A SELECTIVE LEARNING METHOD TO IMPROVE THE GENERALIZATION OF MULTILAYER FEEDFORWARD NEURAL NETWORKS

I. M. GALVÁN, P. ISASI, R. ALER and J. M. VALLS
Universidad Carlos III de Madrid, Avenida de la Universidad, 30, 28911 Leganés (Madrid), Spain

Received 31 July 2000
Revised 1 March 2001
Accepted 13 March 2001

Multilayer feedforward neural networks with backpropagation algorithm have been used successfully in many applications. However, the level of generalization is heavily dependent on the quality of the training data. That is, some of the training patterns can be redundant or irrelevant. It has been shown that with careful dynamic selection of training patterns, better generalization performance may be obtained. Nevertheless, generalization is carried out independently of the novel patterns to be approximated. In this paper, we present a learning method that automatically selects the training patterns more appropriate to the new sample to be predicted. This training method follows a lazy learning strategy, in the sense that it builds approximations centered around the novel sample. The proposed method has been applied to three different domains: two artificial approximation problems and a real time series prediction problem. Results have been compared to standard backpropagation using the complete training data set and the new method shows better generalization abilities.

1. Introduction

Multilayer feedforward neural networks (MFNN) with backpropagation algorithm are passive learners because they passively receive information about the problem domain and attempt to adjust the weights to learn the training samples.[1] Although backpropagation neural networks have been used successfully in many applications, they may suffer from problems inherent to the training data used to get the network learned.

The level of generalization, i.e., the ability to correctly respond to novel inputs is heavily dependent on the quality of the training data. In the traditional way, the neural network is trained with available samples about the domain and the training set is generally chosen in such a way that it represents most widely the problem. Thus, the average amount of novel information per sample decreases as learning proceeds. The reason for this is that as the size of the training set grows, the knowledge of the network about large regions of the input space become more and more confident; so, additional samples from these regions are basically redundant, as they do not contribute considerable to an improvement in generalization ability.

Much research has been done to improve generalization and to reduce the convergence time. This research has mostly been focused on optimal setting of initial weights of perceptron multilayer;[2] optimal learning rates and momentum;[3] finding optimal architectures using pruning and construction techniques;[4] and on sophisticated weight modification rules and optimization techniques;[5,6] and techniques based on mixture of experts, in which several expert networks trained on different partitions of the input space.[7,8]

While techniques previously mentioned are useful for a variety of problems, other authors have paid attention to other additional factors which influence the learning speed and the generalization ability of the networks. One of them is the nature and size of the training set. There is no guarantee that the generalization performance is improved by increasing the training set size.[9] In general, one should choose those examples which are most likely to help the network to solve the problem. It has been shown that with careful dynamic selection of training patterns, better generalization performance may be obtained.[10] This has given rise to new methods named active learning methods in the literature. Active methods consist of any form of learning in which the learning process has some control over the training patterns it is trained with.[10,11] Those strategies allow to dynamically select training patterns from a candidate training set in order to reduce the convergence time and to increase the generalization ability of MFNNs.

Following this approach, different works have appeared in the literature. For classification problems, the selected examples are the border patterns, i.e., the patterns that lie closest to the separating hyperplanes. In Ref. 12, it is shown that a network trained on border patterns generalizes better than a network trained on the same number of examples chosen at random. In Ref. 13, the nearest neighbor criterion is used to distinguished between typical samples and confusing samples. Other works in which different criteria are used for selecting critical examples.[14–16]

The idea of selecting dynamically or actively the patterns to train the network from the available data about the domain is close to our approach. However, the aim in this work is to develop learning mechanisms such that the selection of patterns used in the training phase is based on the novel sample, instead of based on other training patterns. Thus, the network will use its current knowledge of the new sample to have some deterministic control about what patterns should use for training. In addition, novel samples which are not represented in the available training data about the domain could be used in future training of the MFNN.

Most of the supervised learning methods can be considered as eager learning methods, in the sense that generalization is carried out beyond the training data before observing the new sample. When a new pattern is received, eager methods have already chosen their global approximation. That global approximation over the training data representing the domain can lead to poor generalization properties. An alternative approach in supervised learning that tries to solve this problem is to defer the generalization phase until a new sample is obtained, using a selection of patterns of the training set. Thus, the learning methods construct local approximations using a selection of training data instead of using the total set in which irrelevant or redundant information could be given.

