(International Journal of Neural Systems, 5(1): 67-75, 1994)

ACCELERATED LEARNING BY ACTIVE EXAMPLE SELECTION

Byoung-Tak Zhang[*]
AI Division, Institute for Computer Science, University of Bonn
Römerstraße 164, D-53117 Bonn, Germany

Abstract

Much previous work on training multilayer neural networks has attempted to speed up the back-propagation algorithm using more sophisticated weight modification rules, whereby all the given training examples are used in a random or predetermined sequence. In this paper we investigate an alternative approach in which the learning proceeds on an increasing number of selected training examples, starting with a small training set. We derive a measure of criticality of examples and present an incremental learning algorithm that uses this measure to select a critical subset of given examples for solving the particular task. Our experimental results suggest that the method can significantly improve training speed and generalization performance in many real applications of neural networks. This method can be used in conjunction with other variations of gradient descent algorithms.

1 Introduction

One of the most widely used methods for training multilayer feedforward neural networks is the error back-propagation algorithm [1]. This algorithm is a gradient-based method, and suffers from low convergence rate. Most researchers have attempted to speed up the back-propagation procedure using sophisticated weight modification rules. Some have tried limited modifications on the first-order method by adapting the step size dynamically, for example [2]. Others have used second-order derivatives, as in Newton's method, which requires an expensive calculation or estimation of the Hessian matrix (see [3] for a review of recent developments). While these techniques are useful for a variety of problems, there are other additional factors which influence the learning speed and generalization ability of the networks.

One of them is the nature and size of the training set. While there is no guarantee that the generalization performance is improved by increasing the training set size [4], the training time increases as the number of examples increases, In general, one should choose those examples which are most likely to help the network solve the problem. For classification problems, these examples are the border patterns, i.e., the patterns that lie closest to the separating hyperplanes. Several studies have shown that a network trained on border patterns generalizes better than a network trained on the same number of examples chosen at random [5, 6]. However, these studies have been limited to binary problems in which the example space is either small enough to be examined exhaustively, or the problem is simple enough to analyze. General methods for selecting "critical" ex-

[*]Present address: German National Research Center for Computer Science (GMD), Schloß Birlinghoven, D-53757 Sankt Augustin, Germany, e-mail: zhang@gmd.de

amples have not yet been suggested.

In this paper we propose a criterion for selecting critical examples. We also present an efficient method for selecting examples and scheduling their training order based on this criterion. In this new learning scheme the network is trained on incrementally selected examples, rather than on all the available data. In Section 2 the idea of incremental learning will be described in more detail. In Section 3 we derive a measure of criticality and describe the algorithm. In Section 4 the convergence characteristics of the algorithm are shown experimentally in the context of three representative tasks. This is followed by the conclusion in Section 5.

2 Training Multilayer Nets

In multilayer neural networks, the external inputs are presented to the input layer which is fed forward via one or more layers of hidden units to the output layer. In each layer, the output value a_i of unit i is determined by a sigmoid function of the weighted sum of inputs a_j from the previous layer:

$$a_i = \frac{1}{1 + \exp\left(-\left(\sum_{j \in R(i)} w_{ij} a_j + \theta_i\right)\right)} \qquad (1)$$

where $R(i)$ is the index set of predecessor units of i, and θ_i is the bias of i.

The networks learn on the training set of input-ouput pairs:

$$D_N = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}. \qquad (2)$$

The main objective is to achieve a good generalization performance, i.e. the trained network should produce acceptable responses to novel inputs. This problem is a nonlinear optimization problem whose goal is to minimize the additive error functional

$$E(D_N|\mathbf{w}) \;=\; \sum_{p=1}^{N} E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) \qquad (3)$$

where p is the index of examples. A common choice for the error measure $E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w})$ is the sum of squared errors between the desired output $\mathbf{y}_p$ and the actual output $f(\mathbf{x}_p; \mathbf{w})$ of the network, that is $E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) = ||\mathbf{y}_p - f(\mathbf{x}_p; \mathbf{w}_p)||^2$.

