The Dynamic Pattern Selection Algorithm: Effective Training and Controlled Generalization of Backpropagation Neural Networks

A. Röbel
Technische Universität Berlin[1]

March 4, 1994

[1]Institut für Angewandte Informatik, FG Informatik in Natur- und Ingenieurwissenschaften

Abstract

In the following report the problem of selecting proper training sets for neural network time series prediction or function approximation is addressed. As a result of analyzing the relation between approximation and generalization, a new measure, the generalization factor is introduced. Using this factor and cross validation a new algorithm, the *dynamic pattern selection*, is developed.

Dynamically selecting the training patterns during training establishes the possibility of controlling the generalization properties of the neural net. As a consequence of the proposed selection criterion, the generalization error is limited to the training error. As an additional benefit, the practical problem of selecting a concise training set out of known data is likewise solved.

By employing two time series prediction tasks, the results for dynamic pattern selection training and for fixed training sets are compared. The favorable properties of the dynamic pattern selection, namely lower computational expense and control of generalization, are demonstrated.

This report describes a revised version of the algorithm introduced in [Röbel, 1992].

Contents

1 Introduction

Since the formulation of the backpropagation algorithm by Rumelhart, Hinton and Williams [1986] there has been a steadily growing interest on artificial neural networks. Due to some vague analogies between neural networks and the biological nervous systems it has been expected that successful applications of neural networks in fields like Classification, Pattern Recognition, Nonlinear Signal Processing or Control, all areas in which the known technical solutions remain far behind the performance of the biological systems, will be possible in the near future. Concerning the theoretical investigations of neural networks, there exists encouraging results supporting these expectations.

However, the experiences concerning the practical generalization properties of neural networks demonstrated that the widely used backpropagation algorithm does not always achieve the desired generalization precision. This is not surprising, because, as a detailed analysis shows, the two tasks which should be solved during training to represent and to generalize the training examples are not well determined [Poggio and Girosi, 1990].
Mathematically speaking, the conditions for good approximation and good interpolation are only partly related. As a matter of fact, the backpropagation algorithm only considers approximation errors for gradient descent and improved approximation will, in general, not be accompanied by a better interpolation. Therefore, the interpolation obtained is strongly influenced by the random starting conditions of the optimization, and long training times often results in high quality approximations, but insufficient interpolations. This widely known effect often is called *overfitting*.

There are two different strategies to prevent neural networks from overfitting. The first one is especially useful if the available data set is small. It is based on a heuristic argument, which states that the simplest model will in general achieve the best generalization or interpolation. Following these argument one may try to choose as simple a net structure as possible, or state additional constraints on the weights to limit network complexity [Weigend *et al.*, 1990; Ji *et al.*, 1990]. The latter normally are called *Constraint Nets*. However, due to the general foundation of this method, only weak heuristic arguments concerning the interpolation properties find their way into the optimization procedure. An adaption to the special problem under investigation is only obtainable with great additional effort and, consequently, better interpolation is restricted to special well behaved problems. Moreover, the additional optimization expense leads to considerably increased training times.
If there are enough training samples, one may follow another strategy which relies upon the chosen training set. If the training data is selected carefully, it will contain enough information to ensure that the optimal approximating network will have good interpolation properties, too. Up to now, it is a well known practice to achieve this by selecting very large training sets which in general contain a lot of redundancy.

Following the latter strategy, a new method has been developed to ensure valid generalization. This algorithm, the *dynamic pattern selection*, is based on the batch training variant of the backpropagation algorithm and has been proven to be useful in applications with very large data sets. The training data is selected during the training phase, employing

cross validation to revise the actual training set. The error of the net function is used to choose the pattern, which should be added to the steadily growing training set [Röbel, 1992].
The overhead for the selection procedure is small. Due to the initially small training set size the dynamic pattern selection algorithm leads to more effective training then the standard algorithm. Practical experiments have shown that it outperforms current online training variants even in the case of very big and highly redundant data sets.

Plutowski and White [1993] have developed a similar algorithm, which they call *active selection* of training sets. Their algorithm focuses mainly on reducing training set size without considering generalization effects and does not employ cross validation to continually assess the generalization obtained by the training set in use. In contrast to their algorithm, the *dynamic pattern selection* proposed here, validates the training set by continually monitoring the the generalization properties of the net.

In the following section, the relations between approximation, interpolation and overfitting will be discussed on a background of function theory. Subsequently, the known heuristics concerning the number and distribution of training patterns are summarized. The actual methods to choose the training sets for neural nets will be described and the *dynamic pattern selection* will be established. Thereafter, two examples from the field of nonlinear signal processing are investigated to demonstrate the properties of the new algorithm. In the last section, there will be a short discussion concerning data requirements, noise and a comparison to online training methods.
The following explanations are based on the well known backpropagation algorithm as introduced by Rumelhart, Hinton and Williams [1986]. Descriptions of this algorithm are widely spread in the literature and will not be repeated here.

2 Approximation, Interpolation and "Overfitting" in the context of function theory

There have been many publications proving that under weak assumptions simple feedforward networks with a finite number of neurons in one hidden layer are able to approximate arbitrary closely all continuous mappings $R^n \to R^m$ [Hecht-Nielsen, 1989; White, 1990]. Concerning practical applications, however, these results are obviously of limited use, because they are not able to establish the required network complexity to achieve a certain approximation fidelity.

The conditions by which the stepwise improved approximation, achieved with the backpropagation algorithm, is accompanied by a decreasing interpolation error are not precisely known. To understand the basic relations it is useful to analyze the optimization procedure on the background of function theory. The target function $\vec{x} \mapsto \vec{y} = f_t(\vec{x})$ is assumed to be smooth, that is, f_t is a member of $\mathbb{C}^\infty$, the set of functions with continuous derivatives of every order, and the domain $\mathbb{X}$ of f_t to be a compact manifold. In general, there is only a limited set of members $\vec{x} \in \mathbb{X}$ available for which the targets $\vec{y} = f_t(\vec{x})$ are

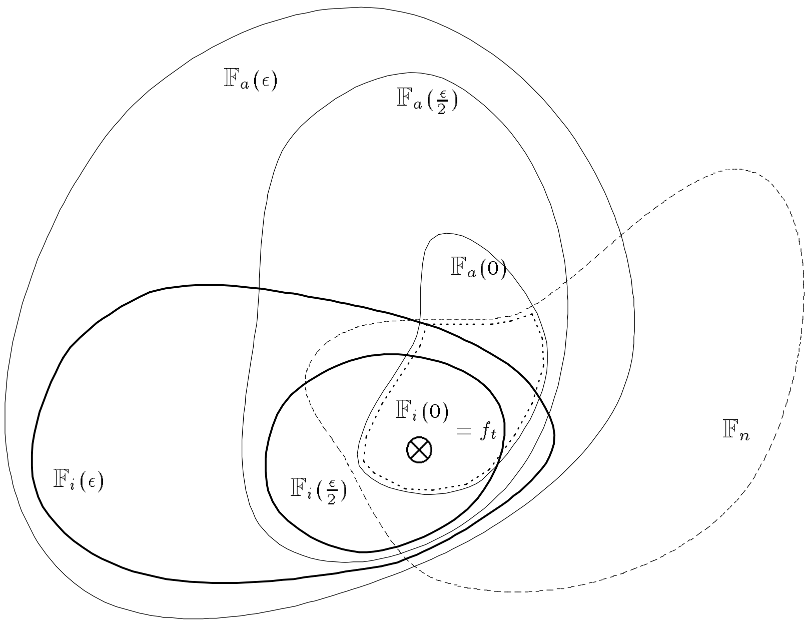

Figure 1: A possible relation between the set of all representable net functions $\mathbb{F}_n$, the sets of all ϵ-approximating functions $\mathbb{F}_a(\epsilon)$, and the sets of all ϵ-interpolating functions $\mathbb{F}_i(\epsilon)$ out of $\mathbb{C}^\infty$.

