Towards Mini-Error with Maxi-Batch SGD

Abstract

Deep neural networks and the stochastic gradient algorithm with which they are optimized have been applied with astonishing success to a variety of problems too large and non-convex for full-batch, shallow methods. With the increasing availability and power of computing resources, "S" in SGD is less of a necessity of data size, but remains a requirement imposed by non-convexity. This work will demonstrate that the use of large mini-batches (*maxi-batches*) diminishes the robustness of SGD to regions of high curvature and causes it to yield solutions of particularly low quality. In doing so, this work also introduces a principled method for measuring the curvature of the loss surface, using this method, offers several new insights into the causes of failure in maxi-batch SGD. The results of this study are synthesized into a hypothesis that may guide further inquiry into this problem.

1 Introduction

In the modern era of large and high-dimensional data, there is an increasing interest in using deep learning to understand it. Due to the non-convexity of the learning problem and the size of the data, (some variant of) the stochastic gradient method (SGD) is the optimization algorithm of choice. The stochasticity is the result of estimating the gradient using a *mini-batch*, a small independent sample of the data. Of course, as the popularity of deep learning grows, so, too, does the capacity of its computing substrates. Practitioners, desiring to rapidly train models, may thus be tempted to overlook the issue of non-convexity and attempt something closer to full-batch gradient descent using a *maxi-batch*. Unfortunately, maxi-batch models fall into poor local minima that have been demonstrated to generalize poorly [7]. Although the stability bounds of SGD [3] suggest an increase in generalization error as batch size increases, what this work will show is that maxi-batch SGD yields models that also *train* poorly. Namely, this is more than a problem of generalization.

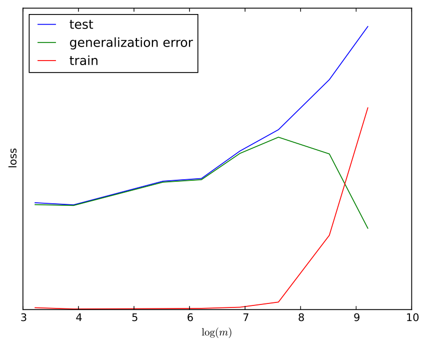

Figure 1: Performance is so impaired that the generalization gap actually decreases!

Over course of numerical experiments, the hypothesis that emerges is that that maxi-batch SGD quickly falls into a steep local minimum and then stays there for the duration of optimization. While this may not be a novel observation, the tools and techniques introduced in the experiments may offer new insight into the problem and methods towards a solution.

MIT 9.520/6.860: Statistical Learning Theory and Applications (Fall 2016).

The remainder of this paper will present the experiments and their results in section 2, offer an interpretation in section 3, and conclude that there is yet more work to be done in section 4.

2 Experiments

To enhance the relevance of the results, the experiments were conducted using a 32-layer ResNet [5] convolutional neural network which, with additional layers, yields state-of-the art performance on several visual task including object and scene recognition. The task of classifying the CIFAR-10 dataset was chosen for its difficulty despite having a size amenable to maxi-batch training. Models were trained using batch sizes $m \in \{50, 100, 250, 500, 1000, 5000, 10000\}$; the largest of which represents one fifth of the training set. Notably, the top performing model uses a batch size of $m = 50$ and learning rate of $\eta = 0.1$ to achieve a test accuracy of 93.1%.

2.1 Isolating the effect of batch normalization

Batch normalization [6], which centers activations within mini-batch using empirical statistics computed on the mini-batch itself, has become ubiquitous for its ability to enable training of very deep networks. As the only model component that explicitly depends on batch size, one may reasonably wonder if this may be the cause of SGD failure. This motivates a preliminary experiment to rule out batch norm as a primary cause of SGD failure. To this end the maxi-batch models are retrained using a modified version of batch norm in which centering is done using statistics computed on a fixed-size subsample of 50 inputs. If batch normalization were the culprit, returning its operation to the mini-batch regime would return performance to the mini-batch levels. The results of this experiment, which actually show a slight decrease in performance, support the original hypothesis that the issue arising during maxi-batch SGD is more systemic than this single component.

