Computing and Informatics, Vol. 25, 2006, 1–15

# IMPROVING THE GENERALIZATION ABILITY OF RBNN USING A SELECTIVE STRATEGY BASED ON THE GAUSSIAN KERNEL FUNCTION

José M. Valls, Inés M. Galván, Pedro Isasi

*Departamento de Informática, Universidad Carlos III de Madrid*
*Avenida de la Universidad 30*
*28911 Leganés (Madrid), Spain*
*e-mail:* {jvalls, igalvan}@inf.uc3m.es, isasi@ia.uc3m.es

Manuscript received 17 June 2005; revised 20 December 2005
Communicated by Vladimír Kvasnička

**Abstract.** Radial Basis Neural Networks have been successfully used in many applications due, mainly, to their fast convergence properties. However, the level of generalization is heavily dependent on the quality of the training data. It has been shown that with careful dynamic selection of training patterns, better generalization performance may be obtained. In this paper, a learning method is presented, that automatically selects the training patterns more appropriate to the new test sample. The method follows a selective learning strategy, in the sense that it builds approximations centered around the novel sample. This training method uses a Gaussian kernel function in order to decide the relevance of each training pattern depending on its similarity to the novel sample. The proposed method has been applied to three different domains: an artificial approximation problem and two time series prediction problems. Results have been compared to standard training method using the complete training data set and the new method shows better generalization abilities.

**Keywords:** Radial Basis Neural Networks, generalization ability, selective learning, kernel functions

## 1 INTRODUCTION

Radial Basis Neural Networks (RBNN) [9, 4] are originated from the use of radial basis functions, as the Gaussian functions, in the solution of the real multivariate

interpolation problem [2, 14]. As the Multilayer perceptron (MLP) they can approximate any regular function [12]. Due to its local behavior and to the linear nature of its output layer, their training is faster than MLP training [12] and this fact makes them useful for a wide variety of applications. Usually, the generalization capability of RBNN is poor because they are too specialized in the training data set. Hidden neurons represent regions of the input space and therefore the search of the appropriate hidden neurons to get good generalization properties may be an arduous task [6].

In order to improve the generalization ability of the networks, some authors have developed optimization methods to allocate the centers of the RBNN and to determine their architecture [13, 18], whereas others have paid attention to the nature and size of the training set. There is no guarantee that the generalization performance is improved by increasing the training set size [1]. In general, only those examples which are most likely to help the network solve the problem should be chosen. It has been shown that with a careful dynamic selection of training patterns, better generalization performance may be obtained [3].

The idea of selecting dynamically – from the available data about the domain – the patterns to train the network 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 novel samples, instead of being based on other training patterns. Thus, the network will use its current knowledge of each new sample in order to decide what patterns are going to be selected for training.

The learning method proposed in this work to train RBNN consists of recognizing, from the whole training data set, the most relevant patterns for each new sample to be processed, discarding data that could worsen the generalization of the new pattern. This subset of useful patterns is used to train a RBNN, and therefore deferring the training until a test pattern is received. Thus, taking advantage of the fast convergence of this kind of networks, a complete RBNN is trained for each test sample. The relevance of a pattern in the training data is obtained using a weighting measure generated by a kernel function. The value assigned by this function to the training pattern depends on its Euclidean distance to the test sample in such a way that the closest patterns get the highest weights.

## 2 WEIGHTED SELECTION OF TRAINING PATTERNS

The method proposed in this work to train RBNN involves storing the training data in memory, and finding relevant data to answer to new patterns. Thus, the decision about how to generalize is carried out when a novel pattern needs to be answered, constructing local approximations with a subset of training patterns. With this purpose, when a novel sample is received, the learning method selects, from the whole training data, an appropriate subset of training patterns in order to improve the answer of the network for that novel pattern. Afterwards, the RBNN is trained using this new subset of selected data. The goal is to show that, if the RBNN is

trained with the most appropriate training patterns, the generalization on the new sample can be improved.

The general idea for training patterns selection is to select those patterns close (in terms of the Euclidean distance and some weighting measure) to the novel sample, and to include once or more times those selected patterns. 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.

