No More Pesky Learning Rates: Supplementary Material

Tom Schaul SCHAUL@CIMS.NYU.EDU
Sixin Zhang ZSX@CIMS.NYU.EDU
Yann LeCun YANN@CIMS.NYU.EDU
Courant Institute of Mathematical Sciences
New York University
715 Broadway, New York, NY 10003, USA

A. Convergence Proof

If we do gradient descent with $\eta^*(t)$, then almost surely, the algorithm converges (for the quadratic model). To prove that, we follow classical techniques based on Lyapunov stability theory (Bucy, 1965). Notice that the expected loss follows

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

Thus $J(\theta^{(t)})$ is a positive super-martingale, indicating that almost surely $J(\theta^{(t)}) \to J^\infty$. We are to prove that almost surely $J^\infty = J(\theta^*) = \frac{1}{2}h\sigma^2$. Observe that

$$\begin{aligned}
J(\theta^{(t)}) - \mathbb{E}[J(\theta^{(t+1)}) \mid \theta^{(t)}] &= \frac{1}{2}h\eta^*(t)\,, \\
\mathbb{E}[J(\theta^{(t)})] - \mathbb{E}[J(\theta^{(t+1)}) \mid \theta^{(t)}] &= \frac{1}{2}h\mathbb{E}[\eta^*(t)]
\end{aligned}$$

Since $\mathbb{E}[J(\theta^{(t)})]$ is bounded below by 0, the telescoping sum gives us $\mathbb{E}[\eta^*(t)] \to 0$, which in turn implies that in probability $\eta^*(t) \to 0$. We can rewrite this as

$$\eta^*(t) = \frac{J(\theta_t) - \frac{1}{2}h\sigma^2}{J(\theta_t)} \to 0$$

By uniqueness of the limit, almost surely, $\frac{J^\infty - \frac{1}{2}h\sigma^2}{J^\infty} = 0$. Given that J is strictly positive everywhere, we conclude that $J^\infty = \frac{1}{2}h\sigma^2$ almost surely, i.e $J(\theta^{(t)}) \to \frac{1}{2}h\sigma^2 = J(\theta^*)$.

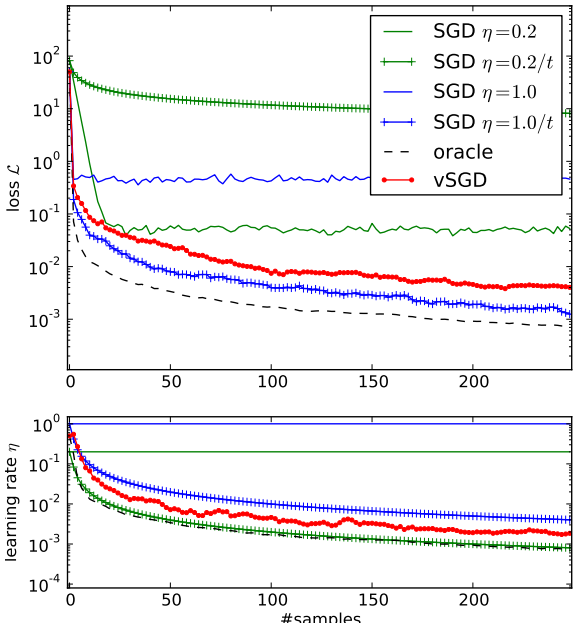

Figure 1. Optimizing a noisy quadratic loss (dimension $d = 1$, curvature $h = 1$). Comparison between SGD for two different fixed learning rates 1.0 and 0.2, and two cooling schedules $\eta = 1/t$ and $\eta = 0.2/t$, and vSGD (red circles). In dashed black, the 'oracle' computes the true optimal learning rate rather than approximating it. In the top subplot, we show the median loss from 1000 simulated runs, and below are corresponding learning rates. We observe that vSGD initially descends as fast as the SGD with the largest fixed learning rate, but then quickly reduces the learning rate which dampens the oscillations and permits a continual reduction in loss, beyond what any fixed learning rate could achieve. The best cooling schedule ($\eta = 1/t$) outperforms vSGD, but when the schedule is not well tuned ($\eta = 0.2/t$), the effect on the loss is catastrophic, even though the produced learning rates are very close to the oracle's (see the overlapping green crosses and the dashed black line at the bottom).