2.2 Minimizer interpolation

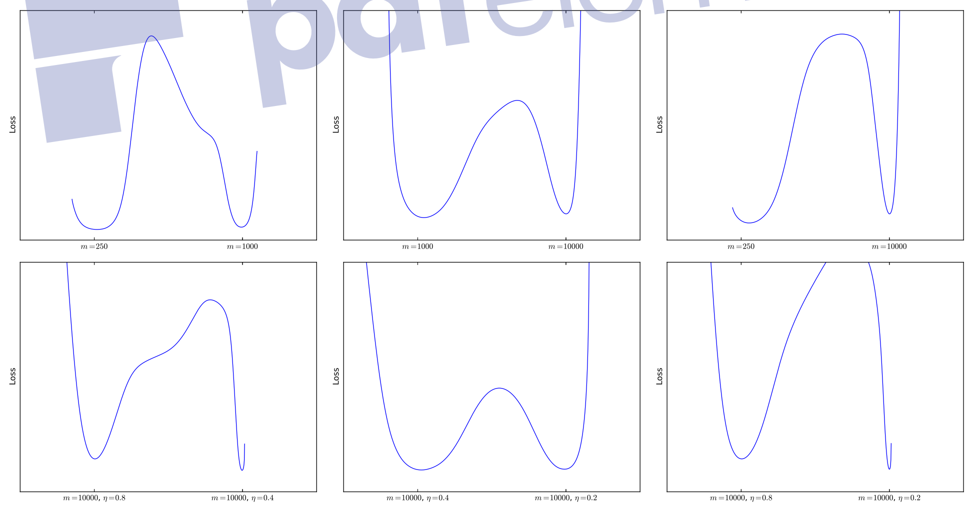

Figure 2: Parametric plots of loss while interpolating between models trained with different batch sizes (top) and, for $m = 10000$, different learning rates (bottom).

With evidence to support that no peculiarity of the model architecture is to blame for the originally stated problem, the first true experiment is a reproduction of the results in [7] that offer convincing evidence of the tendency for maxi-batch SGD to find sharp minima. Specifically, using the method of [1], parametric plots of the loss function can be generated for a convex combination of two models' parameters. In other words, by interpolating between the weights of models trained with different batch sizes, one can obtain a 1D slice of the loss function between them.

As shown in Figure 2, the minima found by SGD become increasingly sharp as batch size is increased. Furthermore, this work extends that of the original to show the effect of varying learning rate in the maxi-batch case; this is motivated by restoring the explorative power of SGD that is missing from the low-variance gradient estimate. Notably, as hypothesized, increasing the learning rate does improve the sharpness of the resulting minimum but still does not offer a full solution to the problem. This theme will recur in further experiments.

2.3 Quantifying sharpness using the $\mathcal{R}\{\cdot\}$ technique

Having demonstrated the existence of minima with varying degrees of convexity, a natural question is whether their local curvatures can be quantified. A previous attempt in this regard [7] led to the notion of "sharpness" which resembles a finite difference. Specifically, given a neighborhood $\mathcal{C}$, the sharpness ϕ is defined as

$$\phi(f, \mathcal{C}) = \max_{y \in \mathcal{C}} \frac{f(x+y) - f(x)}{1 + f(x)}$$

Unfortunately, in addition to being only a rough approximation of the maximum curvature, this measure has the odd property of not being robust to adding a constant to f. Perhaps a better solution would be to measure the Hessian directly. One major contribution of this work is a technique for measuring the sharpness of minima using the matrix of second derivatives.

Although computing the Hessian is computationally infeasible, one may still obtain both the maximum curvature and direction thereof by finding the largest eigenpair. This is enabled by the so-called $\mathcal{R}\{\cdot\}$ operator [10] which is defined as

$$\mathcal{R}_{\mathbf{v}}(f(\mathbf{w})) = \frac{\partial}{\partial r} f(\mathbf{w} + r\mathbf{v})|_{r=0}$$

and allows multiplication of the Hessian by a vector:

$$\mathcal{R}_{\mathbf{v}}(\nabla_w f(\mathbf{w})) = \frac{\partial}{\partial r} f'(\mathbf{w} + r\mathbf{v})|_{r=0} = \mathbf{Hv}$$

In addition to its use in quantifying sharpness, this work contributes an efficient multi-GPU implementation[1] of the $\mathcal{R}$ operator for the Torch7 deep learning framework.

