Algorithms for Learning Good Step Sizes

Brian Zhang (bhz) and Manikant Tiwari (manikant)
with the guidance of Prof. Tim Roughgarden

I. Motivation and Previous Work

Many common algorithms in machine learning, from logistic regression to neural networks and beyond, rely on hyperparameters to function well. In gradient descent, too large a step size can cause the iteration to blow up and not terminate; too small a step size and the iteration doesn't converge fast enough. Tuning these parameters can be very difficult: the manual for the popular linear and integer programming solver CPLEX has a 200-page parameter manual that covers 135 parameters, and advises simply that "you may need to experiment with them" [1].

Existing methods for parameter tuning include just hand-tuning, a brute-force or random search, and grid search on the full set of training data, none of which are especially satisfying. For example, [2] gives a comparison between random and grid search on various problems, concluding that random search is better in many cases. [3] and [4] go further by applying a gradient-descent-like algorithm to the hyperparameters themselves. There is even work on determining whether it is even important to optimize hyperparameters [5]. However, all of these articles focus on hyperparameter optimization by repeatedly training on the full training set with different hyperparameters, and seeing which is best. If the training set is large or the model is complex, this may be highly impractical.

According to the theoretical framework proposed by Prof. Rishi Gupta and Prof. Tim Roughgarden [1], we can learn these hyper parameters by using grid search on a significantly smaller subset of the full training set. We examine specifically the case of tuning the step size ρ used in gradient descent.

More formally, let $\mathcal{D}$ be an hidden[1] probability distribution over a set of functions that can be optimized by gradient descent[2]. For a function f in the support of $\mathcal{D}$, let $\text{Cost}(f; \rho, \nu)$ be the number of steps it takes gradient descent to converge such that the gradient $\nabla f(\rho)$ has norm less than ν. We would like an algorithm that learns the hyperparameter ρ that minimizes the expected cost $C(\rho) \equiv E_{f\sim D}[\text{Cost}(f; \rho, \nu)]$. Intuitively, $\mathcal{D}$ defines a "family" of related optimization problems, and we would like a hyperparmeter ρ that does well for the problems in this family.

It is proposed in [1] that, in order to select a good hyperparameter, one does not need to perform a grid search or random search on the full set of training data; rather, it suffices to run a grid search on a much smaller subset of the training data. In particular, suppose that we want to find the best step size $\rho^* \in [\rho_\ell, \rho_u]$ for some range $0 < \rho_\ell < \rho_u$; there are known constants $L > 0, Z > \nu, c \in (0, 1)$ such that for any function f in the support of our hidden distribution $\mathcal{D}$:

- f is L-smooth; i.e. $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$; intuitively, the gradients do not change too fast
- the starting point x_0 for gradient descent is chosen such that $\|x_0 - x^*\| \leq Z$, where x^* is the minimizer of f; intuitively, we do not start too far away from the minimum
- $\|x - \rho\nabla f(x) - x^*\| \leq (1-c)\|x - x^*\|$; intuitively, there is a guarantee that progress is made toward the minimum.

Under these reasonable conditions, [1] shows that running a grid search with a grid of size $(\rho_u - \rho_\ell)/K$ where $K = \nu c^2/LZ$, and $\tilde{O}(H^3/\tau^2)$ samples from $\mathcal{D}$, we can find ρ that achieves error $\varepsilon(\rho) \equiv C(\rho) - C(\rho^*) < 1 + \tau$ with high probability, where $\tilde{O}$ hides logarithmic factors and $H = \log(\nu/LZ)/\log(1 - c)$ (we discovered a mistake in the analysis of [1], which lists instead $H = \log(\nu/Z)/\log(1 - c)$. This mistake does not affect the rest of the analysis in [1] nor any of our findings.)

II. Empirical Testing: A Simple Case

Our first objective was to test the above hypothesis on some simple functions.

A. A very simple distribution

Consider quadratic functions of the form $f(x) = \frac{1}{2}ax^2$ for some constant $a \in [m, L]$, where $L > m > 0$. Let $\mathcal{D}$ be the distribution generated when a above is distributed with density $p(a)$ on support $[m, L]$, and fix the starting point $x_0 = Z$. Suppose $\rho_u \leq 2/(m + L)$. so that gradient descent does not blow up too much. Arbitrarily, let's set $\rho_\ell = \frac{1}{2}\rho_u$. Then for such functions $f \sim D$, we make a few observations:

- $\nabla f(x) = ax$, so f is L-smooth

[1] i.e. the algorithm does not know $\mathcal{D}$, but it can sample problems $f \sim \mathcal{D}$

[2] under the assumptions laid out in the next paragraph

- The minimum progress condition is satisfied by the value $c = \rho_\ell m$

and so $\mathcal{D}$ satisfies the conditions above.

