ELSEVIER

Neurocomputing 10 (1996) 149–168

NEUROCOMPUTING

Time series forecasting by combining RBF networks, certainty factors, and the Box–Jenkins model

Donald K. Wedding II, Krzysztof J. Cios *

Dept. of Electrical Engineering, University of Toledo, Toledo, OH 43606, USA

Received 11 October 1993; accepted 12 January 1995

Abstract

A method is described for using Radial Basis Function (RBF) neural networks to generate certainty factors along with normal output. When RBF output with low certainty factors values are discarded, the overall accuracy of the network is increased. In this paper, RBF networks are used in a time series application. The RBF neural networks are trained to generate both time series forecasts and certainty factors. Their output is then combined with the Univariant Box-Jenkins (UBJ) models to predict future values of data. This combination approach is shown to improve the overall reliability of time series forecasting. Three possible methods for combining the two forecasts into one hybrid forecast are discussed.

Keywords: RBF neural networks; Box–Jenkins; Forecasting; Certainty factors

1. Introduction

Time series forecasting is the predicting of future values based upon present and past values. Accurate forecasting is important to both government and industry who need to predict such things as future revenues, costs, and demands.

One of the most widely used methods in time series forecasting is the model described by Box and Jenkins [2]. The Univariant Box–Jenkins Method (UBJ) combines moving average and autoregressive models into one unified approach. This approach is both simple and yields accurate results which explains its wide use. The disadvantage of the UBJ method is that it gives simplistic models that only use several previous values to forecast the future. The UBJ is, therefore, unable to find subtle patterns in the time series data.

* Corresponding author. Email: fac1765@uoft01.utoledo.edu

SSDI 0925-2312(95)00021-6

As computer power increases, other approaches to forecasting have been attempted to find subtle trends and relationships. One such method is the use of neural networks. Of the many types of neural networks, one of the most widely used is the Radial Basis Function (RBF) [4,15]. RBFs have been employed in time series prediction with success [6] since they can be trained to find complex relationships in the data. Although their use in forecasting is growing, there is a disadvantage to RBFs. They are very poor at extrapolating an answer when the input data is considerably different from what they were trained on.

As shown below, the RBF network model and the UBJ model compliment one another. The RBF can find subtle patterns in the data, but cannot give meaningful results when the input data deviates from the patterns. The UBJ method gives good results in general, but misses the subtle patterns which also can result in inaccurate forecasts. This paper will describe a method of combining the forecasts from the UBJ method and RBF networks by using certainty factor values derived from the output of the RBF's basis functions. When the certainty factor is high, then the RBF network is indicating that the data matches a previously learned pattern and its forecast is reliable. When the certainty is low, then no pattern was found and the forecast is discarded in favor of the forecast from the UBJ model. The resulting hybrid forecasting model yields more accurate results than using RBF or UBJ forecasting alone.

2. Overview

2.1. Box–Jenkins forecasting method

The Univariant Box–Jenkins (UBJ) method [2] forecasts future time series data with a combination of autoregressive and moving average data. The autoregressive (AR) part of the equation relates the future value to past and present values in a linear fashion. The moving average (MA) component relates the future value to the errors of previous forecasts. The general form of the UBJ forecast equation is given by:

$$\hat{Y}_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} - \theta_1 a_{t-1} - \cdots - \theta_q a_{t-q} \qquad (1)$$

In Eq. (1), the ϕ terms are coefficients determined by linear regression. These coefficients are multiplied to the p previous values in the time series. The θ terms are coefficients which are determined through non linear iterative methods and are multiplied to the previous q forecast errors. The Y values in Eq. (1) are not actually the previous values in the time series. Instead, they are the previous values minus the mean of the training set. The Y values are calculated using the equation:

$$Y_t = Z_t - \mu \qquad (2)$$

where Z_t is the actual value at time t and μ is the mean of all the Z values used in training the UBJ model. The error terms for Eq. (1) are calculated using the formula:

$$a_{t-j} = Y_{t-j} - \hat{Y}_{t-j} \qquad (3)$$

The errors are therefore the actual values at time t minus what they were forecasted to be.

The actual number of terms in the AR and the MA parts of the final model are not chosen arbitrarily. Instead, Box and Jenkins give a structured approach to determining which models will best fit the series in question. Box and Jenkins recommend that the time series models be kept as simple as possible. As a result there are generally not more than three AR or MA terms.

2.2. Radial Basis Function neural networks

The RBF neural network [4,15] is composed of three layers of nodes. The first is the input layer that feeds the input data to each of the nodes in the second or hidden layer. The second layer of nodes differs greatly from other neural networks in that each node represents a data cluster which is centered at a particular point and has a given radius. The third and final layer consists of only one node. It acts to sum the outputs of the second layer of nodes to yield the decision value.

When an input vector is fed into each node of the hidden layer simultaneously, each node then calculates the distance from the input vector to its own center. That distance value is transformed via some function, and the result is output from the node. That value output from the hidden layer node is multiplied by a constant or weighting value. That product is fed into the third layer node which sums all the products and any numeric constant inputs. Lastly, the third layer node outputs the decision value. A graphical representation of an RBF neural network with three data centers is shown in Fig. 1.