With the ability to efficiently compute $\mathbf{Hv}$, the maximum eigenpair of $\mathbf{H}$ may be found using power iteration, which is guaranteed to converge since $\mathbf{H}$ is real and symmetric. This technique can be applied directly within a neural network during training to find $\lambda_{\max}$. This value, in turn, may be interpreted as the convexity/sharpness of a minimum. Accordingly, Figure 3 and Figure 4 show how $\lambda_{\max}$ evolves over the course of training. In these experiments, when computing $\lambda_{\max}$, m is set to 500 to reduce the variance of the estimate.

One may observe, in Figure 3, that $\lambda_{\max}$ is strongly associated with generalization error, which supports the hypothesis that sharp minima generalize poorly. Again, it can be seen that increasing learning rate is helpful but not sufficient. Furthermore, the significant increase in $\lambda_{\max}$ when the learning rate is dropped at epoch 80 suggests that loss surface for this task actually increases in convexity near the minimum (perhaps not dissimilarly to a plot of $\sqrt{|x|}$). Interestingly, this final conjecture runs contrary to the measurements of "sharpness" [7] for which the values decrease as the small batch models converge.

The results in Figure 4, further confirm that maxi-batch SGD is hindered by its tendency to find not only sharp *minima*, but also sharp regions of the loss function, in general. Moreover, a plausible interpretation of large increase in convexity while training the mini- but not maxi-batch model is that the maxi-batch model initially falls into a steep basin/ravine and stays there for the remainder of optimization. This explanation is supported by the following experiments.

2.4 Gradient Noise

The initial explanation by [7] for the tendency of maxi-batch SGD to find sharp minima is that the reduced variance of gradient estimates impair its ability to "hop out of" steep basins. Intuitively, one would expect that noisy gradients cause optimization to more fully explore the parameter space. Figure 5, which plots the trek of the parameters over time, implies that this is the case. Thus, it would seem as though a simple solution is to reintroduce variability during maxi-batch training–specifically

[1]https://github.com/nhynes/th-lib/blob/master/rbackprop.moon

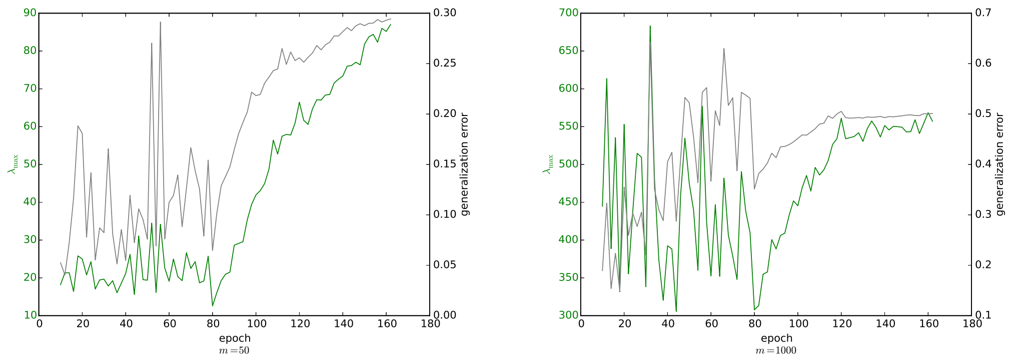

Figure 3: $\lambda_{\max}$ and generalization error during training for $m = 50$ (left) and $m = 1000$ (right).