known. All known pairs $(\vec{x}, \vec{y})$ form the set of available data

$$\mathbb{D}_a = \{(\vec{x_0}, \vec{y_0}), (\vec{x_1}, \vec{y_1}), (\vec{x_2}, \vec{y_2}), ...\}. \tag{1}$$

Given $\mathbb{D}_a$, a neural network, and a real number $\epsilon \geq 0$, we may distinguish between three subsets of $\mathbb{C}^\infty$. First, there is the set of ϵ-approximating functions $\mathbb{F}_a(\epsilon)$, which is the set of functions f approximating the members in $\mathbb{D}_a$ to a given precision

$$\sup_{\mathbb{D}_a} \|f_t(\vec{x}) - f(\vec{x})\| \leq \epsilon. \tag{2}$$

Second, the set of ϵ-interpolating functions $\mathbb{F}_i(\epsilon)$, with distance

$$\sup_{\mathbb{X}} \|f_t(\vec{x}) - f(\vec{x})\| \leq \epsilon \tag{3}$$

to f_t. The third set is the set of functions f_n representable by the the neural net and is denoted as $\mathbb{F}_n$. While the sets $\mathbb{F}_a(\epsilon)$ depend on the set of available data $\mathbb{D}_a$, the sets $\mathbb{F}_i(\epsilon)$ are completely defined by the target function f_t. Using these terms, the target function may be specified as the single member in the set of 0-interpolating functions $\mathbb{F}_i(0)$.

Figure (1) shows a possible relation between the three function sets defined above. Depicted is a special setting in that the target function f_t is a member of $\mathbb{F}_n$ such that f_t

might be represented by f_n without any error. The following statements do not rely on this and therefore remain valid in general.
Note that the backpropagation algorithm generally is used with a squared error function to measure the approximation quality. To compare approximation results achieved with different training sets this measure has to be normalized using the number of elements contained in each training set. Employing this normalized squared error as a measure of distance in equations (2) and (3) would result in more complicated relations between $\mathbb{F}_a(\epsilon)$ and $\mathbb{F}_i(\epsilon)$. Even then however, the following statements remain valid in explaining the principle properties of the backpropagation algorithm.

With respect to the relation $\epsilon_i \leq \epsilon_j$, it is possible to establish the ordering $\mathbb{F}_a(\epsilon_i) \leq \mathbb{F}_a(\epsilon_j)$ on the set of all $\mathbb{F}_a(\epsilon)$ with $\epsilon \geq 0$. A corresponding relation exists for the set of all $\mathbb{F}_i(\epsilon)$. As mentioned above, the smallest set $\mathbb{F}_i(0)$ consists of one element only, in contrast to $\mathbb{F}_a(0)$, which may have infinite cardinality. Note that the set $\mathbb{F}_i(\epsilon)$ is always a subset of $\mathbb{F}_a(\epsilon)$. Concerning $\mathbb{F}_n$ and $\mathbb{F}_a(\epsilon)$, however, it is impossible to find such a simple general relation. Presuming the network weights are bounded, a valid assumption for practical applications, there exists an ϵ_u such that $\mathbb{F}_n \subset \mathbb{F}_a(\epsilon)$ for every $\epsilon > \epsilon_u$. On the other hand, there exists an $\epsilon_l \geq 0$ such that $\mathbb{F}_n \cap \mathbb{F}_a(\epsilon)$ is empty for all $\epsilon < \epsilon_l$. In the case of figure (1), for example, one finds $\epsilon_l = 0$.

The objective of the backpropagation algorithm is to choose a network function f_n, which is in $\mathbb{F}_a(\epsilon_l)$.
The area in figure (1) marked by the dotted line gives the cutting $\mathbb{F}_{an}(\epsilon_l) = \mathbb{F}_a(\epsilon_l) \cap \mathbb{F}_n$, containing all solutions obtainable by gradient descent. It is not possible to ensure that gradient descent optimization will reach $\mathbb{F}_{an}(\epsilon_l)$. Due to the specific error function, the constraints on the initial conditions and the selected training set, there might be no descending connection from the initial function to $\mathbb{F}_{an}(\epsilon_l)$.
As a matter of fact, the generalization of the optimum set $\mathbb{F}_{an}(\epsilon_l)$ is biased through the data contained in $\mathbb{D}_a$. There exist data sets $\mathbb{D}_a$ for which the relation between $\mathbb{F}_i(\epsilon)$ and $\mathbb{F}_a(\epsilon)$, combined with the gradient descend procedure, will result in poor generalization behavior.

For many applications, especially for function approximation, it is sensible to demand that the generalization error be equal to, or lower than, the training error. Formally, $f_n \in \mathbb{F}_a(\epsilon)$ should implicate $f_n \in \mathbb{F}_i(\epsilon)$. To be able to rank the generalization properties of f_n, it is sensible to define the generalization factor

$$\rho(f_n) = \frac{\epsilon_i(f_n)}{\epsilon_a(f_n)}, \tag{4}$$

where $\epsilon_a(f_n)$ is the minimal ϵ such that $f_n \in \mathbb{F}_a(\epsilon)$ and $\epsilon_i(f_n)$ is minimal such that $f_n \in \mathbb{F}_i(\epsilon)$. The generalization factor indicates the error made in optimizing on $\mathbb{D}_a$ instead of $\mathbb{X}$. As a result, we conclude that

$$\rho(f_n) \leq 1.0 \tag{5}$$

is a sensible condition for valid generalization.

3 Choosing the training set

As a result of the previous section, it should be clear that the set $\mathbb{D}_a$ strongly influences the generalization properties of the solutions obtained by gradient descent. Clearly, a further selection of training data out of $\mathbb{D}_a$ in general will lead to an even worse situation. The decreasing number of supporting points leads to increasing sets $\mathbb{F}_a(\epsilon)$ containing less information over the interesting sets $\mathbb{F}_i(\epsilon)$.

Consequently, one might think it would be best to use all available data for training purposes. For many applications, however, this would be awkward due to the immense data sets available. In speech recognition or signal processing $\mathbb{D}_a$ often contains many thousands of samples and a large amount of redundancy. Training on all of the samples can result in unnecessarily expensive computation. Moreover, the redundancy might not be equally distributed over the input space, thereby preventing optimal generalization properties. As a consequence, the question of selecting the proper training set to achieve optimal approximation and interpolation results arises.

Although the sufficiently dense distribution of training patterns on f_t is an important condition for successful training, there exist only some vague statements concerning this issue. Surely the suitable number of training patterns depends on the chosen network structure, the problem and the required precision. The latter relation, often unjustifiably neglected, evidently stems from the fact that $\mathbb{F}_a(\epsilon)$ has to contain more information about $\mathbb{F}_i(\epsilon)$ to achieve an interpolation with higher precision.

