Neural Processing Letters **20:** 105–124, 2004.
© 2004 Kluwer Academic Publishers. Printed in the Netherlands.

Lazy Learning in Radial Basis Neural Networks: A Way of Achieving More Accurate Models

JOSÉ M.VALLS, INÉS M. GALVÁN, and PEDRO ISASI
Departamento de Informática, Universidad Carlos III de Madrid Avenida de la Universidad, 30, 28911 Leganés, Spain. e-mail: jvalls@inf.uc3m.es

Abstract. Radial Basis Neural Networks have been successfully used in a large number of applications having in its rapid convergence time one of its most important advantages. However, the level of generalization is usually poor and very dependent on the quality of the training data because some of the training patterns can be redundant or irrelevant. In this paper, we present a learning method that automatically selects the training patterns more appropriate to the new sample to be approximated. 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: an artificial regression 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.

Key words. improving generalization ability, kernel functions, K-means algorithm, lazy learning, radial basis neural networks

1. Introduction

Radial basis neural networks (RBNN) [1–3] are originated from the use of radial basis functions, as the Gaussian functions, in the solution of the real multivariate interpolation problem [4, 5].

RBNNs can be used for a wide range of application primarily because they can approximate any regular function [6].

Generally, the generalization capability of the RBNN is poor because they are too specialized in the data training. Some authors have paid attention to the nature and size of the training set in order to improve the generalization ability of the networks. There is no guarantee that the generalization performance is improved by increasing the training set size [7]. It has been shown that with careful dynamic selection of training patterns, better generalization performance may be obtained [8]. The idea of selecting the patterns to train the network from the available data about the domain is close of 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 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 be used for training. In this work a selective training strategy has been developed to improve the generalization capabilities of RBNN inspired on lazy strategies [9, 10]. The learning method proposed involve finding relevant data to answer a particular novel pattern and defer the decision of how to generalize beyond the training data until each new sample is encountered. Thus, the decision about how to generalize is carried out when a test pattern needs to be answered constructing local approximations. The main idea is to recognize, from the whole training data set, the most similar patterns for each new pattern to be processed.

2. Lazy Learning Method to Train Radial Basis Neural Networks

The learning method proposed in this work to train RBNN consists of selecting, from the whole training data, an appropriate subset of training patterns in order to improve the answer of the network for a 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 the pattern selection is to include—once or more times—those patterns close—in terms of the Euclidean distance and some weighting 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.

The general idea presented in this work for the selection of patterns consists of establishing an n-dimensional sphere centered at the test pattern, in order to select only those patterns placed into this sphere. Its radius-named r-is a threshold distance, since all the training patterns whose distance to the novel sample is bigger than r will be discarded. Distances may have very different magnitudes depending on the problem domains, due to their different data values and number of attributes. It may happen that for some domains the maximum distance between patterns is many times the maximum distance between patterns for other domains. In order to make the method independent of this fact, both the sphere radius and the training patterns distances will be relative respect to the maximum distance to the test pattern. Thus, the relative threshold distance or relative radius, r_{r}, will be used to select the training patterns situated into the sphere centered at the test pattern, being r_{r} a parameter that must be established before the application of the learning algorithm.

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

$$X = \{(x_k, y_k)\ k = 1 \ldots N, \quad x_k = (x_{k1}, \ldots, x_{kn}), \quad y_k = (y_{ki}, \ldots, y_{km})\}, \tag{1}$$

where x_k are the input patterns and y_k their respective target outputs. When a new sample q must be predicted, the RBNN 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:

Step 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

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

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

Step 2. As it was previously mentioned, in order to make the method independent on the distances magnitude, relative distances must be used. Thus, a relative distance, d_{rk} is calculated for each training pattern. Let $d_{\max}$ be the maximum distance to the novel pattern, this is

$$d_{\max} = \max(d_1, d_2, \ldots, d_N).$$

Then, the relative distance is given by

$$d_{rk} = \frac{d_k}{d_{\max}}. \tag{3}$$

Step 3. As it was mentioned in Section 1, a weighting function or kernel function $K()$ is used to calculate a weight for each training pattern from its distance to the test pattern. The maximum value of the kernel function must be given at zero distance and the function should decrease smoothly as distance increases. The kernel function used in the method is the inverse function $K(d) = 1/d$. An example of this function can be seen in Figure 1, where the x-axis represents the patterns in a one-dimensional domain, being the value of the test pattern 2.25. Thus, the result of evaluating the inverse function of the relative distance calculated at Equation (3) is associated to each training pattern:

$$K(x_k) = \frac{1}{d_{rk}}, \; k = 1, \ldots, N. \tag{4}$$

These values $K(x_k)$ are normalized in such a way that the sum of them equals the number of training patterns in X. The normalized values, named as $K_n(x_k)$, are obtained by