These ideas are known in the literature as lazy learning methods[17,18] because they defer the decision of how to generalize beyond the training data until each new sample is encountered. These methods usually involve finding relevant data to answer a particular novel pattern. Thus, the decision about how to generalize is carried out when a test pattern needs to be classified constructing local approximations. The relevance of a pattern in the training data is often measured using a distance function, with nearby points having high relevance. Once relevant patterns are selected, novel samples are answered by combining the most relevant training data and discarding data that could worsen the generalization of the new pattern. In that sense, it is said that lazy methods construct local approximations because only local data is taken into account in the generalization. Using lazy techniques, better generalization capabilities could be expected, because only relevant patterns are used.

The most basic form of lazy learning is the k-nearest neighbor classifier.[19] It simply stores the entire training set and postpones all effort towards inductive generalization until classification time. That method works by retrieving the k least distant input patterns of the novel sample, i.e., the most similar; afterwards the approximation of those patterns is just the most common value of output patterns among the k training examples nearest to the new sample. Variant of this basic method have been developed improving its accuracy on some learning tasks (e.g., Refs. 20 and 21).

Other form of lazy learning is locally weighted regression.[17,22,23] This method constructs an explicit approximation of the target function over a local region surrounding the new sample. That local approximation might be built up using a linear function, quadratic function or some other functional form. Locally weighted regression uses nearby or distance-weighted training examples to form this local approximation, and the contribution of each training example is weighted by its distance to the novel pattern. The regression of the local function coefficients is based only on data near the novel pattern. That local approximation is then applied to that new pattern, for classification or prediction purposes.

The learning method proposed in this work to train MFNNs is inspired on lazy strategies. Perhaps, the most basic idea — and following the main ideas developed on lazy methods — is to select from the whole training data the k-nearest patterns to the novel pattern and to train the MFNN with these k patterns. This is, when the new sample is found, the subset of the k nearest patterns could be used to train the MFNN. After that, the trained network would be used to approximate the new sample. However, in this work the idea of selecting the k-nearest patterns has been rejected mainly because of two factors. First, determining the best value for k might be a problem which might be solved by a trial and error mechanism. In addition, the k parameter will probably depend on the given problem. Secondly, and the most important issue, the network will always be trained with the same amount of training data for each new sample. That may be a disadvantage because each novel pattern could require a different amount of training data.

The lazy method proposed in this work attempts to solve this problem by selecting, for each new sample, the training patterns in an automatic way. The method selects the most similar training patterns to the new pattern received and discards those useless patterns that may not provide useful knowledge to approximate that new pattern. The selection is based on the inverse of Euclidean distance, thus neighboring patterns will be selected and farther ones will be discarded. The number of selected trained patterns can be different for each new sample. In addition, the method follows the principle that the closer to the novel sample a learning pattern is, the more important it should be considered for learning. This is achieved by replicating it according to its closeness to the novel pattern.

To sum up, instead of using the whole training data, a variable portion of the training data is selected when a new sample is received. This selection depends on the incoming test pattern. Then, a complete neural network is trained from scratch for every new pattern, and then that network is applied to the new sample to predict. It is very important to remark that any lazy strategy — like ours — implies a bigger computational effort than an eager strategy, because a local approximation based on the novel sample must be built for every novel sample, instead of building a global approximation once and for all. However, in most real applications this is not a problem because the available time between the arrival of novel samples is enough to train the network. Besides, if novel samples arrive incrementally and it is desired to use those samples for training purposes, then lazy methods are not worse than eager methods, because both have to generate a network everytime a new pattern arrives. Actually, in that case, lazy methods are better in terms of computational cost, because they use fewer patterns to train the network, hence training time will be smaller.

The purpose of this paper is to empirically show that automatic selection of training patterns helps to improve the generalization capabilities of MFNN when the training input space is not uniformly distributed. That is, when there are few patterns in some regions while there is redundant data in other regions of the input space. The rest of the paper is organized as follows. Section includes a complete description of the lazy training method proposed in this work to train MFNN in order to improve the generalization ability of these networks. Experimental results are presented in Sec. 3. The lazy method is applied to three different domains: two artificial approximation problems and a real time series prediction problem. The first two domains have been considered in order to show the effectiveness of a selection of patterns based on test samples to obtain better network answers and to reduce the error over unknown samples during the training phase. The third domain is a real problem in which the performance of MFNN must be improved. The proposed method is compared against the traditional way of training MFNN that uses the complete training data available. Finally, some conclusions are drawn in Sec. 4.

2. Lazy Learning Method to Train Multilayer Feedforward Neural Networks

The learning method proposed in this work to train MFNN consists in selecting, from the whole training data, an appropriate subset of patterns in order to improve the answer of the network for a novel pattern. Afterward, the MFNN is trained using this new subset data. The goal is to show that if the MFNN is trained with the most appropriate training patterns, the generalization on the new sample can be improved.

