CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung

Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Abschlussbericht

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren

SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autoren:

Kjel Barjenbruch, Andrea de Behr,
Dmitrii Dementevskii, Corinna Feeken,
Thies de Graaff, Nils Hartmann, Kardar Kurdo,
Daniel Patron, Christin Poloczek, Jonas Prellberg,
Sebastian Warsitz

Oldenburg, den 18. April 2017

Zusammenfassung

In diesem Abschlussbericht werden alle relevanten Informationen zum Projekt der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ (PGMAmKS) dokumentiert. Während des Projekts wurde eine Applikation entwickelt, die das Importieren und Verwalten von Daten ermöglicht, die durch den *humotion*-Sensorgürtel aufgezeichnet wurden. Des Weiteren erlaubt die Anwendung, importierte Daten manuell zu beschriften und spezielle Algorithmen darauf anzuwenden. Zu den anwendbaren Algorithmen zählen insbesondere Algorithmen zum Training von maschinellen Lernalgorithmen und zur Klassifikation von Bewegungsmustern mithilfe dieser Algorithmen. Die gewählten maschinellen Lernalgorithmen sind Boosted Decision Tree (BDT), Boosted Decision Stump (BDS), Adaptive Multi-Hyperplane Machine (AMM) und Artificial Neural Network (ANN). Weitere relevante implementierte Algorithmen sind der Algorithmus zur Parameteroptimierung der Klassifikatoren, um ein optimales Klassifikationsergebnis zu erreichen, und der Schritterkennungsalgorithmus, der das Berechnen von Gangparametern wie Schrittdauer, Doppelschrittdauer und Kadenz erlaubt. Die Klassifikation erfolgt nach einer hierarchischen Struktur, welche zuerst den Zustand klassifiziert und anschließend innerhalb der Zustände die Klassifikation der einzelnen Bewegungsmuster vornimmt. Als Ergebnis der Parameteroptimierung wurden durchschnittliche F1-Scores von 96,6 % für die Zustände, 97,3 % bzw. 97,5 % für statische bzw. dynamische Aktivitäten und 94,8 % für Transitionen erreicht.

Innerhalb dieses Dokuments wird neben der Beschreibung der entwickelten Applikation vom Konzept bis zur Realisierung auch auf grundlegende Themen, den Prozess der Entscheidungsfindung und das Projekt als solches eingegangen. Des Weiteren wird beschrieben, wie die Projektgruppe manuelle Beschriftungen vorgenommen hat und welche Zusatzdaten als Ergänzung zu den während der VERSA-Studie aufgenommenen, von der Projektgruppe genutzten Daten generiert wurden. Im Anhang befinden sich die Ausarbeitungen der einzelnen Projektgruppenmitglieder, die jeweils auf einen speziellen thematischen Aspekt des Projektes oder mit dem Projekt verknüpfte Themen eingehen.

Inhalt

Abkürzungen

Die mit einem Stern (*) gekennzeichneten Abkürzungen werden ausführlicher im Glossar beschrieben.

ADL	Activities of Daily Living *
ANN	Artificial Neural Network (deut. künstliches neuronales Netz)
AMM	Adaptive Multi-Hyperplane Machine
aTUG	Ambient Timed Up & Go *
BDT	Boosted Decision Tree (deut. verstärkter Entscheidungsbaum)
BDS	Boosted Decision Stump
CLI	Command Line Interface (deut. Kommandozeilenschnittstelle)
CMJ	Counter Movement Jump
DBMS	Datenbankmanagementsystem
DEMMI	De Morton Mobility Index
FFT	Fast Fourier Transformation *
GUI	graphische Nutzerschnittstelle
HPC	High-Performance Computing *
IADL	Instrumental Activities of Daily Living *
JAMAL	Java Matlab Linking
JAXB	Java Architecture for XML Binding
JNI	Java Native Interface
JVM	Java Virtual Machine *
LDA	Linear discriminant analysis *
MATLAB	Matrix Laboratory
MPG	Medizinproduktgesetz
PGMAmKS	Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“
RMS	Root Mean Square *

SCPT Stair Climb Power Test

SPPB Short Physical Performance Battery

SSH Secure Shell *

TUG Timed Up & Go *

VERSA Vorhersage zum Erhalt der Selbstständigkeit im Alter *

W3C World Wide Web Consortium

XML Extensible Markup Language

XSD XML Schema

Abbildungen

Tabellen

Glossar

Activities of Daily Living Eine Menge von Aktivitäten, die eine Person im alltäglichen Leben bewältigen muss, um zu überleben. Dazu zählen beispielsweise Essen, Schlafen oder sich Bewegen.

AEQUIPA Bezeichnet das Projekt, in dessen Rahmen die VERSA-Studie durchgeführt wurde. Genauere Informationen zu dem AEQUIPA-Projekt sind unter `http://www.aequipa.de/home.html` zu finden.

Akzelerometer Beschleunigungsmesser.

Alltagsbewegungsmuster Folgende Bewegungsmuster, die im Alltag auftreten können: Sitzen, Stehen, Liegen, Aufstehen, Hinsetzen, Hinlegen, Hinhocken/Beugen, Gehen und Treppensteigen.

Ambient Timed Up & Go Durchführung des „Timed Up & Go"-Tests mit Hilfe eines Stuhls, in den verschiedene Sensoren zur Messung der Testdurchführung integriert sind.

Assessment Der Begriff (geriatrisches) Assessment bezeichnet die in der Geriatrie durchgeführte Einschätzung des Patienten bezüglich der medizinischen, psychosozialen und funktionellen Möglichkeiten.

Assessment-Bewegungsmuster Folgende bei einem Assessment auftretende Bewegungsmuster: Frailty-Gehtest, Timed Up & Go (TUG), Short Physical Performance Battery (SPPB), Stair Climb Power Test (SCPT), De Morton Mobility Index (DEMMI), Counter Movement Jump (CMJ) und 6-Minuten-Gehtest. SPPB setzt sich aus den Einzeltests „Statisches-Gleichgewicht-Test", „4-m-Gehgeschwindigkeit" sowie „5-Time-Chair-Rise-Test" zusammen und DEMMI aus den Einzeltests „Bett-Test", „Stuhl-Test", „Statisches-Gleichgewicht-Test", „Gehen-Test" sowie „Dynamisches-Gleichgewicht-Test".

Barthel-Index Ein Bewertungsverfahren, das auf einer gewissen Skala die Fähigkeiten einer Person für alltägliche Aufgaben (Activities of Daily Living (ADL)) beschreibt.

Datenreihe Rohdaten des Sensorgürtels oder abgeleitete Zeitreihen (Zeit $\rightarrow$ Skalar).

Fast Fourier Transformation Die Fast Fourier Transformation (deut. schnelle Fourier-Transformation) ist ein Verfahren zur Berechnung der Frequenzanteile eines digitalen Signals.

Gyrometer Drehbewegungsmesser

HERO Die High End Computing Resource Oldenburg ist ein Server-Verbund der Universität Oldenburg, der als Rechencluster zur Verfügung gestellt wird. HERO wird in Abschnitt 3.6 vorgestellt.

High-Performance Computing Zu Deutsch: Hochleistungsrechnen. Beschreibt die computerbasierte Lösung von Aufgaben mit hohem Rechen- und/oder hohem Speicheraufwand.

Hyperthreading Eine Technologie zur Beschleunigung von Rechnerprozessen, bei der in einem Prozessor mehrere virtuelle Prozessoren gleichzeitig unterschiedliche Operationen durchführen.

IEEE-754-Standard Eine Norm zur Darstellung von binären Gleitkommazahlen.

Java Virtual Machine Die Java Virtual Machine führt Java-Programme aus.

Instrumental Activities of Daily Living Eine Menge von Aktivitäten, die ein Individuum nicht zwangsweise ausführen können muss, um zu überleben (siehe ADL), die jedoch ein selbstständiges Leben ermöglichen. Dazu zählen beispielsweise Hausarbeiten oder die Zubereitung von Essen.

Label-Gruppe Benannte Menge von Label-Informationen.

Label-Informationen Ein bestimmtes Bewegungsmuster, das einem Zeitintervall zugeordnet wurde. Ein Bewegungsmuster kann dabei ein Assessment-Test (z. B. TUG, SPPB-Teiltest) oder eine Alltagsbewegung (z. B. Gehen, Stehen, Hinsetzen) sein.

Linear discriminant analysis Ein Verfahren, das zur Dimensionsreduktion von Vektoren eingesetzt werden kann.

LUCAS-Funktions-Index Ein Bewertungsverfahren zur Bestimmung des geriatrischen Syndroms der klinischen Gebrechlichkeit (Frailty).

Magnetometer Messinstrument zur Erfassung von magnetischer Feldstärke.

Remote Host Als Remote Host bezeichnet man einen entfernten Computer, auf den der Benutzer zwar keinen physikalischen Zugriff hat, der jedoch durch geeignete Software vom eigenen Computer aus gesteuert werden kann.

Root Mean Square Root Mean Square (deut. quadratische Mittel) ist ein Mittelwert, bei dem größere Zahlen stärker in das Ergebnis einfließen.

Secure Shell Secure Shell ist ein Netzwerkprotokoll, durch das verschlüsselte Verbindungen zu anderen Geräten hergestellt werden können, die über ein Netzwerk verbunden sind.

Softwarelebenszyklus Beschreibt die Phasen der Softwareentwicklung von der Anforderungsanalyse bis zur Wartung des Produktes.

Timed Up & Go Ein Mobilitätstest, beschrieben in Abschnitt 3.1.

VERSA Bezeichnet die Studie, in der die Assessments durchgeführt und die hier verwendeten Sensordaten aufgenommen wurden. Der ausgeschriebene Titel der Studie ist „Vorhersage zum Erhalt der Selbstständigkeit im Alter“. Genauere Informationen zur VERSA-Studie sind unter `http://www.uni-oldenburg.de/versa-studie/` zu finden.

1. Einleitung

Im Folgenden wird die Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ (PGMAmKS) vorgestellt und ein kurzer allgemeiner Überblick über das Modul gegeben. Aufbauend auf einer einleitenden Motivation zum Projektgruppenthema und der dazugehörigen Problembeschreibung werden die grundlegenden Ziele der in diesem Projekt zu entwickelnden Software definiert. Darüberhinaus soll diese Software den anschließend beschriebenen Bewertungskriterien genügen. Die Einleitung endet mit einem Überblick zu den einzelnen Kapiteln dieser Dokumentation.

1.1. Allgemein zur Projektgruppe

Die Projektgruppe ist ein Pflichtmodul in den Master-Studiengängen Informatik, Wirtschaftsinformatik und Eingebettete Systeme und Mikrorobotik an der Carl von Ossietzky Universität Oldenburg. Die Gruppe umfasst zwischen 6 und 12 Teilnehmern, von denen jeweils ein Arbeitsaufwand von 720 Stunden verteilt über 12 Monate erwartet wird. Dieser Arbeitsaufwand entspricht 24 Kreditpunkten.

Das Ziel der Projektgruppe besteht dabei darin, gemeinsam eine vorgegebene Problemstellung zu bearbeiten. Dabei können studiengangtypische Kenntnisse und Methoden angewendet werden, sodass ein typischer Ablauf Problemanalyse, Lösungsentwurf, Implementierung und Validierung umfasst. Durch die Teilnahme am Projekt sollen aber auch Soft Skills erlernt und ein praxisnaher Einblick in das Projektmanagement ermöglicht werden. Als Teilleistungen werden von jedem Teilnehmer eine Seminararbeit und eine zugehörige Präsentation eines projektnahen Seminarthemas gefordert. Weiterhin muss die Gruppe als Ganzes eine umfangreiche Dokumentation über das Projekt erstellen und es in einer Abschlusspräsentation vorstellen.

Die Projektgruppe PGMAmKS setzt sich aus folgenden elf Studentinnen und Studenten

zusammen: Kjel Barjenbruch, Andrea de Behr, Dmitrii Dementevskii, Corinna Feeken, Thies de Graaff, Nils Hartmann, Kardar Kurdo, Daniel Patron, Christin Poloczek, Jonas Prellberg und Sebastian Warsitz.

Betreuer und Gutachter der hier vorgestellten Projektgruppe sind Dr. rer. nat. Sebastian Fudickar und Sandra Hellmers.

1.2. Motivation

Die Abteilung Assistenzsysteme und Medizintechnik an der Carl von Ossietzky Universität Oldenburg entwickelt im Rahmen des AEQUIPA-Projektes [PKW] neue Technologien zur individuellen Gesundheitsvorsorge in der Primärprävention von Senioren. In der in diesem Rahmen durchgeführten Studie „Vorhersage zum Erhalt der Selbstständigkeit im Alter (VERSA)“ wurden geriatrische Assessments mit 350 Personen durchgeführt. Dabei wurden mithilfe des humotion-Sensorgürtels Bewegungsrohdaten (Daten der Drei-Achsen-Sensoren: Akzelerometer, Gyrometer und Magnetometer sowie Daten eines Barometers) aufgezeichnet. Ziel der VERSA-Studie ist es, Verfahren zu entwickeln, welche es ermöglichen, den Abbau der Muskulatur und somit den möglichen Verlust von Mobilität der Teilnehmer besser vorhersagen zu können. Dies würde es ermöglichen, dem Muskelabbau und -verlust durch Erarbeitung eines Trainingsplans frühzeitig entgegenzuwirken. Im Rahmen der PGMAmKS soll der Ansatz verfolgt werden, ein solches Verfahren auf Basis der durch den Sensorgürtel aufgezeichneten Bewegungsrohdaten zu entwickeln.

Ein geriatrisches Assessment umfasst eine Reihe von Tests mit eigener Bewertungsskala, die der Erfassung und Einschätzung der motorischen Fähigkeiten der am Assessment teilnehmenden Person dienen. Ein Überblick über die mobilitätsrelevanten Tests, welche während der Assessments durchgeführt wurden, ist in Kapitel 3.1 zu erhalten.

Zusätzlich zur Aufzeichnung der Assessments wurde auch der Alltag einer Woche (direkt im Anschluss an das jeweilige Assessment) für jede Person aufgezeichnet. Es gibt Krankheiten wie beispielsweise Parkinson, welche einen Abbau der motorischen Fähigkeiten oder auch eine Veränderung im Bewegungsverhalten als Symptom aufweisen können. Diese Symptome werden leider häufig zu spät erkannt, was eine effektive Behandlung der Krankheiten verhindert. Die Nutzung von Sensoren zur Gewinnung von Mobilitätsparametern erlaubt eine kostengünstige Langzeitbeobachtung, welche es ermöglichen könnte,

entsprechende Symptome frühzeitig zu erkennen. Diesem Ansatz folgend könnte sich auch die Erkenntnis ergeben, dass die Assessments überflüssig werden, wenn die aus der Langzeitbeobachtung gewonnenen Informationen ausreichend für die Einschätzung der Mobilität der jeweiligen Personen sind.

1.3. Problembeschreibung

Im Laufe der VERSA-Studie wurden Bewegungsrohdaten im Gesamtvolumen von 3 TB gesammelt. Dieses riesige Datenaufkommen erfordert es, die Sensordaten automatisiert mithilfe einer geeigneten Software auszuwerten. Eine solche Auswertungssoftware existiert für den in der VERSA-Studie verwendeten Sensorgürtel allerdings bisher nicht. Daher ist die Neuentwicklung einer solchen Software notwendig. Essentiell ist dabei die Erkennung von Bewegungsmustern, wie z. B. *Gehen* oder *Sitzen*, besonders dann, wenn der Sensorgürtel im Alltag getragen wird. Wünschenswert sind außerdem die Berechnung von bewegungsspezifischen Parametern und die automatisierte Interpretation der berechneten Parameter in Bezug auf schleichenden Muskelabbau.

1.4. Ziele

Im Rahmen dieser Projektgruppe muss eine Auswertungssoftware für den in der VERSA-Studie genutzten Sensorgürtel entwickelt werden. Die Ziele, die sich die Projektgruppe bei der Umsetzung dieser Software gesetzt hat, werden im Folgenden beschrieben.

Damit die Daten des Sensorgürtels überhaupt genutzt werden können, muss die Software zunächst die Grundfunktionalität *Daten einlesen* bereitstellen. Die Software muss anschließend in den Rohdaten des Sensorgürtels während eines geriatrischen Assessments und im Alltag durchgeführte Bewegungen erkennen. Dafür ist es notwendig, die Grundfunktionalitäten *Assessment-Übungen erkennen* und *Alltags-Bewegungen erkennen* umzusetzen. Um Aussagen über die erkannten Bewegungen treffen zu können, sollte außerdem die Funktionalität *erkannte Bewegungen auswerten* umgesetzt werden. Dies umfasst z. B. die Bestimmung der Schrittlänge, wenn die Bewegung *Gehen* erkannt wurde. Zuletzt sollten die ausgewerteten Daten in Bezug auf schleichenden Muskulaturabbau oder Bewegungsmusteränderungen bewertet werden. Dafür wird die Funktionalität *Daten bewerten* benötigt.

Da die Software hauptsächlich von Fitnesstrainern oder medizinischem Personal benutzt werden soll, muss auch die Grundfunktionalität *Daten visualisieren* umgesetzt werden. Die Grundfunktionalitäten werden in Abbildung 30 anhand eines Use-Case-Diagramms verdeutlicht.

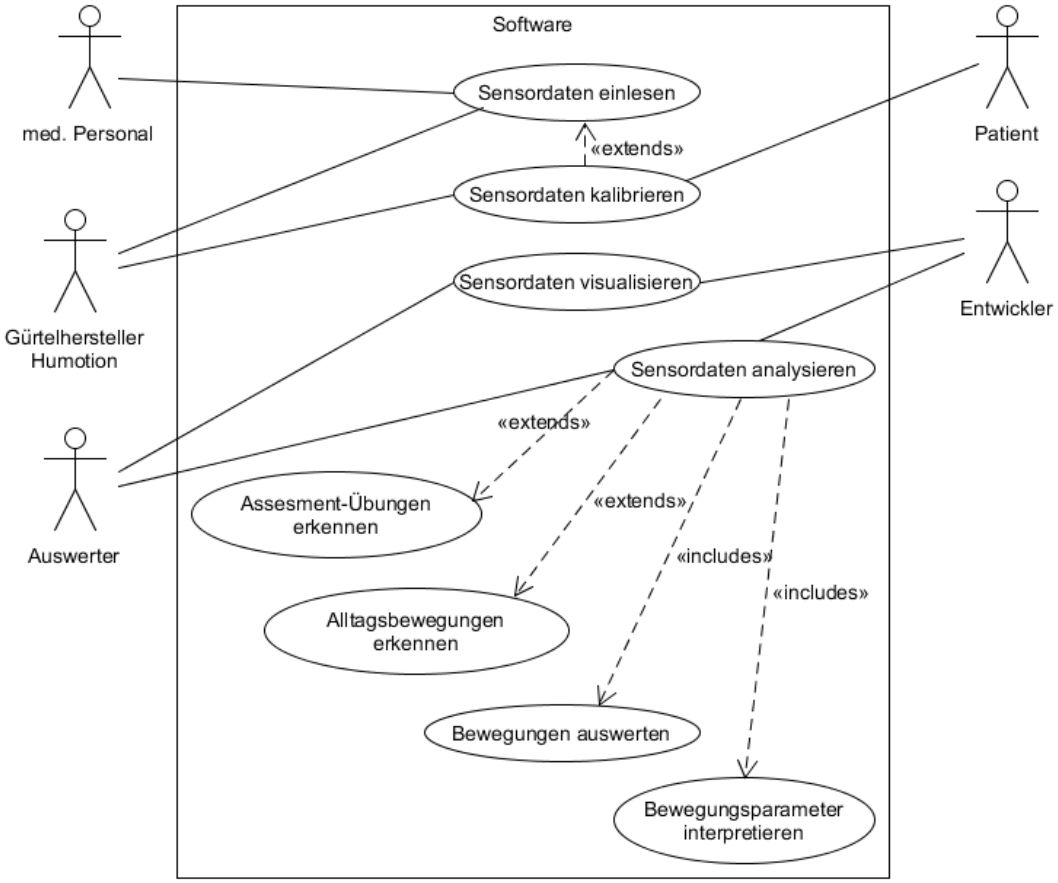


Abbildung 1.: Use-Case-Diagramm der Grundfunktionalitäten der Software

Neben den Grundfunktionalitäten muss die zu entwickelnde Software einige nicht-funktionale Anforderungen erfüllen. Das User-Interface muss den Entwicklern und dem medizinischen Personal einen einfachen Umgang mit der Software ermöglichen. Weiterhin ist Performanz angesichts der großen Datenmengen wichtig. Daher muss die Software rechenintensive Algorithmen parallelisiert in einem Rechencluster ausführen. Der Quellcode der Software muss außerdem gut verständlich und erweiterbar sein. Um das zu gewährleisten, müssen Entwicklungsstandards eingehalten werden. Ebenfalls muss begleitend zum Software-Entwicklungsprozess eine Dokumentation erstellt werden, die die Anforderungen, den Entwicklungsprozess und die Software-Handhabung beschreibt.

1.5. Bewertungskriterien

Um sicherzustellen, dass die zu entwickelnde Software die an sie gestellten Anforderungen erfüllt, wurden die folgenden Bewertungskriterien festgelegt. Zunächst einmal muss die Anwendung in der Lage sein, die fehlerfreien Daten eines Sensorgürtels einzulesen. Darüber hinaus müssen die verschiedenen Assessment-Tests (Frailty-Gehtest, Timed Up & Go (TUG), Short Physical Performance Battery (SPPB), Stair Climb Power Test (SCPT), De Morton Mobility Index (DEMMI), Counter Movement Jump (CMJ) und 6-Minuten-Gehtest) korrekt von der Software erkannt und benannt werden. Hierbei ist zu beachten, dass sich der SPPB aus den Einzeltests „Statisches-Gleichgewicht-Test“, „4-m-Gehgeschwindigkeit„ sowie „5-Time-Chair-Rise-Test“ zusammensetzt und der DEMMI aus den Einzeltests „Bett-Test“, „Stuhl-Test“, „Statisches-Gleichgewicht-Test“, „Gehen-Test“ sowie „Dynamisches-Gleichgewicht-Test“.

Um die Bewertung des Frailty-Gehtests, des TUG, der 4-m-Gehgeschwindigkeit (aus dem SPPB), des 5-Time-Chair-Rise-Tests (aus dem SPPB) und des SCPT vornehmen zu können, genügt es, die Durchführungsdauer jeweils in Sekunden zu erfassen. Auch für die Bewertung der einzelnen Stände (Schlussstand, Semitandemstand, Tandemstand, Stand mit geschlossenen Füßen, Stand auf den Fußspitzen) innerhalb der beiden Statisches-Gleichgewicht-Tests (aus dem SPPB und dem DEMMI) ist die jeweilige Ermittlung der Durchführungsdauer in Sekunden ausreichend. Im Gegensatz dazu ist für die Bewertung des 6-Minuten-Gehtests und des Gehen-Tests (aus dem DEMMI) die jeweils zurückgelegte Strecke in Metern zu erfassen. Für die Bewertung des Bett-Tests, des Stuhl-Tests sowie des Dynamisches-Gleichgewicht-Tests (aus dem DEMMI) wiederum muss erkannt werden können, ob die einzelnen Testbestandteile dem Probanden jeweils selbständig, mit Hilfe oder gar nicht gelingen, wobei beim Sitzen im Stuhl ohne Unterstützung (aus dem Stuhl-Test) auch die Durchführungsdauer in Sekunden benötigt wird. Zur Bewertung der jeweiligen Assessment-Tests müssen im Anschluss die Ergebnisse des Frailty-Gehtests und des SPPB in Punkte, die Ergebnisse des TUG in Kategorien und die Ergebnisse des DEMMI zunächst in Kategorien und dann in Punkte umgerechnet werden.

Des Weiteren muss die Software die einzelnen Assessment-Test-Bewegungsmuster (Gehen, Springen, Hinsetzen, Aufstehen, Balance halten, Treppensteigen) zuverlässig identifizieren. Das Assessment-Test-Bewegungsmuster Gehen findet sich im Frailty-Gehtest, TUG, SPPB, DEMMI und im 6-Minuten-Gehtest. Im TUG, SPPB und DEMMI treten zusätzlich noch

die Assessment-Test-Bewegungsmuster Hinsetzen und Aufstehen auf, während Balance Halten lediglich im SPPB und DEMMI vorkommt. Springen findet sich als Assessment-Test-Bewegungsmuster im DEMMI und CMJ, wohingegen Treppensteigen im SCPT auftritt.

Die erkannten Assessment-Tests und Assessment-Test-Bewegungsmuster muss die Software sowohl für die Entwickler als auch für die Endnutzer visualisieren. Außerdem muss die Software den Entwicklern und Endnutzern die Ausgabe der Bewertung der Assessment-Tests ermöglichen.

Wünschenswert wäre des Weiteren, wenn die zu entwickelnde Software auch Alltagsbewegungsmuster (Gehen, Stehen, Sitzen, Liegen, Treppensteigen, Fahrradfahren) korrekt erfassen könnte. Zusätzlich sollte die Software in der Lage sein, aus den erkannten Assessment-Tests und Bewegungsmustern Bewegungsparameter (Laufzeit, Laufstrecke, Schrittlänge und Schrittgeschwindigkeit beim Gehen und Treppensteigen, Sprunghöhe und Sprungkraft beim Springen, Aufstehkraft, Aufsteh- und Hinsetzgeschwindigkeit beim Aufstehen bzw. Hinsetzen, Balance im Stand) abzuleiten.

Zukünftig sollen dann Bewegungsmusterveränderungen korrekt erkannt und visualisiert werden. So wird den Endnutzern ermöglicht, dem jeweiligen Probanden einen auf ihn abgestimmten Trainingsplan zu erstellen, um seinem weiteren Muskelabbau und Mobilitätsverlust vorzubeugen.

1.6. Aufbau des Projektabschlussberichtes

Nachfolgend wird beschrieben, wie sich der Projektabschlussbericht im Einzelnen zusammensetzt. Zunächst werden im Kapitel Projektmanagement die Rollenaufteilung der einzelnen Projektgruppenmitglieder, die gesamte Projektorganisation sowie der dazugehörige Meilensteinplan beschrieben und es wird auf das während des Projekts geführte Tagebuch verwiesen, das sich im Anhang befindet. Im anschließenden Kapitel werden Grundlagen erläutert, indem auf den Ablauf von Assessments, die Erhebung und manuelle Beschriftung der Sensorgürteldaten, den Prozess zur automatischen Erkennung von Aktivitäten, Schritten und davon abgeleiteten Gangparametern sowie die Nutzung des HERO-Rechenclusters eingegangen wird. Das Kapitel Anforderungsanalyse benennt die für die Anwendung identifizierten Anforderungsfälle, die Anforderungsliste sowie die

Einschränkungen durch das Medizinproduktgesetz. Es folgt im Kapitel Konzeption eine Beschreibung der Architektur der Software und der darauf bezogenen Technologieentscheidungen, eine Auflistung der von der Anwendung unterstützten Programmiersprachen, der Spezifikationen von in der Applikation genutzten Dateiformaten sowie der Möglichkeiten zur Generierung von Referenzinformationen und eine Erläuterung des GUI-Designs, der implementierten Algorithmen, des Aktivitätserkennungsverfahrens mithilfe des Maschinellen Lernens sowie der Parameteroptimierung. Das Kapitel Realisierung geht auf die einzelnen Software-Komponenten Frontend, Kommandozeilenschnittstelle, Shared (für die Datenverarbeitung) und Job-Executor (für die Algorithmusausführung) ein. Im darauffolgenden Kapitel Evaluierung werden die Split- und Kreuzvalidierung trainierter Modelle dargestellt sowie die Auswahlverfahren für die Sliding-Window-Größe bzw. -Schrittweite und die (Nicht-)Verwendung von Nicht-Probandendaten. Das anschließende Kapitel beschreibt die Entwicklung eines Toshiba-Sensors zur Datenübertragung in Echtzeit. Zum Abschluss wird ein Fazit gezogen und ein Ausblick gegeben über die Generierung von Assessment-Labels, die Nachbereitung von Klassifikationsergebnissen, die geringe Klassifikationsgenauigkeit, die Implementierung weiterer Techniken, die Bestimmung weiterer Bewegungsparameter, die Validierung der Gangparameterergebnisse, die Darstellung von Echtzeitdaten und die Aktivitätserkennung anhand von Daheim-Daten. Im Anhang finden sich die jeweiligen Ausarbeitungen der Projektgruppenteilnehmer zu ausgewählten Aspekten des Projektgruppenthemas.

2. Projektmanagement

In diesem Kapitel werden die projektinternen Abläufe und Regelungen beschrieben, die für eine produktive Zusammenarbeit und den geregelten Projektablauf von der Gruppe aufgestellt wurden. Zuerst wird darauf eingegangen, welche Rollen die einzelnen Mitglieder innerhalb der Gruppe übernommen haben. Danach wird die Fairnessvereinbarung der Gruppe vorgestellt, die Regeln für das gegenseitige Miteinander festlegt. Anschließend werden die Projekt-Management-Methode Scrum, die in leicht abgewandelter Form im Projekt genutzt wurde, und die Gruppenstandards erläutert. Zu Letzteren zählen festgelegte Code-Konventionen, der Workflow im Jira und Git ebenso wie der Ablauf für das Schreiben der Unit-Tests sowie die hierfür nutzbaren Frameworks. Es folgt die Beschreibung des Review-Prozesses und damit verbunden die *Definition of Done*. Die zeitliche Planung des Projektes ist im Meilensteinplan festgehalten und zudem wurde ein Projekttagebuch geführt, welches die relevantesten Ereignisse und Entscheidungen beinhaltet.

2.1. Rollenaufteilung

Damit für jeden Tätigkeitsschwerpunkt innerhalb der Projektgruppe ein Hauptverantwortlicher benannt ist und als Ansprechpartner zur Verfügung steht, wurden den Projektgruppenmitgliedern zu Beginn des Projekts entsprechende Rollen zugewiesen. Die Rolle der Projektleitung hat Andrea de Behr übernommen und darüber hinaus fungiert sie als Raumbeauftragte. Als Projektleiterin ist sie für das Einhalten des Zeitplans, das Lösen von Gruppenkonflikten und den internen sowie externen E-Mail-Verkehr zuständig. Ihre Aufgabe als Raumbeauftragte ist die Anforderung benötigter Ausstattungsgegenstände. Jonas Prellberg ist der Scrum-Master der Projektgruppe. In dieser Rolle beantwortet er Fragen zum Vorgehensmodell Scrum und übernimmt die Aufgabe der Scrum-Prozessüberwachung. Thies de Graaff übt die Funktion des Testbeauftragten aus. Die Beantwortung testbezogener Fragen und das Sicherstellen der Qualität von Tests gehören dabei zu seinem Aufgabenbe-

reich. Die Rolle des Git-Beauftragten übernimmt Sebastian Warsitz. Er ist zuständig für das Beantworten von Fragen zur Versionsverwaltung Git, das Einrichten des Git-Repositorys und die Überwachung der Repository-Struktur. Daniel Patron nimmt die Funktion des Jira- und Confluence-Beauftragten wahr. Diese Aufgabe umfasst die Beantwortung von Fragen zum Projektwiki Confluence und zur Projektverfolgung Jira sowie die Überwachung der Confluence-Struktur. Kardar Kurdo übernimmt die Rolle des LaTeX-Beauftragten und des Kassenwarts. Als LaTeX-Beauftragter beantwortet er Fragen zum Textsatzsystem LaTeX, erstellt Vorlagen und ist verantwortlich für die Strukturierung der Projektdokumentation. Um bei allen Projektgruppenmitgliedern die Motivation für einen pünktlichen Sitzungsbeginn zu erhöhen, erhebt Kardar in seiner Funktion als Kassenwart bei Teilnehmern, die unangekündigt verspätet zu einer Sitzung erscheinen, eine Strafgebühr.

2.2. Projektorganisation

Um einen Überblick über das gesamte Projekt, die jeweiligen Meilensteine und die darin anfallenden Aufgaben zu erhalten und um zu gewährleisten, dass die einzelnen Projektaufgaben innerhalb eines festgelegten Zeitraums korrekt und vollständig abgearbeitet werden, ist es notwendig, entsprechende Regeln der Zusammenarbeit und der Aufgabenabwicklung aufzustellen. Hierunter fallen die Fairness-Vereinbarung, die die Regeln für den Umgang der Projektgruppenteilnehmer miteinander aufstellt, der Scrum-Prozess, der das Gesamtprojektziel in kleinere Etappenziele einteilt und so strukturiert, die Gruppenstandards, die den Jira- und Git-Workflow sowie die Code-Konventionen und die Unit-Tests beschreiben, und das Qualitätsmanagement, das durch einen gruppenweiten Review-Prozess umgesetzt wird.

2.2.1. Regeln für Zusammenarbeit

In einer Projektgruppe treffen verschiedene Kenntnisse und Wissensstände aufeinander sowie verschiedene Herangehensweisen. Aufgrund dieser Unterschiedlichkeiten ist es eine Aufgabe der Projektgruppe, alle ihre Mitglieder in das Projekt einzubinden und die Aufgaben so zu verteilen, dass jeder möglichst effizient mitarbeiten kann. Jedem Projektgruppenteilnehmer soll es ermöglicht werden, seine Kenntnisse einzubringen und sich in neue Themen einzuarbeiten. Die Stärke einer Projektgruppe ist dabei, dass man

auf den gemeinsamen Wissens-Pool zugreifen und sich dadurch gegenseitig helfen und unterstützen kann. Um dies zu gewährleisten, ist es notwendig, ein Arbeitsklima zu schaffen, in dem einerseits Fragen gestellt werden dürfen und beantwortet werden und in dem andererseits konstruktive Kritik geäußert und angenommen werden kann.

Ein weiterer wesentlicher Punkt ist die Aufrechterhaltung der Motivation – sowohl der Gruppe als auch des einzelnen Gruppenteilnehmers – über den Zeitraum eines Kalenderjahres. Es gibt zwei Arten von Motivation, wobei die sogenannte intrinsische Motivation aus einem selbst heraus kommt (wenn einem z. B. ein Thema besonders liegt oder Spaß macht), während die extrinsische Motivation eine von außen herbeigeführte ist (durch besondere Privilegien wie beispielsweise eine Gehaltserhöhung oder durch Druck, wenn man z. B. von seinen Eltern zu einer bestimmten Ausbildung gedrängt wird). Über einen längeren Zeitraum gesehen hat die intrinsische Motivation den nachhaltigeren Effekt. Zu beachten ist auch, dass Maßnahmen zur Erhöhung der extrinsischen Motivation die intrinsische Motivation reduzieren oder sogar ganz verdrängen können.

In der nun folgenden Fairnessvereinbarung sind die Regeln, die die Teamarbeit fördern und die intrinsische Motivation aller Projektgruppenmitglieder stärken bzw. erhalten sollen, zusammengefasst.

Fairnessvereinbarung

- Die Projektgruppenteilnehmer begegnen sich mit Freundlichkeit, Fairness, Wertschätzung, Respekt und Zuverlässigkeit.
- Die Team-Mitglieder helfen und unterstützen sich gegenseitig.
- Unterschiedliche Kenntnisse und Wissensstände werden berücksichtigt.
- Kritik wird sachlich und konstruktiv geäußert, sodass der Kritisierte sein Gesicht wahren kann.
- Die Projektgruppenteilnehmer gehen auf lösungsorientierte Weise an Fehler heran.
- Es wird (außer auf ausdrückliche Bitte des Dozenten hin) keine Bewertung der Arbeit anderer Projektgruppenmitglieder vorgenommen.
- Die Selbstkontrolle bzw. Eigenverantwortung aller wird gefördert.

2.2.2. Scrum

Für eine bessere Abstimmung im Projektteam und für die Sicherstellung der Erreichung des Projektziels (siehe Abschnitt 1.4) wird eine strukturierte Aufgabenverteilung bzw. Projektplanung nach der agilen Management-Methode Scrum verwendet. Agil bezieht sich in diesem Kontext auf einen beweglichen und sich entwickelnden Prozess. In diesem Abschnitt wird die Entwicklungsmethode Scrum kurz vorgestellt, auf das vorliegende Projekt bezogen und angepasst. Die Projekt-Management-Methode Scrum hilft einem Team, ein komplexes Projekt durch Regeln und Muster erfolgreich abschließen zu können. Dabei besteht ein Scrum-Projekt aus mehreren Iterationen (Sprints), die in der Regel maximal 30 Tage dauern. In Bezug auf das vorliegende Projekt werden die Sprints als 14-Tage-Abschnitte definiert. Die Abbildung 2 soll dabei helfen, den Ablauf eines Scrum-Prozesses besser nachvollziehen zu können. Während eines Sprints wird jede Anforderung im Kontext des Gesamtprojektes in Teilaufgaben aufgeteilt, die anschließend erarbeitet und umgesetzt werden [HRS09]. Zu einem Scrum-Prozess gehören mehrere Akteure, die nach bestimmten Regeln ein Projekt zum Abschluss bringen. Diese und der Ablauf eines Scrum-Projektes werden im Folgenden dargestellt und auf das vorliegende Projekt bezogen.

Scrum-Team

Ein Scrum-Team besteht aus drei Rollen:

- dem Product Owner
- dem Entwicklungsteam
- dem Scrum Master

Product Owner Die wichtigste Funktion im Scrum obliegt dem Product Owner. Er trifft Entscheidungen und lenkt mit seiner Kompetenz das gesamte Projekt. Vor allem identifiziert der Product Owner die Anforderungen an das Projekt und trägt diese ins sogenannte Product Backlog ein [Max12]. Die Entwicklung des Produktes überlässt der Product Owner dem Entwicklungsteam. Die Funktion des Product Owners übernimmt im vorliegenden Projekt die Projektgruppe selbst, da sie eigenständig die einzelnen Sprints plant und somit die Verantwortung für das Projekt übernimmt.

Entwicklungsteam Die Aufgabe des Entwicklungsteams ist die Umsetzung der vom Product Owner festgehaltenen Anforderungen. Diese Anforderungen werden durch die Projektgruppe selbstständig und iterativ umgesetzt. Im Team selbst gibt es keine Hierarchie, sodass alle Mitglieder des Teams gleichgestellt sind, es gibt jedoch wie im Abschnitt 2.1 beschrieben, eine Verteilung von besonderen Aufgaben. Ein Team sollte so zusammengestellt sein, dass sich die Fähigkeiten der einzelnen Mitglieder ergänzen und somit ein Inkrement gemeinsam erarbeitet werden kann [Max12]. Im vorliegenden Projekt werden die Fähigkeiten der Autoren in Bezug auf die Hardware-Analyse, Software-Entwicklung sowie die Recherche zu Algorithmen der Bewegungsmustererkennung ergänzt.

Scrum Master Aufgabe des Scrum Masters ist es, Sorge zu tragen, dass das Team die Regeln des Scrum-Prozesses einhält, dass die Inhalte des Product Backlogs verstanden sowie dass diese zielgerichtet umgesetzt werden. Ebenso fungiert der Scrum Master als Bindeglied zwischen Team und Product Owner [Max12]. Diese Funktion wird wie im Abschnitt 2.1 beschrieben innerhalb der Projektgruppe durch Jonas Prellberg übernommen.

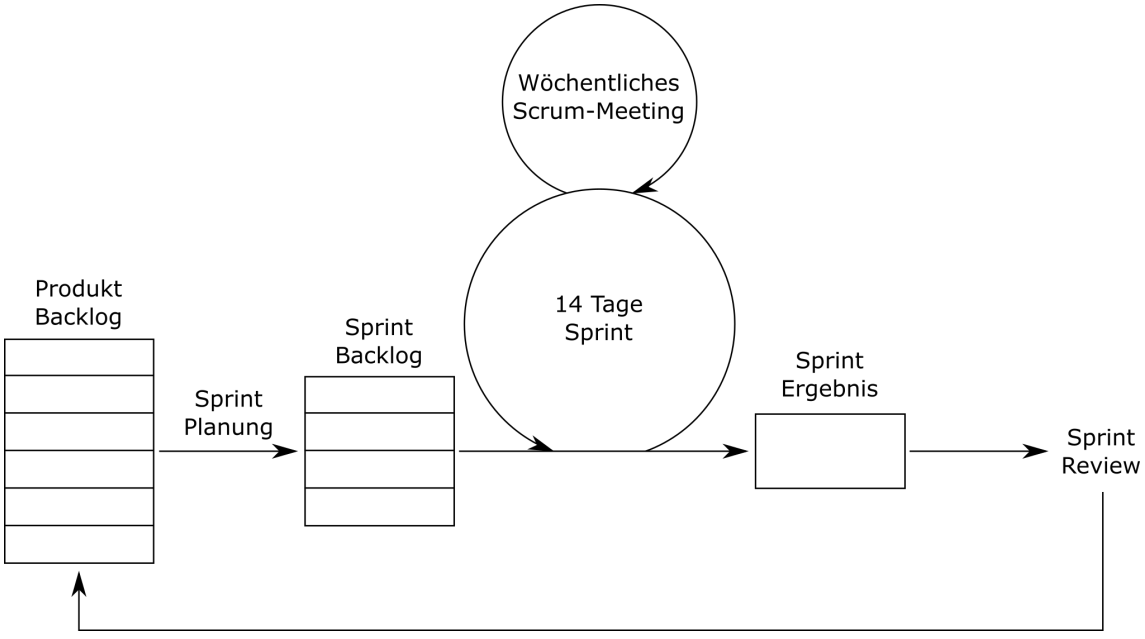


Abbildung 2.: Scrum-Prozess

Product Backlog

Das zentrale Dokument zum Bestimmen der Anforderungen an ein Projekt stellt das Product Backlog dar. In ihm werden alle Anforderungen und Funktionalitäten des Produktes,

in diesem Fall die auszuführenden Entwicklungstätigkeiten, vom Product Owner verwaltet, priorisiert und in Bezug auf ihren Realisierungsaufwand geschätzt [Han10].

Sprints

Im Folgenden werden die fünf wichtigsten Aspekte eines Sprints im Detail vorgestellt und jeweils ein direkter Bezug zum vorliegenden Projekt hergestellt.

Weekly Scrum Unter dem Begriff Weekly Scrum werden wöchentliche Teambesprechungen verstanden. Die Besprechung erfolgt am Anfang der wöchentlichen Teamsitzung und soll möglichst kurz und auf die wichtigsten Informationen begrenzt sein. Im Weekly Scrum informiert jedes Teammitglied die anderen Teammitglieder über den Fortschritt der eigenen Aufgaben. Zum Teil können hier bereits weniger komplexe Probleme und Fragen geklärt werden, auf längere Diskussionen wird an dieser Stelle jedoch verzichtet.

Sprint Planning Zu Beginn eines jeden Sprints wird eine Sprint-Planungssitzung abgehalten. In dieser Sitzung wird durch den Product Owner das Ziel des anstehenden Sprints festgelegt, welches dann vom Entwicklungsteam zu verwirklichen ist [Han10]. Für die Sprint-Planung werden im Product Backlog alle Aufgaben identifiziert, die für die Erreichung des Sprint-Ziels notwendig sind [Han10]. Die Inhalte eines jeden Sprints werden vom Entwicklungsteam diskutiert und so ausgewählt, dass alle Aufgaben nach 14 Tagen abgeschlossen sind. Dabei sind Zeitaufwand der Aufgaben und zur Verfügung stehende Arbeitszeit - hier kann es zu Einschränkungen aufgrund von Urlaub oder Krankheit kommen - gegeneinander abzuwägen. Bezogen auf das vorliegende Projekt werden die Aufgaben des Product Backlog kontinuierlich erstellt, zum Großteil jedoch direkt in der Sprint-Planung.

Estimation Meeting Das Estimation Meeting gehört inhaltlich zur Sprint-Planung. Zur Beurteilung des Zeitaufwands einzelner Aufgaben nutzt die Projektgruppe sogenannte Storypoints. Diese vereinfachen die Abschätzung der zu schaffenden Aufgaben eines Sprints. Jeder Aufgabe im Product Backlog wird dabei eine Punktzahl, die Storypoints, zugeordnet, welche dem Zeitaufwand dieser Aufgabe entsprechen soll. Die Abschätzung der Storypoints einer Aufgabe erfolgt in der Projektgruppe mithilfe der Planning Poker Methode. Bei dieser Methode erhält jeder Teilnehmer des Estimation Meetings Karten mit vordefinierten Storypoints. Um eine Aufgabe abzuschätzen. hält jeder Teilnehmer die

Karte mit den Storypoints hoch, die seiner Beurteilung des Zeitaufwands der Aufgabe entsprechen. Die Storypoints der am häufigsten gezeigten Karte werden danach der Aufgabe zugeordnet. Dadurch soll eine möglichst realistische Abschätzung des Zeitaufwandes gewährleistet werden. Im Anschluss werden nur so viele Aufgaben aus dem Product Backlog in den sogenannten Sprint Backlog übernommen, dass eine vorher bestimmte Anzahl an Storypoints, die der maximal zur Verfügung stehenden Arbeitszeit entsprichen soll, nicht überschritten wird.

Sprint Review Jeder Sprint wird mit dem Sprint Review beendet. In dieser Phase wird das Inkrement des Sprints durch den Product Owner abgenommen. Es werden nur fehlerfreie und vollständige Ergebnisse akzeptiert. Kommt es zu Beanstandungen durch ein Mitglied des Teams oder durch den Product Owner, wird die Aufgabe wieder in den Sprint aufgenommen. Die Beanstandungen werden behoben, bis ein fehlerfreies Ergebnis präsentiert werden kann [Han10]. Die Arbeitsweise der Projektgruppe unterscheidet sich in diesem Fall etwas vom beschriebenen Ablauf aus der Literatur (siehe Unterabschnitt 2.2.4).

Retrospektive In diesem Abschnitt des Scrum-Prozesses steht die Beurteilung des Entwicklungsprozesses selbst im Fokus. Von jeder Person, die im Entwicklungsprozess des Projektes mitwirkt, können Aspekte vorgetragen werden, die als positiv oder negativ im Entwicklungsprozess aufgefallen sind. Dabei findet keine Gewichtung der Kritik statt unabhängig davon, welche Rolle die Person inne hat, sei es als Entwicklungsteammitglied oder als Product Owner. Die gesammelten Anmerkungen werden diskutiert und Verbesserungsvorschläge dazu unterbreitet. Die daraus resultierenden Mehrheitsbeschlüsse sollen im darauffolgenden Sprint umgesetzt werden. Bei der Retrospektive geht es also um die Optimierung des Entwicklungsprozesses und damit der Teamarbeit [Han10].

2.2.3. Gruppenstandards

Die Abarbeitung der jeweiligen Projektaufgaben bzw. Jira-Tasks innerhalb eines Sprints soll termingerecht, vollständig und korrekt im Sinne der Anforderungsliste erfolgen. Um dies zu gewährleisten, wurde im Folgenden festgehalten, wie der Scrum-Prozess im Projekt-Management-Tool Jira umgesetzt, die Abarbeitung der Jira-Tasks in der Versionsverwaltung Git gestaltet, die Qualität des Quellcodes durch Code-Konventionen sichergestellt und die Logik der Software getestet werden soll.

Jira-Workflow

Für die Verwaltung und Organisation der Sprints sowie der zugehörigen Issues, welche die Aufgabenarten Task, Bug, Epic, Story und Repetition umfassen, wurde in dem Projekt das Projekt-Management-Tool Jira verwendet. Das Agile-Board bietet hierbei eine Übersicht des aktuellen Sprints. Issues, die noch nicht in einen Sprint aufgenommen wurden, befinden sich im Backlog. Für alle abgeschlossenen und laufenden Sprints lässt sich ein Burndown-Chart anzeigen, welches den geschätzten und den tatsächlichen Arbeitsaufwand des jeweiligen Sprints grafisch darstellt.

Das Agile-Board beinhaltet vier Workflow-Phasen: „To Do“, „In Progress“, „In Review“ und „Done“. Issues aus der „To Do“-Spalte dürfen sich Gruppenmitglieder selbstständig zuweisen. Wenn die Issue bearbeitet wird, soll sie in die „In Progress“-Spalte verschoben werden. In dieser Spalte sollte sich möglichst nur eine Issue pro Person befinden. Ist die Bearbeitung abgeschlossen oder ein Review notwendig, werden Issues gegebenenfalls mit einem Kommentar, wo die Arbeit zu finden ist, nach „In Review“ verschoben. Issues, die ein Review durchlaufen haben, werden entweder vom Bearbeiter nach „Done“ verschoben, wenn sie korrekt und vollständig bearbeitet wurden, oder auf „To Do“, sollten noch Korrekturen notwendig sein.

Im Allgemeinen dürfen keine neuen Issues in einen laufenden Sprint aufgenommen werden. Eine Ausnahme wird gemacht, wenn in der Sprintplanung der Arbeitsaufwand für die Issues zu hoch eingeschätzt wurde und daher vor Sprintende keine Issues mehr zur Verfügung stehen, um die wöchentlich geforderte Arbeitszeit zu erreichen.

Das Ziel ist es, alle Issues, auch die eventuell nachträglich hinzugefügten, innerhalb des Sprints abzuschließen.

Git-Workflow

Der folgende Abschnitt beschreibt den Workflow, der mit der eingesetzten Versionsverwaltung Git einzuhalten ist.

Fängt man mit der Bearbeitung einer neuen Issue an, so muss mit dem Befehl *git pull* zunächst der aktuelle Stand von dem Server geladen werden.

Mit dem Befehl *git branch* <*branch-name*> wird ein neuer Branch angelegt, dessen Name dem Jira-Key der Issue entspricht (z. B. PGMAMKS-16). Auf diesem Branch kann man

ungestört arbeiten, nachdem man mit *git checkout <branch-name>* auf diesen gewechselt ist.

Möchte man Änderungen committen, so müssen zunächst die betreffenden Dateien via *git add* zu einem Commit gebündelt werden.

Anschließend wird der Commit via *git commit -m <nachricht>* durchgeführt, wobei die Commit-Nachricht auf Englisch beschreibt, was für Änderungen durchgeführt wurden. Nun können weitere Änderungen vorgenommen werden, die später auch committet werden, oder man überträgt seine Commits auf den Server.

Existiert der neu angelegte Branch noch nicht auf dem Server, so wird er mit dem Befehl *git push -u origin <branch-name>* auf den Server geladen. Jetzt existiert der neue Branch mit allen Commits auch auf dem Server. Alle weiteren Commits können durch *git push* hochgeladen werden.

Ist die Arbeit an einem Branch abgeschlossen und wurden alle Review-Anmerkungen umgesetzt, so muss dieser Branch vom Bearbeiter in den Master gemerged werden. Hierzu wird mit dem Befehl *git checkout master* zunächst zurück in den Master gewechselt. Anschließend wird über *git merge <branch-name>* der Merge durchgeführt. Falls es zu Konflikten kommt, müssen diese von Hand aufgelöst werden. Wenn der Konflikt gelöst wurde, muss diese Änderung neu committet werden.

Anschließend wird der Merge durch *git push* auf den Server übertragen.

Code-Konventionen

Die Qualität der Software erhöht sich, wenn der Quellcode leicht verständlich, wiederverwendbar und leicht erweiterbar ist. Die folgenden Programmierkonventionen sollen genau dies sicherstellen:

- Die Formatierung soll im Google Java Style erfolgen. Der Eclipse-Formatter verändert keine Newlines, daher ist darauf zu achten, dass keine doppelten Leerzeilen vorkommen. In Ausnahmefällen kann vom Styleguide abgewichen werden - etwa wenn die Lesbarkeit eines Abschnitts bei manueller Formatierung deutlich besser ist.
- Es sollen keine überflüssigen Imports vorgenommen werden.
- Der Code ist ausreichend zu kommentieren.

- Die Bezeichner sollen aussagekräftig sein.
- Klassen sollen sinnvoll strukturiert werden, was die Reihenfolge und Gruppierung von Variablen und Methoden sowie die Nutzung von Leerzeilen angeht, um die Lesbarkeit zu erleichtern.
- Es sind die richtigen Modifier (public, private , ...) zu wählen.
- Es sind die richtigen Datenstrukturen für die jeweilige Aufgabe auszuwählen.
- Statische Methoden sollen nicht verwendet werden, da diese nicht gemockt werden können.
- Funktionen sollen nicht mehr als 50 Zeilen Code beinhalten.
- Die Logik ist so einfach wie möglich zu halten.
- Der Code soll keine TODOs enthalten.
- Es sollen keine println's oder anderer Debug-Code im Code verbleiben.
- Warnings sind zu beseitigen.
- Es sind Unit-Tests mit sinnvollen Testfällen und einer Code-Coverage von mindestens 80 % zu erstellen.
- Wenn Klassen durch vorgenommene Erweiterungen mehr als eine „Aufgabe“ haben, soll der Code in mehrere Klassen aufgeteilt werden.

Javadoc Klassen und Methoden sollen mithilfe von Javadoc dokumentiert werden:

- Die Javadoc-Dokumentation soll in englischer Sprache erfolgen (und zwar vor dem jeweils zu beschreibenden Element).
- Bei Interfaces ist Javadoc zur Beschreibung des Interfaces, der Fields und der Methoden zu nutzen.
- Bei Klassen soll Javadoc mindestens zur Beschreibung der Klasse sowie der öffentlichen Funktionen, die keine Interface-Methoden implementieren, angewendet werden (für getter und setter nicht zwingend notwendig, aber trotzdem hilfreich).
- Bei Klassen, Interfaces und Fields kann als Beschreibung das Subjekt weggelassen und nur das Objekt genannt werden, z. B. „Ein Button-Label“.
- Der Javadoc-Kommentar wird idealerweise direkt beim Erstellen der Methode hinzuge-

fügt.

- Die Beschreibung von Methoden beginnt mit einem Verb.
- Geschrieben wird aus der dritten Person, beispielsweise bei einer Methode: „Bekommt das Label übergeben.“
- Methoden sollen nicht durch die Wörter beschrieben werden, die im Methodennamen genutzt werden, da dies keine zusätzlichen Informationen bringt.
- Die Tags „@param“ für die Parameter und „@return“ für den Rückgabewert von Methoden sollen auf jeden Fall verwendet werden, weitere sind optional.

Unit-Tests

Um für alle Klassen, die Logik in irgendeiner Form enthalten, sowohl das lt. Anforderungsliste korrekte Verhalten dieser Logik als auch die definierten Sonder- und Fehlerfälle zu testen, müssen Unit-Tests für die jeweiligen Klassen geschrieben werden. Darauf aufbauend werden die Frameworks Hamcrest, Mockito und JaCoCo genutzt, die im Folgenden näher beschrieben werden.

Hamcrest Hamcrest ist ein Framework, mit dem sich Matcher-Objekte beschreiben lassen. Mithilfe von Hamcrests *assertThat*-Methode und diesen Matchern können die eigentlichen *Asserts* von JUnit vermieden und die Leserlichkeit der Tests deutlich erhöht werden. Hamcrest stellt bereits diverse Matcher zur Verfügung, die allgemeine Zwecke abdecken. Darüber hinaus können aber auch eigene Matcher definiert werden, was nützlich ist, wenn in verschiedenen Tests immer die gleichen Attribute getestet werden.

Mockito Ein Mock bezeichnet einen Dummy einer Klasse, der keine Logik beinhaltet. Für die Methoden eines Mocks kann aber festgelegt werden, welchen Rückgabewert ein Aufruf haben soll. Durch dieses Konzept wird gewährleistet, dass die Implementierung der zu testenden Klasse unabhängig von der Implementierung anderer Klassen ist. Umgesetzt wird dieses Konzept mithilfe von Mockito. Da man allerdings mit Mockito keine statischen Methoden mocken kann, ist die Nutzung derselben im Projekt nicht möglich.

JaCoCo Sobald ein Test fertig geschrieben ist, kann mit JaCoCo geprüft werden, wie gut die Code-Coverage dieses Tests ist. Wenn die Tests mithilfe des Build-Tools Gradle

ausgeführt werden, wird auch automatisch ein HTML-Report generiert, in dem die Code-Coverage aller Tests zu sehen ist (alternativ dazu gibt es auch ein Eclipse-Plugin für JaCoCo). Das Ziel eines Tests sollte es sein, den gesamten Code zu testen, von daher sollte die Code-Coverage immer gegen 100 % streben. Für die Projektgruppe wird als Minimum eine Code-Coverage von 80 % festgesetzt.

Schreiben eines Tests

1. Parallel zu der zu testenden Klasse im *src*-Verzeichnis wird im *test*-Verzeichnis der Test angelegt.
2. Mithilfe von JUnit, Hamcrest und Mockito wird die Logik anhand von Beispielen getestet, deren Resultat von vornherein bekannt ist, sodass das Ergebnis des Programms verifiziert werden kann. Hierbei werden sowohl Positiv- als auch Negativbeispiele verwendet, um zu gewährleisten, dass die Implementierung der zu testenden Klasse in allen Fällen (Normalfall, Sonderfall und Fehlerfall) korrekt ist.
3. Es wird mithilfe von JaCoCo geprüft, wie gut die Code-Coverage ist und welche Stellen des Codes nicht abgedeckt sind. Der Test wird dann entsprechend so lange erweitert, bis die Code-Coverage mindestens 80 % erreicht.

Ändern einer Klasse Wenn eine bereits existierende Klasse geändert wird, für die es einen Test gibt, so muss dieser Test nach dem Abschluss der Modifizierung ausgeführt werden. Im Falle eines Fehlschlags muss überprüft werden, ob die neue Implementierung einen Fehler hat oder ob der Test aufgrund der Änderung des Codes entsprechend angepasst werden muss. Unabhängig davon ist zu beachten, dass der Test immer im Einklang mit der Anforderungsdefinition ist. Sollte dies nicht der Fall sein, muss gegebenenfalls die Anforderungsdefinition überarbeitet werden, falls dies erwünscht ist. Andernfalls ist der Test entsprechend anzupassen.

2.2.4. Qualitätsmanagement durch gruppenweiten Review-Prozess

Damit alle Projektaufgaben bzw. Tasks im Sinne der Anforderungsliste korrekt und vollständig abgearbeitet werden, nutzt die Projektgruppe den im Folgenden beschriebenen Review-Prozess, in dem geprüft wird, ob die jeweilige Task die sogenannte Definition of Done erfüllt.

Review-Workflow

Wenn der Bearbeiter einer Task der Meinung ist, dass ein Review sinnvoll ist, stellt er die Task in JIRA auf „In Review“ und mergt den Master-Branch vor dem Anlegen einer Pull-Request in Bitbucket in den bearbeiteten Branch. Sollten dabei Konflikte entstehen, so müssen diese per Hand gelöst und der Branch im Anschluss committet werden.

Danach wird in Bitbucket eine Pull Request mit dem Jira-Key als Überschrift erstellt. Dazu wählt man den bearbeiteten Branch und den Ziel-Branch für das Mergen aus. Damit kann man überprüfen, ob die geänderten Dateien ohne Konflikte fast-forward gemergt werden könnten. Sollte dies nicht der Fall sein, so muss der Master-Branch ein weiteres Mal in den bearbeiteten Branch gemergt, Konflikte müssen per Hand gelöst und committet werden.

Darauf folgend wird ein Prüfer ausgewählt. Dieser und der Bearbeiter erhalten durch das System automatisch eine Benachrichtigung per E-Mail, sollte sich etwas an dem Branch ändern oder wenn ein Kommentar bzw. eine Aktion in Bitbucket vorgenommen wurde.

Der Bearbeiter einer Task kann, wenn nötig, im Kommentarbereich notieren, wo die Arbeit (Dateien/Seiten) zu finden ist und worauf beim Review geachtet werden soll.

Nun soll der Prüfer diese Arbeit begutachten und überprüfen, ob die Task richtig umgesetzt wurde. Handelt es sich bei der Task um eine Programmieraufgabe, so soll die Code-Review-Checkliste beachtet werden.

Sollte es Mängel geben, so können diese in Bitbucket direkt an der entsprechenden Änderung kommentiert werden. Außerdem kennzeichnet der Prüfer in Bitbucket, dass es bei der Task Mängel gibt.

Der Bearbeiter kann nun anhand des Feedbacks die Verbesserungen vornehmen. Hierfür zieht er die Task wieder auf „In Progress“. Der Kommentarbereich kann außerdem zum Dialog zwischen Bearbeiter und Reviewern genutzt werden, falls Unklarheiten vorhanden sind.

Sind die Verbesserungen abgeschlossen, wird die Task wieder auf „In Review“ gestellt und der Prozess beginnt von vorne.

Ist die Task laut Prüfer ohne Mängel, so genehmigt er diese in Bitbucket. Sollte ein direkter fast-forward Merge möglich sein, so führt der Prüfer diesen aus, zieht die Task in JIRA auf „Done“ und schließt sie somit ab.

Sollten beim Merge Konflikte entstehen, so zeigt Bitbucket dies an. Diese Konflikte sind wie oben beschrieben vom Bearbeiter zu beheben.

Definition of Done Die folgenden Voraussetzungen müssen erfüllt sein, damit eine Issue als abgeschlossen gilt und in Jira auf „Done“ gestellt werden darf:

- Die in der Issue beschriebene Aufgabe ist vollständig und korrekt im Sinne der Anforderungsdefinition umgesetzt.
- Der Review-Prozess wurde erfolgreich durchlaufen und alle hierbei gefundenen Mängel sind beseitigt.
- Die finalen Änderungen wurden vom Bearbeiter in den Master gemergt.

Bei Programmieraufgaben müssen zusätzlich noch die folgenden Voraussetzungen erfüllt sein:

- Die Code-Konventionen wurden eingehalten.
- Der Code ist durch einen Test abgedeckt, wobei die Code-Coverage mindestens 80 % betragen soll.
- Alle Tests werden weiterhin erfolgreich ausgeführt.

2.3. Meilensteinplan

Um strukturiert an das Projekt herangehen und die Anforderungen im vorgegebenen Zeitrahmen erfüllen zu können, wurde ein Meilensteinplan erstellt, der eine Orientierung für die Sprintplanung bietet.

geplantes Fertigstellungsdatum	Ziele	tatsächliches Fertigstellungsdatum
31. Mai 2016	Anforderungsanalyse	31. Mai 2016
	Aufnahme der Demodaten	31. Mai 2016
	Festlegung der Entwicklungsstandards	31. Mail 2016
	Ist-Analyse (Hardware, Framework, aufgenommene Daten etc.)	04. August 2016
04. August 2016	Aufsetzen der Applikation	04. August 2016
	Visualisierung der Rohdaten	04. August 2016
	Spezifikation des Dateiformats	18. August 2016
	Implementierung des Command Line Interface (deut. Kommandozeilenschnittstelle) (CLI)	14. September 2016
	Implementierung von Algorithmussequenzen (CLI)	14. September 2016
19. Dezember 2016	Recherche zu Algorithmen für die Mustererkennung in den Sensordaten	13. Oktober 2016
	Erkennung von Bewegungsmustern (Alltag inkl.) in Demodaten	28. November 2016
	Erkennung von Bewegungsmustern (Alltag inkl.) in Geriatriedaten	28. November 2016
19. Dezember 2016	Ergebnisse aller Ausarbeitungen	27. März 2017
09. Januar 2017	Vorträge zu den Ausarbeitungen	27. März 2017
30. Januar 2017	HPC einbinden	13. Februar 2017
	Bestimmung von Assessment-Parametern (Gang)	13. März 2017
13. März 2017	Interpretation der Ergebnisse (Güte der Klassifikationsergebnisse)	27. März 2017
	Code-Freeze	13. April 2017
	Fertigstellung des Toshiba-Sensor-Hardware-Aufbaus und der zugehörigen Dokumentation	wird nachgereicht

06. April 2017	Vorbereitung der Präsentation	13. April 2017
	Umsetzung restlicher Kleinigkeiten	17. April 2017
	Fertigstellung der Dokumentation	18. April 2017

2.4. Projekttagebuch

Während der gesamten Projektlaufzeit wurde ein Projekttagebuch geführt, um wichtige Ereignisse des Projektes mit exakter Datierung in chronologischer Reihenfolge festzuhalten. Dadurch kann auf einfache Weise nachvollzogen werden, was zu welcher Zeit im Projekt geschehen ist. Schon während der Projektlaufzeit hat sich dies als sinnvoll erwiesen, da Projektmitglieder sich so nach vorübergehender Abwesenheit einen schnellen Überblick über den zwischenzeitlichen Projektfortschritt verschaffen können. Darüber hinaus ist so leicht nachvollziehbar, wann welche Projektentscheidungen getroffen wurden, welche Gründe es dafür gab und wie sich diese Entscheidungen jeweils auf den Projekterfolg ausgewirkt haben. Das Projekttagebuch dieses Projektes ist im Anhang in Abschnitt C zu finden.

3. Grundlagen

In diesem Kapitel werden die projektrelevanten Grundlagen erläutert. Dazu zählen der Ablauf der Assessments, die Datenerhebung mit einem Sensorgürtel und die Beschriftung der vom Gürtel erzeugten Daten. Es wird detailliert auf die Schritte der Akivitätserkennung eingegangen und die anschließende Bewertung der Aktivitätsklassifikation eingegangen. Weiterhin werden Gangparameter vorgestellt, welche nach der Aktivitätserkennung mit Hilfe einer Schritterkennung abgeleitet werden können. Schließlich wird ein Überblick des Rechenclusters der Universität Oldenburg gegeben, welches genutzt wurde, um zeitaufwendige Algorithmen auszuführen.

3.1. Assessments

Im Folgenden werden lediglich Testanteile aufgeführt, die für die Betrachtung der vom Sensorgürtel aufgenommenen Daten relevant sind. Weiterhin werden die Tests nur so genau beschrieben, wie es für die Auswertung im Rahmen des Projekts notwendig ist.

Zu Beginn des Assessments wird eine Eingangsbefragung durchgeführt, an die sich eine allgemeine Befragung, Instrumental Activities of Daily Living (IADL), Barthel-Index, LUCAS-Funktions-Index und eine anschließende Blutdrucks- und Wadenumfangsmessung anschließen. Erst danach wird der Gürtel angelegt und mit den Frailty-Kriterien begonnen. Während der ersten drei Frailty-Kriterien Gewichtsverlust, Schwäche und Erschöpfung sitzt der Proband. Dann wird das Ganggeschwindigkeitskriterium erfasst, wonach sich der Proband erneut setzt, um den Frailty-Fragebogen zu beantworten. Gegebenenfalls wird der Fragebogen zuerst beantwortet und dann das Geschwindigkeitskriterium erfasst. Im Anschluss an den Fraility-Test folgen die weiteren Assessments. Nach dem gesamten Ablauf (Frailty bis 6-Minuten-Gehtest) gibt die Person den Sensorgürtel ab und erhält einen neuen Gürtel für die Aufnahme der Heimdaten.

Eine Übersicht des gesamten Ablaufs inklusive der Allgemeinbewegungen ist im Anhang Abschnitt A zu finden. Die Frage- und Bewertungsbögen der einzelnen Tests sind im Anhang Abschnitt B zu finden. Der Raum, in dem ein Großteil der Assessments stattfindet, ist in Abbildung 3 zu sehen.

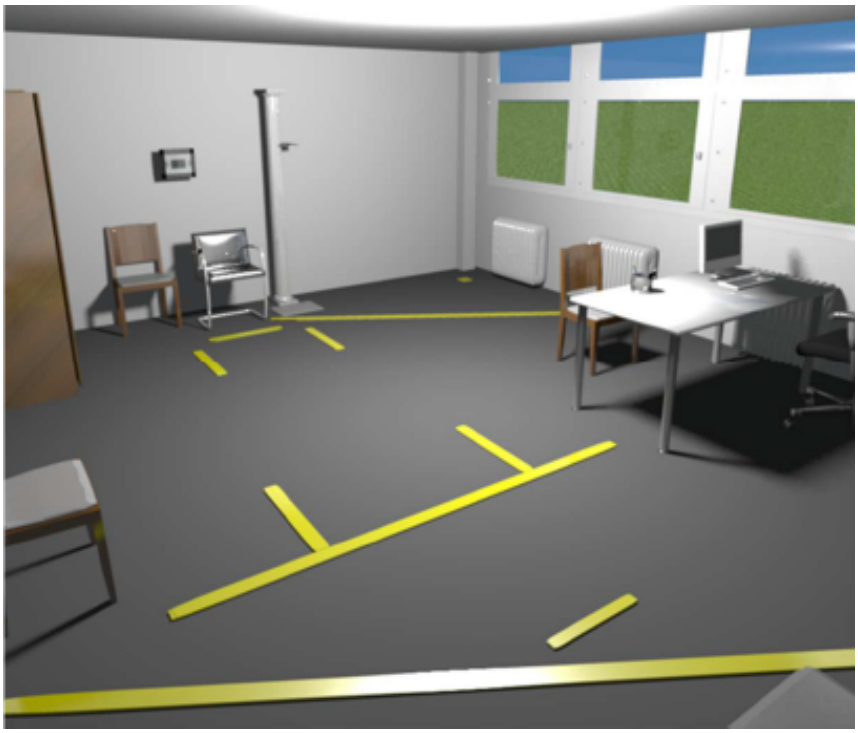


Abbildung 3.: Assessment-Raum

1. Frailty Kriterien

    a) Gehgeschwindigkeit: Die Person stellt sich in eine Ecke des Raums und geht dann beim Signal 4,57 Meter geradeaus zur anderen Seite des Raums.

2. Timed Up & Go (TUG)

   Die Person sitzt auf dem aTUG-Stuhl (siehe Abbildung 4). Beim Signal steht sie auf, geht drei Meter geradeaus, dreht sich um, geht zum Stuhl zurück und setzt sich wieder. Es wird die Zeit vom Aufstehen bis zum Hinsetzen gemessen. Der Test wird zweimal durchgeführt. Die Drehrichtung beim Umdrehen und Hinsetzen ist willkürlich.

3. Short Physical Performance Battery (SPPB)

   Die Person stellt sich zu Beginn auf die Gewichtsplatte (siehe Abbildung 5) und bleibt während der Teilabschnitte auf dieser stehen. Die Person soll während der Teilabschnitte die Haltung für zehn Sekunden halten. Es kann vorkommen, dass die Person dazu nicht fähig ist und vorher abbricht, andernfalls endet der Abschnitt nach zehn Sekunden. Die

Abbildung 4.: aTUG-Stuhl

Arme hängen während des Tests locker an den Seiten herunter. Für den Gehtest steigt die Person von der Platte und geht zur Ausgangsposition.

a) Statisches Gleichgewicht

   i. Schlussstand

      Die Person steht aufrecht auf der Gewichtsplatte mit beiden Füßen direkt nebeneinander.

   ii. Semitandemstand

      Die Person steht aufrecht auf der Gewichtsplatte mit beiden Füßen leicht versetzt, so dass der Fußballen des einen Fußes neben dem Hacken des anderen Fußes steht.

   iii. Tandemstand

      Die Person steht aufrecht auf der Gewichtsplatte mit beiden Füßen hintereinander, so dass die Fußspitze des einen Fußes an den Hacken des anderen Fußes stößt.

b) 4m Gehgeschwindigkeit

   Die Person steht in der Ecke auf einer Ausgangspunktmarkierung. Beim Signal geht sie geradeaus. Dabei überschreitet sie zwei Linien, die vier Meter voneinander entfernt sind, und bleibt beim Signal stehen. Die Person dreht sich dann um und wiederholt den Test beim Signal in umgekehrter Richtung. Die zurückgelegte Distanz ist etwas größer als vier Meter, da die Linien überschritten werden.

c) 5x Chair Rise

Die Person setzt sich auf den aTUG-Stuhl und überkreuzt die Arme. Sie führt dann ein Testaufstehen und -setzen durch. Während des Tests steht die Person fünfmal auf und setzt sich dann wieder hin. Die Zeit des Tests wird vom 1. Aufstehen (Signal) bis zum 5. Aufstehen gemessen.

Abbildung 5.: Bodenkraftmessplatte

4. Stair Climb Power Test (SCPT)

Die Person verlässt den Assessment-Raum und geht zur Treppe. Dort stellt sie sich vor die unterste Stufe. Beim Signal steigt die Person zehn Treppenstufen nach oben. Der Test wird zweimal durchgeführt. Zwischen den Tests steigt die Person die Stufen wieder hinab, um aus derselben Position zu starten.

5. De Morton Mobility Index (DEMMI)

a) Bett

Zu Beginn legt sich die Person mit dem Rücken auf die Liege.

i. Brücke

Die Person setzt die Füße auf, streckt die Arme aus und bildet eine Brücke. Diese hält sie für einen Moment und legt sich wieder ab.

ii. Auf die Seite rollen

Die Person rollt sich auf eine (beliebige) Körperseite.

iii. Vom Liegen zum Sitzen

Die Person setzt sich auf der Liege auf, so dass die Kniekehlen sich auf Höhe der Kante der Liege befinden und die Beine locker herunterhängen.

b) Stuhl

Hierfür setzt sich die Person nicht extra auf einen Stuhl, sondern es wird direkt aus der Sitzposition auf der Liege fortgefahren. Die Liege wird vorher gegebenenfalls

herunter gefahren.

i. Sitzen im Stuhl ohne Unterstützung

Die Person sitzt weiterhin auf der Liege.

ii. Aus dem Stuhl aufstehen

Die Person steht auf, bleibt kurz stehen und setzt sich anschließend wieder.

iii. Aus dem Stuhl aufstehen, ohne die Arme zu benutzen

Die Person überkreuzt die Arme, steht auf und bleibt stehen. (Anschließend setzt sich die Person in den meisten Fällen wieder hin.)

c) Statisches Gleichgewicht ohne Gehhilfe

Die Person stellt sich auf die Gewichtsplatte. Die Arme hängen während dieses Tests locker herunter. Die Abschnitte dauern jeweils zehn Sekunden oder bis zum Abbruch durch die Person.

i. Ohne Unterstützung stehen

Die Person steht aufrecht in selbst gewähltem Stand auf der Platte.

ii. Stehen mit geschlossenen Füßen

Die Person steht aufrecht auf der Gewichtsplatte mit beiden Füßen direkt nebeneinander.

iii. Auf den Fußspitzen stehen

Die Person steht stabil — die Füße nebeneinander, aber mit Abstand zueinander, so dass die Beine geöffnet sind. Sie stellt sich dann aus dieser Haltung auf die Zehenspitzen. Es kommt hierbei vor, dass die Person tippelt oder sogar die Füße umsetzt.

iv. Im Tandemstand mit geschlossenen Augen stehen

Die Person steht aufrecht auf der Gewichtsplatte mit den Füßen hintereinander, so dass die Fußspitze des einen Fußes an den Hacken des anderen Fußes stößt, und schließt dabei die Augen. Es kommt hierbei vor, dass die Person die Arme ausstreckt, um die Balance wiederzufinden. Häufig halten die Probanden die vollen zehn Sekunden nicht durch.

d) Gehen

Die Person verlässt den Assessment-Raum und begibt sich zum Anfang der Wegstrecke. Bei Bedarf wird dieser Test mit Gehhilfe (Gehbock, Stock, Rollator, ...)

durchgeführt.

i. Wegstrecke (mit oder ohne Gehhilfe)

Die Person geht geradeaus, dreht sich nach 20 Metern um (willkürliche Richtung) und geht dann wieder geradeaus. Nach weiteren 20 Metern dreht sie sich erneut um (willkürliche Richtung) und geht weitere 10 Meter. Die Person geht also insgesamt 50 Meter. Es kann vorkommen, dass die Person nicht fähig ist, die gesamte Strecke zu gehen. In diesem Fall wird der Test abgebrochen.

e) Dynamisches Gleichgewicht

Die Person begibt sich wieder in den Assessment-Raum.

i. Stift vom Boden aufheben

Die Person hebt einen Stift vom Boden auf, wenn möglich. Wie sie dies tut, ob durch Hinabbeugen oder Hinhocken, ist ihr überlassen.

ii. Vier Schritte rückwärts

Die Person begibt sich in die Ausgangsposition und geht dann vier Schritte rückwärts.

iii. Sprung

Die Person stellt sich auf die Gewichtsplatte und führt anschließend einen Sprung aus.

6. Counter Movement Jump (CMJ)

Die Person bleibt auf der Gewichtsplatte stehen. Sie führt drei Sprünge hintereinander aus. Zwischen jedem Sprung wird eine Minute gewartet.

7. 6 Minuten Gehtest

Die Person verlässt den Assessment-Raum und stellt sich an die Startposition. Beim Signal geht die Person geradeaus los. Auf Grund der räumlichen Gegebenheiten muss die sich Person alle 20 Meter umdrehen (die Richtung ist willkürlich) und danach in die entgegengesetzte Richtung weiterlaufen. Vermutlich geht die Person etwas mehr als 20 Meter, um die Lichtschranken (siehe Abbildung 6) an der entsprechenden 20 Meter-Marke zu passieren.

Abbildung 6.: Lichtschranken

3.2. Datenerhebung mittels Sensorgürtel

In diesem Projekt wurde zur Datenerhebung ein Messgürtel der Firma Humotion aus Münster (siehe Abbildung 7) verwendet, der im Wesentlichen aus einer Sensoreinheit mit Datenlogger besteht. Der Messgürtel verfügt über verschiedenste Sensoren, um Bewegungsparameter des Trägers über eine Dauer von mehreren Stunden zu erfassen. Der Aufbau des Sensorgürtels und die Funktionsweisen der einzelnen Sensoren werden in den folgenden Abschnitten näher beschrieben.

Abbildung 7.: Foto Sensorgürtel

3.2.1. Aufbau des Sensorgürtels

Die Kommunikation zwischen den Sensoren und dem Mikrocontroller findet vollständig über einen digitalen Signalbus statt. Eine Kommunikation via Datenbus hat den Vorteil, dass sie weniger störanfällig ist als die analoge Datenübertragung per Spannungspegel. Zudem sind mit einem Datenbus höhere Datenraten realisierbar. Die verwendete Bustopologie

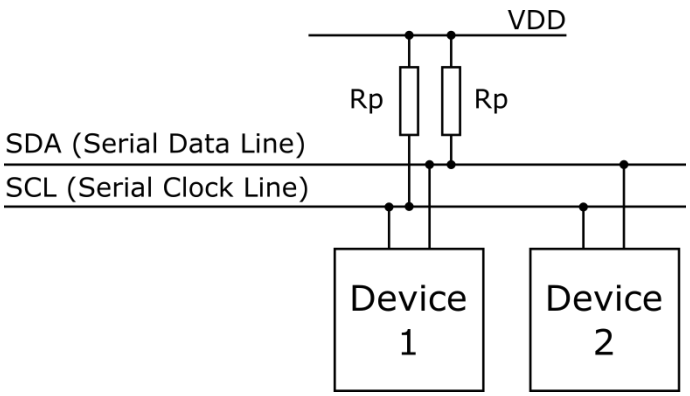


Abbildung 8.: Aufbau eines Sensornetzwerks per I^2C

ist der I^2C-Bus. Dieser ist ein von NXP Semiconductors (ehem. Philips Semiconductors) entwickelter serieller Datenbus, der zur Kommunikation zwei Leitungen benötigt. Eine Leitung dient dabei als Taktsignal (SCL, System Clock) und auf der anderen Leitung werden die Nutzdaten übertragen (SDA, System Data). Der I^2C-Bus ist als Master-Slave-System konzipiert, das auch mehrere Master zulässt. Ein beispielhafter Netzaufbau ist in Abbildung 8 dargestellt. Für die bidirektionale Kommunikation sind Datenraten von bis zu 3,4 Mbit/s möglich (High speed mode), üblich sind allerdings 100 kbit/s (Standard mode). I^2C ist speziell als Bus auf Leiterplatten häufig verwendet, da es so möglich ist, dass ein Mikrocontroller ein ganzes Netzwerk an integrierten Schaltungen mit nur zwei I/O-Pins und einer relativ simplen Software kontrollieren kann. Die Einfachheit des Busses sorgt allerdings auch für Probleme: Er ist störanfällig, da er nicht durch differenzielle Übertragung gegen Übersprechen und Rauschen geschützt ist. Das macht den I^2C-Bus ungeeignet zur Überbrückung größerer Entfernungen, wie es beispielsweise für Feldbusse in industriellen Sensornetzwerken in der Produktionstechnik typisch ist. Da der Bus in dieser Anwendung jedoch nur zur Kommunikation innerhalb der Schaltung über wenige Zentimeter eingesetzt wird, ist das System ausreichend störsicher [NXP14].

Die gesamte Hardware einschließlich aller Sensoren, des Speichers und der Batterie sind direkt im Gürtel integriert und somit für den Probanden nicht sichtbar und geschützt. Der Sensorgürtel lässt sich wie eine handelsüblicher Gürtel tragen und schränkt den Träger daher bei keinerlei Bewegung ein.

Das Tragen der Sensoren mittels eines Gürtels hat den Vorteil, dass die Messinstrumente immer an ihrem richtigen Platz sind. Die einzige Fehlerquelle ist, dass das Anlegen des Sensorgürtels verkehrt herum geschieht. Dies lässt sich jedoch einfach anhand der

Orientierung der Schwerkraft mit Hilfe des Beschleunigungssensors erkennen.

Den Sensorgürtel gibt es in verschiedenen Modellen mit unterschiedlichen Aufnahmeraten. In der Projektgruppe wird mit Sensorgürteln gearbeitet, welche die Aufnahmefrequenzen 100 Hz und 400 Hz bieten. Die selbst entwickelte Anwendung unterstützt prinzipiell beliebige Aufnahmeraten des Sensorgürtels und ist in der Lage gleichzeitig mit Datensätzen gemischter Aufnahmefrequenzen zu arbeiten.

Die Humotion-DataLogger-Software zum Auslesen der Aufnahmen eines Gürtels gibt die Daten unbearbeitet in einer CSV-Datei aus, welche im Unterabschnitt 5.4.1 beschrieben wird.

3.2.2. Sensorik

In diesem Kapitel werden die im Gürtel verbauten Sensoren, ihre Funktion und Besonderheiten näher beschrieben. Die für alle Sensoren identische Ausrichtung bei korrekt angelegtem Gürtel zeigt Abbildung 9.

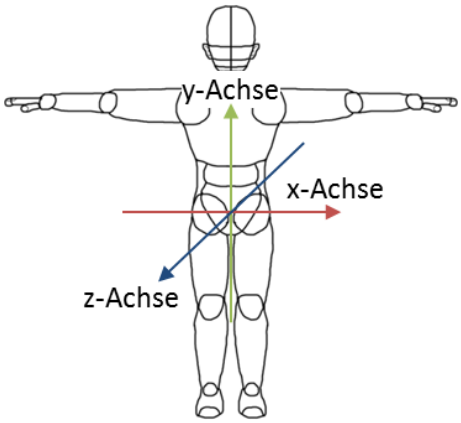


Abbildung 9.: Ausrichtung der Sensoren

Gyroskop

Ein Gyroskop ist ein Sensor, der eine relative Drehbewegung misst. Der Sensor wird bei der Systeminitialisierung kalibriert, weshalb die Initialisierung im Stillstand erfolgen muss. Den aus der Initialisierung gewonnenen Wert nimmt der Sensor als Nulllage an. Da dieser Sensor wie jeder andere einer Messabweichung unterworfen ist und nur relative Winkeländerungen ermittelt werden können, driftet der Messwert mit jeder weiteren

Winkeländerung. Der Fehler potenziert sich also über die Messdauer immer weiter. Allein auf Basis dieses Sensors wäre eine Ermittlung der Position kaum möglich.

Beschleunigungssensor

Der Beschleunigungssensor, auch Accelerometer genannt, misst absolute Beschleunigungen in der SI-Einheit $\frac{m}{s^2}$ (Meter pro Sekunde zum Quadrat) und ist der zweite Sensor zur Bestimmung der Ausrichtung im Raum. Dieser Sensor gibt ein deutlich ungenaueres Signal bei kleinen Winkeländerungen aus als das Gyroskop. Er hat jedoch den Vorteil, dass die Erdanziehungskraft als Beschleunigung auf ihn wirkt und damit eine permanente gerichtete Beschleunigung vorhanden ist. Diese wird bei aufrechtem Stand des Probanden vom Sensor als dauerhafte positive Beschleunigung in y-Richtung gemessen. Die Erdanziehungskraft macht es möglich, den Absolutwert der Winkellage zu ermitteln. Bei dem verbauten Sensor BMA180 von Bosch Sensortec [Bosa] handelt es sich um ein mikro-elektro-mechanisches System (MEMS) mit dem Prinzip eines Feder-Masse-Systems. „Durch die Auslenkung bei Beschleunigung kann zwischen dem gefedert aufgehängten Teil und einer festen Bezugselektrode eine Änderung der elektrischen Kapazität gemessen werden.“ [RWH14, S. 583]

Magnetometer

Das Magnetometer ist ein Sensor zur Ermittlung der magnetischen Flussdichte in der Einheit Gauß. Im Falle unseres Sensors von MEMSIC [MEM] handelt es sich um einen anisotropen magnetoresistiven Sensor, der den gleichnamigen Effekt ausnutzt.

Barometer

Mit dem Barometersensor BMP085 von Bosch Sensortec [Bosb] wird der absolute Luftdruck gemessen. Der Barometersensor wird zur Ermittlung von Vertikalbewegungen, also Höhenänderungen verwendet. Wie in Abbildung 10 zu sehen ist, sinkt mit der Höhe der atmosphärische Luftdruck. Der Sensor misst die Druckunterschiede mittels einer piezoresistiven Elektronik. Bereits integriert ist ein Analog-Digital-Converter. Der Sensor teilt dem System den Luftdruckwert sowie die Temperatur mit.

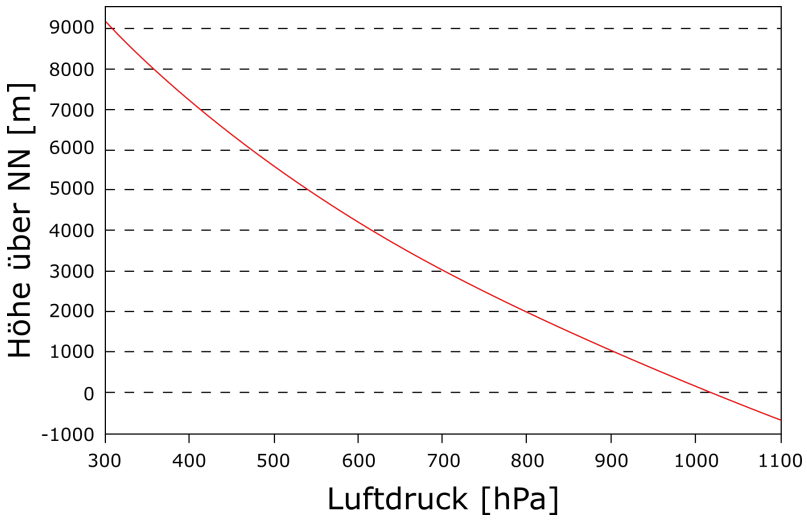


Abbildung 10.: Verlauf des atmosphärischen Luftdrucks bei Höhenänderung

Thermometer

Sowohl der im Sensorgürtel verbaute Beschleunigungssensor als auch der Luftdrucksensor besitzen jeweils ein integriertes Thermometer. Beide Temperaturmesswerte der Sensoren werden ausgegeben.

3.3. Datenbeschriftung

Die Datenbeschriftung ist im Rahmen des maschinellen Lernens für das Training von guten Klassifikationsalgorithmen ein notwendiger Schritt. In diesem Abschnitt werden zunächst die Beschriftungstypen und die zugeordneten Bewegungsarten der Assessment-Datensätze genannt. Anschließend wird anhand einer Vorlage das Beschriften der gewonnenen Assessment-Datensätze veranschaulicht.

3.3.1. Beschriftung der Sensordaten

Die vorliegenden Daten sind unbeschriftet in dem Sinne, dass Beginn und Ende der Assessments nicht mit Zeitstempeln gekennzeichnet sind. Ziel ist es, Bewegungen und darauf aufbauend die Assessments durch Klassifikationsalgorithmen zu erkennen. Auf diese Weise ist das System später auch für Bewegungserkennung im Alltag verwendbar und nicht auf den Assessmentablauf eingeschränkt. Es werden beschriftete Daten benötigt – einerseits um Machine-Learning-Algorithmen damit zu trainieren und andererseits um

eine Grundlage für die Bewertung der Klassifikationsergebnisse zu haben.

Die Daten wurden mit den Beschriftungen aus Tabelle 2 versehen. Jeder Beschriftung ist ein Zustand zugeordnet, die den grundsätzliche Typ der Bewegung angibt. Es wird zwischen dynamischen, statischen und Übergangsbewegungen unterschieden.

Ausgehend von den Assessments wurden Bewegungen identifiziert, die genutzt werden können, um die Assessments zu identifizieren. Diese Zuordnung ist in Tabelle 3 dargestellt.

Problematisch bei der Beschriftung sind die Daten aus den Heim-Aufnahmen der Probanden. Für diese ist verständlicherweise kein Ablauf bekannt und somit existieren keine Referenzen für die Beschriftung. Die manuelle Beschriftung dieser Daten wäre also höchstwahrscheinlich fehlerhaft. Deswegen wurde sich gegen die Nutzung dieser Daten entschieden, da eine Validierung gegen unsichere Referenzdaten nicht besonders sinnvoll erscheint.

Beschriftung	Zustand
ASSESSMENT_FRAILTY_SPEED	ASSESSMENT
ASSESSMENT_TUG	ASSESSMENT
ASSESSMENT_SPPB_BALANCE	ASSESSMENT
ASSESSMENT_SPPB_SPEED	ASSESSMENT
ASSESSMENT_SPPB_CHAIRRISE	ASSESSMENT
ASSESSMENT_SCPT	ASSESSMENT
ASSESSMENT_DEMMI_BED_BRIDGE	ASSESSMENT
ASSESSMENT_DEMMI_BED_ROLL_TO_SIDE	ASSESSMENT
ASSESSMENT_DEMMI_BED_SIT_UP	ASSESSMENT
ASSESSMENT_DEMMI_CHAIR_SIT	ASSESSMENT
ASSESSMENT_DEMMI_CHAIR_STAND_UP	ASSESSMENT
ASSESSMENT_DEMMI_CHAIR_STAND_UP_WITHOUT_HELP	ASSESSMENT
ASSESSMENT_DEMMI_BALANCE_STATIC	ASSESSMENT
ASSESSMENT_DEMMI_WALK	ASSESSMENT
ASSESSMENT_DEMMI_PENCIL	ASSESSMENT
ASSESSMENT_DEMMI_WALK_BACKWARDS	ASSESSMENT
ASSESSMENT_DEMMI_JUMP	ASSESSMENT
ASSESSMENT_CMJ	ASSESSMENT
ASSESSMENT_WALKSIXMINUTES	ASSESSMENT
WALK	DYNAMIC
STAND	STATIC
STAND_SIT	TRANSITION
STAND_LIE	TRANSITION
SIT	STATIC
SIT_STAND	TRANSITION
SIT_LIE	TRANSITION
LIE_ON_BACK	STATIC
LIE_ON_SIDE	STATIC
LIE_ON_BACK_LIE_ON_SIDE	TRANSITION
LIE_ON_SIDE_LIE_ON_BACK	TRANSITION
LIE_SIT	TRANSITION
STAIR_CLIMB	DYNAMIC
BEND_OVER_OR_SQUAT	DYNAMIC
WALK_BACKWARDS	DYNAMIC
JUMP	DYNAMIC

Tabelle 2.: Verwendete Beschriftungen und ihr Zustand

Assessment	Bewegungsbeschriftung
Frailty: 4,57m Gehgeschwindigkeit	WALK
TUG	WALK, STAND_SIT, SIT_STAND
SPPB: Statisches Gleichgewicht	STAND[1]
SPPB: 4m Gehgeschwindigkeit	WALK
SPPB: „5x Chair Rise“-Test	STAND_SIT, SIT_STAND
SCPT	STAIR_CLIMB
DEMMI: Bett: Brücke	ASSESSMENT_DEMMI_BED_BRIDGE[2]
DEMMI: Bett: Auf die Seite rollen	LIE_ON_BACK_LIE_ON_SIDE, LIE_ON_SIDE_LIE_ON_BACK[3]
DEMMI: Bett: Aufsetzen	LIE_SIT
DEMMI: Stuhl: Sitzen	SIT
DEMMI: Stuhl: Aufstehen (2x)	SIT_STAND
DEMMI: Statisches Gleichgewicht	STAND[1]
DEMMI: Gehen	WALK
DEMMI: Stift aufheben	BEND_OVER_OR_SQUAT[4]
DEMMI: Rückwärts gehen	WALK_BACKWARDS
DEMMI: Springen	JUMP
CMJ	JUMP
6 Minuten Gehgeschwindigkeit	WALK

1. Unmöglich von anderem Stehen unterscheidbar und daher als Assessment-Indikator unbrauchbar
2. Keine Alltagsbewegung, wird direkt als Assessment klassifiziert
3. Teilweise rollen sich die Probanden nicht zurück, sondern setzen sich direkt für die nächste Übung auf
4. Es ist dem Probanden überlassen, wie er den Stift aufhebt, daher ist die Bewegung nicht genau bekannt

Tabelle 3.: Zuordnung zwischen Assessments und Bewegungen

3.3.2. Vorlage zum Beschriften von Assessments

Mit der Vorlage zum Beschriften von Assessments anhand eines Beispieldatensatzes (siehe Abbildung 11) soll sichergestellt werden, dass die Trainingsdatensätze mindestens die Beschriftungen (im Folgenden auch Label genannt) enthalten, die innerhalb der jeweiligen Assessment-Tests für das Assessment relevant sind (beispielsweise sind Sitzen oder Stehen, während der Assessment-Test dem Probanden erklärt wird, für das eigentliche Assessment unerheblich). Darüber hinaus dient die Vorlage dazu, eine einheitliche Beschriftung zu gewährleisten, damit die Machine-Learning-Algorithmen entsprechend

trainiert werden können. Wenn nicht anders beschrieben, wird der Assessment-Test selbst (oder der Assessment-Test-Bestandteil) vom ersten Bewegungsmuster-Label bis zum letzten Bewegungsmuster-Label des jeweiligen Assessment-Tests (bzw. Assessment-Test-Bestandteils) beschriftet.

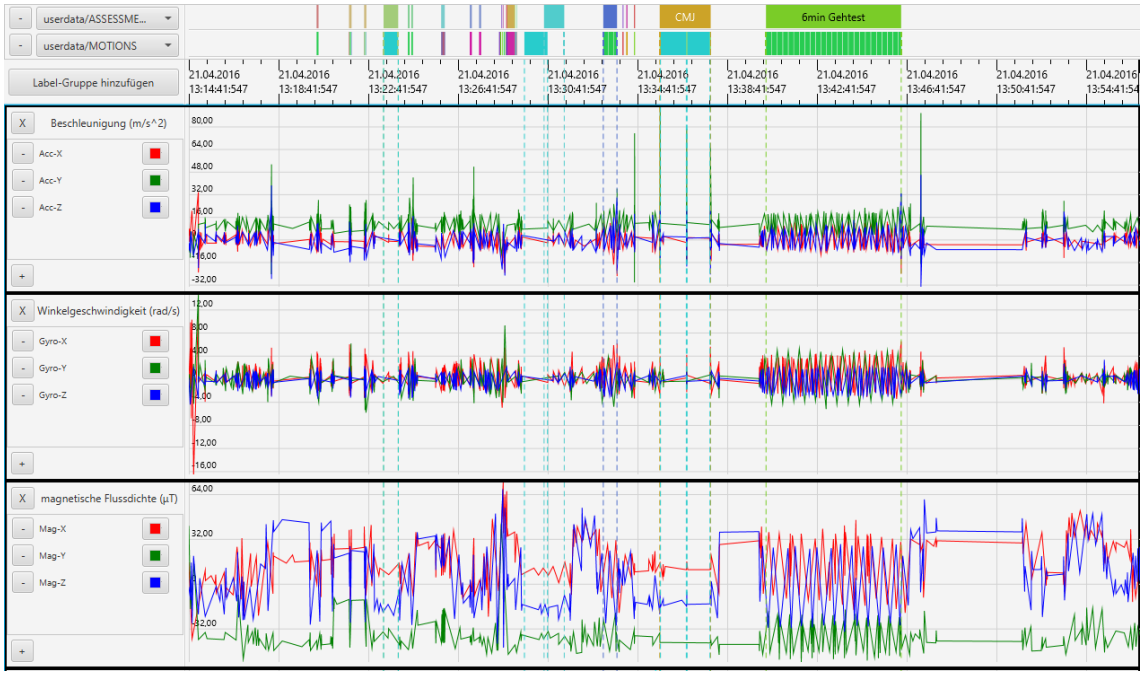


Abbildung 11.: Gesamt-Label-Vorlage

4,57-Meter-Gehgeschwindigkeitstest

Der *4,57-Meter-Gehgeschwindigkeitstest* ist ein Bestandteil des Frailty-Kriterien-Tests. Der Proband geht zur Startposition, dreht sich um und wartet stehend auf sein Startsignal. Gelabelt wird dann das „Gehen“ von der Start- bis zur Endposition. Danach dreht sich der Proband wieder um. Das Umdrehen lässt sich jeweils als Ausschlag der y-Achse des Gyroskops erkennen (siehe Abbildung 12).

Timed-Up-&-Go-Test

Der TUG wird zweimal nacheinander ausgeführt und jeweils einzeln als *TUG* gelabelt. Der Proband geht zum aTUG-Stuhl und setzt sich hin. Beim Startsignal steht der Proband auf (zu labeln als „Sitzen-Stehen“), geht ein paar Schritte (zu labeln als „Gehen“), dreht sich um (wird nicht gelabelt), geht wieder zurück (zu labeln als „Gehen“), dreht sich um (wird nicht gelabelt) und setzt sich wieder hin (zu labeln als „Stehen-Sitzen“, bis der Proband

Abbildung 12.: Label-Vorlage für Frailty-Test: 4,57m-Gehgeschwindigkeit

wieder ruhig sitzt). Dann wartet der Proband sitzend auf das erneute Startsignal (siehe Abbildung 13).

Short-Physical-Performance-Battery-Test

Der Short-Physical-Performance-Battery-Test unterteilt sich in einen Test zur Messung des *statischen Gleichgewichts*, einen *4-Meter-Gehgeschwindigkeitstest* und einen *5x-Chair-Rise-Test*.

Für die Messung des *statischen Gleichgewichts* geht der Proband zur Bodenkraftmessplatte, steigt darauf, dreht sich um und wartet stehend. Dort führt er dann jeweils auf das Startsignal hin nacheinander den Schluss-, Semitandem und Tandemstand aus (zu labeln als „Stehen“). Danach steigt der Proband von der Bodenkraftmessplatte (siehe Abbildung 14).

Auch der *4-Meter-Gehgeschwindigkeitstest* wird zweimal durchgeführt und entsprechend gelabelt. Ähnlich wie beim 4,57-Meter-Gehgeschwindigkeitstest geht der Proband auch hier zur Startposition, dreht sich um und wartet stehend. Mit dem Startsignal geht der Proband vier Meter (zu labeln als „Gehen“), dreht sich um und wartet wieder stehend auf das erneute Startsignal (siehe Abbildung 15).

Abbildung 13.: Label-Vorlage für TUG-Test

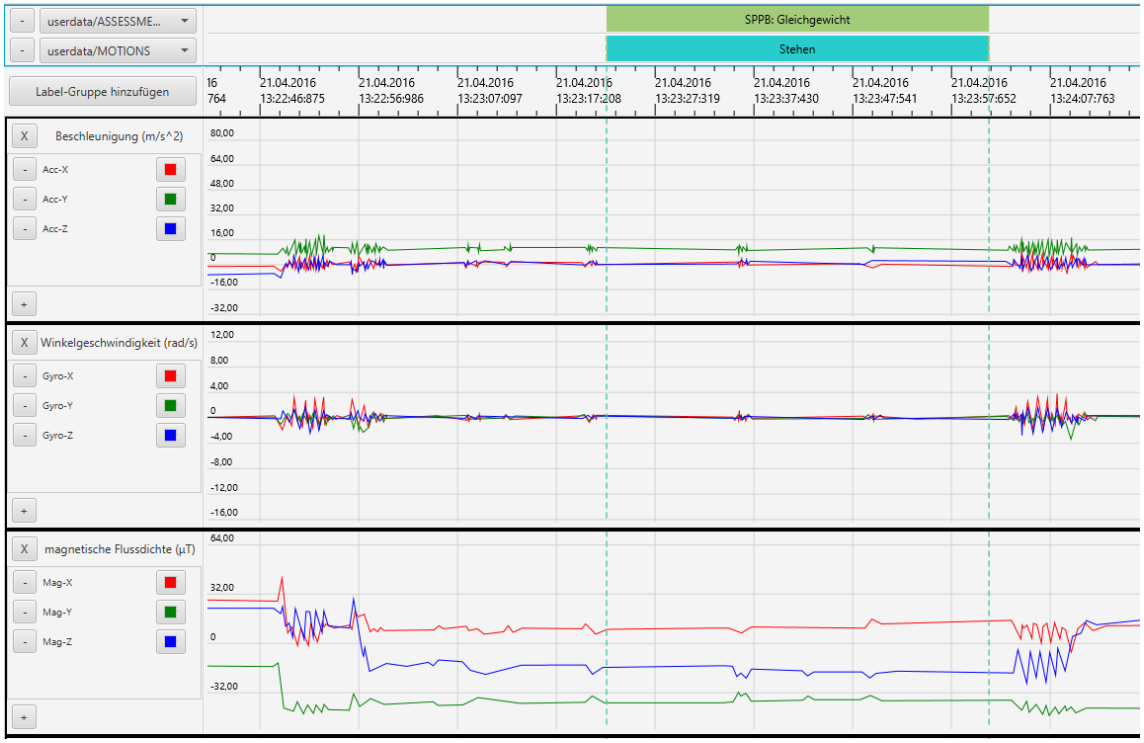


Abbildung 14.: Label-Vorlage für SPPB-Test: Statisches Gleichgewicht

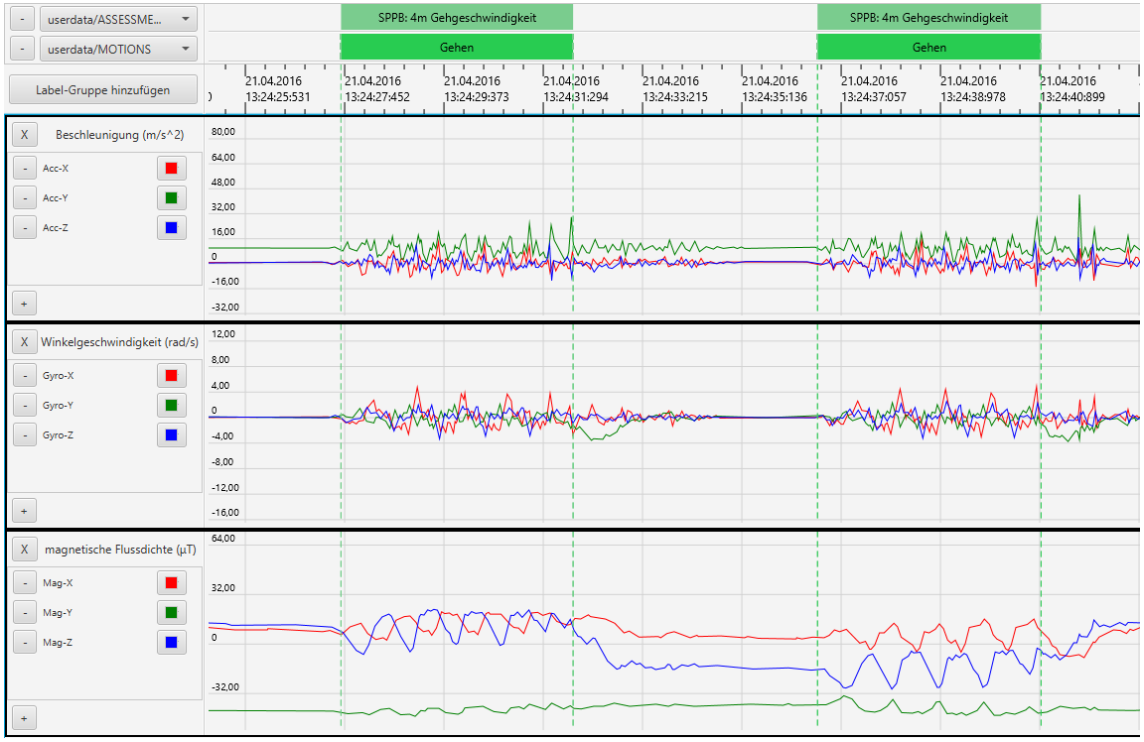


Abbildung 15.: Label-Vorlage für SPPB-Test: 4m-Gehgeschwindigkeit

Für den *5x-Chair-Rise-Test* geht der Proband zum aTUG-Stuhl und setzt sich hin. Es ist möglich, dass er vor dem eigentlichen Test einmal zur Probe aufsteht und sich wieder hinsetzt. Dann wartet der Proband sitzend auf das Startsignal. Daraufhin steht er auf (zu labeln als „Sitzen-Stehen“), setzt sich hin (zu labeln als „Stehen-Sitzen“) und wiederholt das ganze noch viermal. Danach wartet der Proband entweder sitzend oder er steht gleich wieder auf. Zu beachten ist hier, dass der *5x-Chair-Rise-Test* selbst nur bis einschließlich zum fünften Aufstehen gelabelt wird (siehe Abbildung 16).

Stair-Climb-Power-Test

Der *Stair-Climb-Power-Test* wird ebenfalls zweimal nacheinander ausgeführt und beide Male einzeln gelabelt. Hierfür geht der Proband zur Treppe, steigt sie hinab, dreht sich um und wartet stehend. Auf das Startsignal hin steigt er die Treppe hinauf (zu labeln mit „Treppensteigen“). Danach dreht der Proband sich um, steigt die Treppe wieder hinunter und dreht sich erneut um, um abermals stehend das Startsignal abzuwarten (siehe Abbildung 17).

Abbildung 16.: Label-Vorlage für SPPB-Test: 5x-Chair-Rise

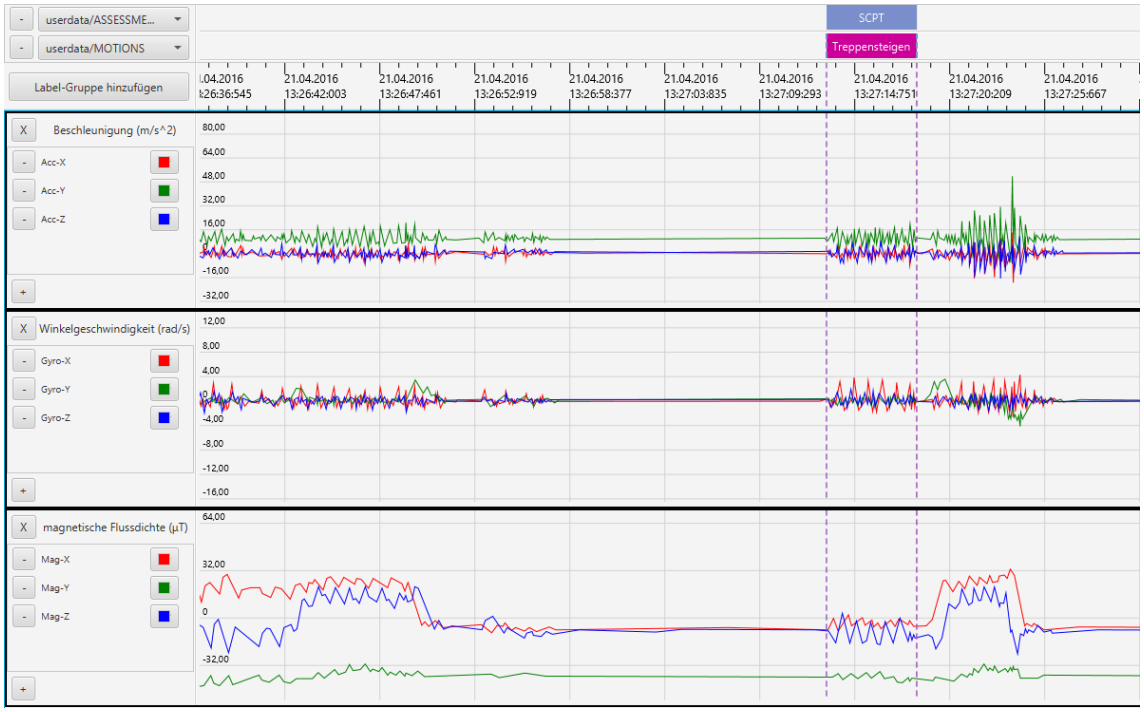


Abbildung 17.: Label-Vorlage für SCPT-Test

De Morton Mobility Index

Der De Morton Mobility Index besteht aus den folgenden Einzeltests: Betttest, Stuhltest, Test für das statische Gleichgewicht, Gehtest und Test für das dynamische Gleichgewicht.

Der Betttest teilt sich ebenfalls auf in drei Einzeltests (siehe Abbildung 18). Zunächst zieht sich der Proband die Schuhe aus (im Stehen oder Sitzen), geht zum Bett und dreht sich um. Dann legt er sich hin (entweder zu labeln als „Stehen-Liegen“ oder als „Stehen-Sitzen“ mit anschließendem „Sitzen-Liegen“) und wartet liegend (zu labeln als „Liegen (Rücken)“). Anschließend bildet der Proband eine Brücke (zu labeln als *DEMMI: Bett: Brücke*) und wartet dann wieder liegend (zu labeln als „Liegen (Rücken)“). Nun rollt er sich auf die Seite (zu labeln als *DEMMI: Bett: Rollen* und „Liegen (Rücken)-Liegen (Seite)“) und wartet dann wieder liegend (zu labeln als „Liegen (Seite)“). Es ist möglich, aber nicht notwendig, dass sich der Proband als nächstes wieder zurück auf den Rücken rollt (zu labeln als „Liegen (Seite)-Liegen (Rücken)“) und liegend wartet (zu labeln als „Liegen (Rücken)“). Im Anschluss setzt sich der Proband auf (zu labeln als *DEMMI: Bett: Aufsetzen* und als „Liegen-Sitzen“).

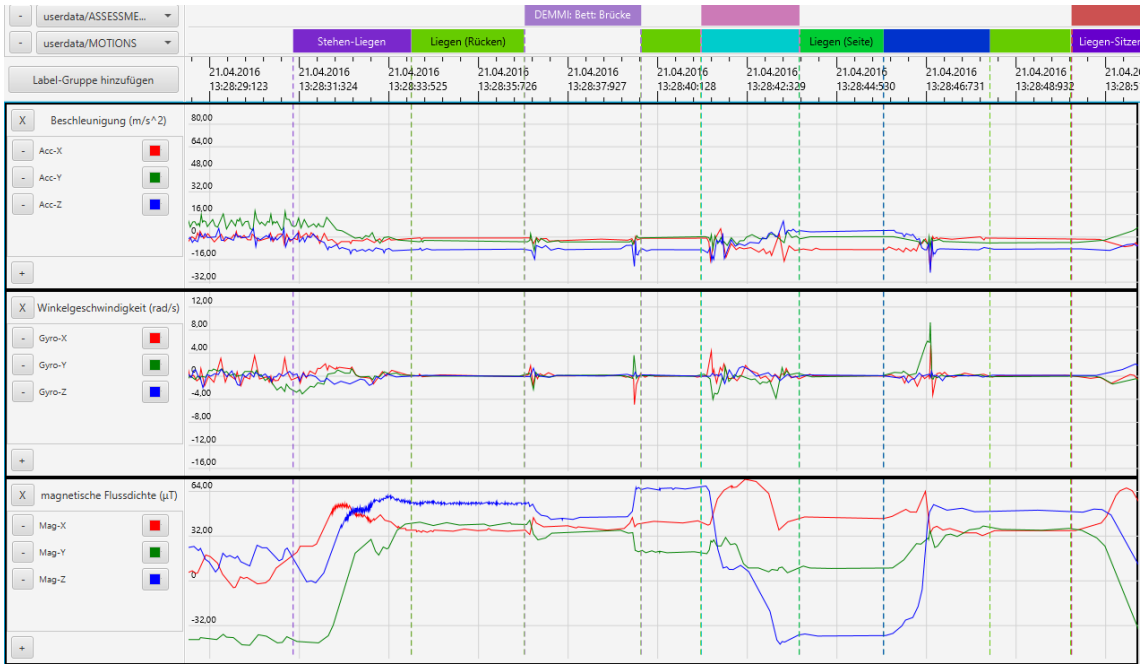


Abbildung 18.: Label-Vorlage für DEMMI-Test: Bett

Auch der Stuhltest beinhaltet zwei Einzeltests (siehe Abbildung 19). Hierfür sitzt der Proband am Anfang ohne Unterstützung auf dem Bett (zu labeln als *DEMMI: Stuhl: Sitzen* und als „Sitzen“). Dann steht er auf (zu labeln als *DEMMI: Stuhl: Aufstehen* und

als „Sitzen-Stehen“) und bleibt stehen (zu labeln als „Stehen“). Anschließend setzt er sich wieder hin (zu labeln als „Stehen-Sitzen“), bleibt sitzen (zu labeln als „Sitzen“) und steht dann, ohne die Arme zu Hilfe zu nehmen, wieder auf (zu labeln als *DEMMI: Stuhl: Aufstehen ohne Hilfe* und als „Sitzen-Stehen“). Danach soll der Proband zwar eigentlich zur Bodenkraftmessplatte gehen, aber in einigen Fällen setzt er sich vorher zwischendurch nochmal wieder aufs Bett.

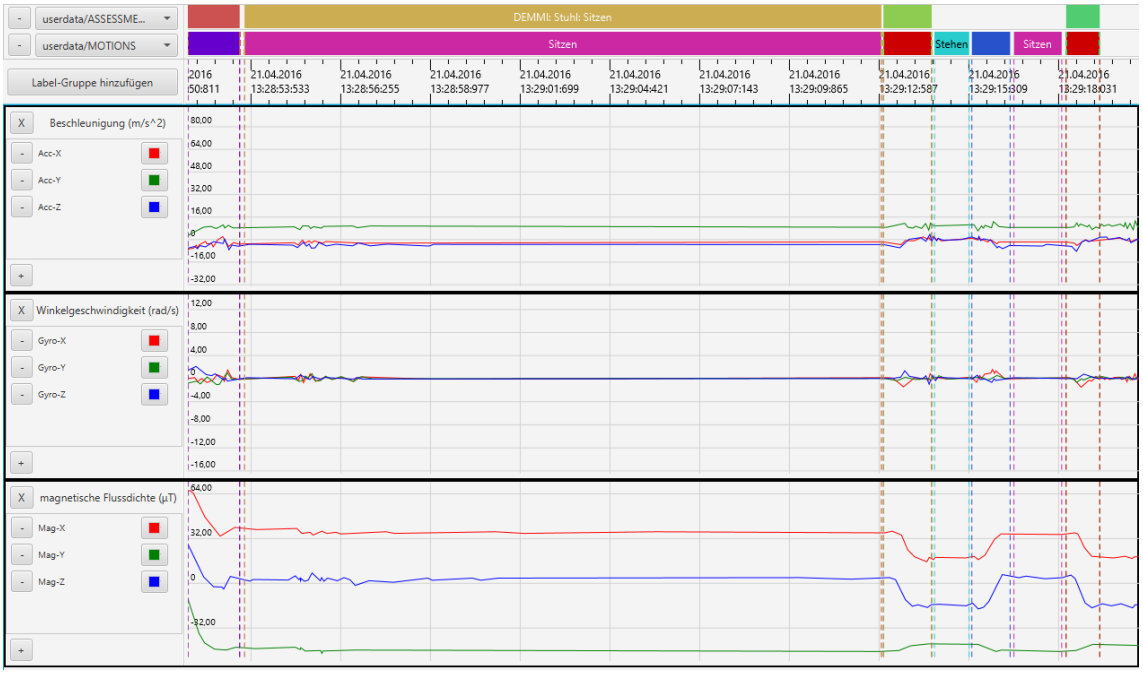


Abbildung 19.: Label-Vorlage für DEMMI-Test: Stuhl

Der Test für das statische Gleichgewicht besteht ebenso aus vier Einzeltests. Zunächst steigt der Proband auf die Bodenkraftmessplatte, dreht sich um und sobald er ruhig steht, wird mit „Stehen“ gelabelt (mit Ausnahme der Phasen, in denen der Proband zu sehr schwankt). Mit *DEMMI: stat. Gleichgewicht* wird durchgängig der Stand ohne Unterstützung, der Stand mit geschlossenen Füßen, der Stand auf Fußspitzen und der Tandemstand mit geschlossenen Augen gelabelt (siehe Abbildung 20). Danach steigt der Proband von der Bodenkraftmessplatte.

Für den 50-Meter-Gehtest zieht sich der Proband zunächst die Schuhe wieder an (im Stehen oder Sitzen). Danach begibt er sich zur Startposition und dreht sich um. Auf das Startsignal hin geht der Proband bis zum Ende des Flurs (zu labeln als „Gehen“) und dreht sich um (wird nicht gelabelt). Dies wird wiederholt, bis fünfzig Meter erreicht sind. Der gesamte Test wird mit *DEMMI: 50m Gehen* gelabelt (siehe Abbildung 21).

Auch der Test für das dynamische Gleichgewicht teilt sich in drei Einzeltests auf (siehe

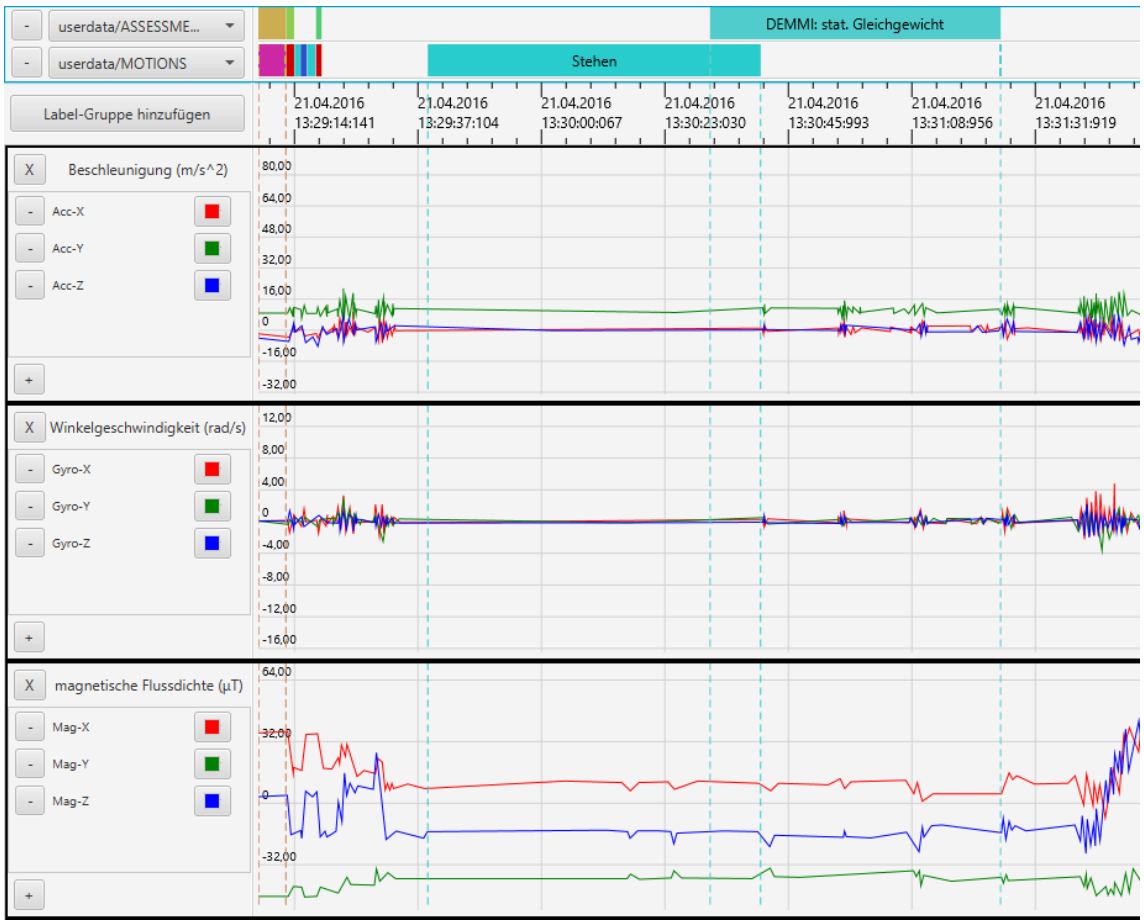


Abbildung 20.: Label-Vorlage für DEMMI-Test: Statisches Gleichgewicht

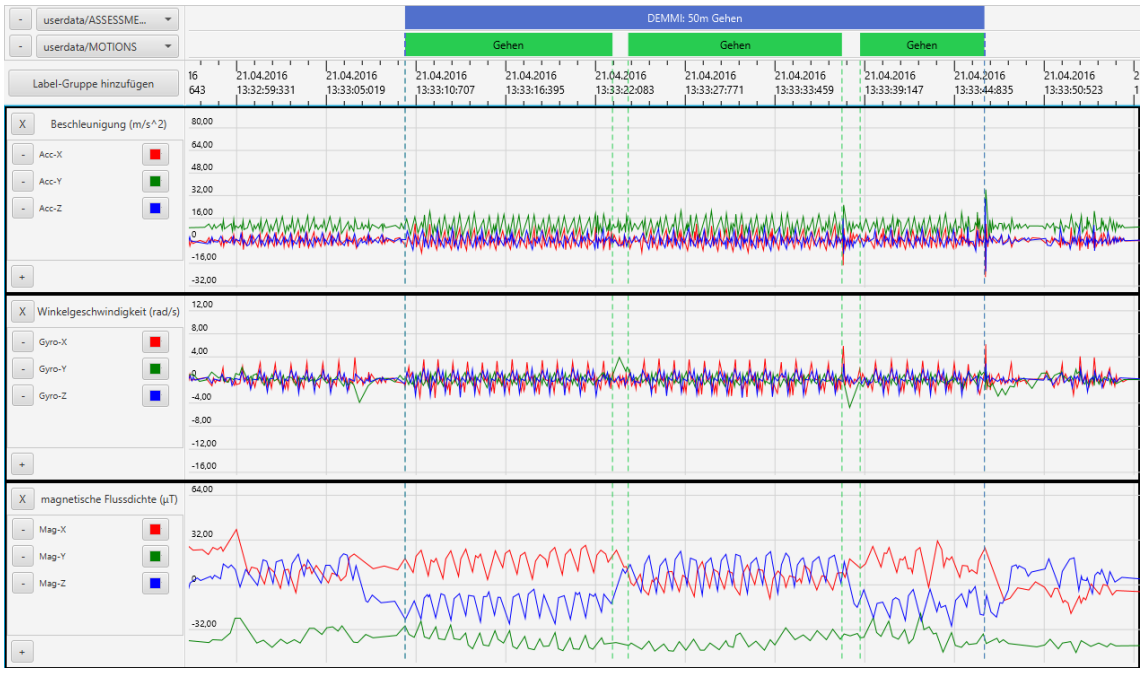


Abbildung 21.: Label-Vorlage für DEMMI-Test: Gehen

Abbildung 22). Zunächst wartet der Proband stehend (wird nicht gelabelt) und hebt dann einen Stift auf (zu labeln als *DEMMI: Stift aufheben* und als „Hinunterbeugen/Hocken“). Dann begibt er sich zur Startposition und wartet stehend. Beim Startsignal geht der Proband vier Schritte rückwärts (zu labeln als *DEMMI: Rückwärts gehen* und als „Rückwärts gehen“). Im Anschluss geht er zur Bodenkraftmessplatte, steigt darauf, dreht sich um und wartet stehend. Dort führt der Proband dann einen Sprung aus (zu labeln als *DEMMI: Springen* und als „Springen“). Danach bleibt er auf der Bodenkraftmessplatte stehen.

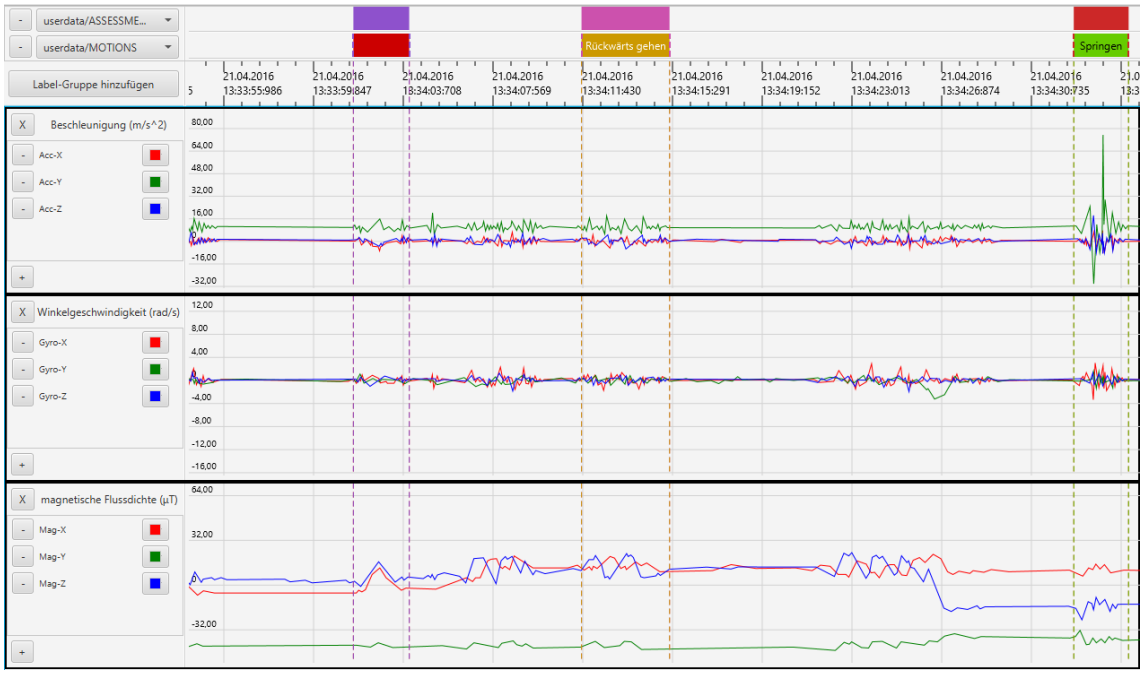


Abbildung 22.: Label-Vorlage für DEMMI-Test: Dynamisches Gleichgewicht

Counter Movement Jump

Für den CMJ-Test befindet sich der Proband bereits vom vorherigen Test auf der Bodenkraftmessplatte und wartet dort stehend. Beim Startsignal springt er (zu labeln mit „Springen“) und wartet dann stehend 60 Sekunden lang (zu labeln mit „Stehen“) bis zum nächsten Startsignal. Dies wird einmal wiederholt und im Anschluss noch ein drittes Mal gesprungen, was ebenfalls mit „Springen“ zu labeln ist. Danach steigt der Proband von der Bodenkraftmessplatte. Der ganze Test vom ersten bis zum dritten Sprung einschließlich wird mit *CMJ* gelabelt (siehe Abbildung 23).

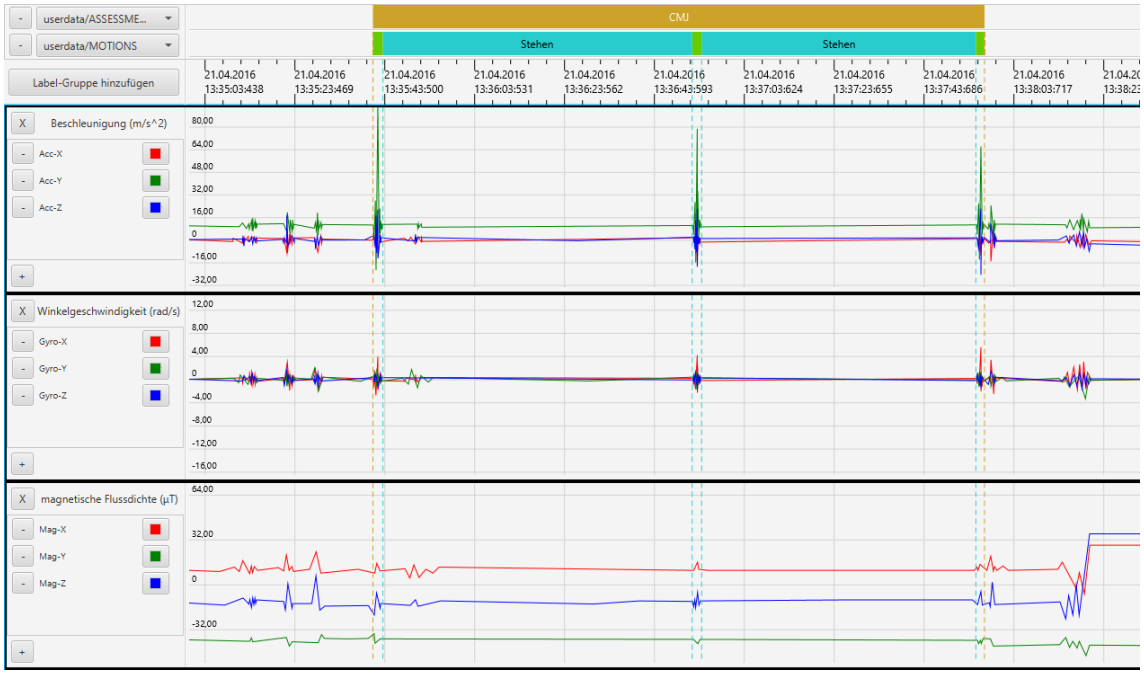


Abbildung 23.: Label-Vorlage für CMJ

6-Minuten-Gehtest

Für den 6-Minuten-Gehtest geht der Proband zur Startposition und wartet dort kurz stehend. Auf das Startsignal hin geht er bis zum Ende des Flurs (zu labeln mit „Gehen“) und dreht dort um (wird nicht gelabelt). Das Ganze wiederholt sich sechs Minuten lang. Dann bleibt der Proband kurz stehen. Der gesamte Test vom Loslaufen an der Startposition bis zum Stehenbleiben (ausschließlich) wird mit *6-Minuten-Gehtest* gelabelt (siehe Abbildung 24).