When a novel sample is gathered, the selection of patterns is carried out establishing a weight for each training pattern, depending on the distance of this pattern to the novel sample. That weight is calculated using a weighting function or kernel function, which must have the following characteristics:

- The function reaches its maximum value when the distance to the novel sample is null.
- The function decreases smoothly as this distance increases.

With this purpose, the kernel function used in this work, to select the most relevant patterns, is the Gaussian function. This function assigns to each training input pattern $\mathbf{x_k}$ a real value or weight according to the following equation:

$$K(\mathbf{x_k}) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{d(\mathbf{q},\mathbf{x_k})^2}{2\sigma^2}} \tag{1}$$

where $\mathbf{q}$ represents the novel sample, $\sigma$ is a parameter named "width", which indicates the width of the Gaussian function, and $d(\mathbf{q}, \mathbf{x_k})$ is the Euclidean distance from the novel sample to the training input pattern $\mathbf{x_k}$. In order to simplify the notation this distance will be represented by $d_k$ and is defined as usual:

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

The weighting function given by Equation (1) allows to associate a weight to each training pattern, reaching its maximum value when the distance is zero and decreasing smoothly as distance increases. As the width parameter decreases, the shape of the function becomes sharper, being higher its maximum value; thus, less training examples will be selected although the weights of the nearest ones will be higher.

The weight values $K(\mathbf{x_k})$ are used to indicate how many times the training pattern $(\mathbf{x_k}, \mathbf{y_k})$, where $\mathbf{y_k}$ is the target output for the input $\mathbf{x_k}$, will be included into the training subset associated to the novel sample $\mathbf{q}$. Hence, those real values must be transformed into natural numbers. The most intuitive way consists on taking the integer part of $K(\mathbf{x_k})$. Thus, each training pattern will have an associated natural number, $n_k = int(K(\mathbf{x_k}))$, which indicates how many times the pattern $(\mathbf{x_k}, \mathbf{y_k})$ will be used to train the RBNN when the new instance $\mathbf{q}$ is reached. If $n_k = 0$ then the $k^{th}$ pattern is not selected and not used to train the RBNN.

Once the training patterns are selected according to the weighting function (Equation (1)), the RBNN is trained with the new subset of patterns. The usual way of training a RBNN is based on the hybrid learning method introduced by Moody and Darken [9]. It involves determining the neuron centers, the dilations or widths, and the weights in two separate stages: a self-organized stage to estimate the centers locations and widths of the radial basis functions on the hidden layer and a supervised stage where the linear weights of the output layer are determined.

The centers $C_i$ are calculated in an unsupervised way using the K-means algorithm to classify the input space. In this context, the K- means algorithm is applied only to the selected training patterns for the new sample. The neurons dilations $d_i$ or widths are evaluated as the geometric mean of the distances from each neuron center to its two nearest centers

$$d_i = \sqrt{\|C_i - C_t\| \ \|C_i - C_s\|} \tag{3}$$

where $C_t$ and $C_s$ are the two nearest centers to center $C_i$.

Finally, the weights of output layer of the RBNN are estimated in a supervised way to minimize the mean square error $E$ measured over the selected training subset:

$$E = \frac{1}{R} \sum_{r=1}^{R} e_r \tag{4}$$

where $R$ is the number of patterns selected and $e_r$ is the error committed by the network for the pattern $x_r$, given by

$$e_r = \frac{1}{2} \sum_{i=1}^{m} (\tilde{y}_{ri} - y_{ri})^2 \tag{5}$$

where $y_r = (y_{r1}, ..., y_{rm})$ and $\tilde{y}_r = (\tilde{y}_{r1}, \ldots, \tilde{y}_{rm})$ are the desired output vector and the output vector of the network, respectively.

In the next, the sequential structure of the selective learning method is summarized.

For each new sample $\mathbf{q}$,

1. The standard Euclidean distances $d_k$ from the pattern $\mathbf{q}$ to each input training pattern are calculated using the equation 2.
2. The Kernel function $K()$, given by Equation 1, is used to calculate a weight for each training pattern from its distance to the new pattern.
3. The real numbers $K(\mathbf{x_k})$ are transformed into natural numbers $n_k$ taking its integer part.
4. A training pattern subset associated to the novel pattern $\mathbf{q}$, named $X_q$, is built up as follows:

   (a) Given a pattern $(\mathbf{x_k}, \mathbf{y_k})$ from the original training set, that pattern is included in the new subset if the value $n_k$ is higher than zero.