A popular learning method is the back-propagation algorithm [1] in which the weights are updated using the recursion (weight modification rule)

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \epsilon \nabla(\mathbf{w}_t) \qquad (4)$$

where $\nabla(\mathbf{w}_t)$ is the t-th estimate of the error gradient with respect to the weights, and the parameter ϵ is the step size. Depending on the method of estimating the error gradient, there are two main training methods: batch and online. The batch training uses the equation $\nabla(\mathbf{w}_t) = \frac{1}{N}\sum_{p=1}^{N} \nabla E_p|_{\mathbf{w}=\mathbf{w}_t}$ while the online training uses $\nabla(\mathbf{w}_t) = \nabla E_p|_{\mathbf{w}=\mathbf{w}_t}$, where $\nabla E_p|_{\mathbf{w}=\mathbf{w}_t}$ is an estimate of the gradient of $E_p = E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w})$ at $\mathbf{w}_t$. Note that the complete training set is repeatedly presented to update the weights during the training process in both training methods.

The main feature of our method is that we do not train the network on the entire training set of size N from the outset. Rather, we start the learning with a small subset of size $N_0 < N$ of given examples and increase the training set incrementally. That is, rather than attempting to minimize the function (3) directly, we try to minimize a set of objective functions

$$E(D_N|\mathbf{w}) = E(D_{N_0}|\mathbf{w}) + \cdots + E(D_{N_s}|\mathbf{w}) \qquad (5)$$

from left to right sequentially. Here N_k is the kth size of the training set satisfying the relation

$$N_0 < N_1 < \cdots < N_s = N. \qquad (6)$$

The advantage of the incremental learning will be illustrated in a simple example. Suppose that we want to approximate the function f depicted in Figure 1. Let the data points $r_p = (x_p, y_p)$, $p = 1, ..., 25$, be given.

When the back-propagation, irrespective of online/batch training, is used, the algorithm tries to satisfy from the start all the constraints described by all the given examples. Because some examples are contradictory and some are redundant, it usually takes a long time to evolve a reasonable approximation curve. A typical progress of this *non-incremental learning* is illustrated in Figure 2.

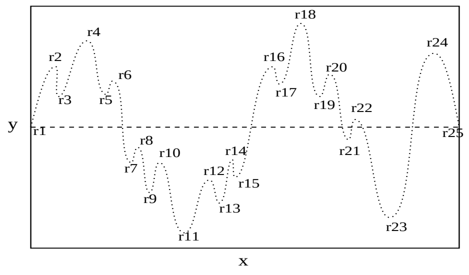

Figure 1: The desired function and data points

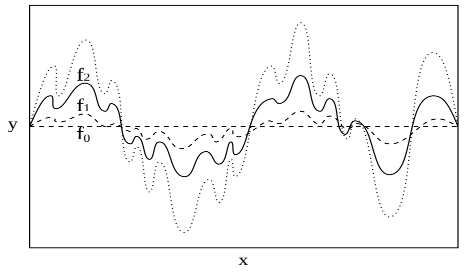

Figure 2: A non-incremental learning

On the other hand if we use a small number of examples to train the network, the training will converge very fast. Although the first approximation may not be satisfactory, we can use this knowledge to select the next set of examples which efficiently improve the current approximation. A typical progress of this *incremental learning* is illustrated in Figure 3.

The comparison suggests that the incremental learning can achieve a reasonable performance using only a small subset of examples. The question is how to select "good" examples. To answer this question we need a measure of goodness.

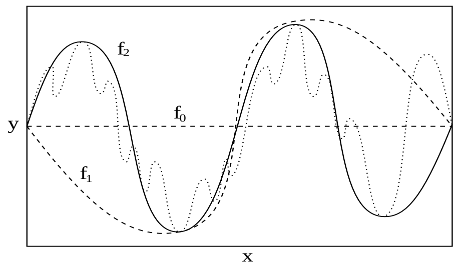

Figure 3: An incremental learning

3 Learning by Selecting Critical Examples

Assume that we have trained the network on the data set D_N. The network can be viewed as a model of the data [7, 8] and we assign the probability of the data as a function of $\mathbf{w}$:

$$P_N(\mathbf{w}) = P(D_N|\mathbf{w}) = \prod_{p=1}^{N} P(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) \qquad (7)$$

where

$$P(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) = \frac{\exp(-\beta E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}))}{Z(\beta)} \qquad (8)$$

Here β is a positive constant which determines the sensitivity of the probability to the error value, i.e. a measure of the presumed noise included in $\mathbf{y}_p$ and $Z(\beta) = \int \exp(-\beta E(\mathbf{y}|\mathbf{x}, \mathbf{w}))d\mathbf{y}$ is a normalizing constant. In the case of the quadratic error function the resulting $P(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w})$ is the Gaussian distribution

$$P(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w})}{2\sigma^2}\right\} \qquad (9)$$

with $\beta = \frac{1}{2\sigma^2}$ and σ^2 is the variance of the noise included in $\mathbf{y}_p$.