3.4. Aktivitätserkennung

Mit der Hilfe eines Algorithmus soll jedem Zeitpunkt eines Datensatzes die ausgeführte Aktivität des Probanden zugeordnet werden können. Ein solcher Algorithmus wird als Aktivitätserkennung bezeichnet und lässt sich in mehrere Teilschritte gliedern. Diese werden nacheinander abgearbeitet, wobei jede Phase Ergebnisse liefert, die wiederum als Eingangsdaten für die nächste Phase dienen. Der letzte Schritt gibt Labelinformationen aus, die Zeitintervalle mit einer Aktivität verknüpfen. Die folgenden Teilschritte werden dabei typischerweise durchlaufen:

1. Vorverarbeitung: Bei der Vorverarbeitung werden unterschiedliche Filter auf die Zeitreihen angewandt um Rauschen zu unterdrücken oder andere Optimierungen durchzufüh-

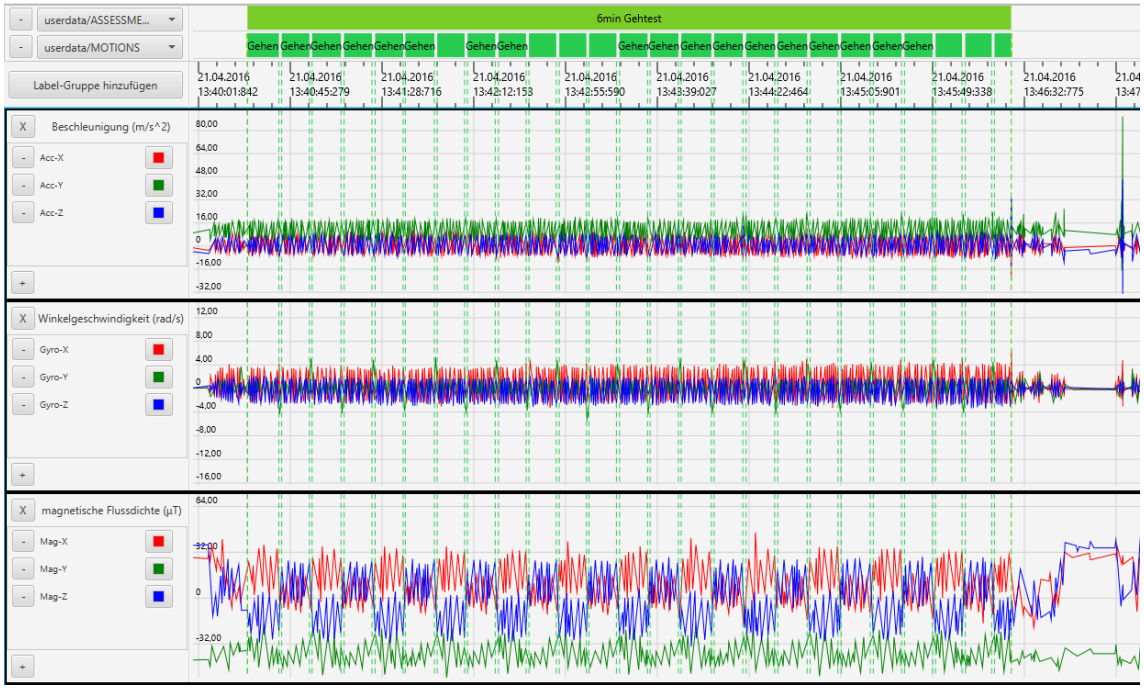


Abbildung 24.: Label-Vorlage für 6-Minuten-Gehtest

ren. Dadurch können die Ergebnisse der Aktivitätserkennung verbessert werden. Auch eine Normalisierung der Zeitreihen kann hier durchgeführt werden.

2. Segmentierung: Bei der Segmentierung wird die Zeitreihen-Repräsentation vereinfacht. Dadurch sollen die Zeitreihen weniger Speicherplatz benötigen, schneller analysierbar sein und schon erste Merkmale für die nachfolgende Merkmalsgewinnung liefern.

3. Merkmalsgewinnung: Bei der Merkmalsgewinnung werden Merkmale für Teilabschnitte der Zeitreihen (Segmente) berechnet. Die Merkmale sollten so gewählt werden, dass diese die unterschiedlichen Aktivitäten gut voneinander abgrenzen. Beispiele sind Mittelwert, Standardabweichung oder Autokorrelation (siehe Unterabschnitt 3.4.3). Die Gesamtheit aller Merkmale eines Segments wird Merkmalsvektor bezeichnet.

4. Klassifizierung: Jedem Segment wird mithilfe seines Merkmalsvektors eine Aktivität zugeordnet. Zum Einsatz kommen dabei Machine-Learning-Techniken wie Support Vector Machines, Gaussian Mixture Models oder Hidden Markov Models (siehe Unterabschnitt 3.4.6).

5. Nachbearbeitung: Zum Schluss werden benachbarte Segmente, denen dieselbe Aktivität zugeordnet wurde, zusammengefasst. Wenn sich benachbarte Segmente überlappen und mit unterschiedlichen Aktivitäten klassifiziert wurden, muss außerdem entschieden werden, welche Aktivität dem überlappenden Teil der Segmente zugeordnet wird. Des

Weiteren können hier auch Heuristiken angewandt werden. Diese dienen dazu, um kurzzeitige Fehlklassifikationen zu entfernen (z. B. ist ein Sprung für 0,1 Sekunden während des Treppensteigens wahrscheinlich ein Fehler).

Nachfolgend werden die oben genannten Teilschritte, die in ihrer Gesamtheit die Aktivitätserkennung darstellen, präziser erläutert.

3.4.1. Vorverarbeitung

In vielen Fällen ist es sinnvoll, dass die Rohdaten nicht direkt als Eingabe für die weiteren Schritte verwendet werden. Stattdessen wird zunächst ein Vorverarbeitungsschritt durchgeführt, der die Rohdaten auf verschiedene Arten transformieren kann. Beispielsweise kann mit einer Rauschunterdrückung das Rauschen aus den Rohdaten entfernt werden, was gegebenenfalls zu besseren Ergebnissen führt. Ein weiteres Beispiel wäre, dass nachgeschaltete Teilschritte Eingabedaten benötigen, die sich in einem bestimmten Intervall befinden. Dies kann durch eine Normalisierung erreicht werden.

In diesem Kapitel werden verschiedene Vorverarbeitungsschritte genauer beschrieben.

Normalisierung

Wie in Kapitel 6.3 im Unterabschnitt D.3 beschrieben, sollten die Eingabevektoren für künstliche neuronale Netze normalisiert sein. Das heißt, dass der Mittelwert aller Werte bei etwa null und die Standardabweichung bei knapp eins liegen.

Das kann einfach erreicht werden, indem beim Training eines Modells anhand der Trainingsdaten die Mittelwerte und Standardabweichungen der verschiedenen Features (Merkmale) berechnet werden. Subtrahiert man nun von den Features aller Trainingsdaten die entsprechenden Mittelwerte und teilt sie anschließend durch die Standardabweichungen, dann normalisieren sich die Werte auf das Intervall $(-1, 1)$. Die berechneten Mittelwerte und Standardabweichungen müssen für das trainierte Modell gespeichert werden, sodass sie bei der Klassifikation auch zur Normalisierung der zu klassifizierenden Daten verwendet werden können.

Rauschunterdrückung

Rauschunterdrückung ist eine der wichtigsten Aufgaben unter anderem in der Sensorik, Messtechnik und Telekommunikationstechnik. Durch Rauschunterdrückung können einerseits die Auflösung und Zuverlässigkeit eines Messsystems verbessert und andererseits die Messunsicherheit reduziert werden. Verschiedene Methoden werden in der Praxis zur Rauschunterdrückung verwendet, um das Signal/Rausch-Verhältnis (SNR) eines Messsystems zu verbessern. Analoge Filter, Trägerfrequenzverfahren und Integrationsverfahren gehören zur typischen analogen Signalverarbeitung. Sie werden durch die Schaltungstechnik implementiert. Mittelwertbildung, numerisches Glätten und digitale Filter werden häufig durch digitale Signalverarbeitung realisiert.

Digitale Filterung Mit einem Filter können Signale unterschiedlicher Frequenz voneinander getrennt werden. Für die analoge Signalverarbeitung werden z. B. für eine Reihe von Filterfunktionen praktische Filterausführungen angegeben. Digitale Filterung wird beispielsweise mit einem Prozessrechner vorgenommen, um nach der Abtastung eines Messsignals die Messwerte zu glätten, die Bandbreite des Messsignals zu begrenzen, nach einer bestimmten Frequenz im Messsignal zu suchen oder um das Signal/Rausch-Verhältnis zu verbessern.

Tiefpass-Filter Ein Tiefpass-Filter ist ein Filter, der Signale unterhalb einer Grenzfrequenz passieren lässt und Signale oberhalb der Grenzfrequenz dämpft. Durch Entfernung einiger Frequenzen erzielt der Filter einen Glättungseffekt. Das heißt, der Filter erzeugt langsame Änderungen in den Ausgabewerten, um die Anzeige von Trends zu vereinfachen und das gesamte Signal/Rausch-Verhältnis bei minimaler Signalverschlechterung zu erhöhen.

Hochpass-Filter Ein Hochpass-Filter (auch als Tiefensperre bezeichnet) dämpft Signale unterhalb einer Grenzfrequenz (dem Sperrbereich) und lässt Signale oberhalb der Grenzfrequenz (dem Durchlassbereich) passieren. Das Ausgangssignal dieses Filters ist direkt proportional zur Änderungsrate des Eingangssignals. Hochpass-Filter werden oft verwendet, um niederfrequente Störsignale zu bereinigen, niederfrequente Trends aus Zeitreihendaten zu entfernen und so die hochfrequenten Trends hervorzuheben.

Gauß-Filter In der Elektronik und Signalverarbeitung ist ein Gauß-Filter ein Filter, dessen Impulsantwort eine Gauß-Funktion, wie in Gleichung 3.1 beschrieben, ist.

$$\int_{-\infty}^{\infty} e^{-\frac{1}{2}t^2} dt = \sqrt{2\pi} \tag{3.1}$$

Gauß-Filter haben die Eigenschaften keine Überschwinger zu einem Schritt des Funktionseingangs bei gleichzeitiger Minimierung der Anstiegs- und Abfallzeit zu haben. Dieses Verhalten hängt eng mit der Tatsache zusammen, dass der Gauß-Filter eine minimal mögliche Gruppenverzögerung hat. Mathematisch gesehen verändert der Gauß-Filter das Eingangssignal durch Faltung mit einer Gauß-Funktion.

Kalman-Filter Der Kalman-Filter ist ein Algorithmus, der den Zustand eines Systems anhand von Messdaten schätzt. Dieser Filter wurde vor allem vom ungarischen Mathematiker Rudolf Kálmán entwickelt, nach dem er benannt ist. Der Algorithmus des Filters ist ein zweistufiger Prozess: Zuerst wird der Zustand des Systems vorausgesagt und dann wird Messrauschen verwendet, um die Schätzung des Systemzustands zu verfeinern. Der Kalman-Filter arbeitet auf einer kontinuierlichen Folge von Messwerten, die aus einem beliebigen Verfahren stammen können. Zum Zeitpunkt t existiert eine Vorhersage x_t, die am Ende von Schritt $t-1$ erstellt wurde. Außerdem liefert das gewählte Verfahren zu jedem Zeitpunkt t einen Messwert z_t. Die Kalman-Gain-Matrix K gewichtet nun abhängig vom Fehler P, ob man dem Messwert z_t oder dem zugrunde liegenden Bewegungsmodell, d.h. der Vorhersage, mehr vertraut. Messwert und Vorhersage werden dann zu einem Schätzwert fusioniert. Der Kalman-Filter ist ein stochastischer Zustandsschätzer mit iterativer Struktur, der in Echtzeitsystemen angewendet werden kann. Es wird angenommen,

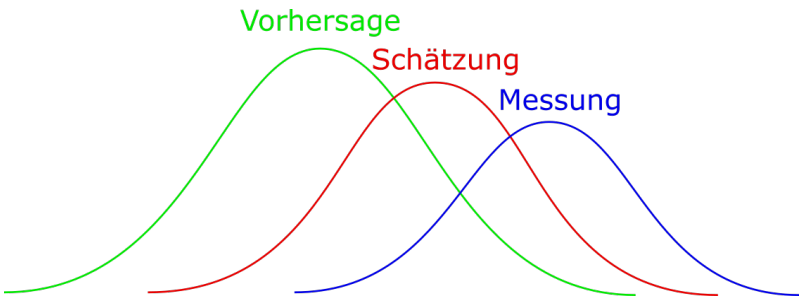


Abbildung 25.: Fusionierung von Vorhersage und Messung

dass Vorhersage und Messung normalverteilt sind. Anhand des Erwartungswerts und der Varianz werden die beiden Werte fusioniert.

Wavelet-Transformation Wavelets („kleine Wellen“) sind mathematische Funktionen, welche ähnlich wie die Sinus- und Kosinus-Funktionen der Fourier-Transformation dazu benutzt werden, eine gegebene Funktion (Signal) auf die in ihr enthaltenen Frequenzbestandteile hin zu untersuchen. Zu diesem Zweck können Wavelets gestreckt beziehungsweise gestaucht (skaliert) werden oder an eine bestimmte Stelle des Signals verschoben werden. In der Transformation selbst kommen dann die entsprechend skalierten und örtlich verschobenen Varianten des Basis-Wavelets zum Einsatz. Diese Varianten werden auch allgemein als Waveletfamilie bezeichnet. Bei der kontinuierlichen Wavelet-Transformation (Continuous Wavelet-Transform: CWT) beruht die Analyse des Signals auf der Erkenntnis, dass die Wavelets beim Strecken oder Stauchen ihre Frequenz ändern, wodurch sie sich den verschiedenen Signalkomponenten automatisch anpassen können. Somit wird bei der Analyse das Signal zunächst mit einem stark gestreckten (also hoch skalierten) Wavelet untersucht, um die tiefen Frequenzen (Grundfrequenzen) zu bestimmen. Anschließend komprimiert man das Wavelet schrittweise – es wird also in verschiedenen Auflösungen dargestellt – um immer feinere Frequenzen zu erfassen. Die durch vollständige Abtastung des Signals mit einem entsprechend skalierten Wavelet entstandene Funktion wird als Wavelet-Transformierte bezeichnet. Sie beschreibt die Übereinstimmung des Signals an allen Stellen mit dem analysierenden Wavelet (mit einem bestimmten Skalierungsfaktor).

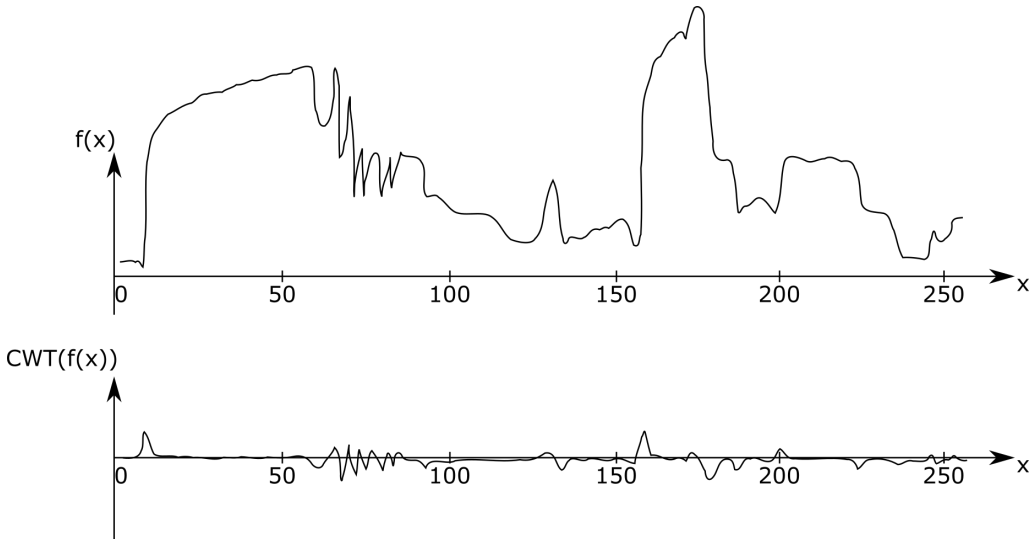


Abbildung 26.: Beispiel für eine Wavelet-Transformation. Die Funktion beschreibt das Ausgangssignal. Die zweite Funktion stellt die jeweiligen Wavelet- Transformierten dar

Stückweise lineare Repräsentation Die intuitive stückweise lineare Darstellung bezieht sich auf die Approximation einer Zeitreihe T, der Länge n, mit K geraden Linien. Da K typischerweise viel kleiner als n ist, macht diese Darstellung die Speicherung, Übertragung und Berechnung der Daten effizienter.

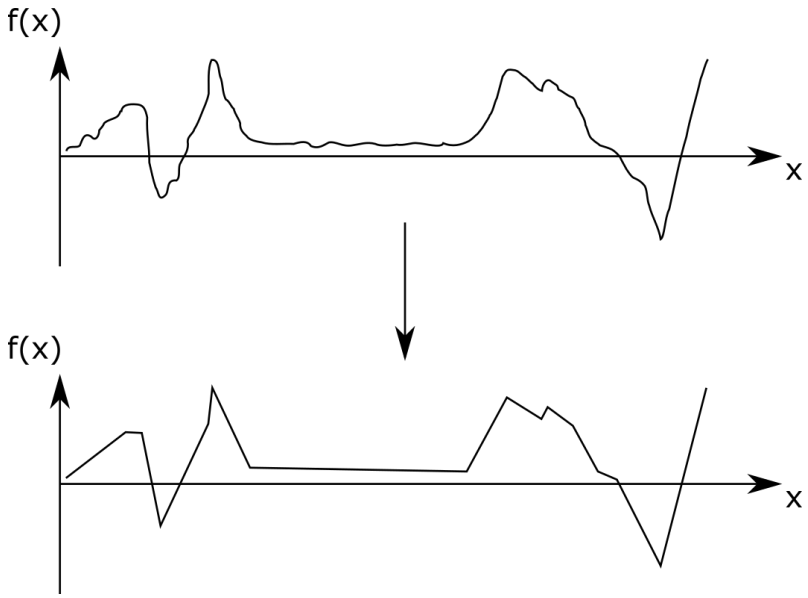


Abbildung 27.: Stückweise lineare Repräsentation von zwei Zeitreihen A) Space Shuttle Telemetry B) Electrocardiogram

Speziell im Rahmen des Data Mining wurde die stückweise lineare Darstellung verwendet, z. B. für:

- die Unterstützung der schnellen Suche von gleichen Zeitreihenteilen
- die Unterstützung neuartiger Distanzmessungen für Zeitreihen, wie „Fuzzy-queries“, „weighted queries“, „multiresolution queries“, „dynamic time warping“ und „relevance feedback“
- die Unterstützung vom gleichzeitigen Datamining von Text- und Zeitreihen
- die Unterstützung neuartiger Clustering- und Klassifikationsalgorithmen

3.4.2. Segmentierung

Die Segmentierung ist ein Verfahren, bei der eine Eingangszeitreihe in eine Folge von Segmenten unterteilt wird [Her10, S. 164ff]. Dies dient vor allem der Vereinfachung der Zeitreihen-Repräsentation, da die Segmente weniger Speicherplatz benötigen, schneller analysierbar sind und bereits erste Merkmale für die nachfolgende Merkmalsgewinnung

liefern. Es wird zwischen Online- und Offline-Segmentierung unterschieden. Online-Segmentierung kann einen eingehenden Datenstrom verarbeiten, während der Offline-Segmentierung der gesamte Datensatz zur Verfügung stehen muss. Einige Verfahren zur Segmentierung werden nachfolgend kurz vorgestellt.

Top-Down (Offline) Bei Top-Down-Algorithmen werden Zeitreihen rekursiv in mehrere Segmente unterteilt, indem die Daten an der bestmöglichen Stelle gespalten werden. Die Daten werden jeweils halbiert, bis alle Segmente einen Approximations-Fehler unter der vorher definierten Grenze besitzen. Die Komplexität liegt in O(n^2K), wobei K die Anzahl der Segmente ist. Der Top-Down-Algorithmus entspricht in etwa dem Douglas-Peucker-Algorithmus (siehe Abschnitt 6.1.5) [Her10, S. 165] [KCHP01, S. 7].

Bottom-Up (Offline) Die Segmentierung beginnt mit der genauesten Approximation, dies sind n/2 Segmente bei einer Zeitreihe der Länge n. Benachbarte Segmente werden anschließend zusammengeführt und bilden ein größeres Segment. Dieses Segment wird solange vergrößert, bis ein Fehlerkriterium erreicht wird. Die Komplexität liegt in O(nL), wobei L die durchschnittliche Länge eines Segments ist [Her10, S. 165ff] [KCHP01, S. 7f].

Sliding Window (SW) (Online) Bei Sliding Window (SW) wird nur ein „Fenster“ , ein Ausschnitt der Zeitreihe betrachtet. Beginnend mit einer Teilsequenz dieser Zeitreihe werden neue Datenpunkte hinzu addiert, bis ein vorher festgelegtes Fehlerkriterium erreicht wird. Diese bereits addierten Datenpunkte bilden ein Segment und der Vorgang wird danach für alle weiteren Punkte wiederholt [KCHP01, S. 7]. Dieses Sliding-Window-Verfahren hat nichts mit dem gleichnamigen Verfahren für die Merkmalsgewinnung in Unterabschnitt 3.4.3 zu tun.

Sliding Window and Bottom-Up (SWAB) (Online) Dies stellt eine Kombination aus Sliding Window und Bottom-Up dar, die das Online-Verhalten von SW und die Vorteile von Bottom-Up vereint. Es wird zuerst SW ausgeführt und anschließend wird auf jedes ermittelte Segment hieraus das Bottom-Up-Verfahren angewendet [KCHP04].

3.4.3. Merkmalsgewinnung

Bei der Merkmalsgewinnung wird eine große Anzahl von Eingangsdaten in eine reduzierte Menge von Merkmalen, dem Merkmalsvektor, transformiert. Das Ziel ist dabei Merkmale

zu finden, die die Hauptcharakteristiken beschreiben und die Eingangsdaten so genau wie möglich repräsentieren. Merkmale können als eine Abstraktion der Eingangsdaten betrachtet werden und sind z. B. der Mittelwert oder die Standardabweichung über den Eingangsdaten [ABMP+10].

Bei der Aktivitätserkennung sind die Eingangsdaten der Merkmalsgewinnung die Daten eines Teilabschnittes (Segment) der Zeitreihen. Für das Segment wird bei der Merkmalsgewinnung der Merkmalsvektor berechnet, der als Eingabe für einen Klassifikations-Algorithmus dient [ABMP+10].

Um die Zeitreihe in Teilabschnitte zu unterteilen, wird hier ein einfaches Sliding-Window-Verfahren angewendet. Bei dem Verfahren wird ein Fenster einer festen Länge (z. B. 2 Sekunde) am Anfang der Zeitreihen positioniert. Das Fenster wird über die Zeitreihen mit einer festen Schrittweite (z. B. 0,5 Sekunden) verschoben und erzeugt so die Segmente, die die Eingaben der Merkmalsgewinnung bilden.

Die von uns für die Aktivitätserkennung verwendeten Merkmale werden nachfolgend beschrieben.

Autokorrelation (AC) Die Autokorrelation beschreibt die Korrelation einer Funktion oder eines Signals mit sich selbst zu einem früheren Zeitpunkt. Sie gibt also an, wie viel Ähnlichkeit die um die Zeit l verschobene Folge X_{i-l} mit der ursprünglichen Folge X_i hat [MSGK14]. Die Autokorrelation wird hier eingesetzt, um sich wiederholende Muster in einem Zeitreihen-Segment zu erkennen. Dadurch lassen sich periodische Aktivitäten (z. B. Gehen) von nicht-periodischen Aktivitäten (z. B. Hinsetzen) unterscheiden. Der Wertebereich der Autokorrelation ist $[0, 1]$, wobei 0 keine Ähnlichkeit und 1 die Gleichheit der beiden Folgen bedeutet. Eine effiziente Berechnung kann mithilfe der Fast Fourier Transformation (FFT) durchgeführt werden und sieht wie folgt aus:

$$
\begin{enumerate}
\item $AC = \text{FFT}(X)$
\item $AC_i = (Re(AC_i) + Im(AC_i)i) \cdot (Re(AC_i) - Im(AC_i)i)$

für jedes Element $AC_1, ..., AC_n$ von AC

mit $Re(AC_i) =$ reeller Anteil von AC_i

und $Im(AC_i) =$ imaginärer Anteil von AC_i
\item $AC = \text{iFFT}(AC)$ mit iFFT = inverse FFT
\item $AC_i = \frac{AC_i}{AC_1}$ für jedes Element $AC_1, ..., AC_n$ von AC
\item $AC = \{AC_2 ... AC_{l+1}\}$
\end{enumerate}
\tag{3.2}
$$

Ein wichtiger Parameter ist die Zeitdifferenz (engl. lag) l mit der die Folge verschoben werden soll. Da die Periodendauer verschiedener Aktivitäten unterschiedlich sein kann, ist der lag-Parameter hier als maximale Zeitdifferenz zu verstehen. Das bedeutet, dass die Autokorrelation für jede Zeitdifferenz i in $[1, l]$ berechnet wird. Das Ergebnis ist ein Vektor mit den Autokorrelationen der unterschiedlichen Zeitdifferenzen.

Korrelation (Cor) Die Korrelation ist ein Begriff aus der Statistik und der Signalverarbeitung und beschreibt die Beziehung zwischen zwei Funktionen oder Signalen. Um den Grad der Beziehung zweier Zeitreihen-Segmente X und Y mit den Messwerten $x_m, x_{m+1}, ..., x_n$ und $y_m, y_{m+1}, ..., y_n$ zu bestimmen, wird der empirische Korrelationskoeffizient wie folgt berechnet:

$$
Cor(X,Y) = \frac{(n-(m-1)) \cdot \sum_{i=m}^{n}(x_i y_i) - (\sum_{i=m}^{n} x_i) \cdot (\sum_{i=m}^{n} y_i)}{\sqrt{((n-m) \cdot \sum_{i=m}^{n} x_i^2 - (\sum_{i=m}^{n} x_i)^2) \cdot ((n-m) \cdot \sum_{i=m}^{n} y_i^2 - (\sum_{i=m}^{n} y_i)^2)}}
\tag{3.3}
$$

Der empirische Korrelationskoeffizient gibt dabei die Stärke der Beziehung der beiden Zeitreihen X und Y an [Bor05, S.206]. Die Formel wurde der deskriptiven Statistik entnommen und besitzt im Gegensatz zur Formel der induktiven Statistik keine Bessel-Korrektur. Bei der Bessel-Korrektur ist der Normierungsfaktor $1/(n-(m-1)-1)$ statt $1/(n-(m-1))$, was für eine bessere Schätzung der Varianz einer Gesamtpopulation anhand einer Stichprobe sorgt. Da dies allerdings in unserem Anwendungsfall keine

Rolle spielt und uns nur der Grad der Korrelation zweier Zeitreihen interessiert, können wir auf die Bessel-Korrektur verzichten.

Arithmetisches Mittel ($\overline{x}$) Als weiteres Merkmal kann das arithmetische Mittel [Kem11, S.32] eines Zeitreihen-Segmentes X mit den Messwerten $x_m, x_{m+1}, ..., x_n$ berechnet werden. Es dient zur Bestimmung des Durchschnittswertes und berechnet sich wie folgt:

$$\overline{x}(X) = \frac{1}{(n-(m-1))} \cdot \sum_{i=m}^{n} x_i \tag{3.4}$$

Pitch (P) Der Pitch (Neigungswinkel) ist ein Merkmal, das die Neigung des Sensorgürtels zwischen der vertikalen Achse des Körpers und der Gravitationsachse zu einem bestimmten Zeitpunkt t beschreibt. Wenn der Proband zu diesem Zeitpunkt steht, sollte der Neigungswinkel 0 Radiant betragen. Der Neigungswinkel wird auf Basis der Y-Achse des Accelerometers AY (mit den Messwerten $ay_m, ay_{m+1}, ..., ay_n$), der Z-Achse des Accelerometers AZ (mit den Messwerten $az_m, az_{m+1}, ..., az_n$) und der X-Achse des Gyrometers GX (mit den Messwerten $gx_m, gx_{m+1}, ..., gx_n$) mithilfe eines Komplementärfilters bestimmt. Dieser muss sich zunächst einschwingen und liefert den Neigungswinkel zum Zeitpunkt der Zeitreihen-Segment-Enden ($t = n$). Da der Neigungswinkel zum Zeitpunkt der Zeitreihen-Segment-Mitten ($t = \left\lfloor \frac{m+n+1}{2} \right\rfloor$) bestimmt werden soll, wird das Zeitintervall bei der Berechnung um $\left\lfloor \frac{n-(m-1)}{2} \right\rfloor$ nach links verschoben. Die Berechnung sieht dann wie folgt aus:

1. $P = 0$
2. for i in $max\left(0, \left\lfloor m - \frac{n-(m-1)}{2} \right\rfloor\right) : \left\lfloor m + \frac{n-(m-1)}{2} \right\rfloor$ do
   $P = \alpha \cdot (P + gx_i \cdot dt) + (1-\alpha) \cdot atan2(ay_i, az_i)$
   end

(3.5)

Dabei gibt der α-Parameter an, wie stark das Gyrometer-Zeitreihen-Segment in die Berechnung einfließen soll. Der Parameter kann im Intervall $[0, 1]$ eingestellt werden. Der dt-Parameter gibt die Zeitdauer zwischen zwei aufeinanderfolgenden Messwerten der Zeitreihen an. Des Weiteren ist $atan2$ eine Abwandlung der Arkustangens-Funktion:

$$atan2(x,y) = \begin{cases} arctan(\frac{y}{x}) & \text{falls } x > 0 \\ arctan(\frac{y}{x}) + \pi & \text{falls } x < 0 \text{ und } y \geq 0 \\ arctan(\frac{y}{x}) - \pi & \text{falls } x < 0 \text{ und } y < 0 \\ \frac{\pi}{2} & \text{falls } x = 0 \text{ und } y > 0 \\ -\frac{\pi}{2} & \text{falls } x = 0 \text{ und } y < 0 \\ 0 & \text{falls } x = 0 \text{ und } y = 0 \end{cases} \tag{3.6}$$

Die Verwendung von $atan2$ hat den Vorteil, dass die Werte zweier Achsen eines Sensors zu einem Zeitpunkt einfach übergeben werden können und der richtige Quadrant und somit auch der korrekte Wert in Gegensatz zu $atan$ immer berechnet werden kann.

Quadratisches Mittel (RMS) Das quadratische Mittel [Kem11, S.33f] (engl. Root Mean Square (RMS)) ist ein Mittelwert, bei dem größere Zahlen stärker in das Ergebnis einfließen. Wird es auf ein Zeitreihen-Segment X mit den Messwerten $x_m, x_{m+1}, ..., x_n$ angewendet, sieht die Berechnungsvorschrift wie folgt aus:

$$RMS(X) = \sqrt{\frac{\sum_{i=m}^{n} x_i^2}{(n-(m-1))}} \tag{3.7}$$

Signal Energy (SE) Mithilfe der SE können Aktivitäten mit hoher Energie (z. B. Springen) von Aktivitäten mit wenig Energie (z. B. Sitzen) unterschieden werden [ABMP+10]. Auf ein Zeitreihen-Segment X mit den Messwerten $x_m, x_{m+1}, ..., x_n$ angewendet, wird die SE wie folgt berechnet:

$$SE(X) = \sum_{i=m}^{n} x_i^2 \tag{3.8}$$

Signal Magnitude Area (SMA) SMA ist ebenfalls ein Merkmal um Aktivitäten mit hoher Energie (z. B. Springen) von Aktivitäten mit wenig Energie (z. B. Sitzen) unterscheiden zu können [ABMP+10]. Angewendet auf ein Zeitreihen-Segment X mit den Messwerten $x_m, x_{m+1}, ..., x_n$ wird es wie folgt berechnet:

$$SMA(X) = \frac{1}{(n-(m-1))} \cdot \sum_{i=m}^{n} |x_i| \tag{3.9}$$

Signal Vektor Magnitude (SMV) Die SMV ist ein Grad für die Bewegungsintensität und ist daher beispielsweise ein wichtiger Wert für die Erkennung eines Sturzes [ABMP+10]. Es wird auf einer Menge von Zeitreihen-Segmente $X_1, ..., X_l$ berechnet. Beispielsweise kann die SMV für ein drei-dimensionales Accelerometer berechnet werden, so dass Bewegungen in allen drei Dimension einfließen. Für die Zeitreihen-Segmente $X_1, ..., X_l$ mit den Messwerten $x_{(1,m)}, x_{(1,m+1)}, ..., x_{(1,n)}, ..., x_{(l,m)}, x_{(l,m+1)}, ..., x_{(l,n)}$ berechnet sich die SMV wie folgt:

$$SMV(X_1, ..., X_l) = \sum_{i=m}^{n} \sqrt{\sum_{j=0}^{l} x_{(j,i)}^2} \tag{3.10}$$

Spektralentropie (PSE) Die PSE ist ein Merkmal zur Unterscheidung von Aktivitäten mit ähnlichen Energiewerten. Ausgangspunkt ist die Anwendung der Fast-Fourier-Transformation zur Abbildung des Zeitreihen-Segmentes X in den Frequenzbereich. Anschließend wird die Shannon-Funktion auf das transformierte Zeitreihen-Segment angewendet. Bei einem absolut unregelmäßigen Zeitreihen-Segment kommt es zur Gleichverteilung der unterschiedlichen Frequenzbereiche und die PSE ist 1. Anderenfalls, also bei einem absolut regelmäßigen Zeitreihen-Segment wie beispielsweise einer einzigen Sinuswelle, beträgt die PSE 0. Die Berechnung der PSE wird wie folgt durchgeführt:

$$\begin{aligned}
&1. \quad P = \text{FFT}(X) \\
&2. \quad P_i = \frac{P_i^2}{N} \quad \text{für jedes Element } P_1, ..., P_n \text{ von } P \\
&3. \quad P_i = \frac{P_i}{\sum_{i=1}^{n} P_i} \quad \text{für jedes Element } P_1, ..., P_n \text{ von } P \\
&4. \quad PSE = -dot(P, log(P)) \quad \text{mit} \quad dot(X, Y) = \sum_{i=1}^{n} x_i \cdot y_i
\end{aligned} \tag{3.11}$$

Dabei ist N ein Parameter, der angibt in wie viele Frequenzbereiche das Frequenzspektrum aufgetrennt werden soll. [ABMP+10]

Standardabweichung (s) Die Standardabweichung [Kem11, S.389] ist ein Maß für die Streuung der Messwerte $x_m, x_{m+1}, ..., x_n$ eines Zeitreihen-Segmentes X. Es kann als Merkmal eingesetzt werden, um statische Aktivitäten (z. B. Stehen) von dynamischen Aktivitäten (z. B. Gehen) unterscheiden zu können. Die Standardabweichung wird wie

folgt berechnet:

$$s(X) = \sqrt{\frac{1}{(n-(m-1))-1} \cdot \sum_{i=m}^{n}(x_i - \overline{x})} \quad mit \quad \overline{x} = \frac{1}{n-(m-1)} \cdot \sum_{i=m}^{n} x_i \tag{3.12}$$

3.4.4. Ausgleich der Klassenimbalance

Die zur Verfügung stehenden Bewegungsdaten haben eine problematische Klassenimbalance: Für einige Bewegungsmuster wie z. B. WALK steht Datenmaterial vieler Stunden zur Verfügung, weil sie Teil vieler Assessments waren oder schlicht über einen längeren Zeitraum ausgeführt wurden. Andere Bewegungen wie z. B. JUMP treten pro Assessment nur viermal auf und haben Zeitdauern im Sekundenbereich. Dementsprechend können durch ein Sliding-Window für WALK deutlich mehr Feature-Vektoren erzeugt werden als für JUMP. Das ist problematisch, da die eingesetzten Machine-Learning-Methoden besser funktionieren, wenn das Verhältnis der Klassen zumindest annähernd ausgeglichen ist.

Es gibt zwei grundlegende Techniken zum Ausgleich solcher Klassenimbalancen: Undersampling und Oversampling. Undersampling wird verwendet, um den Anteil überrepräsentierter Klassen zu vermindern, während Oversampling den Anteil unterrepräsentierter Klassen erhöht.

Undersampling ist ein sehr einfaches Verfahren, das einen einstellbaren Prozentsatz der Trainingsbeispiele entfernt. Die Auswahl geschieht zufällig, sodass im Mittel sowohl repräsentative als auch unwichtige Beispiele entfernt werden.

Oversampling fügt zusätzliche Trainingsbeispiele hinzu. In seiner einfachsten Form würden einige Trainingsbeispiele mehrfach verwendet werden. Im Projekt wird allerdings ein verbesserte Ansatz namens Synthetic Oversampling verwendet. Die Grundidee ist dabei, neue Trainingsbeispiele zu generieren, indem vorhandene Beispiele der Klasse kombiniert werden. Der eingesetzte Algorithmus nennt sich SMOTE (Synthetic Minority Over-sampling Technique) und ist in [CBHK02] beschrieben.

3.4.5. Dimensionsreduktion

In vielen Feldern der Informationsverarbeitung wie Mustererkennung, Datenkomprimierung, Datenbanknavigation und Maschinellem Lernen stellt sich das Problem großer Datenmengen, deren Umfang ein einfaches Verarbeiten zu zeitaufwändig oder gar unmöglich macht. Der Umfang der Daten ergibt sich dabei sowohl durch die hohe Anzahl an Datenpunkten als auch durch die hohe Anzahl an Merkmalen pro Datenpunkt [Gho06] [SAHB14]. Die Verfahren der Dimensionsreduktion zielen darauf ab, eine Darstellung der Daten zu finden, welche deutlich kleiner ist als die der Originaldaten, dabei aber die in den Daten enthaltene „Kerninformation“ zu erhalten. In einigen Fällen kann die resultierende Darstellung sogar zum Clustern und Klassifizieren von Daten genutzt werden. Im Rahmen der Projektgruppe sind solche Verfahren von Interesse, da sie dazu genutzt werden können, eine dimensionsreduzierte Repräsentation des Merkmalraums zu finden, welche anstelle der originalen Merkmale zum Trainieren der maschinellen Lernalgorithmen genutzt werden kann. Dies kann die Algorithmen auf Grund der kleineren Problemgröße schneller machen und kann zudem zu verbesserten Klassifikationsergebnissen führen.

Hauptkomponentenanalyse

Die Hauptkomponentenanalyse (engl. principal component analysis, PCA) ist ein Dimensionsreduktionsverfahren, welches darauf abzielt, eine Datenmenge anhand ihrer Hauptkomponenten (den Eigenvektoren ihrer Kovarianzmatrix) darzustellen und somit auf die wesentlichen Komponenten zu reduzieren. Zur Erinnerung: der Erwartungswert $E(X)$ über einer Zufallsvariable X mit Ausprägungen $X_1, X_2, \ldots, X_n$ ergibt sich durch Gleichung 3.13.

$$E(X) = \frac{\sum_{i=1}^{n} X_i}{n} \tag{3.13}$$

Die Varianz $Var(X)$ von X ist gegeben durch das Quadrat der Standardabweichung $\sqrt{Var(X)}$ also durch Gleichung 3.14.

$$Var(X) = \frac{\sum_{i=1}^{n} (X_i - E(X))^2}{n-1} \tag{3.14}$$

Während der Erwartungswert beschreibt, welchen Wert die Zufallsvariable im Mittel an-

nimmt, geben Standardabweichung und Varianz ein Maß der Streuung um dieses Mittel an. Diese Größen sind allerdings alle 1-dimensional. Für Datensätze höherer Dimension ließen sich diese Werte zwar vergleichen, allerdings geben sie keinen direkten Zusammenhang zwischen mehreren Merkmalen eines Datensatzes an, da sie unabhängig von einander sind. Eine Größe, welche einen Zusammenhang darstellt, ist die Kovarianz.

Die Kovarianz $Cov(X, Y)$ ergibt sich für zwei Zufallsvariablen X und Y durch Gleichung 3.15.

$$Cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - E(X))\dot{(}Y_i - E(Y))}{n-1} \tag{3.15}$$

Sie beschreibt eine Maß der Streuung zwischen den beiden Zufallsvariablen. Was auffällt ist, dass die Berechnung der Varianz (Gleichung 3.14) und die Berechnung der Kovarianz (Gleichung 3.15) sehr ähnlich sind. Tatsächlich gilt $Cov(X, X) = Var(X)$.

Um ein entsprechendes Maß für einen n–dimensionalen Merkmalsraum zu erhalten, lässt sich die Kovarianzmatrix $C_{m,m}$ durch Gleichung 3.16 berechnen, dabei sind $X_1, X_2, \ldots, X_n$ die n Merkmale des Merkmalsraums.

$$C_{m,m} = \begin{pmatrix} Cov(X_1, X_1) & Cov(X_1, X_2) & \cdots & Cov(X_1, X_n) \\ Cov(X_2, X_1) & Cov(X_2, X_2) & \cdots & Cov(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n, X_1) & Cov(X_n, X_1) & \cdots & Cov(X_n, X_n) \end{pmatrix} \tag{3.16}$$

Auf der Diagonalen der Matrix finden sich die Varianzen der einzelnen Merkmale, darüber bzw. darunter die entsprechenden Kovarianz für je zwei Merkmale.

Für die Hauptkomponentenanalyse werden nun die Eigenvektoren der Kovarianzmatrix bestimmt und nach Eigenwerten absteigend sortiert. Die Eigenvektoren mit niedrigem Eigenwert können entfallen, algorithmisch lässt sich dies z. B. dadurch umsetzen, dass nur Vektoren mit Eigenwert größer einem vorgegeben Grenzwert behalten werden.

Zur Erinnerung: Für eine Matrix A heißt jener Vektor $\vec{x}$ Eigenvektor bezüglich A, welcher bei Multiplikation mit der Matrix ein Vielfaches seiner selbst $\lambda \cdot \vec{x}$ ergibt. Für den also gilt:

$$A \cdot \vec{x} = \lambda \cdot \vec{x} \tag{3.17}$$

Der Wert λ heißt dann Eigenwert des Vektors $\vec{x}$ bezüglich Matrix A. Der Eigenwert λ ist unabhängig von der Skalierung des Vektors. In anderen Worten für $\vec{y} = \vec{x} \cdot s$ mit $s \in \mathbb{R}$ gilt $A \cdot \vec{y} = \lambda_y \cdot \vec{y}$ mit $\lambda_y = \lambda$.

Bestimmen lassen sich die Eigenvektoren und -werte z. B. per Gauß-Algorithmus oder Additionsverfahren. Für PCA werden Einheits-Eigenvektoren genutzt, also die Eigenvektoren der Länge 1. Unabhängige Eigenvektoren einer Matrix erfüllen die Eigenschaft, orthogonal zueinander zu stehen. Mithilfe der Eigenvektoren lassen sich weitere Merkmalsvektoren in den dimensionsreduzierten Merkmalsraum projizieren.

Im Folgenden soll die PCA anhand eines Beispiels mit 2-dimensionalen Merkmalen gezeigt werden. Dieses Beispiel ist [Smi02] entnommen.

Tabelle 4.: PCA: Daten des Beispiels

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

Damit PCA korrekt angewendet werden kann, müssen die Daten zu Beginn um den Mittelwert zentriert werden. Die in Tabelle 4 gezeigten Beispieldaten, ergeben die in Tabelle 5 zentrierten Werte durch Subtraktion der Mittelwerte $\bar{x} = 1.81$ und $\bar{y} = 1.91$.

Tabelle 5.: PCA: Mittelwert-zentrierte Daten

x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

Für die Mittelwert-zentrierten Daten ergibt sich die Kovarianzmatrix C als

$$C_{2,2} = \begin{pmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.716555556 \end{pmatrix} \tag{3.18}$$

mit Eigenvektoren $\vec{e}_1 = \begin{pmatrix} -0.73517656 \\ 0.677873399 \end{pmatrix}$ und $\vec{e}_2 = \begin{pmatrix} -0.677873399 \\ -0.735178656 \end{pmatrix}$ mit zugehörigen Eigenwerten $\lambda_{\vec{e}_1} = 0.0490833989$ und $\lambda_{\vec{e}_2} = 1.28402771$.

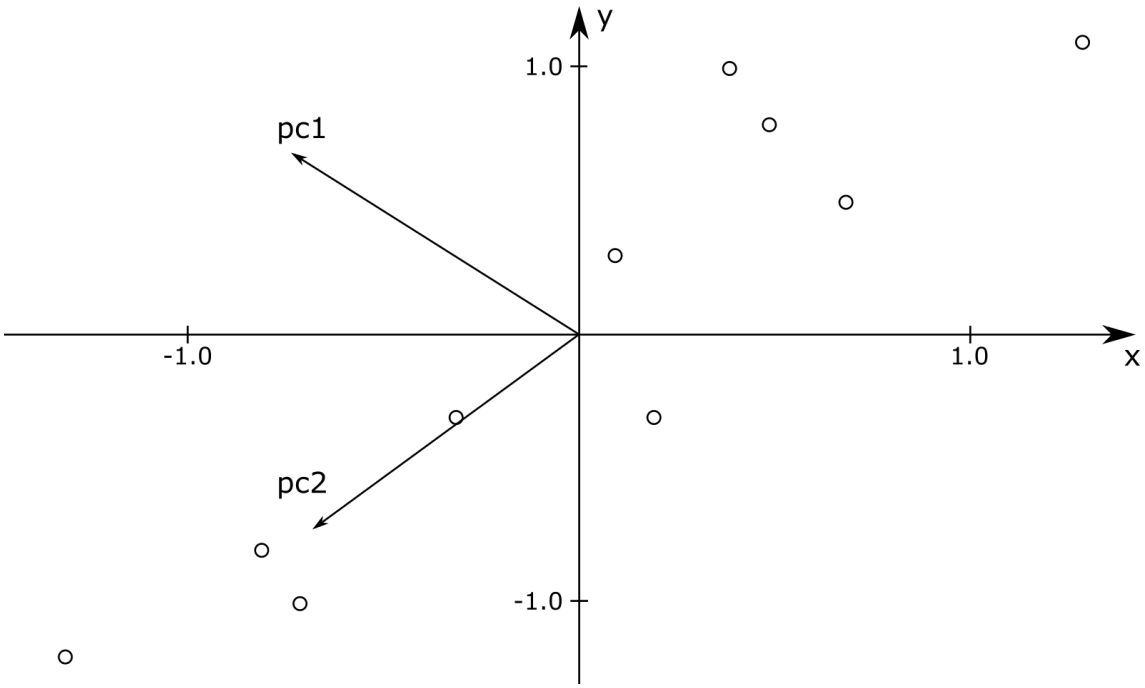


Abbildung 28.: Verteilung der Mittelwert-zentrierten Merkmale und die beiden Hauptkomponenten, **pc1** im Bild entspricht Eigenvektor $\vec{e}_1$ und **pc2** im Bild entspricht Eigenvektor $\vec{e}_2$.

Wie in Abbildung 28 zu sehen, unterliegen die Merkmale einem linearen Zusammenhang, welcher auch in der zugehörigen Kovarianzmatrix zu sehen ist. In der Abbildung sind zudem die beiden Hauptkomponenten $\vec{e}_1$ als **pc1** und $\vec{e}_2$ als **pc2** dargestellt. Die Hauptkomponente $\vec{e}_2$ geht in Richtung der linearen Regressionsgraden durch die Merkmalspunkte, während $\vec{e}_1$ in Richtung der seitlichen Abweichung von der Geraden verläuft. An den Eigenwerten ist zu erkennen, dass die Hauptkomponente $\vec{e}_2$ die Verteilung der Merkmalspunkte gut beschreibt und $\vec{e}_1$ dies deutlich schlechter tut.

Um nun weitere Merkmals-Vektoren in den reduzierten Merkmals-Raum zu projizieren, ist eine Matrix-Multiplikation durchzuführen. Die Matrix P sei die Projektionsmatrix, welche sich für die gewonnen, gewählten Hauptkomponenten $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ mit Eigenwerten

$\lambda_{\vec{e}_1} >= \lambda_{\vec{e}_2} >= \ldots >= \lambda_{\vec{e}_n}$ wie folgt ergibt:

$$P = \begin{pmatrix} (\vec{e}_1)^T \\ (\vec{e}_2)^T \\ \vdots \\ (\vec{e}_n)^T \end{pmatrix} \tag{3.19}$$

Die Matrix P bildet sich also aus den Transponierten der gewählten Hauptkomponenten als Zeilenvektoren. Der reduzierte Merkmal-Vektor $\vec{m}_p$ des originalen Merkmal-Vektors $\vec{m}$ mit Mittelwert-zentriertem Vektor $\vec{m}_c$ ergibt sich dann durch die Multiplikation:

$$\vec{m}_p = P \times \vec{m}_c \tag{3.20}$$

Für das gegebene Beispiel ist die Projektionsmatrix P, wenn beide Hauptkomponenten gewählt werden:

$$P = \begin{pmatrix} \vec{e}_2^T \\ \vec{e}_1^T \end{pmatrix} = \begin{pmatrix} -0.677873399 & -0.735178656 \\ -0.735178656 & 0.677873399 \end{pmatrix} \tag{3.21}$$

Für einen Vektor $\vec{m} = \begin{pmatrix} 2.5 \\ 2.4 \end{pmatrix}$ mit Mittelwert-zentriertem Vektor $\vec{m}_c = \begin{pmatrix} 0.69 \\ 0.49 \end{pmatrix}$ ergibt sich dann die Projektion $\vec{m}_p$ durch:

$$\vec{m}_p = P \times \vec{m}_c = \begin{pmatrix} -0.677873399 & -0.735178656 \\ -0.735178656 & 0.677873399 \end{pmatrix} \times \begin{pmatrix} 0.69 \\ 0.49 \end{pmatrix} = \begin{pmatrix} -8.2797 \\ -0.1751 \end{pmatrix} \tag{3.22}$$

Wählt man stattdessen P ohne die zweite, weniger relevante Hauptkomponente, so ergibt sich die Projektion als:

$$\vec{m}_p = P \times \vec{m}_c = \begin{pmatrix} -0.677873399 & -0.735178656 \end{pmatrix} \times \begin{pmatrix} 0.69 \\ 0.49 \end{pmatrix} = \begin{pmatrix} -8.2797 \end{pmatrix} \tag{3.23}$$

In diesem Fall wird dann also der 2-dimensionale Merkmals-Vektor $\vec{m} = \begin{pmatrix} 2.5 \\ 2.4 \end{pmatrix}$ auf eine 1-dimensionale Repräsentation $\vec{m}_p = \begin{pmatrix} -8.2797 \end{pmatrix}$ projiziert.

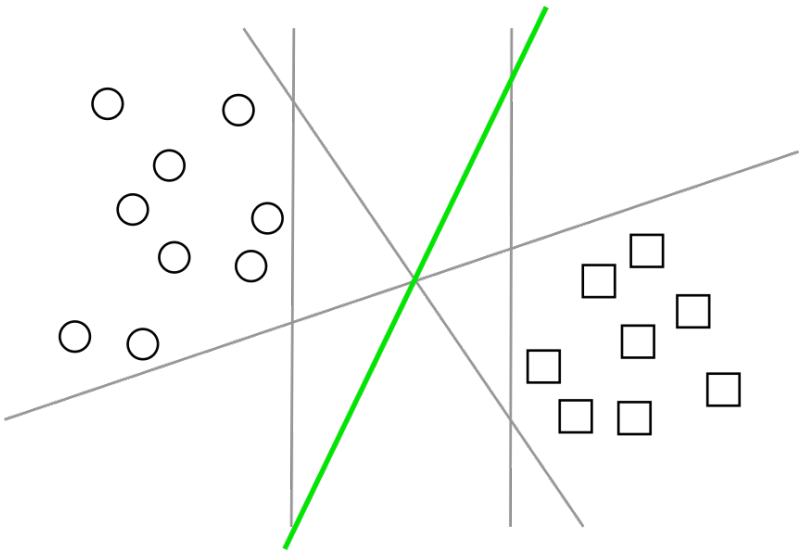


Abbildung 29.: Hyperebene in grün und weitere Klassifikatoren

3.4.6. Klassifikation und Erkennung der Aktivität

„Die Bewegungserkennung kann im Allgemeinen als Klassifikationsproblem aufgefasst werden“ [Nia12]. Aktivitäten oder Ereignisse sind dabei die zu unterscheidenden Klassen [Nia12]. Im folgenden Kapitel werden ausgewählte Methoden zur Klassifikation vorgestellt.

Support Vector Machine (SVM) Support Vector Machines (SVM) wurden im Kontext der Mustererkennung in einer Implementierung zur strukturellen Risikominimierung entwickelt. Eine erweiterte Form der SVM kann bei der Regressionsschätzung genutzt werden. Eine Anwendung können SVMs in linearen und nicht-linearen Systemen finden [DH]. Das Klassifizierungsproblem kann beschränkt werden auf die Berücksichtigung des Zwei-Klassen-Problems ohne an Allgemeingültigkeit zu verlieren. Als Ziel wird gesetzt, eine Klassifizierung zu finden, welche bei unbekannten Fällen funktioniert, also gut verallgemeinern kann. In Abbildung 29 sind mehrere Punkte im Graphen zu sehen, welche von verschiedenen Klassifikatoren separiert werden können, aber lediglich der grüne lineare Klassifikator schafft dies mit einer maximalen Spanne. Dies ist damit die optimale, separierende Hyperebene [Gun98].

Nächste-Nachbarn-Klassifikation Die „Nearest-Neighbour-Klassifikation“ lässt sich auf mehr als zwei Klassen anwenden. Zum klassifizierenden Merkmalsvektor wird die zugehörige Klasse genommen und als Klasse des nächsten Nachbarn festgelegt. In Bezug auf die „k-Nearest-Neighbour-Methode“ wird beim Auffinden einer passenden

Klasse so vorgegangen, dass die Klasse gewählt wird, bei der die meisten Mitglieder zu k nächsten Nachbarn zugeordnet werden können [Ert13].

Hidden-Markov-Modelle „Hidden Markov Modelle (HMM) können mit endlichen Automaten verglichen werden. Die Zustandsübergänge und die Ausgänge beruhen auf Wahrscheinlichkeitsbetrachtungen“ [Nia12]. Dieses Modell umfasst eine Markov-Kette, welche oft eine endliche Menge an Elementen enthält. Es ist jedoch auch möglich, einen beliebigen Zustandsraum vorauszusetzen. Dabei ist wichtig zu nennen, dass diese Kette versteckt ist und nicht beobachtet werden kann. Der Beobachter sieht lediglich den stochastischen Prozess, der die Verteilung regelt, durch die ein Element X der Markov-Kette einem Element Y zugeordnet wird [CMR05].

Naiver Bayes-Klassifikator Als „Naiver Bayes Klassifikator“ wird ein wahrscheinlichkeitsbasiertes Klassifikationsverfahren, das auf dem Satz von Bayes basiert, beschrieben. Die zu der „Klassifikation benötigten Wahrscheinlichkeiten lassen sich aus den [...] beobachteten relativen Häufigkeiten schätzen“ [Run10]. Sofern es sich um eine „deterministische[] Klassifikation“ handelt, liefert der naive Bayes-Klassifikator diejenige Klasse mit der höchsten Wahrscheinlichkeit. Einer der Vorteile des Klassifikators ist die Effizienz, weil die Trainingsdaten zur Bestimmung der relativen Häufigkeit nur ein einziges Mal durchlaufen werden müssen [Run10].

Entscheidungstabellen Unter dem Begriff „Entscheidungstabelle“ versteht man „eine übersichtliche Zusammenfassung von Entscheidungsregeln, die angeben, welche Bedingung oder Kombination von Bedingungen erfüllt sein muß [!sic] bzw. nicht erfüllt sein darf, wenn eine genau definierte Aktion oder Aktionsfolge ausgeführt werden soll“ [Sch13]. Mithilfe solcher Tabellen kann die Logik einer Problemlösung zweidimensional dargestellt werden und es ist auch möglich, mehrere, untereinander verknüpfte Entscheidungstabellen für komplexere Probleme zu nutzen [Sch13].

Entscheidungsbäume Mithilfe von Entscheidungsbäumen können „mehrstufige[] Entscheidungen“ dargestellt werden [MG]. Diese wurden aus einem Zustandsbaum erweitert, „indem [...] neben den erwarteten Umweltzuständen“ auch „die verfügbaren Handlungsalternativen“ betrachtet wurden [MG]. Der Entscheidungsbaum kann für eine komplette „Abbildung der Entscheidungssituation“ genutzt werden, was „jedoch nur bei sehr einfachen Entscheidungsproblemen praktikabel“ ist [MG].

Künstliche neuronale Netze Aufgrund von Problemen werden „effiziente Methoden zur

Verarbeitung und Repräsentation von Wissen gesucht“ [KBB+15]. Es gab „Problemstellungen-Verfahren“ , welche eine Orientierung „an natürlichen bzw. biologischen Prozessen“ gesucht haben [KBB+15]. Zu diesem Bereich gehören auch künstliche neuronale Netze.

Hybride Modelle Die vorangegangenen Methoden zu Entscheidungsbäumen und künstlichen neuronalen Netzen werden mit ihren positiven Eigenschaften in hybriden Modellen kombiniert. Dabei wird davon ausgegangen, dass Wissen ohne Erfahrung beim Entwurf von Entscheidungsbäumen die Klassifikationsgenauigkeit im Allgemeinen verbessern kann [Nia12].

3.4.7. Bewertung der Klassifikationsergebnisse

Die Algorithmen, welche im Rahmen des Projekts entwickelt werden, sollen ein Klassifikationsproblem lösen: Ausgehend von Sensormesswerten (Features) $X_i = (x_1, x_2, \ldots, x_n)_i$ für Zeitpunkte i sollen ausgeführte Bewegungen (Label) y_i durch einen Klassifikator $f(X_i) = \hat{y}_i$ erkannt werden. Um verschiedene Implementierungen des Klassifikators vergleichen zu können, müssen die Klassifikationsergebnisse durch ein geeignetes Verfahren bewertet werden.

Empirisches Risiko

Der erste Bewertungsansatz verwendet das Risiko, das heißt den erwarteten Fehler. Das tatsächliche Risiko einer Klassifikationsfunktion ist abhängig von der unbekannten Wahrscheinlichkeitsverteilung der Features, daher wird stattdessen das empirische Risiko aus Gleichung 3.24 als Näherung genutzt [Pó].

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f\left(X_i\right)\right) \tag{3.24}$$

Die Funktion L ist dabei eine Kostenfunktion. Für ein Klassifikationsproblem bietet sich dafür die *0-1-Loss-Function*

$$L(y_i, f(X_i)) = L(y_i, \hat{y}_i) = I(y_i = \hat{y}_i)$$

an. I ist dabei die Indikatorfunktion, welche 0 zurückliefert, wenn ihre Argumente gleich sind und 1 anderenfalls. Diese Funktion wurde gewählt, um bei der Bewertung nur zwischen

richtiger und falscher Vorhersage zu unterscheiden, denn es lässt sich auf den Bewegungs-Labels keine sinnvolle Distanzmetrik aufstellen.

Je kleiner das empirische Risiko eines Klassifikators ist, desto bessere Vorhersagen trifft er. Dieses Verfahren hat allerdings zwei Probleme:

1. Die Referenzlabel sind lückenhaft, das heißt es gibt nicht für jeden Zeitpunkt der Zeitreihe ein zugewiesenes Label. Dadurch würde an diesen Stellen die Indikatorfunktion eine 1 zurückliefern, sodass der empirische Fehler sehr groß werden würde.
2. Ein einzelner Wert als Vergleichskriterium ist eher ungenau.

Das erste Problem kann gelöst werden, indem nur die Zeitpunkte in die Risikoberechnung einfließen, für die es auch Referenzlabel gibt. Das zweite Problem ist jedoch mit dem verwendeten Ansatz nicht direkt lösbar. Wünschenswert wäre es, wenn die Bewertung des Klassifikators angeben kann, mit welcher Genauigkeit die einzelnen Bewegungen vorhergesagt werden können. Darüber hinaus ist auch interessant, in wie vielen Fällen eine der Bewegungen von dem Klassifikator als eine bestimmte andere Bewegung klassifiziert wird. Um diesen Anforderungen gerecht zu werden, wird im folgenden Kapitel ein weiterer Ansatz, die Konfusionsmatrix, erläutert.

Konfusionsmatrix

Eine Konfusionsmatrix spiegelt die Resultate eines Klassifikators wieder, der auf einem Datensatz angewendet wurde. Dabei sind die wahren Label dieses Datensatzes schon bekannt, sodass Vergleiche mit den Vorhersagen gemacht werden können. Ein Beispiel einer Konfusionsmatrix für ein binäres Klassifikationsproblem ist in Tabelle 6 angegeben. Sie ist immer eine quadratische Matrix und ihre Elemente zählen, wie oft der Klassifikator die verschiedenen Label in Bezug zu dem wahren Label vorhergesagt hat. Beispielsweise wurden von allen 65 Labeln mit dem echten Wert „Richtig“ 50 Stück vom Klassifikator auch als „Richtig“ erkannt, während 15 aber fälschlicherweise als „Falsch“ erkannt wurden. Im ersten Fall spricht man von *True positive* (t_p) und im zweiten Fall von *False negative* (f_n). Wird ein Label korrekt als „Falsch“ vorhergesagt, dann handelt es sich um einen *True negative* (t_n). Der verbleibende Fall wird *False positive* (f_p) genannt. [Pow11]

Ein Klassifikator ist umso besser, je mehr Label korrekt erkannt werden bzw. je weniger *False positives/negatives* auftreten. In der Matrix spiegelt sich das durch möglichst große

Elemente auf der Diagonalen und möglichst kleine restliche Elemente wieder. Um diese Zusammenhänge klarer zu definieren, können verschiedene Maße berechnet werden [Pow11] [MRS08]:

- Sensitivität/*Recall*: Beschreibt das Verhältnis der korrekt klassifizierten Objekte einer Klasse zu der Gesamtzahl der Eingabedaten für diese Klasse. Die Formel dazu lautet

$$Recall = \frac{t_p}{t_p + f_n}.$$

  An dem Beispiel in Tabelle 6 für die Klasse „Richtig“:

$$Recall(Richtig) = \frac{50}{50 + 15} \approx 76,9\%$$

- Genauigkeit/*Precision*: Beschreibt das Verhältnis der korrekt klassifizierten Objekte zu allen Vorhersagen mit diesem Label. Die Formel dazu lautet

$$Precision = \frac{t_p}{t_p + f_p}.$$

  An dem Beispiel in Tabelle 6 für die Klasse „Richtig“:

$$Precision(Richtig) = \frac{50}{50 + 5} \approx 90,9\%$$

  .

- Korrektklassifikationsrate/*Accuracy*: Beschreibt das Verhältnis aller korrekt klassifizierten Objekte zu dem gesamten Datensatz. Die Formel dazu lautet

$$Accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}.$$

  An dem Beispiel in Tabelle 6:

$$Accuracy = \frac{50 + 95}{50 + 95 + 5 + 15} \approx 87,9\%$$

  .

- F-Maß/*F-score*: Das F-Maß ist eine Kombination aus Sensitivität und Genauigkeit mit

Hilfe des gewichteten harmonischen Mittels. Die Formel dazu lautet

$$F_\alpha = (1 + \alpha^2) \cdot \frac{(precision \cdot recall)}{(\alpha^2 \cdot precision + recall)}.$$

Mit dem Parameter α kann die Gewichtung von Sensitivität und Genauigkeit eingestellt werden. Mit $\alpha > 1$ wird die Sensitivität stärker gewichtet als die Genauigkeit und mit $\alpha < 1$ ist es genau umgekehrt. Mit $\alpha = 1$ werden beide Maße gleich stark gewichtet.

Sensitivität und Genauigkeit sind eigentlich nur für die Positivklasse definiert. Bei der Negativklasse spricht man von Spezifität beziehungsweise Segreganz. Diese unterschiedlichen Bezeichnungen sind für Konfusionsmatrizen mit mehr als zwei Klassen nicht sinnvoll. Daher wird im Folgenden immer nur von Sensitivität respektive Genauigkeit gesprochen.

Realität \ Vorhersage	Richtig	Falsch	Sensitivität %
Richtig	50	15	76,9
Falsch	5	95	95,0
Genauigkeit %	90,9	86,3	

Tabelle 6.: Konfusionsmatrix eines binären Klassifikationsproblems

In der zu entwickelnden Softwarelösung zur Erkennung von körperlichen Bewegungen handelt es sich um ein Mehrklassenproblem. Das heißt, dass das vorgestellte Verfahren noch nicht direkt angewendet werden kann. Es kann aber leicht erweitert werden, um diese Anforderungen zu erfüllen. Ein entsprechendes Beispiel ist in Tabelle 7 zu sehen. Die Definition der *True/False positives/negatives* in dieser Mehrklassen-Konfusionsmatrix fällt jedoch etwas anders aus:

- *True positive*: Die *True positives* der verschiedenen Klassen befinden sich auf der Diagonalen der Konfusionsmatrix.
- *False positive*: Die Anzahl der *False positives* einer Klasse ist die Summe der zugehörigen Spalte minus dem *True positve*-Wert.
- *False negative*: Die Anzahl der *False negatives* einer Klasse ist die Summe der zugehörigen Reihe minus dem *True positve*-Wert.
- *True negative*: Die Anzahl der *True negatives* einer Klasse ist die Summe der gesamten Matrix minus der Summe der zugehörigen Reihe und Spalte.

Die vorgestellten Maße können auf demselben Weg berechnet werden.

Insgesamt entsteht so ein Bewertungsansatz, mit dem Klassifikatoren leicht verglichen werden können. Darüber hinaus ist es auch möglich die Schwächen eines Klassifikators zu erkennen, sodass entsprechende Maßnahmen zur Verbesserung der Ergebnisse vorgenommen werden können.

Realität \ Vorhersage	Klasse1	Klasse2	Klasse3	Klasse4	Sensitivität %
Klasse1	40	3	5	2	80,0
Klasse2	5	35	10	1	68,6
Klasse3	0	0	50	0	100,0
Klasse4	5	7	1	35	72,9
Genauigkeit %	80,0	77,8	75,8	92,1	

Tabelle 7.: Konfusionsmatrix eines Mehrklassen-Klassifikationsproblems

3.5. Schritterkennung und abgeleitete Gangparameter

Nachdem eine Aktivitätserkennung durchgeführt wurde, können die erkannten Aktivitäten für weitere Analysen verwendet werden. Insbesondere ist im Kontext der Projektgruppe die Analyse der Gehen-Aktivität von Interesse, da der Gang Teil von vielen Assessment-Übungen ist und anhand einer Ganganalyse Aussagen über das Gangbild eines Probanden gemacht werden können. Damit eine Ganganalyse durchgeführt werden kann, müssen zunächst einzelne Schritte einer erkannten Gehen-Aktivität bestimmt werden.

Der Algorithmus zur Erkennung von Schritten innerhalb erkannter Gehen-Aktivitäten beruht auf einem Peak-Detection-Verfahren, das die Zeitpunkte des Auftretens der Ferse am Boden detektiert. Eine detaillierte Beschreibung des Algorithmus ist in Unterabschnitt D.6 zu finden. Aus den erkannten Schritten können anschließend einige Gangparameter direkt abgeleitet werden. Zu diesen Gangparametern gehören u.a.:

- Die Anzahl der Schritte
- Die minimale, maximale und durchschnittliche Schrittdauer
- Die minimale, maximale und durchschnittliche Doppelschrittdauer
- Die Kadenz (Schrittfrequenz)

Weitere Informationen zu diesen und weiteren Gangparametern, die während einer Ganganalyse betrachtet werden können, werden in Unterabschnitt D.6 sowie auch in Unterab-

schnitt D.7 bereitgestellt. Die Realisierung der Schritterkennung und die Generierung der direkt ableitbaren Gangparameter ist in Unterabschnitt 6.3.9 beschrieben.

3.6. High-End Computing Resource Oldenburg (HERO)

HERO ist ein Rechencluster der Universität Oldenburg, das Angehörigen der Universität das Ausführen rechenintensiver Programme erlaubt.

Im Rahmen des Projekts muss eine große Datenmenge durch verschiedene Algorithmen analysiert werden. Um diesen Prozess zu beschleunigen, sollen die Algorithmen nicht nur auf einzelnen Hosts, sondern auch auf dem HERO-Cluster ausgeführt werden können. Da sich das parallele Arbeiten auf einem einzelnen Host deutlich vom parallelen Arbeiten auf HERO bzw. Rechenclustern im Allgemeinen unterscheidet, muss die Softwarearchitektur des Projekts dieses berücksichtigen. Je besser die Softwarearchitektur zur Hardwarearchitektur des Clusters passt, desto höhere Performanz kann erzielt werden.

Die Haupt-Rechenressource von HERO besteht aus einem Verbund von 150 Compute Nodes, d.h. 150 physikalischen Servern. Diese verfügen zusammen über 1800 CPU-Kerne und 4,1 TB Arbeitsspeicher. Jeder einzelne physikalische Server besitzt 2 CPUs mit jeweils 6 Kernen. Die einzelnen Compute Nodes sind durch ein 10 Gb Ethernet-Netzwerk verbunden, das ausschließlich für die Kommunikation zwischen den CPUs reserviert ist. Der Zugriff auf Massenspeicher und andere Systeme des Clusters geschieht über ein weiteres separates Netzwerk. Die genauen Spezifikationen lassen sich im Web[1] einsehen.

Prozesse, die auf dem Cluster laufen, werden durch das Sun Grid Engine (SGE) Job Management System verwaltet. Die SGE teilt das Cluster in sogenannte Slots auf, wobei ein Slot üblicherweise einem CPU-Kern entspricht. Der SGE können Jobs übergeben werden, welche die Ausführung eines Programms, benötigte Ressourcen u. v. m. beschreiben. Sofern das Cluster über freie Slots verfügt, werden übergebene Jobs aus einer Warteschlange entnommen und auf freie Slots zur Ausführung verteilt. Offensichtlich verfügt ein mit SGE ausgeführter Job, da er in einem Slot ausgeführt wird, nur über einen CPU-Kern.

Um dennoch parallele Jobs mit SGE auszuführen, kann ein Parallel Environment (PE) spezifiziert werden. Es gibt verschiedene PEs, von denen die für unser Projekt interessanten

[1] http://www.uni-oldenburg.de/fk5/wr/hochleistungsrechnen/hpc-facilities/hero/

Umgebungen im Folgenden beschrieben werden.

Shared memory parallelization (smp) Eine Möglichkeit zur parallelen Programmierung ist es, mehrere Threads eines einzelnen Prozesses auf gemeinsamen Speicher arbeiten zu lassen. Dieses Parallelisierungsmodell wird durch die smp-Umgebung unterstützt. Ein Job mit smp-Umgebung wird von SGE daher nur auf einer Compute Node ausgeführt, aber kann auf dieser beispielsweise alle 12 Kerne zugeordnet bekommen.

Es ist zu beachten, dass Jobs mit Anfragen nach 12 Kernen im smp-Modus üblicherweise lange Wartezeiten haben, da ein vollständig ungenutzter physikalischer Server für den Job benötigt wird.

Message Passing Ein anderer Ansatz zur parallelen Programmierung ist es, separate Prozesse auszuführen, die miteinander über ein Netzwerk kommunizieren. Die Prozesse können auf verschiedenen Kernen derselben CPU laufen, auf verschiedenen Kernen verschiedener CPUs im selben Server oder auch auf gänzlich verschiedenen Servern. Über die tatsächliche Verteilung entscheidet letztendlich die SGE.

Die Kommunikation geschieht über das sogenannte Message Passing Interface (MPI). Dieses bietet eine API, über die Prozesse Daten austauschen können – unabhängig davon, wo die verschiedenen Prozesse ausgeführt werden. Allerdings müssen alle Prozesse über ein besonderes MPI Startup Tool gestartet werden, so dass MPI beispielsweise nicht zur Kommunikation zu einem Frontend-Prozess außerhalb des Clusters dienen kann.

Beide Parallel Environments lassen sich mit Java ausnutzen. Für smp reicht es aus, gewöhnliche Java-Threads zu starten. Diese werden ganz gewöhnlich auf den verschiedenen zugewiesenen Kernen ausgeführt. Soll Message Passing genutzt werden, ist es notwendig, eine Implementierung des MPI als Bibliothek einzubinden. OpenMPI bietet Java Bindings an, die für diesen Zweck genutzt werden können.

Eine weitere für unser Projekt kritische Ressource ist der Festplattenspeicher, da die zu verarbeitenden Datenmengen so groß sind. Zwar bietet HERO genügend Speicherkapazität an, jedoch kann auf diesen nur über das Netzwerk zugegriffen werden. Allerdings besitzt jede Compute Node auch etwas eigenen Festplattenspeicher, auf den die Arbeitsdaten vorher übertragen werden können, um schnellere Zugriffszeiten zu ermöglichen.

Es muss erwähnt werden, dass die erstellte Applikation neben den Rechnercluster HERO auch die beiden anderen Rechencluster der Universität, nämlich CARL und EDDY,

verwendet. Es wurde zunächst nur Quellcode für den Zugriff und die Arbeit auf HERO erstellt. Aber als zum Ende der Projektarbeit die Absicht der Universität bestand HERO abzuschalten und stattdessen CARL und EDDY zukünftig für die Nutzung zur Verfügung zu stellen, wurde der Quellcode der Anwendung auf die beiden letztgenannten Rechencluster erweitert.

4. Anforderungsanalyse

Nachfolgend werden die Anwendungsfälle und die Anforderungen der zu entwickelnden Software definiert. Zudem wird anhand des Medizinproduktgesetzes argumentiert, warum die Software als medizinisches Produkt einstufbar ist.

4.1. Anwendungsfälle

Um Anforderungen für die zu entwickelnde Anwendung spezifizieren zu können, werden zunächst die möglichen Anwendungsfälle der Anwendung beschrieben. Abbildung 30 stellt eine Übersicht der Anwendungsfälle der Anwendung und die Akteure mit ihren Beziehungen und Abhängigkeiten zu den Anwendungsfällen dar. Die einzelnen Anwendungsfälle der Anwendung werden anschließend detailliert in tabellarischer Form ausgeführt.

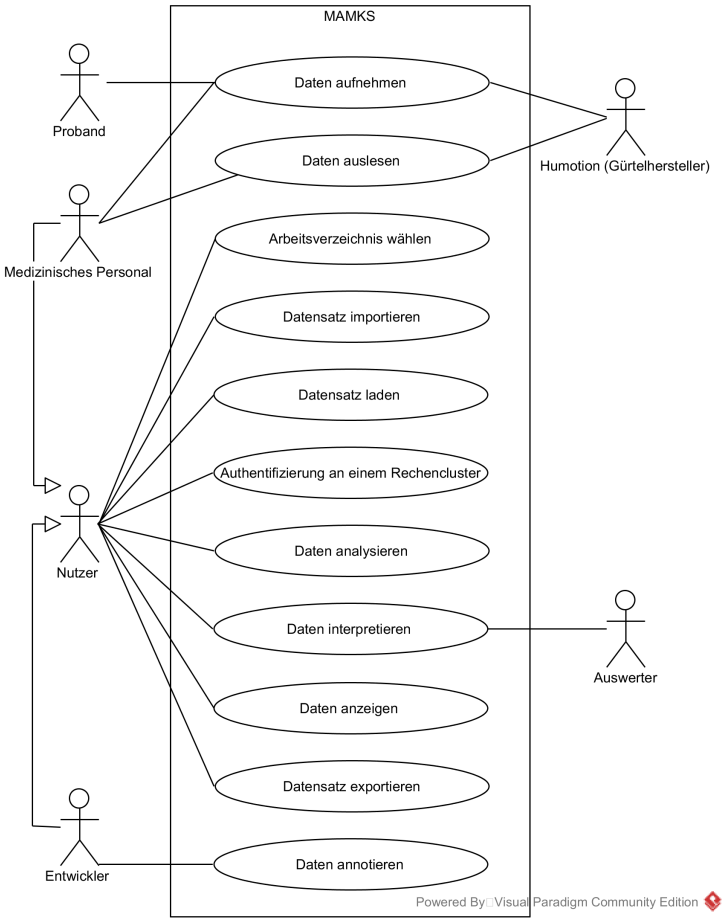


Abbildung 30.: Use-Case-Diagramm

Name	**Daten aufnehmen**
ID	UC-01
Beschreibung	Mithilfe des Sensorgürtels des Herstellers Humotion werden die Daten der Probanden aufgenommen. Dies beinhaltet das Durchführen der Assessments unter Anleitung von medizinischem Personal sowie das Tragen des Gürtels im Alltag.
Akteure	Proband, medizinisches Personal, Gürtelhersteller Humotion
Anfangsbedingung	Es liegen zwei aufnahmebereite Sensorgürtel vor.
Abschlussbedingung	Es liegt ein Sensorgürtel mit den Assessment-Daten und ein Sensorgürtel mit den Heim-Daten vor.
Standardablauf	1. Der Proband befindet sich im Assessment-Raum und legt den Gürtel an. 2. Das medizinische Personal (Physiotherapeut) führt nach einer Instruktion zu dem jeweiligen Test das Assessment mit dem Probanden durch und protokolliert dies (siehe Assessments). 3. Der Proband gibt den Sensorgürtel zurück und erhält im Austausch einen Neuen. 4. Der Proband verlässt den Assessment-Raum und nimmt den Gürtel mit nach Hause. 5. Der Proband trägt den Gürtel täglich und lädt jede Nacht den Akku auf. 6. Der Proband kehrt nach einer Woche in den Assessment-Raum zurück und gibt den Gürtel wieder ab.
Qualitätsanforderungen	
Fehlerfälle	• Die Hardware wird während der Benutzung beschädigt. • Es werden fehlerhafte Daten aufgenommen, z. B. durch Fehler in der Soft- oder Hardware. • Der Proband vergisst den Akku aufzuladen. • Der Proband vergisst den Gürtel anzulegen.

Name	**Daten auslesen**
ID	UC-02
Beschreibung	Die aufgenommenen Daten (Rohdaten und Metadaten) werden aus dem Sensorgürtel ausgelesen.
Akteure	medizinisches Personal, Gürtelhersteller Humotion
Anfangsbedingung	Es liegt ein Sensorgürtel mit aufgenommenen Daten vor.
Abschlussbedingung	Die Rohdaten des Sensorgürtels und die Metadaten liegen in Textdateien vor und stehen dem System zur Verfügung.
Standardablauf	1. Das medizinische Personal stellt mittels USB eine Verbindung zwischen dem Sensorgürtel und dem Rechner her und startet die Humotion-Software. 2. Die Humotion-Software liest die Rohdaten und Metadaten des Sensorgürtels aus und speichert sie in Textdateien ab. 3. Das medizinische Personal stellt dem System die Textdateien zur Verfügung.
Qualitätsanforderungen	
Fehlerfälle	

Name	**Arbeitsverzeichnis wählen**
ID	UC-03
Beschreibung	Es wird ein Arbeitsverzeichnis festgelegt, in dem die importierten Datensätze, berechneten Datenreihen und Label-Informationen gespeichert werden.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Der Nutzer hat Zugriff auf die Anwendung.
Abschlussbedingung	Der gewählte Zielpfad ist im System als Arbeitsverzeichnis abgespeichert.
Standardablauf	1. Der Nutzer startet die Anwendung. 2. Das System öffnet die Anwendung und einen Dialog zur Auswahl des Arbeitsverzeichnisses. 3. Der Nutzer klickt auf den Button zur Auswahl des Verzeichnisses. 4. Das System öffnet einen Datei-Explorer. 5. Der Nutzer navigiert zu dem gewünschten Verzeichnis und bestätigt seine Auswahl. 6. Das System schließt den Datei-Explorer und trägt den Pfad in der Dialogbox in ein Textfeld ein. 7. Der Nutzer wählt, falls gewünscht, die Option aus, dass der Dialog bei erneutem Öffnen der Anwendung nicht mehr angezeigt wird, und klickt auf „OK“. 8. Das System speichert den Pfad sowie die gewählte Anzeige-Option und schließt den Dialog.
Qualitätsanforderungen	• Es muss möglich sein, das Arbeitsverzeichnis nachträglich in der Anwendung zu ändern.
Fehlerfälle	

Name	**Datensatz importieren**
ID	UC-04
Beschreibung	Die Originaldaten werden in das System-Format überführt und im Arbeitsverzeichnis abgelegt.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Es liegen Originaldaten im Textdatei-Format (CSV und INI) vor.
Abschlussbedingung	Eine Repräsentation der zu importierenden Originaldaten befindet sich im Arbeitsverzeichnis.
Standardablauf	1. Der Nutzer wählt in der graphische Nutzerschnittstelle (GUI) unter „Datei“ den Menüpunkt „Datensatz importieren“ oder benutzt den entsprechenden Toolbar-Button oder benutzt das Tastenkürzel „Strg+I„. 2. Das System öffnet einen Dialog. 3. Der Nutzer klickt auf den Button zur Auswahl des Originaldatensatzes. 4. Das System öffnet einen Datei-Explorer. 5. Der Nutzer navigiert zum Ordner, in der sich der gewünschte Originaldatensatz befindet, wählt ihn aus und bestätigt mit einem Klick auf „Ordner auswählen“. 6. Das System schließt den Datei-Explorer und trägt den Pfad im Dialog in einem Textfeld ein. 7. Der Nutzer wählt aus, ob die Studienraumaufnahme oder die Daheimaufnahme importiert werden soll und ob ein Aufnahmesegment automatisch ausgewählt werden soll und klickt auf „OK“. 8. Wenn der gewählte Originaldatensatz mehr als ein zeitlich zusammenhängendes Aufnahmesegment enthält und der Nutzer keine automatische Segment-Auswahl gewählt hat, wird dem Nutzer eine Tabelle mit allen Aufnahmesegmenten, ihren Start-, Endzeitpunkten und Aufnahmedauern präsentiert. Aus diesen kann der Benutzer genau ein Segment zum Import auswählen. 9. Das System liest die Originaldaten und speichert sie im System-Format im Arbeitsverzeichnis. Anschließend erhält der Nutzer eine Benachrichtigung über den erfolgreichen Import.
Qualitätsanforderungen	

Fehlerfälle	• Die Originaldaten liegen in einem unbekannten und daher nicht einlesbaren Format vor. • Es steht nicht genügend Speicherplatz zur Verfügung, um die Daten im System-Format abzuspeichern.

Name	**Datensatz exportieren**
ID	UC-05
Beschreibung	Datenreihen und Labelinformationen, die im System-Format vorliegen, werden ins CSV-Format exportiert.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Es wurde ein Datensatz in die Anwendung geladen.
Abschlussbedingung	Die ausgewählten Datenreihen und Labelinformationen liegen im CSV-Format in dem gewünschten Zielordner bereit.
Standardablauf	1. Der Nutzer wählt in der GUI unter „Datei“ den Menüpunkt „Datennsatz exportieren“ oder benutzt den entsprechenden Toolbar-Button oder benutzt das Tastenkürzel „Strg+E„. 2. Das System öffnet ein Dialog mit einer Liste aller Datenreihen, die im System-Format zu dem geladenen Datensatz vorliegen, und einer Checkbox zur Auswahl, ob Labelinformationen ebenfalls exportiert werden sollen. 3. Der Nutzer wählt die gewünschten Datenreihen, und ob Labelinformationen exportiert werden sollen, aus und drückt den Button zur Angabe des Zielpfades. 4. Das System öffnet einen Datei-Explorer. 5. Der Nutzer navigiert zu dem Ordner, in dem die CSV-Datei gespeichert werden soll, gibt einen Namen für diese Datei an und bestätigt seine Auswahl. 6. Das System schließt den Datei-Explorer und trägt im Dialog in einem Textfeld den Zielpfad ein. 7. Der Nutzer bestätigt mit „OK“. 8. Das System schließt den Export-Dialog, konvertiert die ausgewählten Datenreihen vom System-Format ins CSV-Format und speichert diese in der angegebenen Datei ab. Wurde ausgewählt, dass Labelinformationen ebenfalls exportiert werden sollen, konvertiert das System diese ebenfalls und speichert diese in einer separaten CSV-Datei. In der Statusleiste wird der Nutzer über den erfolgreichen Export informiert.
Qualitätsanforderungen	
Fehlerfälle	• Es ist nicht genug Speicherplatz vorhanden, um die CSV-Datei(en) abzuspeichern.

Name	**Datensatz laden**
ID	UC-06
Beschreibung	Der Nutzer wählt über die GUI den Datensatz im System-Format aus, der zum Betrachten und Analysieren in die Anwendung geladen werden soll.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Es befinden sich bereits importierte Datensätze im Arbeitsverzeichnis.
Abschlussbedingung	Der ausgewählte Datensatz wurde in die Anwendung geladen und die Probanden-ID und das Aufnahmedatum werden in der GUI angezeigt.
Standardablauf	1. Der Nutzer wählt in der GUI unter „Datei“ den Menüpunkt „Datensatz laden“ oder benutzt den entsprechenden Toolbar-Button oder benutzt das Tastenkürzel „Strg+O„. 2. Das System öffnet eine Dialogbox mit jeweils einer Dropdown-Liste für die Probanden-IDs und die dazugehörigen Datensätze (in Form von Datums- und Zeitangabe), die im System-Format im Arbeitsverzeichnis vorliegen. 3. Der Nutzer wählt eine Probanden-ID aus. 4. Das System stellt in der entsprechenden Dropdown-Liste alle zugehörigen Datensätze der gewählten Probanden-ID zur Verfügung. 5. Der Nutzer wählt einen Datensatz und bestätigt mit „OK“. 6. Das System lädt den Datensatz in die Anwendung und macht die Probanden-ID, das Aufnahmedatum und die Datenreihen des Accelerometers, Gyroskops und Magnetometers des Sensorgürtels in der GUI für den Nutzer sichtbar.
Qualitätsanforderungen	• Die Dropdown-Liste des Datensatzes darf erst auswählbar sein, nachdem die Probanden-ID gewählt wurde. • Der „OK“-Button darf seine Funktionalität erst erhalten, nachdem eine Probanden-ID und ein Datensatz gewählt wurde.
Fehlerfälle	

Name	**Authentifizierung an einem Rechencluster**
ID	UC-07
Beschreibung	Der Nutzer muss sich für die Nutzung des Rechenclusters authentifizieren.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Die Anwendung wurde gestartet.
Abschlussbedingung	Der Nutzer hat sich erfolgreich beim Rechencluster authentifiziert und kann das Rechencluster nun nutzen, um Algorithmen auf diesem ausführen.
Standardablauf	1. Der Nutzer klickt auf den entsprechenden Toolbar-Button oder gibt im Dialogfeld zur Ausführung eines Algorithmus an, dass dieser auf einem Rechencluster ausgeführt werden soll. 2. Das System öffnet eine Dialogbox mit einem Textfeld zur Angabe eines Nutzernamens und eines Passwortes. 3. Der Nutzer gibt sein Nutzername und sein Passwort in die beiden Textfelder ein und bestätigt mit „OK“. 4. Das System verwendet den eingegebenen Nutzernamen und das eingegebene Passwort um sich beim Rechencluster zu authentifizieren und gibt das Ergebnis der Authentifizierung (erfolgreich/nicht erfolgreich) aus.
Qualitätsanforderungen	• Das Textfeld zur Angabe des Passwortes muss maskiert angezeigt werden. • Das System muss anzeigen, ob die Authentifizierung erfolgreich war.
Fehlerfälle	• Der Nutzername oder das Passwort ist fehlerhaft. • Es kann keine Internetverbindung hergestellt werden.

Name	**Daten analysieren**
ID	UC-08
Beschreibung	Der geladene Datensatz wird mithilfe von Algorithmen analysiert. Hierbei soll die Erkennung von Assessment-Bewegungsmustern und alltäglichen Bewegungsmustern sowie die Berechnung von Bewegungsparametern und Erzeugung von Reports möglich sein.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Es wurde ein Datensatz in die Anwendung geladen.
Abschlussbedingung	Die Ergebnisse der Algorithmusausführungen befinden sich im System-Format in einer neuen Datei, die Parameterwerte sind in der Log-Datei eingetragen und, falls zuvor ausgewählt, werden die resultierenden Datenreihen in neuen Plots dargestellt.
Standardablauf	1. Der Nutzer wählt in der GUI unter „Bearbeiten“ den Menüpunkt „Algorithmus anwenden“ oder benutzt den entsprechenden Toolbar-Button hierfür. 2. Das System öffnet eine Dialogbox mit einer Liste aller im System integrierten Algorithmen. 3. Der Nutzer wählt einen Algorithmus aus und klickt auf „Weiter“. 4. Das System zeigt in der Dialogbox die Datensätze im Arbeitsverzeichnis an. 5. Der Nutzer wählt die Datensätze aus, für die der Algorithmus ausgeführt werden soll, und klickt auf „Weiter“. 6. Falls der ausgewählte Algorithmus ein ML-Algorithmus ist, zeigt das System wieder eine Liste der Datensätze im Arbeitsverzeichnis an. 7. Falls der ausgewählte Algorithmus ein ML-Algorithmus ist, wählt der Nutzer die Datensätze aus, die bei der Validierung der Klassifikation nicht betrachtet werden sollen, und klickt auf „Weiter“. 8. Das System prüft, welche Eingabeparameter für den gewählten Algorithmus benötigt werden, und stellt diese in der Dialogbox in einer Tabelle dar. 9. Der Nutzer wählt pro Eingabeparameter mindestens einen Wert aus und wählt zusätzlich aus, ob das Job-Management nach dem Start der Algorithmusausführung geöffnet werden soll, und bestätigt anschließend mit „OK“. 10. Das System fügt für jede Parameterkombination einer Algorithmusausführung dem Job-Management einen Job hinzu und trägt die Startzeit ein. Wurde die entsprechende Checkbox ausgewählt, wird das Job-Management geöffnet. 11. Das System schließt die Dialogbox und führt die Jobs nacheinander aus und trägt die Parameterwerte in eine Log-Datei ein. 12. Nach erfolgreicher Ausführung werden die Ergebnisse in einer neu angelegten Datei im Systemformat gespeichert.

Qualitätsanforderungen	• Bei fehlerhaften Eingaben von Parameterwerten muss auf den letzten gültigen Wert zurückgesprungen werden. • Die Parameterwerte aller ausgeführten Algorithmen müssen in einer Log-Datei einsehbar sein. • Die Jobs müssen über das Job-Management abgebrochen werden können. • Der Status und die Start- und Endzeit aller Jobs muss im Job-Management einsehbar sein. • Wählt der Nutzer die resultierenden Datenreihen vor dem Beenden der Anwendung nicht zum Speichern aus, müssen die entsprechenden Dateien wieder gelöscht werden.
Fehlerfälle	• Die Daten sind fehlerhaft. • Es ist nicht genug Speicherplatz vorhanden, um die Ergebnisse zu speichern.

Name	**Datenreihen grafisch anzeigen**
ID	UC-09
Beschreibung	Ausgewählte Datenreihen des geladenen Datensatzes der gleichen Einheit werden in einem Plot visualisiert.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Es wurde ein Datensatz in die Anwendung geladen.
Abschlussbedingung	Die ausgewählten Datenreihen werden in einem Plot dargestellt.
Standardablauf	1. Der Nutzer wählt in der GUI unter „Ansicht“ den Menüpunkt „Plot hinzufügen“ oder benutzt den entsprechenden Toolbar-Button hierfür. 2. Das System öffnet eine Dialogbox zum Auswählen der Einheit des zu erstellenden Plots per Dropdown-Liste und eine Checkbox-Liste der Datenreihen, die die ausgewählte Einheit in der Dropdown-Liste besitzt. 3. Der Nutzer wählt die Einheit des Plots per Dropdown-Liste aus und die Datenreihen in der Checkbox-Liste, die visualisiert werden sollen, und bestätigt mit „OK“. 4. Das System liest die Datenreihen aus dem System-Format aus und fügt der GUI einen Plot mit der ausgewählten Einheit hinzu, in dem die ausgewählten Datenreihen dargestellt werden.
Qualitätsanforderungen	• Es muss möglich sein, die Zeitachse durch Angabe einer Zeiteinheit und eines zugehörigen Wertes zu skalieren. • Es muss möglich sein, die Zeitachse durch Angabe eines Betrachtungsintervalls zu skalieren und zu verschieben. • Es muss möglich sein, die Einheitenachse durch Angabe eines minimalen und maximalen Wertes zu skalieren. • Es muss möglich sein, zu angezeigten Plots weitere Datenreihen der gleichen Einheit hinzuzufügen. • Es muss möglich sein, bereits visualisierte Datenreihen und angezeigte Plots zu entfernen.
Fehlerfälle	

Name	**Datenreihen tabellarisch anzeigen**
ID	UC-10
Beschreibung	Es wird eine Tabelle angezeigt, die die Werte der Datenreihen des geladenen Datensatzes und aller bisher berechneten Datenreihen anzeigt.
Akteure	medizinisches Personal, Entwickler
Anfangsbedingung	Es wurde ein Datensatz in die Anwendung geladen.
Abschlussbedingung	Die Werte aller Datenreihen des geladenen Datensatzes und berechnete Datenreihen werden in einer Tabelle dargestellt.
Standardablauf	1. Der Nutzer wählt in der GUI unter „Ansicht“ den Menüpunkt „Datenreihen-Tabelle anzeigen“ oder benutzt den entsprechenden Toolbar-Button hierfür. 2. Das System liest die Datenreihen aus dem System-Format aus und öffnet eine Tabellenansicht, in der die Werte aller Datenreihen mit zugehörigem Zeitpunkt enthalten sind.
Qualitätsanforderungen	• Die Auswahl an angezeigten Datenreihen muss in der Tabellenansicht angepasst werden können. • Die Spalten der Tabelle müssen verschiebbar sein, damit Datenreihen besser miteinander verglichen werden können.
Fehlerfälle	

Name	**Daten annotieren**
ID	UC-11
Beschreibung	In dem geladenen Datensatz können Zeitintervalle mit Bewegungsmuster- und Assessment-Test-Labeln versehen werden.
Akteure	Entwickler
Anfangsbedingung	Es wurde ein Datensatz in die Anwendung geladen.
Abschlussbedingung	Ein Label wurde einem Zeitintervall zu einer Labelgruppe hinzugefügt und als Labelinformation im Arbeitsverzeichnis gespeichert.
Standardablauf	1. Der Nutzer markiert mit der linken Maustaste innerhalb der Plots, der Zeitleiste oder der Labelgruppen ein Zeitintervall. 2. Das System merkt sich das ausgewählte Zeitintervall und zeigt dieses innerhalb der Plots, der Zeitleiste und der Labelgruppen an. 3. Der Nutzer klickt anschließend mit der rechten Maustaste in den markierten Bereich. 4. Das System öffnet ein Kontextmenü. 5. Der Nutzer wählt im Kontextmenü den Eintrag „Label hinzufügen“. 6. Das System öffnet eine Dialogbox mit Start- und Endzeit des markierten Zeitintervalls, eine Dropdown-Liste mit allen Bewegungsmustern und Assessment-Tests und ein Textfeld zur Angabe der Labelgruppe. 7. Der Nutzer wählt ein Bewegungsmuster oder einen Assessment-Test aus, passt gegebenenfalls die Start- und Endzeit des Zeitintervalls an, gibt die Labelgruppe an, zu der das Label hinzugefügt werden soll und bestätigt mit „OK“. 8. Das System erzeugt ein neues Label mit den angegebenen Daten. 9. Der Nutzer wählt unter „Datei“ den Menüpunkt „Label speichern“ oder benutzt das Tastenkürzel „Strg+S“. 10. Das System speichert alle neu hinzugefügten und bearbeiteten Label als Label-Informationen im Arbeitsverzeichnis.
Qualitätsanforderungen	• Es muss die Möglichkeit geben, bereits hinzugefügte Label zu bearbeiten. • Es muss die Möglichkeit geben, bereits hinzugefügte Label zu entfernen.

Fehlerfälle	• Es ist nicht genug Speicherplatz vorhanden, um die Label-Informationen zu speichern.

Name	**Daten interpretieren**
ID	UC-12
Beschreibung	Die Datensätze eines Probanden werden anhand von historischen Daten und Bewegungsparametern auf medizinischer Grundlage interpretiert.
Akteure	Entwickler, Auswerter
Anfangsbedingung	Es existieren analysierte, historische Datensätze und Bewegungsparameter des Probanden.
Abschlussbedingung	Die Daten wurden von einem Auswerter dahingehend interpretiert, ob Veränderungen in Bewegungsmustern vorliegen oder nicht.
Standardablauf	1. Der Entwickler stellt dem Auswerter die Daten des Probanden zur Verfügung. 2. Der Auswerter vergleicht den aktuellen Datensatz mit historischen Datensätzen des Probanden und interpretiert das Ergebnis.
Qualitätsanforderungen	
Fehlerfälle	

4.2. Anforderungsliste

In diesem Abschnitt werden die funktionalen und nicht-funktionalen Anforderungen des Systems beschrieben. Die Anforderungen sind nach den vier verschiedenen Ebenen des Systems strukturiert, sodass zunächst abstrakte Anforderungen durch die Beschreibung von Anforderungen der nächsten Systemebene immer weiter konkretisiert werden.

Die erste Systemebene ist die *grobe Systemebene*. Anforderungen der *groben Systemebene (z. B. A1)* beschreiben das gesamte System und legen daher die grobe Funktionalität des Systems fest. Die nächst-konkretere Systemebene ist die *funktionale Systemebene*. Anforderungen der *funktionalen Systemebene (z. B. A1.1)* beschreiben Teilvorgänge des Systems. Anforderungen der *fachlichen Systemebene (z. B. A1.1.1)* beschreiben fachlich motivierte Anforderungen, während Anforderungen der *technischen Systemebene (z. B. A1.1.1.1)* technische Aspekte des Systems spezifizieren.

Die beiden folgenden Tabellen stellen eine vollständige Auflistung aller spezifizierten Anforderungen des Systems und der GUI des Systems dar.

Auflistung der Anforderungen an das System

Kürzel	Anforderung
A1	Das System muss Bewegungsmuster in den Bewegungsdaten erkennen.
A1.1	Das System muss Originaldaten importieren.
A1.1.1	Das System muss die Daten aus Dateien im Original-Format auslesen.
A1.1.1.1	Das System muss die Rohdaten aus CSV-kompatiblen Textdateien auslesen.
A1.1.1.2	Das System muss die Metadaten aus INI-Textdateien auslesen.
A1.1.1.3	Wenn mehr als ein zusammenhängendes Aufnahmesegment im Originaldatensatz existiert, muss der Benutzer wählen, welches Segment importiert werden soll.
A1.1.2	Das System muss die Daten in Dateien im System-Format abspeichern.
A1.1.2.1	Das System muss Rohdaten im Binärformat abspeichern.
A1.1.2.2	Das System muss Metadaten im Extensible Markup Language (XML)-Format abspeichern.
A1.1.3	Das System wird die Rohdaten und Metadaten direkt aus dem Sensorgürtel importieren.
A1.2	Das System muss die Bewegungsdaten im System-Format vorhalten.
A1.2.1	Das System muss Bewegungsdaten im System-Format abspeichern und einlesen.
A1.2.1.1	Das System muss eine Verzeichnisstruktur für Bewegungsdaten im System-Format anlegen und verwalten.
A1.2.1.2	Das System muss Rohdaten und abgeleitete Datenreihen im Binärformat abspeichern und einlesen.
A1.2.1.2.2	Jedes Datum einer Datenreihe muss sich einem Zeitstempel zuordnen lassen.
A1.2.1.3	Das System muss Metadaten der Rohdaten, Label-Gruppen, Label-Informationen und Bewegungsparameter im XML-Format abspeichern und einlesen.
A1.2.1.4	Das System muss beliebige weitere Datenreihen mit Zeitbezug und Metainformationen (z.B. Name der Datenreihe) als Teil der Bewegungsdaten speichern.

A1.2.2	Das System muss einen Bewegungsdatensatz zur Zeit öffnen können und diesen für alle Aktionen verwenden.
A1.2.2.1	Das System muss es ermöglichen, den Bewegungsdatensatz anhand einer Probanden-ID und einem Aufnahmezeitpunkt zu identifizieren.
A1.2.2.2	Das System muss Label-Gruppen verwalten.
A1.2.2.2.1	Das System muss alle existierenden Label-Gruppen zurückliefern.
A1.2.2.2.2	Das System muss Label-Gruppen anlegen und löschen können.
A1.2.2.2.3	Das System muss Label-Informationen zu einer Label-Gruppe hinzufügen und entfernen können.
A1.2.2.2.4	Innerhalb einer Label-Gruppe dürfen sich die Zeitintervalle von Label-Informationen nicht überschneiden.
A1.2.2.3	Das System muss alle existierenden Label-Informationen einer Label-Gruppe eines Label-Typs zurückliefern.
A1.2.2.4	Das System muss alle existierenden Label-Informationen einer Label-Gruppe, die innerhalb eines gegebenen Zeitintervalls liegen, eines Label-Typs zurückliefern.
A1.2.3	Das System muss Parameterdaten für Algorithmen als Teil abgeleiteter Einzel-Daten vorhalten.
A1.2.3.1	Das System muss die Möglichkeit bieten, für eine Algorithmus-Ausführung die vom Benutzer eingestellten Algorithmus-Parameter zu speichern.
A1.2.3.2	Das System sollte die Möglichkeit bieten, für eine Algorithmus-Ausführung bereits gespeicherte Parameter wiederzuverwenden.
A1.2.3.3	Das System muss gelernte Modelle von maschinellen Lernalgorithmen (Trainings-Algorithmen) speichern.
A1.2.3.3.1	Das System muss die Möglichkeit bieten, ein Speicherort für ein gelerntes Modell auszuwählen.
A1.2.3.4	Das System muss gelernte Modelle für maschinelle Lernalgorithmen (Klassifizierungs-Algorithmen) laden.
A1.2.3.4.1	Das System muss die Möglichkeit bieten, ein zu verwendendes gelerntes Modell bei Ausführung eines Klassifizierungs-Algorithmus auszuwählen.
A1.3	Das System muss Algorithmen zur Verarbeitung und Erzeugung von Bewegungsdaten bereitstellen.

A1.3.1	Das System muss Algorithmen anbieten, die Rohdaten vorverarbeiten.
A1.3.1.1	Das System muss folgende Algorithmen für die Vorverarbeitung von Rohdaten anbieten: Gauß-Filter Highpass-Filter, Lowpass-Filter und Median-Filter.
A1.3.2	Das System sollte Algorithmen anbieten, die Assessment-Test-Bewegungsmuster korrekt erkennen.
A1.3.2.1	Diese Algorithmen müssen Label-Informationen, die Bewegungsmuster ihren Zeitintervallen zuordnen, erstellen.
A1.3.3	Das System muss Algorithmen anbieten, die Alltags-Bewegungsmuster korrekt erkennen.
A1.3.3.1	Diese Algorithmen müssen Label-Informationen, die Bewegungsmuster ihren Zeitintervallen zuordnen, erstellen.
A1.3.3.2	Diese Algorithmen müssen folgende Alltags-Bewegungsmuster erkennen: Gehen, Treppensteigen, Stehen, Sitzen, Auf dem Rücken liegen, Auf der Seite liegen, Hinsetzen und Aufstehen.
A1.3.4	Die Algorithmen müssen uniforme Ein- und Ausgabeschnittstellen besitzen.
A1.3.4.1	Ein Algorithmus muss eine Spezifikation seiner Schnittstellen bereitstellen.
A1.3.4.2	Ein Algorithmus muss feststellen, ob seine Schnittstellen zu einem anderen Algorithmus kompatibel sind.
A1.3.4.3	Ein Algorithmus erhält einen oder mehrere Bewegungsdatensätze als zusätzliche Eingabe.
A1.3.5	Das System sollte eine sequentielle Ausführung mehrerer Algorithmen ermöglichen, sodass die Ausgabedaten eines Algorithmus als Eingabedaten des Nächsten dienen.
A1.3.5.1	Das System muss die Beschreibung einer Ausführungssequenz aus einer Konfigurationsdatei laden.
A1.3.5.2	Die Konfigurationsdateien zur Beschreibung einer Ausführungssequenz müssen als Textdateien vorliegen, sodass sie mit einem Texteditor bearbeitet werden können.
A1.3.6	Die Algorithmen müssen große Datenmengen (mehrere GB bis TB) in annehmbarer Zeit verarbeiten.
A1.4	Das System muss die Daten im System-Format in CSV-Formate exportieren können.
A1.4.1	Das System muss ausgewählte Datenreihen in eine CSV-Datei schreiben können.
A1.4.2	Das System muss alle Label-Informationen in eine CSV-Datei schreiben können.

A2	Das System sollte für die erkannten Bewegungsmuster (z.B. Gehen) Bewegungsparameter (z.B. Schrittlänge) bestimmen.
A2.1	Das System sollte Algorithmen für die Berechnung der Bewegungsparameter aus Bewegungsdaten bereitstellen.
A2.2	Das System sollte berechnete Bewegungsparameter vorhalten.
A3	Das System wird eine Bewertung anhand der Bewegungsparameter vornehmen.
A4	Das System muss langläufige Operationen durch Jobs organisieren.
A4.1	Das System muss Job-Beschreibungen für vom Benutzer geforderte Algorithmus-Ausführungen generieren.
A4.1.1	Eine Job-Beschreibung zur Ausführung eines Algorithmus enthält eine Algorithmussequenz, Algorithmusparameter, Bewegungsdatensatz-ID und ein Zeitintervall.
A4.2	Das System muss Job-Beschreibungen zwischen verschiedenen Hosts austauschen können.
A4.2.1	Das System muss Job-Beschreibungen in einer Datei speichern können.
A4.2.2	Das System muss Job-Beschreibungen aus einer Datei lesen können.
A4.3	Das System muss Job-Beschreibungen verarbeiten und so einen ausführbaren Job erzeugen.
A4.3.1	Ein Job hat einen Startzeitpunkt.
A4.3.2	Ein Job hat einen Namen.
A4.3.3	Ein Job hat einen der folgenden Zustände: in einer Warteschlage, bei der Ausführung, abgebrochen, beendet oder mit Fehler beendet.
A4.3.4	Das System muss den Job-Prozess überwachen, um Zustandsänderungen zu erkennen.
A4.4	Das System muss Jobs in verschiedenen Umgebungen ausführen.
A4.4.1	Das System muss Jobs auf dem lokalen Host ausführen.
A4.4.2	Das System muss Jobs auf dem HPC-Rechencluster ausführen.
A4.4.2.1	Das System muss gegebene Account- und Verbindungsinformationen verwenden, um sich beim HPC-Rechencluster zu authentifizieren.
A4.4.2.2	Das System muss Eingabedaten nach HPC übertragen, sofern sie nicht bereits vorliegen.

A4.4.2.3	Das System muss die Ergebnisse des Jobs von HPC herunterladen und lokal verfügbar machen.
A4.4.2.4	Es muss ein Arbeitsverzeichnis für den Job ausgewählt werden.
A4.4.3	Das System sollte Jobs auf einem entfernten Host ausführen.
A4.5	Ein Job muss als eigener Prozess ausgeführt werden.

Auflistung der Anforderungen an die GUI des Systems

Kürzel	**Anforderung**
G1	Das System muss eine GUI zur Verfügung stellen.
G1.1	Die GUI-Texte müssen internationalisierbar sein.
G1.2	Die GUI-Texte müssen deutsch lokalisiert sein.
G1.3	Die GUI muss die Möglichkeit bieten, ein Arbeitsverzeichnis für den Datensatzimport und die Datenreihenspeicherung auswählen zu können.
G1.3.1	Die GUI muss die Möglichkeit bieten, ein Standardarbeitsverzeichnis festlegen zu können.
G1.3.2	Die GUI muss die Möglichkeit bieten, das Arbeitsverzeichnis nachträglich zu ändern.
G1.3.3	Die GUI muss die Möglichkeit bieten, über einen Datei-Manager direkt das Arbeitsverzeichnis zu öffnen.
G1.4	Die GUI muss die Möglichkeit bieten, Originaldaten zu importieren.
G1.4.1	Wenn mehr als ein zusammenhängendes Aufnahmesegment in der gewählten Datei existiert, soll das korrekte Segment anhand der zusätzlichen Zeitstempeldaten erkannt werden.
G1.4.2	Wenn mehr als ein zusammenhängendes Aufnahmesegment in der gewählten Datei existiert und die automatische Erkennung des korrekten Segments fehlschlägt, muss die GUI vor dem Import alle Aufnahmesegmente anzeigen.
G1.4.2.1	Die GUI muss für jedes Aufnahmesegment Start-, Endzeitpunkt und Aufnahmedauer anzeigen.
G1.5	Die GUI muss die Möglichkeit bieten, Bewegungsdaten zu laden.
G1.5.1	Die GUI muss die Möglichkeit bieten, einen Datensatz von Bewegungsdaten im Binär-Format auszuwählen.

G1.5.2	Die GUI muss beim Laden eines Datensatzes dem Benutzer die Möglichkeit bieten, diesen anhand der Probanden-ID und der Aufnahmenummer (Zeit-Code) zu identifizieren.
G1.6	Die GUI muss die Möglichkeit bieten, Bewegungsdaten zu betrachten.
G1.6.1	Die GUI muss die Möglichkeit bieten, Datenreihen der Bewegungsdaten in einem Plot darzustellen.
G1.6.1.1	Die GUI muss die Möglichkeit bieten, auszuwählen, welche Datenreihen der Bewegungsdaten angezeigt werden.
G1.6.1.2	Die GUI muss die Möglichkeit bieten, ein zeitliches Betrachtungsintervall vorzugeben.
G1.6.1.3	Die GUI muss die Möglichkeit bieten, das Zeitintervall eines markierten Zeitachsen/Label-Leisten-/Plot-Bereichs als Betrachtungsintervall zu wählen.
G1.6.1.4	Die GUI muss die Möglichkeit bieten, in einem größeren Betrachtungsintervall die Kurven der Bewegungsdatenreihen mit Hilfe des Douglas-Peucker-Algorithmus zu glätten.
G1.6.1.5	Die GUI muss die Möglichkeit bieten, die Zeitachse und die horizontale (zeitliche) Auflösung der Plots zu skalieren.
G1.6.1.6	Die GUI muss die Möglichkeit bieten, die vertikale Auflösung der Plots zu skalieren.
G1.6.2	Die GUI muss die Möglichkeit bieten, die Datenreihen der Bewegungsdaten in einer Tabelle darzustellen.
G1.6.2.1	Die GUI muss die Möglichkeit bieten, auszuwählen, welche Datenreihen der Bewegungsdaten in der Tabelle angezeigt werden.
G1.7	Die GUI muss die Möglichkeit bieten, mit Label-Informationen zu arbeiten.
G1.7.1	Die GUI muss Label-Informationen auf der Zeitachse darstellen.
G1.7.1.1	Die GUI muss eine beliebige Anzahl von Label-Gruppen als eigene Zeilen auf der Zeitachse darstellen.
G1.7.1.2	Die GUI muss die Möglichkeit bieten, eine Label-Gruppe zur Ansicht hinzuzufügen.
G1.7.1.3	Die GUI muss die Möglichkeit bieten, eine Label-Gruppe aus der Ansicht zu entfernen.
G1.7.2	Die GUI muss die Möglichkeit bieten, Label-Informationen innerhalb einer Label-Gruppe anzulegen.
G1.7.2.1	Die GUI muss die Möglichkeit bieten, einem neuen Label ein markiertes Zeitintervall zuzuordnen.

G1.7.2.2	Die GUI muss die Möglichkeit bieten, einem neuen Label ein eingegebenen Zeitintervall zuzuordnen.
G1.7.2.3	Die GUI muss die Möglichkeit bieten, als Label-Typ Assessment-Tests und Alltags-Bewegungsmuster zu verwenden.
G1.7.3	Die GUI muss die Möglichkeit bieten, Label-Informationen zu bearbeiten.
G1.7.4	Die GUI muss die Möglichkeit bieten, Label-Informationen zu entfernen.
G1.8	Die GUI sollte die Möglichkeit bieten, Algorithmen auf Bewegungsdaten anzuwenden.
G1.8.1	Die GUI muss die Wahl bieten, Algorithmen lokal oder auf dem HPC-Rechencluster auszuführen.
G1.8.1.1	Die GUI muss eine Möglichkeit bieten, Verbindungsinformationen (Internetadresse, Benutzername, Passwort) für das HPC-Rechencluster einzugeben.
G1.8.1.2	Die GUI muss sensible Verbindungsinformationen (Passwort) maskiert anzeigen.
G1.8.2	Die GUI muss die Möglichkeit bieten, Parameter für den gewählten Algorithmus anzugeben.
G1.8.2.1	Die GUI muss die Möglichkeit bieten, einen oder mehrere Bewegungsdaten, auf denen der Algorithmus arbeitet, auszuwählen.
G1.8.2.2	Die GUI muss mit optionalen Parametern (d.h. Parameter wird mit Standardwert belegt, wenn vom Benutzer nicht angegeben) und versteckten Parametern (d.h. Parameter für den Benutzer nicht sichtbar) umgehen.
G1.8.3	Die GUI muss die Auswahlmöglichkeit bieten, das Job-Management nach der Algorithmusauswahl automatisch zu öffnen.
G1.9	Die GUI muss die Möglichkeit bieten, eine Log-Datei mit u.a. den bisher verwendeten Algorithmusparametern zu öffnen.
G1.10	Die GUI muss die Möglichkeit bieten, Label-Informationen zu speichern.
G1.11	Die GUI muss die Möglichkeit bieten, Bewegungsdaten zu exportieren.
G1.11.1	Die GUI muss die Möglichkeit bieten, ausgewählte Datenreihen eines Bewegungsdatensatzes als CSV-Datei zu exportieren.
G1.11.2	Die GUI muss die Möglichkeit bieten, alle Label-Informationen eines Bewegungsdatensatzes als CSV-Datei zu exportieren.

G1.12	Die GUI muss eine Ansicht bieten, um laufende Jobs zu verwalten.
G1.12.1	Die GUI muss eine Übersicht aller Jobs anbieten, die den jeweiligen Start- und Endzeitpunkt der Jobs sowie deren Zustand enthält.
G1.12.2	Die GUI muss es ermöglichen, Jobs abzubrechen.
G1.12.3	Der Benutzer muss über Statusänderungen von Jobs benachrichtigt werden.
G1.12.3.1	Der Benutzer muss für fehlgeschlagene Jobs von der Benachrichtigung aus Informationen zum Fehlschlag aufrufen können.
G1.12.3.2	Der Benutzer muss für erfolgreiche Jobs von der Benachrichtigung aus die resultierenden Datenreihen abrufen können.

4.3. Medizinproduktgesetz

Die Software, die im Rahmen dieser Projektgruppe entsteht, wird als Produkt zum Sensorgürtel entwickelt. Der Sensorgürtel ist zweckbestimmt, als Fitnessgerät auf den Markt gebracht zu werden. Da aber die Option offen gehalten werden soll, es als medizinisches Produkt zu vermarkten, liegt ein großes Augenmerk auf der Einhaltung von Kriterien, welche die Zulassungsbestimmungen vom Gesetz über Medizinproduktgesetz (MPG) vorschreiben, um eine reibungslose und schnelle Zulassungsbescheinigung zu erwirken. Das MPG ist ein deutsches Gesetz, das am 02.08.1994 angefertigt und zuletzt am 31.08.2015 geändert wurde. Das Gesetz entstand als Umsetzung der beschlossenen Richtlinien vom Rat der Europäischen Wirtschaftsgemeinschaft von 1990 und 1993 sowie der Richtlinie vom Europäischen Parlament im Jahr 1998.
Der Zweck des Gesetzes ist es,

> „[. . .] den Verkehr mit Medizinprodukten zu regeln und dadurch für die Sicherheit, Eignung und Leistung der Medizinprodukte sowie die Gesundheit und den erforderlichen Schutz der Patienten, Anwender und Dritter zu sorgen.“ [dJufV94]

Das MPG benennt explizit eine Software als ein Medizinprodukt, da es mit einem Instrument für diagnostische Zwecke im Verbund stehen kann [dJufV94]. Es lässt eine spätere Eignung des Sensorgürtels und der Auswertesoftware als Medizinprodukt zu, da es in Paragraf 3, Absatz 1 heißt, dass die Produkte u. a. zum Zwecke von:

„a) der Erkennung, Verhütung, Überwachung, Behandlung oder Linderung von Krankhei-

ten“ [dJufV94]

sowie

„c) der Untersuchung, der Ersetzung oder der Veränderung des anatomischen Aufbaus oder eines physiologischen Vorgangs [...]“ [dJufV94]

dienen sollen. Der Sensorgürtel kann durch die hardware-technische und software-technische Ausstattung später mit Hinweis auf die Erfüllung dieser beiden Punkte zur Erkennung und Vorbeugung von Erkrankungen im hohen Alter zugelassen werden. Jedoch muss die Software zunächst bestimmte Anforderungen erfüllen. Dazu zählt zunächst die Achtung von Datenschutz hinsichtlich identifizierender Informationen über die Benutzer der Hardware [dJufV94]. Zudem muss die Software für den deutschen Markt Informationen in deutscher Sprache vorliegen (Paragraf 11, Absatz 2). In begründeten Fällen kann andere für die Anwender leicht verständliche Sprache gewährt werden. Nach der Richtlinie 93/42/EWG des Rates vom 14. Juni 1993 über Medizingesetze muss mit einer Software auch ein Handbuch angefertigt werden, da die technischen Kenntnisse der Anwender der Software berücksichtigt werden müssen [Gem93]. Zudem sind den Benutzern die Genauigkeitsgrenzen der Auswertungen schriftlich zu vermitteln [Gem93]. Für die Zertifizierung und spätere Verlängerung der notwendigen Zertifizierung mit der CE-Kennzeichnung ist nach Paragraph 26, Absatz 3 eine Prüfstelle oder Prüfbehörde zuständig. Hierfür müssen u. a. Unterlagen zur Entwicklung der Software vorliegen. Für die Prüfstelle muss die Software „[...] entsprechend dem Stand der Technik validiert werden, wobei die Grundsätze des Software-Lebenszyklus, des Risikomanagements, der Validierung und Verifizierung zu berücksichtigen sind.“ [Gem93].

Das Qualitätssicherungssystem muss schriftlich einsehbar sein und alle „Einzelheiten, Anforderungen und Vorkehrungen, die der Hersteller für sein Qualitätssicherungssystem zugrunde legt“ [Gem93] beinhalten. Die Verschriftung muss systematisch und geordnet in einer Dokumentation erfolgen, „beispielsweise in Form von Programmen, Plänen, Handbüchern und Aufzeichnungen zur Qualitätssicherung“ [Gem93]. Zu den Unterlagen gehören auch die anzuwendenden Normen und die Ergebnisse der Risikoanalyse sowie die Beschreibung der Lösungen zur Einhaltung grundlegender Anforderungen [Gem93].

5. Konzeption

In diesem Kapitel findet eine ausführliche Beschreibung der Konzepte statt, die für die Realisierung des Projektes genutzt wurden. Als erstes werden die Architektur der Software dargestellt und Technologientscheidungen begründet. Es werden Möglichkeiten zur Nutzung verschiedener Programmiersprachen dargelegt und die Dateiformate der Anwendung spezifiziert. Die Erstellung von Referenzinformationen, zu dem unter anderem die manuelle Beschriftung der Daten gehört, wird nachfolgend beschrieben und das Design der GUI wird anhand eines Mockups vorgestellt. Das anschließende Unterkapitel beschäftigt sich mit den verwendeten Algorithmen, welche von einfachen Filtern bis hin zu Klassifikations- und Schritterkennungsalgorithmen reichen. Anschließend wird das Konzept des maschinellen Lernens bezüglich der Verwendung im Projekt sowie die damit verbundendene Paramateroptimierung erläutert.

5.1. Architektur

Im Rahmen des Projekts werden vier Softwarekomponenten entwickelt: eine GUI, eine Kommandozeilenanwendung (CLI), der Job-Executor (JE) und eine Sammlung von Code, der von allen drei Komponenten verwendet wird (shared). Diese Architektur ist in Abbildung 31 dargestellt.

Die GUI wird benötigt, um die aufgenommenen Daten zu visualisieren und zu bearbeiten. Durch die GUI ist es einfach möglich, die verschiedenen Sensordaten zu betrachten, einzelne Segmente zu markieren und mit einer Beschriftung zu versehen. Weiterhin kann der Datensatz zugeschnitten werden. Diese Funktionalitäten sind essentiell zur Vorbereitung der Daten für den Einsatz von Machine Learning. Schließlich können alle entwickelten Algorithmen über die GUI ausgeführt werden. Dies geschieht entweder in Threads, einem externen Prozess oder auf einer per SSH erreichbaren Maschine wie dem High-

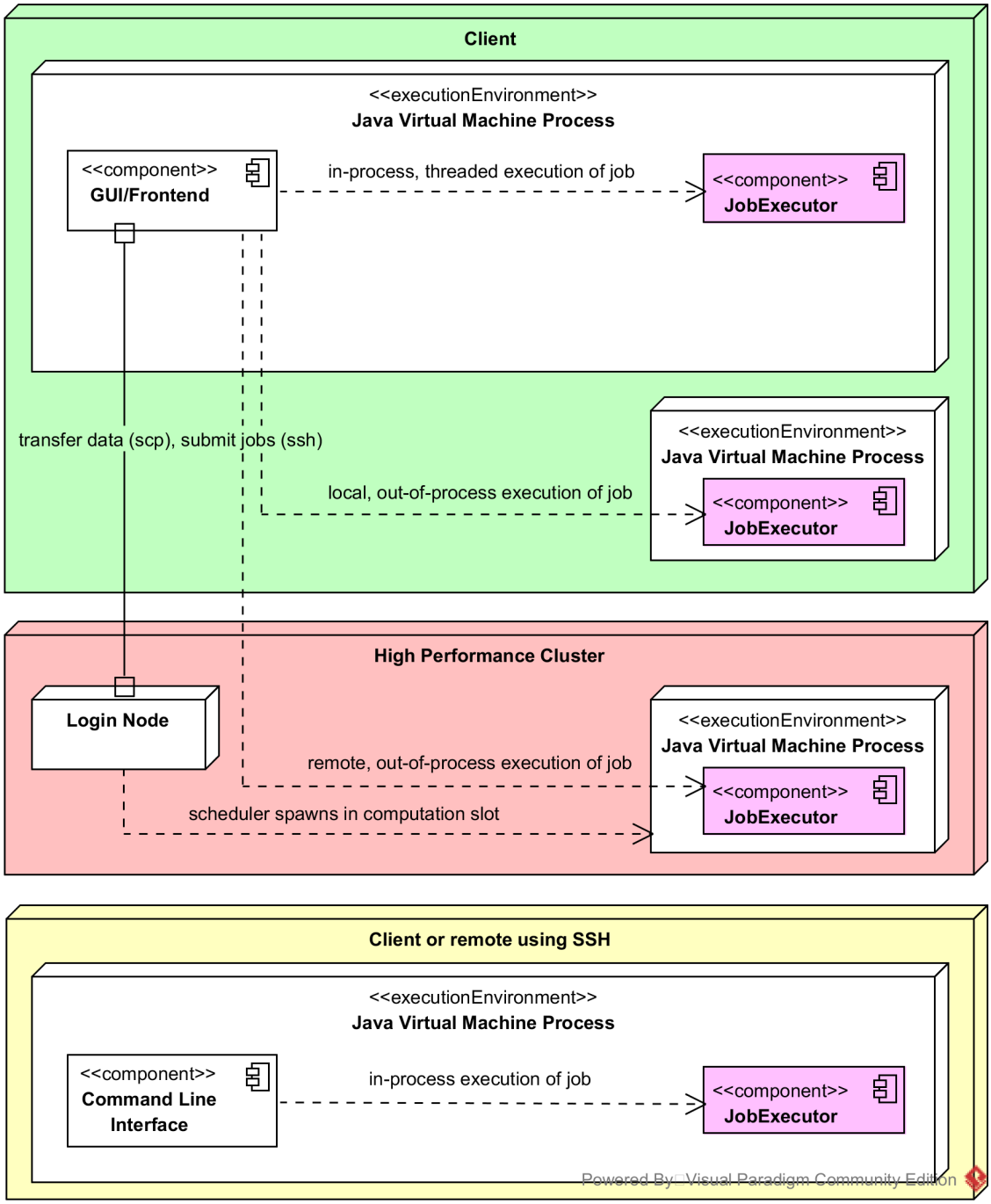


Abbildung 31.: Die Architektur der Software

Performance-Cluster der Universität.

Die CLI stellt überwiegend dieselbe Funktionalität wie die GUI zur Verfügung, allerdings auf der Kommandozeilenebene. Auf diese Weise lassen sich bestimmte Anwendungsszenarien als Skript festhalten und schnell erneut ausführen. Das ist hilfreich sowohl bei der Entwicklung der Anwendung als auch bei Aufgaben, die Automatisierung benötigen. Außerdem ließ sich die Anwendung so auf dem HPC der Universität nutzen, bevor die Integration in die GUI abgeschlossen war.

Der JE wird benötigt, um Algorithmen auszuführen. Dies kann einerseits lokal genutzt werden, ist jedoch überwiegend für den Einsatz auf dem HPC gedacht. Die GUI soll per SSH mit dem HPC kommunizieren und den JE nutzen, um einen Job auszuführen.

5.2. Technologieentscheidung

Zu Beginn des Projektes mussten einige grundlegende Technologieentscheidungen getroffen werden, damit die zu entwickelnde Software die gestellten Anforderungen aus Abschnitt 4.2 optimal erfüllen kann. Eine wichtige Entscheidung war dabei die Auswahl von Java als Programmiersprache. Anschließend wurde Fastutil als eine geeignete Primitive-Collection-Bibliothek gewählt, da diese schnelle Lese- und Schreibzugriffe auf die Sensordaten im Hauptspeicher gewährleistet. Als geeignete Methode zur Datenverwaltung wurde ein Float-Binärdateiformat gewählt, da dieses die besten Zugriffszeiten ermöglicht und dabei wenig Speicherplatz belegt. Die Technologieentscheidungen werden in den folgenden Abschnitten detailliert erläutert.

5.2.1. Auswahl der Programmiersprache

Für die Entwicklung der Software im Rahmen der Projektarbeit ist es notwendig eine für die Lösung der Aufgabe geeignete Programmiersprache zu finden. Zur engeren Auswahl standen die beiden objektorientierten Programmiersprachen Matrix Laboratory (MATLAB) und Java.
MATLAB ist als kommerzielle Programmiersprache und Programmierumgebung seit 1984 erhältlich. Sie erlaubt funktional, imperativ, prozedural und objektorientiert zu programmieren. Zu den Anwendungsgebiet gehören laut Gramlich [Gra09]: Numerische Berechnungen aller Art, Entwicklung von Algorithmen, Modellierung, Simulation und Entwicklung von

Prototypen technischer und wirtschaftlicher Probleme, Analyse, Auswertung und grafische Darstellung von insbesondere größerer und häufig automatisiert erfasster Datenmengen [Sch11], Visualisierungen, wissenschaftliche und technische Darstellungen sowie Applikationsentwicklung mit Aufbau einer grafischen Benutzerschnittstelle.
Zu den besonderen Eigenschaften von MATLAB gehören laut Gramlich [Gra09]:

- Einfache und intuitive Syntax
- Möglichkeit in einer höheren Programmiersprache zu programmieren
- Möglichkeit der interaktiven Arbeit, die leichte Fehlersuche erlaubt
- Viele Grafik- und Visualisierungsmöglichkeiten
- Erweiterungsmöglichkeiten durch Toolboxen

Die Programmiersprache Java erschien im Jahr 1995 und liegt als Plattform sowohl in freier, als auch unfreier Form zur Nutzung bereit. Mit Java ist es möglich sowohl imperativ, als auch objektorientiert zu programmieren. Es existieren zur Nutzung mehrere Entwicklungsumgebungen in den Lizenz-Formen: proprietär, freeware und open source. Die Einsatzgebiete für Java sind sehr vielfältig und für Anwendungen mit und ohne GUI einsetzbar. Großen Einsatz findet Java vor allem in verteilten Systemen. Java-Programme haben nach Goll und Heinisch [GH16] die Eigenschaft, dass sie einfach, stabil, objektorientiert, verteilbar, sicher und portierbar sind. Im Folgenden werden die Eigenschaften kurz erläutert:

- Einfachheit und Stabilität: Java wurde so konzipiert, dass es im Vergleich zu den beiden Programmiersprachen C und C++ keine unsicheren Sprachkonstrukte enthält.
- Objektorientierung: Durch die objektorientierte Programmierung ergeben sich viele Vorteile. Dazu zählen: bessere Abstraktionsmöglichkeit, starke Möglichkeit der hierarchischen Abstraktion, methodische Durchgängigkeit, evolutionäre Entwicklung wird begünstigt, Vereinfachung von Gruppenarbeiten und vor allem höhere Sicherheit und Qualität der Software [Sla14]
- Verteilbarkeit: Java-Programme können auf verschiedene Rechner verteilt eingesetzt werden.
- Sicherheit: „Java hat ein mehrstufiges Sicherheitskonzept, das die Ausführung von kritischen oder für das System gefährlichen Operationen verhindert“ [GH16].
- Portierbarkeit: Durch die Erzeugung eines Byte-Codes sind die Java-Programme auf un-

terschiedlichen Betriebs- und Hardwaresystemen einsetzbar, wenn vorher eine Virtuelle Maschine für Java installiert wird [GH16].

Obwohl auf den ersten Blick MATLAB für unser Vorhaben besser geeignet scheint, haben wir uns dennoch für die Programmiersprache Java entschieden. Hierfür sind verschiedene Punkte verantwortlich gewesen. Der entscheidende Faktor ist die Ausführungszeit. Da eine große Datenmenge verarbeitet werden muss, ist es wünschenswert, dass die Programmiersprache schnelle Ausführungszeiten ermöglicht. In einer Studie der beiden amerikanischen Universitäten University of Maryland und University of Pennsylvania 2014 wird die Performanz verschiedener Sprachen verglichen, darunter auch MATLAB und Java. Als Benchmark wurde ein stochastisches Wachstumsmodell in den unterschiedlichen Programmiersprachen implementiert und auf Windows- und Mac-Rechnern berechnet.Die genaueren Hardware-Spezifikationen können der Studie entnommen werden [AFV14]. Für die Java-Implementierung wurde das JDK8u5 verwendet, bei MATLAB kam die Version 2014a zum Einsatz. Die Ergebnisse des Benchmarks zeigen, dass MATLAB circa 4 mal langsamer als Java ist.

Eine zweite Rolle spielt die deutlich höhere Verbreitung von Java, im Vergleich zu MATLAB. Mit dem „TIOBE Programming Community Index“ wird seit 2001 monatlich ein aktualisiertes Ranking von den 20 populärsten Programmiersprachen herausgegeben [sB16]. Java hat seit 2001 durchgängig entweder den ersten oder zweiten Platz belegt, MATLAB hingegen immer Plätze im unteren Bereich bekommen. Die Beliebtheit von Java ist nicht zuletzt auf die oben erwähnten Eigenschaften der Sprache zurückzuführen. Daher wird auch an Univesitäten oftmals Java als Lehrsprache eingesetzt, sodass die Teilnehmer dieser Projektgruppe bereits gute Kenntnisse und Erfahrungen mit Java haben. Auch in Hinblick auf die Wartbarkeit und Erweiterbarkeit der zu entwickelnden Software ist eine populäre Programmiersprache vorteilhaft, da sich so die EInarbeitungszeit auf ein Minimum reduziert.

Zusammengefasst haben die schnellere Ausführung von mathematischen Algorithmen sowie der höhere Kenntnisstand bei den Teilnehmern der Projektgruppe zur Entscheidung für die Programmiersprache Java geführt. Durch die Integrierbarkeit anderer Programmiersprachen in Java ist es dennoch möglich, dass bestimmte Funktionen in einer performanteren Programmiersprache implementiert und in die Java-Anwendung eingebunden werden. Dieses Prinzip ist in Abschnitt 5.3 genauer beschrieben.

5.2.2. Auswahl einer geeigneten Primitive-Collection-Bibliothek

Wie im Abschnitt 3.2 beschrieben wurde, werden die Datenreihen der einzelnen Sensoren des Gürtels in einer CSV-Datei als reelle Werte hinterlegt. Damit das System auf diese Werte zugreifen kann, müssen sie im Java-Programm geladen werden. Die einzelnen Werte der CSV-Datei sollen dabei in Java als Datentyp double gespeichert werden. Die Schreib- und Lesezugriffe auf diese Werte müssen möglichst schnell sein, um effiziente Algorithmenlaufzeiten zu erzielen.

Um eine variable Anzahl von double-Werten in Java zu speichern, bietet Java einige Möglichkeiten wie z. B. ArrayList. Solche Java-Collections sind allerdings wenig effizient, da Java primitive Datentypen wie double in entsprechende Java-Objekte einpackt. Grund dafür ist, dass Java-Collections nur Objekte speichern können, welche von der Klasse Object erben und primitive Datentypen erben in Java nicht von Object.

Um eine effiziente Speicherung der double-Werte zu gewährleisten, muss daher auf optimierte Bibliotheken zurückgegriffen werden. In Frage kommen die Bibliotheken Colt[fNR04], Trove[EPRF12] und Fastutil[Vig]. Aufgrund des hohen Alters von Colt (aktuelle Version datiert von 2004), wurde Colt allerdings nicht näher betrachtet.