$$K_n(x_k) = VK(x_k),$$

where

$$V = \frac{N}{\sum_{k=1}^{N} K(x_k)},$$

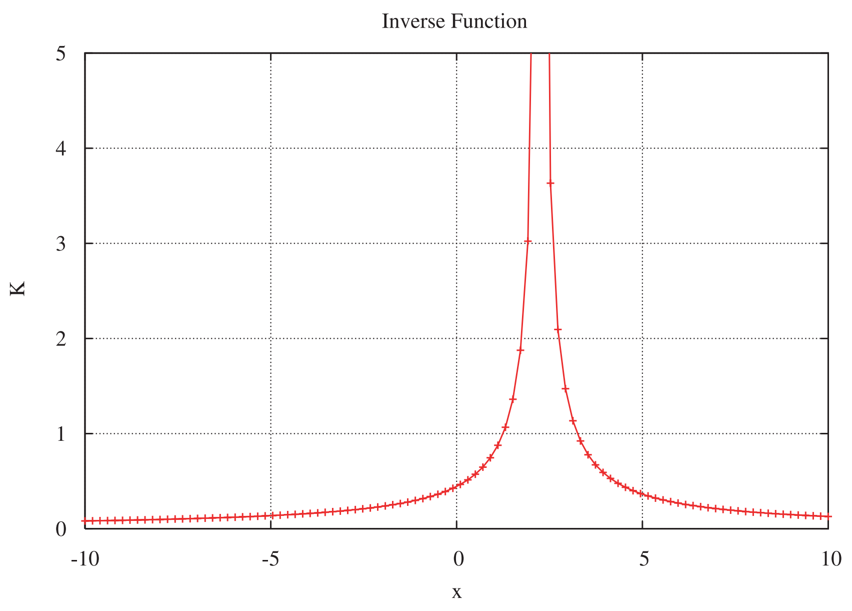

Figure 1. The inverse function as kernel function.

thus

$$\sum_{k=1}^{N} K_n(x_k) = N.$$

In order to simplify the notation, henceforth $K_n(x_k)$ will be named f_{nk}, normalized frequency.

Step 4. Both the relative distance $d_{\mathrm{r}k}$ calculated in step 2 and the relative frequency f_{nk} calculated in step 3 are used to decide whether the training pattern (x_k, y_k) is selected, and—if the pattern is selected—how many times is included in the training subset. Hence, they are used to generate a natural number, n_k, following the next rule:

$$\begin{array}{lll} \text{if} & d_{\mathrm{r}k} < r_{\mathrm{r}} & \text{then} \\ & n_k = int(f_{nk}) + 1 & \\ \text{else} & & \\ & n_k = 0. & \end{array} \tag{5}$$

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) will be used to train the RBNN when the new instance q is reached. If the pattern is selected, $n_k > 0$ otherwise $n_k = 0$.

Step 5. 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. Once the training patterns are selected, the RBNN is trained with the new subset of patterns, X_q. As usually, training a RBNN involves to determine the centers, the dilations or widths, and the weights. The centers are calculated in an unsupervised way using the K-means algorithm in order to classify the input space formed by all the training patterns included in the subset X_q. The k-means algorithm initialization has been modified in order to avoid the situation where many classes have no patterns at all. Thus, the initial values of the centers are set in the following way:

- M_q, the centroid of the set X_q, is evaluated.
- k centers $(c_{1q}, c_{2q}, \ldots, c_{kq})$ are randomly generated, such as $\|c_{jq} - M_q\| < \epsilon$, $j = 1, 2, \ldots, k$, where ϵ is a very small real number.

In this way, the k centers expand from their initial positions around the centroid M_q, and if the number of centers is smaller than the number of patterns, no classes will remain empty when k-means finishes. Once the neurones centers have been calculated, the neurones dilations or widths are evaluated as the geometric mean of the distances from each neuron center to its two nearest centers. Lets d_i the width and C_i the center of the i_{th} neurone:

$$d_i = \sqrt{\|C_i - C_t\| \; \|\mathrm{C_i} - \mathrm{C_s}\|}. \tag{6}$$

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

Finally, the weights of the RBNN are estimated in a supervised way to minimize the mean square error E measured in the training subset X_q:

$$E = \frac{1}{R}\sum_{r=1}^{R} e_r, \tag{7}$$

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

$$e(r) = \frac{1}{2}\sum_{i=1}^{m} (\tilde{y}_{ri} - y_{ri})^2, \tag{8}$$

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

3. Experimental Results

The lazy learning method presented in this work has been applied to three different problems: One artificial regression problem,—the Hermite Polynomial-whose dimension is 1, and two n-dimensional time series prediction problems, an artificial one—the Mackey–Glass time series- and a real one: a time-series describing the behavior of the water level at Venice Lagoon. In this set of experiments, the proposed method has been applied using RBNN with different architectures—i.e., different number of hidden neurones—and setting the relative radius to different values in order to study the influence of these parameters in the performance of the method. This performance has been measured in terms of the RBNNs mean errors over the whole test set. The mean error, e, for the test set is evaluated as

$$e = \frac{1}{n}\sum_{k=1}^{n} e_k, \tag{9}$$

where n is the number of patterns in the test set and e_k represents the error for the kth test pattern, calculated as $e_k = |\, \tilde{y}_k - y_k \,|$, 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.

In order to show whether the selective method proposed in this work is able to improve the generalization capability of RBNN, another set of experiments have been carried out training the RBNN 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. The results obtained by both methods, the proposed selective method and the traditional one, are compared. In Section 3.1. the experimental set-up description and results are presented.

3.1. AN ARTIFICIAL APPROXIMATION PROBLEM: THE HERMITE POLYNOMIAL

The Hermite polynomial (see Figure 2) is given by the following equation:

$$f(x) = 1.1(1 - x + 2x^2)e^{-(1/2)x^2}. \tag{10}$$

This domain has been widely used in RBNN literature 11, 12, 13.

A random sampling with an uniform distribution over the interval $[-4, 4]$ is used in order to obtain 40 input–output points for the training data. The test set is composed by 200 input–output points that are generated in the same way as the points in the training set. Both sets have been normalized in the interval $[0, 1]$.

The lazy learning method described in Section 2 has been applied to this problem for RBNN with different architectures, from 3 to 21 neurones, and they have been trained during 300 learning cycles varying the relative radius from 0.04 to 0.28. As it was previously commented, the aim of these experiments consists of studying the influence of the relative radius on the generalization ability of the networks. In Table 1 mean errors over the whole test set for each architecture and for each relative radius are shown. Figure 3 shows graphically these results.