(b) In addition, the pattern $(\mathbf{x_k}, \mathbf{y_k})$ is placed $n_k$ times randomly in the training set $X_q$.

5. The RBNN is trained using the new subset $X_q$.

## 3 EXPERIMENTAL VALIDATION

In this section, the selective learning method based on the weighting Gaussian function has been applied to three different problems. Two of them are domains widely used in the literature of RBNN: an artificial regression problem – the Hermite Polynomial – and an artificial time series prediction problem – the Mackey-Glass time series. The third one is a real time series prediction problem describing the behavior of the water level at Venice Lagoon.

In the next subsections, the features of the different problems, the experimental set-up description and results obtained for each domain are presented and analyzed.

### 3.1 Experimental Definition

The selective learning method to train RBNN has been validated in different domains: the Hermite Polynomial, the Mackey-Glass time series and the water level at Venice Lagoon times series. In the next the characteristics of all of them are presented.

**The Hermite Polynomial approximation**

The Hermite Polynomial is a one-dimensional approximation problem given by the following equation:

$$f(x) = 1.1(1 - x + 2x^2)e^{-\frac{1}{2}x^2}. \tag{6}$$

The training and test patterns are obtained using a random sampling with a uniform distribution over the interval $[-4, 4]$. As in the works that appear in the literature about this domain [5, 11, 18], 40 and 200 input-output points are generated for the training and test data sets, respectively. Both sets have been normalized in the interval $[0, 1]$.

**The Mackey-Glass time series prediction**

This time series is widely regarded as a benchmark for comparing the generalization ability of RBNN [13, 18, 9, 17]. It is a chaotic time series created by the Mackey-Glass delay-difference equation [7]:

$$\frac{dx(t)}{dt} = -bx(t) + a\frac{x(t-\tau)}{1 + x(t-\tau)^{10}}. \tag{7}$$

Following the studies mentioned above, the series has been generated using the next values for the parameters: $a = 0.2$, $b = 0.1$, and $\tau = 17$. The task for the

RBNN is to predict the value of the time series at point $x[t+P]$ from the earlier points $(x[t], x[t-6], x[t-12], x[t-18])$. The number of sample steps $P$ has been set to 50, as in [18]. Thus, the function (whose dimension is 4) to be learned by the network is:

$$x(t) = f(x(t-50), x(t-50-6), x(t-50-12), x(t-50-18)). \quad (8)$$

Fixing $x(0) = 0$, 5 000 values of the time series are generated using the Equation (8). The initial 3 500 samples are discarded in order to avoid the initialization transients. 1 000 data points, corresponding to the sample time between 3 500 and 4 499, have been chosen for the training set. The test set is composed by the points corresponding to the time interval $[4\,500, 5\,000]$. All data points are normalized in the interval $[0, 1]$.

**Prediction of water level at Venice Lagoon**

The time series describing the water level at Venice Lagoon represents a real time series prediction problem. The prediction of high tides has always been the subject of intense interest to mankind, not only from a human point of view, but also from an economic one, and the water level of Venice Lagoon is a clear example of these events [10, 8]. 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 meters 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 level in Venice, mainly for the prediction of the high water phenomenon [15].

Different approaches have been developed for the purpose of predicting the behavior of sea level at the Venice Lagoon [15, 16]. Multilayer feedforward neural networks have also been used to predict the water level [19] obtaining same advantages over linear and traditional models.

The goal in this work is to predict only the next sampling time and a nonlinear model using the six previous sampling times, i.e. data of the six previous hours, may be appropriate. Thus, the function to be learned, whose dimension is 6, is:

$$x(t) = f(x(t-1), x(t-2), x(t-3), x(t-4), x(t-5), x(t-6)). \quad (9)$$

A training data set of 3000 points corresponding to the water level 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 Figure 1). High-water situations are considered when the level of water is not lower than 110 cm. Test samples have also been extracted from the available data and they represent a situation when the water level is higher than 110 cm (see Figure 2).