The general idea for the pattern selection is to include several times those patterns close — in terms of the Euclidean distance and some frequency measure — to the novel sample. Thus, the network is trained with the most useful information, discarding those patterns that not only do not provide any knowledge to the network, but might confuse the learning process.

To develop this idea, let us consider q an arbitrary novel pattern described by an n-dimensional vector, $q = (q_1, \dots, q_n)$, where q_i represents the attributes of the instance q. Let X be the whole available training data set:

$$X = \{(x_i, y_i)\ i = 1 \cdots N;\ x_i = (x_{i1}, \dots, x_{in})\,; \\ y_i = (y_{i1}, \dots, y_{im})\}$$

where x_i are the input patterns and y_i their respective target outputs. When a new sample q must be predicted, the MFNN is trained with a subset, which is named X_q, from the whole training data X. The steps to select the training set X_q are the following:

1. A real value, d_k, is associated to each training pattern $(x_k,\ y_k)$. That value is defined in terms of the standard Euclidean distance from the pattern q to each input training pattern. More precisely, it is defined as:

$$d_k = d(x_k, q) = \sqrt{\sum_{i=1}^{n} (x_{ki} - q_i)^2} \\ k = 1 \cdots N\,.$$

   That distance provides a measure to determine the nearest training patterns to the novel pattern.

2. A measure of frequency, f_k, is associated to each training pattern (x_k, y_k). That frequency value is inversely proportional to d_k and it is calculated as follows:

$$f_k = \frac{1}{d_k} \quad k = 1 \cdots N\,.$$

   In order to obtain a relative frequency, the values f_k are normalized in such a way that the sum of the frequencies equals the number of training patterns in X. The relative frequencies, named as fn_k, are obtained by:

$$fn_k = \frac{f_k}{S} \quad \text{where} \quad S = \frac{1}{N}\sum_{k=1}^{N} f_k\,.$$

$$\text{Thus} \sum_{k=1}^{N} fn_k = N\,.$$

3. The values fn_k previously calculated will be used to indicate how many times the training pattern (x_k, y_k) is repeated into the new training subset. Hence, they must be transformed to natural numbers. The most intuitive way to perform that transformation is to take the integer part of the real value fn_k. Thus, the times that the pattern (x_k, y_k) is repeated, named n_k, is calculated as: $n_k = \text{Int}(f_{n_k})$. At this point, each training pattern in X has an associated natural number, n_k, which indicates how many times the pattern (x_k, y_k) has been used to train the MFNN when the new instance q is reached.

4. A new training pattern subset associated to the novel pattern q, X_q, is built up. Given a pattern (x_k, y_k) from the original training set X, that pattern is included in the new subset if the value n_k is higher than zero. In addition, the pattern (x_k, y_k) is placed n_k times randomly in the training set X_q.

5. Once the training patterns are selected, the MFNN is trained using the backpropagation algorithm. The network weights can be randomly initialized or they can be fixed to the weights obtained for the training of the previous sample test. At this point it is necessary to establish a criterion to stop the network training. Since the target output for the new sample is unknown, that criterion must be established using the training patterns. It is difficult to determine when is the most convenient moment to stop the training of the network because it is well known that backpropagation networks could fall down in local minima in the earlier training cycles or they can specialize in the training patterns if the training is carried out during a large period. In this work, three criteria have been combined to decide when to stop the network.

On one hand, a maximum number of learning cycles has been fixed. In real applications, that number must be in accordance with the available time among samples as they are received. On the other hand, the derivative of the training error is measured and the training is stopped when that derivative does not suffer important changes. And finally, the training is also stopped when the answer of the network for the test sample does not suffer important changes. That criterion may avoid an excessive specialization of the network in the training patterns.

3. Experimental Results

Experiments using the lazy method to train MFNN have been carried out with three different problems: two artificial approximation problems and a real prediction problem. First, a theoretical function — a piecewise-defined function — has been chosen as a study case. That function has been chosen because there are a few patterns for which the traditional backpropagation algorithm finds difficulties to generalize them. The goal is to show that with a selection of patterns, the approximations of those patterns can be improved. Secondly and with the purpose of showing that selective learning could improve the approximation capabilities of MFNN, another artificial example has been used. Finally, real data has been used to validate the proposed lazy learning method. The data represents a real time series describing the behavior of the water level at Lagoon Venice and the lazy method is used for the purpose of one-step prediction problem, that is to predict the next sample of data based on previous samples. In addition, that study case is also appropriate to show that the proposed method scales well with the complexity of the problem by considering a learning problem in high-dimensional spaces. The two artificial examples are one-dimensional problems, but to succeed in the prediction of the water level at Lagoon Venice the dimension of the input space must be higher than one.

