Descending Epsilon in Back-Propagation: A Technique for Better Generalization[1]

Yeong-Ho Yu
Robert F. Simmons

Department of Computer Sciences
and
Artificial Intelligence Laboratory
Taylor Hall 2.124, University of Texas at Austin,
Austin, Texas 78712-1188

March 1990 AI90-130

Abstract

There are two measures for the optimality of a trained feed-forward network for the given training patterns. One is the global error function which is the sum of squared differences between target outputs and actual outputs over all output units of all training patterns. The most popular training method, back-propagation based on the Generalized Delta Rule, is to minimize the value of this function. In this method, the smaller the global error is, the better the network is supposed to be. The other measure is the correctness ratio which shows, when the network's outputs are converted into binary outputs, for what percentage of training patterns the network generates the correct binary outputs. Actually, this is the measure that often really matters. This paper argues that those two measures are not parallel and presents a technique with which the back-propagation method results in a high correctness ratio. The results show that the trained networks with this technique often exhibit high correctness ratios not only for the training patterns but also for novel patterns.

To Appear in the Proceedings of the International Joint Conference on Neural Networks 1990

[1]This work is sponsored by the Army Research Office under contract DAAG29-84-K-0060.

1 Introduction

There are two optimality measures of a trained feed-forward network for the given training patterns. One is the global error function shown below.

$$E = \sum_{p=1}^{P} \sum_{i=1}^{N} (t_{p_i} - o_{p_i})^2$$

where P is the number of training patterns and N is the number of output units.
t_{p_i} is the i-th target output of the p-th training pattern
while o_{p_i} is the i-th actual output of it.

The most popular training method for a feed-forward network, *back-propagation* based on the *Generalized Delta Rule*(GDR), uses this measure [RHW 86]. That is, it minimizes the global error through steepest gradient-descent in the $N+1$ dimensional error space where N is the number of connections[2] in the network. At each stage of the training, back-propagation always takes the direction where the slope is the *largest* from the current point in the error space so that it can reach a minimum as quickly as possible. In some cases, this leads the network to the global minimum in which the global error is the smallest in the whole error space. However, often it leads the network into a local minimum in which the global error is smaller than any neighboring points in the error space, but not the smallest in the entire space. Even though a local minimum is not the optimal solution, it may be accepted as a pseudo-optimal solution if it is very close to the global minimum. In practice, since the value of the global minimum is unknown, a small global error is considered to be sufficient.

Another measure for the optimality is the *correctness ratio* of the binary outputs for the given inputs of the training patterns. To measure the correctness ratio of a network, the network's actual outputs are converted into binary numbers through a threshold, often 0.5. Then, they are compared to the binary target values of the training patterns. If the binary output is the same as the target output, it is considered to be correct and the correctness ratio represents for what percentage of training patterns the network generates the correct binary outputs. The optimal solution by this measure is definitely that of 100% correctness ratio. Even though this is the measure that often really matters, unfortunately there is no known training method based on this measure.

Then, the question arises; are those two measures parallel? That is, does a low global error mean a high correctness ratio? In other words, if the global error is decreased by back-propagation, will the correctness ratio always go up? The answer is no. It is possible to have a network with a small global error that obtains a low correctness ratio or a network with a high global error but a high correctness ratio. Actually, it is not uncommon that a teacher trains a network up to a very small global error but a low correctness ratio and is puzzled about what is wrong.

[2]Here, biases are considered as the connections to a constant, 1.0.

