Deferring the Learning for Better Generalization in Radial Basis Neural Networks

José María Valls, Pedro Isasi, and Inés María Galván

Carlos III University of Madrid, Computer Science Department. Avd Universidad, 30, 28911, Leganés, Madrid
jvalls@inf.uc3m.es

Abstract. The level of generalization of neural networks 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 most appropriate training patterns to the new sample to be predicted. The proposed method has been applied to Radial Basis Neural Networks, whose generalization capability is usually very poor. The learning strategy slows down the response of the network in the generalisation phase. However, this does not introduces a significance limitation in the application of the method because of the fast training of Radial Basis Neural Networks.

1 Introduction

The radial basis neural networks (RBNN) [1,2] are originated from the use of radial basis functions, as the Gaussian functions, in the solution of the real multivariate interpolation problem. RBNNs can be used for a wide range of application primarily because they can approximate any regular function [3]. Generally, the generalisation capability of the RBNN is poor because they are too specialised 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 [4]. It has been shown that with careful dynamic selection of training patterns, better generalisation performance may be obtained [5,6].

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 generalisation capabilities of RBNN inspired on lazy strategies [7,8]. The learning method proposed involve finding relevant data to answer a particular novel pattern and defer the decision of how to generalise beyond the training data until each new sample is encountered. Thus, the decision about how to generalise is carried out when a test pattern needs to be answered constructing local approximations. The main idea is to recognise from the whole training data set, the most similar patterns for each new pattern to be processed.

2 Automatic Selection of Training Data

The method proposed in this work to train RBNN consists of selecting, from the whole training data, an appropriate subset of patterns in order to improve the answer of the network for a novel test pattern. The general idea for the selection of patterns is to include only and several times those patterns close to the novel sample. To develop this idea, let us consider q an arbitrary test pattern described by a n-dimensional vector, $q=(q_1,..., q_n)$, where q_i represents the attributes of the instance q. Let $X = \{(x_i, y_i)/i = 1,..., N\}$ be the whole available training data set, where x_i are the input patterns and y_i their respective target outputs. The steps to select the training set associated to the pattern q, which is named 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.

Step 2. A measure of frequency, $f_k=1/d_k$, where k=1,...N ,is associated to each training pattern (x_k,y_k). In order to obtain a relative frequency, the values f_k are normalised in such way that the sum of the frequencies is equal to the number of training patterns in X. The relative frequencies, named as fn_k, are obtained by:

$$fn_k = \frac{f_k}{S} \text{ where } S = \frac{1}{N}\sum_{i=1}^{N} f_k \qquad \text{Thus: } \sum_{i=1}^{N} fn_k = N$$

Step 3. The values fn_k's 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 are transformed to natural numbers as: $n_k=\text{int}(fn_k)$.

At this point, each training pattern in X has associated a natural number, n_k, which indicates how many times the pattern (x_k, y_k) has been used to train the RBNN when the new instance **q** is reached.

Step 4. A new training pattern subset associated to the test pattern **q**, X_q, is built up. Once the training patterns are selected, the RBNN is trained with the new subset of patterns, X_q. 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 to classify the input space. After that, the dilations coefficients are calculated as the square root of the product of the distances from the respective center to its two nearest neighbours. Finally, the weights of the RBNN are estimated in a supervised way to minimise the mean square error measured in the training subset X_q.

Let's $W_q \subset X_q$ the set resulting of removing the repeated patterns. In this work, the K-means algorithm 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 as:

- M_q, the centroid of the set W_q, is evaluated.
- k centers are randomly generated $(c_{1q}, c_{2q}, \ldots c_{kq})$,

such as $\left| c_{jq} - M_q \right| < \varepsilon$, j=1,...k, and ε is a very small real number.

3 Experimental Results

The deferred learning method proposed in this work has been applied to two different approximation problems. In order to validate the proposed method, RBNNs have also been trained as usual, this is, the network is trained using the whole training data set.

3.1 Hermite Polynomial Approximation

The Hermite polynomial is given by the equation:

$F(x)=1.1(1-x+2x^2)\exp(-1/2x^2)$

A random sampling of the interval [-4 , 4] is used in obtaining 160 input-output points for the training set and 65 input-output data for the test set.

