Understanding Short-Horizon Bias in Stochastic Meta-Optimization

Yuhuai Wu[*1,2], **Mengye Ren**[*1,2,3], **Renjie Liao**[1,2,3], **Roger B. Grosse**[1,2]

[1]University of Toronto
[2]Vector Institute for Artificial Intelligence
[3]Uber Advanced Technologies Group
{ywu, mren, rjliao, rgrosse}@cs.toronto.edu

Abstract

Careful tuning of the learning rate and its schedule can be crucial to effective neural net training. There has been much recent interest in gradient-based meta-optimization, where one tunes hyperparameters, or even learns an optimizer, in order to minimize the expected loss when the training procedure is unrolled. But because the training procedure must be unrolled thousands of times, the meta-objective must be defined with an orders-of-magnitude shorter time horizon than is typical for neural net training. We show that such short-horizon meta-objectives cause a serious bias towards small step sizes, an effect we term short-horizon bias. We believe short-horizon bias is a fundamental problem that needs to be addressed if meta-optimization is to scale to practical neural net training regimes.

1 Introduction

The learning rate is one of the most important and frustrating hyperparameters to tune in deep learning. While a fixed learning rate often works well for simpler problems, good performance on the ImageNet [13] benchmark requires a carefully tuned schedule. A variety of decay schedules have been proposed for different architectures, including polynomial (YOLO) [12], exponential (GoogLeNet) [18], staircase (VGG, ResNet) [16, 4]. A related hyperparameter is momentum; typically fixed to a reasonable value such as 0.9, careful tuning can also give significant performance gains [17]. Because of this, it is not surprising that there have been many attempts to adapt learning rates, either online during optimization [15], [14], by learning a policy offline corresponding to a learning rate schedule [10], or even with an optimizer [1], [8], [3]. All of these approaches are forms of meta-optimization, where one defines a meta-objective, the expected loss after some number of optimization steps, and tunes the hyperparameters to minimize this meta-objective. But because the meta-optimization is a lot more expensive than the base-level optimization. the meta-objective must be defined with a much smaller time horizon (e.g. hundreds of updates) than we are ordinarily interested in for large-scale optimization. The hope is that the learned hyperparameters or optimizer will generalize well to much longer time horizons. However, we show that this is not achieved because of a strong tradeoff between short-term and long-term performance, which we refer to as *short-horizon bias*.

To demonstrate and understand the short-horizon bias, we first analyze a noisy quadratic cost function based on [14]. In our noisy quadratic problem, the dynamics of SGD with momentum can be analyzed exactly, allowing us to derive the 1-step-horizon-optimal learning rate and momentum in closed form, as well as to fit the long-horizon-optimal schedule using gradient descent. We

[*]Equal contribution.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

analyze the differences between the short-horizon and long-horizon schedules. Because short-horizon performance is dominated by high-curvature directions, the short-horizon optimizer neglects low-curvature directions, leading to poor long-term performance. Next, we consider measuring short-horizon bias in a practical setting. In particular, we consider an offline algorithm which fits a learning rate decay schedule by running optimization many times from scratch. We show that short-horizon meta-optimizers cause base-level optimization progress to slow to a crawl, even with moderately long time horizons (e.g. 100 or 1000 steps) similar to those used in prior work on gradient-based meta-optimization .

In short, we expect that any meta-objective which doesn't correct for short-horizon bias will probably fail dramatically when run for a much longer time horizon than it was trained on. There are applications where short-horizon meta-optimization is directly useful, such as few-shot learning [6]. In those settings, short-horizon bias is by definition not an issue. But much of the appeal of meta-optimization comes from the possibility of using it to speed up or simplify the training of large neural networks. In these settings, short-horizon bias is a fundamental obstacle that must be addressed for meta-optimization to be practically useful.

2 Noisy quadratic problem

In this section, we develop a toy problem which demonstrates the short-horizon bias, but that we can analyze analytically. In particular, we borrow the noisy quadratic model of [14]; the true function being optimized is a quadratic, but in each iteration, we observe a "noisy" version with perturbed center but exact curvature. We analyze the dynamics of SGD with momentum on this example, determine both the optimal and short-horizon-optimal learning rate schedules.