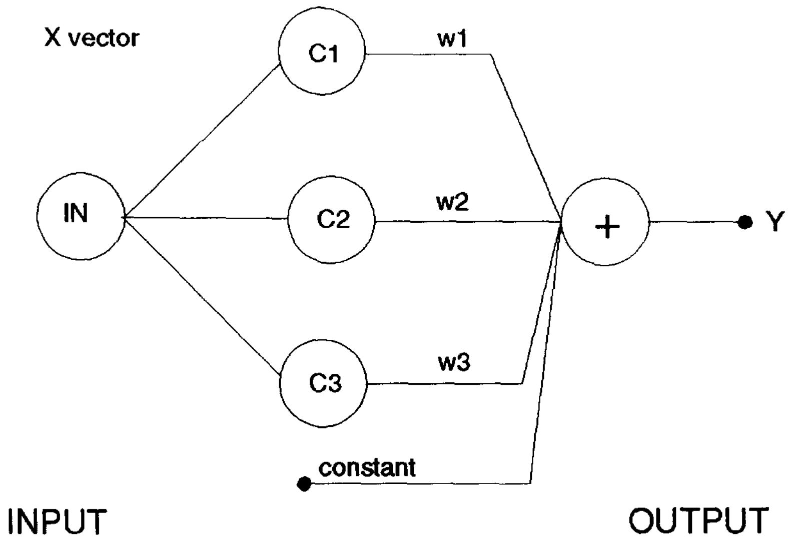


Fig. 1. Radial basis function neural network.

An RBF neural network computes a decision function from a given input. The network receives a k dimensional input vector **x** and outputs a scalar value using the general formula:

$$f_r(\bar{\mathbf{x}}) = \lambda_0 + \sum_{i=1}^{n} \lambda_i \phi(v_i) \tag{4}$$

In Eq. (4), the value n refers to the number of nodes in the hidden layer of the RBF network. The lambdas in the formula are weight values.

The vector **v** is an array of length n and contains the distance measure of the input vector **x** to each of the centers. Usually, the euclidean norm is used to calculate distance, but any other metric can also be used. The equation for the euclidean norm is given by:

$$\mathbf{v}_i = \sqrt{\sum_{j=1}^{k} (\mathbf{x}_j - \mathbf{c}_{ji})^2} \tag{5}$$

The value **c** in Eq. (5) represents a cluster center for any of the given nodes in the hidden layer. Lastly, the function ϕ(v) in Eq. (4) represents the function that the hidden layer nodes use to transform the distance value before multiplying by the lambda values. The function chosen for this application is the gaussian function:

$$\phi(v) = \exp^{-(v^2/\beta^2)} \tag{6}$$

Note that in Eq. (6) the value of β is the radius of the cluster represented by the node.

Complex nonlinear systems such as time series data are generally difficult to model using standard linear regression. Unlike regression, neural networks are nonlinear. Their parameters are determined or trained using techniques such as error backpropagation [16]. The major fault with standard neural nets comes in the area of determining their parameters. The various techniques can be time consuming and have a tendency to get stuck at local minimum points [7]. RBFs overcome the above problems since the only lambda values that need to be trained are the ones in the hidden layer of the network. Finding their values is the solution of a linear problem and can be obtained through interpolation [3]. Therefore, the lambda values are found much faster than in other neural networks. Also, the network can usually reach near perfect accuracy on the training data set without running into the problem of local minima.

The drawback to RBF neural networks occurs when an input vector is a large distance away from the training data. In the case of the gaussian functions, as the input vectors get farther away from the center of a cluster the resulting output of the gaussian function gets exponentially smaller and quickly approaches zero (see Fig. 2). The significance of this is RBFs using gaussian transforms are good only when the input data stays near or within the RBF centers. When this data is far away, then the RBF will give poor results. In other words, RBFs are particularly bad at taking 'educated guesses' when given data they have not seen before.

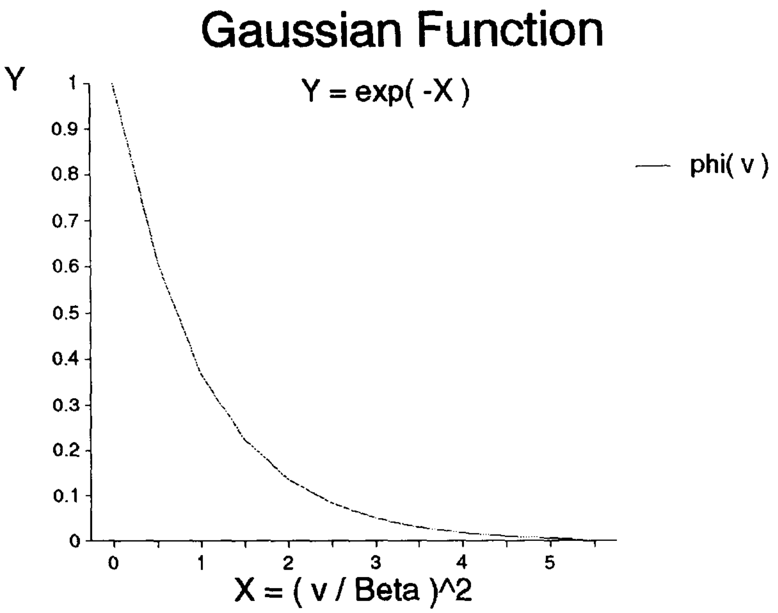


Fig. 2. Graph of the Gaussian function.

2.3. Certainty factors

Certainty factors were introduced by Buchanan and Shortliffe [5] for the medical diagnostic system, MYCIN. Certainty factors are based on heuristic measurements of belief and disbelief of facts.

To explain the concept, assume that some fact, X, is asserted to be true. If the certainty factor of this assertion is 1.0 then it is believed to be 100% true. If the certainty factor of this assertion is 0.0, then it is believed to be 0% true, or 100% false. Any degree within the range of 0.0 and 1.0 inclusive is allowed, and can all be interpreted to mean percent belief that a fact is true.