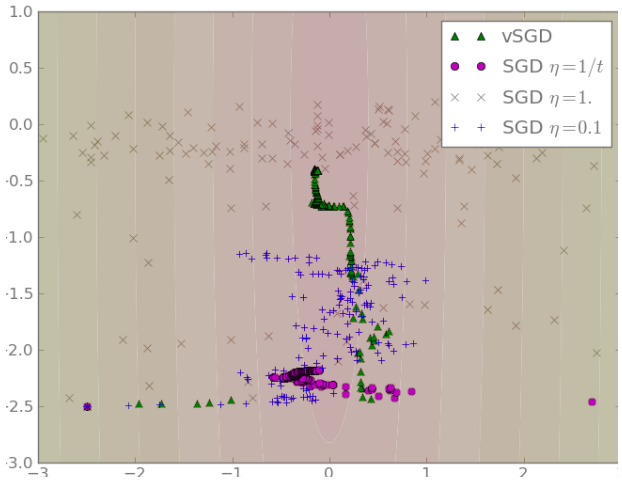

Figure 2. Illustration of the dynamics in a noisy quadratic bowl (with 10 times larger curvature in one dimension than the other). Trajectories of 400 steps from vSGD, and from SGD with three different learning rate schedules. SGD with fixed learning rate (crosses) descends until a certain depth (that depends on η) and then oscillates. SGD with a $1/t$ cooling schedule (pink circles) converges prematurely. On the other hand, vSGD (green triangles) is much less disrupted by the noise and continually approaches the optimum.

B. Derivation of Global Learning Rate

We can derive an optimal global learning rate η_g^* as follows.

$$\begin{aligned}
\eta_g^*(t) &= \arg\min_\eta \mathbb{E}\left[\mathcal{J}\left(\boldsymbol{\theta}^{(t+1)}\right) \mid \boldsymbol{\theta}^{(t)}\right] \\
&= \arg\min_\eta \sum_{i=1}^{d} h_i \Big((1-\eta h_i)^2(\theta_i^{(t)}-\theta_i^*)^2 \\
&\qquad\qquad +\sigma_i^2+\eta^2 h_i^2\sigma_i^2\Big) \\
&= \arg\min_\eta \Bigg[\eta^2 \sum_{i=1}^{d}\left(h_i^3(\theta_i^{(t)}-\theta_i^*)^2+h_i^3\sigma_i^2\right) \\
&\qquad\qquad -2\eta\sum_{i=1}^{d} h_i^2(\theta_i^{(t)}-\theta_i^*)^2\Bigg]
\end{aligned}$$

which gives

$$\eta_g^*(t) = \frac{\sum_{i=1}^d h_i^2(\theta_i^{(t)}-\theta_i^*)^2}{\sum_{i=1}^d\left(h_i^3(\theta_i^{(t)}-\theta_i^*)^2+h_i^3\sigma_i^2\right)}$$

The adaptive time-constant for the global case is:

$$\tau_g(t+1) = \left(1-\frac{\sum_{i=1}^d \overline{g_i}^2}{\bar{l}(t)}\right)\cdot\tau_g(t)\ +\ 1$$

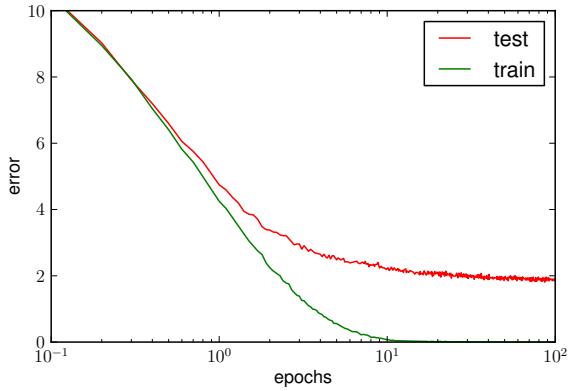

Figure 3. Learning curves for full-length runs of 100 episodes, using vSGD-l on the M1 benchmark with 800 hidden units. Test error is shown in red, training error is green. Note the logarithmic scale of the horizontal axis. The average test error after 100 epochs is 1.87%.