B. Computing ρ^*

This distribution is straightforward enough that we can compute the optimal value ρ^* mathematically if we assume that the costs are large enough that they can be treated as continuous. Let $f(x) = \frac{1}{2}ax^2 \sim D$. Notice first that for any $\rho \in [\rho_\ell, \rho_u]$, by similar logic to that used in [1] to compute H, we have

$$\begin{aligned}
\text{Cost}(f;\rho) &= \frac{\log(\nu/aZ)}{\log(|1-a\rho|)} \\
C(\rho) &= \mathop{\mathbb{E}}_{\alpha\sim p}\left[\frac{\log(\nu/\alpha Z)}{\log(|1-\alpha\rho|)}\right] \\
&= \int_m^L \frac{\log(\nu/\alpha Z)p(\alpha)}{\log(|1-\alpha\rho|)}\,\mathrm{d}\alpha \\
\rho^* &= \operatorname*{argmin}_\rho C(\rho) \\
&= \operatorname*{argmin}_\rho \int_m^L \frac{\log(\nu/\alpha Z)p(\alpha)}{\log(|1-\alpha\rho|)}\,\mathrm{d}\alpha
\end{aligned}$$

While this integral is difficult to solve analytically, both the integral and optimization above are easily solved by Scipy's `integrate` and `optimize` libraries, allowing us to easily compute the actual error $\varepsilon(\rho) = C(\rho) - C(\rho^*)$ for any given ρ.

C. Methods

If the bounds in Section I hold precisely, we expect that, if we use a grid of size $(\rho_u - \rho_\ell)/K$ and a sample size $\tilde{O}(H^3)$, then, in the average case, we should be able to find ρ that achieves error $\varepsilon(\rho)$ that does not increase with H or $1/K$ increase. Here, we attempt to confirm this theoretical result.

We will use the parameter range $[\rho_\ell, \rho_u]$ defined above. Notice first that, using these parameters, H and K do not depend on the exact values of ν, Z, m, L, but only on the ratios ν/Z and m/L. We can therefore WLOG set $Z = L = 1$ and focus only on ν and m. Further, we notice that K is tiny for any reasonable values of these parameters, so we take the liberty of making the grid size $O((\rho_u - \rho_\ell)/K)$ (adding a constant factor). We pick the default values $\nu = 2^{-23}, m = 2^{-9}$, and attach a constant factor of 10^{-10} onto both the grid size and the sample size. These values are somewhat arbitrary, but they were picked because (a) they are somewhat reasonable in practice: typically we have no idea what the "eigenvalues" of our gradient descent are, so $m \ll L$, and we want to stop at a very small gradient, so $\nu \ll Z$; (b) the constant factors mean that the sample size and grid size are not too large, so that the runtime of our experiments is reasonable; and (c) ν and m are small enough that the gradient descent takes long enough that errors are measurable.

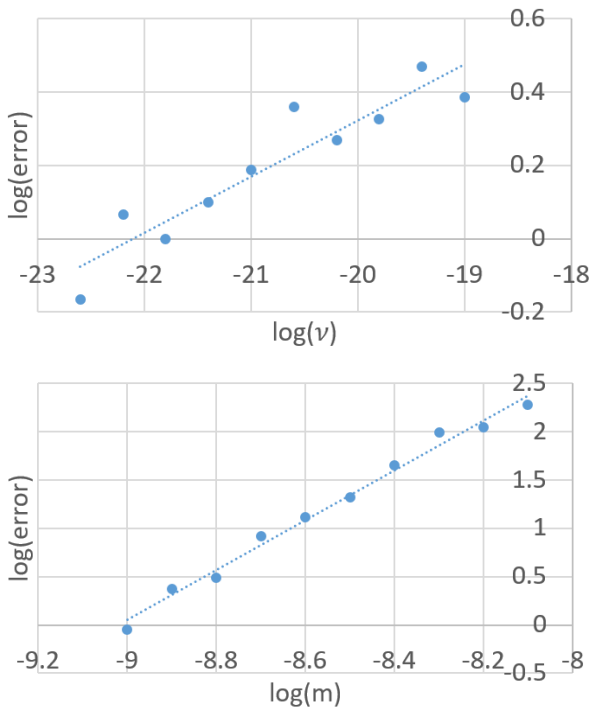

Fig. 1. Top: Plot of ν against average error, while holding m fixed. Bottom: Plot of m against average error, while holding ν fixed. Both plots are log-log plots; all logs are taken base 2

For concreteness, using with the values $m = 2^{-9}, \nu = 2^{-23}$ (which are used in the tests below) and plugging in Prof. Roughgarden's formulas gives $H \approx 8200$ and $K \approx 4.5 \times 10^{-13}$, which immediately makes clear the need for scaling the grid and sample sizes by a very small constant factor. After scaling, we use a sample size of 54 and grid size of 220, which is entirely reasonable to work with.