Certainty factors should not be confused with probabilistic measures, since there is only a small correlation between the two. For example, a man may assert that he is 90% certain that he will pick up milk on the way home from work. This does not mean the same thing as there is a 90% probability that the man will purchase the milk. In the first case, it has the meaning that the man is reasonably confident that he will buy the milk; this is what most people understand the man to mean from his statement. The second meaning implies that all things being equal, the man will buy milk exactly 9 times out of 10.

3. RBFs generating certainty factors

RBF neural networks are adept at giving answers when the input data is familiar; but when the data is unfamiliar then the answers arrived at are virtually

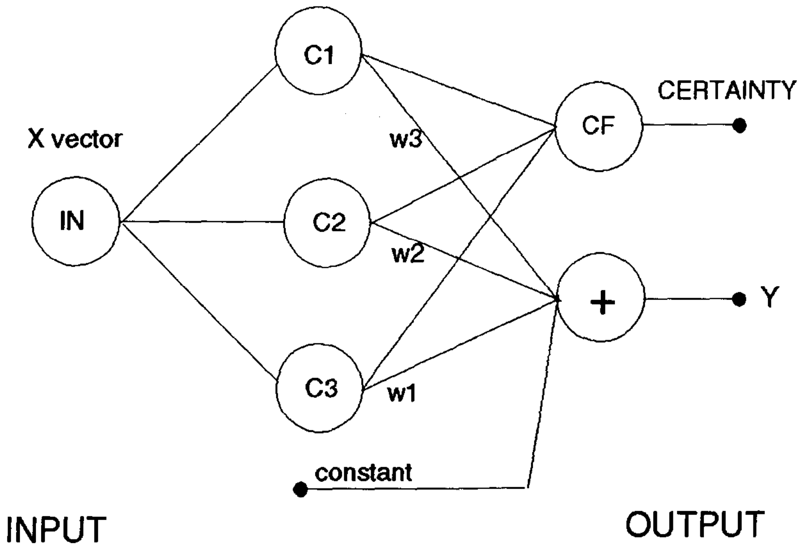


Fig. 3. RBF neural network modified to generate certainty factors along with normal output.

meaningless. Therefore, to get the most accurate results from the RBF neural network it is essential to identify good and bad answers. Once identified, the good results are maintained while the bad results are discarded. Having the RBF neural network itself indicate good and bad results is the most direct approach to this problem.

Using RBF neural networks to generate a reliability measure is not a unique idea. Lee [11] used this principle in a handwritten digit recognition program and achieved good results. Lee used as a measure of confidence the numeric difference between the two output nodes with the highest values. Leonard, Kramer, and Unger [12,13] explored two methods of measuring the reliability of prediction: the maximum activation value function and the Parzen estimator and concluded the Parzen method superior. The approach proposed in this paper is a variation of this idea.

This paper will describe a new reliability measure for RBF neural networks. This new reliability measure is the certainty factor. Certainty factors can be generated by RBF's by using the $\phi(v)$ values from the basis functions in the hidden layer. Referring to Eq. (6) and its graph in Fig. 2, it is obvious that this function returns values between 1.0 and 0.0 which is also the same range as allowed for by certainty factors.

When the output $\phi(v)$ of a hidden layer node is high (near 1.0), then this indicates that a point lies near the center of a cluster. This means that the data is familiar to that particular node and therefore the overall network is confident in the answer. A value of 1.0 exactly indicates the data falls directly upon the center of a node. A value that is very small (near 0.0) will lie far outside of a cluster. This data is therefore unfamiliar to that particular node. If all the nodes have a small

ϕ(v) value, then the network's certainty in the answer is small and the network is not confident in the result.

The problem with this scheme is that N values of ϕ(v) are generated, or one for each node in the hidden layer of the RBF neural network. The question arises of how a confidence measure should be determined from all these values. There are many possible methods of combining them into a certainty factor. The obvious approach is to take the maximum ϕ(v) and consider it to be the certainty factor. This equation is the same as the maximum activation value function proposed by Leonard, Kramer, and Unger [12,13] to measure if enough training data was in the vicinity of the test point for the neural network to make a reliable prediction. In their work, they asserted that the use of the maximum value is not an ideal measure of the sufficiency of training because it only indicates that some data are in the area of the test point, and not the amount or density of the training data in the region.

From a certainty factor standpoint, the disadvantage of using only the maximum ϕ(v) value when determining the certainty factor is that it disregards the certainty of the other nodes. This may result in an instance when two or more of the hidden nodes are reasonably familiar with an input data, but no node is extremely familiar with it. Therefore, the calculated certainty factor would be lower than expected which could cause good results to be discarded.

In order to factor in the certainty of other nodes, a variation of the maximum ϕ(v) equation is used which calculates the certainty using the recursive formula:

$$CF_i = CF_{i-1} + (1 - CF_{i-1})\max_i(\phi(v)) \tag{7}$$

In Eq. (7), the value of i refers to the number of nodes that will be used to determine the certainty factor of the RBF network. In general, the value of i should be set to N (the number of nodes in the hidden layer), but it can also be set to a lower value if computational time is critical to the application. The term CF_{i-1} refers to the certainty factor of the network using only i − 1 nodes to calculate the confidence. The value of CF_0 (the certainty when no nodes in the hidden layer are used) is defined to be 0.0. Lastly, the term $\max_i(\phi(v))$ is defined as the ith largest value of ϕ(v). So, $\max_1$ would mean the largest ϕ(v) value and $\max_2$ would be the second largest value of ϕ(v), etc. Eq. (7) can best be explained with the following example.

Example. Assume that a neural network has only four nodes (N = 4) in its hidden layer. For a given input vector, assume that the four ϕ(v) values calculated were respectively 0.6, 0.1, 0.5, and 0.2. The certainty factors for the values of i = 1..4 are given in Table 1.