A first hint towards the necessary number of training samples can be obtained by analyzing the number of free parameters of f_n, given by the number of network weights. Consequently, one of the first suppositions concerning the suitable number of training patterns, stated as a rule of thumb for simple linear networks by Widrow [1987], demands that the number of training patterns should be ten times the number of free parameters. This rule has been utilized for nonlinear neural nets as well [Morgan and Boulard, 1990]. Although there exists a unique solution for considerably fewer supporting points, the overhead of information will result in a lower generalization error.

The necessary number of training samples heavily depends on their distribution over the input set $\mathbb{D}_a$. It is common to choose the training samples randomly out of $\mathbb{D}_a$. This is thought to reproduce the density of the underlying distribution. If there is no further information available this may be sensible, but because the properties of f_t (maxima, minima, curvature, ...) and $\mathbb{F}_n$ are not involved this will, in general, result in suboptimal training sets. One of the main advantages of random selection is its easy implementation. Moreover, the generalization properties of neural networks trained on randomly selected training sets may be investigated theoretically. In analyzing certain classes of networks and randomly selected training sets, Baum and Haussler [1989], for example, obtained some coarse estimates of the relation between training set size and achievable generalization precision.

In practical experiments, considerably fewer training patterns than stated in the above mentioned investigations suffice to give good generalization [Morgan and Boulard, 1990].

Aside from random selection, there are other approaches to obtain proper training sets by carefully choosing the training data out of the domain $\mathbb{X}$. One might try, for example, to choose the distance between adjacent training patterns to be almost constant. This, however, depends on a meaningful method to measure distances in $\mathbb{X}$. In signal processing applications, it is sensible to choose the Euclidean distance. Some experimental results obtained by the author shows that such an equal distance distribution of training samples leads to considerably better training/generalization properties then the random distribution. For other applications, however, it might be difficult to find a meaningful distance measure.

A more adept approach to select a suitable training set would be to adapt the training set by dynamically selecting training patterns while training proceeds. Atlas, Cohn and Ladner [1990] have proposed an algorithm which, by investigation of the network state, decides which patterns are to be added to the training set. Although their algorithm shows better results then random selection of training sets, it requires expensive computations and is therefore difficult to use in practical applications.

As previously mentioned, another dynamic approach was established by Plutowski, Cottrell and White [1993]. They train with a specific training set until the error stalls and then search in $\mathbb{D}_a$ for the element which possesses a gradient vector most similar to the average gradient of the entire set. This element is chosen to enlarge the training set. To prevent overfitting of the initially small training set, the initial network is rather small, with further hidden units added if the capabilities of the actual network to fit the growing training set are exhausted. This strategy will lead to very small training sets. For high precision approximation, however, the selection of the proper training set is computationally very expensive, as the neural net is trained to the desired precision for all intermediate training sets.

4 Online Cross Validation

The dynamic selection of training patterns proposed in the following section uses a well known tool from the field of estimation theory, called *cross validation*. Cross validation, described in detail by Stone [1974], is often used to revise statistical models by applying them to test sets. In the field of neural computation, cross validation has been used to verify parameter settings [Finnoff *et al.*, 1992], the network structure or generalization properties. The last is of great interest here and therefore will be explained further. As Hecht-Nielsen [1990] has proposed, the provided data set has to be divided into a training and a validation set, the latter not being used for training purposes. Applying the network function to the validation set, it is possible to estimate the generalization error of the net. This can already be done during the training phase. In the beginning of optimization, the estimated generalization error will generally decrease with the training error. After some time, the generalization error will reach a minimum, and start to increase while the training error decreases further. This is interpreted as the beginning of overfitting and Hecht-Nielsen suggests stopping training at this point.

Cross validation is a very flexible tool. However, there unfortunately exist two contrary demands, which refer to the necessary division of the data into training and validation sets. On the one hand, one would like to choose the test set as large as possible to achieve valid estimations for the generalization properties, on the other hand, all data contained in the test set can not be used for training and its information is lost to the training process. This will especially be a problem in situations where the available number of training data is very limited.

4.1 Dynamic selection of training patterns

In the preceding discussion, the principal relations between approximation and generalization have been clarified. The two results essential for the understanding of the proposed dynamic pattern selection are, in short form:

1. The number and the distribution of training patterns have an important influence on the resulting generalization properties of a neural network, but there exist only incomplete findings concerning practical solutions for selecting the training data.
2. Online cross validation is a useful tool to monitor the generalization properties of the network and can be applied during the training phase.

If one is willing to select the training patterns dynamically, two basic questions arise. Starting with an empty training set, the first question becomes: *Which* pattern should be chosen? There are several possible answers. The easiest, excluding random selection, is to select the pattern which has the highest error contribution. Compared to other possibilities, for example the sophisticated Δ-*ISB* criterion proposed by Plutowski and White [1993], the maximum error criterion is very easy to compute and has the advantage of being directly coupled to the generalization factor, equation (4). Therefore this criterion is used for the dynamic pattern selection algorithm [Röbel, 1992].

The second question, *at which time* the next pattern should be selected, turns out to be more tricky. There are two objectives. First, as is given by equation (5), the generalization factor ought to be less than one, and second, to prevent overfitting, the selection of new data should take place as early as necessary. The first objective may be achieved by estimating the generalization factor and inserting a new training pattern whenever it grows beyond one.
A number of experiments have shown this straightforward strategy results in reasonable training sets, which in many cases leads to better results then comparable sized fixed training sets. However, by employing this criterion alone, the generalization factor tends to oscillate between one and a value considerably below, at about 0.3-0.8.[1] Each selected pattern effects that the generalization factor decreases and reaches a minimum. Then it slowly increases again and due to the long time it takes the generalization factor to reach one, the selection of proper training sets for high precision training takes a long time. It would obviously be better to catch the generalization factor at its minimum and select

[1]This is especially true for training to very small errors

a new training pattern just when it starts to increase. Following this we have found the second objective to aim for: keep the generalization factor at its minimum.

There is one problem left now, which is to obtain valuable estimates of the generalization factor and its tendency without extensive computational efforts. We do not want to compute f_n for the whole data $\mathbb{D}_a$, but try to estimate the generalization factor by comparing the error function on the selected training set and a validation set. Following the method of cross validation, section **4**, the available data $\mathbb{D}_a$ is divided into subsets $\mathbb{D}_T$ and $\mathbb{D}_V$. $\mathbb{D}_T$ contains all possible training patterns and is referred to as the training store. $\mathbb{D}_V$, the validation store, contains all possible validation patterns. The actual training set $\mathbb{D}_t \subset \mathbb{D}_T$ and the validation set $\mathbb{D}_v \subset \mathbb{D}_V$ are selected from the respective stores.