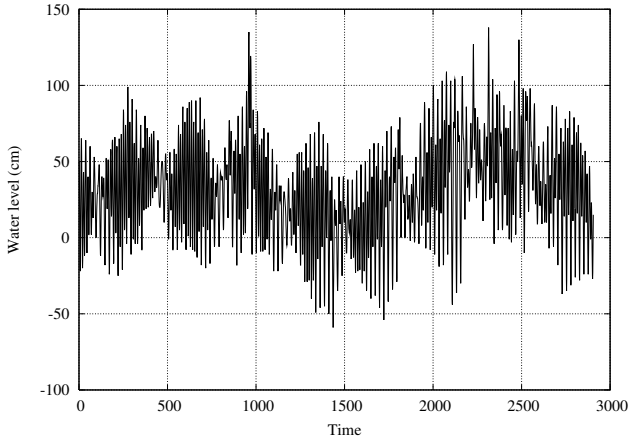


Fig. 1. Water level at Venice Lagoon during four months. Training set.

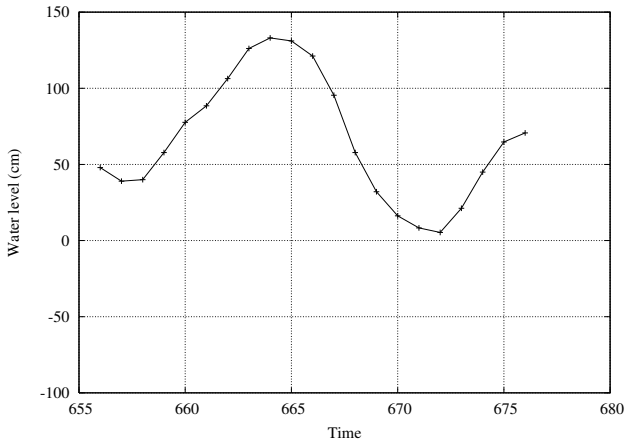


Fig. 2. Water level at Venice Lagoon. Test set.

### 3.2 Experimental Conditions

To apply the selective learning method proposed in this work to train RBNN, the width of the Gaussian Kernel function must be fixed as an external parameter of the method. That parameter determines the number of patterns selected to train the RBNN for each sample test. Hence, it is important to study the influence of that parameter in the performance of the selective learning method. With this purpose a set of experiments varying the value of the width parameter has been carried out and the performance of the method has been measured. In all cases, the width parameter is varied from 0.0 to 0.4 with a step of 0.05. Those maximum an minimum

values of the width have been chosen in such a way that the shape of the Gaussian function allows the selection of some training patterns.

In addition, experiments varying the number of hidden neurons have also been realized in order to observe whether the number of hidden neurons is a crucial parameter when a selection of training patters is made. For all domains, architectures with 5, 9, 13 and 17 hidden neurons have been tested. The different RBNN architectures have been trained during 300 learning cycles to reach the convergence of the networks.

In order to show whether the selective learning method is able to improve the generalization capability of RBNN, a set of experiments training the RBNN as usual – traditional learning method – has been carried out. That is, networks are also trained using the whole available training data set, and after the training phase they are used to approximate the test o validation samples. For this set of experiments, RBNN with different number of hidden neurons have been trained until they reach the convergence.

In both cases – selective and traditional learning methods – the generalization capability of the RBNN has been measured in terms of the mean error over the test data set, which is given by

$$e = \frac{1}{n} \sum_{k=1}^{n} e_k \tag{10}$$

where $n$ is the number of patterns in the test set and $e_k$ represents the error for the $k_{th}$ test pattern, calculated as $e_k = \mid \tilde{y}_k - y_k \mid$, being $\tilde{y}_k$ the output of the network and $y_k$ the desired output for that pattern. In all the studied domains the output is a real number.

### 3.3 Experimental Results

In this subsection, the results obtained with the selective learning Gaussian method to train the RBNN for the different application domains, are shown and analyzed. The results will be also compared with those obtained when RBNN are trained as usual.

#### 3.3.1 Approximation of Hermite Polynomial

The mean errors obtained for this approximation problem when a selection of patterns is made to train the RBNN are shown in Table 1. In Figure 3 the variation of those errors regard to the width parameter for the different architecture of RBNN can be graphically appreciated.