From Table 1, it is observed that incorporating each additional hidden layer node increases the certainty factor by a smaller and smaller amount until the network reaches its true certainty factor value. This example of diminishing returns indicates that it is not necessary to actually use i = N to arrive at the optimum value of network certainty. A value of less than N will likely give an answer close to the true certainty. Setting the value of i too low, however, can give bad estimates.

Table 1
RBF certainty factor value using i nodes of the four node hidden layer

i	CF_{i-1}	$1-CF_{i-1}$	$\max_i(\phi(v))$	CF_i
1	0.00	1.00	0.60	0.60
2	0.60	0.40	0.50	0.80
3	0.80	0.20	0.20	0.84
4	0.84	0.16	0.10	0.86

A value of $i=1$ (which would be analogous to using Leonard, Kramer, and Unger's [12,13] maximum activation function) would greatly underestimate the network's certainty factor.

It should be noted, however, that the most accurate results will occur when i is set to N. Approximating with values smaller than N should only be done when the sorting time for calculating the $\max_i$ values becomes a significant factor in obtaining the network's output.

4. Combining UBJ and RBF models

The proposed new model for time series forecasting is shown in Fig. 4. The diagram shows that input is fed simultaneously into a trained RBF network and a UBJ forecasting model. The RBF network generates a time series forecast and a certainty factor as described above while the UBJ model generates only a time

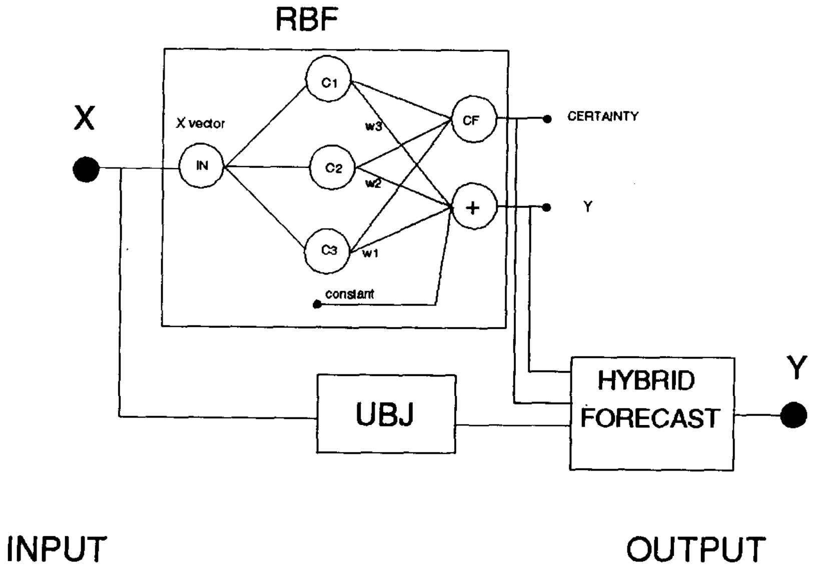


Fig. 4. RBF/UBJ hybrid time series forecasting architecture.

series forecast. The two forecasts along with the certainty factor are entered into the Hybrid Forecast block of Fig. 4. In this block, a final forecast is generated. An important consideration is exactly how the RBF and UBJ forecasts should be combined into a single forecast. Although virtually any method can be used, this paper examines three possible methods which can be used to generate the hybrid forecast.

From the RBF and UBJ models, three values are generated:
- an RBF forecast value,
- an RBF certainty value,
- a UBJ forecast value.

There are three obvious approaches that might be employed to combine these numbers into a single forecast. Each will be discussed, and then all will be experimentally tested to determine which gives the most reliable forecast.

4.1. Method 1: Winner take all approach

In this method, the user specifies a certainty cutoff point, for example 0.5. If the RBF's output certainty is greater than or equal to the cutoff value, then the RBF's forecast is used. If the RBF's certainty is less than the cutoff value, then the forecast from the UBJ model is used. There is no combination of the two forecasts in this method. The formula for this method is given as:

```
if $(Certainty \geq Cutoff)$ then
    $Forecast = RBF$
else
    $Forecast = UBJ$                                        (8)
```

where RBF and UBJ are the forecasts from the RBF network and the UBJ model respectively and Certainty is the confidence value output from the RBF neural network.

4.2. Method 2: Equal weighting approach

This method is different from the first approach in that Method 1 selects either the RBF or the UBJ forecast depending upon certainty with no combination of the two values. Although it may be reasonable to only use the value for UBJ when the confidence of the RBF output is low, the converse may not necessarily be true. Method 2, therefore attempts to always utilize the forecast of the UBJ method.

In this approach, the user again specifies a certainty cutoff value. If the RBF's output certainty is less than the cutoff, then only the UBJ value is used. If it is greater than or equal to the cutoff, then the average of the two is used.

```
if $(Certainty \geq Cutoff)$ then
    $Forecast = (RBF + UBJ)/2$
else
    $Forecast = UBJ$                                        (9)
```

The cutoff certainty used for Methods 1 and 2 must be chosen through trial and error to determine the value which yields the optimal certainty.

4.3. Method 3: Parliamentary weighting approach

This method takes the RBF's certainty factor value literally and weights the answer accordingly. If the RBFs certainty factor is, for example 0.80, then the forecast is taken as 80% RBF and 20% UBJ. The more certain the RBF network, the more the RBF value is weighted. The less certain the RBF network, the more the UBJ is weighted.

$$Forecast = (CF * RBF) + (1.0 - CF) * UBJ \tag{10}$$

The value of CF is the certainty factor calculated by the RBF neural network using Eq. (7).