Analysis The problem is to find the optimal learning rate and momentum for optimizing a noisy quadratic problem. For every quadratic, one can choose a coordinate system such that the Hessian is diagonal. We assume the gradient noise variance to be co-diagonalizable with Hessians. The loss is written as,

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{c}\sim\mathcal{P}}[\sum_i \frac{1}{2} h_i(\theta_i - c_i)^2] = \frac{1}{2}\sum_i h_i((\theta_i - \theta_i^*)^2 + \sigma_i^2)$$

where the expectation is taken over a data distribution $\mathcal{P}$, and θ_i, c_i are the i^{th} coordinate of the parameter $\boldsymbol{\theta}$ and the sample $\mathbf{c}$, and $\mathbb{E}_{\mathbf{c}\sim\mathcal{P}}[c_i] = \theta_i^*$, $\mathbb{V}[c_i] = \sigma_i^2$. The stochastic gradient is written as $\nabla_{\theta_i}\mathcal{L} = h(\theta_i - c_i)$. WLOG, we assume $\theta^* = 0$, so the stochastic gradient becomes,

$$\nabla_{\theta_i}\mathcal{L} = h_i\theta_i - h_i\sigma_i\xi, \quad \xi \sim \mathcal{N}(0,1)$$

The expected loss at timestep t is,

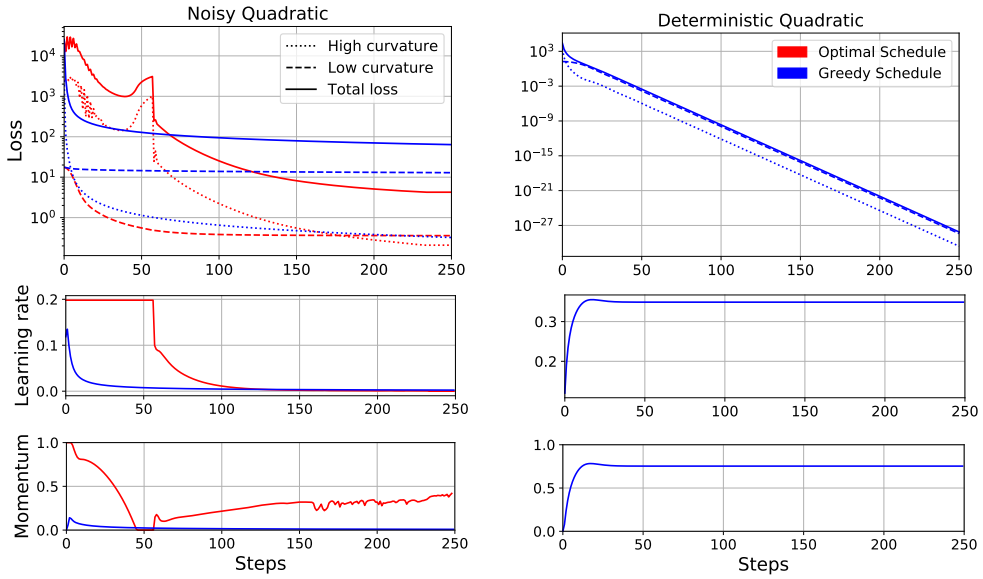
$$\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \sum_i \frac{1}{2} h_i((\theta_i^{(t)})^2 + (\sigma_i^{(t)})^2) = \sum_i \frac{1}{2} h_i(\mathbb{E}[\theta_i^{(t)}]^2 + \mathbb{V}[\theta_i^{(t)}] + (\sigma_i^{(t)})^2) \tag{1}$$

Greedy Optimal Learning Rate and Momentum We consider the setting where the curvature already accounts for preconditioners, and hence the learning rate and momentum are shared across all dimensions. We derive the dynamics of the SGD with momentum (in Appendix), from which we can then solve for the optimal learning rate $\alpha^{(t)*}$ and momentum $\mu^{(t)*}$ that optimizes the loss at timestep $t+1$, $\mathcal{L}(\boldsymbol{\theta}^{(t)})$. We call this as the *greedy* optimal learning rate and momentum.

