Accelerating SGD for Distributed Deep Learning Using Approximted Hessian Matrix

Séb Arnold - arnolds@usc.edu *Chunming Wang* - cwang@math.usc.edu

University of Southern California

Abstract

We present a novel Quasi-Newton method for large scale optimization in the distributed regime. Our contribution takes advantage of the differences in the gradient information of each replica to estimate the singular values and directly computes $H^{-1}v$, the product of the inverse Hessian and an arbitrary vector. More importantly, it paves the way for distributed optimization algorithms that take advantage of local information to infer higher-order statistics. We also provide preliminary benchmark results on MNIST and CIFAR-10.

Introduction

This time is spent finding optimal parameters w_{opt} for the network F, which is best phrased as a minimization problem of the expected loss $\mathcal{L}$ given input-target pairs (x, y) sampled from distribution χ:

$$w_{opt} = \arg\min_{w} \mathbb{E}_{(x,y)\sim\chi}[\mathcal{L}(y, F(x; w))]$$

It is common to use Stochastic Gradient Descent (SGD) - or its derivates[1] - to solve the optimization problem. In order to reduce variance in the estimate of the gradients, it is common to use *batches* of input-target pairs and average the individual gradients. This makes parallelizing SGD trivial, as each replica is attributed a batch of size $\frac{B}{M}$:

$$w_{t+1} = w_t - \frac{\alpha}{M} \cdot \sum_{m}^{M} \nabla_{w_t} \mathcal{L}(y_m, F(x_m; w_t))$$

Problem: *Even in a distributed setting, training large networks takes weeks.*

This problem arises from intrisic properties of the deep learning setup (deep models on big datasets), and of convergence properties of SGD. In particular, Dauphin & al. have suggested that the loss surface of deep neural networks doesn't consist of many local minimas - they become exponentially rarer as dimensionality increases - but have numerous saddle-points and long plateaus surounding those minimas. These regions have very small magnitude in the gradients, which might drastically slow down SGD and other first-order methods. (See Figure 1)

Insight: *Combine the local information of each replica to estimate the curvature.*

By leveraging the curvature, we would be able to approximate Newton's method which uses information from the second-order derivative - the Hessian H. Concretely, the equation for Newton's method is given by

$$w_{t+1} = w_t - \alpha \cdot H_{\mathcal{L}}^{-1} g \tag{1}$$

where $g = \nabla_{w_t}\mathcal{L}$. Note that implementing Newton's method is rather challenging since it involves the inversion of a $N \times N$ matrix, where N is the number of parameters of the neural network and on the order of several millions.

[1] C.f. our blog post on popluar improvements to SGD: https://seba-1511.github.io/dist_blog/

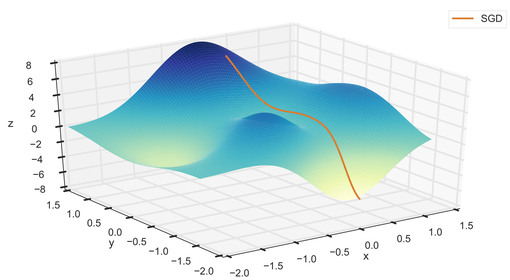

Figure 1: Saddle-points and plateaus slow down SGD

Method

Our method is based on the observation that an approximation of the inverse Hessian to gradient vector product can be obtained by a first order finite difference of the gradient

$$H_{\mathcal{L}}(w_k - w_j) \approx \nabla\mathcal{L}(w_k) - \nabla\mathcal{L}(w_j) = g_k - g_j \tag{2}$$

where we use a shorthand notation $g_i = \nabla\mathcal{L}(w_i) = \nabla_{w_i}\mathcal{L}(y_d, F(x_d; w_i))$ for the gradient of the ith replica with parameters w_i. We define $\tilde{g} = \frac{1}{M}\sum_m^M g_m$ the mean gradient and $\tilde{w} = \frac{1}{M}\sum_m^M w_m$ the mean parameters, as well as matrices

$$G = [g_1 - \tilde{g}, \dots, g_M - \tilde{g}] \text{ and } W = [w_1 - \tilde{w}, \dots, w_M - \tilde{w}]$$

