Affine Invariant Stochastic Optimization

Jose V. Alcala-Burgos (vidal@cimat.mx)
www.cimat.mx/~vidal/

CONACYT Consejo Nacional de Ciencia y Tecnología

CIMAT Centro de Investigación en Matemáticas, A.C.

Abstract

Training of deep neural networks is a computational challenge in machine learning applications. The most successful training methods are online algorithms, like stochastic gradient descent (SGD). Online methods can take advantage of parallel computing only on the step design; thus, we look for algorithms that converge in fewer steps at the cost of extra parallelizable work on each step.

We introduce the Affine Invariant Stochastic Gradient Descent (AISGD), an online algorithm with a built-in approximation of the Hessian matrix of the objective function. Moreover, most of the work on each step of this algorithm is plain matrix multiplication, and therefore full parallelization is feasible.

Acknowledgements

The authors wish to thank J. Goodman for many helpful discussions throughout the duration of this project.

Problem setup

The problem at hand is to find θ_* minimizing $f(\theta)$ when we have samples W_θ; such that,

$$\nabla f(\theta) = E[W_\theta]\,. \tag{1}$$

A general Robbins-Monro iteration takes the form

$$\widehat{\theta}_{n+1} = \widehat{\theta}_n - \frac{1}{n} K\, W_n\,, \tag{2}$$

The optimal K is the inverse of the Hessian of f at the optimal θ_*. We follow Bottou and aim to minimize

$$\mathcal{L} = E[f(\widehat{\theta}_n) - f(\theta_*)]. \tag{3}$$

The numerical analysis is based on the spectral decompositions with increasing eigenvalues:

$$\begin{aligned} H &= P\Lambda P^T = D^2 f(\theta_*) \\ \Sigma &= QDQ^T = \mathrm{Cov}\,(W_{\theta_*}, W_{\theta_*}) \end{aligned} \tag{4}$$

Lower bound for SGD error

The sharp asymptotic lower bound for the SGD algorithm ($K = \frac{1}{\lambda_1} I$) is

$$\frac{1}{2}\mathrm{tr}\left[H^{-1}\Sigma\right] + \frac{1}{4} s\, \frac{(k_* - 1)^2}{(k_* - \frac{1}{2})} \leq \mathcal{L}\,, \tag{5}$$

with

$$\begin{aligned} M &= Q^T P, \qquad & c_i^2 &= \sum_{k=1}^{p} d_k M_{ki}^2\,, \\ s &= \mathrm{tr}[\Sigma], \qquad & k_* &= \frac{1}{\frac{1}{s}\sum_{i=1}^{p} \frac{\lambda_1}{\lambda_i} c_i^2}\,. \end{aligned} \tag{6}$$

The second term in the LHS of (5) disappears when $K = H^{-1}$.

H^{-1} online estimator

Let X_n and Y_n be the corresponding $n \times (p+1)$ and $n \times p$ matrices with entries

$$x_k = (\widehat{\theta}_k, 1), \qquad y_k = W_{\widehat{\theta}_k}\,,$$

consider the linear regression $Y = XB$ and denote the first p rows of a matrix M by $\overline{M}$. We calculate the natural estimators

$$\begin{aligned} B_n &= (X_n^T X_n)^{-1} X_n^T Y, \qquad & H_n &= \overline{B_n} \\ G_n &= \overline{B_n}^{-1}, \qquad & K_n &= \frac{G_n + G_n^T}{2}\,, \end{aligned} \tag{7}$$

and use K_n as our H^{-1} estimator. We use the *online update*

$$\begin{aligned} s_{n+1} &= \frac{1}{1 + x_{n+1} P_n x_{n+1}^T} \\ u_{n+1} &= s_{n+1} \overline{P_n x_{n+1}^T} \\ v_{n+1} &= y_{n+1} - x_{n+1} B_n \\ t_{n+1} &= \frac{1}{1 + v_{n+1} G_n u_{n+1}} \\ G_{n+1} &= G_n - t_{n+1} G_n u_{n+1} v_{n+1} G_n \\ B_{n+1} &= B_n + s_{n+1} P_n x_{n+1}^T v_{n+1} \\ P_{n+1} &= P_n - s_{n+1} P_n x_{n+1}^T x_n P_n^T \end{aligned} \tag{8}$$

The algorithm uses $O(p^2)$ *parallelizable operations* and new matrices are computed through *rank one updates*.