The estimation of the generalization factor ρ is obtained by selecting a random validation set $\mathbb{D}_v$ and computing

$$\rho_v = \frac{E(\mathbb{D}_v)}{E(\mathbb{D}_t)}, \tag{6}$$

with $E()$ denoting the error function of the backpropagation optimization. To achieve comparable statistical properties of $E(\mathbb{D}_v)$ and $E(\mathbb{D}_t)$, one chooses $|\mathbb{D}_v| = |\mathbb{D}_t|$[2]. Having computed the generalization factor estimate, it now remains to use this value to estimate the generalization factor tendency. We compute the average a_ρ and standard deviation σ_ρ of the generalization factors from a fixed number of preceding epochs M[3]. To catch the increasing of the generalization factors as early as possible, we choose the threshold for the generalization factor to be

$$\xi_\rho(n) = \min(\xi_\rho(n-1), a_\rho + \sigma_\rho, 1.0) \tag{7}$$

and select a new training pattern whenever

$$\rho(n) > \xi_\rho(n). \tag{8}$$

Here the argument n reflects the number of training epochs computed so far. Note that ξ_ρ is monotonically decreasing. As the optimal threshold ξ might increase, it is appropriate to allow a small increase of the ξ after the selection of a new training pattern. Therefore the threshold is initialized after each selection to

$$\xi_\rho(n+1) = \min(\rho(n), 1.0) \tag{9}$$

and is fixed at this level for the number of epochs M used to calculate the statistical properties a_ρ and σ_ρ. Note that, due to the selection criterion (8), in case of a selection $\rho(n)$ is always above $\xi_\rho(n)$.

Regarding section **4**, there is a sensible extension of the algorithm. Calculating $E(\mathbb{D}_V)$ we are able to achieve a good, less fluctuating estimate of the generalization error of the actual net. This estimate may then be used to select the net which achieves best generalization properties during training. Moreover, a further investigation of the relation

[2] $|\mathbb{A}|$ here denotes the cardinality of $\mathbb{A}$

[3] In the following experiments $M = 100$ is used.

between generalization error and training set size $|\mathbb{D}_t|$ helps get further insight into the reasons for bad training results.
At first, the growing of $\mathbb{D}_t$ will be accompanied by a decrease of the generalization error estimation. If f_t is not contained in $\mathbb{F}_n$ or, due to the actual state of the network, not achievable by gradient descent, there will be a certain minimal generalization error. After having reached this limit, the further decreasing training error obtained by the backpropagation algorithm results in increasing generalization errors. The dynamic pattern selection algorithm prevents overfitting by frequently inserting new training patterns into $\mathbb{D}_t$. As a result, the generalization error will fluctuate around the minimum value with $\mathbb{D}_t$ slowly increasing. When this situation arises, the training process could be stopped, as the generalization will not improve further.
While this effect is due to the limited net complexity and may easily be prevented by choosing a larger network, there exists another situation which results in a fast growing of $\mathbb{D}_t$ up to $\mathbb{D}_T$ accompanied by an increasing generalization error. This rapid growth is due to missing information in $\mathbb{D}_T$ compared to $\mathbb{D}_V$. Therefore, we are able to decide which generalization precision might be achieved with the available data $\mathbb{D}_a$, by monitoring the rate of selection and the generalization error.

Whenever the set of available data $\mathbb{D}_a$ is rather small, the partitioning of the data into training and validation sets can impede successful training. This is the case if the information contained in each of these sets is incomplete. For such situations, there exists a modified version of the dynamic pattern selection algorithm. The modification consists in choosing both sets $\mathbb{D}_T$ and $\mathbb{D}_V$ to completely cover $\mathbb{D}_a$. As a consequence, the assertion obtained by the validation tests are less reliable. However, the experiments which are partly presented in the following section showed that the selected training sets remains quite reasonable, as long as they remain small. If $\mathbb{D}_t$ grows to more than half $\mathbb{D}_a$ one will in general not expect to obtain sound generalization estimates.

In the following, the dynamic pattern selection algorithm will be described more formally using a very general application example. The task which is to be learned consists in learning a target function f_t which is given, in accordance to equation (1), by a set of examples only. Formally the error function $E(\mathbb{D}_a)$ has to be minimized and the net function f_n ought to generalize to f_t outside the given examples.

The algorithm

Initialization:
The neural net weights are initialized with small random numbers, as is usual with the backpropagation algorithm. This initialization establishes a random net function f_n which is used to select the member $\vec{d_i} = (\vec{x_i}, \vec{y_i})$ out of $\mathbb{D}_T$ that shows the maximal error with respect to the error function E. The training set $\mathbb{D}_t$ is now set up containing just this maximal error element. The threshold of the generalization factor is set to one.
The minimal generalization error $E(\mathbb{D}_V)_{min}$ is initialized using the random start error on $\mathbb{D}_V$, which is $E(\mathbb{D}_V)$.

Training:
After each training epoch[4] one selects a random validation set $\mathbb{D}_v \subset \mathbb{D}_V$ holding as much elements as $\mathbb{D}_t$.
Whenever

$$E(\mathbb{D}_v) > \xi_\rho \cdot E(\mathbb{D}_t), \tag{10}$$

$\mathbb{D}_t$ will be enlarged by adding $\vec{d}_i \in \mathbb{D}_T \backslash \mathbb{D}_t$, the element from $\mathbb{D}_T$ which contributes most to $E(\mathbb{D}_T)$ and is not already a member of $\mathbb{D}_t$. At each training epoch, the threshold ξ_ρ has to be updated as per equations (7) and (9).
In the case that $E(\mathbb{D}_V)$ is smaller than $E(\mathbb{D}_V)_{min}$, the actual net will be stored and $E(\mathbb{D}_V)_{min}$ will be updated. After checking the respective stopping criteria, training continues.

Break:
If the generalization error stalls although $\mathbb{D}_t$ is growing or if any other stopping criterion matches, training will be finished and the optimal net stored so far represents the result of optimization.

The procedure just described has several advantages:

- The training patterns will be inserted whenever and wherever the information contained in $\mathbb{D}_t$ fails to yield a regular convergence of the net function f_n to the target function f_t. The algorithm could be interpreted as a dynamic adaption of $\mathbb{F}_a(\epsilon)$ to $\mathbb{F}_i(\epsilon)$, whereby especially the critical regions are formed.

- The redundancy contained in $\mathbb{D}_t$ remains small, and the number of training patterns is related to the reached training and generalization errors.

- The interpolation is controlled using the full information contained in $\mathbb{D}_a$ without using all the data for training. An overfitting of the training patterns is suppressed.

- Analyzing the relation between $|\mathbb{D}_t|$ and $|\mathbb{D}_T|$ may give some hints about the redundancy contained in $\mathbb{D}_T$ and, moreover, about the validity of the achieved generalization.

- The slowly growing training set leads to a slowly increasing complexity of the trained task, which, as was shown by Jacobs [1988], will often result in better training results.

Several additional remarks concerning the proposed algorithm are in turn:

[4]As long as the changes of the net function f_n remains small, it is possible to train several epochs without correcting the training set. Otherwise, the adaption of the training set stays too far behind the actual network state and training and generalization error diverge.

Between two enlargements of $\mathbb{D}_t$ there should be at least one training cycle to ensure that the additional information could have had some effect on the net function. This is ensured by the threshold initialization of equation (9), which takes place after each training data selection. Because the data points in the neighborhood of the latest added training pattern often show comparable errors, one might otherwise select a number of patterns in the actual critical place. This would lead to groups of redundant patterns and thereby prevent a regular convergence.

