In *Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN'01),* 115-119.

The Need for Small Learning Rates on Large Problems

D. Randall Wilson
fonix Corporation
180 West Election Road
Draper, Utah, USA
WilsonR@fonix.com

Tony R. Martinez
Computer Science Department
Brigham Young University
Provo, Utah, USA
martinez@cs.byu.edu

Abstract

In gradient descent learning algorithms such as error back-propagation, the learning rate parameter can have a significant effect on generalization accuracy. In particular, decreasing the learning rate below that which yields the fastest convergence can significantly improve generalization accuracy, especially on large, complex problems. The learning rate also directly affects training speed, but not necessarily in the way that many people expect. Many neural network practitioners currently attempt to use the largest learning rate that still allows for convergence, in order to improve training speed. However, a learning rate that is too large can be as slow as a learning rate that is too small, and a learning rate that is too large or too small can require orders of magnitude more training time than one that is in an appropriate range. This paper illustrates how the learning rate affects training speed and generalization accuracy, and thus gives guidelines on how to efficiently select a learning rate that maximizes generalization accuracy.

1 Introduction

Training a neural network using an algorithm such as error back-propagation [1,2,3,4] usually requires a lot of time on large, complex problems. Such algorithms typically have a *learning rate* parameter that determines how much the weights can change in response to an observed error on the training set. The choice of this learning rate can have a dramatic effect on generalization accuracy as well as the speed of training. Almost anyone who has used such training algorithms has been faced with the problem of choosing the learning rate, but there is seldom much guidance on what value to use, since the best value to use depends on the particular task.

Several algorithms exist for automatically tuning the learning rate parameter [6, 7, 8], but such methods typically concentrate on improving the speed of convergence and fail to focus on generalization accuracy.

Many neural network practitioners currently use the largest learning rate that allows convergence, in an attempt to speed up training. However, on large and complex problems, a learning rate that is too large hurts generalization accuracy and also slows down training. On the other hand, once the learning rate is small enough, further reductions in size waste computational resources without any further improvement in generalization accuracy.

It should be stated at the outset that this paper assumes the use of *on-line* training (where weights are updated after the presentation of each training instance) rather than *batch* training (where weight changes are accumulated and applied at the end), since batch training requires more training time than on-line training on most problems of practical interest without any corresponding improvement in accuracy [5].

This paper shows the typical effect of the size of the learning rate on training speed and generalization accuracy. It then shows how to find the learning rate which will yield the fastest training, and then discusses how to decide when a learning rate is small enough to yield the maximum generalization accuracy.

2 The Effect of Learning Rates on Training Speed and Accuracy

When using a gradient descent learning algorithm, the error gradient (or an approximation thereof) is calculated at the current point in weight space, and the weights are changed in the opposite direction of this gradient in an attempt to lower the error. However, although the gradient may indicate what direction the weights should be moved, it does not specify how far the weights may safely be moved in that direction before the error quits decreasing and starts increasing again.

Therefore, a learning rate that is too large often moves *too far* in the "correct" direction, resulting in overshooting a valley or minimum in the error surface, thus hurting accuracy. Because of this effect, a learning rate that is too large takes longer to train, because it is continually overshooting its objective and "unlearning" what it has learned, thus requiring expensive backtracking or causing unproductive oscillations. This instability often causes poor generalization accuracy as well, since the weights can never settle down enough to move all the way into a minimum before bouncing back out again.

Once the learning rate is small enough to avoid such overcorrections, it can proceed in a relatively smooth path through the error landscape, finally settling in a minimum. Reducing the learning rate further can make this path more smooth, and doing so can significantly improve generalization accuracy. However, there comes a point at which reducing the learning rate any more simply wastes time, resulting in taking many more steps than necessary to take the same path to the same minimum.

2.1 Digit Speech Recognition Experiments

In order to illustrate the effect of learning rates on training speed and generalization accuracy, experiments were run on a phoneme classification task. A multilayer perceptron with 130 inputs (plus one bias), 100 hidden nodes, and 178 outputs was trained on 21,085 training instances. The output class of each instance corresponded to one of 178 context-dependent phoneme categories from a digit vocabulary. Each context-dependent phoneme belonged to one of 23 base phoneme classes. For each instance, one of the 178 outputs had a target of 1, and all other outputs had a target of 0. The targets of each instance were derived from hand-labeled phoneme representations of a set of training utterances.

The neural network was trained using 15 different learning rates from 100 down to 0.00001. Each neural network began with random initial weights, and the training instances were presented in a different random order each training iteration (or *epoch*). For all learning rates, the same random number seeds were used, so the initial weights and order of presentation of the training instances were identical.