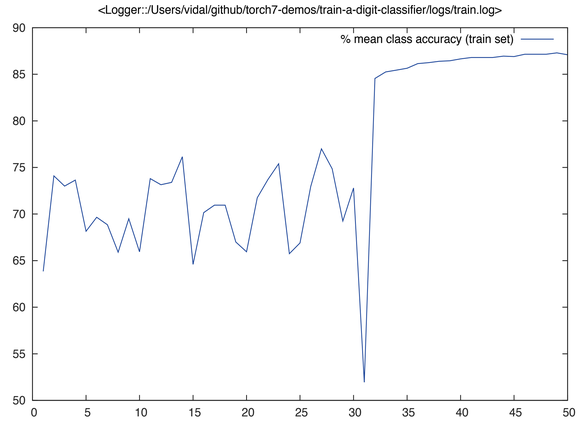
Simulations in Torch7

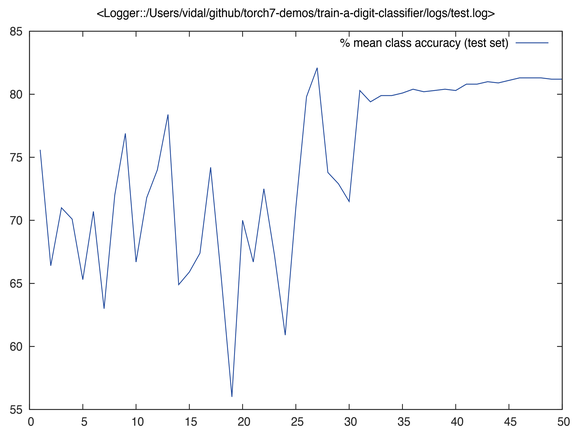
Torch7 is is an open source library maintained by Idiap, NYU and NEC.

- Supports CUDA and OpenMP
- Recent neural networks are implemented, like *Dropout*

MNIST

- convolutional neural network (cnn) with 4,000 parameters.
- epochs 1-30 with SGD and 31-50 with AISGD





https://bitbucket.org/vidalalcala/affine-invariant-sopt/

Low rank approximation

Given the *thin SVD* decomposition $G = USV^T$, with $S \in M_{r\times r}$, find the decomposition

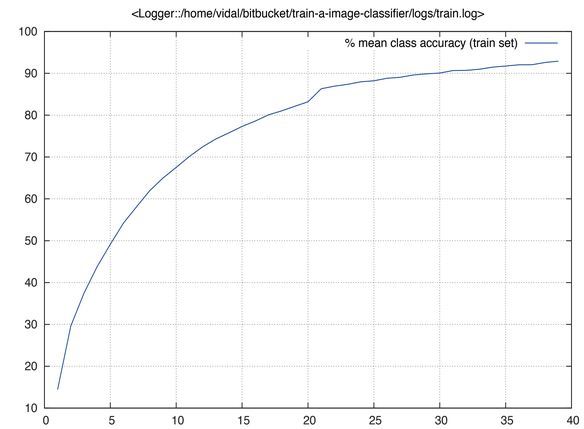
$$G + ab^T = \overline{U}\,\overline{S}\,\overline{V}^T, \qquad \overline{S} \in M_{(r+1)\times(r+1)}\,,$$

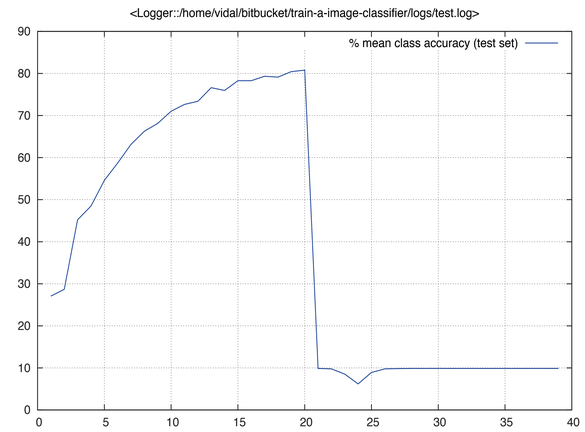
with the steps

$$\begin{aligned} &m = U^T a; \quad p = a - Um, \quad R_a = \|p\|; \quad P = R_a^{-1} p\,, \\ &n = V^T b; \quad q = b - Vn, \quad R_b = \|q\|; \quad Q = R_b^{-1} q\,, \\ &K = \begin{bmatrix} S + mn^T & \|q\| m \\ \|p\| n^T & \|p\|\|q\| \end{bmatrix}, \quad K = U' S' V'^T \\ &\overline{U} = [U\ P]\, U'; \qquad \overline{S} = S'; \qquad \overline{V} = [V\ Q]\, V' \end{aligned}$$

CIFAR-10

- cnn with Dropout and 9,000,000 parameters.
- epochs 1-20 with SGD and 21-40 with AISGD





https://bitbucket.org/vidalalcala/train-a-image-classifier