Adaptive Hyperparameter Search for Regularization in Neural Networks

Devin Lu
Stanford University
Department of Statistics
devinlu@stanford.edu

June 13, 2017

Abstract

In this paper, we consider the problem of whether we can achieve comparable or better validation accuracy from a regularization constant that varies during training than from a fixed hyperparameter. Also, can we consider the problem of developing a policy that can change the regularization parameter based on feedback from training rather than using a fixed schedule.

1 Introduction

Traditionally, regularization constants and other hyperparameters are fixed for a model throughout training. Optimizing these hyperparameters is usually done through splitting out a portion of the training data into an evaluation set separate from the test data specifically for use in hyperparameter optimization. The model is then trained repeatedly using different hyperparameter settings and an optimal choice is made from performance on the dev set. The downside of this approach is that this is often very expensive, such as when training the model is expensive or when data volume is large enough that training examples cannot all be held in memory. These are issues that often arise with neural networks. Here, we examine whether using a regularization parameter that varies during training can accelerate the process of achieving a certain level of accuracy or even achieve a model that performs better than what is achieved the traditional hyperparameter optimization. We also examine the question of whether we can develop a policy that determines changes in regularization rather than using a fixed schedule.

2 Previous Work

The concept of an adaptive regularization parameter in a different domain has been considered before in "Adaptive regularization parameter adjustment for reconstruction problems" by Watzenig, et al (2004). In that paper, they considered the field of reconstruction problems and used an approach based on condition number.

3 Problem and Data

To examine the problem of adapative regularization, we chose to try an image classificatin task.
We used the CIFAR-10 dataset. This dataset is 60,000 images each of which is 32 x 32. Each image is colored, so with the three color channels each image consists of 32 x 32 x 3 = 3072 floating point values. Each image falls into one of 10 classes. The dataset was split into a training set consisting of 50,000 images and a test set consisting of 10,000 images.
The total size of the dataset in memory is approximately 163MB.

4 Neural Network Comparison

4.1 Pull Down/Up Schedule

As a first test we implemented a feed forward neural network. We tried using a schedule that started with high regularization that gradually reduced. We compared this against using any of the individual regularization parameters in the schedule.

We implemented a two-layer neural network for the classification task and compared using a "pull-down" regularization schedule (starting with a high regularization and subsequently decreasing) and a "pull-up" regularization schedule (starting with a low regularization schedule and subsequently increasing) vs using a fixed regularization schedule.

Fig 1 shows the increase in accuracy from using a varying schedule over using the best fixed regularization parameter. To make the comparison fair, each comparison model was trained for as many iterations in total as during the varying schedule training.

To illustrate what we mean by this, suppose we used a regularization schedule [3.0, 2.0, 1.0, 0.0], training the neural net for 1000 iterations for each value. In total, this means we trained the neural net for 4000 iterations. To compare, we would then train four separate neural networks, one each with regularization 3.0, 2.0, 1.0, and 0.0. We would train each of these for 4000 iterations.

Fig 1 shows the gain in accuracy for using a pull-down regularization schedule over the best model found this way using a constant regularization term. This is equivalent to the gain from using a varying regularization schedule over identifying the best constant regularization parameter with grid search and using that as the baseline. This is effectively a speedup of a factor of n, where n is the number of regularization parameters to search over. Interestingly, we find that in addition to the speedup, we can find an absolute performance gain by using the pull-down schedule.

We found that a pull-down schedule generally worked better than a pull-up schedule. Figure 2 shows generalization accuracy of a pull-up vs pull-down regularization schedule as a function of the number of hidden units. This was for a fixed 1000 iterations per constant.

4.2 Analysis

As shown in Figure 2, a pull down schedule (high regularization to start and decreasing with time) generally performed better than a pull up schedule (low regularization to start and increasing with time). We suspect this is because in the beginning of training, essentially all training is overfitting since there is literally nothing else. In other words, at the beginning of training, the model will tend to overfit to the idiosyncrasies of the early batches. If regularization is strong at this point, weights will not get updated to fit the false positive signals that exist simply because of the random distribution of the initial batches. While it is theoretically possible to eventually learn to ignore these false signals, it will take longer to "unlearn" rather than to simply not learn in the first place, especially if learning rate decay is heavy. Therefore, an early heavy regularization will prevent more of these false starts when the model is most vulnerable to it, resulting in faster convergence to better accuracy.

As can be seen in Figure 1, the varying regularization schedule generally helped more as the number of hidden units increased. We believe this is because as hidden unit numbers increase, the complexity of the loss surface increases exponentially. Thus it is likely there will be more bad local minima that happen to exist because simply because of the particularities of the training data. Since changing regularization changes the loss surface, "weak" local minima are more likely to no longer become critical points if regularization changes. Thus, the network will probably escape these points, an effect that would be pronounced with higher hidden units.