In order to show the effectiveness of the lazy method proposed in this work, MFNN has also been trained as usual, that is, the network is trained using the whole available training data set, and then it is used to approximate the novel samples.

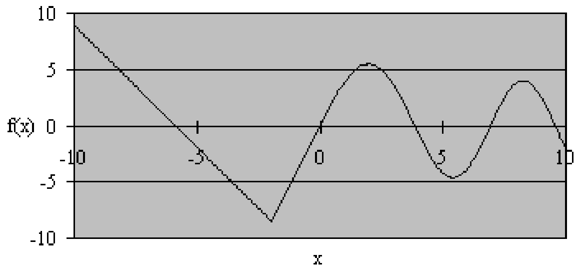


Fig. 1. Piecewise-defined function.

In the next subsections the experimental set-up description and results are presented.

3.1. *Artificial Example 1: A piecewise-defined function*

The piecewise-defined function (see Fig. 1) is a single variable function given by Eq. (1).

$$f(x)=\begin{cases} -2.186x-12.864 & \text{if } -10\leq x<-2 \\ 4.246x & \text{if } -2\leq x<0 \\ 10e^{-0.05x-0.5}\sin((0.03x+0.7)x) & \text{if } 0\leq x\leq 10 \end{cases} . \quad (1)$$

The training set is composed of 120 input–output points randomly generated by an uniform distribution in the interval $[-10, 10]$. Two sets of 80 input–output points generated in the same way are used as test and validation patterns, respectively. Data are normalized in the interval [0,1].

After several simulations varying the number of hidden units, 20 hidden neurons were chosen to approximate the function by a MFNN. Thirty different experiments have been carried out, each one starting with different random weights. The process we have followed in each experiment is described next. First, the MFNN has been trained using the whole training set. In this case, the training and validation errors are measured every learning cycle and the evolution of those errors has been obtained for 1,000,000 cycles. At this point, both training and validation errors were stabilized and learning was stopped. That number of learning cycles is required to get an appropriate generalization of the network. Next, the mean square error over a new test set composed by unseen paterns is measured. The average, standard deviation, minimum, and minimum of the experiments are shown in Table 1.

Table 1. Performance of different training methods for the piecewise-defined function.

	Average Mean Square Error	Standard Deviation	Min	Max
Traditional	0.0063734	0.004683512	0.004052	0.015315
Lazy	0.0021216	0.00048242	0.001549	0.003219

After that, the MFNN is trained with the proposed lazy method. For each test pattern a selection of training patterns is made and a complete training phase is carried out with the training patterns selected following the steps described in Sec. 2. In this case, the training phase for each test sample is carried out until a maximum number of learning cycles (500,000) is reached or until the derivative of training error is near zero or until the output of the network for the test input does not undergo large changes, as it was described in Sec. 2. The mean square error on the test patterns obtained is shown in Table 1.

As it is shown in Table 1, the average mean square error has been significatively reduced when the MFNN is trained with a selection of patterns for each test pattern. Evidently, the computational cost is higher when the lazy method is used, because for each test pattern a complete network is trained. However, as it was previously mentioned, if the MFNN is trained with the whole training set during a large number of cycles, the test error cannot be reduced.

Most of the experiments have a similar behavior. Figure 2 shows the error for each test pattern using both learning methods, traditional and lazy for one of the experiments. As it can be observed, there are two patterns for which the MFNN trained as usual find difficulties to approximate them. The network trained with the whole training data has some difficulties to approximate the points in which the function changes their tendency. The approximation of those patterns is improved when the MFNN is trained with a lazy learning strategy. The use of a reduced and appropriate training set helps the network to find a better approximation. The approximation of the rest of the patterns is also improved, but this is not so relevant. Thus, patterns that are difficult to approximate using the traditional method can be improved when an appropriate selection of patterns is made.

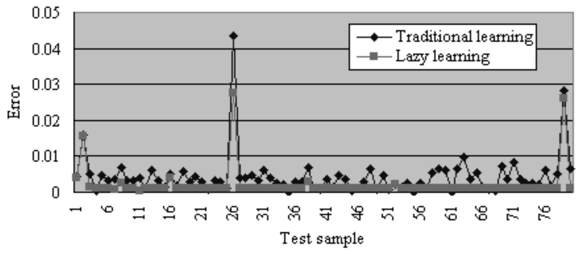
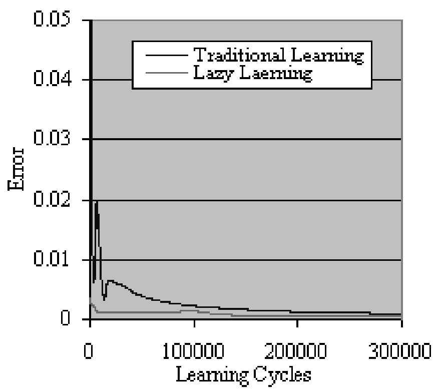


Fig. 2. Errors for each test sample for the piecewise-defined function.

