# Combining approximate inference methods for efficient learning on large computer clusters

Zhenwen Dai[1,2], Jacquelyn A. Shelton[1], Jörg Bornschein[1], Abdul Saboor Sheikh[1], Jörg Lücke[1,2]

[1] Frankfurt Institute for Advanced Studies, Germany; [2] Physics Dept., Goethe-University Frankfurt, Germany

## Summary


- A framework of parallel Expectation Maximization (EM) learning
- Parallelization based on MPI
- Approximate inference with Expectation Truncation (ET)
- Dynamic data repartitioning
- Hybrid parallelization with GPUs
- Efficient inference in high dimensions with sampling


## Parallel EM Learning Framework

### A Typical Sparse Coding Generative Model:

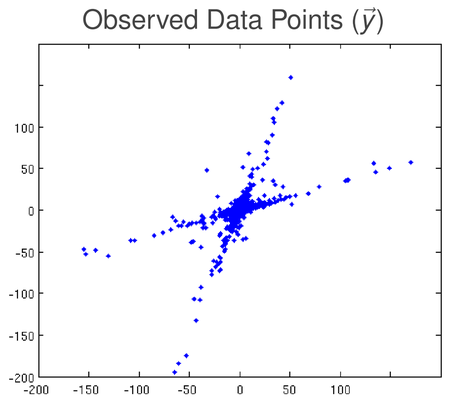
$$p(\vec{s}\,|\,\Theta) = \prod_{i=1}^{m} \mathcal{C}(s_i)\,, \text{ where } \mathcal{C}(s_i) = \frac{1}{\pi\,(1+s_i^2)}$$

$$p(\vec{y}\,|\,\vec{s},\Theta) = \mathcal{N}(\vec{y};\, W\vec{s},\, \sigma^2\,1|)$$

where

| | |
|---|---|
| $\vec{y} \in \mathbb{R}^D$ | observed variables |
| $\vec{s} \in \mathbb{R}^H$ | hidden variables |
| $W \in \mathbb{R}^{D\times H}$ | basis functions |
| $\sigma$ | noise level |
| $\pi$ | prior parameter |

Observed Data Points ($\vec{y}$)




Note:

- $p(\vec{s}\,|\,\Theta)$ can be replaced by other distributions, e.g. Bernoulli, Laplace, or spike-and-slab distributions.
- $W\vec{s}$ can be replaced by other superposition rules, e.g. maximum or occlusion.

### Maximum Likelihood Learning via EM:

$$\Theta^* = \arg\max_{\Theta}\{\mathcal{L}(\Theta)\} \text{ with } \mathcal{L}(\Theta) = \log\left(p(y^{(1)},\ldots,y^{(N)}|\Theta)\right) = \sum_{n=1}^{N} \log p(y^{(n)}\,|\,\Theta),$$

$$\text{where } p(y\,|\,\Theta) = \int_{\vec{s}} p(y\,|\,\vec{s},\Theta)\,p(\vec{s}\,|\,\Theta)d\vec{s}.$$

In general, the derived update equations in the M-step take the form:

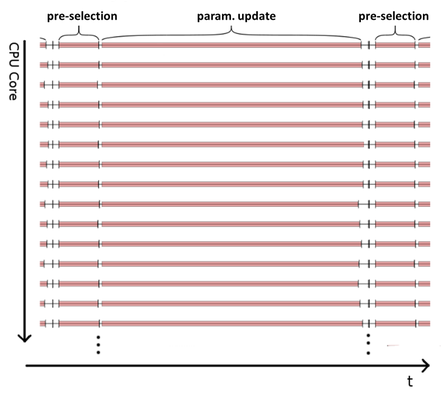
$$\theta^{\text{new}} = \left(\sum_{n=1}^{N}\langle f(y^{(n)},\vec{s})\rangle_{q_n(\vec{s})}\right)\left(\sum_{n=1}^{N}\langle g(\vec{s})\rangle_{q_n(\vec{s})}\right)^{-1},$$

where $\theta$ is some parameter to update, $f$ and $g$ are model dependent update functions, and $\langle\cdot\rangle_{q_n}$ are their expectation values w.r.t. the distribution $q_n$.

### Parallelization Framework:

- partition according to data points
- compute sufficient statistics on local sets of data points
- use (sum-)reductions to aggregate statistics in M-step

Typical Runtime Trace




## Expectation Truncation

The posterior distribution is approximated by truncating the true posterior distribution on a subset $\mathcal{K}_n$ of the state space:

$$p(\vec{s}\,|\,\vec{y}^{(n)},\Theta) \approx \frac{p(\vec{s},\vec{y}^{(n)}\,|\,\Theta)}{\sum_{\vec{s}'\in\mathcal{K}_n} p(\vec{s}',\vec{y}^{(n)}\,|\,\Theta)}\,\delta(\vec{s}\in\mathcal{K}_n)$$