Um eine fundierte Entscheidung für eine der beiden anderen Bibliotheken treffen zu können, wurde eine Performance-Evaluierung durchgeführt. Da entweder die Klasse TDoubleArrayList von Trove oder die Klasse DoubleArrayList von Fastutil verwendet werden sollen, wurden beide Klassen auf die Zugriffszeiten (Schreib- und Leseoperationen der darin gespeicherten double-Werte) evaluiert. Als Referenz wurde ArrayList<Double> ebenfalls evaluiert. Die Evaluierung wurde mit den Datengrößen 1.000.000 Werte und 20.000.000 Werte durchgeführt. Für beide Datengrößen wurde die Evaluierung fünfmal wiederholt und die durchschnittlichen Schreib- und Lesezeiten ermittelt. Die Tabelle 5.2.2 zeigt die Ergebnisse der Performance-Evaluierung.

Aufgrund der deutlich kürzeren Schreib-Zugriffszeit bei der Datensatz-Größe von 20.000.000 Werte fiel die Entscheidung für die Verwendung von Fastutil.

5.2.3. Datenverwaltung

Das Projekt stellt hohe Anforderungen an die zu entwickelnde Software bzgl. der Performanz. Um diesen Anforderungen gerecht zu werden, muss nicht nur die Software

	java.util. ArrayList<Double>	Trove TDoubleArrayList	Fastutil DoubleArrayList
Datensatz-Größe: 1.000.000 Werte			
Schreibzeit [ns]	5.492.610	2.489.361	2.239.330
Lesezeit [ns]	1.430.251	846.611	830.270
Datensatz-Größe: 20.000.000 Werte			
Schreibzeit [ns]	635.054.399	90.206.783	63.517.613
Lesezeit [ns]	13.259.024	5.748.034	5.724.367

performant implementiert sein, sondern auch das Ein- und Ausgabeformat der Software spielt hierfür eine entscheidende Rolle. Das zu wählende Format soll einerseits so wenig Speicherplatz wie möglich zur Speicherung der Daten verwenden, damit der Dateitransfer zwischen verschiedenen Rechnern möglichst schnell ist. Neben der Dateigröße sind aber auch die Lese- und die Schreibgeschwindigkeit des Dateiformats von entscheidender Bedeutung, da die Anwendung nicht unnötig viel Zeit mit Ein- und Ausgabe verschwenden soll. Diese Zeit wäre in die Analyse der Daten wesentlich besser investiert.

Aus diesem Grund wurden einige relevante Dateiformate betrachtet und entsprechend der zuvor erwähnten Kriterien evaluiert. Zur generellen Auswahl standen hierbei die Speicherung der Daten als Textdateien, in einer Datenbank oder als Binärdateien.

Während Textdateien nicht weiter unterschieden werden können, stellt sich bei Binärdateien noch die Frage nach der Präzision, sprich wie viele Bits zur Speicherung eines Datums verwendet werden. In Unterabschnitt 5.2.1 wurde bereits erläutert, dass die Anwendung in der Programmiersprache Java geschrieben wird. Java bietet zur Speicherung von Gleitkommazahlen die zwei Datentypen „float“ und „double“ an, die gemäß des IEEE 754-Standards mit 32 bzw. 64 Bit kodiert sind [IEE85]. Daher muss evaluiert werden, welcher Datentyp sich besser eignet.

Die Suche nach einer geeigneten Datenbank ist etwas schwerer, da es eine Vielzahl an Datenbankmanagementsystem (DBMS) gibt. Jedes fällt dabei in eine bestimmte Klasse von verschiedenen Datenbankmodellen. Im Folgendem werden diese kurz vorgestellt und anhand verschiedener Kriterien wird eine Auswahl getroffen, die im Anschluss in die Dateiformatevaluation Einzug findet.

Datenbankmodelle

Es existieren verschiedene Datenbankmodelle, die sich in ihrer Art unterscheiden, wie sie die Daten abspeichern. Im Folgendem werden kurz das relationale und das objektorientierte Datenbankmodell angesprochen. Es gibt noch einige weitere Modelle, diese sind aber nicht mit unseren Daten vereinbar, da bspw. das hierarchische Datenbankmodell hierarchische Daten voraussetzt.

Relationales Datenbankmodell Das relationale Datenbankmodell ist das am weitesten verbreitete Modell, das in der Datenbankentwicklung als Standard genutzt wird. Eine Datenbank kann man sich als eine Ansammlung von Tabellen und Beziehungen vorstellen, die miteinander verknüpft sind. Da das relationale Modell lediglich Tupelmengen kennt, die aus Werten bestehen, müssen komplexe Anwendungsobjekte aus den einzelnen Relationen wiederhergestellt werden. Dies kann zu unübersichtlichen Abfragen führen.

Objektorientiertes Datenbankmodell Mit dem Aufkommen objektorientierter Programmiersprachen wurden zunehmend Objektdatenbanken angeboten. Damit können Objekte aus OO-Sprachen wie Java direkt in der Datenbank gehalten werden – eine Abbildung der Objekte auf die relationale Tabellenstruktur, das objektrelationale Mapping, ist dann nicht mehr notwendig. Objektdatenbanken haben jedoch noch immer Nachteile gegenüber relationalen Datenbanken bei der Verarbeitung großer Datenmengen.

Fazit und Auswahl Im Hinblick auf die Anforderungen, hier im speziellen, dass große Datenmengen verarbeitet werden müssen, kann das objektorientierte Datenbankmodell von der Auswahl ausgeschlossen werden. Somit fällt die Wahl auf das relationale Datenbankmodell. Den Nachteil bezüglich der unübersichtlichen Abfragen, den dieses Modell mit sich bringt, kann voraussichtlich umgangen werden, da das geplante Datenmodell überschaubar ist. Auch alle Anforderungen aus Abschnitt 4.2 können mit diesem Modell abgedeckt werden.

Datenbankmanagementsysteme

Für den praktischen Einsatz von relationalen Datenbankmodellen ist eine Vielzahl von DBMS geeignet. Anhand von konkreten Bewertungskriterien muss diese Menge gegenein-

ander abgewogen werden, um ein für das Projekt geeignetes DBMS auswählen zu können. Zu diesen Kriterien gehören: Popularität, Performanz, unterstützte Betriebssysteme und JDBC-Verfügbarkeit.

Popularität Die Firma „solid IT“ bietet auf der Website db-engines.de eine Übersicht zu der Popularität vieler DBMS. Weit abgeschlagen von anderen Systemen belegen Oracle, MySQL und Microsoft SQL Server die ersten drei Plätze. Danach folgen noch PostgreSQL, IBM DB2 und Microsoft Access. Aus diesem Grund fiel die grobe Auswahl zunächst auf diese DBMS.

Performanz In der Ausgabe vom Februar 2012 des „Journal of Computer Science & Research“ [Bas12] wurde die Performanz aller zuvor genannten DBMS gegeneinander verglichen, mit Ausnahme von PostgreSQL. Daher wurde dieses System vorab aus der Bewertung ausgeschlossen. Der Benchmark betrachtet einmal die Ausführungszeit von Anfragen, sowie die CPU- und Speicherauslastung. Die Durchschnittswerte sind in Abbildung 32 zu sehen und fließen mit in das Fazit ein.

Betriebssystem Interessant ist auch, auf welchen Betriebssystemen die jeweiligen DBMS laufen. Während MySQL eine ganze Bandbreite unterstützt, läuft bspw. Microsoft Access nur auf Windows, und auch die anderen DBMS unterliegen einigen Einschränkungen.

JDBC Da die Software mit Java entwickelt wird, ist es notwendig, dass das verwendete DBMS einen JDBC-Treiber besitzt. Durch diesen kann man von Java aus direkt mit dem DBMS kommunizieren. Bis auf Microsoft Access gibt es für jedes System diese Schnittstelle.

Fazit und Auswahl Um eine Auswahl treffen zu können, werden für die Popularität, Ausführungszeit, CPU- und RAM-Auslastung eine Reihenfolge erstellt und jeweils 1 bis 5 Punkte vergeben, wobei 5 Punkte die bestmöglichste Punktzahl ist. Da fast alle DBMS sowohl Windows als auch Linux unterstützen, wird hier die Bewertung auf 1 bis 3 Punkte reduziert. Bei der Bewertung der Verfügbarkeit eines JDBC-Treibers kann nur zwischen vorhanden (1 Punkt) und nicht vorhanden (0 Punkte) unterschieden werden. Da diese Schnittstelle aber einige wertvolle Vorteile mit sich bringt, wird das Vorhandensein mit

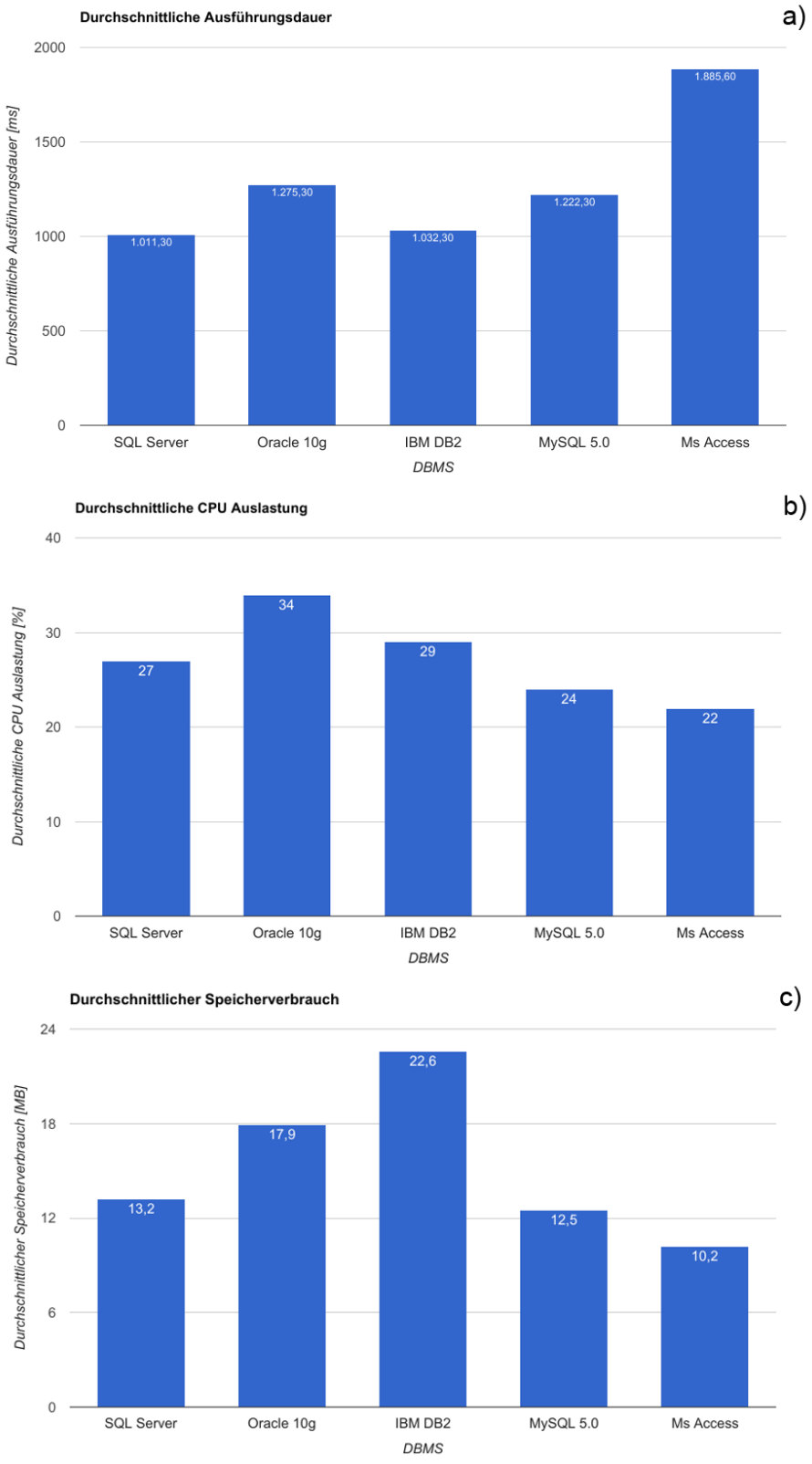


Abbildung 32.: Durchschnittsergebnisse der Benchmarks: a) Ausführungszeit von Anfragen, b) CPU-Auslastung und c) Speicherauslastung

	MS SQL Server	MySQL	IBM DB2	Oracle	MS Access
Popularität	3	4	2	5	1
Ausführungszeit	4	3	5	2	1
CPU-Auslastung	3	4	2	1	5
RAM-Auslastung	3	4	1	2	5
Betriebssysteme	2	3	2	2	1
JDBC-Treiber	3	3	3	3	0
Summe	18	21	15	15	13

Tabelle 23.: Bewertung der Datenbankmanagementsysteme

dreifacher Wertung gewichtet. Wie in Tabelle 23 zu sehen ist, erfüllt MySQL die Kriterien am besten.

Evaluation der Dateiformate

Nun standen die zu evaluierende Dateiformate zur Speicherung der Daten fest: Textdatei, Float-Binärdatei, Double-Binärdatei und eine MySQL-Datenbank. Kurzfristig wurde auch noch entschieden, SQLite mit in die Evaluierung aufzunehmen. SQLite ist eine Programm-Bibliothek, die ohne Datenbank-Server auf einer einzelnen Datei arbeitet. In der Datei sind alle Daten enthalten und der Zugriff läuft wie auch bei MySQL über SQL.

Für die Tests wurden zwei künstliche Datensätze erstellt, die für jeden Sensor eine Million bzw. 10 Millionen Werte enthalten. Wie eingangs erwähnt, werden die Dateiformate in Hinsicht auf Dateigröße, Lese- und Schreibgeschwindigkeit verglichen. Hierfür wurden verschiedene Programme geschrieben, die diese Werte für die verschiedenen Dateiformate ermitteln. Um die Nachvollziehbarkeit zu gewährleisten, wird der dafür verwendete Programmcode mit dem gesamten Projekt abgegeben. Bei der Implementierung wurde viel Wert auf eine effiziente Ausführung gelegt. Hierfür wurden beispielsweise Buffer verwendet, um die Menge an I/O-Operation mit dem Speichermedium zu reduzieren.

Die Ergebnisse dieser Programme sind in Abbildung 38 zu sehen. Es ist zu erkennen, dass die Binärformate in Bezug auf die Performanz am besten abschneiden, während die Datenbanken deutlich zu langsam für die Anforderungen sind. MySQL war sogar so langsam, dass das Schreiben und Lesen für 10 Millionen Werte pro Sensor nicht in hinnehmbarer Zeit ausgeführt werden konnte.

Und auch bei der Dateigröße liegen die Binärformate vor den Konkurrenten, während das Float-Binärformat nochmal deutlich besser ist als das Double-Binärformat.

Die ermittelten, absoluten Werte sind natürlich immer von dem System abhängig, auf dem die Messungen durchgeführt werden. Dennoch ist das relative Verhältnis der verschiedenen Dateiformate gültig, sodass dadurch eine Entscheidung für ein bestimmtes Dateiformat getroffen werden kann.

Aus diesen Gründen fiel die Entscheidung zugunsten der Float-Binärdateien aus. Die Lese- und Schreibgeschwindigkeiten sind mit am besten und der reduzierte Speicherverbrauch ist von Vorteil für einen Dateitransfer. Die mangelnde Präzision im Vergleich zu den Double-Binärdateien kann vernachlässigt werden, da die Rohdaten des Sensorgürtels nur bis auf einige Nachkommastellen genau sind und daher kein Genauigkeitsverlust befürchtet werden muss. Intern in der Anwendung wird aber weiterhin mit dem Datentyp double gerechnet, damit es hier zu keinen Ungenauigkeiten kommt.

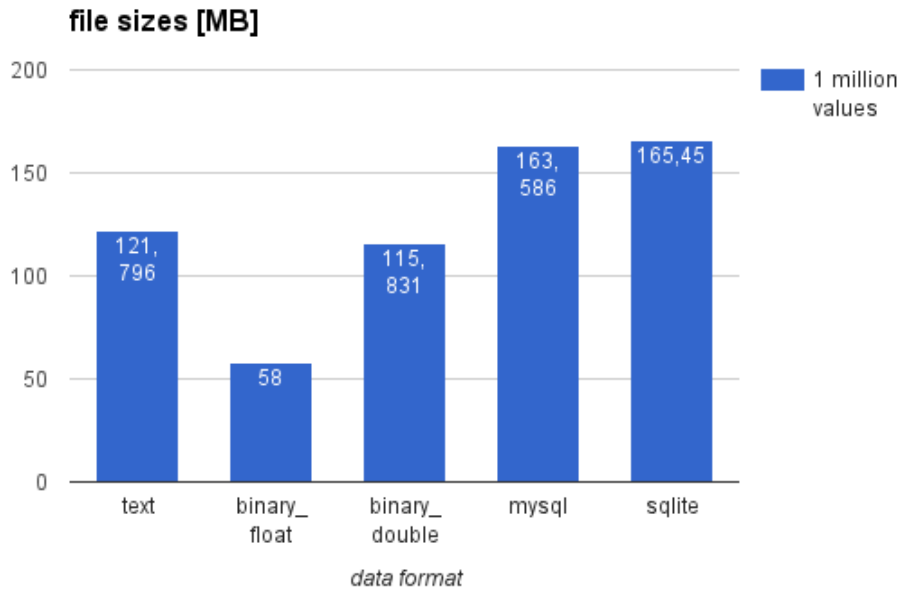


Abbildung 33.: Evaluationsergebnisse der Dateiformate - file size [MB] 1 million values

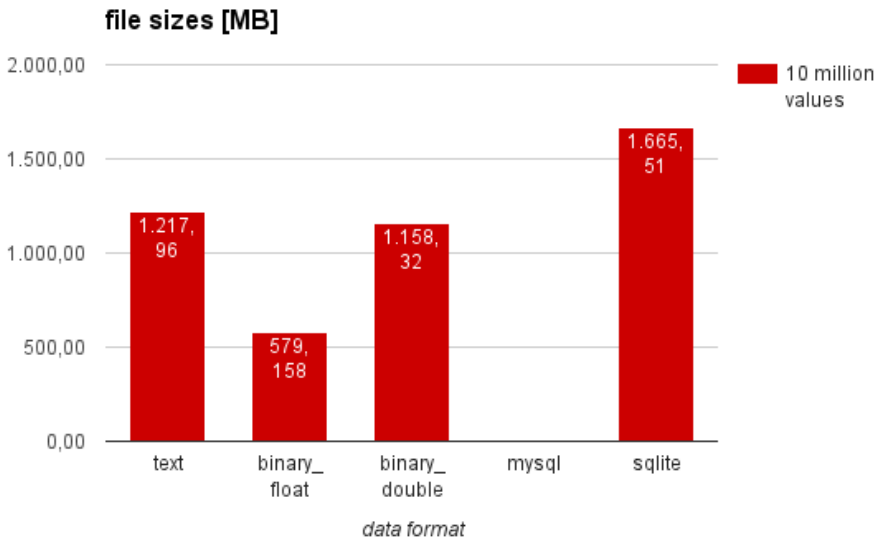


Abbildung 34.: Evaluationsergebnisse der Dateiformate - file size [MB] 10 million values

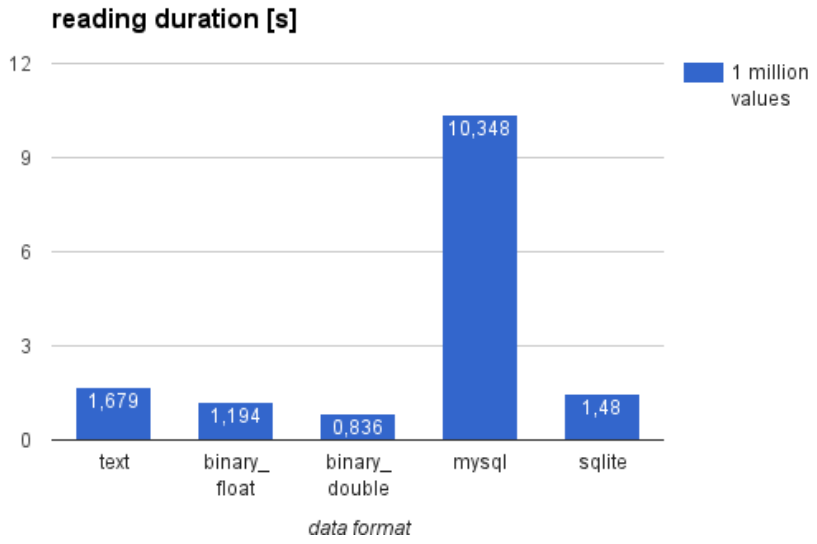


Abbildung 35.: Evaluationsergebnisse der Dateiformate - reading duration [s] 1 million values

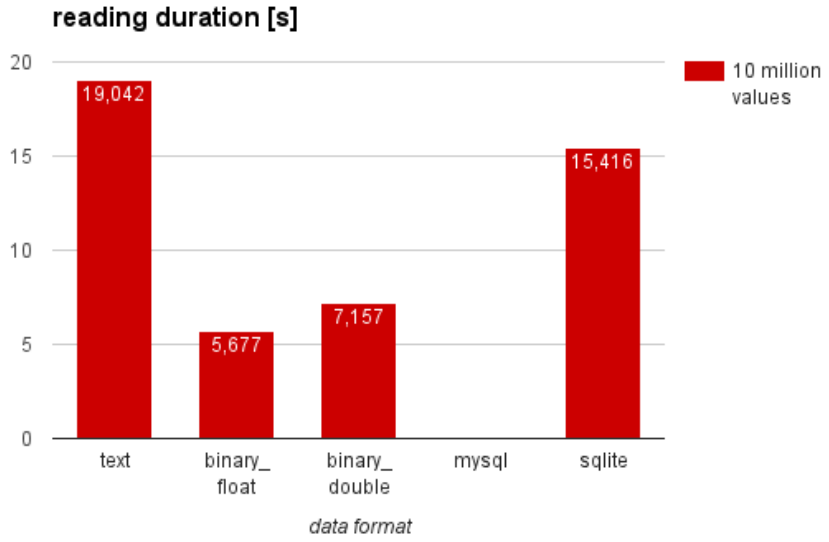


Abbildung 36.: Evaluationsergebnisse der Dateiformate - reading duration [s] 10 million values

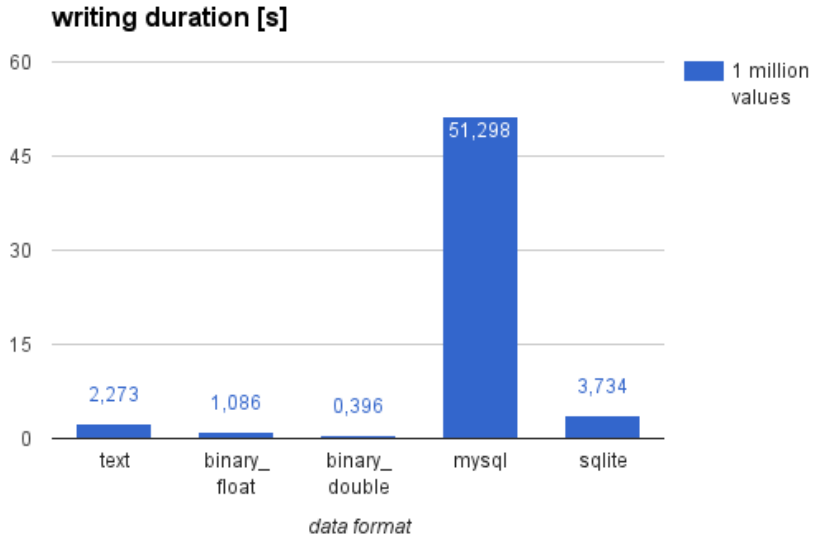


Abbildung 37.: Evaluationsergebnisse der Dateiformate - writing duration [s] 1 million values

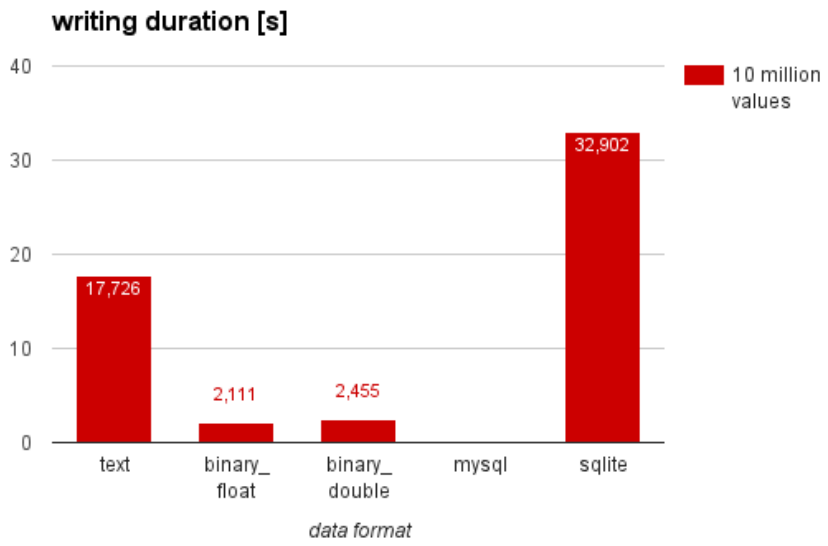


Abbildung 38.: Evaluationsergebnisse der Dateiformate - writing duration [s] 10 million values

5.3. Unterstützung anderer Programmiersprachen

Die im Rahmen dieser Projektgruppe entwickelte Java-Anwendung unterstützt unterschiedliche Programmiersprachen. Dazu zählen alle Java Virtual Machine (JVM)-kompatiblen Sprachen sowie über Java Native Interface (JNI) auch Programme in C oder C++. Programme, die mit MATLAB erstellt wurden, lassen sich ebenfalls einbinden.

5.3.1. JVM-Sprachen

Als JVM-Sprachen werden diejenigen Programmiersprachen bezeichnet, deren Programme durch eine JVM ausführbar sind. Im Gegensatz zu Java zeichnen sie sich beispielsweise durch reduzierten Aufwand beim Schreiben des Quellcodes, erhöhte Flexibilität oder geringe Komplexität aus ([Wä12] f.). Diese Programme können sowohl in Kombination mit der Programmiersprache Java als auch selbstständig eingesetzt werden. Eine Integration der JVM-Sprachen ist sehr einfach, da beim Kompilieren des Quellcodes Java-Byte-Code erzeugt wird, der dann in der JVM ausgeführt wird. Die Liste dieser JVM-Sprachen ist relativ lang, bekannteste Vertreter sind beispielsweise Scala, Python, JavaScript oder Apache Groovy. Die Einbindung der verschiedenen Sprachen in ein Java-Programm variiert

leicht, ist allgemein aber einfach durchzuführen.

5.3.2. JNI

Mit Hilfe von JNI wird eine Schnittstelle zwischen Java und den Programmiersprachen C und C++ gebildet. Zunächst muss der Code der anderen Programmiersprache zu einer dynamisch ladbaren Bibliothek gebunden werden, welche die nativen Methoden implementiert [Ull12]. Die Bibliothek kann dann in einer Java-Anwendung geladen werden und Funktionsaufrufe werden dann an die entsprechende native Implementierung weitergeleitet [Ull12]. Der Entwicklungsprozess sieht im Allgemeinen folgendermaßen aus [Hä00]:

1. Schreiben einer Java-Klasse, die entsprechende Methoden als nativ deklariert
2. Compilieren dieser Java Klasse und Generieren einer C-kompatiblen Header-Datei
3. Implementierung der nativen Methoden in C/C++ und Compilieren der Bibliothek

5.3.3. MATLAB

Um MATLAB-Funktionen aus einer Java-Anwendung heraus auszuführen, kann die Open-Source Java-Bibliothek Java Matlab Linking (JAMAL) genutzt werden. JAMAL besteht aus zwei Teilen [mat12]:

1. **RMI-Server**: Läuft auf der MATLAB-Seite, empfängt Befehle des Client, führt die entsprechenden MATLAB-Funktionen aus und liefert die Ergebnisse zurück.
2. **RMI-Client**: Läuft in der Java-Anwendung, ruft Funktionen des RMI-Server mit entsprechenden Parametern auf und erhält die Ergebnisse zurück.

RMI steht für Remote Method Invocation und bedeutet, dass ein Zugriff auf eine Methode eines Java-Objekts erfolgt, das sich nicht auf der selben JVM wie die Applikation befindet.

5.4. Dateiformatspezifikationen

Entsprechend der Anforderungen im Abschnitt 4.2 muss die Anwendung mit verschiedenen Dateiformaten arbeiten. Dazu zählen Dateien

- die von der Anwendung importiert werden müssen,

- in denen Daten durch die Anwendung gespeichert werden und aus denen die Anwendung diese Daten auch wieder ausliest und
- in die Daten exportiert werden können.

Im Folgenden werden diese verschiedenen Dateiformate beschrieben.

5.4.1. Importdateiformate

Die Anwendung muss in der Lage sein, die von der „Humotion DataLogger“-Anwendung generierten Dateien, welche die Daten des Sensorgürtels enthalten, zu importieren. Wie im Unterabschnitt 3.2.2 beschrieben, enthält der Sensorgürtel ein Akzelerometer, Gyroskop, Magnetometer und einen Luftdrucksensor. Diese Sensoren werden in verschiedene Datenreihen gegliedert, die folgendermaßen benannt sind:

- Time-Date: Zeitstempel der jeweiligen Aufnahme
- Acc-X: Beschleunigung in Richtung der X-Achse
- Acc-Y: Beschleunigung in Richtung der Y-Achse
- Acc-Z: Beschleunigung in Richtung der Z-Achse
- Acc-Temp: Temperaturmessung des Thermometers der Akzelerometer-Komponente
- Gyro-X: Drehrate um die X-Achse
- Gyro-Y: Drehrate um die Y-Achse
- Gyro-Z: Drehrate um die Z-Achse
- Mag-X: Magnetische Flussdichte in Richtung der X-Achse
- Mag-Y: Magnetische Flussdichte in Richtung der Y-Achse
- Mag-Z: Magnetische Flussdichte in Richtung der Z-Achse
- Pressure-P: Luftdruck
- Pressure-T: Temperaturmessung des Thermometers der Luftdrucksensor-Komponente
- Power-Voltage: Vom Sensorgürtel bezogene Spannung

Der Sensorgürtel liefert verschiedene Dateien, von denen zwei Textdateien von Relevanz sind:

- eine .txt-Datei, die die Sensordaten zu den einzelnen Sensorreihen enthält und

- eine .ini-Datei, die Metadaten, wie z. B. den Aufnahmezeitraum oder die Einheit einer Sensorreihe, enthält.

Der Aufbau dieser Textdateien und die für dieses Projekt relevanten Informationen werden in den folgenden Unterkapiteln beschrieben. Da beide Dateien zeitbezogene Informationen speichern, muss beachtet werden, dass die Aufnahme in dem Moment beginnt, wenn der Sensorgürtel initialisiert wird. Es kann aber durchaus sein, dass diese Vorbereitung einige Tage vor der eigentlichen Messung durchgeführt wird. Umgekehrt werden die Daten nach einer Messung nicht immer direkt ausgelesen, sondern vielleicht erst Tage später. In der Zwischenzeit wird der Gürtel aufgeladen und misst in dieser Zeit keine Daten. Die Zeit läuft aber trotzdem voran, jedoch werden erst wieder Daten gemessen und gespeichert, wenn der Gürtel vom Ladegerät entfernt wird. Dadurch kann es zu Zeitsprüngen innerhalb der Aufnahme kommen. Durch diese Zeitsprünge kann die Messreihe in mehrere zusammenhängende Segmente geteilt werden, von denen mindestens eines eine Messung beschreibt. Handelt es sich um eine Assessmentaufnahme, so wird es wahrscheinlich nur ein Segment mit einer Messung geben. Stattdessen wird es bei einer Alltagsmessung für jeden Tag ein Segment geben.

Sensordatendateiformat

Die .txt-Datei, die die Sensordaten enthält, folgt einem einfachen, strikten Aufbau. Die einzelnen Werte einer Aufnahme sind durch ein Trennsymbol getrennt, genauer gesagt durch einen Tabulator. Sie entsprechen dadurch einer komma-separierten Textdatei, wobei statt einem Komma zur Trennung der Werte ein Tabulator genutzt wird. Man kann diese Datei daher auch als eine Tabelle betrachten, sodass im Folgenden auch von Zeilen und Spalten gesprochen wird.

Die erste Zeile der .txt-Datei ist ein Header, in dem die Namen der einzelnen Sensorreihen beschrieben sind. Diese Namen entsprechen den in der Aufzählung in Unterabschnitt 5.4.1 verwendeten Begriffen, sind aber zusätzlich in Anführungszeichen gesetzt. Für eine korrekte und vollständige Funktionsweise der Anwendung wird vorausgesetzt, dass solch eine .txt-Datei für alle 14 Sensorreihen Daten enthält. Die Header-Zeile muss also dementsprechend folgendermaßen aussehen:

„Time-Date “ „Acc-X“ „Acc-Y“ „Acc-Z“ „Acc-Tmp“ „Gyro-X“ „Gyro-Y“ „Gyro-Z“ „Mag-X“ „Mag-Y“ „Mag-Z“ „Pressure-P“ „Pressure-T“ „Power-Voltage“

Ab der zweiten Zeile beginnen die Sensorwerte, wobei jede Spalte der entsprechenden Sensorreihe zugeordnet wird. Die Anzahl an Zeilen ist theoretisch beliebig groß, wird aber durch den Aufnahmezeitraum und die Aufnahmerate begrenzt. Vor- und Nachkommastellen der Sensorwerte sind durch ein Komma getrennt, wie es in Deutschland üblich ist. Das muss besonders beachtet werden, da Programmiersprachen üblicherweise einen Punkt zur Trennung von Ganzzahl und Bruchteil voraussetzen. Dies entspricht der Schreibweise aus dem englischsprachigen Raum.

Die Werte der Spalte „Time-Date“ repräsentieren Zeitstempel. Je nach Einstellung des „Humotion Datalogger“ können diese Werte unterschiedlich aussehen. Eine Möglichkeit ist, dass die Werte bei null starten und dann in Millisekunden hochgezählt werden (je nach Abtastrate des Sensorgürtels). Dabei kann es vorkommen, dass gebrochene Millisekunden mit Nachkommastellen auftreten. Diese repräsentieren dann Mikrosekunden. In Verbindung mit dem Aufnahmezeitraum aus der Metadaten-Datei lassen sich aus diesen Werten die konkreten Zeitpunkte berechnen. Eine andere Möglichkeit ist, dass die Werte Unix-Zeitstempel in Dezimalform kodieren. Unix zählt die Sekunden ab dem 1. Januar 1970, 00:00 UTC. In diesem Fall werden dann die Millisekunden im Nachkommateil gespeichert.

Die Werte zu den restlichen Sensoren sind absolut und können sowohl positive als auch negative Werte annehmen. Wie allgemein üblich wird bei positiven Zahlen das vorangestellte Pluszeichen nicht mitgeschrieben, während negative Zahlen durch ein vorangestelltes Minuszeichen gekennzeichnet werden.

Metadatendateiformat

Auch die .ini-Datei, die die Metainformationen zu einer Aufnahme enthält, folgt einem einfachen, strikten Aufbau. Es gibt verschiedene, benannte Sektionen, die jeweils ihre speziellen Attribute beinhalten.

Die Textdatei hat folgende allgemeine Eigenschaften:

- Jede Zeile der Datei enthält nur eine Information, also entweder den Namen der Sektion oder ein Attribut mit zugehörigem Wert. Es gibt aber auch Leerzeilen sowohl zu Beginn beziehungsweise Ende der Datei als auch zwischen den verschiedenen Sektionen.
- Die Namen der Sektionen sind jeweils in eckige Klammern gesetzt.

- Ein Attribut beginnt zunächst mit einem Namen, gefolgt von einem Gleichheitszeichen und seinem Wert. Dabei gibt es keine Trennung durch Leerräume, wie z. B. ein Leerzeichen. Die einzige Ausnahme ist das Attribut „Freqs“ in der Sektion „SuperFrame.1“, dass keinen zugewiesenen Wert hat.

Im Folgenden ist eine Auflistung der verschiedenen Sektionen zu sehen, bei der auch beschrieben wird, welche Informationen für dieses Projekt relevant sind.

- Ini: Enthält keine für dieses Projekt relevanten Informationen.
- Channels: Das Attribut Count beschreibt die Anzahl an Messreihen. Es sollte daher immer den Wert 14 haben.
- Compression: Das Attribut RecordsPerSecond beschreibt die Aufnahmerate in Hertz.
- SuperFrame.1: Enthält keine für dieses Projekt relevanten Informationen.
- Chan.1 bis Chan.14: Enthalten Informationen zu den verschiedenen Messreihen. Die Attribute Type und ChanName beschreiben dabei den Namen der Messreihe.
  - Chan.1: Beschreibt die Zeitmessung. Kann zusätzlich noch das Attribut IsUnixTimeStamp enthalten, das mit einem Wert von eins angibt, dass die Zeitstempel der Sensordatendatei im Unix-Format kodiert sind. Exisitert das Attribut nicht, oder es hat den Wert null, dann beginnen die Zeitstempel bei null.
  - Chan.2 bis Chan.14: Beschreiben die Sensorreihen. Enthalten zusätzlich noch das Attribut Unit, das die Einheit der Messreihe beschreibt.
- Measure: Enthält Informationen über den Aufnahmezeitraum. Die Attribute Start und Stop beschreiben den Start- und Endzeitpunkt der Aufnahme. Das Format ist dabei durch die allgemeine Form YYYY-MM-DDT:hh:mm:ss:fff gegeben, wobei das „T“ zur Trennung von Datum und Uhrzeit dient und „fff“ den Sekundenbruchteil beschreibt. Bei diesen Werten ist aber zu beachten, dass der Start- und der Endzeitpunkt nicht zwangsweise die wirkliche Messung beschreiben (siehe Unterabschnitt 5.4.1).

5.4.2. Lese- und Schreibdateiformat

In Unterabschnitt 5.4.1 wurde bereits beschrieben, welche Dateiformate von der Anwendung importiert werden müssen. Für dieses Projekt wurde aber entschieden, dass für die Speicherung der Daten andere Dateiformate genutzt werden sollen. Dadurch soll Spei-

cherplatz eingespart und eine bessere Performanz beim Einlesen und Zurückschreiben der Daten erreicht werden. In Abschnitt 5.2.3 ist eine Evaluation von verschiedenen Dateiformaten beschrieben, die als Ergebnis eine Verwendung von Binärdateien zur Speicherung der Rohdaten nahelegt. Es muss daher eine Spezifikation dieses Binärformats erfolgen. Zusätzlich müssen auch Dateiformate zur Speicherung der Metainformationen zu einer Datenreihe, sowie zur Speicherung von Labelinformationen definiert werden. All diese Dateien müssen durch eine Verzeichnisstruktur logisch gruppiert werden, sodass ein gewünschter Datensatz schnell und einfach gefunden werden kann.

Verzeichnisstruktur

Der Benutzer hat die Möglichkeit ein Verzeichnis anzugeben, in dem die Anwendung all ihre Dateien verwaltet. Der Aufbau des Arbeitsverzeichnisses ist beispielhaft in Listing 5.1 dargestellt. Für jeden Probanden wird ein Ordner angelegt, dessen Name der Probanden-ID entspricht. In diesem Verzeichnis befinden sich wiederum Verzeichnisse, die jeweils einen Datensatz dieses Probanden repäsentieren. Die Namen entsprechen den Startzeitpunkten in ISO 8601.

Jedes Datensatzverzeichnis enthält die Datei metainfo.xml, welche Metainformationen über den Datensatz, wie den Aufnahmezeitraum, speichert. Sonst befinden sich im Verzeichnis nur weitere Ordner, von denen zwei eine besondere Bedeutung haben.

Im Verzeichnis sensorbelt befinden sich die Datenreihen, welche aus dem Originaldatensatz importiert wurden. Der Dateiname einer Datenreihe setzt sich aus dem Präfix data- und dem Namen der Datenreihe zusammen. Die Dateiendung für solche Binärdateien ist .bin. Beispielsweise ist in der Datei data-Mag-X.bin eine Datenreihe namens Mag-X (Magnetometer, X-Achse) gespeichert. Die Namen der Sensordatenreihen entsprechen denen des Originaldatensatzes, der importiert wurde. Weiterhin befindet sich in diesem Ordner die Datei label-reference.xml, welche Referenzlabel enthält, die beim Import aus zusätzlichen Informationen wie Lichtschrankendaten gewonnen wurden. Die Daten im sensorbelt-Verzeichnis sind für den Benutzer aus der Anwendung heraus nicht bearbeitbar und dienen nur als Datengrundlage.

Das Verzeichnis userdata enthält nur Labelgruppen-Dateien. Jede Labelgruppe kann beliebig viele Label enthalten. Wenn der Benutzer in der Anwendung manuell Label anlegt, werden diese im userdata-Verzeichnis gespeichert. Der Name der Datei ergibt

sich dabei durch die Konkatenation des Präfix labels- und dem Labelgruppenname. Heißt die Labelgruppe beispielsweise MOTIONS so heißt die erzeugte, zugehörige Datei label-MOTIONS.xml.

Schließlich kann das Datensatzverzeichnis beliebig viele weitere Verzeichnisse enthalten, wie z. B. algorithmXYZ-1489672649377-0. Solche Verzeichnisse werden durch die Ausführung von Algorithmen angelegt und enthalten die Algorithmusergebnisse in Form von Datenreihen und Label-Informationen. Der Name der Verzeichnisse ergibt sich als die Konkatenation der Algorithmus-ID, der Ausführungszeit als Unix-Epoch-Zeitstempel und einer Ausführungsnummer. Die Ausführungsnummer wird relevant, wenn ein Algorithmus mit einer Kombination von Parametern ausgeführt wird, da dann für jede Ausführung ein Verzeichnis angelegt wird. Für Algorithmen, die keine Datenreihen oder Label-Informationen erzeugen, wird kein Verzeichnis angelegt. Ein Algorithmusverzeichnis kann wie das sensorbelt-Verzeichnis Datenreihen und wie das userdata-Verzeichnis Label-Informationen enthalten, zusätzlich enthält ein Algorithmusverzeichnis eine algorithm-execution-information.xml in der die Ausführungsinformationen des Algorithmus gespeichert sind.

Listing 5.1: Beispielhafte Verzeichnisstruktur

```
working-directory/
| -- 10534/
|    | -- 2015-12-21T15-23-10-129Z/
|    |    | -- sensorbelt/
|    |    |    | -- data-Mag-X.bin
|    |    |    | -- data-Acc-Z.bin
|    |    |    | ...
|    |    |    | -- label-reference.xml
|    |    |
|    |    | -- userdata/
|    |    |    | -- label-abc.xml
|    |    |    | -- label-def.xml
|    |    |
|    |    | -- algorithmXYZ-1489672649377-0/
|    |    |    | -- data-Acc-X-filtered.bin
|    |    |    | -- label-classification-output.xml
|    |    |    | -- algorithm-execution-information.xml
|    |    |
|    |    | -- metainfo.xml
|    |
|    | -- 2016-02-13T05-23-10-100Z/
|         | -- ...
|
| -- 10666/
```

25 | — ...

Binärformat

Die Anwendung muss Binärformate für die Speicherung der verschiedenen Sensorreihen unterstützen.

Wie in Unterabschnitt 5.4.1 beschrieben, speichert der Sensorgürtel die Sensorwerte als vorzeichenbehaftete Dezimalzahlen. Die Anwendung wird zur Speicherung dieser Daten den Datentyp „float“ verwenden (siehe Abschnitt 5.2.3). Dies entspricht einer 32-Bit Gleitkommazahl nach dem IEEE-754-Standard [IEE85]. Dabei soll für jede Sensorreihe eine einzelne Binärdatei erzeugt werden, die die zugehörigen Werte enthält. Die Namen der Dateien entsprechen den jeweiligen Sensorreihen.

Somit kann die Wertereihe eines Sensors als Sequenz vom Datentyp „float“ abgespeichert werden. In der Binärdatei entsprechen dann alle vier Bytes einem Sensorwert und können dementsprechend wieder eingelesen und zu einem „float“ zusammengesetzt werden. Die Anzahl an gespeicherten Werten in einer solchen Float-Binärdatei lässt sich durch die Division der Dateigröße durch vier Bytes berechnen.

XML-Format

Um die verschiedenen Metainformationen abspeichern zu können, die zu einer Aufnahme des Sensorgürtels gehören, muss ein geeignetes Dateiformat gefunden werden. Dasselbe gilt auch für die Speicherung der Labelinformationen oder der Informationen zu einer Algorithmusausführung. Anstatt ein komplett neues Format zu erfinden, wird die bewährte Auszeichnungssprache XML verwendet, mit der die Daten hierarchisch gespeichert werden können. Die hierarchische Form eines XML-Dokuments kann auch als eine Baumstruktur angesehen werden. Es gibt eine Wurzel, in der alle weiteren Elemente enthalten sind. Ein Element ist durch ein Tag gekennzeichnet und kann weitere Elemente beinhalten. Dies ist lediglich eine grobe Darstellung und für tiefere Einblicke sei auf die XML-Spezifikation des World Wide Web Consortium (W3C) verwiesen [W3C08].

Um ein anwendungsspezifisches XML-Format zu definieren, empfiehlt das W3C die Verwendung eines XML Schema (XSD). Dies ist selbst ein XML-Dokument, das die Struktur des XML-Formats beschreibt und Einschränkungen vornehmen kann. Der große Vorteil

bei Verwendung von XSD ist, dass ein XML-Parser prüfen kann, ob ein XML-Dokument die Spezifikation des XSD einhält. Daher werden im Folgenden die Spezifikationen der benötigten XML-Formate mittels XSD vorgenommen und erläutert.

Metainformationen Das XSD für Metainformationen ist in Listing 5.2 zu sehen. Es verfügt über ein Element metaInformation des komplexen Typs MetaInformationType. Der Typ MetaInformationType verfügt über das Element timeInterval des komplexen Typs TimeIntervalType, welches das Zeitintervall des beschriebenen Datensatzes angibt, und über das Element recordCount, welches die Anzahl an Aufnahmen innerhalb des Datensatzes angibt. Der Typ TimeIntervalType verfügt über ein Element start in welchem die Startzeit und ein Element end in welchem die Endzeit angegeben wird. Die Zeit wird nach ISO 8601 mit Nanosekundengenauigkeit angegeben.

So werden alle relevanten Informationen gespeichert, die für die gesamte Messung gelten. Informationen wie z. B. die Einheit einer Datenreihe des Sensorgürtels werden nicht gespeichert, da diese als unveränderlich vorausgesetzt werden. Beim Einlesen der Sensorgürteldateien wird trotzdem validiert, ob die zu importierenden Dateien das erwartete Format erfüllen.

Listing 5.2: XSD für Metainformationen

```
1  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
2  <xsd:annotation>
3    <xsd:documentation xml:lang="en">
4      Schema for the meta information of motion data.
5    </xsd:documentation>
6  </xsd:annotation>
7
8  <xsd:element name="metaInformation" type="MetaInformationType"/>
9
10 <xsd:complexType name="MetaInformationType">
11   <xsd:sequence>
12     <xsd:element name="timeInterval" type="TimeIntervalType"/>
13     <xsd:element name="recordCount" type="xsd:integer"/>
14   </xsd:sequence>
15 </xsd:complexType>
16
17 <xsd:complexType name="TimeIntervalType">
18   <xsd:sequence>
19     <xsd:element name="start" type="xsd:string"/>
20     <xsd:element name="end" type="xsd:string"/>
21   </xsd:sequence>
22 </xsd:complexType>
23
```

```
24 </xsd:schema>
```

Labelinformationen Das XSD für Labelinformationen ist in Listing 5.3 zu sehen. Das Schema enthält ein Element labelInformation des komplexen Typs LabelInformationType. Dieser komplexe Typ wiederum kann eine unbegrenzte Anzahl an Elementen label des komplexen Typs LabelType enthalten. Jedes dieser label enthält ein Element description in welchem der Labelname als Text gespeichert wird und ein Element interval vom komplexen Typ IntervalType. IntervalType enthält ein Element start, welches den Startindex des Labels angibt, und ein Element end, welches den Endindex des Labels angibt.

Sowohl im userdata- als auch in jedem Algorithmus-Verzeichnis können beliebig viele Labelinformationsdateien angelegt werden, jede Datei beschreibt dann die Label der entsprechenden Labelgruppe. Die Dateien müssen sich namentlich unterscheiden.

Listing 5.3: XSD für Labelinformationen

```
1  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
2  <xsd:annotation>
3    <xsd:documentation xml:lang="en">
4      Schema for the label information of motion data.
5    </xsd:documentation>
6  </xsd:annotation>
7
8  <xsd:element name="labelInformation" type="LabelInformationType"/>
9
10 <xsd:complexType name="LabelInformationType">
11   <xsd:sequence>
12     <xsd:element name="label" type="LabelType" minOccurs="1" maxOccurs="unbounded"/>
13   </xsd:sequence>
14 </xsd:complexType>
15
16 <xsd:complexType name="IntervalType">
17   <xsd:sequence>
18     <xsd:element name="start" type="xsd:long"/>
19     <xsd:element name="end" type="xsd:long"/>
20   </xsd:sequence>
21 </xsd:complexType>
22
23 <xsd:complexType name="LabelType">
24   <xsd:sequence>
25     <xsd:element name="interval" type="IntervalType"/>
26     <xsd:element name="description" type="xsd:string"/>
27   </xsd:sequence>
28 </xsd:complexType>
29
30 </xsd:schema>
```

Algorithmusausführung Das XSD für Informationen über eine Algorithmusausführung ist in Listing 5.4 zu sehen. Das Schema enthält ein Element algorithmExecutionInformation des komplexen Typs AlgorithmExecutionInformationType. Ein solcher Typ enthält ein Element algorithmName in dem der Name des Algorithmus gespeichert wird, eine beliebige Anzahl an Elementen dataSeriesInfo des komplexen Typs DataSeriesInfoType, die jeweils Informationen zu einer durch den Algorithmus erzeugten Datenreihe enthalten, eine beliebige Anzahl an Elementen inputParameter des komplexen Typs ParameterType, welche jeweils Informationen zu einem Eingabeparameter des Algorithmus enthalten, und eine beliebige Anzahl an Elementen outputParamter des komplexen Typs ParameterType, die jeweils Informationen zu einem Ausgabeparameter des Algorithmus enthalten.

Der komplexe Typ DataSeriesInfoType verfügt über ein Element name und ein Element unit, in denen der Name und die Einheit der Datenreihe abgelegt werden.

Der komplexe Typ ParameterType verfügt über ein Element name und ein Element value, in denen der Name und der Wert des Parameters abgelegt werden.

Listing 5.4: XSD für Informationen zu einer Algorithmusausführung

```
1  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
2  <xsd:annotation>
3    <xsd:documentation xml:lang="en">
4      Schema for the information about an algorithm execution.
5      It stores information about the created DataSeries as well as input and output
           parameter values.
6    </xsd:documentation>
7  </xsd:annotation>
8
9  <xsd:element name="algorithmExecutionInformation" type="
       AlgorithmExecutionInformationType"/>
10
11 <xsd:complexType name="AlgorithmExecutionInformationType">
12   <xsd:sequence>
13     <xsd:element name="algorithmName" type="xsd:string"/>
14     <xsd:element name="dataSeriesInfo" type="DataSeriesInfoType" minOccurs="0" maxOccurs
           ="unbounded"/>
15     <xsd:element name="inputParameter" type="ParameterType" minOccurs="0" maxOccurs="
           unbounded"/>
16     <xsd:element name="outputParameter" type="ParameterType" minOccurs="0" maxOccurs="
           unbounded"/>
17   </xsd:sequence>
18 </xsd:complexType>
19
20 <xsd:complexType name="ParameterType">
21   <xsd:sequence>
22     <xsd:element name="name" type="xsd:string"/>
```

```
23      <xsd:element name="value" type="xsd:string"/>
24    </xsd:sequence>
25  </xsd:complexType>
26
27  <xsd:complexType name="DataSeriesInfoType">
28    <xsd:sequence>
29      <xsd:element name="name" type="xsd:string"/>
30      <xsd:element name="unit" type="xsd:string"/>
31    </xsd:sequence>
32  </xsd:complexType>
33
34  </xsd:schema>
```

5.4.3. Exportdateiformat

Die Anwendung soll Daten in ein Format exportieren können, das leicht lesbar ist und von gängigen Tabellenkalkulationsprogrammen unterstützt wird. Hierfür eignet sich das CSV-Format (Comma-separated values), das Informationen als Textdatei abspeichert. Der Aufbau ähnelt dabei einer Tabelle, wobei Spalten durch ein spezielles Zeichen voneinander getrennt werden. Wie der Name schon sagt, ist ein Komma zur Trennung üblich, doch es können auch andere Zeichen wie das Semikolon oder ein Leerraum verwendet werden. Ein Komma ist leicht problematisch, da im deutschsprachigen Raum das Komma zur Trennung von Vor- und Nachkommastellen bei Dezimalzahlen verwendet wird. Um dieses Problem zu vermeiden und um eine Konsistenz mit den CSV-Dateien zu schaffen, die vom Sensorgürtel generiert werden (siehe Abschnitt 5.4.1), werden Tabulatoren zur Trennung der Spalten verwendet.

Es muss zwischen zwei Exportformaten unterschieden werden, die in den folgenden Abschnitten beschrieben sind.

Export der Datenreihen

Die erste Zeile der CSV-Datei ist ein Header, der den verschiedenen Spalten einen Namen mit ihrer Maßeinheit zuordnet. In den darauf folgenden Zeilen werden dann die Sensorwerte den entsprechenden Spalten zugeordnet. Die Zeitpunkte werden indexbasiert und durch das entsprechende menschenlesbare UTC-Datumsformat mit Nanosekundengenauigkeit dargestellt.

Export der Label-Informationen

Der Export der Label-Informationen enthält alle Label-Informationen aller Label-Gruppen des Bewegungsdatensatzes. Die exportierte CSV-Datei besitzt folgende Spalten:

Data set enthält den Identifikationsstring des Datensatzes, zu dem das Label gehört.

group enthält den Identifikationsstring der Label-Gruppe, zu der das Label gehört.

name enthält den Namen des Labels.

Interval-Start [Index] enthält den Startindex der Label-Markierung.

Interval-End [Index] enthält den Endindex der Label-Markierung.

Interval-Start [UTC] enthält den Startzeitpunkt der Label-Markierung als UTC-Datumsformat mit Nanosekundengenauigkeit.

Interval-End [UTC] enthält den Endtzeitpunkt der Label-Markierung als UTC-Datumsformat mit Nanosekundengenauigkeit.

5.5. Generierung von Referenzinformationen

Die während der Assessments aufgenommenen Daten, die der PGMAmKS zur Verfügung stehen, enthalten neben den Daten des Sensorgürtels weitere Daten, wie z. B. Daten die vom Ambient Timed Up & Go (aTUG)-Stuhl aufgenommen wurden, Ereigniseinträge mit Zeitstempelangaben, von Schaltern, die vor bestimmten Assessments betätigt wurden, und von Lichtschranken, die während der Assessments ausgelöst wurden. Darauf, wie diese Daten im Rahmen der PGMAmKS genutzt wurden, welche Kenntnisse diesbezüglich gewonnen und welche daraus resultierenden Entscheidungen bezüglich der Nutzung dieser Daten getroffen wurden, soll in diesem Kapitel eingegangen werden.

Darüber hinaus soll in diesem Kapitel darauf eingegangen werden, wie durch manuelles Beschriften der Daten zusätzliche Informationen gewonnen wurden, wie diese genutzt wurden und welche Bedeutung sie für den Prozess der Bewegungsmuster-Erkennung haben.

5.5.1. Lichtschranken

Einige der Übungen, die während eines Assessment durchgeführt werden, werden durch Lichtschranken überwacht. Dazu zählen das Gehen von $4.57\,m$ beim Frailty-Test, der $4.00\,m$ Gehtest, sowie der Stair-Climb-Power-Test und der 6-Minuten-Gehtest. Immer wenn eine Lichtschranke auslöst wird ein Eintrag in eine CSV-Datei geschrieben, der einen Zeitstempel sowie weitere Informationen enthält. In der Theorie sollten aus diesen Daten dann einfach entsprechende Label für Gehen oder Treppensteigen generiert werden.

Ein Blick in die Lichtschrankendaten verschiedener Assessments hat aber gezeigt, dass die Lichtschrankendaten nicht sehr zuverlässig sind. Des öfteren werden Fehlermeldungen geloggt oder aber es lösen Lichtschranken in einer Reihenfolge aus, die eigentlich nicht möglich sein sollte. Dies ließe sich beispielsweise durch das Durchschreiten der Lichtschranken von anderen Personen als dem Probanden erklären. Da die zu generierenden Labels aber als 100% vertrauenswürdige Referenzen dienen sollen, dürfen solche fehlerbehafteten Daten keine Verwendung finden.

Dementsprechend wurde ein robuster Algorithmus entwickelt, der diese Gefahren berücksichtigt und vertrauenswürdige Referenzinformationen liefert. Die Anwendung dieses Algorithmus auf verschiedene Datensätze liefert aber nur eine geringe Anzahl an Labeln, die von einem etwas geübten Menschen auch in wenigen Sekunden manuell erstellt werden könnten.

5.5.2. Weitere Zusatzdaten

Die Entwicklung eines Algorithmus zur Extraktion von Referenzinformationen aus den zusätzlichen Daten, die während der verschiedenen Übungen aufgezeichnet werden, hat sich am Beispiel der Lichtschrankendaten als eher schwierig erwiesen. Für die Entwicklung eines robusten und korrekten Algorithmus, der zuverlässige Referenzinformationen generiert, muss viel Zeit aufgewendet werden. Und selbst dann liefert das Verfahren keinen wirklichen Mehrwert.

Aus diesen Gründen wurde die Entscheidung getroffen, dass von der Implementierung weiterer Algorithmen abgesehen wird, die weitere Zusatzdaten verwenden. Ein geübter Mensch kann mit der in diesem Projekt entwickelten Anwendung schnell und einfach die entsprechenden Zeitintervalle manuell beschriften.

5.5.3. Manuelle Beschriftung

Als „Manuelle Beschriftung“ bezeichnet die PGMAmKS ein Verfahren, bei dem der Nutzer einem zeitlichen Intervall innerhalb des Aufnahmezeitraums eine Beschriftung zuweist. Der Nutzer wählt bei der manuellen Beschriftung die Intervalle dabei so, dass sie ein optisch deutlich erkennbares Muster in den Sensordaten umschließen. Folglich bedarf der Nutzer einer gewissen Fachkenntnis über die zu beschriftenden Muster, deren Aussehen und muss die vom Sensorgürtel aufgezeichneten Daten zu einem gewissen Rahmen interpretieren können. In Abschnitt 3.3 wurde bereits ein ausführlicher Überblick über die relevanten Muster und über das Vorgehen bei der Beschriftung gegeben.

Wie in Unterabschnitt 5.5.1 und Unterabschnitt 5.5.2 beschrieben, ist eine automatische Gewinnung von Referenzinformationen kaum bzw. nur sehr eingeschränkt möglich, somit ist die „Manuelle Beschriftung“ das einzige Verfahren, mit dem Referenzinformationen gewonnen werden können.

Die durch die „Manuelle Beschriftung“ der Daten gewonnenen Referenzinformationen sind für die Aktivitätserkennung der vom Sensorgürtel aufgenommenen Daten von entscheidender Bedeutung, da sie genutzt werden, um die Klassifikatoren zu trainieren. Wie die gewonnen Referenzinformationen für die Aktivitätserkennung mithilfe von Techniken des maschinellen Lernens genutzt werden, wird in Abschnitt 5.8 erläutert.

Die Prozessschritte der Aktivitätserkennung wurden bereits in Abschnitt 3.4 detailliert beschrieben. Zusammengefasst werden auf den vom Sensorgürtel aufgenommenen Rohdaten per Schiebefenster Merkmalsvektoren berechnet, denen durch einen Klassifikator Zielklassen (in unserem Fall Beschriftungen) zugeordnet werden. Damit ein Klassifikator dies leisten kann, muss er vorher während der sogenannten Trainingsphase trainiert werden. Während der Trainingphase wird mithilfe der manuell beschrifteten Intervalle, auch Trainingsbeispiele genannt, ein Modell erzeugt, das anschließend neuen Intervalle automatisch Beschriftungen zuordnen kann. Dabei werden die Parameter des Modells so angepasst, dass das Modell einem zeitlichen Intervall (repräsentiert durch seinen Merkmalsvektor) möglichst eine korrekte Beschriftung zuordnen kann.

Zu Beginn der Projektgruppe haben zwei Projektgruppenmitglieder jeweils ein Assessment unter Anweisung des medizinischen Personals, das auch die anderen Assessments begleitet hat, durchgeführt. Jeder Schritt des Ablaufs wurde sowohl schriftlich, als auch auf Video dokumentiert. Ausgehend von dem so gewonnen Eindruck über die Assessmentdurch-

führung und unter Zuhilfenahme weiterer Informationsgrundlagen, wie die Assessment- und Befragungsbögen (siehe Abschnitt B) wurde eine der beiden Aufnahmen manuell beschriftet (die Aufnahme des anderen Assessments war beschädigt) und ein Assessment-Ablaufplan (siehe Abschnitt A) erstellt. Damit wurden ausreichende Hilfsmittel gegeben, um die Aufnahmen der Probanden-Assessments manuell zu beschriften.

Die Projektgruppe hat insgesamt 38 Aufnahmen von Probanden-Assessments beschriftet. Die Aufnahmedauer aller 38 Aufnahmen beträgt insgesamt 42 Stunden, 5 Minuten und 50,48 Sekunden. Diesen wurden sowohl allgemeine Bewegungsbeschriftungen als auch Assessmentbeschriftungen hinzugefügt. Insgesamt wurden 22,66% der Gesamtaufnahme (9 Stunden, 35 Minuten, 15,61 Sekunden) als Assessments beschriftet , 41,34% (17 Stunden, 24 Minuten, 9,21 Sekunden) wurden als statische Bewegung beschriftet , 12,04% (5 Stunden, 3 Minuten, 59,69 Sekunden) wurden als dynamische Bewegung beschriftet und 1,62% (40 Minuten, 48,12 Sekunden) wurden als Transitionsbewegung beschriftet.

An diesen Zahlen wird ersichtlich, dass die Beschriftungen deutlich ungleich verteilt sind. Beispielsweise sind 25-mal mehr Daten mit statischen Bewegung, als mit Transitionsbewegungen beschriftet worden. Dies ist problematisch, da die Trainingsbeispiele der Klassifikatoren idealerweise gleichverteilt sein sollten, um Überanpassungen an bestimmte Muster zu vermeiden.

Da sich das Ungleichgewicht in den Daten nur bedingt durch Oversampling bzw. Undersampling (siehe Unterabschnitt 3.4.4) korrigieren lässt, wurden weitere Daten aufgenommen und beschriftet. Als Probanden dieser Aufnahmen haben die Mitglieder der PGMAmKS gedient. Vorab wurde ein Testdurchlauf durchgeführt, um zu verifizieren, ob die PGMAmKS-Mitglieder als Probanden für einen solchen Zweck überhaupt geeignet sind. Es bestand nämlich die Befürchtung, dass die Mitglieder aufgrund deutlicher Alters-, als auch Fitnessunterschiede im Vergleich zu den Geriatrie-Probanden andere Muster erzeugen. Der Durchlauf hat allerdings gezeigt, dass die aufgenommenen Zusatzdaten des Testdurchlaufs das Klassifikationsergebnis verbessern, weshalb mit der Aufnahme von Zusatzdaten fortgefahren wurde. Es wurden dabei fast ausschließlich dynamische Bewegungen und Transitionsbewegungen aufgenommen um die Ungleichverteilung auszugleichen. Insgesamt wurden 31 Zusatzdaten aufgenommen, mit einer Gesamtaufnahmelänge von ca. 31 Stunden. Von den Zusatzdaten wurden 0,45% (ca. 8 Minuten) als statische Bewegung (Liegen auf der Seite, Liegen auf dem Rücken), 10% (1 Stunde, 45 Minuten, 8,98 Sekunden) als dynamische Bewegung (Treppesteigen, Springen) und 17,19% (5 Stunden,

19 Minuten, 54,72 Sekunden) als Transitionsbewegung (Aufstehen, Hinsetzen) beschriftet.

Ohne die grafische Benutzeroberfläche und die dahinter liegende Anwendung, welche im Rahmen dieser Projektgruppe entwickelt wurde, wäre dieser Umfang an manueller Datenbeschriftung kaum möglich, bzw. weit schwieriger gewesen, da die entwickelte Anwendung speziell für diesem Zweck optimiert wurde.

5.6. Grafische Benutzeroberfläche

Gemäß den formulierten Anforderungen an das System in Abschnitt 4.2, wurde eine GUI entwickelt, die Funktionen zum Betrachten und Bearbeiten von Datensätzen bietet und die Ausführung von Algorithmen ermöglicht. Sie wurde vor Allem für eine einfache und schnelle Erzeugung von Referenzinformationen mittels der im vorigen Abschnitt beschriebenen „Manuelle Beschriftung“ realisiert, ihre Umsetzung ist allerdings aus verschiedenen weiteren Gründen sinnvoll. Einerseits sind Fitnesstrainer und medizinisches Personal die Hauptzielgruppe dieser Anwendung (siehe Abschnitt 1.4). Daher sollte die Anwendung einfach und ohne technisches Wissen bedient werden können. Andererseits sollen Datensätze einfach und schnell analysiert und bearbeitet werden können. Durch eine Visualisierung der Datensätze und ein intuitives Bedienkonzept zur Bearbeitung der visualisierten Datensätze wird dies gewährleistet.

Das Design der GUI wird nachfolgend konzeptionell beschrieben.

5.6.1. Design der grafischen Benutzeroberfläche

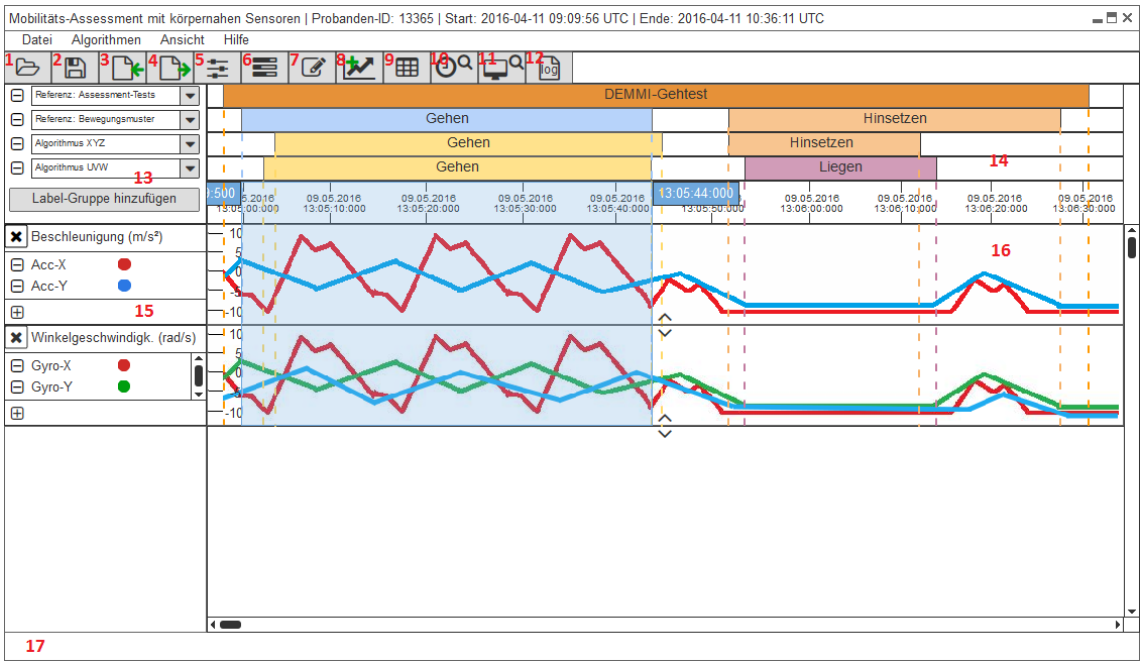


Abbildung 39.: Hauptfenster des GUI-Mockups

Zur Umsetzung der GUI wurde zunächst ein sogenanntes Mockup erstellt, also ein Modell der GUI. Im Folgenden werden die einzelnen Elemente des Mockups, die in Abbildung 39 durchnummeriert sind, genauer erläutert. Zu Beachten ist jedoch, dass die GUI im Projektverlauf kontinuierlich optimiert wurde und das tatsächliche Design der GUI daher vom hier beschriebenen ersten Konzept des Designs abweichen kann.

Titelleiste

Sobald ein Datensatz geladen wurde, wird in der Titelleiste die Probanden-ID und der Aufnahmezeitraum des ausgewählten Datensatzes angezeigt.

Datei:

- **Arbeitsverzeichnis wählen**

  Bevor die Anwendung genutzt werden kann, muss zunächst einmal ein Arbeitsverzeichnis festgelegt werden, in das die Datensätze importiert und die Datenreihen gespeichert werden sollen. Es ist möglich, das gewählte Verzeichnis als Standardarbeitsverzeichnis eintragen zu lassen.

  Mit dem Menüpunkt „Arbeitsverzeichnis wählen “ kann man das Arbeitsverzeichnis

nachträglich noch ändern.

- **Arbeitsverzeichnis extern öffnen**

  Das Arbeitsverzeichnis wird extern in einem Datei-Browser geöffnet. Die Datenreihen und Label-Informationen der in das Arbeitsverzeichnis importierten Datensätze lassen sich hier komfortabel einsehen oder löschen.

- **Datensatz laden (1)**

  Anhand der Probanden-ID und des Aufnahmezeitraums kann aus den bereits in die Anwendung importierten Datensätzen einer ausgewählt werden, für den die Assessment-Tests und Bewegungsmuster visualisiert und analysiert werden sollen.

- **Label speichern (2)**

  Sobald neue Label-Informationen (z. B. auf Grund ausgeführter Algorithmen) vorliegen, erscheint ein Sternchen hinter dem Anwendungsnamen in der Titelleiste. Die entsprechenden Label-Informationen können dann im XML-Format gespeichert werden.

- **Datensatz importieren (3)**

  Um einen Datensatz mit Rohdaten importieren zu können, muss zunächst mit Hilfe eines Datei-Browsers das Verzeichnis des gewünschten Probanden ausgewählt werden und, ob die Studienraum- oder die Daheimaufnahme visualisiert werden soll. Enthält die entsprechende Datei mehr als ein Aufnahmesegment, lässt sich in einer Dialog-Box eines dieser Segmente auswählen, die nach Start- und Endzeitpunkt sowie Aufnahmedauer unterscheidbar sind. Anschließend wird der jeweilige Datensatz vom CSV-Format ins Binärformat übersetzt, sodass dieser in der Anwendung verarbeitet werden kann.

- **Datensatz exportieren (4)**

  Mit dieser Funktion kann man Datenreihen vom Binärformat ins CSV-Format übertragen und mit Hilfe eines Datei-Browsers sowohl Datenreihen als auch Label-Informationen im gewünschten Zielordner ablegen.

- **Anwendung beenden**

  Mit dem Menüpunkt „Beenden“ oder dem Kreuz in der rechten oberen Ecke lässt sich die Anwendung beenden.

Algorithmen:

- **Algorithmus anwenden (5)**

Aus einer Liste wird zunächst ein Algorithmus oder eine Algorithmussequenz ausgewählt. Danach lassen sich gegebenenfalls Parameter einstellen. In Dropdown-Listen hat man die Möglichkeit, über Checkboxen eine oder mehrere auf den Algorithmus anwendbare Datenreihen auszuwählen. Möchte man denselben Algorithmus nacheinander mit verschiedenen Parameterwerten ausführen, so kann man im dazugehörigen Parameterwertfeld die unterschiedlichen Parameterwerte durch Semikolon getrennt eingeben. Eine Sequenz von Parameterwerten, die gleich weit auseinanderliegen, lässt sich im Parameterwertfeld eingeben, indem man den Sequenzstart- und -endwert sowie die Schrittweite jeweils durch Doppelpunkt trennt. Den Status sowie Start- und Endzeitpunkt der einzelnen Algorithmusausführungen kann man sich in der Job-Management-Tabelle anzeigen lassen.

Über den Erfolg oder Misserfolg der Algorithmusanwendung erhält man am rechten unteren Bildschirmrand eine Nachricht. Bei erfolgreicher Algorithmusausführung ist es hier darüber hinaus möglich, sich die errechneten Datenreihen gemäß ihrer Einheiten in Plots ausgeben zu lassen.

- **Job-Management (6)**

  Das Job-Management stellt eine Liste der auszuführenden Algorithmen dar, der sogenannten Jobs. Diese Job-Management-Tabelle gibt Auskunft über den Start- und Endzeitpunkt sowie den Status (wartend, in Ausführung, abgebrochen oder beendet) aller Jobs. Des Weiteren ist es in dieser Tabelle möglich, einen Job abzubrechen.

- **Label hinzufügen (7)**

  Um einem Zeitabschnitt des Datensatzes ein Assessment-Test- oder Bewegungsmuster-Label zuordnen zu können, muss zunächst mit Hilfe einer Dropdown-Liste der gewünschte Label-Typ ausgewählt werden. Darunter lässt sich dann das Intervall, für das das Label eingefügt werden soll, angeben. Das den Bildschirm ausfüllende Betrachtungsintervall ist hier voreingestellt.

- **Auf Selektion zuschneiden**

  Wenn ein Datensatz geladen ist und mit der Maus ein Bereich aufgezogen wurde, ist es möglich unter „Bearbeiten“ die Funktion „Auf Selektion zuschneiden“ auszuwählen. Hiermit wird ein neuer Datensatz erzeugt, der dem originalen Datensatz identisch ist, aber lediglich den selektierten Bereich beinhaltet. Der zugeschnittene Datensatz ist in dem entsprechenden Probanden-Ordner zu finden und wird nach der Erzeugung direkt geladen, sodass der Benutzer nach dem Zuschneiden hiermit weiterarbeiten kann.

Ansicht:

- **Plot hinzufügen (8)**

  Um einen Plot hinzuzufügen, wählt man zunächst die Einheit aus, mit der der Plot dargestellt werden soll. Danach kann man aus den dieser Einheit entsprechenden Datenreihen die zu visualisierenden aussuchen.

- **Douglas-Peucker-Algorithmus auf Plots anwenden**

  Die Anwendung dieses Algorithmus auf die Plots führt bei einer Vergrößerung des Betrachtungsintervalls zur Kurvenglättung. Diese Technik ist in Abschnitt 6.1.5 genauer beschrieben.

- **Datenreihen-Tabelle (9)**

  Mit dieser Funktion lassen sich alle Datenreihen-Werte aus dem gewählten Datensatz in Tabellenform darstellen. Mit dem Plus-Button rechts oben kann man einzelne Datenreihen ausblenden und so die Tabellenansicht eingrenzen.

- **Zeitachse skalieren (10)**

  Die Skalierung der Zeitachse lässt sich in Abhängigkeit zur Bildschirmgröße anpassen.

- **Betrachtungsintervall wählen (11)**

  Die Werte des gewählten Betrachtungsintervalls werden innerhalb der durch den Bildschirm gegebenen Plot-Breite dargestellt. Das den Bildschirm aktuell ausfüllende Betrachtungsintervall ist hier standardmäßig eingetragen.

- **Zeit relativ anzeigen** Soll die Zeit in der Zeitleiste relativ angezeigt werden, so wird der Startzeitpunkt der Aufnahme als Zeitpunkt 0 und alle weiteren Zeitpunkte relativ zum Startzeitpunkt angezeigt. Bei absoluter Anzeige der Zeit in der Zeitleiste werden die tatsächlichen Zeitpunkte der Aufnahme angezeigt.

- **Log (12)**

  Der Log protokolliert die vom Nutzer ausgeführten Aktionen, gibt gegebenenfalls Fehlermeldungen aus und speichert diese in Textform ab. Beispielsweise werden für einen ausgeführten Algorithmus die eingestellten Parameter, die Start- und Endzeitpunkte sowie vom Algorithmus erzeugte zusätzliche Informationen im Log ausgegeben.

Label-Gruppen-Auswahl (13), Label-Leisten und Zeitachse (14)

Über der Zeitachse befinden sich die Leisten zur Anzeige der Label-Informationen. Der Benutzer kann mit den Steuerelementen links neben den Label-Leisten auswählen, welche Label-Gruppe in welcher Leiste angezeigt werden soll und wie viele Label-Leisten angezeigt werden sollen. Zur Verdeutlichung der Start- und Endzeitpunkte einzelner Labels werden gestrichelte Linien über die darunter liegenden Plots gezeichnet.

Durch einen Rechtsklick auf ein Label öffnet sich ein Kontextmenü mit den Einträgen „Label bearbeiten“, „Label entfernen“, „Label hinzufügen“ und „Betrachtungsintervall wählen“. Zur Label-Bearbeitung ändert man entweder mit Hilfe der Dropdown-Liste im „Label bearbeiten“-Dialog das Assessment-Test- bzw. Bewegungsmuster-Label oder das dazugehörige Intervall.
Die Markierungen der Zeitachse sind beschriftet mit dem Datum und der Zeitangabe im Format „Stunde:Minute:Sekunde:Millisekunde“.

Plot-Info-Box (15)

Hier befinden sich die Informationen zu einem Plot und darüber hinaus ist es in diesem Bereich möglich, den Plot nachträglich zu bearbeiten. Mit einem Klick auf das Kreuz oben links lässt sich der Plot aus der Ansicht entfernen. Rechts daneben wird die Einheit der Datenreihen des jeweiligen Plots angezeigt. Darunter liegt die Legende, in der sich durch einen Klick auf den Minus-Button Datenreihen aus dem Plot entfernen lassen. Die Legende befindet sich in einer Scrollbox, damit bei vertikaler Verkleinerung des Plots alle Datenreihen sichtbar bleiben. In der unteren linken Ecke lässt sich mit dem Plus-Button eine Datenreihe hinzufügen. Es öffnet sich hierfür eine Dialogbox mit den noch nicht im Plot dargestellten Datenreihen der zugehörigen Einheit.

Plot-Ansicht, -Skalierung und -Scroll-Balken (16)

In diesem Bereich werden die Datenreihen grafisch dargestellt. Auf der linken Seite eines jeden Plots befindet sich die jeweilige Einheitenachse.
Innerhalb der Plots und der darüber liegenden Zeitachse lässt sich ein Zeitabschnitt markieren, indem man die linke Maustaste gedrückt hält. Durch anschließenden Rechtsklick in diesen Abschnitt öffnet sich je nachdem, ob die Abschnittsmarkierung in der Zeitachse erfolgte oder in einem der Plots, entweder der Zeitachse-skalieren-Dialog oder ein Kontextmenü mit den Einträgen „Label hinzufügen“, „Betrachtungsintervall wählen“ und „Y-Achse skalieren “.

Bei „Label hinzufügen“ öffnet sich eine Dialogbox, in der dem markierten Bereich mit Hilfe einer Dropdown-Liste ein Assessment-Test- oder Bewegungsmuster-Label zugeordnet werden kann. Darunter lässt sich das Intervall, in dem das Label eingefügt werden soll, ändern. Das Intervall des markierten Bereichs im Plot ist hier voreingestellt.
Die Kontextmenüoption „Betrachtungsintervall wählen“ öffnet eine Dialogbox zur Einstellung eines Intervalls, dessen Werte dann über die durch den Monitor begrenzte Plot-Breite dargestellt werden. Voreingestellt ist der im Plot markierte Bereich.
Mit dem Kontextmenüpunkt „Y-Achse skalieren “ öffnet sich für den Plot, aus dem heraus die Abschnittsmarkierung erfolgte, eine Dialogbox, in der die Extremwerte über alle in diesem Plot dargestellten Datenreihen angezeigt werden. Hier lassen sich Minimum und Maximum anpassen.
Rechts neben den angezeigten Plots befindet sich ein Scroll-Balken, mit denen man die angezeigten Plots in vertikaler Richtung verschieben kann. Ein weiterer Scroll-Balken befindet sich am unteren Fensterrand zur Verschiebung der Zeitachse und damit des angezeigten Betrachtungsintervalls.

Statusleiste (17)
In dieser Leiste werden Benachrichtigungen an den Nutzer ausgegeben, wie z. B. die erfolgreiche Durchführung eines Datensatzexports.

5.7. Algorithmen

Wie Anwendungsfall UC-08 in Abschnitt 4.1 beschreibt, sollen mithilfe der Anwendung Algorithmen zur Analyse von Datensätze ausgeführt werden können. Die Anwendung ist flexibel aufgebaut und erlaubt es daher neue Algorithmen einfach zu integrieren. Dies ist sinnvoll, da die aufgenommenen Datensätze auf unterschiedlichsten Wegen analysiert und ausgewertet werden können. Die bereits in die Anwendung integrierten Algorithmen werden nachfolgend beschrieben.

5.7.1. Madgwick (MadgwickAlgorithm)

Der Madgwick-Algorithmus berechnet die Orientierung des Sensorgürtels im dreidimensionalen Raum. Zur Berechnung werden alle Datenreihen der drei Sensoren Accelerometer,

Gyroskop und Magnetometer des Sensorgürtels benötigt.

Eingabeparameter

Parametername	Bedeutung
acc-x	Die Id der Datenreihe, die die Werte der X-Achse des Accelerometers speichert.
acc-y	Die Id der Datenreihe, die die Werte der Y-Achse des Accelerometers speichert.
acc-z	Die Id der Datenreihe, die die Werte der Z-Achse des Accelerometers speichert.
gyro-x	Die Id der Datenreihe, die die Werte der X-Achse des Gyroskops speichert.
gyro-y	Die Id der Datenreihe, die die Werte der Y-Achse des Gyroskops speichert.
gyro-z	Die Id der Datenreihe, die die Werte der Z-Achse des Gyroskops speichert.
mag-x	Die Id der Datenreihe, die die Werte der X-Achse des Magnetometers speichert.
mag-y	Die Id der Datenreihe, die die Werte der Y-Achse des Magnetometers speichert.
mag-z	Die Id der Datenreihe, die die Werte der Z-Achse des Magnetometers speichert.
beta	Gibt den „Algorithm Gain“ des Madgwick-Algorithmus an. Dieser Parameter sollte mit einen Wert im Intervall [0, 1] eingestellt werden.

5.7.2. Gauß-Filter (GaussianFilterAlgorithm)

Dieser Algorithmus wendet den Gauß-Filter auf eine Datenreihe an und kann bei einer Aktivitätserkennung zur Vorverarbeitung benutzt werden. Der Gauß-Filter wurde bereits in Abschnitt 3.4.1 beschrieben.

Eingabeparameter

Parametername	Bedeutung
seriesInId	Die Id der Datenreihe, auf die der Gauß-Filter angewendet werden soll.
sigma	Der Glättungsgrad des Gauß-Filters. Je höher dieser Wert eingestellt wird, desto höher mehr werden die Werte der Datenreihe geglättet.

5.7.3. Highpass-Filter (HighpassFilterAlgorithm)

Dieser Algorithmus wendet den Hochpass-Filter auf eine Datenreihe an und kann bei einer Aktivitätserkennung zur Vorverarbeitung benutzt werden. Der Hochpass-Filter wurde bereits in Abschnitt 3.4.1 beschrieben.

Eingabeparameter

Parametername	Bedeutung
seriesInId	Die Id der Datenreihe, auf die der Hochpass-Filter angewendet werden soll.
cutoff-freq-hz	Die Grenzfrequenz des Hochpass-Filters in Hertz.
start	Der Startwert des Hochpass-Filters.

5.7.4. Lowpass-Filter (LowpassFilterAlgorithm)

Dieser Algorithmus wendet den Tiefpass-Filter auf eine Datenreihe an und kann bei einer Aktivitätserkennung zur Vorverarbeitung benutzt werden. Der Tiefpass-Filter wurde bereits in Abschnitt 3.4.1 beschrieben.

Eingabeparameter

Parametername	Bedeutung
seriesInId	Die Id der Datenreihe, auf die der Tiefpass-Filter angewendet werden soll.
cutoff-freq-hz	Die Grenzfrequenz des Tiefpass-Filters.
start	Der Startwert des Tiefpass-Filters in Hertz.

5.7.5. Median-Filter (MedianFilterAlgorithm)

Dieser Algorithmus wendet den Medianfilter auf eine Datenreihe an und kann bei einer Aktivitätserkennung zur Vorverarbeitung benutzt werden. Der Medianfilter sorgt für eine Rauschunterdrückung in einer Datenreihe, da für jeden Wert der Datenreihe der Median aus diesem und seinen Nachbarwerten gebildet wird.

Eingabeparameter

Parametername	Bedeutung
seriesInId	Die Id der Datenreihe, auf die der Medianfilter angewendet werden soll.
windowRadius	Gibt an, wie viele linke Nachbarwerte beziehungsweise rechte Nachbarwerte bei der Berechnung des Median für den betrachteten Wert der Datenreihe berücksichtigt werden sollen.

5.7.6. Merkmalsvisualisierung (FeatureVisualization)

Dieser Algorithmus exportiert die für eine Aktivitätserkennung berechneten Merkmalsvektoren in CSV-Dateien, damit diese mithilfe einer externen Anwendung (z. B. MATLAB oder R) visualisiert und ausgewertet werden können. Außerdem erzeugt dieser Algorithmus eine Bilddatei, die die berechneten Merkmalsvektoren grafisch darstellt. Der Algorithmus eignet sich daher um festzustellen, welche Merkmale für die Aktivitätserkennung sinnvoll sind und welche Merkmale keine Aussagekraft in Bezug auf die Unterscheidung von Aktivitäten besitzt.

Eingabeparameter

Parametername	Bedeutung
applyPrincipalComponentAnalysis	Gibt an, ob die Hauptkomponentenanalyse zur Dimensionsreduktion für die berechneten Merkmalsvektoren angewendet werden soll.
exportDestinationFolderPath	Der Ordnerpfad, in der die generierten CSV-Dateien gespeichert werden sollen.

labelGroupId	Die Id der Labelgruppe, die zur Berechnung der Merkmalsvektoren verwendet werden soll.
plotToImageAtPath	*Optional.* Der Dateipfad, in der die erzeugte Bilddatei mit der grafischen Darstellung der Merkmalsvektoren gespeichert werden soll.

5.7.7. Labelgruppen-Zeitdauer-Evaluierung (LabelGroupTemporalEvaluation)

Dieser Algorithmus speichert die Gesamtzeitdauer der unterschiedlichen Aktivitäten einer Labelgruppe in eine CSV-Datei ab. Dadurch kann für das Training eines Modells zur Aktivitätserkennung abgeschätzt werden, für welche Aktivitäten genug Trainingsbeispiele vorhanden sind, für welche Aktivitäten noch Aufnahmen benötigt werden oder für welche Aktivitäten Techniken des Under- beziehungsweise Over-Sampling angewendet werden sollten. Dieser Algorithmus kann auch nach einer Aktivitätserkennung ausgeführt werden um Informationen über die Zeitdauer getätigter Aktivitäten eines Probanden zu erhalten.

Eingabeparameter

Parametername	**Bedeutung**
exportDestinationFile	Der Dateipfad, in der die CSV-Datei mit der Gesamtzeitdauer der unterschiedlichen Aktivitäten gespeichert werden soll.
labelGroupId	Die Id der Labelgruppe, über die die Zeitdauer der Aktivitäten berechnet werden soll.

5.7.8. Parameter-Optimierung (ParameterOptimizationAlgorithm)

Der Algorithmus zur Optimierung von Parametern ist ein evolutionärer Algorithmus, der in einem Werteraum die optimalen Werte für die Eingabeparameter von Klassifikatoren sucht, die eine Aktivitätserkennung mithilfe von Techniken des maschinellen Lernens durchführen. Eine Beschreibung des eingesetzten Verfahrens zur Parameteroptimierung ist in Abschnitt 5.9 zu finden.

Eingabeparameter

Parametername	Bedeutung
classifier	Der Klassifikator, für welchen die Parameter-Optimierung angewendet werden soll.
generateMissingValues	Gibt an, ob fehlende Individuen oder fehlende Werte in der Properties-Datei für die initiale Population mit Zufallswerten im zugehörigen Intervall belegt werden sollen oder der Algorithmus mit einer Fehlermeldung abgebrochen werden soll.
initialPopulationFile	Dateipfad zur Properties-Datei, in der die initiale Population für den evolutionären Algorithmus gespeichert ist.
labelGroupId	Die Id der Labelgruppe, die manuell beschriftete Aktivitäten enthält, die als Trainingsbeispiele bei der Parameter-Optimierung benutzt werden.
logPopulations	Gibt an, ob die erzeugten Populationen des evolutionären Algorithmus in eine Datei gespeichert werden sollen.
offspring	*Optional.* Gibt an, wie viele Kinder der evolutionäre Algorithmus pro Population erzeugen soll.
outputDirectory	Der Ordnerpfad, in der das Ergebnis der Paramater-Optimierung gespeichert werden soll.
parentsPerChild	Die Anzahl der Eltern pro Individuum des evolutionären Algorithmus.
threadCount	*Optional.* Die Anzahl der Java-Threads mit der die Parameter-Optimierung ausgeführt werden soll. Die einzelnen Schritte des evolutionären Algorithmus werden soweit wie möglich auf den Java-Threads nebenläufig ausgeführt.
timeBudgetMinutes	Gibt an, nach wie vielen Minuten der Algorithmus abgebrochen werden soll und die bis dahin gefundene beste Wertebelegungen der Parameter ausgegeben werden sollen.

5.7.9. Konfigurierbares Training (ConfigurableTraining)

Dieser Algorithmus verwendet Techniken des maschinellen Lernens, um ein Modell für die Erkennung von Aktivitäten zu trainieren. Das Training wird mithilfe von Properties-

Dateien für die vier Klassifikatoren (State, Static, Transition, Dynamic) konfiguriert.

Eingabeparameter

Parametername	**Bedeutung**
labelGroupId	Die Id der Labelgruppe, die die manuell beschrifteten Aktivitäten enthält und daher die Trainingsbeispiele für das maschinelle Lernen liefert.
modelFileOutput	Der Dateipfad, in der das trainierte Modell abschließend gespeichert werden soll.
properties-dynamic, properties-state, properties-static, properties-transition	Die Pfade zu den Properties-Dateien der vier Klassifikatoren. Eine Properties-Datei beschreibt • die anzuwendenden Vorverarbeitungsverfahren • die Fenstergröße und Schrittweite für das Sliding-Window-Verfahren • die Merkmale, die für einen Merkmalsvektor berechnet werden sollen • die anzuwendenden Dimensionsreduktionsverfahren • das anzuwendende Klassifikationsverfahren und die Werte für seine Parameter
trainer	Gibt an, wie das maschinelle Lernen durchgeführt werden soll. Es steht ein Split-Trainer zur Verfügung, der die Datensätze in Datensätze für das Training und Datensätze für die Validierung im angegebenen Verhältnis trennt (z. B. split(0.5)). Des Weiteren steht ein Trainer zur Verfügung, der die Datensätze in die Anzahl der angegebenen Teilmengen unterteilt und anschließend eine Kreuzvalidierung durchführt (z. B. cv(5)). Wird der Kreuzvalidierungs-Trainer ausgewählt, so wird keine Modelldatei ausgegeben.

5.7.10. Klassifikation (ClassificationAlgorithm)

Der Algorithmus zur Klassifikation führt eine Aktivitätserkennung für ausgewählte Datensätze durch. Die Klassifikation beschriftet einzelne Zeitintervalle eines Datensatzes mit einer Aktivität. Dabei kann der Klassifikations-Algorithmus die in Abschnitt 4.2 (Anforderung 1.3.3.2) aufgeführten Alltags-Bewegungsmuster erkennen. Die Klassifikation wird auf Basis eines trainierten Modells durchgeführt, das vorher mithilfe des Algorithmus „Konfigurierbares Training“ erzeugt werden kann.

Eingabeparameter

Parametername	**Bedeutung**
labelGroupId	*Optional.* Die Id der Labelgruppe, die als Referenz für die Berechnung einer Konfusionsmatrix dient.
modelFileInput	Der Dateipfad, in der das trainierte Modell gespeichert ist mit dem die Klassifikation durchgeführt werden soll. Dieses Modell kann mithilfe des Algorithmus „Konfigurierbares Training“erzeugt werden.

5.7.11. Label-Scoring (LabelScoring)

Der Label-Scoring-Algorithmus berechnet für zwei Labelgruppen eine Konfusionsmatrix um ein Maß für die Gleichheit der beiden Labelgruppen zu erhalten.

Eingabeparameter

Parametername	**Bedeutung**
comparisonMode	Gibt an, ob die Konfusionsmatrix nur für die Aktivitätszustände (Dynamisch, Statisch, Transition) oder für die tatsächlichen Aktivitäten berechnet werden soll.
refLabelGroupId	Die Id der Labelgruppe, die als Referenz zur betrachteten Labelgruppe bei der Berechnung der Konfusionsmatrix verwendet werden soll.
testLabelGroupId	Die Id der Labelgruppe, für die die Konfusionsmatrix gegenüber einer Referenz-Labelgruppe berechnet werden soll.

5.7.12. Schritterkennung (StepDetectionAlgorithm)

Der Algorithmus zur Schritterkennung erkennt einzelne Schritte innerhalb bereits erkannter „Gehen“-Segmenten. Das bedeutet, dass einzelne Zeitintervalle innerhalb der „Gehen“-Segmente Schritte zugeordnet werden.

Eingabeparameter

Parametername	**Bedeutung**
Peak Width (in ms)	Gibt ein Zeitintervall an, in der die rechten beziehungsweise linken Nachbarwerte bei der Erkennung eines Extrempunktes berücksichtigt werden sollen. Es wird der Mittelwert dieser Nachbarwerte gebildet und mit dem betrachteten Wert verglichen und dann entschieden, ob es sich beim betrachteten Wert um ein Extrempunkt handelt.
Peak Threshold	Schwellenwert zur Erkennung eines Extrempunktes. Je höher dieser Wert ist, desto weiter muss der betrachtete Wert von seinen Nachbarwerten abweichen um als Extrempunkt erkannt zu werden.
Scale factor of acceleration energy	Skalierungsfaktor, der angibt, wie groß der Einfluss der Energie der Beschleunigungssensoren auf die Wahrscheinlichkeit des Fersenauftritt haben soll.
Heel strike Probability Peak Width (in ms)	Gibt ein Zeitintervall an, in der die rechten beziehungsweise linken Nachbarwerte bei der Erkennung von Hochpunkten in der Zeitreihe der Wahrscheinlichkeit des Fersenauftritts berücksichtigt werden sollen.
Minimal distance between two heel strikes (in ms)	Gibt die minimale zeitliche Distanz zwischen zwei möglichen Fersenauftritte an um Fehlerkennungen zu vermeiden.
labelGroupId	Gibt die Id der Labelgruppe an, für die die Schritterkennung durchgeführt werden soll. Die Schritterkennung wird für alle „Gehen“-Label der Labelgruppe durchgeführt.

5.7.13. Gangparameter (GaitParameterAlgorithm)

Der Algorithmus zur Bestimmung von Gangparametern ist eine Erweiterung des Algorithmus zur Schritterkennung. Dieser Algorithmus bestimmt nach der Durchführung der Schritterkennung die in Abschnitt 3.5 aufgelisteten direkt ableitbaren Gangparameter und gibt diese im Form eines Reports aus.

Eingabeparameter

siehe Abschnitt 5.7.12.

5.8. Maschinelles Lernen

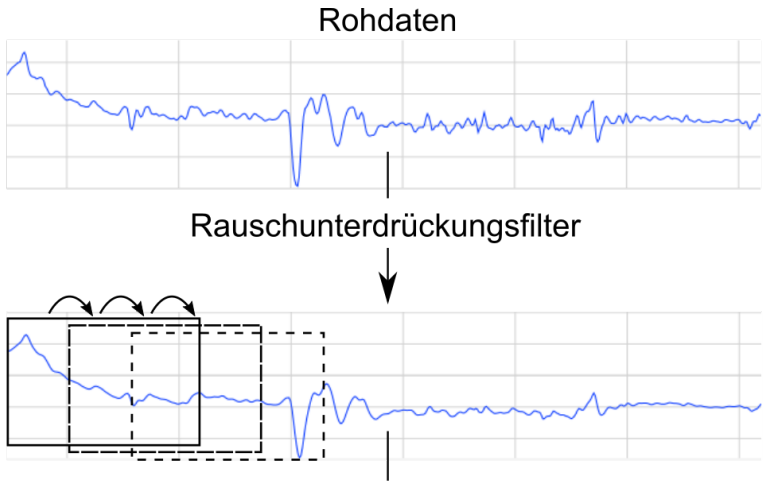
Im Grundlagenkapitel wurden viele Techniken eingeführt, die zur Umsetzung der Aktivitätserkennung mit Hilfe von Machine Learning notwendig oder hilfreich sind. In diesem Kapitel wird erläutert, wie diese Bausteine im Rahmen des Projekts verwendet wurden.

5.8.1. Überblick

Es wird eine hierarchische Klassifikation durchgeführt. Dass heißt, es wird zunächst ein Klassifikator genutzt, um zu entscheiden, ob eine vorliegende Aktivität dynamisch, statisch oder im Übergang ist. Weiterhin werden drei Klassifikatoren für diese drei Klassen trainiert, die entsprechend der Entscheidung des ersten Klassifikators angewendet werden, um die genaue Bewegung zu bestimmen.

Jeder dieser vier Klassifikatoren kann durch eine beliebige Machine-Learning-Technik umgesetzt werden. Im Rahmen des Projekts wurden dafür vier Techniken implementiert: AMM, Multilayer Perceptrons, BDT und BDS.

Ein solcher Klassifikator muss zunächst trainiert werden, bevor er zur Klassifikation eingesetzt werden kann. In beiden Fällen werden zu Beginn Featurevektoren extrahiert. Dieser Ablauf ist in Abbildung 40 dargestellt. Im Anschluss findet entweder ein Training wie in Abbildung 41 zu sehen statt oder eine Klassifikation wie in Abbildung 42.



Featurevektor	Beschriftung	Fenster
(0.1, 0.75, -12.3, 3.7, -2.0)	STAND	[0; 1000]
(-7.7, -1.26, 1.71, 4.1, 0.22)	STAND	[400; 1400]
(14.2, -4.51, 0.29, 2.6, -0.8)	SIT	[800; 1800]

Dimensionsreduktion

Featurevektor	Beschriftung	Fenster
(0.5, 0.5)	STAND	[0; 1000]
(-1.2, 0.78)	STAND	[400; 1400]
(3.1, 7.2)	SIT	[800; 1800]

Normalisierung

Featurevektor	Beschriftung	Fenster
(-0.3, -2.33)	STAND	[0; 1000]
(-2, -2.05)	STAND	[400; 1400]
(2.3, 5.37)	SIT	[800; 1800]

Abbildung 40.: Feature-Extraction

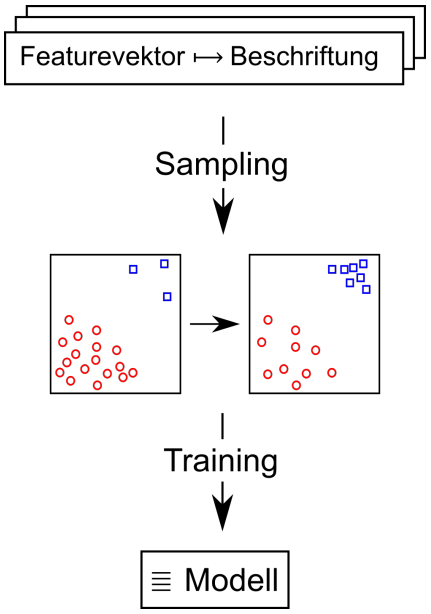


Abbildung 41.: Training eines Klassifikators

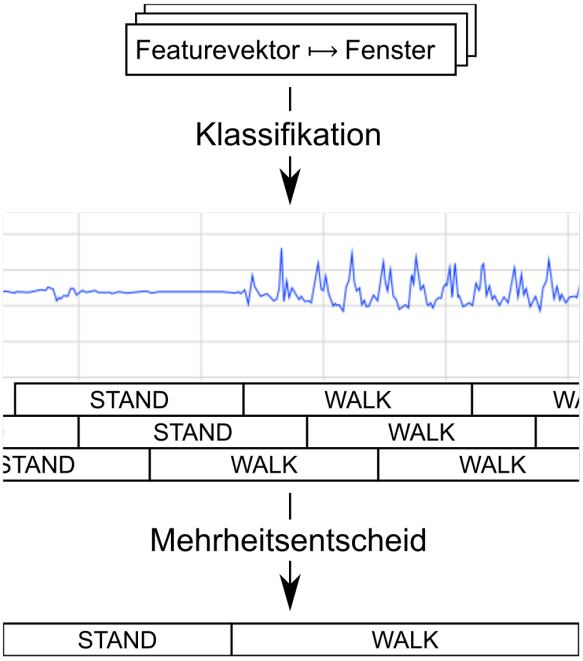


Abbildung 42.: Ausführung eines Klassifikators

5.8.2. Featureextraktion

Der erste Schritt der Feature-Extraktion, die Anwendung eines Rauschunterdrückungsfilters, ist optional. Es besteht die Möglichkeit einen Lowpass-, Highpass-, Gauss- oder Medianfilter anzuwenden, um die Sensorsignale zu verbessern.

Anschließend wird ein Sliding-Window-Verfahren eingesetzt, um Featurevektoren aus den Datenreihen zu extrahieren. Die Featurevektoren werden zusammegesetzt aus einzelnen Features, wie sie in Unterabschnitt 3.4.3 beschrieben sind. Im Rahmen einer Ausarbeitung (siehe Unterabschnitt D.8) wurden vielversprechende Kombinationen von Features bestimmt und als Feature-Sets in der Anwendung definiert. Es kann daher zwischen verschiedenen Feature-Sets zur Extraktion gewählt werden.

Die letzten beiden Schritte sind wieder optional. Auf die Featurevektoren kann eine Dimensionsreduktionsmethode angewandt werden, um die zu verarbeitende Datenmenge zu verringern. Bisher ist nur die Hauptkomponentenanalyse implementiert. Wenn ein Multilayer Perceptron trainiert wird, muss weiterhin eine Normalisierung stattfinden, bei der die Daten so transformiert werden, dass sie einen Mittelwert von null und eine Standardabweichung von eins erhalten.

5.8.3. Training

Nach der Featureextraktion liegt eine Menge von Featurevektoren mit ihrer zugehörigen Beschriftung vor. Die Beschriftung der Featurevektoren wurde aus der Referenzbeschriftung des Datensatzes gewonnen, während die Featureextraktion mit dem Sliding Window durchgeführt wurde. Um dem Ungleichgewicht zwischen verschiedenen Klassen (Beschriftungen) entgegenzuwirken, werden Undersampling und SMOTE eingesetzt. Mit den aufbereiteten Trainingsdaten wird dann ein Klassifikator trainiert.

5.8.4. Klassifikation

Bei der Klassifikation entstehen durch die Featureextraktion eine Menge von Featurevektoren mit zugehörigen Fensterpositionen. Die Beschriftung ist im Allgemeinen unbekannt, da neue Datensätze klassifiziert werden sollen. Die Featurevektoren werden dem Klassifikator als Eingabe übergeben und es wird eine Vorhersage für die Beschriftung zurückgeliefert. Durch den Einsatz des Sliding-Window-Verfahrens entstehen allerdings Überlappungen.

Es wird daher ein Mehrheitsentscheid für jeden Zeitpunkt durchgeführt, um die endgültige Beschriftung zu bestimmen.

5.9. Parameteroptimierung

Die in Abschnitt 5.8 beschriebene Machine-Learning-Pipeline besitzt zahlreiche einstellbare Parameter. Einige davon sind Parameter der Machine-Learning-Techniken selber, wie beispielsweise die Anzahl der Schichten eines MLPs, andere sind unabhängig davon, wie z. B. die Fensterlänge. Je besser diese Parameter eingestellt sind, desto besser wird das tranierte Modell klassifizieren können. Da die Menge und der Wertebereich der Parameter sehr groß sind, wurde eine automatische Parameteroptimierung entwickelt.

Sie basiert auf evolutionärer Optimierung, sodass die Parameterkonstellationen als Individuen dargestellt werden und ihre Fitness der mit diesen Parametern erreichten F1-Score entspricht. Wiederholt werden Eltern-Individuen rekombiniert, mutiert und die besten Individuen für die nächste Generation selektiert. Es wird ein $\mu + \lambda$-EA genutzt, wobei μ (Anzahl der Eltern pro Generation) und λ (Anzahl der Kinder pro Generation) frei einstellbar sind. Die Plus-Selektion führt dazu, dass die nächste Elterngeneration in jedem Fall auch das bisher beste Individuum enthält. Ebenfalls kann die Anzahl der Eltern, welche zu einem Kind rekombiniert werden sollen, gewählt werden.

Weitere Informationen sind in der Ausarbeitung D.9 zu finden.

6. Realisierung

In diesem Kapitel wird die technische Realisierung der Anwendung beschrieben. Zunächst wird auf die grafische Oberfläche und die einzelnen Komponenten eingegangen. Anschließend wird die Kommandozeilenschnittstelle erklärt. Mithilfe der CLI ist es möglich, Befehle ohne grafische Oberfläche auszuführen. Darauf folgend werden die Datenstrukturen für die Verarbeitung näher erläutert. Zum Schluss wird im Punkt Job-Executor die Ausführung der Algorithmen erklärt.

6.1. Frontend

Im Frontend ist die GUI der Anwendung, die in Abschnitt Abschnitt 5.6 beschrieben wurde, realisiert. Die realisierte GUI bietet Funktionen zum Betrachten und Bearbeiten von Datensätzen sowie zum Ausführen von Algorithmen.

Für die Implementierung der GUI wurde das Framework JavaFX verwendet. JavaFX erlaubt die Beschreibung des Aussehens und der Anordnung der unterschiedlichen Komponenten einer GUI wie z. B. Menüs, Buttons oder Textfelder in einer externen XML-Datei, genannt FXML-Datei. Die Funktionen der so beschriebenen neuen Komponenten werden anschließend in einer Controller-Klasse implementiert. Weitere Informationen zum JavaFX-Framework sind in [(Or13] zu finden.

Es wurden zur Darstellung des Hauptfensters des Frontends fünf JavaFX-Komponenten entwickelt:

FXML	Controller-Klasse	Package
MainApplicationView	MainApplicationController	mamks.frontend
MotionDataView	MotionDataController	mamks.frontend.motiondataview
LabelGroupView	LabelGroupController	mamks.frontend.motiondataview.label

TimelineView	TimelineController	mamks.frontend.motiondataview.timeline
PlotView	PlotController	mamks.frontend.motiondataview.plot

Tabelle 36.: Entwickelte JavaFX-Komponenten

Die MainApplicationView bildet das Hauptfenster des Frontends und die anderen Komponenten wurden in die MainApplicationView integriert. Des Weiteren wurden noch eine Vielzahl von Dialogfenstern entwickelt, die es dem Benutzer erlauben, Einstellungen für die unterschiedlichen Funktionen der Anwendung vorzunehmen und diese auszuführen. Die Dialogfenster können über die Menüleiste, die Buttonleiste oder die MotionDataView aufgerufen werden. Die Logik der GUI ist so aufgebaut, dass Funktionen zur Anzeige und Bearbeitung eines MotionData-Objektes zunächst deaktiviert sind. Nachdem ein MotionData-Objekt in die Anwendung geladen wurde, werden diese Funktionen aktiviert und können ausgeführt werden. In den nachfolgenden Abschnitten werden die in Tabelle 36 aufgelisteten Komponenten detailliert beschrieben. Ebenso wird nachfolgend die Realisierung des Dialogfensters zur Ausführung von Algorithmen beschrieben, das verschiedene Einstellungsmöglichkeiten für die Parameter der Algorithmen bereitstellt.

6.1.1. MainApplicationView

Die MainApplicationView bildet das Hauptfenster der GUI. Sie besteht aus einer Menü- und einer Buttonleiste zum Ausführen bestimmter Funktionen der Anwendung sowie einer MotionDataView zur Darstellung des geladenen MotionData-Objektes und einer Statusleiste zur Anzeige von Informationen. Die MotionDataView integriert weitere Komponenten (Abbildung 43).

Abbildung 43.: MainApplicationView-Komponente

Für die Umsetzung der Funktionen der MainApplicationView wurden neben der Controller-Klasse weitere Klassen für eine bessere Strukturierung implementiert. Diese Klassen und ihre Beziehungen zueinander sind in Abbildung 44 dargestellt. Das MainApplication-Controller-Objekt implementiert das MainApplicationListener-Interface. Dadurch werden Methoden implementiert, die aufgerufen werden, wenn ein neues MotionData-Objekt geladen wird, wenn Systemdaten geändert oder gespeichert werden und zum Ändern des Textes der Statusleiste. Des Weiteren enthält das MainApplicationController-Objekt ein App-Objekt, das Methoden zum Anzeigen von Dialogfenstern bereitstellt. Außerdem werden Methoden zum Ausführen von Algorithmen und Exportieren von Labelgruppen und Datenreihen im App-Objekt bereitgestellt. Das MainApplicationController-Objekt injiziert ebenso ein PreferenceHandler-Objekt, das Methoden zum Speichern und Laden von Benutzer-Einstellungen bereitstellt. Die Einstellungen bleiben auch nach dem Beenden der Anwendung bestehen.

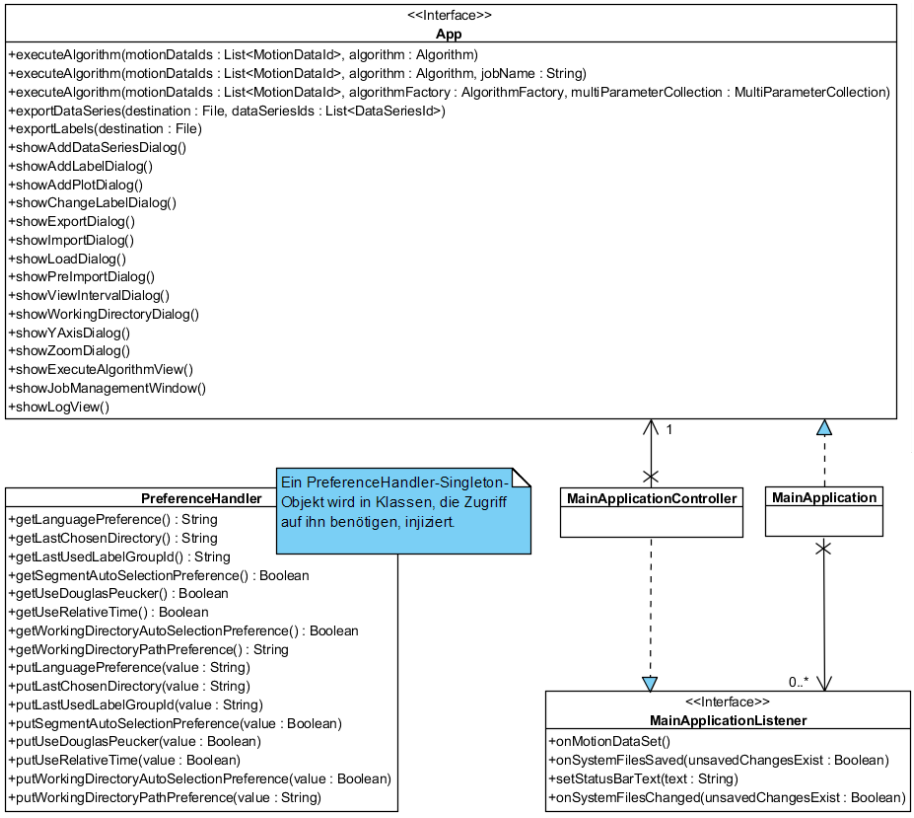


Abbildung 44.: Realisierung der MainApplicationView

Dialogfenster

Neben den fünf Hauptkomponenten wurden eine Vielzahl von Dialogfenstern entwickelt. Die Dialogfenster lassen sich über die Buttonleiste, Menüleiste und MotionDataView öffnen. Das im MainApplicationController-Objekt enthaltene App-Objekt bietet dabei die Methoden zur Anzeige der Dialogfenster an. Eine Übersicht über alle Dialogfenster ist in Tabelle 37 dargestellt. Die Funktionen der Dialogfenster sind dem Javadoc des Quellcodes zu entnehmen.

FXML	Controller-Klasse	Package
DataTableView	DataTableController	mamks.frontend
ExportDialogView	ExportDialogController	mamks.frontend
ImportDialogView	ImportDialogController	mamks.frontend
JobManagementView	JobManagementController	mamks.frontend
JobNotification	JobNotificationController	mamks.frontend
LoadDialogView	LoadDialogController	mamks.frontend

LogView	LogController	mamks.frontend
PreImportDialogView	PreImportDialogController	mamks.frontend
WorkingDirectory-DialogView	WorkingDirectory-DialogController	mamks.frontend
ExecuteAlgorithm-View	ExecuteAlgorithm-Controller	mamks.frontend.algorithm
LabelDialogView	LabelDialogController	mamks.frontend.labeldialog
AddPlotDialogView	AddPlotDialogController	mamks.frontend.motiondataview
ViewIntervalDialog-View	ViewIntervalDialog-Controller	mamks.frontend.motiondataview
AddDataSeries-DialogView	AddDataSeriesDialog-Controller	mamks.frontend.motiondataview.plot
YAxisDialogView	YAxisDialogController	mamks.frontend.motiondataview.plot
ZoomDialogView	ZoomDialogController	mamks.frontend.motiondataview.zoom

Tabelle 37.: Entwickelte JavaFX-Dialogfenster

6.1.2. MotionDataView

Die MotionDataView ist für die Anzeige des geladenen MotionData-Objektes bestimmt. Ein MotionData-Objekt bündelt Informationen über aufgenommene Sensordaten und abgeleitete Daten. Es werden unter anderem aufgenommene Datenreihen, abgeleitete Datenreihen, Labels gruppiert in Labelgruppen und Aufnahmezeitraum in diesem Objekt gespeichert.

Zur Anzeige einer Labelgruppe des MotionData-Objektes wird eine LabelGroupView-Komponente verwendet. Es können beliebig viele LabelGroupView-Komponenten in die MotionDataView dynamisch eingefügt und wieder entfernt werden. Zur Anzeige einer Datenreihe wird eine PlotView-Komponente verwendet. Es können auch hier beliebig viele PlotView-Komponenten in die MotionDataView dynamisch eingefügt und wieder entfernt werden. Da sowohl für die Anzeige der Labelgruppen als auch für die Anzeige der Datenreihen dieselbe x-Achse zur Darstellung des Aufnahmezeitraums benötigt wird, wurde

diese in eine eigene Komponente, der TimelineView-Komponente, umgesetzt. Die MotionDataView besteht dabei stets aus einer TimelineView-Komponente und aus beliebig vielen LabelGroupView-Komponenten und PlotView-Komponenten. Abbildung 45 stellt eine MotionDataView mit der TimelineView-Komponente, zwei LabelGroupView-Komponenten und drei PlotView-Komponenten dar.

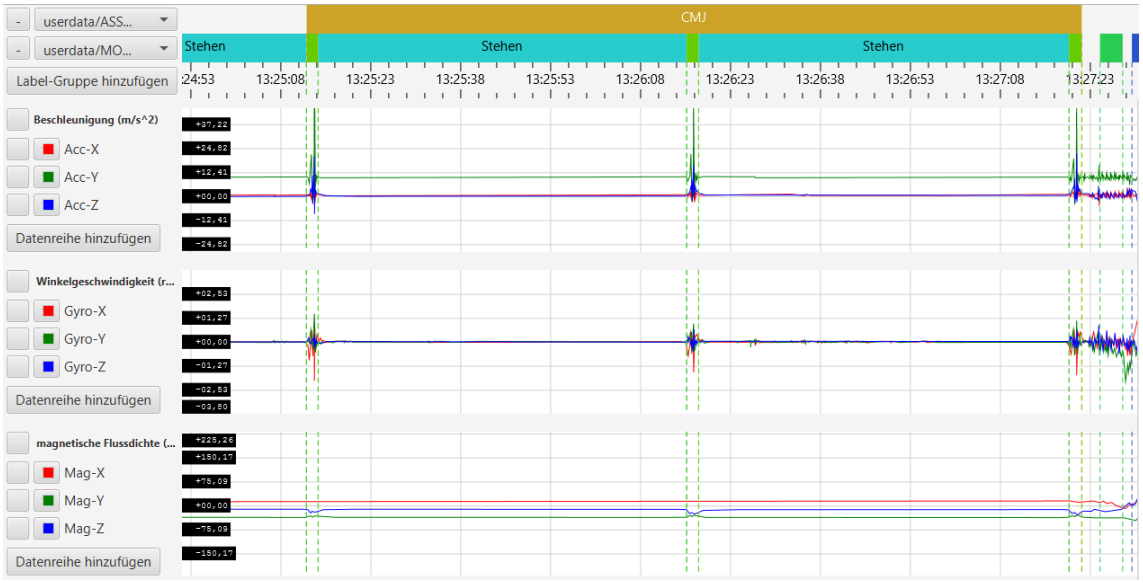


Abbildung 45.: MotionDataView-Komponente

Abbildung 46 stellt die für die Umsetzung der Funktionen der Motiondataview implementierten Klassen und ihre Beziehungen zueinander dar. Die Controller-Klasse der MotionDataView wurde in der MotionDataController-Klasse implementiert. Ein MotionDataController-Objekt speichert dabei die Controller-Klassen-Objekte der eingefügten Komponenten, damit diese hier zentral gesteuert werden können. Zudem enthält das MotionDataController-Objekt ein MotionDataViewState-Objekt, das alle Informationen für die Darstellung des geladenen MotionData-Objektes speichert. Diese Informationen sind unter anderem die Breite des Hauptfensters, das anzuzeigende Betrachtungsintervall, der eingestellte Zoom-Wert (in Zeitdauer pro 100 Pixel) und eventuell das Intervall, das vom Benutzer selektiert wurde. Auch die Einstellungen, ob der Douglas-Peucker-Algorithmus (sieh Abschnitt 6.1.5) auf die angezeigten Datenreihen angewandt werden soll und ob der Aufnahmezeitraum in der Zeitleiste absolut oder relativ zum Aufnahmebeginn angezeigt werden soll, werden hier gespeichert. Außerdem werden im MotionDataViewState-Objekt aus Performanzgründen noch die sichtbaren Labels im anzuzeigenden Betrachtungsintervall gecached, damit diese nicht ständig neu berechnet werden müssen. Da die eingefügten

Komponenten dieselbe x-Achse zur Darstellung des Aufnahmezeitraums nutzen, werden im MotionDataViewState-Objekt ebenfalls Methoden (indexToPosX(index:Integer) und posXToIndex(posX:Double)) zur Konvertierung eines Aufnahmezeitpunktes in eine x-Position auf der GUI und umgekehrt bereitgestellt und entsprechend von den Komponenten zur Darstellung der Labelgruppen, des Aufnahmezeitraums und der Datenreihen genutzt.

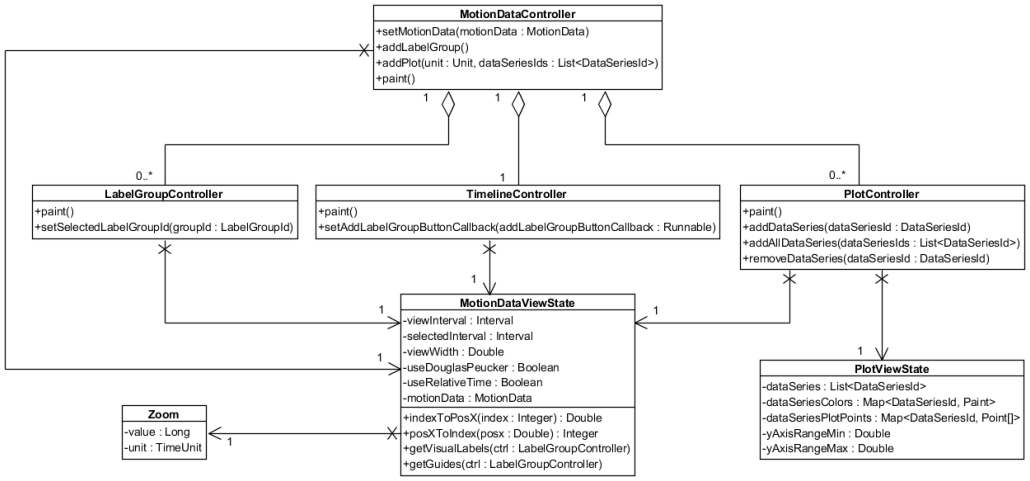


Abbildung 46.: Realisierung der MotionDataView

Wird eine Einstellung vorgenommen, die eine Aktualisierung der MotionDataView erfordert (z. B. durch scrollen oder zoomen), wird die paint-Methode des MotionDataController-Objektes aufgerufen. Diese Methode ruft daraufhin die paint-Methoden der Controller-Klassen-Objekte der derzeit eingefügten Komponenten auf und veranlasst diese damit den Inhalt ihrer Komponenten zu aktualisieren.

6.1.3. LabelGroupView

Eine LabelGroupView ist für die Anzeige der Labels einer Labelgruppe des in der Anwendung geladenen MotionData-Objektes bestimmt. Diese besteht aus einem Button zur Entfernung der LabelGroupView aus der MotionDataView-Komponente, einer Combobox zum Auswählen der anzuzeigenden Labelgruppe und einer Zeichenfläche (Canvas) zum Zeichnen der Labels (Abbildung 47).

Abbildung 47.: LabelGroupView-Komponente

Durch Aufruf der paint-Methode des LabelGroupController-Objektes wird eine Neuzeichnung der Labels ausgelöst. Die im Betrachtungsintervall sichtbaren Labels der ausgewählten Labelgruppe werden vom MotionDataViewState-Objekt durch Aufruf der getVisualLabels-Methode angefordert und anschließend gezeichnet. Neben den Labels werden außerdem gestrichelte Linien gezeichnet. Diese stellen die Grenzen der Labels dar, die von LabelGroupView-Komponenten angezeigt werden, welche sich in der GUI über dieser LabelGroupView befinden. Die Informationen über die anzuzeigenden Linien werden im MotionDataViewState-Objekt gespeichert und können durch Aufruf der getGuides-Methode angefordert und anschließend ebenso gezeichnet werden.

6.1.4. TimelineView

Die TimelineView bildet die x-Achse der MotionDataView. Sie zeigt den Aufnahmezeitraum des in der Anwendung geladenen MotionData-Objektes in einer Zeitleiste an. Aus Platzgründen ist der Button zum Hinzufügen von LabelGroupViews ebenfalls in der TimelineView integriert (Abbildung 48).

Abbildung 48.: TimelineView-Komponente

Ein TimelineController-Objekt enthält ein MotionDataViewState-Objekt, das alle Informationen speichert, die zur Anzeige der Zeitleiste benötigt werden. Wird das Betrachtungsintervall geändert (durch zoomen oder scrollen), wird die paint-Methode des TimelineController-Objektes aufgerufen und die Zeitleiste neu gezeichnet. Neben der Zeitleiste werden außerdem gestrichelte Linien gezeichnet, die die Grenzen der von LabelGroupViews angezeigten Labels darstellen sollen.

6.1.5. PlotView

Aufgrund der riesigen Anzahl an Datenpunkten, die eine Datenreihe typischerweise enthält, und den damit verbundenen Performance-Problemen bei der Verwendung bestehender Komponenten zur Anzeige dieser Datenreihen wurde entschieden, eine eigene Komponente zu entwickeln, die Datenreihen selber effizient zeichnen soll. Die entwickelte PlotView-Komponente besteht aus einem Zeichenfeld (Canvas) für die Anzeige der Datenreihen und einer Legende, die Namen und Farbe der angezeigten Datenreihen darstellt (Abbildung 49).

Abbildung 49.: PlotView-Komponente

Das PlotController-Objekt enthält ein PlotViewState-Objekt, das Informationen speichert, die zur Darstellung des Plots benötigt werden. Diese Informationen sind unter anderem die Liste der aktuell im Plot angezeigten Datenreihen, die Farben, in der die Datenreihen angezeigt werden, die aktuelle Skalierung der Y-Achse und die X- und Y-Positionen im Plot der Datenpunkte der angezeigten Datenreihen. Zusammen mit dem enthaltenen MotionDataViewState-Objekt sind somit alle Informationen verfügbar, um den Plot im Zeichenfeld zu zeichnen.

Der Plot muss dabei immer neu gezeichnet werden, wenn eine Einstellung vorgenommen wurde, die das PlotViewState-Objekt oder das MotionDataViewState-Objekt verändert. Dabei wird die paint-Methode im PlotController-Objekt aufgerufen und zum Zeichnen des Plots ein PlotCanvas-Objekt erzeugt. Dieser bekommt das PlotViewState-Objekt und das MotionDataViewState-Objekt übergeben und zeichnet daraufhin die Datenpunkte der Datenreihen im aktuellen Betrachtungsintervall, eine Y-Achse, ein vertikales und horizontales Gitter und die gestrichelten Linien zur Kennzeichnung der Grenzen der angezeigten Labels. Letzteres wird wieder durch Aufruf der getGuides-Methode angefordert. Falls ein Intervall ausgewählt wurde, wird dieses ebenfalls gezeichnet, und falls sich der Mauszeiger innerhalb der MotionDataView befindet, wird eine vertikale Leiste mit Plot-Informationen zur aktuellen Mausposition gezeichnet.

Douglas-Peucker-Algorithmus

Um eine performante Darstellung der Sensordaten als Plots zu ermöglichen, wird der Douglas-Peucker-Algorithmus eingesetzt. Je nach gewählter Zoom-Stufe der Ansicht würden viele Datenpunkte auf dasselbe Pixel abgebildet werden und sind demnach irrelevant. Daher kann die Darstellung beschleunigt werden, indem die Kurve der Plots nur zwischen relevanten Datenpunkten gezeichnet wird.

Der Douglas-Peucker-Algorithmus vereinfacht eine gegebene Folge von Punkten, die einen Streckenzug darstellen, und versucht dabei seine Gestalt möglichst gut beizubehalten. Als Ergebnis wird eine Untermenge der Eingabe-Punkte zurückgeliefert, welche eine gute Approximation der Kurve bieten. Das Gütekriterium des Algorithmus, mit welchem die zu entfernenden Punkte gewählt werden, ist eine Abstandsmessung zwischen der approximierten Kurve und den weggelassenen Punkten der ursprünglichen Kurve.

6.1.6. TimeIntervalControl-Komponente

Viele Dialogfenster benötigen eine Anzeige zur Bearbeitung von Zeitintervallen. Zu diesen Dialogfenstern gehören das Dialogfenster zur Auswahl eines Betrachtungsintervalls und die Dialogfenster zur Hinzufügung und Bearbeitung von Labels. Um Codeklone zu verhindern wurde eine eigene Komponente zur Anzeige und Bearbeitung von Zeitintervallen entwickelt, die anschließend in die Dialogfenster integriert wurde. Die Funktionalität dieser Komponente ist in der TimeIntervalControl-Klasse realisiert und das Aussehen in der TimeIntervalControl.fxml-Datei definiert. Abbildung 50 stellt die Komponente dar.

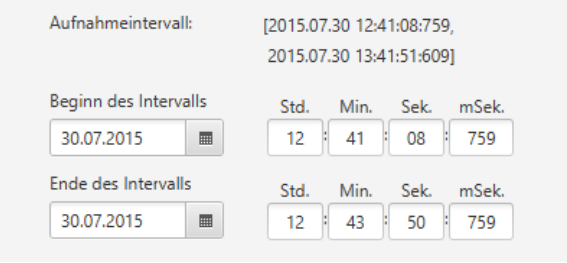


Abbildung 50.: Die TimeIntervalControl-Komponente

Ein Zeitintervall wird durch die Angabe seines Startzeitpunktes und seines Endzeitpunktes beschrieben. Ein Zeitpunkt besteht aus der Angabe eines Datums, einer Stunde, einer

Minute, einer Sekunde und der Millisekunden. Das Datum des Start- und Endzeitpunktes kann in der TimeIntervalControl-Komponente mithilfe einer DatePicker-Komponente bearbeitet werden. Die anderen Bestandteile des Start- und Endzeitpunktes können mithilfe von Textfelder bearbeitet werden. Dabei wird bei jeder Änderung des Wertes eines Textfeldes überprüft, ob die Änderung gültig ist. Ein Textfeld ist gültig, wenn mit seinem Wert und mithilfe der anderen Angaben ein ZonedDateTime-Objekt erzeugt werden kann. Ist dies nicht der Fall (weil z. B. ein Buchstabe eingegeben wurde), wird der Wert des betroffenen Textfeldes mit dem alten Wert zurückgesetzt.

6.1.7. Ausführung von Algorithmen

Über das Frontend können Algorithmen ausgeführt werden. Wird über die Buttonleiste oder die Menüleiste der Button zur Ausführung eines Algorithmus gedrückt, öffnet sich zunächst ein Wizard. Die Funktionalität des Wizards wird von der ExecuteAlgorithmController-Klasse implementiert und das Aussehen von der ExecuteAlgorithmView.fxml definiert. Der Wizard beinhaltet mehrere Seiten, die die verschiedenen Einstellungsmöglichkeiten zur Ausführung eines Algorithmus anzeigen. Eine Seite besteht aus einer VBox-Komponente, die Einstellungsmöglichkeiten bereitstellt. Der Wizard enthält insgesamt vier Seiten, durch die mithilfe des Weiter- und des Zurück-Buttons des Wizards geblättert werden kann und die folgende Einstellungsmöglichkeiten realisieren:

1. Auswahl des auszuführenden Algorithmus
2. Auswahl der Datensätze, für die der Algorithmus ausgeführt werden soll
3. Auswahl der Datensätze, die bei der Validierung einer Klassifikation nicht betrachtet werden sollen (optional, nur für ML-Algorithmen)
4. Einstellung der Eingabeparameter des auszuführenden Algorithmus

Zunächst muss der Algorithmus ausgewählt werden, der ausgeführt werden soll. Alle für die Anwendung implementierten Algorithmen sind in der AlgorithmRegistry-Klasse eingetragen und können so vom Wizard ausgelesen und auf einer Seite in einer ListView-Komponente angezeigt werden.

Auf der nächsten Seite muss angegeben werden, für welche Datensätze der ausgewählte Algorithmus ausgeführt werden soll. Dazu werden alle im Arbeitsverzeichnis befindlichen Datensätze auf der Seite in einer CheckList-Komponente angezeigt, sodass auch mehrere

Datensätze ausgewählt werden können.

Die nächste Seite ist optional. Sie dient der Auswahl von Datensätze, die bei der Validierung einer Klassifikation nicht betrachtet werden sollen und wird nur dann angezeigt, wenn ein Algorithmus zur Ausführung ausgewählt wurde, der einen entsprechenden Eingabeparameter besitzt. Dies trifft auf ML-Algorithmen zu. Der Aufbau dieser Seite gleicht ansonsten dem Aufbau der vorherigen Seite. Über eine CheckList-Komponente können die Datensätze ausgewählt werden, die bei der Validierung der Klassifikation nicht betrachtet werden sollen.