RBNNs with different number of neurones have been trained using the whole training data. The training process is stopped either when 150 cycles are performed or when the derivative of the train error equals zero. The generalization capability of the trained networks has been measured using the test set and the mean square errors achieved by the networks. In figure 1 (a), the mean errors obtained for different architectures are shown. The best results have been achieved using a RBNN with 25 neurones, although no significant differences have been found for networks between 10 and 25 neurones. It is observed that the test error can not be improved even if more learning cycles are performed using the whole training data set.

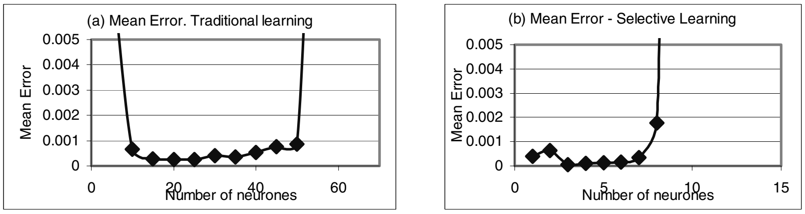

Fig. 1. Mean error on the test set for the Hermite function achieved by different architectures.

The selective learning method proposed in this work has also been used to train RBNNs with different architectures during 150 learning cycles, and their generalization capability has been tested. Mean square errors on the test set achieved by these networks are shown in figure 1 (b). In this case, only 3 neurones are necessary to obtain the best results, and using more than 8 increases the error to the level of the RBNN trained with the classical method.

The value of the mean square test error achieved by the best network, the one with 25 neurones, trained with the whole training set, is shown in Table 1. Using the specific learning method proposed in this work, the best network, the one with 3 neurones, has also been evaluated. In that case, the mean square error over the test set is 2.5 times lower.

Table 1. Performance of different training methods for the Hermite function

	Mean square error	Number of Neurones
Training with the whole data set	2.49×10^{-4}	25
Training with a selection of data	0.649×10^{-4}	3

To show the difference between both learning methods, the errors for each test pattern are represented in figure 2. In this figure it can be observed that the generalization error corresponds initially to some difficult regions of the function. It is specially in these regions where our approach is able to drastically reduce the error and to find a good approximation.

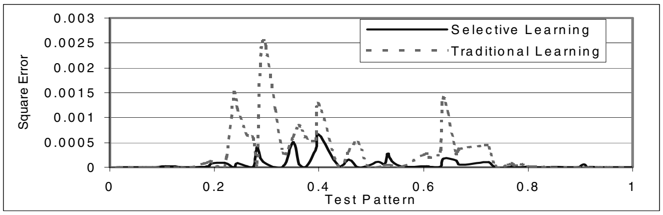

Fig. 2. Hermite function: Square errors for each test patterns.

3.2 Piecewise-Defined Function Approximation

This function has been chosen because of the poor generalisation performance that RBNN presents when approximating it. The function is given by the equation:

$$f(x) = \begin{cases} -2.186x - 12.864 & \text{if } -10 \le x < -2 \\ 4.246x & \text{if } -2 \le x < 0 \\ 10e^{(-0.05x - 0.5)} \sin\left[(0.03x + 0.7)x\right] & \text{if } 0 \le x \le 10 \end{cases}$$

The original training set is composed by 120 input-output points randomly generated by an uniform distribution in the interval [-10,10]. The test set is composed by 80 input-output points generated in the same way as the points in the training set.

As in the previous experiment, RBNNs with different number of neurones have been trained, using the whole training data until the convergence of the network has been reached, that is, either when 150 cycles are performed or when the derivative of the train error equals zero. In figure 3 (a), the mean square errors obtained for different architectures are shown. The best results have been achieved using a RBNN with 20 neurones, although no significant differences have been found for networks between 2 and 30 neurones.

The selective learning method proposed in this work has also been used to train RBNNs with different architectures during 150 learning cycles, and their generalization capability has been tested. Mean square errors on the test set achieved by these networks are shown in figure 3 (b). In this case, only 4 neurones are necessary to obtain the best results, and using more than 8 increases the error to the level of the RBNN trained with the clasical method.

The mean square errors obtained in both cases over the test set, and the learning cycles involved are shown in table 2. As it is possible to observe in table 2, the mean square error over the test set is significantly reduced when an appropriate selection of patterns is made.