It is possible to observe that when the relative radius is lower than a certain value, mean errors for all architectures are very high–higher as the number of neurons grow up-. This is due to the small number of patterns that are selected with such a small radius, insufficient to allow an appropriate training of the network. This training process with an insufficient number of patterns will be worse as the number of neurones increases. As the relative radius grows up, the behavior of the networks

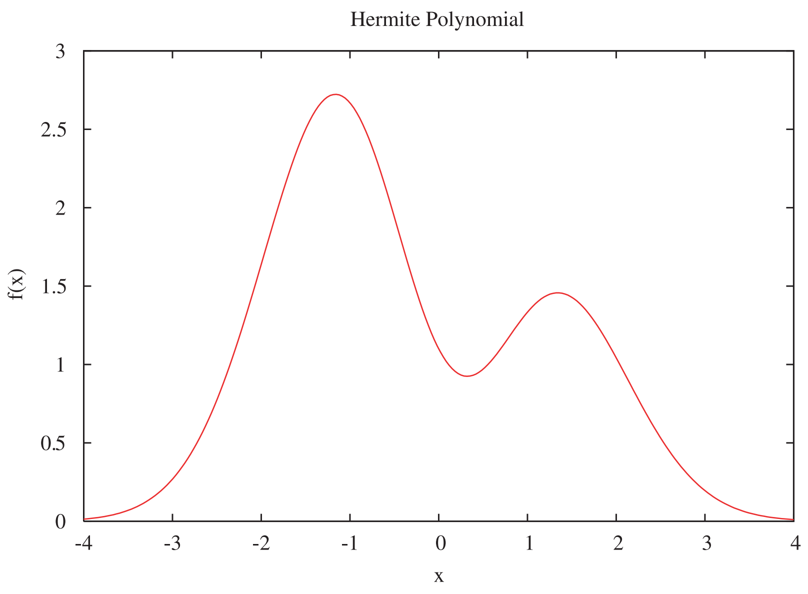

Figure 2. Hermite polynomial.

depends on their architecture: if the number of neurones is small -3 or 5 neurones—the error reaches a minimum when $r_r = 0.08$ and increases again as r_r rises. If the number of neurones is higher -9 or more- the mean error decreases and when r_r is 0.01 or bigger the error does not change significatively as the radius increases. This behavior is explained as follows: as the radius grows up, more patterns are selected allowing the network to perform better with the test set. If the number of neurones is very small, the network can not generalize properly when the number of training patterns is high, and that is why the mean error increases when the radius is greater than 0.08. If the number of neurones is bigger, the network can fit the training set, even if the number of patterns is high, keeping the mean error its value relatively

Table 1. Mean errors with the selective learning method. Hermite polynomial.

	Hidden neurones					
Relative radius	3	5	9	13	17	21
0.04	0.03321	0.02755	0.04067	0.07421	0.11557	0.68453
0.08	0.01037	0.00701	0.00345	0.00999	0.01792	0.02219
0.12	0.01498	0.01112	0.00439	0.00395	0.01093	0.00901
0.16	0.02176	0.01645	0.00635	0.00413	0.01012	0.00561
0.2	0.02902	0.01854	0.00703	0.00502	0.00421	0.00526
0.24	0.03425	0.02204	0.01065	0.00687	0.00488	0.00464
0.28	0.04230	0.02777	0.01450	0.00847	0.00493	0.00547

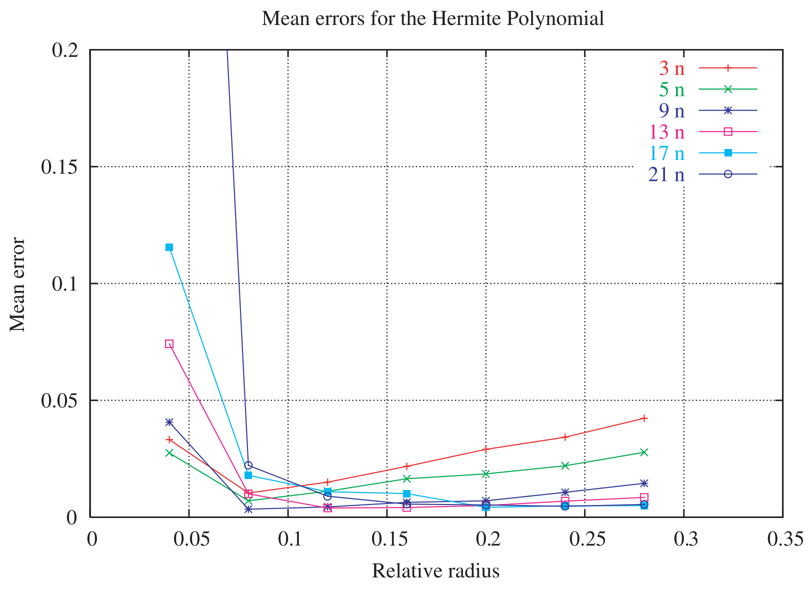

Figure 3. Mean errors with the selective learning method. Hermite polynomial.

constant. As it can be seen in Table 1 a network with 9 neurons obtains the best results using a relative radius of 0.08.

In order to study whether the lazy method can improve the generalization capability of RBNN, networks with different number of neurons have also been trained as usual, that is, using the whole training data, until the convergence of the network has been reached. In this work, the traditional learning has been carried out using a training and a validation data set. The training and validation errors have been measured every learning cycle and the iterative process has been stopped when both errors become stabilized. In Table 2, mean errors obtained for different architectures are shown. When the number of neurones is higher than 30 the mean error maintains its value near 0.02.

The best results for each method are shown in Table 3. A significative improvement of the generalization capability of RBNNs is obtained when the proposed lazy learning method is used.