Because the estimation of the generalization is a computationally expensive task for large sets $\mathbb{D}_V$ and because the changes of the generalization error are usually very slow, especially at the end of the optimization procedure, one may choose to evaluate this estimation after a number of training epochs only. This will generally have a negligible effect on the achieved results.
As the generalization is controlled by adapting the training set this estimation of the generalization error can even be omitted at all. The experiments have shown, that the differences between the optimal and the final state of the optimization procedure is fairly small.

The proposed algorithm results in a monotonously increasing training set $\mathbb{D}_t$. There have been some experiments done with decreasing training sets, by removing the best pattern from $\mathbb{D}_t$. In general this results in poorer performance. Even if the training process has adapted f_n to f_t in the neighborhood of the selected patterns, it seems that they retain their importance in ensuring regular interpolation.

5 Experimental results

In the following section the favorable properties of the dynamic pattern selection are demonstrated. The learning algorithm used here is an accelerated batch mode back-propagation algorithm with dynamically adapted learning rate and momentum [Salomon, 1991; Röbel, 1993]. All neural nets are simple feedforward structures with sigmoidal activation function of the hidden units and linear output units.

The tasks to solve stems from the field of nonlinear signal processing. Precisely speaking, two examples of continuous, nonlinear system functions shall be represented by the neural nets. After successful training, the net function may be used as a predictor for the future behavior of the dynamical system. The underlying theory is beyond the scope of this article. A suitable introduction has been given by Lapedes and Farber [1987a; 1987b].

5.1 Predicting the Henon model

In the first experiment, the network is trained to predict the chaotic dynamic of the henon model [Grassberger and Procaccia, 1983a]. This two dimensional model is given by the difference equation

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} y_n + 1 - a * x_n^2 \\ b * x_n \end{pmatrix}. \tag{11}$$

Choosing the parameters $a = 1.4$ and $b = 0.3$ will result in a solution showing chaotic behavior. A simple transformation of equation (11) gives the prediction function

$$x_{n+2} = 1 + b * x_n - a * x_{n+1}^2, \tag{12}$$

which is to be approximated by the neural net.

The available data set

$$\mathbb{D}_a = \{(x_n, x_{n+1}, x_{n+2}), (x_{n+1}, x_{n+2}, x_{n+3}), ...\} \tag{13}$$

consists of 331 vectors, built from the solution of (11) and the initialization $x_0 = 0.6, y_0 = 0.18$. An additional independent validation set

$$\mathbb{D}_u = \{(x_{n+333}, x_{n+334}, x_{n+335}), (x_{n+334}, x_{n+335}, x_{n+336}), ...\} \tag{14}$$

has been built from the following 667 vectors of the solution.

The neural net architecture used for this task is chosen to have an input layer with two units, one hidden layer with seven units and an output layer with one unit. The initial weights are randomly chosen out of the interval $[-1, 1]$. It turns out that equation (12) is easily approximated by this fairly small neural net, and therefore this task is well suited to compare the training results of the dynamic pattern selection with the results obtained with fixed training sets of varying sizes. Moreover, by virtue of the low dimensionality of the problem, it is possible to get a visual impression of the distribution of the selected training samples on f_t.

The fixed training sets are subsets of $\mathbb{D}_a$, and are chosen to approximate with varying density a uniform covering of $\mathbb{D}_a$. While, in the case of fixed training sets, each training run consists of 6000 epochs, the dynamic pattern selection algorithms were stopped after 2000 epochs. This results in similar average generalization errors. To obtain statistically valid assertions in comparing the results, 20 different initial weight settings have been used with each training algorithm.

The average training and generalization results are presented in table (1). The error bars are proportional to the variance of the results. In the upper part of this table the results using the different sized fixed training sets are shown, in the lower part one finds the results using both variants of the dynamic pattern selection. Here, the first line represents the dynamic pattern selection with an independent validation set. For the training and validation repertoire, the relations $\mathbb{D}_T = \mathbb{D}_a$ and $\mathbb{D}_V = \mathbb{D}_u$ are chosen. The second line shows the results for the modified version without a special validation set. In this case, the data sets are chosen following the relation $\mathbb{D}_T = \mathbb{D}_V = \mathbb{D}_a$.

The number of training patterns in the fixed training sets and the average number of selected training patterns at the end of the training are listed in column one. The columns

training set size	training - error $E(\mathbb{D}_t)$	generalization - error $E(\mathbb{D}_u)$	generalization - factor ρ_u	forward-/backward propagations
5	4.50e-2 ± 4.84e-2	3.75e-1 ± 6.60e-2	8.33	0.18e+6
15	4.29e-3 ± 2.00e-3	2.50e-2 ± 1.66e-2	5.83	0.54e+6
25	3.77e-3 ± 1.79e-3	4.80e-3 ± 2.04e-3	1.27	0.90e+6
35	2.46e-3 ± 1.24e-3	2.90e-3 ± 1.35e-3	1.18	1.26e+6
50	3.02e-3 ± 1.61e-3	3.46e-3 ± 1.75e-3	1.15	1.80e+6
75	2.99e-3 ± 1.54e-3	3.14e-3 ± 1.65e-3	1.05	2.70e+6
100	2.90e-3 ± 1.55e-3	3.21e-3 ± 1.73e-3	1.10	3.60e+6
150	4.53e-3 ± 2.11e-3	5.08e-3 ± 2.38e-3	1.12	5.40e+6
300	3.56e-3 ± 2.19e-3	3.90e-3 ± 2.42e-3	1.09	10.8e+6
69 ± 10	3.69e-3 ± 1.36e-3	3.06e-3 ± 1.10e-3	0.83	0.74e+6
65 ± 7	2.65e-3 ± 5.78e-4	2.24e-3 ± 4.84e-4	0.84	0.69e+6

Table 1: Predicting the Henon model. Comparison of the average training and generalization results using a 2-7-1 neural net and different training sets.

two and three contain the training and generalization results measured by means of the normalized prediction error

$$E(\mathbb{D}) = \sqrt{\frac{\sum_{x\in\mathbb{D}}(f_t(\vec{x}) - f_n(\vec{x}))^2}{\sigma_{\mathbb{D}}^2|\mathbb{D}|}}, \tag{15}$$

where $\sigma_{\mathbb{D}}$ represents the standard deviation of the target function measured on the data set $\mathbb{D}$.

All the errors contained in table (1) have been measured using the *optimal* weight set, which was determined during the training phase by cross validating the generalization error on $\mathbb{D}_V$. Using equation (6) the generalization factor ρ has been estimated to be

$$\rho_u = \frac{E(\mathbb{D}_u)}{E(\mathbb{D}_t)}. \tag{16}$$

Column five shows the average computational expense neede to achieve the generalization results for the respective training set. It is estimated by the number of propagation passes through the net. Here the simple approximation of equal computational efforts for forward and backward propagations is made. The acceleration algorithm needs four additional forward propagations for each training pattern and each training epoch, so each training cycle consists of 5 forward and 1 backward propagation for a selected training pattern. For the dynamic selection there is an additional overhead of 1 forward propagation for each pattern in the randomly chosen validation set $\mathbb{D}_v$. Moreover, each pattern selection results in 1 forward propagation for each member of $\mathbb{D}_T$ to select the training pattern with the maximum error contribution.