To measure the accuracy of the trained neural networks, a hold-out set of 9,306 instances was used. The output node with the highest activation was chosen as the "winning" output. The output was considered correct if its context-dependent phoneme belonged to the same base phoneme as the target of the instance.

A summary of the results of these experiments is presented graphically in Figure 1. The bars indicate training time in epochs (using the scale on the left) before the maximum generalization accuracy was reached. The overlaid line indicates the maximum generalization accuracy achieved by each learning rate. These values are also presented numerically in Table 1, along with the total number of epochs tested for each learning rate.

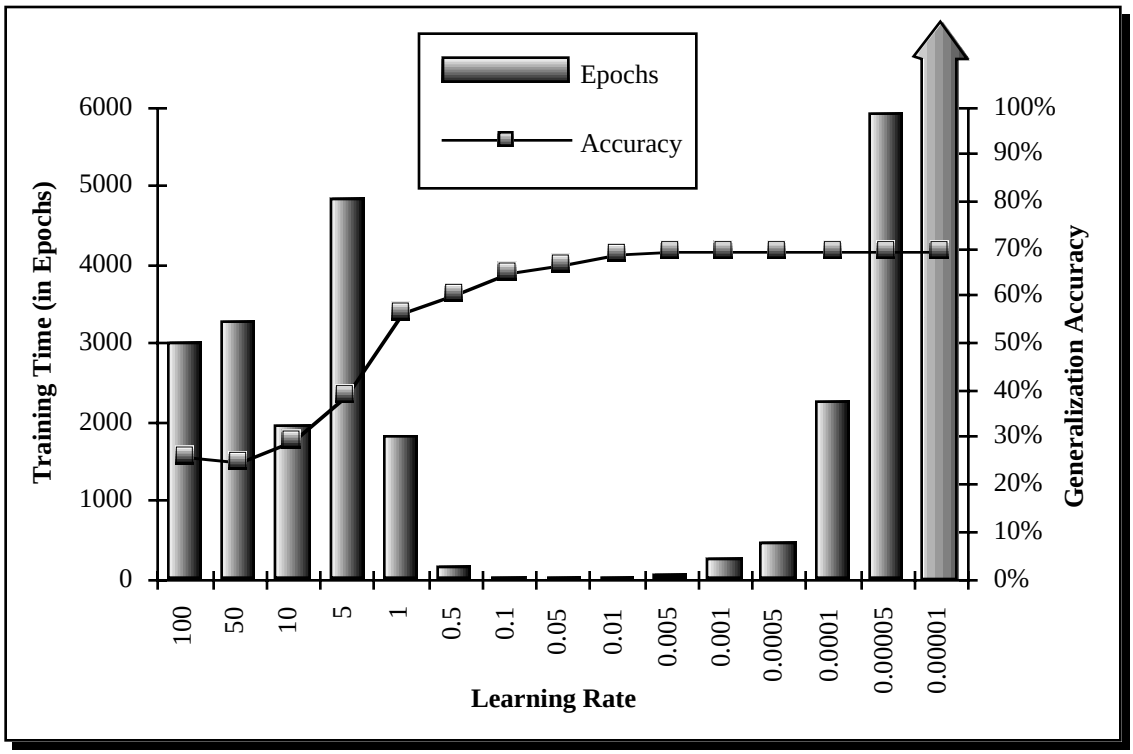

Figure 1: Training time (in epochs) and maximum hold-out set phoneme generalization accuracy for each learning rate, on a digit speech recognition task. The bars indicate the number of epochs needed to reach the maximum generalization accuracy, and the line indicates what the maximum accuracy was for each learning rate.

Table 1: Training time (in epochs) and maximum hold-out set phoneme generalization accuracy for each learning rate, on a digit speech recognition task.

Learning Rate	Best Training Epoch	Maximum Generalization Accuracy	Tested Training Epochs
100	3,017	25.63%	5,000
50	3,282	24.70%	5,000
10	1,955	29.21%	5,000
5	4,821	38.66%	5,000
1	1,817	56.14%	1,000
0.5	160	59.96%	1,000
0.1	7	64.86%	1,000
0.05	9	66.62%	1,000
0.01	14	68.71%	1,000
0.005	39	69.04%	1,000
0.001	242	69.48%	1,000
0.0005	473	69.47%	1,000
0.0001	2,252	69.43%	5,000
0.00005	5,913	69.47%	10,000
0.00001	18,595	69.46%	30,000

As can be seen from Figure 1 and Table 1, the maximum generalization accuracy is quite poor for large learning rates, and improves as the learning rate is reduced to a level of 0.001. Learning rates smaller than this show no improvement (nor, in fact, any significant difference).