It is possible to observe, on one hand, that errors decrease as the width increases, although they reach high values if the width is bigger than 0.35. On the other hand, it is also observed that architectures with few hidden neurons provide the smallest errors. When the width parameter is small, the Gaussian function is tight and high and few patterns are selected. Hence, the performance of the networks with

| | Hidden neurons | | | |
|---|---|---|---|---|
| Width | 5 | 9 | 13 | 17 |
| 0.01 | 0.0982 | 0.2182 | 0.3432 | 0.3821 |
| 0.05 | 0.0283 | 0.1633 | 0.3238 | 0.3849 |
| 0.1 | 0.0121 | 0.1180 | 0.2352 | 0.3891 |
| 0.15 | 0.0134 | 0.0944 | 0.2113 | 0.3519 |
| 0.2 | 0.0133 | 0.0738 | 0.2133 | 0.2672 |
| 0.25 | 0.0123 | 0.0566 | 0.1832 | 0.3044 |
| 0.3 | 0.0134 | 0.0887 | 0.1923 | 0.2920 |
| 0.35 | 0.0135 | 0.0973 | 0.2398 | 0.3412 |
| 0.4 | 0.4513 | 0.4492 | 0.4457 | 0.4321 |

Table 1. Hermite Polynomial: Mean errors with the selective learning

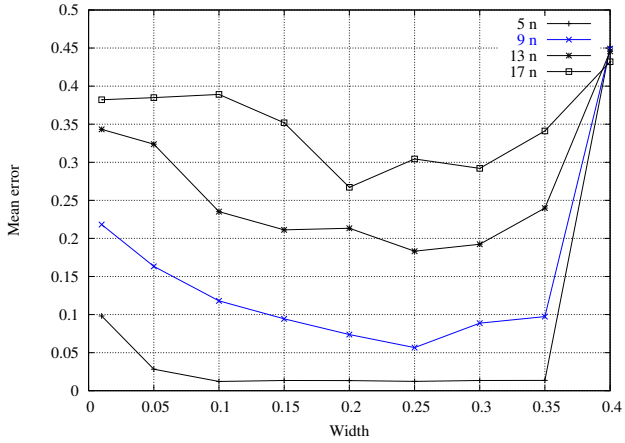


Fig. 3. Hermite Polynomial: Mean errors with the selective learning

few hidden neurons is better. When the width parameter increases, the Gaussian function is wider and lower, selecting more training patterns and improving the performance of the network. However, if the width parameter is bigger than 0.35, a lot of patterns are selected and the generalization capabilities of the networks gets worse.

### 3.3.2 Prediction of Mackey-Glass Time Series

In the same way as in the previous case, Table 2 shows the mean test errors obtained for different values of the width parameter and different architectures. Those results are graphically represented in Figure 4.

In the obtained results, it is possible to observe that the performance of the selective learning strategy to predict the Mackey-Glass time series does not depend significantly on the width parameter and on the number of hidden neurons in the

network. There is a wide interval of width values in which the errors are very similar for all architectures. Only when width parameter is fixed to small values, for instance 0.01, or to values bigger than 0.35, the generalization of the selective method is very poor as also happened in the previous domain. This is due to the small number of selected patterns, insufficient to construct an approximation.

| | Hidden neurons | | | |
|---|---|---|---|---|
| Width | 5 | 9 | 13 | 17 |
| 0.01 | 0.3047 | 0.3138 | 0.3217 | 0.3376 |
| 0.05 | 0.0463 | 0.0467 | 0.0622 | 0.0808 |
| 0.1 | 0.0434 | 0.0349 | 0.0386 | 0.0466 |
| 0.15 | 0.0535 | 0.0381 | 0.0359 | 0.0393 |
| 0.2 | 0.0652 | 0.0398 | 0.0374 | 0.0411 |
| 0.25 | 0.0664 | 0.0425 | 0.0355 | 0.0406 |
| 0.3 | 0.0647 | 0.0417 | 0.0377 | 0.0413 |
| 0.35 | 0.0605 | 0.0427 | 0.0417 | 0.0511 |
| 0.4 | 0.5752 | 0.5752 | 0.5752 | 0.5752 |

Table 2. Mean errors with the selective learning method. Mackey-Glass time series.