C. SMD Implementation

The details of our implementation of SMD (based on a global learning rates) are given by the following updates:

$$\begin{aligned}
\boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t - \eta_t\nabla_{\boldsymbol{\theta}} \\
\eta_{t+1} &\leftarrow \eta_t \exp\left(-\mu\nabla_{\boldsymbol{\theta}}^\top \mathbf{v}_t\right) \\
\mathbf{v}_{t+1} &\leftarrow (1-\tau^{-1})\mathbf{v}_t - \eta_t\left(\nabla_{\boldsymbol{\theta}} + (1-\tau^{-1})\cdot\mathbf{H}_t\mathbf{v}_t\right)
\end{aligned}$$

where $\mathbf{Hv}$ denotes the Hessian-vector product with vector $\mathbf{v}$, which can be computed in linear time. The three hyper-parameters used are the initial learning rate η_0, the meta-learning rate μ, and a time constant τ for updating the auxiliary vector $\mathbf{v}$.

D. Additional Results and Analysis

Figures 1 through 8 complement the results from the main paper. First, for an intuitive understanding of the effects of the optimal adaptive learning rate method, and the effect of the approximation, we illustrate the oscillatory behavior of SGD (Figure 2), and compare the decrease in the loss function and the accompanying change in learning rates on the noisy quadratic loss function (Equation 2 of the main paper): Figure 1 shows learning curves, contrasting the effect of fixed rates or fixed schedules to adaptive learning rates, whether in approximation or using the oracle. Complementing the MNIST experiments in the main paper, Figure 3 shows the learning curve on the M1 setup over 100 epochs, much longer than the remainder of the experiments. Further, Figure 4 visualizes test and training error for different algorithm settings on the three CIFAR benchmarks, and Figure 5 visualizes test and training error for all algorithms and their

settings on all benchmarks (zoomed out from the region of interest to give different perspective). Figure 6 shows the evolution of (minimal/maximal) learning rates over time, emphasizing the effects of slow-start initialization in our approach. Figure 7 and 8 show the effect of tuning for all benchmarks and a subset of the algorithms.

E. Sensitivity to Initialization

Figure 10 shows that the initialization parameter C does not affect performance, so long as it is sufficiently large. This is not surprising, because its only effect is to slow down the initial step sizes until accurate exponential averages of the interesting quantities can be computed.

There is a *critical* minimum value of C, blow which the algorithm is unstable. Figure 9 shows what those critical values are for 13 different setups with widely varying problem dimension. From these empirical results, we derive our rule-of-thumb choice of $C = d/10$ as a 'safe' pick for the constant (in fact it is even a factor 10 larger than the observed critical value for any of the benchmarks, just to be extra careful).

References

Bucy, R. S. Stability and positive supermartingales. *Journal of Differential Equations*, 1(2):151–155, 1965.

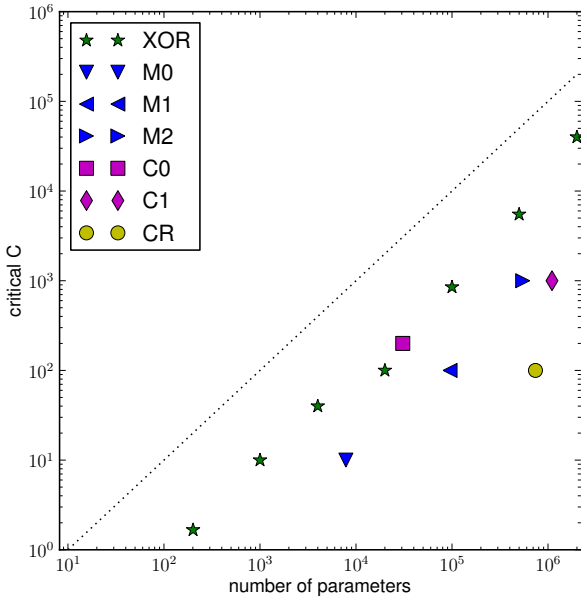

Figure 9. Critical values for initialization parameter C. This plot shows the values of C below which vSGD-l becomes unstable (too large initial steps). We determine the critical C value as the largest for which at least 10% of the runs give rise to instability. The markers correspond to experiments with setups on a broad range of parameter dimensions. Six markers correspond to the benchmark setups from the main paper, and the green stars correspond to simple the XOR-classification task with an MLP of a single hidden layer, the size of which is varied from 2 to 500000 neurons. The black dotted diagonal line indicates, our 'safe' heuristic choice of $C = d/10$.