In Fig. 3, the evolution of the mean square errors for the two conflictive test patterns during the training phase of the MFNN is shown for the previous experiment. In that case, the MFNN has been trained using both methods, and the error for each test sample has been measured at every learning cycle. The goal of that experiment is to observe the influence of the patterns used in the training phase over the generalization capability of the network. As it is observed, when the whole training data is used, the test error is higher than if a selection of patterns (the most relevant) is used. In addition, it is also important to point out that the convergence is slower. This implies that the network can easily generalize when it is trained with the most useful patterns and when irrelevant patterns are discarded.

3.2. *Artificial Example 2: A smooth continuous function*

In this subsection the results obtained with an artificial function are shown. The goal of this experimental case is to show the performance of the lazy method in those functions with several changes of tendency (see Fig. 4). That single function is not



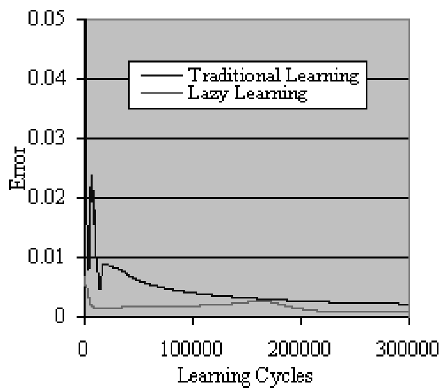


Fig. 3. Evolution of the error during the training for two conflictive patterns.

Table 2. Performance of different training methods for the Smooth Continuous Function.

	Average Mean Square Error	Standard Deviation	Min	Max
Traditional	0.0393358	0.00445235	0.033005	0.046313
Lazy	0.0063768	0.001267353	0.004038	0.008222

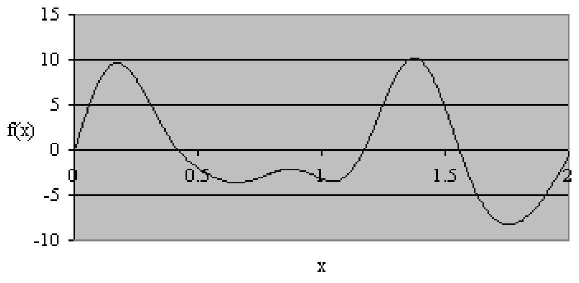


Fig. 4. Smooth continuous function.

a difficult task for the standard MFNN. However, the regions in which the function tendency changes are generally difficult to get accurate approximations. The study case is a single variable function which is given by Eq. (2):

$$f(x) = 9\sin(6x) + 4\sin(10x) + \sin(15x)$$
$$x \in [0,2]\,. \tag{2}$$

In this case, the points have been uniformly generated on the interval [0,2]. From them, two set of 80 input–output points has been randomly extracted and used as validation and test sets, respectively. Data are normalized on [0,1].

After several simulations, ten hidden neurons have been chosen. In this case, 800,000 learning cycles have been required to get an appropriate mean square validation error (see Table 2) when the whole training set is presented to the network. As in the experimental case 1, the error over the training and validation sets are measured every learning cycle and the training is stopped when the validation error is stabilized. Subsequently, the lazy method has been used to approximate the new test set of unseen patterns. In Table 2, some statistics of the error of all the performed experiments are shown both for traditional and lazy methods. As in the previous study case, the mean validation error cannot be reduced when the complete training data is used even if more learning cycles are performed. For the lazy method, a maximum number of learning cycles — 300,000 — is allowed and the training is finished when one of the three criteria described in Sec. 2 are reached.

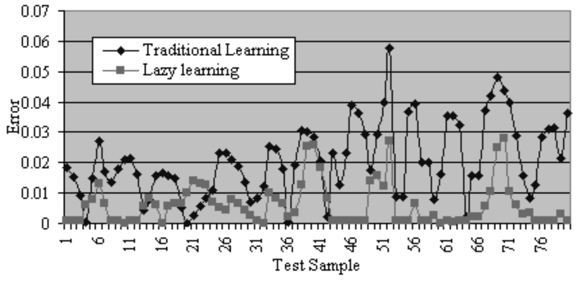


Fig. 5. Errors for each test sample for the smooth continuous function.