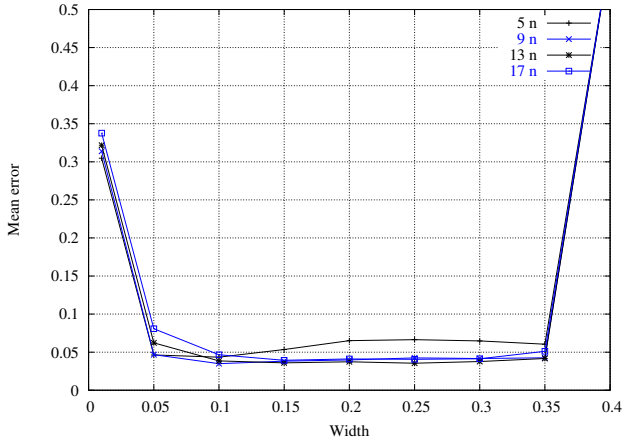


Fig. 4. Mean errors with the selective learning method. Mackey-Glass time series.

### 3.3.3 Prediction of Water Level at Venice Lagoon

The results obtained in this application domain are shown in Table 3 and displayed in Figure 5. As in the previous prediction problem, it is possible to observe that the performance of selective learning method is, generally, not influenced by the value of the width parameter and by the number of hidden neurons fixed in the network.

The tendency is similar to the one observed with the Mackey-Glass time series: the mean errors maintain its value nearly constant for all architectures when the width parameter is bigger than 0.05 and lower than 0.35. As in the previous case, the big errors committed by the networks when the width value is outside of that interval are due to the shortage of selected training patterns.

| | Hidden neurons | | | |
|---|---|---|---|---|
| Width | 5 | 9 | 13 | 17 |
| 0.01 | 0.6344 | 0.6344 | 0.6344 | 0.6344 |
| 0.05 | 0.1490 | 0.1315 | 0.1324 | 0.1350 |
| 0.1 | 0.1059 | 0.1120 | 0.1255 | 0.1259 |
| 0.15 | 0.1239 | 0.1180 | 0.1072 | 0.0905 |
| 0.2 | 0.1300 | 0.1155 | 0.1164 | 0.1068 |
| 0.25 | 0.1342 | 0.1191 | 0.1138 | 0.1118 |
| 0.3 | 0.1298 | 0.1170 | 0.1115 | 0.1337 |
| 0.35 | 0.1023 | 0.1134 | 0.1332 | 0.1357 |
| 0.4 | 0.6344 | 0.6344 | 0.6344 | 0.6344 |

Table 3. Mean errors with the selective learning method. Venice lagoon time series.

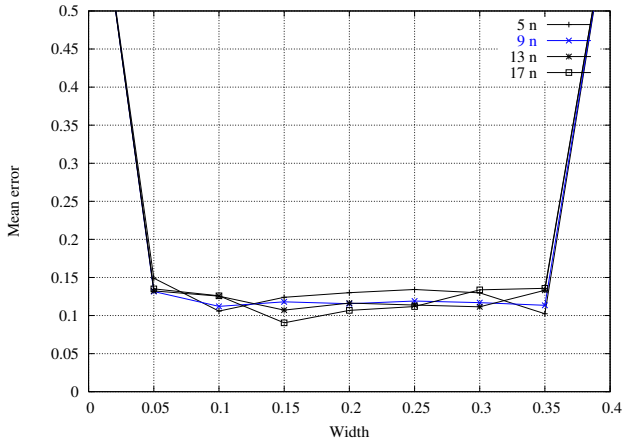


Fig. 5. Mean errors with the selective learning method. Venice lagoon time series.

The experiments seem to point out that the width parameter does not affect the performance of the network significantly if the width parameter neither too high nor too small. The Gaussian function seems to be a good way of selecting training patterns in a lazy way because the width of the function is not a critical parameter.

