No More Pesky Learning Rates

Tom Schaul **Sixin Zhang** **Yann LeCun**

Courant Institute of Mathematical Sciences
New York University, 715 Broadway, 10003, New York
{schaul,zsx,yann}@cims.nyu.edu

1 Introduction

Large-scale learning problems require algorithms that scale benignly (e.g. sub-linearly) with the size of the dataset and the number of trainable parameters. This has lead to a recent resurgence of interest in *stochastic gradient descent* methods (SGD). Besides fast convergence, SGD has sometimes been observed to yield significantly better generalization errors than batch methods [1]. In practice, getting good performance with SGD requires some manual adjustment of the initial value of the learning rate (or step size) for each model and each problem, as well as the design of an annealing schedule for stationary data. The problem is particularly acute for non-stationary data.

The contribution of this paper is a novel method to *automatically adjust* learning rates (possibly different learning rates for different parameters), so as to minimize some estimate of the expectation of the loss at any one time. The performance of the methods *obtained without any manual tuning* are reported on a variety of convex and non-convex learning models and tasks. They compare favorably with an "ideal SGD", where the best possible learning rate was obtained through systematic search, as well as previous adaptive schemes.

2 Background

SGD methods have a long history in adaptive signal processing, neural networks, and machine learning, with an extensive literature (see [2, 1] for recent reviews). While the practical advantages of SGD for machine learning applications have been known for a long time [3], interest in SGD has increased in recent years due to the ever-increasing amounts of streaming data, to theoretical optimality results for generalization error [4], and to competitions being won by SGD methods, such as the PASCAL Large Scale Learning Challenge [5], where Quasi-Newton approximation of the Hessian was used within SGD. Still, practitioners need to deal with a sensitive hyper-parameter tuning phase to get top performance: each of the PASCAL tasks used very different parameter settings.

Learning rates in SGD are generally decreased according a schedule of the form $\eta(t) = \eta_0(1 + \gamma t)^{-1}$. Originally proposed as $\eta(t) = O(t^{-1})$ in [6], this form was recently analysed in [7, 8] from a non-asymptotic perspective to understand how hyper-parameters like η_0 and γ affect the convergence speed.

A promising first step towards adaptive learning rates was introduced in [9], in an approach called 'ADAGRAD ', where the learning rate takes the form $\eta_i(t) = \frac{\eta_a}{\sqrt{\sum_{s=0}^{t} \nabla_{\theta_i}^{(s)}}}$,for each problem dimension i, where $\nabla_{\theta_i}^{(s)}$ is gradient of the ith parameter at iteration s.

3 Optimal Adaptive Learning Rates

In this section, we derive an optimal learning rate schedule, using an idealized quadratic and separable loss function. We show that using this learning rate schedule preserves convergence guarantees of SGD. In the following section, we find how the optimal learning rate values can be estimated from available information, and describe a couple of possible approximations.

The samples, indexed by j, are drawn i.i.d. from a data distribution $\mathcal{P}$. Each sample contributes a *per-sample loss* $\mathcal{L}^{(j)}(\boldsymbol{\theta})$ to the expected loss:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{j\sim\mathcal{P}}\left[\mathcal{L}^{(j)}(\boldsymbol{\theta})\right] \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the trainable parameter vector, whose optimal value is denoted $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$. The SGD parameter update formula is of the form $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta^{(t)}\nabla_{\boldsymbol{\theta}}^{(j)}$, where $\nabla_{\boldsymbol{\theta}}^{(j)} = \frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{L}^{(j)}(\boldsymbol{\theta})$ is the gradient of the the contribution of example j to the loss, and the learning rate $\eta^{(t)}$ is a suitably chosen sequence of positive scalars (or positive definite matrices).

We assume that the per-sample loss functions are smooth around minima, and can be locally approximated by a quadratic function. We also assume that the minimum value of the per-sample loss functions are zero:

$$\mathcal{L}^{(j)}(\boldsymbol{\theta}) = \frac{1}{2}\left(\boldsymbol{\theta} - \mathbf{c}^{(j)}\right)^\top \mathbf{H}^{(j)}\left(\boldsymbol{\theta} - \mathbf{c}^{(j)}\right) \qquad\qquad \nabla_{\boldsymbol{\theta}}^{(j)} = \mathbf{H}^{(j)}\left(\boldsymbol{\theta} - \mathbf{c}^{(j)}\right)$$

where $\mathbf{H}_i$ is the Hessian matrix of the per-sample loss of sample j, and $\mathbf{c}^{(j)}$ is the optimum for that sample. The distribution of per-sample optima $\mathbf{c}^{(j)}$ has mean $\boldsymbol{\theta}^*$ and variance $\boldsymbol{\Sigma}$.