Die letzte Seite ermöglicht die Einstellung der Eingabeparameter des ausgewählten Algorithmus. Der Wizard mit dieser Seite zeigt Abbildung 51. Da die implementierten Algorithmen unterschiedliche Parameter besitzen, muss diese Seite generisch in Abhängigkeit vom ausgewählten Algorithmus erstellt werden. Die Seite besitzt ein TableView-Objekt mit 2 Spalten zur Anzeige der Parameternamen und der Parameterwerte des ausgewählten Algorithmus. Für den Wert des Parameters wird abhängig von seinem Datentyp eine entsprechende Komponente erzeugt, die für den Benutzer eine möglichst einfache Methode zur Bearbeitung des Wertes bieten soll. Daher wird der Inhalt einer Zelle des TableView-Objektes zur Anzeige eines Parameterwertes mithilfe eines ParameterValueTableCell-Objektes erzeugt. Dieses erzeugt in Abhängigkeit des Datentyps des Parameters die entsprechenden GUI-Komponenten zur Bearbeitung des Wertes in der entsprechenden Zelle des TableView-Objektes. Für ein Parameter vom Datentyp Boolean wird beispielsweise in der Zelle zur Bearbeitung seines Wertes eine Checkbox-Komponente erzeugt und angezeigt.

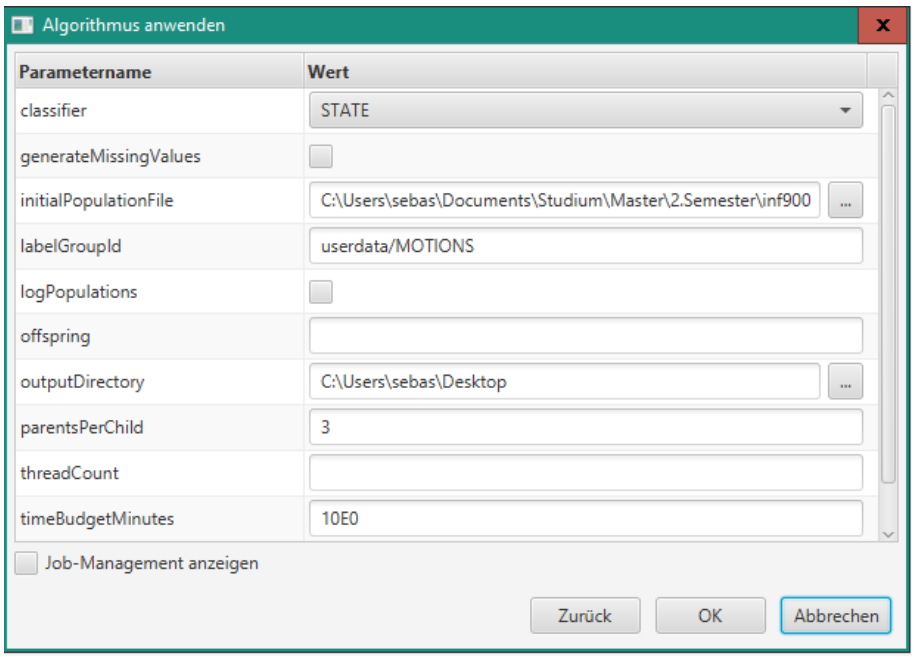


Abbildung 51.: Wizard zur Ausführung eines Algorithmus im Frontend

Außerdem besteht für einen Eingabeparameter die Möglichkeit, diesem auch mehrfache Werte zuzuweisen. Dazu werden die Eingabeparameter mit allen eingestellten Werten in einem MultiParameterCollection-Objekt gespeichert. Werden einem Eingabeparameter mehrere Werte zugewiesen, so muss der ausgewählte Algorithmus mehrfach, nämlich mit jedem eingestellten Wert, ausgeführt werden.

Sobald der Benutzer auf der letzten Seite des Wizards den OK-Button drückt, wird die executeAlgorithm-Methode des App-Objektes aufgerufen. Die Methode benötigt zunächst eine Liste aller möglichen Wertekombinationen der eingestellten Eingangsparameter, für die der Algorithmus ausgeführt werden soll. Um diese Liste zu erhalten, wird die getCombinations-Methode des MultiParameterCollection-Objektes aufgerufen. Diese liefert eine Liste von ParameterCollection-Objekten zurück. Ein ParameterCollection-Objekt speichert für jeden Parameter einen einzigen Wert und wird einem Algorithmus durch den Aufruf der setInputCollection-Methode vor seiner Ausführung übergeben. Für jedes ParameterCollection-Objekt der Liste wird ein Algorithmus erzeugt und durch Aufruf der submit-Methode des JobManager-Objektes zur Ausführung gebracht.

6.1.8. Job-Management

Wenn der Benutzer einen Algorithmus ausführen möchte, wird zunächst ein Job angelegt. Dieser bündelt alle notwendig Informationen um den Algorithmus auszuführen. Je nachdem, ob der Job auf dem lokalen Rechner oder auf einem Remote Host ausgeführt werden soll, wird der Job an den entsprechenden JobManager übergeben, der die Verwaltung des Jobs übernimmt. Der Benutzer hat die Möglichkeit aktuelle Informationen über alle gestarteten Jobs mit Hilfe der JobManagementView einzusehen.

In den folgenden Kapiteln soll ein detaillierter Einblick in die Implementierung der verschiedenen JobManager, sowie der JobManagementView gegeben werden.

JobManager

JobManager ist ein Interface, das die verschiedenen Schnittstellenoperationen zur Kontrolle von Jobs definiert. Die wichtigsten Funktionen sind:

- submit(Job job): Füge den Job zur Warteschlange hinzu und starte ihn sobald es möglich ist,
- cancel(Job job): Breche die Ausführung des Jobs ab.

Alle konkreten JobManager-Implementierung müssen entweder dieses Interface oder eine Erweiterung davon implementieren.

Lokale JobManager

Die Applikation bietet dem Benutzer die Möglichkeit die verschiedenen Algorithmen lokal auf dem eigenen Rechner auszuführen. Das ist besonders für Algorithmen geeignet, die im Allgemeinen eine relativ kurze Ausführungsdauer haben und nicht viel Arbeitsspeicher beanspruchen. Dazu zählen beispielsweise die verschiedenen Rauschunterdrückungsfilter oder der Madgwick-Algorithmus.

Wie in Abschnitt 5.1 beschrieben, erlaubt die Anwendung einerseits die Jobs in dem gleichen Prozess des Frontends abzuarbeiten (LocalThreadPoolJobManager). Durch die Verwendung von Threads blockiert die Ausführung des Algorithmus aber nicht das Frontend, sodass der Benutzer weiterhin Eingaben tätigen kann. Andererseits können Jobs aber auch in eigenständigen Prozessen ausgeführt werden (LocalProcessPoolJobManager).

Beide Klassen implementieren das Interface LocalJobManager, das eine Erweiterung von JobManager ist. Bei Aufruf der zusätzlichen Methode updateJobStatus() sollen die Status aller aktuellen Jobs überprüft werden um bei entsprechenden Änderungen die Listener zu benachrichtigen. Damit eine solche Aktualisierung nicht manuell gestartet werden muss, wurde die Klasse LocalPollingJobManager implementiert, die regelmäßig eine Überprüfung der Status aller Jobs anstößt.

LocalThreadPoolJobManager erlaubt die Ausführung von Algorithmen in der gleichen JVM, in der auch das Frontend läuft. Ein Vorteil dieser Variante ist, dass Debugging sehr leicht möglich ist.

Der LocalThreadPoolJobManager verwendet einen ExecutorService, der die Abarbeitung der verschiedenen Jobs parallelisiert. Zur konkreten Ausführung wird ein Job-Executor-Objekt verwendet (siehe Abschnitt 6.4).

LocalProcessPoolJobManager erlaubt die Ausführung von Algorithmen in einer neuen JVM. Dafür wird von der Applikation ein eigenständiger Prozess erstellt und entsprechend parametrisiert, sodass der Prozess einen gegebenen Job mit Hilfe des Job-Executors ausführt. Die Ausgabeströme des Prozesses werden in spezifizierte Logdateien geschrieben, sodass diese nach Beendigung des Prozesses vom Frontend geladen werden können. Dadurch kann der Benutzer nachvollziehen, wie die Abarbeitung des Jobs abgelaufen und was für Fehler gegebenenfalls aufgetreten sind.

Remote JobManager

Die Applikation soll nicht nur die Möglichkeit bieten, dass Jobs lokal auf dem eigenen Rechner ausgeführt werden können, sondern auch auf einem beliebigen Remote Host. Das ist vor allem sinnvoll um lange Berechnungen mit hohem Ressourcenbedarf auslagern zu können, insbesondere auf ein High-Performance Computing (HPC) System.

Für die Realisierung wurde ein allgemeiner Ansatz entwickelt, der die Übertragung von benötigten Dateien sowie die Ausführung von Befehlen auf einem Remote Host via Secure Shell (SSH) durchführt (MamksRemoteConnection). Einige Operationen sind aber abhängig von dem eingesetzten Job-Scheduler des Remote Hosts. Die Carl von Ossietzky Universität Oldenburg betreibt ein HPC System, das mit dem Job-Scheduler SLURM arbei-

tet, sodass eine entsprechende Implementierung in der Klasse SlurmRemoteConnection vorgenommen wurde. Weitere Remote Hosts mit gegebenenfalls anderen Job-Schedulern können einfach ergänzt werden.

Der Ansatz zur Ausführung von Jobs auf einem Remote Host charakterisiert sich durch bestimmte Schritte, die nacheinander abgearbeitet werden müssen:

1. Zunächst wird ein Job erstellt, der die lokale Ausführung des Algorithmus beschreibt. Dieser Job wird aber nicht ausgeführt sondern als Basis verwendet um einen RemoteJob zu generieren, der die Ausführung auf dem Remote Host beschreibt. Der RemoteJob unterscheidet sich im Wesentlichen nur durch Anpassung aller Dateipfade, die mit der Algorithmusausführung verknüpft sind, da die lokalen Dateipfade nicht für den Remote Host übernommen werden können. Daher werden die lokalen Dateipfade auf Dateipfade für den Remote Host abgebildet. Durch eine Speicherung des lokalen Jobs können die lokalen Dateipfade den remote Dateipfaden eindeutig zugeordnet werden.
2. Es wird ein Skript generiert, das auf dem Remote Host ausgeführt werden soll und die Ausführung des Job-Executors mit dem entsprechenden Job starten soll. Gegebenenfalls können auch Parameter festgelegt werden, beispielsweise wie viele Prozessorkerne für den Job verwendet werden sollen.
3. Alle benötigten Dateien werden auf den Remote Host hochgeladen, sodass der Job ausgeführt werden kann. Zu diesen Dateien zählen
   - der serialisierte Job, der auf dem Remote Host ausgeführt werden soll,
   - die Eingabedateien des auszuführenden Algorithmus,
   - gegebenenfalls die Datensätze, auf denen der Algorithmus operieren sollte, falls diese noch nicht auf dem Remote Host existieren,
   - das Skript, das die Ausführung starten soll.
4. Das Skript wird ausgeführt, sodass der Job auf dem Remote Host gestartet wird.
5. Es wird regelmäßig überprüft, ob die Ausführung des Jobs beendet wurde. Dabei könnte der Job erfolgreich bearbeitet werden oder aber abstürzen.
6. Ist die Ausführung beendet worden, werden die Ergebnisse heruntergeladen. Falls der Job abgestürzt ist handelt es sich dabei nur um die Logdateien, die anschließend vom Frontend geladen werden, sodass der Benutzer Einsicht in den Ausführungsverlauf erhalten kann. War die Algorithmusausführung hingegen ein Erfolg, so werden zusätzlich

alle vom Algorithmus erzeugten Daten heruntergeladen.

7. Die heruntergeladenen Dateien werden an die entsprechenden Stellen kopiert, die vom Benutzer ursprünglich spezifiziert wurden.

Verläuft dieser ganze Prozess ohne Probleme, dann merkt der Benutzer quasi nicht, dass der Job nicht lokal auf dem eigenen Rechner ausgeführt wird. Im Falle eines Fehlers werden entsprechende Exceptions im Log angezeigt.

JobManagementView

Mit Hilfe der JobManagementView soll dem Benutzer eine Übersicht über alle gestarteten Jobs und deren Status gegeben werden.

Das Aussehen der JobManagementView ist in der JobManagementView.fxml-Datei definiert und die Funktionalität in der JobManagementController-Klasse realisiert. Dieser implementiert das JobListener-Interface, das die Methoden jobChanged(Job job), jobAdded(Job job) und jobRemoved(Job job) deklariert, die jedes mal dann aufgerufen werden, wenn sich ein Attribut eines Jobs ändert, ein Job hinzugefügt wird beziehungsweise ein Job entfernt wird. Somit wird die JobManagementView immer auf den aktuellsten Stand gehalten.

Da Algorithmen generell lokal oder auf einem Remote Host ausgeführt werden können, besitzt der JobManagementController einen LocalJobManager-Objekt und einen RemoteJobManager-Objekt. Diese speichern Informationen über jeden Algorithmus, der lokal beziehungsweise auf einem Remote Host ausgeführt wird, wie z. B. seinen Namen, den Startzeitpunkt seiner Ausführung und seinen aktuellen Zustand.

Die JobManagementView besteht aus einem TableView-Objekt, das Spalten zur Anzeige der Namen der Algorithmen, der Startzeitpunkte sowie der Endzeitpunkte, der Hostnamen, der aktuellen Zustände der Algorithmen und der Abbrechen-Buttons bereitstellt. Die Algorithmen, die derzeit im LocalJobManager-Objekt oder im RemoteJobManager-Objekt gespeichert sind, werden in einem ObservableList<Job>-Objekt eingetragen und dieses zur Anzeige an das TableView-Objekt übergeben. Für Algorithmen, die auf einen Remote Host ausgeführt werden, wird zusätzlich ein Tooltip angezeigt, wenn sich der Mauszeiger über der entsprechende Zeile im TableView-Objekt befindet. Dieser zeigt die Ausführungsparameter des Jobs an (z. B. Anzahl eingestellter CPUs, eingestellter RAM oder die eingestellte maximale Ausführungsdauer).

Wird der Abbrechen-Button eines Algorithmus gedrückt, so wird die Ausführung des Algorithmus zum Abbruch aufgefordert. Dabei wird die onJobCancelButtonClicked(Job job)-Methode des JobManagementControllers aufgerufen, welcher wiederum die cancel(Job job)-Methode des LocalJobManager-Objektes oder des RemoteJobManager-Objektes aufruft und den Job somit zum Abbruch zwingt.

6.1.9. Report-Dialoge

Einige Algorithmen erzeugen Reports, deren Inhalte mit Hilfe der hier beschriebenen Report-Dialog-Realisierung dem Nutzer nach der Algorithmusausführung visualisiert werden können. Die Realisierung der Reports ist in Unterabschnitt 6.3.10 beschrieben.

Ein Algorithmus kann ein Objekt einer Implementierung des Report-Interfaces als Parameter an seine Ausgabeparametersammlung übergeben, in diesem Fall wird im Anschluss an die Algorithmusausführung vom JobEventListener, die Methode MainApplication#showReport(Report) aufgerufen, welche den Report-Dialog anzeigt (JobEventListener und MainApplication sind im Projekt mamks-frontend im Paket mamks.frontend realisiert). Handelt es sich bei der Report-Implementierung um die Klasse GaitParameterReport oder LabelGroupEvaluationReport wird ein spezieller Reportdialog angezeigt, andernfalls wird ein Standarddialog angezeigt.

Die Report-Dialoge sind im Projekt mamks-frontend im Paket mamks.reportview realisiert. Die abstrakte Implementierung AbstractReportDialog stellt einige grundlegende Funktionen zur Verfügung und ist von anderen Report-Dialog-Implementierungen zu erweitern. In einer abgeleiteten Implementierung des AbstractReportDialog muss die Methode #setReport(Report) erweitert werden. Wird in dieser Methode die Super-Methode (also die Implementierung von AbstractReportDialog) verwendet, dann fügt dieser automatisch die Textfelder für Eingabe- und Ausgabeparametersammlungen des Reports, sowie die Schalter für das Schließen des Dialogs und das Speichern der angezeigten Report-Inhalte an. Diese Elemente werden vom AbstractReportDialog gesteuert, hierbei ist zu beachten, dass das Wurzelelement des jeweiligen Dialogs mit der fx:id root versehen ist, damit der AbstractReportDialog auf diese zugreifen kann.

Es gibt drei konkrete Implementierungen der abstrakten Klasse AbstractReportDialog:

- DefaultReportDialog: Dieser verfügt über ein Textfeld, in dem die Inhalte des Reports über dessen Methode #getContentAsSimpleText() als einfache Textrepräsentation

dargestellt werden.

- GaitParamReportDialog: Dieser verfügt über ein Textfeld, in dem die Gangparameter angezeigt werden. Der Vorteil ist, dass im Gegensatz zur DefaultReportDialog-Implementierung die Internationalisierung genutzt werden kann, um den Text in der richtigen Sprache anzuzeigen.
- LabelGroupEvaluationReportDialog: Dieser verfügt über zwei Textfelder, im oberen wird die Information zu den ausgewählten Datensätzen und der abgeleiteten Gesamtdauer dargestellt, im unteren werden die Evaluationsergebnisse angezeigt. Wie beim GaitParamReportDialog kann auch dieser Dialog von der Internationalisierung Gebrauch machen.

Für genauere Informationen zu den jeweiligen Algorithmen siehe Abschnitt 5.7.

6.2. Kommandozeilenschnittstelle

Das Teilprojekt, welches die Kommandozeilenschnittstelle realisiert, besteht hauptsächlich aus Code zum Parsen der Kommandozeilenparameter und nutzt dann den JobExecutor zur Ausführung des aufgerufenen Befehls. Weiterhin gibt es einige Dateiverwaltungsbefehle, die nur in der CLI implementiert sind.

Das Parsing wird durch die Bibliothek JCommander umgesetzt. Für jeden Befehl existiert im Package mamks.cli.commands eine Klasse, welche die notwendigen und optionalen Parameter spezifiziert. Der Parser erstellt aus den Kommandozeilenparametern das korrekte Command-Objekt. Weiterhin wird ein Environment-Objekt erstellt, das den Inhalt einiger Umgebungsvariablen speichert. Diese können als Ersatz für einige Befehlsparameter dienen. Diese beiden Objekte werden dann an den zugehörigen CommandRunner-Objekt übergeben, welches den Befehl ausführt. Üblicherweise geschieht dies, indem Code aus dem shared-Projekt aufgerufen wird.

Um bestimmte, wiederkehrende Kommandozeilenparameter wie z. B. Motion-Data-Ids zu parsen, gibt es die StringValueFactory. Sie stellt Methoden bereit, um Strings in die korrekten Objekte umzuwandeln.

Eine Sonderstellung nimmt der Befehl plot-features ein, da mit ihm eine minimalistische grafische Oberfläche geöffnet werden kann, die Plots anzeigt. Die Implementierung

befindet sich in der Klasse PlotFrame.

6.3. Shared

In Shared wurde Quellcode realisiert, der sowohl vom Frontend, der Kommandozeilenschnittstelle und dem Job-Executor genutzt wird. Dieser Quellcode umfasst die Datenstruktur der Anwendung, die Klassen zur Darstellung der Domänendaten bereitstellt sowie einen DataStorage, der die Originaldaten des Sensorgürtels in für die Anwendung lesbare XML-Dateien im Binärformat konvertiert und im Arbeitsverzeichnis ablegt. Außerdem wurden in Shared die Algorithmen realisiert, die über das Frontend oder die Kommandozeilenschnittstelle im Job-Executor ausgeführt werden können. Dazu zählen insbesondere die Machine-Learning-Algorithmen, welche in diesem Kapitel ebenso wie die dazugehörigen Strukturen wie das Sliding-Window-Verfahren und die in Unterabschnitt 3.4.3 beschriebenen Merkmale erläutert werden. Des Weiteren wurden Funktionen für den Export von Merkmalen in ein CSV-Dateiformat realisiert, damit diese unter Verwendung einer geeigneten Anwendung (z. B. *R* oder *MATLAB*) visualisiert und analysiert werden können.

In den folgenden Abschnitten wird die Realisierung der Datenstruktur, des DataStorage, der Algorithmen, des Sliding-Window-Verfahrens, der Merkmale sowie des Merkmalsexports beschrieben.

6.3.1. Datenstruktur

Die Datenstruktur zur Darstellung der Domänendaten muss zahlreiche Informationen enthalten.

Offensichtlich müssen die Messwerte des Sensorgürtels gespeichert werden. Diese bestehen aus mehreren Datenreihen der verschiedenen Sensoren. Weiterhin müssen Metainformationen wie der Aufnahmezeitraum, die Anzahl der Messwerte und die Zeitdauer zwischen zwei aufeinanderfolgenden Messwerten zur Verfügung stehen. Das gesamte Modell ist in Abbildung 52 zu sehen (siehe Abschnitt 4.2).

Diese Informationen werden in der Klasse MotionData gebündelt. Diese verfügt jeweils über ein Feld für die MotionDataId, für eine Sammlung von DataSet-Instanzen, für das

TimeInterval, für die Anzahl an Messwerten innerhalb einer Datenreihe, für den zeitlichen Abstand zweier aufeinanderfolgenden Messwerte und für eine Sammlung an MotionData-Listener-Instanzen.

Abbildung 52.: Grundlegende Datenstruktur

In der MotionDataId wird sowohl eine eindeutige ID, als auch der zugehörige Proband der Aufnahme in Form einer ProbandId gespeichert. Eine ProbandId speichert wiederum die eindeutige ID des Probanden, sowie eine beliebige Anzahl an MotionDataId-Instanzen, da einem Probanden mehrere Bewegungsdaten zugeordnet werden können z. B. die unterschiedlichen Assessment-Aufnahmen des Probanden.

Der Aufnahmezeitraum wird mithilfe der Klasse TimeInterval beschrieben, die einen Start- und Endzeitpunkt speichert. Der Start- und Endzeitpunkt wird jeweils durch ein ZonedDateTime-Objekt aus dem Java-Package java.time repräsentiert.

Die im MotionData-Objekt registrierten MotionDataListener werden bei Änderung am MotionData-Objekt alarmiert, damit sie reagieren können. Dies findet beispielsweise in der GUI Verwendung um die Darstellung der GUI entsprechend der Änderungen zu aktualisieren.

Die Klasse DataSet ist eine logische Gruppierung von Daten um ihre Datenquellen zu differenzieren. Dabei werden die Rohdaten des Sensorgürtels, vom Benutzer manuell erzeugte Daten und die durch Algorithmusausführung erzeugte Daten unterschieden. Hierfür speichert ein DataSet einen Namen, wobei dieser für die Rohdaten des Sensorgürtels immer „sensorbelt“ lautet, während manuell erzeugte Daten durch „userdata“ bezeichnet werden. Jede Algorithmusausführung die entweder Labelinformationen oder Datenreihen erzeugt, verfügt über ein DataSet in dem diese Informationen und weitere Informationen gespeichert werden. Zu diesen weiteren Informationen zählen der DataSet-Name, der sich aus dem Namen des Algorithmus sowie seinem Ausführungszeitpunkt zusammen setzt, und die Ein- und Ausgabeparameter von Algorithmen, diese können im DataSet als ParameterCollection abgespeichert werden. Die ParameterCollection-Attribute sind optional, da das Sensorgütel-DataSet oder das Benutzer-DataSet nicht über Parameter verfügen. Die Datenreihen eines DataSet werden in einer Sammlung von DataSeries gespeichert, die Labelinformationen werden in Mengen von MotionLabel-Objekten gespeichert, die nach Labelgruppen gruppiert sind. Die zugehörige Labelgruppe wird mithilfe einer LabelGroupId angegeben.

In der Klasse DataSeries wird eine Zeitreihe von Messwerten gespeichert. Hierfür liegen die Messwerte in einem DoubleList-Objekt aus der Fastutil-Bibliothek vor, die intern durch Arrays implementiert ist. Dadurch soll eine schnelle Zugriffsgeschwindigkeit erzielt werden. Weitere Informationen zu Fastutil und anderen Primitive-Collection-Bibliotheken sind in Unterabschnitt 5.2.2 beschrieben. Zusätzlich enthält ein DataSeries-Objekt eine DataSeriesId, die aus dem Namen des DataSet, einem Datenreihen-Namen und der Messeinheit der Datenreihe besteht. Für die vom Sensorgürtel erzeugten Messreihen sind die DataSeriesId bereits vordefiniert.

Um eine Klassifikation der Bewegungsdaten vornehmen zu können, sollen Zeitintervalle

innerhalb des Aufnahmezeitraums durch Label beschriftet werden. Hierfür enthält die Klasse MotionLabel ein Interval und einen MotionType. Das Interval-Objekt speichert den Start- und Endindex der Messwerte. Mithilfe der indexToTime(Interval interval)-Methode kann dies allerdings in ein TimeInterval-Objekt konvertiert werden, welches das Intervall per Start- und Endzeit beschreibt. Sowohl der Startindex und die Startzeit sind dabei inklusive, während der Endindex und die Endzeit exklusive sind. Das MotionType-Objekt speichert den Typ der Beschriftung (z. B. ein Assessment wie „TUG“ oder eine Aktivität wie „Gehen“). Die MotionLabel-Objekte werden im DataSet in der Form Map<LabelGroupId, Set<MotionLabel>> gespeichert, damit jedem MotionLabel seine zugehörige Labelgruppe zugeordnet werden kann. Die Intervalle von MotionLabel einer Labelgruppe dürfen sich nicht überschneiden, sonst wird beim Hinzufügen eines MotionLabel eine LabelIntersectsException geworfen.

6.3.2. Datenverwaltung/-eingabe/-ausgabe

In den folgenden Abschnitten wird beschrieben, wie die Eingabe, die Verwaltung und die Ausgabe von Daten durch die Anwendung realisiert ist. Jeder Abschnitt befasst sich mit einem dieser Punkte. Zusätzlich ist in Abschnitt 6.3.2 die Ausgabe von Merkmalen beschrieben. Alle beschriebenen Realisierungen sind als Teile des Unterprojektes mamks-shared realisiert, damit sie sowohl der CLI als auch der GUI zur Verfügung stehen.

Datenverwaltung

Die Datenverwaltung ist in der Klasse DataStorage im Paket mamks.datastorage, sowie in den Klassen der zugehörigen Unterpakete mamks.datastorage.io, mamks.datastorage.io.metadata und weiteren realisiert. Das Konzept der Datenverwaltung ist ausführlich in Unterabschnitt 5.2.3 beschrieben, kurz zusammengefasst: Die Datenreihen werden in Binärdateien, alle anderen Informationen wie Label, Metainformationen und Algorithmusausführungs-Informationen werden in XML-Dateien verwaltet. Die Datenverwaltung nutzt hierfür eine eigene Verzeichnishierarchie von Probanden-, Bewegungsdaten- und Datensatzverzeichnissen.

Das Schreiben und Lesen der Binärdateien ist in der Klasse BinaryFloatIo im Paket mamks.datastorage.io implementiert. Im selben Paket sind zudem XmlLabelIo (für das Schreiben und Lesen von Labels), XmlMetadataIo (für das Schreiben und Lesen von

Metadaten) und XmlAlgorithmExecutionIo (für das Schreiben und Lesen von Algorithmusausführungs-Informationen) realisiert.

In den Unterpaketen von mamks.datastorage.io sind die Typ-Klassen für das Schreiben der weiteren Informationen mit Java Architecture for XML Binding (JAXB), Exception-Klassen, sowie einige weitere Hilfsklassen realisiert. Auf diese wird hier allerdings nicht im Detail eingegangen, da es sich dabei um keine wesentlichen Klassen handelt.

Die in der Klasse DataStorage realisierte Datenverwaltung verwendet die oberen Lese- und Schreibrealisierungen, um die Gesamtheit von Probanden-Daten zu lesen und zu schreiben, zudem verfügt sie über Methoden, um die benötigte Verzeichnisstruktur herzustellen und, um zu validieren, dass das angegebene Arbeitsverzeichnis über die korrekte Struktur verfügt.

Für genauere Informationen zu den Lese- und Schreibdateiformaten siehe Unterabschnitt 5.4.2.

Import

Die Anwendung erlaubt die Eingabe von Gürteldaten in dem in Unterabschnitt 5.4.1 beschriebenen Format. Diese Funktion ist in der Klasse Importer im Paket mamks.importer beziehungsweise in der vom Importer genutzten Klasse SensorBeltImport realisiert. Während in SensorBeltImport das eigentliche Einlesen der Daten der Importdatei in die Anwendung implementiert ist, sind im Importer nützliche Methoden realisiert, die es zum Beispiel erlauben die Probanden-ID aus dem Ordnername abzuleiten oder es ermöglichen, dass unterschiedliche Importquellen (direkt vom Gürtel exportierte Verzeichnisse, Verzeichnisse in der VERSA-Verzeichnishierarchie) erlaubt werden. Der Importer erlaubt zudem für Verzeichnisse in Probanden-Superordner-Hierarchie die Wahl des Imports der Daheim-/Studienraum-Aufnahme.

Export

Die Anwendung erlaubt die Ausgabe der Datenreihen in eine CSV-Datei und die Ausgabe von Labels in eine CSV-Datei. Für genauere Informationen zu den Exportdateiformaten siehe Unterabschnitt 5.4.3. Hierzu wurde der Exporter im Paket mamks.exporter implementiert, welcher das Schreiben von Datenreihen und Labels in Dateien im angegeben

Format ermöglicht. Zur Vereinfachung wird hierfür die org.apache.commons.csv-Library verwendet.

Merkmalsexport

Neben dem Export von Datenreihen und Labels ermöglicht die Anwendung auch den Export von Merkmalen. Hierzu dient der FeatureVisualization-Algorithmus, welcher im Paket mamks.featurevisualization realisiert wurde. Momentan kann die Wahl der zu exportierenden Merkmale nur codeseitig getroffen werden, indem der Wert des Feldes SLIDING_WINDOW_SETUP des Algorithmus als die zu nutzende Implementierung der abstrakten Klasse AbstractSlidingWindowSetup angegeben wird. Aktuell stehen die konkreten Implementierungen SlidingWindowSetupAllFeatures, SlidingWindowSetupAlternativeFeatures, SlidingWindowSetupGyroFeatures und SlidingWindowSetupSomeFeatures zur Verfügung.

Detaillierter soll hier auf den Merkmalsexport nicht eingegangen werden, da das Thema ausführlich in der Ausarbeitung in Unterabschnitt D.5 beschrieben ist, in deren Rahmen der Merkmalsexport realisiert wurde.

6.3.3. Algorithmen

Algorithmen sind in diesem Projekt die Abstraktion über alle möglichen Operationen auf den Bewegungsdatensätzen. Sie werden genutzt, um Daten zu transformieren (z. B. Hochpassfilter, Madgwick-Algorithmus), Machine-Learning-Modelle zu trainieren und mit diesen Klassifizierungen vorzunehmen. Andere Anwendungen sind natürlich möglich.

In Abbildung 53 ist zu sehen, dass alle Algorithmen durch das Interface Algorithm standardisiert sind. Jeder Algorithmus besitzt zwei ParameterSpecification-Objekte, die angeben, welche Parameter der Algorithmus erwartet und zurückliefert sowie von welchem Typ sie sind. Parametern können Standartwerte zugewiesen werden. Das ist hilfreich, um Algorithmen später empirisch erprobte Parameter zuordnen zu können oder um sich bei Algorithmen, die fast immer auf den gleichen Sensordatenreihen ausgeführt werden, das manuelle Einstellen zu sparen. Weiterhin können Parameter als versteckt oder optional markiert werden. Versteckte Parameter werden dem Benutzer nicht angezeigt. Sie sind nützlich für Machine-Learning-Algorithmen, um ihnen bereits angelernte Modell-Objekte zu über-

geben. In einer ParameterCollection können die Werte zu den verschiedenen Parametern einer ParameterSpecification abgespeichert werden. Ist für einen Parameter sowohl ein Wert in der ParameterCollection als auch ein Standardwert in der ParameterSpecification gegeben, so wird der Wert aus der ParameterCollection bevorzugt.

Die abstrakte Klasse AbstractAlgorithm implementiert die Schnittstellenoperationen von Algorithm und Validierungsaufgaben, die sonst jede Algorithmus-Implementierung selbst durchführen müsste. Es werden mehrere abstrakte Methoden definiert, die von konkreten Algorithmen implementiert werden müssen. Die abstrakten Methoden configureInputSpecification und configureOutputSpecification bekommen ParameterSpecification-Objekte übergeben, durch die deklariert werden kann, welche Parameter erwartet und zurückgeliefert werden.

Die wichtigsten Methoden sind allerdings before, execute und after. Jeder Algorithmus wird durch den JobExecutor ausgeführt. Dieser erhält ein Job-Objekt mit einer Beschreibung der zu erledigenden Aufgabe. Insbesondere ist eine Liste von MotionDataIds enthalten auf denen der Algorithmus arbeiten soll. Der Ablauf ist als Pseudocode in Listing 6.1 dargestellt. Das dargestellte vorgehen verhindert, dass alle Daten auf einmal im Arbeitsspeicher gehalten werden müssen. Das hat allerdings zur Folge, dass alle Lernalgorithmen *online* arbeiten müssen, das heißt sie können iterativ mit Daten trainiert werden.

Listing 6.1: Ausführen eines Jobs im JobExecutor

```
algorithm ← job.getAlgorithm()
ids ← job.getMotionDataIds()
algorithm.before()
for id ∈ ids:
        motionData ← dataStorage.load(id)
        dataSet ← algorithm.execute(motionData)
        dataStorage.save(dataSet)
algorithm.after()
```

Die before-Methode kann vom Algorithmus zur Vorbereitung der Berechnung genutzt werden. ML-Algorithmen können beispielsweise ein bereits trainiertes Modell laden, um es weiter zu trainieren. Die execute-Methode enthält die tatsächliche Berechnung und liefert ihre (optionalen) Ergebnisse als DataSet zurück. Der ML-Trainingsalgorithmus wird üblicherweise kein DataSet zurückliefern, da das Ergebnis ein neues Modell ist und als Ausgabeparameter zurückgeliefert wird. Die after-Methode kann für abschließende Schritte wie das Ablegen eines gelernten ML-Modells in den Ausgabeparametern genutzt werden.

Weiterhin ist es durch die after-Methode möglich auch *offline* Lernalgorithmen zu nutzen. Es müssten dann alle Daten die execute übergeben bekommen, gesammelt werden und das tatsächliche Lernen würde in after stattfinden. Natürlich ist dies nur bei Datenmengen möglich, deren vollständiger Umfang im Arbeitsspeicher gehalten werden kann.

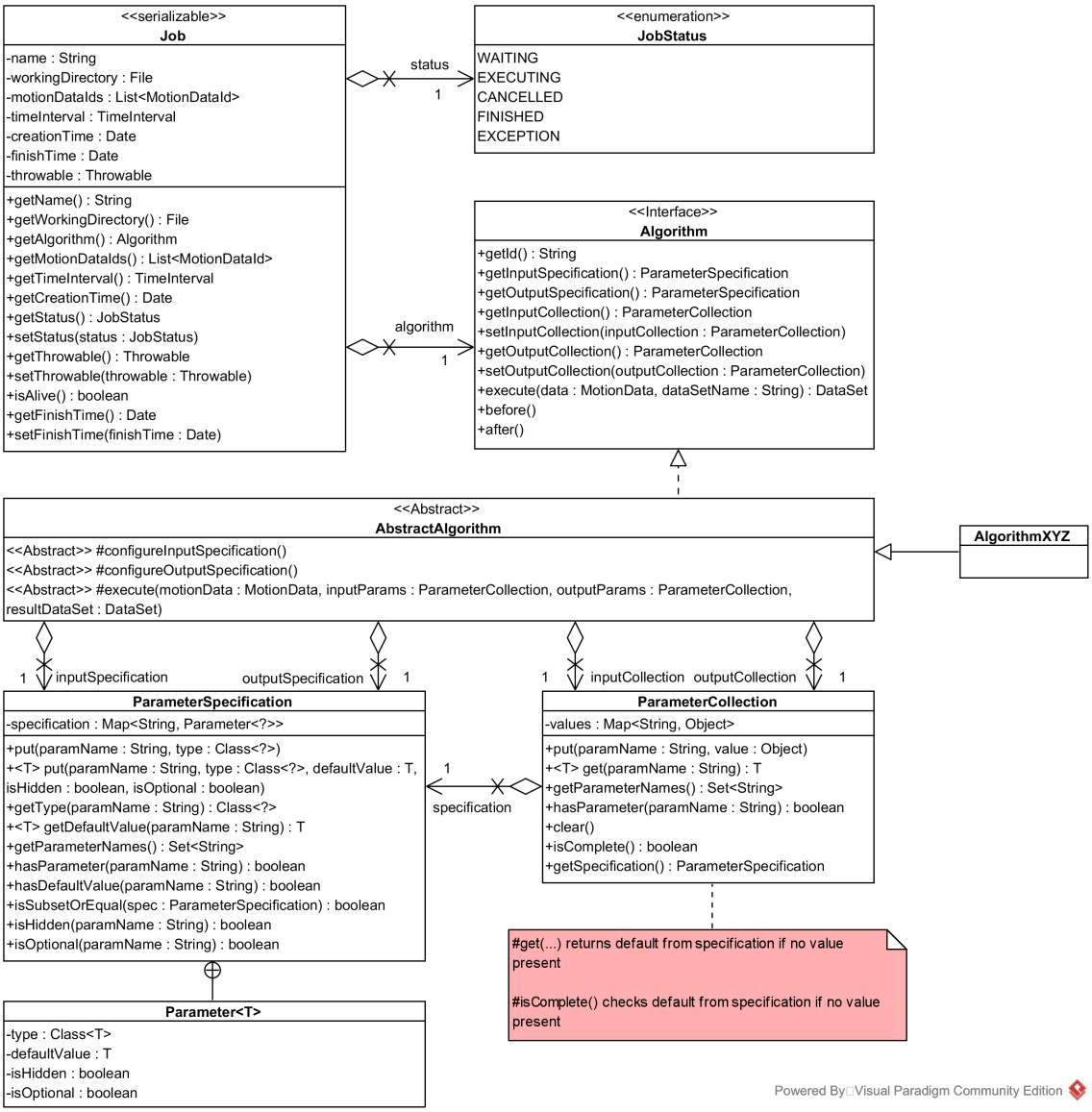


Abbildung 53.: Jobs und Algorithmen

6.3.4. Filter

Im Paket mamks.algorithm.filters wurden Filter realisiert, die eine Rauschunterdrückung in den Datenreihen erlauben und z. B. zur Vorverarbeitung genutzt werden können. Es wurde das Gauß-, Tiefpass-, Hochpass- und Median-Filter implementiert. Die genauen Funktionen der Filter sind in Abschnitt 3.4.1 beschrieben. Alle Filter implementieren

das Interface Filter, welches die Methodendeklaration DoubleList apply(DoubleList values) beinhaltet und das Anwenden eines Filters auf eine DoubleList ermöglicht. In jeder Filter-Klasse liefert diese Methode das Ergebnis für die gesamte eingegebene Liste als DoubleList zurück. In der Klasse GaussianFilter gibt es zusätzlich noch eine überladene apply-Methode, die einen Start- und Endindex als Parameter erwartet und den Filter nur auf die Elemente der übergebenen DoubleList anwendet, die zwischen den beiden Indizes liegen. Zu jeder Filter-Klasse existiert eine Algorithm-Klasse, die von AbstractAlgorithm (siehe Unterabschnitt 6.3.3) erbt und jedes Filter somit als eigenständiger Algorithmus in der Anwendung ausgeführt werden kann.

6.3.5. Sliding Window und Feature Calculator

Die realisierte Umsetzung des maschinellen Lernens nutzt nicht direkt die Datenreihen sondern Merkmale, die per Schiebefenster, welches über die Datenreihen „geschoben“ wird, berechnet werden, als Eingabe. Der realisierte Schiebefenster-Ansatz ist in Unterabschnitt 3.4.2 beschrieben. Die Umsetzung des Schiebefensters ist durch die Klasse SlidingWindow gegeben, die Merkmale werden als Klassen realisiert, welche das Interface FeatureCalculator erweitern. Die Klassen sind im Paket mamks.algorithm.features zu finden.

Das Schiebefenster verfügt über die Parameter length, welcher die Größe des Fensters in Index-Schritten angibt, und step, welche den Versatz des Schiebefensters in Index-Schritten, um den das Schiebefenster bei jedem Schritt „verschoben“ wird, beschreibt. Diese werden bei der Erzeugung des Schiebefenster per Konstruktor gesetzt. Durch die Methode addFeature(FeatureCalculator featureCalculator) können dem Schiebefenster beliebig viele Merkmale hinzugefügt werden, welche beliebige numerische Ausgaben haben können und auf beliebigen Datenreihen operieren können. Das Schiebefenster ist als Iterator implementiert und kann per Methode next() einen Schritt „weitergeschoben“ werden. Die Ausgabe dieser Methode ist dabei gerade der Merkmalsausprägungsvektor des aktuellen Schrittes, welcher als Array des primitiven Typen double ausgegeben wird. Dieser Vektor ergibt sich dadurch, dass von allen Merkmalen, die dem Schiebefenster übergebe wurden, die aktuellen Ausprägungen berechnet und in diesem Vektor aggregiert werden. Die Merkmalsausprägungen werden in der Reihenfolge, in der die zugehörigen Merkmale dem Schiebefenster übergeben wurden, aggregiert.

Damit die Merkmale auf diese Weise vom Schiebefenster genutzt werden können, müssen sie das Interface FeatureCalculator erweitern, und die Methoden get(int startIndex, int endIndex) : double[], die die Merkmalsausprägung des angegeben Index-Intervalls zurückgibt, die Methode getFeatureLength(), die die Größe des Merkmalsausprägungsvektors zurückgibt (bei 1-dimensionalen Merkmalen wäre diese z. B. 1), und die Methode getDataLength(), die die Anzahl an Datenpunkten in den Daten, die dem Merkmal übergeben wurden, zurückgibt, überschreiben. Ein Beispiel einer solchen Realisierung wäre der MeanFeatureCalculator der den Mittelwert als Merkmal berechnet.

6.3.6. Dimensionsreduktion

Im Rahmen des Projekts wurde nur die Hauptkomponentenanalyse als Dimensionsreduktionsmethode verwendet, aber die Struktur des Codes wurde flexibel angelegt, sodass weitere Methoden einfach implementiert werden können. Dimensionsreduktionsmethoden müssen üblicherweise auf einem Datensatz trainiert werden, wodurch eine Transformationsfunktion entsteht, welche auf alle Daten angewandt wird, um ihre Dimensionalität zu verringern. Um diese Anwendungsfälle zu unterstützen wurde das Interface DimensionalityReductionMethod geschaffen. Alle Implementierungen müssen Serializable sein, sodass sie zusammen mit dem trainierten Machine-Learning-Modell gespeichert werden können. Das ist notwendig, damit bei der Klassifikation von neuen Daten dieselbe Datentransformation vorgenommen werden kann wie beim Training. Die Implementierung PrincipalComponentAnalysis speichert beispielsweise eine Projektionsmatrix ab.

6.3.7. Sampling

Um ein relativ ausgewogenes Gleichgewicht der Anzahl an Trainingsbeispielen für die verschiedenen Aktivitätsklassen zu erhalten, wurden in Unterabschnitt 3.4.4 die Techniken Undersampling und Oversampling erläutert. Die Realisierung beider Techniken ist in dem Package mamks.algorithm.ml.sampling zu finden.

Undersampling wurde simpel implementiert, indem für jedes Trainingsbeispiel per Zufall entschieden wird, ob es verworfen werden soll oder nicht. Es ist dabei einstellbar, wie wahrscheinlich es ist, dass ein Trainingsbeispiel verworfen wird.

SMOTE nutzt den k-Nearest-Neighbor-Algorithmus um ausgehend von einer Menge von

Trainingsbeispielen weitere Datenpunkte zu generieren. Ausgehend von einem gegebenen Trainingsbeispiel werden dessen k nächste Nachbarn in der Trainingsmenge bestimmt. Durch eine zufallsbasierte Kombination der Vektoreinträge des aktuell betrachteten Trainingsbeispiels und einem seiner nächsten Nachbarn kann ein neues Trainingsbeispiel erzeugt werden. Die Anzahl der zu erzeugenden synthetischen Datenpunkten kann beliebig eingestellt werden. Zur Lösung des k-Nearest-Neighbor-Problems wird auf die Java-Bibliothek SMILE zurückgegriffen, die verschiedene Lösungsmethoden bietet. Je nach Menge der Datenpunkte wird entweder ein K-D-Tree oder Locally Sensitive Hashing eingesetzt. Letzteres ist zwar nur ein approximatives Verfahren, aber ab einer gewissen Menge an Datenpunkten sind exakte Verfahren nicht mehr mit vertretbarem Zeitaufwand ausführbar.

Undersampling und SMOTE erwarten beide einen Parameter, der die Stärke des Samplings definiert. Es wäre sehr mühsam, wenn diese Parameter manuell eingestellt werden müssten, da sie abhängig von der Anzahl an Trainingsbeispielen pro Klasse sind. Diese Anzahl kann aber je nach Menge an verwendeten Datensätzen zur Generierung der Trainingsbeispiele stark schwanken. Aus diesem Grund wurde die Klasse DynamicSampling implementiert, die die Verteilung der Trainingsbeispiele in den jeweiligen Klassen betrachtet und dann entsprechende Klassen over- oder undersampled. Hierbei wird ein Zielwert an Trainingsbeispielen pro Klasse angegeben, der durch das DynamicSampling erreicht werden soll. Falls eine Klasse sehr stark unterrepräsentiert sein sollte, so müsste sie sehr stark oversampled werden. Dies ist tendenziell eher schlecht, da so das Training auf überwiegend künstlichen Daten stattfinden würde. Daher wird dieser Zielwert entsprechend nach unten korrigiert, falls eine Klasse zu stark unterrepräsentiert sein sollte. So reduziert sich die Stärke des oversamplings und die Klassenbalance kann trotzdem annähernd gewährleistet werden.

6.3.8. Merkmalsvisualisierung

Die Merkmalsvisualierung ist durch den Algorithmus FeatureVisualization im Paket mamks.featurevisualization im Projekt mamks-shared umgesetzt (die weiteren beschriebenen Klassen sind im selben Paket realisiert). Dieser Algorithmus ermöglicht zum einen den Export von Merkmalen (siehe Abschnitt 6.3.2) und optional deren Visualisierung in Form einer Streudiagrammmatrix, die in eine Bilddatei gezeichnet wird. Die Streudiagrammmatrix ist innerhalb der Klasse ScatterPlotMatrix implementiert. Die Auswahl der Merkmale

ist momentan nur direkt im Programmcode möglich, hierzu dient die abstrakte Klasse AbstractSlidingWindowSetup mit den konkreten Implementierungen SlidingWindowSetupAllFeatures, SlidingWindowSetupAlternativeFeatures, SlidingWindowSetupGyroFeatures und SlidingWindowSetupSomeFeatures. Die verwendete Instanz ist als Wert des Feldes SLIDING_WINDOW_SETUP im Algorithmus FeatureVisualization zu setzen.

Die Merkmalsvisualierung wurde im Rahmen einer Ausarbeitung implementiert, in dieser ist die Visualisierung ausführlicher beschrieben. Siehe dazu Unterabschnitt D.5.

6.3.9. Schritterkennung und Bestimmung abgeleiteter Gangparameter

Der Algorithmus zur Erkennung von Schritten aus bereits erkannten „Gehen“-Segmenten ist konzeptionell in Unterabschnitt D.6 beschrieben. Er wurde in der Anwendung als ausführbarer Algorithmus in der StepDetectionAlgorithm-Klasse implementiert. Die Eingangsparameter des Algorithmus wurden bereits in Unterabschnitt 5.7.12 in Abschnitt 5.7.12 aufgelistet und beschrieben.

Zunächst wird eine Liste aller Zeitintervalle erzeugt, die in der Labelgruppe mit der eingegebenen labelGroupId mit dem „Gehen“-Label markiert wurden. Anschließend wird die Schritterkennung für jedes dieser „Gehen“-Segmente einzeln nacheinander durchgeführt. Dabei wird die Wahrscheinlichkeit für den Fersenauftritt am Boden zu jedem Zeitpunkt im „Gehen“-Segment berechnet. Diese Wahrscheinlichkeit beruht auf der Dichte der Extrempunkte und der Energie, die jeweils über den Zeitreihen der drei Achsen des Beschleunigungssensors berechnet werden.

Um die Dichte der Extrempunkte im „Gehen“-Segment zu bestimmen, müssen zunächst Extrempunkte in den Zeitreihen des Beschleunigungssensors erkannt werden. Die Erkennung von Extrempunkten in den Zeitreihen wird mithilfe der detect-Methode der PeakDetection-Klasse durchgeführt. Damit die Dichte der Extrempunkte zuverlässig bestimmt werden kann, werden die Zeitreihen des Beschleunigungssensors vor der Erkennung der Extrempunkte mit verschiedenen Sigmas (Glättungslevel) mithilfe eines Gaußfilters geglättet. Das Gaußfilter wurde in der GaussianFilter-Klasse realisiert. Nach der Glättung werden die Extrempunkte jeweils für alle Zeitreihen und alle Sigmas berechnet. Die Ausgabe ist anschließend eine neue Zeitreihe, die zu jedem Zeitpunkt des „Gehen“-Segments die Anzahl der erkannten Extrempunkte aus den geglätteten Zeitreihen angibt.

Anschließend wird der zweite Wert, die Energie der drei Zeitreihen des Beschleunigungssensors zu jedem Zeitpunkt im „Gehen“-Segment, berechnet. Die berechnete Energie wird danach ebenfalls mit unterschiedlichen Sigmas geglättet. Die geglätteten Zeitreihen werden miteinander und mit dem eingegebenen Skalierungsfaktor multipliziert.

Die Zeitreihen für beide berechneten Werte werden danach miteinander multipliziert, um eine Zeitreihe für die Wahrscheinlichkeit des Fersenauftritts am Boden zu erhalten. Die lokalen Hochpunkte dieser Zeitreihen sind die vermuteten Zeitpunkte für den Fersenauftritt am Boden und markieren somit den Beginn beziehungsweise das Ende eines Schrittes. Die Zeitpunkte der lokalen Hochpunkte werden berechnet und in einer Liste gespeichert. Abschließend wird eine neue Labelgruppe erzeugt und für jedes Zeitintervall zwischen zwei lokalen Hochpunkte ein „Schritt“-Label erzeugt und in der Labelgruppe gespeichert. Es werden sowohl die Labelgruppe mit den erzeugten „Schritt“-Labels als auch die Zeitreihe des Fersenauftritts am Boden vom Algorithmus ausgegeben.

Der Algorithmus zur Schritterkennung kann auch in Verbindung mit der Bestimmung der abgeleiteten Gangparameter erfolgen. Dafür wurde die GaitParameterAlgorithm-Klasse realisiert, die von der StepDetectionAlgorithm-Klasse erbt und diese erweitert. Wird dieser Algorithmus ausgeführt, wird zuerst die oben beschriebene Schritterkennung durchgeführt und anschließend ein GaitParameterReport-Objekt, das für die erkannten Schritte die in Abschnitt 3.5 beschriebenen Parameter berechnet und speichert, erzeugt und ausgegeben.

Die Berechnung der Parameter minimale, maximale und durchschnittliche Schrittdauer (Step Time) erfolgt dabei so, dass die vom StepDetectionAlgorithm erzeugte Labelgruppe mit „Schritt“-Labels durchlaufen und die minimale Dauer sich als die Dauer des Labels von kürzester Länge und die maximale Dauer sich als Dauer des Labels von längster Länge gemerkt wird. Die durchschnittliche Schrittdauer wird so berechnet, dass die Dauer aller Labels summiert und anschließend durch die Anzahl der Labels geteilt wird, sie ergibt sich also als das arithmetische Mittel über der Dauer der Labels der Labelgruppe. Die Kadenz ergibt sich durch Division einer Sekunde durch die durchschnittliche Schrittdauer und der anschließend Multiplikation mit 60. Sie ergibt sich also als die hochgerechnete Anzahl der Schritte pro einer Minute. Für die Berechnung der Doppelschrittdauer (Stride Time) wird die Labelgruppe mit „Schritt“-Labels erneut durchlaufen. Es wird dann immer die Dauer zweier direkt aufeinanderfolgender Labels (zwei Label, wobei das Intervall des ersten Labels zu dem Zeitpunkt endet, wo das Intervall des zweiten Labels beginnt) berechnet. Über diesen berechneten Werten werden sich das Minimum als die minimale

Doppelschrittdauer und das Maximum als die maximale Doppelschrittdauer gemerkt, zudem werden alle Werte summiert und durch die Anzahl an so berechneten Werten geteilt, um die durchschnittliche Doppelschrittdauer zu erhalten.

6.3.10. Reports

Einige Algorithmen erzeugen Report-Instanzen, welche z. B. durch die Report-Dialoge, die im Projekt mamks-frontend realisiert sind (siehe hierzu Unterabschnitt 6.1.9), dem Nutzer präsentiert werden können. Zu diesem Zweck wurde im Projekt mamks-shared im Paket mamks.report das Report-Interface angelegt. Eine allgemeine, abstrakte Implementierung dieses Interfaces ist durch die AbstractReport-Klasse gegeben. In einem Report können beliebige Inhalte innerhalb der contentMap abgelegt werden, der jeweilige Report stellt Methoden zur Verfügung, die den direkten Zugriff oder den Zugriff per Schlüssel auf die Report-Inhalte erlauben. Zudem verfügt jeder Report über eine Eingabe- und eine Ausgabeparametersammlung. Diese sind in der Klasse ParameterCollection im Paket mamks.algorithm implementiert. Eine solche Parametersammlung enthält die Eingabeparameter, die zur Erzeugung des Reports verwendet wurden, beziehungsweise die Ausgabeparameter, die mit den im Report gespeicherten Informationen verknüpft sind. Aktuell werden Report-Objekte nur während der Ausführung von Algorithmen erzeugt. Die Parametersammlungen des Reports werden in diesem Fall als die Parametersammlungen des jeweiligen Algorithmus gesetzt, der zur Erzeugung des Reports geführt hat. Es gibt zum derzeitigen Zeitpunkt vier konkrete Report-Implementierungen, diese sind jeweils im Paket mamks.report zu finden. Für eine Beschreibung der einzelnen Realisierung siehe Tabelle 38.

Jeder Report bietet durch die Methode #getContentAsSimpleText() beziehungsweise #getContentAsSimpleText(showInputParameters: boolean, showOutputParameters: boolean) die Möglichkeit, die in ihm enthaltenen Inhalte und wahlweise auch die Parametersammlungen in einem einfachen Textformat zurückzugeben.

6.3.11. Maschinelles Lernen

Das Maschinelle Lernen (ML) zur Aktivitätserkennung ist im Paket mamks.algorithm.ml realisiert. Das Paket umfasst eine recht große Menge an Klassen sowie weitere Pake-

Tabelle 38.: Eingabeparameter des Algorithmus zur Schritterkennung

Implementierung	**Information**
ConfusionMatrixReport	Die im Paket mamks.evaluation realisierte ConfusionMatrix bietet die Möglichkeit per Methode #getReport(inputParameterCollection, outputParameterCollection) einen Report zu erzeugen, der die Matrix sowie die angegeben Parametersammlungen vorhält.
GaitParameterReport	In diesem Report werden die Gangparameter über den vom Schritterkennungsalgorithmus erkannten Schritten berechnet und gespeichert. Für weitere Informationen zum Schritterkennungsalgorithmus und zu den berechneten Gangparametern siehe Unterabschnitt 6.3.9.
HierarchicalConfusionMatrixReport	Da das Training hierarchisch erfolgt (zuerst werden die Zustände und dann die Bewegungen innerhalb der jeweiligen Zustände klassifiziert), besteht das Problem, dass bei der Validierung während des Trainings vier Konfusionsmatritzen generiert werden. Da im ConfusionMatrixReport die hierarchische Zusammenhangsinformation verloren gehen würde, wurde, um diese zu erhalten, dieser Report implementiert, der mehrere Konfusionsmatritzen mit ihren zugehörigen ClassifierType-Informationen speichern kann.
LabelGroupEvaluationReport	Dieser Algorithmus speichert die Ergebnisse des LabelGroupEvaluators (aus dem Paket mamks.evaluation) und bietet als zusätzliche Funktion die Möglichkeit die Information zu einer Datei im TDF-Format (tab delimited) zu exportieren.

te, die logische Untergruppen bilden. Für die Pakete mamks.algorithm.ml.dimred und mamks.algorithm.ml.sampling sei auf Unterabschnitt 6.3.6 sowie Unterabschnitt 6.3.7 verwiesen. Das Paket mamks.algorithm.ml.featuresets enthält eine Vielzahl sogenannter FeatureSets. Ein FeatureSet ist eine Menge von Features die gemeinsam geeignet sein sollen, um die Eingabe für einen Klassifikator zu bilden. Für weitere Details sei auf die Seminararbeit D.8 verwiesen, die sich der Bestimmung geeigneter FeatureSets widmet. Das Paket mamks.algorithm.ml.optimization realisiert die Parameteroptimierung mit Hilfe von evolutionären Algorithmen. Diese Realisierung wird in Unterabschnitt 6.3.12 beschrieben.

Die verbleibenden Klassen des Pakets sind grob in drei Kategorien unterteilbar: Modell, Training und Klassifikation.

Modell

Die Trainings- und Klassifikationsalgorithmen sind unabhängig von der konkret genutzten Bibliothek für die verschiedenen Techniken des Maschinellen Lernens. Um dies zu erreichen wurden mehrere Modell-Klassen entwickelt, die die generelle Funktionalität einer „Supervised Learning“-Technik umsetzen (SupervisedModel). Die entsprechenden Implementierungen bieten so eine einheitliche Schnittstelle, können intern aber die gewünschte ML-Bibliothek verwenden (AMMModel, BoostModel, MLPModel). Darüber hinaus speichert das SupervisedModel die verwendete Funktion zur Generierung des SlidingWindows, sowie die verwendete Dimensionreduktionsmethode. Das ist notwendig, weil die Serialisierung der Modell-Klassen als Speicherformat dient. Wenn das Modell später wieder geladen wird, muss für eine Klassifikation bekannt sein, wie das Schiebefenster und die Dimensionsreduktion eingestellt waren.

Die Klasse HierarchicalModel fasst vier SupervisedModel-Instanzen zusammen, um die Idee der hierarchischen Klassifikation umzusetzen. Welches Modell für welche Klassifikation verwendet werden soll, wird mit Hilfe der Aufzählung ClassifierType definiert.

Training

Ein Trainingsbeispiel wird durch die Klasse TrainingExample dargestellt, dass einem Merkmalsvektor ein Label zuordnet. Eine TrainingExampleSource ist eine Aggregation von TrainingExamples, die aus MotionData-Objekten extrahiert werden. Das hierfür benötigte SlidingWindow wird dynamisch durch eine Function<MotionData, SlidingWindow> erzeugt, mit der die TrainingExampleSource instanziiert wurde. In der Klasse TrainingAlgorithm werden so vier TrainingExampleSource-Instanzen für die vier Klassifikatoren STATE, STATIC, DYNAMIC, TRANSITION mit den Trainingsbeispielen der Datensätze befüllt.

Wenn alle Daten aufgesammelt wurden, kann das eigentliche Training gestartet werden. Hierfür wird ein SupervisedHierarchicalTraining-Objekt erzeugt, dem die view TrainingExampleSource-Instanzen, das zu trainierende hierarchische Modell, eine Trainingsmethode sowie die Dimensionreduktionsmethoden übergeben werden. Die Trainingsmethode ist vom Typ Trainer und aktuell sind der SplitValidationTrainer und der CrossValidationTrainer umgesetzt. Während der SplitValidationTrainer die extrahierten Trainingsbeispiele in eine Training- und Validationsmenge unterteilt, führt der CrossValidationTrainer eine Kreuzvalidierung durch. Die Dimensionreduktionsmethoden sind vom Typ DimensionalityReduction-

Method, wobei aktuell nur die Hauptkomponentenanalyse (PrincipalComponentAnalysis) implementiert ist.

Das SupervisedHierarchicalTraining trainiert nun die vier verschiedenen Klassifikatoren mit den entsprechenden TrainingExampleSources. Dabei wird der gesetzte Trainer verwendet, der gleichzeitig auch eine Evaluation vornimmt. So erhält man neben dem trainierten Modell auch dessen Güte.

Klassifikation

Bei der Klassifikation soll nun ein Datensatz mit Hilfe eines antrainierten Modells klassifiziert werden. Analog zu dem SupervisedHierarchicalTraining existiert die Klasse SupervisedHierarchicalClassification, die diese Aufgabe übernimmt und weiter delegiert. Im ersten Schritt wird mit dem STATE-Klassifikator der Datensatz in seine Zustände (STATIC, DYNAMIC und TRANSITION) klassifiziert. Danach werden die gefundenen Teilbereiche mit den entsprechenden Klassifikatoren klassifiziert, um die konkrete Aktivität zu erhalten. Dies wird für die drei Klassifikatoren parallel ausgeführt.

Die konkrete Klassifikation wird von der Klasse MotionDataClassifier wahrgenommen, die über gegebene Intervalle iteriert und mit Hilfe eines SlidingWindows die Features extrahiert. Da das SlidingWindow überlappen kann, können für einen Zeitpunkt mehrere Klassifikationen vorliegen, die sich voneinander unterscheiden können. Daher wird per Mehrheitsentscheid bestimmt, welches Label für den Zeitpunkt gewählt werden soll. Diese Funktionalität ist in der Klasse LabelVoting implementiert.

Nach der Klassifikation existiert ein Label zu jedem Zeitpunkt. Diese Label sind aber alle nur so lang wie die Überlappung des SlidingWindow. Daher werden in einem letzten Schritt die Label zusammengefügt, welche zeitlich aneinander angrenzen und das gleiche Label tragen.

6.3.12. Parameteroptimierung

Die Parameteroptimierung ist im Paket mamks.algorithm.ml.optimization realisiert. Der einzige für den Benutzer sichtbare Algorithmus ist die Klasse ParameterOptimizationAlgorithm. In dieser ist die Hauptschleife des evolutionären Algorithmus implementiert. Weiterhin ist nennenswert, dass beim Laden der Bewegungsdaten alle nicht benötigten Da-

tenreihen aus dem Arbeitsspeicher gelöscht werden, um Speicherplatz zu sparen. Welche Datenreihen dies sind, kann bisher nur im Code eingestellt werden.

Zu Beginn der Optimierung wird die Initialpopulation aus Properties-Dateien geladen. Deren Format ist im Benutzerhandbuch der Anwendung in Detail beschrieben. Es entsteht dadurch ein Population-Objekt, welches wiederum einzelne Individual-Objekte enthält. Weiterhin werden Methoden angeboten, die die Rekombination, Mutation, Fitness-Bewertung und Selektion durchführen. Die Fitness-Bewertung ist der aufwändige Teil des Algorithmus, da für jede Fitness-Bewertung ein vollständiges Training mit den gewählten Parametern durchgeführt werden muss. Jedoch kann die Auswertung der Individuen unabhängig voneinander erfolgen, sodass hier Threads genutzt werden, um die Bestimmung der Fitness zu parallelisieren. Weiterhin werden in jedem Schleifendurchlauf die aktuell beste Fitness, das aktuell beste Individuum, sowie die aktuelle Population in Dateien ausgegeben, um sie anschließend auswerten zu können. Dies wird von den Klassen FitnessWriter und PropertiesIO übernommen.

Die Klasse Individual handhabt die Repräsentation der Parameter. Sie stellt eine Zuordnung von Gen-Namen zu ihren Werten, ihrem Wertebereich und ihrer Mutationsrate zur Verfügung. Diese Informationen sind selbst innerhalb einer Gene-Klasse gekapselt. Es werden drei verschiedene Arten von Genen unterstützt: reellwertig, ganzzahlig und kategorisch. Die Gene implementieren eine eigene mutate-Methode, welche eine dem Typ angemessene Mutation durchführt. Jedes Gen speichert seine Mutationsrate (entspricht der Stärke der Mutation) selber und mutiert diese auch selber, da ein selbstadaptives Verfahren eingesetzt wird. Dadurch ist es nicht notwendig, die Mutationsrate manuell zu wählen.

Die Klasse Population handhabt eine Menge von Individual-Objekten. Die Methode recombinate(k: int, ppc: int) erzeugt aus den Individuen k Nachfahren, indem jeweils ppc zufällig ausgewählte Individuen rekombiniert werden. Die Rekombination wählt dann für jedes Gen zufällig den Wert einer der ppc Elternteile aus. Die mutate-Methode ruft lediglich die mutate-Methode jedes Individuums auf. Nach Rekombination und Mutation ist eine neue Population mit veränderten Parametern entstanden, die nun ausgewertet werden müssen. Dazu wird die Methode evaluateFitness(:FitnessEvaluator, :ExecutorService): double bereitgestellt. Ihr werden ein ExecutorService zur Ausführung der einzelnen Auswertungen in Threads und ein FitnessEvaluator zur Fitnessauswertung bereitgestellt und sie gibt die beste F1-Score der Population zurück.

Der FitnessEvaluator hat die Aufgabe, einem Individual seine Fitness oder genauer F1-Score zuzuordnen. Dazu wird die Machine-Learning-Pipeline entsprechend den Parameterwerten des Individuums parametrisiert und ein Training ausgeführt. Anschließend werden einige speziell für die Validierung ausgezeichneten Datensätze, welche nicht zum Training verwendet wurden, mit dem trainierten Modell klassifiziert. Dabei wird eine Konfusionsmatrix gegenüber den manuelle angelegten Referenzlabeln erstellt, aus welcher sich die F1-Score ableitet.

6.4. Job-Executor

Wie in Abschnitt 5.1 erläutert, müssen Algorithmen in eigenen Prozessen ausgeführt werden. Zu diesem Zweck existiert der JobExecutor. Seine Aufgabe ist es, eine Job-Beschreibung aus einer Datei zu lesen und den dort beschriebenen Auftrag auszuführen. In Abbildung 54 ist der Ablauf einer Algorithmusausführung zu sehen, der nun im Folgenden beschrieben wird.

Zur Ausführung eines Algorithmus werden vom Frontend zunächst die gewünschten Algorithmus-Klassen instanziiert und mit Parametern gefüllt. Anschließend wird daraus ein Job erzeugt, welcher dem JobManager zur Ausführung übergeben wird. Die Job-Klasse ist serialisierbar, sodass sie in eine Job-Datei geschrieben werden kann. Je nach konkreter Implementierung des JobManagers wird der Job auf verschiedene Weisen zur Ausführung gebracht.

Der LocalThreadPoolJobManager instanziiert einen neuen JobExecutor, dem dann der Job übergeben wird. Für die Ausführung von Algorithmen in anderen Prozesses legt der LocalProcessPoolJobManager ein temporäres Verzeichnis an und serialisiert den Job in eine Datei in diesem Verzeichnis. Wenn der Job auf einem Remote Host mit Hilfe des RemoteProcessJobManagers ausgeführt werden soll, werden notwendige Daten und auch die Job-Datei vorher übertragen.

Anschließend wird ein neuer Prozess des JobExecutors gestartet, dessen Arbeitsverzeichnis das angelegte Verzeichnis ist. Der JobExecutor erwartet in seinem Arbeitsverzeichnis die Job-Datei und liest sie ein. Daraus wird der Algorithmusaufbau mitsamt Parametern rekonstruiert. Bevor der Job zur Ausführung kommt, werden benötigte Bewegungsdaten in die ParameterCollections des Algorithmus eingefügt. Dazu wird ein DataStorage-Objekt

genutzt, welches die Daten vom lokalen Massenspeicher des Hosts liest. Nach der Ausführung werden Ergebnisse ebenfalls mit Hilfe des DataStorage-Objekts zurückgeschrieben.

Wenn während der Ausführung eines Jobs Exceptions auftreten, werden diese in eine Datei serialisiert. Das Frontend kann diese Datei einlesen und die Exception verarbeiten.

Da bisher auf Grund der zusätzlichen Komplexität keine Kommunikation zwischen den Algorithmus-Prozessen und dem Frontend-Prozess vorgesehen ist, muss das Frontend den Job-Status durch regelmäßiges Polling überwachen.

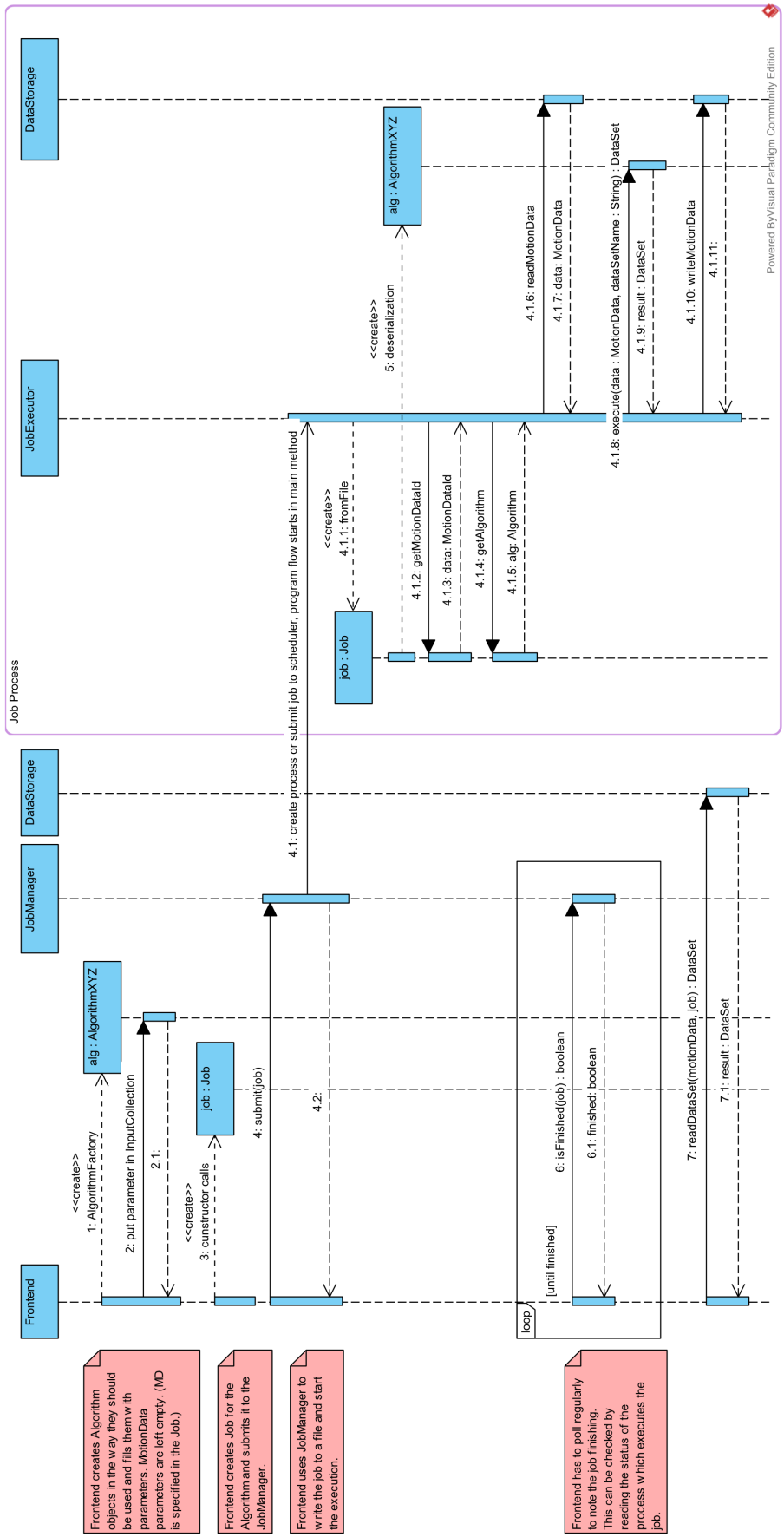


Abbildung 54.: Algorithmusausführung

7. Evaluierung

Nachfolgend werden die im Projekt verwendeten Evaluierungsmethoden vorgestellt und eine Evaluierung der Fenstergrößen und Schrittweiten des Sliding-Window-Verfahrens durchgeführt, um eine geeignete Wahl dieser Parameter zu finden. Zudem wird der Einfluss von Zusatzdaten, die nicht von Assessment-Probanden stammen, auf die Klassifikationsergebnisse untersucht.

7.1. Split-Validierung

Eine sehr einfache Methode, um ein Klassifikationsmodell zu trainieren und anschließend zu testen, ist es, die Daten in eine Trainingsmenge und eine Validierungsmenge aufzuteilen. Mit der Trainingsmenge wird das Training durchgeführt und auf der Validierungsmenge wird die Güte der Klassifikation bestimmt. Diese Methode ist extrem simpel zu implementieren, da der gesamte, zufällig durchmischte Datenbestand einfach nur nach einem gewünschten Verhältnis in die beiden Mengen aufgeteilt werden muss. Ein Nachteil ist, dass die zufällige Auswahl der Daten der Validierungsmenge das Ergebnis deutlich beeinflussen kann. Die Kreuzvalidierung löst dieses Problem.

7.2. Kreuzvalidierung

Um Klassifikationsalgorithmen bzw. Klassifikationsmodelle zu testen, bietet es sich an, die Kreuzvalidierung zu nutzen. Eine Kreuzvalidierung wird verwendet, damit die Güte von statistischen Modellen gemessen werden kann. Dabei wird nicht auf dem gesamten Datensatz zur gleichen Zeit trainiert und getestet, sondern der Datensatz wird in einen Trainings- und einen Testdatensatz eingeteilt. Zu beachten ist, dass es verschiedene Abwandlungen der Kreuzvalidierung gibt - Beispiele dafür sind Leave-One-Out-Kreuzvalidierung,

Leave-P-Out-Kreuzvalidierung und k-fache Kreuzvalidierung. [Hyn10]

Zur Überprüfung der Klassifikationsmodelle und -algorithmen wird in dieser Anwendung die k-fache Kreuzvalidierung verwendet. Da bei der k-fachen Kreuzvalidierung jeder Datensatz einmal zum Test, aber auch zum Training genutzt wird, haben wir uns für diese Möglichkeit entschieden. Dabei wird der Datensatz in k möglichst gleich große Teildatensätze unterteilt. Bei jedem Durchgang wird dann einer der k Teildatensätze als Testdatensatz genutzt und die Gesamtheit der jeweils anderen Teildatensätze zum Training. Dies wird insgesamt k-mal wiederholt, bis jeder der k Teildatensätze einmal als Testdatensatz verwendet wurde.

In der Implementierung wird dann für die Kreuzvalidierung zum einen der zu validierende Klassifizierer übergeben und zum anderen eine Liste der Daten, wobei dem jeweiligen Feature-Vektor auch noch das dazugehörige Label zugewiesen ist, damit in der Konfusionsmatrix ein Vergleich mit dem korrekten Label erfolgen kann. [Sch97]

Als Ergebnis wird von der Hauptmethode der Kreuzvalidierung für jeden Klassifikationstyp (state, static, dynamic, transition) der Durchschnitt der F_1-Score-Werte zurückgegeben, die pro Label das harmonische Mittel von Genauigkeit und Sensitivität angeben. [Bro14]

Der Grund dafür, dass in diesem Fall die k-fache Kreuzvalidierung genommen wurde, liegt darin, dass die k-fache Kreuzvalidierung den gesamten Datensatz in k ungefähr gleich große Teile aufteilt und anschließend jeder Teildatensatz einmal zur Validierung genutzt wird - also insgesamt k-mal.

Bei der Leave-One-Out-Kreuzvalidierung wird jeweils ein Datensatz als Testdatensatz genommen und die restlichen Daten zur Validierung [Hyn10]. Das wird dann n-mal wiederholt, wobei n für die Menge an Datensätzen insgesamt besteht, also in diesem Fall bei mehreren zehntausend. [Hyn10]

Weiterhin gibt es noch die Leave-P-Out-Kreuzvalidierung, wobei dort jeweils P Teildatensätze aus dem kompletten Satz entfernt werden [sld]. Für N Teildatensätze wird dabei eine Training-Validierungspaarmenge von $\binom{N}{P}$ erzeugt. [sld]

7.3. Auswahl einer geeigneten Fenstergröße und Schrittweite für das Sliding-Window-Verfahren

Um eine möglichst gute Erkennung der Bewegungen gewährleisten zu können, müssen eine geeignete Fenstergröße und Schrittweite für das Sliding-Window-Verfahren (siehe Unterabschnitt 3.4.2) ausgewählt werden. Aus diesem Grund wurde eine Evaluierung von unterschiedlichen Fenstergrößen und Schrittweiten durchgeführt.

Die Evaluierung wurde mit einem AMM-Modell (siehe Unterabschnitt D.4) auf allen beschrifteten Datensätzen durchgeführt, da dieses schnelle Trainingszeiten bietet und die Einstellung der geeigneten Fenstergröße und Schrittweite unabhängig vom Modell ist. Es wurden verschiedene Szenarien getestet, die sich in Fenstergrößen und Schrittweiten des Sliding-Windows unterscheiden.

Es wurde die in der Software integrierte 5-fache Kreuzvalidierung verwendet und als Bewertungskriterium für die getesteten Szenarien wurde der F_1-Score jeder Klasse betrachtet. Die F_1-Scores aller Klassen für jedes getestete Szenario sind in Tabelle 39 dargestellt.

Wie zu erkennen ist, zeigen die vier Klassifizierer jeweils für unterschiedliche Szenarien die besten F_1-Scores, sodass unterschiedliche Fenstergrößen und Schrittweiten für die vier Klassifizierer gewählt werden sollten. Die jeweils beste Einstellung des Sliding-Windows für die Klassifizierer aufgrund der F_1-Scores sind in Tabelle 40 zu sehen.

7.4. Klassifikationsoptimierung per Cluster

Die implementierte Umsetzung des Klassifikationskonzepts unter Verwendung von maschinellen Lernalgorithmen hat eine Vielzahl an einstellbaren Parametern z. B. Länge und Schrittweite des Schiebefensters, Wahl der zu verwendenden ML-Technik und Merkmalen, Einstellparameter der ML-Techniken. Diese Parameter haben einen wesentlichen Einfluss auf das Klassifikationsergebnis beziehungsweise auf die Klassifikationsgenauigkeit. Damit ergibt sich das Problem, wie diese Parameter einzustellen sind, um das optimale Klassifikationsergebnis zu erhalten.

Dieses Problem ist keines Falls trivial, da der Lösungsraum auf Grund der hohen Anzahl an einstellbaren Parametern sehr groß ist. Manuell ist dieses Problem nicht zu lösen, da der Lösungsraum nicht in seiner Gänze „erkundet“ werden kann und somit nie garantiert

		Fenstergröße/Schrittweite					
Klassifizierer	Klasse	50/50	25/75	75/75	100/20	100/34	100/100
STATE	ANY_STATIC	0,969	0,978	0,977	0,987	0,986	0,984
	ANY_DYNAMIC	0,885	0,925	0,911	0,952	0,949	0,925
	ANY_TRANSITION	0,843	0,862	0,866	0,888	0,895	0,896
STATIC	STAND	0,889	0,898	0,891	0,904	0,900	0,892
	SIT	0,880	0,894	0,883	0,899	0,896	0,884
	LIE_ON_BACK	0,965	0,967	0,966	0,965	0,967	0,958
DYNAMIC	WALK	0,636	0,682	0,506	0,778	0,730	0,659
	STAIR_CLIMB	0,560	0,707	0,667	0,789	0,777	0,624
	JUMP	0,770	0,830	0,804	0,841	0,852	0,823
TRANSITION	STAND_SIT	0,869	0,900	0,848	0,907	0,904	0,937
	SIT_STAND	0,836	0,874	0,785	0,882	0,878	0,928

		Fenstergröße/Schrittweite					
Klassifizierer	Klasse	125/25	125/42	125/125	150/30	150/50	150/150
STATE	ANY_STATIC	0,992	0,992	0,990	0,995	0,995	0,992
	ANY_DYNAMIC	0,968	0,968	0,956	0,978	0,979	0,968
	ANY_TRANSITION	0,907	0,923	0,939	0,900	0,918	0,932
STATIC	STAND	0,904	0,903	0,883	0,901	0,903	0,879
	SIT	0,897	0,902	0,876	0,898	0,899	0,876
	LIE_ON_BACK	0,971	0,972	0,967	0,971	0,976	0,971
DYNAMIC	WALK	0,813	0,804	0,677	0,820	0,763	0,563
	STAIR_CLIMB	0,799	0,806	0,580	0,792	0,745	0,674
	JUMP	0,846	0,853	0,850	0,797	0,825	0,833
TRANSITION	STAND_SIT	0,914	0,910	0,914	0,906	0,861	0,850
	SIT_STAND	0,876	0,875	0,891	0,847	0,835	0,847

		Fenstergröße/Schrittweite					
Klassifizierer	Klasse	175/35	175/59	175/175	200/29	200/40	200/67
STATE	ANY_STATIC	0,996	0,996	0,995	0,997	0,997	0,997
	ANY_DYNAMIC	0,986	0,986	0,975	0,988	0,988	0,987
	ANY_TRANSITION	0,889	0,903	0,912	0,836	0,864	0,887
STATIC	STAND	0,907	0,897	0,865	0,909	0,909	0,906
	SIT	0,905	0,894	0,871	0,906	0,907	0,905
	LIE_ON_BACK	0,968	0,974	0,966	0,974	0,976	0,968
DYNAMIC	WALK	0,818	0,726	0,487	0,844	0,822	0,778
	STAIR_CLIMB	0,812	0,699	0,618	0,810	0,799	0,774
	JUMP	0,731	0,592	0,552	0,656	0,547	0,637
TRANSITION	STAND_SIT	0,921	0,903	0,900	0,853	0,869	0,816
	SIT_STAND	0,815	0,789	0,731	0,658	0,438	0,583

Tabelle 39.: Testszenarien zur Auswahl der Sliding-Window-Größe und -Schrittweite

Klassifizierer	Fensterlänge	Schrittweite
STATE	150	150
STATIC	200	40
DYNAMIC	125	42
TRANSITION	100	100

Tabelle 40.: Gewählte Fensterlängen und Schrittweiten

werden kann, dass das globale Optimum gefunden wurde. Tatsächlich ist das Problem so komplex, dass Verfahren wie Rastersuche den Lösungsraum nicht in absehbarer Zeit durchlaufen können. Dies wird insbesondere dadurch verstärkt, dass die Zielfunktion dieses Optimierungsproblems die Genauigkeit der Klassifikation ist, die sich z. B. durch den F1-Score beschreiben lässt. Dies bedeutet, dass für das Testen einer Parametereinstellung ein vollständiges Training mit anschließender Ergebnisvalidierung durchgeführt werden muss. Je nach ML-Technik kann ein Training mehre Minuten dauern, womit das Optimierungsproblem neben dem Problem des großen Lösungsraum auch das Problem des hohen zeitlichen Aufwandes aufweist.

Um den zeitlichen Aufwand der Optimierung zu reduzieren, wird als Verfahren der Optimierung ein evolutionärer Algorithmus verwendet und eine Vorauswahl an wählbaren Merkmalen per FeatureSets getroffen. Dies bedeutet, dass unter Umständen das globale Optimum nicht gefunden werden kann. Es garantiert allerdings, dass in absehbarer Zeit zumindest „gute“ lokale Optima gefunden werden können. Dieses Problem ist weiterhin äußert rechenaufwendig. Aus diesem Grund wurde der Optimierungsprozess auf dem Rechencluster CARL der Carl-Von-Ossietzky Universität Oldenburg durchgeführt. Die Optimierung wurde in mehreren Durchläufen durchgeführt, wobei die initialen Individuen eines Folgedurchlaufs immer als die optimalen Individuen des vorherigen Durchlaufs gewählt wurden.

Weitere Informationen zu dem Verfahren der evolutionären Algorithmen und den dazugehörigen Ergebnissen sowie eine Interpretation der Ergebnisse der Klassifikationsoptimierung sind in Abschnitt 5.9 gegeben.

7.5. Einfluss von Nicht-Probandendaten auf das Klassifikationsergebnis

Nachdem Probandendaten manuell beschriftet wurden und erste Tests mit Klassifikatoren durchgeführt wurden, fiel auf, dass insbesondere die Bewegungen Springen, Aufstehen und Hinsetzen schlecht erkannt werden. Es lag die Vermutung nahe, dass zu wenige Trainingsdaten für diese Bewegungen vorliegen, da sie in den Assessments nur selten vorkommen. Daher wurde entschieden, weitere Aufnahmen anzulegen, bei denen diese Bewegungen sehr oft ausgeführt werden.

Aus organisatorischen Gründen konnten nicht direkt Probanden des entsprechenden Alters gebeten werden, diese Aufnahmen durchzuführen, sondern die Bewegungen wurden von Projektgruppenmitgliedern im Alter von 22 bis 44 Jahren durchgeführt. Da das Ziel der Projektgruppe aber die Bewegungsklassifizierung von Personen im Seniorenalter ist, muss der Einfluss der Nicht-Probandendaten auf das Klassifikationsergebnis geprüft werden.

Um den Einfluss der zusätzlich aufgenommenen Daten für die Labels SIT_STAND, STAND_SIT und JUMP zu prüfen, wurde ein DTBoost-Modell (siehe Unterabschnitt D.1) mit und ohne den zusätzlichen Daten trainiert und evaluiert. Die Evaluation fand nur auf Probandendaten statt, da diese letztendlich zu erkennen sind. Leider konnte aus diesem Grund die in die Software integrierte Kreuzvalidierung nicht verwendet werden. Stattdessen wurde eine 3-fache Kreuzvalidierung manuell durchgeführt, indem die Probandendatensätze in Gruppen aufgeteilt wurden. Die getesteten Szenarien sind in Tabelle 41 dargestellt und die F_1-Scores der drei Label für jedes Szenario sind in Tabelle 42 zu sehen. Einfluss der zusätzlich aufgenommenen Daten für die Labels SIT_STAND, STAND_SIT und JUMP zu prüfen, wurde ein DTBoost-Modell (siehe Seminararbeit D.1) mit und ohne zusätzliche Daten trainiert und evaluiert. Die Evaluation fand nur auf Probandendaten statt, da diese letztendlich zu erkennen sind. Leider konnte aus diesem Grund die in die Software integrierte Kreuzvalidierung nicht verwendet werden. Stattdessen wurde eine 3-fache Kreuzvalidierung manuell durchgeführt, indem die Probandendatensätze in Gruppen aufgeteilt wurden. Die getesteten Szenarien sind in Tabelle 41 dargestellt und die F1-Scores der drei Labels für jedes Szenario sind in Tabelle 42 zu sehen. Es werden jeweils Szenarien A*n* und B*n* verglichen, da diese dieselben Probandendaten zum Training verwenden, aber B*n* zusätzlich mit den Nicht-Probandendaten trainiert wurde. Die Veränderung der F1-Score ist als Δ eingetragen. Im Durschnitt steigt die F1-Score um 4 % an. In zwei

Szenario	Probanden G1	Probanden G2	Probanden G3	Nicht-Probanden
A1	Training	Training	Validierung	
A2	Training	Validierung	Training	
A3	Validierung	Training	Training	
B1	Training	Training	Validierung	Training
B2	Training	Validierung	Training	Training
B3	Validierung	Training	Training	Training

Tabelle 41.: Testszenarien zur Prüfung des Einflusses von Nicht-Probandendaten auf das Klassifikationsergebnis

F_1-Scores	A1	B1	Δ	A2	B2	Δ	A3	B3	Δ
STAND_SIT	0,54	0,63	0,09	0,67	0,69	0,02	0,64	0,59	**-0,05**
SIT_STAND	0,55	0,59	0,04	0,60	0,58	**-0,02**	0,50	0,56	0,06
JUMP	0,28	0,33	0,05	0,43	0,52	0,09	0,51	0,57	0,06

Tabelle 42.: F1-Scores der Testszenarien

Szenarien ist für eine Klasse eine niedrigere F1-Score erkennbar. Dies ist nicht verwunderlich, da sowohl der Trainingsprozess der Entscheidungsbäume als auch die Gruppierung der Trainingsdaten Zufallseinflüssen unterlegen sind. Zusammenfassend lässt sich sagen, dass mindestens die Erkennung von Sprüngen deutlich von den Nicht-Probandendaten profitiert, während Hinsetzen und Aufstehen eine leicht positive Tendenz aufweisen. Es werden jeweils Szenarien A*n* und B*n* verglichen, da diese dieselben Probandendaten zum Training verwenden, aber B*n* zusätzlich mit den Nicht-Probandendaten trainiert wurde. Die Veränderung der F_1-Score ist als Δ eingetragen. Im Durschnitt steigt die F_1-Score um 4 % an. In zwei Szenarien ist für eine Klasse eine niedrigere F_1-Score erkennbar. Dies ist nicht verwunderlich, da sowohl der Trainingsprozess der Entscheidungsbäume als auch die Gruppierung der Trainingsdaten Zufallseinflüssen unterlegen sind.

Zusammenfassend lässt sich sagen, dass mindestens die Erkennung von Sprüngen deutlich von den Nicht-Probandendaten profitiert, während Hinsetzten und Aufstehen eine leicht positive Tendenz aufweisen.

8. Toshiba-Sensor

Im Rahmen einer Hardware-Entwicklung wurde die Nutzung des vollintegrierten ARM Sensorchips TZ1011 von Toshiba geprüft. Die Ergebnisse dazu sind in der Seminararbeit D.11 zu finden, die gemäß Absprache mit Dr. Sebastian Fudickar nachgereicht wird.

9. Fazit

Im Rahmen der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ (PGMAmKS) wurde die PGMAmKS-Applikation entwickelt. Diese ermöglicht das Importieren und Verwalten von Daten, welche durch *humotion*-Sensorgürtel aufgenommen und mit Hilfe der *humotion*-Anwendung in ein CSV-Format überführt wurden. Zur Verwaltung der Daten wird ein eigenes Dateiformat verwendet, welches die Rohdaten und abgeleiteten, zugehörigen Daten in Binär- beziehungsweise XML-Dateien speichert und aus diesen in die Anwendung lädt. Die Entscheidung zu diesem Format wurde nach einer Evaluation unterschiedlicher Datenverwaltungsmöglichkeiten getroffen, da es gegenüber den anderen Verfahren und dem ursprünglichen CSV-Format deutliche Vorteile in den Aspekten Speichereffizienz sowie Lese- und Schreibzugriffszeit bringt. Neben der Verwaltung der Rohdaten erlaubt die Applikation die automatische Generierung abgeleiteter Daten durch Algorithmen und das manuelle Markieren der Daten mit Beschriftungen (Labels).

Um allen Anwendungsfällen gerecht zu werden, wurden zwei Schnittstellen für die Applikation implementiert: graphische Nutzerschnittstelle (GUI) und Command Line Interface (deut. Kommandozeilenschnittstelle) (CLI). Die GUI dient im Kern der Visualisierung und der manuellen Beschriftung der Daten. Sie bietet darüber hinaus jedoch den vollen Funktionsumfang der Applikation. Die CLI bietet die Möglichkeit, die Funktionen der Applikation, wie z. B. das Ausführen von Algorithmen, ohne den GUI-Overhead per Konsole auszuführen. Sie erlaubt z. B. auch die Verkettung mehrerer Funktionsaufrufe der Applikation durch Bash-Skripte.

Die Applikation bietet außerdem ein Job-Management zur Überwachung und Steuerung von gestarteten Jobs. Ein Job dient dabei der Kapselung von Algorithmusausführungen. Mithilfe dieser Kapselung können Algorithmen sowohl Thread- als auch Prozess-basiert, lokal oder auf einem Remote-Host, ausgeführt werden.

Die Möglichkeit der Ausführung von Algorithmen auf einem Remote-Host (z. B. einem

Rechen-Cluster) ist eine weitere wesentliche Funktion, die durch die Applikation bereitgestellt wird.

Wesentliche Algorithmen, die innerhalb der Applikation implementiert wurden, sind die Trainings- und Klassifikationsalgorithmen. Erstere erlauben das Trainieren von speziell ausgewählten maschinellen Lernalgorithmen wie Boosted Decision Tree (BDT), Boosted Decision Stump (BDS), Adaptive Multi-Hyperplane Machine (AMM) und Artificial Neural Network (ANN). Letztere erlauben die Klassifikation mithilfe der trainierten Algorithmen.

Bei all diesen Algorithmen handelt es sich um überwachte Lernalgorithmen, für die beschriftete Trainingsdaten benötigt werden. Um ausreichend viele Trainingsdaten zu erhalten, wurden 87 Assessment-Aufnahmen beschriftet mit einer Gesamtaufnahmedauer von circa 88 Stunden. Es wurden hierbei sowohl Assessmentmuster (wie TUG) mit einer Gesamtdauer von circa elf Stunden als auch Bewegungsmuster (wie Gehen) mit einer Gesamtdauer von circa 50 Stunden beschriftet. Die aufgenommen Assessments wurden mit Probanden ab einem Alter von 70 Jahren im Rahmen der VERSA-Studie durchgeführt. Die Auswahl an zu beschriftenden Assessment- und Bewegungsmustern, sowie die Auswahl an letztendlich zu klassifizierenden Bewegungen, wurde durch die Projektgruppe getroffen.

Für die meisten maschinellen Lernalgorithmen ist es notwendig, eine möglichst gleichmäßige Verteilung der Trainingsdaten zwischen den einzelnen Klassen zu erreichen. Ungünstiger Weise waren in den Assessment-Aufnahmen einige Klassen deutlich seltener vertreten als andere, z. B. kommt Springen deutlich seltener vor als Gehen. Um dieser Ungleichverteilung entgegenzuwirken, wurden durch die Projektgruppe Zusatzdaten mit einer Gesamtdauer von circa 31 Stunden erzeugt, indem die Bewegungen Aufstehen, Hinsetzen, Springen und Treppesteigen von den Projektgruppenmitgliedern und Betreuern durchgeführt, per Sensorgürtel aufgezeichnet und anschließend manuell beschriftet wurden. Dabei wurden circa 8 Stunden an zusätzlichen beschrifteten Trainingsdaten erzeugt.

Ein weiterer wesentlicher Algorithmus, der innerhalb der Applikation implementiert wurde, ist der Parameteroptimierungsalgorithmus. Mit diesem wurde die Einstellung und Auswahl an Parametern und Merkmalen, welche durch vorab ausgewählte Feature-Sets eingeschränkt wurden, für die Klassifikation von Bewegungsmustern optimiert. Diese Optimierung wurde für jeden der vier maschinellen Lernalgorithmen durchgeführt, was die anschließende Wahl des optimalen Algorithmus beziehungsweise die Wahl der optimalen Teilalgorithmen erlaubt. Die Klassifikation der Bewegungen durch den Klassifikations-

algorithmus und das zugehörige Training erfolgt hierarchisch: Zuerst klassifiziert ein Klassifikator nach den drei Zuständen (dynamisch, statisch und Transition) und anschließend werden für jeden der drei Zustände die einzelnen Bewegungsmuster mit Hilfe eines eigenen Klassifikators klassifiziert. Für die Klassifikation von statischen Bewegungsmustern kann somit z. B. ein anderer Klassifikator verwendet werden als für dynamische Bewegungsmuster. Das Ergebnis der Parameteroptimierung ist in Tabelle 43 zu sehen.

	Zustand	**statisch**	**dynamisch**	**Transition**
BDT	**96,6**%	96,9%	95,4%	93,1%
BDS	95,5%	94,7%	93,6%	89,9%
MLP	96,4%	**97,3**%	**97,5**%	**94,8**%
AMM	95,2%	97,1%	93,8%	91,2%

Tabelle 43.: Ergebnis der Parameteroptimierung

Die Tabelle zeigt die F_1-Scores jeweils pro Klassifikator und verwendetem maschinellen Lernalgorithmus. Der F_1-Score ist das harmonische Mittel von Präzision und Sensitivität. MLP bezeichnet hierbei Multi-Layer Perceptrons - eine spezielle Art von künstlichem neuronalem Netz. Diesem Ergebnis nach scheint BDT als Verfahren für die Zustands-Klassifikation und MLP als Verfahren für alle anderen Klassifikatoren als beste Wahl. Aufgrund einer Umstellung des Validierungsverfahrens, während der Parameteroptimierung, gibt es unterschiedliche Klassifikationsergebnisse. Für genauere Informationen zur Parameteroptimierung siehe **??**.

Innerhalb der Applikation wurde außerdem ein Schritterkennungsalgorithmus implementiert. Dieser bestimmt innerhalb von als „Gang“ beschrifteten Segmenten die einzelnen Fersenaufschläge und markiert die Segmente zwischen zwei Fersenaufschlägen mit „Schritt“-Beschriftungen. Über diesen Segmenten können durch die Applikation (im Gangparameter-Algorithmus) die Gangparameter Schrittdauer, Doppelschrittdauer und Kadenz für eine Aufnahme bestimmt werden. Der Gangparameter-Algorithmus und weitere Algorithmen können ihre Ergebnisse innerhalb der GUI in Form von Reports visualisiert werden.

Von jedem Projektgruppenmitglied wurde zudem eine Ausarbeitung zu einem für die Projektgruppe relevanten und durch die Gruppe ausgewähltem Thema verfasst (siehe Anhang D). In einer Ausarbeitung wurde dabei an einer alternativen Sensorplattform gearbeitet, die später den *humotion*-Sensorgürtel ersetzen und die Möglichkeit zur Echtzeit-Datenübertragung bieten soll.

Vorstellbare Themen im Anschluss an die Projektgruppe sind: die automatische Erzeu-

gung von Assessmentmuster-Beschriftungen, die Anwendung der automatischen Bewegungserkennung auf die Daheim-Daten, die Gewinnung weiterer Gangparameter und die Gewinnung von Parameter über anderen Bewegungen wie Aufstehen, Springen und Treppensteigen. Aber auch die Fortführung der Parameteroptimierung, gegebenenfalls unter Wahl anderer Feature-Sets oder sogar gänzlich anderer Optimierungsverfahren wäre eine sinnvolle Fortsetzung. Diese Ansätze und weitere werden im nachfolgenden Kapitel 10 noch weiter ausgeführt.

10. Ausblick

Während dieses Projekts wurden einige relevante und interessante Funktionen identifiziert, welche im Rahmen dieses Projektes jedoch nicht umgesetzt werden konnten. Diese sollen in diesem Kapitel kurz beschrieben werden, um Ansatzpunkte für anschließende Arbeiten zu bieten.

10.1. Generierung von Assessment-Labels

Die Anwendung kann aktuell lediglich grundlegende Bewegungsmuster wie „Gehen“ erkennen und auf diesen weitere Algorithmen wie z. B. die Ganganalyse (auf „Gehen“-Segmenten) anwenden. Ein weiteres wesentliches Ziel ist die automatische Erkennung von Assessment-Mustern wie dem TUG-Test, da dies den grundlegenden Schritt zu automatisch berechneten Assessments darstellt. Anschließende Schritte wären, nachdem die automatische Assessmentmuster-Erkennung gelungen ist, die automatische Berechnung der relevanten Parameter über die erkannten Assessments.

Die Idee ist die, dass die Assessment-Muster nach der Erkennung der Bewegungsmuster über den bestimmten Bewegungsbeschriftungen abgeleitet werden. Beispielsweise wäre der TUG-Test als Abfolge der Bewegungen „Sitzen-Stehen“, „Gehen“, („Umdrehen“), „Gehen“, („Umdrehen“) und „Stehen-Sitzen“ zu identifizieren. Umdrehen ist aktuell keines der automatisch erkannten Bewegungsmuster. Während dieses Projekts wurden allerdings bereits erste Schritte in Richtung Assessment-Erkennung unternommen. So wurden „Umdrehen-Links“ und „Umdrehen-Rechts“ in den Umfang an Bewegungsbeschriftungen aufgenommen und auch bereits einige Datensätze mit diesen Beschriftungen versehen, da davon ausgegangen wird, dass Umdrehen als Bewegung für Assessments wie TUG oder den 6-Minuten-Gehtest (aufgrund der Ausführung während der VERSA-Studie) wesentlich ist. Im Speziellen wurde über den Einsatz von Hidden-Markov-Modellen zur

Assessmentmuster-Erkennung nachgedacht (siehe Unterabschnitt D.2 für eine detaillierte Beschreibung dieser Modelle).

Die Schwierigkeit bei diesem Ansatz ist die, dass tatsächliche Assessment-Muster identifiziert werden sollen, während Bewegungsabfolgen, die zufälliger Weise aus den gleichen Bewegungen bestehen wie das Assessment-Muster, nicht erkannt werden sollen.

10.2. Nachbereitung der Klassifikationsergebnisse

Aktuell schließen sich an die automatische Beschriftung von Bewegungsmustern keine Nachverarbeitungsschritte an. Es könnten aber verschiedene Mechanismen umgesetzt werden, um die Güte der Klassifikation zu verbessern.

Denkbare wäre beispielsweise ein Verfahren, das die bestimmten Beschriftungen betrachtet und Entscheidungen darüber fällt, ob eine bestimmte Beschriftung im Bezug auf seinen Kontext auch wirklich sinnvoll ist. Wenn z. B. eine längere „Sitzen“-Beschriftung von einer kurzen „Stehen“-Beschriftung unterbrochen wird, die entsprechenden Transitionsbewegungs-Beschriftungen „Sitzen-Stehen“, „Stehen-Sitzen“ jedoch fehlen, scheint die „Stehen“-Beschriftung unplausibel und könnte entsprechend behandelt werden.

Die Frage, die sich hierbei stellt, ist, ob in jedem Fall, indem eine wahrscheinlich falsche Beschriftung identifiziert wurde, der Korrekturschritt eindeutig ist. Wie soll beispielsweise entschieden werden, wenn eine „Sitzen-Stehen“-Beschriftung eine „Sitzen“-Beschriftung unterbricht? Ist dann eindeutig, dass die „Sitzen-Stehen“-Beschriftung falsch ist oder könnte es sein, das die Person tatsächlich kurz aufgestanden ist, die „Stehen-Sitzen“-Transition jedoch nicht erkannt wurde?

10.3. Geringe Klassifikationsgenauigkeit

Einige verwendete Klassifikatoren geben eine eindeutige Klassenzuordnung als Ausgabe zu einem eingegebenen Merkmalsvektor, während Andere einen Wahrscheinlichkeitsvektor als Ausgabe liefern, der für jede Klasse angibt, wie wahrscheinlich sie dem Merkmalsvektor zugeordnet wird.

Aktuell wird stattdessen im letzteren Fall eine Beschriftung für die Klasse erzeugt, die die höchste Wahrscheinlichkeit hat. Es wäre aber denkbar, dass in dem Fall, dass bei der

Klassifikation eines Datenpunkts tendenziell nur geringe Wahrscheinlichkeiten für alle Klassen erreicht werden, keine Beschriftung erzeugt wird, da sonst gegebenenfalls falsche Beschriftungen erzeugt werden.

10.4. Implementierung weiterer Techniken

Im Rahmen der Seminararbeiten wurden viele verschiedene Techniken des Maschinellen Lernens vorgestellt. Es konnten jedoch nicht alle Techniken implementiert und evaluiert werden. Alle aktuell implementierten Verfahren werden dem überwachten Lernen zugeordnet, bei dem jedem Trainingsbeispiel eine Zielklasse zugeordnet sein muss.

Der Projektgruppe wurden viele Datensätze bereitgestellt, die zum Training der Modelle verwendet werden können. Jedoch sind diese Datensätze alle unbeschriftet, sodass viel Zeit investiert werden musste, um diese Datensätze manuell zu beschriften (Zuordnung der Daten zu einer Zielklasse). Eine automatische Generierung verschiedener Beschriftungen hat sich als eher unpraktikabel erwiesen (Abschnitt 5.5).

Die Menge an Trainingsdaten ist somit aktuell relativ beschränkt, weshalb Lernalgorithmen ausgeklammert wurden, die eine große Menge an Trainingsbeispielen fordern. Dazu zählen beispielsweise die Recurrent Multilayer Perceptrons und die Convolutional Neural Networks (siehe Unterabschnitt D.3). Diese Techniken erreichen eventuell bessere Klassifikations-Ergebnisse, als die aktuell verwendeten Verfahren, insofern kann ihre Verwendung bei größerer Menge an Trainingsdaten lohnend sein.

Interessant wäre auch der Einsatz von Co-Training (siehe Unterabschnitt D.1), da dieses Verfahren zu dem semi-überwachten Lernen zählt. Das heißt, dass die Trainingsmenge sowohl aus beschrifteten als auch unbeschrifteten Daten besteht. Was unter dem Gesichtspunkt, dass es aktuell eine gute Basis an beschrifteten Datensätzen und eine große Menge an unbeschrifteten Datensätzen gibt, von wesentlichem Interesse sein könnte.

Neben dem Einsatz anderer Verfahren des Maschinellen Lernens könnten beispielsweise auch andere Merkmale oder andere Dimensionsreduktionsmethoden (z. B. Linear discriminant analysis (LDA)) als die aktuell Verwendeten umgesetzt werden.

10.5. Bestimmung weiterer Bewegungsparameter

Aktuell werden die Fersenaufschläge berechnet und somit einzelne Schritte bestimmt. Zudem werden die folgenden Gangparameter bestimmt:

- minimale, maximale und durchschnittliche Schrittdauer,
- minimale, maximale und durchschnittliche Doppelschrittdauer,
- Kadenz.

Es könnte von wesentlichem Interesse sein weitere Gangparameter wie Schrittlänge und Doppelschrittlänge zu bestimmen. Da sich damit zum einen die zurückgelegte Strecke und durch Kombination mit der Schrittdauer die Gehgeschwindigkeit bestimmen ließen, welche insbesondere für eine Reihe von Assessments als auch für Alltags-Untersuchungen wesentliche Parameter darstellen. In den Ausarbeitungen Unterabschnitt D.6 und Unterabschnitt D.7 werden auf entsprechende weitere Gangparameter eingegangen.

Neben den Gangparametern könnten auch zu Bewegungsmustern wie z. B. „Springen“, „Sitzen-Stehen“, „Treppensteigen“ Parameter bestimmt werden um eine umfassendere Fitness-Bestimmung zu ermöglichen. Auch eine detaillierte Betrachtung von Assessments, die im Stehen stattfinden, ist möglich. Aktuell werden sie lediglich als „Stehen“ erkannt.

10.6. Validierung der Gangparameterergebnisse

Die aktuellen Ergebnisse der Ganganalyse sind nicht validiert, da keine Referenzen vorliegen. Um eine solche Validierung durchzuführen, sollte der Gang verschiedener Personen sowohl per Sensorgürtel und durch ein oder mehrere andere Systeme wie z. B. ein Motion-Capture-System, Sensoranzug oder schlicht einer Kamera erfasst werden. Über diese Systeme können dann die zeitlichen Parameter bestimmt und mit den über den Sensorgürteldaten berechneten Parametern verglichen werden.

10.7. Darstellung von Echtzeitdaten

Durch die Entwicklung eines Toshiba-Sensorsystems können Messdaten über eine kabellose Verbindung in Echtzeit empfangen werden. Es wäre wünschenswert, wenn diese

Echtzeitdaten direkt von der Applikation dargestellt werden könnten, aktuell ist dies nicht gegeben. Es müssten dafür einerseits der Empfang der Daten über ein kabelloses Protokoll (Bluetooth, WiFi, ...) implementiert, als auch die Integration von Echtzeitdaten in die Anwendung umgesetzt werden.

10.8. Daheimdaten

Die Projektgruppe hat sich im Wesentlichen nur auf die Daten konzentriert, die bei Assessment-Aufnahmen gesammelt wurden, und die Daheim-Daten, die im Alltag des Probanden aufgezeichnet wurden, nicht näher betrachtet. Die entwickelten Algorithmen lassen sich selbstverständlich auch auf die Daheim-Daten anwenden. Es ist allerdings zu erwarten, dass Bewegungen in der Daheim-Situation unter Umständen anders ausgeführt werden, als während einer Assessment-Situation. Insofern werden die Klassifikationsergebnisse in Alltags-Situation für Klassifikatoren, die über Assessment-Trainingsdaten trainiert wurden, vermutlich schlechter ausfallen als in Assessment-Situationen. Zumal im Alltag eine Reihe unbekannter Bewegungen auftauchen können, die den erkannten Bewegungsmustern ähneln und somit falsch klassifiziert werden würden.

Es könnte zudem zu Problemen mit dem benötigten Arbeitsspeicher kommen, da die Daheim-Daten häufig über einen ganzen Tag aufgenommen wurden und somit einen entsprechend größeren Datenumfang aufweisen. Gegebenenfalls ist eine Vorbereitung der Daten z. B. per Aufteilen nach Zeitintervallen vor dem Import notwendig.

In diesem Projekt wurden die Daheim-Daten nicht betrachtet, da es keine Referenzinformationen für diese Datensätze gibt. Zudem wird die manuelle Beschriftung von Daten dadurch erschwert, dass der Proband Bewegungen im Alltag beliebig ausführt und das Spektrum unbekannter Bewegungen deutlich größer ist. Bei den Assessment-Aufnahmen kann sich am Assessment-Ablauf orientiert werden, sodass die Bewegungsmuster anhand der Daten leicht zu identifizieren sind. Dies ist bei den Daheim-Daten nicht ohne Weiteres möglich.

Die Betrachtung der Daheim-Daten und insbesondere deren Beschriftung scheint sinnvoll, um für die Daheim-Situation „realistischere“ Trainingsdaten und in der Folge „realistischere“ Klassifikatoren und Klassifikationsergebnisse zu gewinnen. Hierfür ist allerdings vermutlich eine Form einer möglichst detaillierten, ausführlichen Dokumentation des

Daheim-Ablaufes (z. B. per Kamera, um einen präzisen Zeitbezug herzustellen) notwendig und vermutlich ist die manuelle Generierung der entsprechenden Trainingsdaten selbst dann äußerst aufwendig.

Literatur

[ABMP+10] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *23th International Conference on Architecture of Computing Systems 2010*, pages 1–10, Feb 2010.

[AFV14] Boragan Aruoba and Jesus Fernandez-Villaverde. A comparison of programming languages in economics, 05 2014.

[Bas12] Youssef Bassil. A comparative study on the performance of the top dbms systems. *Journal of Computer Science & Research (JCSCR), Vol. pp.20-31, 2012*, Journal pp.20-31, 2012:JournalofComputerScience&Research(JCSCR),Vol.1,No.1, pp.20–31,2012, May 2012.

[Bor05] Jürgen Bortz. *Statistik für Human- und Sozialwissenschaftler*. Springer Medizin Verlag Heidelberg, Heidelberg, 2005.

[Bosa] Bosch Sensortec. *BMA180 Digital, triaxial acceleration sensor*.

[Bosb] Bosch Sensortec. *BMP085 Digital pressure sensor*.

[Bro14] Jason Brownlee. Classification accuracy is not enough: More performance measures you can use, 3 2014.

[CBHK02] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 2002.

[CMR05] Olivier Cappe, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer, 2005.

[DH] P. Drezet and R.F. Harrison. Suppport vector machines for system identification. Technical report, Department of Automatic Control & Systems Engineering, The University of Sheffield, Sheffield, S1 3JD, ????

[dJufV94] Bundesministeriums der Justiz und für Verbraucherschutz. Gesetz über medizinprodukte, 08 1994.

[EPRF12] Rob Eden, Johan Parent, Jeff Randall, and Eric D. Friedman. Trove - high performance collections for java, 03 2012.

[Ert13] Wolfgang Ertel. *Grundkurs künstliche Intelligenz*. Springer Vieweg, 2013.

[fNR04] CERN European Organization for Nuclear Research. Colt, 09 2004.

[Gem93] Der Rat Der Europäischen Gemeinschaften. Richtlinie 93/42/ewg des rates vom 14. juni 1993 über medizinprodukte, 07 1993.

[GH16] Joachim Goll and Cornelia Heinisch. *Java als erste Programmiersprache*. Springer Vieweg, 8 edition, 2016.

[Gho06] Ali Ghodsi. Dimensionality reduction a short tutorial, 2006.

[Gra09] Günter Gramlich. Eine einführung in matlab, 06 2009.

[Gun98] Stefe R. Gunn. Support vector machines for classification and regression. Technical report, Faculty of Engineering, Science and Mathematics, School of Electronics and Computer Science, 1998.

[Hä00] Michael Härtfelder. Kaffee mit vitamin c - jni als ultimativer zweikomponentenkleber, 2000.

[Han10] Eckhart Hanser. *Agile Prozesse: Von XP über Scrum bis MAP* -. Springer-Verlag, Berlin Heidelberg New York, 2010.

[Her10] Gernot Herbst. Unscharfe verfahren für lokale phänomene in zeitreihen. *Deutsche Nationalbibliothek*, 2010.

[HRS09] Peter Hruschka, Chris Rupp, and Gernot Starke. *Agility kompakt* -. Springer Science & Business Media, Berlin Heidelberg, 2. aufl. edition, 2009.

[Hyn10] Rob J. Hyndman. Why every statistician should know about cross-validation, 10 2010.

[IEE85] IEEE standard for binary floating-point arithmetic, 1985.

[KBB+15] Rudolf Kruse, Christian Borgelt, Christian Braune, Frank Klawonn, Christian Moewes, and Matthias Steinbrecher. *Computational Intelligence*. Springer Vieweg, 2015.

[KCHP01] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[KCHP04] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting

time series: A survey and novel approach. *Data mining in time series databases*, 57:1–22, 2004.

[Kem11] Arnfried Kemnitz. *Mathematik zum Studienbeginn*. Springer Fachmedien Wiesbaden: Wiesbaden, Wiesbaden, 2011.

[mat12] matlab4java.wordpress.com. Calling matlab from java, 05 2012.

[Max12] Dominik Maximini. *Scrum - Einfürung in der Unternehmenspraxis - Von starren Strukturen zu agilen Kulturen*. Springer-Verlag, Berlin Heidelberg New York, 1. aufl. edition, 2012.

[MEM] MEMSIC, Inc. *Ultra Small 3-axis Magnetic Sensor, With I2C Interface MMC314xMR*.

[MG] Prof. Dr. Günter W. Maier and Prof. Dr. Robert Gillenkirch. Entscheidungsbaum.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[MSGK14] Dan Morris, T Scott Saponas, Andrew Guillory, and Ilya Kelner. Recofit: using a wearable sensor to find, recognize, and count repetitive exercises. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3225–3234. ACM, 2014.

[Nia12] Khalil Niazmand. *Medizinische Bewegungserkennung durch Beschleunigungsmessung*. dissertation, Technischen Universität München, 2012.

[NXP14] NXP. User manual - um10204, 04 2014.

[(Or13] Monica Pawlan (Oracle). What is javafx?, 04 2013.

[Pó] Barnabás Póczos. Introduction to machine learning cmu-10701 - risk minimization.

[PKW] Prof. Dr. Iris Pigeot-Kübler and Dipl. Cornelia Wiedemeyer. Aequipa.

[Pow11] David Martin Ward Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, 2(1):37–63, 2011.

[Run10] Thomas A. Runkler. *Data Mining*. Vieweg+Teubner, 2010.

[RWH14] Cord-Christian Rossow, Klaus Wolf, and Peter Horst. *Handbuch der Luftfahrzeugtechnik*. Carl Hanser Verlag GmbH Co KG, 2014.

[SAHB14] Rosaria Silipo, Iris Adae, Aaron Hart, and Michael Berthold. Seven techniques for dimensionality reduction, 2014.

[sB16] TIOBE software BV. Tiobe index for april 2016, 04 2016.

[Sch97] Jeff Schneider. Cross validation, 2 1997.

[Sch11] Universitäre Fernstudien Schweiz. Wozu kann man matlab gebrauchen?, 12 2011.

[Sch13] Paul Schmitz. *Die Wirksamkeit von Programmiersprachen*. Springer-Verlag, 2013.

[Sla14] Thomas Slawig. Objektorientierte programmierung - grundlagen und -begriffe, 2014.

[sld] scikit-learn developers. 3.1. cross-validation: evaluating estimator performance.

[Smi02] Lindsay I. Smith. A tutorial on principal components analysis, 2002.

[Ull12] Christian Ullenboom. Java native interface (jni), 2012.

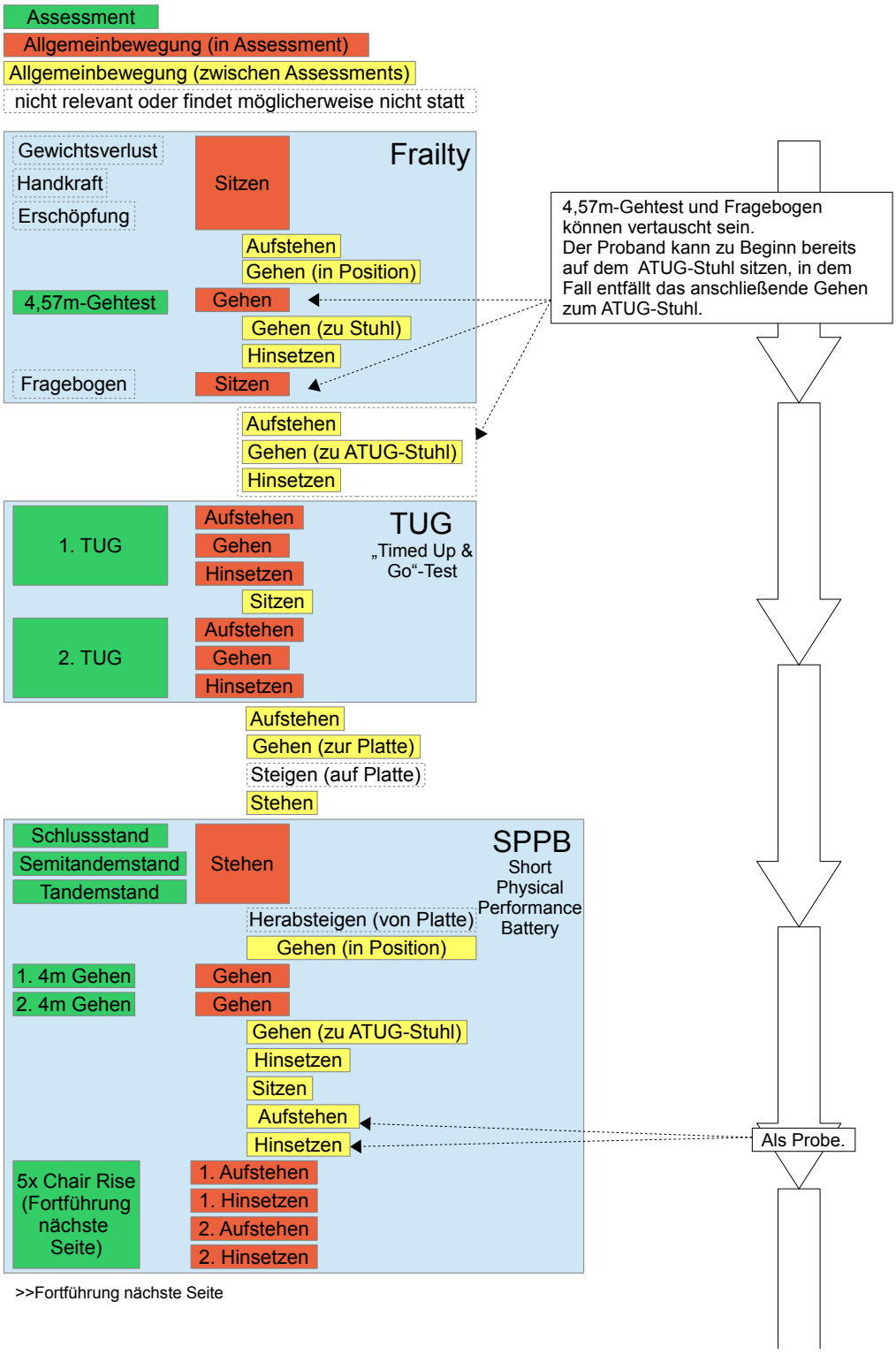
[Vig] Sebastiano Vigna. fastutil: Fast and compact type-specific collections for java.

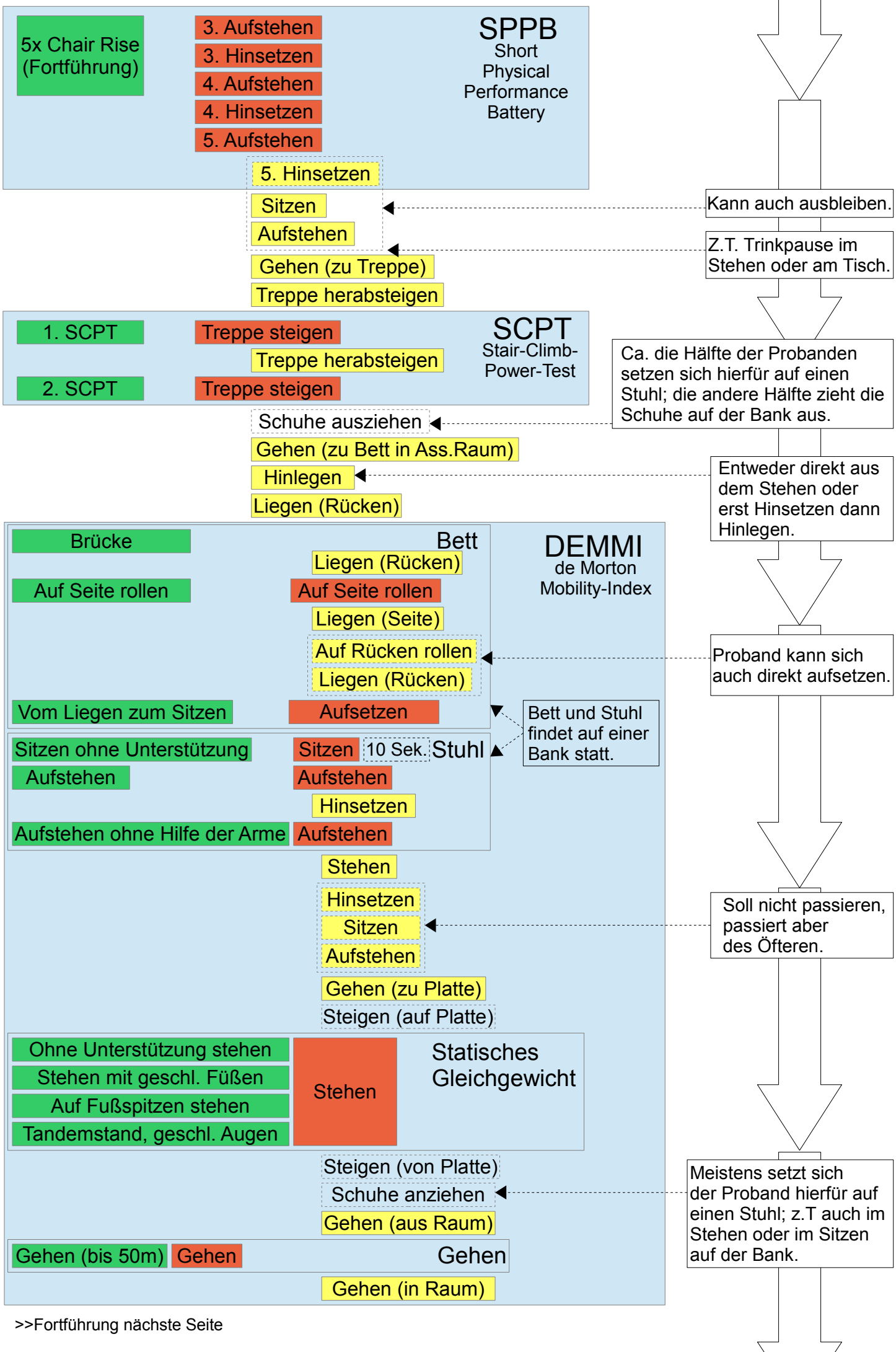
[Wä12] Kai Wähner. Sprachen für die jvm im zusammenspiel mit java, 02 2012.

[W3C08] Extensible markup language (xml) 1.0 (fifth edition), 2008.

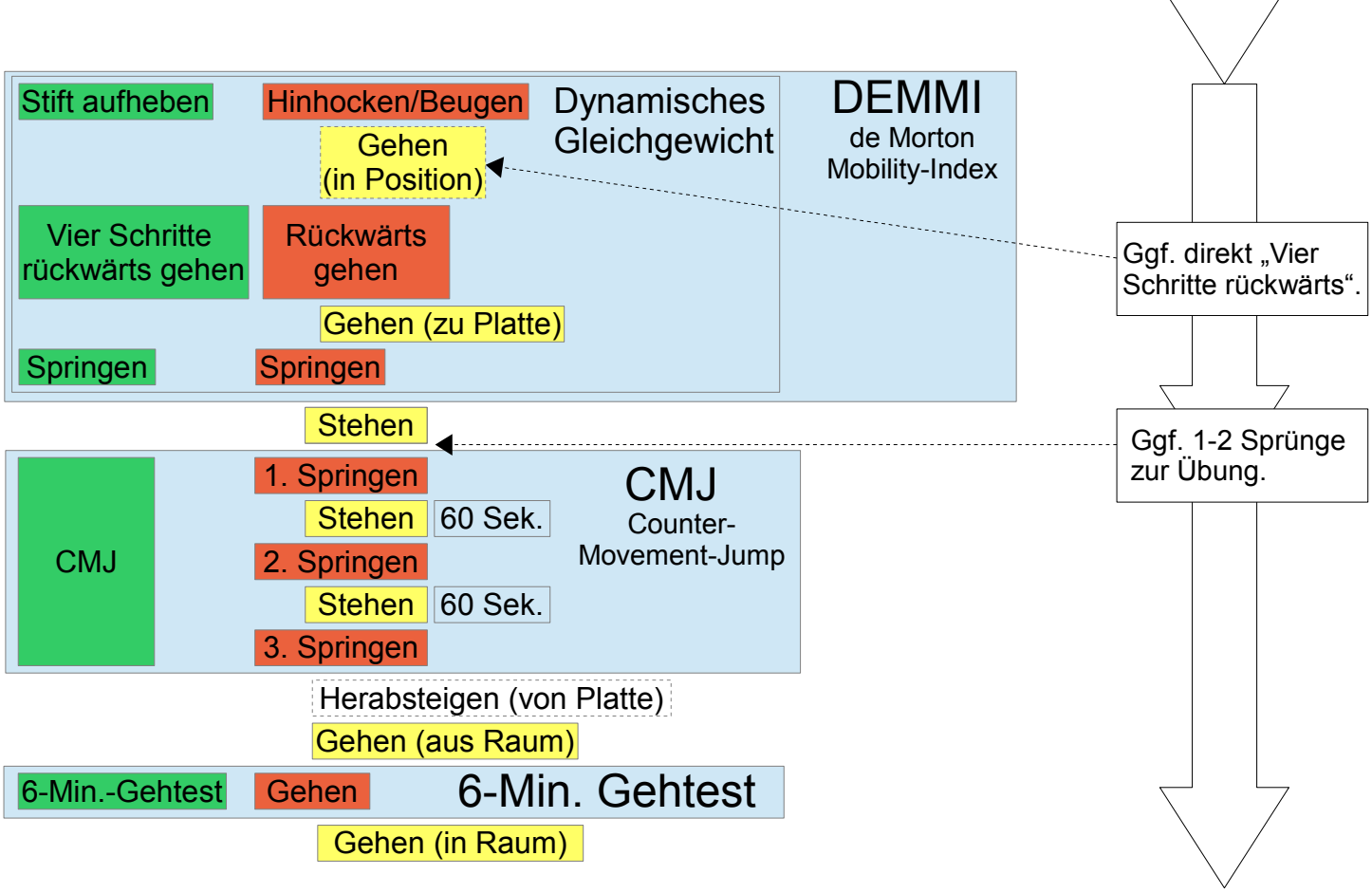
Anhang

A. Ablauf der Assessments





>>Fortführung nächste Seite


Stift aufheben
Hinhocken/Beugen
Dynamisches Gleichgewicht
DEMMI
de Morton Mobility-Index
Gehen (in Position)
Vier Schritte rückwärts gehen
Rückwärts gehen
Gehen (zu Platte)
Springen
Springen
Ggf. direkt „Vier Schritte rückwärts“.
Stehen
Ggf. 1-2 Sprünge zur Übung.
CMJ
1. Springen
Stehen
60 Sek.
2. Springen
Stehen
60 Sek.
3. Springen
CMJ
Counter-Movement-Jump
Herabsteigen (von Platte)
Gehen (aus Raum)
6-Min.-Gehtest
Gehen
6-Min. Gehtest
Gehen (in Raum)

B. Assessment- und Befragunsbögen

Einstiegsfragen

(Herz- oder Hirn-)Schrittmacher oder implantierten Defibrillator, elektronische Implantate?
□ ja □ nein

Schwere Erkrankungen (z.B. Lunge, Niere, Herz)
□ ja □ nein

Haben Sie Beschwerden unter geringen alltäglichen Belastungen?
(Z.B. eine Etage Treppen steigen, 10 min. spazieren gehen, einen vollen Wäschekorb durch die Wohnung tragen?)
□ ja □ nein

Haben Sie schnell Atemnot bei Belastung (Herz/Lunge)?
□ ja □ nein

Wie oft gehen Sie zum Arzt?

Haben Sie einen Diabetes?
□ ja □ nein

Falls ja, kommen Sie damit im Alltag gut zurecht? (stabile Werte, wenig Ausreißer)
□ ja □ nein

Fähigkeit selbstständig in den Studienraum zu kommen
□ ja □ nein

TuG <20 Sekunden Zeit:

Drop out
□ ja □ nein Grund:

Messung durch:

Uhrzeit Assessment Beginn:

Uhrzeit Assessment Ende:

Assessment Nr.: ☐ 1 ☐ 2 ☐ 3

Allgemeine Informationen

Daten in Ordner Probandenliste eintragen!

Daten im <u>Programm</u> eingeben.

Temperatur [°C]	
Luftfeuchtigkeit [%]	
Assessmentdurchführung	☐vormittags ☐mittags ☐nachmittags

Probanden ID	
Wann sind sie geboren? (YYYY) Wie alt sind Sie?	
Geschlecht?	☐männlich ☐weiblich
Größe (cm) **(ohne Schuhe)**	
Gewicht (kg)	
Ihr BMI	
Bluthochdruck	☐Nein ☐Ja
Diabetes Mellitus	☐Nein ☐Ja
Schlaganfall	☐Nein ☐Ja
Schrittmacher	☐Nein ☐Ja -> keine BIA!!
Künstliches Knie	☐Nein ☐Ja ☐ li ☐ re
Künstliche Hüfte	☐Nein ☐Ja ☐ li ☐ re
COPD	☐Nein ☐Ja
Gibt es gesundheitlich etwas Neues bei Ihnen, seit wir uns das letzte Mal gesehen haben?	

