An Empirical Comparison of Optimizers for Deep Learning

Terkel Bo Olsen[a], Jesper Wohlert Hansen[a], Anmol Porwal[b]
Department of Mathematics[a], *Department of Computer Science*[b], *EPFL, Switzerland*
{terkel.olsen, jesper.hansen, anmol.porwal}@epfl.ch

I. Introduction

In this report, different optimizers and their convergence properties are tested in an empirical study concerning fitting of two different sized neural networks to a small dataset. Concretely, the use of Quasi-Newton methods for optimizing small neural networks are investigated and compared to general and adaptive gradient methods. The investigated Quasi-Newton method is the L-BFGS method which is compared to gradient- and stochastic gradient descent based optimizers, as well as more advanced adaptive gradient methods like Adam and RMSprop. In the analysis, both convergence and resource usage of each optimizer is examined. The optimizers in question are those implemented in the PyTorch [1] framework.

II. Comparison of optimizers

In order to investigate the performance of different optimizers on an empirical basis, we employ the Iris flower data set [2] and construct two fully connected deep neural network. The first, a *small* single hidden layer network consisting of 260 parameters and a second *large* network with 3 hidden layers totalling 4451 parameters.

A. Gradient methods

We denote the learning rate by γ and consider the following gradient methods, classical gradient descent (GD, $\gamma = 3e^{-2}$), mini-batch stochastic gradient descent and true stochastic gradient descent (SGD, $\gamma = 9e^{-2}$). For the first experiment, these optimizers are compared to the Quasi-Newton method L-BFGS. In order to ensure convergence of the stochastic methods, we shrink the learning rate exponentially by a factor of 0.001 after each epoch. Figure 1 displays the test loss as a function of gradient steps for these methods. A similar procedure is performed with regards to the larger network, for which the loss is shown in figure 2. From an initial analysis of the plots, we notice that stochastic methods induces large fluctuations on the loss curve which is caused by a resampling of the loss function at each iteration. This might seem like an issue, but by reducing the learning rate exponentially, the stochastic optimizers still converges in expectation to the same minimum as their non-stochastic counterpart. It is also clear that the convergence of L-BFGS is much faster, which is a consequence of the near quadratic convergence obtained by the fact that an approximation of the Hessian matrix is made. Albeit the Quasi-Newton method might seem superior to the first-order methods on the small network, it appears to

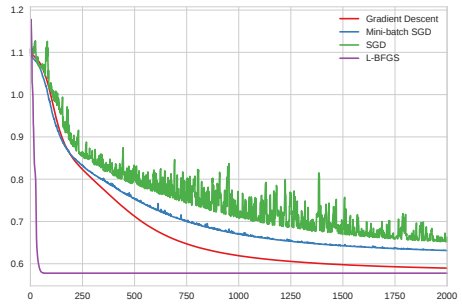

Fig. 1. Convergence of the test loss for gradient descent, single-sample SGD, mini-batch SGD and L-BFGS on the small neural network.

stutter more on the larger network. To better understand this behaviour, we perform an auxiliary experiment by making ten random restarts of the L-BFGS optimizer and recording the test loss on the large network - shown in Figure 3. What we see is that L-BFGS is very fragile to initial conditions in the context of deep neural networks. A likely explanation for this phenomena is that the optimization landscape of deep networks is complex, consisting of many local minima and saddle points. Some authors argue that Quasi-Newton methods converge to these saddle-points, which does not provide a good generalization to the test set [3]. The best minimum in deep networks are not global minima, because it can be shown by permutation of weights that there are several of these, but rather a good minimum is a flat minimum. We theorize that

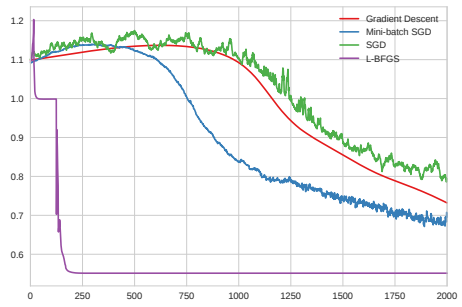

Fig. 2. Convergence of the gradient based methods for the larger network.

introducing a backtracking line search algorithm in the L-BFGS algorithm would substantially stabilize its convergence in larger networks, however this will further increase the resource usage and hence might not be suitable in practical applications.