To simplify the analysis, we assume for the remainder of this section that the Hessians of the per-sample losses are identical for all samples, and that the problem is separable, i.e., the Hessians are diagonal, with diagonal terms denoted $\{h_1, \ldots, h_i, \ldots, h_d\}$. Further, we will ignore the off-diagonal terms of $\boldsymbol{\Sigma}$, and denote the diagonal $\{\sigma_1^2, \ldots, \sigma_i^2, \ldots, \sigma_d^2\}$. Then, for any of the d dimensions, we thus obtain a one-dimensional problem (all indices i omitted).

$$J(\theta) = \mathbb{E}_{i\sim\mathcal{P}}\left[\frac{1}{2}h(\theta - c^{(j)})^2\right] = \frac{1}{2}h\left[(\theta-\theta^*)^2 + \sigma^2\right] \tag{2}$$

The gradient components are $\nabla_\theta^{(j)} = h\left(\theta - c^{(j)}\right)$, with

$$\mathbb{E}[\nabla_\theta] = h(\theta - \theta^*) \qquad \mathbb{V}\,\mathrm{ar}[\nabla_\theta] = h^2\sigma^2 \tag{3}$$

and we can rewrite the SGD update equation as

$$\theta^{(t+1)} \;=\; \theta^{(t)} - \eta h\left(\theta^{(t)} - c^{(j)}\right) = (1-\eta h)\theta^{(t)} + \eta h\theta^* + \eta h\sigma\xi^{(j)} \tag{4}$$

where the $\xi^{(j)}$ are i.i.d. samples from a zero-mean and unit-variance Gaussian distribution. Inserting this into equation 2, we obtain the expected loss after an SGD update

$$\mathbb{E}\left[J\left(\theta^{(t+1)}\right) \mid \theta^{(t)}\right] \;=\; \frac{1}{2}h\cdot\left[(1-\eta h)^2(\theta^{(t)}-\theta^*)^2 + \eta^2h^2\sigma^2 + \sigma^2\right]$$

We can now derive the optimal (greedy) learning rates for the current time t as the value $\eta^*(t)$ that minimizes the expected loss after the next update

$$\eta^*(t) \;=\; \arg\min_\eta\left[(1-\eta h)^2(\theta^{(t)}-\theta^*)^2 + \sigma^2 + \eta^2h^2\sigma^2\right] = \frac{1}{h}\cdot\frac{(\theta^{(t)}-\theta^*)^2}{(\theta^{(t)}-\theta^*)^2+\sigma^2} \tag{5}$$

In the classical (noiseless or batch) derivation of the optimal learning rate, the best value is simply $\eta^*(t) = h^{-1}$. The above formula inserts a corrective term that *reduces* the learning rate whenever the sample pulls the parameter vector in different directions, as measured by the gradient variance σ^2. The reduction of the learning rate is larger near an optimum, when $(\theta^{(t)}-\theta^*)^2$ is small relative to σ^2. In effect, this will reduce the expected error due to the noise in the gradient. Overall, this will have the same effect as the usual method of progressively decreasing the learning rate as we get closer to the optimum, *but it makes this annealing schedule automatic*.

4 Algorithm

In practice, we are not given the quantities σ_i, h_i and $(\theta_i^{(t)} - \theta_i^*)^2$. However, based on equation 3, we can estimate them from the observed samples of the gradient:

$$\eta_i^* = \frac{1}{h_i}\cdot\frac{\left(\mathbb{E}[\nabla_{\theta_i}]\right)^2}{\left(\mathbb{E}[\nabla_{\theta_i}]\right)^2 + \mathbb{V}\,\mathrm{ar}[\nabla_{\theta_i}]} = \frac{1}{h_i}\cdot\frac{\left(\mathbb{E}[\nabla_{\theta_i}]\right)^2}{\mathbb{E}[\nabla_{\theta_i}^2]} \tag{6}$$