Finally, we observed that the gain from the varying regularization schedules tended to decrease as the iterations per constant increased. This suggests to us that part of the benefit in varying regularization schedules is in accelerating convergence to a certain accuracy, even if given sufficient computational resources, a vanilla grid search could eventually get to a similar performance.

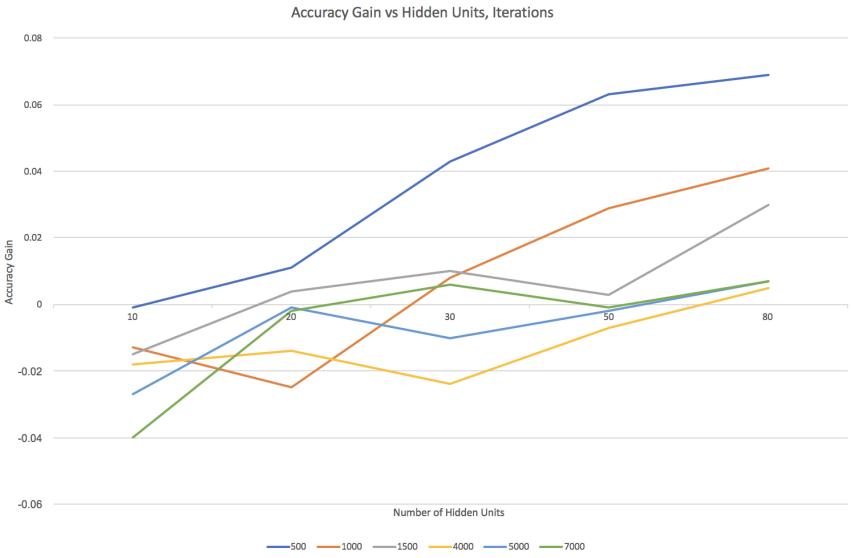

Figure 1: Accuracy gain from a pull-down regularization schedule, as a function of the number of hidden units and iterations per constant.

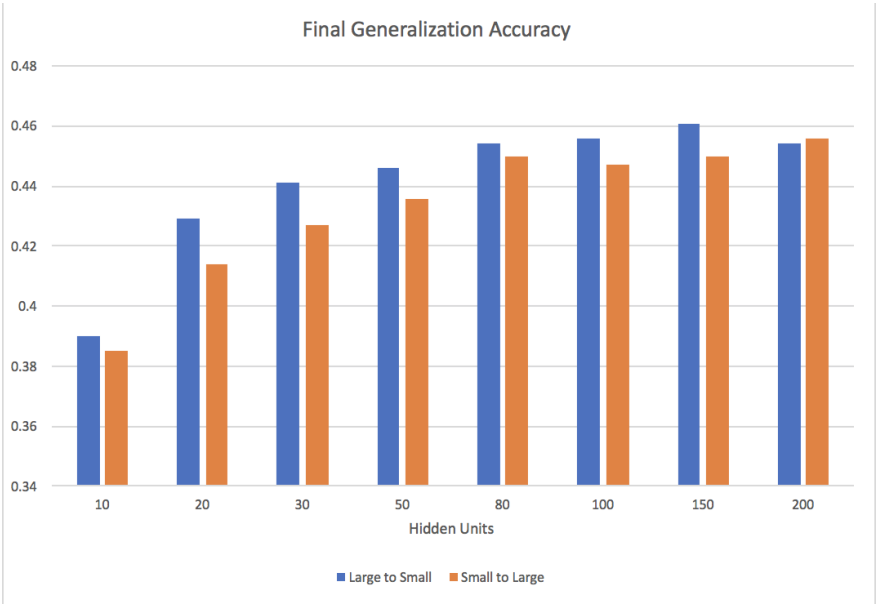

Figure 2: Difference in final generalization accuracy for a pull-down (Large to Small) vs pull-up (Small to Large) schedule. Iteration number fixed at 1000.

5 Adaptive Regularization

In addition to using a fixed regularization schedule, we also experimented with an adaptive regularization parameter. In this approach, we would during training continuously observe the training and validation accuracy and use this as feedback for determining how to adjust the regularization parameter. Specifically, if the training accuracy is significantly higher than the validation accuracy, we adjust the regularization constant up because this indicates overfitting. If the training accuracy is very close to or lower than the validation accuracy, we adjust the regularization constant down.

We examined this approach both with our feed-forward neural network we used in section 4 and with a three-layer convolutional neural network. In Figure 3, we show the results for the feed-forward neural net. In general, there is a positive gain from using this adaptive method, but the gain tends to decrease with the number of hidden units.