The smallest of the fixed training sets contain just five and fifteen training patterns and the net function f_n is obviously under-determined. The examples are learned quickly and

with high precision but the generalization error minimum stalls at a very early state. All the other fixed training sets lead to comparable precisions. A remarkable result is the flat minimum for training and generalization error, when using a training set with 35-100 elements. The increasing error for larger training sets is a result of poorly distributed training patterns. The full set $\mathbb{D}_T$ with all 331 training patterns obviously does not provide an equally spaced distribution of the patterns. The more patterns that have to be chosen out of this set, the harder it is to select a training set with proper distribution in the input space.
Using dynamic pattern selection, 2000 training epochs are sufficient to reach the same average generalization error as with the best fixed training set. Compared to the fixed training set with the minimal generalization error, which is the set containing 35 training patterns, the dynamic training sets contains a higher number of patterns. This is to ensure that the generalization error stays below one, which is not obtained by any of the fixed training sets.
In comparing both variants of the dynamic selection algorithm, one finds that the training sets selected by the modified algorithm are significantly smaller. This is due to the less severe validation, without an additional validation set. In the case of the henon model, the loss in model reliability does not affect the generalization properties and therefore the training results are even improved. In general this is not the case. An example for different behavior will be found in the next experiment.

As mentioned earlier, all experiments are done with a fixed number of training cycles. Due to the varying training sets this leads to varying computational costs for the different runs. For the fixed training sets, the total number of forward and backward propagations increase from about 200 thousand up to 10.8 million. The total computational cost for the dynamic pattern selection is comparable to that for 25 patterns in a fixed training set. However, compared to this fixed training set size, the dynamic pattern selection achieves considerably better generalization results. The smaller cost of the dynamic selection algorithm is a result of the initially small training sets, which compensate for the larger training sets in the final training phase.

In figure (2), the dependency between generalization factor ρ_u and the number of training epochs is shown. As can be seen, the number of training patterns considerably effects the generalization properties. As the fixed training sets are enlarged the reliability of the generalization is increased, but with a higher number of training epochs and improved training error this reliability decreases. For very long training or small training sets overfitting takes place and the generalization properties tend to become random.

The evolution of the generalization factor for fixed training sets with 25 and 50 patterns shows that there is a randomly varying training epoch for which the information contained in the training set does not ensure the desired generalization quality. This limit is moved towards smaller training errors if the training set is enlarged. Regarding the dynamic selection it is shown that the generalization factor is controlled to stay well below one[5].

[5]The initially higher variance of the generalization factor is a consequence of the learning rate adaption. The learning rate is fairly high in the first few hundred epochs. This leads to a quickly varying generalization error, which nevertheless is controlled by inserting new training patterns.

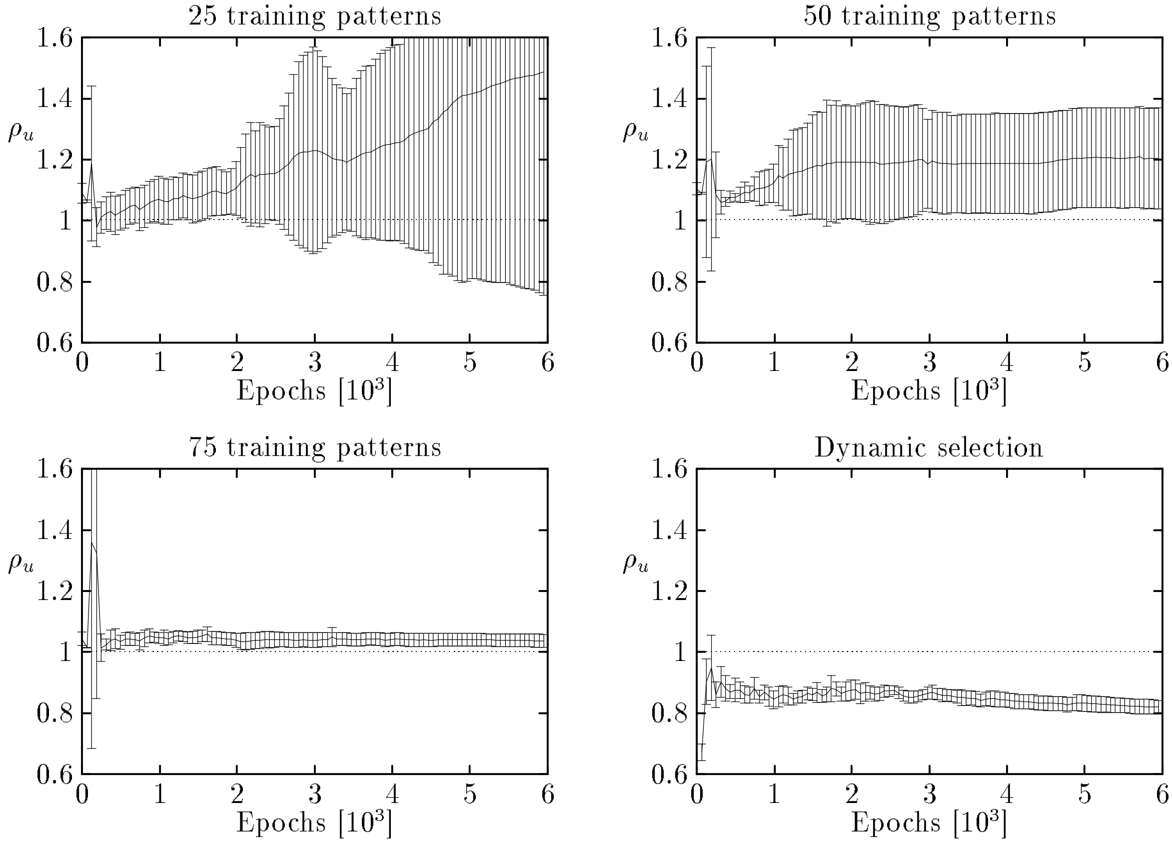

Figure 2: Generalization factor ρ_u as a function of the number of training epochs for learning to predict the henon model.

As was already mentioned, despite the size of the training set, the distribution of the training patterns has a considerable effect on the generalization results. A typical distribution obtained by the dynamic pattern selection is depicted in figure (3). The critical regions, as maxima, minima, margins and parts of high curvature, are occupied. The distribution is not uniform but reflects the error distribution of the net function. This indicates the advantage of the dynamic selection in contrast to the fixed training sets, which must be selected based on general heuristics and without any knowledge of the intermediate network state.

5.2 Predicting the Mackey-Glass model

After having shown the basic properties of the dynamic pattern selection by means of a fairly easy problem, the more complicated task of predicting the chaotic behavior of the

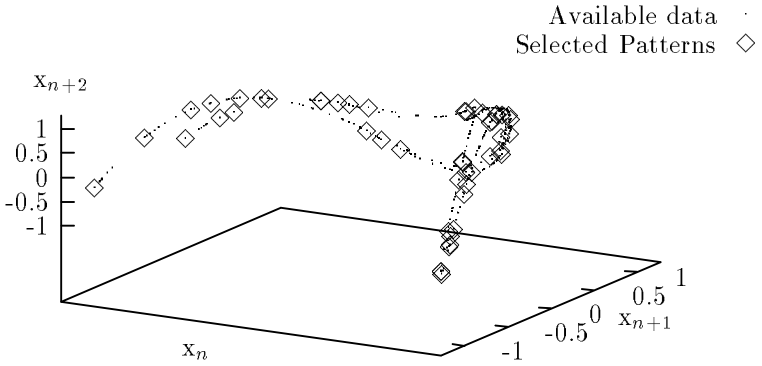

Figure 3: A typical distribution of training patterns on the prediction function of the henon model. Depicted is the distribution of training patterns subsequent to 1000 training cycles.