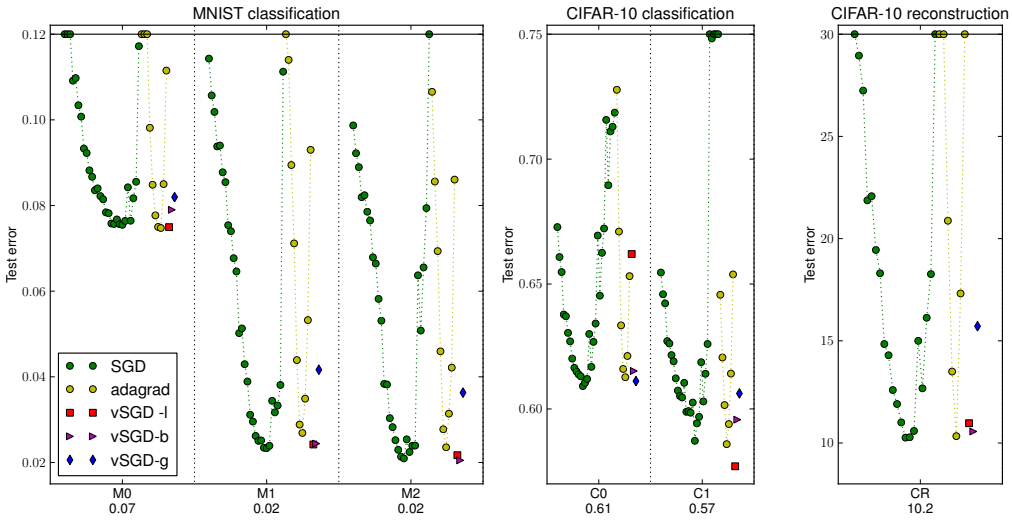

Figure 1: Final performance on test set, on all the practical learning problems (after six epochs) using architectures trained using SGD, ADAGRAD or vSGD. We show all 28 SGD settings (green circles), and 11 ADAGRAD settings (yellow circles), in contrast to tables 1 and 2, where we only compare the best SGD with the vSGD variants. SGD runs (green circles) vary in terms of different learning rate schedules, and the vSGD variants correspond to the local-global approximation choices described in section 4. Symbols touching the top boundary indicate runs that either diverged, or converged too slow to fit on the scale. Note how SGD tuning is sensitive, and the adaptive learning rates are typically competitive with the best-tuned SGD, and sometimes better. See also Figure 4 in the supplementary material for the corresponding figure on the training set.

The optimal learning rate is decomposed into two factors, one term which is the inverse curvature (as is the case for batch second-order methods), and one novel term that depends on the noise in the gradient, relative to the expected squared norm of the gradient. Below, we approximate these terms separately. For the investigations below, when we use the true values instead of a practical algorithm, we speak of the 'oracle' variant.

We use an exponential moving average with time-constant τ (the approximate number of samples considered from recent memory) for online estimates of the quantities in equation 6. We want the size of the memory to increase when the steps taken are small (increment by 1), and to decay quickly if a large step is taken, which is obtained naturally, by the following update

$$\tau_i(t+1) \quad = \quad \left(1 - \frac{\overline{g_i}(t)^2}{\overline{v_i}(t)}\right) \cdot \tau_i(t) \; + \; 1$$

There exist a number of methods for obtaining an online estimates of the diagonal Hessian. We adopt the "bbprop" method, which computes positive estimates of the diagonal Hessian terms for a single sample $h_i^{(j)}$, using a back-propagation formula [3].

The simplest version of the method views each component in isolation. This form of the algorithm will be called "vSGD" (for "variance-based SGD"). In realistic settings with high-dimensional parameter vector, it is not clear a priori whether it is best to have a single, global learning rate (that can be estimated robustly), "vSGD-g", a set of local, dimension-specific rates, "vSGD-l", or block-specific learning rates (whose estimation will be less robust), "vSGD-b".

5 Experiments

We test the new algorithm on two widely used standard datasets to test the different algorithms; the MNIST digit recognition dataset [10], and the CIFAR-10 small natural image dataset [11], both to learn image classification and reconstruction. We use four different architectures (convex and non-convex) of feed-forward neural networks, MLPs with 0,1, or 2 hidden layers for classification, and an autoencoder architecture for regression.

	best-tuned SGD	best-tuned adagrad	vSGD -l	vSGD-b	vSGD-g
CIFAR classif. (C0)	54.78%	54.36%	**45.61%**	52.45%	56.16%
CIFAR classif. (C1)	47.12%	45.20%	**33.16%**	45.14%	54.91%
CIFAR regression (CR)	**9.77**	**9.80**	10.64	10.13	15.37
MNIST classif. (M0)	7.05%	6.97%	**6.72%**	7.63%	8.20%
MNIST classif. (M1)	0.30%	0.58%	**0.18%**	0.78%	3.50%
MNIST classif. (M2)	0.46%	0.41%	**0.05%**	0.33%	2.91%

