A NEW FORMULATION FOR DEEP NEURAL NET LEARNING

Miguel Á. Carreira-Perpiñán and Weiran Wang.
EECS, University of California, Merced.

BayLearn
Bay Area Machine Learning Symposium 2012

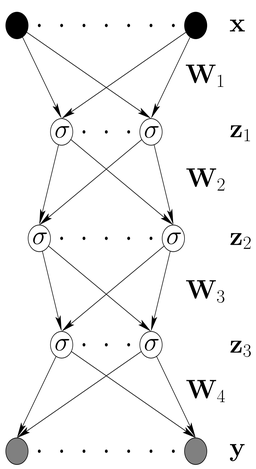
1 The difficulty of optimizing deep nets

- Training mappings with **many hidden layers** (deep nets) is a long standing problem and remains an art. Slowness of the optimization is caused by ill-conditioning of the objective function, which is deeply nested:

$$E_1(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n; \mathbf{W})\|^2,$$
$$\mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathbf{f}_{K+1}(\dots \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}; \mathbf{W}_1); \mathbf{W}_2) \dots; \mathbf{W}_{K+1}),$$

  where each layer function has the form $\mathbf{f}_k(\mathbf{x}; \mathbf{W}_k) = \sigma(\mathbf{W}_k \mathbf{x})$, i.e., a linear transformation followed by a squashing nonlinearity.
- Most widespread methods are stochastic gradient descent (SGD) and off-the-shelf optimizers (CG and L-BFGS), taking tiny steps towards a minimum. SGD is hard to parallelize and requires carefully tuned learning rates. Second order methods have limited application due to large size of the Hessian.
- Speeding up the optimization will free up computer time that can be spent on testing different architectures, cross-validating hyperparameters and trying different initializations.

2 The method of auxiliary coordinates (MAC)



Net with $K = 3$ hidden layers ($\mathbf{W}_k$: weights, $\mathbf{z}_k$: auxiliary coordinates).

- We introduce one **auxiliary variable** per data point and per hidden unit and define the following **equality-constrained** optimization problem:

$$E(\mathbf{W}, \mathbf{Z}) = \sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1})\|^2$$
$$\text{s.t.} \left\{ \begin{array}{l} \mathbf{z}_{K,n} = \mathbf{f}_K(\mathbf{z}_{K-1,n}; \mathbf{W}_K) \\ \cdots \\ \mathbf{z}_{1,n} = \mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1) \end{array} \right\} n = 1, \dots, N.$$

- One way to solve this problem is **quadratic-penalty method**. We optimize the following function over $(\mathbf{W}, \mathbf{Z})$ for fixed $\mu > 0$ and drive $\mu \to \infty$:

$$E_2(\mathbf{W}, \mathbf{Z}; \mu) = \sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1})\|^2 + \frac{\mu}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \|\mathbf{z}_{k,n} - \mathbf{f}_k(\mathbf{z}_{k-1,n}; \mathbf{W}_k)\|^2.$$

3 Optimization of the quadratic-penalty objective

- This defines a continuous path $(\mathbf{W}^*(\mu), \mathbf{Z}^*(\mu))$ which, under mild assumptions, converges to a minimum of the constrained problem.
- The MAC formulation breaks the functional dependencies in the nested mapping $\mathbf{f}$ and unfolds it over layers. Every squared term involves only a **shallow** mapping, improving the conditioning of the problem; and the derivatives required are simpler.
- We apply **alternating optimization** over $\mathbf{Z}$ and $\mathbf{W}$.
- $\mathbf{W}$-step: a **separate nonlinear, least-squares regression for each hidden unit**, of the form $\min_{\mathbf{w}_{kd}} \sum_{n=1}^{N} (\mathbf{z}_{kd,n} - f_{kd}(\mathbf{z}_{k-1,n}; \mathbf{w}_{kd}))^2$, $k = 1, \dots, K$ and $d = 1 \dots, H$. We solve each of these KH problems with a Gauss-Newton approach with a simple line search procedure, which practically converges in 1–2 iterations.
- $\mathbf{Z}$-step: minimizing over $\mathbf{Z}$ for fixed $\mathbf{W}$ **separates over each $\mathbf{Z}_n$ for $n = 1, \dots, N$**. The problem is also a nonlinear least-squares fit, formally very similar to those of the $\mathbf{W}$-step. We optimize it again with Gauss-Newton method.
- **Because each $\mathbf{W}$- or $\mathbf{Z}$-step operates over very large blocks of variables, the decrease in the quadratic-penalty objective function is large in each iteration, unlike the tiny decreases achieved in the nested function.**
- One should increase μ as quickly as possible, in order to approach the solution faster, but not too fast that ill-conditioning would prevent progress. Early stopping criterion can be used.
- **MAC affords a good parallelization, due to the decoupling (on $\mathbf{W}$ or on $\mathbf{Z}$) into many small independent problems.**