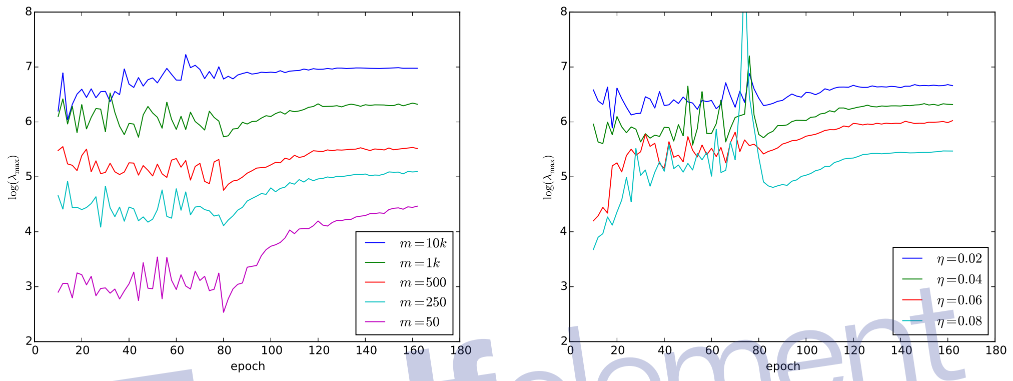

Figure 4: Average $\lambda_{\max}$ over the test set for each epoch of training. The plot on the left varies m while the plot on the right fixes $m = 10000$ but varies the learning rate, η.

through the addition of noise to the gradient. Indeed, the technique of gradient noise has been applied in similar deep networks [9] to improve generalization. To both test whether decreased variance is the main issue and whether gradient noise is an effective solution, two tests are conducted: the first is a straightforward implementation of [9] in which pointwise independent Gaussian noise with a fixed, global variance is added to the gradients entering each convolutional layer; the second test modifies the first to use noise variances computed empirically from the $m = 50$ model. The motivation behind the second setup, which is well supported by observations, is that variance is higher in lower layers and, overall, decreases over the course of training.

In each case, despite manual fine-tuning, the maxi-batch models' training, in the best case, does not improve and, in the worst case, becomes unstable. One explanation for this is that the *type* of noise (i.e., non-diagonal covariance) plays a significant role in how mini-batch SGD explores the space. One avenue of future work is to train many mini-batch models and estimate a spatial noise process for use in the maxi-batch models.[2]

2.5 Adaptive linearization

This final experiment is based on the idea of *continuation* [8], which involves optimizing a smoothed version of an objective function while progressively reducing the amount of smoothing until a solution to the original problem is found. The previous results–particularly those in Figure 5–motivate its use here in that maybe if optimization can avoid falling into a minimum near the origin, it will venture outward to explore the region traversed by mini-batch SGD. Furthermore, mini-batch SGD may itself be interpreted as a form of continuation via noise injection.

[2]A promising approach might be to use multi-task RLS in which models are tasks.

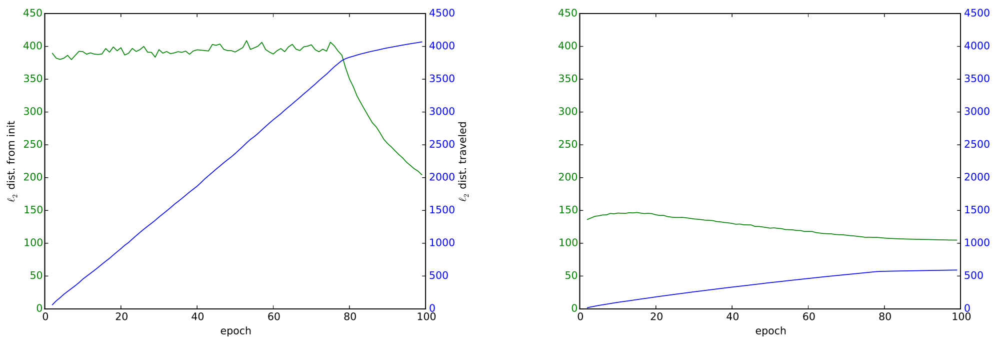

Figure 5: Distances travelled parameters by as training progresses for $m = 50$ and $m = 1000$. The shrinkage towards the origin is due to weight decay.

One method of continuation, introduced in [2], smooths the objective by linearizing the network. Since the only curvature-inducing components in ResNets–and most common neural network architectures–are the softmax layer and rectifier nonlinearities, a plausible hypothesis might be that, by temporarily linearizing the rectifiers, optimization may avoid sharp minima long enough to find a more favorable location in parameter space. Thus, in a manner similar to a Parameterized Rectified Linear Unit (PReLU) [4], the rectifiers in the proposed model are equipped with a parameter that controls the slope of the rectifier in the domain $x < 0$. Unlike PReLU, however, the parameter can not easily be optimized directly. Instead, a simple heuristic is used: if the value of the dominant eigenvector is positive for the parameter, then it is decreased by δ_- or increased by δ_+, otherwise; the value is constrained to lie in $[0, 1]$. While the dominant eigenvector is likely to change significantly during training, the hope is that its general direction will remain the same, which motivates the use of this heuristic.