Welche Medikamente nehmen Sie momentan ein?	
Haben Sie bewusst etwas verändert, seit Sie an der Studie teilnehmen? (heutiger, tatsächlicher Stand)	☐ körperliche Aktivität verändert ☐ viel weniger ☐ etwas weniger ☐ etwas mehr ☐viel mehr ☐ Sonstiges: _____ ☐ viel weniger ☐ etwas weniger ☐ etwas mehr ☐viel mehr ☐ nichts verändert
Rauchen Sie?	☐Nein ☐Ja
	Wenn ja wie viel? ☐ leichter Raucher (unter 10 Zigaretten am Tag) ☐ mittelstarker Raucher (zwischen 10 und 20 Zigaretten am Tag) ☐ Starker Raucher (20 oder mehr Zigaretten am Tag)
Trinken sie Alkohol?	☐Nein ☐Ja
	Wenn ja wie oft? ☐ weniger als 1 Mal die Woche ☐ 1-3 mal die Woche ☐ 3-5 mal die Woche ☐ 5-7 mal die Woche
Sind Sie in den letzten 12 Monaten gefallen?	☐Nein ☐Ja
Besteht eine Pflegestufe ?	☐keine ☐ 0 ☐ 1 ☐ 2 ☐ 3

Jetzt bitte zur Liege gehen!

iADL

(nach Lawton und Broody, 1969)

Telefon:	
Benutzt Telefon aus eigener Initiative, wählt Nummern	☐ 1
Wählt einige bekannte Nummern	☐ 1
Nimmt ab, wählt nicht selbständig	☐ 1
Benutzt das Telefon überhaupt nicht	☐ 0
Einkaufen:	
Kauft selbständig die meisten benötigten Sachen ein	☐ 1
Tätigt wenige Einkäufe	☐ 0
Benötigt bei jedem Einkauf Begleitung	☐ 0
Unfähig zum Einkaufen	☐ 0
Kochen:	
Plant und kocht erforderliche Mahlzeit selbständig	☐ 1
Kocht erforderliche Mahlzeit nur nach Vorbereitung durch Drittpersonen	☐ 0
Kocht selbständig, hält aber benötigte Diät nicht ein	☐ 0
Benötigt vorbereitete und servierte Mahlzeiten	☐ 0
Haushalt:	
Hält Haushalt instand oder benötigt zeitweise Hilfe bei schweren Arbeiten	☐ 1
Führt selbständig kleine Hausarbeiten aus	☐ 1
Führt selbst kleine Hausarbeiten aus, kann aber Wohnung nicht rein halten	☐ 1
Benötigt Hilfe in allen Haushaltsverrichtungen	☐ 1
Nimmt überhaupt nicht teil an täglichen Verrichtungen im Haushalt	☐ 0
WÄSCHE:	
Wäscht sämtliche eigene Wäsche	☐ 1
Wäscht kleine Sachen	☐ 1
Gesamte Wäsche muss auswärts versorgt werden	☐ 0
TRANSPORTMITTEL:	
Benutzt unabhängig öffentliche Transportmittel, eigenes Auto	☐ 1
Bestellt und benutzt selbständig Taxi, jedoch keine öffentlichen Transportmittel	☐ 1
Benutzt öffentliche Transportmittel in Begleitung	☐ 1
Beschränkte Fahrten im Taxi oder Auto in Begleitung	☐ 0
Reist überhaupt nicht	☐ 0
MEDIKAMENTE:	
Nimmt Medikamente in genauer Dosierung und zum korrekten Zeitpunkt eigenverantwortlich	☐ 1
Nimmt vorbereitete Medikamente korrekt	☐ 0
Kann korrekte Einnahme von Medikamenten nicht handhaben	☐ 0
GELDHAUSHALT:	
Regelt finanzielle Geschäfte selbständig (Budget, Schecks, Einzahlung, Gang zur Bank)	☐ 1
Erledigt tägliche kleinere Ausgaben, benötigt aber Hilfe bei Einzahlung, Bankgeschäften	☐ 1
Ist nicht mehr fähig mit Geld umzugehen	☐ 0
SUMME	

Barthel Index

(nach Mahoney und Barthel, 1965) **Hamburger Manual**

A1.	Essen	Isst komplett selbständig, benutzt Geschirr und Besteck	☐ 10
		Braucht Hilfe bei mundgerechter Zubereitung, z. B.	☐ 5
		Völlig hilfsbedürftig	☐ 0
A2.	Bett/ (Roll-) Stuhltransfer	Unabhängig aus liegender Position in (Roll-)Stuhl und	☐ 15
		Aufsicht oder geringe Hilfe (ungeschulte Hilfe)	☐ 10
		Erhebliche Hilfe (geschulte Laienhilfe, professionelle	☐ 5
		Bettlägerig, wird faktisch nicht aus dem Bett transferiert	☐ 0
A3.	Waschen	Komplett selbständig beim Waschen von Gesicht und Händen, Kämmen, Zähneputzen, Rasieren, Frisieren	☐ 5
		Nicht selbständig bei o. g. Tätigkeiten	☐ 0
A4.	Toilettenbenutzung	Selbstständige Nutzung von Toilette, inkl. Spülung	☐ 10
		Benötigt Hilfe oder Aufsicht, z. B. bei Gleichgewicht, Kleidung aus- und anziehen, Spülung, Reinigung	☐ 5
		Kann nicht auf Toilette / Nachtstuhl	☐ 0
A5.	Baden	Selbständiges Baden/Duschen, inkl. Reinigen,	☐ 5
		Badet oder duscht mit Hilfe	☐ 0
A6.	Aufstehen & Gehen	Ohne Aufsicht/Hilfe vom Sitz in den Stand kommen und mind. 50 m ***ohne Gehwagen*** gehen (aber ggf. Stöcken/Gehstützen)	☐ 15
		Ohne Aufsicht/Hilfe vom Sitz in den Stand kommen und mind. 50 m ***mit Gehwagen*** gehen	☐ 10
		Mit Hilfe oder Gehwagen vom Sitz in den Stand kommen und Strecken im Wohnbereich bewältigen	☐ 5
		Kann sich nicht (mind. 50 m) fortbewegen	☐ 0
A7.	Treppensteigen	Selbständig (auch mit Gehilfe) ein Stockwerk hinauf- und hinuntersteigen	☐ 10
		Mit Hilfe ein Stockwerk hinauf und hinunter	☐ 5
		Kann auch mit Hilfe nicht Treppen steigen	☐ 0
A8.	An- und Auskleiden	Selbständig bei Kleidung und Schuhe anziehen, in angemessener Zeit	☐ 10
		Kleidet mind. den Oberkörper in angemessener Zeit selbständig an	☐ 5
		Völlig hilfsbedürftig	☐ 0
A9.	Stuhlkontrolle	Ständig kontinent	☐ 10
		Gelegentlich inkontinent, maximal 1x/Woche	☐ 5
		Mehr als 1x/Woche stuhlinkontinent	☐ 0
A10.	Urinkontrolle	Ständig kontinent, ggf. unabhängig bei DK/Cystofix.	☐ 10
		Gelegentlich inkontinent, max. 1x/Tag oder benötigt Hilfe bei der Versorgung des Harnkathetersystems	☐ 5
		Mehr als 1x/Tag harninkontinent	☐ 0
	Ergebnis	**Gesamtpunktzahl (max. 100)**	

LUCAS Funktions-Index

Markerfragen FRAIL (6 Punkte maximal)

1. Haben Sie in den letzten 6 Monaten unbeabsichtigt 5kg oder mehr abgenommen?	☐ nein (0) ☐ ja (1)
Haben Sie in den letzten 12 Monaten aus gesundheitlichen oder körperlichen Gründen die Art und Weise verändert, mit der Sie **2.) ...einen Kilometer zu Fuß gehen?** z.B. Sie gehen seit dem letzten Jahr langsamer oder vorsichtiger, haben Ihre Haltung oder Ihren Schritt verändert, verwenden seit dem letzten Jahr neu einen Stock oder andere Gehhilfen oder legen häufiger Ruhepausen ein als vorher. **3.) …10 Treppenstufen steigen?** z.B. Sie steigen seit dem letzten Jahr langsamer oder vorsichtiger, haben Ihren Schritt oder Ihre Gehweise verändert, legen häufiger Ruhepausen ein oder benutzen in den letzten 12 Monaten häufiger das Geländer. **4.) …in ein Auto, in einen Bus oder in einen Zug ein- oder aussteigen?** z.B. Sie stützen sich in den letzten 12 Monaten vermehrt mit den Händen ab oder ziehen sich mit den Armen hoch, Sie nehmen sich mehr Zeit oder lassen sich neu seit dem letzten Jahr von anderen helfen.	☐ nein (0) ☐ ja (1) ☐ nein (0) ☐ ja (1) ☐ nein (0) ☐ ja (1)
5.) An wie vielen Tagen der letzten Woche waren Sie aus irgendeinem Grund <u>zu Fuß außerhalb Ihrer Wohnung</u> unterwegs wie z.B. zum Spazieren, Einkaufen, für Besuche oder andere Tätigkeiten?	☐ Nie (1) ☐ 1-2 Tage (1) ☐ 3-4 Tage (0) ☐ 5-7 Tage (0)
6.) Sind Sie im Laufe der letzten 12 Monate jemals hingefallen?	☐ nein (0) ☐ ja (1)

Markerfragen ROBUST (6 Punkte maximal) und **Markerfragen FIT**

		a	b
1.a) Etwa 500 Meter zu Fuß gehen	☐ Selbstständig ohne Schwierigkeiten (1) ☐ Selbstständig aber mit Schwierigkeiten (0) ☐ Möglich nur mit Hilfsmittel (0) ☐ Möglich aber nur mit Hilfsperson (0) ☐ Nicht möglich (stark gehbehindert oder Rollstuhlfahrer) (0)		
1.b) Fahren Sie Fahrrad?	☐ Nein, nie gelernt (0) ☐ Nein, aufgehört (0) ☐ Ja, gelegentlich (1) ☐ Ja, regelmäßig, mindestens einmal pro Woche (1)		

2.) An wie vielen Tagen der letzten Woche waren Sie aus irgendeinem Grund <u>zu Fuß außerhalb Ihrer Wohnung</u> unterwegs wie z.B. zum Spazieren, Einkaufen, für Besuche oder andere Tätigkeiten?	☐ Nie (0) ☐ 1-2 Tage (0) ☐ 3-4 Tage (1) ☐ 5-7 Tage (1)		
3.a) An wie vielen Tagen der letzten Woche haben Sie <u>mäßig anstrengenden Sport</u> getrieben wie z.B. Gymnastik (auch Wassergymnastik), Freizeitschwimmen, Tanzen, Wandern, Radfahren zum Einkaufen oder zum Vergnügen o.ä.? **b) Treiben Sie regelmäßig mindestens 1x pro Woche <u>mäßig anstrengenden Sport</u>?**	☐ Nie (0) ☐ 1-2 Tage (1) ☐ 3-4 Tage (1) ☐ 5-7 Tage (1) ☐ **Nein** ☐ Ja: **Krafttraining** z.B. Wassergymnastik, Krafttraining an Geräten, Therabandübung ☐ Ja: **Balancetraining** z.B. Radfahren, Tai Chi, Tanzen ☐ Ja: **Ausdauertraining** z.B. Wandern im Gelände, Walking, Jogging, Schwimmen *(Dieser FIT-Marker ist erfüllt, wenn mindestens eine „Ja"-Kategorie angekreuzt wurde)*		
4.a) An wie vielen Tagen der letzten Woche haben Sie <u>stark anstrengenden Sport</u> getrieben wie z.B. Joggen, Sportschwimmen, Radfahren (als Sport oder auf Hometrainer), Tennis, Aerobic, Ballsport, Skifahren (Alpin und Langlauf) oder ähnliches? **b) Treiben Sie regelmäßig mindestens 1x pro Woche <u>stark anstrengenden Sport</u>?**	☐ Nie (0) ☐ 1-2 Tage (1) ☐ 3-4 Tage (1) ☐ 5-7 Tage (1) ☐ Nein ☐ Ja: **Krafttraining** z.B. Zirkeltraining an Krafttrainings-Geräten, Bankdrücken ☐ Ja: **Balancetraining** z.B. Turniertanz ☐ Ja: **Ausdauertraining** z.B. Sportschwimmen, Spinning *(Dieser FIT-Marker ist erfüllt, wenn mindestens eine „Ja"-Kategorie angekreuzt wurde)*		
5.) Leisten sie zurzeit ehrenamtliche Arbeit (freiwilliges bürgerschaftliches Engagement)?	☐ Nein (0) ☐ Ja, Teilzeit (1) ☐ Ja, Vollzeit (1)		
6.) Schränken Sie gewisse Tätigkeiten ein, weil Sie Angst haben, hinzufallen?	☐ Nein (1) ☐ Ja (0)		
SUMME:			

Auswertung LUCAS Funktions-Index (international a)		
Summe	3 - 6 **ROBUST**-Faktoren	0 - 2 **ROBUST**-Faktoren
0 – 2 **FRAIL**-Faktoren	Robust ☐	preFrail ☐
3 – 6 **FRAIL**-Faktoren	postRobust ☐	Frail ☐

* Zu dem ROBUST Marker gehören die Fragen 1a) und 2-6.

Auswertung LUCAS Funktions-Index (regional b)		
Summe	3 - 6 **FIT**-Faktoren	0 - 2 **FIT**-Faktoren
0 – 2 **FRAIL**-Faktoren	Fit ☐	preFrail ☐
3 – 6 **FRAIL**-Faktoren	postFit ☐	Frail ☐

* Zu dem FIT Marker gehören die Fragen 1b), 2, 3b), 4b), 5 und 6.

BIA

RZ		**TBW**		**FFM**		**MM**	
XC		**ECW**		**BCM**			
PA		**ICW**		**FM**			

Blutdruck- Messung:

	1. Messung	Ggf. 2. Messung
Zeit		
Blutdruck **(links)**		
Puls		

		Rechts	**Links**
	**Wadenumfang 37.5 cm**		
	Maximaler Wadenumfang		

Frage an Proband, ob er Lust hat, <u>nach</u> VERSA an einer weiteren Studie teilzunehmen:

☐ ja ☐ nein

Gürtel – Nr:

Frailty Kriterien

(nach Fried et al. 2001)

(1 Punkt, wenn Kriterium erfüllt, 0 Punkte, wenn nicht)

1. Gewichtsverlust:

Gewicht vor 12 Monaten _____ kg (z.B.: auch 62-63 kg)

Ungewollt > 4,5 kg Verlust im vergangenen Jahr bzw. ungewollt >5 % im Follow-up/Jahr ☐ Pkt.

2. Schwäche: Handkraftmessung (Dynamometer, isometrisch), Mittelwert von 3 Messungen, dominanten Mittelwert für Score berücksichtigen.

☐ **Rechtshänder** ☐ **Linkshänder**

	Links [kg]	Rechts [kg]
Messung 1		
Messung 2		
Messung 3		
Mittelwert		

☐ Pkt.

Teilnehmer hatte **Schwierigkeiten**? ☐ Nein ☐ Ja

Welche: _____

3. Erschöpfung:

In der vergangenen Woche:	(0) < 1d	(1) 1-2 d	(2) 3-4 d	(3) 5-7 d
• Empfand ich alles als anstrengend				
• Bin ich nicht „in Gang“ gekommen				

☐ Pkt.

Es gibt einen Punkt sobald der Proband mind. einmal (2) 3-4 d (3) 5-7 d angibt.

4. Ganggeschwindigkeit

4,57 m gehen in gewöhnlicher Geschwindigkeit _____ Sekunden

☐ Pkt.

5. Körperliche Aktivität

Freizeitaktivität	**Intensitäts-Code (I)**	**Gesamtdauer D [min] der letzten 2 Wochen**	**I x D [kcal]**
Spazierengehen	3,5		
Rasen mähen	6,0		
Gartenarbeit	4,5		
Wandern	6,0		
Fahrradfahren	4,0		
Tanzen	5,5		
Joggen/Walking	6,0		
Schwimmen	6,0		
Segeln	3,0		
Tennis	8,0		
Badminton	7,0		
Golf zu Fuß, ziehend	5,0		
Summe Energieverbrauch I x D			

Energieverbrauch pro Woche: ∑ (I x D) / 2: _____ kcal/Woche ☐ Pkt.

Gesamt-Score

Klassifizierung

Score	**Klassifizierung**	
0	healthy	□
1 – 2	pre-frail	□
> 3	frail	□

☐ Pkt.

Grenzwerte für die einzelnen Frailty-Kriterien

Timed „Up & Go“

(nach Podsiadlo & Richardson, 1991)

	Es wurde eine Gehhilfe benutzt	Nein ☐	Ja, welche:
	Benötigte Zeit in Sekunden 1. Versuch		**Sek.**
	Benötigte Zeit in Sekunden 2. Versuch		**Sek.**

Kategoriale Auswertung

□ unter 10 Sekunden	keine Einschränkungen
□ 11 bis 19 Sekunden	Mobilität eingeschränkt
□ 20 bis 29 Sekunden	ausgeprägte Mobilitätseinschränkung

Short Physical Performance Batterie

(nach Guralnik et al. 1994)

Statisches Gleichgewicht

Sobald eine der 3 Übungen im Stand nicht für mindestens 10 Sekunden gehalten werden kann, wird mit dem 4m Gehtest fortgefahren. (Timing: kurz (1-2 Sek.) Positionen probeweise einnehmen, dann Messung starten, festhalten lassen, Fernbedienung drücken, „loslassen“, Stoppuhr.)

- **Schlussstand**
  □ < 10 Sekunden □ > 10 Sekunden (1 Punkt)
- **Semitandemstand**
  □ < 10 Sekunden □ > 10 Sekunden (1 Punkt)
- **Tandemstand**
  □ < 3 Sekunden □ > 3 Sekunden (1 Punkt) □ > 10 Sekunden (2 Punkte)

4-m Gehgeschwindigkeit

Der Proband soll möglichst 2m Platz vor und hinter der Gehteststrecke zum Ein- und Auslaufen haben. Der beste von 2 Versuchen zählt. Wir messen Hin- und Rückstrecke.

Zeitbedarf:

1. Versuch
2. Versuch Sekunden

□ < 4,82 (4 Punkte)
□ 4,82 – 6,20 (3 Punkte)
□ 6,21 – 8,70 (2 Punkte)
□ > 8,70 (1 Punkt)
□ Nicht fähig (0 Punkte)

5 time chair riseTest

Zeitbedarf: Sekunden

□ < 11,19	4 Punkte
□ 11,20 – 13,69	3 Punkte
□ 13,70 – 16,69	2 Punkte
□ > 16,70	1 Punkt

Punkte SPPB

Stair-Climb-Power-Test

(nach Bean et al., 2007)

1. Versuch	
2. Versuch	
Mittelwert (1. Wert + 2. Wert)/2	

de Morton Mobility Index (DEMMI)

	0	1	2
Bett			
1. Brücke	☐ nicht möglich	☐ möglich	
2. Auf die Seite rollen	☐ nicht möglich	☐ möglich	
3. Vom Liegen zum Sitzen	☐ nicht möglich	☐ geringe Unterstützung ☐ Supervision	☐ selbständig
Stuhl			
4. Sitzen im Stuhl ohne Unterstützung	☐ nicht möglich	☐ 10 Sek.	
5. Aus dem Stuhl aufstehen	☐ nicht möglich	☐ geringe Unterstützung ☐ Supervision	☐ selbständig
6. Aus dem Stuhl aufstehen, ohne die Arme zu Hilfe zu nehmen	☐ nicht möglich	☐ möglich	
Statisches Gleichgewicht (ohne Gehhilfe)			
7. Ohne Unterstützung stehen	☐ nicht möglich	☐ 10 Sek.	
8. Stehen mit geschlossenen Füßen	☐ nicht möglich	☐ 10 Sek.	
9. Auf den Fußspitzen stehen	☐ nicht möglich	☐ 10 Sek.	
10. Im Tandemstand mit geschlossenen Augen stehen	☐ nicht möglich	☐ 10 Sek.	
Gehen			
11. Wegstrecke +/- Gehhilfe *Gehhilfe (kennzeichnen):* *keine/ Gehbock/ Stock/ Rollator/ andere*	☐ nicht möglich ☐ 5m	☐ 10m ☐ 20m	☐ 50m
12. Selbständiges Gehen	☐ nicht möglich ☐ geringe Unterstützung ☐ Supervision	☐ selbständig mit Gehhilfe	☐ selbständig ohne Gehhilfe
Dynamisches Gleichgewicht (ohne Gehhilfe)			
13. Stift vom Boden aufheben	☐ nicht möglich	☐ möglich	
14. vier Schritte rückwärts gehen	☐ nicht möglich	☐ möglich	
15. Springen	☐ nicht möglich	☐ möglich	
ERGEBNISSE DER SPALTEN			

Schritte über 20m :

Schrittlänge in cm:

leicht

Sitzen ohne Unterstützung

Brücke

Ohne Unterstützung stehen

Aus dem Stuhl aufstehen

Auf die Seite rollen

Vom Liegen zum Sitzen

Stehen mit geschlossenen Füßen

Stift vom Boden aufheben

Rückwärts gehen

Wegstrecke

Aus dem Stuhl aufstehen ohne Arme

Selbständiges Gehen

Springen

Zehenstand

Tandemstand geschlossene Augen

schwer

Auf BKMP; Socken oder Barfuß

ROHWERT
(Summe der Spaltenergebnisse) /19

DEMMI- Rohwert Umrechnungstabelle

DEMMI SCORE
(MDC_{90} = 9 Punkte; MCID = 10 Punkte) /100

Rohwert	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
DEMMI score	0	8	15	20	24	27	30	33	36	39	41	44	48	53	57	62	67	74	85	100

Name Patient: _____
Datum: _____
Name Tester: _____
Unterschrift: _____

Kommentare:

Counter Movement Jump

Besonderheiten:

BKMP abspeichern!

6 Min-Gehtest

Bahn	Zeit in Sek (Hin)	Bahn	Zeit in Sek (Zurück)
1		2	
3		4	
5		6	
7		8	
9		10	
11		12	
13		14	
15		16	
17		18	

Datum: PID:

19		**20**	
21		**22**	
23		**24**	
25		**26**	
27		**28**	
29		**30**	
31		**32**	
33		**34**	
35		**36**	
37		**38**	
39		**40**	

Meter letzte angefangene Bahn (wenn letzte Bahn „voll“, dann Null eintragen)	
Strecke in Metern gesamt	

BKMP abgespeichert?		**Ja** ☐ **Nein** ☐
Nummer des Gürtels der ausgehändigt wurde:		
Wie erfolgt die Gürtelrückgabe?	☐	**Gürtel muss abgeholt werden + Kalendereintrag!**
	☐	**Gürtel wird zurückgeschickt**
	☐	Gürtel wird vom Probanden zur Post gebracht
	☐	Gürtel muss von der Post beim Probanden abgeholt werden
	☐	**Gürtel wird zurückgebracht**

Proband hat auffälliges Gangbild?	☐Nein ☐Ja
Beschreibung:	

Nachbereitung/ fehlende Dokumente
Special case?

Besonderheiten bei der Testdurchführung

- ☐ **Paket erklären (Fragebogen, Tagebuch -> 1. Datum eintragen und ggf. heften, Ladeanweisung)**
- ☐ **Termin vereinbaren und in Kalender eintragen**
- ☐ **Terminzettel und Liste mit fehlenden Dokumenten mitgeben**
- ☐ **Kontodaten („Auf welches Konto dürfen wir die Fahrtkostenerstattung überweisen?“) + USB – Kabel in Probandenpaket**
- ☐ **Eintrag Laborbuch**
- ☐ **PIDs auf Akte und Bögen?**
- ☐ **Termin auf Post-it mit Gürtel-ID?**

Uhrzeit Assessment Ende -

C. Projekttagebuch

12.04.2016:

- Einstellungen und Entscheidungen wurden im Hinblick auf Git (Commit-Style, Benennung der Tasks), JIRA (Workflow, Dokumentation der Arbeitszeit in Jira, Sicherung der Daten in Jira), Scrum (Sprintlänge, Aufwandsabschätzung), LaTeX (Dokumentation, Kardar als Beauftragter, fortführbare Dokumentation) getroffen.
- Der erste Sprint wurde gestartet, welcher hauptsächlich organisatorische Aufgaben zum Start des Projektes (Einstellen von Software, Anlegen von Seiten für Informationen, Vorbereitungen für die Arbeit) beinhaltet.

19.04.2016:

- JIRA soll auf einem Rechner der IT-Services mit der dazugehörigen Software (Confluence, Fisheye, Crucible) neu aufgesetzt werden, da der Rechner im PG-Raum dafür nicht ausgerichtet ist (Backup, kein richtiger Server). Der Workload muss nach dem Umzug übertragen werden.
- Es wurden folgende Standards bzw. Vereinbarungen festgelegt:
  - Programmiersprache: Java
  - Kodierung: UTF-8
  - Die Dokumentation soll über die gesamte Projektdauer erstellt werden und darf, solange sie für einen Bereich noch in Arbeit ist, in Stichpunkten bearbeitet werden. Anschließend soll sie ausformuliert werden.
  - Sitzungstagesordnungen sollen bis 18:00 am Sonntag hochgeladen werden. Das Protokoll soll bis 24:00 am selben Tag hochgeladen werden.
- Es wurden die folgenden Rollenverteilungen festgelegt:
  - Git-Beauftragter: Sebastian
  - Jira-/Confluence-Beauftragter: Daniel
  - Scrum-Master: Jonas
  - Projektmanagerin: Andrea
  - Test-Beauftragter: Thies

  - Latex-Beauftragter: Kardar

26.04.2016:

- Es wurde entschieden, dass die Ergebnisse in Form von Eugen Langes Software nicht für uns relevant sind. Eugens Programmiersprache- und Framework-Auswahl können wir nutzen, um unsere Auswahl zu begründen.
- Der Dienstags-Termin wird weiter als Gruppentreffen beibehalten, da ansonsten nur zweistündige Termine möglich sind.
- Es wurde ein Probe-Assessment durchgeführt mit Kjel und Christin.
- Unser Produkt wird nicht als Medizinprodukt, sondern als Fitness-Produkt kategorisiert.
- Am Donnerstag, 28.04.2016 um 14:00 Uhr, findet ein zusätzliches Gruppentreffen statt, um die Anforderungsliste zu erarbeiten.
- Für die Dokumentation der Arbeitszeit wird nun statt JIRA eine Google-Docs-Tabelle genutzt, da es zahlreiche Probleme mit JIRA in diesem Zusammenhang gab.

03.05.2016:

- Python und MATLAB sind im Vergleich zu Java und C++ zu langsam als Programmiersprachen, deswegen kommen diese nicht in Frage. Es wurde sich endgültig auf Java geeinigt, weil es alle beherrschen.
- Zur Erstellung von UML-Diagrammen wird das Programm Visual Paradigm benutzt, da es dafür kostenlose Community-Lizenzen gibt.
- Es gab Überlegungen zum ersten Mockup.
- Jedes Mitglied soll sich einen Google-Account für die Arbeitszeittabelle erstellen.

10.05.2016:

- Am Donnerstag (12.05.2016) soll ein Gruppentreffen von 14-18 Uhr auf freiwilliger Basis stattfinden.
- Das Git-Repository soll weiterhin in der ARBI laufen, zum Ende des Projektes soll dieses jedoch mit ausgehändigt werden.
- Durch Probleme mit den Beschäftigungsverhältnissen für wissenschaftliche Hilfskräfte wird der Server, den wir eigentlich bekommen sollten, uns doch nicht zur Verfügung gestellt. Über die Uni können wir jedoch das Rechencluster der Uni nutzen.

- Sebastian F. hat während der Sitzung einen Überblick über einen Zwischenstand des Projektes bekommen.
- Sebastian F. versucht noch einmal Kontakt mit Merle aufzunehmen. Sofern sie sich bis zum nächsten Dienstag (17.05.2016) nicht meldet, wird sie wieder ausgeschlossen.
- Der Review-Prozess wurde angepasst und ist in neuerer Form in den Gruppenstandards zu finden.

17.05.2016:

- Merle wird kein Mitglied der Projektgruppe.
- Das Abspeichern der Daten des Gürtels mithilfe von Floats wird in einem Binärformat geschehen, da dies ohne Datenverlust möglich ist. Intern wird zur Verbesserung der Genauigkeit mit Double weitergerechnet.
- Da für unsere Anwendung auch das Hero-Rechencluster der Universität verwendet werden kann, fällt unser zuvor geplanter Backend-Service weg, da es hier nicht möglich ist ein 24/7-Backend-Service zur Verfügung zu stellen. Zunächst soll die Berechnung lokal ausgeführt werden können. Anschließend soll sich darum gekümmert werden, dass auch die Berechnung auf Hero stattfinden kann.
- Es soll ein Projekttagebuch geführt werden, um u.a. Entscheidungen (auch Fehlentscheidungen) festzuhalten und zu begründen.

18.05.2016:

- Es wurde uns per Mail von Sebastian F. eine Liste mit Bewegungsarten geschickt, welche interessant wären von der Software erkannt zu werden. Dabei wird nicht erwartet, dass diese gesamte Liste erkannt wird.

19.05.2016:

- Es fand ein weiteres Assessment mit einer Probandin statt, welches durch Daniel und Nils begleitet wurde und wobei auch die Zeiten und die Tests dokumentiert wurden.

24.05.2016:

- Es wurden verschiedene Systemfragen geklärt, die beispielsweise das Logging für den Benutzer, das Speichern und die Einstellungen beim Start der Anwendung betreffen.
- Es wurden verschiedene Verbesserungsvorschläge für die LaTex-Dokumentation besprochen.

- Daniel, Dmitrii und Thies berichteten über ihre jeweiligen Ergebnisse ihrer Arbeiten (neue Version der Architektur, Performance-Evaluierung von primitive collections librarys und Dateiformat-Spezifikation).
- Der Vorschlag von Andrea zur Einführung einer Fairness-Vereinbarung wurde einstimmig positiv aufgenommen.

27.05.2016:

- Im Zuge der Dateiformat-Spezifikation ist aufgefallen, dass in den Sensordaten Zeitsprünge vorhanden sind. Diese begründen sich dadurch, dass die Gürtel 1-2 Tage vor dem Assessment programmiert werden, beim Laden die Messwerte 0 anzeigen und nicht mehr messen sowie nach dem Assessment einige Zeit im Schrank liegen.

30.05.2016:

- Der Vorschlag zur Fairness-Vereinbarung wurde verschriftlicht.

02.06.2016:

- In einer weitergeleiteten Mail von Sebastian F. wurden wir darüber informiert, dass das HPC-Cluster erneuert wird und sich deshalb dabei Bereiche ändern, die wohl auch unsere Software betreffen.

07.06.2016:

- Da es keinen Sinn macht mit sich ständig verändernden Anforderungen zu Arbeiten, wird alle Arbeit im Bezug auf das HPC zunächst eingestellt. Die Architektur ist bereits auf die Eigenheiten eines HPC angepasst, sodass die Einbindung zum Ende des Projekts ohne Schwierigkeiten möglich sein dürfte.
- Die für den Monat Mai gesteckten Zeitplan-Ziele wurden erreicht.

14.06.2016:

- Labels sollen in unterschiedlichen Gruppen einteilbar sein. So kann es z. B. eine Gruppe „Referenz-Labels“ und eine andere Gruppe „berechnete Labels“ geben, sodass man diese unterschiedlichen Label-Gruppen miteinander vergleichen kann.
- Aufgrund von Performance-Problemen am Plot soll nur der Ausschnitt der Datenserien berechnet werden, welche auch tatsächlich angezeigt werden. Beim Scrollen oder Zoomen soll eine Neuberechnung des Plots erfolgen.
- Lichtschranken-Daten und/oder Knopf-Druck-Daten während eines Assessments sollen

verwendet werden, um automatisch ein zu importierendes Daten-Segment auszuwählen oder Referenz-Labels automatisch zu erzeugen.

21.06.2016:

- Es wurde festgestellt, dass in der bisherigen Implementierung Interfaces für den MotionDataController (ursprünglich MainWidget), das PlotWidget und das TimelineWidget fehlten. Es wurde ein Entwurf der Interfaces angelegt, welche in der zukünftigen Implementierung genutzt werden sollen. Wegen der schlechten Performance von JavaFX beim Zeichnen der Plots und der Zeitleiste, wurde sich darauf geeinigt, dass die Plots und die Zeitleiste in der zukünftigen Implementierung direkt auf einer Canvas gezeichnet werden.

28.06.2016:

- Der Umzug unseres Git-Repositorys auf Bitbucket (auf einer Windows-VM) wurde veranlasst.
- Um die Reviews zu beschleunigen und besser über die Gruppe zu verteilen, bestimmt der Bearbeiter eines Tasks zukünftig einen (bei einfacheren Tasks) oder zwei bis drei Reviewer (bei schwierigeren Tasks).
- Aus einem Algorithmus resultierende Datenreihen und Label-Informationen werden im selben Ordner gespeichert, der bei Bedarf gelöscht werden kann. Die resultierenden Label-Informationen sollen nicht manuell überschrieben werden können.
- Lichtschranken-, Knopfdruck- und Bodenmessplatten-Daten liegen in sehr unterschiedlichen Formaten vor. Dies muss bei der Erzeugung von Referenz-Labels berücksichtigt werden.

05.07.2016:

- Für den Review-Prozess wird ab sofort Bitbucket genutzt, welches kurz von Jonas vorgestellt wurde.
- Aus der Retrospective ging hervor, dass etwas gegen die häufigen Verspätung zur Sitzung getan werden muss. Deshalb wurde sich darauf geeinigt, dass fünf Euro gezahlt werden, wenn man zu spät kommt. Das Geld wird gesammelt und für Pizza o.ä. beim Spieleabend ausgegeben.

19.07.2016:

- Nils wird nächste Woche nach Münster fahren, um Informationen über Hardware und Tool zu sammeln. Es wird darauf hingewiesen, dass andere auch teilnehmen können und weitere Fragen gesammelt werden sollen.
- Es erfolgte ein Redesign des Datenmodells und des Datenformates.
- Es wird festgelegt, das Reviews hohe Priorität bekommen. Wer mergt, soll den Task auch auf Done verschieben.

26.07.2016:

- Nils, Christin und Sebastian F. waren bei Humotion und konnten dort Einblicke in deren Projekte erhalten.
- Rückblickend haben wir unser Daten-/Dateimodell sehr komplex designt. Daher wird der Einsatz von Cassandra in Kombination mit Spark evaluiert.

01.08.2016:

- Ein Treffen mit Herrn Harfst vom Rechencluster fand statt, bei dem Jonas und Sandra anwesend waren. Dabei wurden Änderungen zum neuen Cluster besprochen.

04.08.2016:

- Der neue Sitzungstermin findet in der vorlesungsfreien Zeit Donnerstags von 10:00-13:30 Uhr statt. Dementsprechend verschiebt sich auch die Frist für das Fertigstellen der Aufgaben für den Review-Prozess von Freitag auf Sonntag.
- Durch die Programmiersprache von Spark und das umständliche Nutzen der Java-API, sowie durch die Einarbeitungszeit in Cassandra, werden diese beiden nicht genutzt, sondern unser eigenes Binärformat. Dadurch kann Zeit eingespart werden.
- Eine Bewertung der Gruppenmitglieder soll geschehen, da im Moment die investierten Zeiten weit auseinandergehen. Dazu sollen bis zum 09.08.2016 Bewertungskriterien an Sebastian geschickt werden. In der darauffolgenden Woche sollen dann die Bewertungen der Mitglieder untereinander geschehen.
- Der Zeitplan wurde um die Integration vom Cluster erweitert, dies soll im November geschehen.
- Weiterhin wird die Erkennung von Alltagsmustern verworfen.

11.08.2016:

- Sebastian F. möchte eine Übersicht über die bisherige Arbeit haben, daher soll in zwei Wochen, am 25.08.2016, eine kurze Präsentation stattfinden. Diese wird von Kjel, Thies, Dmitrii und Kardar vorbereitet.
- Es wird bis zur nächsten Woche eine gegenseitige, anonyme Bewertung der Gruppenmitglieder vorgenommen.

18.08.2016:

- Die GUI ist weitestgehend fertiggestellt. Diese soll nun auf Fehler bzw. mögliche Verbesserungen überprüft werden.

25.08.2016:

- Die Auswertung der Gruppenbewertung wurde an die PG-Teilnehmer weitergeleitet.
- Sebastian F. und Sandra wurde mit einer Präsentation der aktuelle Stand des Projekts dargestellt. Alle Diskussions- und Kritikpunkte wurden direkt in neuen Tasks im Backlog festgehalten.

01.09.2016:

- Es gibt Probleme (fehlerhafte Datensätze, falsch gedrückte Buttons) bei der automatischen Erzeugung von Referenzlabels. Mögliche diskutierte Lösungsansätze: Skript um herauszufinden wie viele Datensätze standardkonform sind oder vom Idealfall ausgehen und schauen für wie viele Datensätze automatisch Referenzlabel erzeugt werden können.
- Es fand die Vorstellung der Ergebnisse zur Literaturrecherche über die Segmentierung von Zeitreihen statt.

08.09.2016:

- Es wurden Themen im Bereich des maschinellen Lernens und der Gangerkennung für die Vorträge bestimmt, die das Projekt inhaltlich vorantreiben sollen:
  - Neural Networks (NN)
  - Support Vector Machines (SVM) bzw. Adaptive AMM: AMMs arbeiten im Gegensatz zu SVMs online, was für das Projekt vorteilhaft scheint, da die Gesamtheit der Daten nicht vorgeladen werden muss.
  - Gangerkennung: Dieser Vortrag soll als Einstieg ins Thema Gangerkennung dienen und einen umfassenden Überblick hierüber bieten.

- Es wurde eine Gruppierung von Assessment-Mustern zu Bewegungsmustern vorgenommen. Zudem wurde sich darauf geeinigt, dass der Fokus zu Beginn auf Gangerkennung und Erkennung von Ruhephasen gelegt werden soll. Ersteres weil dies für eine Vielzahl von Assessments von Bedeutung ist (SPPB: 4m-Gehen, Frailty: 4,57m-Gehen, TUG, DEMMI: 50m-Gehen, 6-Min-Gehtest), letzteres weil diese vermutlich relativ einfach erkennbar sind.

15.09.2016:

- Task-Reviews können zukünftig auch von einem anderen als dem vom Bearbeiter bestimmten Reviewer durchgeführt werden.
- Es wurde ein Assessment-Ablauf-Diagramm mit allen Bewegungsmustern erstellt.
- Code-Beispiele sollen in die Dokumentation mit dem Code-Makro und einem Sans-Serif-Font eingefügt werden.
- Wird ein Algorithmus ausgeführt mit verschiedenen MotionData-Objekten als Parametern, so findet man die resultierenden Datenreihen in den jeweiligen Verzeichnissen der entsprechenden MotionData-Objekte.
- Im Meilensteinplan wurde der Fertigstellungstermin für die Bewegungsmustererkennung nach hinten korrigiert und die HPC-Einbindung ergänzt.

22.09.2016:

- Für die Dokumentation von Ausarbeitungen wird die Vorlage der Projektdokumentation angepasst. Die Struktur und der Stil sollen gleich sein, weil diese Ausarbeitungen später in die Dokumentation eingefügt werden sollen.
- Die Präsentationen von Ausarbeitungen sollen so früh wie möglich gehalten werden, da dies wichtig ist für die weitere Sprint-Planung und für Folgeausarbeitungen.
- Alle MotionTypes aus Assessments sollen noch klassifiziert werden, um die Arbeit mit Algorithmen zu erleichtern. Es wurden vier verschiedene Bewegungstypen angelegt: ASSESSMENT, STATIC, TRANSITION und DYNAMIC

29.09.2016:

- Wir benötigen für das maschinelle Lernen noch deutlich mehr gelabelte Datensätze.
- Für Aktivitäten, die selten in den Assessments vorkommen (z.B. Springen), sollten wir noch Extra-Trainingsdaten aufnehmen (Christin fragt bei Altenpflegerin nach).

- Die ersten Vorträge sind in ca. zwei Wochen fertig.
- Fürs Reviewen gilt ab sofort: Der Prüfer soll innerhalb von zwei Tagen nach dem Erstellen der Pull-Request eine Rückmeldung per Bitbucket an den Prüfer senden, wann er dazu kommt das Review durchzuführen.

06.10.2016:

- Das Labeln weiterer Datensätze schreitet gut voran und auch die Ausarbeitungen werden voraussichtlich kommende Woche fertiggestellt. Daher werden am nächsten Donnerstag auch die entsprechenden Präsentationen gehalten.
- Es wurden sich genauere Gedanken über den kommenden Ablauf der Implementierung vom maschinellen Lernen gemacht. Dabei wurden Verfahren identifiziert, die für jeden Klassifikator notwendig sein werden, sowie auch Aufgaben, die in den einzelnen Gruppen bearbeitet werden müssen.

13.10.2016:

- Insgesamt wurden 28 weitere Datensätze innerhalb dieses Sprints gelabelt.
- 3/4 der Vorträge/Ausarbeitungen wurden fertiggestellt und präsentiert.
- Aus allen drei Vortragsthemen werden Gruppenarbeiten definiert, welche in den nächsten Sprints weiter ausgearbeitet und implementiert werden.
- Aus der Ausarbeitung von Sebastian W. werden weitere Ausarbeitungsthemen hervorgehen.

17.10.2016:

- Es fand die Besprechung der Gruppen NN, SVM/AMM und Decision Trees zum Trainieren des Modells und zur Klassifikation statt.
- Zur Nutzung von NN ist ein Vorverarbeitungsalgorithmus notwendig.
- Die Sliding Windows sollen jeweils eine Länge von zwei Sekunden haben und sich zu 66% überlappen um Voting zu ermöglichen.
- Aus den Ausarbeitungen ergibt sich, dass noch FeatureCalculators für TiltAngle, Standardabweichung, Spektralenergie und Signal Energy benötigt werden.
- Für NN soll die Library deeplearning4j, für SVM/AMM und Decision Trees die Libraries JSAT und Smile eingebunden werden.

- Es ist jeweils eine Klasse zum Trainieren des Modells und zur Klassifikation nötig.

24.10.2016:

- Sebastian W. hat seinen Vortrag zur Gangerkennung und Einführung in die Ganganalyse gehalten.
- Das Gruppentreffen findet ab jetzt beim Sprintende Montags ab 08:15, ansonsten um 09:15 statt, da die Zeit zwischen den Sprintenden meist nicht so lang für die Gruppentreffen ausfällt.
- Sandra hat uns Informationen zu Abweichungen vom Testablauf auf dem Netzlaufwerk zur Verfügung gestellt.

31.10.2016:

- Die einzelnen Gruppen melden Fortschritte bei der Implementierung der Algorithmen. Allerdings wird noch Zeit benötigt um entdeckte Probleme zu beseitigen.

07.11.2016:

- Es wurde ein allgemeines Konzept für die Machine-Learning-Algorithmen erstellt.
- Es wurden Themen für die verbleibenden Ausarbeitungen gefunden und diese vergeben.

14.11.2016:

- Sebastian W. hat die Anforderungen aktualisiert und einige offene Fragen bezüglich der Änderungen wurden in der Gruppensitzung beantwortet.

21.11.2016:

- Das Sliding-Window-Verfahren in seiner aktuellen Umsetzung führt dazu, dass kurze Bewegungen unterhalb der Fensterlänge ignoriert werden. Mögliche Lösungen sollen evaluiert werden.
- Die Seminararbeiten zum Parametertuning sollen begonnen werden.

28.11.2016:

- Tests zum Sliding-Window-Verfahren ergab, dass eine Fenstergröße von 150 und eine Schrittweite von 50 für die Sliding-Windows das Problem der Vorwoche löst.
- Es sollen mithilfe der erkannten Bewegungen nun auch die dazugehörigen Assessments erkannt werden.
- Ein Termin im Fitness-Studio zur Aufnahme neuer Daten war aufgrund der geringen

Teilnahme nicht sehr erfolgreich. Es gibt allerdings noch weitere Termine.

- Nils möchte ein Hardware-Board für einen Toshiba-Chip entwickeln, welcher aufgenommene Daten in Echtzeit zu einem PC übermittelt. Dadurch wäre die Erkennung von Bewegungsmustern in Echtzeit möglich. Eine Abstimmung darüber, ob dies durchgeführt werden soll, findet in der nächsten Sitzung statt.

05.12.2016:

- Um in den Assessments seltener vorkommende Bewegungsmuster sicher erkennen zu können, werden wir nochmals selbst Bewegungsdaten erzeugen und aufnehmen, da die ersten Aufnahmen bereits eine 4%-ige Verbesserung gebracht haben. Die Anwerbung von Probanden wird abgebrochen, da sie nicht so erfolgreich war und die Aufnahmen wohl erst zu spät zur Verfügung stehen würden.
- Die Assessment-Tests TUG, SPPB-5x-Chair-Rise, SCPT, CMJ und 6-Minutengehtest sollen vorrangig von unserer Anwendung erkannt und gelabelt werden.
- Nils wird ein Hardware-Board für einen Toshiba-Sensorchip bauen und die dadurch erzeugten Livestream-Daten sollen in unserer Anwendung visualisiert werden.

12.12.2016:

- Die noch ausstehenden Ausarbeitungen werden erst im neuen Jahr fertig.
- Da die von uns selbst aufgenommenen Zusatzdaten (mit in den Assessments seltener vorkommenden Bewegungsmustern) die Klassifikation verbessert haben, werden weitere Aufnahmen angefertigt.
- Im Code soll die Vereinheitlichung des Zeitkonzeptes (inklusiv/exklusiv) durchgeführt werden.

19.12.2016:

- Die restlichen Ausarbeitungen werden am 09.01. und 16.01.2017 präsentiert.
- Zum 01.01.2017 soll eine neue Gruppenbewertung stattfinden.

02.01.2017:

- Es werden drei Seminararbeiten zum Thema Parametertuning/Optimierung der ML-Algorithmen erarbeitet.
- Vier weitere Seminararbeiten beschäftigen sich mit: Uni-Cluster; Gangparameter;

Hidden-Markov-Models, Naive Bayes und Gaussian Mixture Models; Feature-Visualisierung

- Desweiteren wurden durch die Projektgruppe unterrepräsentiere Bewegungen gelabelt um eine verbesserte Ausgangssituation für die ML-Optimierung zu schaffen.

19.01.2017:

- Nach Gesprächen unter Nils, Daniel, Kjel und Andrea bezüglich der Ausarbeitungen zum Parametertuning und zusammen mit Sebastian F. hat sich herausgestellt, dass das bisherige Vorgehen in diese Richtung (manuelles Ausprobieren von Parametern) nicht zielführend ist und überdacht/überarbeitet werden muss. Dabei kam die Erkenntnis, dass ein automatisiertes Vorgehen dabei bisher nicht/sehr schwierig umzusetzen ist. Ein Umbau unserer Anwendung ist dazu notwendig. Weiterhin sollen die drei Ausarbeitungen zum Parametertuning trotzdem grob in diesem Bereich angesiedelt bleiben, jedoch mit veränderten Inhalten.
- Viele zusätzliche Bewegungsdaten wurden in dem Sprint gelabelt und ins Netzlaufwerk geladen, damit diese für das Parametertuning verwendet werden können.

23.01.2017:

- In der Gruppe wurde das neue Vorgehen für das Parametertuning diskutiert. Mit Hilfe von einem 1+1-EA soll die Optimierung der ML-Algorithmen vollautomatisiert ablaufen. Hierfür muss entsprechend neuer Code entwickelt werden und es sind auch einige Umbauarbeiten notwendig.
- Es sollen weitere geriatrische Datensätze gelabelt werden um eine breite Trainingbasis zu haben.
- Aus den Labeln von Datensätzen sollen künftig statistische Informationen generiert werden können, wie z. B. die Art der Bewegung, ihr prozentualer Anteil oder die Häufigkeit. Damit wird eine Basis für eine Art Fitnessreport geschaffen.

30.01.2017:

- Der Code für die Parameteroptimierung sollte bis zum 17.02. fertig sein, damit die Berechnungen während Daniels Urlaub ausgeführt werden können und er anschließend die Ergebnisse im Rahmen seiner Seminararbeit interpretieren kann.

06.02.2017:

- Es wurde abgeschätzt wie viele Ressourcen nötig sind für eine Ausführung der Parameteroptimierungs-

Algorithmen auf dem Rechencluster. Da nicht so viele Ressourcen benötigt werden, müssen keine Ressourcen auf dem Rechencluster extra gebucht werden.

13.02.2017:

- Die Umbau-Arbeiten für die Umsetzung der Optimierung der Klassifizierer ist abgeschlossen.
- Corinna hat ihren Vortrag zu „Gaussian Mixture Models, Hidden Markov Models und Naive Bayes“ gehalten.
- Es wurde festgehalten, dass Drehungen (nach Links, nach Rechts) bzw. Umdrehen (180° , nach Links/Rechts) als zu erkennende Bewegungen relevant sein können, um Assessments eindeutig zu identifizieren.

20.02.2017:

- Es wurde festgestellt, dass die gestarteten Job auf dem Rechencluster bezüglich der Parameteroptimierung sehr lange in der Warteschlange verweilen.
- Andrea hat ihren Vortrag zu Feature Sets gehalten.
- Die Vorträge von Nils und Daniel sollen am 20.03.2017 stattfinden.

27.02.2017:

- Da wir kein HPC-Gruppenverzeichnis erhalten, wurde ein Netzlaufwerk für uns angefordert (bis dahin gibt Jonas seinen Ordner für uns frei). Auf dem neuen Rechencluster sind noch Kapazitäten frei, daher werden dort nun unsere Jobs ausgeführt. Die Parameteroptimierung läuft für AMM und BDT gut (95 bis 99%) und ist für MLP und BDS noch zu verbessern.
- Drehen soll manuell gelabelt werden.

06.03.2017:

- Die bisherigen besten Klassifikationsergebnisse liegen für State mit BDT bei 99,88%, für Static mit BDT bei 98,36%, für Dynamic mit AMM bei 98,38% und für Transition mit BDT bei 100%.
- Um den Projektabschluss vorzubereiten, ist die Projektgruppe gemeinsam die Dokumentation durchgegangen. Ebenso soll die Anwendung von jedem Gruppenmitglied durchgetestet werden.

13.03.2017:

- Die Klassifikationsergebnisse haben sich im Vergleich zum letzten Durchlauf verschlechtert (ca. 96-97%).
- Es ist nun möglich über die GUI Algorithmen auf dem Cluster auszuführen.
- Es soll keine Klassifikation mit „Drehen“ gelabelten Daten mehr stattfinden, da zu wenige gelabelte Datensätze hiervon existieren.

20.03.2017:

- Die Applikation unterstützt nun neben der Sprache Deutsch auch Englisch. Weitere Arbeiten an dem Programm werden in der kommenden Woche fertiggestellt und es werden dann keine weiteren Features implementiert.
- Thies, Kjel, Sebastian und Nils haben sich bereiterklärt die Abschlusspräsentation auszuarbeiten und vorzutragen.
- Daniel hat seinen Vortrag zur Interpretation der Projektgruppen-Ergebnisse gehalten.

31.03.2017:

- Die Abschlusspräsentation wurde auf den 19.04.2017 (16-18 Uhr) gelegt.

02.04.2017:

- Des Ende des aktuellen Sprints wurde aufgrund eines verschobenen Sitzungstermins auf Donnerstag, den 06.04.2017 verschoben.

06.04.2017:

- In der Gruppe werden die letzten Dokumentations-Tasks für den Fein-Schliff zur Bearbeitung freigegeben. In der Sitzung wurde über die Inhalte der geplanten Abschlusspräsentation diskutiert.

19.04.2017:

- Die Abschlusspräsentation wurde gehalten. Das Projekt ist damit erfolgreich abgeschlossen.

D. Seminararbeiten

D.1. Boosted Decision Trees, K-Means und Co-Training

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Boosted Decision Trees, K-Means und Co-Training

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Kardar Kurdo

Oldenburg, den 7. April 2017

Inhalt

Abbildungen

1 Einleitung

Nach der zivilen Nutzung des Internets in Form des „World Wide Webs“ seit 1991 ist das digitale Datenvolumen sehr stark angestiegen. Waren im Jahr 2005 ein geschätztes weltweites Datenvolumen von 130 Exabyte vorhanden, so lag der Wert zehn Jahre später, im Jahr 2015, bereits bei 8.591 Exabyte. Im Jahr 2020 wird es nach einer Prognose schon 40.026 Exabyte an Datenvolumen geben [Sta16](Siehe Abbildung 1).

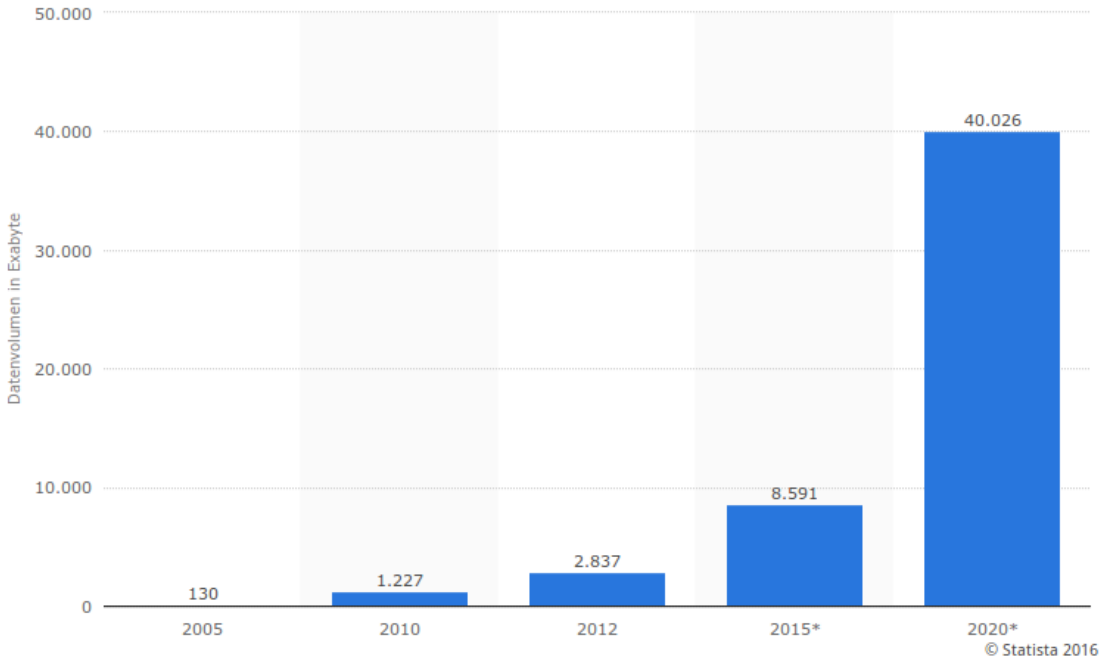


Abbildung 1: Prognose des weltweiten jährlich generierten Datenvolumens (Angaben in Exabyte)
(Quelle: [Sta16])

Durch die große Anzahl an Datenvolumen hat sich ein immer stärker werdendes ökonomisches Interesse entwickelt, die Daten effizient über intelligent entwickelte Software schnell zu verarbeiten und auch neue Informationen daraus zu gewinnen. Dabei hat sich Machine Learning (ML) zu einem festen Bestandteil von künstlicher Intelligenz in der Informatik entwickelt. Von Interesse ist die Anwendung von Machine Learning im Bereich medizinischer Informatik. Den damit wird ermöglicht, dass große Mengen an gesammelte

Daten über Patienten effektiv und effizient verarbeitet werden, um Gesundheitsverläufe zu beobachten und Erkrankungen frühzeitig vorbeugen zu können.
Im Rahmen der Projektarbeit in den Masterstudiengängen Informatik, Wirtschaftsinformatik und ESMR ist die Zusammenarbeit in einer Gruppe über zwei Semestern Bestandteil der jeweiligen Prüfungsordnung. Der Verfasser dieser Ausarbeitung wirkt in der Projektgruppe: „Mobilitäts-Assessments mit körpernahen Sensoren“ im SoSe 2016 - WiSe 2017 mit. Als Bestandteil der Gruppenteil sind Vorträge zu halten. Mit dieser Ausarbeitung kommt der Autor dieser Reglung nach und verfasst aus dem Themenbereich Machine Learning einen Vortrag über die drei Verfahren Boosted Decision Trees, K-Means und Co-Training. Diese Ausarbeitung ist folgend gegliedert. Zunächst wird kurz der Oberbegriff Machine Learning definiert, unterteilt, Anwendungsfelder genannt und eine Unterscheidung von den darunterfallenden Algorithmen gegeben. Der darauf folgende Punkt bildet Supervised Learning. Nachdem auch hier eine kurze Definition und mögliche Anwendungsfelder gegeben werden, wird zunächst das Verfahren Decision Trees beschrieben und Gründe für das Boosting geliefert. Danach wird das Verfahren Boosted Decision Trees behandelt.Der nächste Punkt bildet Unsupervised Learning. Nach einer kurzen Definition und das Aufzeigen von Anwendungsfeldern wird das Verfahren K-Means beschrieben. Im letzten Punkt wird das Semi-Supervised Learning thematisiert. Nach einer kurzen Definition und das Aufzeigen von Anwendungsfeldern wird auf das Verfahren Co-Training eingegangen.

2 Machine Learning (ML)

In den folgenden Punkten wird auf das Themengebiet Machine Learning eingegangen.

2.1 Definition

Maschinelles Lernen ist ein Teilbereich des Forschungsgebietes von Künstlicher Intelligenz und beschreibt die Fähigkeit eines Programms auf einem Computersystem aus Erfahrungen zu lernen. Sie bildet eine effektive Strategie, um mithilfe der statistischen Mathematik „Daten zu analysieren und auszuwerten und die gewonnenen Erkenntnisse zu verwenden, um Informationen über weitere Daten zu erhalten“ ([Nit05], S. 4).

2.2 Anwendungsfelder

Maschinelles Lernen wird in Gebieten mit automatisierten Abläufen ohne Nutzerinteraktion, Situationen mit nicht ausreichendem Expertenwissen, wie zum Beispiel bei der DNA-Forschung, Situationen mit dynamischen Anforderungen, die zum Beispiel in der Wirtschaft sehr häufig auftreten (Data Mining), sowie personalisierte Programmen eingesetzt ([Nit05], S. 4).

2.3 Unterteilung von ML-Algorithmen

Die Algorithmen für das Maschinelle Lernen lassen sich in sechs Bereiche einteilen: Supervised Learning, Unsupervised Learning, Semi-Supervised Learning, Reinforcement Learning, Transduction und Learning to Learn.
Supervised Learning bedeutet, von vorgegebenen Zuordnungen zu lernen([Gra07], Folie

66). Dabei wird eine Klassifikationsstruktur durch das Trainieren mit Beispieldaten erzeugt, die später eingesetzt wird ([Nit05], S. 5). Beim Unsupervised Learning findet eine Zuordnung eines Modells zu Datenpunkten statt ([Gra07], Folie 66). Dabei kommen keine Trainingsdaten zum Einsatz. Das zu erlernende Wissen über die Aufteilung von Daten wird durch die Analyse der eigentlichen Daten gewonnen ([Nit05], S. 5). Semi-Supervised Learning bildet eine Mischung aus Supervised und Unsupervised Learning. Beim Reinforcement Learning wird eine Taktik zum Handeln in einem speziellen Kontext erlernt, die durch Feedbacks aus der Umgebung geformt wird ([Nit05], S. 6). Das Lernen mittels Transduction ähnelt der beim Supervised Learning. Es werden zwar Trainingsdaten zum Lernen genommen, aber mit dem Unterschied, dass Vorhersagen auf Eingabedaten und deren Ergebnissen bei den Trainingsdaten basieren ([Nit05], S. 6). Beim Learning to Learn findet das Erlernen durch „eine Adaption des Lernalgorithmus selbst auf Basis der früheren Erfahrung“ ([Nit05], S. 6) statt.

3 Supervised Learning

Das Ziel beim Supervised Learning (Überwachtes Lernen) liegt darin die Zukunft vorherzusagen [Klo15]. Dabei ist eine Menge von Datenpunkten, sowohl Ein- als auch Ausgabewerten, bereits gegeben. Anhand von bereits vorliegenden Eingabewerten und den zugehörigen Ausgabewerten zielt das angestrebte Ergebnis auf das „Lernen einer Funktion, welche Zuordnungen automatisch auf neuen, noch nicht gesehenen Datenpunkten trifft“ ([Gra07], Folie 68). Zu den typischen Werkzeugen von überwachtem Lernen gehören die Klassifizierung, bei der Daten in Kategorien sortiert werden und die Regressionsanalyse, wo eine Vorhersage aufgrund von Eingabewerten bestimmt wird [Klo15]. Zu den bekannten Algorithmen gehören unter anderem Decision Trees, Boosted Decision Trees, Nearest Neighbor Algorithmus und Support Vector Machines. Im Folgenden werden die Anwendungsfelder genannt und anschließend wird schrittweise auf das Verfahren von Boosted Decision Trees (BDT) eingegangen.

3.1 Anwendungsfelder

Zu den typischen Anwendungsgebieten von Supervised Learning gehören: Textklassifikation, Kontexterkennung, Ranking von Suchergebnissen, Gen-Daten-Analyse, Bildanalyse, Spracherkennung, Robotik und Quantenphysik ([Gra07], Folie 70).

3.2 Decision Trees

Ein Verfahren aus der Kategorie überwachtes Lernen bildet Boosted Decision Trees (BDT). Da das BDT auf dem Verfahren Decision Tree basiert, wird dies zunächst beschrieben.

3.2.1 Definition

Ein Decision Tree (Entscheidungsbaum) ist ein Modell zur multivariaten Analyse von Datensätzen ([Sch13] S. 23). Dabei werden die Datensätze stufenweise durch binäre Entscheidungen, die häufig mit Ja oder Nein zu beantworten sind, in immer kleinere Teilmengen aufgeteilt ([Sch13], S. 23). Die dabei entwickelte Struktur ähnelt am Ende einer Baumstruktur.

3.2.2 Algorithmus-Beschreibung

Als Basis-Elemente dienen innere Knoten, welche mit einem Attribut beschriftet sind und die Kanten der Knoten (Vater-Knoten), die Bedingungen enthalten. Der Wertebereich von den Bedingungen der Kanten eines Vater-Knotens überschneiden sich dabei nicht ([Gra16b], Folie 30) (Siehe Abbildung 2)

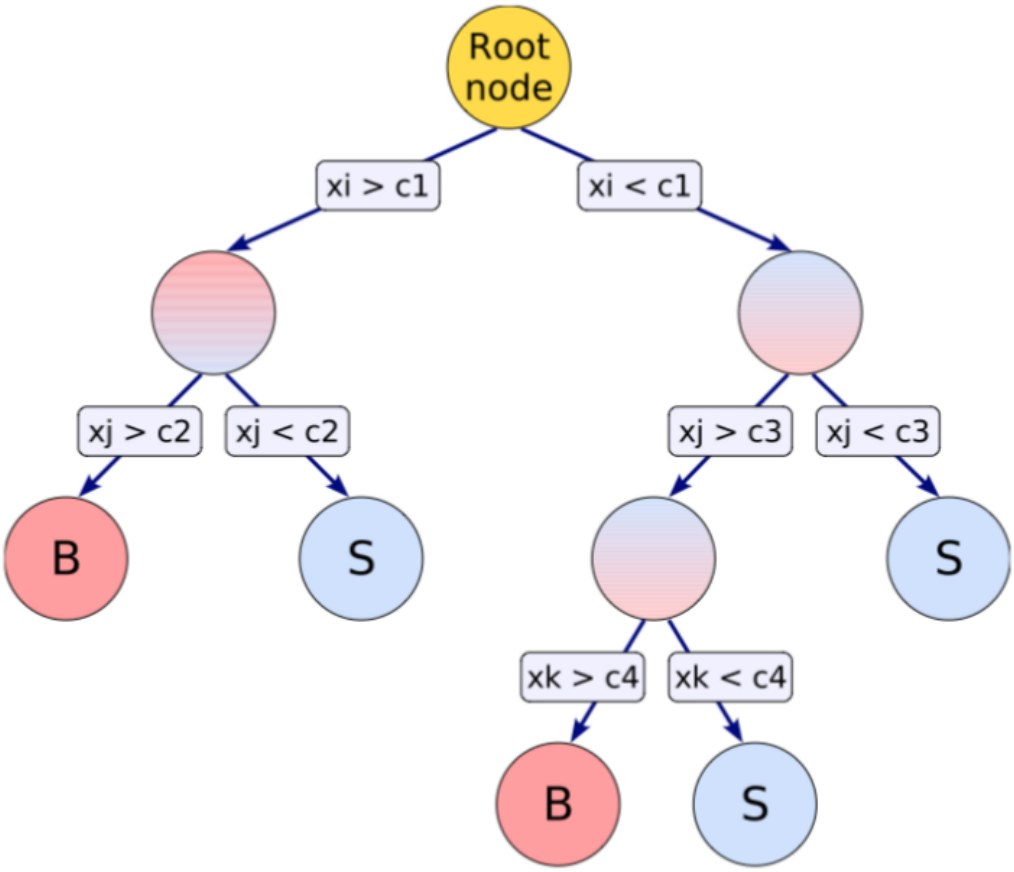


Abbildung 2: Aufbau eines Entscheidungsbaumes
(Quelle: [Gie12], S. 11)

Der Algorithmus zum Aufbau einer Klassifikationsstruktur mit Hilfe von Trainingsdaten

ist folgend ([Gra16b], Folie 34):

1. Wähle einen Attribut und Bedingungen dafür (Splits)
2. Partitioniere die Trainingsmenge gemäß Bedingungen in m Partitionen. Bei binären Entscheidungsbäumen beträgt m = 2. Als Ergebnis erhält man einen inneren Knoten.
3. Das Verfahren wird rekursiv für jede der Partitionen angewandt, solange „nicht alle Objekte einer Partition der gleichen Klasse angehören oder ein anderes Stopp-Kriterium erfüllt ist“ ([Gra16b], Folie 34).

Als Stopp-Kriterium gilt zum Beispiel, wenn „die Teilmenge in einem Knotenpunkt eine bestimmte Anzahl von Ereignissen, welche als Parameter mitgegeben wird, unterschreitet“ ([Sch13], S. 23). Die Werte eines Endknotens, der nicht mehr aufteilbar ist, werden dabei einer Klasse zugewiesen. Wichtig dabei ist, dass zunächst ein Auswahlkriterium (Variable) gesucht wird, das die beste Trennung des Ausgangsdatensatzes liefert. „Ist diese gefunden, wird der Schnittwert festgelegt, bei dem diese Trennung am besten funktioniert“ ([Sch13], S. 23). Zu den klassischen Algorithmen gehören ID3, CART, C4.5 und C5.0, bei denen das Ziel darin besteht, die Anzahl der erwarteten Vergleiche zu minimieren, also den Baum zu balancieren und niedrig zu halten ([Gra16b], Folie 41). Dazu wird der Split so gewählt, dass die Entropie der neuen Partition maximal kleiner wird ([Gra16b], Folie 41). Dabei ist die Entropie einer Trainingsmenge T bezüglich der Klassen $C = c1, \ldots, cn$ definiert als ([Gra16b], Folie 41):

$$entropy(T) = -\sum_{i=1}^{n} p_i * log(p_i) \tag{3.1}$$

pi bildet die relative Häufigkeit der Klasse ci in der Trainingsmenge. Während die Entropie durch die Partitionierung maximal verringert wird, soll der Information-Gain maximal sein. Definiert ist der Information-Gain einer Partition P einer Grundmenge T mit $|P| = m$ als ([Gra16b], Folie 46):

$$gain(T, P) = entropy(T) - \sum_{i}^{m} entropy(P_i) \tag{3.2}$$

Der gewichtete Information-Gain lautet dabei ([Gra16b], Folie 46):

$$wgain(T, P) = entropy(T) - \sum_{i}^{m} \frac{|P_i|}{|T|} entropy(P_i) \tag{3.3}$$

Für die Berechnung des besten Splits gelten dabei folgende Schritte ([Gra16b], Folie 50):

1. Berechne den gewichteten Information-Gain für alle Attribute mit den jeweiligen Partitionierungen
2. Wähle den Split, bei dem der gewichtete Information-Gain am größten ist
3. Sollte kein Split möglich sein, dann muss es für diese Partition gestoppt werden
4. Partitioniere die Trainingsmenge entlang der Attribute in den entsprechenden Partitionen
5. Das Verfahren muss für jede noch übrig gebliebene Partition rekursiv angewandt werden

3.2.3 Richtige Größe der Parameter

Dabei ist es wichtig die richtige Größe der Parameter für den Datensatz, der für die Mindestanzahl an Objekten bzw. Ereignissen in letzten Knoten verantwortlich ist, zu wählen. „Ist der Parameter zu klein gewählt, wird der Tree zu komplex werden und zu Overtraining neigen“ ([Sch13], S. 23). Andererseits kann der Entscheidungsbaum wichtige Eigenschaften des Datensatzes nicht mehr auflösen ([Sch13], S. 23).

3.2.4 Vorteile

Zu den Vorteilen von Entscheidungsbäumen gehören (vgl. [Aut07], Folie 10), dass sie einfach zu verstehen und zu interpretieren sowie schnell zu implementieren sind. Entscheidungsbäume sind außerdem seit langer Zeit bekannt und bewährt. An jeden Knoten ist die Entscheidung immer nachvollziehbar. Zudem wird kein Ereignis vollständig entfernt und keine Information geht verloren. Außerdem existieren dafür viele Anwendungen und sie haben eine hohe Genauigkeit ([SLP+12], Folie 69).

3.2.5 Nachteile

Zu den Nachteilen von Entscheidungsbäumen zählen, dass sie nicht robust gegenüber Rauschen (fehlerhaft Daten) ([SLP+12], Folie 70) und dass sie nicht stabil sind. Denn eine kleine Änderung bzw. Schwankung der Daten kann einen großen Unterschied im Ergebnis machen ([Aut07], Folie 10). Außerdem tendieren sie zum Overfitting (Überanpassung). Fehler durch Overfitting können verursacht werden durch Fehlerhafte (verrauschte) Trai-

ningsdaten, Anpassung an sehr speziellen (korrekten) Beispielen und starke Spezialisierung durch zu langes Lernen ([Bur09], Folie 21).

3.2.6 Gründe für Boosting

Die Gründe für das Verwenden von Boosting sind zu einem, dass BDT leistungsstärker als die Decision Trees sind und dass „statistische Fluktuationen ausgeglichen und damit im Endergebnis weitgehend vermieden werden“ ([Gie12], S.12). Denn ein „einzelner Entscheidungsbaum[es] hat den Nachteil stark unterschiedliche Ergebnisse schon für geringe Veränderungen im Trainingsdatensatz zu erzielen“ ([Gie12], S.11).

3.3 Boosted Decision Trees

Im Folgenden wird das Verfahren Boosted Decision Trees definiert und der Algorithmus beschrieben. Am Ende folgen die Vor- und Nachteile.

3.3.1 Definition

Boosted Decision Trees (gewichtete Entscheidungsbäume) ist ein Verfahren aus der multivariaten Analyse, welche aus einer Sammlung von Decision Trees (Entscheidungsbäumen) besteht ([Sch13], S. 23). Die Zuweisung einer Eingabe zu einer Klasse findet in einer Mehrheitsentscheidung mehrerer Entscheidungsbäumen statt ([Lan07a], S. 51). Beim BDT werden Entscheidungsbäume mit demselben Datensatz trainiert und unterschiedlich gewichtet ([Sta13], S. 21). Wird ein Ereignis falsch klassifiziert, so werden diese für das Training in dem nächsten Entscheidungsbaum höher gewichtet ([Lan07a], S. 51f.).

3.3.2 Ziel

Das Ziel beim BDT ist es durch das ständige Lernen vom Training eine optimale Verzweigungsstruktur zu finden, um Objekte oder Ereignisse am besten zu sortieren. In der Praxis ist das Verwenden von bis zu 200 Entscheidungsbäumen üblich ([Sch13], S. 25).

3.3.3 Algorithmus-Beschreibung

Für Boosted Decision Trees existieren verschiedene Verfahren. In diesem Punkt wird das Adaptive Boosting beschrieben. Das Adaptive Boosting (AdaBoost) ist eine sehr bekannte BDT-Methode, „dass die Klassifizierungsfunktion boostet, indem es einfache (schwache) Klassifizierer zu einem starken Klassifizierer kombiniert“ ([Fau07], S. 6). Der Algorithmus ist wie folgt aufgebaut: Am Anfang erhält jedes Ergebnis das gleiche Gewicht. Bevor ein weiterer Entscheidungsbaum aufgebaut wird, wird die „misclassification rate“ des vorherigen Baumes bestimmt, das als ein Maß für die Anzahl falsch klassifizierter Ereignisse gilt ([Sch13], S. 25).

$$err_k = \frac{\sum_{i=1}^{N} w_i I(x_i)}{\sum_{i=1}^{N} w_i} \tag{3.4}$$

Die „misclassification rate“ beträgt per Definition $\leq 0,5$. Ist der Fehler größer als 0,5, so wird das Training frühzeitig abgebrochen. „Hierbei ist w_i die Gewichtung des i-ten Ereignisses, N die Gesamtzahl der Ereignisse. Die Funktion $I(x_i)$ ist dann 1, wenn Ereignis x_i falsch eingestuft wurde und 0 sonst“ ([Sch13], S. 25). Als nächstes wird der Gewichtungsfaktor α_k bestimmt, wobei β als Parameter wählbar ist.

$$\alpha_k = \left(\frac{1 - err_k}{err_k}\right)^{\beta} \tag{3.5}$$

„Im k+1-ten Tree werden die Gewichte aller falsch klassifizierten Ereignisse mit α_k multipliziert“ ([Sch13], S. 25). Die Gewichtung erhöht sich, wenn folgendes gilt: $\alpha_k \geq 1$. Nachdem der Wald aus Entscheidungsbäumen erstellt worden ist, wird über die Klassifikation $T(x_i)$ berechnet, „wie es im Mittel über alle Trees einsortiert wird, unter Berücksichtigung des Boosting Faktors“ ([Sch13], S. 25).

$$T(x_i) = \sum_{k=1}^{N_{Tree}} \alpha_k T_k(x_i) \tag{3.6}$$

Dabei bildet $T_k(x_i)$ die Ausgabe eines einzelnen Entscheidungsbaumes. Die Ergebnisse umfassen einen Werteintervall zwischen +1 und -1. Ein Wert von +1 oder Werte größer als Null sind der ersten Klasse zuzuordnen. Werte von -1 oder Werte kleiner als Null sind in der zweiten Klasse einzustufen ([Gie12], S.12).

3.3.4 Vorteile

Als Vorteil ist die schnelle, einfache und leichte Implementierung sowie die Tatsache, dass außer der Anzahl der Trainingsrunden keine Parameter zu adaptieren sind zu nennen ([Bis02], S. 6). Außerdem kann jeder beliebige einfache Klassifikator als Basis genutzt werden ([Bis02], S. 6). Zudem sorgt sie dafür, dass eine Variable, die eine sehr schlechte Trennung bietet, nicht dazu führen kann, dass die Methode schlechter arbeitet ([Sch13], S. 25). Zu beachten ist, dass die tatsächliche Performanz stark von der Menge der Trainingsdaten und der Art des einfachen oder schwachen Basisklassifikators abhängt, der weder zu schwach noch zu komplex sein darf ([Bis02], S. 6).

3.3.5 Nachteile

Als Nachteil gilt, dass nicht die gesamte Separationsfähigkeit aller Variablen berücksichtigt wird ([Sch13], S. 25). Außerdem ist eine Anfälligkeit für das Rauschen vorhanden. Allerdings kann eine starke Gewichtung von Ausreißern auch zu dessen Identifikation genutzt werden ([Bis02], S. 6).

4 Unsupervised Learning

Beim Unsupervised Learning (Unüberwachtes Lernen) ist das Ziel vorhandene Daten zu verstehen [Klo15] und versteckte Strukturen in den Daten zu entdecken [Klo15]. Im Gegensatz zu Supervised Learning existieren hierfür keine Trainingsdaten mit Ein- und Ausgabewerten. Dabei handelt sich bei den Arbeitsdaten also um eine Menge von Datenpunkten ohne Zuordnung. Eine Annäherung an das Ziel wird durch das Anwenden von mathematischen Modellen erreicht, unter anderem durch das Clustering, Reinforcement-Learning, die Dimensionalitätsreduktion oder eine Wahrscheinlichkeitsfunktion ([Gra07], Folie 84). Zu den bekannten Algorithmen gehören unter anderem K-Means, Mixture Models, Hierarchical Clustering und der Expection-Maximization Algorithmus. Im Folgenden werden mögliche Anwendungsmöglichkeiten genannt und anschließend auf K-Means eingegangen.

4.1 Anwendungsfelder

Ein mögliches Anwendungsfeld ist die Segmentierung mit Clustering, bei der Daten mit Gemeinsamkeiten gruppiert werden [Klo15]. Mit der Clustering-Methode ergeben sich im Text Mining folgende mögliche Anwendungsfelder: Automatische Extraktion von Themen einer Suche, Finden von Plagiaten bzw. Mutationen von Texten, Extraktion von Konzepten in einem Informationsraum sowie die Zusammenfassung von Texten ([Gra07], Folie 93). Auch gehört die Komprimierung von Daten mithilfe der Hauptkomponentenanalyse zu einem Anwendungsfall [Klo15].

4.2 K-Means

In den folgenden Abschnitten werden das K-Means-Verfahren definiert, Ziele genannt und der Algorithmus beschrieben. Anschließend werden die Vor- und Nachteile aufgezählt.

4.2.1 Definition

Bei K-Means handelt es sich um einen iterativen Clustering-Algorithmus, bei dem im Voraus die Anzahl der Cluster bekannt sein muss ([Nit05], S. 12).

4.2.2 Ziel

Das Ziel ist ähnliche Objekte durch eine Abstandsfunktion (in der Regel Euklidisches Abstandsmaß) zu den Clusterzentren zu Clustern zusammenzufassen.

4.3 Algorithmus-Beschreibung

Der Algorithmus ist so auszuführen, dass zunächst k Datenpunkte als Cluster-Zentren ausgewählt werden. Als Nächstes werden alle Datenpunkte durch das Abstandsmaß zum nächstgelegenen Cluster-Zentrum zugewiesen. Anschließend werden die Cluster-Zentren durch den Mittelwert aller zugewiesenen Datenpunkte ersetzt. Die Zuweisungen der Datenpunkte zu den nächstgelegenen Cluster-Zentren sowie das Ersetzten der Cluster-Zentren durch den Mittelwert aller zugewiesenen Datenpunkte ist solange zu wiederholen, bis sich die Werte der Cluster-Zentren sich nicht mehr ändern ([Gie02], Folie 9). Als Beispiel zur Veranschaulichung dient die Abbildung 3.

4.4 Varianten

Eine mögliche Variante ist, dass die initialen Cluster-Zentren gleichmäßig im Raum verteilt werden. Dies wäre von Vorteil bei relativ gleichmäßig verteilten Datenpunkten, allerdings schlecht für stark geclusterte Daten, da der Verschiebungsabstand höher wäre ([Gra16a], Folie 29). Eine andere Variante ist es, das Verfahren zu stoppen, sobald nur noch wenige Objekte ihre Zugehörigkeit zu Cluster-Zentren ändern. Das würde das Verfahren schneller

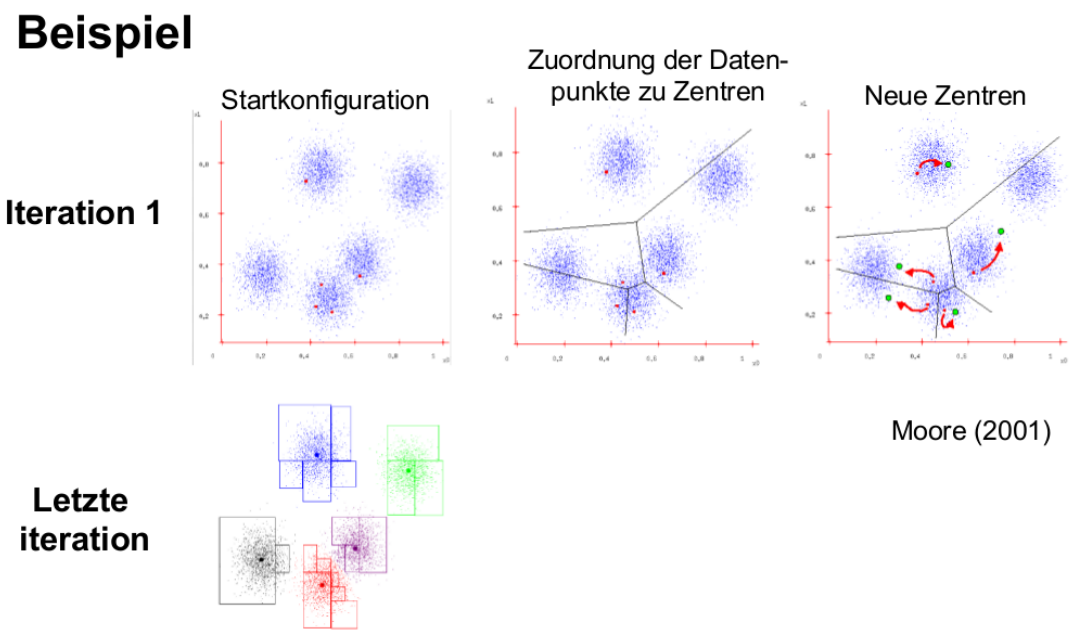


Abbildung 3: Beispiel beim Clustering mit K-Means
(Quelle: [Gie02], Folie 12)

machen, allerdings wären leichte Ungenauigkeiten vorhanden ([Gra16a], Folie 29). Zum Schluss ist es möglich K-Means mehrmals mit unterschiedlichen Startpunkten zu starten und dabei das beste Ergebnis nehmen, was eine Standard-Methode darstellt, um schlechten initialen Cluster-Zentren entgegenzuwirken ([Gra16a], Folie 29).

4.5 Vorteile

Zu den Vorteilen von K-Means gehört, dass es durch die schnelle Laufzeit effizient ist und einfach zu erlernen bzw. schnell zu implementieren ist. Außerdem kommt es mit vielen unterschiedlichen Datentypen zurecht [Clu09]. Zudem wird das Problem von der Partitionierung von multidimensionalen Daten in mehreren Clustern durch „die Suche von lokalen Minima in der Fehlerfunktion“ ([Nit05], S. 13) gelöst. Eine Suche nach dem globalen Minimum würde Probleme mit Ausprägung der Komplexitätsklasse NP-Hart verursachen ([Nit05], S. 13).

4.6 Nachteile

Zu den Nachteilen von K-Means gehört, dass es Probleme hat, Cluster zu erkennen, die keine kugelförmige Struktur haben oder „die große Abweichungen in Dichte und Größe aufweisen“ [Clu09] (Siehe Abbildung 4).

Abbildung 4: Clustering-Probleme bei K-Means
(Quelle: [Gra16a], Folie 34)

Außerdem haben Ausreißer großen Einfluss auf die Ergebnisse [Clu09]. Des Weiteren kann das Clusterergebnis von der Wahl der Startpunkte abhängen [Hü06]. Daher wird geraten mehrere Versuche mit unterschiedlichen Startpositionen zu unternehmen [Loh12]. Auch kann es „zu leeren Klassen kommen, bei denen bestimmte[n] Clusterzentren keine Daten zugeordnet werden“ [Loh12]. Zu erwähnen ist auch, dass die Resultate von der Anzahl der Cluster abhängig sind, wobei bislang keine allgemeine Regel für die Festlegung einer optimalen Anzahl an Cluster in Abhängigkeit zu der Menge von Datenobjekten existiert ([Nit05], S. 13).

5 Semi-Supervised Learning

Wie bereits beim Punkt Unterteilung der Machine-Learning-Algorithmus erwähnt, ist das Semi-Supervised Learning (halb-Überwachtes Lernen) eine Kombination aus Supervised Learning und Unsupervised Learning. Dabei findet das Lernen mit großen Datenmengen statt, bei dem nur für einen kleinen Teil der Eingabedaten Zuordnungen vorhanden sind, jedoch ein großer Teil der Eingabedaten über keine Zuordnungen verfügt. Aus den Daten ohne bekannte Zuordnung „können Informationen über die Nachbarschaft oder Cluster der Daten gewonnen werden, was dabei helfen kann die Trainingspunkte mit bekanntem Wert besser einzuordnen“ ([Sch14], S. 70). Zu den bekannten Algorithmen gehören unter anderem das Co-Training, Self-Training, Mixture of Gaussian distributions und Semi-supervised Support Vector Machines. Im Folgenden werden Anwendungsfelder genannt und im Anschluss auf das Verfahren Co-Training eingegangen.

5.1 Anwendungsfelder

Das teil-überwachte Lernen kommt gerne in Situationen zum Einsatz, bei denen vollständig gelabelte Daten rar, zeit- oder kostenintensiv und ungelabelte Datensätze kostengünstig sowie schnell zu haben sind. Das Lernen der Zuteilung durch die Verwendung von gelabelten und ungelabelten Daten bildet dabei einen ökonomischen Anreiz. Ein maßgebendes Anwendungsfeld von Semi-Supervised Learning ist das Web Data Mining. Dabei wird es unter anderem zur Bild-Kategorisierung, Website-Klassifizierung und Spracherkennung angewendet ([Zhu07], Folie 36). Außerdem findet es auch Einsatz im „natural language processing (word sense disambiguation, document categorization, named entity classification, sentiment analysis, machine translation), computer vision (object recognition, image segmentation), bioinformatics (protein function prediction), and cognitive psychology“ ([Zhu09], S. 8).

5.2 Co-Training

Im Folgenden wird das Verfahren Co-Training vorgestellt. Dazu wird das Verfahren definiert und die Ziele aufgezeigt. Anschließend folgen die Algorithmus-Beschreibung sowie die Vor- und Nachteile.

5.3 Definition

Das Co-Training ist eine Methode, die im Bereich Semi-überwachtes Lernen angewendet wird, um einen guten Klassifikator mit Hilfe von ungelabelten Daten zu erhalten. Vorausgesetzt wird, dass es möglich ist, die Features der Daten in zwei disjunkte Mengen aufzuteilen ([Lan07b], S. 1). Dabei müssen die beiden Teilmengen redundant ausreichende Features bilden, das heißt, dass bereits eines der Features ausreichen muss, um die Daten eindeutig klassifizieren zu können ([Lan07b], S. 3). Die beiden Teilmengen geben dabei bedingt unabhängig die Klasse wieder ([Zhu07], Folie 106). Es reicht hierbei aus, dass nur eine Teilmenge die Klassifikationsfunktion lernt ([Lan07b], S. 1). „Mit diesen beiden Teilmengen werden dann zwei Klassifikatoren trainiert, die sich gegenseitig beim Klassifizieren und Lernen der ungelabelten Daten unterstützen“ ([Lan07b], S. 1).

5.4 Ziel

Das Ziel von Co-Training ist es die große Menge an ungelabelten Daten zu verwenden, „um das Ergebnis eines Klassifikators, der zuvor mit einer kleinen Menge gelabelter Daten trainiert wurde, signifikant zu verbessern“ ([Lan07b], S. 1).

5.5 Algorithmus-Beschreibung

Der Algorithmus für diese Methode beginnt damit, dass zunächst zwei schwache Klassifikatoren mit den wenig vorhandenen gelabelten Daten trainiert werden. „Ein schwacher Klassifikator ist ein Klassifikator, der nur geringfügig besser sein muss als eine zufällige Wahl der Klasse“ ([Lan07b], S. 3). Dabei trainiert jeder schwache Klassifikator mit jeweils eines der beiden vorher aufgeteilten Teilmengen von gelabelten Datenpunkten

([Lan07b], S. 3). Anschließend werden mit den beiden Klassifikatoren separat ein Teil der nicht zugeordneten Datenpunkte klassifiziert. „Aus diesen Datenpunkten werden zu jedem Label [...] die [...] Datenpunkte ausgewählt, denen dieses Label mit der größten Konfidenz zugewiesen wurde“ ([Lan07b], S. 3). Ist die erledigt, so werden die Ergebnisse der meist zuversichtlichen Zuweisung der einen Funktion zu den gelabelten Datenpunkten der zweiten Funktion eingefügt und umgekehrt. Ein Datenpunkt muss dabei von beiden Klassifikationsfunktionen das gleiche Label erhalten. Der Algorithmus wiederholt sich bis alle ungelabelten Datenpunkte mit zusammenhängenden Hintergrundinformationen gelabelt werden. Zur Veranschaulichung soll ein bipartiter Graph dienen (Siehe Abbildung 5).

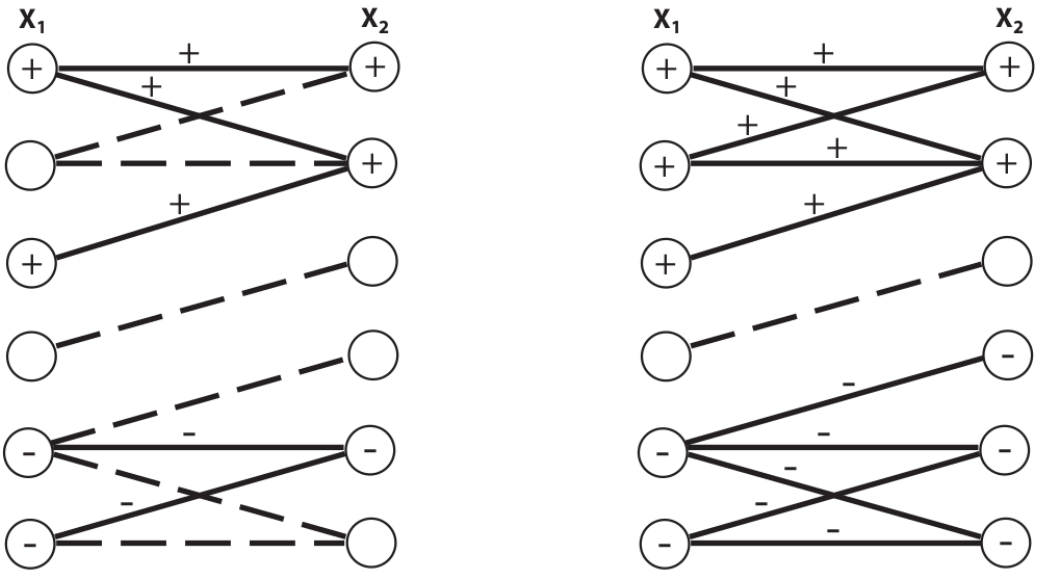


Abbildung 5: Darstellung des Algorithmus des Co-Trainings durch einen bipartiten Graphen
(Quelle: [Lan07b], S. 5)

Die Abbildung 5 stellt zwei Iterationen dar. Auf der linken Seite ist die erste Iteration zu sehen. Dabei repräsentieren die Knoten jeweils einen Datenpunkt für die erste beziehungsweise zweite Teilmenge. Die Knoten sind mit gestrichelten oder durchgezogenen Kanten verbunden, die angeben, dass die Wahrscheinlichkeit des Datenpunktes für die Verteilung ungleich Null ist ([Lan07b], S. 4). Durchgezogene Kanten verbinden gelabelte Datenpunkte. Gestrichelte Kanten verbinden ungelabelte Datenpunkte. Die Plus- und Minus-Zeichen symbolisieren lediglich zwei Labels als Beispiele. Zwei Knoten eines zusammenhängenden Teilgraphen, die durch eine Kante miteinander verbunden sind, müssen von beiden Funktionen das gleiche Label zugewiesen werden ([Lan07b], S. 4). In der ersten Iteration

werden die Klassifikatoren mit bereits gelabelten Datenpunkten trainiert (durchgezogene Kanten). In der zweiten Iteration auf der rechten Seite werden die ungelabelten Datenpunkte, die mit bereits gelabelten Datenpunkten verbunden sind, gelabelt. „Diese werden dann im Falle der Graph-Notation zu durchgezogenen Linien und entsprechend mit dem zugewiesenen Label versehen. Danach wiederholt sich der Vorgang“ ([Lan07b], S. 4).

5.6 Vorteile

Zu den Vorteilen von dem Verfahren Co-Training gehört, dass es eine einfache Methode darstellt. Auch ist sie weniger empfindlich bei Fehlern als die Methode Selbsttraining ([Zhu07], Folie 108). Außerdem arbeitet sie gut, wenn die bedingte Unabhängigkeit hält. Zu beachten ist dabei: „Only works when one classifier correctly labels a data that the other classifier misclassified“ ([Sab13], Folie 13).

5.7 Nachteile

Es kann vorkommen, dass für gelabelte Daten keine natürlichen Eigenschaften zum Spalten in zwei Teilmengen existieren. Außerdem würde das Modell bessere Ergebnisse liefern, wenn beide Features genutzt werden ([Zhu07], Folie 108).

Literatur

[Aut07] Christian Autermann. Boosted Decision Trees. A modern method of data analysis. Website, 2007. Online erhältlich unter http://wwwiexp.desy.de/users/auterman/talks/20070706_hh_bdt.pdf; abgerufen am 23.09.2016.

[Bis02] Manuela Bischoff. AdaBoost. Website, 2002. Online erhältlich unter http://www.manuela-b.de/karriere/HS_NI.pdf; abgerufen am 29.09.2016.

[Bur09] Hans-Dieter Burkhard. Moderne Methoden der KI: Maschinelles Lernen. Entscheidungsbäume. Website, 2009. Online erhältlich unter http://www2.informatik.hu-berlin.de/~hdb/LV/MMKI09/mmki-09-Entscheidungsbaeume.pdf; abgerufen am 23.09.2016.

[Clu09] Team Clusteranalyse. K-Means. Website, 2009. Online erhältlich unter https://www-m9.ma.tum.de/material/felix-klein/clustering/Methoden/K-Means.php; abgerufen am 25.09.2016.

[Fau07] Stefan Fausser. Object recognition with Boosting. Website, 2007. Online erhältlich unter http://www.informatik.uni-ulm.de/ni/Lehre/SS07/SeminarNI/ausarbeitungen/Fausser.pdf; abgerufen am 28.09.2016.

[Gie02] Martin Giese. Vorlesung Lernmethoden in Computervision und Computer Grafik. Thema: Unüberwachtes Lernen I. Website, 2002. Online erhältlich unter http://www.informatik.uni-ulm.de/ni/Lehre/WS02/LMCVCG/skript/Vorlesung9.pdf; abgerufen am 01.10.2016.

[Gie12] David Gier. Einsatz eines Boosted Decision Trees (BDT) für die Messung von Neutrinos aus Richtung des Galaktischen Zentrums. Website, 2012. Bachelorarbeit an der RWTH-Aachen. Online erhältlich unter https://www.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaakaige; abgerufen am 23.09.2016.

[Gra07] Michael Granitzer. Text Mining. Website, 2007. Online erhältlich unter http://markusstrohmaier.info/courses/WS2007-08/

707.000_web-science/slides/week-web-mining2.pdf; abgerufen am 20.09.2016.

[Gra16a] Marco Grawunder. Vorlesung Informationssysteme II. Thema Data Mining: Clustering. Website, 2016. Uni-Oldenburg. Sommersemester 2016.

[Gra16b] Marco Grawunder. Vorlesung Informationssysteme II. Thema Data Mining: Klassifikation. Website, 2016. Uni-Oldenburg. Sommersemester 2016.

[Hü06] Mike Hüftle. Clusterverfahren. Partitionierende Clusterverfahren: K-Means-Algorithmus. Website, 2006. Online erhältlich unter http://www.optiv.de/Methoden/ClustMet/index.htm?14; abgerufen am 25.09.2016.

[Klo15] Olivia Klose. Machine Learning (2) - Supervised versus Unsupervised Learning. Website, 2015. Online erhältlich unter http://www.oliviaklose.com/machine-learning-2-supervised-versus-unsupervised-learning/; abgerufen am 22.09.2016.

[Lan07a] Christoph Langenbruch. Bestimmung des Verzweigungsverhältnisses des Zerfalls $B+ \rightarrow w' + v'$ mit dem BABAR-Experiment. Website, 2007. Diplomarbeit an der Universität Heidelberg. Online erhältlich unter http://www.pi.uni-hd.de/Publications/diploma_langenbruch07.pdf; abgerufen am 22.09.2016.

[Lan07b] Andreas Lanz. Semi-Überwachtes Co-Training. Website, 2007. Online erhältlich unter http://www.informatik.uni-ulm.de/ni/Lehre/SS07/SeminarNI/ausarbeitungen/Lanz.pdf; abgerufen am 08.10.2016.

[Loh12] Hans Lohninger. Grundlagen der Statistik. k-Means Clusteranalyse. Website, 2012. Online erhältlich unter http://www.statistics4u.info/fundstat_germ/ee_kmeans_clustering.html; abgerufen am 25.09.2016.

[Nit05] Christian Nitschke. Wichtige Algorithmen für Machine Learning. Website, 2005. Online erhältlich unter http://ii.ist.i.kyoto-u.ac.jp/~nitschke/Site/Research_files/TIRA_Seminar_Machine_Learning_Algorithmen_Skript.pdf; abgerufen am

19.09.2016.

[Sab13] Jasneet Sabharwal. Semi-Supervised Learning. Website, 2013. Online erhältlich unter https://www.cs.ubc.ca/~schmidtm/MLRG/Semi-Supervised%20Learning.pdf; abgerufen am 07.10.2016.

[Sch13] Sebastian Schmidt. Verbesserung der Untergrundunterdrückung eines neuen CAST Detektors mittels multivariater Methoden aus TMVA. Website, 2013. Bachelorarbeit an der Friedrich-Wilhelms-Universität Bonn. Online erhältlich unter https://www.lhc-ilc.physik.uni-bonn.de/ergebnisse/dateien/t00000040.pdf?c=t&id=40; abgerufen am 22.09.2016.

[Sch14] Alexander Schier. Discrete Exterior Calculus im Maschinellen Lernen. Website, 2014. Online erhältlich unter http://wissrech.ins.uni-bonn.de/research/pub/schier/thesis/thesis_schier.pdf; abgerufen am 07.10.2016.

[SLP+12] Christoph Sawade, Niels Landwehr, Paul Prasse, Dominik Lahmann, and Tobias Scheffer. Maschinelles Lernen. Entscheidungsbäume. Website, 2012. Online erhältlich unter https://www.cs.uni-potsdam.de/ml/teaching/ws12/ml/Entscheidungsbaeume.pdf; abgerufen am 23.09.2016 .

[Sta13] Martin Stahlberg. Verbesserung der Datenselektion in der Neutrinooszillationsanalyse mit IceCube. Website, 2013. Bachelorarbeit an der RWTH Aachen. Online erhältlich unter https://www.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaakaihc; abgerufen am 22.09.2016 .

[Sta16] Statista. Prognose zum Volumen der jährlich generierten digitalen Datenmenge weltweit in den Jahren 2005 bis 2020 (in Exabyte). Website, 2016. Online erhältlich unter https://de.statista.com/statistik/daten/studie/267974/umfrage/prognose-zum-weltweit-generierten-datenvolumen/; abgerufen am 22.09.2016.

[Zhu07] Xiaojin Zhu. Semi-Supervised Learning Tutorial. Website, 2007. Online erhältlich unter http://pages.cs.wisc.edu/~jerryzhu/pub/sslicml07.pdf; abgerufen am 07.10.2016.

[Zhu09] Xiaojin Zhu. Semi-Supervised Learning. Website, 2009. Online erhältlich unter http://pages.cs.wisc.edu/~jerryzhu/ssl/pub/SSL_EoML.pdf; abgerufen am 07.10.2016.

D.2. Hidden Markov Models, Naive-Bayes-Klassifizierer und Gaussian Mixture Models

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Hidden Markov Models, Naive-Bayes-Klassifizierer und Gaussian Mixture Models

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Corinna Feeken

Oldenburg, den 13. April 2017

Inhalt

Abkürzungen

HMM Hidden Markov Model

GMM Gaussian Mixture Model

EM-Algorithmus Expectation-Maximization-Algorithmus

CHMM continuous hidden Marcov Model

Abbildungen

1 Einleitung

Im Rahmen der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ an der Carl von Ossietzky Universität Oldenburg findet die Auswertung von Sensoraufnahmen statt, welche während einer Studie von mehreren Geriatrie-Probanden aufgenommen werden. Die Daten werden mit einem Sensorgürtel aufgenommen, der ein Accelerometer, ein Gyrometer, ein Magnetometer, einen Drucksensor, einen Beschleunigungssensor und zwei Thermometer enthält. Die Probanden tragen diesen Gürtel sowohl jeweils eine durchgehende Woche Zuhause als auch bei den Assessments, die in der Universität durchgeführt werden.
Bei den Assessments werden sechs verschiedene Mobilitäts-Tests durchgeführt, welche teilweise selbst mehrere Untertests beinhalten. Ziel der Projektgruppe ist es eine Software zu entwickeln, die diese Assessment-Tests automatisch klassifiziert. Da die Tests aus einzelnen Alltagsbewegungen bestehen - z.B. setzt sich der „Timed up & go“-Test aus „Aufstehen“, „Gehen“, „Gehen“ und „Hinsetzten“ zusammen - wurden zunächst diese Bewegungen mithilfe von maschinellen Lernalgorithmen klassifiziert. Im nächsten Schritt müssen die Assessment-Tests klassifiziert werden. Als möglicher Lösungsansatz wurde die Klassifizierung durch „Hidden Markov Models“ ins Auge gefasst, da diese die Modellierung von zeitlichen Abhängigkeiten ermöglichen. Ob die Klassifizierung der Assessments-Tests mit den „Hidden Markov Models“ anhand der Reihenfolge ihrer Alltagsbewegungs-Sequenzen in Frage kommt, ist daher Thema dieser Ausarbeitung.
Im folgenden Kapitel 2 werden zuerst einige Grundlagen der Stochastik erläutert. In Kapitel 3 folgt die Definition, ein Beispiel und die drei Grundprobleme des „Hidden Markov Models“. In den Kapiteln 4 und 5 werden zwei weitere stochastische Klassifikatoren bzw. Clustering-Methoden, der „Naive-Bayes-Klassifizierer“ und das „Gaussian Mixture Model“, vorgestellt, welche auch im Zusammenhang mit dem „Hidden Markov Model“ verwendet werden können. Kapitel 6 befasst sich mit der

Anwendung der vorgestellten Modelle in der Aktivitätserkennung. Eine abschließende Zusammenfassung ist in Kapitel 7 zu finden.

2 Grundlagen

Bevor die stochastischen Modelle und Klassifizierer behandelt werden, müssen zunächst einige Grundlagen geklärt werden, deren Definitionen und Erklärungen aus [Zwe15] stammen.

Eine **Zufallsvariable** X ist eine Funktion, welche jedes Ergebnis eines Zufallssexperiments mit der Ergebnismenge $\{x_1, ..., x_n\}$ auf eine reele Zahl abbildet:

$$X := \{x_1, ..., x_n\} \to \mathbb{R}$$

Unterschieden wird zwischen diskreten und stetigen Zufallsvariablen. Diskrete Zufallsvariablen besitzen abzählbar endlich oder unendlich viele Werte (z.B. Anazahl an Würfelaugen), während stetige Zufallsvariablen überabzählbar unendlich viele Werte besitzen (z.B. Körpergröße).

Die **Wahrscheinlichkeitsfunktion** einer diskreten Zufallsvariablen X ist definiert als:

$$f(x) = P(X = x) := \begin{cases} p_i & \text{, wenn } x = x_i, i = 1, ..., n \\ 0 & \text{, } sonst \end{cases}$$

und beschreibt die Wahrscheinlichkeit, dass X den Wert x_i annimmt. Dabei gilt $p_i \in [0, 1]$ und $\sum_{i=1}^{n} p_i = 1$.

Die **Verteilungsfunktion** einer stetigen Zufallsvariablen X

$$F(X) = P(X \leq x) := \int_{-\infty}^{x} f(u)\,\mathrm{d}u$$

gibt die Wahrscheinlichkeit an, dass X einen Wert kleiner oder gleich x annimmt.

Die **Dichtefunktion** einer stetigen Zufallsvariablen X ist der Differentialquotient

der Verteilungsfunktion:

$$f(x) := \frac{dF(x)}{dx}$$

Durch diese Funktion wird der Verlauf der Wahrscheinlichkeiten angegeben. Die Wahrscheinlichkeiten selbst sind Flächeninhalte unterhalb der Dichtefunktion:

$$P(a < X < b) = P(a \leq X \leq b) = F(b) - F(a) = \int_a^b f(x)\,\mathrm{d}x$$

Der **Erwartungswert** μ, die **Varianz** σ^2 und die **Standardabweichung** σ einer diskreten Zufallsvariablen sind gegeben durch

$$\mu = E(X) := \sum\ x \cdot f(x),$$

$$\sigma^2 = V(X) := E[X - E(X)]^2 = E(X^2) - \mu^2 = \sum (x-\mu)^2 \cdot f(x) \quad \text{und}$$

$$\sigma = \sqrt{\sigma^2}.$$

Bei stetigen Zufallsvariablen werden diese Parameter definiert durch

$$\mu = E(X) := \int_{-\infty}^{+\infty} x \cdot f(x)\,\mathrm{d}x,$$

$$\sigma^2 = V(X) := \int_{-\infty}^{+\infty} (x-\mu)^2 \cdot f(x)\,\mathrm{d}x \quad \text{und}$$

$$\sigma = \sqrt{\sigma^2}.$$

Eine **Normalverteilung** (auch Gaußsche Normalverteilung genannt) ist eine spezielle Dichtefunktion. Sie besitzt eine Glockenform und verläuft symmetrisch zur x-Achse. Das Maximum liegt an der Stelle x=μ und je kleiner die Streuung um den Erwarungswert ist, d.h. je kleiner die Varianz ist, desto steiler verläuft die Kurve (siehe Abbildung 1).
Die formale Definition der Normalverteilung lautet:

$$f(x|\mu,\sigma) = \mathcal{N}(\mu,\sigma) := \frac{1}{\sigma \cdot \sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Die **Kovarianz** ist ein Maß für den Zusammenhang zweier Zufallsvariablen. Hierbei geht es um das gleichzeitige Abweichen zweier Variablen. Die Formel für die Kovarianz

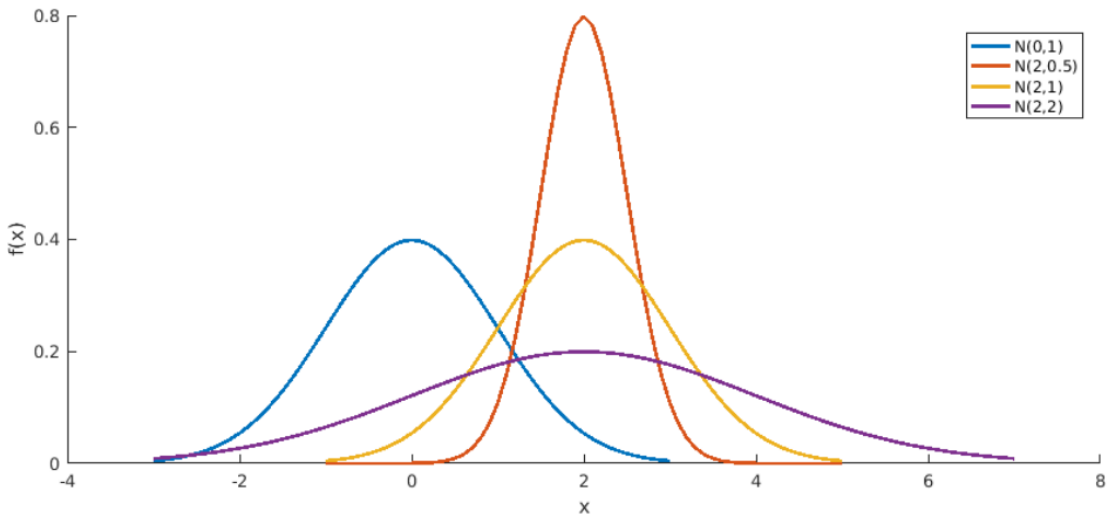


Abbildung 1: Normalverteilungen mit verschiedenen Erwartungswerten und Standardabweichungen

lautet

$$\sigma_{xy} := \frac{1}{N} \cdot \sum_{i=1}^{N} (x_i - \mu_x) \cdot (y_i - \mu_y) = \frac{1}{N} \cdot \sum_{i=1}^{N} (x_i y_i - \mu_x \mu_y).$$

Werden mehr als zwei Variablen betrachtet, können die Kovarianzen in einer sogenannten Kovarianzmatrix zusammengefasst werden.

3 Hidden Markov Models

Das Hidden Markov Model (HMM) ist ein stochastisches Modell mit dessen Hilfe Abhängigkeiten von Zuständen modelliert werden können. Es lässt sich als ein endlicher Zustandsautomat beschreiben, bei dem sichtbare Ausgaben durch nicht-sichtbare Zustände erzeugt werden. In Kombination mit verschiedenen Algorithmen, die in 3.2 vorgestellt werden, kann dieses Modell als Klassifikator genutzt werden. Das Modell wird für häufig für Spracherkennung und auch für Aktivitätserkennung eingesetzt [DHS00].
Im Folgenden wird zuerst der Begriff der Markov-Kette erläutert, anschließend folgt die Definition und ein Beispiel für das Hidden Marvov Model.

3.1 Definition

Eine **Markov-Kette n-ter Ordnung** ist definiert als ein stochastischer Prozess (Abfolge von zufälligen Ereignissen), welcher die Eigenschaft $P(X_{t+1} = S_{t+1}|X_t = S_t, ..., X_n = S_n) = P(X_{t+1} = S_{t+1}|X_i = S_i, i \in 0, ..., t)$ erfüllt. D.h. die Wahrscheinlichkeit in einen Folgezustand zu wechseln ist von den n vorherigen Zuständen abhängig.
Bei einer **Markov-Kette 1-ter Ordnung** hängt die Wahrscheinlichkeit in den Folgezustand S_{t+1} zu wechseln also nur von dem aktuellen Zustand S_t und nicht von den davorigen Zuständen ab. Die Eigenschaft $P(X_{t+1} = S_{t+1}|X_t = S_t) = P(X_{t+1} = S_{t+1}|X_i = S_i, i \in 0, ..., t)$ dieser Markov-Kette wird auch **Markov-Eigenschaft** genannt.
Eine Markov-Kette ist **zeit-homogen** (auch homogene Markov-Kette genannt), wenn

$$\forall t \in T, \forall i, j \in S : P(X_{t+1} = i|X_t = j) = P(X_t = i|Xt - 1 = j)$$

gilt, d.h. wenn die Zustandsübergangs-Wahrscheinlichkeiten zu jedem Zeitpunkt für denselben Zustand den gleichen Folgezustand liefern.
Für eine homogene Markov-Kette kann eine **Transitions-Matrix**

$$M_S = (s_{ij}), \forall i, j \in S : s_{ij} \geq 0 \wedge \sum_{j \in S} s_{ij} = 1$$

und ein **initialer Verteilungsvektor** $\pi = (\pi_i)$ mit $\pi_i = P(X_0 = i), i \in S$ aufgestellt werden [Koh07].

Formal kann ein HMM als 5-Tupel beschrieben werden: $\Omega = (S, M_S, \pi, E, M_E)$. Hierbei stellt das Tripel (S, M_S, π) eine homogene Markov-Kette 1-ter Ordnung dar. Abweichend von der Markov-Kette, sind die Zustände $s_t \in S$ zum Zeitpunkt t nicht beobachtbar (daher der Name „*Hidden* Markov Model“). Allerdings erfolgt zu jedem Zeitpunkt t im System eine Ausgabe $e \in E$, welche beobachtet werden kann. Welche Ausgabe aus der Menge von Ausgaben zum Zeitpunkt t erfolgt, ist durch Wahrscheinlichkeiten bedingt und nur vom aktuellen Zustand abhängig. Diese Wahrscheinlichkeiten können in einer Matrix

$$M_E = (e_{ij}), \forall i, j \in E : e_{ij} \geq 0 \wedge \sum_{j \in E} e_{ij} = 1$$

notiert werden. Damit ergeben sich zusammenfassend folgende Tupel-Objekte für das HMM [Koh07]:

- S: Menge an nicht-beobachtbaren Zuständen
- M_S: Transitionsmatrix
- π: initialer Verteilungsvektor
- E: Menge an beobachtbaren Ausgaben
- M_E: Matrix für Ausgabe-Wahrscheinlichkeiten

Beispiel

Ein Beispiel für ein HMM ist in Abbildung 2 dargestellt. Die Zustandübergänge mit der Wahrscheinlichkeit null (entspricht den Schlaufen) wurden zur besseren Übersicht weggelassen. Hierbei ist die Menge der nicht sichtbaren Zustände als

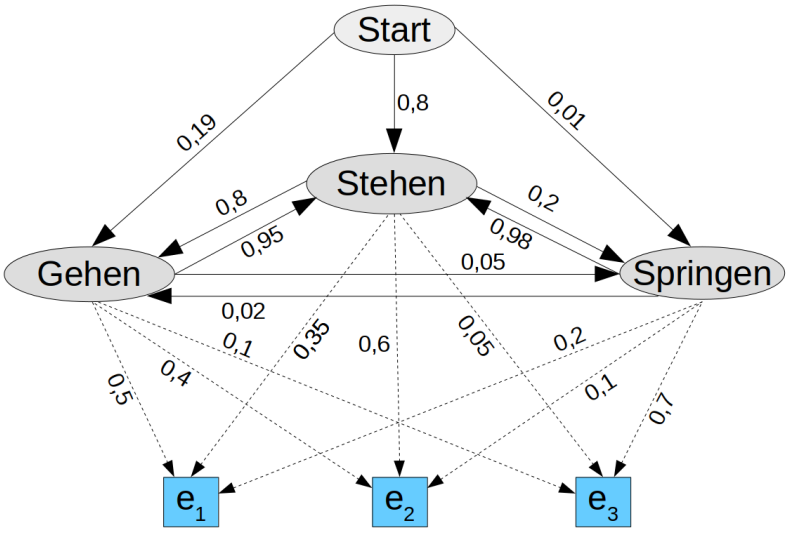


Abbildung 2: Beispiel eines Hidden Markov Models

$S = \{Gehen, Stehen, Springen\}$ und die Menge der beobachtbaren Ausgaben als $E = \{e_1, e_2, e_3\}$ gegeben. Die Anfangswahrscheinlichkeiten für die Bewegungsmuster sind $\pi = (0,19; 0,8; 0,01)$. Anhand der Transitions-Wahrscheinlichkeiten und den Ausgabe-Wahrscheinlichkeiten lassen sich die beiden Matritzen

$$M_S = \begin{array}{c} \\ Gehen \\ Stehen \\ Springen \end{array} \begin{array}{c} \begin{array}{ccc} Gehen & Stehen & Springen \end{array} \\ \begin{pmatrix} 0 & 0,95 & 0,05 \\ 0,8 & 0 & 0,2 \\ 0,02 & 0,98 & 0 \end{pmatrix} \end{array}$$

und

$$M_E = \begin{array}{c} \\ Gehen \\ Stehen \\ Springen \end{array} \begin{array}{c} \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{pmatrix} 0,5 & 0,4 & 0,1 \\ 0,35 & 0,6 & 0,05 \\ 0,2 & 0,1 & 0,7 \end{pmatrix} \end{array}$$

aufstellen.

3.2 Probleme des Hidden Markov Models

Nach dem Modellieren des HMMs können verschiedene Fragestellungen formuliert werden, welche mithilfe des Modells und entsprechenden Algorithmen beantwortet werden können. Angenommen ein HMM $\Omega = (S, M_S, \pi, E, M_E)$, sowie eine Sequenz

von beobachtbaren Ausgaben sind bekannt. Zum einen kommt die Frage auf, wie hoch die Wahrscheinlichkeit ist, dass diese Sequenz von dem gegebenen HMM produziert wird (**Evaluationsproblem**) und zum anderen welche Sequenz von unbeobachtbaren Zuständen die höchste Wahrscheinlichkeit besitzt diese Ausgabe-Sequenz produziert zu haben (**Decodierungsproblem**). Ein weiteres Problem ist das **Lernproblem**, bei welchem das HMM trainiert wird und bei gegebener Menge der unbeobachtbaren Zuständen S und einer Beobachtungs-Sequenz die Transitionsmatrix M_S und die Ausgabematrix M_E zurückgegeben werden [DHS00] [JM16].

3.2.1 Evaluationsproblem

Das Evaluationsproblem lässt sich folgendermaßen definieren:
Gegeben sei ein HMM $\Omega = (S, M_S, \pi, E, M_E)$ und eine Beobachtungssequenz $O^T = o_1, ..., o_T$ der Länge T ($o_i \in E$), welche von der nicht bekannten Zustandssequenz $S^T = s(1), ..., s(t)$ ($s(i) \in S$) produziert wird. Gesucht ist die Wahrscheinlichkeit $P(O^T|\Omega)$, d.h. die Wahrscheinlichkeit, dass die Sequenz O^T von dem vorliegendem HMM Ω produziert wurde.
Ein grundlegender Ansatz für die Lösung dieses Problems wäre es mithilfe der Transitionsmatrix die Wahrscheinlichkeiten für jeden Pfad im HMM, der die Beobachtungssequenz produzieren kann, zu berechnen und mithilfe der Ausgabematrix die Wahrscheinlichkeiten für die jeweiligen Ausgaben im entsprechenden Zustand hinzu zu multiplizieren. Über all diesen Wahrscheinlichkeiten für die Pfade muss anschließend noch die Summe gebildet werden. Es wird also

$$P(O^T) = \sum_{r=1}^{r_{max}} \prod_{t=1}^{T} P(s(t)|s(t-1)) \cdot P(o_t|s(t))$$

berechnet, wobei r_{max} die Anzahl an möglichen Zustandssequenzen ist, von denen O^T produziert werden kann und $s(t)$ bzw. o_t einen Zustand bzw. die Ausgabe zum Zeitpunkt t beschreibt. Dieses Verfahren liegt jedoch in der Komplexitätsklasse $\mathcal{O}(|S|^T \cdot T)$, sodass der Rechenaufwand bei einer großes Zustandsmenge und einer langen Sequenz sehr hoch werden kann. Ein effizienteres Verfahren mit der Komplexitätsklasse $\mathcal{O}(|S|^2 \cdot T)$ stellt der **Forward-Algorithmus** dar. Der Forward-Algorithmus benutzt eine Tabelle um die Zwischenwerte der Wahrscheinlichkeiten für die Pfade, die die gegebene Ausgabe produzieren können, zu speichern anstatt

sie jedes Mal erneut zu berechnen. $P(O^T)$ kann so rekursiv mit der Vorschrift

$$\alpha_i(t) := \begin{cases} 0 & \text{, wenn } t = 0 \text{ und } i \neq start \\ 1 & \text{, wenn } t = 0 \text{ und } i = start \\ \sum_{j=1}^{|S|} \alpha_j(t-1) \cdot s_{ji} \cdot e_i(o_t) & \text{, sonst} \end{cases}$$

berechnet werden, wobei $i \in 1, ..., |S|$ und $e_i(o_t)$ für die Ausgabewahrscheinlichkeit von o_t im Zustand s_i steht. Mit $\alpha_i(t)$ wird demnach die Wahrscheinlichkeit ausgedrückt, dass das HMM sich zum Zeitpunkt t im Zustand s_i befindet und die ersten t Ausgaben von der Sequenz O^T produziert hat, also $\alpha_i(t) = P(o_1, ..., o_t, s_i(t)|\Omega)$. $\alpha_i(t)$ wird auch **Forward-Wahrscheinlichkeit** genannt. Die Lösung des Evaluationsproblems stellt also die Wahrscheinlichkeit $\alpha_i(T)$ dar. Der Pseudocode für den Forward-Algorithmus ist in 3.1 zu finden [DHS00] [JM16].

Listing 3.1: Forward-Algorithmus

```
1 initialize o(1), t=0, s_ij, e_ij, V^T, α(0) = 1
2         for t ← t+1
3                 α_i(t) ← Σ_{j=1}^{|S|} α_j(t−1) · s_ji · e_i(o_t)
4         until t = T
5 return P(O^T) ← α_i(T)
6 end
```

3.2.2 Decodierungsproblem

Beim Decodierungsproblem wird zu einem gegebenen HMM $\Omega = (S, M_S, \pi, E, M_E)$ und einer gegebenen Ausgabe-Sequenz $O^T = o_1, ..., o_T$ der Länge T die Abfolge von nicht-sichtbaren Zuständen $S' = s(1), ..., s(T)$ gesucht, die die größte Wahrscheinlichkeit besitzen O^T produziert zu haben. Analog zum Evaluationsproblem ist der Lösungungsansatz nacheinander jeden Pfad auszuprobieren sehr rechenaufwendig. Stattdessen kann der **Viterbi-Algorithmus** verwendet werden, welcher dem Forward-Algorithmus sehr ähnelt. Der Unterschied besteht darin, dass beim Viterbi-Algorithmus bei der Berechnung in den einzelnen Zellen nicht die Summe, sondern das Maximum berechnet wird. Für die Viterbi-Pfad-Wahrscheinlichkeit, die

zu jedem Zeitpunkt für jeden Zustand berechnet wird, ergibt sich somit:

$$\beta_i(t) = \max_{j=1}^{|S|} \beta_j(t-1) \cdot s_{ji} \cdot e_i(o_t)$$

Nachdem alle Wahrscheinlichkeiten berechnet wurden, wird mittels Backtracing für jeden Zeitpunkt t der Zustand s_i ermittelt, welcher das maximale $\beta_i(t)$ liefert. Der Pseudo-Algorithmus ist in 3.2 zu finden [DHS00] [JM16].

Listing 3.2: Viterbi-Algorithmus

```
1  begin initialize Path = {}, t = 0
2          for t ← t+1
3                  i = 0, β0 = 0
4                  for i ← i+1
5                          βi(t) ← max_{j=1}^{|S|} βj(t−1) · sji · ei(ot)
6                  until i = |S|
7                  j' ← argmax_j βj(t)
8                  Append sj' to Path
9          until t = T
10 return Path
11 end
```

Zu beachten ist, dass es unter Umständen möglich ist, dass die gefundene Sequenz von Zuständen nicht erlaubt ist, weil die Übergangswahrscheinlichkeit zweier Zustände den Wert 0 beträgt, aber diese beiden Zustände zu zwei aufeinanderfolgenden Zeitpunkten die maximale Viterbi-Pfad-Wahrscheinlichkeit besitzen [DHS00].

3.2.3 Lernproblem

Das Lernproblem beschäftigt sich damit für ein HMM Ω mit der gegebenen Zustandsmenge S, der Ausgabemenge E und einer Ausgabesequenz $O^T = o_1, ..., o_T$ der Länge T die Transitionsmatrix M_S und die Ausgabe-Matrix M_E zu trainieren. Hierfür wird der **Forward-backward-Algorithmus** (auch **Baum-Welch-Algorithmus**) verwendet, welcher ein Expectation-Maximization-Algorithmus (EM-Algorithmus) darstellt. Ausgehend von initialen Parameterwerten werden mehrere Iteratonen durchlaufen, in denen abgeschätzt wird wie gut das HMM mit den zugrundeliegenden Parametern die Trainingsdaten modelliert und anschließend jeweils die Matrixeinträge angepasst werden, bis ein Abbruchkriterium (z.B. Konvergenz) eintritt. Zusätzlich zur Forward-Wahrscheinlichkeit $\alpha_i(t)$ aus Kapitel 3.2.1 wird die **Backward-Wahrscheinlichkeit**

$\gamma_i(t)$ definiert. Sie drückt aus wie wahrscheinlich es ist, dass das HMM sich im Zustand s_i befindet und die noch folgenden Ausgaben von O^T produzieren wird.

$$\gamma_i(t) := \begin{cases} 0 & \text{, wenn } t = T \text{ und } s_i(t) \neq \text{ Endzustand für Ausgabesequenz} \\ 1 & \text{, wenn } t = T \text{ und } s_i(t) = \text{ Endzustand für Ausgabesequenz} \\ \sum_{j=1}^{|S|} s_{ij} \cdot e_j(o_{t+1}) \cdot \gamma_j(t+1) & \text{, sonst} \end{cases}$$

Da die Parameter s_{ij} und $e_i(o_{t+1})$ nicht bekannt sind, wird zuerst die Wahrscheinlichkeit dafür berechnet, dass ein Zustandsübergang von s_i zu s_j zum Zeitpunkt t zu $t+1$ stattfindet und die vollständige Ausgabesequenz durch eine beliebige Zustandssequenz erzeugt wurde:

$$\rho_{ij}(t) := \frac{\alpha_i(t) \cdot s_{ij} \cdot e_j(o_{t+1}) \cdot \gamma_j(t+1)}{P(O^T|\Omega)}$$

mit

$$P(O^T|\Omega) = \alpha_i(T) = \gamma_0(T) = \sum_{j=1}^{|S|} \alpha_j(t) \cdot \gamma_j(t)$$

Um nun eine Schätzung für die Transitionsmatrix zu erhalten, wird das Verhältnis von der angenommenen Anzahl der Zustandsänderungen von s_i zu s_j und der Anzahl aller Zustandswechsel aus s_i berechnet:

$$s'_{ij} := \frac{\sum_{t=1}^{T} \rho_{ij}(t)}{\sum_{t=1}^{T} \sum_{k=1}^{|S|} \rho_{ik}(t)}$$

Für die Schätzung der Ausgabe-Matrix wird ähnlich verfahren. Hier wird die Häufigkeit einer bestimmten Ausgabe ins Verhältnis gesetzt mit der Häufikeit einer beliebigen Ausgabe.

$$\rho'_j(t) := \frac{\alpha_j(t) \cdot \gamma_j(t)}{P(O^T|\Omega)}$$

und

$$e'_j(o_k) := \frac{\sum_{t=1, O(t)=o_k}^{T} \rho'_j(t)}{\sum_{t=1}^{T} \rho'_j(t)}$$

Der Index $t = 1, O(t) = o_k$ bedeutet, dass zum Zeitpunkt t die Ausgabe o_k erfolgt sein muss, damit der Summand für den Index t in die Summme mit eingeht.

Der Pseudocode für diesen Algorithmus ist in 3.3 zu finden [DHS00] [JM16].

Listing 3.3: Forward-Backward-Algorithmus

```
1  begin initialize s_ij , e_ij , O^T , Konvergenzkriterium ϵ
2  iterate until convergence
3          ρ_ij(t) ← α_i(t)·s_ij·e_j(o_{t+1})·γ_j(t+1) / P(O^T|Ω) ∀t, i, j
4          ρ'_j(t) ← α_j(t)·γ_j(t) / P(O^T|Ω) ∀t, j
                      Σ_{t=1}^{T} ρ_ij(t)
5          s'_ij ← ------------------------------
                   Σ_{t=1}^{T} Σ_{k=1}^{|S|} ρ_ik(t)
                              Σ_{t=1, O(t)=o_k}^{T} ρ'_j(t)
6          e'_j(o_k) ← ------------------------------
                       Σ_{t=1}^{T} ρ'_j(t)
7  return s'_ij , e'j(o_k)
8  end
```

4 Naiver-Bayes-Klassifizierer

Der **Naive-Bayes-Klassifizierer** ist ein linearer Klassifizierer, der auf dem Satz von Bayes beruht. Er gilt als sehr einfacher, aber effizienter Klassifizierer und geht zur Vereinfachung davon aus, dass die Merkmale unabhängig voneinander sind. Diese Unabhängigkeiten treffen aber in der Realität meist nicht zu, weshalb der Klassifizierer „*Naiver*-Bayes-Klassifizierer“ genannt wird. Trotzdem erzielt er in der Praxis meist gute Ergebnisse [Ras14].

Satz von Bayes:
Seien $x_1, ..., x_n$ und y Zufallsvariablen, dann gilt

$$P(y|x_1, ..., x_n) = \frac{P(x_1, ..., x_n|y) \cdot P(y)}{P(x_1, ..., x_n)}.$$

Im Kontext der Klassifizierung können $x_1, ..., x_n$ als die möglichen Merkmale und y als die resultierende Klasse betrachtet werden. Da die Kombinationsmöglichkeiten für die bedingte abhängige Wahrscheinlichkeit $P(x_1, ..., x_n|y)$ bei einer großen Anzahl an Merkmalen sehr hoch werden kann, werden die Merkmale zur Vereinfachung als bedingt unabhängig angenommen [Ert16]:

$$P(x_1, ..., x_n|y) = P(x_1|y) \cdot ... \cdot P(x_n|y).$$

Durch diese Annahme erhält man:

$$P(y|x_1, ..., x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}$$

Von allen Klassen y_j wird nun das mit der höchsten Wahrscheinlichkeit unter der

Bedingung $x_1, ..., x_n$ ermittelt [Ert16]:

$$y = \underset{j=1,...,k}{\operatorname{argmax}} P(y_j|x_1, ..., x_n)$$

$$= \underset{j=1,...,k}{\operatorname{argmax}} \frac{P(y_j) \cdot \prod_{i=1}^{n} P(x_i|y_j)}{P(x_1, ..., x_n)}$$

$$= \underset{j=1,...,k}{\operatorname{argmax}} P(y_j) \cdot \prod_{i=1}^{n} P(x_i|y_j)$$

5 Gaussian Mixture Model

Ein Gaussian Mixture Model (GMM, deutsch: Gaußsche Mischverteilung) ist ein Modell zur Beschreibung von Merkmalsvektor-Ausprägungen mittels Gaußschen Normalverteilungen und kann als Clustering-Methode verwendet werden. Soll eine Klassifizierung erfolgen, so muss jede Klasse durch ein Gaussian Mixture Model (GMM) repräsentiert werden, welches die Gesamtverteilungsdichte aller Merkmalsvektoren dieser Klasse beschreibt. Durch die anschließende Bestimmung welches GMM für die zu klassifizierenden Daten die maximale Wahrscheinlichkeit liefert (z.B. durch den Naive-Bayes-Klassifizierer), kann die Klasse bestimmt werden.

5.1 Definition

Zur Beschreibung der Merkmalsvektor-Ausprägungen wird die Verteilungsdichte berechnet. Da sich die Verteilungsdichte durch nur eine Gaußsche Normalverteilung schwer darstellen lässt, erfolgt eine bessere Approximation durch eine Mischverteilung, d.h. durch eine gewichtete Summe mehrerer Normalverteilungen. Dies ist zur besseren Veranschaulichung in Skizze 3 dargestellt.

Die Verteilungsdichtefunktion eines GMM ist wie folgt definiert:

$$p(x|\lambda) := \sum_{i=1}^{M} w_i \cdot \mathcal{N}_i(x)$$

$$\text{mit} \quad \sum_{i=1}^{M} w_i = 1 \quad \text{und}$$

$$\mathcal{N}_i(x) := \frac{1}{(2\pi)^{N/2}\sqrt{|C_i|}} exp\left(-\frac{1}{2} \cdot (x-\mu_i) \cdot C_i^{-1} \cdot (x-\mu_i)^T\right)$$

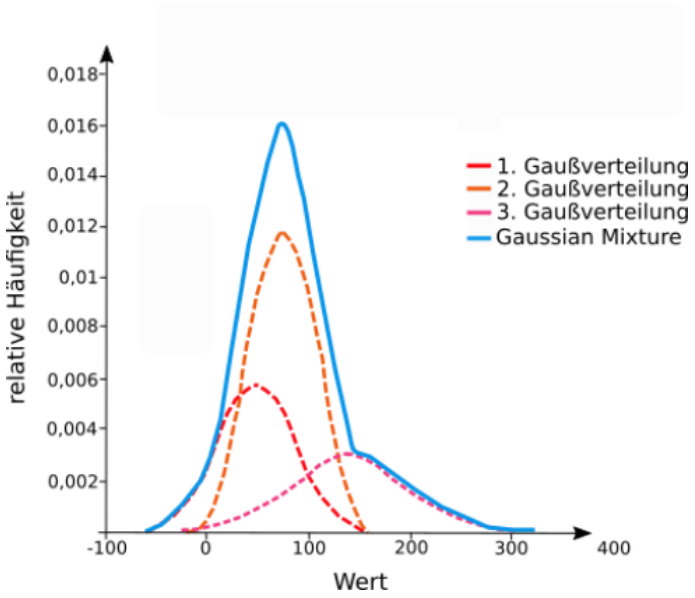


Abbildung 3: Skizze: Approximation der Verteilungsdichte durch meherere Normalverteilungen (vgl. [Rho03])

Hierbei seien

N die Anzahl an Merkmalen,

T die Anzahl an Vektoren von Merkmalsausprägungen,

M die Anzahl an Gaußschen Normalverteilungen,

$X \in \mathbb{R}^{T \times N}$ eine Matrix mit Merkmalsausprägungen,

$x_t \in \mathbb{R}^{1 \times N}$ der t-ter Zeilenvektor von X (ein Merkmalsausprägungs-Vektor),

$\mu \in \mathbb{R}^{M \times N}$ eine Erwartungswert-Matrix,

$C \in \mathbb{R}^{M \times N \times N}$ eine Kovarianzmatrix und

$w \in \mathbb{R}^{M \times 1}$ ein Gewichtungsvektor.

Der Index i indexiert jeweils die Gaußsche Komponente. Die Parameter μ und C für ein GMM werden zusammengefasst als $\lambda = \{\mu, C\}$ [Rho03]. Ein Beispiel eines GMMs mit vier Komponenten ist in Abbildung 4 dargestellt.