In Figure 4 errors for each test pattern are displayed for both learning methods. These results correspond to the situations indicated in Table 3 where only the mean values of the errors are shown.

It is possible to observe that, for the majority of patterns, the error is smaller when the lazy learning method is used. Most of the test samples of the Hermite polynomial can be more accurately approximated when the RBNN is trained with an appropriate selection of patterns—the most relevant examples—instead of the whole training set. The computational cost is higher when the deferred training method is used, although, on the other hand, the number of neurons is smaller, *Table 2*. Mean errors with traditional learning method. Hermite polynomial.

Neurons	Mean error
10	0.11569
20	0.02702
30	0.02134
40	0.01904
50	0.02272
60	0.02215
70	0.02066
80	0.02263
90	0.02628
100	0.02145
110	0.02143
120	0.02338
130	0.02508

Table 3. Performance of different training methods for the Hermite polynomial.

Selective method	Traditional method
0.00345	0.01904
$r_r = 0.08$, 9 neurons	40 neurons

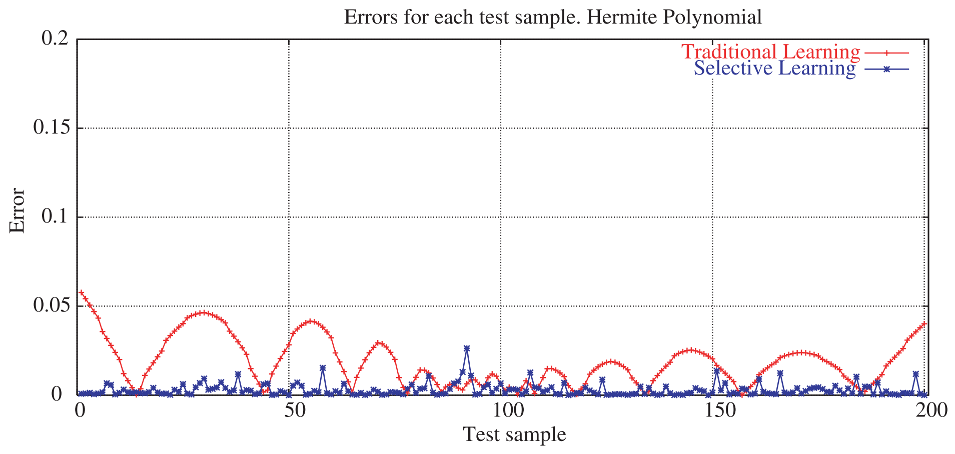

Figure 4. Errors for each test sample for the Hermite polynomial.

and the RBNN is trained in a shorter time. In both cases, the RBNN has been trained until it reaches the convergence. Thus, the generalization capability of the network using the whole training data can not be improved if it is trained for more learning cycles.

3.2. AN ARTIFICIAL TIME SERIES PREDICTION PROBLEM: THE MACKEY–GLASS SERIES

The Mackey-Glass series, based on the Mackey–Glass differential equation [7] is widely regarded as a benchmark for comparing the generalization ability of RBNN: 1, 13, 15, 12. This series is a chaotic time series governed by the following time-delay ordinary differential equation:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^{10}}. \tag{11}$$

Following the studies cited above, the series has been generated using the next values for the parameters: $a = 0.2, b = 0.1$, and $\tau = 17$. As in the mentioned studies, 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 [13]. Thus, the function to be learned -whose dimension is 4- by the network is:

$$x(t) = f(x(t-50), x(t-50-6), x(t-50-12), x(t-50-18)). \tag{12}$$

The initial 3500 samples are discarded in order to avoid the initialization transients. 1000 data points form the training set, corresponding to the sample time between 3500 and 4499. The test set is composed by the points corresponding to the time interval $[4500, 5000]$. In Figure 5 is shown the representation of the time series corresponding to the test set. All data points are normalized in the interval $[0, 1]$.

In this subsection the selective proposed lazy learning method has been applied to this artificial time series, where—in the same way as in the previous subsection—RBNN of different architectures have been trained during 300 learning cycles varying the relative radius from 0.05 to 0.3. The results -mean errors over the whole test set for each architecture and for each relative radius- are shown in Table 4. As in the previous case, the goal of this experiments is to study the influence of the relative radius in the performance of the networks.

Figure 6 displays these results and, as in the previous example, it is possible to observe that the performance of the network is influenced by the value of the relative radius and the architecture of the network. The tendency is similar to the one observed with the Hermite Polynomial: when the number of neurons is bigger than a certain value -5 neurons in this case-, the mean error decreases with the radius until $r_r = 0.1$, and then it maintains its value nearly constant. On the other hand, the network with 5 neurons makes smaller errors than architectures with more neurons, reaching a minimum error value when $r_r = 0.1$ and growing up again as the radius