5. Tests of forecast accuracy

The most important criterion for choosing a forecast method is its accuracy. Accuracy in this case is defined as how closely the forecast predicts an actual event [1]. There are several methods of measuring a time series model's accuracy, but one method that is widely used is the Mean Absolute Deviation (MAD). The MAD is calculated by summing the absolute value of all the forecast errors and dividing that sum by the number of forecasts.

$$MAD = \frac{1}{T}\sum_{i=1}^{T}\left(Y_i - \hat{Y}_i\right) \tag{11}$$

The value of T in Eq. (11) is the number of one period ahead forecasts predicted by the model. The Y_i term is the actual value of Y from Eq. (2) at time i, while $\hat{Y}_i$ is the one period ahead forecast at time i − 1.

6. Experimental results

Before testing the hybrid approach presented in this paper, an RBF network and a UBJ model had to be developed for each of the data sets used. The learning sets for the RBF networks and UBJ models came from the beginning of each of the time series and the test sets came from the end of each time series. The size of the learning sets was deliberately kept as low as possible to allow for large test sets. In real world applications, however, the training sets would be larger than presented here. The RBF centers for each data set were selected from the learning vectors using the Orthogonal Least Squares (OLS) method [6]. The UBJ model was determined using the Minitab Statistical Software [14] and was generated using the same learning set that was used to train the RBF network.

The hybrid model was tested on real world applications using all three approaches described for combining the RBF and the UBJ forecasts. The data sets used were the Wolfer Sunspot Data [18], West German monthly unemployment figures [17], and the number of monthly housing starts for single unit structures from 1959 to 1994 [8]. All three approaches were tested at certainty cutoffs ranging from 0.0 to 1.0 in increments of 0.1. Note that Method 3 is independent of a cutoff value, so its value will be a constant horizontal line in all the error graphs. It is presented in this manner for comparison purposes with Methods 1 and 2 which are dependent upon a cutoff value.

To determine the accuracy of exclusively using RBF or UBJ as opposed to this combination approach, the reader can examine the Method 1 line on Hybrid Forecasts – MAD plots. At a 0.0 cutoff of the Method 1 one (the far left of Hybrid Forecasts – MAD plot), all RBF values are considered acceptable so this would be the same as using RBF forecasts exclusively. At the 1.0 cutoff, virtually all of the RBF points are discarded as unacceptable, so only UBJ forecasts will be used. Therefore, the far right of the plot is the accuracy of using only UBJ forecasts.

6.1. Sunspot forecasting

The first test was conducted on the Wolfer sunspot data [18]. This data set represents the average number of sunspots observed each day for an entire year for the years from 1700 to 1979. Typically the data is within the range of 0 to 200 sunspots. Both the RBF network and the UBJ model were trained on the data 1700 to 1795 and the hybrid test was conducted on the remaining data.

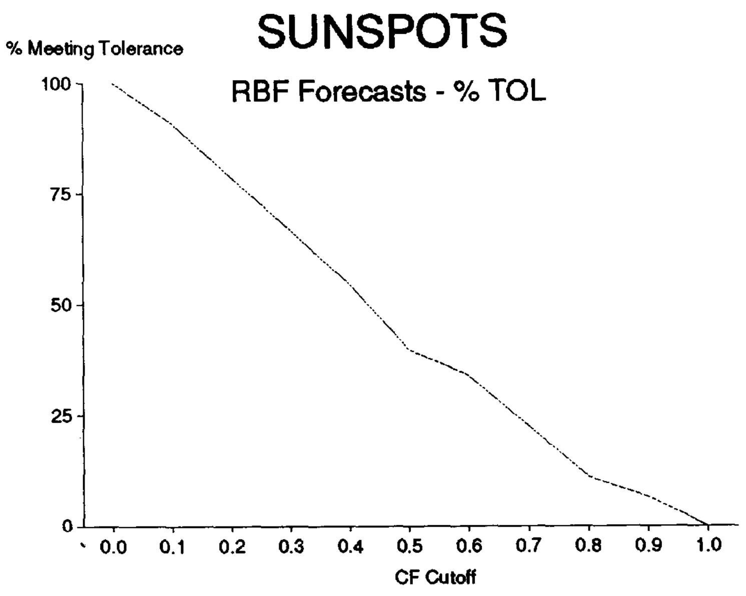


Fig. 5. RBF forecasts – percent meeting tolerance.

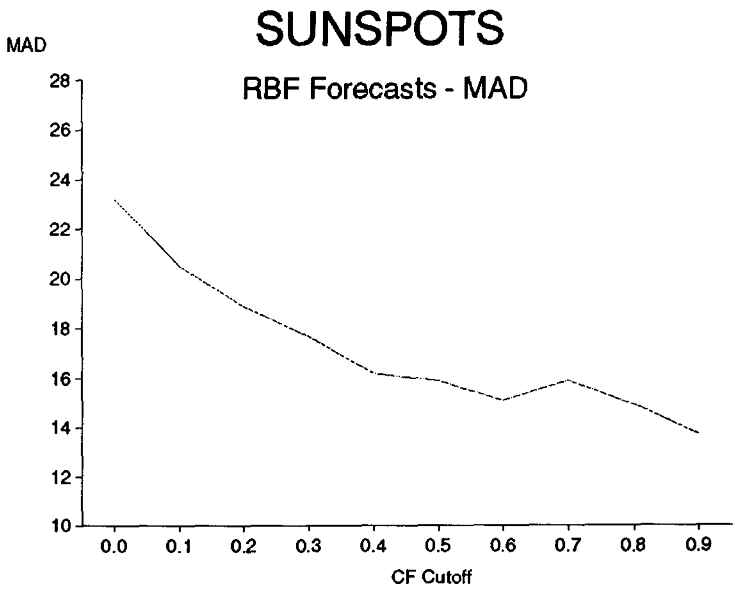


Fig. 6. RBF forecasts – mean absolute deviation.