5.2 Training

Für das Training des Modells wird der EM-Algorithmus verwendet. Hierfür wird für jede Klasse ein Modell antrainiert. Als Eingabe dient die Matrix X, welche aus den Trainingsdaten alle Merkmalsausprägungs-Vektoren der entsprechenden Klasse

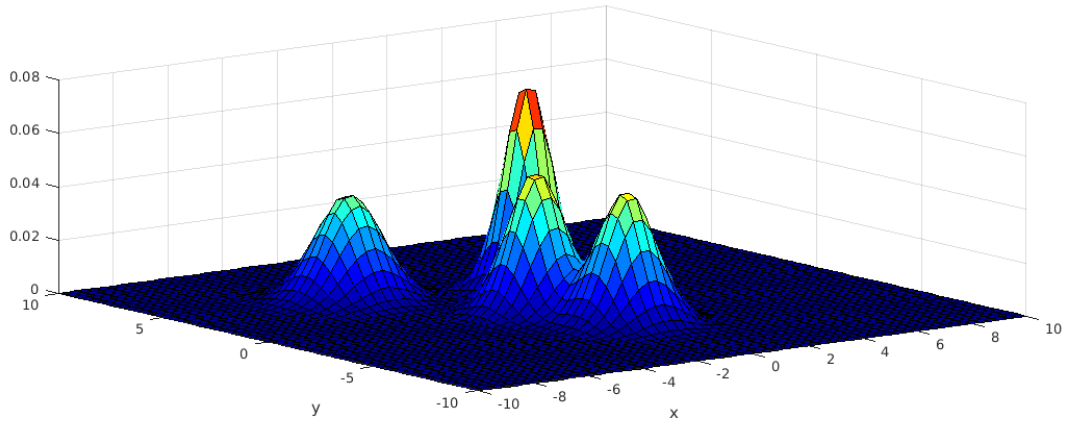


Abbildung 4: Gaussian Mixture Model mit vier Komponenten

beinhaltet. Für die Parameter w, μ und C müssen zunächst initiale Werte festgelegt werden (häufig mithilfe von K-Means) und die Anzahl der Normalverteilungen M und ein Konvergenzkriterium müssen festgelegt werden. Ziel des Algorithmus ist es die Modellparameter so anzupassen, dass die Auftrittswahrscheinlichkeit der Trainingsdaten X maximiert wird [AR11]:

$$\operatorname{argmax} p(X|\lambda) = \operatorname{argmax} \sum_{t=1}^{T} p(x_t|\lambda)$$

Der Algorithmus führt in jeder Iteration den E-Schritt und danach den M-Schritt aus. Im E-Schritt erfolgt die Berechnung der Auftritts-Wahrscheinlichkeit für die Trainingsdaten X bei gegeben Parametern, die in der letzten Iteration berechnet wurden (bzw. für die initialen Parameterwerte). Dies entspricht einer Evaluation der abgeschätzten Modellparameter, es wid also überprüft wie gut das GMM mit den vorliegenden Parametern die Trainingsdaten modelliert. Im darauffolgenden M-Schritt wird diese Wahrscheinlichkeit maximiert, indem die Parameterwerte neu berechnet werden. Der Algorithmus terminiert, wenn ein Konvergenzkriterium erfüllt wurde. Dies kann z.B. die Unterschreitung eines Schwellwerts für die Differenz zweier Wahrscheinlichkeiten aus aufeinanderfolgenden Iterationen sein, d.h. die Wahrscheinlichkeit für das Auftreten von X durch λ ändert sich nur noch kaum merklich.

Im Folgenden wird der EM-Algorithmus für GMMs ausführlicher beschrieben. Eine

vollständige Herleitung und Erklärung ist in [Bil98] zu finden.

Zunächst wird die **Log-Wahrscheinlichkeits-Funktion** eingeführt, welche maximiert werden soll:

$$\mathcal{L}(\lambda|X) = log\left(\prod_{t=1}^{T} p(x_t|\lambda)\right) = \sum_{t=1}^{T} log\left(\sum_{i=1}^{M} w_i \cdot \mathcal{N}_i(x_t)\right)$$

Da diese Funktion den Logarithmus einer Summe beinhaltet, ist sie noch schwer zu berechnen. Daher wird zur Vereinfachung angenommen, dass X „unvollständig“ ist und zusammen mit einem neu eingeführten Vektor Y einen vollständigen Datenssatz bildet. Die Komponenten von $Y = \{y_1, ..., y_T\}$ mit $y_i \in \{1, ..., M\}$ sind unbekannt und geben an welches Element x_t von welcher Dichtefunktion produziert wird. Es gilt $y_t = m$, wenn x_t von der m-ten Dichtefunktion generiert wurde. Für die neue Log-Wahrscheinlichkeits-Funktion, die Y berücksichtigt, ergibt sich somit:

$$\begin{aligned} log(\mathcal{L}(\sigma|X,Y)) &= log(P(X,Y|\lambda)) \\ = \sum_{t=1}^{T} log(P(x_t|y_t) \cdot P(y)) &= \sum_{t=1}^{T} log(w_{y_t} \cdot p_{y_t}(x_t|\lambda_{y_t})) \end{aligned} \tag{5.1}$$

Es wird nun eine Hilfsfunktion

$$Q(\lambda, \lambda^{i-1}) := E[logp(X,Y|\lambda)|X, \lambda^{i-1}]$$

eingeführt, wobei λ^{i-1} die derzeit abgeschätzten Parameter sind, die zur Evaluierung genutzt werden und λ die neuen Parameter sind, die optimiert werden sollen. Um die Wahrscheinlichkeit zu maximieren, muss in jeder Iteration

$$\mathcal{L}(\lambda) > \mathcal{L}(\lambda^{i-1})$$

gelten. Anders ausgedrückt kann stattdessen auch die Differenz der beiden Wahrscheinlichkeiten zweier aufeinanderfolgenden Iterationen maximiert werden. Die Hilfsfunktion Q kann als Schranke dieser Differenz ausgedrückt werden:

$$\mathcal{L}(\lambda) - \mathcal{L}(\lambda^{i-1}) \geq Q(\lambda, \lambda^{i-1})$$

Statt die Log-Wahrscheinlichkeit zu maximieren kann auch die Funktion Q maximiert

werden und so zur Abschätzung der neuen Parameter dienen:

$$\lambda = \underset{\lambda}{\operatorname{argmax}}\, Q(\lambda, \lambda^{i-1})$$

Aus Gleichung 5.1 und der Definition der Hilfsfunktion Q ergibt sich

$$Q(\lambda, \lambda^{i-1}) = \sum_{m=1}^{M} \sum_{t=1}^{T} log(w_m) \cdot p(m|x_t, \lambda^{i-1}) + \sum_{m=1}^{M} \sum_{t=1}^{T} log(p_m(x_t|\lambda_m) \cdot p(m|x_t, \lambda^{i-1})$$

Somit haben sich zwei unabhängige Terme gebildet, welche nun einzeln maximiert werden. Während der erste Term von dem Gewichtsvektor abhängig ist, ist der zweite Term von λ abhängig. Die Abschätzung in jeder Iteration für die neuen Parameter wird also folgendermaßen berechnet, wobei die Reihenfolge eingehalten werden muss, da Abhängigkeiten bestehen:

$$w_i = \frac{1}{T} \sum_{t=1}^{T} P(i|x_t, \lambda)$$

$$\mu_i = \frac{\sum_{t=1}^{T} P(i|x_t, \lambda) \cdot x_t}{\sum_{t=1}^{T} P(i|x_t, \lambda)}$$

$$C_i = \frac{\sum_{t=1}^{T} P(i|x_t, \lambda) \cdot x_t^2}{\sum_{t=1}^{T} P(i|x_t, \lambda)} - \mu_i^2$$

Hierbei beschreibt

$$P(i|x_t, \lambda_i) = \frac{w_i \cdot \mathcal{N}_i(x_t)}{\sum_{m=1}^{M} w_m \cdot \mathcal{N}_m(x_t)}$$

die Wahrscheinlichkeit, dass x_t bei gegebenen λ zur Gaußschen Komponenete i gehört [AR11] [Bil98].

6 Anwendung in der Aktivitätserkennung

Nachdem bereits die Theorie der Klassifizierer bzw. der Clustering-Verfahren erläutert wurde, sollen nun Beispiele gegeben werden wie die Anwendung hiervon in der praktischen Aktivitätserkennung aussehen kann.

Neben der Verwendung als eigenständiger Klassifikator kann das Hidden Markov Model auch zur Korrektur von bereits klassifizierten Daten genutzt werden. Dieses Verfahren wird in [YMJ15] anhand von Daten eines Video-Motion-Capture-Systems erläutert. Im ersten Schritt wurden die Daten mit einem Klassifizierer in die unterschiedlichen Bewegungsklassen eingeteilt. Im zweiten Schritt sollten die mit der höchsten Wahrscheinlichkeit falsch klassifizierten Bewegungen von einem HMM korrigiert werden. Dazu wurden ein HMM mit folgenden Anfangsparametern trainiert: Die Transitionsmatrix wurde anhand der klassifizierten Daten aus dem ersten Schritt berechnet, die Ausgabematrix wurde durch die Confusion-Matrix des ersten Klassifizieres berechnet und die initialen Wahrscheinlichkeiten sind gleichverteilt. Durch die anschließende Klassifizierung mit dem HMM konnte die Klassifizierungsgenauigkeit um 7,79 % verbessert werden. Auch die Laufzeit erzielte mit einem 1.6 GHz Intel Core i7-Prozessor und Matlab (mit einem Thread) Ergebnisse, die Echtzeit-Ausführungen ermöglichen würden. Für ein HMM mit einem Fenster mit weniger als 150 Beobachtungen betrug die Rechenzeit weniger als eine Sekunde.

In [RC17] wurde eine Klassifizierung anhand von mehreren HMMs durchgeführt. Es wurden die Accelerometer- und Gyroskopdaten eines Smartphones, welches an der Taille getragen wurde und eine Aufnahmefrequnez von 50 Hz hat, verwendet. Insgesamt wurden 30 Datensätze aufgenommen, 21 (70 %) hiervon wurden zum Training

benutzt und 9 (30 %) zum Testen. Klassifiziert wurde nach „Stehen“, „Sitzen“, „Liegen“, „Gehen“, „Treppe hochsteigen“ und „Treppe runtersteigen“. Als Beobachtungen wurden die Sensorwerte verwendet. Da diese aber im reelen Zahlenbereich liegen, wurden CHMM verwendet. Bei diesen werden die Ausgaben als GMMs modelliert. Für die Merkmalsauswahl wurde „Random forest variable importance“ (RF VI), eine Kombination mehrerer Lernalgorithmen zur Merkmalsauswahl, eingesetzt. Die Grundidee bei dieser Klassifizierung ist, dass es einer hierarchischen Struktur folgt. Im ersten Schritt wird nach „statisch“und „dynamisch“ trainiert bzw. klassifiziert und im zweiten Schritt wird die genaue Aktivität ermittelt.
Beim Training wird wie folgt vorgegangen: Die Daten werden zunächst vorverarbeitet, dies beinhaltet die Merkmalsextraktion und Merkmalsauswahl durch RF VI und anschließender Normierung. Danach dienen die Trainingsdaten für statische Aktivitäten als Beobachtung für ein CHMM und die Daten für dynamische Aktivitäten für ein anderes CHMM mit jeweils zwei Zuständen. Dieser Trainingsprozess bildet die erste Schicht. Beim Training der zweiten Schicht wird ähnlich verfahren. Die Merkmalsauswahl erfolgt diesmal seperat, sodass es ein Merkmals-Set für alle statischen und ein Merkmals-Set für alle dynamischen Aktivitäten gibt. Anschließend erfolgt das Training, indem die Trainingsdaten jeder Aktivität als Beobachtung für jeweils ein eigenes CHMM dienen. Insgesamt gibt es in der zweiten Schicht also sechs Modelle. Diese besitzen anfänglich drei Zustände, dies wurde später jedoch noch teilweise angepasst.
Für die Klassifikation werden die gleichen Merkmale verwendet wie beim Training und die Wahrscheinlichkeiten, dass die resultierenden Beobachtungen durch die CHMMs für dynamische bzw. statische Aktivitäten erzeugt wurden, wird berechnet. Das CHMM, das die größere Wahrscheinlichkeit zurückliefert, bestimmt ob die Daten von einer statsichen oder dynamischen Aktivität stammen. Danach werden in der zweiten Schicht alle als statisch bzw. dynamisch klassifizierten Daten nach dem selben Prinzip der Klasse „Stehen“, „Sitzen“ oder „Liegen“ bzw. „Gehen“, „Treppe hochsteigen“ oder „Treppe runtersteigen“ zugeordnet. Eine Veranschaulichung des Training- und Klassifikationsprozesses ist in Abbildung 5 und 6 zu finden.

Insgesamt wurden mithilfe von 50% überlappenden Sliding Windows, welche jeweils 2,56 Sekunden lang sind, 561 Merkmale berechnet. Eine sehr gute Klassifikationsgenauigkeit in der ersten Schicht wurde mit zwei Baum-Welch-Iterationen im Training

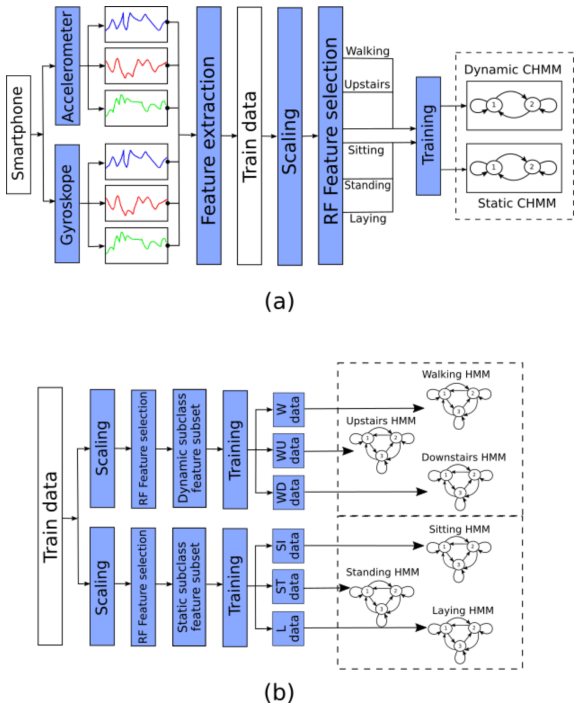


Abbildung 5: Ablauf des Trainingsprozesses eines CHMMs für das grobe (a) und das feine (b) Training (vgl. [RC17])

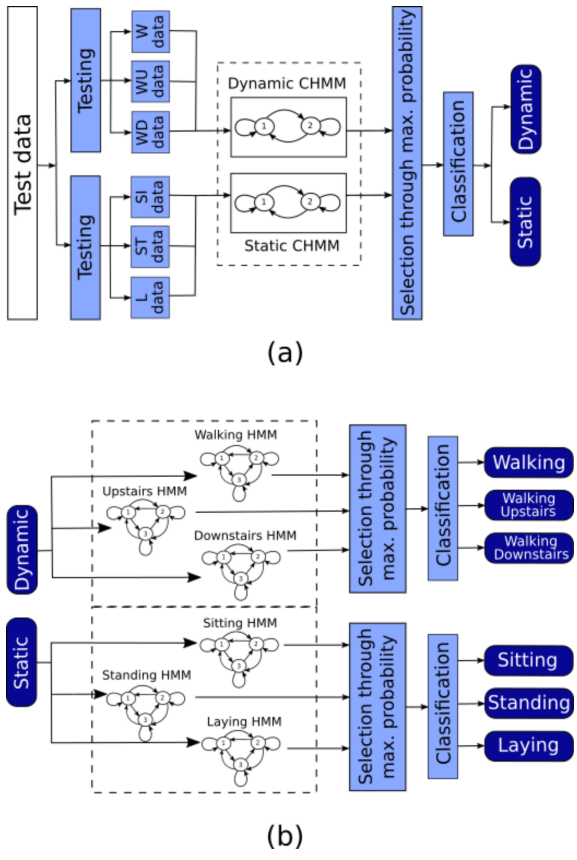


Abbildung 6: Ablauf des Klassifikationsprozesses eines CHMMs für die grobe (a) und die feine (b) Klassifizierung (vgl. [RC17])

erzielt. In der zweiten Schicht wurden bis zu 70 Iterationen durchgeführt. Die Anzahl an Iterationen für die statsichen Trainingsdaten liegt bei 58 mit einer Klassifikationsgenauigkeit von 93,38 %, für die dynamischen Trainingsdaten liegt die Anzahl bei 27 mit einer Genauigkeit von 92,44 %, ohne dass das Modell über- oder unterangepasst ist.

Für die drei CHMMs der dynamischen Aktivitäten wurden zusätzlich zu der Anzahl der Iterationen auch die Anzahl der Zustände verändert. Die besten Ergebnisse für „Gehen“ wurden mit zwei Zuständen und 45 Iterationen (96,98 % Genauigkeit), für „Treppe hochsteigen“ mit drei Zustände und 25 Iterationen (96,18 % Genauigkeit) und für „Treppe runtersteigen“ ebenfalls mit drei Zustände und 25-26 Iterationen (89,05 % Genauigkeit) erzielt. Aufgrund des durchgeführten Experiments wird vermutet, dass die CHMMs für komplexere Aktivitäten mehr Zustände benötigen als die für einfachere Bewegungen. Außerdem scheinen Modelle mit weniger Zuständen mehr Iterationen für das Training zu benötigen als Modelle mit mehr Zuständen.

Als abschließenden Vergleich wurden die Modelle mit den besten Ergebnissen mit anderen Klassifizierern verglichen, darunter nicht-hierarchische CHMMs, künstliche neuronale Netze und der Naive-Bayes-Klassifizierer. Die höchste Gesamt-Genauigkeit erzielte das hierarchische CHMM mit 93,18 %. Die Klassifikationsgenauigkeit von künstlichen neuronalen Netzen betrug ca. 90 %, vom Naive-Bayes-Klassifizierer ca. 85 % und von nicht-hierarchische CHMMs lediglich 65 %. Letzteres Ergebnis lässt nahe liegen, dass die hierarchische Variante des CHMM deutlich besser zur Aktivitätserkennung geeignet ist als die einfache Variante.

7 Zusammenfassung

In dieser Ausarbeitung wurden das Hidden Marcov Model, der Naive-Bayes-Klassifizierer und das Gaussian Mixture Model vorgestellt. Das HMM basiert auf Markov-Ketten und kann somit zeitliche Abhängigkeiten modellieren. Es wird häufig als Klassifizierer in der Aktivitätserkennung eingesetzt und kann zudem zur Nachbereitung von schon klassifizierten Daten genutzt werden.
Der Naive-Bayes-Klassifizierer ist ein sehr einfacher Klassifizierer, der die Klasse mit der höchsten Wahrscheinlichkeit zurückliefert. Trotz seiner Einfachheit liefert er in der Praxis recht gute Ergebnisse. In der Projektgruppe wird er jedoch höchstwahrscheinlich keine Anwendung mehr finden, da er keine zeitlichen Abfolgen (im Falle des Projekts die Reihenfolge der Alltagsbewegungen pro Assessment-Test) berücksichtigt.
Das Gaussian Mixture Model kann zum Clustern von Merkmalsvektor-Ausprägungen mithilfe von Normalverteilungen verwendet werden. Es kann z.B. die Ausgaben für ein Hidden Marcov Model modellieren ,wie im vorherigen Kapitel erläutert.
Da keine Literatur für den Anwendungsfall der Projektgruppe gefunden wurde, kann keine Aussage darüber getroffen werden, ob und wie gut sich HMMs zur Klassifikation der Assessment-Test eignen. Zumindest in der Theorie müssten die Modellierung hierfür folgendermaßen aussehen:
Pro Assessment-Test muss ein HMM modelliert werden und die Ausgabemenge jedes Modells ist die Menge an Alltagsbewegungen. Anhand der bereits klassifzierten Daten können die Modelle mit dem Forward-backward-Algorithmus für ihren jeweiligen Assessment-Test trainiert werden. Wenn nun z.B. die Beobachtung „AAAAGGGGGGGGGHHH“ (A: Aufstehen, G: Gehen, H: Hinsetzten) gemacht wird, wird mit dem Forward-Algorithmus die Wahrscheinlichkeit berechnet, dass die Beobachtung von dem jeweiligen Modell produziert wurde. Anschließend wird das HMM mit der höchsten Wahrscheinlichkeit gewählt. Zu beachten ist hierbei, dass auch unbeschriftete Daten innerhalb eines Assessment-Tests existieren, die

entsprechend behandelt werden müssen. Außerdem müsste noch eine sinnvolle Länge der Beobachtungssequenz bestimmt werden.

Literatur

[AR11] Maria Alvarez Rodriguez. Implementation of gaussian mixture models in .net technology for automatic speech recognition. Master's thesis, University of Science and Technology in Krakow, 2011.

[Bil98] Jeff Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, 1998.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification, 2nd Edition*. Wiley, 2000.

[Ert16] Wolfgang Ertel. *Grundkurs Künstliche Intelligenz – Eine praxisorientierte Einführung, 4. Auflage*. Springer, 2016.

[JM16] Daniel Jurafsky and James H. Martin. Speech and language processing– chapter 9: Hidden marcov models. 2016.

[Koh07] Christian Kohlschein. An introduction to hidden marcov models. Hauptseminarvortrag an der RWTH Aachen, 2007.

[Ras14] Sebastian Raschka. Naive bayes and text classification I - introduction and theory. *CoRR*, abs/1410.5329, 2014.

[RC17] Charissa Ann Ronao and Sung-Bae Cho. Recognizing human activities from smartphone sensors using hierarchical continuous hidden markov models. *International Journal of Distributed Sensor Networks*, 13(1):1550147716683687, 2017.

[Rho03] Thomas Rhodenburg. Klassifikation von audio-signalen. Master's thesis, Universität Bremen, 2003.

[YMJ15] Rabih Younes, Thomas L. Martin, and Mark Jones. Activity classification at a higher level: What to do after the classifier does its best? In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ISWC '15, pages 83–86, New York, NY, USA, 2015. ACM.

[Zwe15] Karlheinz Zwerenz. *Statistik. Einführung in die computergestützte Datenanalyse*. De Gruyter Oldenbourg Wissenschaftsverlag; Berlin, Boston, 2015.

D.3. Künstliche neuronale Netze zur menschlichen Aktivitätserkennung

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Künstliche neuronale Netze zur menschlichen Aktivitätserkennung

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Jonas Prellberg

Oldenburg, den 17. April 2017

Abstract

Ziel dieser Arbeit ist eine Einführung in neuronale Netze im Rahmen der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren”. Es werden Mutlilayer Perceptrons und Recurrent Mutlilayer Perceptrons sowie jeweils Trainingsalgorithmen für diese Netzwerke vorgestellt. Als Alternative zur manuellen Feature-Selektion werden Convolutional Neural Networks eingeführt. Anschließend wird erörtert, ob und wie diese Techniken eingesetzt werden können. Die Nutzung von Multilayer Perceptrons erscheint sinnvoll und kann durch Convolutional Neural Networks ergänzt werden. Problematisch könnte vor allem die recht kleine Menge beschrifteter Trainingsdaten sein.

Inhalt

Abbildungen

Tabellen

1 Einleitung

Im Rahmen der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren" entsteht das Bedürfnis, menschliche Bewegungen wie Gehen, Springen, Sitzen und viele mehr anhand von Daten eines Sensorgürtels zu erkennen. Aktivitätserkennung mit körpernahen Sensoren ist ein viel untersuchtes Gebiet und es gibt zahlreiche Möglichkeiten zur Umsetzung dieser Aufgabe [PGK+09]. Techniken des maschinellen Lernens (ML) sind dabei von Interesse, da sie mit aufgenommenen Daten trainiert werden und auch komplizierte Zusammenhänge erlernen können, ohne dass Domänenexperten zeitaufwändig ein Regelsystem oder spezielle Algorithmen entwickeln müssen. Teilweise erreichen ML-Systeme sogar höhere Klassifikationsgenauigkeiten als mit Expertenwissen aufgebaute Systeme [PEK+06].

Eine ML-Technik, die zur Klassifikation verwendet werden kann, ist das künstliche neuronale Netz (NN). Die Namensgebung und Idee ist an die Struktur des menschlichen Gehirns aus Neuronen und Synapsen angelehnt, allerdings enden die Gemeinsamkeiten hier. Ein NN ist ein gerichteter Graph aus Berechnungsknoten („Neuronen"), welche Funktionen auf die an ihren Kanten („Synapsen") anliegenden Signale anwenden. Die Signale werden mit Kantengewichten multipliziert, welche die freien Parameter des Systems darstellen. Durch einen Trainingsprozess werden passende Gewichte bestimmt, sodass ein funktionaler Zusammenhang zwischen Eingaben und Ausgaben des Netzwerks anhand der präsentierten Trainingsbeispiele gelernt wird. Die Stärke eines erfolgreich trainierten NNs besteht in der Fähigkeit zur Generalisierung: Auch für bisher unbekannte Eingaben wird (hoffentlich) eine angemessene Ausgabe erzeugt.

Ein sehr häufig verwendeter NN-Typ ist das *Multilayer Perceptron*. Es besteht aus vollständig, azyklisch und gerichtet verbundenen Schichten von Berechnungsknoten und kann vielfältig zur Aktivitätserkennung eingesetzt werden [KLLK10, WALC07, PEK+06, GFH09, SBZM05, PFN06, HFLC05, GMM08, MHS01, YWC08]. Wegen ihrer Popularität und weil sie die Grundlage für viele weitere Netzwerktypen bilden, werden sie in diesem

Dokument vorgestellt. Ein weiterer NN-Typ ist das *Recurrent Multilayer Perceptron*, welches beispielsweise in [MT91] zur Gesten-Klassifizierung anhand von Beugungswinkeln der Finger oder in [PPLNS01] zur Erkennung der Alzheimer-Krankheit anhand von Elektroenzephalografie-Aufnahmen genutzt wird. Im Unterschied zum Multilayer Perceptron enthält die Netzwerkarchitektur Zyklen, sodass Zustandsinformationen über die Zeit gespeichert werden können. Recurrent Multilayer Perceptrons werden in diesem Dokument vorgestellt, da sie eine einfach verständliche Erweiterung der Multilayer Perceptrons darstellen und es ermöglichen, dynamische Systeme zu modellieren. *Convolutional Neural Networks*, welche oft mit dem Schlagwort Deep Learning bezeichnet werden, erzielen in der Bilderkennung beachtliche Erfolge [KSH12]. Im Bereich der Aktivitätserkennung findet sich nach Wissen des Autors bisher keine Publikation, die Convolutional Neural Networks einsetzt[1]. Jedoch ist dies ein interessanter Ansatz, da im Gegensatz zu anderen Techniken keine handausgewählten Features (siehe Kapitel 2) benötigt werden. Neben den beschriebenen NNs gibt es noch unzählige weitere, auf die im Rahmen dieser Arbeit aus Platzgründen nicht eingegangen wird.

Der Aufbau dieses Dokuments ist wie folgt: Zunächst werden in Kapitel 2 einige Grundlagen eingeführt, die alle ML-Techniken gemeinsam haben. In Kapitel 3 werden dann das Multilayer Perceptron, seine Nutzbarkeit als Klassifikator und Trainingsalgorithmen vorgestellt. In Kapitel 4 wird aufbauend auf dem Wissen über Multilayer Perceptrons dessen Erweiterung, das Recurrent Multilayer Perceptron, eingeführt und ein Trainingsalgorithmus betrachtet. Es folgt ein kurzer Einblick in Convolutional Networks in Kapitel 5. Schließlich wird in Kapitel 6 die Anwendung der Techniken in der Projektgruppe besprochen. Das Dokument endet mit einer Zusammenfassung in Kapitel 7.

[1] Allerdings nutzt [ZLC+14] Bewegungserkennung als Evaluationsbeispiel für Convolutional Neural Networks.

2 Grundlagen

Die Aufgabe eines Klassifikators besteht darin, einem Eingabevektor $\boldsymbol{x}$ die korrekte Klasse aus c verschiedenen Möglichkeiten zuzuordnen. Durch *überwachtes Lernen* wird er anhand von Beispielen bestehend aus Eingabe und gewünschter, zugehöriger Ausgabe trainiert. Wenn die Beispiele dabei nicht alle gemeinsam vorliegen müssen, sondern nacheinander präsentiert werden können, handelt es sich um *online-learning*.

Üblicherweise enthält der Eingabevektor keine Rohdaten, sondern von ihnen abgeleitete *Features*. Ein Feature kann zum Beispiel eine statistische Größe wie die Varianz oder Autokorrelation sein. Durch Features wird die Klassifikation vereinfacht, indem eine Vorauswahl der wesentlichen Informationen, die in den Rohdaten enthalten sind, vorgenommen wird. Wenn mit Zeitreihen gearbeitet wird, bietet sich weiterhin das sogenannte *Sliding-Window*-Verfahren an. Dabei wird ein kleines Zeitfenster mit einem festen Versatz über die Rohdaten geschoben und jeweils ein Feature-Vektor berechnet. Dadurch reduziert sich die zu verarbeitende Datenmenge und die Klassifikation muss nicht allein anhand einer Momentaufnahme geschehen.

ML-Trainingsverfahren haben kein definiertes Ende, da beliebig viele Beispiele zum Training verwendet werden können. Es gibt algorithmusspezifische Terminierungsbedingungen wie zum Beispiel das Beobachten einer Fehlerfunktion. Sobald deren Änderung pro Trainingsschritt einen Schwellwert unterschreitet, wird der Prozess abgebrochen. Alternativ kann die Fähigkeit zur Generalisierung regelmäßig getestet werden und das Training wird beendet, sobald die gewünschte Klassifikationsgenauigkeit erreicht ist. Dazu werden Validierungsbeispiele benötigt, die nicht bereits zum Training benutzt wurden. Ansonsten kann es zu *overfitting* kommen, das heißt der Klassifikator lernt die Trainingsbeispiele auswendig und erreicht damit auf den Trainingsdaten eine extrem geringe Fehlerrate, verliert allerdings die Fähigkeit zur Generalisierung. Ein beliebtes Verfahren zur Vermeidung von overfitting ist Kreuzvalidierung. Die Trainingsdaten werden dabei in k gleich große Mengen aufgeteilt und alle bis auf eine der Mengen werden zum Training verwendet. Die

ungenutzte Menge wird anschließend zur Validierung genutzt. Dieser Prozess kann k mal mit wechselnden Validierungsmengen wiederholt werden

3 Multilayer Perceptrons

In diesem Kapitel wird zunächst erläutert, wie ein Multilayer Perceptron aufgebaut ist und wie es als Klassifikator verwendet werden kann. Anschließend werden zwei Algorithmen zum Training des Netzwerks vorgestellt. Die Informationen dieses Kapitels stützen sich, sofern nicht anders angegeben, auf [Hay09, FR96].

3.1 Grundlagen

Abbildung 1 zeigt die generelle Struktur eines Multilayer Perceptrons. Es besteht aus mehreren Schichten: einem *input layer*, einem *output layer* und mindestens einem *hidden layer*. Der *input layer* reicht Eingabesignale, die dem Netzwerk zur Verfügung gestellt werden, lediglich weiter, während die anderen Schichten aus Berechnungsknoten bestehen. Alle Berechnungsknoten einer Schicht sind durch gerichtete Kanten mit allen Berechnungsknoten der nächsten Schicht verbunden. Da es keine Zyklen gibt, wird diese Architektur auch als *feed-forward* bezeichnet.

Die Berechnungsknoten werten eine Funktion auf ihren Eingabesignalen aus und stellen das Ergebnis als Ausgabesignal zur Verfügung. Daher lässt sich ein Multilayer Perceptron auch als Netzwerk von Funktionen verstehen. Mit diesem Ansatz können beliebige Eingabe-Ausgabe-Abbildungen beliebig genau approximiert werden—vorausgesetzt es stehen genügend Berechnungsknoten zur Verfügung (Universal Approximation Theorem).

Die Funktion des oder der hidden layer(s) besteht darin, die Eingaben so zu transformieren, dass problemspezifische, gewünschte Eigenschaften vom output layer einfach erkannt werden können. Für eine Klassifikationsaufgabe kann beispielsweise die Darstellung der Datenpunkte in einem anderen Raum die Separierung der Klassen deutlich vereinfachen.

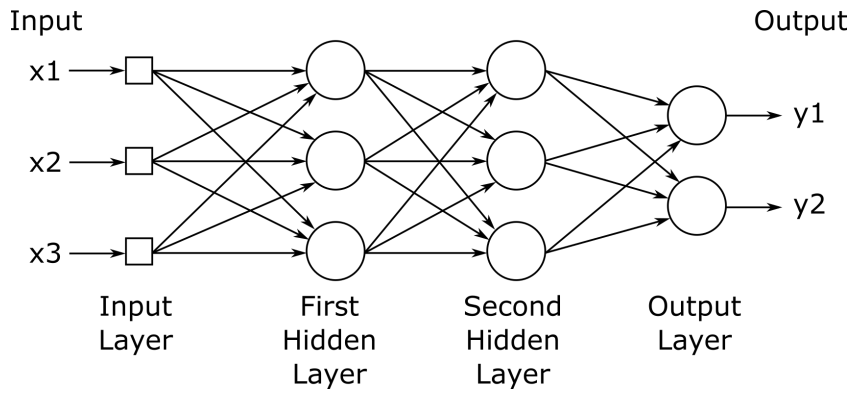


Abbildung 1: Struktur eines Multilayer Perceptrons [Hay09]

Berechnungsknoten

Ein einzelner Berechnungsknoten j mit m eingehenden Signalen ist in Abbildung 2 dargestellt. Die reellen Werte y_1 bis y_m entsprechen den Ausgaben der m Berechnungsknoten der vorherigen Schicht und stellen die Eingabesignale des Berechnungsknotens dar. Zunächst wird über den Eingabesignalen eine gewichtete Summe mit den reellwertigen Gewichten w_{ji} zwischen Knoten i der vorherigen Schicht und Knoten j der aktuellen Schicht gebildet. Auf den aggregierten Wert wird eine Aktivierungsfunktion φ angewendet. Die Rolle der Aktivierungsfunktion wird in Kapitel 3.3 genauer betrachtet und kann bis dahin als beliebige nichtlineare Funktion verstanden werden. Zusammenfassend lässt sich das Verhalten eines einzelnen Berechnungsknotens j als

$$y_j = \varphi\left(v_j\right), \quad v_j = \sum_{i=0}^{m} w_{ji} y_i$$

beschreiben, wobei m die Anzahl der eingehenden Signale ist und v_j induzierter lokaler Körper des Berechnungsknotens j genannt wird.

Zur anschaulichen Darstellung der Funktionsweise eines Berechungsknotens lässt er sich als binärer Klassifikator interpretieren. In der Tat arbeiteten die ersten Berechnungsknoten nach dem Mulloch-Pitts-Modell mit binären Ein- und Ausgaben. Die m Eingabesignale eines Berechnungsknotens bilden einen m-dimensionalen Raum. Durch diesen wird eine Hyperebene gelegt, welche die Klassen voneinander abgrenzt. Ihre Lage ist durch die Kantengewichte und die Aktivierungsfunktion bestimmt. Die binäre Ausgabe wäre davon abhängig, auf welcher Seite der Hyperebene ein Eingabevektor liegt. Bisher unbeachtet blieb das Signal y_0 aus Abbildung 2. Es wird *bias* genannt und verhält sich wie eine weitere Berechnungseinheit der vorgeschalteten Schicht mit konstanter Ausgabe. Erst durch das

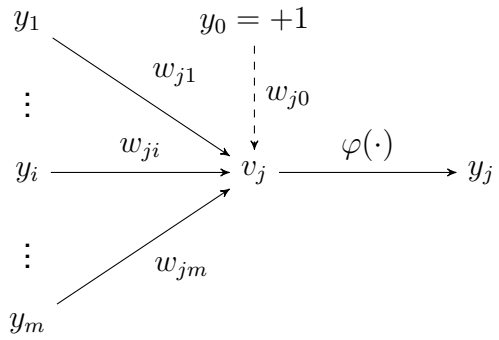


Abbildung 2: Struktur eines Berechnungsknotens. Das optionale Bias ist mit einer gestrichelten Kante gekennzeichnet. Nach [Hay09, bearbeitet].

Bias werden Hyperebenen ermöglicht, die nicht durch den Ursprung des Eingaberaums verlaufen [Sar02]. Für manche Probleme macht diese Einschränkung keinen Unterschied, jedoch ist sie im Allgemeinen hinderlich.

Netzwerkarchitektur

Unabhängig von der Aufgabe des Netzwerks enthält der input layer immer dim $(\boldsymbol{x})$ Eingabeknoten, sodass jedem Knoten ein Element des Eingabevektors $\boldsymbol{x}$ zugeordnet werden kann.

Die Anzahl der hidden layer und auch die Anzahl der Berechnungsknoten dieser Schichten ist schwierig festzulegen. Nach dem Universal Approximation Theorem lassen sich bereits mit einem einzigen hidden layer aus einer unbestimmten Anzahl von Berechnungsknoten beliebige Funktionen approximieren [Cor04, Hay09]. Jedoch bedeutet das keineswegs, dass immer nur eine Schicht verwendet werden sollte, da diese unter Umständen extrem viele Knoten enthalten muss, um eine Funktion mit der gewünschten Genauigkeit zu approximieren. Häufig wird die Anzahl der Berechnungsknoten im hidden layer experimentell festgelegt, aber es gibt auch Ansätze zur Metaoptimierung dieses Problems. Für einen Überblick solcher Techniken sei auf [Cor04] verwiesen.

Um ein Multilayer Perceptron als Klassifikator einzusetzen, gibt es zwei verschiedene Ansätze für den Aufbau des output layers. Der empfohlene und häufig verwendete Ansatz [Cor04, KLLK10, WALC07, PEK+06, MHS01, YWC08] ist es, c Berechnungsknoten in der Ausgabeschicht zu verwenden, sodass jeder Ausgabewert als Indikator für die Zugehörigkeit zu einer der c Klassen dient. Ebenfalls kann für diese Schicht eine besondere

input layer	hidden layer	output layer	freie Parameter	Publikation
33	2×10	5	505	[WALC07]
6	1×15	7	217	[PEK+06]
9	1×3	1	6×34	[GFH09]
24	1×43	4	1251	[MHS01]

Tabelle 1: Anzahl der Knoten pro Netzwerkschicht aus verschiedenen Publikationen und die daraus berechnete Anzahl freier Parameter (Kantengewichte, Bias). [GFH09] nutzt 6 separate Netzwerke zur Erkennung von 6 Klassen.

Aktivierungsfunktion wie *softmax*[1] verwendet werden, um die Ausgabe als Wahrscheinlichkeitsmaß interpretieren zu können [Sar02].

Ein weiterer Ansatz ist es, nur einen einzigen Ausgabeknoten zu verwenden [GFH09, SBZM05]. In diesem Fall müssen allerdings c Netzwerke trainiert werden und für jede Klassifikation alle Netzwerke ausgewertet werden. Im Vergleich zum ersten Ansatz ensteht sowohl beim Training als auch bei der Klassifizierung ein höherer Rechenaufwand, sofern die Netzwerke (abgesehen vom output layer) gleich aufgebaut sind. Unter Umständen lässt sich jedoch eine einfachere Struktur mit weniger Berechnungsknoten verwenden, da die Aufgabe jedes Netzwerks spezifischer ist.

Einige Publikationen zum Thema Aktivitätserkennung mit Multilayer Perceptrons, die die verwendete Netzwerkstruktur angegeben haben, sind in Tabelle 1 aufgelistet und können als Anhaltspunkte verwendet werden.

3.2 Training

Die Netzwerkstruktur und Aktivierungsfunktionen des Netzwerks werden vor dem Training festgelegt. Die freien Parameter des Netzwerks sind die Kantengewichte, welche im Trainingsprozess bestimmt werden sollen. Sei $\boldsymbol{x}$ das Eingabesignal eines Trainingsbeispiels und $\boldsymbol{d}$ das gewünschte, zugehörige Ausgabesignal. Das Ziel des Trainings ist es, die Kantengewichte des Netzes so anzupassen, dass die Ausgabe des Netzwerks $\boldsymbol{y}$ bei Eingabesignal $\boldsymbol{x}$ möglichst nah am gewünschten Ausgabesignal $\boldsymbol{d}$ liegt. Die Idee der Trainingsalgorithmen besteht darin, eine Fehlerfunktion $\mathscr{E}$ in Abhängigkeit der Gewichte

[1] softmax : $\mathbb{R}^n \to \mathbb{R}^n$ bildet $\boldsymbol{x}$ auf $\boldsymbol{y} = (y_1 \ \ldots \ y_n)$ mit $\sum y_i = 1$ und $0 < y_i < 1$ ab.

zu minimieren. Eine übliche Wahl ist

$$\mathscr{E} = \frac{1}{2}\sum_{j\in\mathscr{A}} e_j^2 = \frac{1}{2}\sum_{j\in\mathscr{A}} (d_j - y_j)^2 \tag{3.1}$$

wobei $\mathscr{A}$ alle Indizes der Berechnungsknoten des output layer enthält und e_j den Fehler an Berechnungsknoten j bezeichnet. Die Abhängigkeit der Fehlerfunktion von den Gewichten ist nicht explizit dargestellt.

Die Fehlerfunktion wird für ein oder mehrere Trainingsbeispiele ausgewertet und die Gewichte werden so korrigiert, dass der Fehler sinkt. Es lassen sich die Trainingstechniken *online-learning* und *batch-learning* unterscheiden. Beim online-learning wird ein Trainingsbeispiel zur Zeit betrachtet und daraufhin eine Gewichtskorrektur vorgenommen, während beim batch-learning alle Trainingsbeispiele auf einmal betrachtet werden und dann eine Gewichtskorrektur vorgenommen wird. Der Kompromiss zwischen beiden Techniken nennt sich *mini-batch*. Dabei werden mehrere, aber nicht alle Trainingsbeispiel gemeinsam betrachtet [LZCS14]. In [Hay09] wird online-learning empfohlen, da der Lernprozess mit geringerer Wahrscheinlichkeit in lokalen Minima endet, weniger Arbeitsspeicher während des Trainings benötigt wird und Redundanz in Trainingsbeispielen besser ausgenutzt wird. Daher wird im Folgenden nur dieser Fall betrachtet. Die Fehlerfunktion aus Gleichung 3.1 ist bereits für online-learning geeignet.

Im Folgenden werden die Trainingsalgorithmen Backpropagation (BP) und dessen Verbesserung Resilient Backpropagation (RPROP) vorgestellt. Es gibt weitere Algorithmen wie Levenberg-Marquardt und Optimized Levenberg-Marquardt with Adaptive Momentum, welche beispielsweise in [GFH09, SBZM05] verwendet werden. Durch diese Algorithmen wird die Fehlerfunktion potenziell besser minimiert, jedoch ist die Zeit- und Platzkomplexität der Algorithmen höher als bei BP und RPROP. Dieses Dokument beschränkt sich aus Platzgründen auf die häufiger verwendeten Algorithmen BP und RPROP und verzichtet auf Herleitungen, welche in [Hay09] nachgeschlagen werden können.

3.3 Backpropagation

Backpropagation ist ein Trainingsalgorithmus der nach dem Gradientenabstiegsverfahren arbeitet. Zunächst wird dem Netzwerk das Eingabesignal $\boldsymbol{x}$ eines Trainingsbeispiels präsentiert und das Netzwerk mit den aktuellen Gewichten im sogenannten *forward pass*

ausgewertet. Daraufhin kann der Fehlerterm aus Gleichung 3.1 bestimmt werden. Es soll nun für jedes Gewicht w_{ji} eine Korrektur Δw_{ji} bestimmt werden, die das Gewicht in Richtung des absteigenden Gradienten des Fehlerterms korrigiert. Das geschieht durch die partielle Ableitung des Fehlerterms in Abhängigkeit vom jeweiligen Gewicht:

$$\Delta w_{ji} = -\eta \frac{\partial \mathscr{E}}{\partial w_{ji}} = \eta \delta_j y_i \tag{3.2}$$

Dabei bezeichnet η die (beliebig einstellbare) Lernrate und δ_j den lokalen Gradienten an Berechnungsknoten j. Der Wert des lokalen Gradienten unterscheidet sich je nach Position des Berechnungsknotens im Netzwerk:

$$\delta_j = \begin{cases} \varphi'(v_j)\, e_j & \text{Knoten im output layer} \\ \varphi'(v_j) \sum_k \delta_k w_{kj} & \text{Knoten im hidden layer} \end{cases}$$

Wenn sich der Knoten im output layer befindet, kann der Fehler e_j des Knotens direkt bestimmt werden, da die gewünschte Ausgabe d_j bekannt ist. Befindet sich der Knoten in einem hidden layer, muss eine gewichtete Summe der lokalen Gradienten der Knoten aus der nachfolgenden Schicht gebildet werden. In beiden Fällen ist es notwendig, die Ableitung der Aktivierungsfunktion φ zu bestimmen und auf den induzierten lokale Körper des Knotens anzuwenden.

Bisher wurde es vermieden, konkrete Ausprägungen für Aktivierungsfunktionen anzugeben. Ohne die Aktivierungsfunktion wäre es dem Netzwerk nur möglich, lineare Zusammenhänge abzubilden, da die verwendeten gewichteten Summen nur lineare Funktionen sind. Wählt man jedoch eine nichtlineare Aktivierungsfunktion, können auch nichtlineare Zusammenhänge approximiert werden. Prinzipiell ist jede nichtlineare Funktion (außer Polynomen) geeignet [Sar02, Cor04], doch um Backpropagation anwenden zu können, muss sie differenzierbar sein. Üblichweise werden Sigmoidfunktionen („S-förmige Funktionen") wie die logistische Funktion oder der Tangens Hyperbolicus verwendet. Da der Tangens Hyperbolicus zusätzlich eine ungerade[2] Funktion seines Arguments ist und dies Vorteile bei der Geschwindigkeit des Lernprozesses bringt [LBOM98], wird tanh gegenüber der logistischen Funktion bevorzugt.

Ausgehend vom output layer werden die lokalen Gradienten der Knoten bestimmt und im

[2] Eine Funktion $f(x)$ heißt ungerade genau dann, wenn $f(-x) = -f(x)$.

sogenannten *backward pass* genutzt, um die lokalen Gradienten der direkt davor liegenden Schicht zu bestimmen. Auf diese Weise können rekursiv alle lokalen Gradienten bestimmt werden. Anschließend kann mit Gleichung 3.2 die Korrektur jedes Gewichts bestimmt werden und die Gewichte werden vor dem nächsten Trainingsschritt angepasst.

Ein bisher unbetrachtetes Problem ist die Wahl der Lernrate η, welche vom Benutzer eingestellt werden muss. Wird sie zu klein gewählt, dauert der Lernprozess sehr lange, da sich die Gewichte nur langsam verändern. Wird sie zu groß gewählt, kann der Lernprozess oszillieren, sodass der Fehler nicht sinkt. Einer von vielen Ansätzen ist es, sie während des Lernprozesses langsam zu verringern, sodass das Training zu Anfang beschleunigt wird, aber dennoch konvergiert. Der in Abschnitt 3.4 beschriebene Algorithmus nimmt dem Nutzer die Wahl der Lernrate ab.

3.4 Resilient Backpropagation

Wie im letzten Abschnitt erwähnt, ist das Setzten der Lernrate des Backpropagation-Algorithmus problematisch. Es gibt zahlreiche Heuristiken, welche die Lernrate während des Trainingsprozesses anpassen, jedoch ist die Gewichtskorrektur Δw_{ji} letztendlich immer auch von der Größe des Gradienten $\partial \mathscr{E} / \partial w_{ji}$ abhängig. Der Vorteil einer gut eingestellten Lernrate kann daher durch unvorhersehbar große oder kleine Gradienten zunichte gemacht werden. Der Resilient-Backpropagation-Algorithmus löst dieses Problem, indem die Gewichtskorrektur unabhängig von der Größe des Gradienten durchgeführt wird und stattdessen die zeitliche Veränderung des Gradientenvorzeichens als Basis für die Größe der Korrektur verwendet wird [RB93].

Jedes Gewicht bekommt eine individuelle Korrekturrate Δ_{ji} zugewiesen. Diese wird nach der Regel

$$\Delta_{ji}^{(t)} = \begin{cases} \eta^+ \Delta_{ji}^{(t-1)} & \text{wenn } \frac{\partial \mathscr{E}}{\partial w_{ji}}^{(t-1)} \cdot \frac{\partial \mathscr{E}}{\partial w_{ji}}^{(t)} > 0 \\ \eta^- \Delta_{ji}^{(t-1)} & \text{wenn } \frac{\partial \mathscr{E}}{\partial w_{ji}}^{(t-1)} \cdot \frac{\partial \mathscr{E}}{\partial w_{ji}}^{(t)} < 0 \\ \Delta_{ji}^{(t-1)} & \text{sonst} \end{cases}$$

mit $0 < \eta^- < 1 < \eta^+$ verändert. Ändert der Gradient sein Vorzeichen, ist das ein Hinweis darauf, dass ein lokales Minimum übersprungen wurde. Daraufhin wird die Korrekturrate verringert. Behält der Gradient sein Vorzeichen bei, wird die Korrekturrate erhöht, um den Lernprozess zu beschleunigen. Die resultierende Gewichtskorrektur ist ebenfalls

vom Vorzeichen des Gradienten abhängig: Bei positivem Gradienten bzw. steigender Fehlerrate wird $\Delta w_{ji}^{(t)} = -\Delta_{ji}^{(t)}$ gewählt, sodass sich das Gewicht verringert. Bei negativem Gradienten bzw. sinkender Fehlerrate wird $\Delta w_{ji}^{(t)} = \Delta_{ji}^{(t)}$ gewählt, sodass sich das Gewicht erhöht. Weiterhin wird bei einem Vorzeichenwechsel des Gradienten, das heißt wenn ein Minimum übersprungen wurde, auch das Vorzeichen der Gewichtskorrektur umgekehrt.

Die freien Parameter des Algorithmus sind die Initialwerte $\Delta_{ji}^{(0)}$ für alle Gewichte und η^+, η^-. Nach [RB93] sind unabhängig vom Problem $\Delta_{ji}^{(0)} = 0{,}1$ und $\eta^+ = 1{,}2$ sowie $\eta^- = 0{,}5$ eine gute Wahl. Der Vorteil gegenüber des Backpropagation-Algorithmus ist somit, dass keine Parameter eingestellt werden müssen und der Algorithmus trotzdem schneller konvergiert.

4 Recurrent Multilayer Perceptrons

In diesem Kapitel wird aufbauend auf dem Wissen aus Kapitel 3 die Idee des Recurrent Multilayer Perceptrons eingeführt. Nach der Vorstellung ihrer Struktur wird eine Variation des Backpropagation-Algorithmus zum Training von Recurrent Neural Networks erläutert. Die Informationen dieses Kapitels stützen sich, sofern nicht anders angegeben, auf [Hay09, FR96, JM99].

4.1 Grundlagen

Recurrent Multilayer Perceptrons, welche im Folgenden vorgestellt werden, gehören zur Klasse der Reccurrent Neural Networks. Es sei darauf hingewiesen, dass es weitere Ausprägungen wie NARX und Second-Order Recurrent Networks gibt. Weiterhin lässt sich sowohl mit kontinuierlicher als auch diskreter Zeit arbeiten. In diesem Dokument wird nur letzterer Fall betrachtet.

Recurrent Multilayer Perceptrons erweitern die feed-forward-Architektur der Multilayer Perceptrons um Feedback-Schleifen. Auf diese Weise ist es dem Netzwerk möglich, Zustandsinformationen zu speichern und diese in zukünftigen Berechnungen zu nutzen. Das ermöglicht Eingaben variabler Länge zu verarbeiten, indem diese stückweise als Eingabesignal angelegt werden.

Der Aufbau eines solchen Netzwerks ist in Abbildung 3 dargestellt. Die Ausgaben der Berechnungsknoten jedes hidden layers werden in imaginären Kontextknoten zwischengespeichert und verhalten sich im nächsten Berechnungsschritt wie Ausgaben der vorherigen Schicht. Das heißt die Kontextknoten sind mit allen Berechnungsknoten des hidden layers verbunden, liefern den gespeicherten Wert als Eingabesignal und besitzen zugehörige Kantengewichte, die im Trainingsprozess angepasst werden. Abgesehen von dieser Erweiterung verhält sich das Netzwerk wie ein Multilayer Perceptron.

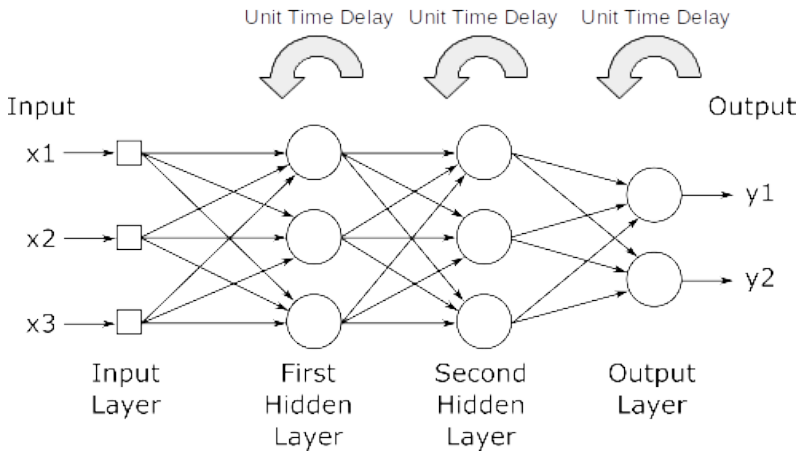


Abbildung 3: Struktur eines Recurrent Multilayer Perceptrons [Hay09]

Recurrent Neural Networks können beliebige dynamische Systeme beliebig genau approximieren—vorausgesetzt es stehen genügend Berechnungsknoten zur Verfügung [Jae02].

4.2 Training

Die grundlegende Idee des Trainings von Recurrent Multilayer Perceptrons ist dieselbe wie für Multilayer Perceptrons: Es werden Gewichte gesucht, die eine Fehlerfunktion auf Trainingsbeispielen minimieren. Allerdings ist es für Recurrent Neural Networks notwendig, die Trainingsbeispiele in einer logischen Reihenfolge zu präsentieren, da die Reihenfolge wegen der Feedback-Schleifen Teil der gelernten Informationen ist.

Auch für Reccurrent Multilayer Perceptrons lässt sich das Training auf zwei Arten durchführen: epochenweise oder kontinuierlich. Die Trainingsarten sind mit den bereits beschriebenen Arten batch- und online-learning verwandt. Epochenweises Training wird verwendet, wenn klar abgrenzbare Eingabesequenzen in den Trainingsdaten existieren, die gelernt werden sollen. Kontinuierliches Training wird verwendet, wenn keine solche Unterteilung der Eingabesequenzen möglich ist.

Im Folgenden wird eine Variante des Backpropagation-Algorithmus vorgestellt. Es sei darauf hingewiesen, dass es weitere Trainingsalgorithmen wie *real-time recurrent learning* oder Ansätze mit Hilfe des *extended Kalman filter*s gibt. Keiner der Algorithmen ist den anderen generell überlegen [Jae02].

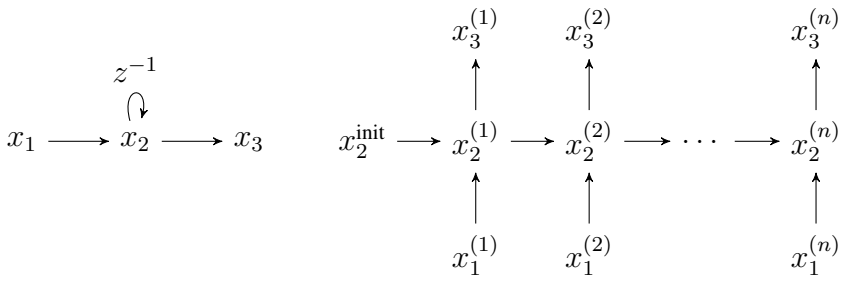


Abbildung 4: Abrollen eines einfachen Recurrent Multilayer Perceptrons (links) in ein äquivalentes Multilayer Perceptron (rechts). z^{-1} ist die Unit-Time-Delay-Operation.

4.3 Backpropagation through time

Der Backpropagation-Algorithmus kann für Recurrent Multilayer Perceptrons nicht direkt eingesetzt werden, weil das Netzwerk Zyklen enthält. Diese Einschränkung lässt sich umgehen, indem das Netzwerk durch „Abrollen der Zeitschritte" in ein feed-forward Multilayer Perceptron transformiert wird. In Abbildung 4 ist dieses Verfahren an einem einfachen Beispiel dargestellt. Wenn das Netzwerk über n Zeitschritte abgerollt wird, entstehen in dem abgerollten Netzwerk n Schichten, die jeweils eine Kopie aller Knoten des ursprünglichen Netzwerks enthalten.

Auf dem abgerollten Netzwerk kann dann eine Variante des Backpropagation-Algorithmus ausgeführt werden. Im Folgenden wird das epochenweise Training beschrieben. Da epochenweises Training mehr Gemeinsamkeiten mit batch-learning als online-learning aufweist, muss (wie auch beim batch-learning) eine andere Fehlerfunktion als in Gleichung 3.1 angegeben verwendet werden. Sei $[n_0, n_1]$ das Zeitintervall der Trainingssequenz. Dann lässt sich

$$\mathscr{E}_{\text{total}} = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j \in \mathscr{A}} e_{j,n}^2$$

mit Indexmenge $\mathscr{A}$ der Knoten, für die eine gewünschte Ausgabe bekannt ist, als Fehlerfunktion verwenden.

Wie schon beim Backpropagation-Algorithmus wird zunächst ein forward pass durchgeführt, bei dem das Netzwerk mit den gegebenen Trainingsbeispielen und aktuellen Gewichten ausgewertet wird. Die induzierten lokalen Körper der Berechnungsknoten und die Ausgaben des Netzwerks werden dabei für jeden Zeitschritt gespeichert. Im anschließenden backward pass werden wieder die lokalen Gradienten berechnet, jedoch auf eine

andere Weise als bisher:

$$\delta_{j,n} = \begin{cases} \varphi'(v_{j,n})\, e_{j,n} & \text{wenn } n = n_1 \\ \varphi'(v_{j,n}) \left(e_{j,n} + \sum_{k \in \mathscr{A}} w_{jk} \delta_{k,n+1} \right) & \text{wenn } n_0 < n < n_1 \end{cases}$$

Es ist zu beachten, dass alle Werte mit dem Zeitpunkt n parametrisiert sind. Die Anzahl der Berechnungsschritte des forward und backwards pass steigt linear mit der Länge der Trainingssequenz, da sie für jeden Zeitschritt durchgeführt werden müssen. Schließlich werden die Gewichtskorrekturen nach

$$\Delta w_{ji} = -\eta \frac{\partial \mathscr{E}_{\text{total}}}{\partial w_{ji}} = \eta \sum_{n=n_0+1}^{n_1} \delta_{j,n} y_{i,n-1}$$

bestimmt, wobei η die Lernrate ist und $y_{i,n-1}$ die Ausgabe des Berechnungsknotens i zum Zeitpunkt $n-1$ ist.

5 Convolutional Neural Networks

Die Idee von *Convolutional Neural Networks* ist es, das Netzwerk auch die Feature-Extraktion übernehmen zu lassen, sodass es nicht mehr notwendig ist, Features manuell zu bestimmen. Die Begriffe *deep learning* und Convolutional Neural Networks umfassen im Vergleich zu Multilayer Perceptrons flexiblere Netzwerkstrukturen aus zusätzlichen Bausteinen, welche im Folgenden beschrieben werden. Die Informationen stammen, sofern nicht anders angegeben, aus [ZLC+14, Nie15].

Die Feature-Extraktion wird mit zwei Konzepten bewerkstelligt: *convolutional layers* und *pooling layers*. Die Ergebnisse der Feature-Extraktion werden dann durch weitere Netzwerkschichten, wie sie beispielsweise aus (Recurrent) Multilayer Perceptrons bekannt sind, weiterverarbeitet.

Convolutional layers funktionieren ähnlich wie hidden layers in Multilayer Perceptrons. Der entscheidende Unterschied ist, dass sie nicht vollständig mit der vorherigen Schicht verbunden sind, sondern jeder Berechnungsknoten nur mit Eingaben aus einem kleinen, lokal begrenzten Bereich versorgt wird. Dies ist in Abbildung 5 dargestellt und sehr ähnlich zum Sliding-Window-Verfahren. Weiterhin teilen sich alle Knoten des convolution layers die Kantengewichte und das Bias. Das heißt in Abbildung 5 gibt es nur 3 Kantengewichte und 1 Bias, obwohl mehr Kanten und Knoten vorhanden sind. Die Aktivierungsfunktion

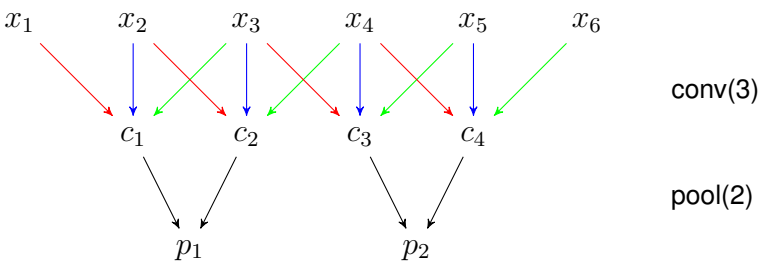


Abbildung 5: Funktionsweisen eines convolutional layers (kernel size 3, stride 1) und eines pooling layers (subsampling-factor 2)

dieser Schichten kann wie für hidden layers gewählt werden, allerdings ist ReLU[1] in Convolutional Neural Networks sehr erfolgreich [ZLC+14].

Die geteilten Parameter vereinfachen den Trainingsprozess, aber führen auch dazu, dass jeder Knoten des convolutional layers dasselbe Feature erkennt. Daher werden mehrere parallel angeordnete convolutional layers benötigt, um verschiedene Features zu erkennen.

Auf einen convolutional layer folgt üblicherweise ein pooling layer, wie in Abbildung 5 dargestellt. Dieser reduziert die Anzahl der Knoten, indem Ausgaben des convolutional layers gruppenweise zusammengefasst werden. Dies geschieht zum Beispiel durch das Bilden des Maximums. Durch diesen Schritt wird das Netzwerk weniger anfällig für verrauschte Daten und außerdem sinkt die Menge der in nachfolgenden Schritten zu verarbeitenden Daten.

Diese beiden Schichttypen können mehrfach hintereinander angewandt werden, um zunächst grundlegende und später immer speziellere Features zu extrahieren. Anschließend können die Ergebnisse zum Beispiel durch hidden und output layer, wie sie aus Multilayer Perceptrons bekannt sind, weiterverarbeitet werden. Ein größeres Beispiel-Netzwerk ist in Abbildung 6 dargestellt. Das Training eines solchen Netzwerks kann mit Varianten von Backpropagation geschehen. Es gilt zu beachten, dass RPROP zum Training von Convolutional Networks, anders als für (Recurrent) Multilayer Perceptrons, dem BP-Algorithmus nicht ohne Anpassung überlegen ist [MM15].

Obwohl die Anzahl der freien Parameter nur geringfügig steigt, erhöht die Feature-Extraktion durch convolutional layers die Trainingszeit signifikant, da viele zusätzliche Rechenoperationen pro Trainingsschritt durchgeführt werden müssen.

[1] Rectified Linear Unit, $\varphi(x) = \max(0, x)$

6 Einsatz in der Projektgruppe

Im Rahmen der Projektgruppe liegen große Datenmengen in Form von Sensoraufnahmen vor. Um von diesen Daten überwacht zu lernen, müssen sie manuell beschriftet werden. Nach dem aktuellen Stand sollen 16 Klassen unterschieden werden, die sich weiter in dynamische (Gehen, Springen, ...), statische (Sitzen, Liegen, ...) und Übergangsaktivitäten (Hinsetzen, Hinlegen, ...) unterteilen lassen. Der Ansatz einer hierarchischen Klassifikation wie in [KLLK10, YWC08] erscheint auf Grund der Unterschiede in diesen Bewegungen, den exzellenten Ergebnissen dieser Publikationen und Ähnlichkeiten zu der Aufgabe der Projektgruppe sinnvoll. Dabei wird zunächst ein Klassifikator trainiert, der nur entscheidet, ob eine vorliegende Aktivität dynamisch, statisch oder im Übergang ist. Weiterhin werden 3 Klassifikatoren für diese 3 Klassen trainiert, die entsprechend der Entscheidung des ersten Klassifikators angewendet werden, um die genaue Bewegung zu bestimmen. In Convolutional Networks ist dieses Vorgehen wahrscheinlich unnötig, da das Netzwerk durch mehrere convolutional layers selbst hierarchische Features extrahieren kann.

6.1 Feature-Extraktion

Unabhängig davon, ob convolutional layers genutzt werden, sollte wie in Kapitel 2 beschrieben ein Sliding Window eingesetzt werden, um die Zeitreihen zu verarbeiten. Die Länge des Zeitfensters und der Grad der Überlappung sind Parameter, deren optimale Werte nur durch Experimente bestimmt werden können. Als Anhaltspunkte lassen sich 3,2 Sekunden ohne Überlappung und 5,12 Sekunden mit 50 % Überlappung aus [KLLK10, YWC08] nennen.

Werden keine convolutional layers verwendet, müssen die Features manuell festgelegt werden. Problematisch dabei ist, dass die Wahl der Features stark anwendungsabhängig und von Publikation zu Publikation verschieden ist. Daher ist es notwendig, unterschiedliche

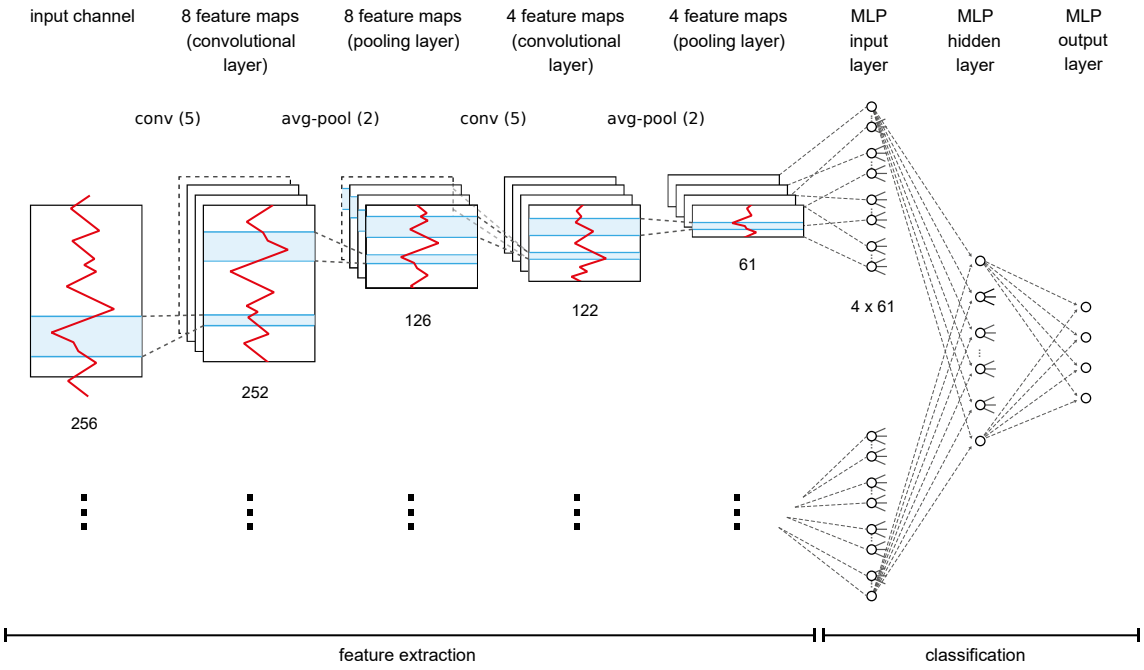


Abbildung 6: Convolutional Network zur Klassifikation multivariater Zeitreihen. Netzwerkarchitektur aus [ZLC+14].

Features zu testen und zu vergleichen.

Mit dem Einsatz von convolutional layers entfällt die Featureauswahl. Eine mögliche Netzwerk-Architektur zur Klassifizierung von Aktivitäten ist in Abbildung 6 dargestellt.

Die Netzwerkeingabe ist ein 256-elementiger Vektor pro Sensordatenreihe, welcher beispielsweise durch ein Sliding-Window-Verfahren mit 2,56 s Fenstergröße bei 100 Hz erzeugt werden könnte. Die Feature-Extraktion wird für jede Datenreihe unabhängig ausgeführt. Es werden 8 convolutional layers mit kernel size 5 verwendet, sodass 8 Schichten mit je 252 Knoten entstehen. Da alle Gewichte und Biase geteilt werden, müssen trotzdem nur 5 Gewichte und 1 Bias pro convolutional layer gelernt werden. Bei 8 Schichten entstehen also lediglich 48 weitere freie Parameter. Es folgen 4 weitere convolutional und pooling layers, sodass letztendlich 61 Knoten pro convolutional layer und Sensordatenreihe verbleiben. Diese dienen dann als Eingabe eines Multilayer Perceptrons, welches die Klassifikation vornimmt.

Diese Architektur wird in [ZLC+14] mit einem Multilayer Perceptron und 1-nearest-neighbor mit dynamic time warping als Abstandsmaß unter verschiedenen Sliding-Window-Fenstergrößen verglichen. Sie schlägt die Konkurrenz in allen Aspekten.

Es erscheint daher sinnvoll, auch mit Convolutional Neural Networks zu experimentieren.

6.2 Netzwerk

Sowohl Multilayer Perceptrons als auch Recurrent Multilayer Perceptrons ermöglichen es, mehr als zwei Klassen zu unterscheiden: durch ein *multiclass* Netz mit c Ausgabeknoten oder c klassenspezifische Netze mit einem Ausgabeknoten.

Für Multilayer Perceptrons sind multiclass-Netze entsprechend den Empfehlungen und Erfahrungen der Literatur eine gute Wahl, jedoch können auch klassenspezifische Netze nützlich sein: Die Trainingsdaten der Projektgruppe sind sehr ungleichmäßig über die Klassen verteilt, aber zum Training eines Netzwerks sind Beispiele der Klassen in ähnlichem Verhältnis vorteilhaft. Also müssten die Daten aller Klassen sehr stark durch *sampling* (siehe Abschnitt 6.3) manipuliert werden, um die Verhältnisse anzugleichen. Wenn aber klassenspezifische Netzwerke trainiert werden, kann für jedes Netzwerk in Abhängigkeit der vorliegenden Verteilung entschieden werden, wie stark sampling eingesetzt wird.

Für Recurrent Multilayer Perceptrons kann ein Argument für c Netzwerke mit einem Ausgabeknoten gegeben werden: Zur Unterscheidung der verschiedenen temporalen Muster wären in einem einzigen Netz wahrscheinlich sehr viele Berechnungsknoten notwendig, was das Training verlangsamt und die Anzahl der notwendigen Trainingsbeispiele erhöht.

Die Anzahl der hidden layer und auch die Anzahl der darin enthaltenen Knoten muss experimentell bestimmt werden. Nach Tabelle 1 kann aber davon ausgegangen werden, dass höchstens 2 hidden layer notwendig sein werden.

Die Klassifikation ist mit beiden Netzwerktypen sehr schnell, da alle Operationen als Matrixmultiplikationen ausgedrückt werden können. Deutlich zeitaufwendiger ist das Training. Für Multilayer Perceptrons hängt die benötigte Trainingszeit von den Konvergenzeigenschaften des genutzten Trainingsalgorithmus ab. Mit Resilient Backpropagation steht ein erprobter online-Algorithmus zur Verfügung, jedoch ist die Konvergenz auch stark von der Problemstellung, der Netzwerkstruktur und den gewählten Initialbedingungen abhängig und kann nicht garantiert werden. In jedem Fall ist der Trainingsprozess aber schneller als mit Backpropagation-through-time für Recurrent Multilayer Perceptrons, da der Zeit- und Speicheraufwand hier linear mit der Länge des betrachteten Zeitintervalls wächst und das Training aus dem Algorithmus inhärenten Gründen nicht mehr online geschehen kann. Weiterhin ist die Anzahl der freien Parameter bei gleicher Knotenanzahl wegen der Feedback-Schleifen doppelt so hoch, wodurch mehr Trainingsdaten notwendig sind.

In der Literatur werden Multilayer Perceptrons deutlich häufiger eingesetzt als Recurrent Multilayer Perceptrons. Sie sind einfacher zu nutzen, schneller zu trainieren und benötigen weniger Trainingsdaten. Daher sollte eine Wahl zwischen den beiden Verfahren zunächst auf Multilayer Perceptrons fallen.

6.3 Training

Der Trainingserfolg von neuronalen Netzen wird durch viele Faktoren beeinflusst. Im Folgenden werden Hinweise gegeben, um möglichst effektiv zu trainieren.

Normalisierung der Eingabevektoren Die Eingaben sollten verschoben werden, sodass ihr Mittelwert etwa bei null liegt und sie sollten normalisiert werden [Hay09, Sar02]. Im Rahmen der Projektgruppe kann die Normalisierung entweder personenbasiert oder global auf allen Daten durchgeführt werden. In [WALC07] wird ein personenbasierter Ansatz genutzt, der für jede Person Features aus 5 Sekunden Gehaktivität bestimmt und mit diesen die restlichen Daten der Person normalisiert. Alternativ können die Daten sowohl global als auch personenbasiert mit einem konstanten, im Voraus bestimmten Faktor skaliert werden, sodass sie danach im gewünschten Bereich liegen. Die Unterschiede beider Methoden sollten evaluiert werden.

Dekorrelation der Eingabevektoren Insbesondere wenn keine convolutional layers genutzt werden, sollten die Elemente der Eingabe unkorreliert sein. Das kann durch die Anwendung von Dimensionsreduktionsmethoden wie Hauptkomponentenanalyse oder Linear discriminant analysis erreicht werden. Beispiele für solche Vorverarbeitungen finden sich in [GFH09, KLLK10, WALC07, ZLC+14].

Normalisierung der Trainings-Ausgabevektoren Die gewünschten Ausgaben der Trainingsbeispiele sollten vorverarbeitet werden, sodass ihr Wertebereich deutlich innerhalb der Grenzwerte der gewählten Aktivierungsfunktion liegt. Durch die Sigmoidform der Aktivierungsfunktion ist ihre Ableitung nahe der Grenzwerte sehr klein. Um den Trainingsprozess nicht unnötig durch kleine Gradienten zu verlangsamen, sollte es vermieden werden, Berechnungsknoten in diesem Bereich arbeiten zu lassen. Für $\varphi(x) = a \tanh(bx)$ mit $a = 1{,}7159$ und $b = 0{,}\bar{6}$ ist $(-1, 1)$ ein vernünftiger Zielbereich [Hay09].

Initialisierung der Gewichte Die Wahl der Initialgewichte ist von der Aktivierungsfunk-

tion abhängig. Für tanh mit den eben beschrieben Parametern sollten die initialen Gewichte aus einer stetigen Gleichverteilung mit Erwartungswert null und Varianz $1/m$ gezogen werden, wobei m die Anzahl der Verbindungen eines Berechnungsknotens ist [Hay09].

Informationsgehalt der Trainingsbeispiele maximieren Für einen effektiven Lernprozess sollte der Informationsgehalt jedes präsentierten Trainingsbeispiels möglichst hoch sein. Das kann mit Beispielen erreicht werden, die einen möglichst hohen Fehler hervorrufen oder sehr unterschiedlich zu bisher gelernten Beispielen sind. In der Praxis lässt sich das für Klassifikationsprobleme erreichen, indem die Reihenfolge der Trainingsbeispiele zufällig gewählt wird. Auf diese Weise wird selten mehrfach hintereinander ein Beispiel derselben Klasse trainiert. Die zufällige Ordnung der Trainingsbeispiele wird unter anderem in [KLLK10] verwendet.

Regularisierung Regularisierung ist eine Technik zur Vermeidung von overfitting. Regularisierung kann unter anderem durch eine neue Fehlerfunktion umgesetzt werden, welche die Gewichte mit einbezieht und große Gewichtswerte bestraft. Dadurch verändert sich die Netzwerkausgabe bei kleinen Änderungen in der Eingabe weniger [Nie15]. Eine weitere erwähnenswerte Technik ist *dropout* [KSH12], welche während des Trainingsprozesses zufällig die Ausgaben einiger Knoten auf null setzt.

Ergebnisvalidierung Im Rahmen der Aktivitätserkennung ist zu beachten, dass die aufgenommenen Daten üblicherweise von unterschiedlichen Probanden stammen. Bei einer Kreuzvalidierung könnten die Probanden also entweder in k Mengen aufgeteilt und die zugehörigen Daten zum Training verwendet werden oder die Daten aller Probanden werden zufällig durchmischt und dann in k Mengen aufgeteilt. Letzteres Verfahren führt zu besseren Ergebnissen, da beim Training Daten einer größeren Anzahl verschiedener Personen zur Verfügung stehen und sich das Netzwerk nicht zu sehr an die speziellen Bewegungsmuster einzelner Probanden anpasst [WALC07].

Doch selbst wenn alle diese Hinweise befolgt werden, kann es schwierig sein, gute Trainingsergebnisse zu erhalten. Das größte Problem der Anwendung neuronaler Netze in der Projektgruppe dürfte die sehr kleine Menge zur Verfügung stehender *beschrifteter* Trainingsdaten sein. Im Folgenden wird anhand von Schätzungen bestimmt, wie viele Daten benötigt werden. Etwaige convolution layer werden dabei nicht berücksichtigt.

Bei einer hierarchischen Klassifikation werden zur konkreten Bewegungsklassifikation

3 Netzwerke mit jeweils 4-7 Ausgabeknoten trainiert. Nach Tabelle 1 sollte die Anzahl der freien Parameter eines solchen Netzwerks in der Größenordnung von 1000 liegen. Es gibt verschieden Aussagen über den Zusammenhang zwischen freien Parametern und Trainingsdatenmenge, jedoch hängt dieser auch immer sehr stark vom Problem und der Repräsentativität der Trainingsbeispiele ab. Nach [Hay09] werden für eine Ziel-Klassifikationsgenauigkeit von 90 % etwa 10 mal mehr Beispiele als Parameter benötigt—also 10 000 Trainingsbeispiele. Mit einer Fenstergröße von 3 Sekunden ohne Überlappung entspricht das 8,3 Stunden beschrifteter Daten aus der groben Klasse (dynamisch, statisch, Übergang). Andere Quellen [Sar02] geben an, dass 30 mal mehr Beispiele als Parameter notwendig sind—also 30 000 Trainingsbeispiele. Das entspricht 25 Stunden beschrifteten Daten.

Weiterhin sind die verschiedenen Klassen in den vorliegenden Daten unterschiedlich stark repräsentiert: Beispielsweise tritt Springen nur einige Male für wenige Sekunden pro Aufnahme (etwa 1 Stunde) auf. Allerdings ist es für den Trainingsprozess vorteilhaft, Positiv- und Negativbeispiele in einem gleichmäßigen Verhältnis zu präsentieren. Mögliche Techniken mit dieser Problematik umzugehen sind *undersampling*, *oversampling* und *synthetic oversampling*. Beim undersampling wird nur ein kleiner Teil der Beispiele aus überrepräsentierten Klassen verwendet. Problematisch dabei ist, dass auch sehr gute, repräsentative Beispiele zufällig entfernt werden könnten. Beim oversampling werden Beispiele aus unterrepräsentierten Klassen mehrfach verwendet. Das kann allerdings zu overfitting führen. Synthetic oversampling erzeugt neue Trainingsbeispiele für die unterrepräsentierten Klassen, indem ihre Beispiele auf verschiedene Weisen kombiniert werden. Synthetic oversampling erzielt bessere Ergebnisse als die anderen beiden Ansätze [Cha09].

7 Zusammenfassung

In diesem Dokument werden zunächst Multilayer Perceptrons eingeführt, welche in vielen Publikationen zum Thema Aktivitätserkennung mit Erfolg genutzt werden. Sie bestehen aus einem Netzwerk von Berechnungsknoten und können durch einen Trainingsprozess funktionale Zusammenhänge erlernen. Der Trainingsprozess basiert auf einem Gradientenabstiegsverfahren über einer Fehlerfunktion, welche von Kantengewichten des Netzwerks abhängt. Diese Kantengewichte werden durch das Training angepasst, um das gewünschte Verhalten des Netzwerks zu erreichen. Von den zwei vorgestellten Trainingsalgorithmen ist RPROP der bessere.

Anschließend werden Recurrent Multilayer Perceptrons vorgestellt. Sie erweitern das Modell der Multilayer Perceptrons um Feedback-Schleifen und erhalten damit eine Art Gedächtnis. Auf diese Weise können dynamische Systeme modelliert werden, aber es müssen mehr freie Parameter bestimmt werden. Der Aufwand des erläuterten Backpropagation-through-time-Algorithmus wächst linear mit der Länge der gelernten Sequenzen. Ebenfalls ist die Anzahl wissenschaftlicher Publikationen zur Aktivitätserkennung mit Recurrent Multilayer Perceptrons stark begrenzt.

Convolutional Neural Networks bieten die Möglichkeit, die Feature-Extraktion vom Netzwerk übernehmen zu lassen. Dieser Ansatz erzielt in der Bilderkennung große Erfolge, aber ist im Rahmen der Aktivitätserkennung bisher kaum untersucht. Nachteilig ist hauptsächlich der erhöhte Rechenaufwand während des Trainings.

Schließlich wird der Einsatz neuronaler Netze in der Projektgruppe besprochen. Die Feature-Extraktion lässt sich manuell oder mit convolutional layers durchführen. Beides hat seine eigenen Nachteile: In ersterem Fall ist Domänenwissen erforderlich, um die richtigen Features für die Aktivitätserkennung zu selektieren. In letzterem Fall wird die Netzwerkarchitektur komplexer und der Trainingsprozess dauert länger. In beiden Fällen bietet es sich an, Multilayer Perceptrons als Klassifikator zu verwenden. Die gegebenen

Trainingshinweise sollten befolgt werden, um die Chance auf gute Ergebnisse zu erhöhen. Die größten Hindernisse dürften die Menge der beschrifteten Trainingsbeispiele und die stark unterschiedlichen Mengenverhältnisse zwischen den Klassen sein. Letzteres könnte durch synthetic oversampling eingedämmt werden.

Zusammenfassend bieten neuronale Netze ein mächtiges Werkzeug zur Aktivitäts-Klassifikation, das in vielen Publikationen hohe Genauigkeit bewiesen hat. Allerdings ist der Unterschied zu anderen ML-Techniken üblicherweise gering und stark anwendungsabhängig [PGK+09], sodass in der Projektgruppe mit verschiedene Techniken experimentiert werden sollte. Convolutional Neural Networks sind im Rahmen der Aktivitätserkennung bisher kaum untersucht, aber erscheinen sehr vielversprechend.

Literatur

[Cha09] Nitesh V. Chawla. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer, 2009.

[Cor04] Paulo Cortez. MLP application guidelines. Invited lecture at NN2003/NN2004 Summer School of Neural Networks in Classification, Regression and Data Mining, ISEP, Oporto, Portugal, 2004.

[FR96] J. Feldman and R. Rojas. *Neural Networks: A Systematic Introduction*. Springer Berlin Heidelberg, 1996.

[GFH09] Norbert Győrbíró, Ákos Fábián, and Gergely Hományi. An activity recognition system for mobile phones. *Mobile Networks and Applications*, 14(1):82–91, 2009.

[GMM08] Daniele Giansanti, Velio Macellari, and Giovanni Maccioni. New neural network classifier of fall-risk based on the mahalanobis distance and kinematic parameters assessed by a wearable device. *Physiological Measurement*, 29(3):N11, 2008.

[Hay09] S.S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 2009.

[HFLC05] Michael E. Hahn, Arthur M. Farley, Victor Lin, and Li-Shan Chou. Neural network estimation of balance control during locomotion. *Journal of Biomechanics*, 38(4):717–724, 2005.

[Jae02] Herbert Jaeger. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. Technical report, German National Research Center for Information Technology, 2002.

[JM99] L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1999.

[KLLK10] A. M. Khan, Y. K. Lee, S. Y. Lee, and T. S. Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1166–1172, Sept 2010.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classifi-

cation with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[LBOM98] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Effiicient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK, 1998. Springer-Verlag.

[LZCS14] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 661–670, New York, NY, USA, 2014. ACM.

[MHS01] Jani Mäntyjärvi, Johan Himberg, and Tapio Seppänen. Recognizing human motion with multiple acceleration sensors. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*. Institute of Electrical & Electronics Engineers (IEEE), 2001.

[MM15] Alan Mosca and George D. Magoulas. Adapting resilient propagation for deep learning. *CoRR*, abs/1509.04612, 2015.

[MT91] Kouichi Murakami and Hitomi Taguchi. Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 237–242, New York, NY, USA, 1991. ACM.

[Nie15] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[PEK+06] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen. Activity classification using realistic data from wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):119–128, Jan 2006.

[PFN06] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. *Feature Selection and Activity Recognition from Wearable Sensors*, pages 516–527. Springer, Berlin, Heidelberg, 2006.

[PGK+09] Stephen J. Preece, John Y. Goulermas, Laurence P. J. Kenney, Dave Howard, Kenneth Meijer, and Robin Crompton. Activity identification using body-mounted sensors—a review of classification techniques. *Physiological Measurement*, 30(4):R1, 2009.

[PPLNS01] A. A. Petrosian, D. V. Prokhorov, W. Lajara-Nanson, and R. B. Schiffer. Recurrent neural network-based approach for early recognition of Alzheimer's disease in EEG. *Clinical Neurophysiology*, 112(8):1378–1387, 2001.

[RB93] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

[Sar02] Warren S. Sarle. Neural network FAQ. Periodic posting to the Usenet newsgroup comp.ai.neural-nets, 2002.

[SBZM05] E. S. Sazonov, T. Bumpus, S. Zeigler, and S. Marocco. Classification of plantar pressure and heel acceleration patterns using neural networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 5, pages 3007–3010 vol. 5, July 2005.

[WALC07] N. Wang, E. Ambikairajah, N. H. Lovell, and B. G. Celler. Accelerometry based classification of walking patterns using time-frequency analysis. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4899–4902, Aug 2007.

[YWC08] Jhun-Ying Yang, Jeen-Shing Wang, and Yen-Ping Chen. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern Recogn. Lett.*, 29(16):2213–2220, December 2008.

[ZLC+14] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. *Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks*, pages 298–310. Springer International Publishing, Cham, 2014.

D.4. Support Vector Machine und Adaptive Multi-Hyperplane Machine

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Support Vector Machine und Adaptive Multi-Hyperplane Machine

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren

SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Thies de Graaff

Oldenburg, den 7. April 2017

Abstract

Support Vector Machines sind eine verbreitete Methode im Maschinellen Lernen. Durch die Transformation von Daten in einen hochdimensionalen Raum kann eine lineare Trennbarkeit der Daten erreicht werden, sodass diese durch eine Hyperebene getrennt werden können. Diese Hyperebene dient dann als Klassifikator um neue Datenpunkte einer der Klassen zuordnen zu können.

Dieses Prinzip wurde auf viele Arten erweitert, sodass diese Ausarbeitung einen Einblick in solche Erweiterungen geben wird. Betrachtet Beispiele sind die Support Vector Data Description und die Adaptive Multi-Hyperplane Machines.

Im Anschluss wird reflektiert, wie Support Vector Machines in der Literatur bereits eingesetzt werden, um verschiedene Aktivitäten zu klassifizieren.

Davon ausgehend wird die Integration in das Projekt „Medizinische Assessments mit körpernahen Sensoren“ evaluiert.

Inhalt

Abkürzungen

AMM	Adaptive Multi-Hyperplane Machine
MAmkS	Medizinische Assessments mit körpernahen Sensoren
MM	Multi-Hyperplane Machine
OAA	One Against All
OAO	One Against One
RBF	Radial Basis Function
SGD	Stochastic Gradient Descent
SVDD	Support Vector Data Description
SVM	Support Vector Machine

Abbildungen

Tabellen

1 Einleitung

Es gibt viele Anwendungsfälle, in denen man eine bestimmte Menge an Merkmalen hat, anhand derer eine Klassifikation vorgenommen werden soll. Ein Beispiel wäre die Bestimmung einer Fischart mittels verschiedener physiologischer Ausprägungen, wie z.B. die Farbe, die Länge, das Gewicht, etc. Diese Merkmale kann man als einen Vektor betrachten, dem dann eine Fischart zugewiesen ist.

Um eine solche Klassifikation vornehmen zu können, sind Verfahren notwendig, die ausgehend von einem Trainingsdatensatz ein Modell erlernen können, mit dem sich zu einem neuen Vektor von Merkmalen eine Vorhersage treffen lässt, welche Klasse die wahrscheinlichste Lösung ist. Solche Verfahren fasst man unter dem Begriff *Maschinelles Lernen* zusammen.

Es wurden bereits viele verschiedene solcher Verfahren erforscht, eines davon ist die sogenannte Support Vector Machine (SVM). SVMs sind in der Mitte der Neunziger Jahre von Vapnik eingeführt und seitdem stark erweitert worden [Vap95]. Es gibt viele verschiedene Derivate des ursprünglichen Ansatzes, wodurch SVMs sehr vielseitig eingesetzt werden können. Eine Erweiterung des Verfahrens ist die Adaptive Multi-Hyperplane Machine (AMM).

In dieser Ausarbeitung wird das generelle Verfahren von SVMs erläutert und einige wichtige Derivate vorgestellt. Darüber hinaus wird ein tieferer Einblick in die AMMs gegeben. Im Anschluss wird vorgestellt, wie beide Verfahren bereits in der Literatur zur Aktivitätserkennung eingesetzt wurden. Ausgehend davon wird evaluiert, wie sich die Verfahren in das Projekt „Medizinische Assessments mit körpernahen Sensoren (MAmkS)“ integrieren lassen und welches am Besten geeignet ist.

2 Grundlagen

Bevor das Prinzip von SVMs erläutert werden kann, müssen zunächst einige grundlegende Definitionen gegeben werden, die zentrale Bestandteile des Verfahrens sind.

Ein Hilbertraum $\mathcal{H}$ ist ein Vektorraum mit einem Skalarprodukt. Dazu gibt es eine Featuremap Φ, die Elemente aus einer nichtleeren Menge X in diesen Hilbertraum abbildet:

$$\Phi : X \to \mathcal{H}$$

Die Bildpunkte werden Features genannt und sollen $x \in X$ möglichst gut charakterisieren, sodass mehr Struktur zur Beschreibung der Daten als in X entsteht [Bre15, S. 54]. Zu dieser Featuremap Φ existiert ein sogenannter Kern $k : X \times X \to \mathbb{R}$ mit

$$k(x, z) := \langle \Phi(x), \Phi(z) \rangle_{\mathcal{H}} .$$

Mit Hilfe eines Kerns lässt sich also das Skalarprodukt in dem zugehörigen Hilbertraum berechnen.

3 Support Vector Machine

3.1 Grundidee

Der Grundansatz für SVMs wurde durch den Generalized Portrait Algorithm von Vapnik & Lerner gegeben [VL63]. Dieser trennt eine widerspruchsfreie Trainingsmenge durch eine Hyperebene. In der Literatur wird dieses Verfahren oft auch als „Lineare SVM“ bezeichnet.

Sei $X \subset \mathbb{R}^d$ der Datenraum, $Y = \{-1, 1\}$ und $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ der Trainingsdatensatz, sodass eine Vektor $w \in \mathbb{R}^d$ mit $\|w\|_2 = 1$ und eine reelle Zahl $b \in \mathbb{R}$ existieren, sodass gilt

$$\begin{aligned} \langle w, x_i \rangle + b > 0 \quad & \forall i \text{ mit } y_i = +1 \\ \langle w, x_i \rangle + b < 0 \quad & \forall i \text{ mit } y_i = -1. \end{aligned}$$

$\langle w, x_i \rangle + b$ definiert nun eine Hyperebene, die die Trainingsdaten optimal trennt. Hier zeigt sich aber auch gleich die große Schwäche des Verfahrens, denn es wird vorausgesetzt, dass die Daten linear trennbar sind. Dies ist aber für echte Anwendungen oftmals nicht der Fall, sodass sich hier der Generalized Portrait Algorithm als ungeeignet erweist. Dies gilt auch für die Daten des Projektes MAmkS. Außerdem wäre es sinnvoll, dass man im Falle von verrauschten Daten eine Fehlerkorrektur anwenden könnte. Der hier vorgestellte Algorithmus bietet auch hierfür keine Möglichkeit.

Aufgrund dieser Probleme wurde das Verfahren erweitert und stellt in seiner Gesamtheit die heute bekannte SVM oder auch „Kernel SVM“ dar[Vap95]. Hierfür bildet man die Trainingsdaten mittels einer Featuremap in einen höherdimensionalen Hilbertraum $\mathcal{H}_0$ ab. Durch diese Abbildung der ursprünglichen, nicht-linear trennbaren Trainingsdaten in den höherdimensionalen Raum lässt sich eine lineare Trennbarkeit erreichen, um anschließend den Generalized Portrait Algorithm in diesem Featureraum anzuwenden. So erhält man eine

trennende Hyperebene in dem Hilbertraum, die zurückabgebildet in den Ursprungsraum die Klassen optimal trennt. Um nun zusätzlich noch die gewollte Fehlertoleranz umzusetzen, wird das Verfahren um Schlupfvariablen erweitert. Das gesamte Optimierungsproblem für eine SVM sieht dann wie folgt aus:

$$\begin{aligned} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n}\xi_i \to min \quad & , w \in \mathcal{H}_0, b \in \mathbb{R}, \xi \in \mathbb{R}^n \\ \text{s.t. } y_i\left(\langle w, \Phi(x_i)\rangle + b\right) \geq 1 - \xi_i \quad & , 1 \leq i \leq n \\ \xi_i \geq 0 \quad & , 1 \leq i \leq n \end{aligned}$$

Um zu vermeiden, dass zu viele Berechnungen in dem Hilbertraum vorgenommen werden müssen, nutzt man die Lagrangemethode um die folgende duale Problemstellung zu erhalten:

$$\begin{aligned} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j)\rangle \to max \quad & \alpha \in [0, C]^n \\ \text{s.t. } \sum_{i=1}^{n} y_i \alpha_i = 0, & \end{aligned}$$

wobei die $\alpha_{i/j}$ die Lagrange-Multiplikatoren sind. Lediglich der Term $\langle \Phi(x_i), \Phi(x_j)\rangle$ muss noch in $\mathcal{H}_0$ berechnet werden, doch mit Hilfe eines Kerns k kann auch dieses Skalarprodukt in $\mathbb{R}^d$ berechnet werden, sogar ohne Kenntnis über die eigentliche Featuremap.

Das Ergebnis dieses Verfahrens ist eine Entscheidungsfunktion $f_D(x)$ die für den Vektor x bestimmt, auf welcher Seite der Hyperebene er liegt, d.h. zu welcher Klasse er gehört.

$$\begin{aligned} & f_D(x) = sgn\left(\langle w_D, \Phi(x)\rangle + b^*\right) \\ \text{mit} \quad & w_D = \sum_{i=1}^{n}\alpha_i^* y_i \Phi(x_i) \\ \text{und} \quad & b^* = y_j - \sum_{i=1}^{n}\alpha_i^* y_i k(x_i, x_j) \\ \Rightarrow \quad & f_D(x) = sgn\left(\sum_{i=1}^{n}\alpha_i^* y_i k(x_i, x) + y_j - \sum_{i=1}^{n}\alpha_i^* y_i k(x_i, x_j)\right) \end{aligned}$$

Die Berechnung von w_D lässt erkennen, dass nur Trainingsvektoren mit $\alpha_i^* \neq 0$ in die Berechnung der Hyperebene mit einfließen. Daher werden gerade diese Vektoren die Supportvektoren genannt und sind auch namensgebend für das gesamte Verfahren.

3.2 Multiclass SVMs

In den vorherigen Definitionen wurde immer nur von zwei Klassen gesprochen, die durch eine Hyperebene mit maximalem Margin getrennt werden sollen. In vielen Anwendungsfällen liegt aber kein binäres Klassifikationsproblem vor. Stattdessen müssen die Merkmals-Vektoren in viele verschiedene Klasse eingeordnet werden können. Um dieses Problem zu lösen gibt es verschiedene Ansätze, die auf unterschiedliche Art und Weise das bisher vorgestellte Konzept erweitern. Die bekanntesten Vertreter sind „One Against All (OAA)“ und „One Against One (OAO)“[HL02]. Diese Verfahren trainieren mehrere binäre SVMs und kombinieren ihre Ergebnisse bei einer Klassifikation um eine Vorhersage treffen zu können.

Es gibt auch Ansätze, die ein Muliclass Problem direkt durch eine SVM lösen wollen, die mehrere Hyperebenen aufspannt. Jedoch wird das Optimierungsproblem dadurch sehr komplex und benötigt entsprechend viel Rechenzeit. Daher sind solche Verfahren gerade für große Datenmengen, wie auch beim Projekt MAmkS, eher ungeeignet.

Im Folgenden sei k die Anzahl an Klassen, die in einem Klassifikationsproblem differenziert werden sollen.

3.2.1 One Against All

Beim Training einer OAA-SVM wird iterativ jede einzelne Klasse betrachtet, wobei die restlichen Klassen zu einer einzelnen Klasse (dem „Rest“) zusammengefasst werden. So erhält man jeweils binäre Teilprobleme, die durch eine binäre SVM gelöst werden können. Insgesamt ergeben sich so k verschiedene SVMs, für jede Klasse genau eine. Dieser ganze Prozess ist an einem Beispiel in Abbildung 1 veranschaulicht.

Ein Problem stellt sich nun aber noch durch die Verwendung der Signum-Funktion in der binären SVM. Dadurch könnte einem neuen Datenpunkt mehr als eine Klasse zugeordnet werden bzw. es könnte keine Klassifizierung vorgenommen werden (siehe Abbildung 2). Daher wird die Signum-Funktion weggelassen und es wird direkt mit den kontinuierlichen Werten der Entscheidungsfunktion die Klasse ausgewählt, die den höchsten Wert hat. So entstehen keine undefinierten Bereiche und eine vollwertige Multiclass SVM wurde konstruiert.

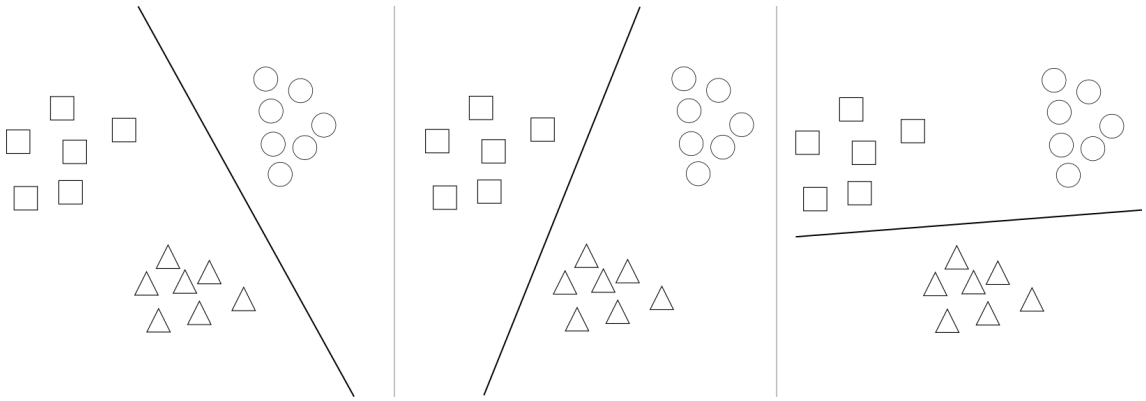


Abbildung 1: Darstellung des One Against All-Verfahrens

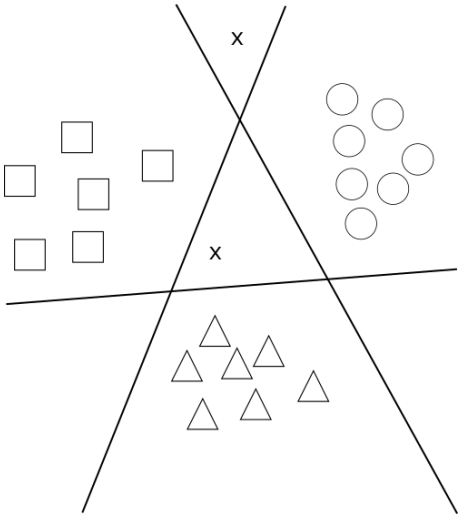


Abbildung 2: Zwei Beispiele sind durch ein „X“ markiert, die bei Verwendung der Signumfunktion nicht korrekt klassifiziert werden können.

3.2.2 One Against One

Im Gegensatz zu dem OAA-Ansatz, der eine Klasse immer von ihrem Rest trennt, wird beim OAO-Ansatz für jede paarweise Kombination aller Klassen jeweils eine SVMkonstruiert. Dabei werden dann nur die Datenpunkte der beiden betrachteten Klassen in das Training einbezogen, der Rest wird ignoriert. Die Anzahl an zu trainierenden SVMs wird daher durch den Term $\frac{k(k-1)}{2}$ bestimmt.

Die Klassifikation eines neuen Datenpunktes ist bei diesem Verfahren nicht strikt vorgegeben. Es ist aber üblich, mit jeder trainierten SVM die Klassifikation durchzuführen und die Stimmen für die einzelnen Klassen zu zählen. Am Ende wird die Klasse ausgewählt, die die meisten Stimmen hat. Bei einem Gleichstand kann z. B. die Klasse mit dem kleinsten Index ausgewählt werden.

3.3 One-class SVMs

Bisher wurden nur Verfahren betrachtet, die versuchen mehrere Klassen voneinander zu trennen. Es gibt aber auch Anwendungsfälle, in denen ein Datensatz nur aus Positivbeispielen besteht. Aus den Daten soll ein Modell gelernt werden, das beurteilen kann ob ein neuer Datenpunkt in dem Raum von gültigen Lösungen liegt, der von den Trainingsdaten aufgespannt wird. Man spricht dann von „One-class Klassifikation“. Schölkopf et al. modifizieren hierfür eine normale SVM so, dass durch eine aufgespannte Hyperebene der Raum der Positivklasse von dem Rest gut abgeschirmt wird[SWS$^+$00].

Ein anderer Ansatz ist die sogenannte Support Vector Data Description (SVDD)[TD01]. Die Idee ist, die Trainingsdaten so in einen hochdimensionalen Hilbertraum abzubilden, sodass sie von einer möglichst minimalen Hypersphäre (im folgendem auch als Kugel bezeichnet) eingeschlossen werden. Das Urbild dieser Kugel bildet dann eine Fläche, die den Trainingsdatensatz umschließt (siehe Abbildung 3). Je kleiner die gefundene Hypersphäre ist, desto besser schmiegt sich die Fläche an die Daten an.

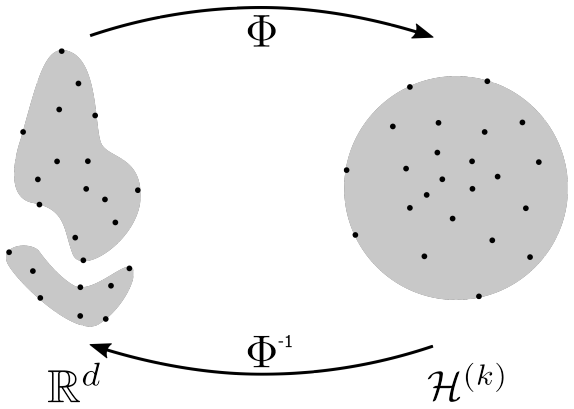


Abbildung 3: Visualisierung des SVDD-Verfahrens.

Zur Abbildung der Daten wird wie bei der SVM eine Featuremap Φ verwendet. Die Ungleichung $\|\Phi(x_i) - a\|^2 \leq R^2 + \xi_i, \forall i$ beschreibt dann die gesuchte Hypersphäre, wobei a der Mittelpunkt der Kugel, $\|\cdot\|$ der euklidische Abstand und R der Radius der Kugel ist. Außerdem wurden noch Schlupfvariablen ξ_i eingeführt, um eine Toleranz für fehlerhaft klassifizierte Datenpunkte zu ermöglichen. Je nach Wahl der Featuremap Φ können sich unterschiedliche Hypersphären ergeben, die die Datenpunkte umfassen. Es ergibt sich nun also ein Optimierungsproblem, das die Featuremap so wählen soll, sodass der Radius der Kugel minimal ist. Dieses Optimierungsproblem lässt sich mit der Lagrangemethode formalisieren und in weiteren Schritten umformen. Für detaillierte Einblicke sei auf [TD01]

und [Bre15, S. 61-62] verwiesen. Als Ergebnis erhält man folgende Wolfe Dualform des Optimierungsproblems:

$$W(\beta) = \sum_i \Phi(x_i)^2 \beta_i - \sum_{i,j} \beta_i \beta_j \Phi(x_i)\Phi(x_j).$$

Hierbei ist β ein Vektor, der die Werte der Langrange'schen Multiplikatoren beschreibt und als einziges Funktionsargument frei wählbar ist. Diese Funktion soll nun maximiert werden, sodass

$$0 \leq \beta_i \leq C$$
$$\sum_i \beta_i = 1.$$

Der Parameter C ist von Benutzer zu Beginn einstellbar und beeinflusst die Fehlertoleranz. Ist kein Fehler in den Trainingsdaten zu erwarten, so kann $C = 1$ gewählt werden. Falls jedoch Ausreißer zugelassen werden sollen, so kann auch ein $C < 1$ gewählt werden. Somit werden die Lagrange-Multiplikatoren in ihrem Wertebereich eingeschränkt und die Größe der Hypersphäre schrumpft. Die Trainingsvektoren, die nun außerhalb der Kugel liegen, werden auch als gebundene (bounded) Supportvektoren bezeichnet.

Mit Hilfe eines Kerns k können die Berechnungen der Skalarprodukte in $\mathcal{H}$ wieder in $\mathbb{R}^d$ durchgeführt werden. Bei der Supportvektor Data Description hat sich ein Gaußkern (oder auch Radial Basis Function (RBF)) allgemein bewährt:

$$k(x_i, x_j) = e^{-\frac{1}{2\sigma^2}\|x_i - x_j\|^2}.$$

σ beschreibt die Bandbreite des Gaußkerns.

Insgesamt erhält man so

$$W(\beta) = \sum_i k(x_i, x_i)\beta_i - \sum_{i,j} \beta_i \beta_j k(x_i, x_j).$$

Somit ist die Bestimmung des Maximums wieder ohne Kenntnis über die eigentliche Featuremap Φ möglich. Als Ergebnis erhält man die optimale Belegung des Vektors β, um die kleinste Hypersphäre zu beschreiben. Die Trainingsvektoren mit einem Lagrange-Multiplikator $\beta_i = 0$ werden in das innere der Hyperkugel abgebildet und sind somit für ihre Beschreibung irrelevant. Trainingsvektoren mit $0 < \beta_i < C$ werden stattdessen

auf die Kugeloberfläche abgebildet und stellen somit die Supportvektoren dar. Falls ein $C < 1$ gewählt wurde, kann es auch $\beta_i = C$ geben. Die zugehörigen Vektoren liegen dann außerhalb der Kugel und sind gebundene Supportvektoren.

Zuletzt ist nun noch eine Entscheidungsfunktion gesucht, die für Eingabevektoren bestimmt, ob sie innerhalb bzw. auf der Oberfläche der Kugel liegen. Dies geschieht durch die Bestimmung des Abstandes des Eingabevektors vom Mittelpunkt der Kugel und anschließendem Vergleich dieses Abstands mit dem Radius der Hypersphäre. Der Mittelpunkt der Kugel kann dabei durch $a = \sum_i \beta_i \Phi(x_i)$ beschrieben werden [Bre15, S. 63]. Der Abstand eines Datenpunktes $z \in \mathbb{R}^d$ von a in $\mathcal{H}$ bestimmt sich dann wie folgt:

$$\begin{aligned} R^2(z) &= \|\Phi(z) - a\|^2 \\ &= \left\| \Phi(z) - \sum_i \beta_i \Phi(x_i) \right\|^2 \\ &= \Phi(z) \cdot \Phi(z) - 2 \cdot \sum_i \beta_i \Phi(x_i)\Phi(z) + \sum_{i,j} \beta_i \beta_j \Phi(x_i)\Phi(x_j) \end{aligned}$$

Mit einem Kern k lassen sich die Skalarprodukte ersetzen zu

$$R^2(z) = k(z,z) - 2 \cdot \sum_i \beta_i k(x_i, z) + \sum_{i,j} \beta_i \beta_j k(x_i, x_j).$$

Falls nun noch der bereits oben beschriebene Gaußkern verwendet wird, so vereinfacht sich die Gleichung zu

$$R^2(z) = 1 - 2 \cdot \sum_i \beta_i k(x_i, z) + \sum_{i,j} \beta_i \beta_j k(x_i, x_j).$$

Zuletzt muss noch der Radius der konstruierten Hypersphäre bekannt sein, damit der berechnete Abstand eines beliebigen Datenpunktes zum Mittelpunkt auch interpretiert werden kann. Da alle Supportvektoren mit $0 < \beta_i < C$ auf der Hülle der Kugel liegen, kann mit einem beliebigen dieser Vektoren und der Radiusgleichung auch der Radius $R_{\mathcal{S}}$ der Hypersphäre berechnet werden. Somit kann die Positivklasse folgendermaßen definiert werden:

$$\mathcal{X} = \{x | R(x) \leq R_{\mathcal{S}}\}.$$

Um nun die SVDD für das Projekt MAmkS nutzbar zu machen, kann analog zu den „One Against All“- oder „One Against One“-Verfahren für jede Klasse eine minimale

Hypersphäre berechnet werden. Bei der Klassifizierung würde dann der Abstand des neuen Vektors zu dem Mittelpunkt aller Hypersphären berechnet werden und es wird die Klasse mit dem geringsten Abstand ausgewählt. Es können aber auch noch erweiterte Entscheidungsfunktionen gewählt werden[YMT11].

Im Gegensatz zu den anderen Multiclass Verfahren kann dieser Ansatz auch feststellen, ob ein neuer Vektor überhaupt zu einer Klasse gehört. Denn wenn der neue Datenpunkt in keine der Hypersphären abgebildet wird, so scheint es sich entweder um eine neue Klasse oder um eine Abweichung zu handeln. Letzteres könnte durch eine gewisse Fehlertoleranz noch ausgeglichen werden.

4 Adaptive Multi-Hyperplane Machine

4.1 Motivation

Die AMM ist ein im Jahre 2011 eingeführtes Verfahren, das ausgehend von einer Linearen SVM einige Erweiterungen und Optimierungen vornimmt, um die folgenden Ziele zu erreichen[WDCV11]

- Lösung von Multiclass Problemen,
- Lösung von nicht-linearen Problemen,
- effizientes Trainieren und Vorhersagen.

4.2 Realisierung

Zunächst wird der binäre Klassifikator zu einem Multiclass Modell erweitert, indem der Gewichtsvektor w des Optimierungsproblems so erweitert wird, dass jede Klasse einen eigenen Gewichtsvektor zugewiesen bekommt. Die Entscheidungsfunktion dieses Modells wählt dann die Klasse mit der größten Ähnlichkeit (*similarity score*) zu einem neuen Datenpunkt durch folgende Formel aus:

$$f(x) = \arg\max_{i \in Y} \langle w_i, x \rangle.$$

Dieses Problem wird in einem weiteren Schritt nochmal erweitert, indem nun jeder Klasse nicht nur ein einziger Gewichtsvektor zugewiesen wird, sondern beliebig viele. Das j-te Gewicht der i-ten Klasse wird durch $w_{i,j}$ beschrieben. Ein Datenpunkt wird nun genau dann korrekt klassifiziert, wenn der *similarity score* mindestens eines der j Gewichte der i-ten Klasse größer ist als alle anderen Ergebnisse der falschen Klassen. Durch diese Erweiterung erreicht das Verfahren eine erhöhte Aussagekraft, da die Klassen nur komplexer

beschrieben werden können. Die resultierende Entscheidungsfunktion:

$$f(x) = \arg\max_{i \in Y} \left(\max_j \langle w_{i,j}, x \rangle \right).$$

Letzteres Verfahren wird als Multi-Hyperplane Machine (MM) bezeichnet und ist der Grundbaustein der AMM. Auf die konkreten Optimierungsprobleme der obigen Ansätze wird in dieser Ausarbeitung nicht eingegangen und kann in [WDCV11] und [AS05] eingesehen werden.

Entscheidend ist, dass die Optimierungsprobleme recht komplex sind und durch konvexe Optimierung gelöst werden. Das kann unter Umständen aber sehr lange dauern, sodass eine AMM eine alternative Optimierungsstrategie wählt. Durch die Verwendung von Stochastic Gradient Descent (SGD) kann das Optimierungsproblem iterativ in Richtung des Optimums angenähert. Durch ein Abbruchkriterium kann dieser Prozess dann unterbrochen werden, sodass eine entsprechende Approximation des Optimums gefunden wurde. Während des Ablaufs können für eine Klasse beliebig viele neue Gewichtsvektoren erstellt werden. Diese Eigenschaft gibt dem Verfahren das Präfix „Adaptiv“, da das Modell so kontinuierlich angepasst werden kann[WDCV11].

Im Groben wird durch SGD pro Iteration jeder Gewichtsvektor Richtung null reduziert, mit Ausnahme der Gewichte, die vom aktuell betrachteten Trainingsvektor betroffen sind. Das können einerseits die Gewichte der aktuellen Klasse sein, diese werden in Richtung des Datenpunktes bewegt. Das Gewichte mit der größten Ähnlichkeit der falschen Klassen wird von diesem Datenpunkt entfernt. So charakterisieren die Gewichte ihre eigenen Klassen und werden möglichst von den anderen Klassen getrennt. Da viele Gewichte auf diese Art gegen null streben, wird noch ein Schritt namens „Weight Pruning“ eingeführt. Dabei werden alle Gewichte, die unter einem gewissen Grenzwert liegen, auf null gesetzt. So kann die Spezifizierung des Modells an den Trainingsdatensatz verhindert werden[WDCV11].

Durch kleinere Modifizierungen an diesem Verfahren kann auch eine Online AMM kreiert werden. Dabei kann aber nicht garantiert werden, dass das Verfahren zum lokalen Optimum konvergiert. Dafür wird in einem Schritt der AMM-Optimierung der gesamte Datensatz benötigt. Als Vorteile werden so eine bessere Performanz sowie die Reduzierung des Speicherverbrauchs erreicht. Experimentelle Ergebnisse haben gezeigt, dass die Ergebnisse der Online AMM nicht ganz so präzise wie sein Offline Pendant sind, aber dennoch im vertretbaren Rahmen liegen[WDCV11].

Trainingsdatensatz			Fehlerrate (%)			Trainingszeit (Sekunden)		
Größe	Klassen	Dimensionen	AMM	Online AMM	RBF SVM	AMM	Online AMM	RBF SVM
32.561	2	123	15.03±0.11	16.44±0.23	14.97	2	0.2	99
1.976.130	2	3.231.961	1.34±0.21	2.87±1.49	-	400	24	-
8.000.000	2	784	0.53±0.05	0.54±0.03	0.43	3084	300	2 Tage*
8.000.000	8	784	3.20±0.16	3.36±0.20	0.67	13864	1200	8 Tage

Tabelle 1: Vergleich von AMM, Online AMM und Kernel SVM. Durchgeführt auf einem einzelnen Prozessor. Ausnahmen sind mit einem * markiert und wurden parallelisiert auf 512 Prozessoren durchgeführt. Auszug aus [WDCV11].

4.3 Vergleich mit SVM

AMMs wurden ausgiebig getestet um eine Vergleichbarkeit mit SVMs in Hinsicht auf Performanz und Genauigkeit zu erreichen. Diese Tests haben generell gezeigt, dass eine AMM von der Präzision zwischen einer Linearen SVM und einer Kernel SVM liegt. Die Performanz hingegen ist vergleichbar mit der einer Linearen SVM und deutlich besser als die einer Kernel SVM[WDCV11].

Konkretere Ergebnisse sind in Tabelle 1 zu sehen. Die Ergebnisse zeigen, dass bei großen Datensätzen mit mehreren Klassen eine Kernel SVM viele Tage Rechenzeit benötigt, während eine AMM im Stundenbereich liegt. Dabei ist die Genauigkeit etwas schlechter. Der Vergleich zeigt auch, dass eine Online AMM zwar schneller als eine Offline AMM, dafür aber auch etwas ungenauer ist[WDCV11].

5 Aktivitätserkennung mit SVM/AMM

In der Literatur lassen sich viele Arbeiten finden, die eine Aktivitätserkennung mit Hilfe von SVMs lösen. Viele dieser Arbeiten nutzen die Sensoren eines Handys bzw. Smartphones um Trainingsdaten zu sammeln und anschließend dann auch die zu erkennende Bewegung zu messen. Zwischen diesen einzelnen Arbeiten lassen sich dann aber doch deutliche Unterschiede in der konkreten Realisierung finden.

In einem Paper werden lediglich die Accelerometerdaten verwendet, um darauf die Features Durchschnitt, Varianz, Korrelation, FFT energy und frequency domain entropy zu berechnen. Dabei wird ein halb-überlappendes Sliding Window verwendet, dessen Länge in verschiedenen Tests variiert wurde, um eine geeignete Länge zu identifizieren. Die Länge von 4 Sekunden hat sich als am besten geeignet erwiesen. Als konkrete SVM wird eine Kernel SVM mit einem Gaußkern verwendet. Damit sollen dann folgende Aktivitäten differenziert werden: Stationär, Gehen, Rennen, Fahrradfahren, Treppen hochgehen, Treppen heruntergehen und Autofahren. Die Erfolgsrate liegt dabei insgesamt bei ca. 93%. Konkreter zeigt sich, dass folgende Aktivitätspaare etwas schwerer zu unterscheiden sind: Stationär und Autofahren, Treppen hoch- und heruntergehen[TP16].

Auch eine andere Arbeit verwendet nur die Accelerometerdaten eines Smartphones. Verwendete Features sind unter anderem Durchschnitt, Standardabweichung, signal magnitude area, Entropie und Korrelation. Es wird ein halb-überlappendes Sliding Windows mit ein Länge von 2,56 Sekunden verwendet. Es wird eine Hardware-Friendly SVM verwendet, die durch OAA erweitert wird um ein Multiclass Problem zu lösen. Als Kernel wird ein Laplacekernel verwendet, da dieser ressourcensparender ist. Insgesamt sollen die Aktivitäten Laufen, Treppen hochgehen, Treppen heruntergehen, Stehen, Sitzen und Liegen unterschiedene werden. Die erreichte Genauigkeit der trainierten SVM liegt bei ca. 90%, wobei der größte Genauigkeitsverlust durch die Unterscheidung von Treppen hoch- und heruntergehen vorliegt. Die anderen Aktivitäten können zu ca. 95% erkannt werden, Liegen wird sogar fehlerfrei erkannt[AGO+12].

Weitere Recherchen ergaben keine Ergebnisse, in denen eine Aktivitätserkennung mit Hilfe von AMM durchgeführt wurde. Dies mag darauf zurückzuführen sein, dass das Verfahren im Jahr 2011 vorgestellt wurde und daher noch recht jung ist.

Auch konnten keine Arbeiten gefunden werden, in denen eine One-class SVM zu einer Mutliclass SVM erweitert wurde (wie in Abschnitt 3.3 vorgeschlagen), um eine entsprechende Analyse durchzuführen.

6 Integration in das Projekt MAmkS

Support Vector Machines sind vom Prinzip her nur offline und nur durch einige Modifikationen auch online anwendbar. Da aktuelle Überlegungen in der Projektgruppe aber gezeigt haben, dass auch ein Offlinetraining denkbar wäre, würden SVMs ein mögliches Verfahren sein, das zur Klassifizierung der Aktivitäten eingesetzt werden kann. Bereits andere Arbeiten haben gezeigt, dass durch SVMs eine gute Erkennungsquote von rund 90% erreicht werden kann. Dabei haben diese Ansätze nur Accelerometerdaten und daraus abgeleitete Features verwendet. Durch die größere Vielfalt an Sensordaten ist in dem Projekt gegebenenfalls eine bessere Erkennungsrate möglich.

Eine Adaptive Multi-Hyperplane Machine ist genau wie eine SVM in das Projekt integrierbar. Darüber hinaus kann aber auch eine Online AMM verwendet werden, sodass dieser Verfahren mehr Freiheiten beim Lernen bietet. Es ist wahrscheinlich eine etwas geringere Genauigkeit als bei den SVMs zu erwarten, jedoch muss dies innerhalb des Projekts genauer evaluiert werden.

Der in Abschnitt 3.3 vorgestellte Ansatz zur Kombination mehrere One-class SVMs zur Konstruktion eines Multiclass Klassifikators wäre auch denkbar. Diese Idee ist aber recht unpopulär und müsste daher vermutlich von der Projektgruppe implementiert werden. Da der vorgestellte Ansatz aber recht einfach mit Hilfe eines Frameworks zur Lösung des SVDD-Problems umgesetzt werden kann, ist diese Herangehensweise durchaus möglich.

Literatur

[AGO+12] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Lecture Notes in Computer Science*, pages 216–223. Springer Science + Business Media, 2012.

[AS05] Fabio Aiolli and Alessandro Sperduti. Multiclass classification with multi-prototype support vector machines. *J. Mach. Learn. Res.*, 6:817–850, December 2005.

[Bre15] Jörg Bremer. Constraint-handling mit supportvektor-dekodern in der verteilten optimierung, 2015.

[HL02] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, mar 2002.

[SWS+00] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection, 2000.

[TD01] D. Tax and R. Duin. One-class classification, 2001.

[TP16] C. Tran and D. Phan. Human activities recognition in android smartphone using support vector machine. 7th International Conference on Intelligent Systems, Modelling and Simulation, 2016.

[Vap95] Vladimir Vapnik. *Nature of Statistical Learning Theory*. Springer New York, 1995.

[VL63] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[WDCV11] Zhuang Wang, Nemanja Djuric, Koby Crammer, and Slobodan Vucetic. Trading representability for scalability: Adaptive multi-hyperplane machine for nonlinear classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*. Association for Computing Machinery (ACM), 2011.

[YMT11] Lei Yang, Wei-Min Ma, and Bo Tian. New multi-class classification method based on the SVDD model. In *Advances in Neural Networks – ISNN 2011*,

pages 103–112. Springer Science + Business Media, 2011.

D.5. Merkmals-Visualisierung

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Merkmals-Visualisierung

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Kjel Barjenbruch

Oldenburg, den 7. April 2017

Abkürzungen

ANN	artificial neural network (deut.: künstliches neuronales Netz)
AMM	adaptive multi-hyperplane machine
BDT	boosted decision tree, deut.: verstärkter Entscheidungsbaum
CLI	command line interface, deut.: Kommandozeilen-Schnittstelle
csv	comma separated values, deut.: Komma-getrennte Werte
DEMMI	De Morton Mobility Index
GUI	graphical user interface, deut.: graphische Benutzeroberfläche
ML	maschinelle Lernalgorithmen
PCA	principal component analysis, deut.: Hauptkomponentenanalyse
PGMAMKS	Projektgruppe - Mobilitäts-Assessments mit körpernahen Sensoren
SMA	signal magnitude area
TDF	tab delimited format (deut.: Tabulator-getrenntes Format)
TUG	„Timed Up & Go“-Test
VERSA	„Vorhersage zum Erhalt der Selbstständigkeit im Alter“ -Studie

Abbildungen

Tabellen

1 Einleitung

Für die Gesundheit von Personen — insbesondere Personen im höheren Alter — ist die eigene körperlicher Mobilität von entscheidender Bedeutung. Um das notwendige Maß an Mobilität zu erreichen, sind Muskelkraft und Balance erforderlich. Insofern ist der Erhalt von Muskelkraft und Balance im Alter von wesentlicher Relevanz [unib].

Im Rahmen der „Vorhersage zum Erhalt der Selbstständigkeit im Alter“ -Studie (VERSA) wurden an der Universität Oldenburg mit 350 Probanden ab einem Alter von 70 Jahren im Zeitraum vom Juli 2015 bis Januar 2016 Assessments durchgeführt. Jeder Proband wurde während der Assessments von medizinischem Personal beobachtet und zusätzlich mit unterschiedlicher Sensorik aufgezeichnet. Die Assessments setzen sich zusammen aus unterschiedlichen Mobilitäts-Tests, wie zum Beispiel dem De Morton Mobility Index (DEMMI) oder dem „Timed Up & Go“-Test (TUG) [unib][unia][aeq].

Zu der, während der Assessments eingesetzten Sensorik, zählt unter anderem ein Sensorgürtel. Bei diesem Gürtel handelt es sich um eine Kombination eines Mikrocontroller mit einem Beschleunigungssensor, einem Gyroskop, einem Magnetometer, einem Drucksensor und zwei Thermometern. Zusammen mit dem im Gürtel verbauten Akkumulator und einer Speichereinheit, bildet der Gürtel somit eine körpernahe Sensorplattform.

Im Rahmen der
Projektgruppe - Mobilitäts-Assessments mit körpernahen Sensoren (PGMAMKS) wird eine Applikation entwickelt, welche die vom Sensorgürtel aufgezeichneten Daten klassifiziert. Hierbei ist das Ziel die Klassifikation von Bewegungsmustern wie Gehen, Sitzen, Stehen, etc. und darauf aufbauend die Klassifikation der Mobilitäts-Tests wie DEMMI, welche als Bestandteil der Assessments durchgeführt werden.

Für die Klassifikation werden maschinelle Lernalgorithmen (ML) wie artificial neural network (deut.: künstliches neuronales Netz) (ANN), adaptive multi-hyperplane machine (AMM) und boosted decision tree, deut.: verstärkter Entscheidungsbaum (BDT) eingesetzt.

Diese werden mit manuell markierten (engl.: labeled) Datensätzen trainiert, sodass sie anschließend selbstständig, ohne manuelle Unterstützung Datensätze klassifizieren können. Sowohl beim Training als auch beim Klassifizieren arbeiten die MLs nicht direkt auf den Rohdaten, sondern auf Merkmalen (engl.: feature), welche mit Hilfe von Schiebefenstern (engl.: sliding window) über den Rohdaten gewonnen werden.

Bis zum aktuellen Zeitpunkt stellt sich für die PGMAMKS das Problem, dass keine Kenntnisse über die tatsächlich berechneten Merkmalsausprägungen vorliegen. Dies ist dadurch bedingt, dass bis dato keine Möglichkeit existiert Informationen über die berechneten Merkmale aus der Applikation zu extrahieren und diese anschließend zu visualisieren. Im Rahmen dieser Ausarbeitung soll nun zum einen die Umsetzung des Exports der berechneten Merkmalsausprägungen und zum anderen die Umsetzung derer Visualisierung, welche als Teil dieser Ausarbeitung implementiert wurden, vorgestellt werden.

2 Grundlagen

Im Folgenden soll eine kurze Einführung in die Umsetzung der Merkmalsgewinnung per Schiebefenster innerhalb der PGMAMKS-Applikation gegeben werden, für eine detaillierte Beschreibung ist ein Blick in die Dokumentation der Applikation zu werfen.

2.1 Merkmalsgewinnung

Die vom Gürtel aufgezeichneten Daten werden innerhalb der Anwendung in Objekten vom Typ MotionData gehalten. Genauer wird in einem DataSet-Objekt namens „sensorbelt“ für jeden Kanal z. B. für die X-Achse des Beschleunigungssensors ein DataSeries-Objekt angelegt, welches die eigentlichen Daten in Form eines DoubleList-Objekts hält.

Neben den Gürteldaten enthält das MotionData-Objekt die zugehörige zeitliche Information in Form eines TimeInterval-Feldes.

Für das Training der MLs sind zudem die manuell angelegten *Labels* (deut.: Markierungen) relevant. Diese werden, insofern vorhanden, innerhalb eines Set-Objekts von MotionLabel-Objekten eines DataSet-Objekts namens „userdata“ des MotionData-Objektes gehalten.

Innerhalb der entsprechenden Trainings-Algorithmus-Implementierungen werden dann durch eine TrainingExampleSource auf den übergebenen MotionData-Objekten mithilfe übergebener SlidingWindow-Instanzen eine Liste von TrainingExample-Objekten erzeugt, welche genutzt werden um die MLs zu trainieren. Dabei werden die Merkmale vorab über entsprechende FeatureCalculator-Implementierungen an die genutzte SlidingWindow-Instanz übergeben. Die Information, auf welchen Datenreihen die Merkmale berechnet werden, wird vorab — während der Konstruktion der FeatureCalculator — angegeben.

Die entsprechenden Klassen finden sich im Unterprojekt mamks-shared. Die Modellklassen sind im Paket mamks.model, Klassen der allgemeinen Algorithmusumsetzung sind im Paket mamks.algorithm, Klassen der allgemeinen ML-Umsetzung sind im Paket

mamks.algorithm.ml und die Merkmalsumsetzungen sind im Paket mamks.algorithm.feature zu finden.

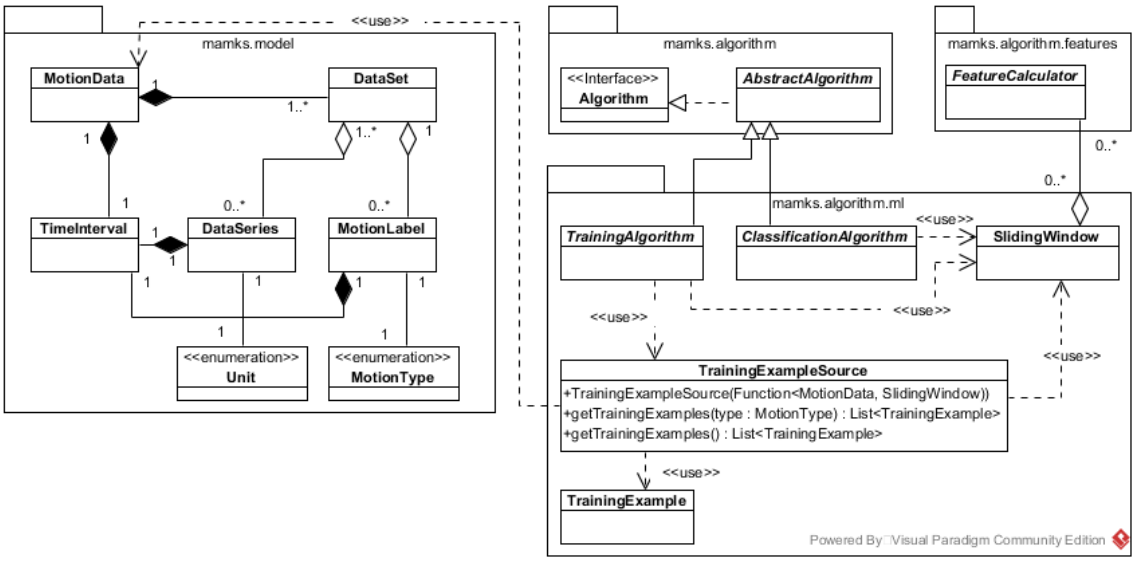


Abbildung 1: Bewegungsdaten und Algorithmen

3 Export

Wie in dem Kapitel 1 beschrieben, stellt sich für die PGMAMKS das Problem, dass die berechnete Merkmalsausprägungen bis dato nicht einfach verfügbar sind. Bisher werden die Ausprägungen bei der Ausführung eines Trainings- oder Klassifizierungsalgorithmus für ein angegebenes Schiebefenster über ausgewählten Bewegungsdaten berechnet. Innerhalb eines Trainingsalgorithmus werden diese Werte anschließend genutzt um ein Klassifikationsmodell zu trainieren. Innerhalb eines Klassifizierungsalgorithmus werden die Werte genutzt, um die Schiebefensterintervalle Klassen zu zuordnen. Dies bedeutet die berechneten Werte liegen nur temporär im Arbeitsspeicher vor und werden frühestens nach der Übergabe an den Klassifizierer und spätestens nach dem Abschluss des Algorithmus durch die JAVA-*Garbage Collection* (deut.: Müllsammlung) aus dem Arbeitsspeicher gelöscht.