Training time was quite fast for learning rates of 0.1 to 0.005, and reasonably fast from 0.5 to 0.0005. Larger learning rates, however, took thousands of epochs to reach their maximum accuracy, which was ultimately poor anyway. Learning rates smaller than 0.001 took longer and longer to reach the same accuracy as 0.001, but yielded no further improvement in accuracy.

Those seeking simply for the learning rate that produces the fastest convergence would probably settle on a learning rate of 0.1. However, doing so would yield a generalization accuracy of only 64.86%, which is significantly lower than the accuracy of 69.48% achievable by using a smaller learning rate.

3. How to Choose a Learning Rate

As illustrated in the above example, if the learning rate is too large, accuracy will be poor and training speed will also be poor. As the learning rate decreases, generalization accuracy improves and training speed also improves. As the learning rate is dropped further, accuracy continues to improve slightly while training speed starts to once again get worse. Finally, there comes a point at which accuracy flattens out and does not improve with a smaller learning rate, and further reductions in the learning rate only waste computational resources.

The obvious question, then, is how to choose a learning rate that is small enough to achieve good generalization accuracy without wasting computational resources.

3.1 Avoiding Learning Rates that are too Large

First we look at how to rule out learning rates that are so large that they train slower *and* are less accurate than a smaller learning rate. This can be done by examining either the generalization accuracy on a hold-out set or, more conveniently, by examining the *total sum squared (tss) error* calculated during the training process itself.

Figure 2 shows the *tss* error for each learning rate after 1, 5, 10, 100 and 1000 training epochs. In every case, the learning rate of 0.05 has the smallest error, indicating that it is the "fastest" learning rate, in terms of reduction in *tss* error. In this example, any learning rate larger than 0.05 can be discarded from further consideration, since it is slower and most likely less accurate, and thus has no advantage over the "fastest" learning rate.

In order to avoid learning rates that are too large, then, it is possible to train a collection of neural networks for just one epoch using learning rates at different orders of magnitude (using the same initial weights and order of presentation for each network). The learning rate that results in the lowest *tss* error is then considered the "fastest" learning rate, and all learning rates larger than that are discarded from further consideration.

3.2 Maximizing Accuracy Using Smaller Learning Rates

The "fastest" learning rate is quite often not the most accurate one, though it is almost always at least as accurate as any larger, slower learning rate. On large, complex problems, a learning rate smaller than the "fastest" one often results in higher generalization accuracy, so it is often worth spending additional training time using smaller learning rates in order to improve accuracy.

One approach to doing this is as follows.

1. Train the neural network for one epoch using learning rates of different orders of magnitude (e.g., 100 down to .00001) in order to find the learning rate *L* with the smallest *tss* error.
2. Continue training the network using the learning rate *L* until hold-out set accuracy begins to drop.
3. Record the maximum generalization accuracy for this learning rate.
4. Reduce the learning rate by a constant factor (e.g., 3) and retrain the network, using the same initial weights and order of presentation.
5. Repeat steps 3 and 4 until the maximum accuracy fails to improve.