The optimal RBF model used a training set of 93 vectors each with a length of k = 4. From these training points, 48 were determined to be center points using the Orthogonal Least Squares (OLS) [6] algorithm. The radii of these center points were all preset to a constant value of 0.1. For the calculations of the RBF network, all the training and test vectors were normalized and their forecasts had a bias term added to it equal to the mean value of the training set. The optimal UBJ model was found to have 3 autoregressive and 1 moving average components.

Before examining the accuracy of the hybrid approach, a brief explanation will be given illustrating how pruning low certainty results yields higher accuracy. Referring to Fig. 5, it is seen that when the certainty cutoff is set to 0.0 then all of the test vectors are able to meet this tolerance and so 100% of the forecasts are accepted. As the certainty factor cutoff is increased, fewer of the test points are able to meet tolerance and more are rejected. Eventually, at a certainty cutoff of 1.0, no test point meets tolerance and so 0% of the test points remain. The effect that pruning has on the error is seen in Fig. 6. As the results having low certainty factors are removed, the MAD gets progressively lower. Referring again to Fig. 5 and Fig. 6, when the certainty cutoff is 0.0 and 100% of the data is used, then the MAD is 23.2. For a cutoff of 0.9, the MAD drops to 13.7. The drawback to setting the cutoff to 0.9 is that only 6.6% of the results are used. A time series model that can only be used 6.6% of the time is of little use in real world applications. This demonstrates both the need for the RBF certainty factors and the RBF-UBJ hybrid approach.

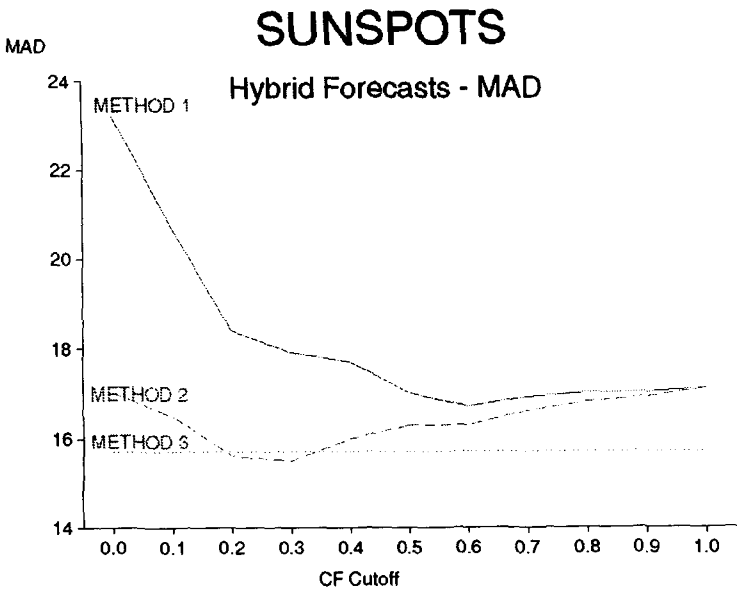


Fig. 7. RBF-UBJ hybrid – mean absolute deviation.

The RBF network was combined with the UBJ model using three previously defined methods. The results are shown in Fig. 7. Examining Methods 1 and 2, it is observed that their accuracy is best in the mid range certainty cutoff points. This can be explained by considering the fact that when the cutoff is set very low, then a large percent of the RBF forecasts are used no matter how unreliable. When the cutoff is set at a very high level, virtually no RBF forecast can pass the tolerance test and so the forecast is the same as using the UBJ method alone. A very high certainty causes the hybrid model to lose the advantage of the RBF's pattern recognition ability. As stated before, Method 3 does not depend upon a cutoff value, so it is superimposed as a straight line on the graph for the purpose of comparison with Methods 1 and 2. Further examination of Fig. 7 shows that Method 2 is always a better choice in this application than is Method 1 and the best overall results occur when the cutoff of Method 2 is set to 0.3. Method 3 gives excellent results as well without the worry of selecting the correct cutoff value.

6.2. West German unemployment forecasting

For the second test set, the RBF and UBJ models were trained and tested using West German monthly unemployment figures from 1948–1980 [17]. This data has wide variations ranging from as low as 90 thousand to as high as 2.3 million. Both the RBF network and the UBJ model were trained on the data 1948 to 1964 and the hybrid test was conducted on the remaining data.

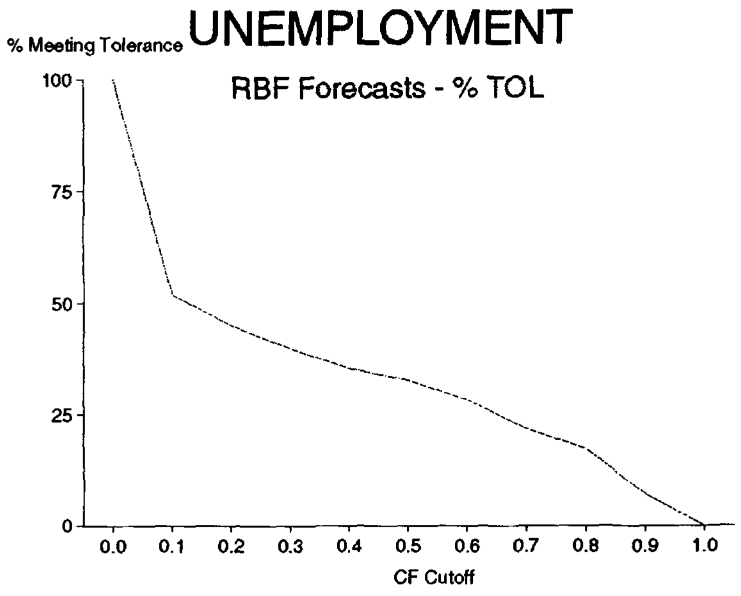


Fig. 8. RBF forecasts – percent meeting tolerance.