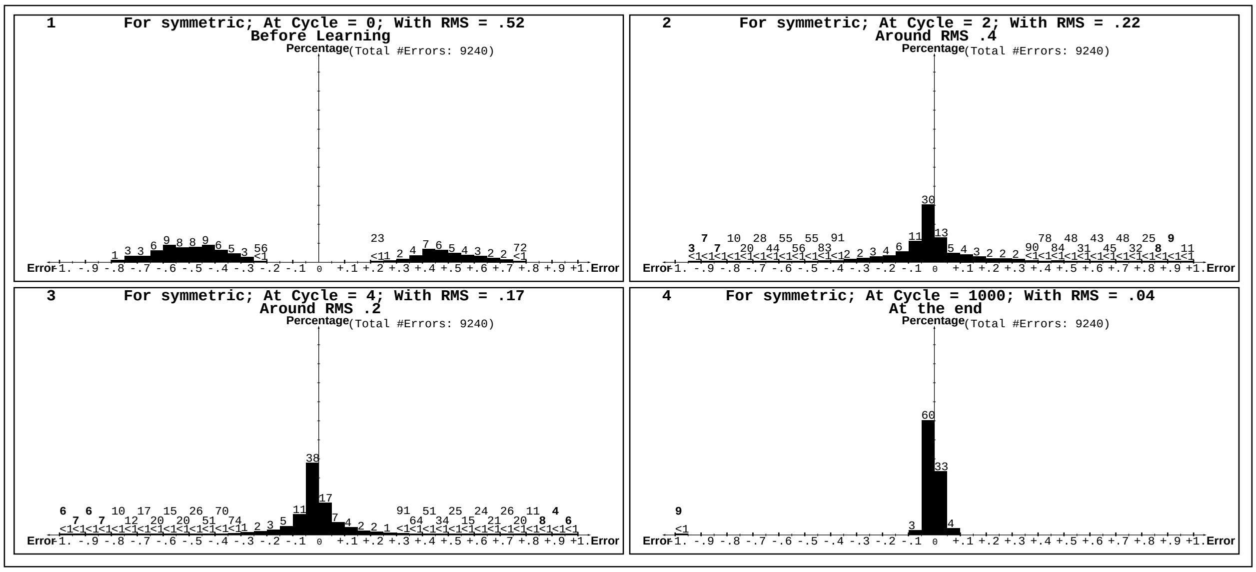

Figure 1: Error Distribution Graphs for Normal Back-Propagation

In the next section, it is shown why back-propagation based on GDR often leads to a low global error but a low correctness ratio.

2 Error Distribution Graph

Imagine a network in which most of the output errors (i.e., $t_{p_i} - o_{p_i}$) of training patterns are near 0.0 while the rest are near 1.0. Then, the correctness ratio may not be high even though the global error is fairly small. Imagine another network in which all the output errors of training patterns are all near 0.1. Then the correctness ratio will be 100% even though the global error is larger than the first one.

The first situation is well shown in the *Error Distribution Graphs* in Figure 1. In these graphs, the x-axis represents the groups of all errors by size, while y-axis represents the percentages of errors in some specific groups. The numbers on top of the bars are percentages while "< 1" means less than 1%. In this case, the numbers above "< 1" show the actual numbers of errors.

The graphs in this figure were snapshots taken during the learning by *auto-associative back-propagation of English/German sentence pairs* (see Section 5). The network is a 70-20-70 feed forward network and there are 132 training patterns resulting in the total of $132 \times 70 = 9240$ possible output errors.

The first graph shows the error distribution of the initial network and the second through fourth graphs show the error distributions during the learning phase. Specifically, the second graph shows the distribution of the errors when the root-mean-square (RMS) error per output unit[3] is around 0.4, while the third and last graph show those of the RMS errors of 0.2 and 0.04. respectively. Each graph shows how many cycles (or epochs) it took to achieve its RMS.

The comparison of the first and second graphs shows that even though the values of most errors have been reduced, there are some errors whose values have increased. Note that, in the beginning, there were no errors whose magnitudes were larger than 0.8. However, after 2 cycles, there are 70 such errors. This reveals the fact that some of the errors are reduced at the expense of others. The third graph shows an improved situation, but there are still many errors whose magnitudes are large. In the final graph, even though most of errors— 99.9% — are within 0.1, still some errors— about 0.01% — are near -1.0. The learning process converged around the 80th cycle and showed little further progress up to 1000 cycles.

The question at this point is whether the network has reached the global minimum, and achieved the best set of weights. Even though the global error value (0.04) above is very small, the final network above has correctness ratios of 93.2% for the training patterns and 50% for 12 novel patterns — unsatisfactory results. Unfortunately, this is a typical result of a back-propagation-based learning with a large number of training patterns [Allen 87]. Often, in cases like this, inappropriate topologies (i.e., too few or two many hidden units), insufficient training patterns, or a wrong combination of some parameters like *learning rate* and *momentum* are blamed [Hornik 89].

The real problem, however, is the fact that the back-propagation based on GDR does not use the correctness measure at all. In training feed-forward network through the back-propagation method, a set of training patterns act like *weak* constraints. They are called *weak* since some of them may not be satisfied at the end of the learning phase. Since the criterion for completing back-propagation depends only on the global error, some weight changes during the learning phase result in larger errors for some patterns even though the global error gets smaller.

The following section describes a technique which controls the reduction of errors in back-propagation so that the final network may achieve a high correctness ratio.

3 Descending Epsilon

To prevent the phenomenon described in the previous section, it is necessary for the back-propagation algorithm to attend to each individual error and make sure that the reductions of some errors do not increase others. However, this is impossible in most cases. Therefore, the proposed technique approximates it by balancing the errors. This is done by manipulating the optional parameter called *epsilon* which represents the margin of errors tolerable in the training. If it is used in learning, back-propagation is initiated only when the error of an output unit is larger than *epsilon*. The original intention for this parameter was to reduce

[3] $RMS = \sqrt{\frac{E}{N*O}}$ when E is the global error, N is the number of training patterns, and O is the number of output units. It is a measure independent of the number of patterns or the number of output units.