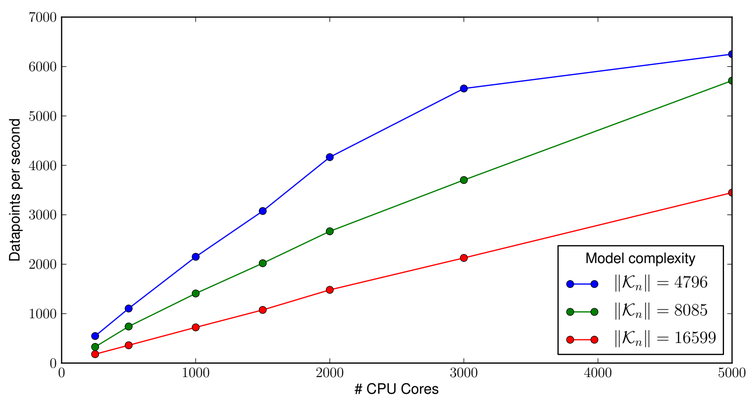
The subsets $\mathcal{K}_n$ are chosen in a data-driven way using a deterministic selection function $\mathcal{S}_h$. Appropriate selection functions $\mathcal{S}_h(\vec{y},\Theta)$, e.g. for sparse coding models, can be realized as any efficiently computable function $f(\vec{y},\Theta)$ with a norm that correlates with the probabilities $p(s_h = 1\,|\,\vec{y}^{(n)},\Theta)$; here a $\mathcal{S}_h(\vec{y}^{(n)})$ yielding a reasonable definition of $\mathcal{K}_n$ is:

$$\mathcal{S}_h(\vec{y}^{(n)}) = (\vec{W}_h^{\mathrm{T}}\,/\,||\vec{W}_h||)\,\vec{y}^{(n)}, \quad \text{with } ||\vec{W}_h|| = \sqrt{\textstyle\sum_{d=1}^{D}(W_{dh})^2}\,.$$

Computer Clusters

| Name | # CPU cores | # GPUs |
|---|---|---|
| GPU-Scout | 144 | 108 |
| FIAS | 500 | 12 |
| Fuchs CSC | ∼4500 | 0 |
| Loewe CSC | ∼19000 | 786 |

Performance Evaluation for a Sparse Coding Model




## ET Based Dynamic Data Repartitioning for Parallel EM Learning

EM based optimization of latent causes models can also involve state ($\vec{s}$) dependent computationally expensive operations. e.g. consider a SC model with a spike-and-slab prior (combining continuous $\vec{z}$ and discrete $\vec{s}$ hidden variables):

$$p(\vec{s}\,|\Theta) = \prod_{h=1}^{H}\pi_h^{s_h}(1-\pi_h)^{1-s_h} = \text{Bernoulli}(\vec{s};\vec{\pi}) \text{ and } p(\vec{z}\,|\Theta) = \mathcal{N}(\vec{z};\,\vec{0},\,1|_H) \quad \text{(Gaussian)},$$

$$\text{with} \quad p(\vec{y}\,|\,\vec{s},\vec{z},\Theta) = \mathcal{N}(\vec{y};\; W(\vec{s}\odot\vec{z}),\sigma^2 1|_D) \text{ where } (\vec{s}\odot\vec{z})_h = s_h z_h \quad \text{for all } h,$$

The (ET) truncated posterior of the model takes the following form:

$$p(\vec{s},\vec{z}\,|\,\vec{y}^{(n)},\Theta) \approx \frac{\mathcal{N}(\vec{y}^{(n)};\vec{\mu}_{\vec{s}},C_{\vec{s}})\,\text{Bernoulli}(\vec{s};\vec{\pi})\,\mathcal{N}(\vec{z};\,\vec{\kappa}_{\vec{s}}^{(n)},\Lambda_{\vec{s}})}{\sum_{\vec{s}'\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{(n)};\vec{\mu}_{\vec{s}'},C_{\vec{s}'})\,\text{Bernoulli}(\vec{s}';\vec{\pi})}\,\delta(\vec{s}\in\mathcal{K}_n). \tag{1}$$

$$\text{where } C_{\vec{s}} = \tilde{W}_{\vec{s}}\tilde{W}_{\vec{s}}^{\mathrm{T}} + \sigma^2 1|_D, \qquad (\tilde{W}_{\vec{s}})_{dh} = W_{dh}s_h, \qquad M_{\vec{s}} = \tilde{W}_{\vec{s}}^{\mathrm{T}}\tilde{W}_{\vec{s}} + \sigma^2\,1|_H, \tag{2}$$

$$\Lambda_{\vec{s}} = \sigma^2\,(M_{\vec{s}})^{-1} \quad \text{and} \quad \vec{\kappa}_{\vec{s}}^{(n)} = (M_{\vec{s}})^{-1}\,\tilde{W}_{\vec{s}}^{\mathrm{T}}\,\vec{y}^{(n)}. \tag{3}$$

Computation of the posterior is expensive, It requires parameters (2) to (3), and it also involves inverting and taking determinant of $C_{\vec{s}}$.