Theorem 1 (Greedy optimal learning rate and momentum)**.** *The global optimal learning rate and momentum is given by,*

$$\alpha^{(t)*} = \frac{\sum_i \Big(h_i^2 A(\theta_i^{(t)})\Big(\sum_j h_j A(v_j^{(t)})\Big) - \Big(\sum_j h_j \mathbb{E}[\theta_j^{(t)} v_j^{(t)}]\Big) h_i^2 \mathbb{E}[\theta_i^{(t)} v_i^{(t)}]\Big)}{\sum_i \Big(\Big(h_i^3(A(\theta_i^{(t)}) + (\sigma_i)^2)\Big)\Big(\sum_j h_j A(v_j^{(t)})\Big) - \Big(\sum_j (h_j)^2 \mathbb{E}[\theta_j^{(t)} v_j^{(t)}]\Big) h_i^2 \mathbb{E}[\theta_i^{(t)} v_i^{(t)}]\Big)}$$

$$\mu^{(t)*} = -\frac{\sum_i h_i(1 - \alpha^{(t)*} h_i)\mathbb{E}[\theta_i^{(t)} v_i^{(t)}]}{\sum_i h_i A(v_i^{(t)})}$$



(a) (b)

Figure 1: Comparisons of the optimal learning rate and momentum trained by gradient descent and greedy learning rate and momentum in both noisy (a) and deterministic (b) quadratic settings. In the deterministic case, our optimized schedule matched the greedy one, just as the theory predicts.

Analytically solving for the optimal learning rate and momentum for greater than 1-step look ahead is infeasible. However, since we derived the dynamics of how , we can numerically solve the problem of T-steps lookahead with Autodiff softwares and do gradient descent on the loss at timestep T with respect to the learning rate and momentum.

Experiments We compare the results we obtain if we do one-step lookahead vs. T-steps lookahead, where $T = 250$. We chose a 1000 dimensional quadratic with the curvature distribution adopted from [9], that which CG achieves the worst convergent rate. Since the Fisher matrix is an approximation to the Hessian matrix, we assume that $h_i = \mathbb{V}[\nabla_{\theta_i} \mathcal{L}]$, and hence $\sigma_i = \frac{1}{\sqrt{h_i}}$. The results and the corresponding learning rate and momentum are shown in Figure 1 (a). The optimal total loss decreases to a much lower value (4.25) than the loss obtained by greedy schedule (63.86).

In addition to the total loss, we also look at the sum of the losses on the 50 highest curvature directions and 50 lowest curvature directions separately. We find that by following a greedy schedule, the learning rate and momentum became small very early on, which made the loss on the high curvature directions decrease instantly, along with the total loss. However, we notice that the learning rate was too small to make any progress on low curvature directions. On the other hand, the optimal schedule kept at a high learning rate and momentum from the beginning, so the loss on the low curvature directions decreases significantly. Later on as the learning rate decreased, the loss on the high curvature directions also decreased, hence it converged to a much higher loss in the end. As we will also see later in the real datasets, short horizon objective will often encourage the learning rate and momentum to be conservative, so as to achieve largest gain in short term, but performing poorly in the long term.

On the other hand, we find the short term horizon bias is the result of stochasticity in the objective. As observed by [11], the greedy learning rate is the optimal learning rate schedule in the deterministic case. We also validated it by setting the noise σ_i to zero and run greedy schedule compared to the schedule found by gradient descent. Both curves aligned, shown in Figure 1 (b). The noise in the problem adds uncertainty to the objective, resulting in failures of greedy schedule.

3 Gradient-based meta-optimization

We now turn our attention to gradient-based hyperparameter optimization. A variety of approaches have been proposed which tune hyperparameters by doing gradient descent on a meta-objective [15, 10, 1]. We empirically analyze an idealized meta-optimization algorithm which is simplified

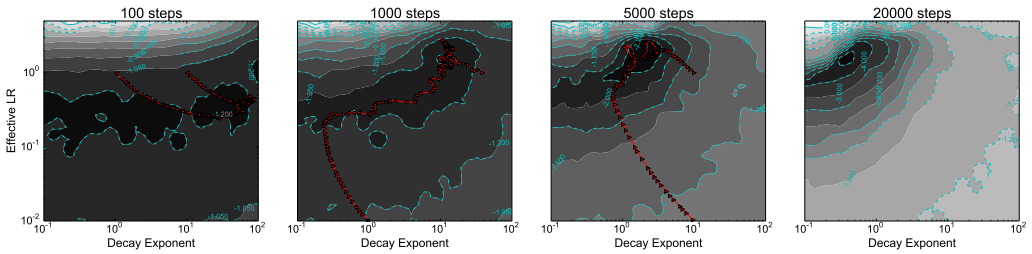


Figure 2: SMD trajectories optimizing initial effective learning rate and decay exponent by looking ahead {100, 1k, 5k, 20k} steps. 2.5k random samples with Gaussian interpolation per plots are used to illustrate the loss surface. Loss values displayed in logarithmic scale. The training losses obtrained using the selected hyperparameters are $\{4.8 \times 10^{-2}, 1.3 \times 10^{-1}, 1.3 \times 10^{-3}, 4.0 \times 10^{-5}\}$.