D. Experiments and Results

For our first test, we consider a simple uniform distribution p (i.e. $p(a) = \frac{1}{L-m}$). we leave m at 2^{-9} and vary ν between 2^{-23} and 2^{-19}. At each value of ν, we run the algorithm $T = 1000$ times, achieving T different empirically computed step sizes $\rho_1, \ldots, \rho_T$, and then compute the average error $\frac{1}{T}\sum_{i=1}^{T}\varepsilon(\rho_i)$. For our second test, we leave ν at 2^{-23} and vary m between 2^{-8} and 2^{-9}, using the same procedure.

Graphs of the results of the above tests can be found in Fig. 1. In both cases, we see clearly that, as m and ν decrease (so that H and $1/K$ are both increasing), the expected error decreases almost linearly in the log-log plot. As stated before, the theoretical result would predict

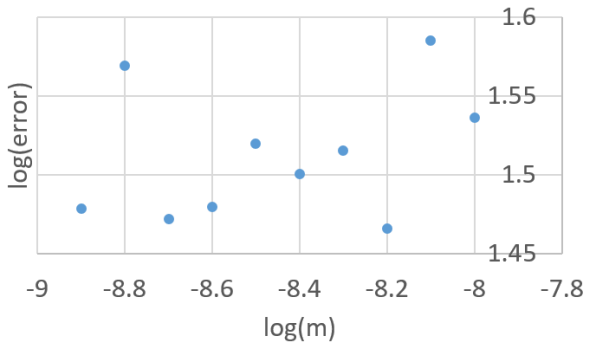

Fig. 2. The m vs. error plot, like the one in Fig. 1, but this time with sample size $O(\sqrt{H})$ instead of $O(H^3)$. Notice the now complete lack of correlation.

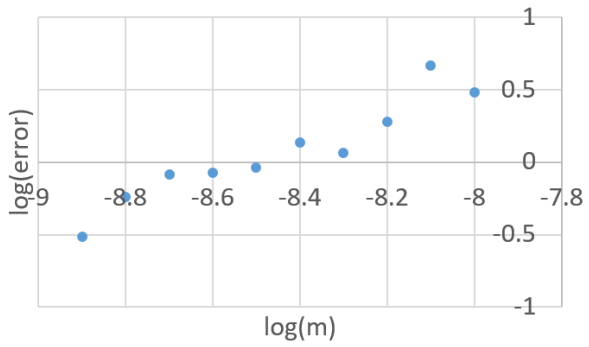

Fig. 3. The m vs. error plot, like the one in Fig. 2, but this time with a log-uniform distribution.

that error should not change as we vary the parameters. The linear relationship shown in the graph suggests that the theoretical bound can be tightened in some practical cases such as this. We now seek what exponent on H will be needed to make these bounds tight; in other words, try to sample $O(H^k)$ samples for different values of k, and find the k for which the resulting graphs have near-zero linear relationship. Remarkably, some testing with various exponents yields that $k \approx 1/2$ is the proper exponent. Using $O(\sqrt{H})$ instead of $O(H^3)$ samples for each test, we end up with the graph in Fig. 2, in which we find nearly no relationship between the complexity of the problem (which grows with decreasing m) and the error rate. This suggests that the number of samples required in this simple case is around $O(\sqrt{H})$.

Another case to test was that of different distributions: up to now we have been using p as a uniform distribution $p(a) = \frac{1}{L-m}$. To investigate whether these above observations also applied to other distributions, we also tested a log-uniform distribution; i.e. one for which $\log a$ is uniformly distributed between $\log m$ and $\log L$. It can be easily checked that the density function for this distribution is

$$p(a) = \frac{1}{a(\log L - \log m)}$$

We run the test by simply swapping out the uniform density function for this new density. Even in this case, we find similar results; in particular, a plot with $O(\sqrt{H})$ samples is shown in Fig. 3, demonstrating that this number of samples is also more than sufficient in the log-uniform case.

E. Analysis

The above experiments show clearly that the bound on the number of samples required is even smaller than $O(\sqrt{H})$ for multiple distributions $\mathcal{D}$ over one-dimensional quadratics. Further quick experiments (plots not shown) yield negative correlations when the number of samples is decreased to $O(1)$, indicating that the bound lies somewhere between $O(1)$ and $O(\sqrt{H})$ for more than one distribution. This is surprising, because it is a far cry from the $O(H^3)$ samples recommended by [1].