1. $epsilon = initial\_epsilon$
2. $correct = 0$
3. **loop** *p for each pattern*

    $no\_error = true$

    loop *i for each output unit*

        if $absolute(t_{p_i} - o_{p_i}) > epsilon$ **then** $error_i = (t_{p_i} - o_{p_i})$; $no\_error = false$
        otherwise $error_i = 0.0$.

    end_of_loop

    if no_error **then** $correct = correct + 1$.
    otherwise *back-propagate.*

   end_of_loop

4. **if** $correct =$ the number of patterns **then** *go to step 5.*
   otherwise *go to step 2.*
5. **if** $epsilon \leq final\_epsilon$ **then** *finish the training.*
   otherwise $epsilon = epsilon - epsilon\_step$; *go to step 2.*

Figure 2: Algorithm of Descending Epsilon

the training time by ignoring small errors.

The new technique is called *Descending Epsilon* and, as the name implies, it manipulates the value of *epsilon* in a decreasing pattern. The algorithm is presented in Figure 2.

One set of possible values for initial epsilon, epsilon step, and final epsilon used in the examples in this paper are 0.5, 0.1, and 0.1 respectively. Therefore, in the beginning, only the errors larger than 0.5 are back-propagated while the others are not. After all the errors are within 0.5, the value of *epsilon* is reduced to 0.4 and the network reduces the error within the new limit. The process goes until all the errors are within the value of *final epsilon*. This gradual error reduction prevents the network from reducing some errors too rapidly at the expense of increasing other errors. Instead, by balancing all the errors while the limit is decreased, the algorithm results in a network with a high correctness ratio while possibly achieving a low global error if the training finishes successfully.

Figure 3 shows the error distribution graphs during the learning phase of the translation task using the Descending Epsilon technique. Each graph shows that all the errors are contained within the current *epsilon* value. Even though the RMS error at the end, 0.05, is about the same as 0.04 obtained from normal back-propagation, the correctness ratio is 100%.

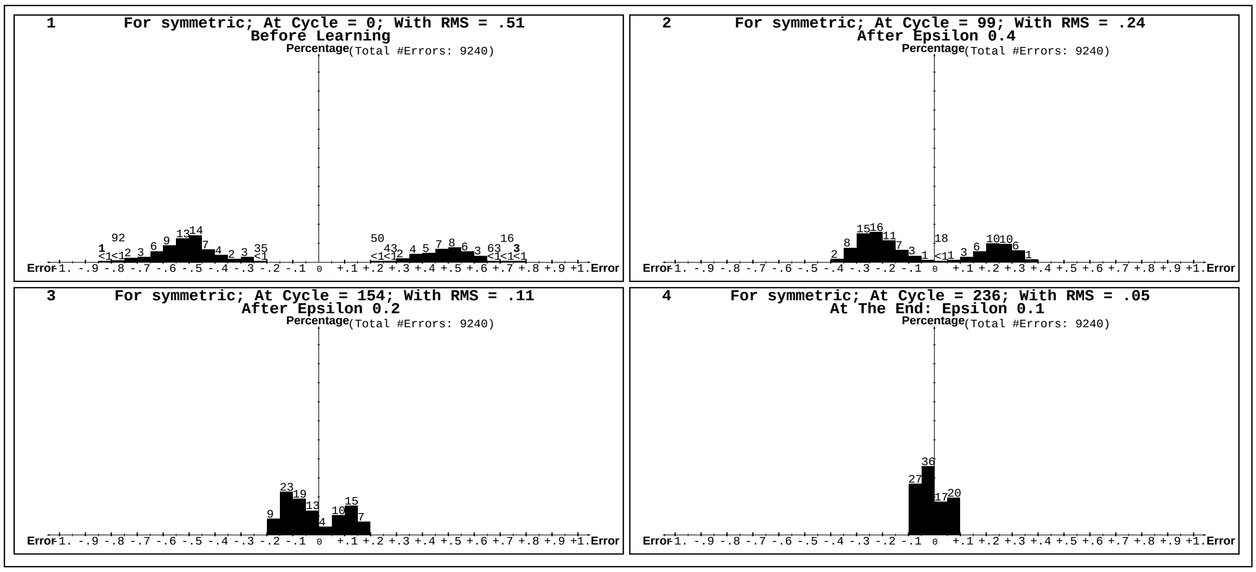

Figure 3: Error Distribution Graphs for Descending Epsilon

4 Generalization

One important goal of the learning is *generalization*: a high correctness ratio for novel patterns. Unfortunately, it is often the case that, a trained network has a high correctness ratio for the training patterns, but a low correctness ratio for novel patterns. Even in some cases, a lower global error and a higher correctness ratio for the training set result in a lower correctness ratio than a higher global error and a low correctness ratio.