Equation 2, leads to $G = \hat{H}_{\mathcal{L}} W$, where $\hat{H}_{\mathcal{L}}$ represents key characteristics of the Hessian matrix. We first obtaina singular value decomposition of $G = U_G \Sigma_G V_G^H$ by finding eigenvalues and eigenvectors of $G^T G$. Note that $G^T G$ has dimensions $M \times M$ and is thus rather small. This gives us

$$V_G = [v_1^{(G)}, \dots, v_M^{(G)}] \text{ together with } \Sigma_G = [e_1\sigma_1^{(G)}, \dots, e_m\sigma_M^{(G)}]$$

where $v_i^{(G)}$, $\sigma_i^{(G)}$ are the ith eigenvector and eigenvalue of $G^T G$, respectively. Using equation 2, we get

$$\hat{H}_{\mathcal{L}}^{-1} G = W \implies \hat{H}_{\mathcal{L}}^{-1} U_G \Sigma_G = X V_G$$

We now introduce matrices $P = \hat{H}_{\mathcal{L}}^{-1} U_G$ and $Y = X V_G$, and notice that the ith column of P is given by $P_i = \frac{1}{\sigma_i^{(G)}} Y_i$. Thus for any vector z we have

$$\hat{H}_{\mathcal{L}}^{-1} z \approx P U_G^H Y z$$

It remains to find the M (necessary) orthonormal columns of U_G which are computed with

$$u_i^{(G)} = \frac{G v_i^{(G)}}{||G v_i^{(G)}||_2}$$

A nice property of the above method is that when working with gradients (or in fact any vector z such that $z = \sum_m^M z^H u_m^{(G)} u_m^{(G)} + z^{\perp}$, provided G as rank $\geq M$), we can write

$$\hat{H}_{\mathcal{L}}^{-1} z = z^{\perp} + \sum_m^M z^H u_m^{(G)} \sigma_m^{-1} X_g v_m^{(G)}$$

Note: Our method directly approximates the largest eigenvalue $\sigma_{\max}$, which according to LeCun al. can be used to compute the optimal learning rate $\alpha_{\text{opt}} = \frac{1}{\sigma_{\max}}$.

Results

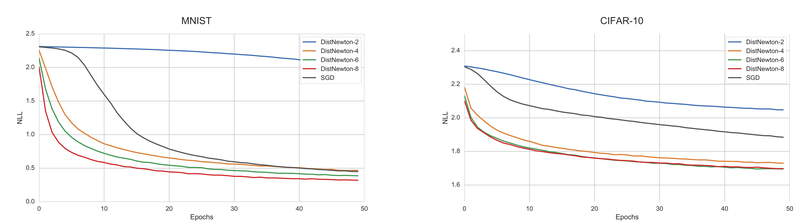

Figure 2: Results on MNIST and CIFAR-10 datasets

We now present preliminary results on two widely used datasets (MNIST and CIFAR-10) and compare the convergence rate of our proposed method against SGD. Furthermore, we restrict our study to the 1-step synchronous case. Since we are only interested in the optimization performance, we keep most of our hyper-parameters constant, including learning rates (0.0003 and 0.01), activation functions (ReLU and tanh), and a global batch size of 256. Our model is a 5 layer convnet with about 16'000 parameters, which we train each time for 50 epochs. We report the negative log-likelihood on the train dataset at every epoch, and for up to 8 replicas. Note that since the global batch size is fixed, the SGD convergence curves are identical and thus only reported once.

Our results clearly demonstrate convergence improvements as we scale to a larger number of replicas, and consistently outperforms stochastic gradient descent in the most distributed case. Interestingly the latter is true even with a relatively small number of replicas, as we observe a much faster convergence with $m = 4$ in both experiments. However, when the number of replicas is not sufficient to properly estimate the most influential singular values the method converges to poor minimas and is slower than distributed SGD.

Summary

- Newton's method offers several attractive properties for high-dimensional optimization.
- In the distributed setting, we propose a method to compute a scalable estimate of $H^{-1}v$, as well as the eigenvalues/eigenvectors of the Hessian.
- Our method is trivially implemented in both the Parameter-Server and Tree-Reduction frameworks.

Acknowledgements

Sponsorship of the Living With a Star Targeted Research and Technology NASA/NSF Partnership for Collaborative Space Weather Modeling is gratefully acknowledged.

References

- Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex op- timization. In Advances in neural information processing systems, pages 2933–2941, 2014.
- Yann LeCun, Patrice Y Simard, and Barak Pearlmutter. Automatic learning rate maximization by on-line estimation of the hessians eigenvectors.