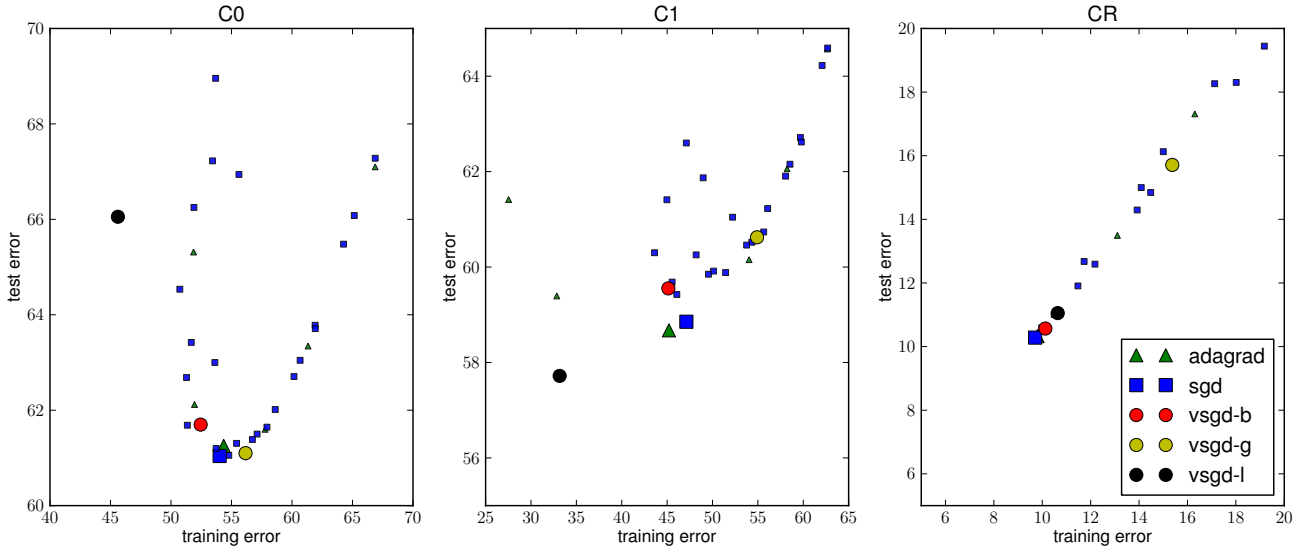

Figure 4. Training error versus test error on the three CIFAR setups (after 6 epochs). Different symbol-color combinations correspond to different algorithms, with the best-tuned parameter setting shown as a much larger symbol than the other settings tried. The axes are zoomed to the regions of interest for clarity. Note how there is much more overfitting here than for MNIST, even with vanilla SGD.