in two ways. First, we allow the meta-optimizer more computation than would be economical in practice. Second, we limit the representational power of our meta-model: and we focus on the much simpler problem of adapting learning rate and momentum hyperparameters, or schedules thereof. The aim of these two simplifications is that we would like to do a good enough job of optimizing the meta-objective that any failures arise from deficiencies in the meta-objective itself (i.e. short-horizon bias) rather than incomplete meta-optimization. However, we believe our findings are applicable to the more general setting without these two simplifications. We don't see any reason why cheaper meta-optimization methods would address the shortcomings of the meta-objective.

We design an offline experiment on a multi-layered perceptron (MLP) on MNIST [7]. Since the dynamics of training are different at the very start compared to later stages, we pretrain the network first with manually tuned learning rate and momentum. We obtain gradients to the hyperparameters by using forward-mode automatic differentiation [2] on the training iterations, which allows us to get the exact gradients over thousands of training steps to find out hyperparameters for both short and long-horizon.

Learnable decay schedule. We include a parameterized learning rate decay schedule, on top of the SGD with momentum updates. A standard decay schedule is the inverse time decay [19]: $\alpha_t = \frac{\alpha_0}{(1+\frac{t}{K})^\beta}$, where t is the number of training steps, β is the learning rate decay exponent, and K is the time constant. In the SMD procedure, we optimize β together with α and μ.

Experimental details. The network layer size is [784-100-100-10], with ReLU activations. network is pretrained with 1k SGD with momentum steps, with $\alpha = 0.1, \mu = 0.9, \beta = 0$. We train the all hyper-parameters on log space using Adam optimizer [5] for 2k steps. After one meta-step, we restore the weights back to its pretrained initialization. For meta-optimization purpose, we re-parameterized the momentum with $(1 - \mu)$, the learning rate with the effective learning rate $\alpha_{\text{eff}} = \frac{\alpha}{1-\mu}$.

Figure 2 plots SMD optimization trajectories on the loss surface, which is obtained by using random samples of hyperparameters. Multiple trajectories were run to ensure the local minima match with the ones plotted on the surface. The final hyperparameter values are not influenced much by their initializations. Importantly, looking ahead for longer steps ends up with larger initial learning rate α, and much smaller learning rate decay exponent β. The loss surface exhibit distinct landscapes, and the final chosen β differs by two orders of magnitude between 100 and 20k steps. If we use the optimal hyperparameters for 100 steps, the resulting training loss at 20k steps is three orders of magnitude larger than 20k steps.

4 Conclusion

In this paper we look at the short-horizon bias in the meta-optimization objective. We find short horizon objective will often encourage the learning rate and momentum to be conservative, so as to achieve largest gain in short term, but performing poorly in the long term. We demonstrated this effect on a noisy quadratic problem as well as in a practical setting. We believe short-horizon bias is a fundamental problem that needs to be addressed if meta-optimization is to scale to practical neural net training regimes.

References

[1] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3981–3989, 2016.

[2] A. G. Baydin, B. A. Pearlmutter, and A. A. Radul. Automatic differentiation in machine learning: a survey. *CoRR*, abs/1502.05767, 2015.

[3] S. L. Chelsea Finn, Pieter Abbeel. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3th International Conference on Learning Representations*, 2015.

[6] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learningof simple visual concepts. In *In Proceedings of the 33th Annual Meeting of the Cognitive Science Society*, 2011.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[8] K. Li and J. Malik. Learning to optimize. In *5th International Conference on Learning Representations*, 2017.

[9] R. Li. Sharpness in rates of convergence for CG and symmetric lanczos methods. Technical report, 2005.

[10] D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning*, July 2015.

[11] J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *ICML*, 2015.

[12] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Hawaii, HI, USA, June 21-26, 2017*, 2017.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[14] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 343–351, 2013.

[15] N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 569–574 vol.2, 1999.

[16] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3th International Conference on Learning Representations*, 2015.

[17] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147. JMLR.org, 2013.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[19] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688, 2011.