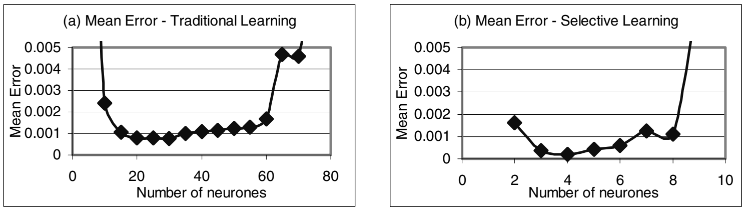

Fig. 3. Mean error on the test set for the Piecewise function achieved by different architectures.

Table 2. Performance of different training methods for the piecewise function.

	Mean square error	Number of Neurones
Training with the whole data set	7.8×10^{-4}	20
Training with a selection of data	2.07×10^{-4}	4

As in the previous experiments, the computational cost is higher when the deferred training method is used, although, on the other hand, the number of neurones is smaller, and the RBNN is trained in a shorter time. In that case, as in the previous approximation problem, the RBNNs has been trained until they reach the convergence. Thus, the generalisation capability of the network using the whole training data can not be improved if it is trained for more learning cycles.

It has been observed how the network trained with the whole training data has some difficulties to approximate the points in which the function changes their tendency. This can be described as a deficiency in the generalisation capabilities of the network. However, the generalisation is improved when an appropriate selection of patterns is made. The proposed method is able to provide better approximations of points even when the tendency of the function is changed. Figure 4 shows the errors committed by the different learning strategy for each test pattern. Most of the test patterns are better approximated when the specific learning method is used to train RBNN.

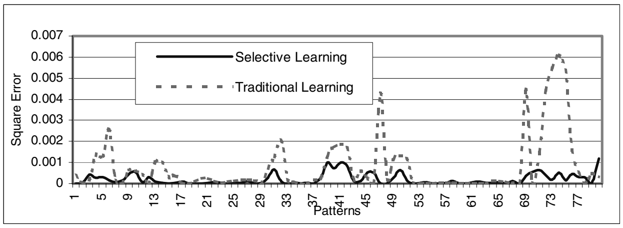

Fig. 4. Piecewise-defined function: Square errors for each set patterns.

4 Conclusions

The results presented in the previous section show that if RBNNs are trained with a selection of training patterns, the generalisation performance of the network is improved. The selection of the most relevant training patterns, helps to obtain RBNN‘s able to better approximate complex functions. The selective method seems to be more sensitive to the number of hidden neurones than the traditional one. However, when the selective method is used less experiments must be realized because it gets better generalization results with less hidden neurones.

Due to the reduced input space, the election of the initial centeres for the K-means algorithm is extremely important. Previous experiments have shown that 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 behaviour. The proposed method to determine the initial centroid of the cluster avoids this problem.

The specific learning methods proposed in this work involves storing the training data in memory, and finding relevant data to answer a particular test pattern. Thus, the decision about how to generalise 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 when a new sample test is presented. However, that is not a disadvantage of the method because in many cases that computational effort can be broached and to achieve lower approximation errors is an important advantage. Moreover, the number of neurones of the network trained with the selective method is much lower, thus, the computational effort is not as high as it appears to be.

The proposed method uses the Euclidean distance as similarity measure. However, the method is flexible to incorporate other similarity measures, which will be studied in the future.

References

1. Moody J.E. and Darken C.J.: 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. Proceedings of the IEEE, 78, (1990), 1481-1497.
3. Park J. and Sandberg I.W.: Approximation and Radial-Basis-Function Networks. Neural Computation, 5, (1993), 305-316.
4. Abu-Mostafa Y. S.: The Vapnik-Chervonenkis dimension: information versus complexity in learning. Neural Conputation 1, (1989), 312-317.
5. Cohn D., L. Atlas and R. Ladner: Improving Generalisation with Active Learning, Machine Learning, Vol 15, (1994), 201-221.
6. Vijayakumar S. And H. Ogawa: Improving Generalization Abolity through Active Learning. IEICE Transactions on Information and Sytems. Vol E82-D,2, (1999), 480-487.
7. Atkeson C. G., A. W. Moore and S. Schaal. Locally Weighted Learning. Artificial Intelligence Review 11, (1997), 11-73.
8. Wettschereck D., D.W. Aha and T. Mohri: A review and Empirical Evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review 11, (1997), 273-314.