Notice also the stronger positive correlation and generally smaller errors observable in Fig. 3, which uses the same sample sizes as Fig. 2. These differences suggest that in some sense the log-uniform distribution is "easier to learn" than the uniform distribution.

Further, we must note that the constant factors on the grid and step sizes are not large (in the first experiment we have set both to 10^{-10}), and the average errors that we are finding are also not large at all: on the order of $2^3 = 8$ iterations or fewer. This means that, in practice, at least for gradient descent on one-dimensional quadratics, even for very extreme values of ν/Z and m/L, very few samples (< 100) and a very small grid (also < 100) are needed to get a good idea of what the best step size is.

Lastly, some preliminary experimentation indicates that similar error rates are achieved for higher dimensional general quadratic forms $f(x) = \frac{1}{2}x^T Ax$. Sadly, these claims are harder to verify, since it is not as straightforward to compute the cost function $C(\rho)$ in these cases, and so the error ε is harder to compute as well. However, since none of the calculations we performed have any dependence on the dimension of the problem, it is reasonable to predict that the errors would not change.

This work shows that, empirically, in many cases, a small sample is sufficient to discern the best step size. It must be said that the numerical calculations performed here to determine the required step and grid size are difficult to do in practice, since the values, in particular Z and c, are hard to know in advance. But the general framework–sample a small number of training examples, find the step size that works best on these examples, and claim that this step size should work well on the

whole training set–applies regardless. In the next section, we apply this general framework to a neural network–a highly complex system for which these calculations are nearly impossible to perform, and for which convergence to a global optimum is not even guaranteed–to see how these observations hold in practice.

III. Application: Neural Networks for Digit Classification

To test this theory on a practical problem, we build a simple neural network to be used on the classic problem of digit classification. Our inputs are 28×28 grayscale images of digits, and our output is a digit, 0-9. We set up a neural network with one input layer of size $784 = 28^2$, a varying number of hidden layers each with 24 neurons, and an output layer of size 10 (one output neuron for each digit). We used the sigmoid function

$$\sigma(z) = 1/(1 + \exp(-z))$$

as our activation function, and the standard backpropagation algorithm. In the final layer, we use the softmax function for output:

$$\hat{y}_j = \frac{\exp(z_j)}{\sum_{k=0}^{9} \exp(z_k)}$$

where z_j is the value in the jth output neuron, and $\hat{y}_j$ can be interpreted as our confidence that this particular example encoded the digit j. Further, to smooth out costs, we use the cross-entropy cost function:

$$\text{Cost} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=0}^{9} \Big[y_j^{(i)} \log \hat{y}_j^{(i)} + (1 - y_j^{(i)}) \log\Big(1 - \hat{y}_j^{(i)}\Big) \Big]$$

where $y_j^{(i)}$ is an indicator for whether the ith training example was an image of the digit j, and $\hat{y}_j^{(i)}$ is the value of the jth output. All of the above techniques are well known and detailed in e.g. [6]

We decided to test a grid of three step sizes: $2.5, 0.25, 0.025$, and sample sizes ranging from 10 to 10000. These highly varying step and sample sizes were sufficient to give us a good idea of the general picture. As stated before, gathering actual numerical data such as Z or c in the prior analysis would be nearly impossible for this problem, so we settle for applying the general idea of using a small grid and small sample size.

It quickly became clear that more than 50 samples would be required even for a neural net with just one hidden layer. However, 100 samples sufficed for the one-layer case, as can be seen in the first and third graphs in Fig. 4: with 100 samples (top graph), it is clear