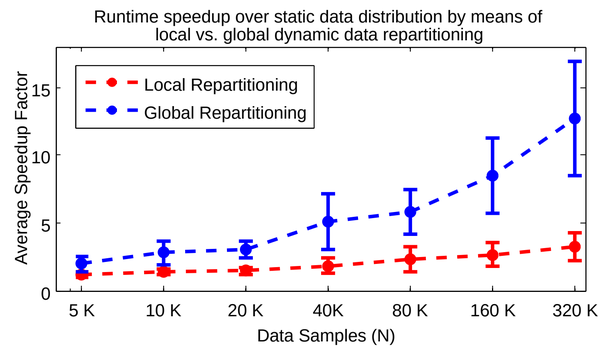
Note:

- The parameters $\vec{\mu}_{\vec{s}}$, $C_{\vec{s}}$ and $\Lambda_{\vec{s}}$ entirely depend on a state $\vec{s}$ of causes and $\vec{\kappa}_{\vec{s}}^{(n)}$ also takes prefactors that can be precomputed given $\vec{s}$
- ET preselection of the most probable hidden causes defines a sub-state-space $\mathcal{K}_n$ for each $\vec{y}^{(n)} \in \mathcal{Y} = \{\vec{y}^{(1)},\ldots,\vec{y}^{(N)}\}$
- Data points associated with the same subspaces can share the computations involved in (1) - (3)
- In a parallel setting, maximizing the similarity among data points assigned to individual processing units can minimize redundant computations overall

### Dynamic Data Repartitioning Parallelization Framework:

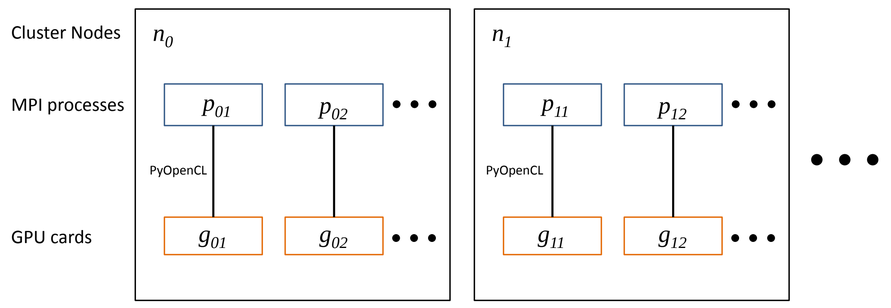
- Prior to each E-step, cluster data based on ET subspace preselection
- To avoid unfair workload distribution, split large clusters
- Distribute clusters evenly among computing nodes.
- Use (sum-)reductions (as before) to aggregate statistics in M-step

E-step runtime speedup over the static data distribution strategy taken as a baseline. The red plot shows the speedup when initially uniformly distributed data samples were only clustered locally by each processing unit, while the blue plot shows the speedup as a result of globally clustering and redistributing the data. The runtimes include the time taken by data clustering and repartitioning modules.



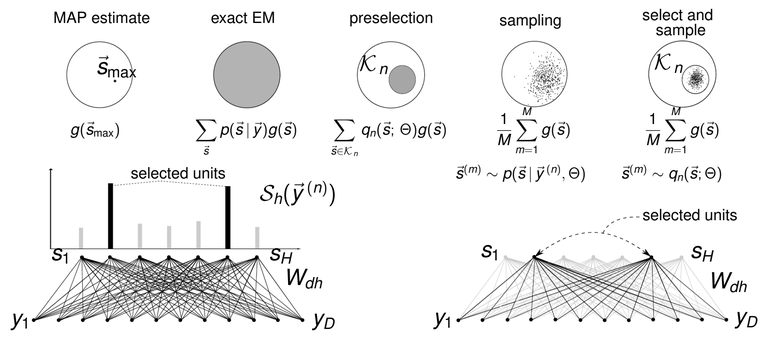

## Hybrid Parallelization with GPUs

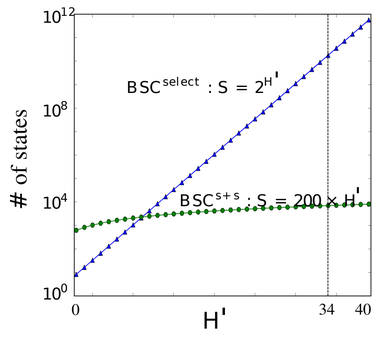
- Divide data points according to the number of GPUs.
- Assign every GPU a dedicated CPU (MPI) process.
- Replace the sufficient statistic computation by specialized GPU kernels.
- Control CPU-GPU synchronization via PyOpenCL.
- Observed about 10-20 times speed up than only using CPUs.




## Sampling

- Straight forward to integrate with sampling.
- Gibbs-, MCMC or more advanced sampling methods.
- E.g. Select and Sample Sparse Coding with $H = 1600$ latent variables on $40 \times 40$ image patches.




**Acknowledgement** This project was supported by the German Federal Ministry of Education and Research (BMBF) within the "Bernstein Focus: Neurotechnology Frankfurt" through research grant 01GQ0840 and by the German Research Foundation (DFG) in the project LU 1196/4-1.