The optimal RBF model used a training set of 195 vectors each with a length of $k = 3$. From these training points, 170 were determined to be center points using the OLS algorithm. The radii of these center points were preset to a constant value of 0.015. For the calculations of the RBF network, all the training and test vectors were normalized, and their forecasts had a bias term added to it equal to the minimum value of the training set. The optimal UBJ model was found to have 1 autoregressive and 2 moving average components.

Again before examining the accuracy of the hybrid approach, the accuracy of the RBF neural network alone will be examined. Observing Fig. 8 and Fig. 9, it is seen that when the cutoff is set to 0.0 and all of the RBF network forecasts are used then the MAD is almost 400,000. This error drops considerably as the certainty cutoff is increased. The optimal forecast results are achieved at a 0.7 cutoff. At this cutoff, the percent meeting tolerance is 21.7% while the MAD has dropped from 400,000 to under 19,000. At the higher cutoff level of 0.9, the MAD increases slightly to approximately 27,000 while only 7.1% of the forecasts used. This increase is caused by the extensive pruning which results in only a small number of forecasts remaining. When few forecasts remain, error from noise or insufficient network training becomes a more significant factor. Even with this condition of the MAD slightly increasing, the overall accuracy shows a drastic improvement from the original MAD error.

Examining the RBF-UBJ hybrid results of Fig. 10, it is again observed that Method 2 outperforms Method 1 consistently at almost every certainty cutoff point. The only exception occurs at a certainty cutoff of 0.7 when Method 1 has a

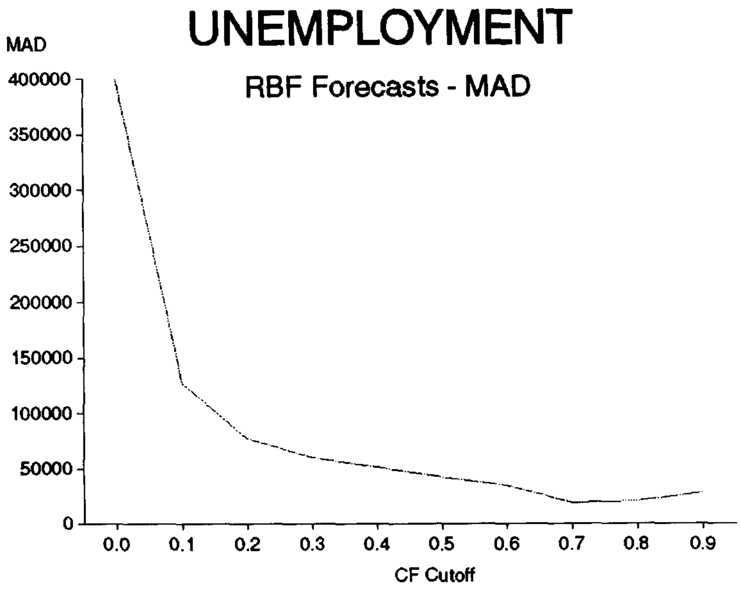


Fig. 9. RBF forecasts – mean absolute deviation.

MAD of 47,300 while Method 2 has a MAD of 48,300. It is also observed that for Method 1 and Method 2 the best results are achieved at mid range certainty cutoff points with the optimal cutoff level being at 0.7. Lastly it is again observed that

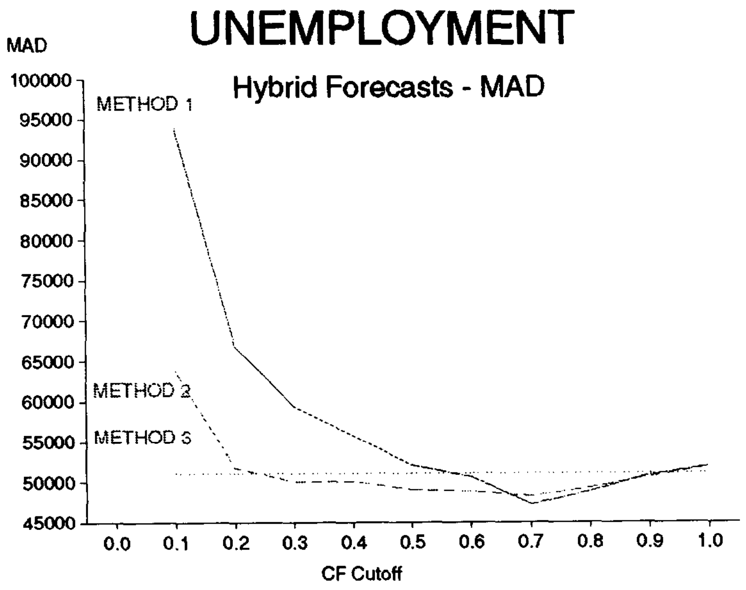


Fig. 10. RBF-UBJ hybrid – mean absolute deviation.