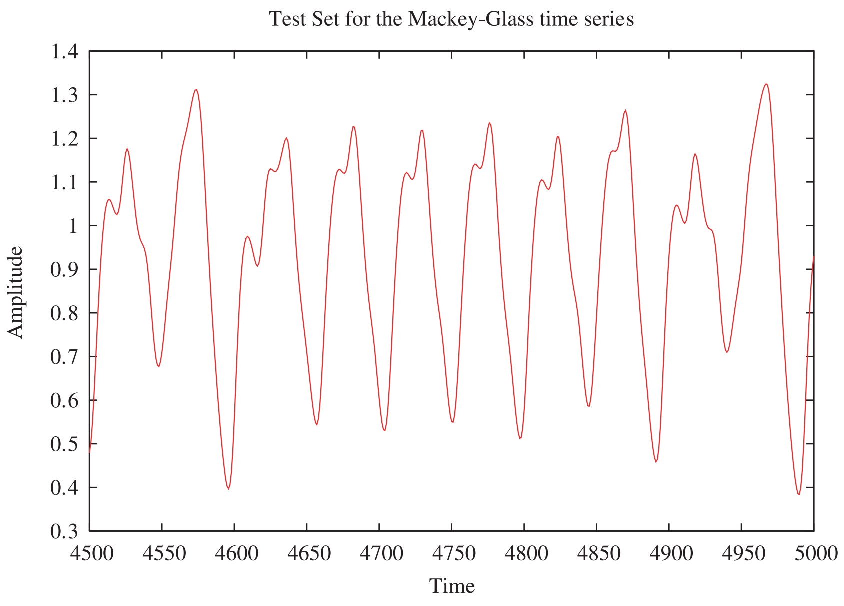

Figure 5. Mackey–Glass time series. Test set.

Table 4. Mean errors with the selective learning method for the Mackey–Glass time series.

	Hidden neurones					
Relative radius	5	10	15	20	25	30
0.05	0.0635	0.0814	0.11313	0.12392	0.16735	0.16419
0.1	0.0374	0.0265	0.02171	0.02104	0.02005	0.01877
0.15	0.0456	0.0233	0.01721	0.01863	0.01651	0.01711
0.2	0.0581	0.0204	0.01996	0.01691	0.01722	0.01713
0.25	0.0665	0.0226	0.02076	0.01824	0.01873	0.01739
0.3	0.0672	0.0248	0.02228	0.01953	0.01858	0.01837

increases. As in the previous case, the big errors committed when the radius is very small are due to the shortage of selected training patterns. In this case, a network with 25 neurons obtains the best results using a relative radius of 0.15.

As it was done in the Hermite polynomial case and having the aim of comparing both learning strategies, RBNNs with different number of neurons have been trained, using the whole training data until the convergence of the network has been reached. As in was done in part 1, a training and a validation data set has been used. In Table 5, mean errors obtained for different architectures are shown. The best results have been achieved using a RBNN with 110 neurons.

In Table 6 the best results for each learning method are displayed, being possible to observe that the performance of RBNN can be significatively enhanced when the

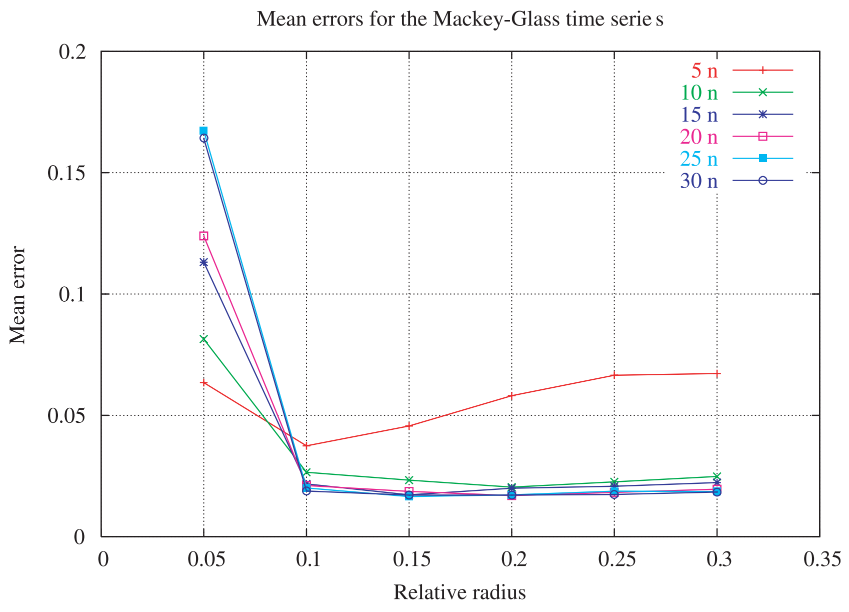

Figure 6. Mean errors with the selective learning method for the Mackey–Glass time series.

Table 5. Mean errors with the traditional learning method. Mackey–Glass time series.

Neurons	Mean error
10	0.1330
20	0.1356
30	0.1271
40	0.1277
50	0.1123
60	0.1052
70	0.1274
80	0.1115
90	0.1177
100	0.1163
110	0.1027
120	0.1114
130	0.1277

lazy learning method is used. Moreover, nearly all the results achieved with the selective method are better than the best result produced by the traditional one.

As in the previous case, although the mean error comparison shows that the lazy learning strategy behaves better than the usual one, it is interesting to verify that this

Table 6. Perfomance of different training methods for the Mackey–Glass time series.

Selective method	Traditional method
0.01651	0.1027
$r_r = 0.15$, 25 neurons	110 neurons

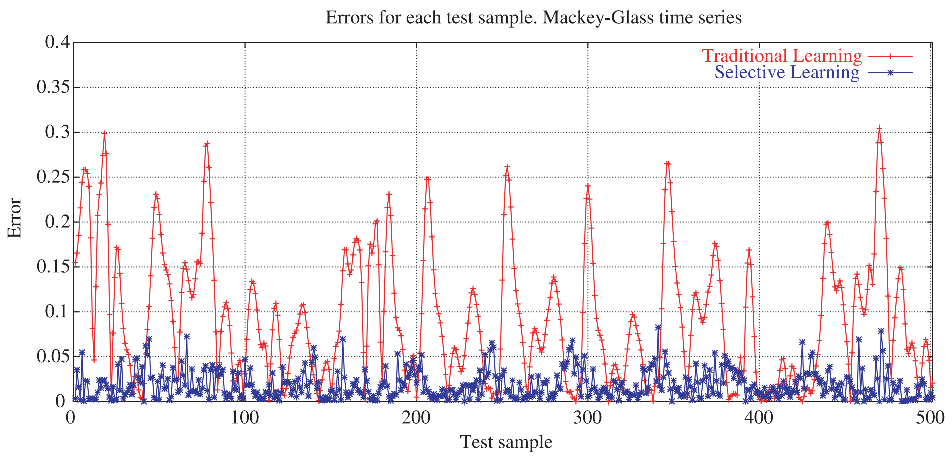

Figure 7. Errors for each test sample for the Mackey–Glass time series.

better behavior occurs for the majority of the test patterns. This can be seen in Figure 7 where the absolute errors for each test sample are displayed. Most of them are predicted with more accuracy when the selection of training patterns is made. As it happened in the Hermite polynomial case, the computational cost is higher when the selective strategy is utilized, but, on the other hand, less neurones are needed and the training of the network is faster.