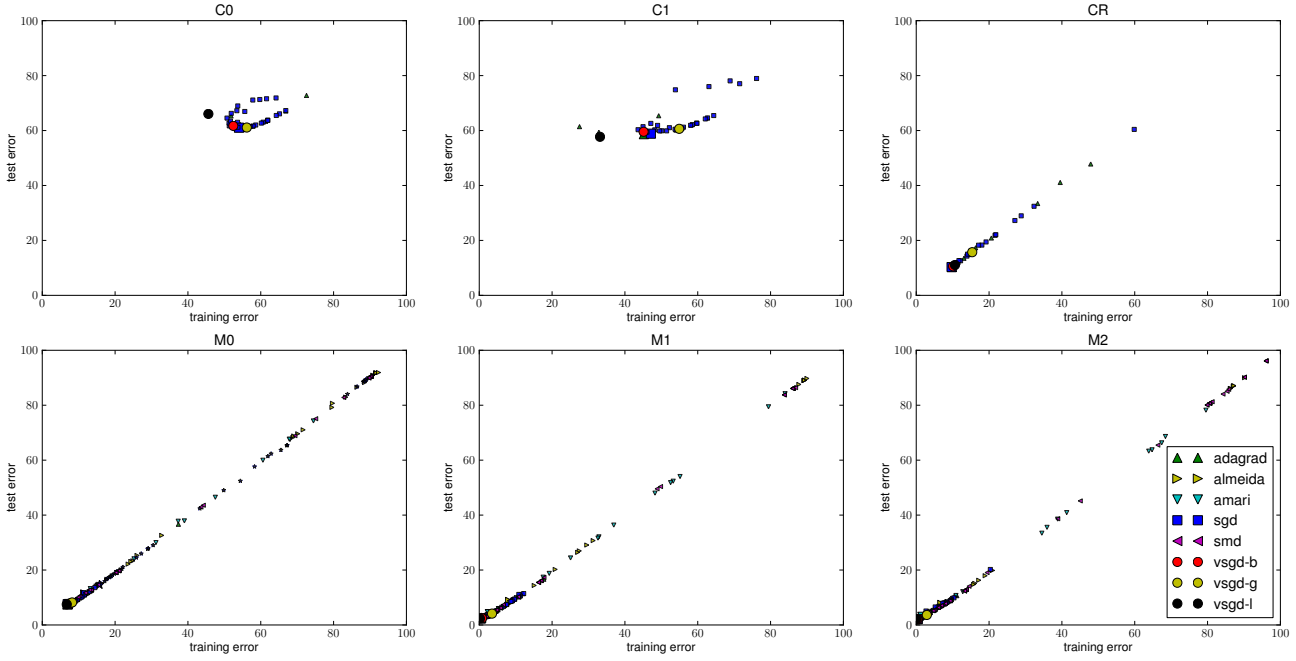

Figure 5. Training error versus test error on all 6 setups, global perspective. Different symbol-color combinations correspond to different algorithms, with the best-tuned parameter setting shown as a much larger symbol than the other settings tried.

