A NEW FORMULATION FOR DEEP NEURAL NET LEARNING
Miguel Á. Carreira-Perpiñán and Weiran Wang.
EECS, University of California, Merced.

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:
  \[ E_1(W) = \sum_{n=1}^{N} \| y_n - f(x_n; W) \|_2^2, \]
  \[ f(x; W) = f_{K+1} \cdots f_1(x; W_1; W_2) \cdots W_{K+1}, \]
  where each layer function has the form \( f_k(x; W_k) = \sigma(W_k x) \), 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.

The method of auxiliary coordinates (MAC)

- We introduce one auxiliary variable per data point and per hidden unit and define the following equality-constrained optimization problem:
  \[ E(W, Z) = \sum_{n=1}^{N} \| y_n - f_{K+1}(z_{K,n}, W_{K+1}) \|_2^2 \]
  \[ s.t. \{ z_{K,n} = f_k(z_{k-1,n}, W_k) \}, n = 1, \ldots, N. \]

  - One way to solve this problem is quadratic-penalty method. We optimize the following function over \( (W, Z) \) for fixed \( \mu > 0 \) and drive \( \mu \to \infty; \)
  \[ E_d(W, Z, \mu) = \sum_{n=1}^{N} \| y_n - f_{K+1}(z_{K,n}, W_{K+1}) \|_2^2 + \frac{\mu}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \| z_{k,n} - f_k(z_{k-1,n}, W_k) \|_2^2. \]

Net with \( K = 3 \) hidden layers (\( W_k; \) weights, \( z_k; \) auxiliary coordinates).

Optimization of the quadratic-penalty objective

- This defines a continuous path \( [W^*(\mu), Z^*(\mu)] \) which, under mild assumptions, converges to a minimum of the constrained problem.

- The MAC formulation breaks the functional dependencies in the nested mapping \( f \) 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 \( Z \) and \( W \).

  - \( W \)-step: a separate nonlinear, least-squares regression for each hidden unit, of the form \( \min_{w_{K+1}} \sum_{n=1}^{N} (y_n - f_{K+1}(z_{K,n}; W_{K+1}))^2 \), \( k = 1, \ldots, K \) and \( d = 1, \ldots, H \).
  
    - 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.

  - \( Z \)-step: minimizing over \( Z \) for fixed \( W \) separates over each \( z_{K,n} \) for \( n = 1, \ldots, N \). The problem is also a nonlinear least-squares fit, formally very similar to those of the \( W \)-step. We optimize it again with Gauss-Newton method.

- Because each \( W \)- or \( Z \)-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 \( \mu \) 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 \( W \) or on \( Z \)) into many small independent problems.

Experimental results

- Reconstructing USPS input image through a nonlinear, deep autoencoder. Training/validation set size: \( 5000 \).

- The architecture consists of \( K \) layers of sigmoids 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 \( 92000 \)).

- 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 \( E_1(W) \) as a function of run time. Markers shown every epoch (SGD), every 100 iterations (CG, L-BFGS) or every 1 iteration (MAC, HF).

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:

- Embracing parallel steps.

- The \( Z \)-step is expensive (linear system of \( \dim z_k = KH \) 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.