### 3.3.4 Selective Learning Versus Traditional Learning

Having the aim of comparing both learning strategies, selective and traditional ones, RBNN with different number of hidden neurons have been trained, using the whole training data until the convergence of the network has been reached. In Table 4, test mean errors obtained for different application domains are shown. It is important to notice that the test mean errors cannot be improved even if more learning cycles are performed using the whole training data set. In this case, the number of hidden neurons must be higher because very poor results are obtained when using less than 10 neurons.

| Hidden Neurons | Hermite Polynomial | Mackey-Glass time series | Venice Lagoon time series |
|---|---|---|---|
| 10 | 0.1157 | 0.1330 | 0.2365 |
| 20 | 0.0270 | 0.1356 | 0.1341 |
| 30 | 0.0213 | 0.1271 | 0.1117 |
| 40 | 0.0190 | 0.1277 | 0.1120 |
| 50 | 0.0227 | 0.1123 | 0.0961 |
| 60 | 0.0221 | 0.1052 | 0.1029 |
| 70 | 0.0206 | 0.1274 | 0.1022 |
| 80 | 0.0226 | 0.1115 | 0.1065 |
| 90 | 0.0262 | 0.1177 | 0.1214 |
| 100 | 0.0214 | 0.1163 | 0.1254 |
| 110 | 0.0214 | 0.1027 | 0.1290 |
| 120 | 0.0233 | 0.1114 | 0.1380 |
| 130 | 0.0251 | 0.1277 | 0.1415 |

Table 4. Test mean errors with the traditional learning

In Table 5, the best results obtained in the different domains for both methods, selective and traditional ones, are shown.

| | Hermite Polynomial | Mackey-Glass time series | Venice Lagoon time series |
|---|---|---|---|
| Selective Method | 0.0121 $\sigma = 0.1$, 5 neurons | 0.0349 $\sigma = 0.1$, 9 neurons | 0.0905 $\sigma = 0.15$, 17 neurons |
| Traditional Method | 0.0190 40 neurons | 0.1027 110 neurons | 0.0961 50 neurons |

Table 5. Selective learning versus traditional learning

As it is possible to observe, in the Hermite polynomial and Mackey-Glass time series the performance of RBNN can be enhanced when a weighted selection of training patterns is made, reaching better precision levels. Moreover, in spite of the different values of the parameters, nearly all the results achieved with the selective method are better than the best result obtained by the traditional method. Due

to the special characteristics of the Venice lagoon domain, the selection of patterns made by the Gaussian function might not be appropriate since more training examples might be needed. As can be seen in the domain description (Section 3.1), the test set patterns represent a high water situation, this kind of situations being scarcely represented in the training set.

## 4 CONCLUSIONS

The generalization capabilities of RBNN depend not only on the learning methods but also on the quality of the data used to train the network. The learning method presented in this work provides an automatic mechanism to select the most appropriate training data in terms of the novel sample. It is inspired by the idea that the use of the whole training data available about the domain might not be the best choice to reach good generalization properties of RBNN.

The results presented in the previous sections show that if RBNN 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, taken from the neighborhood region around the novel sample, and the replication of those patterns helps RBNN obtain better results on approximation functions and time series prediction. The Gaussian kernel function decides the relevance of training patterns depending on its similarity to the novel pattern, measuring this similarity in terms of the Euclidean distance. The number of patterns selected depends on the width of the Gaussian function. If the width parameter is small, only patterns very close to the new sample will be selected and they will be repeated a lot of times; if the width parameter is big, more training patterns will be selected but they will be repeated less times. However, the experimental results show that the performance of the selective strategy does not depend significantly on the width parameter. There is a wide interval of width values in which the errors are very similar. Only when the width parameter is too small or too big, the generalization of the selective method is very poor.

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. Anyway, 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. This fact implies a large computational cost because the network has to be trained each time a new sample is presented. However, the goal of this paper is to improve the generalization capability even if the computational cost is higher. In some 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.

## REFERENCES


[1] Abu-Mostafa, Y.: The Vapnik-Chervonenkis Dimension: Information Versus Complexity in Learning. Neural Computation, Vol. 1, 1989, pp. 312–317.

[2] Broomhead, D.—Lowe, D.: Multivariable Functional Interpolation and Adaptative Networks. Complex Systems, Vol. 2, 1988, pp. 321–355.

[3] Cohn, D.—Atlas, L.—Ladner, R.: Improving Generalization with Active Learning. Machine Learning, Vol. 15, 1994, pp. 201–221.

[4] Ghosh, J.—Nag, A.: An Overview of Radial Basis Function Networks. R. J. Howlett and L. C. Jain (Eds), Physica Verlag, 2000.

[5] Leonardis, A.—Bischof, H.: An Efficient MDL-Based Construction of RBF Networks. Neural Networks, Vol. 11, 1998, pp. 963–973.