Now we want to select an $(N+1)$-th example to improve the performance of the network. This example can be selected by measuring the information gain of the network when we receive the new example $\mathbf{y}_{N+1}$. Let the probability distributions of the parameters before and after we receive the datum $\mathbf{y}_{N+1}$ be $P_N(\mathbf{w})$ and $P_{N+1}(\mathbf{w})$. According to information theory [9], the mean information for discrimination between $P_N(\mathbf{w})$ and $P_{N+1}(\mathbf{w})$ is given by

$$I(P_{N+1}, P_N) = \int P_{N+1}(\mathbf{w}) \ln \frac{P_{N+1}(\mathbf{w})}{P_N(\mathbf{w})} d\mathbf{w} \qquad (10)$$

The greater the value of $I(P_{N+1}, P_N)$, the less resemblance between the two distributions, and the more information is gained about $\mathbf{w}$. Given a fixed distribution $P_N(\mathbf{w})$, the maximum information gain is thus achieved by maximizing the difference of $P_{N+1}(\mathbf{w})$ from $P_N(\mathbf{w})$.

According to the assumed relation (8) between the likelihood and the error of the data, we can maximize this difference by selecting the example whose addition to D_N leads to the greatest $E(D_{N+1}|\mathbf{w})$ with the current parameters $\mathbf{w}$. Hence the example that maximizes

$$\Delta E_{N+1} = E(D_{N+1}|\mathbf{w}) - E(D_N|\mathbf{w}) \qquad (11)$$

should be included in the training set. The training method is also assumed to be able to reduce the error to a desired accuracy.

This example can be found by presenting the candidates $\mathbf{x}_p$ to the partially trained network and computing their errors $E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w})$ and selecting the kth example satisfying

$$E(\mathbf{y}_k|\mathbf{x}_k, \mathbf{w}) = \max_p \{ E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) \}. \qquad (12)$$

In situations where we have to select examples from a large data set, we need only the relative error of each example, not the value itself. In general, we define the *criticality* of an example $(\mathbf{x}_p, \mathbf{y}_p)$ as

$$\begin{aligned} e_{\mathbf{w}}(\mathbf{x}_p) &= \frac{E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w})}{\dim(\mathbf{y}_p)} \\ &= \frac{1}{m}\sum_{i=1}^{m}(y_{pi} - f_i(\mathbf{x}_p; \mathbf{w}))^2 \qquad (13) \end{aligned}$$

where f_i denotes the activation value of the ith output unit and m, the number of the output units, is used to normalize the value. The measure $e_{\mathbf{w}}(\mathbf{x}_p)$ has a value $0 \le e_{\mathbf{w}}(\mathbf{x}_p) \le 1$ if the sigmoid output function $f_i(s) = \frac{1}{1+e^{-s}}$ is used. By definition an example is the most critical if it causes the largest error. (Note that in the previous work [10, 11] we have used the square root of the criticality, called *interestingness*).

Now that we can describe the algorithm. The *training set*, denoted by D, is defined to be the set of data that are currently used to train the network. The rest of the given examples is called the *candidate set* and denoted by C. The training set is initialized with a small set of randomly chosen *seed examples*. The candidate set is initialized with the rest of the given examples. The learning process consists of iteration of the training of the network and the selection of the training set.

In the training phase, the connection weights of the network are updated using only the examples in the training set D. The weights of the network are initialized randomly with values from the interval $-\omega \le w_{ij} \le +\omega$. Each time an example is presented, the weights are modified by

$$\Delta w_{ij}(t) = -\epsilon \frac{\partial E_p}{\partial w_{ij}(t)} + \eta \Delta w_{ij}(t-1) \qquad (14)$$

where $E_p = E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w})$ and $\frac{\partial E_p}{\partial w_{ij}}$ is approximated by the error back-propagation procedure described in the previous section—although we are using an online training here, it is also possible to use a batch training or any other weight modification rule. In the above equation, ϵ and η are the step size and the momentum factor, respectively.