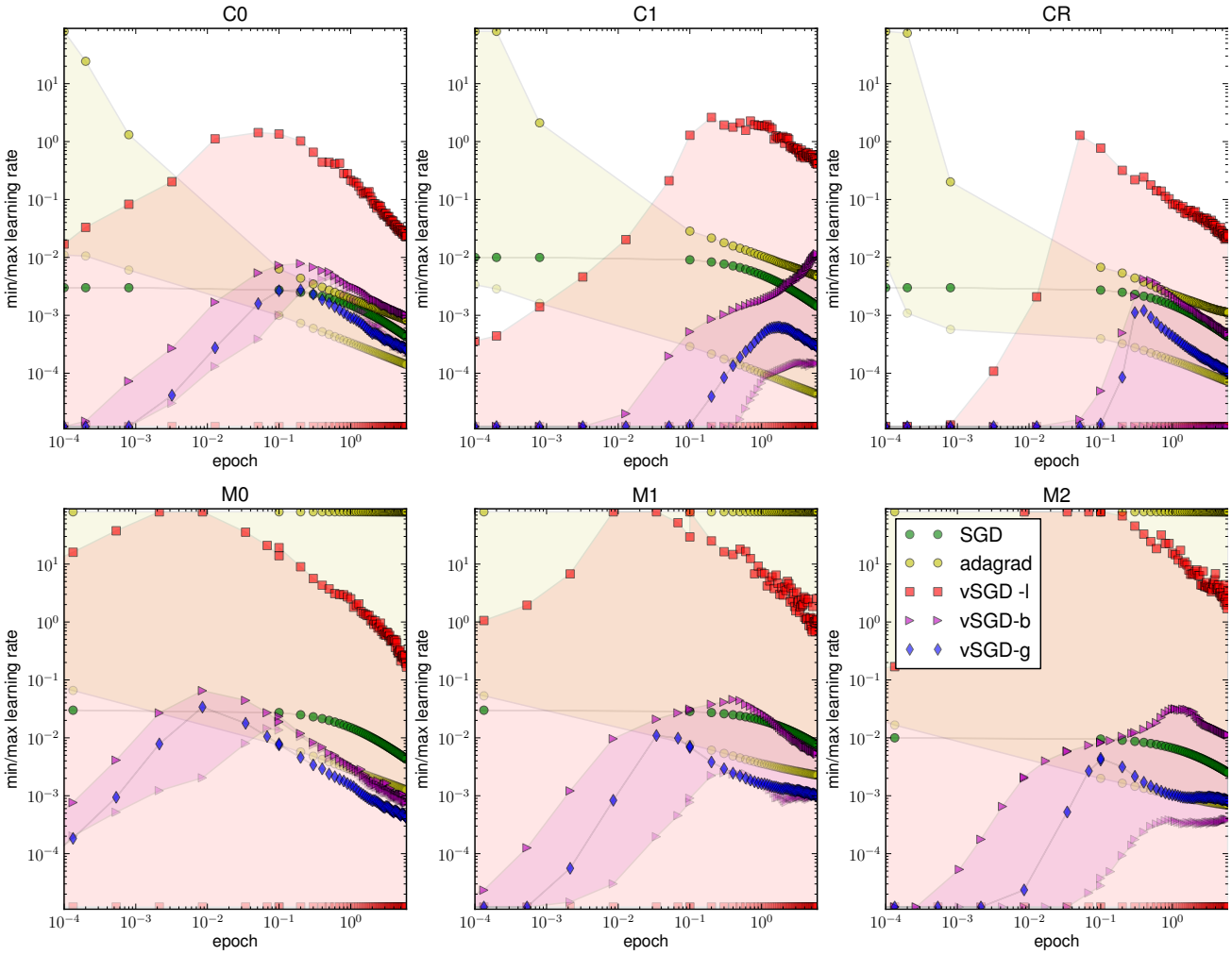

Figure 6. Evolution of learning rates. It shows how the learning rates (minimum and maximum across all dimensions) vary as a function of the epoch. Left: CIFAR classification (no hidden layer), right: MNIST classification (no hidden layer). Each symbol/color corresponds to the median behavior of one algorithm. The range of learning rates (for those algorithms that don't have a single global learning rate) is shown in a colored band in-between the min/max markers. The log-log plot highlights the initial behavior, namely the 'slow start' (until about 0.1 epochs) due to a large C constant in out methods, which contrasts with the quick start of AdaGrad . We also note that AdaGrad (yellow circles) has drastically different ranges of learning rates on the two benchmarks.