[6] Lowe, D.: Adaptative Radial Basis Function nonlinearities, and the Problem of Generalization. First IEE International Conference on Artificial Neural Networks, pp. 171–175, 1989.

[7] Mackey, M.—Glass, L.: Oscillation and Chaos in Physiological Control Systems. Science, Vol. 197, 1977, pp. 287–289.

[8] Michelato, A.—Mosetti, R.—Viezzoli, D.: Statistical Forecasting of Strong Surges and Aplication to the Lagoon of Venice. Boll. Ocean. Teor. Appl., Vol. 1, 1983, pp. 67–83.

[9] Moody, J.—Darken, C.: Fast Learning in Networks of Locally Tuned Processing Units. Neural Computation, Vol. 1, 1989, pp. 281–294.

[10] Moretti, E.—Tomasin, A.: Un Contributo Matematico All-Elaborazione Previsionale dei Dati di Marea a Venecia. Boll. Ocean. Teor. Appl., Vol. 1, 1984, pp. 45–61.

[11] Orr, M. J. L.: Introduction to Radial Basis Neural Networks. Technical Report. Centre for Cognitive Science, University of Edinburgh, 1996.

[12] Park, J.—Sandberg, I. W.: Universal Approximation and Radial-Basis-Function Networks. Neural Computation, Vol. 5, 1993, pp. 305–316.

[13] Platt, J.: A Resource-Allocating Network for Function Interpolation. Neural Computation, Vol. 3, 1991, pp. 213–225.

[14] Powell, M.: The Theory of Radial Basis Function Approximation in 1990. Advances in Numerical Analysis, Vol. 3, 1992, pp. 105–210.

[15] Tomasin, A.: A Computer Simulation of the Adriatic Sea for the Study of Its Dynamics and for the Forecasting of Floods in the Town of Venice. Comp. Phys. Comm., Vol. 5, 1973, pp. 51.

[16] Vittori, G.: On the Chaotic Features of Tide Elevation in the Lagoon Venice. Proc. of the ICCE-92, 23rd International Coference on Coastal Engineering, pp. 4–9, 1992.

[17] Whitehead, B. A.—Choate, T. D.: Cooperative – Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction. IEEE Transactions on Neural Networks, Vol. 5, 1995, pp. 15–23.

[18] Yingwei, L.—Sundararajan, N.—Saratchandran, P.: A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks. Neural Computation, Vol. 9, 1997, pp. 461–478.

[19] Zaldvar, J.—Gutiérrez, E.—Galván, I.—Strozzi, F.—Tomasin, A.: Forecasting High Waters at Venice Lagoon Using Chaotic Time Series Analysis and Nonlinear Neural Networks. Journal of Hydroinformatics, Vol. 2, 2000, pp. 61–84.

**José M. Valls** received his Ph. D. in computer science at Universidad Carlos III of Madrid (Spain) in 2004. He joined the Computer Science Department at the same university in 1998, being associate professor since 2004. He is enrolled in the Neural Networks and Evolutionary Computation Laboratory of this university. His current research focuses on the application of neural networks, evolutionary computation and other soft computing techniques to engineering problems.

**Inés M. Galván** received a doctorate-fellowship as research scientist in the European Commission, Joint Research Centre Ispra (Italy) from 1992 to 1995. She received her Ph. D. in computer sience at Universidad Politecnica de Madrid (Spain) in 1998. She has joined the Computer Science Department at the University Carlos III of Madrid in 1995 and she is associate professor of that department from 2000. Her current research focuses on artificial neural networks and other soft computing techniques, such as evolutionary computation and multiagent systems. Her research interests cover also applications fields, such as time series prediction and control of dynamic process.

**Pedro Isasi** received his Ph. D. in computer science from Politecnica de Madrid University in 1994. He joined the Computer Science Department of Carlos III de Madrid University in 1991, he was associate professor from 1997 until 2001, and he is now full professor of that department. He is the founder and director of the Neural Network and Evolutionary Computation Laboratory. His current research focuses on the application of soft computing techniques (NN, evolutionary computation, fuzzy logic and multiagent systems) to engineering problems such as power plant control, robot control, cryptography or finances, mainly in domains of prediction, classification, optimization and times series.