The weight adjustment is repeated until the sum of errors of the current training set is reduced to a specified performance level, i.e.

$$E(D|\mathbf{w}) \le \kappa \qquad (15)$$

where κ is defined as

$$\kappa = \frac{h(n+m)}{\tau} \qquad (16)$$

Here n, m and h are the number of input, output and hidden units, respectively. τ is a constant determining the allowable error tolerance per connection. This is based on the fact that the learning capacity is proportional to the total number of adjustable connections in the network [4]. In a variety of discrete problems the τ value of $100 \le \tau \le 200$ turned out to be reasonable.

In the selection phase, examples in the current candidate set C are tested by computing the criticality $e_{\mathbf{w}}(\mathbf{x}_q)$ of the examples (Eq. 13) with respect to the current model $\mathbf{w}$. If $|C| \ge \lambda$, then a λ number of the most critical examples $(\mathbf{x}_q, \mathbf{y}_q)$ are selected from the candidate set C and moved to the training set D:

$$\begin{aligned} D &\leftarrow D \cup \{(\mathbf{x}_q, \mathbf{y}_q)\} \qquad (17) \\ C &\leftarrow C - \{(\mathbf{x}_q, \mathbf{y}_q)\} \qquad (18) \end{aligned}$$

Otherwise, all examples in C are selected into D. Using the expanded training set D, the next cycle of training and selection is done. The iteration is repeated until the specified performance level is achieved or the candidate set C is empty. Notice that the network has generalized correctly to the candidate set C if the algorithm halts with nonempty C.

We now analyze the behavior of the algorithm. Let N denote the size of given data set. Let N_s denote the size of training set after the sth selection step, then

$$N_s = N_0 + \lambda s \tag{19}$$

where N_0 is the number of seed examples and λ is the increment of training set size. The maximum number of selection steps is given by

$$s_m = \left\lceil \frac{N - N_0}{\lambda} \right\rceil \tag{20}$$

where $\lceil x \rceil$ is the ceiling function. Let D_{N_s} denote the training set after the sth selection stage.

The objective function optimized in the selective incremental learning is given by

$$\begin{aligned} &E_{SEL}(D_N|\mathbf{w}) \\ &= \sum_{s=0}^{s_m} E(D_{N_s}|\mathbf{w}) \\ &= E(D_{N_0}|\mathbf{w}) + E(D_{N_1}|\mathbf{w}) + \cdots + E(D_{N_{s_m}}|\mathbf{w}) \\ &= \sum_{p=1}^{N_0} E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) + \cdots + \sum_{p=1}^{N_{s_m}} E(\mathbf{y}_p|\mathbf{x}_p, \mathbf{w}) \end{aligned} \tag{21}$$

with the relation

$$D_{N_0} \subset D_{N_1} \subset \cdots \subset D_{N_{s_m}} = D_N. \tag{22}$$

Notice that minimizing the local objective functions $D_{N_i}, i = 0, 1, ..., s-1$ leads also to minimization of $D_{N_k}, k = i, i+1, ..., s$. Comparing this objective function with Eq. (3) we can see that the SEL algorithm minimizes the usual objective function eventually, since $N_{s_m} = N$. Also observe that the incremental learning reduces to the non-incremental one such as usual back-propagation if the entire data set is used as the training set from the start. In this sense the selective incremental learning procedure is a generalization of the back-propagation procedure.

We now consider the computational requirements of the algorithm. Let t_s denote the total number of sweeps through the training set D_{N_s} in the sth learning stage. The total number of weight modifications for this training set is given by

$$T(D_{N_s}) = \sum_{t=1}^{t_s} \sum_{p=1}^{N_s} K \tag{23}$$

where K is the number of adjustable weights in the network. Using this equation, the total time of the selective learning is proportional to

$$T_{SEL}(D_N) = \sum_{s=0}^{s_m} T(D_{N_s}) = \sum_{s=0}^{s_m} \sum_{t=1}^{t_s} \sum_{p=1}^{N_s} K \tag{24}$$

The costs for computing the criticality is neglected since the total number of selections is much smaller than the total number of training epochs, i.e. $s_m \ll \sum_{s=0}^{s_m} t_s$, and the criticality computation involves only a forward pass for the candidate examples. On the other hand, the total training time of the usual back-propagation procedure is given by

$$T_{BP}(D_N) = \sum_{t=1}^{t_{max}} \sum_{p=1}^{N} K \tag{25}$$

To compare the learning time (24) and (25), we consider the growth rate, r_s, of the sth training set:

$$r_s = \frac{N_s}{N_{s-1}} = \frac{N_0 + \lambda s}{N_0 + \lambda(s-1)} \tag{26}$$

The rate r_s is a monotonically decreasing function of s ($r_s \to 1$ as $s \to s_m$).