3.3. A REAL TIME SERIES PREDICTION PROBLEM: PREDICTION OF WATER LEVEL AT VENICE LAGOON

Unusually high tides 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 17, 18. 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

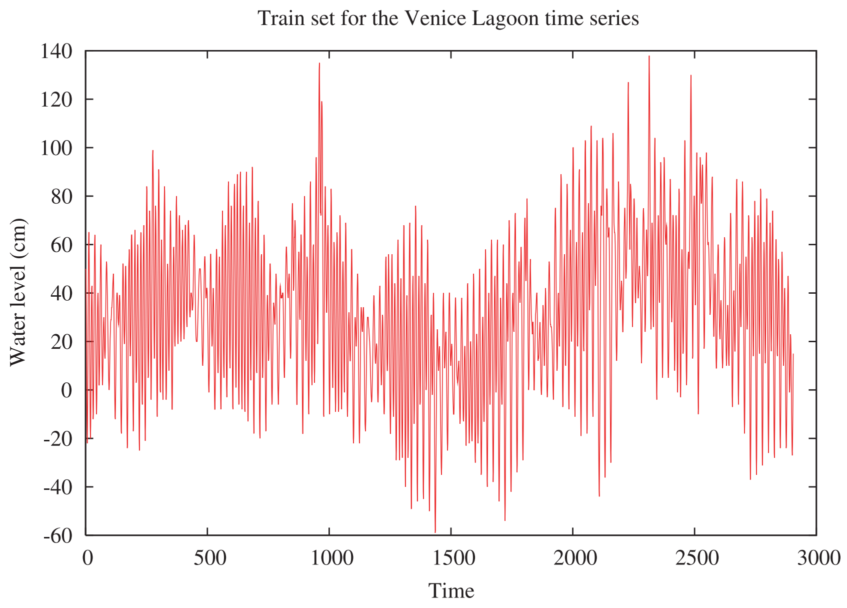

Figure 8. Water level at Venice Lagoon during four months. Training set.

predicting sea levels in Venice and mainly for the prediction of the high water phenomenon [19].

Different approaches have been developed for the purpose of predicting the behavior of sea level at the Lagoon Venice [18, 20]. Multilayer feedforward neural networks have also been used to predict the water level [21] obtaining same advantages over linear and traditional models.

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 RBNN 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 Figure 8). High-water situations are considered when the level of water is no lower than 110 cm. 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 Figure 9). 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

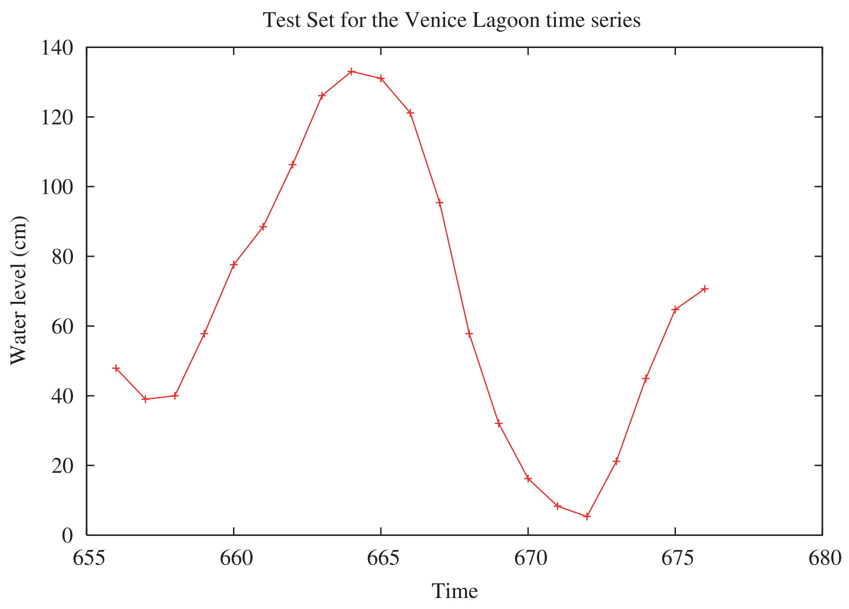

Figure 9. Water level at Venice Lagoon. Test set.

interesting to predict the high water phenomenon but also what will happen around that phenomenon.

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. The aim in this context is to observe whether a lazy strategy may help to obtain better predictions of high water phenomena. The selective learning method described in Section 2 has also been used to train RBNNs with different architectures and different relative radius (r_r) during 300 learning cycles, and their generalization capability has been measured. Mean errors on the test set achieved by these networks are shown in Table 7.

As in previous examples, the performance of the network is influenced by the value of the relative radius and the architecture of the network (see Figure 10). It is possible to observe that, as in previous cases, when the relative radius is small, mean errors are very high, due to the shortage of selected training patterns, and as the relative radius increases, the mean error decreases and then does not change significatively. In this case, a network with 15 neurons obtains the best results using a relative radius of 0.12.