The error for each test pattern obtained by the traditional and lazy methods for a typical experiment are shown in Fig. 5. In this case, the superiority of the lazy method is not only observed in a few patterns as in the previous study case (Sec. 3.1), but in most of the test patterns.

3.3. *A real problem: Prediction of water level at Venice Lagoon*

Unusually high tides, or sea surges, result from a combination of chaotic climatic elements in conjunction with the more normal, periodic, tidal systems associated with a particular area. The prediction of such events has always been the subject of intense interest to mankind, not only from a human point of view, but also from an economic one. The water level of Venice Lagoon is a clear example of these events.[24,25] The most famous example of flooding in the Venice Lagoon occurred in November 1966 when, driven by strong winds, the Venice Lagoon rose by nearly 2 m above the normal water level. That phenomenon is known as "high water" and many efforts have been made in Italy to develop systems for predicting sea levels in Venice and mainly for the prediction of the high water phenomenon.[26]

Different approaches have been developed for the purpose of predicting the behavior of sea level at the Lagoon Venice.[26–28] Recently, multilayer feedforward neural networks have been also used to predict the water level[29] obtaining same advantages over linear and traditional models. However, prediction capability of MFNN must be improved, mainly the predictions of the high water phenomenon.

There is a great amount of data representing the behavior of the Venice Lagoon time series. However, the part of data associated to the stable behavior of the water is very abundant as opposed to the part associated to high water phenomena. This situation leads to the following: the MFNN trained with a complete data set is not very accurate in predictions of high water phenomena. It seems natural that if the network is trained with selected patterns, the predictions will improve.

In this work, a training data set of 3000 points corresponding to the level of water measured each hour has been extracted from available data (water level of Venice Lagoon between 1980 and 1994 sampled every hour). This set has been chosen in such a way that both stable situations and high water situations appear represented in the set (see Fig. 6). High-water situations are considered when the level of water is no less than 110 cm. Validation and test samples have also been extracted from the available data and they represent a situation when the level of water is higher than 110 cm (see Fig. 7). Evidently, that situation differs from those appearing in the training set. It is necessary to point out that when the high water occurs, the time series representing the level of water suffers strong variations that are difficult to predict. Hence, it is interesting to predict the high water phenomenon but also what will happen around that phenomenon.

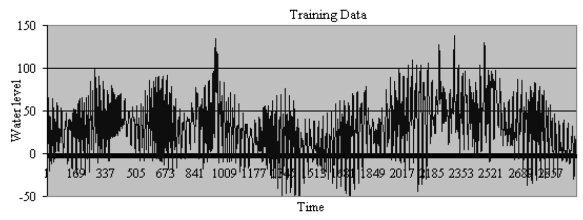


Fig. 6. Water level at Venice Lagoon during four months.

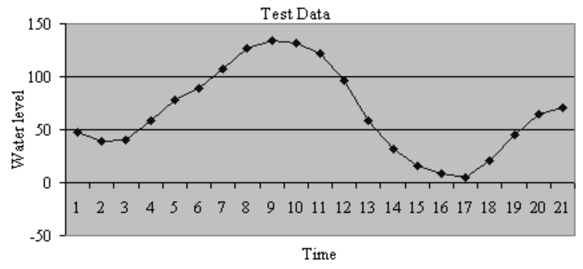


Fig. 7. Water level at Venice Lagoon used as test samples.

Table 3. Performance of different training methods for Venice Lagoon time series.

	Average Mean Square Error	Standard Deviation	Min	Max
Traditional	68.279195607	3.268950185	65.52579465	73.99644855
Lazy	31.038980871	2.253680115	28.4414382	34.9320421

Since the goal in this work is to predict only the next sampling time, a nonlinear model using the six previous sampling times, i.e., data of the six previous hours, may be appropriate. When a long-term prediction has to be made, models with a more extensive information through the input are more convenient.[30] The aim in this context is to observe if a lazy strategy may help to obtain better predictions of high water phenomena. Thus, a MFNN with six input units, 20 hidden neurons and one output neuron representing the level of water at the next instant has been considered and trained with traditional and lazy learning strategies.

The MFNN is trained using the complete training data during 600,000 learning cycles. As in the previous experimental cases, training is stopped when the validation error does not suffer changes. The average, standard deviation, minimum, and maximum of the mean square error over the test samples are shown in Table 3. It is noticed that the validation error goes steady in the cycle 200,000 for all the experiments. That implies that the MFNN has converged and it cannot produce better predictions.