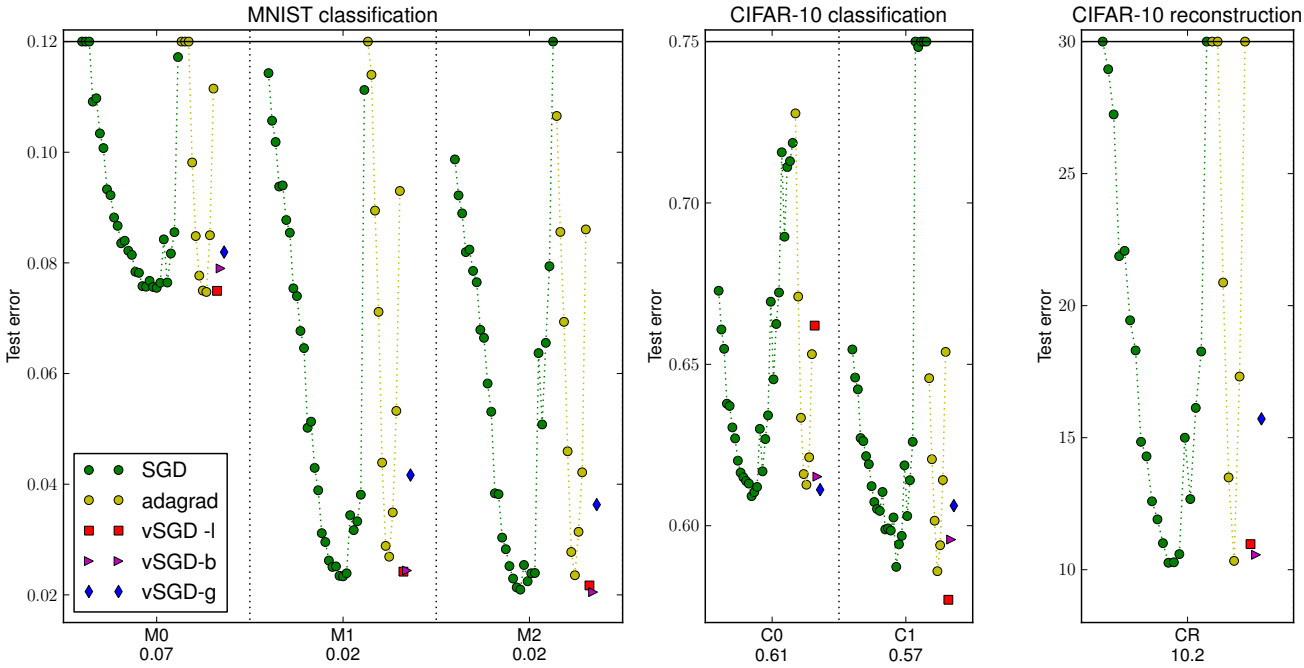

Figure 7. Final performance on test set, on all the practical learning problems (after six epochs) using architectures trained using SGD, AdaGrad or vSGD. We show all 28 SGD settings (green circles), and 11 AdaGrad settings (yellow circles), in contrast to tables 2 and 3, where we only compare the best SGD with the vSGD variants. See also Figure 8 for the corresponding performance on the training set. 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 5. 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.