RBNNs of different architectures have been trained with the usual method, that is, using the whole training data until the convergence of the network has been reached, in the same way as in previous cases. In Table 8, the corresponding mean errors are displayed, and the best results have been obtained by an architecture with 50 neurons, although no significant differences have been found for networks between 20 and 130 neurons. It is observed that the test mean error can not be

Table 7. Mean errors with the selective learning method for the Venice Lagoon time series.

	Hidden neurones						
Relative radius	3	5	7	9	11	13	15
0.04	0.21442	0.26172	0.34971	0.40387	0.40387	0.48629	0.52912
0.06	0.08597	0.11279	0.10640	0.13074	0.13074	0.13165	0.14915
0.08	0.05360	0.04531	0.05006	0.05773	0.05773	0.04944	0.05745
0.1	0.03171	0.03438	0.03039	0.03148	0.03148	0.02866	0.02547
0.12	0.03495	0.02797	0.02744	0.02578	0.02578	0.02434	0.01951
0.14	0.03303	0.02842	0.02745	0.02906	0.02906	0.02648	0.02562
0.16	0.03741	0.02867	0.02881	0.02371	0.02371	0.03063	0.03075
0.18	0.04060	0.03059	0.02916	0.02354	0.02354	0.02818	0.02784
0.2	0.04882	0.03292	0.03075	0.02603	0.02603	0.02759	0.02546

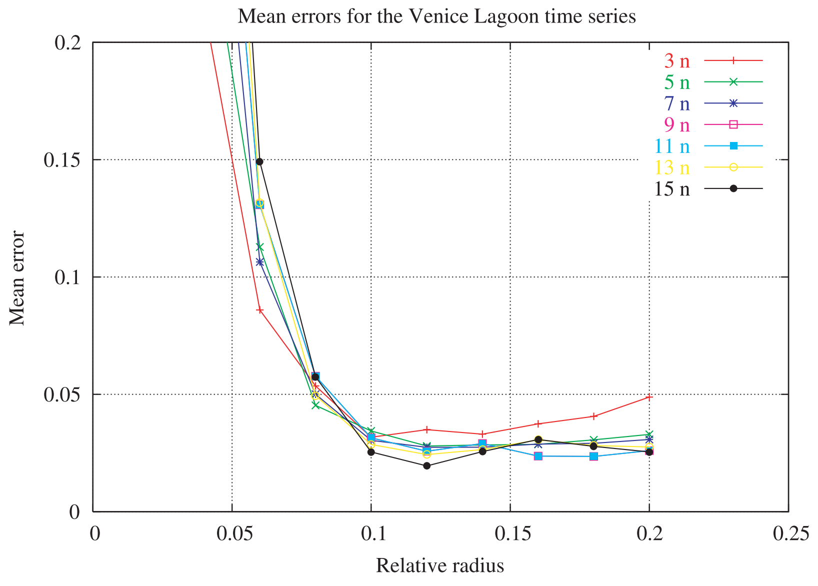

Figure 10. Mean errors with the selective learning method for the Venice Lagoon time series.

improved even if more learning cycles are performed using the whole training data set.

As in the previous example, in order to compare the proposed lazy learning method with the traditional one, mean errors over the test set obtained by both methods are shown in Table 9 for the best architectures. As it can be observed, the mean error over the test set is reduced when the network is trained with an appropriate selection of patterns.

Table 8. Mean errors for different architectures with the traditional learning method. Venice Lagoon time series.

Neurons	Mean error
10	0.2365
20	0.1341
30	0.1117
40	0.1120
50	0.0961
60	0.1029
70	0.1022
80	0.1065
90	0.1214
100	0.1254
110	0.1290
120	0.1380
130	0.1415

Table 9. Perfomance of different training methods for the Venice lagoon time series.

Selective method	Traditional method
0.01951	0.0961
$r_r = 0.12$, 25 neurons	50 neurons

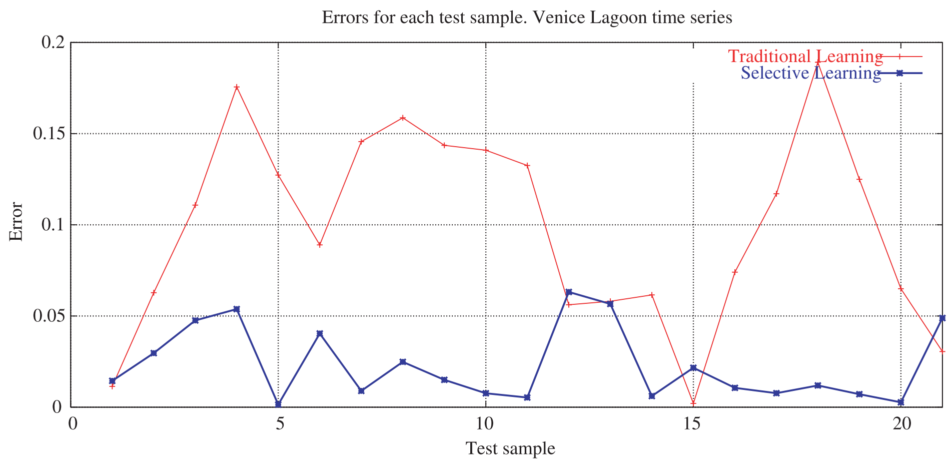

Figure 11. Errors for each test sample for the Venice Lagoon time series.