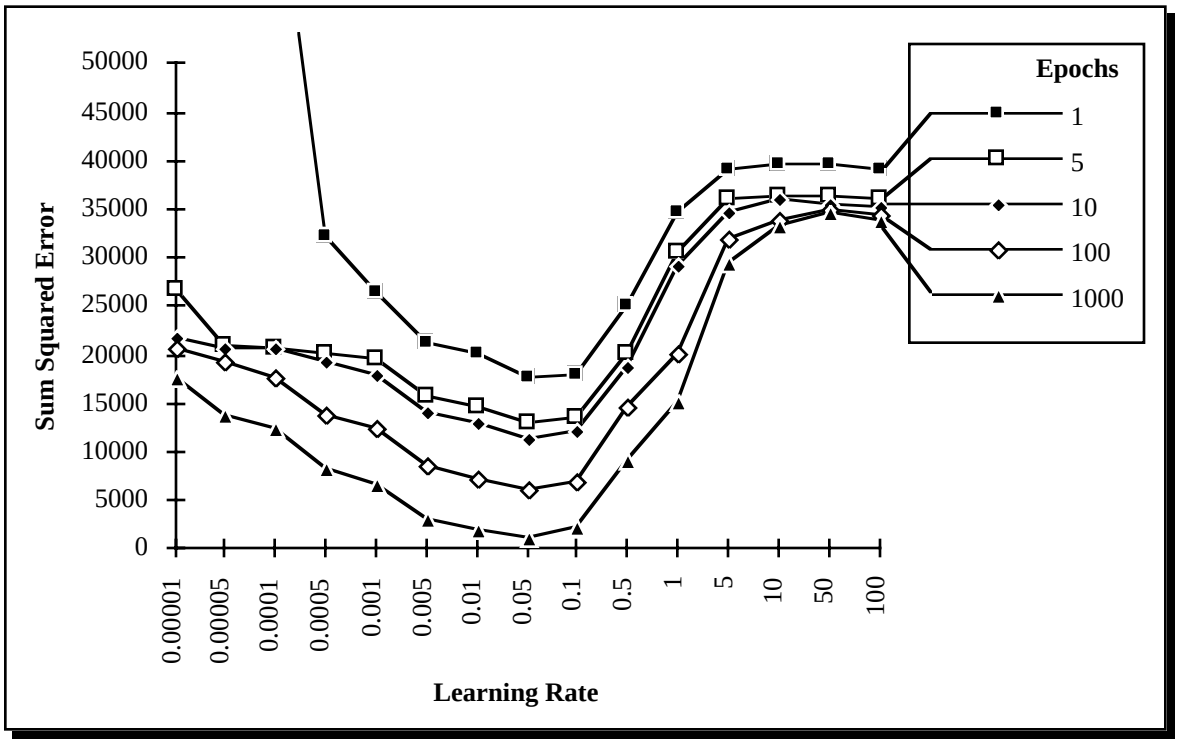

Figure 2: Total sum squared error of each learning rate after various numbers of training epochs. The learning rate of 0.05 consistently had the lowest *tss* error.

While this method does require training the neural network several times, the training is done first with the fastest learning rate and thus goes relatively quickly. Subsequent passes are slower, but only a few passes should be required before the accuracy ceases to improve. All of these passes put together will typically take about as long as the next slower learning rate.

In the speech recognition example, if we had used only powers of 10 between 100 and 0.00001 (rather than using 5, 0.5, etc.), 1 epoch of training would have been done for each of 8 learning rates, and 0.1 would be chosen as the "fastest" learning rate. Step 2 would require 7 epochs to reach a maximum, plus about 5 more to determine that the accuracy was dropping. Using our non-logarithmic reductions (i.e., dropping by alternating factors of 2 and 5), Step 4 would reduce L to 0.05, 0.01, 0.005 and 0.001 before reaching the maximum accuracy, followed by 0.005 to find that the accuracy had stopped improving.

Using the training times from Table 1, this process would require $7 + 9 + 14 + 39 + 242 + 473 = 784$ epochs to reach the maximums, plus another 10% or so (≈80 epochs) to determine that each had reached its maximum. Together with the 8 epochs required in Step 1, an estimated total of about 872 epochs of training would be needed in order to achieve the maximum generalization accuracy.

One advantage to this method is that it provides the confidence that the maximum accuracy has been reached. In addition, it avoids the extremely long training time required by even smaller learning rates (e.g., 18,595+ epochs required by 0.00001). If a very small learning rate were to be used right from the start, not only could much time be wasted, but there would still be guarantee that the best learning rate was really used.

3.3 Stopping Criteria

In order for the above method to work efficiently, it is important to use the correct stopping criteria. While c*onvergence* is often defined as reaching a certain level of *tss* error, this criteria is prone to overfitting and requires the user to select the error threshold.

Instead, hold-out set accuracy should be used to decide when the neural network has been trained long enough for a particular learning rate.

When the learning rate is at least as small as the "fastest" learning rate, the generalization accuracy typically moves up smoothly, starts to do some backtracking as it levels off, and then drops back down as the network overfits, as demonstrated in Figure 3. This figure also demonstrates how the learning rate of 0.05 reaches its maximum quickly and drops off while 0.01 reaches a higher maximum after a few more epochs. 0.005 eventually overtakes 0.01 and reaches its maximum after 39 epochs. 0.0001, not shown

in the chart, proceeds much more slowly, but eventually beats 0.005 by a half a percentage in accuracy.

In our tests we trained much longer than necessary in order to be able to be more sure of our conclusions. However, in practice one would need only train until the downward trend was observed in order to determine the maximum hold-out set generalization accuracy for each learning rate. This downward trend can be identified by seeing when a smoothed running average of the accuracy drops some fraction below the maximum (smoothed) accuracy seen so far.

4. Conclusions and Future Work

Choosing a learning rate has always been a bit of a magic art for neural network practitioners. This paper has shown how to fairly easily avoid learning rates that are so large that they take longer to train *and* are less accurate than networks produced by smaller, faster learning rates.