Observe that the large growth rate in the early stages means that the probability of any additional examples for changing the current model is high, and that it is expected a large number of training sweeps t_s is required. In spite of this, the learning time will not be long, because N_s is small. As learning proceeds r_s approaches 1, meaning the relative information gain decreases and the probability of any additional examples for changing the model reduces. In spite of a large N_s, the training time will also not be long, since t_s is small. Consequently, one can expect the selective incremental learning to converge faster than the usual back-propagation training. We will confirm this by experiments in the next section.

4 Convergence Characteristics

The method has been applied to several discrete and continuous problems. In this section we compare the convergence characteristics of the incremental method with the conventional backpropagation training in the context of three representative tasks. A more detailed description can be found in [12].

4.1 Majority Function

Majority is a binary mapping problem from an input space $X = \{0,1\}^n$ to the output space $Y = \{0,1\}$. This function returns 1 if the majority of the inputs is 1, and returns 0, otherwise. For an odd n, one half of the examples are positive (output 1) and the other half are negative (output 0). This property allows an exact analysis of the example selection behavior of the algorithm and the results can be verified objectively.

# input	λ	N	N_{min}	N_{avg}
5	2	20	12	18.7
	2	25	10	17.9
	2	30	8	16.8
15	5	100	67	98.5
	5	150	67	119.8
	5	200	52	116.0
25	10	300	152	234.0
	10	400	152	232.7
	10	500	132	198.8

Table 1: Training set size for perfect generalization

Table 1 shows the training set sizes used for perfect learning of majority functions with varying data size N. Generalization was evaluated by a test set of 1000 random examples drawn independently from the same distribution. Observe that the incremental learning can solve the 5-input majority problem using only 8 training examples, although there are ${}_5C_3 + {}_5C_2 = 20$ border patterns. For the cases of $n = 15$ and 25, the size of the training set sufficient for correct generalization was much lower than the theoretical bound (all border patterns). The selected examples for the 5-majority are shown in Figure 4, in which the numbers in the squares indicate the selection sequence of examples. The results suggest that not all border patterns are necessary for perfect generalization, and the selective incremental learning can find such subset.

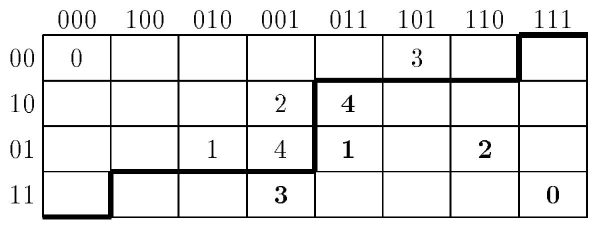

Figure 4: Selected examples for 5-majority

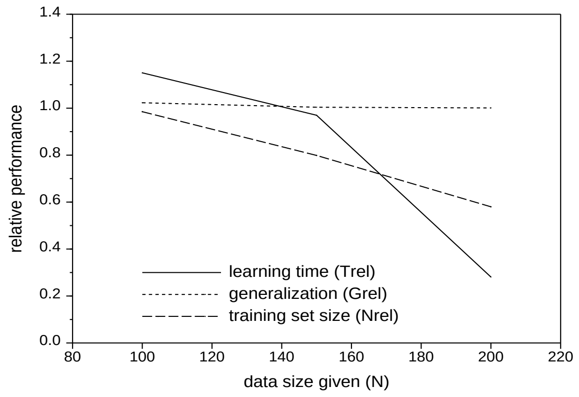

Figure 5: Relative performance for 15-majority

Figure 5 depicts the relative performance of the selective incremental learning method compared with the non-incremental learning, i.e. $T_{rel} = T_{SEL}/T_{BP}$ etc., for 15-majority. The learning parameters in the adaptation phase of both algorithms were the same. In the figure, it can be seen that as the size N of the initially known data set grows, the relative learning efficiency of the incremental learning is superior than the non-incremental one. This is because SEL tends to take a smaller fraction of the given data as the known data size increases. The results suggest that the incremental learning is especially useful when the data is large.