Table 1: Final classification error (and reconstruction error for CIFAR-2R) on the **training** set, obtained after 6 epochs of training, and averaged over ten random initializations. Variants are marked in bold if they are not statistically significantly from the best one ($p = 0.05$) Note that the both the SGD tuning parameter, and the ADAGRAD tuning parameters are different for each setup. We observe the best results with the full element-wise learning rate adaptation ('vSGD-l'), almost always significantly better than the best-tuned SGD or best-tuned ADAGRAD .

	best-tuned SGD	best-tuned adagrad	vSGD -l	vSGD-b	vSGD-g
CIFAR classif. (C0)	**61.06%**	**61.25%**	66.05%	61.70%	**61.10%**
CIFAR classif. (C1)	58.85%	58.67%	**57.72%**	59.55%	60.62%
CIFAR regression (CR)	**10.29**	**10.33**	11.05	10.57	15.71
MNIST classif. (M0)	**7.60%**	**7.52%**	**7.50%**	7.89%	8.20%
MNIST classif. (M1)	**2.34%**	2.70%	**2.42%**	**2.44%**	4.14%
MNIST classif. (M2)	**2.16%**	2.34%	**2.16%**	**2.05%**	3.65%

Table 2: As Table 1, but on the **test** set. The outcome is more balanced, with vSGD-l being better or statistically equivalent to the best-tuned SGD in 4 out of 6 cases. The main outlier (C0) is a case where the more aggressive element-wise learning rates led to overfitting.

SGD is one of the most common training algorithms in use for (large-scale) neural network training. The experiments in this section compare the three vSGD variants introduced above with SGD and ADAGRAD . We exhaustively search for the best hyper-parameter settings among $\eta_0, \eta_a \in \{10^{-4}, 3*10^{-4}, 10^{-3}, 3*10^{-3}, 10^{-2}, 3*10^{-2}, 10^{-1}, 3*10^{-1}, 10^{0}, 3*10^{0}, 10^{1}\}$, $\gamma \in \{0, 1/3, 1/2, 1\}$/#traindata, as determined by their lowest test error.

Figure 1 illustrates the sensitivity to hyper-parameters of both SGD and ADAGRAD , while very similar performance is achieved using the tuning-free vSGD variants. For each benchmark, ten independent runs are averaged and reported in Table 1 (training set) and Table 2 (test set). They show that the best vSGD variant, across the board, is vSGD-l, which most aggressively adapts one learning rate per dimension. It is almost always significantly better than the best-tuned SGD or best-tuned ADAGRAD in the training set, and better or statistically equivalent to the best-tuned SGD in 4 out of 6 cases on the test set.

6 Conclusion

Starting from the idealized case of quadratic loss contributions from each sample, we derived a method to compute an optimal learning rate at each update, and (possibly) for each parameter, that optimizes the expected loss after the next update. The method relies on the square norm of the expectation of the gradient, and the expectation of the square norm of the gradient. We showed different ways of approximating those learning rates in linear time and space in practice. The experimental results confirm the theoretical prediction: the adaptive learning rate method completely eliminates the need for manual tuning of the learning rate, or for systematic search of its best value.

Given the successful validation on a variety of classical large-scale learning problems, we hope that this enables for SGD to be a truly user-friendly 'out-of-the-box' method.

References

[1] Bottou, L and Bousquet, O. The tradeoffs of large scale learning. In Sra, S, Nowozin, S, and Wright, S. J, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, 2011.

[2] Bottou, L. Online algorithms and stochastic approximations. In Saad, D, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

[3] LeCun, Y, Bottou, L, Orr, G, and Muller, K. Efficient backprop. In Orr, G and K., M, editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[4] Bottou, L and LeCun, Y. Large scale online learning. In Thrun, S, Saul, L, and Schölkopf, B, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[5] Bordes, A, Bottou, L, and Gallinari, P. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, July 2009.

[6] Robbins, H and Monro, S. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[7] Xu, W. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *ArXiv-CoRR*, abs/1107.2490, 2011.

[8] Bach, F and Moulines, E. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[9] Duchi, J. C, Hazan, E, and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. 2010.

[10] LeCun, Y and Cortes, C. The mnist dataset of handwritten digits. 1998. http://yann.lecun.com/exdb/mnist/.

[11] Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009.