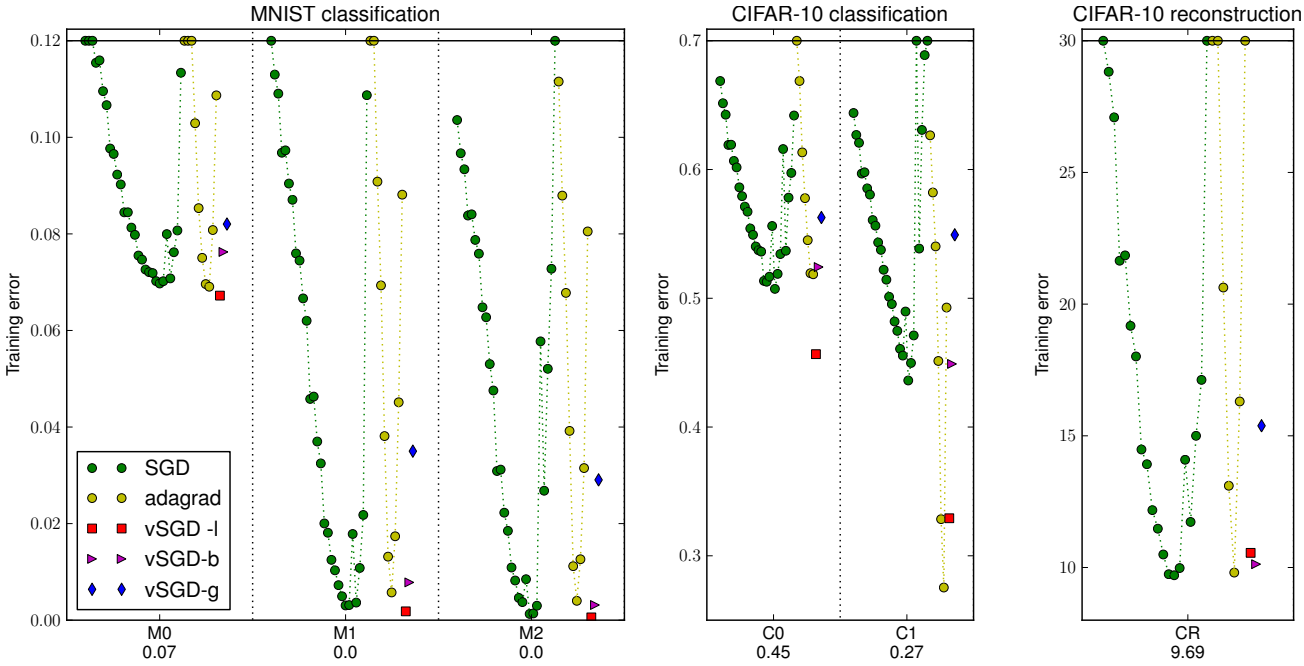

Figure 8. Final performance on training set, on all the practical learning problems (after six epochs) using architectures trained using SGD, AdaGrad or vSGD. We show all 28 SGD settings (green circles), and 11 adaSGD settings (yellow circles). We can observe some clear overfitting in the M2, M3 and C2 cases.

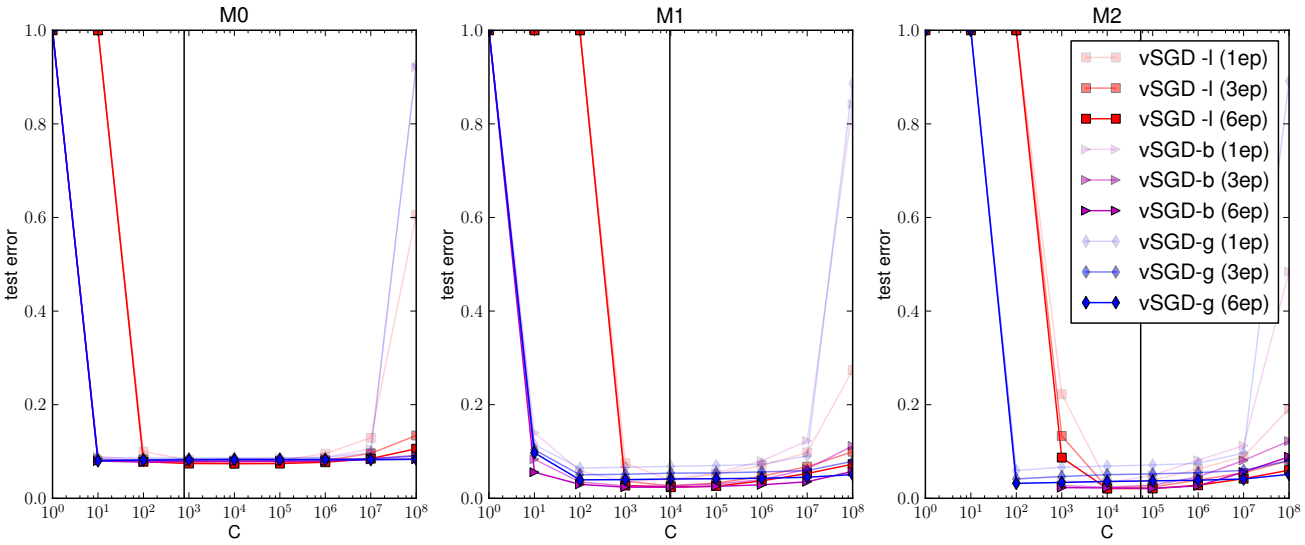

Figure 10. Parameter study on hyper-parameter C. These plots demonstrate that the algorithm is insensitive to the choice of initial slowness parameter C. For each of the setups on the MNIST classification benchmark (with vastly differing parameter dimension d, see Table 1 in the main paper, we show the sensitivity of the test set performance as we vary C over 8 orders of magnitude. Each plot shows the test errors after 1, 3 and 6 epochs (different levels of transparency), for the three adaptive variants (l, b, g, in different colors). In all cases, we find that the updates are unstable if C is chosen too small (the element-wise 'l' variant being most affected), but otherwise C has very little effect, up until when it becomes extremely large: for $C = 10^8$, this initialization basically neutralizes the whole first epoch, and is still felt at epoch 6. The black vertical line indicates, for the three setups, our 'safe' heuristic choice of $C = d/10$. Its only purpose is to avoid instability upon initialization, and given that its 'sweet spot' spans many orders of magnitude, it does not need to be tuned more precisely.