4.2 Nonlinear Continuous Mapping

The majority function discussed above is a discrete classification problem. The performance in continuous problems will be demonstrated on a simple function approximation task. For the graphical verification of the performance, we chose the function

$$z(x,y) = \frac{1}{2}\left(x^2 + y^2\right) \tag{27}$$

with inputs x and y from the interval $[-1, +1]$. In this domain the output z has a range of $[0, +1]$. We used a network of 2-5-1 architecture. The x, y, and z values were encoded to use activation values from the interval $[0.1, 0.9]$ on the input and output units.

A total of $N = 21 \times 21 = 441$ data points (x, y) and associated z values are given initially. Learning was started with two seed examples using a randomly chosen weights from the interval $[-0.1, +0.1]$. In each selection step, 20 new examples were selected to the training set. The intermediate results after training at selection steps of $s = 0, 1, 3, 7, 11, 15$ are shown in Figure 6. Here one can see how the incremental method searches the input space and in which strategy it tries to learn the unkown function; the algorithm learns examples from the regions where the current approximation error is large. The selective learning network approximated the desired function 98% correctly (within 0.1 of the target value) using only 68% of given training examples in 15 selection steps. Similar reduction of the training set size and the improvement in training speed have also been observed in robot arm control tasks [12].

4.3 Handwritten Digit Recognition

The performance of the algorithm was evaluated on the recognition of handwritten digits. The data was collected from 10 persons. Each person wrote by hand 680 digits $(0 - 9)$ on two sheets of paper. The digits were read by a scanner to be converted in 15×10 bitmaps. One sheet of each writer was chosen randomly to build the data set (3400 digits). The other sheets were collected to build the test set (3400 digits).

We used a feed-forward network with a 150-40-10 architecture. 10 digits of 0 to 9 were randomly chosen from the data set to initialize the training set. In the selection phase, 200 new examples were chosen to expand the training set.

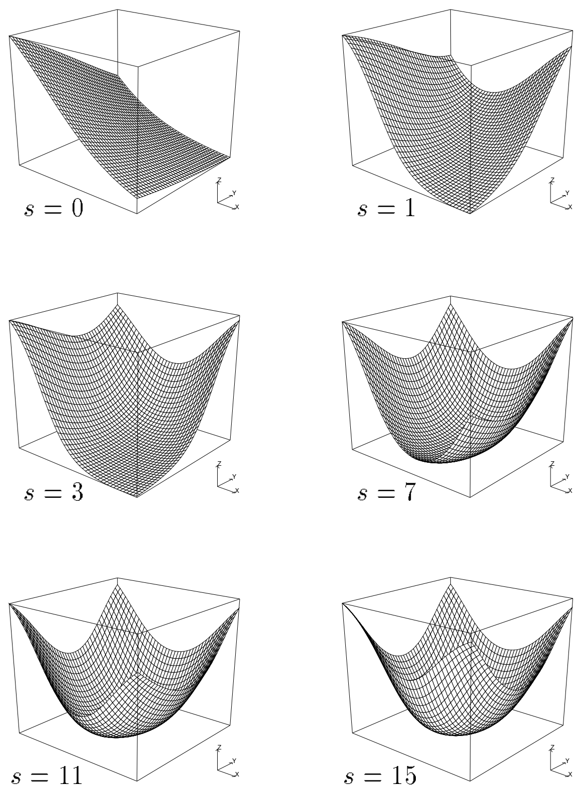

Figure 6: Learning results at different stages

For comparison, we trained the same size of network using an online back-propagation procedure (BP). All the learning parameters were the same as those used in the adaptation phase of SEL. Figure 7 shows the average performance of both learning procedures for ten runs each, as a function of the training time measured in cpu minutes on a Sparcstation. The SEL method converged to a 99% performance, on average, four times faster than the simple BP training. For both algorithms the maximum generalization accuracy was about 85% for the data used.