Figure 4 shows the results for a convolutional neural network with fixed architecture as a function of the epoch number. We see that the adaptive method usually gives a gain, especially in the earlier epochs, but the gap tends to close as the model trains for many iterations. Note however that the adaptive method reaches its final validation accuracy earlier, while the fixed schedule tends to continue improving, eventually reaching roughly the same level as the adaptively trained model. This is similar to what we saw in Section 4, which suggested the varying regularization schedules were speeding up convergence.

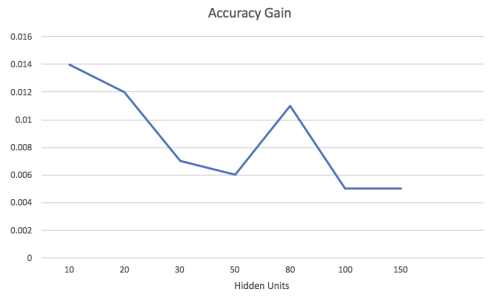

Figure 3: Accuracy gain over the feed-forward neural network as a function of the number of hidden units.

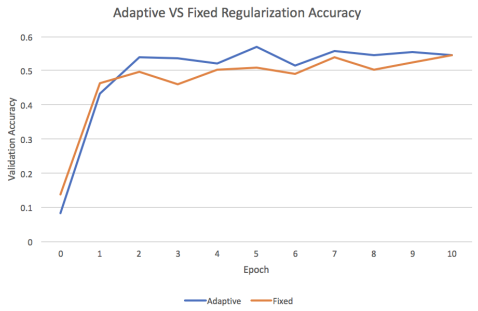

Figure 4: Accuracy gain using an adaptive regularization schedule over a fixed schedule for a three-layer convolutional neural network.

6 Nonconvex Regularization

Usually, ℓ_1 or ℓ_2 norm is used for regularization. The reason is that ℓ_1 and ℓ_2 norms are both convex functions.

Convex functions are preferred in many machine learning applications because they have the property that there is at most one locally optimal value, which coincides with the global optimum, if it exists. Additionally, the set of points achieving this optimal value is a connected, convex set. Therefore any optimization method that eventually converges to a local optimum (like stochastic gradient descent) also converges to a global optimum. Convex functions are thus easy to optimize.

Many traditional machine learning algorithms, such as logistic regression, are formulated with convex loss functions for this reason. The space of convex functions have the nice property that it is closed under addition, so adding an ℓ_1 or ℓ_2 regularization term to a convex loss term keeps the loss convex. If the regularization term used a non-convex function, such as a loss of the form ℓ_m for $m < 1$, there is no guarantee the resulting loss function would be convex, and so there could exist multiple local minima.

However, neural network losses are already highly non-

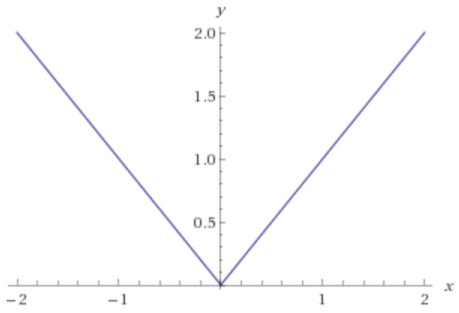

Figure 5: Loss of L1-regularization

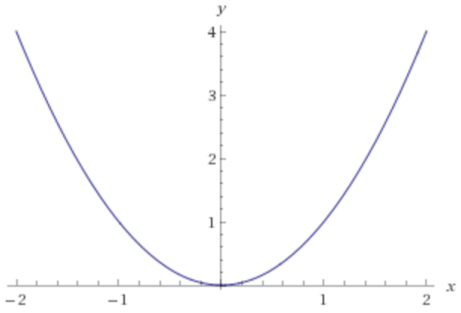

Figure 6: Loss of L2-regularization.

convex. Therefore, the reason for restricting regularization terms to ℓ_1 or ℓ_2 norms is already lost.

Therefore, we also experimented with using an $\ell_{\frac{1}{2}}$ loss, i.e., $||x||_{\frac{1}{2}} = |x|^{\frac{1}{2}}$. We expect this to have a sparsifying effect on our weights. To see why, consider Figures 5-7. Suppose a feature has weight w. Reducing any particular feature weight by δ in ℓ_1 regularization reduces the regularization loss by a factor linear in δ. In ℓ_2 regularization, $\frac{d}{dw}w^2 = 2w$, so reducing a feature weight by δ reduces the regularization loss by $O(w\delta)$. However, in ℓ_{frac12} regularization, $\frac{d}{dw}w^{\frac{1}{2}} = \frac{1}{2\sqrt{w}}$, so reducing a feature weight by δ reduces the regularization loss by $O(\frac{\delta}{\sqrt{w}}) = O(\frac{1}{\sqrt{w}})$. Therefore, as $w \to 0$, ℓ_2 regularization rewards reducing w less and less, ℓ_1 keeps the reward constant, and $\ell_{\frac{1}{2}}$ rewards it more and more. Therefore, we would expect that $\ell_{\frac{1}{2}}$ regularization promotes sparsity even more than ℓ_1.