Test samples have also been approximated using a selection of patterns to train the MFNN. In this case, the maximum number of learning cycles for each test pattern is fixed to 100,000, although it is not necessary to reach that number, because if the derivative of the training error is near to zero or the answer of the network does not change, the training is stopped as well. The mean test error is reduced by the lazy strategy.

Figure 8 shows the error for each test pattern in both learning methods for a typical experiment. The errors show that the performance of lazy strategies is better than traditional learning generalization because the errors are generally lower. The high water phenomenon can be predicted more accurately when the MFNN is trained with a selection of patterns, instead of the whole training data.

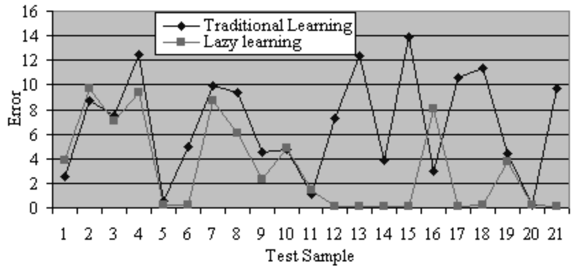


Fig. 8. Errors for each test sample for the water level at Venice Lagoon.

4. Conclusions

The generalization capabilities of MFNNs depends not only on the learning methods used but also on the quality of the data used to train the network. The use of the whole training data available about the domain may not be the most efficient way to obtain the best generalization properties of neural networks. This is specially true when the data presents different regimes. The generalization in those regions in the pattern space that do not follow the general tendency is distorted by the most stable regions.

The lazy learning method presented in this work provides an automatic mechanism to select the most appropriate training data, by using the novel sample to focus the selection process. Thus, those regions that do not follow the general tendency are not ignored by the network, while the rest are equally considered. The results presented in the previous sections show that if MFNNs are trained with such a selection of training patterns, the generalization performance of the network is improved. The selection of the most relevant training patterns — closer patterns in Euclidean distance terms in this case — and the replication of those patterns helps MFNN to obtain better quality on approximation functions and time series prediction.

Besides, the lazy method makes an automatic selection of training patterns for each test sample allowing that the number of patterns to train is variable. If a k-nearest-neighbors strategy were used, several troubles would arise: first, the determination of k, and second, it always selects the same number (k) of patterns, which might not be appropriate in all the cases. Those problems are overcome in the lazy method presented in this work.

However, the proposed method has also some disadvantages. They are mainly given by the use of the Euclidean distance to select the most appropriate patterns. It is well known that in some domains the Euclidean distance does not provide a good similarity measure. Evidently, in those cases, the proposed method will not work in an efficient way. For instance, some classification domains, in which similar patterns belong to different classes, the proposed method will not work. However, the method is flexible to incorporate other different similarity measures.

It is also necessary to mention some aspects related to the computational cost of the lazy learning method proposed. The method involves storing the training data, and finding relevant data to answer a particular test pattern. Thus, the decision about how to generalize is carried out when a test pattern needs to be answered constructing local approximations. That implies a large computational cost because the network has to been trained everytime a new sample is presented. However, the goal of this paper is to improve the generalization capability even if the computational cost is higher. Moreover, in applications (for instance, time series prediction) in which enough time is available between samples to train the network, the computational cost required by the method is not a disadvantage, as long as the generalization capability is improved.

Besides, if novel samples arrive incrementally and it is desired to use those new samples for training purposes, then lazy methods are not worse than eager methods, because both have to generate a network everytime a new pattern arrives. Actually, in that case, lazy methods are better in terms of computational cost, because they use fewer patterns to train the network, hence training time will be smaller.