Es wäre von Vorteil die Merkmalsausprägungen in einer permanenten Form z. B. einer Textdatei zur Verfügung zustellen. Dies würde es ermöglichen die Werte mithilfe der PGMAMKS-Applikation oder auch jeder anderen Anwendung wie z. B. *MatLab*- oder *R*-Entwicklungsumgebungen zu betrachten, zu bearbeiten, zu analysieren, etc., ohne sie vorab erneut berechnen zu müssen. Für die Visualisierung der Merkmals-Werte bietet sich hiermit der Vorteil eines vom Export-Schritt entkoppelten Visualisierungsschrittes. Damit ist der Nutzer im wesentlichen frei in der Wahl des Visualisierungswerkzeugs.

Der Export soll diese Funktion bieten, indem er die Merkmalsausprägungen quasi analog zu den Trainingsalgorithmen berechnet und anschließend in eine Textdatei schreibt, anstatt ein Modell damit zu trainieren.

3.1 Export-Implementierung

Für die Umsetzung des Exports wurde das Paket mamks.featurevisualization als Teil des Unterprojekts mamks-shared angelegt. Zur feineren Untergliederung wurde außerdem das Paket mamks.featurevisualization.io angelegt. Letzteres enthält die neuen Importer- und Exporter-Klassen sowie das ConstantInformation-Enum. Ersteres enthält die Klassen FeatureVisualization und FeatureData, sowie die abstrakte Klasse AbstractSlidingWindowSetup, sowie die davon abgeleiteten Klassen SlidingWindowSetUpAllFeatures und SlidingWindowSetUpSomeFeatures. Die Klassen sind im Diagramm 2 dargestellt.

Die Klasse FeatureVisualization ist von der Klasse AbstractAlgorithm abgeleitet. Es handelt sich hierbei also um eine Algorithmus-Umsetzung. Diese kann vom Benutzer über die graphical user interface, deut.: graphische Benutzeroberfläche (GUI) oder über die command line interface, deut.: Kommandozeilen-Schnittstelle (CLI) ausgeführt werden. Der Algorithmus erzeugt dann unter Nutzung einer, im Code festgelegten AbstractSlidingWindowSetup-Implementierung, ein SlidingWindow über den an den Algorithmus übergebenen MotionData-Objekten. Mit Hilfe dieses Schiebefensters wird dann eine TrainingExampleSource erzeugt. Es wird dann eine Exporter-Instanz erzeugt, an die die genutzte AbstractSlidingWindowSetup-Implementierung, die erzeugte TrainingExampleSource, sowie die an den Algorithmus übergebenen Informationen übergeben werden. Der Exporter durchläuft dann alle TrainingExample der TrainingExampleSource und schreibt diese unter Verwendung der in der AbstractSlidingWindowSetup-Implementierung enthaltenen Zusatzinformation in die angegebene Zieldatei.

Die TrainingExampleSource berechnet für jeden Schiebefensterschritt ein TrainingExample, welches die Ausprägungen aller in der AbstractSlidingWindowSetup-Implementierung gesetzten FeatureCalculator für diesen Schritt enthält. Durch den Export der TrainingExample werden also alle berechneten Merkmalsausprägungen exportiert.

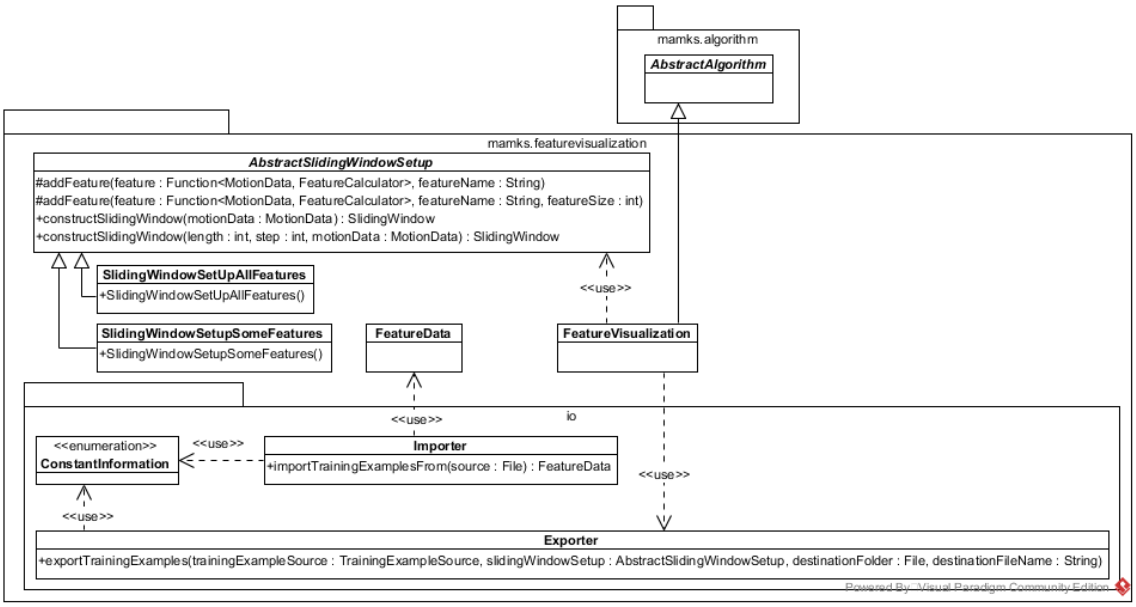


Abbildung 2: Bewegungsdaten und Algorithmen

3.2 Export-Nutzung

Der Export-Algorithmus kann sowohl per graphischer Benutzeroberfläche oder per CLI ausgeführt werden.

Um den Algorithmus per Benutzeroberfläche auszuwählen, ist der Knopf „Algorithmus anwenden“ zu klicken. Dadurch gelangt man in den „Algorithmus anwenden“-Dialog. Hier wird der Algorithmus „FeatureVisualization“ gewählt. Anschließend sind die Bewegungsdatensätze auszuwählen, für welche die Merkmalsausprägungen berechnet und exportiert werden sollen. Es öffnet sich ein Dialog wie in Abbildung 3 dargestellt. In diesem können drei Parameter gewählt werden:

applyPrincipalComponentAnalysis Für diesen Parameter kann als Wert true angegeben werden, wenn auf den Merkmalsausprägungen eine principal component analysis, deut.: Hauptkomponentenanalyse (PCA) durchgeführt werden soll und die Werte derer Hauptkomponenten ausgegeben werden sollen. Andernfalls kann der Parameter als false gesetzt werden, wenn keine PCA auf den Merkmalswerten ausgeführt werden soll und diese selbst exportiert werden sollen.

exportDestinationFolderPath Für diesen Parameter ist der Pfad zu dem Verzeichnis anzugeben, in das die Export-Datei geschrieben werden soll.

labelGroupId Hier ist die Label-Gruppe anzugeben, deren Label den exportierten Merkmalsausprägungen als Klassen zugeordnet werden sollen.

plotToImageAtPath Dieser Parameter ist optional. Als Wert kann der Pfad zu einer *png*-Bilddatei angegeben werden. Es werden dann die Merkmalsausprägungen exportiert und zudem direkt in eine Streudiagramm-Matrix gezeichnet, die in dem angegebenen Bild gespeichert wird. (Dieser Parameter ist neu und wurde ergänzend nach dem Vortrag zu dieser Ausarbeitung hinzugefügt.)

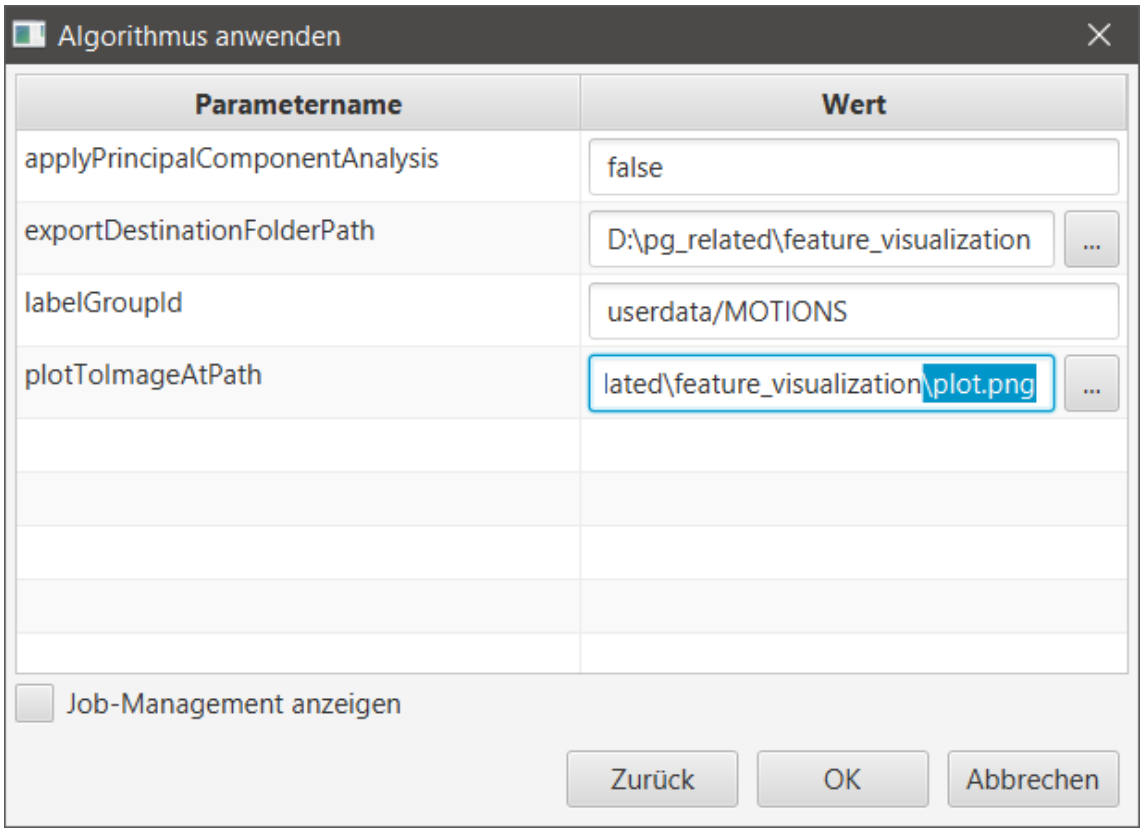


Abbildung 3: Export-Algorithmus-Dialog

Um den Algorithmus per CLI auszuführen, ist die mamks-cli.jar wie folgt per Konsole aufzurufen:

```
java -jar mamks-cli.jar exec
        -a FeatureVisualization
        -w D:\pg_related\workdir
        -M 10722/2015-07-29T17-16-49-269Z
        -P exportDestinationFolderPath=
                D:\pg_related\featureVisualization
        -P applyPrincipalComponentAnalysis=false
        -P labelGroupId=userdata/MOTIONS
```

Die angegeben Parameter entsprechen den oben beschrieben Parameters des Algorithmus.

Merkmalsauswahl

In der vorgestellten Umsetzung lässt sich die Auswahl der zu exportierenden Merkmale ausschließlich im Code ändern. Die abstrakte Klasse AbstractSlidingWindowSetup kann zu diesem Zweck mit einer eigenen Implementierung erweitert werden. Die Klassen SlidingWindowSetUpAllFeatures und SlidingWindowSetUpSomeFeatures sind zwei Beispiele einer solchen Erweiterung. Die entsprechende Umsetzung muss dann in der Klasse FeatureVisualization als Wert des Feldes SLIDING_WINDOW_SETUP gesetzt werden. Das SlidingWindowSetUpAllFeatures enthält alle zum aktuellen Zeitpunkt implementierten Merkmale, die mit Ausnahme des Nick-Winkel-Merkmals (PitchFeatureCalculator) auf den drei Akzelerometer-Achsen bzw. einer Kombination der Achsen im Falle von Merkmalen, die mehr als einen Eingangparameter erfordern, operieren. Das SlidingWindowSetUpSomeFeatures enthält alle Feature, die aktuell für das Training und die Klassifikation genutzt werden, dazu zählt die Autokorrelation, die signal magnitude area (SMA) und der Nick-Winkel (Pitch).

3.3 Export-Datei

Der Export-Algorithmus schreibt die berechneten Merkmalswerte in eine comma separated values, deut.: Komma-getrennte Werte (csv)-Datei, genauer eine csv-Datei im tab delimited format (deut.: Tabulator-getrenntes Format) (TDF)-Format. Ein Beispiel einer solchen Export-Datei sieht wie folgt aus:

START_TIME_UNIX	END_TIME_UNIX	LABELED_CLASS	LABELED_CLASS_ORDINAL	...
1438262144689000	1438262146689000	LIE_ON_SIDE	27	...
1438262145359000	1438262147359000	LIE_ON_SIDE	27	...

Die ... symbolisieren hier, dass die entsprechende Zeile noch weit länger ist.

In der ersten Spalte START_TIME_UNIX und in der zweiten Spalte END_TIME_UNIX findet sich die Startzeit und Endzeit des Intervalls auf dem das Merkmal berechnet wurde als Epoch-Zeitstempel in Mikrosekunden. In der dritten Spalte LABELED_CLASS befindet sich die Information darüber, welche Bezeichnung für das Merkmal gegeben war. In der vierten Spalte LABELED_CLASS befindet sich die Repräsentation der Klasse als Zahl.

Alle folgenden Spalten enthalten die einzelnen Merkmale und ihre Ausprägungen.

4 Visualisierung

Der Export der Merkmale ermöglicht es, diese auch außerhalb der Anwendung zu nutzen. Somit könnten die berechneten Merkmalsausprägungen z. B. auch an andere Anwendungen zum Training, zur Klassifikation oder zur Analyse übergeben werden. Er ermöglicht es zudem auch die Merkmale graphisch zu visualisieren. Dies könnte z. B. unter Verwendung von in *R* oder in *MatLab* geschrieben Programmen erfolgen.

4.1 Streudiagramm-Matrix

Bei einem Streudiagramm (eng.: scatterplot) handelt es sich um eine der am häufigsten eingesetzten graphischen Darstellungsform, welche konstruiert wird, indem die Wertepaare $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ als Punkte im einen rechtwinkligen Koordinatensystem dargestellt werden. Der Verwendungszweck des Streudiagrammes ist die Untersuchung des Zusammenhanges zwischen zwei quantitativen Variablen [NBOH96]. Ein Beispiel eines Streudiagramms ist in Abbildung 4 dargestellt.

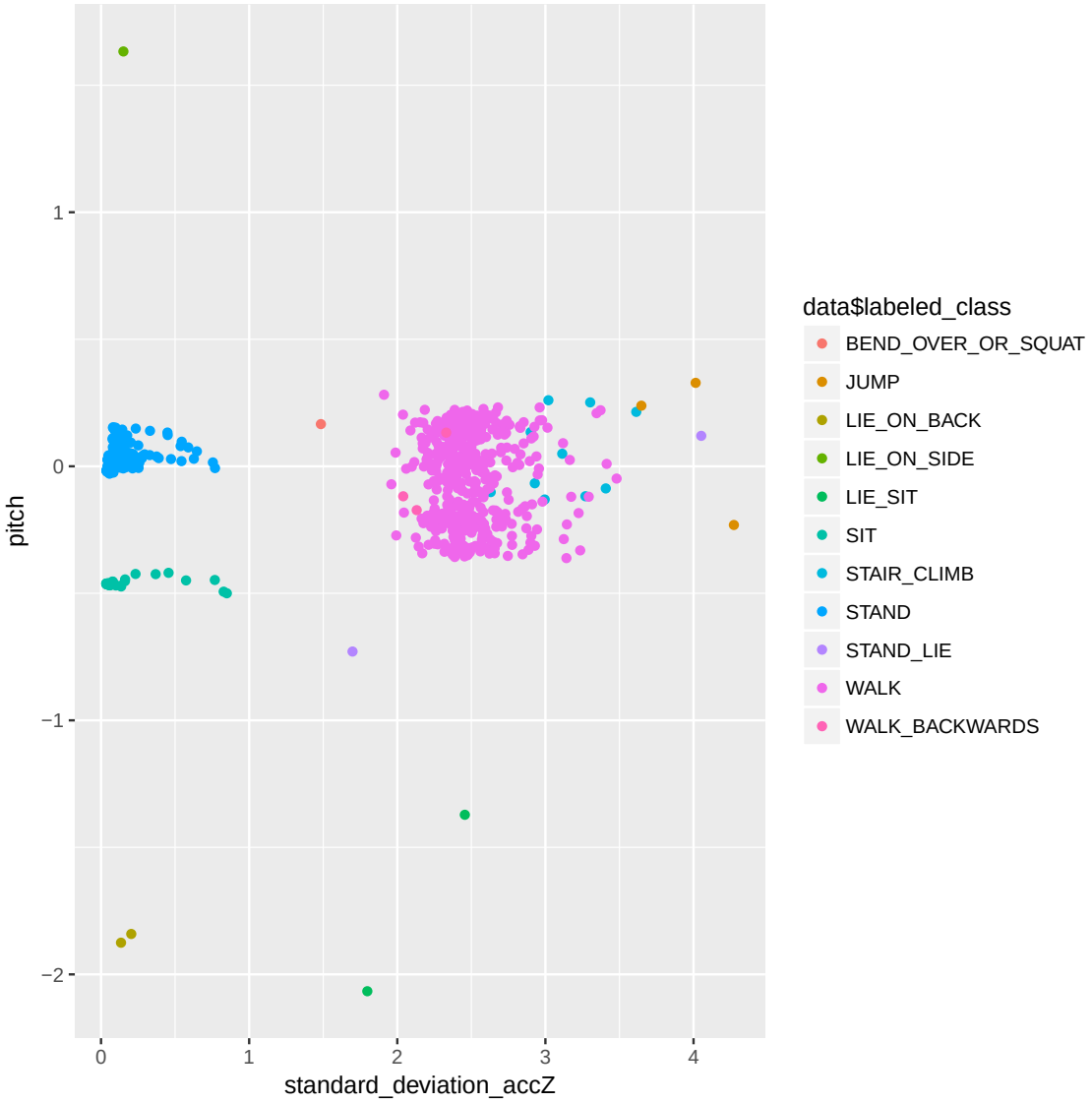


Abbildung 4: Streudiagramm Beispiel

In diesem Beispiel werden die Werte der Merkmale Nick-Winkel (Pitch) und Standardabweichung auf der Akzelerometer-Z-Achse (standard_deviation_accZ) als Streudiagramm dargestellt. Die einzelnen Punkte sind nach den ihnen zugeordneten Bezeichnungen bzw. Klassen gefärbt.

Das Problem, welches sich im Zusammenhang mit den Merkmalen, die von der PGMAMKS-Anwendung berechnet werden, stellt, ist, dass nicht nur zwei Merkmale berechnet werden sondern weit mehr. Ein 2-dimensionales Streudiagramm kann allerdings lediglich zwei Merkmale gegeneinander darstellen. Eine einfache Lösung um dieses Problem zu beheben ist die Streudiagramm-Matrix. Dabei handelt es sich um eine Matrix von Streudiagrammen. Eine Streudiagramm-Matrix über n Merkmalen ergibt sich als $n \times n$-Matrix von Streudiagrammen, wobei jedes $(i, j) - te$ Streudiagramm ein Streudiagramm der Merkmale mit Index i und Index j ist mit $i, j \in [0, n-1]$. Die Diagramme für den Fall $i = j$, also alle Diagramme entlang der Diagonalen der Matrix, werden meist nicht dargestellt, da

hier das entsprechende Merkmal gegen sich selbst dargestellt werden würde. In einigen Fällen werden auch alle Diagramme unterhalb der Diagonalen nicht dargestellt, da es sich hierbei letztlich um die gleichen Diagramme wie oberhalb der Diagonalen handelt, welche lediglich entlang der Diagonalen gespiegelt sind. In Abbildung 5 ist ein Beispiel einer Streudiagramm-Matrix zu sehen.

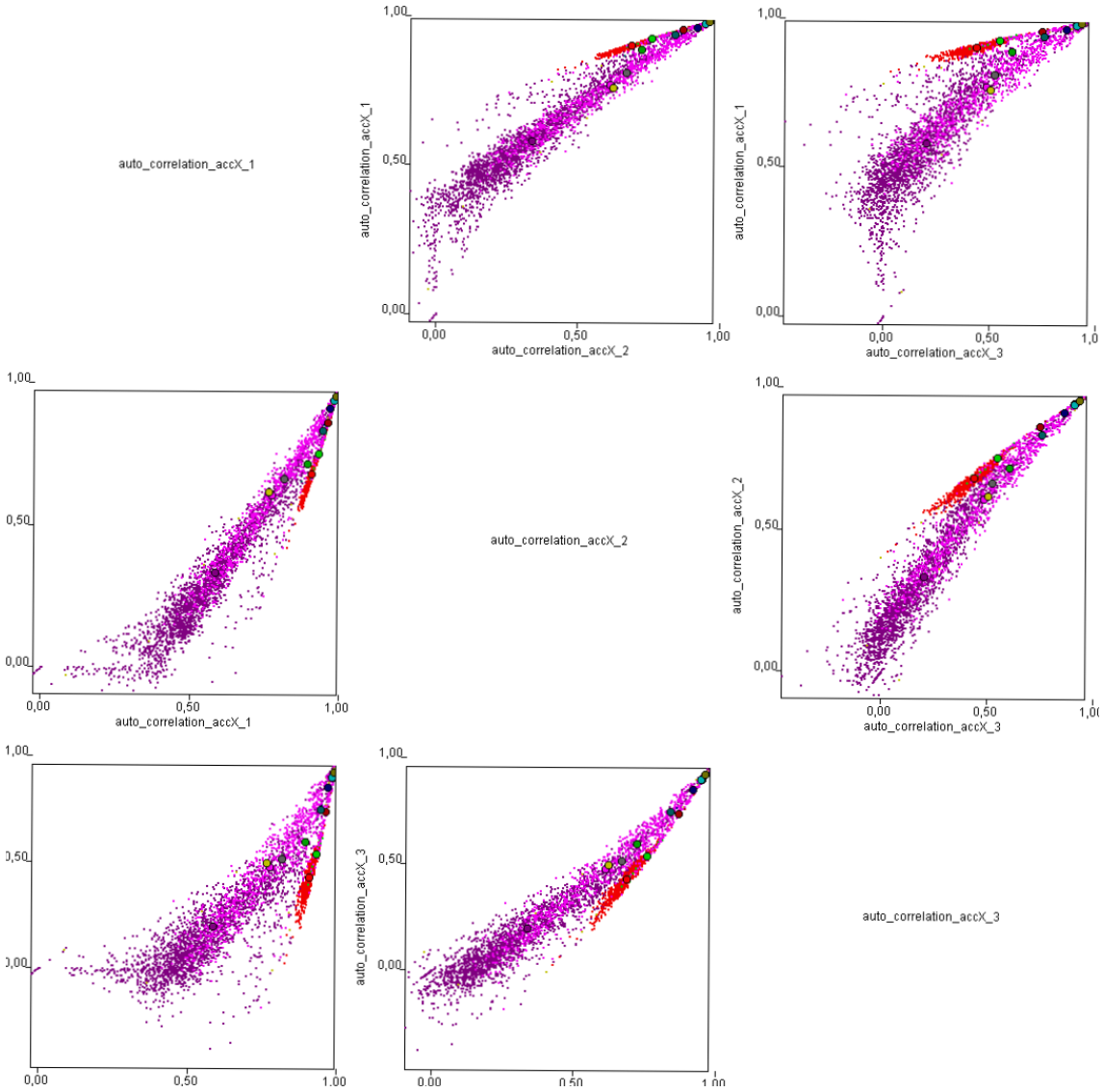


Abbildung 5: Streudiagramm-Matrix Beispiel

In diesem Diagramm sind drei Komponenten der Autokorrelation auf der Akzelerometer-X-Achse jeweils gegeneinander dargestellt. Die Punkte sind nach Beschriftungen bzw. Klassen gefärbt. Die größeren Punkte stellen die jeweiligen Mittelwerte der entsprechenden Klasse dar.

4.1.1 Streudiagramm-Matrix-Implementierung

Die in Abbildung 5 dargestellte Streudiagramm-Matrix ist tatsächlich ein Ausschnitt aus einer weit größeren Streudiagramm-Matrix, die mit Hilfe der, im Rahmen dieser

Ausarbeitung geschriebenen Streudiagramm-Matrix-Implementierung, erzeugt wurde. Im diesem Abschnitt soll beschrieben werden, wie diese Implementierung umgesetzt ist, in Abschnitt 4.1.3 ist beschrieben, welche Probleme es im Zusammenhang mit der Streudiagramm-Matrix gibt und warum die Implementierung so umgesetzt wurde wie hier beschrieben.

Die Streudiagramm-Matrix wurde in der Klasse ScatterPlotMatrix im Paket mamks.cli.visualization des Unterprojekts mamks-cli implementiert. Bei der Klasse handelt es sich um eine von JPanel erbende Klasse. Die Nutzung der *swing*-Bibliothek anstelle der ansonsten in der PGMAMKS-Applikation verwendeten *javafx*-Bibliothek ist im wesentlichen der Tatsache verschuldet, dass ursprünglich eine Implementierung unter der Verwendung der *smile*-Bibliothek, welche die *swing*-Bibliothek nutzt, angedacht war. Obwohl die *smile*-Bibliothek letztlich nicht genutzt wurde, waren alle Schnittstellen auf die *swing*-Bibliothek ausgelegt, weshalb diese in der aktuellen Lösung eingesetzt wird.

In dem mamks.cli.visualization Paket wurden zudem die Klassen PlotFrame und MotionTypeColorer implementiert. Bei ersterer handelt es sich um die Klasse des graphischen Container in den die Streudiagramm-Matrix dargestellt wird. Er ermöglicht es die Matrix sowohl vertikal, als auch horizontal zu scrollen und bietet einen Menüeintrag um die Matrix als Bilddatei abzuspeichern. Das entsprechende Fenster ist in Abbildung 6 dargestellt. Bei letzterer Klasse handelt es sich um eine Utility-Klasse mit deren Hilfe die Farbe der Beschriftung bzw. Klasse der einzelnen Punkte bestimmt werden kann.

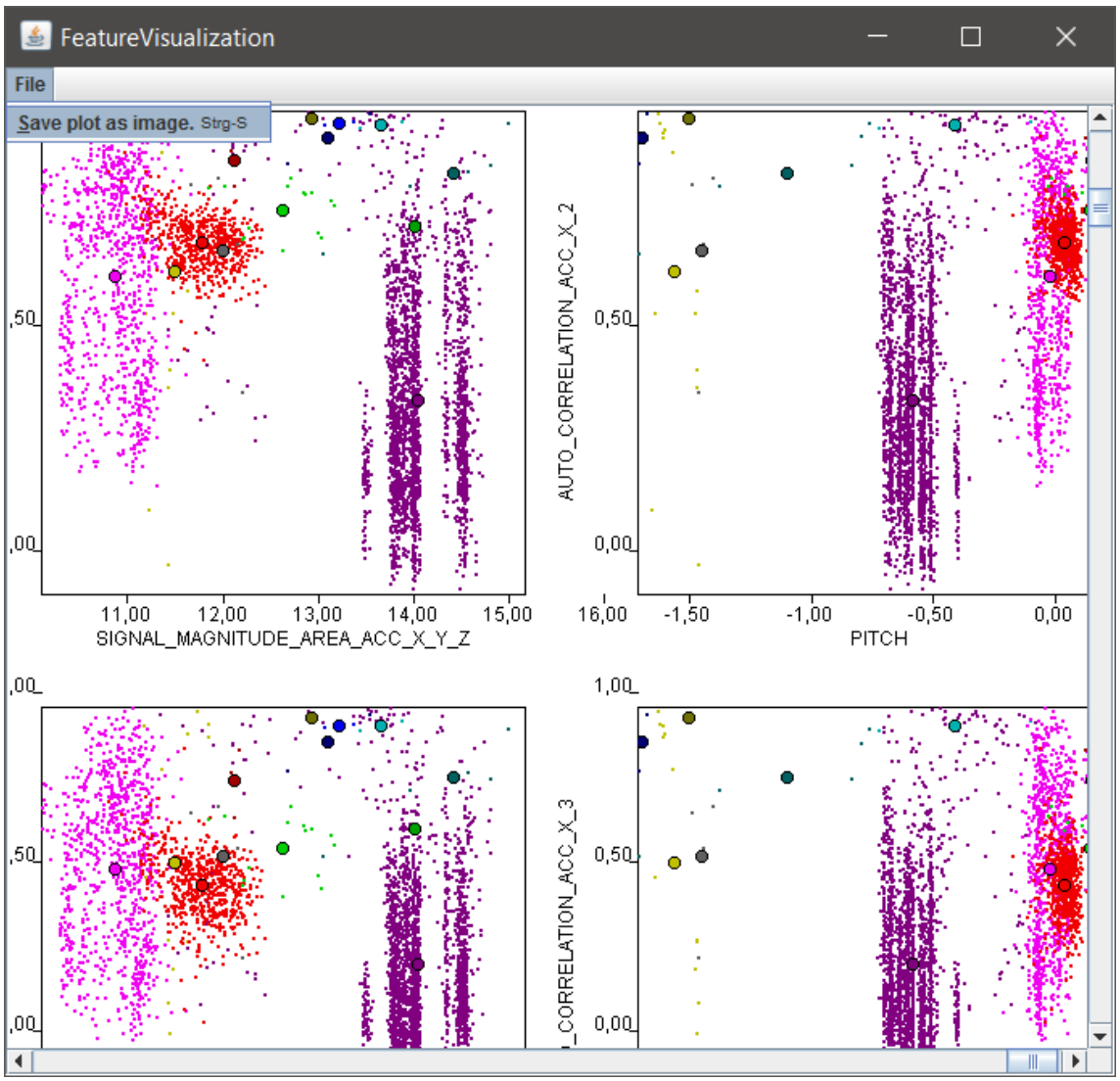


Abbildung 6: Streudiagramm-Matrix-Fenster

Damit der Benutzer auf die ScatterPlotMatrix zugreifen kann, wurden zudem im Paket mamks.cli.commands die PlotFeaturesCommand-Klasse und im Paket mamks.cli.runners die PlotFeaturesCommandRunner-Klasse angelegt. Diese beschreiben, wie der CLI-Befehl zum Zeichnen des Diagramms aussieht, bzw. implementieren dessen Ausführung. Wie der Befehl aussieht und wie er zu nutzen ist, ist in Abschnitt 4.1.2 beschrieben.

4.1.2 Streudiagramm-Matrix-Nutzung

Der „Zeichne Streudiagramm-Matrix“ -Befehl ist wie folgt aufgebaut.

```
plot-features -s <sourcePath>
                        [-c <classToDraw1> <classToDraw2> ...]
                        [-dm <drawMode>]
                        [-img <directImageExportDestination>]
```

Der Parameter -s ist zwingend zu setzen, als sein Wert <sourcePath> ist der Pfad zur Quelldatei also der Pfad zu der Datei, in die die Merkmale exportiert wurden, anzugeben. Die folgenden Parameter sind optional. Wenn der Parameter -c nicht gesetzt wird, werden alle verfügbaren Klassen im Diagramm dargestellt. Sollen nur bestimmte Klassen dargestellt werden, können diese für diesen Parameter als Werte angegeben werden. Die Namen müssen dabei den Namen der Klassen in der Quelldatei entsprechen. Wenn der Parameter -dm nicht gesetzt wird, wird in den Diagrammen das Streudiagramm inklusive der Mittelwert-Punkte der einzelnen Klassen dargestellt. Wird der Wert von -dm als 0 angegeben, werden jeweils Streudiagramm, Mittelwert-Punkte und Standardabweichungs-Kreise gezeichnet, wird der Wert als 1 angegeben, wird jeweils ausschließlich das Streudiagramm gezeichnet, wird der Wert als 2 angegeben, entspricht dies dem Standardfall und wird als Wert 3 angegeben, dann wird das Streudiagramm nicht gezeichnet sondern nur Mittelwert-Punkte und Standardabweichungs-Kreise. Wird der Parameter -img nicht gesetzt, wird das Streudiagramm in dem in Abschnitt 4.1.1 beschriebenen PlotFrame gezeichnet. Alternativ kann der Wert des Parameters als Pfad zu einer Bilddatei angegeben werden, in die die Streudiagramm-Matrix gezeichnet wird, in diesem Fall öffnet sich kein Fenster.

Beispiel

```
plot-features
        -s D:\pg_related\featureVisualization\trainingExamples.csv
        -img D:\pg_related\featureVisualization\plot.png
```

Das Beispiel zeichnet die Merkmale der Datei trainingExamples.csv im Ordner D:\pg_related\featureVisualization\ mit allen enthaltenen Klassen direkt in die Bilddatei

plot.png im selben Ordner.

4.1.3 Streudiagramm-Matrix-Problematik

Das Kernproblem mit der Streudiagramm-Matrix ist ihre Größe bzw. die Menge an darzustellenden Punkten. Bei n Merkmalen und m Schiebefensterschritten enthält die Streudiagramm-Matrix $n \cdot n$ Streudiagramme bzw. $(n \cdot n) - n$, wenn die Streudiagramme auf der Diagonalen nicht gezeichnet werden.

Angenommen eine Assessment-Aufnahme dauert $1h$ und ca. 33% der Aufnahme sind beschriftet, dann würden pro Datensatz eines Probanden $1h \cdot \frac{1}{3} = 20min$ an beschrifteten Daten anfallen. Ursprünglich haben die Trainer mit einem Schiebefenster mit $2sec$-Schritten und 66%-Überlappung gearbeitet. Die Anzahl der Schiebefensterschritte lässt sich nicht exakt bestimmen, da die Schiebefenster bei jeder Beschriftung beginnen und die überlappenden Schiebefenster dementsprechend erst nach $2sec$ bzw. $4sec$ bestimmen. Überschlagen ergeben sich ca. $s = \frac{20min}{2sec} \cdot 3 = \frac{1200sec}{2sec} \cdot 3 = 1800$ Schiebefensterschritte s pro Datensatz eines Probanden. Sei $x = 50$ die Anzahl an Datensätzen deren Merkmale exportiert wurden. Dann ergibt sich bei diesen Einstellungen $m = x \cdot s = 50 \cdot 1800 = 90.000$ Schiebefensterschritte. Jedes der $(n \cdot n) - n$ Streudiagramme enthält also 90.000 Punkte. Es ist möglich, dass zu späterem Zeitpunkt kleinere Fenstergrößen eingesetzt werden, die Anzahl an Merkmalen wäre dementsprechend größer.

Aktuell verwenden die Trainer die in SlidingWindowSetUpSomeFeatures angegebenen Merkmale. Dies sind zum einen der Nick-Winkel (Pitch), die SMA über den drei Akzelerometer-Achsen und die Autokorrelation (jeweils auf jeder der drei Akzelerometer-Achsen). Der Nick-Winkel und die SMA sind 1-dimensional, die Autokorrelation ist je nach Einstellung mehrdimensional. Bei der aktuellen Einstellung ist die Autokorrelation 11-dimensional. Damit ergeben sich $n = 1 + 1 + 11 \cdot 3 = 35$ Merkmale.

Für die Streudiagramm-Matrix ergeben sich damit $(n \cdot n) - n = (35 \cdot 35) - 35 = 1190$ Streudiagramme, die jeweils $m = 90.000$ Punkte enthalten. Insgesamt ergeben sich also $1190 \cdot 90.000 = 107.100.000$ Punkte, die in der Streudiagramm-Matrix darzustellen sind.

Die ursprüngliche Idee die exportierten Merkmale über einem in *R* geschriebenes Programm zu visualisieren, hat sich als nicht durchführbar herausgestellt. Das Programm zum Zeichnen der Streudiagramm-Matrix hat nicht terminiert und die Programme zum Zeichnen einzelner Diagramme, wie das in Abbildung 4 Dargestellte, hatten eine zu lange

Laufzeit. Der zweite Ansatz war der, die Diagramme über die Kommandozeilenschnittstelle unter der Verwendung existierender Bibliotheken wie *smile* durch ein Java-Programm zu zeichnen. Es war allerdings unter Verwendung entsprechender Bibliotheken nicht möglich eine entsprechend große Streudiagramm-Matrix zu zeichnen. Aus diesem Grund wurde als dritte Lösung eine eigene Lösung implementiert. Diese zeichnet sich im wesentlichen dadurch aus, dass die Punkte direkt auf ein Bild gezeichnet und ausschließlich das Bild nicht aber jeder Datenpunkt vorgehalten wird. Dies macht es möglich mit entsprechend großen Datenmengen umzugehen. Es führt allerdings auch zu offensichtlichen Nachteilen wie z. B., dass für eine neue Skalierung der Diagramme das gesamte Bild neu gezeichnet werden muss.

4.2 Streudiagramm-Interpretation

Die Interpretation der Streudiagramme kann unter Umständen schwierig sein. Bei der Betrachtung des Diagramms 7 kommt die Vermutung auf, dass beide Merkmale sehr gut für die Klassifikation geeignet sind, da die einzelnen Klassen deutlich erkennbare Cluster bilden. Betrachtet man jedoch das Diagramm 8 würde man vermuten, dass die beiden Merkmale für die Klassifikation nicht gut geeignet sind, da es zu deutlichen Überschneidungen zwischen den Klassen kommt. Der Unterschied zwischen den Diagrammen ist ausschließlich wie viele Probanden-Daten für die Streudiagramme herangezogen wurden. Eine Betrachtung von Diagramm 9 lässt wiederum vermuten, dass der Mittelwert auf der Z-Achse des Akzelerometers womöglich doch geeignet ist um die Klassen WALK und STAIR_CLIMB also „Gehen“ und „Treppensteigen“ zu unterscheiden.

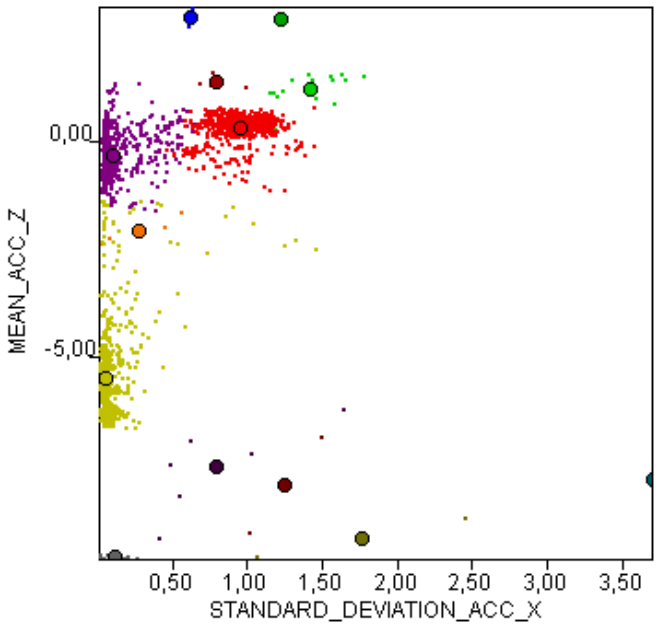


Abbildung 7: Streudiagramm - Proband 10722 - Mittelwert-Akz.-Z, Standardabweichung-Akz.-X

Das Diagramm ist Ausschnitt einer Streudiagramm Matrix über den Daten von Proband 10722. Es sind der Mittelwert auf der Z-Achse und die Standardabweichung auf der X-Achse des Akzelerometers gegeneinander dargestellt.

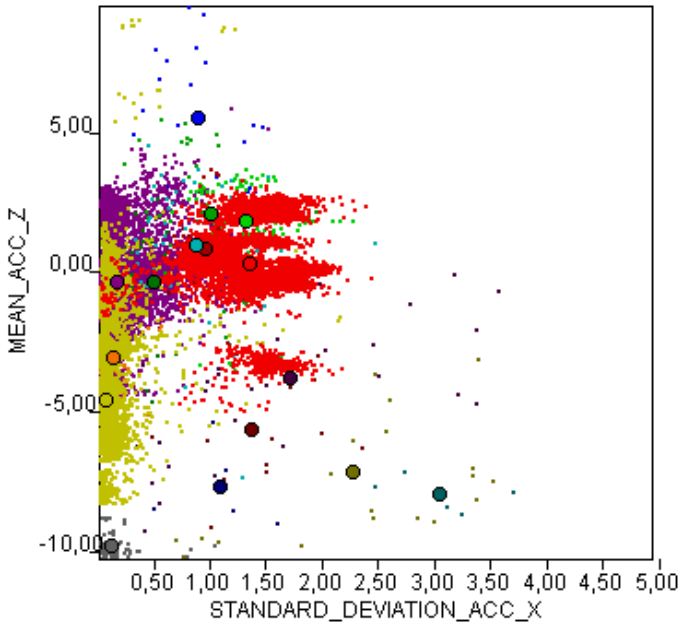


Abbildung 8: Streudiagramm - Mehrere Probanden - Mittelwert-Akz.-Z, Standardabweichung-Akz.-X

Das Diagramm ist Ausschnitt einer Streudiagramm Matrix über den Daten von Proband 10722 und neun weiteren Probanden. Es sind der Mittelwert auf der Z-Achse und die Standardabweichung auf der X-Achse des Akzelerometers gegeneinander dargestellt.

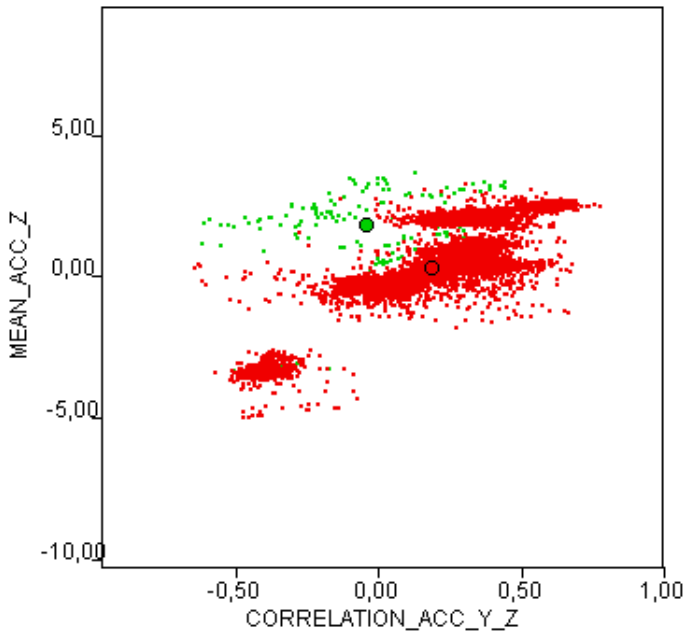


Abbildung 9: Streudiagramm - Mehrere Probanden - Mittelwert-Akz.-Z, Korrelation-Akz.-Y-Z

Das Diagramm ist Ausschnitt einer Streudiagramm Matrix über den Daten von Proband 10722 und neun weiteren Probanden. Es sind der Mittelwert auf der Z-Achse und die Korrelation der Y- und Z-Achse des Akzelerometers gegeneinander dargestellt. Die Klasse WALK ist rot und die Klasse STAIR_CLIMB ist grün gefärbt.

Ausgehend von durch Abbildung 9 abgeleiteten Annahme, dass der Mittelwert der Z-Achse des Akzelerometers und die Korrelation der Y- und der Z-Achse des Akzelerometers geeignete Merkmale sind um die Klassen „Gehen“ und „Treppensteigen“ zu unterscheiden, wurde mit Hilfe der AMM ein Training auf 68 Datensätzen (darin enthalten sind auch Datensätze über Zusatzdaten) und eine Klassifikation dieser Datensätze auf Basis des trainierten Modells durchgeführt. Im ersten Durchlauf wurden die bisher eingesetzten Merkmale verwendet. Im zweiten Durchlauf wurden zu diesen Merkmalen die hier genannten Merkmale hinzugefügt. Das Ergebnis ist in Tabelle 1 dargestellt.

Die Merkmale verbessern also das Klassifikationsergebnis.

Die Streudiagramme haben zudem eine weitere neue Erkenntnis geschaffen. Nämlich die, dass in der aktuellen Implementierung der Autokorrelation das erste Merkmal immer den Wert 1 annimmt. Folglich hat dieses Merkmal keinen Einfluss auf die Klassifikatoren und ist prinzipiell überflüssig. Dies gilt tatsächlich in allen drei Fällen der Nutzung der Autokorrelation für deren erstes Merkmal. In Abbildung 10 ist dieses Problem in einem Ausschnitt der Streudiagramm-Matrix zu sehen.

Wert \ Durchlauf	1. Durchlauf (ohne)	2. Durchlauf (mit)
Sensitivität (im Mittel)	0,39272	**0,44038**
Präzision (im Mittel)	0,71458	**0,75651**
Genauigkeit (im Mittel)	0,96775	**0,97548**
F1 (im Mittel)	0,65527	**0,73643**
Sensitivität (Gehen)	0,38094	**0,63042**
Präzision (Gehen)	**0,97575**	0,96577
Genauigkeit (Gehen)	0,89841	**0,93665**
F1 (Gehen)	0,54796	**0,76287**
Sensitivität (Treppensteigen)	**0,95792**	0,92889
Präzision (Treppensteigen)	0,31488	**0,38713**
Genauigkeit (Treppensteigen)	0,90221	**0,92910**
F1 (Treppensteigen)	0,47397	**0,54650**

Tabelle 1: Darstellung der Klassifikation-Ergebnisse

In der Spalte „1. Durchlauf (ohne)“ sind die Klassifikationsergebnisse des 1. Durchlaufs dargestellt, der ohne den Mittelwert auf der Akzelerometer-Z-Achse und ohne die Korrelation der Akzelerometer-Y- und Akzelerometer-Z-Achse als Merkmale durchgeführt wurde. In der Spalte „2. Durchlauf (mit)“ sind die Klassifikationsergebnisse des 2. Durchlaufs dargestellt, der mit den beiden Merkmalen durchgeführt wurde.

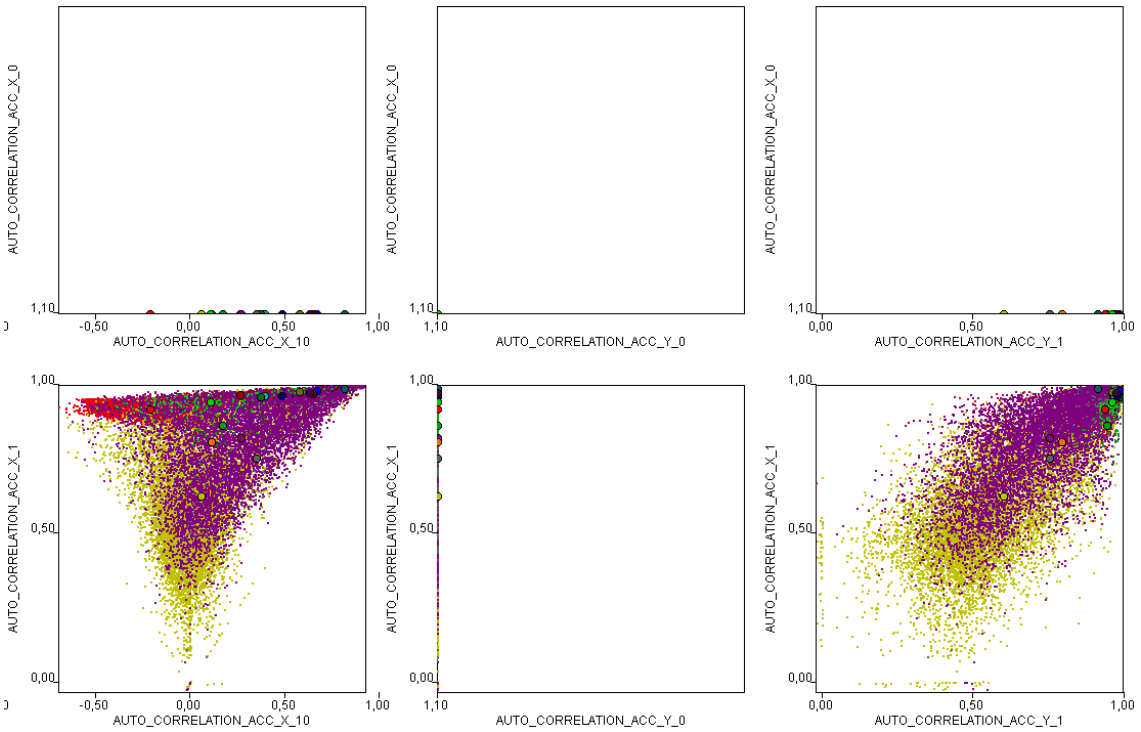


Abbildung 10: 1. Merkmal der Autokorrelation

Dieser Ausschnitt der Streudiagramm-Matrix zeigt: Das 1. Merkmal der Autokorrelation ist immer 1, wie hier für die Autokorrelation auf der X- und auf der Y-Achse des Akzelerometers zu sehen.

5 Zusammenfassung

Im Rahmen dieser Ausarbeitung wurde eine Möglichkeit geschaffen die Merkmale, die von der PGMAMKS-Applikation für das Training der Klassifikatoren berechnet werden, zu exportieren. Dies ermöglicht es die Merkmale in andere Anwendungen z. B. für Analysen oder zum Training von anderen Klassifikatoren einzusetzen. Hierbei ist allerdings zu beachten, dass die Auswahl der zu exportierenden Merkmale ausschließlich im Programmcode der PGMAMKS-Anwendung getroffen werden kann. Dies ist im wesentlichen dadurch begründet, dass die Nutzung im Rahmen der PGMAMKS so deutlich einfacher zu implementieren als auch zu nutzen ist. Hier bietet sich die Möglichkeit eine besser Lösung zu finden.

Als eine Möglichkeit der Nutzung der exportierten Daten wurde die in die Kommandozeilenschnittstelle der PGMAMKS-Applikation integrierte Streudiagramm-Matrix implementiert. Diese stellt ein einfaches Beispiel zur Visualisierung der Merkmale dar. Die Streudiagramm-Matrix hat allerdings auch einige Probleme, wie die Dimensionsproblematik und die unter Umständen schwierige Interpretation. Diese wurden in dieser Ausarbeitung aufgezeigt. Es gibt eine Reihe weiterer Möglichkeiten der Merkmalsvisualisierung, wie z. B. die Grand Tour, Projection Pursuit oder auch Methoden, die z. B. die Daten erst durch Verfahren wie multidimensionale Skalierung in darstellbare Dimensionen bringen [CBCH95] [Bas01]. Insofern ist auch hier ein großes Potential für alternative Lösungswege gegeben.

Zudem wurde unter Einsatz der vorgestellten Visualiserungsmethode ein Problem mit den aktuell genutzten Merkmalen und eine Möglichkeit zur Optimierung der Klassifikationsergebnisse (zumindest für die AMM) aufgezeigt.

Literatur

[aeq] Technologiegestützte Interventionen am Beispiel der Prävention des schleichenden funktionellen Abbaus im Alter. https://www.uni-oldenburg.de/versorgungsforschung/abteilungen/medizintechnik/medizintechnikforschung/projekte/aequipa/. Stand: 17.10.2016.

[Bas01] Wojciech Basalaj. Proximity visualisation of abstract data. Technical report, University of Cambridge - Computer Laboratory, 2001.

[CBCH95] Dianne Cookand, Andreas Buja, Javier Cabrera, and Catherine Hurley. Grand Tour and Projection Pursuit. *Journal of Computational and Graphical Statistics, 1 September 1995, Vol.4(3), pp.155-172*, 4(3):155, 1995.

[NBOH96] Matthias Nagel, Axel Benner, Rüdiger Ostermann, and Klaus Henschke. *Graphische Datenanalyse*. Gustav Fischer Verlag, 1996.

[unia] Technologiegestützte Interventionen am Beispiel der Prävention des schleichenden funktionellen Abbaus im Alter. https://www.uni-oldenburg.de/versorgungsforschung/abteilungen/medizintechnik/medizintechnikforschung/projekte/aequipa/. Stand: 17.10.2016.

[unib] Versa-Studie – Selbständigkeit wahren. https://www.uni-oldenburg.de/versa-studie/. Stand: 17.10.2016.

D.6. Gangerkennung und Einführung in die Ganganalyse

CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Gangerkennung und Einführung in die Ganganalyse

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Sebastian Warsitz

Oldenburg, den 7. April 2017

Abstract

Ziel des Dokumentes ist ein Überblick über Algorithmen zur Gangerkennung anhand von einfachen tragbaren Sensoren (wie z.B Accelerometer, Gyroskop, Magnetometer) zu geben und zu erörtern, welche sich im Kontext der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ am besten eignen. Dabei zeigte sich, dass die Gangerkennung in der Projektgruppe allgemeiner in Form einer Aktivitätserkennung durchgeführt werden sollte. Die Aktivitätserkennung sollte dann in der Projektgruppe mithilfe von Techniken des maschinellen Lernens umgesetzt werden. Des Weiteren wird eine Einführung in die Ganganalyse geboten, in der ein Gangzyklus und seine unterschiedlichen Phasen beschrieben werden. Es werden Parameter vorgestellt, welche für eine Ganganalyse betrachtet werden sollten. Für die Berechnung der Parameter ist es erforderlich einzelne Schritte eines Ganges zu erkennen, sodass die *Peak Detection* Methode zur Schritterkennung erläutert wird. Weiterhin wird beschrieben, wie die Schrittlänge aus Accelerometer-Daten berechnet werden kann, welcher ein wichtiger Parameter der Ganganalyse darstellt.

Inhalt

Abbildungen

Tabellen

1 Einleitung

Im Rahmen der Projektgruppe „Mobilitäts-Assessment mit körpernahen Sensoren“ an der Carl von Ossietzky Universität Oldenburg soll eine automatische Auswertung von Daten eines Sensorgürtels, welche während Assessments mit Senioren aufgenommen wurden, realisiert werden. Eine Hauptaufgabe auf diesem Weg ist die automatische Erkennung von Aktivitäten wie Gehen, Sitzen oder Liegen und die Erkennung der durchgeführten Assessment-Übungen. Insbesondere ist im Kontext der Projektgruppe die Erkennung und Analyse des Gehens von Interesse, da es Teil in vielen Assessment-Übungen ist und anhand einer Ganganalyse Aussagen über das Gangbild der Probanden gemacht werden können. Mithilfe des Gangbildes kann bei abweichender Norm ein Rehabilitätsprogramm erstellt werden [GN06, S.9].

In der Literatur werden Methoden wie z.B. verschiedene Schwellenwert-basierte Verfahren [BH13] oder Verfahren mithilfe von Techniken des maschinellen Lernens [ABMP+10] zur Gangerkennung vorgestellt. Wie in Kapitel 2 beschrieben wird, eignen sich Schwellenwert-basierte Verfahren im Kontext der Projektgruppe eher nicht für eine Gangerkennung. Verfahren mithilfe von Techniken des maschinellen Lernens hingegen können für eine Gangerkennung in der Projektgruppe benutzt werden, vor allem wenn sie mit der allgemeineren Form, der Aktivitätserkennung praktiziert werden.

Die Aktivitätserkennung anhand Daten von körpernahen Sensoren wird häufig mithilfe von Techniken des maschinellen Lernens durchgeführt, da sie komplexe Zusammenhänge in den Daten automatisch erlernen können und darüber hinaus eine sehr hohe Erkennungsgenauigkeit aufweisen [PEK+06].

Das Dokument stellt die Schritte bei der Aktivitätserkennung in Kapitel 2 vor und zeigt in Kapitel 3 auf, welche Techniken in der Projektgruppe zum Einsatz kommen sollten. In Kapitel 4 wird anschließend eine Einführung in die Ganganalyse gegeben. Dabei werden die einzelnen Phasen eines Gangzyklus und die zu betrachteten Parameter beschrieben. Anschließend wird die *Peak Detection* Methode für die Erkennung einzelner Schritte

präsentiert. Außerdem wird eine Methode vorgestellt, um die Schrittlänge eines erkannten Schrittes zu berechnen. Abgerundet wird das Dokument durch eine Zusammenfassung in Kapitel 5.

2 Gangerkennung

Methoden zur Gangerkennung werden in [BH13] vorgestellt. Dabei handelt es sich um verschiedene Schwellenwert-basierte Verfahren, welche Betrag, Autokorrelation oder die Kurzzeit-Fourier-Transformation aus Accelerometer-Daten nutzen. Diese werden berechnet und wenn sie über ein vorher festgelegter Schwellenwert liegen, so wird das für die Berechnung betrachtete Segment als *Gehen* identifiziert. Es lassen sich mithilfe dieser Verfahren Aktivitäten unterschiedlicher Intensitäten erkennen [ABMP+10]. Dies ist im Zusammenhang mit der Projektgruppe aber wenig sinnvoll, da während der Assessments unterschiedliche Aktivitäten derselben Intensität wie z.B. *Stehen* und *Sitzen* oder *Gehen* und *Treppe steigen* durchgeführt werden und diese mit Schwellenwert-basierte Verfahren nicht zuverlässig unterschieden werden können.

Eine Gangerkennung kann ebenfalls mithilfe von Techniken des maschinellen Lernens realisiert werden. Wie in Kapitel 1 bereits beschrieben, sollte dies in der allgemeineren Form, der Aktivitätserkennung, realisiert werden. Bei der Aktivitätserkennung mit körpernahen Sensoren geht es darum, verschiedene Aktivitäten wie z.B Gehen, Sitzen oder Liegen mithilfe von Sensoren, welche am Körper getragen werden, zu erkennen. Obwohl dabei unterschiedliche Techniken des maschinellen Lernens eingesetzt werden, sind die Schritte bei der Aktivitätserkennung in vielen relevanten Forschungsarbeiten gleich [BI04] [BGpK+] [SZL+10] [KLLK10] [ABMP+10] [PCLP13] [PFN06]. Diese Schritte sind in Abbildung 1 dargestellt und werden in den folgenden Abschnitten kurz beschrieben.

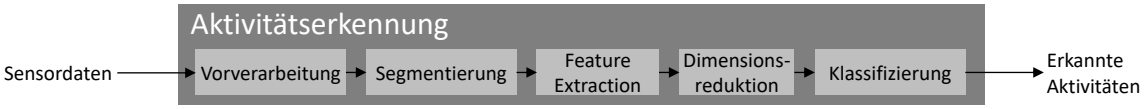


Abbildung 1: Prozess der Aktivitätserkennung

2.1 Vorverarbeitung

Eine Vorverarbeitung der aufgenommenen Sensordaten ist in einigen Fällen sinnvoll und steigert die Genauigkeit der Erkennung der Aktivitäten. Es können dabei folgende Vorverarbeitungsschritte angewendet werden.

Entfernung von Ausreißern. Um Ausreißer aus den Sensordaten zu entfernen, welche aus Mess- oder Übertragungsfehler resultieren, kann ein Mittelwert-Filter [KLLK10] verwendet werden. Dadurch werden die weiteren Prozessschritte nicht von diesen Datenfehlern beeinflusst. [BGpK$^+$]

Reduzierung von Rauschen. Um Hochfrequenzrauschen aus den aufgenommenen Sensordaten zu reduzieren, werden Tiefpass-Filter wie Medianfilter, Laplace-Filter oder Gauß-Filter verwendet. [ABMP$^+$10]

Rekonstruktion von Spitzen. Wenn der Wertebereich, welches ein Sensor erfassen kann, kleiner ist als tatsächlich auftretende Werte, müssen diese Werte rekonstruiert werden. Ein Verfahren zur Rekonstruktion dieser Werte ist in [BGpK$^+$] beschrieben.

Herausrechnung der Erdbeschleunigung. Die Erdbeschleunigung kann aus den aufgenommenen Sensordaten herausgerechnet werden. Zu diesem Zweck können Hochpass-Filter verwendet werden, welche zwischen Erdbeschleunigung und dynamischer Beschleunigung unterscheiden können. [ABMP$^+$10]

2.2 Segmentierung

Die Erkennung von Aktivitäten aus den vorverarbeiteten Sensordaten, welche sich über ein größeren Zeitraum erstrecken, ist schwierig. Aus diesem Grund werden die vorverarbeiteten Sensordaten in kleinere Abschnitte segmentiert. Die meisten Segmentierungsverfahren lassen sich in drei Gruppen kategorisieren: *Aktivitätsbasierte Fenster*, *Eventbasierte Fenster* und *Gleitendes Fenster*. [BGD$^+$14]

Aktivitätsbasierte Fenster. Bei diesem Verfahren werden die vorverarbeiteten Sensordaten basierend auf der Detektion von Aktivitätsänderungen segmentiert. Dabei können verschiedene Verfahren zur Detektion von Aktivitätsänderungen angewendet werden. Zum Beispiel lassen sich Aktivitätsänderungen durch Unterschiede in den Frequenzeigenschaften identifizieren. Um Aktivitätsänderungen besser detektieren zu können, kann mittels heuristischer Methoden zwischen statischen und dynamischen Aktivitäten unterschieden werden. [BGD+14]

Eventbasierte Fenster. Bei diesem Verfahren werden die vorverarbeiteten Sensordaten an identifizierten Events getrennt. Dieses Verfahren eignet sich besonders gut in der Ganganalyse, da das Aufsetzen der Ferse und Absetzen der Ferse beim Gehen mithilfe dieses Verfahrens detektiert werden können. [BGD+14]

Gleitendes Fenster. Bei diesem Verfahren werden die vorverarbeiteten Sensordaten in gleich große Segmente unterteilt, sodass es keine Überlappungen und Lücken zwischen den Segmenten gibt. In einigen Forschungsarbeiten wird das Verfahren auch mit Überlappungen zwischen benachbarten Segmenten verwendet. Das Gleitende Fenster Verfahren eignet sich besonders gut für die Erkennung von periodischen Aktivitäten (z.B Gehen, Laufen) und statischen Aktivitäten (z.B. Stehen, Sitzen). Für die Erkennung von sporadischen Aktivitäten (z.B Essen zubereiten) eignet sich dieses Verfahren nicht. [BGD+14]

2.3 Feature Extraction

Der nächste Schritt ist die für ein Segment charakteristischen Merkmale zu finden, welche diesen so genau wie möglich repräsentieren. Eine Auflistung von häufig in der Aktivitätserkennung verwendeten Merkmalen zeigt Tabelle 1.

Zeitdomäne	Mittelwert
	Standardabweichung (bzw. Varianz)
	Autokorrelation
	Anzahl Null-/Mittelwert-Durchgänge
Frequenzdomäne	Energie
	Entropy
Zeit- und Frequenzdomäne	Wavelet
Heuristische Merkmale	Signal Magnitude Area (SMA)
	Signal Vector Magnitude (SMV)
	Korrelation zwischen Achsen

Tabelle 1: Häufig verwendete Merkmale [ABMP+10]

Die Auswahl der zu berechnenden Merkmale wirkt sich signifikant auf die Genauigkeit der zu erkennenden Aktivitäten aus [ABMP+10]. Die Menge der ausgewählten Merkmale wird *Merkmalsvektor* genannt.

2.4 Dimensionsreduzierung

Das Ziel der Dimensionsreduzierung ist eine höhere Genauigkeit der Erkennung der Aktivitäten zu erzielen und den Berechnungsaufwand zu reduzieren. Es gibt dabei zwei Vorgehensweisen wie man die Dimension eines Merkmalsvektors reduzieren kann. [ABMP+10]

Auswahl von Merkmalen. Es wird ein neuer Merkmalsvektor erzeugt, welcher eine Teilmenge des Merkmalsvektors aus dem letzten Schritt ist. Der neue Merkmalsvektor besteht danach nur aus Merkmalen, welche aussagekräftig und nicht redundant sind. [ABMP+10]

Transformation von Merkmalen. Um die Dimension des Merkmalsvektors zu reduzieren, werden aus verschiedenen Merkmalen des Merkmalsvektor neue Merkmale berechnet

und die für die Berechnung verwendeten Merkmale aus dem Merkmalsvektor entfernt. Ein Verfahren hierfür ist die Hauptkomponentenanalyse (Principal Component Analysis (PCA)). Bei diesem Verfahren wird der Merkmalsvektor durch eine geringe Anzahl von Linearkombinationen der Merkmale vereinfacht. [ABMP+10]

2.5 Klassifizierung

Der abschließende Schritt ist, die Segmente mithilfe ihrer Merkmalsvektoren zu klassifizieren d.h. jedem einzelnen Segment eine Aktivität zuzuordnen. Dabei werden Algorithmen des maschinellen Lernens eingesetzt um Muster in den Merkmalsvektoren zu erkennen. In Forschungsarbeiten im Gebiet der Aktivitätserkennung werden oft Klassifizierer mit überwachten Lernverfahren verwendet, bei welcher vor der eigentlichen Mustererkennung eine Trainingsphase erfolgt. Beispiele für solche Verfahren, welche zur Erkennung von Aktivitäten verwendet werden können, sind **Enscheidungsbäume** [BI04] [HKG+06] [KNM+06], **Hidden-Markov-Modelle** [WLT05], **Gaussian-Mixture-Modelle** [AALC06], **Nächste-Nachbarn-Klassifikation** [HKG+06] [PFN06], **Naive-Bayes-Klassifikation** [HKG+06], **Support Vector Machine** [SZL+10] [PCLP13] oder **neuronale Netze** [KLLK10] [KWM11]. In [BI04] wurde festgestellt, dass sich für die Erkennung von alltäglichen Aktivitäten mithilfe von Accelerometern die Klassifizierung mittels Entscheidungsbäumen und die Nächste-Nachbarn-Klassifikation besonders gut eignen.

3 Anwendung der Aktivitätserkennung im Projekt

Die oben beschriebene Vorgehensweise zur Erkennung verschiedener Aktivitäten sollte im Projekt ebenfalls umgesetzt werden. Ein besonders interessanter Ansatz wurde dabei in [KLLK10] vorgestellt. Hier wurde ein hierarchisches Verfahren angewendet. Dies bedeutet, dass die zu erkennenden Aktivitäten zuerst in die drei Klassen *statische Aktivitäten*, *Transitionen* und *dynamische Aktivitäten* klassifiziert wurden und anschließend mithilfe von *künstlichen neuronalen Netzwerken* die eigentliche Aktivitätserkennung durchgeführt wurde. Dabei wurden die *künstlichen neuronalen Netzwerke* für die drei Klassen mit unterschiedlichen Trainingsdaten trainiert.
Ein hierarchischer Ansatz wäre im Projekt ebenfalls sinnvoll. Abbildung 2 stellt eine mögliche Vorgehensweise vor.

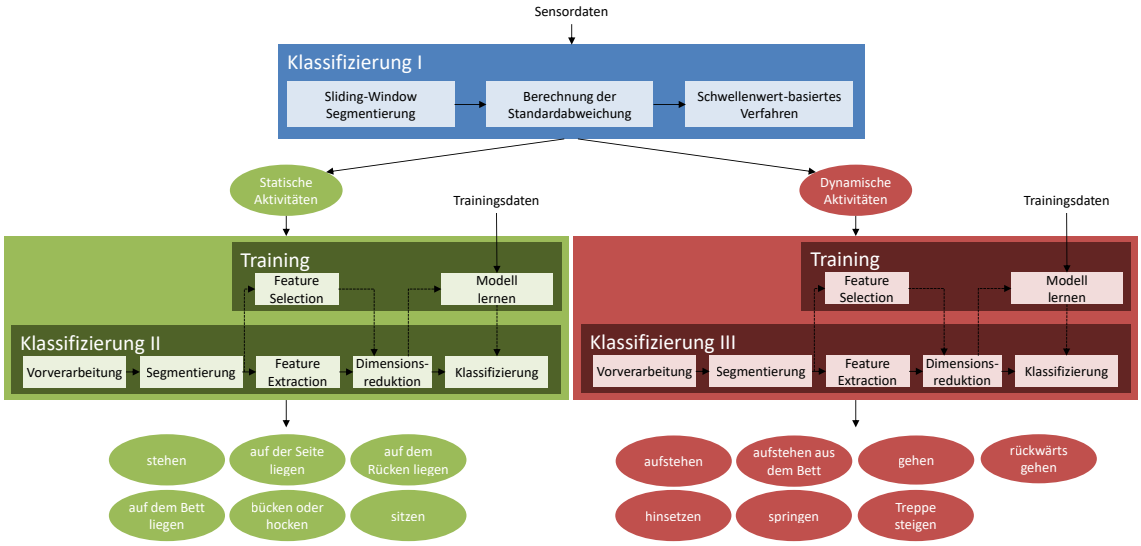


Abbildung 2: Umsetzung im Projekt

Zuerst werden die aufgenommen Sensordaten mithilfe eines Schwellenwert-basierten

Verfahrens in statische Aktivitäten und dynamische Aktivitäten segmentiert. Dabei wird für die Segmentierung das *Gleitende Fenster* Verfahren verwendet und anschließend für jedes Segment die Standardabweichung bestimmt. Ist die Standardabweichung für das Segment größer als ein zu bestimmender Schwellenwert, wird das Segment in die Klasse der dynamische Aktivitäten klassifiziert, ansonsten in die Klasse der statischen Aktivitäten. Danach können die statischen und dynamischen Aktivitäten bei der Erkennung der Aktivitäten getrennt betrachtet werden, was die Genauigkeit der Aktivitätserkennung erhöhen sollte. Es müssen für beide Klassen die beste Fenstergröße, die besten Merkmale und der beste Klassifizierer gefunden werden, um die höchstmögliche Genauigkeit zu erreichen. Diese sind vom jeweiligen Anwendungsfall abhängig und daher können die Parameter im Kontext der Projektgruppe nicht fundiert vorausgesagt werden und müssen durch eine geeignete Evaluation bestimmt werden. Als Klassifizierer sollten unbedingt Entscheidungsbäume und die Nächste-Nachbarn-Klassifikation betrachtet werden, da sich diese für eine Aktivitäteserkennung besonders gut eignen sollen [BI04].

4 Einführung in die Ganganalyse

4.1 Gangzyklus und seine Phasen

Der Begriff **Gangzyklus** bezeichnet den Zeitraum, der zwischen zwei aufeinanderfolgenden initialen Bodenkontakte desselben Fußes während des Gehens liegt. Der initiale Bodenkontakt zu Beginn des Gangzyklus wird als 0%-Punkt des Gangzyklus bezeichnet. Der Zeitpunkt des nächsten Bodenkontakt desselben Fußes markiert das Ende eines Gangzyklus und wird daher auch 100%-Punkt bezeichnet. Das andere Bein vollzieht währenddessen die gleiche Bewegungsabfolge, allerdings um einen halben Gangzyklus verschoben. [GN06]

Ein Gangzyklus wird in 8 Phasen unterteilt. Für die Bezeichnung der Phasen existieren in der Literatur zwei verschiedene Terminologie-Systeme, die traditionelle Terminologie und die Rancho Los Amigos Terminologie. Dieses Dokument verwendet die Rancho Los Amigos Terminologie, da sie die Phasen eindeutiger bezeichnet [GN06]. Die Phasen des Gangzyklus sind in Abbildung 3 dargestellt und werden nun kurz erläutert.

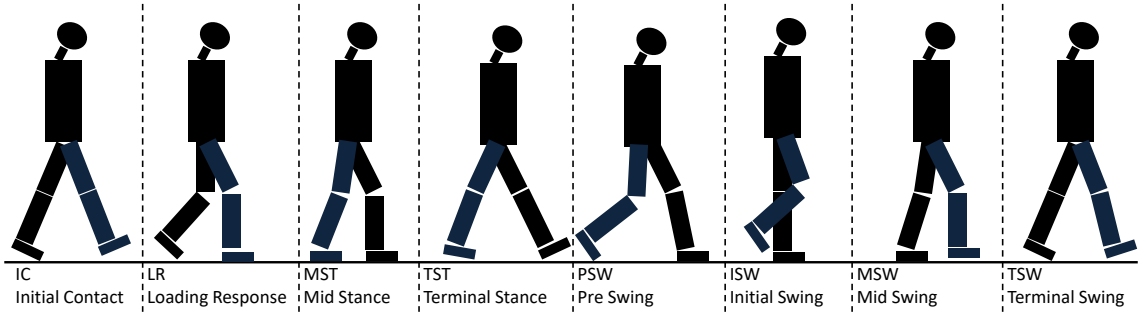


Abbildung 3: Phasen eines Gangzyklus [Str]

1. **initialer Kontakt** (Initial contact, 0%): Diese Phase umfasst nur den Moment, in der der Fuß initialen Kontakt zum Boden hat.
2. **Stoßdämpfungsphase** (Loading response, 0%-12%): Diese Phase beginnt mit dem initialen Bodenkontakt des Fußes und endet, wenn der kontralaterale Fuß zum Schwung

holen angehoben wird.

3. **Mittlere Standphase** (Mid stance, 12%-31%): Diese Phase beginnt, wenn der kontralaterale Fuß angehoben wird und endet mit der Anhebung der Ferse.
4. **Terminale Standphase** (Terminal stance, 31%-50%): Diese Phase beginnt mit der Anhebung der Ferse und endet mit dem initialen Kontakt des kontralateralen Fußes zum Boden.
5. **Vorschwungphase** (Pre-swing, 50%-62%): Die Phase beginnt mit dem initialen Kontakt des kontralateralen Fußes und endet, wenn die Fußspitze vom Boden angehoben wird.
6. **initiale Schwungphase** (Initial swing, 62%-75%): Die Phase beginnt sobald die Fußspitze vom Boden angehoben wurde und endet, wenn der Fuß sich gegenüber des anderen Fußes befindet.
7. **mittlere Schwungphase** (Mid swing, 75%-87%): Sobald sich der Fuß gegenüber des anderen Fußes befindet, wird in diese Phase übergegangen. Die Phase endet, wenn sich das Schienbein in vertikaler Position befindet.
8. **terminale Schwungphase** (Terminal swing, 87%-100%): Die letzte Phase beginnt, wenn sich das Schienbein in vertikaler Position befindet und endet, sobald der Fuß wieder auf dem Boden auftritt.

Die ersten fünf Phasen werden auch zusammengefasst als **Standbeinphase** bezeichnet, da der Fuß während dieser Phase Bodenkontakt hat. Sie macht bei einem normalen Gang in etwa 60-62% des Gangzyklus aus. Die anderen drei Phasen werden zusammengefasst als **Schwungbeinphase** bezeichnet. Der Fuß befindet sich während dieser Phase in der Luft. Die Schwungbeinphase macht in etwa 38-40% des Gangzyklus eines normalen Ganges aus. [Kir06] [GN06]

Da die Standbeinphasen für jeweils beide Gliedmaßen 60-62% in Anspruch nehmen, folgt daraus, dass beide Füße für 20% des Gangzyklus gleichzeitig Kontakt zum Boden haben. Dieser Zeitraum wird daher auch **Zweibeinstandphase** (double support) genannt. Es gilt, dass die Dauer der Zweibeinstandphase mit höherer Ganggeschwindigkeit abnimmt bis sie schließlich beim Laufen 0% beträgt. [Kir06]

4.2 Zeitlich/räumliche Parameter eines Gangzyklus

Um eine grundsätzliche Beschreibung über das Gangbild eines Menschen zu erhalten, können zeitliche und räumliche Parameter eines Gangzyklus betrachtet werden [Kir06]. Im Folgenden werden die wichtigsten Parameter kurz aufgelistet und beschrieben.

Doppelschrittdauer (Stride time). Die Doppelschrittdauer bezeichnet die Dauer, die für einen Doppelschritt benötigt wird. Die Zeitmessung beginnt dabei beim Aufsetzen auf dem Boden der Ferse und endet beim nächsten Aufsetzen derselben Ferse. Ein Doppelschritt entspricht also einem Gangzyklus.

Schrittdauer (Step time). Die Schrittdauer bezeichnet die Dauer, die für einen einzelnen Schritt benötigt wird. Die Zeitmessung beginnt dabei beim Aufsetzen der einen Ferse und endet beim Aufsetzen der anderen Ferse auf dem Boden.

Dauer in der Standbeinphase (Stance time). Dieser Parameter bezeichnet die Dauer innerhalb eines Gangzyklus, in der sich ein Gliedmaße in der Standbeinphase befindet.

Dauer in der Schwungbeinphase (Swing time). Dieser Parameter bezeichnet analog die Dauer innerhalb eines Gangzyklus, in der sich ein Gliedmaße in der Schwungbeinphase befindet.

Dauer in der Zweibeinstandphase (Double limb time). Dieser Parameter bezeichnet die Dauer innerhalb eines Gangzyklus, in der beide Füße gleichzeitig Kontakt zum Boden haben.

Geschwindigkeit (Speed). Die Geschwindigkeit bezeichnet die zurückgelegte Distanz pro Zeiteinheit. In der Ganganalyse wird dies häufig in Meter pro Sekunde (m/s) gemessen.

Kadenz (Cadence). Die Kadenz bezeichnet die Anzahl der Schritte pro Zeiteinheit. Als Einheit wird oft Schritte pro Minute gewählt.

Schrittlänge (Step length). Die Schrittlänge bezeichnet den Abstand zwischen dem Auftreten der linken und rechten Ferse (siehe Abbildung 4). Bei einem normalen Gang sollte die Schrittlänge bei einem rechten und einem linken Schritt gleich groß sein [VDO92, S.12].

Doppelschrittlänge (Stride length). Die Doppelschrittlänge bezeichnet den Abstand zwischen dem Auftreten der Verse desselben Fußes (siehe Abbildung 4).

Spurbreite (Step width). Die Spurbreite bezeichnet den seitlichen Abstand der beiden Fersen zueinander (siehe Abbildung 4).

Winkel des Fußes (Foot angle). Dieser Parameter bildet die Positionierung des Fußes beim Aufsetzen des Fußes auf dem Boden ab. Dabei wird eine gerade Linie von der Ferse in Laufrichtung gezogen und eine Linie von der Ferse in Richtig des zweiten Zehs. Anschließend wird der Winkel der beiden Linien bestimmt (siehe Abbildung 4).

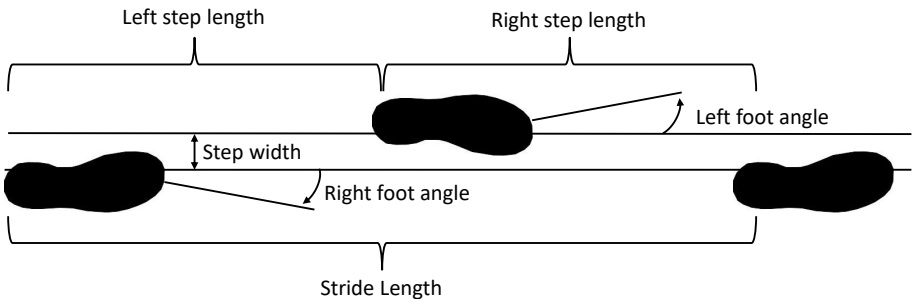


Abbildung 4: räumliche Parameter eines Gangzyklus [VDO92, S.12]

Um die oben beschriebenen Parameter berechnen zu können, ist es zwingend erforderlich nicht nur Phasen des Gehens in den Sensordaten zu erkennen, sondern auch die einzelnen Schritte.

4.3 Erkennung eines Schrittes

Für die Berechnung der in Abschnitt 4.2 vorgestellten Parameter müssen die einzelnen Schritte der bereits erkannten Gehen-Segmente identifiziert werden. In der Literatur werden einige unterschiedliche Methoden zur Schritterkennung vorgestellt wie *Peak De-*

tection [kJHP04] [RDM03], *Zählung der positiven Nullübergänge* [GRSK11] [Bea06], *Normalized Auto-correlation based Step Counting (NASC)* [MTN+11] [12], *Wavelet Transformation* und einige Andere [BH13]. Ein Vergleich dieser Methoden wurde in [BH13] durchgeführt. Dabei wurden die Accelerometer-Daten eines Smartphones verwendet, welcher beliebig am Körper getragen werden durfte. Demnach zeigte die *Peak Detection* Methode die höchste Genauigkeit bei der Schritterkennung.

In [TMN+12] wurde ein dreidimensionaler Accelerometer verwendet, welcher im unteren Rückenbereich der Probanden angebracht wurde. Dies entspricht in etwa dem Szenario der Projektgruppe, in welcher die Sensoren in einem Gürtel integriert sind. Anschließend wurde auf den Accelerometer-Daten die *Peak Detection* Methode durchgeführt um eine Schritterkennung zu realisieren. Diese Methode wird im Folgenden beschrieben.

Wie in Abschnitt 4.1 beschrieben, kann ein Gangzyklus in Standbein- und Schwungbeinphase unterteilt werden. Bei einem normalen Gang ist es so, dass zu Beginn der Standbeinphase die Ferse am Boden aufschlägt. Der Impuls, der durch die Kollision der Ferse mit dem Boden entsteht, wird über das Bein zur Körpermitte übertragen, was zu einer kurzen ruckartigen Bewegung der Körpermitte führt. Folglich registriert der getragene Accelerometer ein starkes Signal im Moment des Aufsetzens der Ferse. Innerhalb eines Gangzyklus werden zwei solch starke Signale registriert für den linken und den rechten Fuß, dargestellt in Abbildung 5.

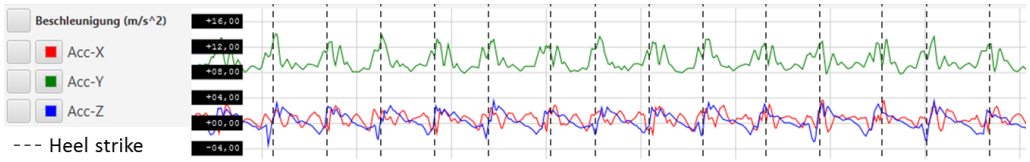


Abbildung 5: Accelerometer-Signal eines Ganges [TMN+12]

Folglich kann das Ereignis des Aufsetzen der Ferse auf dem Boden verwendet werden, um einzelne Schritte aus den bereits erkannten Segmenten des Gehens zu erkennen. Dabei wird aus den Daten des dreidimensionalen Accelerometers die Wahrscheinlichkeit des Aufsetzen der Ferse für jeden Zeitpunkt berechnet. Die Berechnung der Wahrscheinlichkeit des Aufsetzen der Ferse basiert auf zwei Beobachtungen in den Daten des Accelerometers:

- Die Dichte von lokalen Extrempunkten in allen drei Dimensionen des Accelerometers ist relativ hoch.
- Die Energie der Accelerometer-Signale ist relativ hoch.

Um für die Dichte der lokalen Extrempunkten eine robuste Berechnung zu garantieren,

werden die Accelerometer-Daten zunächst mithilfe verschiedener Gauß-Filter und Glättungslevel geglättet. Danach werden alle Extrempunkte für jeden Glättungslevel in den geglätteten Accelerometer-Daten markiert (siehe Abbildung 6(b)). Abschließend wird die Wahrscheinlichkeitsdichtefunktion der markierten Extrempunkte p_f^t zum Zeitpunkt t mithilfe eines Kerndichteschätzers berechnet (siehe Abbildung 6(c)).
Als zweites wird die Energie der Accelerometer-Daten als Wahrscheinlichkeit für das Aufsetzen der Ferse am Boden betrachtet. Die Energie e_t zum Zeitpunkt t ist definiert als der Betrag aller drei Accelerometer-Daten zum Zeitpunkt t. Um auch hier eine bessere Robustheit zu garantieren, werden verschiedene geglättete Energiewerte $e_{w_l,t}$ mit Glättungsparameter w_l und Glättungslevel l (siehe Abbildung 6(d)) berechnet. Die Wahrscheinlichkeit des Aufsetzen der Ferse basierend auf die berechneten Energiewerte ist $p_t^e = p * \prod_l e_{w_l,t}$ wobei p ein Skalierungsfaktor ist.

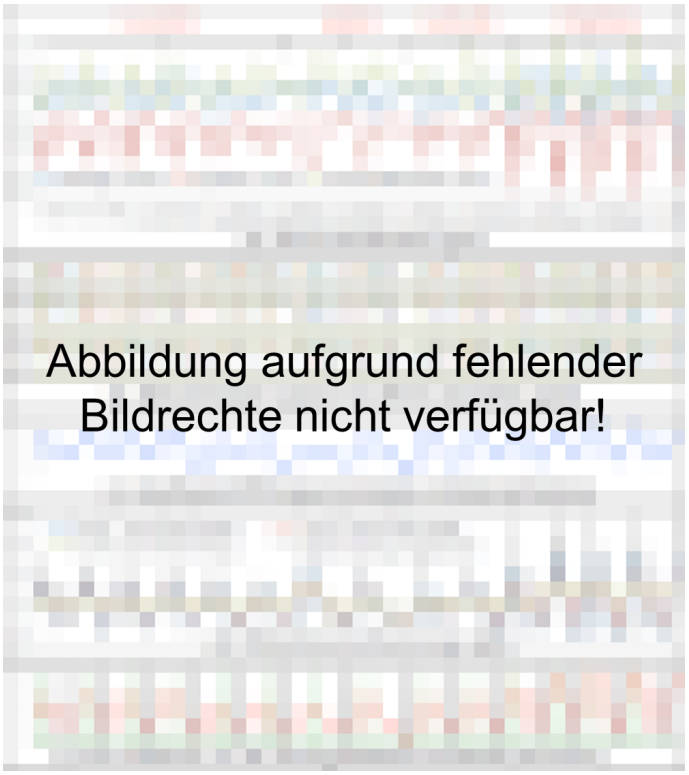


Abbildung 6: Algorithmus für die Schritterkennung [TMN+12]

Im letzten Schritt wird das Produkt beider berechneter Wahrscheinlichkeiten p_f^t und p_t^e gebildet, um eine Wahrscheinlichkeitsfunktion für das Aufsetzen der Ferse am Boden zu erhalten:

$$p_t = p_f^t * p_t^e \tag{4.1}$$

Die lokalen Hochpunkte dieser Wahrscheinlichkeitsfunktion markieren näherungsweise die Zeitpunkte für das Aufsetzen der Ferse am Boden (siehe Abbildung 6(e)).

4.4 Berechnung der Schrittlänge und der zurückgelegten Distanz

Die Schrittlänge ist ein wichtiger Parameter in der Ganganalyse. In [SPK+07] wird eine Methode zur Berechnung der Schrittlänge vorgestellt. Es wurde ein zweidimensionaler Accelerometer verwendet, welcher beliebig am Körper angebracht werden kann. Nachdem zunächst eine Schritterkennung durchgeführt wurde, wird die Schrittlänge jedes einzelnen Schrittes berechnet und diese aufsummiert, um die zurückgelegte Distanz zu erhalten. Die Schrittlänge wird dabei durch eine lineare Regression wie folgt berechnet:

$$a * f + b * v + c \tag{4.2}$$

mit den Parameter a, b und c, welche in einer Trainingsphase gelernt oder konfiguriert werden müssen. Die Schrittlänge ist abhängig von der Gangfrequenz f_i und der Varianz des Accelerometers v_i. Diese ergeben sich zu

$$f_i = \frac{1}{t_i - t_{i-1}} \tag{4.3}$$

$$v_i = \sum_{t=t_{i-1}}^{t_i} (\frac{a_t - \bar{a}_i)^2}{N}) \tag{4.4}$$

wobei t_i die Zeit des i-ten erkannten Schrittes bezeichnet, a_t den Wert des Accelerometers zum Zeitpunkt t, $\bar{a}_i$ den Durchschnitt der Werte des Accelerometers während des i-ten Schrittes und N die Anzahl der Samples des Accelerometers während des Schrittes.
Die zurückgelegte Distanz ergibt sich dann aus der Summe aller Schritte

$$\sum_{i=1}^{n}(a * f_i + b * v_i + c) \tag{4.5}$$

wobei n die Anzahl der erkannten Schritte ist.
Das lineare Regressionsmodell kann nicht gleichzeitig für Gehen und Laufen angewendet werden, da die Berechnung der Schrittlänge sonst zu ungenau wird. Gehen und Laufen müsste getrennt mit zwei unterschiedlichen linearen Regressionsmodellen betrachtet werden [SPK+07]. Im Kontext der Projektgruppe wird allerdings nur die Aktivität des Gehens betrachtet, so dass dies hier nicht notwendig ist.

5 Zusammmenfassung

In diesem Dokument wurden zunächst Methoden zur Gangerkennung kurz vorgestellt und beschrieben, warum nur die Aktivitätserkennung mithilfe von Techniken des maschinellen Lernens sinnvoll für den Einsatz in der Projektgruppe ist. Die einzelnen Schritte der Aktivitätserkennung mithilfe von Techniken des maschinellen Lernens sind dabei Vorverarbeitung, Segmentierung, Feature Extraction, Dimensionsreduzierung und Klassifizierung. Anschließend wurde eine Einführung in die Ganganalyse gegeben. Bei der Ganganalyse werden einzelne Gangzyklen eines Ganges betrachtet, wobei ein Gangzyklus als der Zeitraum zwischen zwei aufeinanderfolgenden initialen Bodenkontakten desselben Fußes definiert ist. Ein Gangzyklus kann in Standbeinphase (Fuß auf dem Boden) und Schwungbeinphase (Fuß in der Luft) unterteilt werden, wobei die Standbeinphase 60% und die Schwungbeinphase 40% des Gangzyklus ausmachen. Zudem gibt es noch eine Zweibeinstandphase, in welcher beide Füße Kontakt zum Boden haben. Diese macht etwa 20% des Gangzyklus aus. Aus einem Gangzyklus lassen sich für die Ganganalyse zeitliche Parameter (Doppelschrittdauer, Schrittdauer, Dauer in der Standbeinphase, Dauer in der Schwungbeinphase, Dauer in der Zweistandphase, Geschwindigkeit, Kadenz) und räumliche Parameter (Schrittlänge, Doppelschrittlänge, Spurbreite, Winkel des Fußes) gewinnen. Um diese Parameter berechnen zu können müssen einzelne Schritte eines Ganges erkannt werden und so wurde die *Peak Detection* Methode zur Schritterkennung beschrieben. Dabei wird der Zeitpunkt des Aufsetzen der Ferse auf dem Boden genutzt, da die Kollision mit dem Boden ein starkes Signal in den Accelerometer-Daten erzeugt.
Zum Schluss wurde ein Algorithmus zur Berechnung der Schrittlänge aus Accelerometer-Daten vorgestellt. Dabei wurde festgestellt, dass die Schrittlänge abhängig ist von der Gangfrequenz und der Varianz der Accelerometer-Daten. Mithilfe der erkannten Schritte und berechneten Schrittlänge konnte auch die zurückgelegte Distanz bestimmt werden.

Literatur

[12] Venkat Padmanabhan Rijurekha Sen Krishna Chintalapudi , Krishna Kant Chintalapudi. Zee : Zero-effort crowdsourcing for indoor localization. In *Mobicom*, August 2012.

[AALC06] F. R. Allen, E. Ambikairajah, N. H. Lovell, and B. G. Celler. An adapted gaussian mixture model approach to accelerometry-based movement classification using time-domain features. In *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, pages 3600–3603, Aug 2006.

[ABMP+10] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *Architecture of Computing Systems (ARCS), 2010 23rd International Conference on*, pages 1–10, Feb 2010.

[Bea06] S. Beauregard. A helmet-mounted pedestrian dead reckoning system. In *Applied Wearable Computing (IFAWC), 2006 3rd International Forum on*, pages 1–11, March 2006.

[BGD+14] Oresti Banos, Juan-Manuel Galvez, Miguel Damas, Hector Pomares, and Ignacio Rojas. Window size impact in human activity recognition. *Sensors*, 14(4):6474, 2014.

[BGpK+] Thomas Bernecker, Franz Graf, Hans peter Kriegel, Christian Moennig, and Christoph Tuermer. 1 activity recognition on 3d accelerometer data (technical report).

[BH13] Agata Brajdic and Robert Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 225–234, New York, NY, USA, 2013. ACM.

[BI04] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. pages 1–17. Springer, 2004.

[GN06] Kirsten Götz-Neumann. *Gehen verstehen - Ganganalyse in der Physiotherapie*. Georg Thieme Verlag, 2 edition, 2006.

[GRSK11] P. Goyal, V. J. Ribeiro, H. Saran, and A. Kumar. Strap-down pedestrian dead-reckoning system. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pages 1–7, Sept 2011.

[HKG+06] E. A. Heinz, K. S. Kunze, M. Gruber, D. Bannach, and P. Lukowicz. Using wearable sensors for real-time recognition tasks in games of martial arts - an initial experiment. In *2006 IEEE Symposium on Computational Intelligence and Games*, pages 98–102, May 2006.

[Kir06] Chris Kirtley. *Clinical Gait Analysis - Theory and Practise*. Elsevier, 2006.

[kJHP04] Jeong W. kim, Han Jin Jang, Dong-Hwan Hwang, and Chansik Park. A Step, Stride and Heading Determination for the Pedestrian Navigation System. In *Presented at GNSS 2004, Sydney, Australia*, December 2004.

[KLLK10] A. M. Khan, Y. K. Lee, S. Y. Lee, and T. S. Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1166–1172, Sept 2010.

[KNM+06] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler. Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):156–167, Jan 2006.

[KWM11] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, March 2011.

[MTN+11] Yasushi Makihara, Ngo Thanh Trung, Hajime Nagahara, Ryusuke Sagawa, Yasuhiro Mukaigawa, and Yasushi Yagi. *Phase Registration of a Single Quasi-Periodic Signal Using Self Dynamic Time Warping*, pages 667–678. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[PCLP13] Zhaoqin Peng, Chun Cao, Qiusheng Liu, and Wentao Pan Pan. Human walking pattern recognition based on kpca and svm with ground reflex pressure signal. *Mathematical Problems in Engineering*, 2013, 2013.

[PEK+06] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen. Activity classification using realistic data from wearable sensors. *IEEE*

Transactions on Information Technology in Biomedicine, 10(1):119–128, Jan 2006.

[PFN06] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. Feature selection and activity recognition from wearable sensors. In *Proceedings of the Third International Conference on Ubiquitous Computing Systems*, UCS'06, pages 516–527, Berlin, Heidelberg, 2006. Springer-Verlag.

[RDM03] C. Randell, C. Djiallis, and H. Muller. Personal position measurement using dead reckoning. In *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, pages 166–173, Oct 2003.

[SPK+07] S. H. Shin, C. G. Park, J. W. Kim, H. S. Hong, and J. M. Lee. Adaptive step length estimation algorithm using low-cost mems inertial sensors. In *2007 IEEE Sensors Applications Symposium*, pages 1–5, Feb 2007.

[Str] Streifenender. The eight phases of human gait cycle.

[SZL+10] Lin Sun, Daqing Zhang, Bin Li, Bin Guo, and Shijian Li. *Activity Recognition on an Accelerometer Embedded Mobile Phone with Varying Positions and Orientations*, pages 548–562. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[TMN+12] N. T. Trung, Y. Makihara, H. Nagahara, Y. Mukaigawa, and Y. Yagi. Inertial-sensor-based walking action recognition using robust step detection and inter-class relationships. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3811–3814, Nov 2012.

[VDO92] Christopher L. Vaughan, Brian L. Davis, and Jeremy C. O'Conner. *Gehen verstehen - Ganganalyse in der Physiotherapie*. Kiboho Publishers, 2 edition, 1992.

[WLT05] Jamie A. Ward, Paul Lukowicz, and Gerhard Tröster. Gesture spotting using wrist worn microphone and 3-axis accelerometer. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, sOc-EUSAI '05, pages 99–104, New York, NY, USA, 2005. ACM.

D.7. Gangparameter

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI - Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Gangparameter

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Christin Poloczek

Oldenburg, den 22.01.2017

Abstract

Die vorliegende Seminararbeit soll einen Überblick über den Gangzyklus mit seinen acht Gangphasen und anschließend den gangspezifischen Parametern geben. Bei diesen handelt es um die Gehgeschwindigkeit, der Doppelschrittlänge und Schrittlänge, der Doppelschrittdauer sowie Schrittdauer, die Dauer der einzelnen Gangphasen, die Kadenz, Spurbreite und Symmetrie. Im Laufe dieser Seminararbeit wird deutlich, dass diese Parameter, außer der Spurbreite, sehr zuverlässig mithilfe von Accelerometern und einer bereits implementierten Schritterkennung bestimmt werden können. Außerdem wird mithilfe einer japanischen Projektarbeit die Genauigkeit einer Ganganalyse bei unterschiedlichen Positionen des Sensors betrachtet.

Inhalt

Abbildungen

Tabellen

1 Einleitung

Im Rahmen der Projektgruppe „Mobilitäts-Assessment mit körpernahen Sensoren“ an der Carl von Ossietzky Universität Oldenburg soll eine Software entwickelt werden, die Daten eines Sensorgürtels automatisch auswertet. Bei diesen Daten handelt es sich um Aufnahmen die während mehreren unterschiedlichen Assessments mit Senioren als Probanden aufgenommen wurden. Ein wesentlicher Aspekt der automatischen Auswertung liegt darin, dass Gangparameter bestimmt und an den Nutzer der Software ausgegeben werden sollen. Mithilfe solcher Parameter kann anschließend der Gang analysiert und auf Abweichungen zu der Norm untersucht werden. In dieser Arbeit wird zunächst in Kapitel 2 auf den Gangzyklus mit seinen charakteristischen Phasen eingegangen. Danach findet sich im Kapitel 3 eine Auflistung der typischen Gangparameter mit ihrer jeweiligen Beschreibung und mögliche Berechnungen. Im vierten Kapitel wird ein Projekt der NTT Corporation und der Wacoal Corporation aus Japan betrachtet, bei dem mithilfe von Accelerometern, statistischen und gangspezifischen Merkmalen überprüft wurde, ob die Probandin einen sogenannten „schönen Gang“ vollführt hat. Im letzten Kapitel wird schließlich darauf eingegangen, ob und wie sich Gangparameter im Zuge der Projektarbeit umsetzen lassen.

2 Gangzyklus

Der natürliche Gang des Menschen ist ein zyklischer Vorgang, der in der Literatur mithilfe unterschiedlicher Terminologien beschrieben wird. Im Folgenden wird von der Rancho Los Amigos – Terminologie ausgegangen, die vor allem im klinischen Umfeld Gebrauch findet und die Phasen eindeutiger beschreibt, als die traditionelle Terminologie (vgl. Götz-Neumann).

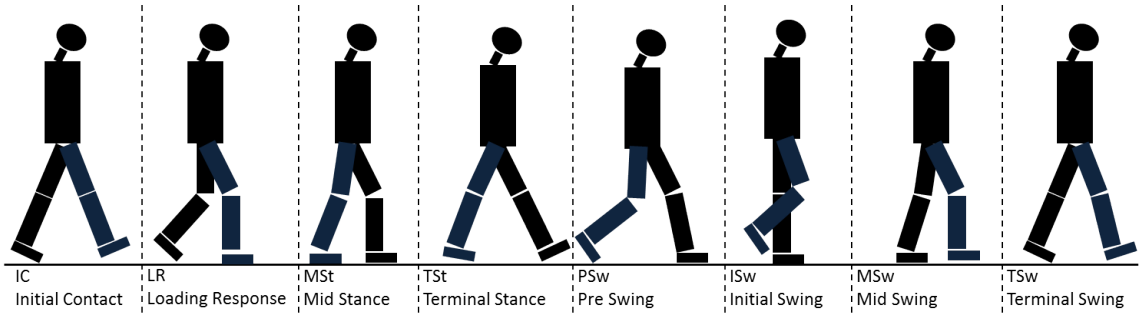


Abbildung 1: Gangzyklus mit den einzelnen Phasen (Vgl. [Neu13])

Laut Götz-Neumann wird als Gangzyklus derjenige Zeitraum definiert, der zwischen zwei aufeinander folgenden initialen Kontakten mit dem Boden desselben Fußes liegt. Auch andere sich wiederholende Ereignisse des Gehens, die eindeutig sind, könnten zur Begriffsdefinition herangezogen werden. Da sich jedoch der initiale Fußkontakt mit dem Boden sehr gut beobachten lässt, sowohl mit dem Auge als auch mit Sensoren wie dem Accelerometer, wird in der Regel der Beginn eines Gangzyklus durch den initialen Kontakt festgelegt. Der Gangzyklus erfüllt dabei drei wesentliche funktionelle Aufgaben: Die der Gewichtsübernahme, dem Einbeinstand und der Vorwärtsbewegung des Schwungbeines. In der Abbildung 1 wird der Zyklus des rechten Beines (Referenzbein) betrachtet, der nun näher mit den einzelnen Phasen und ihrer jeweiligen Funktion beschrieben wird. Gleichzeitig vollzieht das andere Bein (kontralaterales Bein) die gleichen Bewegungsabläufe wie das Referenzbein, jedoch um den Ablauf eines halben Gangzyklus verschoben.

Die entsprechenden korrespondierenden Phasen beider Beine werden in der Tabelle 1 gegenübergestellt. [GN06]

1. Die Phase des initialen Kontakts (engl. initial contact, Abk.: IC) beschreibt lediglich den Moment, in dem der Fuß des Referenzbeines den ersten Kontakt mit dem Boden hat und dient der Vorbereitung für die Stoßdämpfung des Schrittes.

2. Die Stoßdämpfungsphase (engl. loading response, Abk.: LR) beginnt mit dem initialen Kontakt und endet in dem Augenblick, in dem der in der Abbildung 1 hintere (kontralaterale) Fuß den Boden verlässt. Diese Phase dient der Stoßdämpfung, der Gewährleistung der gewichtstragenden Beinstabilität und dem Erhalt der Vorwärtsbewegung.

3. Die mittlere Standphase (engl. mid stance, Abk.: MSt) gewährleistet die Vorwärtsbewegung bis über den Vorfuß, der die Mittelfußknochen und die Zehen beinhaltet, und die Bein- sowie Rumpfstabilität. Diese Phase beginnt mit dem Abheben des kontralateralen Fußes und endet, wenn die Ferse des Fußes vom Referenzbein abhebt.

4. Die Phase des terminalen Standes (engl. terminal stance, Abk.: TSt) beginnt mit dem Abheben der Ferse und endet mit dem initialen Kontakt des hinteren Beines. Diese Phase dient der Vorwärtsbewegung des Körpers über das stützende Bein (Standbein) hinaus.

5. Die Vorschwungphase (engl. pre-swing, Abk.: PSw) hat als spezifische Leistung die vorbereitende Positionierung für die Schwungphase. Die Vorschwungphase beginnt mit dem ersten Kontakt des hinteren Fußes und endet mit dem Abheben des Vorfußes vom Boden.

6. Die initiale Schwungphase (engl. initial swing, Abk.: ISw) beginnt in dem Augenblick des kompletten Abhebens des Fußes vom Boden und endet, wenn, lateral gesehen, sich beide Füße hintereinander befinden, ähnlich dem Endzustand der mittleren Standphase. Die initiale Schwungphase dient dem Abheben des Fußes vom Boden und dem Nachvornebringen des Beines.

7. Die mittlere Schwungphase (engl. mid swing, Abk.: MSw) beginnt, wenn sich beide Füße lateral gesehen hintereinander liegen. Sie wird beendet sobald sich das Schienbein (lat. Tibia) in vertikaler Position befindet. Auch diese Phase dient dem Nachvornebringen des Beines, darüber hinaus wird der Abstand zwischen Fuß und Boden gewährleistet.

8. Die terminale Schwungphase (engl. terminal swing, Abk.: TSw) beginnt bei der verti-

kalen Position des Schienbeines und wird mit dem Auftreten des Fußes auf den Boden beendet. Diese Phase dient der Vollendung des Nachvornebringens des Beines und Vorbereitung auf den bevorstehenden initialen Kontakt.

Tabelle 1: Korrespondierende Phasen des Referenz- und kontralateralen Beines (vgl. [GN06])

Referenzbein	Kontralaterales Bein
Initialer Kontakt / Stoßdämpfungsphase	Vorschwungphase
Mittlere Standphase	Initiale und mittlere Schwungphase
Terminale Standphase	Terminale Schwungphase
Vorschwungphase	Initialer Kontakt / Stoßdämpfungsphase
Initiale Schwungphase	Mittlere Standphase
Mittlere Schwungphase	Mittlere Standphase
Terminale Schwungphase	Terminale Standphase

Des Weiteren werden die ersten fünf Phasen in der Literatur von Götz-Neumann und Winter et al. zu der Standbeinphase zusammengefasst, da sich der Fuß währenddessen in Bodenkontakt befindet. Die Standbeinphase beschreibt etwa 62 Prozent des Gangzyklus. Die restlichen 38 Prozent werden durch die sogenannte Schwungbeinphase beschrieben. Hierbei handelt es sich um eine Zusammenfassung der Phasen, bei denen sich der Fuß des Referenzbeines in der Luft befindet. Darüber hinaus gibt es zusätzlich die Definition der Zweibeinstandphase.

Diese Phase beschreibt den Zeitraum, bei dem beide Füße gleichzeitig Kontakt zum Boden haben. Nimmt jedoch die Gang- beziehungsweise Laufgeschwindigkeit zu, so nimmt die Dauer der Bodenkontaktzeit prozentual ab, bis schlussendlich keine Zweibeinstandphase mehr vorhanden ist. Bei hoher Laufgeschwindigkeit kommt es dann zu zwei sogenannten Schwebephasen, bei der jeglicher Kontakt zum Boden fehlt. Die Tabelle 2 zeigt bei unterschiedlichen Geschwindigkeiten die prozentuale Verteilung der Standphase zur Flug- beziehungsweise Schwungphase. [GN06, WPFW90]

Tabelle 2: Phasen im Laufschritt, Sprint und Gehen sowie ihre prozentuale Verteilung (vgl. [KdQSS08])

Laufschritt von 5 m/s:	Standphase (Kontaktphase): Flugphase	= 30 % : 70 %
Sprint von 5 m/s:	Standphase (Kontaktphase): Flugphase	= 20 % : 80 %
Im Vergleich zum Gehen (1.4 m/s):	Standphase (Kontaktphase): Schwungphase	= 62 % : 38 %

3 Parameter

Die folgende Aufzählung der Gangparameter unterliegt nicht einer bestimmten Reihenfolge und stellt keine Hierarchie dar, da die Parameter sich unter Umständen voneinander ableiten lassen und somit voneinander abhängig sind.

3.1 Gehgeschwindigkeit

Beim freien Gehen wird laut Kramers et al. vom Menschen in der Regel jene Ganggeschwindigkeit gewählt, die den geringsten Energieverbrauch darstellt. Eine Beschleunigung oder ein Verlangsamen erhöhen diesen Energieverbrauch. Der Norm-Wert für die physiologische freie Ganggeschwindigkeit von Erwachsenen liegt bei 1.20 bis 1.50 m/sec (vgl. Argstatter et al., Kramers et al., Winter et al.). Bei etwa 2 m/s fängt der Mensch instinktiv an zu laufen (vgl. Winter et al.).

Die Gehgeschwindigkeit (engl. Velocity) wird laut wissenschaftlichen Regelungen entsprechend der international standardisierten Vorgaben für Messungen des „Système International“ mit m/sec angegeben. Bei klinischen Anwendungen wird aufgrund der Kompatibilität zur Kadenz die Gehgeschwindigkeit mit m/min bevorzugt.

Die durchschnittliche Ganggeschwindigkeit entspricht dem Produkt der zurückgelegten Strecke und die dafür benötigte Zeit. Alternativ kann auch das Produkt aus Schrittlänge und der Schrittfrequenz gewählt werden. [GN06, WPFW90, AHTB07, KdQSS08]

3.2 Doppelschrittlänge und Schrittlänge

Die Doppelschrittlänge (engl. Stride length) steht für die Distanz eines Gangzyklus beziehungsweise die Länge von 2 Schritten, also je einem Schritt pro Bein, und wird in Meter (m) angegeben. Durch bloße Beobachtung kann laut Götz-Neumann die Doppelschrittlän-

ge nicht abgeschätzt werden. Zur Bestimmung müssen deshalb laut Kramers et al. und Argstatter et al. die Länge der Strecke und die Anzahl an benötigten Schritten für diese Strecke bekannt sein. Mithilfe der Formel 3.1 kann dann die Doppelschrittlänge berechnet werden.

$$Doppelschrittlänge = Weg\ (m) * 2\ /\ benötigte\ Schritte \tag{3.1}$$

Die Schrittlänge (eng. Step length) beschreibt die Distanz eines einfachen Schrittes und somit den Abstand, der während eines Gangzyklus mit jeweils einem Bein zurückgelegt wird. In Abbildung 2 ist dies schematisch dargestellt. Bei einem normalen und gesunden Gang sollten laut Götz-Neumann die Schrittlängen beider Schritte, links und rechts, etwa gleich groß sein. Kleine Abweichungen können bereits durch eine asymptomatische Beinlängendifferenz von bis zu 6 mm hervorgerufen werden.

Der Norm-Wert der Doppelschrittlänge beträgt bei Männern 1.46 m und bei Frauen 1.28 m in der Literatur. Der Unterschied in der Norm zwischen Männern und Frauen wird durch die durchschnittliche Körpergröße und die entsprechend langen Beine verursacht.

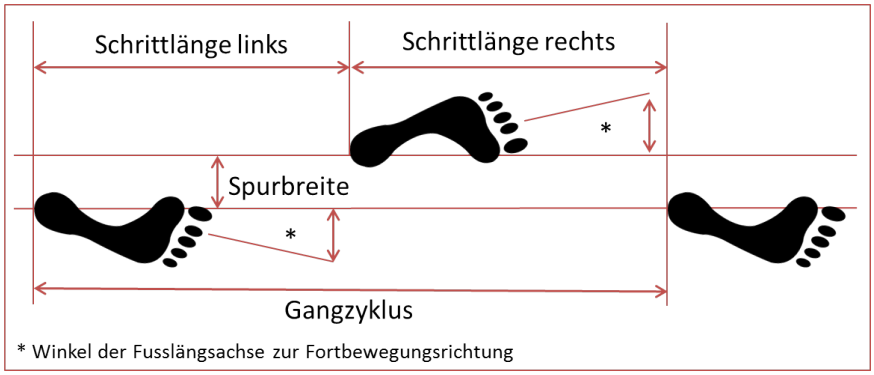


Abbildung 2: Darstellung der Schrittlänge und Spurbreite (Vgl. [GN06])

Zur Bestimmung der mittleren Schrittlänge sind die gleichen Parameter wie für die Doppelschrittlänge von Nöten. Denn dessen Ergebnis wird halbiert, um die durchschnittliche Schrittlänge zu erhalten. [GN06, AHTB07, KdQSS08]

3.3 Doppelschrittdauer und Schrittdauer

Die Doppelschrittdauer (engl. Stride time) bezeichnet laut Götz-Neumann die Dauer, die für einen Doppelschritt und somit einen Gangzyklus benötigt wird. Während bei der

Schrittdauer (engl. Step time) lediglich der Schritt eines Beines betrachtet wird. Für die Zeitmessung sind daher die Phasen des initialen Kontaktes mit dem Boden wichtig. Die Doppelschrittdauer findet zwischen dem ersten Aufsetzen des Fußes und dem nächsten Aufsetzen desselben Fußes statt, während die Schrittdauer nur bis zum Aufsetzen des anderen Fußes gemessen wird. [GN06]

3.4 Dauer der Phasen

Die jeweilige Dauer der einzelnen Gangphasen, wie sie oben beschrieben sind, lässt sich nicht einfach bestimmen. Für genaue Berechnungen werden laut Kramers et al. Video-Aufzeichnungen, Motion-Capture-Systeme oder ähnliches benötigt. Um ohne solche zeit- und/oder kostenintensive Berechnungen dennoch an Kennziffern zu gelangen, kann man die jeweilige Dauer approximieren. Dafür wird die Doppelschrittdauer auf die acht Gangphasen gemäß ihrem absoluten prozentualen Anteil, wie in Tabelle 3 entnommen werden kann, segmentiert. [KdQSS08]

Tabelle 3: Beziehung zwischen den 8 Gangphasen, ihrem zeitlichen sowie absoluten Anteil und den funktionellen Aufgaben (vgl. [GN06])

	Standphase: 62%					Schwungphase: 38%		
Funktion	Gewichtsübernahme		Einbeinstand		Schwungbeinvorwärtsbewegung			
Bezeichnung	IC	LR	MSt	TSt	PSw	Isw	MSw	TSw
zeitlicher Anteil	0%	0-12%	12-31%	31-50%	50-62%	62-75%	75-87%	87-100%
abs. Anteil	0%	12%	19%	19%	12%	13%	12%	13%

3.5 Kadenz

Die Kadenz (engl. Cadence) gibt laut Götz-Neumann die Anzahl Schritte pro Minute an und wird auch Schrittfrequenz genannt. „Wie bei einem Pendel schwingen die Beine beim Gehen in einer bestimmten Frequenz entgegengesetzt proportional zu ihrer Länge.“ [GN06] Das hat zu bedeuten, dass kleinere Personen in der Regel eine höhere Kadenz besitzen als große Personen. Der normale Bereich, der je nach Alter, Geschlecht und Beinlänge variiert, liegt laut Götz-Neumann und Argstatter et al. bei 100 – 130 Schritten die Minute. Der durchschnittliche Wert von Männern liegt bei 111 Schritten/min und von Frauen bei 117 Schritten/min (vgl. Kramers et al.).

Um die Kadenz zu bestimmen reicht es aus, für eine beliebige Zeitdauer, jedoch mindestens 10 Sekunden, die Schritte zu zählen. „Ganz präzise ist die Messung zwar nicht, aber die Abweichung kann als unbedeutend angesehen werden.“ [GN06] Diese Abweichung wird proportional geringer bei längerer Messung, wobei eine Messung über 60 Sekunden bereits nur noch eine Abweichung von 2 % beträgt. Die Formel 3.2 gibt an, wie die Kadenz anschließend berechnet wird (vgl. [GN06, MNH04]).

$$Kadenz\ (Schritte/min) = gezählte\ Schritte * 60\ /\ gestoppte\ Zeit\ (sec) \tag{3.2}$$

Das Team um Emma Fortune des Motion Analysis Laboratory der Mayo Clinic in Rochester (USA) fand die Formel 3.3 um die Kadenz mithilfe von Accelerometer-Daten zu bestimmen.

$$Cadence = (n-1)/(t_n - t_1) \tag{3.3}$$

Bei dieser Formel beschreibt n die Anzahl an Schritten in dem ausgewählten Segment, t_n den Zeitpunkt in Minuten des initialen Kontaktes des n-ten Schrittes und t_1 den Zeitpunkt in Minuten des ersten initialen Kontaktes. [GN06, AHTB07, KdQSS08, FLMK14, MNH04]

3.6 Spurbreite

Für die Definition des Begriffs der Spurbreite, wurde folgende aus dem Buch „Gehen verstehen: Ganganalyse in der Physiotherapie“ von Götz-Neumann gewählt:

„Der Begriff Spurbreite (engl. Step width) ist [wie in Abbildung 2 zu sehen ist, C.P.] durch die Distanz zwischen den beiden Fersenzentren definiert. Die Entfernung wird senkrecht zur Fortbewegungslinie gemessen und liegt normalerweise im Bereich von 5-13 cm.“ [GN06]

3.7 Symmetrie

Für die Feststellung eines symmetrischen Gangs werden laut Götz-Neumann die Schrittlänge, das Standbeinverhältnis, der Armschwung und die Kopfposition betrachtet. Im Hinblick darauf, dass lediglich Sensordaten vorliegen, wird im Folgenden eine vereinfachte Variante des Begriffs der Symmetrie betrachtet. Dabei wird das Augenmerk auf die Schrittlänge

und Schrittdauer gelegt.

Um festzustellen, ob ein Gang symmetrisch oder asymmetrisch ist, werden die Differenz zwischen der Schrittlänge des rechten Beines und der Schrittlänge des linken Beines sowie die Differenz der Schrittdauer des rechten Beines mit der des linken Beines ermittelt. Bei einem gesunden und somit symmetrischen Gang liegen laut Winter et al. diese Differenzen bei null. [WPFW90, GN06]

4 Berechnung mithilfe eines Accelerometers

Bei einem Projekt der NTT Corporation und der Wacoal Corporation in Japan, das in dem Paper „Estimation of beautiful gait using an accelerometer“ von Niijima et al. beschrieben wird, wurde getestet, wie zuverlässig ein sogenannter „schöner Gang“ (eng. Beautiful gait) mithilfe eines Accelerometers detektiert werden kann. Bei dem „schönen Gang“ handelt es sich um einen gesunden Gang mit gerader Körperhaltung.

Für dieses Experiment wurden 41 weibliche Testpersonen zwischen 20 und 70 Jahren mit jeweils drei Accelerometern (MVP-RF8-HC, MicroStone, Abtastrate: 500 Hz) ausgestattet. Diese wurden an unterschiedlichen Stellen des Körpers befestigt: vorne an der Brust, hinten an der Hüfte und am linken Handgelenk. Ein bewährtes Motion-Capture-System wurde parallel eingesetzt, um dem „schönen Gang“ festzustellen und den Ergebnissen der Auswertung der Accelerometer gegenüberzustellen. Jede Testperson musste dafür sechsmal eine gerade Strecke von 10 Metern gehen.

Für die Auswertung wurden jeweils 4 Sekunden der Aufnahmen gewählt, in denen ein durchgehender Gang stattfindet. Auf die Rohdaten wurde anschließend ein Breitbandfilter angewandt. Hier wurden zwei Einstellungen getestet: 0.3 – 3 Hz und 0.05 – 10 Hz.

Für die Analyse wurden 6 statistische Merkmale und 3 Gangparameter berechnet. Bei den 6 statistischen Merkmalen handelt es sich um die Durchschnittswerte und die Standardabweichungen der Achsen des Accelerometers. Als Gangparameter wurden die Balance, Kadenz und Stärke beim Aufsetzen der Ferse genutzt. Für die weitere Verarbeitung mit Machine Learning Algorithmen wurden alle Merkmale nach der Berechnung standardisiert.

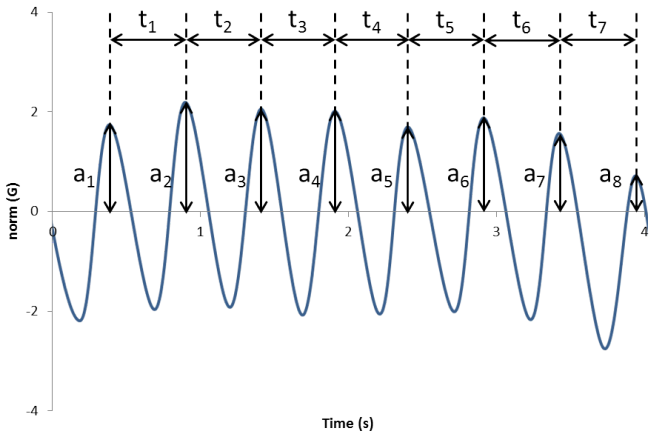


Abbildung 3: Beispiel für die Standardnorm von gefilterten Accelerometer Daten (Standardisiert) (Vgl. [NYM+16])

Die Durchschnittswerte wurden mithilfe der Rohdaten und die Standardabweichungen mithilfe der gefilterten Daten (von 4 Sekunden Länge) bestimmt. Für die Balance wurde zunächst die Standardnorm, oder auch euklidische Norm genannt, der gefilterten Daten gebildet. Die Balance berechnet sich anschließend aus dem Maximum des Autokorrelations-Koeffizienten der Standardnorm. Die Kadenz wurde mit der Formel 4.1 berechnet, wobei t_k die Intervalle zwischen zwei aufeinanderfolgenden Maxima der Standardnorm und n die Anzahl an Maxima der gefilterten Daten angibt, wie in Abbildung 3 veranschaulicht wird.

$$Cadence = 1/(n-1) \sum_{k=1}^{n-1} 60/t_k \tag{4.1}$$

Die Stärke des initialen Kontaktes mit dem Fußboden wurde mit der Formel 4.2 berechnet, hierbei ist a_k die Amplitude der Maxima der Standardnorm und n die Anzahl an Maxima der gefilterten Daten.

$$impact\ at\ heel\ strike = 1/n \sum_{k=1}^{n} a_k \tag{4.2}$$

Für die Klassifikation wurden mehrere Ansätze des Machine Learning getestet. Darunter befanden sich die Nächste-Nachbarn-Klassifikation (k-NN), die logistische Regression (LR) und Gaussian naive Bayes (GNB) als Klassifikation. Jeder Ansatz wurde einmal mit allen Merkmalen getestet und ein zweites Mal, bei dem nur die sechs statistischen Merkmale genutzt wurden. Die Genauigkeit jeder einzelnen Einstellung wurde mit zwei unterschiedlichen Arten der Kreuzvalidierung berechnet. Zum einen mit der Leave-One-Out Kreuzvalidierung (mit 245 Training- und einem Testdurchlauf) und zum anderen mit

der k-fachen Kreuzvalidierung (240 Trainings- und 6 Testdurchläufe), wobei k dem Wert 41 entspricht.

Das Ergebnis von Niijima et al. wird in den Tabellen 4 und 5 dargestellt. Sie zeigen die besten Parameter für jede Position des Sensors. Dabei fällt auf, dass bei der Leave-One-Out Kreuzvalidierung (Tabelle 4) alle Genauigkeiten sehr nah beieinander liegen. Bei der 41-fachen Kreuzvalidierung (Tabelle 5) jedoch liegt die höchste Genauigkeit bei einem an der Taille getragenen Sensors mit einem Breitbandfilter von 0.3 – 3 Hz, den statistischen Merkmalen und Gaussian naive Bayes als Klassifikation bei 78 %. [NYM+16]

Tabelle 4: Die höchsten Genauigkeiten der Leave-One-Out Kreuzvalidierung (vgl. [NYM+16])

Position	Filter	Dimension	Classifier	Accuracy
chest	0.3 - 3 Hz	9	k-NN	80.1 %
waist	0.05 - 10 Hz	6	k-NN	82.1 %
wrist	0.3 - 3 Hz	6 or 9	k-NN	82.9 %

Tabelle 5: Die höchsten Genauigkeiten der 41-fachen Kreuzvalidierung (vgl. [NYM+16])

Position	Filter	Dimension	Classifier	Accuracy
chest	0.3 - 3 Hz	6	LR	73.2 %
waist	0.3 - 3 Hz	6	GNB	78.0 %
wrist	0.3 - 3 Hz	6	k-NN	58.1 %

5 Anwendung im Projekt

Im Vorfeld der Projektarbeit wurden Probanden während des Durchlaufens mehrerer Assessments mit einem Sensorgürtel ausgestattet. Die Aufnahmen beinhalten für die Bestimmung der Gangparameter zwei wesentliche Assessments: der Frailty-Gehtest mit dem 4,57 m langen Gehtest und der 6-Minuten-Gehtest. Mit diesen beiden Assessments und der bereits implementierten Schritterkennung können ein Großteil der oben genannten Parameter abgeleitet werden.

Mit der Schritterkennung können direkt die Parameter Doppelschrittdauer und Schrittdauer abgeleitet werden, da die Schritterkennung mit der Erkennung des initialen Kontaktes verbunden ist. Dies bedeutet aber auch, dass Schritte gezählt und somit die Kadenz berechnet werden kann. Dies kann jedoch auch mithilfe des 6-Minuten-Gehtests und der Anzahl der Schritte, die in der Zeit gemacht wurden, berechnet werden.

Der 4,57 m lange Gehtest kann sowohl dazu genutzt werden, um die Gehgeschwindigkeit zu bestimmen, als auch für die Berechnung der Doppelschrittlänge beziehungsweise der Schrittlänge. Mit der Länge und der Dauer der (Doppel-)Schritte kann somit auch die Gangsymmetrie beschrieben werden.

Lediglich die Spurbreite lässt sich von den anderen Parametern nicht ableiten. Aber vielleicht lässt sich mithilfe der Berechnung der Balance aus dem Projekt der NTT Corporation und der Wacoal Corporation aus Japan dieser Wert ableiten oder den Parameter der Symmetrie verfeinern. Bei dem Kennwert der Balance muss entsprechend erst noch geprüft werden, wie aussagekräftig dieser wirklich ist. Im besten Fall kann die Balance als zusätzlicher Parameter für die Bestimmung der Symmetrie verwendet werden.

Literatur

[AHTB07] H Argstatter, TH Hillecke, M Thaut, and HV Bolay. Musiktherapie in der neurologischen rehabilitation. *Neurol Rehabil*, 13(1):42–48, 2007.

[FLMK14] Emma Fortune, Vipul Lugade, Melissa Morrow, and Kenton Kaufman. Validity of using tri-axial accelerometers to measure human movement–part ii: Step counts at a wide range of gait velocities. *Medical engineering & physics*, 36(6):659–669, 2014.

[GN06] Kirsten Götz-Neumann. *Gehen verstehen: Ganganalyse in der Physiotherapie; 18 Tabellen*. Georg Thieme Verlag, 2006.

[KdQSS08] Inès A Kramers-de Quervain, Edgar Stüssi, and Alex Stacoff. Ganganalyse beim gehen und laufen. *Schweizerische Zeitschrift für Sportmedizin und Sporttraumatologie*, 56(2):35–42, 2008.

[MNH04] Rolf Moe-Nilssen and Jorunn L Helbostad. Estimation of gait cycle characteristics by trunk accelerometry. *Journal of biomechanics*, 37(1):121–126, 2004.

[Neu13] Donald A Neumann. *Kinesiology of the musculoskeletal system: foundations for rehabilitation*. Elsevier Health Sciences, 2013.

[NYM+16] Arinobu Niijima, Kazuhiro Yoshida, Osamu Mizuno, Yumiko Tanmatsu, Naoki Asanoma, Tomoki Watanabe, Tsubasa Nakayama, and Makoto Oyama. Estimation of beautiful gait using an accelerometer. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 169–172. ACM, 2016.

[WPFW90] David A Winter, Aftab E Patla, James S Frank, and Sharon E Walt. Biomechanical walking pattern changes in the fit and healthy elderly. *Physical therapy*, 70(6):340–347, 1990.

D.8. Ermittlung von Feature-Sets zur Optimierung der Klassifikationsparameter

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Ermittlung von Feature-Sets zur Optimierung der Klassifikationsparameter

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Andrea de Behr

Oldenburg, den 7. April 2017

Abstract

Die Aktivitätserkennung mit Sensoren erfordert eine sorgfältige Auswahl der für die Klassifikation verwendeten Kombinationen aus Features und Sensordatenreihen, da zum einen nicht jede Merkmal-Sensor-Kombination für jede Klassifikation geeignet ist und zum anderen die Rechenzeit im überschaubaren Rahmen bleiben muss. Es gilt also die Anzahl der Merkmal-Sensor-Kombinationen so einzugrenzen, dass nur diejenigen verwendet werden, die tatsächlich das Klassifikationsergebnis signifikant verbessern. Diese Ausarbeitung wurde im Rahmen der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ erstellt, wobei zu beachten ist, dass die Mobilitäts-Assessments eine hohe Anzahl verschiedener Bewegungsmuster beinhalten. Aus diesem Grund wird die Klassifikation auf hierarchische Weise durchgeführt, sodass zunächst die Bewegungsmuster in Static, Dynamic und Transition eingeteilt werden, um dann erst im nächsten Schritt die einzelnen Bewegungsmuster zu identifizieren. Die vorliegende Ausarbeitung beschäftigt sich sowohl mit der Auswahl einer geeigneten Fenstergröße und Schrittweite der verwendeten Sliding Windows als auch mit der Eingrenzung der möglichen Merkmals-Sensor-Kombinationen zur Klassifikationsoptimierung.

Inhalt

Abkürzungen

RMS	Root Mean Square
SMA	Signal Magnitude Area
SMV	Signal Vector Magnitude
SFC	Single Feature Classification
SFS/SFFS	Sequential Forward Selection/Sequential Forward Floating Search

Abbildungen

Tabellen

1 Einleitung

Eines der Ziele der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ ist es, in den während eines Mobilitäts-Assessments mit einem Sensorgürtel aufgenommenen Bewegungsdaten eines Probanden einzelne Bewegungsmuster möglichst präzise zu erkennen. Diese sogenannte Klassifikation setzt voraus, dass die einzelnen Datenreihen (Daten der x-, y- und z-Achsen im Zeitraum des Mobilitäts-Assessments) der jeweiligen Sensoren (Akzelerometer, Gyroskop und Barometer) vorab auf Merkmale (Features) untersucht werden, die die anschließende Klassifikation ermöglichen. Die Merkmalsgewinnung findet innerhalb eines Sliding Windows statt, also eines Fensters mit einer festen Größe, das sich mit einer bestimmten Schrittweite über den gesamten Datensatz bewegt. Anhand des Feature-Vektors eines Sliding Windows bestimmt dann der State-Klassifikator, ob das Bewegungsmuster innerhalb des Sliding Windows ein statisches (Static), dynamisches (Dynamic) oder Übergangsbewegungsmuster (Transition) ist. Im zweiten Schritt dieser hierarchischen Klassifikation bestimmt dann der entsprechende Motion-Klassifikator (Static, Dynamic oder Transition), um welches Bewegungsmuster es sich handelt. Dabei unterscheidet der Static-Klassifikator in sit, stand, lie-on-back und lie-on-side, der Dynamic-Klassifikator in walk, stair climb und jump und der Transition-Klassifikator in sit-stand oder stand-sit.

Allerdings gibt es zu viele Merkmal-Sensor-Kombinationen und Parameter, um eine optimale Klassifikation zu gewährleisten. Die Merkmale Autokorrelation, Mean bzw. Arithmetisches Mittel, Root Mean Square (RMS) bzw. Quadratisches Mittel, Standardabweichung, Signalenergie und Spektralentropie lassen sich jeweils mit allen drei Sensoren kombinieren, wohingegen sich Korrelation, Signal Magnitude Area (SMA) und Signal Vector Magnitude (SMV) jeweils nur mit Akzelerometer und/oder Gyroskop kombinieren lassen. Das Feature Pitch (Neigungswinkel) wiederum benötigt sowohl Akzelerometer- als auch Gyroskopdaten. Es gibt also insgesamt 25 (= 6x3 + 3x2 + 1) Merkmal-Sensor-Kombinationen bzw. Feature-Sets, die man wieder ihrerseits kombinieren könnte, sodass

insgesamt $2^{25} = 33.554.432$ Feature-Sets möglich wären - wohlgemerkt pro Klassifikator (State, Static, Dynamic und Transition) und pro Machine-Learning-Algorithmus (Adaptive Multi-Hyperplane Machines, Multilayer Perceptrons, Boosted Decision Trees und Boosted Decision Stumps). Darüber hinaus wurden bei dieser Berechnung sonstige Parameteranpassungen (bei Rauschunterdrückungsfiltern, Sliding Windows, Features und Machine-Learning-Algorithmen) noch nicht berücksichtigt. Es ist also eine Einschränkung des Parameterraums für die Optimierung der Klassifikation notwendig, die mit Hilfe eines evolutionären Algorithmus erfolgen soll. Zu diesem Zweck sollen in der hier vorliegenden Ausarbeitung geeignete Feature-Sets ermittelt werden, die dem evolutionären Algorithmus eine Annäherung an ein Klassifikationsoptimum ermöglichen.

2 Sliding Windows

Um mit Hilfe der Sliding Windows die Merkmalsgewinnung durchführen zu können, müssen die Fenstergröße und die Schrittweite festgelegt werden. In diesem Kapitel werden entsprechende wissenschaftliche Veröffentlichungen mit den Projektgruppenerfahrungen verglichen und Vorschläge für weitere Testszenarien unterbreitet.