However, we observed that using both a pull-up and pull-down schedule harmed final accuracy with $\ell_{\frac{1}{2}}$ regularization (Fig 8). Generally, the larger the number of hidden units, the greater loss we encountered.

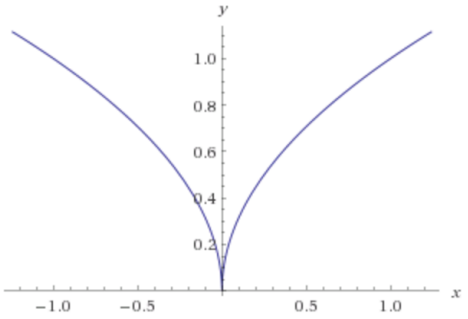

Figure 7: Loss of L$\frac{1}{2}$-regularization.

7 Conclusion

We completed a study on using a regularization parameter that changes over time instead of a fixed parameter. We found that it can help in certain neural network models, either by accelerating convergence to a particular accuracy level or sometimes achieving an accuracy level beyond what can be obtained by a grid search-based optimal hyperparameter selection over the same parameters as used in the varying schedule. We also examined using $\ell_{\frac{1}{2}}$ regularization and found a varying regularization schedule tended to hurt performance.

Future work is needed to develop more sophisticated feedback policies for adaptively changing the regularization parameter and understanding why using $\ell_{\frac{1}{2}}$ regularization produced the opposite result to the typical ℓ_2 case.

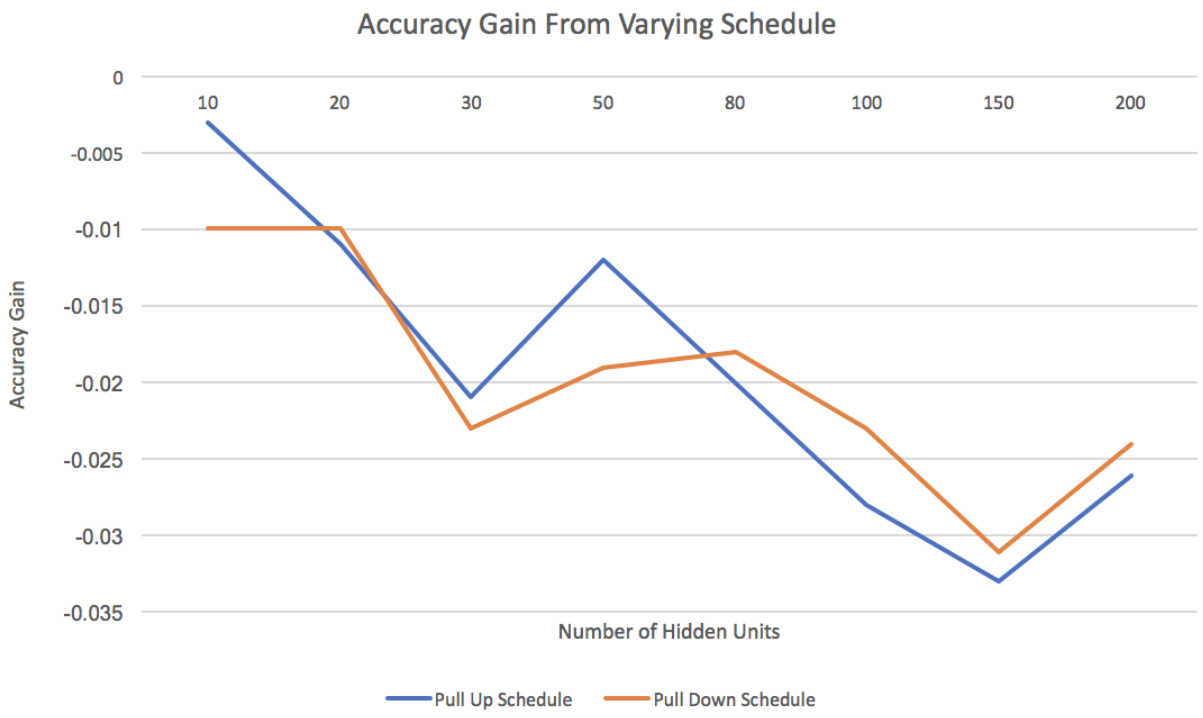

Figure 8: Accuracy loss from a varying regularization schedule under $\ell_{\frac{1}{2}}$ loss. Loss grows more severe with greater number of hidden units.

8 References

Watzenig, et al (2004). "Adaptive regularization parameter adjustment for reconstruction problems." IEEE Transactions on Magnetics. Volume 40, Issue 2, March 2004.