The Descending Epsilon technique seems to work very well as far as generalization is concerned. In all our experiments, Descending Epsilon outperformed the normal back-propagation in generalization— usually 100% for novel patterns. Even though it is not clear why it works better in generalization, the balance among errors seems to play a big role. To make the errors balanced, the learning process settles to the weights which serve each training pattern equally well and, in doing so, it achieves the most abstract mapping between inputs and outputs. Conversely, if the network just memorizes the mappings of each individual pair of input and output, the errors are not likely to be balanced.

5 Performance

We studied the performance of the Descending Epsilon technique in three tasks. The first one is the 3-bit parity problem which is an extension of the famous XOR problem. It is known that 3-bit parity has more risk of falling into a local minimum than the XOR problem. This task is chosen to show how the Descending Epsilon technique works in complex error spaces.

The second task is to analyze simple English sentences into the values for cases such as *agent, object, action,* and *instrument.* Through a set of simple grammar rules, 360 sentences

task	ε	α	*topology*	*trials*	*finished*	*avg cyc*	*avg RMS*	*avg correct*	*generalization*
parity-3	0.3	0.9	3-3-1	25	22	2212	0.1	98.6	n.a.
case	0.1	0.9	40-10-20	5	5	144	0.03	100	77.6
symmetric	.05	0.9	70-20-70	10	0	600+	0.04	93	57.5

(a) Normal Back-Propagation.

task	ε	α	*topology*	*trials*	*finished*	*avg cyc*	*avg RMS*	*avg correct*	*generalization*
parity-3	0.4	0.9	3-3-1	25	25	3674	0.1	100	n.a.
case	0.2	0.9	40-10-20	5	5	192	0.04	100	83.2
symmetric	0.1	0.9	70-20-70	10	10	211	0.06	100	100

(b) Descending Epsilon.

ε: learning rate; α: momentum; *finished*: number of trials finished with 100% correctness within the maximum cycles below.
avg cyc: average cycle of finished trials; *avg RMS, avg correct*: averages of all trials.
maximum cycle: 10000 for parity, 400 for case, and 600 for symmetric.

Figure 4: Parameter Values and Performances

have been generated. Among them, 90 (25%) randomly chosen sentences are used in training while the other 270 are used as novel patterns. Each sentence may have up to 8 words, each of which is represented by 5 input units. Each value for a role in the output also takes 5 output units.

The last task is the auto-associative German/English symmetric translation mentioned in previous sections. In this task, the inputs and outputs are each pairs of English and German sentences. Some of the inputs have only English or German parts while the other parts are left blank. The outputs are always the complete pairs. Therefore, after training this auto-associative memory, the network should be able to generate a pair of English and German sentences given only one of them — a *symmetric translation*.

Figure 4 tabulates the values of various parameters and performances of the normal back-propagation method, and the Descending Epsilon technique. As this figure shows, the descending epsilon technique outperforms the normal back-propagation method in the number of finished trials, the correctness ratio, and generalization. One drawback of the Descending Epsilon technique is that it generally takes more cycles than the back-propagation. Also, the RMS of the Descending Epsilon technique is larger than that of the normal back-propagation. However, the RMS can be reduced to a smaller value by reducing the value of the final epsilon, if it is necessary.

6 Discussion

While training patterns are treated as *weak* constraints in normal back-propagation, they are treated as *strong* constraints in Descending Epsilon. Therefore, in reducing *epsilon* step by step, all those constraints should be satisfied. This requirement has two unfortunate side effects.

First, the Descending Epsilon technique cannot learn from training patterns with a contradiction — i.e. two or more training patterns have identical inputs but different outputs. These training patterns are not 100% satisfiable and the Descending Epsilon technique will not be able to reduce the value of *epsilon* beyond a limit.

Secondly, the Descending Epsilon technique takes more cycles than normal back-propagation since it requires more rigorous conditions for a back-propagation to occur and it has do it many times for different values of *epsilon.* Both of these problems may be eased by relaxing the condition for decreasing the value of *epsilon* by requiring that 90% of the errors fall within the value of *epsilon* and the other errors be within the previous value of *epsilon.* However, such a relaxation might degrade the correctness ratio and generalization.

Despite these problems, our experiments show Descending Epsilon improves back-propagation with respect to satisfactory completion of training, higher correctness ratios, and improved generalization to novel examples.

References

[Allen 87] R. Allen, Several Studies on Natural Language and Back-Propagation, *IEEE International Conference on Neural Networks*, 1987.

[Hornik 89] K. Hornik, Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, Vol 2, pp. 359-366, 1989.

[RHW 86] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Representations by Error Propagation. In D.E. Rumelhart, J.L. McClelland and the PDP Research Group (Eds.), *Parallel Distributed Processing*, Vol 1, pp. 318-362, MIT Press, 1986.