Mackey-Glass model is investigated. Due to the time delay, the Mackey-Glass model

$$\dot{x}(t) = \frac{a \cdot x(t-\tau)}{(1 + x(t-\tau)^{10})} - b \cdot x(t) \qquad (17)$$

has infinite degrees of freedom. The stationary motion is, however, governed by a low dimensional attractor [Farmer, 1982; Grassberger and Procaccia, 1983b]. This is the reason why the Mackey-Glass model is often used as an emulation of real world systems and predicting this model has been established widely as a kind of benchmark for testing predictors [Farmer and Sidorowich, 1988; Lapedes and Farber, 1987b; Crowder, 1990]. As Lapedes and Farber [1987b] have shown, the prediction of the Mackey-Glass model, with the parameter set $a = 0.2, b = 0.1$ and $\tau = 30$, might be attained by using a neural net with six input units, two hidden layers with ten units each and a linear output unit. This settings will be used here as test for the dynamic pattern selection, too.

Solving the equation (17) has been done using a second order Runge-Kutta method, where the first 1000 steps of the solution were skipped to reach the steady state. The data set, $\mathbb{D}_a$, is built from the following 5000 steps of the solution, with a total of 4964 vectors $\vec{v_i} = \{x_i, x_{i+6}, x_{i+12}, x_{i+18}, x_{i+24}, x_{i+30}, x_{i+36}\}$. Following Lapedes and Farber, the vectors are constructed with a time delay of six steps and the fixed training set contains 500 vectors out of $\mathbb{D}_a$, which has been selected with nearly uniform distribution in input space. The independent generalization test set contains 2000 vectors chosen randomly out of the following 10000 steps of the solution.

The fixed training set and each of the dynamic selection algorithms has been used for training with ten initial weight sets. Each run consists of 20000 cycles and the average results are shown in table (2). The labeling here is identical to that in table (1).

The fixed training sets give rise to a generalization factor considerably beyond one. As is

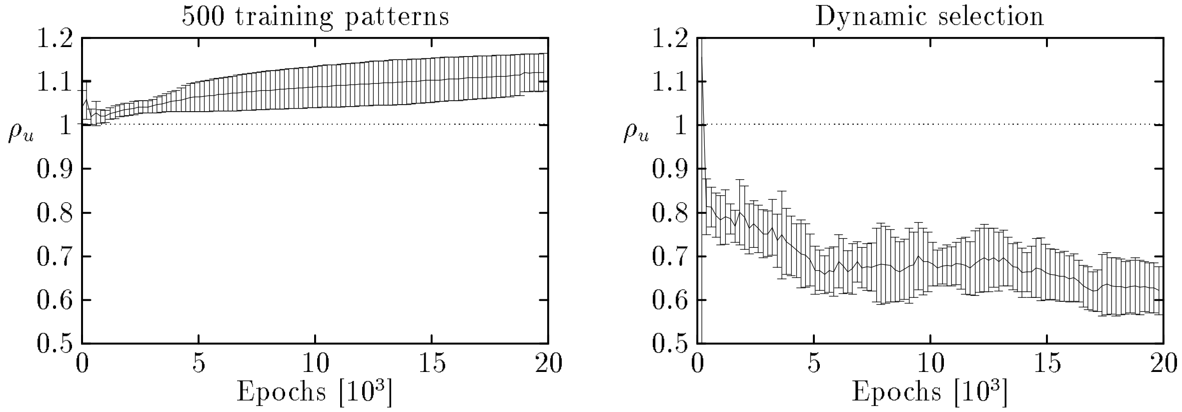

Figure 4: Generalization factor ρ_u as a function of the number of training epochs for learning to predict the Mackey-Glass model.

training set size	training - error $E(\mathbb{D}_t)$	generalization - error $E(\mathbb{D}_u)$	generalization - factor ρ_u	forward-/backward propagations
500	2.47e-2 $\pm$ 3.6e-3	2.73e-2 $\pm$ 3.4e-3	1.11	60e+6
207 $\pm$ 11	4.62e-2 $\pm$ 3.9e-3	2.74e-2 $\pm$ 1.4e-3	0.59	18e+6
202 $\pm$ 13	4.27e-2 $\pm$ 3.2e-3	2.74e-2 $\pm$ 1.5e-3	0.64	18e+6

Table 2: Predicting the Mackey-Glass model. Comparison of the average training and generalization results using a 6-10-10-1 neural net and different training sets.

depicted in figure (4) the generalization factor increases monotonically. In contrast to this, the generalization factor of the dynamic selection algorithm is well below one, indicating a more reliable generalization and the selected sets contains, on average, less than half the number of training patterns. The generalization results being approximately equal, the computational expenses of the dynamic selection algorithms are less by a factor of 0.3. Note that there is no need to do any initial investigations to determine the optimal training set.

6 Discussion

After having demonstrated the basic properties of the dynamic pattern selection, some further topics, concerning possible extensions, data requirements and noise shall be discussed.

Obviously, the application of the dynamic pattern selection is limited to problems where generalization is possible. Otherwise, any selection of a training subset will fail to end up with usable network performance. Up to now, there have been no investigations with problems other than the approximation of continuous functions. To apply the dynamic pattern selection to classification problems with binary target values, the criterion for proper selection times probably has to be modified. In those cases, it seems quite reasonable to expect the dynamic pattern selection to reveal clustering properties and further investigation might lead to interesting results.

As a matter of fact, the dynamic pattern selection does not depend on a special error function. Therefore one might use the dynamic training sets in combination with other backpropagation extensions, as for example constrained networks, thus getting benefits from each of the methods.

6.1 Data requirements

As should be clear from the discussion above, an important precondition for the utilization of the dynamic pattern selection is a sufficient number of training patterns in the data set $\mathbb{D}_a$. In fact, even with very small data sets, the dynamic selection algorithm is a favorable choice, due to the increased control over network results. Although the training set will cover the whole training store more or less quickly, one gains the possibility to estimate the achievable generalization by measuring the training error at the time when the training repertoire is exhausted. After having selected all available training patterns, the different versions of the algorithm are equivalent to the standard backpropagation algorithm, with or without cross validation. As a consequence of this limiting behavior, the dynamic pattern selection has the same data requirements as the standard backpropagation algorithm.

6.2 Noise

In the preceding discussion, one question has been left unasked that is of great interest for the practical virtue of the proposed algorithm. What happens if there is considerable noise in the data?
There is one important difference between dynamic and fixed training sets, which might result in a somewhat higher sensitivity to noise in the dynamic case. If only some of the available training patterns are disturbed, the selection process will probably select most of these patterns trying to improve the bad generalization results. Due to the concentrated training sets the averaging between the selected samples is smaller and consequently the training results are affected more by the additional noise then in the case of fixed training sets.