References

1. D. Rumelhart, G. Hinton and R. J. Williams 1986, “Learning internal representations by error propagation,” in *Parallel Distributed Processing* (MIT Press, Cambridge).
2. T. Denoeux and R. Lengelle 1993, “Initializing back propagation networks using prototypes,” *Neural Networks* **6**(3), 351–363.
3. X. H. Hu and G. A. Chen 1997, “Efficient back-propagation learning using optimal learning rate and momentum,” *Neural Networks* **10**(3), 517–527.
4. C. Schittenkopf, G. Deco and W. Brauer 1997, “Two strategies to avoid overfitting in feedforward neural networks,” *Neural Networks* **10**(3), 505–516.
5. R. Battiti 1992, “First and second-order methods for learning: Between steepest descent and newton’s methods,” *Neural Computation* **4**, 141–166.
6. B. E. Rosen and J. M. Goodwin 1997,*Optimizing Neural networks using Very fast Simulated Annealing*, Parallel and Scientific Computations.
7. K. Chen, L. Xu and H. Chi 1999, “Improved learning algorithms for mixture of experts in multiclass classification,” *Neural Networks* **12**, 1229–1252.
8. M. I. Jordan and R. A. Jacobs 1994, “Hierarchical mixture of experts and the EM algorithm,” *Neural Computation* **6**(2), 181–214.
9. Y. S. Abu-Mostafa 1989, “The Vapnik–Chervonenkis dimension: Information versus complexity in learning,” *Neural Conputation* **1**, 312–317.
10. D. Cohn, L. Atlas and R. Ladner 1994, “Improving generalization with active learning,” *Macine Learning* **15**, 201–221.
11. S. Vijayakumar and H. Ogawa 1999, “Improving generalization ability through active learning,” *IEICE Transactions on Information and Sytems* **E82-D**(2), 480–487.
12. K. Huyser and A. M. Horowitz 1988, “Generalization in connectionist networks that realise Boolean functions,” in *Proc. 1988 Connectionist Models Summer School*, eds. D. Touretzky, G. Hinton and T. Sejnowski (Morgan Kaufman, Palo Alto, CA), pp. 191–200.
13. M. Wann, T. Hediger and N. N. Greenbaun 1990, “The influence of training sets on generalization in feed-forward neural netwok,” *Proc. of the International Joint Conference on Neural Networks*, vol. 3, San Diego, pp. 137–142.
14. R. Cheung, I. Lusting and A. L. Kornhauser 1992, “Relative effectiveness of training set patterns for back propagation,” in *Proc. of the IEEE Intrnational Conference on Neural Networks*, vol. 1, San Diego, pp. 673–678.

15. M. Hasenjager and H. Ritter 1998, "Active learning with local models," *Neural Processing Letters* **7**, 107–117.
16. A. P. Engelbrecht and I. Cloete 1998, "Selective learning using sensitivity analysis," *IEEE International Conference on Neural Networks*, pp. 1150–1155.
17. C. G. Atkeson, A. W. Moore and S. Schaal 1997, "Locally weighted learning," *Artificial Intelligence Review* **11**, 11–73.
18. D. Wettschereck, D. W. Aha and T. Mohri 1997, "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artificial Intelligence Review* **11**, 273–314.
19. B. V. Dasarathy (ed.) 1991, *Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques* (IEEE Computer Society Press, Los Alamitos, CA).
20. J. D. Kelly and L. Davis 1991, "A hybrid genetic algorithm for classification," in *Proceedings of the Twelfth International Joint Conference On Artificial Intelligence*, Sydney, Australia (Morgan Kaufman), pp. 645–650.
21. D. W. Aha 1992, "Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms," *International Journal of Man-Machine Studies* **36**, 267–287.
22. V. Vapnik 1992, "Principles of risk minimization for learning theory," in *Advances in Neural Infromation Processing Systems*, *4*, eds. J. E. Moody, S. J. Hanson and R. P. Lippman (Morgan Kaufman, San Mateo, CA), pp. 831–838.
23. V. Vapnik and L. Bottou 1993, "Local algoritms for pattern recognition and dependencies estimation," *Neural Computation* **5**(6), 893–909.
24. E. Moretti and A. Tomasin 1984, "Un contributo metematico all-elaborazione previsionale dei dati di marea a Venecia," *Boll. Ocean. Teor. Appl.* **1**, 45–61.
25. A. Michelato, R. Mosetti and D. Viezzoli 1983, "Statistical forescasting of strong surges and apllication to the lagoon of venice," *Boll. Ocean. Teor. Appl.* **1**, 67–83.
26. A. Tomasin 1973, "A computer simulation of the adriact sea for the study of its dynamics and for the forecasting of floods in the town of Venice," *Comp. Phys. Comm.* **5**, 51.
27. G. Vittori 1992, "On the choatic features of tide elevation in the lagoon Venice," in *Proc. of the ICCE-92*, 23rd International Conference on Coastal Engineering, 4–9, Venice, pp. 361–362.
28. L. M. Serio Bergamasco, A. R. Osborne and L. Cavaleri 1995, "Finite correlation dimension and positive Lyapunov exponents for surface wave data in the adriatic sea near Venice," *Fractals* **3**, 55–78.
29. J. M. Zaldívar, E. Gutiérrez, I. M. Galván, F. Strozzi and A. Tomasin 2000, "Forecasting high waters at Venice Lagoon using chaotic time series analysis and nonlinear neural networks," *Journal of Hydroinformatics* **2**(1), 61–84.
30. I. M. Galván, J. M. Alonso and P. Isasi 2000, "Improving multi-step time series prediction with recurrent neural modelling," *New Frontiers in Computational Intelligence and its Applications*, ed. Masuod Mohammadian (IOS Press).