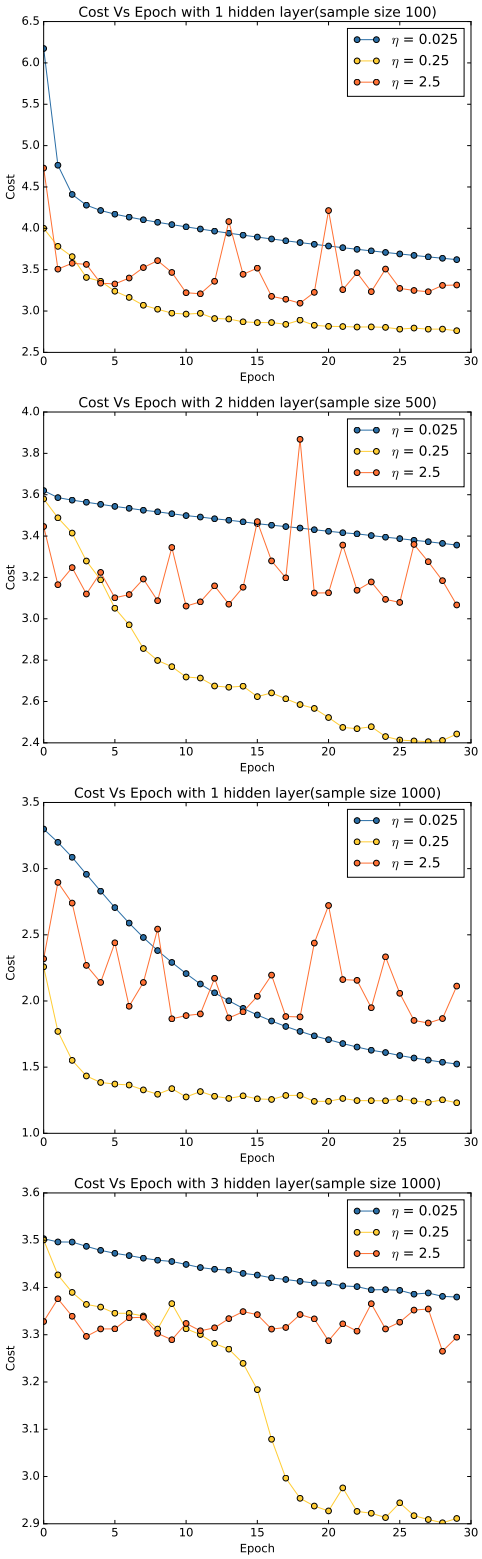

Fig. 4. Number of iterations (x-axis) vs entropy training cost (y-axis) for various choice of number of hidden layers and sample sizes

that 0.25 is the best step size; this pattern continues up through 1000 samples (third graph) and beyond, so we can safely conclude that 0.25 is indeed a good step size that generalizes to the full training set.

As we increased the number of layers, the sample size required to find a reasonable step size also increases. With two layers and the same sample size 100 (graph not shown), step sizes 2.5 and 0.25 had nearly the same learning curve. Increasing to 500 samples sufficed to clearly discern the difference between 0.25 and 2.5 for two layers (second graph in Fig. 4). For three layers, 1000 samples sufficed. It is clear, then, that the number of samples necessary increases with the complexity of the neural network. This is of course expected intuitively, but it needs some justification since the theoretical results include no mention of the number of variables in the function. One justification for this phenomenon is that adding layers can have an effect on the minimum progress condition c and the starting distance Z from the local optimum, both of which are components of the formula in Section I for the number of required samples. However, this is again hard to verify, because Z and c are difficult if not impossible to compute in this scenario.

We notice that even for a reasonably sized neural network with three hidden layers, we do not require such a large sample (only 1000 training examples) to determine a good step size to use in gradient descent. This is extremely important, because this means that, in practice, finding a good step size, even for an objective as complex as a neural network,

IV. Conclusions

In this report, we examined the practical implications of a theoretical framework for identifying good hyperparameters; in particular, for learning good step sizes. We applied the theoretical framework to a simple quadratic optimization problem, and found two main results: first that the theoretical bounds given were not at all tight (we were able to demonstrate a bound in practice of $O(\sqrt{H})$ instead of $O(H^3)$), and that, in practice, generally a very small sample and very small grid are sufficient to determine a very good step size. We then applied these general observations to a more complex system–a neural network–and found that, even as the complexity of the neural network grows very large, a reasonable number of training examples (< 1000) can be used to determine a good step size, which can then be applied to the full training set. This means that, in practice, one can find a good step size to use without too much effort using a small training set, and then say with some confidence that this step size will also work on the full training set.

V. Further Work

More work can be done to figure out what distributions are "harder" than others to learn: maybe there is some distribution p so difficult that it does require $O(H^3)$ samples to learn. Further, it could be possible to prove the irrelevance of dimension–doing this amounts to just proving that the error $\varepsilon(\rho)$ does not depend on dimension, which appears to be empirically true. Finally, it could be possible to come up with estimates of Z and c for the neural network case that are good enough that we can directly apply the calculations shown in the previous parts.

References

[1] R. Gupta and T. Roughgarden, "A PAC approach to application-specific algorithm selection," in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pp. 123–134, ACM, 2016.

[2] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[3] 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*, 2015.

[4] Y. Bengio, "Gradient-based optimization of hyperparameters," *Neural computation*, vol. 12, no. 8, pp. 1889–1900, 2000.

[5] H. Hoos, U. CA, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," 2014.

[6] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016.