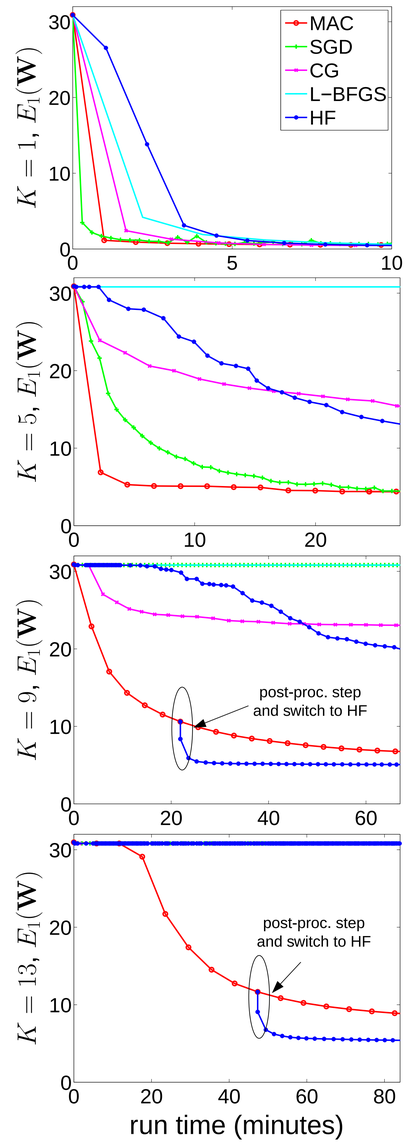
5 Discussion

Even with a simple optimization (quadratic-penalty with exact steps) and without parallelism or GPUs, MAC is competitive with heavily engineered state-of-the-art methods. Many easy speedups are possible:

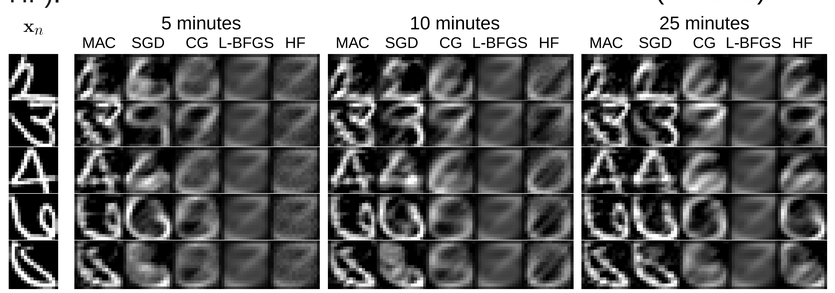
- Embarrassingly parallel steps.
- The Z-step is expensive (linear system of $\dim \mathbf{z}_n = KH$ variables for each data point), but there are many ways to approximate it so its cost is comparable to a backprop step.
- Stochastic updates using minibatches.
- etc.

4 Experimental results

- Reconstructing USPS input image through a nonlinear, deep autoencoder. Training/validation set size: 5 000.
- The architecture consists of K layers of sigmoidal hidden units, and each layer contains the same number of units H, with another linear output layer.
- We focus on the **depth** of the architecture and vary K from 1 to 13, while keeping the total number of weight parameters (around $(K-1)H^2 + 2DH$) approximately constant (about 92 000).
- Report reconstruction error vs runtime on a **single processor** for different algorithms, with carefully tuned hyper-parameters: method of auxiliary coordinates (MAC), stochastic gradient descent (SGD), nonlinear conjugate gradient (CG), limited memory BFGS (L-BFGS), Hessian Free (HF). Weights randomly initialized using the fan-in rule.



⇒ Mean squared error per input $E_1(\mathbf{W})$ as a function of run time. Markers shown every epoch (SGD), every 100 iterations (CG, L-BFGS) or every 1 iteration (MAC, HF).



Autoencoder ($K = 5$) output of several difficult examples $\mathbf{x}_n$ in the validation set obtained by each algorithm at three times.