For large, complex problems, a learning rate smaller than the "fastest" one can often significantly improve generalization accuracy. For most problems the improved generalization accuracy is worth the additional time needed by smaller learning rates. However, using a learning rate that is too small is a waste of time and using any one single learning rate does not guarantee that the best accuracy has been found.

By starting with the "fastest" learning rate and reducing it on subsequent experiments, the maximum generalization accuracy can be achieved without the risk of training with learning rates that are too much smaller than the optimal one. This comes at a cost that is dominated by the number of epochs needed by the last pass.

While this paper presents guidelines on choosing a learning rate, there remains a need to incorporate these guidelines into an automated algorithm that will automatically find the "fastest" learning rate and guide training towards an optimal learning rate without the need for user intervention.

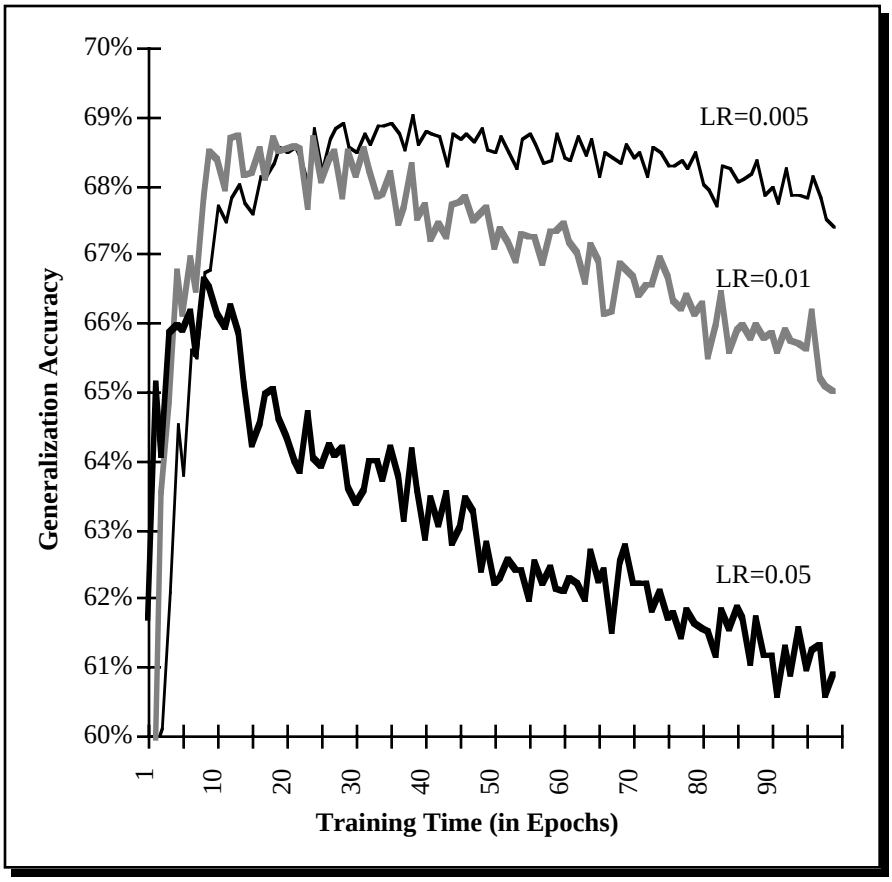

Figure 3: Generalization accuracy on a digit phoneme recognition task for learning rates 0.005, 0.01 and 0.05.

5 References

[1] Bishop, C. M., Neural Networks for Pattern Recognition, New York, NY: Oxford University Press, 1995.

[2] Hassoun, M., Fundamentals of Artificial Neural Networks, MIT Press, 1995.

[3] Rumelhart, D. E., and J. L. McClelland, Parallel Distributed Processing, MIT Press, 1986.

[4] Wasserman, P. D., Advanced Methods in Neural Computing, New York, NY: Van Nostrand Reinhold, 1993.

[5] Wilson, D. R., and T. R. Martinez, "The Inefficiency of Batch Training for Large Training Sets", in Proceedings of the International Joint Conference on Neural Networks (IJCNN2000), Vol. II, pp. 113-117, July 2000.

[6] Jacobs, R. A., "Increased Rates of Convergence through Learning Rate Adaptation," *Neural Networks*, Vol. 1, No. 4, pp. 295-307, 1988.

[7] Tollenaere, T., "SuperSAB: Fast Adaptive Back-propagation with Good Scaling Properties", *Neural Networks*, Vol. 3, No. 5, pp. 561-573, 1990.

[8] Vogl, T. P., J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the Convergence of the Back-propagation method", *Biological Cybernetics*, Vol. 59, pp. 257-263, 1988.