In many cases, however, the noise will be uniformly distributed over the data and in this cases the dynamic training sets will yield similar averaging properties as the fixed sets. In many experiments the dynamic selection has been applied for training neural network predictors of real world signals, which are disturbed by noise levels that are small compared to the training error [Röbel, 1993]. In all cases the dynamic selection proved to be stable with superior training results then the standard algorithm.

6.3 Comparison with online training

It has been argued that the dynamic pattern selection is obviously well suited to process very large data sets containing highly redundant data. In contrast to this, it is often assumed that, in the case of redundant data, the online training variant of backpropagation will yield superior results. Therefore, we have compared the computational expenses for an up to date online training method, the *Search-Then-Converge* learning rate schedule as is set forth in [Darken and Moody, 1991] and the batch mode dynamic pattern selection algorithm. The neural networks have been trained to predict a real world piano signal consisting of 15000 samples and the learning rate adaption of the online training method has been optimized in a number of preceding runs. For the dynamic pattern selection the automatic learning rate adaption already mentioned in the previous sections has been employed.

Despite the preceding optimization necessary for the online training algorithm and the fact that there have been only 50 patterns selected by the dynamic selection, which results in a big overhead for the selection process, the overall expense for the batch mode, dynamic pattern selection training has been lower than the online training expense by a factor of 0.3.[6]

As a consequence of this experiment, it follows that the proposed dynamic pattern selection combined with an accelerated batch mode training is the method of choice in all cases where the complete training set $\mathbb{D}_a$ is available at training time. Only in cases where the net has to be adapted to time varying situations during application should online training be used.

7 Conclusion

Based on the fact that the generalization properties of neural networks are heavily determined by the training sets, an extension to the standard backpropagation algorithm, the dynamic pattern selection, has been introduced. The proposed algorithm has been tested on two problems from the area of nonlinear signal processing. Comparing the results to standard backpropagation training on optimized fixed training sets it has been shown that the dynamic pattern selection algorithm achieves the same average generalization results with less computational expense and without any preceding investigation of the available data.

The dynamic pattern selection has especially proven to be useful for very large and highly redundant data sets which are often used in signal processing applications. In these cases, one should select a reasonable subset as a training and validation store. Further selection

[6] I would like to thank Jens Ehrke for his support on online training experiments.

will then be done automatically. As a special feature, the investigation of the dynamically selected training sets allows to qualitatively estimate the data redundancy and moreover gives some hints on the reliability of the training results.

References

[Atlas *et al.*, 1990] L. Atlas, D. Cohn, and R. Ladner. Training connectionist networks with queries and selective sampling. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 2 (NIPS 89)*, pages 566–573. Morgan Kaufmann Pub., 1990.

[Baum and Haussler, 1989] E. Baum and D. Haussler. What size nets give valid generalization? *Neural Computation*, 1(1):151–160, 1989.

[Crowder, 1990] R. Crowder. Predicting the mackey-glass timeseries with cascade-correlation learning. In *Proc. of the Connectionsts Models Summer School*, pages 117–123, 1990.

[Darken and Moody, 1991] C. Darken and J. Moody. Note on learning rate schedules for stochastic optimization. In R. L. and. J.E. Moody and D. Touretzky, editors, *Neural Information Processing Systems 3 (NIPS 90)*, pages 832–838. Morgan Kaufmann Pub., 1991.

[Farmer and Sidorowich, 1988] J. D. Farmer and J. J. Sidorowich. Predicting chaotic dynamics. *Dynamic Patterns in Complex Systems, World Scientific*, :265–292, 1988.

[Farmer, 1982] J. Farmer. Chaotic attractors of an infinite-dimensional dynamical system. *Physica D*, 4:366–393, 1982.

[Finnoff *et al.*, 1992] W. Finnoff, F. Hergert, and H. Zimmermann. Improving generalization performance by nonconvergent model selection methods. In I. Aleksander and J. Taylor, editors, *Proc. of the Inter. Conf. on Artificial Neural Networks 2 (ICANN 92)*, pages 233–236. Elsevier Sience Publisher, 1992.

[Grassberger and Procaccia, 1983a] P. Grassberger and I. Procaccia. Estimation of the Kolmogerov entropy from chaotic signal. *Physical Review A*, 28(4):2591–2593, 1983.

[Grassberger and Procaccia, 1983b] P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica D*, 9:189–208, 1983.

[Hecht-Nielsen, 1989] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Proceedings of the Intern. Joint Conf. on Neural Networks*, pages I–593–I–605. IEEE TAB Neural Network Comittee, June 1989. Washington D.C.

[Hecht-Nielsen, 1990] R. Hecht-Nielsen. *Neurocomputing.* Addison-Wesley Publishing Company, 1990.

[Jacobs, 1988] R. A. Jacobs. Initial experiments on constructing domains of expertise and hierarchies in connectionist systems. *Proc. of the Connectionsts Models Summer School*, pages 144–153, 1988.

[Ji *et al.*, 1990] C. Ji, R. Snapp, and D. Psaltis. Generalizing smoothness constraints from discrete samples. *Neural Computation*, 2(2):188–197, 1990.

[Lapedes and Farber, 1987a] A. Lapedes and R. Farber. How neural nets work. *IEEE Conference on Neural Information Systems*, 1:442–457, 1987.

[Lapedes and Farber, 1987b] A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical Report LA-UR–87-2662, Los Alamos National Laboratory, 1987.

[Morgan and Boulard, 1990] N. Morgan and H. Boulard. Generalization and parameter estimation in feedforward nets: Some experiments. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 2 (NIPS 89)*, pages 630–637. Morgan Kaufmann Pub., 1990.

[Plutowski and White, 1993] M. Plutowski and H. White. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, 4(2):305–318, 1993.

[Plutowski *et al.*, 1993] M. Plutowski, G. Cottrell, and H. White. Learning Mackey-Glass from 25 examples, plus or minus 2. *Preprint*, 1993.

[Poggio and Girosi, 1990] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[Röbel, 1992] A. Röbel. Dynamic selection of training patterns for neural networks: A new method to control the generalization. Technical Report 92-39, Technical University of Berlin, 1992. In German.

[Röbel, 1993] A. Röbel. *Neural Models of nonlinear dynamical systems and their application to musical signals*. PhD thesis, Technical University of Berlin, 1993.

[Rumelhart *et al.*, 1986] D. Rumelhart, G. Hinton, and R. Williams. *Learning Internal Representations by Error Propagation*, volume 1: Foundations of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 318–362. Rumelhard & McClelland (Eds.), MIT Press, 1986.

[Salomon, 1991] R. Salomon. *Verbesserung konnektionistischer Lernverfahren, die nach der Gradientenmethode arbeiten*. PhD thesis, Technische Universität Berlin, 1991.

[Stone, 1974] M. Stone. Cross-validatory choice and assessment of statistical predictors (with discussion). *Journal of the Royal Statistic Society*, 36:111–147, 1974.

[Weigend *et al.*, 1990] A. Weigend, B. Huberman, and D. Rumelhart. Predicting the future: a connectionist approach. *Intern. Jou. of Neural Systems*, 1(3):193–209, 1990.

[White, 1990] H. White. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, 3(5):535–549, 1990.

[Widrow, 1987] B. Widrow. ADALINE and MADALINE – 1963. *Proc. IEEE 1st Inter. Conf. on Neural Networks*, 1:145–157, 1987.