2.1 Literaturrecherche zu Sliding Windows

Zur Merkmalsgewinnung in der Aktivitätserkennung werden Sliding Windows mit einer Fenstergröße von ca. 2.000 bis 2.500 ms gewählt [ZS11] [AGO+13] oder auch bis zu 6.700 ms [BAK+14] [GD14] [KLLK10]. Dies ist dadurch begründet, dass die zu erkennenden nichtperiodischen Bewegungsmuster (wie z.B. Hinsetzen oder Springen) normalerweise maximal 2,5 s in Anspruch nehmen. Dies wird üblicherweise mit einer Überlappung der Fenster von 50% kombiniert [ZS11] [AGO+13] [BAK+14] [GD14] [PHO11]. Das Sliding Window wird also jeweils um 1.000 ms bzw. 1.250 ms verschoben. Dies entspricht der Mindestdauer eines nichtperiodischen Bewegungsmusters.

2.2 Vergleich mit Testszenarien

Da im Rahmen der Projektgruppe eine vergleichsweise große Menge einzelner Bewegungsmuster unterschieden werden soll, wird im Gegensatz zur üblichen Herangehensweise an die Aktivitätserkennung dafür die bereits eingangs beschriebene hierarchische Merkmalsgewinnung verwendet. So lässt sich zur Verbesserung des Klassifikationsergebnisses die Fenstergröße und die Schrittweite auf die jeweilige Gruppe von Bewegungsarten (Static, Dynamic, Transition) bzw. Bewegungsmustern (z.B. walk, stair climb, jump) abstimmen. Zu diesem Zweck wurde für die Projektgruppe bereits eine Tabelle mit Testszenarien er-

stellt, die sich in der Abschlussarbeit im Abschnitt „Auswahl einer geeigneten Fenstergröße und Schrittweite für das Sliding-Window-Verfahren“ im Kapitel „Evaluierung“ befindet. Es wurden Fenstergrößen von 250 bis 2.000 ms getestet und Schrittweiten von 200 bis 1.750 ms. Auf die Ergebnisse des Tests wird im Folgenden kurz eingegangen und es werden Vorschläge unterbreitet, in welche Richtung noch weiter getestet werden könnte:

State-Sliding-Window

- Fenstergröße (momentan 1.500 ms): beste Erkennung von Transitionen bei 1.500 ms (90 bis 100%)
- Schrittweite (momentan 300 ms): je größer die Schrittweite, desto besser die Erkennung von Transitionen => zu 1.500 ms ändern (Erkennung von 93 bis 99%)

Static-Sliding-Window

- Fenstergröße (momentan 2.000 ms): beste Erkennung bei größter Fenstergröße (91 bis 98%) => bis 6.700 ms ausprobieren
- Schrittweite (momentan 400 ms): beste Erkennung bei Schrittweiten von 200 bis 500 ms (91 bis 98%)

Dynamic-Sliding-Window

- Fenstergröße (momentan 1.200 ms): beste Erkennung von jump (85%) und vergleichsweise akzeptable Erkennung von walk und stair climb (58 bis 81%) bei 1.250 ms
- Schrittweite (momentan 400 ms): beste Erkennung von walk und stair climb bei 200 bis 400 ms (80 bis 81%) => Schrittweiten zwischen 200 und 400 ms ausprobieren

Transition-Sliding-Window

- Fenstergröße (momentan 1.000 ms) [BAK+14]: beste Erkennung bei 1.000 bzw. 1.250 ms (88 bis 94%) => Fenstergrößen zwischen 1.000 und 1.250 ms ausprobieren
- Schrittweite (momentan 1.000 ms): beste Erkennung bei 1.000 bzw. 1.250 ms (89 bis 94%), allerdings variieren die Ergebnisse recht stark

Außer bei statischen Bewegungsmustern liefern Fenstergrößen, die kleiner als 2.000 ms

sind, bessere Ergebnisse. Dies liegt wahrscheinlich daran, dass ein Schritt ca. 500 ms in Anspruch nimmt und Hinsetzen bzw. Aufstehen meist 1.000 bis 1.500 ms dauert. Im Durchschnitt ergeben die Überlappungen zwar ca. 50%, weichen aber doch für die einzelnen Gruppen von Bewegungsarten bzw. Bewegungsmustern stark davon ab.

3 Feature-Sets

Ein Feature-Set ist eine Menge von Merkmal-Sensor-Kombinationen und dient zur Merkmalsgewinnung für die anschließende Klassifikation. Im Folgenden wird beschrieben, welche Features im Projekt „Mobilitäts-Assessments mit körpernahen Sensoren“ verwendet werden (siehe hierzu auch den Unterabschnitt „Merkmalsgewinnung“ im Abschnitt „Aktivitätserkennung“ des Kapitels „Grundlagen“ der Abschlussarbeit) und für welche Klassifikatoren sie jeweils geeignet sind. Auf der Basis logischer Begründungen, manueller Tests und der dazugehörigen Literaturrecherche wurden dann entsprechende Feature-Sets zusammengestellt, die zur Optimierung der Klassifikation beitragen sollen.

3.1 Verwendete Features

Bei der **Korrelation** vergleicht man jeweils die Messwerte zweier Datenreihen miteinander. Sind sie gleichläufig, ergibt der Korrelationsfaktor 1, sind sie gegenläufig, ergibt er -1 und, wenn sie gar nicht korrelieren, ergibt er 0.

Für **Mean** und **RMS** berechnet man den Durchschnittsmesswert über eine Achse, wobei RMS zum einen stärkere Signale verstärkt und zum anderen nicht wie Mean nach Vorzeichen unterscheidet (durch das Quadrieren).

Bei der **Standardabweichung** wird der Durchschnittsmesswert über alle Messwerte jeweils abzüglich des arithmetischen Mittels gebildet.

Die **Spektralentropie** unterscheidet geordnete Signale (der Entropiewert ist 0) von nicht vorhersagbaren (der Entropiewert ist 1).

Für die **Autokorrelation** vergleicht man eine Datenreihe mit sich selbst zu einem späteren Zeitpunkt. Ähnlich wie bei der Korrelation ergibt die Autokorrelation bei Gleichläufigkeit einen Wert signifikant größer als 0, bei Gegenläufigkeit einen Wert signifikant kleiner als 0 und, wenn die Datenreihe gar nicht mit sich selbst korreliert, einen Wert von 0.

Bei **SMA/SMV** berechnet man den Durchschnittswert über alle drei Achsen eines Sensors, wobei SMA stärkere Signale verstärkt.

Für das **Pitch**-Feature wird der Nickwinkel (Drehung um die x-Achse) berechnet.

Bei der **Signalenergie** werden die Quadrate der Messwerte einer Datenreihe aufsummiert, aber ohne dass diese durch die Anzahl der Messwerte geteilt werden.

3.2 Eignung der Features in der Klassifikationshierarchie

Wie bereits im Kapitel „Sliding Windows“ erwähnt wird in den meisten wissenschaftlichen Veröffentlichungen zur Aktivitätserkennung mit Sensoren keine hierarchische Klassifikation verwendet, wie sie in diesem Projekt genutzt wird [KLLK10], sondern es wird versucht, gleich im ersten Schritt alle Bewegungsmuster zu erkennen. Darüber hinaus werden überwiegend nur die Akzelerometerdaten zur Merkmalsgewinnung herangezogen. Der für die Mobilitäts-Assessments genutzte Sensorgürtel stellt aber über das Akzelerometer hinaus noch ein Gyroskop und ein Barometer zur Verfügung, deren jeweilige Daten zur Feature-Berechnung genutzt werden können. Nachfolgend werden die Feature-Sensor-Kombinationen jeweils den Klassifikatoren zugeordnet, die sich aus logischen Begründungen, manuellen Tests oder der Literaturrecherche ergeben.

Signalenergie

Die Signalenergie ist weder für die State- noch für die Motion-Erkennung geeignet, da entweder das Programm aufgrund der sehr schnell wachsenden Werte abstürzt oder diese zu weit auseinanderfallen, um vergleichbar zu sein.

Barometer

Auch die Barometerdaten sind für die State- oder Motion-Erkennung eher ungeeignet. Am ehesten lassen sich damit noch die Dynamic-Bewegungsmuster walk, stair climb (der Luftdruck fällt ein wenig ab) und jump (der Luftdruck steigt kurz an) unterscheiden.

State-Klassifikator

Dieser Klassifikator unterscheidet nach Static (in Abbildung 3 und Abbildung 4), Dynamic (in Abbildung 1 und Abbildung 2) und Transition (in Abbildung 4).

Korrelation [AGO+13] [KLLK10]
Die Korrelation auf den Akzelerometerdaten ist gut zur Unterscheidung der Bewegungsarten geeignet, da Static stark korreliert, Dynamic merklich korreliert, wohingegen Transition kaum korreliert.

Mean/RMS [ZS11] [AGO+13] [BDP+12] [GD14] [KLLK10] [PHO11] [Yan09]
Die Mittelwerte der Akzelerometerdaten sind gut geeignet, um Static von Dynamic bzw. Transition zu unterscheiden [BAK+14]. Die Unterscheidung zwischen Dynamic und Transition hingegen ist augenscheinlich schwieriger.

Standardabweichung [AGO+13] [BDP+12] [GD14] [KLLK10] [PHO11] [Yan09]
Die Standardabweichung ist für die Static-Bewegungsmuster gleich 0, sodass sich diese gut von den Dynamic- und Transition-Bewegungsmustern abgrenzen lassen [BAK+14], und die Dynamic-Bewegungsmuster sollten wiederum eine deutlich höhere Standardabweichung aufweisen als die Transition-Bewegungmuster.

Spektralentropie [ZS11] [AGO+13] [BAK+14] [BDP+12] [GD14] [KLLK10] [PHO11] [Yan09]
Die Spektralentropie ist gut geeignet, um Static-Bewegungsmuster, die geordnet sind und daher einen Wert von 0 ergeben, von Dynamic- und Transition-Bewegungsmustern, die ungeordneter sind und daher einen höheren Wert ergeben, zu unterscheiden.

Motion-Klassifikatoren

Die Motion-Klassifikatoren umfassen den Static-Klassifikator, der nach sit und stand (beide in Abbildung 4) sowie lie on side und lie on back (beide in Abbildung 3) unterscheidet, den Dynamic-Klassifikator, der nach walk und stair climb (beide in Abbildung 1) sowie jump (in Abbildung 2) unterscheidet, und den Transition-Klassifikator, der nach sit-stand und stand-sit (beide in Abbildung 4) unterscheidet.

Autokorrelation [BDP+12]
Bei den Transition-Bewegungsmustern ergibt die Autokorrelation zwischen sit-stand und stand-sit eine gegenläufige Korrelation und ist somit gut zur Unterscheidung der beiden geeignet. Bei den Dynamic-Bewegungsmustern ergibt die Autokorrelation zwischen walk bzw. stair climb und jump keine oder nur wenig Korrelation und zwischen walk und stair climb zwar eine gleichläufige Korrelation, aber die beiden Kurven sind eben nicht völlig identisch, sodass die Autokorrelation auch hier gut geeignet ist zur Unterscheidung. Für

die Static-Bewegungsmuster weist die Autokorrelation allerdings für sit, stand, lie on side und lie on back allesamt eine sehr hohe gleichläufige Korrelation aus, sodass hier die Unterscheidung weniger offensichtlich ist.

SMA/SMV [AGO+13] [KLLK10]
Die SMA- oder SMV-Werte der Akzelerometerdaten sind gut geeignet, die beiden Transitions-Bewegungsmuster sit-stand und stand-sit voneinander zu unterscheiden, da beide Bewegungsmuster zwar seitenverkehrt sehr ähnlich, aber eben nicht identisch sind und daher über alle drei Sensorachsen gerechnet verschiedene Werte herauskommen müssen. Auch die Dynamic-Bewegungsmuster walk und stair climb lassen sich mit dem SMA- bzw. SMV-Feature auf den Akzelerometerdaten gut auseinanderhalten, da sie sich zwar ähneln, aber nicht gleich sind und diese Unterschiede sich über drei Achsen gerechnet verstärken müssten. Die Static-Bewegungsmuster lie on side und lie on back sowie sit bzw. stand lassen sich ebenfalls mit dem SMA- oder SMV-Feature gut abgrenzen, da auch hier die Summe der Werte aller drei Achsen die Unterschiede zwischen den Bewegungsmustern stärker herausstreicht. [BAK+14]

Pitch [KLLK10] [PHO11]
Das Pitch-Feature ist, wie man sich vorstellen kann, gut geeignet, um bei den Static-Bewegungsmustern lie on side bzw. lie on back von sit bzw. stand abzugrenzen. Die Dynamic-Bewegungsmuster walk bzw. stair climb unterscheiden sich weniger stark von jump, wie auch das Transition-Bewegungsmuster sit-stand sich weniger stark von stand-sit unterscheidet.

Korrelation [AGO+13]
Die Korrelation (hauptsächlich auf den Akzelerometerdaten) ist gut zur Unterscheidung der Dynamic-Bewegungsmuster geeignet, da walk, stair climb und jump sehr individuell korrelieren [ZS11]. Die Static- und Transition-Bewegungsmuster hingegen korrelieren jeweils auf sehr ähnliche Weise (die Transition-Bewegungsmuster nur seitenverkehrt), sodass sich hier kaum Verbesserungen erzielen lassen.

Mean/RMS [ZS11] [AGO+13] [BDP+12] [GD14] [PHO11] [Yan09]
Bei den Static-Bewegungsmustern sind die Mittelwerte der Akzelerometerdaten gut zur Unterscheidung von lie on side, lie on back, sit und stand geeignet [BAK+14]. In den Transition-Bewegungsmustern ergeben beide Mittelwerte der Akzelerometerdaten und auch die arithmetischen Mittelwerte der Gyroskopdaten von sit-stand etwas niedrigere

Werte als die von stand-sit. Die Mittelwerte der Dynamic-Bewegungsmuster scheinen nicht stark voneinander abzuweichen.

Standardabweichung [AGO+13] [BAK+14] [BDP+12] [GD14] [PHO11] [Yan09]
Die Standardabweichung ist für alle Static-Bewegungsmuster gleich 0 und daher hier nicht zur Unterscheidung geeignet. Bei den Dynamic-Bewegungsmustern ergibt die Standardabweichung für jump deutlich höhere Werte als für walk und stair climb [ZS11]. Die Standardabweichung von sit-stand unterscheidet sich nur wenig von der für stand-sit.

Abbildung 1: Bewegungsmusterbeispiel für stand (türkis), stair climb (pink) und walk (grün)

3.3 Feature-Set-Vorschläge

Die sich aus der Literaturrecherche, den logischen Begründungen und manuellen Tests ergebenden Merkmals-Sensor-Kombinationen sind im Folgenden tabellarisch zusammengefasst. In den Spalten zu den Features Mean/RMS und SMA/SMV steht in einigen Fällen der Kommentar „(beide)“hinter den zu verwendenden Sensordatenreihen. Damit ist gemeint, dass ein Feature-Set mit dem jeweils ersten Feature getestet werden soll und ein weiteres Feature-Set mit dem jeweils zweiten Feature. Wenn nach den zu benutzenden Sensordatenreihen als Kommentar eines der Features angegeben ist, dann soll die Merkmals-Sensor-Kombination auch nur mit diesem Feature getestet werden. Die letzte Spalte beinhaltet eventuelle Parameter der Features, die im Falle der Autokorrelation

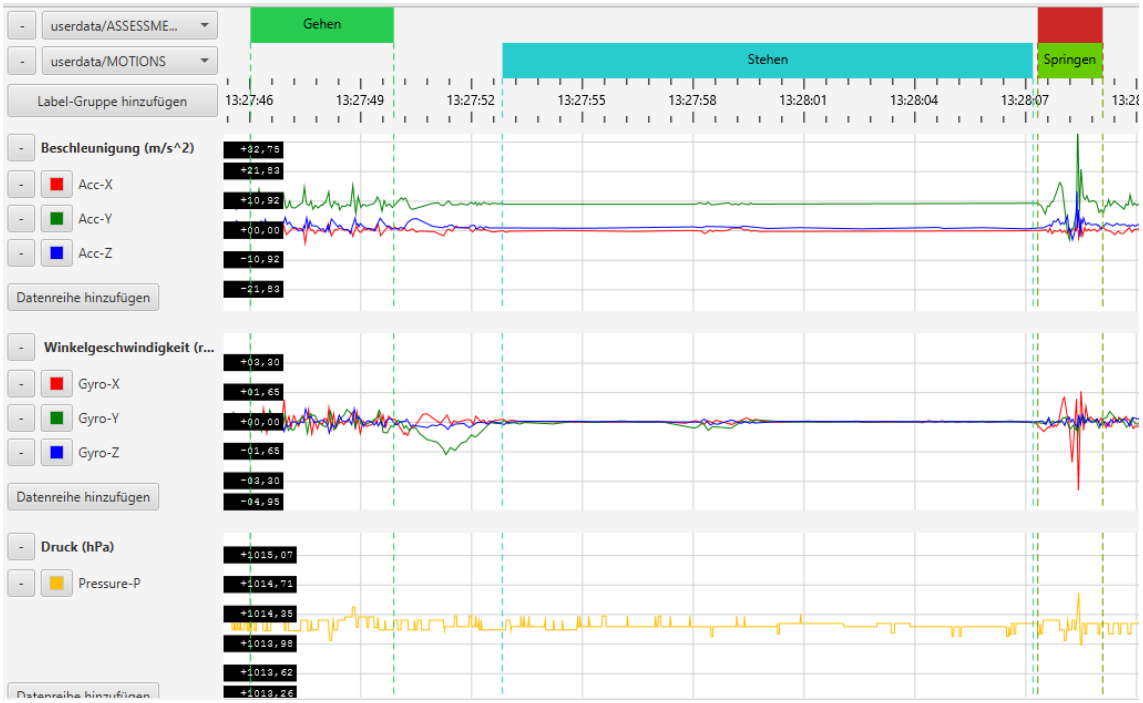


Abbildung 2: Bewegungsmusterbeispiel für walk (grün links), stand (türkis) und jump (grün rechts)

und der Spektralentropie angepasst werden können. Bei der Autokorrelation steht der Parameter „NumLags“ für die Anzahl der Verschiebungen um jeweils 10 ms innerhalb eines Sliding Windows. Die Autokorrelationswerte des dabei entstehenden Teil-Feature-Vektors können dann bei der Klassifikation mit den Autokorrelationswerten eines anderen Teil-Feature-Vektors derselben Datenreihe zu einem späteren Zeitpunkt verglichen werden. Der Parameter „NumLags“ muss kleiner sein als die Fenstergröße des jeweiligen Sliding Windows und sollte auch nicht zu groß gewählt werden, um die Rechenzeit nicht in die Höhe zu treiben. Momentan ist „NumLags“ = 10, aber „NumLags“ könnte bis zur Messpunkteanzahl der Dauer eines Schrittes, der das kürzeste vollständige Bewegungsmuster darstellt) erhöht werden. Bei der Spektralentropie steht der Parameter „N“ für die Anzahl der Frequency Bins, in die der durch die Fourier-Transformation entstandene Frequenzbereich aufgeteilt werden kann. Momentan ist „N“ = 5, aber „N“ könnte bis zur Anzahl der Messpunkte innerhalb der jeweiligen Sliding-Window-Größe erhöht werden.

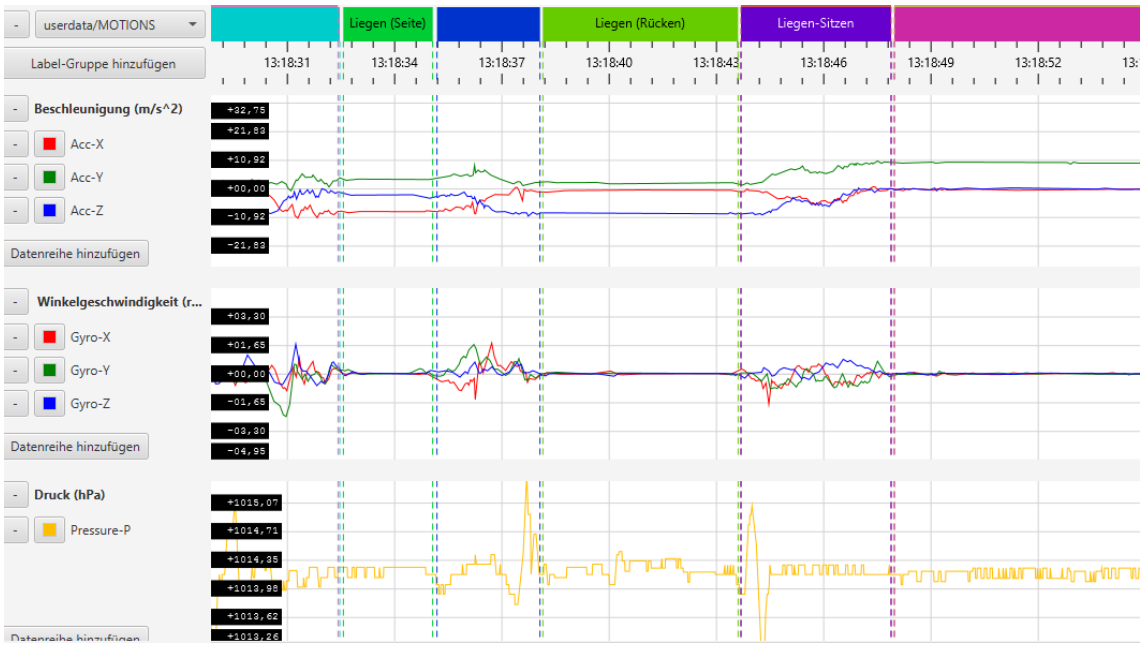


Abbildung 3: Bewegungsmusterbeispiel für lie on side (grün links), lie on back (grün Mitte) und sit (pink)

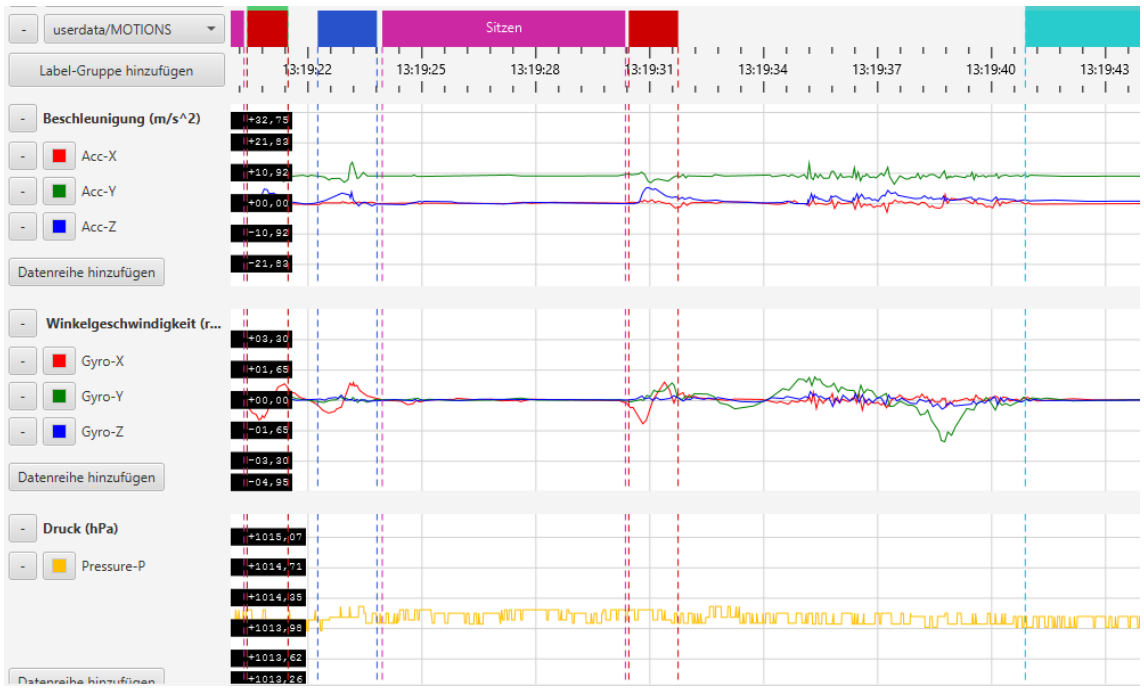


Abbildung 4: Bewegungsmusterbeispiel für stand-sit (blau), sit (pink), sit-stand (rot Mitte) und stand (türkis)

State-Klassifikator	**Feature-Set 1**	**Feature-Set 2**	**Feature-Set 3**	**Feature-Set 4**	**Feature-Parameter**
Auto-korrelation	Acc				NumLags = 10 bis 50
Korrelation	Acc	Acc	Acc	Acc	
Mean/RMS	Acc (beide) Gyr (RMS)	Acc (beide) Gyr (RMS)	Acc (beide)	Acc (Mean)	
Standard-abweichung	Acc Gyr	Acc Gyr	Acc	Acc	
Spektral-entropie	Acc Gyr	Acc	Acc	Acc	N = 5 bis 150
SMA/SMV					
Pitch					

Tabelle 1: Feature-Set-Vorschläge für den State-Klassifikator

Static-Klassifikator	**Feature-Sets 1 - 4**	**Feature-Sets 5 - 8**	**Feature-Sets 9 - 12**	**Feature-Sets 13 - 16**	**Feature-Parameter**
Auto-korrelation	Acc Gyr	Acc Gyr	Acc Gyr	Acc Gyr	NumLags = 10 bis 50
Korrelation	Acc Gyr	 Gyr			
Mean/RMS	Acc (beide)	Acc (beide)	Acc (beide)	Acc (beide)	
Standard-abweichung					
Spektral-entropie			Acc Gyr		N = 5 bis 200
SMA/SMV	Acc (beide)	Acc (beide) Gyr (beide)	Acc (beide)	Acc (beide)	
Pitch	Acc Gyr	Acc Gyr	Acc Gyr	Acc Gyr	Alpha = 98

Tabelle 2: Feature-Set-Vorschläge für den Static-Klassifikator

Dynamic-Klassifikator	**Feature-Sets 1 bis 4**	**Feature-Sets 5 und 6**	**Feature-Sets 7 und 8**	**Feature-Parameter**
Auto-korrelation	Acc Gyr Press	Acc Gyr Press	Acc Gyr	NumLags = 10 bis 50
Korrelation	Acc Gyr	Acc	Acc	
Mean/RMS	Acc (beide)			
Standard-abweichung	Acc Gyr	Acc	Acc	
Spektral-entropie				
SMA/SMV	Acc (beide) Gyr (beide)	Acc (beide)	Acc (beide)	
Pitch	Acc Gyr	Acc Gyr	Acc Gyr	Alpha = 98

Tabelle 3: Feature-Set-Vorschläge für den Dynamic-Klassifikator

Transition-Klassifikator	**Feature-Sets 1 bis 4**	**Feature-Sets 5 bis 8**	**Feature-Sets 9 bis 12**	**Feature-Parameter**
Auto-korrelation	Acc Gyr Press	Acc Gyr	Acc Gyr	NumLags = 10 bis 50
Korrelation	Acc Gyr	Acc		
Mean/RMS	Acc (beide) Gyr (Mean)	Acc (beide) Gyr (Mean)	Acc (beide) Gyr (Mean)	
Standard-abweichung	Acc Gyr	Acc		
Spektral-entropie	 Gyr			N = 5 bis 100
SMA/SMV	Acc (beide) Gyr (beide)	Acc (beide)	Acc (beide)	
Pitch	Acc Gyr	Acc Gyr	Acc Gyr	Alpha = 98

Tabelle 4: Feature-Set-Vorschläge für den Transition-Klassifikator

4 Fazit

Die Fenstergrößen und Schrittweiten der verschiedenen Sliding Windows sind nach den bisherigen Testergebnissen schon recht gut eingestellt, aber vielleicht führen eine erhöhte Fenstergröße oder noch zu testende Zwischenwerte bei Fenstergrößen und Schrittweiten zu einer Verbesserung des Klassifikationsergebnisses.

Das Feature-Set des State-Klassifikators ergab ohnehin schon eine gute Erkennung von Static-, Dynamic- und Transition-Bewegungsmustern, aber möglicherweise lässt sich mit den zusätzlichen Gyroskopdatenreihen, dem RMS-Feature und einer Autokorrelation mit einer auf die Schrittdauer angepassten Verschiebungsanzahl eine Verbesserung der Klassifikation erzielen. Auch das Feature-Set des Static-Klassifikators ergab schon recht gute Ergebnisse, doch eventuell kann die Erweiterung um das Feature Mean bzw. RMS sowie um die Gyroskopdatenreihen für die Autokorrelation das Klassifikationsergebnis steigern. Die Ergänzung des Dynamic-Klassifikators um die Standardabweichung, die Korrelation und die Gyroskop- sowie Barometerdatenreihen bei der Autokorrelation (kombiniert mit einer Anpassung der Verschiebungsanzahl) soll ebenfalls zu einer optimierten Erkennung führen. Auch für die Klassifikationsoptimierung bei den Transition-Bewegungsmustern könnten die Gyroskopdatenreihen hilfreich sein, ebenso wie die Akzelerometerdatenreihen des Mean- bzw. RMS-Features.

4.1 Ausblick

Sollte nach der Parameteroptimierung mit Hilfe eines evolutionären Algorithmus noch weiterer Verbesserungsbedarf bestehen, wird nachfolgend kurz erläutert, welche Optimierungsmöglichkeiten sich noch aus der Literaturrecherche ergeben haben.

4.1.1 Einsatz weiterer Features

- Spektralenergie:
  $Energy_{x-Axis} = \frac{\sum_{i=1}^{n} FFT_{x_i}^2}{n} - Mean_{x-Axix}$ [BAK+14]
- Maximal Difference Acceleration:
  Das Feature Maximal Difference Acceleration gibt die Differenz zwischen Maximum und Minimum einer Akzelerometerdatenreihe innerhalb eines Sliding Windows an. [GD14]
- Physical Features (z.B. arithmetisches Mittel und Varianz der Bewegungsintensität), die speziell für die Aktivitätserkennung entwickelt wurden und die verschiedene Sensordatenreihen kombinieren, um die menschliche Bewegung physikalisch zu beschreiben [ZS11]

4.1.2 Einsatz von Feature-Learning-Frameworks

Feature-Learning-Frameworks werden zur automatischen Entdeckung geeigneter Merkmale genutzt. Die so gefundenen Features verbessern das Klassifikationsergebnis der üblichen statistischen Merkmale deutlich. Eine Klassifizierungsoptimierung ist nach der Nutzung eines solchen Frameworks nicht mehr nötig. Nachfolgend werden zwei dieser Frameworks aufgelistet. [PHO11]

- Deep Learning [PHO11]
- PCA-based Feature-Learning [PHO11]

4.1.3 Einsatz von Feature-Selection-Methoden

Feature-Selection-Methoden bewerten einen existierenden Satz Merkmale nach ihrer individuellen Fähigkeit, eine anschließende Klassifikation zu ermöglichen. Im Folgenden werden drei dieser Methoden vorgestellt.

Filter-based Relief-F

Bei der Relief-F-Feature-Selection-Methode werden Features mit einem Gewicht versehen, das angibt, wie gut das Merkmal eine Klasse von den anderen abgrenzen kann. Wenn das Feature-Gewicht einen entsprechenden Schwellenwert übersteigt, wird es als relevant ange-

sehen und dem Feature-Set hinzugefügt. Redundante Merkmale werden hierbei allerdings leider nicht erkannt. [ZS11] [GD14]

Wrapper-based Single Feature Classification (SFC)

Die SFC-Feature-Selection-Methode verleiht den Features jeweils einen Rang, der die Genauigkeit der Klassifikation bewertet. Dem Feature-Set wird dann jeweils das Merkmal mit dem höchsten Rang hinzugefügt. Auch bei dieser Methode werden redundante Merkmale nicht entfernt. [ZS11]

Wrapper-based Sequential Forward Selection/Sequential Forward Floating Search (SFS/SFFS)

Auch die SFS/SFFS-Methode zur Merkmalsauswahl fügt immer die Features hinzu, die die höchste Klassifikationsgenauigkeit aufweisen, aber sie löscht auch die Merkmale mit der geringsten Klassifizierungspräzision, wenn sie das Ergebnis verschlechtern. Diese Methode erkennt redundante Merkmale und enthält eine Abbruchbedingung zur Vermeidung einer Endlosschleife. [ZS11] [GD14]

=> Die SFS/SFFS-Methode benötigt erheblich weniger Features als die Relief-F- und die SFC-Methode zur Merkmalsauswahl und erreicht dabei eine größere Genauigkeit. Allerdings nimmt sie dafür auch erheblich mehr Rechenzeit in Anspruch. [ZS11] [GD14]

Literatur

[AGO+13] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J.L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.

[BAK+14] Sebastian D. Bersch, Djamel Azzi, Rinat Khusainov, Ifeyinwa E. Achumba, and Jana Ries. Sensor data acquisition and processing parameters for human activity classification. In *Ambient Assisted Living (AAL): Sensors, Architectures and Applications*, pages 4239–4270. MDPI AG, Basel, Switzerland, 2014.

[BDP+12] Oresti Banos, Miguel Damas, Hector Pomares, Alberto Prieto, and Ignacio Rojas. Daily living activity recognition based on statistical feature quality group selection. *Expert Systems with Applications: An International Journal*, 39(9):8013–8021, jul 2012.

[GD14] Piyush Gupta and Tim Dallas. Feature selection and activity recognition system using a single triaxial accelerometer. *IEEE Transactions on Biomedical Engineering*, 61(6):1780–1786, mar 2014.

[KLLK10] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y. Lee, and Tae-Seong Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1166–1172, jun 2010.

[PHO11] Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. *IJCAI '11 Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, 2:1729–1734, jul 2011.

[Yan09] Jun Yang. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. *IMCE '09 Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, pages 1–10, oct 2009.

[ZS11] Mi Zhang and Alexander A. Sawchuk. A feature selection-based framework for human activity recognition using wearable multimodal sensors.

In *BodyNets '11 Proceedings of the 6th International Conference on Body Area Networks*, pages 92–98. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) ICST, Brussels, Belgium, nov 2011.

D.9. Interpretation der Projekt-Ergebnisse

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Interpretation der Projekt-Ergebnisse

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren

SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Daniel Patron

Oldenburg, den 7. April 2017

Abstract

Das Ziel dieser Arbeit ist es, die Ergebnisse der Klassifikation der Machine-Learning Techniken, welche dieser Arbeit zugrunde liegen, zu interpretieren und zu bewerten. Diese Aufgabe geschieht im Rahmen der Projektgruppe "Mobilitäts-Assessments mit körpernahen Sensoren". Dabei wird zunächst in die Grundlagen eingeführt, welche aus den Daten und der Be- sowie Verarbeitung dieser zusammen mit den Labeln zu den Datensätzen, der verwendeten Technologie in Form eines Sensorgürtels sowie aus den Machine-Learning Verfahren und dem genutzten Konzept der evolutionären Algorithmen bestehen. Anschließend folgt die Interpretation der Ergebnisse, die innerhalb dieser Arbeit gesammelt wurden und eine Bewertung dieser. Zum Abschluss folgt eine Zusammenfassung und ein Ausblick.

Inhalt

Abkürzungen

BDT	Boosted Decision Tree
BDS	Boosted Decision Stump
MLP	Multilayer Perceptron
AMM	Adaptive Multi-Hyperplane Machine
SVM	Support Vector Machine

Abbildungen

Tabellen

1 Einleitung

Innerhalb der Projektgruppe “Mobilitäts-Assessments mit körpernahen Sensoren“ gibt es die Notwendigkeit, mithilfe eines Sensorgürtels Bewegungsmuster wie beispielsweise Gehen, Sitzen, Aufstehen und Treppensteigen zu erkennen. Dieses Ziel wurde automatisiert mit verschiedenen Machine-Learning Techniken erreicht, zu denen Multilayer Perceptrons, Boosted Decision Trees, Boosted Decision Stumps und Adaptive Multi-Hyperplane Machines gehören.

Nachdem diese Algorithmen implementiert wurden, konnten auch schon erste Ergebnisse in der Erkennung von Bewegungsmustern auf den Daten des Sensorgürtels erreicht werden. Da diese aber noch nicht den Anforderungen an die Erkennungsgenauigkeit entsprachen, sollten diese durch die Anpassung von Parametern in Bezug auf ihre Genauigkeit und weitere Größen verbessert werden, was durch das Tunen von Parametern erreicht werden kann. Da dieses Parameter-Tuning manuell nicht in annehmbarer Durchführungszeit möglich ist, haben wir uns dazu entschieden, die Durchführung des Tunings von Evolutionären Algorithmen übernehmen zu lassen und so in annehmbarer Rechenzeit auf ein gutes Ergebnis zu kommen.

Die daraus resultierenden Ergebnisse wurden dann zum Einstellen der Machine-Learning Techniken genutzt und mithilfe dieser Grundlage kann dann auf den Daten trainiert werden. Die anschließende Klassifikation kann so verbessert werden, da eine möglichst annähernd optimale Lösung für die Klassifikation gefunden werden kann.

2 Motivation

Am Ende dieser Projektgruppe gibt es zu dem Ziel, mithilfe von Algorithmen Bewegungsmuster erkennen zu können auf zuvor aufgenommenen Daten, verschiedene Klassifikationsergebnisse und Klassifikationsgenauigkeiten, welche ausdrücken können, wie erfolgreich diese Algorithmen in Bezug auf dieses Ziel arbeiten. Dabei ist es aber äußerst wichtig, diese Ergebnisse nicht unkommentiert als gegeben hinzunehmen, sondern sie zu interpretieren und zu bewerten, da diese ansonsten nicht richtig verstanden und beispielsweise in Relation zu anderen Arbeiten und anderen Ergebnissen gesetzt werden können. Die Interpretation unserer Daten und der dazugehörigen Ergebnisse sowie eine Bewertung dieser ist das Ziel dieser wissenschaftlichen Ausarbeitung.

3 Thematische Grundlagen

Für ein grundlegendes Verständnis der in dieser Ausarbeitung hauptsächlich behandelten Interpretation der Ergebnisse der Klassifikation ist es notwendig, dass gewisse Grundlagen beim Leser vorhanden sind und unsere Arbeit verstanden wird, damit davon ausgehend die Daten interpretiert sowie bewertet werden können. Aus diesem Grund werden in diesem Kapitel grundlegende Informationen zu unseren verwendeten Daten, Label, Technologien, zu Machine-Learning Techniken sowie zu evolutionären Algorithmen beschrieben.

3.1 Machinelles Lernen

Unter maschinellem Lernen kann man das automatisierte Lernen verstehen. Dabei kann ein Computer durch verfügbare Daten aus diesen "lernen". Bei diesen verfügbaren Daten handelt es sich um einen sogenannten Trainingsdatensatz [SSBD14, p. 19].

In diesem Projekt wird dann nach dem Lernen diese Technik dazu genutzt, einen unbekannten Datensatz zu klassifizieren und so aus dem angelernten Wissen zu profitieren und das ganze auf einem neuen Problem zu nutzen. Die Klassifikation wird dabei auf einer hierarchischen Klassifikation genutzt, wobei zunächst der State-Klassifikator angewandt wird, durch den eingeteilt wird, ob es sich grundsätzlich um eine statische (static) oder dynamische (dynamic) Bewegung handelt, oder eben um eine Transition (transition). Anschließend werden die soeben genannten weiteren drei Klassifikatoren zur genaueren Erkennung genutzt.

Um in diesem Bereich nicht zu eingeschränkt vorzugehen und verschiedene Alternativen auszuprobieren, wurde entschieden, dass mehrere Machine-Learning Techniken eingesetzt werden - zu diesen gehören die BDTs, Boosted Decision Stump (BDS)s, Adaptive Multi-Hyperplane Machine (AMM)s und die Multilayer Perceptron (MLP)s. Zu jeder dieser Techniken werde ich folgend nun kurz Informationen liefern, damit diese Konzepte

grundlegend verstanden werden:

3.1.1 Boosted Decision Tree

Entscheidungsbäume sind weit verbreitet als Vorhersagemodell. Dabei ist dieses Modell eine Kombination aus inneren Knoten und Blattknoten. Jeder innere Knoten dieses Baumes besteht dabei aus einem Test, der auf ein einzelnes Attribut angewendet wird und die davon abgehenden Zweige repräsentieren die möglichen Ergebnisse dieses Tests. Ein Weg von dem Wurzelknoten bist zu einem Blattknoten repräsentiert dann eine Vorhersage zu einer spezifischen Klasse. Ein wichtiger Schritt bei Entscheidungsbäumen ist das split-Kriterium. Das zu splittende Attribut sollte anschließend in den daraus resultierenden Mengen an den Zweigen ein Ergebnis produzieren, das so rein wie möglich ist. Bezüglich der Auswahl dieses split-Kriteriums gibt es verschiedene Algorithmen wie “ID3“, “C4.5“ und “CART“ [WSX16, p. 1]. Das “Boosting“ hingegen beschreibt das Verfahren, bei dem sogenannte weak-learner, also schwache Lernalgorithmen, die leicht besser als bloßes Raten sind, miteinander kombiniert werden, um so einen besseren bzw. stärkeren Lernalgorithmus zu erschaffen [Sch99, p. 1]. Ein Beispiel dafür wäre AdaBoost oder SAMME, welches für Probleme mit mehreren Klassen genutzt werden kann und auf AdaBoost aufbaut.

3.1.2 Boosted Decision Stump

Ein BDS kann in fast gleicher Form genutzt werden, wie es schon im vorherigem Abschnitt zu den BDTs erläutert wurde - nur dass in diesem Fall als weak-learner anstatt eines Decision Trees ein Decision Stump genutzt wird.

3.1.3 Multilayer Perceptron

Das MLP ist eine Form der künstlichen neuronalen Netze. Dabei gibt es einmal einen “input vector“ und einen “output vector“, wobei die Daten daraus einmal in den input layer eintreten und aus dem output-layer austreten, wobei diese Layer über Knoten, oder sogenannte Neuronen, miteinander verbunden sind [GD98, p. 1]. Diese Knoten können in mehreren sogenannten “Hidden Layers“ auftreten [GD98, p. 2]. Dieses Zusammenspiel sorgt dafür, dass das MLP ein Modell darstellt, welches eine nicht-lineare Abbildung von einem Eingangsvektor auf einen Ausgangsvektor ermöglicht [GD98, p. 1]. Der Output

eines Knotens oder Neurons ist dabei der Input des nächsten Knotens, wobei die Verlaufsrichtung immer zu einer weiteren Ebene läuft und nicht in die entgegengesetzte Richtung. Dies ist von der Gewichtung der Kanten zwischen den Knoten abhängig. [GD98, p. 2]

3.1.4 Adaptive Multi-Hyperplane Machines

Grundlegend ist zu erwähnen, dass Support Vector Machine (SVM)s zu den bekanntesten und erfolgreichsten Algorithmen zur Klassifikation gehören. In diesem Kontext versuchen AMMs die Lücke zwischen linearen und nicht-linearen SVMs zu schließen und so schnelles Training und die Lösung von nicht-linearen Klassifikationsproblemen ermöglicht. Das Modell besteht dabei aus sogenannten “Hyperplanes“ (Gewichte), wobei jede zu einer Klasse gehört und sagt dabei die Klasse hervor, die mit ihrem Gewicht die größte Vorhersage bietet [WCDV11, p. 1]. Die Gewichte werden von einem iterativen Algorithmus bestimmt, welcher auf dem “stochastic gradient descent algorithm“ beruht, welcher eine Garantie dafür bietet, auf jeden Fall ein lokales Optimum zu finden [WCDV11, p. 1].

3.2 Technologie und Daten

Innerhalb dieser Projektgruppe werden Daten mithilfe eines Sensorgürtels der Firma Humotion aufgenommen und können dann weiter verarbeitet werden. Dieser Gürtel wird an der gleichen Position wie ein herkömmlicher Gürtel getragen. Zu den Sensoren, mit denen Daten während des Tragens aufgenommen werden, gehören Akzelerometer, Gyrometer, Magnetometer ein Drucksensor sowie ein Thermometer.

Die Daten, welche innerhalb der Projektgruppe zur Verarbeitung verwendet werden, stammen einmal aus der Versa-Studie, bei der Assessments mit Probanden durchgeführt wurden. Hierbei werden Senioren ab dem „70“. Lebensjahr innerhalb von Assessments getestet - dies geschieht im Hinblick auf „Muskelkraft“, „Gleichgewichts-„ und „Gehfähigkeit“ [VER]. Weiterhin wurden auch zusätzliche Daten von den Mitgliedern der Projektgruppe aufgenommen, um die Anzahl an Daten zu erhöhen.

Die Bewegungen und Bewegungsmuster können in drei verschiedene Kategorien eingeteilt werden, wobei diese verschiedene Bewegungstypen haben:

1. Statisch

a) Stehen

b) Sitzen

c) Liegen (Rücken)

d) Liegen (Seite)

2. Dynamisch

a) Treppensteigen

b) Gehen

c) Springen

d) Gehen (Rückwärts)

3. Transition

a) Stehen-Sitzen

b) Sitzen-Stehen

c) Liegen-Sitzen

d) Liegen(Rücken)-Liegen(Seite)

e) Liegen(Seite)-Liegen(Rücken)

f) Hinunterbeugen (Hocken)

g) Umdrehen (links)

h) Umdrehen (rechts)

Während der Assessments mit den Probanden wurden in der Regel diese Bewegungen ausgeführt (bzw. es wurde der Versuch unternommen, sofern möglich). Im Falle der Zusatzaufnahmen durch Projektmitglieder wurde nur eine kleine Teilmenge dieser Daten zusätzlich aufgenommen, und zwar betrifft dies das Treppensteigen, seitliches Liegen, Aufstehen und Hinsetzen sowie das Springen.

Diese Bewegungen sind in den Datensätzen vorhanden, die von unserer Projektgruppe verarbeitet werden, jedoch werden nicht all diese Bewegungsmuster von den Machine-Learning Techniken trainiert und schlussendlich klassifiziert. Zu diesen gehören Folgende:

1. Statisch

a) Stehen

   b) Sitzen

   c) Liegen (Rücken)

   d) Liegen (Seite)

2. Dynamisch

   a) Laufen

   b) Springen

   c) Treppensteigen

3. Transition

   a) Sitzen-Stehen

   b) Stehen-Sitzen

Diese Bewegungsmuster müssen dann innerhalb der anderen Daten für das Training nutzbar gemacht und letztendlich klassifiziert werden.

3.3 Label für die Datensätze

Damit diese Klassifikation erst einmal möglichst korrekt ablaufen kann, ist es notwendig, dass ein vorheriges Training der Machine-Learning Techniken geschieht. Mithilfe von Labeln wird markiert, welcher Datenbereich welchem Bewegungsmuster entspricht. Dadurch ist es möglich, dass der Algorithmus weiß, wie beispielsweise das Treppensteigen in Bezug auf die Daten der Sensoren aussieht und kann daraus lernen. Diese Label wurden bei uns nicht durch eine Methode mit einem Gold-Standard oder durch Referenzsysteme wie Lichtschranken erstellt, sondern manuell mithilfe einer Software, die im Rahmen dieser Projektgruppe erstellt wurde. Die Personen, die diese Daten gelabelt haben, waren dabei auch die Mitglieder dieser Projektgruppe oder Mitarbeiter der Abteilung, in der dieses Projekt stattfindet. Es gibt für jede Bewegung bzw. für jedes Bewegungsmuster, welches in den Assessments auftauchen kann, ein dazugehöriges Label in der Software. Da jedoch nur eine Teilmenge aller Bewegungsmuster von den Algorithmen erkannt wird, wurde dabei der Fokus auf das Labeln dieser Bewegungen gelegt und Weitere teils ausgelassen.

3.4 Evolutionäre Algorithmen

Evolutionäre Algorithmen sind grundlegend inspiriert worden aus der Biologie und werden zur Optimierung genutzt. In den letzten Jahrzehnten wurden sie erfolgreich zur Lösung von Optimierungsproblemen angewendet [Kra16, p. abstract]. Das zugrundeliegende Konzept dahinter stammt von “Charles Darwin“, der als erster dieses vorgeschlagen hat [Kra16, p. 6]. Dies ist eine Erklärung zur biologischen Entwicklung durch Selektion und der Annahme von “survival of the fittest“ [Kra16, p. 6]. In der folgenden Abbildung ist der Ablauf der Evolution schematisch dargestellt.

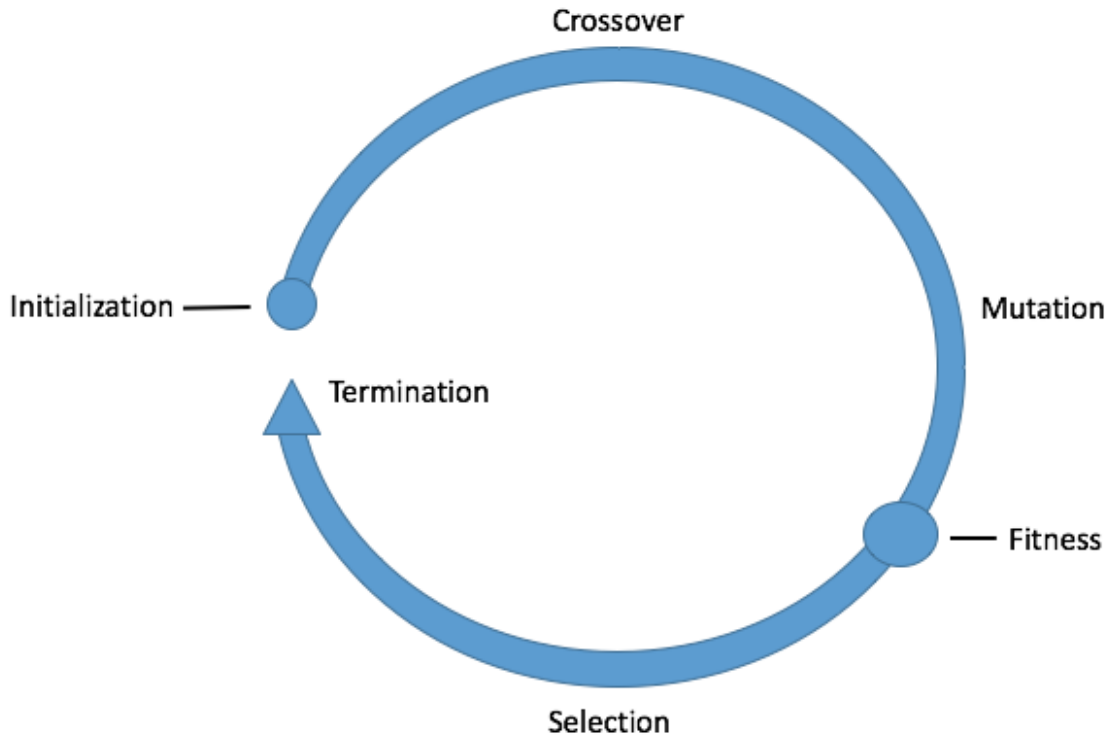


Abbildung 1: Schematische Darstellung des Ablaufs der Evolution (In Anlehnung an [Kra16, p. 6])

Hierbei beginnt der Prozess mit zufällig ausgewählten oder manuell eingestellten Lösungen. Anschließend werden zwei oder mehr Lösungen durch “[C]rossover“ rekombiniert [Kra16, p. 7]. Beim “Crossover“ Operator wird das genetische Material von zwei oder mehr Lösungen miteinander kombiniert [Kra16, p. 16]. Die daraus hervorkommenden Lösungen werden anschließend mutiert [Kra16, p. 7]. Bei der Mutation werden zufällige Änderungen an der Lösung vorgenommen - die Stärke dieser Änderung hängt von der sogenannten Mutationsrate ab [Kra16, p. 19]. Zum Schluss wird mithilfe der Fitness der Lösungen geschaut, welche Lösungen am besten sind und diese werden für die nächste Generation ausgewählt. Dieser Kreislauf startet an dieser Stelle dann so lange erneut von vorne, bis

eine festgelegte Abbruchbedingung eintritt. [Kra16, p. 7]

4 Interpretation

Dieses Kapitel stellt das Kernstück dieser wissenschatlichen Ausarbeitung dar. Hier möchte ich außerdem ansprechen, wie und durch was die Daten und Ergebnisse beeinflusst wurden und wie genau diese verarbeitet und verändert wurden. Dies werde ich in einige Unterabschnitte einteilen. Anschließend folgt die Interpretation. Zum Abschluss werde ich die Daten bewerten.

4.1 Einflüsse auf die Interpretation

In diesem Abschnitt werde ich zunächst auf die Einflüsse eingehen, die bei der Interpretation zu beachten sind und grundlegend die Sicht auf die Daten und die Ergebnisse beeinflussen.

4.1.1 Daten, Aufnahme und Label

Wie schon zu Beginn dieser Arbeit beschrieben, wurden die Daten mithilfe eines Sensorgürtels der Firma Humotion während geplanten Assessments und außerplanmässigen Zusatzaufnahme-Treffen erhoben. Weiterhin gab es keinen Standard, der die Daten fehlerfrei zu einem Bewegungsmuster zuordnen konnte oder sonstige Referenzen, die wir für diesen Zweck nutzen konnten. Lediglich ein (annähernd) standardisierter Ablauf der Assessments und der Zusatzaufnahme-Treffen hat dafür gesorgt, dass wir ausgehend von diesen Abläufen abschätzen konnten oder genau wussten, zu welchem Zeitpunkt welche Daten ungefähr welcher Bewegung entsprechen mussten - dieses Wissen wurde dann dazu genutzt, die Daten manuell zu labeln und so den Daten eindeutig ein Bewegungsmuster zuzuordnen. Dabei ist zu vermerken, dass die Software, mit der dieses manuelle Einfügen von Labeln ermöglicht wurde, von den Mitgliedern der Projektgruppe programmiert wurde und dass auch entweder die Mitglieder des Projektes oder Mitarbeiter aus der Abteilung,

in der dieses Projekt stattfand, die Daten per Hand gelabelt haben. Dabei wurde teilweise auch eine größere Menge an Daten nacheinander gelabelt, was zum Einen einiges an Zeit beansprucht hat und weiterhin relativ monotone Arbeit war - was insgesamt dazu führen konnte, dass über diese Zeit Fehler nicht rechtzeitig gesehen oder selber hinzugefügt wurden. Dadurch ist es möglich und wahrscheinlich, dass die Daten, mit denen die Machine-Learning Algorithmen trainieren, fehlerhaft gelabelte Daten aufweisen können sowie ungenau platzierte Label, aber auch die Klassifikationsdatensätze können ebenfalls fehlerhaft sein, sodass auch die Ergebnisse, ob nun gut oder schlecht, kritisch begutachtet werden müssen. Weiterhin kann es jedoch möglich sein, dass in einer Teilmenge der Datensätze nicht alle Bewegungen enthalten sind, da eventuell Probanden bestimmte Bewegungen nicht ausführen konnten. Zudem wurden nicht erkennbare Bewegungen auch nicht gelabelt und verändern somit die Anzahl an erkennbaren und erlernbaren Bewegungen unter den Datensätzen.

Weiterhin ist zu beachten, dass die Verteilung der Datensätze wie folgt aussieht. Insgesamt gibt es 87 Datensätze, die aus der Versa-Studie kommen und somit Assessmentaufnahmen enthalten und an Zusatzaufnahmen gibt es insgesamt 31 Datensätze. Innerhalb dieser Datensätze gibt auf der Seite der Assessmentaufnahmen grundsätzlich alle 16 Bewegungsmuster, die darin enthalten sein können. Wie im vorherigen Abschnitt beschrieben, können aus den genannten Gründen auch bestimmte Bewegungsmuster nicht vorhanden sein. In den Zusatzaufnahmen wurden nur bestimmte Bewegungen wie Treppesteigen, Springen und Aufstehen sowie Hinsetzen und seitliches Liegen aufgenommen sowie gelabelt. Somit liegt der Verarbeitung der Daten eine große Datenmenge mit vielen unterschiedlichen Bewegungsmustern zugrunde, die dann möglichst gut klassifiziert werden sollen.

Ein weiterer Punkt, der jedoch dabei beachtet werden sollte, ist der Zustand, dass nicht jede Klasse von der Häufigkeit her gleich oft in den Datensätzen vertreten ist, was dafür sorgt, dass unterschiedlich viele Daten zum Training vorhanden sind. Dies wird mithilfe von Over- und Undersampling versucht zu verhindern, da in der Implementierung angegeben wird, wie viele Beispiele zum Training benötigt werden - so wird für eine bessere Verteilung der Daten gesorgt.

4.1.2 Machine Learning und Klassifikation

Um zu erkennen, welcher Machine-Learning Algorithmus nun die Daten wie gut klassifizieren kann, werden dazu verschiedene Messgrößen angewandt. Dazu gehört einmal der sogenannte “Recall“, der angibt, wie viele wirklich positive Werte auch als positiv klassifiziert worden sind [Pow07, p. 2]. Die “Precision“ hingegen gibt an, wie viele positiv klassifizierte Werte auch in Wirklichkeit positiv benannt werden [Pow07, p. 2]. “Accuracy“ gibt weiterhin an, wie viele Werte als positiv klassifiziert worden - egal, ob dies auch den Tatsachen entspricht, oder nicht [Pow07, p. 3]. Der “F-Score “, mit ß=1 als generelles Maß, ist ein gewichtetes harmonisches Mittel von “Recall“ und “Precision“ [Pow, p. 1].

Diese Informationen sind im Rahmen dieser Interpretation wichtig zu kennen, da durch die Messgrößen ermittelt wird, wie erfolgreich eine Machine-Learning Technik funktioniert und gearbeitet hat und wie gut sie nach dem Trainieren auf verschiedenen Datensätzen einen weiteren Datensatz klassifiziert hat. Der Wert des F-Scores wird aber in unserem Fall auch weiterführend verwendet, da der verwendete evolutionäre Algorithmus zur Bestimmung der optimalen Parameterkonstellation diesen Wert als Fitness der einzelnen Individuen einer Population verwenden.

4.1.3 Umsetzung der evolutionären Algorithmen

In diesem Zusammenhang ist es auch wichtig, speziell auf den verwendeten evolutionären Algorithmus und dessen Spezialisierung einzugehen. Dabei soll auch darauf eingegangen werden, wie die Konzepte eines solchen Algorithmus in unserem Fall angewandt wurden.

Im Falle dieser Projektgruppe wird eine Abwandlung des “(1+1)“-Evolutionärer Algorithmus, bei dem normalerweise ein Elternteil mutiert wird und dann zum Kind wird, also in der nächsten Generation weiterverwendet wird [Kra16, p. 7]. In unserer Implementierung können X Elternteile genutzt werden, um Y Kinder zu erstellen. Im Fall dieser Projektgruppe werden immer drei Elternteile genutzt, um insgesamt neun Individuen daraus zu reproduzieren. Dadurch ist es möglich, mehrere verschiedene Nachkommen zu erzeugen, die durch Rekombination verschiedener Eltern zusammengesetzt werden und auch durch Mutation verändert werden. Die Mutation von Genen erfolgt dabei zufällig, jedoch wurden die zwei Parameter dabei speziell berücksichtigt, und zwar die Größe des Sliding Windows und die Schrittweite, da diese voneinander abhängig sind, denn wenn die Schrittweite des

Sliding Windows größer als die eigentliche Größe ist, werden Daten übersprungen.

Das Erbgut dieser Individuen besteht aus Genen, welche verschiedene Parameter repräsentieren. Hierbei sind dies einerseits Parameter, die die jeweilige Machine-Learning Technik betreffen, sodass je nach Technik, die gerade von den evolutionären Algorithmen optimiert wird, sich diese Parameter in Bezug auf Boosted Decision Trees, Boosted Decision Stumps, Adaptive Multi-Hyperplane Machines und Multilayer Perceptrons verändern. Dazu gehört natürlich auch die Fitness, die den F-Score dieser Einstellungen der Parameter für ein Individuum berechnet - also wird mit der jeweiligen Machine-Learning Technik ein Training sowie eine Klassifikation ausgeführt und das dazugehörige Ergebnis betrachtet. Speziell dabei zu beachten ist dabei die größte Anzahl der Parameter, welche in unserer Umsetzung zum Einen zuvor mit Startwerten festgelegt wurden, zum Anderen ein Minimal und Maximalwert festgelegt wurde. Dieses Prinzip der vorherigen Auswahl und Festlegung von Parametern wurde auch in Bezug auf FeatureSets festgelegt, die je nach Klassifikator (State, Static, Dynamic oder Transition) im Vorraus ausgesucht wurden. Dies geschah in den Fällen mit vorheriger Recherche und durch Testen, jedoch grenzt das den Suchraum deutlich ein und damit auch den Lösungsraum und es ist natürlich möglich, dass bei dieser Auswahl auch Fehler geschehen sind und diese sich so auf den weiteren Verlauf und die Optimierung auswirken. Dadurch ist es möglich, dass bestimmte Lösungen, wozu natürlich auch ein eventuelles Optimum im Lösungsraum gehören kann, nicht mehr gefunden werden können, da die Grenzen um Dieses gelegt wurden. Der Grund für diese Maßnahmen war das Problem, dass eine vollständige Suche bei einem so großen Lösungsraum in der uns verfügbaren Zeit und mit der Rechenleistung nicht umsetzbar gewesen wäre. Da wir darauf angewiesen waren, dass die Ergebnisse in relativ kurzer Zeit zur Verfügung standen, wurden insgesamt fünf Durchläufe auf dem Rechencluster “CARL“[UOL] der Universität Oldenburg durchgeführt. Die ersten beiden Optimierungsdurchläufe wurden dabei noch mit dem initialen Bewertungsverfahren durchlaufen, die nächsten drei (der aktuellste Stand dieser Ausarbeitung) wurden mit einem veränderten Bewertungsverfahren durchlaufen - auf die genauen Spezifikationen und Veränderungen dieser werde ich in der hauptsächlichen Interpretation eingehen, in der Durchläufe aus jeder Phase miteinander verglichen werden. Wichtig zu erwähnen ist, dass diese Optimierungsdurchläufe aufeinander aufbauen, sodass diese nicht unabhängig sind, sondern ein neuer Durchlauf die Ergebnisse des vorherigen Durchlaufes (außer, es ist der initiale Durchlauf, dort werden die Startwerte manuell festgelegt) als Startwert übergeben bekommt und mit diesen dann

weiter arbeitet und die Ergebnisse weiter optimierung will.

Bei evolutionären Algorithmen ist es notwendig, dass eine Abbruchbedingung angegeben wird, bei der dieser Algorithmus terminiert [Kra16, p. 7]. Diese Abbruchbedingung ist in unserem Fall nicht dass erreichen beispielsweise einer Lösung, die ausreichend gut ist oder dem Zustand, dass über einen Zeitraum x keine bessere Lösung gefunden wird, sondern es wird ein Budget an Zeit zuvor festgelegt, wobei der Algorithmus dann terminiert, wenn dieser Zeitpunkt erreicht ist. Die evolutionären Algorithmen im zweiten sowie vierten und fünften Durchlauf wurden jeweils für 24 Stunden auf dem Rechencluster der Universität belassen - beim ersten sowie dritten Durchlauf gab es Probleme und die Jobs sind nicht ganz durchgelaufen. Dies könnte eventuell an Speicherproblemen der Multilayer Perceptrons liegen, da diese in jedem der vier Durchläufe Probleme mit diesem Bereich hatten und so nicht zu vollständigen Ergebnissen gelangen können.

Im Falle dieser Implementierung wird zur Berechnung der Fitness eines Individuums, von dem es bis zu neun innerhalb einer Population geben kann, jeweils ein Training auf den Daten (die Datenmenge wurde zuvor in dieser Ausarbeitung angsprochen) ausgeführt und anschließend eine Klassifikation, für die der F-Score berechnet wird und damit die Fitness angibt.

4.2 Interpretation in Bezug auf Optimierungsdurchläufe und Klassifikationsergebnisse

In diesem Abschnitt werde ich nun speziell auf die Interpretation der Optimierungsdurchläufe der evolutionären Algorithmen und Klassifikationsergebnisse eingehen.

Wie schon im vorangegangenen Abschnitt erwähnt, wurde zwischen den ersten beiden Durchläufen und den anschließenden drei Durchläufen das Bewertungsverfahren verändert. Dieses betrifft die Berechnung der Fitness der Individuen, da sich die Bewertung auf das Trainung und somit auf die Klassifikation auswirkt.

Bis zum Einsatz einer neuen Bewertungsfunktion wurde nämlich das Training während der Optimierungsdurchläufe einzelner Individuen so durchgeführt, dass der SplitTrainer eine Datenmenge von 0%, wodurch nur vom Nutzer direkt markierte Daten als ausschließliche Trainingsdaten verwendet werden. Dadurch wurden keine echten Daten zum Training

genutzt. Mit der Veränderung der Datenmenge beim SplitTrainer zum Training auf 1.0 wird der gesamte Datensatz zum Training und zur Validierung bereitgestellt.

Im Normalfall würde durch den aufeinander folgenden Ablauf unserer Optimierung, wobei die Ergebnisse des einen Durchlaufs als Startwerte des nächsten Durchlaufs genutzt werden, dazu führen, dass die Ergebnisse immer besser werden, bis sie ein lokales oder globales Optimum gefunden haben und sich diesem immer weiter annähern. Durch die Veränderung der Bewertungsfunktion ist es aber nun so gekommen, dass die Ergebnisse sich verschlechtert haben. Dies habe ich nun einmal exemplarisch vorbereitet im Fall der Boosted Decision Trees bei der Erkennung des statischen Klassifikators im zweiten Optimierungsdurchlauf:

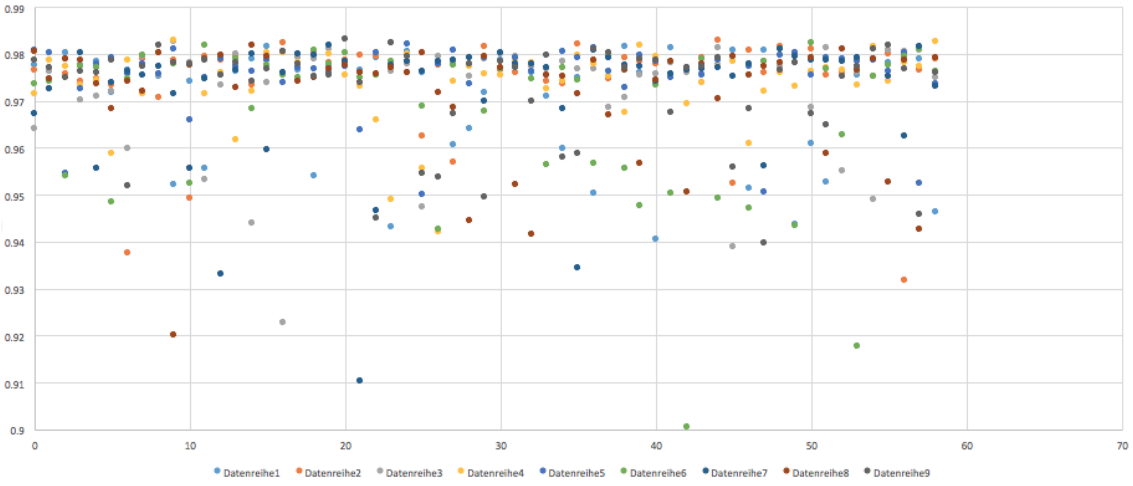


Abbildung 2: Zweiter Optimierungsdurchlauf der BDTs beim Static-Klassifikator

Diese Grafik beschreibt den Verlauf der Fitness von insgesamt neun verschiedenen Individuen in jeweils einer Population, die sich über mehrere Generationen verändern durch Reproduktion, Selektion und Mutation. Dabei wird auf der vertikalen Achse die Fitness angegeben, wobei ein Wert von 1 dabei 100% entsprechen würde. Auf der horizontalen Achse befinden sich dabei die verschiedenen Generationen. Die unterschiedlich farbigen Punkte, die in der Legende als Datenreihe 1-9 betitelt sind, repräsentieren dabei ein Individuum in der jeweiligen Generation. Dabei ist zu erkennen, dass der beste Wert der Fitness bei 0.983 liegt. Andere Individuen auch schon in früheren Generationen hatten aber auch schon einen grundlegend hohen Fitness-Wert.

In der nächsten Abbildung ist nun hingegen das erneuerte Bewertungsverfahren genutzt worden, welches die Ergebnisse eindeutig verändert. Dies ist auch am Beispiel der Boosted Decision Trees bei der Erkennung des statischen Klassifikators im vierten Optimierungs-

durchlauf zu sehen. Der fünfte Durchlauf wurde nicht mehr innerhalb dieser Ausarbeitung verwendet, da eine signifikante Verbesserung der Daten nicht mehr zu vermerken ist und der fünfte Durchlauf erst während der Auswertung der Daten vollendet wurde, sodass ich mich dazu entschieden habe, diese Daten nicht mehr für diese Ausarbeitung zu nutzen.

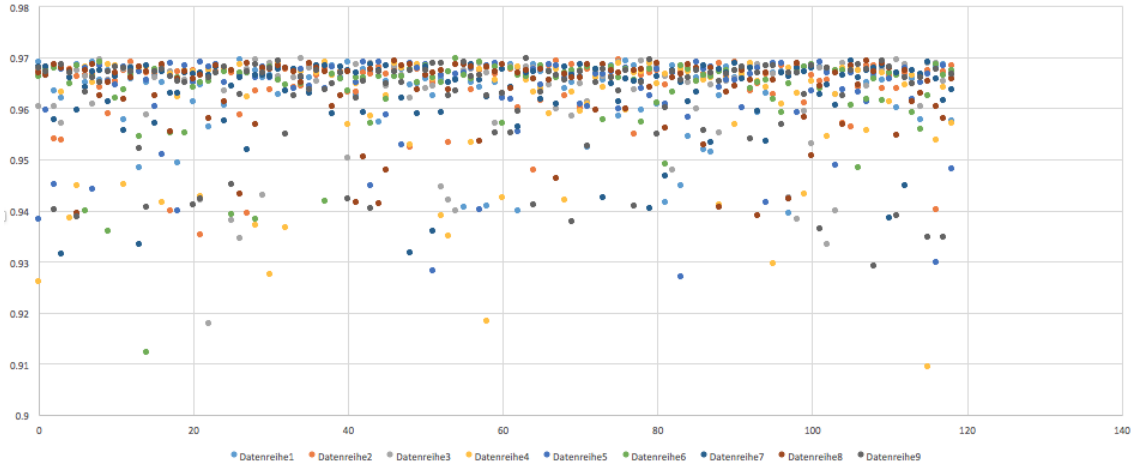


Abbildung 3: Vierter Optimierungsdurchlauf der BDTs beim Static-Klassifikator

Bei dieser Grafik ist zu erkennen, dass die besten Lösungen schlechter als 97% Fitness aufweisen, obwohl noch eine Generation der Optimierung dazwischen liegt und sich eigentlich die Fitness im Normalfall weiter gebessert haben müsste - durch die Veränderung des Bewertungsverfahrens jedoch hat sich die Fitness insgesamt verschlechtert. Der beste Wert liegt hier nun bei 0.969. Weiterhin ist typisch für evolutionäre Algorithmen zu erkennen, dass zu Beginn zwar durch unsere Vorauswahl der Parameter schon hohe Fitness-Werte vorzufinden sind, doch durch das Annäherungsverfahren und die Selektion der Nachkommen bei evolutionären Algorithmen nähern sich die Lösungen der verschiedenen Individuen zum Ende hin immer weiter einer optimalen Lösung an. Die Ausschläge in den positiven sowie negativen Bereich sind dadurch zu erklären, dass mithilfe von Rekombination und Mutation auch Nachkommen erzeugt werden können, die mit ihren neu erschaffenen Genen entweder sehr gut an die Umgebung angepasst sind, oder aber sehr schlecht. Diese werden dann aber im negativen Falle nicht weiter verwendet und werden nicht zur Rekombination genutzt. Da es nicht auszuschließen ist, dass auch durch Wechselwirkungen von bestimmten Parameterkombinationen besonders gute Lösungen hervorgebracht werden können, ist es nicht in jedem Fall möglich zu sagen, weshalb ein Individuum nun aufgrund der dazu enthaltenen Parameter eine besonders gute oder schlechte Fitness aufweist.

Diese Veränderung bzw. Verschlechterung der Fitness, die insgesamt zu bemerken ist,

wird auf das Training mitt realen Daten zurückzuführen sein, was sich aus dem neuen Bewertungsverfahren ergeben hat.

Im Folgenden werde ich nun auf die Klassifikationsergebnisse des vierten Durchlaufs eingehen. Dazu werde ich die Ergebnisse der einzelnen Klassifikatoren bei den verschiedenen Machine-Learning Techniken vorstellen und dies exemplarisch mit Diagrammen darstellen.

In der folgenden Tabelle wurden die Ergebnisse der einzelnen Klassifikatoren zu den Machine-Learning Techniken zusammengestellt. Innerhalb der Zellen befindet sich der F-Score:

Tabelle 1: Ergebnisse des vierten Optimierungsdurchlaufs

	State	Static	Dynamic	Transition
BDT	0.966	0.969	0.954	0.931
BDS	0.955	0.947	0.936	0.899
MLP	0.964	0.973	0.975	0.948
AMM	0.952	0.971	0.938	0.912

Aus dieser Tabelle wird deutlich, dass die Multilayer Perceptrons in jeder Kategorie mit den Ergebnissen beim F-Score führend sind, bis auf den State-Klassifizierer, den die Boosted Decision Trees mit einem kleinen Vorsprung besser klassifizieren können. Hierbei ist jedoch zu beachten, dass die Multilayer Perceptrons bei den Klassifikatoren Dynamic sowie Transition Speicherprobleme im Log aufgewiesen haben.

Weiterhin habe ich der Vollständigkeit halber auch die Ergebnisse des zweiten Optimierungsdurchlaufs tabellarisch (Tabelle 2) aufbereitet.

Tabelle 2: Ergebnisse des zweiten Optimierungsdurchlaufs

	State	Static	Dynamic	Transition
BDT	0.998	0.983	0.983	1.0
BDS	0.981	0.940	0.967	0.964
MLP	0.991	0.986	0.976	0.973
AMM	0.989	0.966	0.983	0.964

Aufgrund der Auswertung dieser Tabelle wurden die optimalen Parameter der besten Machine-Learning Technik für den jeweiligen Klassifikator ausgewählt. Folgende Datensätze wurden dabei zum Training genutzt mit einem SplitTrainer(0.7): 11547, 11663, 11686, 11948, 12117, 12160, 12281, 12527, 12743, 12903, 13529.

Anschließend sollte mit den optimalen Parametern von beiden Durchläufen eine Training

sowie mit dem trainierten Modell eine Klassifikation durchgeführt werden - dies war aber aufgrund von Fehlern lokal sowie auf dem Rechencluster nicht möglich. Vermutlich ist dies zu einem späteren Zeitpunkt mögich.

Weiterhin habe ich ausgewählte Parameter des zweiten Durchlaufs auf Auffälligkeiten untersucht und dementsprechend interpretiert. Zunächst werde ich auf den Parameter "Window Stride" eingehen, die die Schrittweite des Sliding Windows darstellt. Dabei ist keinerlei Konvergenz oder Annäherung an einen bestimmten Wert im zweiten Lauf zu vermerken. Die Werte wirken dabei zufällig verteilt und haben ab und zu Ausreißer. Zur Verdeutlichung dient folgendes Diagramm (4):

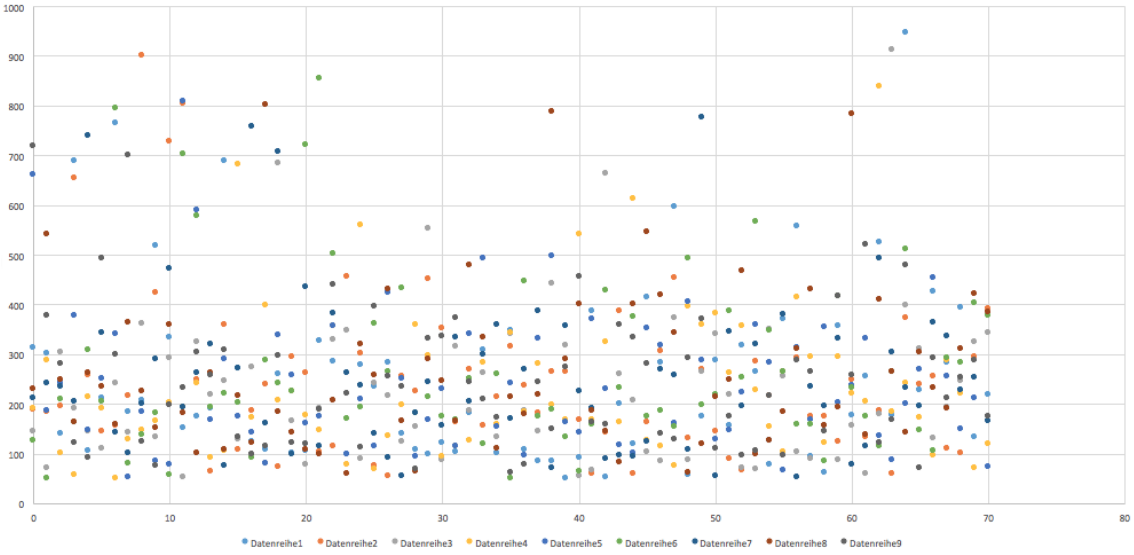


Abbildung 4: Sliding Window Stride bei den Boosted Decision Trees mit Dynamic-Klassifikator

In 4 sind auf der X-Achse die Generationen zu sehen und auf der Y-Achse die Stride-Größe als Integer, wobei dieser die Größe in Indizes repräsentiert. Die neun verschiedenen Datenreiehen in unterschiedlichen Farben stellen die neun verschiedenen Individuen innerhalb einer Population zu der jeweiligen Generation X dar. Insgesamt kann man diesem Diagramm (stellvertretend für die anderen Klassifikatoren mit den Machine-Learning Techniken) entnehmen, dass kaum Konvergenz oder bestimmte auffällige Annäherung an einen Wert zu vermerken sind und dass sich lediglich die meisten Schrittweiten zwischen 100 und 400 Indizes verteilen. Bei weiteren Klassifikatoren und Machine-Learning Techniken sind auch grobe Bereiche zu erkennen, wie beispielsweise bei den Boosted Decision Stumps mit dem State-Klassifikator, der Adaptive Multi-Hyperplane Machine mit dem Static-Klassifikator und Weiteren. Dabei ist aber nur ein solcher Bereich zu erkennen und noch kein genaues Ergebnis - dieses Verhalten ist jedoch typisch für Evolutionäre

Algorithmen und dementsprechend ein Optimierungsverlauf, der sich einer optimalen Lösung annähert.

Bei dem nächsten Parameter, der Fenstergröße ist teilweise aufgrund der Optimierungsdurchläufe eher ein Annäherungsverhalten zu bemerken. Dazu beziehe ich mich einmal beispielhaft auf die Grafiken im folgenden Abschnitt, welche den gleichen Aufbau haben wie die Vorangegangenen, jedoch befindet sich auf der Y-Achse die Größe des Sliding Windows in Indizes.

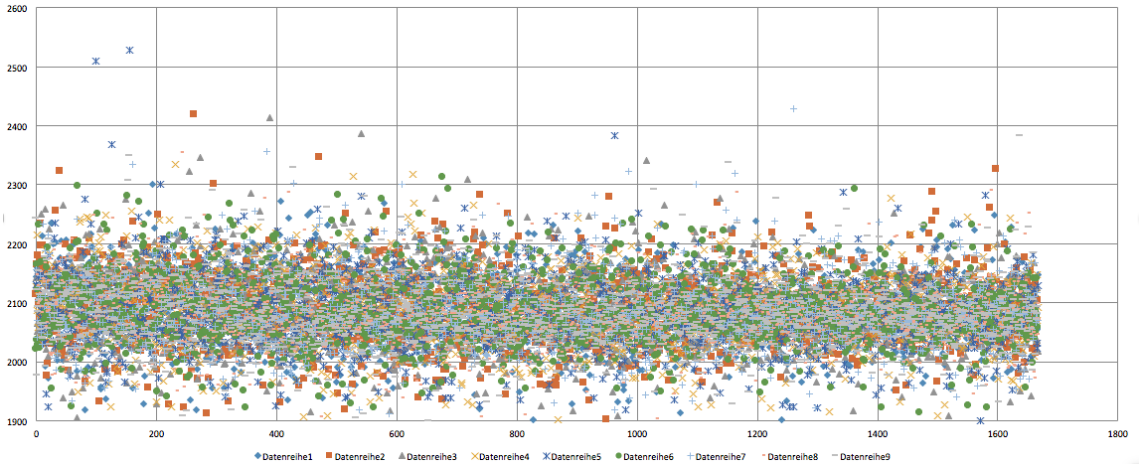


Abbildung 5: Sliding Window Length bei den Adaptive Multi-Hyperplane Machines mit State-Klassifikator

In 5 ist nun eindeutig eine Annäherung der einzelnen Individuen innerhalb der Population pro Generation an einen Bereich zwischen 1900 und 2200 Indizes zu vermerken. Dies bedeutet, dass für diesen Klassifikator bei der Machine-Learning Technik der Adaptive Multi-Hyperplane Machines eine Optimierungskurve zu vermerken ist, sodass eine Annäherung hier schon weiter fortgeschritten sein muss.

In der nächsten Abbildung(6) hingegen ist eine solche Konvergenz noch nicht so abzusehen. Dort ist die Verteilung der Werte noch eher scheinbar durch Zufall bestimmt und verläuft noch nicht in eine eindeutige Richtung, sodass sich die hauptsächlichen Werte auch meist zwischen 1000 und 2600 Indizes befinden.

Im Hinblick auf den “Noise Reduction Filter“ wäre es auch hilfreich und interessant zu wissen, wie oft innerhalb einer Population zum Zeitpunkt X (bzw. in der dazugehörigen Generation) welcher Filter genutzt wurde. Um dies zu veranschaulichen, wurden innerhalb mehrere Punkte-Diagramme diese Informationen visualisiert. Dies wurde nur in der optimalen Parameterkonstellation einer Machine-Learning Technik und dem zugehörigen Klassifikator betrachtet.

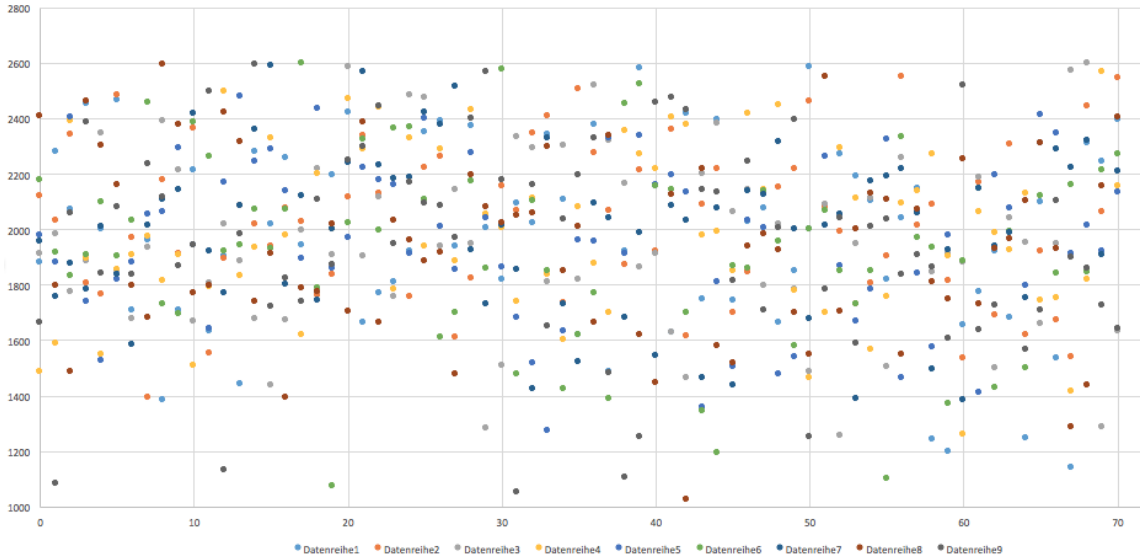


Abbildung 6: Sliding Window Length bei den Boosted Decision Trees mit Dynamic-Klassifikator

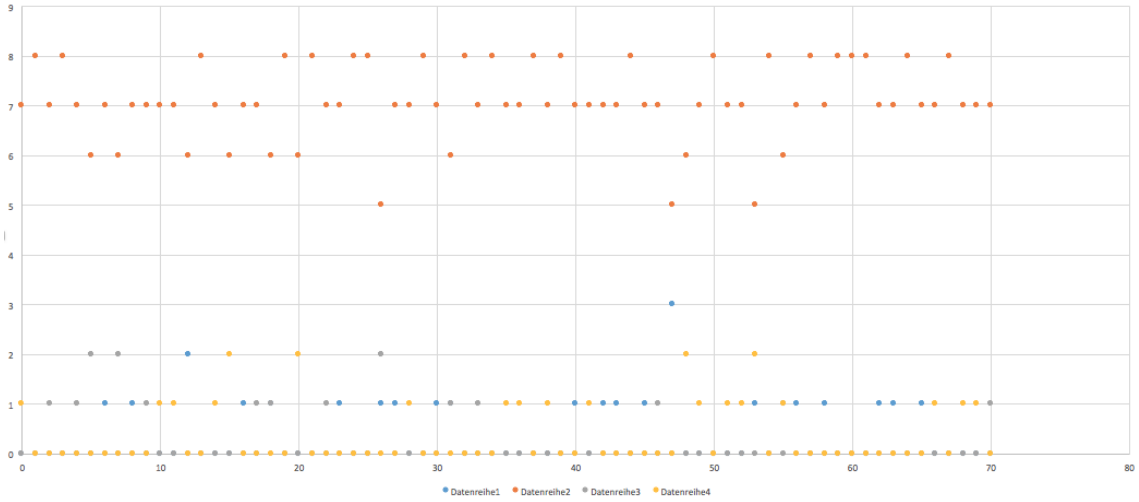


Abbildung 7: Verwendung des Noise-Filters bei den Boosted Decision Trees mit Dynamic-Klassifikator

Eine kurze Erklärung: Dieses Diagramm (7) zeigt die Verteilung der verschiedenen Noise-Filter in den Generationen (X-Achse) mit dem jeweiligen Vorkommen als natürliche Zahl (Y-Achse). Die vier farbigen Datenreihen stehen dabei für die folgenden Daten: Datenreihe1 = Kein Filter, Datenreihe2 = LowpassFilter, Datenreihe3 = Gaussfilter, Datenreihe4 = Medianfilter. Hierbei ist ohne Probleme zu erkennen, dass hauptsächlich der LowpassFilter verwendet wird.

Beim State-Klassifikator (8) hingegen wird hauptsächlich kein Noise-Filter verwendet, was sich fast durch den kompletten Opimierungsdurchlauf zieht.

Im Fall des Transition-Klassifikators (9) hingegen wird, wie beim Dynamic-Klassifikator,

Abbildung 8: Verwendung des Noise-Filters bei den Boosted Decision Trees mit State-Klassifikator

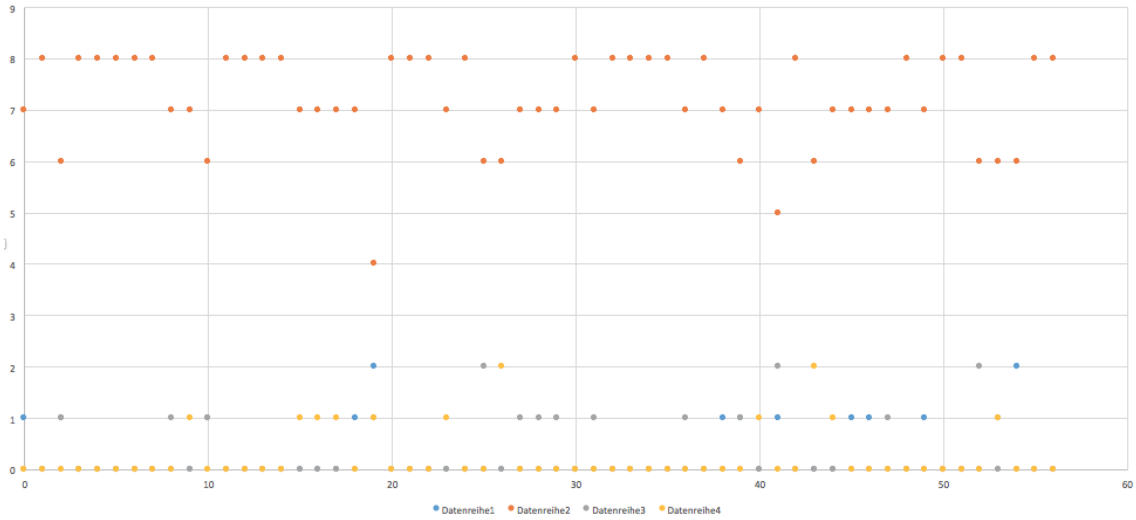


Abbildung 9: Verwendung des Noise-Filters bei den Boosted Decision Trees mit Transition-Klassifikator

in allen Fällen am meisten auf den Lowpass-Filter zurückgegriffen.

Auch im Fall der Multilayer Perceptrons beim Static Klassifikator (10) ist eine eindeutige Präferenz des Lowpass-Filters zu vermerken.

In einer weiteren Ausarbeitung im Rahmen dieser Projektgruppe wurden Feature-Sets ausgearbeitet, die als Parameter in die Optimierung eingeflossen sind. Dabei hat sich ergeben, dass bestimmte Machine-Learning Techniken bei den Klassifikatoren eindeutig vermehrt auf bestimmte Feature-Sets konvergiert sind. Diese Zuordnung werde ich hier zunächst einmal darstellen mithilfe einer Aufzählung und anschließend darauf eingehen, ob dies der vorherigen Abschätzung entsprochen hat.

1. AMM Dynamic - FeatureSet 0

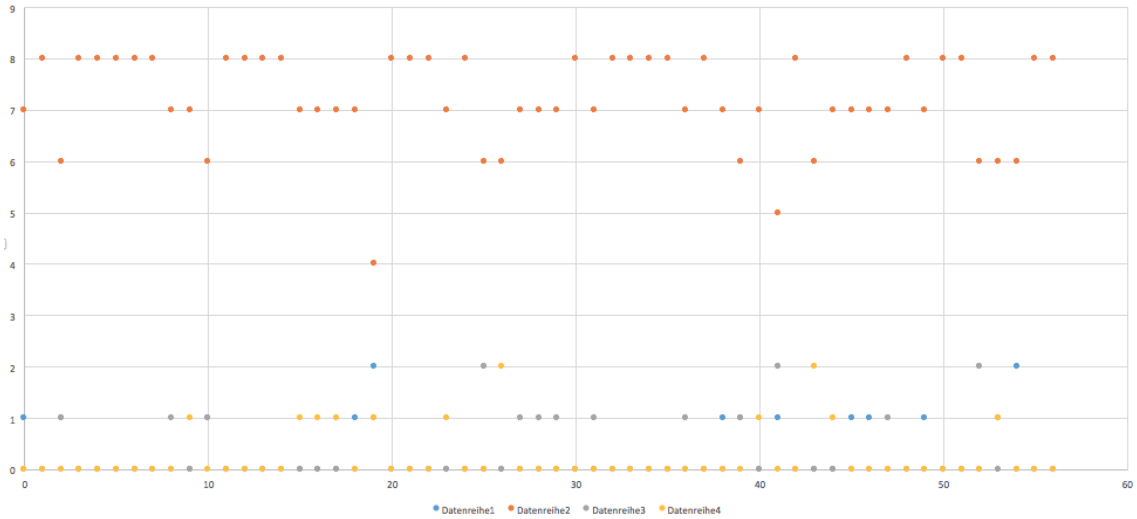


Abbildung 10: Verwendung des Noise-Filters bei den Multilayer Perceptrons mit Static-Klassifikator

2. AMM State - FeatureSet 0
3. AMM Static - FeatureSet 2
4. AMM Transition - FeatureSet 0
5. BDS Dynamic - FeatureSet 0
6. BDS State - FeatureSet 0
7. BDS Static - FeatureSet 0
8. BDS Transition - FeatureSet 0 & 5
9. BDT Dynamic - FeatureSet 0
10. BDT State - FeatureSet 0
11. BDT Static - FeatureSet 0
12. BDT Transition - FeatureSet 0 & 1
13. MLP Dynamic - FeatureSet 0
14. MLP State - FeatureSet 0 & 3
15. MLP Static - FeatureSet 0
16. MLP Transition - FeatureSet 0

Innerhalb der Ausarbeitung der Feature-Sets wurden pro Klassifikator einige verschiedene Feature-Sets ausgearbeitet, die nach einer Recherche eine bestmögliche Erkennung ermöglichen sollten. Nach der Auswertung der letztendlichen, hauptsächlichen Nutzung

der Feature-Sets stellt sich heraus, dass bei jedem Klassifikator meistens das erste implementierte Feature-Set genutzt wurde und dieses sich auch meist sehr eindeutig gegenüber den anderen Feature-Sets innerhalb der Parameter-Optimierung durch evolutionäre Algorithmen durchgesetzt hat. Die genaue Auswahl der Features im jeweiligen Feature-Set ist in der dazugehörigen Ausarbeitung von Andrea De Behr nachzulesen.

4.3 Bewertung

Im Anschluss an die Interpretation der Daten möchte ich noch einmal kurz eine Bewertung dieser Daten vornehmen.

Vorweg sollte natürlich grundsätzlich festgehalten werden, dass die Daten und Ergebnisse der Projektgruppe kritisch zu betrachten sind - aufgrund der im vorherigen Kapitel angemerkten Einflüsse auf die Daten und die Erhebung sowie Verarbeitung dieser.

Da es eine große Menge an aufgenommenen Daten gibt, welche eine große Vielfalt an Daten aufweist und diese mehrere unterschiedliche Bewegungen enthält, in der auch noch mehrere unterschiedliche Bewegungsmuster gelabelt wurden und so zu erkennen sind, ist dies ein schwierig zu lösendes Problem. Trotz der Schwierigkeiten, in einem solchen Raum an Lösungen die Richtige zu finden, gelingt dies gerade bei statischen Mustern zuverlässig. Auch Transitionen können richtig erkannt werden. Durch die Umstellung des Bewertungsverfahrens konnte auch das Problem des Overfittens verkleinert und somit die Klassifikation verbessert werden.

Somit ist auf jeden Fall dennoch festzustellen, dass an der Klassifikationsgenauigkeit der Algorithmen weiter Verbesserungen möglich sind.

5 Zusammenfassung

In dieser wissenschaftlichen Ausarbeitung wird zunächst in maschinelles Lernen eingeführt. Dabei werden die innerhalb dieser Projektgruppe verwendeten Machine-Learning Techniken Boosted Decision Trees, Multilayer Perceptrons, Boosted Decision Stumps und Adaptive Multi-Hyperplane Machines genannt und kurz grundlegend erläutert. Weiterhin wird die verwendete Technologie in Form eines Sensorgürtels kurz angesprochen und Informationen zu den erhobenen Daten und den dazugehörigen Labeln vermittelt. Dabei wird darauf eingegangen, welche Bewegungsmuster in den Datensätzen enthalten sind und welche dabei von den Algorithmen erkannt werden sollen. Die Erhebung der Daten erfolgte dabei im Rahmen der Versa-Studie. Die Label in den Datensätzen wurden manuell von Mitarbeitern oder Mitgliedern der Projektgruppe erstellt. Zum Ende des Kapitels der thematischen Grundlagen werden evolutionäre Algorithmen erklärt. Dabei handelt es sich um von Charles Darwin's Evolutionstheorie inspirierte Algorithmen [Kra16, p. 6].

Im nächsten Kapitel werden als erstes die Einflüsse auf die zu interpretierten Ergebnisse angesprochen. Dabei sind Aspekte gemeint wie die Herkunft der Datensätze, die Erstellung der Referenzlabel und die Erstellung der Zusatzaufnahmen. Weiterhin wird die Bewertung der Ergebnisse der Machine-Learning Techniken besprochen, was letzendlich über den F-Score vollzogen wird.

Die spezielle Umsetzung der evolutionären Algorithmen im Rahmen dieses Projektes ist dann Thema des nächsten Abschnitts. Dort wird der Vorgang der Rekombination, Selektion und Mutation in unserem Fall angebracht und die Bewertung der Fitness der einzelnen Individuen über den F-Score. Weiterhin wird darüber informiert, dass während der Optimierung der Parameter der Machine-Learning Techniken das Bewertungsverfahren für die Algorithmen während des Trainings geändert wurde, was dazu führt, dass nun mit realen Daten gearbeitet wird.

Dies führt zu insgesamt schlechteren Fitness-Werten der Individuen in einem neuen Durch-

lauf - dabei ist die Klassifikationsgenauigkeit eher ein kleines bisschen schlechter, woraus man entweder schließen könnte, dass die Einstellung der Parameter mithilfe des ersten Bewertungsverfahrens besser ablief, oder dass die Daten beim ersten Bewertungsverfahren overfitted sind.

Zusammenfassend lässt sich festhalten, dass die Klassifikationsergebnisse mit den neuesten Optimierungsdurchläufen für die Parameter der Machine-Learning Techniken eine bei bestimmte Klassifikatoren gute Klassifikationsgenauigkeit besitzen, jedoch auf den ersten Durchläufen leicht besser war, jedoch auch einige falsche Klassifikationen zu vermerken sind, was noch Verbesserungspotential aufzeigt.

6 Ausblick

Im Hinblick auf die Klassifikationsgenauigkeit der Algorithmen, welche noch Verbesserungspotential aufweist, wäre es durch verschiedene Maßnahmen möglich, diese eventuell zu verbessern. Einerseits wäre eine größere Anzahl an gelabelten Datensätzen zu Bewegungsmustern hilfreich, welche bisher prozentual eher wenig vertreten waren im Vergleich zu anderen Bewegungen. Als grobes Beispiel sind dabei statische Bewegungsmuster und Transitionen zu nennen. Weiterhin würden mögliche Fehler beim manuellen Erstellen von Labeln dadurch verhindert werden, wenn es die Möglichkeit für eindeutige Referenzlabel oder einen Goldstandard geben würde. Dadurch wäre eine Verbesserung des Trainings möglich. Zusätzlich könnte man abschätzen, inwieweit es möglich wäre, die Parameteroptimierung mit einem größeren Bereich an Parametern auszuprobieren, sodass die Parameter nicht von Beginn an so eingeschränkt werden müssten, wie es in unserem Fall notwendig war und dass man noch beispielsweise weitere Features und Feature-Sets in die Optimierung einbringen könnte. Dies könnte man dadurch erreichen, dass man die Optimierungsalgorithmen für einen längeren Zeitraum mit einem erweiterten Wertebereich für bestimmte Parameter auf einem Rechencluster laufen lässt.

Literatur

[GD98] M.W. Gardner and S.R. Dorling. Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences, 1998.

[Kra16] Oliver Kramer. Computational intelligence 1 genetic algorithms, 2016.

[Pow] David M W Powers. What the f-measure doesn't measure..., ?

[Pow07] David M W Powers. Evaluation: From precision, recall and f-factorto roc, informedness, markedness & correlation, 2007.

[Sch99] Robert E. Schapire. A brief introduction to boosting, 1999.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[UOL] Carl (named after carl von ossietzky). `https://www.uni-oldenburg.de/fk5/wr/hochleistungsrechnen/hpc-facilities/carl/`. Aufgerufen: 19.03.2017.

[VER] Versa-studie - selbstständigkeit wahren. `https://www.unioldenburg.de/versa-studie/`. Aufgerufen: 18.03.2017.

[WCDV11] Zhuang Wang, Koby Crammer, Nemanja Djuric, and Slobodan Vucetic. Trading representability for scalability: Adaptive multi-hyperplane machinefor nonlinear classification, 2011.

[WSX16] Yisen Wang, Chaobing Song, and Shu-Tao Xia. Unifying the split criteria of decision trees usingtsallis entropy, 2016.

D.10. HPC Integration

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Fakultät VI – Medizin und Gesundheitswissenschaften
Department für Versorgungsforschung
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: HPC-Integration

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Dmitrii Dementevskii

Oldenburg, den 7. April 2017

Inhalt

Abbildungen

1 Einleitung

Aufgrund der technologischen Entwicklung mit erhöhter Datenmenge steigt auch die Anforderung an die Leistungskapazität von Rechnern. In vielen Bereichen der Wissenschaft ist es heutzutage nahezu unmöglich die Berechnung von schweren Algorithmen am lokalen Computer durchzuführen. Deswegen wurden für diese Zwecke Hochleistungsrechner (ein Cluster von Rechnern), die eine hohe Rechenleistung und Speicherkapazität aufweisen, gebaut. Die Universität Oldenburg stellt für viele wissenschaftliche Gruppen ebenfalls ein High-Performance-Computing (HPC)-Cluster zur Verfügung.

Im Rahmen der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ (SoSe 2016 – WiSe 2017) wurde eine Software entwickelt, die mithilfe von „Maschinelles Lernen“-Algorithmen die Daten eines Sensorgürtels analysieren kann. Einige der Algorithmen benötigen zur Berechnung sehr viel Zeit bzw. hohe Leistungskapazitäten. Deshalb wurde entschieden, eine Möglichkeit zur Integration mit dem HPC-Cluster von der Universität Oldenburg zu entwickeln. Das Ziel dieser Ausarbeiting ist die Beschreibung wichtiger Merkmalen von HPC und auch die Programmierung von dem Teil der Anwendung, das für die Kommunikation mit dem Cluster verantwortlich ist.

Im ersten Teil dieser Seminararbeit werden Komponenten und Architekturen von High-Performance Computing definiert, dann wird beschrieben, welches System von Hochleistungsrechnern die Universität Oldenburg hat. Am Ende werden Informationen über die Implementierung der HPC Integration der Anwendung dargestellt.

2 High-Performance Computing

Unter High-Performance Computing (HPC) wird die Nutzung paralleler Datenverarbeitung für den effizienten, zuverlässigen und schnellen Einsatz anspruchsvoller Anwendungsprogramme verstanden. Der Ausdruck HPC wird gelegentlich als Synonym für Supercomputing verwendet, obwohl ein Supercomputer eigentlich ein System ist, das an der oder nahe der jeweils aktuellen Obergrenze der Leistungsfähigkeit von Computern operiert ([Rou07]).

2.1 Anwender

Die üblichen Anwender von HPC-Systemen finden sich unter wissenschaftlichen Forschern, Ingenieuren und akademischen Institutionen. Den jeden Tag sollen immer mehr wissenschaftliche Daten bearbeitet werden, bei denen viele Rechnerkapazitäten benötigt werden.

2.2 Entwicklung von Hochleistungs-Computer

Das 1965 von Gordon Moore formulierte empirische „Moore's Law“ besagt, dass sich die Transistordichte auf einem Computerchip etwa alle 18 Monate verdoppelt. Dieser Langzeittrend der technologischen Entwicklung führte in den letzten Jahrzehnten zu immer höher getakteten und damit schnelleren Prozessoren. Eine weitere Steigerung der Transistordichte stößt jedoch mittlerweile an physikalische Grenzen wie bspw. Wärmeeffekte auf dem Chip. Stattdessen wird nunmehr die Zahl der Prozessoren, sog. Cores, auf einem Chip ständig gesteigert, was zu zunehmend massiver Parallelität der Rechner führt. Die maximal erzielbare aggregierte Rechenleistung steigt daher auch weiterhin exponentiell an (Siehe Abbildung 1) und es ist davon auszugehen, dass sich dieses Wachstum bis mindestens

2020 fortsetzen wird ([Gru12], S. 13).

Hinweise auf die weltweit verfügbaren HPC-Kapazitäten liefert – nach einzelnen Ländern aufgeschlüsselt – die sogenannte TOP-500-Liste, die zweimal jährlich die 500 leistungsfähigsten Rechner der Welt und deren akkumulierte Gesamtleistung listet. Laut der Fassung der TOP-500-Liste vom November 2016 stehen 6,2 % der leistungsfähigsten Rechner der Welt in Deutschland.

Abbildung 1: Die Kapazitäten von den 500 leistungsfähigsten Rechner vom November 2016
(Quelle: [SDSM16])

2.3 Parallelisierung

Für viele Probleme in der Wissenschaft wird Parallelisierung angewendet, wenn verschiedene Teilprozesse gleichzeitig durchgeführt werden müssen. Es gibt dabei drei Möglichkeiten zur Parallelisierung:

- die Verteilung von der Arbeit;

- die Verteilung vom Data;
- die Verteilung von Domain.

Es ist zu erwähnen, dass massive Parallelität prinzipielle Beschränkungen aufweist. Die erste Beschränkung ist, dass nicht alle Teile eines Programms sich parallelisieren lassen können. Durch massive Parallelität ist es möglich die Ausführungszeit des parallelisierbaren Anteils beliebig klein zu machen, der sequentielle Anteil bleibt davon jedoch unberührt. Bei einem Anteil von nur 1% nicht-parallelisierbare Anweisungen ist der maximale Speedup gleich 100, unabhängig davon wie viele Prozessoren eingesetzt werden. Dieses Argument wurde bereits 1967 von Gehe Amdahl publiziert und ist unter Amdahls Gesetz bekannt (Siehe Abbildung 2). Aber der sequentielle Anteil ist nicht die einzige Beschränkung paralleler Effizienz, da die Prozessoren in Regel noch ihre Arbeit koordinieren müssen ([HS06], S. 11).

Abbildung 2: Amdahls Gesetz
(Quelle: [HS06])

2.4 Arten der Nutzung von HPC-Ressourcen

Grundsätzlich sind im Bereich der Anwendungen drei Arten der Nutzung von HPC-Ressourcen zu unterscheiden: Capability Computing, Capacity Computing und Real-Time-Computing. Als Capability Computing wird das Rechnen einzelner komplexer Probleme oder Modelle bezeichnet, die sich idealerweise durch gute Parallelisierbarkeit auszeichnen. Anwendungen des Capability Computing sind häufig zeitkritisch im Sinne eines Wettbewerbs um Durchbrüche in der Grundlagenforschung. Aufgrund der Anforderung eines schnellen Outputs sowie der Komplexität der Fragestellungen werden für Anwendungen des Capability Computing Rechensysteme der obersten Leistungsklasse benötigt. Unter Capacity Computing wird hingegen das gleichzeitige Rechnen vieler Probleme, Modelle oder Modellparameter verstanden, die weniger als einzelne Anforderung, sondern vielmehr in der Summe Anforderungen an die Leistungsfähigkeit der Rechensysteme stellen. Zunehmende Bedeutung erlangt das Real-Time-Computing (z. B. als Einsatz in die Visualisierungen), das sich von den beiden genannten Anwendungsformen insbesondere durch die Interaktivität abhebt ([Gru12], S. 12).

2.5 Arten von Architekturen

Für die Effektivität müssen die Code-Architektur und Hardwarearchitektur zueinander passen. Dafür gibt es verschiedene Parallel-Hardware Topologien, die im Folgenden genannt werden.

2.5.1 Multiprozessor – Architektur

Die meisten Mehrprozessorsysteme weisen heute die Multiprozessor-Architektur (Siehe Abbildung 3) auf, welche folgende Eigenschaften aufweist:

- ein globaler Speicher;
- alle Cores sind verbunden mit der gleichen Geschwindigkeit;
- Konzept von Uniform Memory Access (UMA) und
- Symmetrisches Multiprozessorsystem (alle Prozessoren sind gleich).

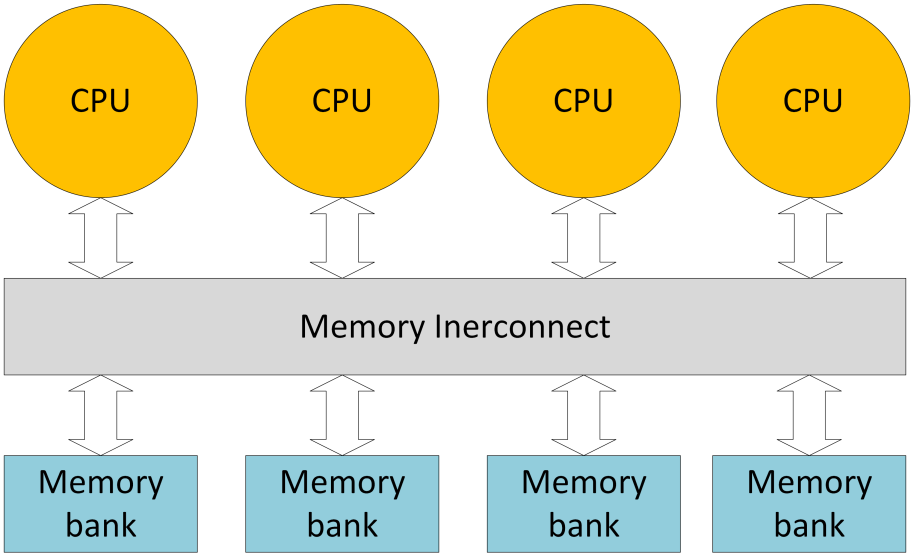


Abbildung 3: Die Darstellung von einem Multiprozessorsystem (eigene Darstellung)

2.5.2 Multicomputer-Architektur

Eine andere Systemarchitektur bildet die Multicomputer-Architektur (Siehe Abbildung 4). Diese werden in Fällen eingesetzt, wenn unterschiedliche Prozessorenoder Rechner verbunden werden sollen. Zu den Eigenschaften eines Multicomputers gehören:

- ein verteilter Speicher;
- Nodes sind verbunden mithilfe von Node-Interconnect;
- Konzept von Cache-only-Memory-Access (CoMA);
- verschiedene Prozess- und Netzwerktopologie.

2.5.3 SMP/ccNuma Nodes

Die meisten heutigen HPC-Systeme (z.B. Flow und Hero von der Universität Oldenburg) wurden als ein Cluster von SMP/ccNuma Nodes gebaut (Siehe Abbildung 5). Dabei sind gleiche Nodes, in den es jeweils viele Prozessoren gibt, miteinander mit durch den Node Interconnect verbunden ([AH15], S. 7f.).

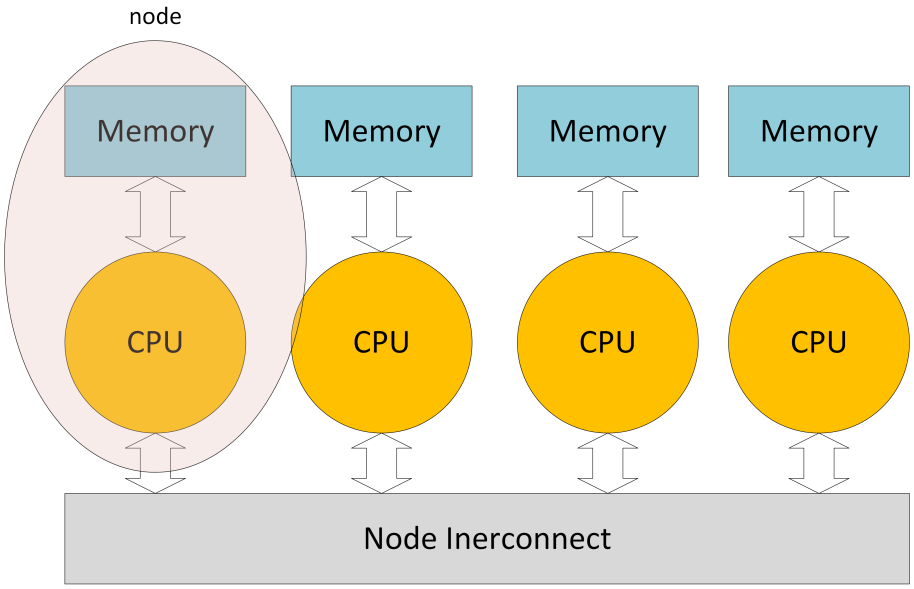


Abbildung 4: Die Darstellung von einem Multicomputersystem (eigene Darstellung)

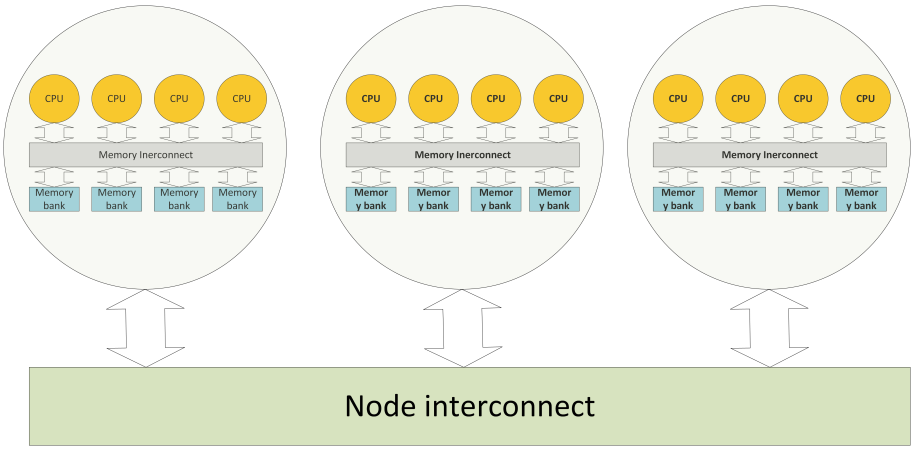


Abbildung 5: Die Darstellung von einem Cluster mit SMP-Nodes (eigene Darstellung)

3 Hochleistungsrechner an der Universität Oldenburg

Das wissenschaftliche Rechnen (Scientific Computing) bildet eine interdisziplinäre Querschnittsaktivität der Fakultät V für Mathematik und Naturwissenschaften an der Universität Oldenburg. Da viele rechenintensive Forschungsprojekte nicht auf einzelnen Workstations oder Computer Servern durchführbar sind, ist eine adäquate Versorgung der Arbeitsgruppen mit hochwertiger Parallel-Rechenleistung unverzichtbar. Der 2004 bis 2006 beschaffte und überwiegend von Mitarbeitern in den Studiengängen Chemie und Physik genutzte Parallelrechencluster GOLEM mit 400 Prozessorkernen wird dem stark gestiegenen Rechenbedarf nicht mehr gerecht. Einige Projekte zeichnen sich durch sehr hohe Kommunikationsanforderungen und/oder sehr großen Hauptspeicherbedarf (mind. 256 GB) aus, wofür ein Multiprozessor (Shared-Memory) Subsystem mit 64-128 Cores benötigt wurde ([Klu10]).

Der an der Carl von Ossietzky Universität Oldenburg in 2011 in Betrieb genommene Rechencluster HERO (Siehe Abbildung 6) wird aktuell von etwa 40 verschiedenen Arbeitsgruppen aus den Fakultäten Mathematik und Naturwissenschaften, Medizin und Gesundheitswissenschaften sowie Informatik, Wirtschafts- und Rechtswissenschaften für die Durchführung von rechenintensiven Forschungsprojekten genutzt. Zu den wissenschaftlichen Disziplinen an der Universität, die Arbeiten an den Rechnercluster HERO durchführen, gehören unteranderem ([Klu10]):

- Theoretischen Chemie/Computerchemie: mit Schwerpunkt für das theoretische Verständnis von molekularen Prozessen an Oberflächen.
- Theoretischen Physik: :Forschungsthema Simulation ungeordneter Systeme, z. B. zur Bestimmung von exakten Grundzuständen von sehr großen Zufallsfeldsystemen, einem der wichtigsten Modelle für ungeordnete Magnete.

- Hörforschung und Akustik: Rechnungen in verschiedenen Projekten der Grundlagenforschung, insbesondere Analyse und Modellierung neuronaler Daten sowie im Rahmen von angewandten Projekten der automatischen maschinellen Spracherkennung, Ereigniserkennung und der akustischen Signalverbesserung.
- Informatik: im interdisziplinären Forschungsverbund Smart Nord verteilte Steuerungskonzepte zur koordinierten, dezentralen Bereitstellung von Wirkleistung, Regelleistung und Blindleistung in verteilten Netzen

Abbildung 6: Foto von dem HERO-Cluster
(Quelle: [Alb15])

3.1 Technischer Überblick über die eingesetzten Systeme

Der zentrale Hochleistungsrechner der Universität Oldenburg hat drei Systeme ([Alb15],s. 15-16):

1. FLOW (Facility for Large-Scale Computations in Wind Energy Research): IBM iDataPlex cluster solution, 2232 CPU cores, 6 TB of (distributed) main memory, QDR InfiniBand interconnect. Theoretical peak performance: 24 TFlop/s.
2. HERO (High-End Computing Resource Oldenburg) hat zwei Komponenten:
   a) IBM iDataPlex cluster solution, 1800 CPU cores, 4 TB of (distributed) main memory, Gigabit Ethernet interconnect. Theoretical peak performance: 19.2 TFlop/s.

   b) SGI Altix UltraViolet shared-memory system (SSMP component"), 120 CPU cores, 640 GB of globally addressable memory, NumaLink5 interconnect. Theoretical peak performance: 1.3 TFlop/s.

3. GOLEM: older, AMD Opteron-based cluster with 390 cores and 800 GB of (distributed) main memory. Theoretical peak performance: 1.6 TFlop/s.

3.2 Bright Cluster Manager

Für die Kontrolle von dem Clusterzustand wurde das Cluster Manager(eine Software für das Management eines Computerclusters), Bright Cluster Manager gewählt. Er steht für Verwaltungsvorgänge im Cluster zur Verfügung und automatisiert meist Vorgänge, wie den Failover vom Primärsystem zum Standby-System im Fehlerfall sowie Switchover für Wartungszwecke. Der Bright Cluster Manager vereinfacht dabei folgende Tätigkeiten ([BC10]):

- Monitoring: mithilfe des Bright Cluster Managers können umfangreiche Hardware- und Software-Metriken auf unterschiedliche Weise überwacht, visualisiert und analysiert werden.
- Installierung: das Installieren mithilfe Bright Cluster Manager ist einfach und erfordert nur ein wenig Linux, HPC sowie Hadoop oder OpenStack Know-how. Er ist auch sehr flexibel: viele Optionen werden angeboten und fast jede Einstellung, die während der Installation konfiguriert wird, kann danach geändert werden.
- Node Versorgungsprozess: die Bereitstellung von Software-Images auf Knoten ist eine zentrale Aufgabe jeder Cluster-Management-Plattform. Leistungsfähige und flexible Knotenbereitstellung und Software-Image-Management sind für die Clusterinstallation und der Verwaltung, insbesondere für größere und komplexere Cluster, von großer Bedeutung.
- Cluster Management GUI (Siehe Abbildung 7): der Bright Cluster Manager enthält eine grafische Benutzeroberfläche (GUI), die das Cluster-Management vereinfacht. Die intuitive GUI von Bright bietet eine einheitliche Sichtbarkeit und Kontrolle von Objekten, ohne die Notwendigkeit mehrere GUIs in Ihrer Infrastruktur anzuzeigen und zu betreiben.

Abbildung 7: GUI des Bright Cluster Managers
(Quelle: [BC10])

3.3 Oracle Grid Engine

Für die Prozessaufteilung ist Oracle Grid Engine (oder CODINE - Computing in Distributed Networked Environments) verantwortlich. Grid Engine wird typischerweise auf einem Hochleistungscomputer (HPC)-Cluster verwendet und ist für das Annehmen, Planen, Verteilen und Verwalten von großen Mengen paralleler Jobs zuständig ([Cor10]).

3.4 Betriebssystem

Als ein Betriebssystem in jedem Node steht Scientific Linux zur Verfügung. Scientific Linux ist eine Linux-Distribution, die auf Distribution Red Hat Enterprise Linux basiert und ihren Schwerpunkt auf die verschiedene Ansprüche von wissenschaftliche Institutionen legt.

4 HPC Integration für die MAmKS Anwendung

Im Rahmen der Projektgruppe MAmKS wurde eine Anwendung für die Analyse von Daten in Versa-Studie genutzten Sensorgürtel entwickelt. Die Software besteht aus zwei Teilen: eine Benutzeroberfläche und eine Ausführungsumgebung für die Ausführung von Algorithmen. Einige von diesen Algorithmen weisen hohe Bedürfnisse der Rechnerkapazität auf(z. B. Maschinelles Lernen). Deswegen wurde vereinbart, die Integration mit dem HPC Cluster von der Universität Oldenburg zu entwickeln.

Als Programmiersprache für die Anwendung wurde Java gewählt, da Java u. a. auch Möglichkeiten zur Parallelisierung anbietet. Bei modernen Betriebssystemen gehört zu jedem Prozess mindestens ein Thread, der den Java Programmcode ausführt. Damit werden also genau genommen die Prozesse nicht mehr parallel ausgeführt, sondern nur die Threads. Innerhalb eines Prozesses kann es mehrere Threads geben, die alle zusammen in demselben Adressraum ablaufen. Die einzelnen Threads eines Prozesses können untereinander auf ihre öffentlichen Daten zugreifen. Die Programmierung von Threads ist in Java dabei einfach möglich und die quasi parallel ablaufenden Aktivitäten ergeben für den Benutzer den Eindruck von Gleichzeitigkeit. Zudem ist es in Java das Ausführen von multithreaded Software auch möglich, wenn das Betriebssystem des Rechners keine Threads direkt verwendet. In diesem Fall simuliert die virtuelle Maschine die Parallelität, indem sie die Synchronisation und die verzahnte Ausführung regelt. Unterstützt das Betriebssystem Threads direkt, bildet die JVM die Thread-Verwaltung in der Regel auf das Betriebssystem ab ([Ull11]).

Die Software der Projektgruppe bietet zwei Möglichkeiten: die Arbeit mit Threads und mit Prozessen (Siehe Abbildung 8). Eigentlich wird die Ausführung von Algorithmen im lokalen Rechner mithilfe von Threads durchgeführt (ein Thread für ein Algorithmus, aber

die Implementierung von dem Algorithmus kann auch mehrere Threads schaffen). Ein Algorithmus wird als ein Prozess im Cluster (ein Prozess kann auch viele Threads haben) während der Arbeit ausgeführt. Dafür wird ein SGE -Skript mit Job-Parameters ausgeführt. In dem SGE-Skript wurden alle Einstellungen für die Ausführung von der Arbeit von Grid Engine rein gestellt.

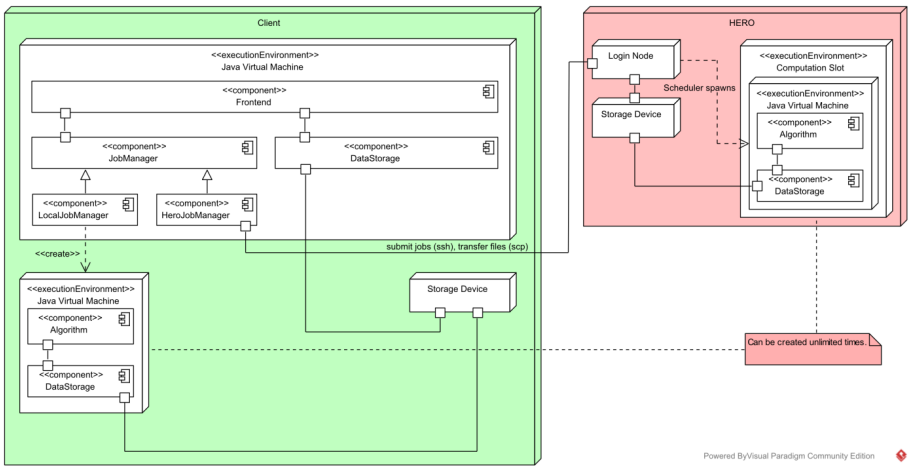


Abbildung 8: Architektur der MAmKS Anwendung
(eigene Darstellung)

Als ein Verbindungsprotokoll wird SSH (Secure Shell) verwendet. JSch von JCraft wurde als JAVA-Bibliothek genommen, welche verschiedene Mittel zur Arbeit mit SSH zur Verfügung stellt (Befehle ausrufen, Files kopieren usw.).

Im Benutzerverzeichnis des Cluster soll eine Struktur von Ordner geschafft werden:

- /MAMKS – der Hauptordner der Anwendung
- /MAMKS/jobs –für benötige Daten von Algorithmen
- /MAMKS/jobs_out – für Ausgabedateien von Prozessen des Clusters
- /MAMKS/motiondata – für alle Motiondata – Dateien

Auch sollen sich im Hauptordner die letzte Version von Job-Executor (mamks-job-executor.jar) und das SGE-Skript (run_job.sge) befinden. Für jeder Algorithmus soll ein Parameter „Execute on cluster“ („true“ oder „false“, standardmäßig „false“) ausgefüllt werden. Wenn der Parameter „true“ ist, dann wird der Algorithmus auf dem Cluster ausgeführt, ansonsten lokal auf dem Rechner.

Für die Arbeit mit dem Cluster soll der Benutzer ein Login/Password-Kombination haben (die vorher im System registriert werden sollte) (Siehe Abbildung 9).

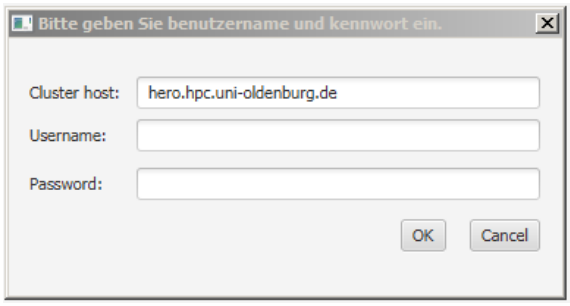


Abbildung 9: Das Fenster für die Eingabe von Login/Password
(eigene Darstellung)

4.1 Möglichkeiten durch Integration mit HPC-Cluster

Die Integration mit dem HPC-Cluster bringt folgende Möglichkeiten:

1. die Ausführung von Jobs
    a) die Vorbereitung von nötigen Dateien im lokalen Rechner (Job-Datei, Motion-data Dateien, die bei dem Algorithmus benutzen werden, Dateien für Machine-Learning-Algorithmen),
    b) die Archivierung von den lokalen Dateien
    c) die Sendung von dem Archiv,
    d) die Extrahierung des Archives auf dem Cluster,
    e) die Löschung des Archives und
    f) die Ausführung von SGE-Skript
2. die Kontrollierung des Status von Jobs
3. die Übertragung von Ergebnisse
    a) die Übertragung von dem Archiv und
    b) die Extrahierung des Archives auf dem lokalen Rechner
4. sowie die Entfernung von dem Job im Cluster

Die Verwendung von dem Cluster führt meistens zur Leistungsverbesserung für Algorithmen, die viele Rechenkapazitäten und Zeit brauchen. Die Benutzer müssen allerdings berücksichtigen, dass alle Prozessen auf dem Cluster vor die Aufführung noch in der Warteschlange stehen müssen und die Übertragung von den großen Datenmengen zeitaufwändig ist.

5 Fazit

Heutzutage werden die Leistungskapazitäten der HPC-Systeme in vielen Bereichen der Wissenschaft verwenden. Die Universität Oldenburg bietet für viele Studierende und verschiedene Gruppen von Wissenschaftler die Möglichkeit mithilfe von einem Computercluster die Forschungen durchzuführen. Für die Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren" kann die Integration mit dem HPC-Cluster die Überarbeitung von großen Datenmengen vereinfachen und gibt eine positiven Ausblick für die weitere Entwicklung des Projekts.

Literatur

[AH15] Stefan Albensoeder and Stefan Harfst. Introductionto MPI. Website, 2015. Online erhältlich unter https://wiki.hpcuser.uni-oldenburg.de/images/4/4c/Introduction_MPI_Course1015.pdf; abgerufen am 14.01.2017.

[Alb15] Stefan Albensoeder. Introduction to HPC. Website, 2015. Online erhältlich unter https://wiki.hpcuser.uni-oldenburg.de/images/8/8a/Introduction_HPC_Course2015.pdf; abgerufen am 14.01.2017.

[BC10] Inc. Bright Computing. Bright Cluster Manager. Website, 2010. Online erhältlich unter http://www.padovatech.com/wp-content/uploads/2010/09/Bright-Cluster-Manager-Brochure.pdf; abgerufen am 14.01.2017.

[Cor10] Oracle Corporation. Oracle Grid Engine: An Overview. Website, 2010. Online erhältlich unter http://www.oracle.com/technetwork/oem/host-server-mgmt/twp-gridengine-overview-167117.pdf; abgerufen am 14.01.2017.

[Gru12] Thomas Gruenewald. Strategische Weiterentwicklung des Hoch- und Höchstleistungsrechnens in Deutschland. Website, 2012. Online erhältlich unter www.wissenschaftsrat.de/download/archiv/1838-12.pdf; abgerufen am 14.01.2017.

[HS06] Bauke Heiko and Mertens Stephan. *Cluster Computing*, chapter Von Megahertz zu Gigaflops. Springer-Verlag Berlin Heidelberg, 2006.

[Klu10] Thorsten Kluener. High-Performance Computing Cluster. Website, 2010. Online erhältlich unter http://gepris.dfg.de/gepris/projekt/177092163; abgerufen am 14.01.2017.

[Rou07] Margaret Rouse. High-Performance Computing (HPC). Website, 2007. Online erhältlich unter http://www.searchdatacenter.de/definition/High-Performance-Computing-HPC; abgerufen am 14.01.2017.

[SDSM16] Erich Strohmaier, Jack Dongarra, Horst Simon, and Martin Meuer. PERFORMANCE DEVELOPMENT. Website, 2016. Online erhältlich unter `https://www.top500.org/statistics/perfdevel/`; abgerufen am 14.01.2017.

[Ull11] Christian Ullenboom. *Java ist auch eine Insel*, chapter 14 Threads und nebenläufige Programmierung. Rheinwerk Verlag, 2011.

D.11. Entwicklung einer Sensorplattform zur Erfassung von Bewegungsparametern

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Masterstudiengänge ESMR, Informatik und Wirtschaftsinformatik

Seminararbeit: Entwicklung einer Sensorplattform zur Erfassung von Bewegungsparametern

Projektgruppe:
Mobilitäts-Assessments mit körpernahen Sensoren
SoSe 2016 - WiSe 2017

Themensteller: Prof. Dr.-Ing. Andreas Hein
Betreuer: Dr. rer. nat. Sebastian Fudickar, Sandra Hellmers

Autor: Nils Hartmann

Oldenburg, den 18. April 2017

Abstract

Im Rahmen der Projektgruppe „Mobilitäts-Assessments mit körpernahen Sensoren“ wurden mit Hilfe einer Sensorplattform Bewegungsparameter aus geriatrischen Assassments gesammelt. Die dabei verwendete Hardware wurde durch die Firma Humotion aus Münster für diesen Zweck entwickelt und der Abteilung Medizintechnik zur Verfügung gestellt. Die Plattform umfasst mehrere Sensoren darunter ein drei-Achs Beschleunigungssensor, ein drei-Achs Gyroskop, ein drei-Achs Magnetometer, einen Luftdrucksensor sowie zwei Thermometer. Die Sensorik ist zusammen mit einem Akku innerhalb eines Gürtels verbaut und wird während der Assassments wie ein konventioneller Gürtel um die Hüfte getragen. Den Sensorgürtel gibt es in verschiedenen Ausführungen, so wurde er überwiegend mit einer Abtastrate von 100Hz verwendet, jedoch liegen zum Teil auch bereits 400Hz Messungen vor. Grundsätzlich bietet die Plattform darüber hinaus vielerlei Einstellmöglichkeiten welche der Projektgruppe jedoch aufgrund der nicht quelloffenen Hardware verborgen geblieben sind. Durch die Projektinitiatoren kam der Wunsch auf, das Sensorsystem um eine Liveschnittstelle in der Projektgruppenapplikation zu erweitern. Dies war der Projektgruppe jedoch ebenfalls mit der geschlossenen Humotionhardware nicht möglich. Diese Gründe nahmen die Projektsteller sowie der Autor zum Anlass eine eigene Hardware auszuarbeiten. Diese soll vollständig quelloffen, weitgehend variabel in den Möglichkeiten der Weiterentwicklung gehalten und mit einer Liveschnittstelle versehen sein. Im Folgenden wird die grundsätzliche Konzeption, die Umsetzung in Form eines Schaltplans sowie der Aufbau eines Layouts erläutert.

Diese Ausarbeitung wird in Absprache mit Dr. Sebastian Fudickar nachgereicht.