Notice that although BP improves its performance relatively fast at the early stages, and converges at the end, the convergence was unstable and therefore very slow. The simple BP method seems to concentrate too much on the statistical

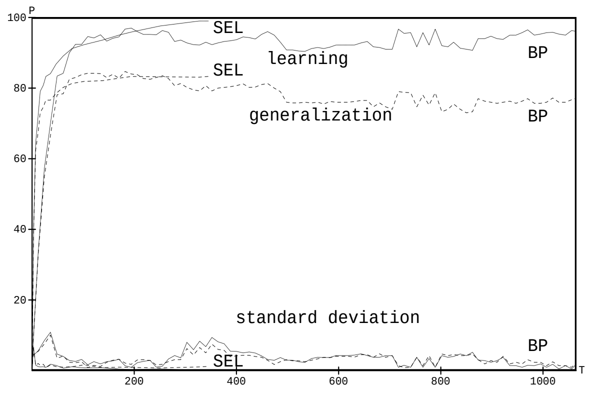

Figure 7: Comparison of learning curves

average of all the available patterns, which occasionally shows good performance but frequently gets stuck in local optima. Further training usually leads to overfitting, and an early stopping or pruning must be used to resolve this problem [13].

On the other hand, the SEL method improves its performance monotonically and finally converges faster than BP, achieving its maximum generalization accuracy at convergence time. This suggests that SEL may be used as a method of avoiding local minima and overfitting problems. The monotonicity of generalization in SEL is also useful for deciding if more data is needed to improve the final performance.

We also note that the SEL performance shows a very rapid improvement (region A) followed by a slow but steady increase afterwards (region B). The training set size between region A and B ranges approximately 500 to 800. This indicates that the algorithm has found a moderately critical subset around this size. The standard deviation suggests that the performance of SEL is robust against the initial weight values and the seed examples.

5 Conclusion

There are several theoretical studies on the minimal size of the training set. For example, Baum and Haussler [14] show that, to achieve a fraction of $1-\epsilon$ generalization, one needs $m \geq O\left(\frac{W}{\epsilon} log \frac{N}{\epsilon}\right)$ random examples to train a feedforward network of linear threshold functions with N neuronal units and W synaptic weights. Such a rule is theoretically interesting, but is not very useful in practice since this is an worst case analysis. Another study [6] suggests that the entire set of border patterns must be presented to guarantee a perfect generalization.

However, the inherent generalization ability of neural networks does not require all the border patterns. This was confirmed by the experimental results on the majority function where a perfect generalization was achieved using only a small subset of the border patterns. On the other hand, when the input space is prohibitively large or infinite, such as in pattern recognition or continuous function approximation, one can not expect a perfect generalization. However, even in these cases one can select data to improve the convergence. This is supported by the results on the digit recognition and the approximation of the simple continuous function where a small subset of the given data was sufficient to achieve the same generalization performance as the original data.

The criticality measure for selecting examples is related to the work of MacKay [8] who also studied active sampling of examples. Plutowski and White [15] describes a similar approach to finding concise training sets from clean data sets. Both works require calculation or approximation of Hessian which could be quite expensive. The statistical interpretation of network learning has been studied by Tishby *et al.* [7]. The analysis enables the evaluation of the probability of a correct prediction of an independent example, after training the network on a given training set. However, their average case analysis did not propose methods of how to select the critical examples efficiently, nor how to use the examples to speed up or improve the generalization.

In contrast, we presented a general method to selecting critical examples for solving a particular task. The method does not need to calculate second derivatives. Our experimental results show that the selective incremental learning finds and uses only a critical subset of given examples, which leads to a considerable enhancement in training speed and generalization performance. We expect the incremental learning will be more efficient than a non-incremental one if the given data contains enough information for correct generalization with small noise. The learning behavior of this method on very noisy data, such as time series data [16], remains to be studied. In a previous study [12], we found that the incremental learning can be more expensive than a simple backpropagation if the available data is too sparse or already critical. This is because the SEL method eventually uses all the examples if there is no redundancy in the data set. In fact, unless we have an oracle telling us which are redundant or noisy data, the best learning starategy would be to use all the examples.