As it is shown in Figure 11, where the errors committed by the different learning strategies for each test pattern are shown, most of the test patterns are better approximated when the selective strategy is used. The error, when the network is trained in the traditional way is significantly higher, for the majority of patterns, than the corresponding to the selective learning method, when an appropriate selection of patterns is made. As it was previously commented, the test set contains a ‘‘high water’’ situation, and very few training patterns represent this kind of situation. When the network is trained as usual, all the training patterns are used and most of them represent the ‘‘periodic’’ situation. When a selection of patterns is made, only the most similar—training patterns representing the ‘‘high water’’ situation—are used to train the RBNN, achieving a more accurate prediction.

4. Conclusions

The generalization capabilities of RBNNs depends not only on the learning methods but also on the quality of the data used to train the network. The use of the whole training data available about the domain might not be the best choice, specially when data from some pattern space regions behave differently from the rest. The generalization performance in those special regions that do not follow the general tendency is distorted by the characteristics of the rest of regions.

The lazy learning method presented in this work provides an automatic mechanism to select the most appropriate training data in terms of the novel sample. Thus, all regions in the pattern space, even those that do not follow the general tendency, are properly considered. The results presented in the previous sections show that if RBNNs 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 form the neighborhood region around the novel sample—and the replication of those patterns helps RBNNs to obtain better results on approximation functions and time series prediction. The relevance of training patterns depends on its similarity to the novel pattern, measuring this similarity in terms of the Euclidean distance.

The extension of the neighborhood region around the novel sample is determined by a parameter named relative radius. The presented results show that if the relative radius reaches a minimum value and the network has a sufficient number of neurons, the generalization error keeps its low value relatively constant.

The K-means algorithm has an important part on the RBNN performance and previous experiments have shown that with the selective learning method a problem related to K-means algorithm arises. If the K-means algorithm is used as usual a lot of clusters remain empty and many hidden neurones in the RBNN are useless prejudicing the network behavior. This is explained because the input space corresponding to the selected training patterns is usually very small. Due to this reduced space, the election of the initial centers for the K-means algorithm is extremely important. The proposed method to determine the initial centroid of the cluster avoids this problem.

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 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.

Another interesting property of the proposed method is that it could be used in any supervised neural network model. Although it has been associated to RBNN, it actually can be applied to different neural network models since the procedure to select the most relevant training data does not depend on the type of neural network employed. Thus, once the training subset is selected, it can be used to train different types of neural networks; this feature of the proposed approach is an additional advantage since it increases its usefulness and generality.

References

1. Moody, J. E. and Darkn, C.: Fast learning in networks of locally tuned processing units. *Neural Computation*, **1** (1989), 281–294.
2. Poggio, T. and Girosi, F.: Networks for approximation and learning. *Proc. IEEE*, **78** (1990), 1481–1497.
3. Ghosh, J. and Nag, A.: *An Overview of Radial Basis Function Networks*. R. J. Howlett and L.C. Jain (Eds). Physica Verlag, (2000).
4. Broomhead, D. S. and Lowe, D.: Multivariable functional interpolation and adaptative networks. *Complex Systems*, **2** (1988), 321–355.
5. Powell, M.: The theory of radial basis function approximation in 1990. *Advances in Numerical Analysis*, **3** (1992), 105–210.
6. Park, I. J. and Sandberg, W.: Universal approximation and radial-basis-function networks. *Neural Computation*, **5** (1993), 305–316.
7. Abu-Mostafa, Y. S.: The vapnik-chervonenkis dimension: information versus complexity in learning. *Neural Computation*, **1** (1989), 312–317.
8. Cohn, D. Atlas, L. and Ladner.R.: Improving generalization with active learning. machine learning. *Machine Learning*, **15** (1994) 201–221.

9. Atkenson, C.G. Moore, A.W. and Schaal, S.: Locally weighted learning. *Artificial Intelligence Review*, **11** (1997) 11–73.
10. Wettschereck, D. and Dietterich, T.: Improving the perfomance of radial basis function networks by learning center locations. *Advances in Neural Information Processing Systems*, **4** (1992), 1133–1140.
11. Leonardis, A. and Bischof, H.: An efficient mdl-based construction of rbf networks. *Neural Networks*, **11** (1998), 963–973, .
12. Orr. M. J. L.: Introduction to radial basis neural networks. *Technical Report. Centre for Cognitive Science, University of Edinburgh*, (1996).
13. Yingwei, L. Sundararajan, N. and Saratchandran, P.: A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, **9** (1997), 461–478.
14. Mackey, M. C. and Glass, L. Oscillation and chaos in physiological control systems. *Science*, **197** (1977) 287–289.
15. Platt, J.: A resource-allocating network for function interpolation. *Neural Computation*, **3** (1991), 213–225.
16. Whitehead, B. A. and Choate, T. D.: Cooperative - competitive genetic evolution of radial basis function centeres and widths for time series prediction. *IEEE Transactions on Neural Networks*, **5** (1995), 15–23.
17. Moretti, E. and Tomasin, A.: Un contributo matematico all-elaborazione previsionale dei dati di marea a Venecia. *Boll. Ocean. Teor. Appl.*, **1** (1984), 45–61.
18. Michelato, A. Mosetti, R. and Viezzoli. D.: Statistical forecasting of strong surges and aplication to the lagoon of Venice. *Boll. Ocean. Teor. Appl.*, **1** (1983), 67–83.
19. 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.*, **5** (1973), 51.
20. Vittori, G.: On the chaotic features of tide elevation in the lagoon Venice. *Proc. of the ICCE-92, 23rd International Conference on Coastal Engineering*, pages (1992), 4–9.
21. Zaldívar, J. M. Gutrrez, E. Galván, I. M. Strozzi, F. and Tomasin, A.: Forecasting high waters at Venice Lagoon using chaotic time series analysis and nonlinear neural networks. *Journal of Hydroinformatics*, **2** (2000), 61–84.