Conversely, we observe from Figure 2 that stochastic optimization methods do not fall prey to saddle-points in the same way. The reason is likely that noisy gradient estimates enable the methods to escape these points by finding better parameters in the neighbourhood. This is essentially a major point of contention between deterministic second-order methods and stochastic first-order methods that shows that performance is highly dependent upon the optimization landscape.

B. Adaptive methods

Naive stochastic gradient descent methods have recently been overtaken in popularity by adaptive methods for deep learning. Methods such as Adam [4] and RMSprop [5] save simple statistics of the gradient updates over time in order to provide a better estimate of future gradients. Figure 4 shows that these methods converge at a faster, more steady pace than the basic first-order methods described previously due to the fact that they can adjust their learning rate per parameter at each gradient step. Further, it is clear that these methods eventually converge to the same minimum as the L-BFGS optimizer and that the convergence is faster than for gradient descent. This makes adaptive methods highly preferred as they seem to combine the best parts of previous methods even for large networks. In Figure 5 it is now clear that having increased the network size, the convergence of the full gradient descent algorithm has become much slower while the adaptive methods ($\gamma = 1e^{-2}$) suffer no considerable losses when optimizing highly parametric models.

III. Discussion and Conclusion

In classical optimization theory, it is known that using information provided by the Hessian (or an approximation thereof) enables the optimizer to be invariant under linear transformations, leading to faster convergence. As gradient

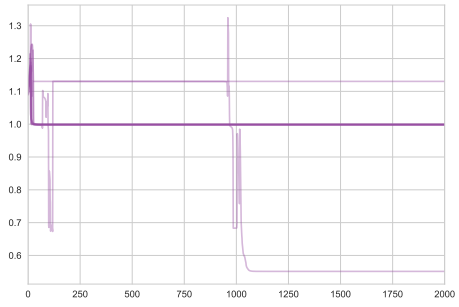

Fig. 3. Convergence of 10 different initializations of L-BFGS on the larger network. Opacity indicate a new restart. Only one of the 10 runs succeed in converging to the same minimum as found in Figure 2 and 5

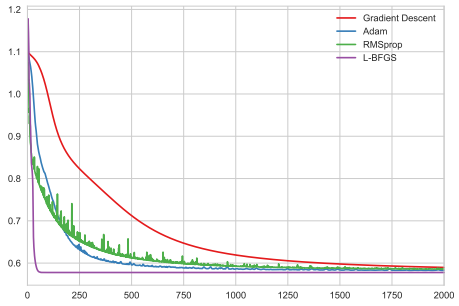

Fig. 4. Convergence of adaptive methods on the small neural network.

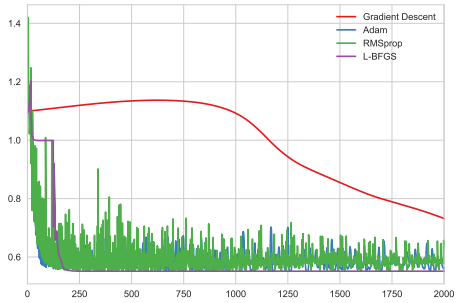

Fig. 5. Convergence of adaptive methods on the larger network.

methods do not use this information, they might be at a disadvantage in comparison to Quasi-Newton methods such as L-BFGS, which could explain the slower convergence on larger networks.

However, Quasi-Newton methods are impractical when working with real world problems due to the computational complexity of approximating the Hessian matrix. If we consider networks with millions of parameters, the dimension of the Hessian matrix scales by $\mathcal{O}(d^2)$, which cannot be stored in memory. Further, Quasi-Newton methods tend to find local minima like saddle points which do not generalize well to the test set.

One can argue that by basing the gradient estimates on the moments of previous estimates for each coordinate, as with RMSProp and Adam, is actually equivalent to a noisy approximation of the preconditioning matrix [6], which can be done in $\mathcal{O}(d)$ time. This makes adaptive methods highly desirable for practical purposes.

Therefore, for all practical problems, one should seek optimizers that strikes a balance between use of computational resources, convergence properties and generalization to a test set. A similar discussion follows for the use of SGD, where using a batch-size of 1 is likewise computationally inefficient as it fails to leverage the batch computation capabilities of modern GPUs. Additionally, it was clear that mini-batch training significantly stabilized the convergence to the minimum.

REFERENCES

[1] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[2] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of human genetics*, vol. 7, no. 2, pp. 179–188, 1936.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[5] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[6] Y. N. Dauphin, H. de Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," *arXiv preprint arXiv:1502.04390*, 2015.