Experiments in which $\delta_- = 0.1$ and $\delta_+ \in \{0.02, 0.05, 1\}$, showed a limited amount of success in the reduction of the early $\lambda_{\max}$s and the final generalization gap, but these came at the cost of slightly reduced classification accuracy. From a continuation perspective, this may be explained by the sharp minima being smoothed but remaining *relatively sharp* so that maxi-batch SGD continues fall into them when the convexity of the problem is increased.

3 Discussion

The results presented in this work suggest that the performance of SGD is diminished when using maxi-batches due to the prevalence of steep local minima into which the algorithm enters but from which it cannot escape. Furthermore, these local minima, which do not generalize well, also seem to represent parameter settings which are bad in a general sense–this is to say that they offer decidedly lower performance than the best empirically found minimum.

Judging by the failure of i.i.d. Gaussian noise to restore the performance of maxi-batch SGD, the original belief that the efficacy of mini-batches is due solely to increased stochasticity requires further scrutiny. Additionally, from the mostly unimpressive outcomes of increasing the learning rate, one may also conjecture that, in non-convex problems, finding a minimum as quickly as possible is not necessarily the best approach.

This challenging problem deserves further study and a guiding hypothesis might be the following:

> The loss surface of deep neural networks is fraught with minima of varying steepness and quality with those near the origin tending to be more of the former and less of the latter. The high variance of gradient estimates in small-batch SGD allows optimization to bypass these early minima and explore more eccentric parameter settings; the covariance of this noise is integral to its utility. Although maxi-batch SGD finds a minimum quickly, it may well be worth the extra time to explore the parameter space more fully.

4 Conclusion and Future Work

This work has presented the results of several experiments including minimizer interpolation, gradient noise, and adaptive linearzation. Additionally, it has introduced the dominant eigenvalue of the Hessian as a means of quantifying the steepness of minima and an efficient method for computing it.

Continued work will focus primarily on gaining further insight into this problem with the goal of obtaining a theoretical basis and a corresponding principled solution. Several concrete next steps are to determine if injecting the same type of noise present during mini-batch SGD will encourage exploration of the parameter space; understand whether (and, if so, why) minima near the origin are so suboptimal; and attempt to estimate the density of minima so to balance exploration versus exploitation.

Although the empirical results presented herein cannot fully explain the cause of the problem thus posed, they will hopefully provide a basis for developing a greater understanding of neural network optimization and possibly non-convex optimization via gradient methods, in general.

5 Acknowledgements

Thanks to Antonio Torralba and TIG for providing and maintaining the many GPUs required to run the maxi-batch experiments. Also, the use of the $\mathcal{R}$ operator and the intuitions for the loss surface of a neural network would not have been realized without the instructional expertise of the 9.520 staff.

References

[1] Ian J. Goodfellow and Oriol Vinyals. "Qualitatively characterizing neural network optimization problems". In: *CoRR* abs/1412.6544 (2014).

[2] Çaglar Gülçehre et al. "Mollifying Networks". In: *CoRR* abs/1608.04980 (2016).

[3] Moritz Hardt, Benjamin Recht, and Yoram Singer. "Train faster, generalize better: Stability of stochastic gradient descent". In: *CoRR* abs/1509.01240 (2015).

[4] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *CoRR* abs/1502.01852 (2015).

[5] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*. 2016, pp. 630–645.

[6] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015, pp. 448–456.

[7] Nitish Shirish Keskar et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *CoRR* abs/1609.04836 (2016).

[8] Hossein Mobahi. "Training Recurrent Neural Networks by Diffusion". In: *CoRR* abs/1601.04114 (2016).

[9] Arvind Neelakantan et al. "Adding Gradient Noise Improves Learning for Very Deep Networks". In: *CoRR* abs/1511.06807 (2015).

[10] Barak A. Pearlmutter. "Fast Exact Multiplication by the Hessian". In: *Neural Computation* 6.1 (1994), pp. 147–160.