Selecting patterns with large squared error raises another question: Doesn't the sampling method forster overfitting? A related point is that starting with the outliers might send the network off to a region very far away from a good solution and likely to get stuck in local minima. To answer these questions one should notice that training and selection are interleaved in the SEL method, instead of just repeating training or selection separately. If an outlier is selected, the successive training stage will learn this outlier to some extent. However, the subseqent selections would correct this mistake by choosing more regular examples because the outlier has biased the network and this time the regular examples will have larger errors than the outliers. Thus the method may contain some outliers, but if a sufficiently large number of selection steps are used, these problems will not occur. The low variance among the various learning trials using different random seed examples in digit recognition experiments is an example that confirms this conclusion.

The current work can be extended in two directions. First, the criticality measure may be used for further purposes. Although it was originally developed for selecting examples, the measure can also be used to generate new useful examples. Preliminary results on this point were reported in [11], where new examples are created by recombining existing critical examples through genetic operators. Secondly, one can combine the idea of incremental data selection with architecture selection algorithms [17], especially with the constructive algorithms. In a series of experiments reported in [18] we were able to optimize the number of hidden units of a feed-forward network using only a critical subset of examples, instead of using the whole data set from the start, and thereby improved the optimization efficiency.

Acknowledgement

The author would like to thank A. Weigend for helpful discussions.

References

[1] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, Vol. I, eds. D. E. Rumelhart and J. L. McClelland (MIT Press, 1986) pp. 318-362.

[2] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks* **1**, 295–307 (1988).

[3] R. Battiti, "First- and second-order methods for learning between steepest descent and Newton's method," *Neural Computation* **4**, 141–166 (1992).

[4] Y. S. Abu-Mostafa, "The Vapnik-Chervonenkis dimension: information versus complexity in learning," *Neural Computation* **1**, 312–317 (1989).

[5] S. Ahmad and G. Tesauro, "Scaling and generalization in neural networks: a case study," in *Proc. 1988 Connec. Models Summer School* (Morgan Kaufmann, San Mateo, 1989) pp. 3-10.

[6] K. A. Huyser and M. A. Horowitz, "Generalization in connectionist networks that realize Boolean functions," in *Proc. 1988 Connec. Models Summer School* (Morgan Kaufmann, 1989) pp. 191-200.

[7] N. Tishby, E. Levin and S. A. Solla, "Consistent inference of probabilities in layered networks: predictions and generaliation," *Proc. IJCNN*, Vol. II (IEEE, 1989) pp. 403–409.

[8] D. J. C. MacKay, *Bayesian Methods for Adaptive Models*, Ph.D. thesis, Caltech, Pasadena, CA. (1992).

[9] S. Kullback, *Information Theory and Statistics*, (Wiley, New York, 1959).

[10] B. T. Zhang and G. Veenker, "Focused incremental learning for improved generalization with reduced training sets," in *Artificial Neural Networks: Proc. ICANN*, Vol. I, eds. T. Kohonen *et al.* (Elsevier, 1991) pp. 227-232.

[11] B. T. Zhang and G. Veenker, "Neural networks that teach themselves through genetic discovery of novel examples," in *Proc. IJCNN*, Vol. I, (IEEE, 1991) pp. 690-695.

[12] B. T. Zhang, *Learning by Genetic Neural Evolution*, Ph.D. thesis in German, Univ. of Bonn (ISBN 3-929037-16-5, Infix-Verlag, Sankt Augustin, 1992).

[13] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Predicting the future: A connectionist approach," *Int'l J. Neural Systems*, **1**, 193–209 (1990).

[14] E. B. Baum and D. Haussler, "What size net gives valid generalization?", in *Advances in Neural Inform. Pro. 1*, ed. D. S. Touretzky (Morgan Kaufmann, 1989) pp. 81–90.

[15] M. Plutowski and H. White, "Selecting concise training sets from clean data," *IEEE Trans. Neural Networks*, **4**, 305–318 (1993).

[16] A. S. Weigend, N. A. Gershenfeld, Eds. *Time Series Prediction*, Addison-Wesley, 1993.

[17] B. T. Zhang and H. Mühlenbein, "Genetic programming of minimal neural nets using Occam's razor," *Proc. Int. Conf. Genetic Algorithms* (Morgan Kaufmann, San Mateo, 1993) pp. 342–349.

[18] B. T. Zhang, "Self-development Learning: Constructing optimal size neural networks via incremental data selection," Tech. Rep. No. 768, German National Research Center for Computer Science, (Sankt Augustin, 1993).