Method 3 gives good results as its MAD of 51,000 is better than using only RBF networks or UBJ networks but it is not significantly better in this application than using UBJ forecasting alone which has a MAD of 51,800. It was seen in this application that Method 2 again outperformed Method 1 at all levels except the optimal cutoff which had Method 1 only marginally better than Method 2. Method 3 again gave good results, but was not significantly better than using only the UBJ model.

6.3. Housing start forecasting

For the final test set, the RBF and UBJ models were trained and tested using the seasonally and annually readjusted number of monthly housing starts for single unit structures from 1959 to 1994 [8]. All numbers expressed in this data set are in units of 1000's. The training set had wide fluctuations with housing starts ranging from 596 to 1432 units. Both the RBF network and the UBJ model were trained on the data 1959 to 1975 and the hybrid test was conducted on the remaining data.

The optimal RBF model used a training set of 190 vectors each with a length of $k = 3$. From these training points, 116 were determined to be center points using the OLS algorithm. The radii of these center points were preset to a constant value of 40.0. For the calculations of the RBF network, the training and test vectors were not normalized and their forecasts had a bias term added to it equal to the mean value of the training set. The optimal UBJ model was found to have 1 autoregressive and 1 moving average components.

Examining Fig. 11 and Fig. 12 it is again observed that pruning low certainty output causes the MAD of the time series forecasts to decrease. When the cutoff

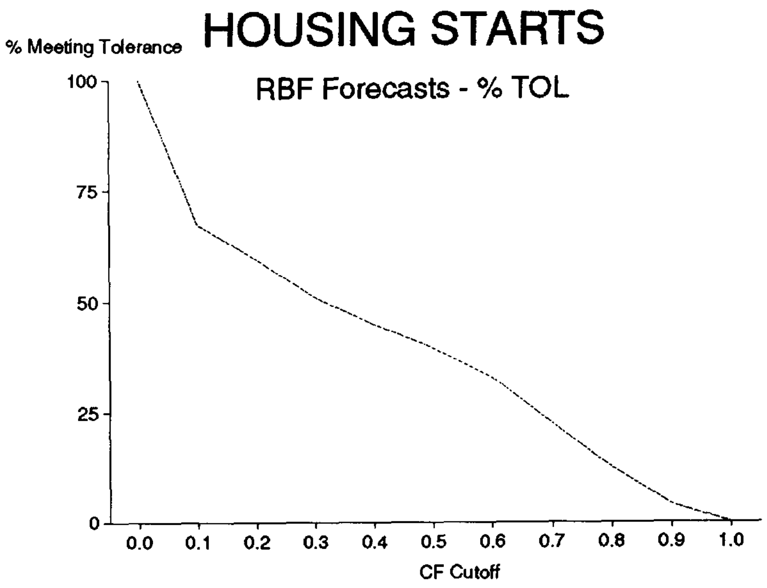


Fig. 11. RBF forecasts – percent meeting tolerance.

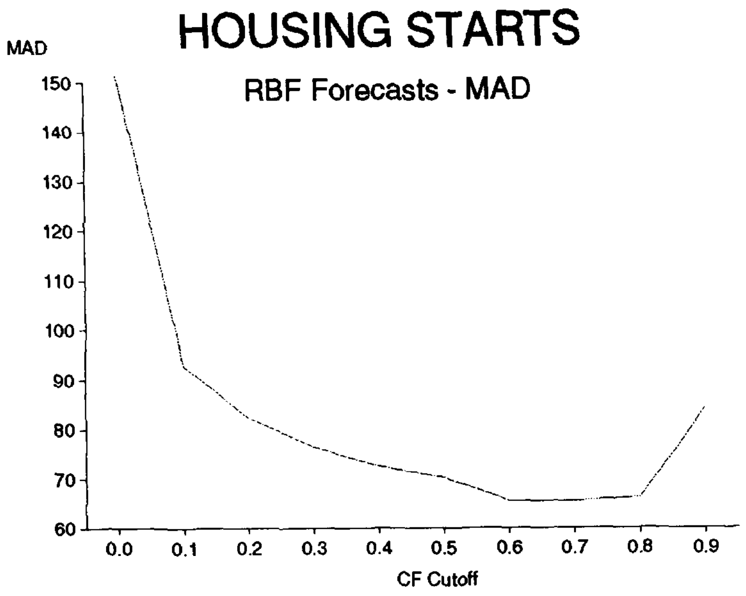


Fig. 12. RBF forecasts – mean absolute deviation.

value is set to 0.0 and all data is accepted, the MAD is 151.4. It is observed from the two figures that as the certainty cutoff increases the MAD is reduced until the optimal accuracy is achieved at a certainty cutoff of 0.7. At this cutoff, the MAD is only 65.5. As seen in the previous data sets, increasing the certainty cutoff to levels approaching 1.0 causes the MAD to increase slightly. Again, this can be explained by the fact that at very high cutoffs only a very small amount of the test data remain. Any errors that occur at this high cutoff will have a greater impact on the MAD than if there were a larger number of test vectors remaining.

The results of the hybrid approach are shown in Fig. 13. It is again seen that Method 2 significantly outperforms Method 1 at all levels and both methods outperform Method 3 at higher cutoff levels. The optimal forecasting results are achieved in this application by using Method 2 at a certainty cutoff factor of 0.7 which yields a MAD value of 64.9. As the cutoff approaches 1.0, again the MAD begins to increase to 65.7. This is due to the fact that at this cutoff level, none of the RBF forecasts were used in the hybrid model. It is also important to note that Method 3 had a MAD of 67.8 which was slightly higher than using only the UBJ model alone. This may indicate that the RBF network trained on this housing start data may need to be retrained with more data.

7. Conclusions

This paper introduced a new concept with Radial Basis Function neural networks: the certainty factor. This paper described how an RBF neural network

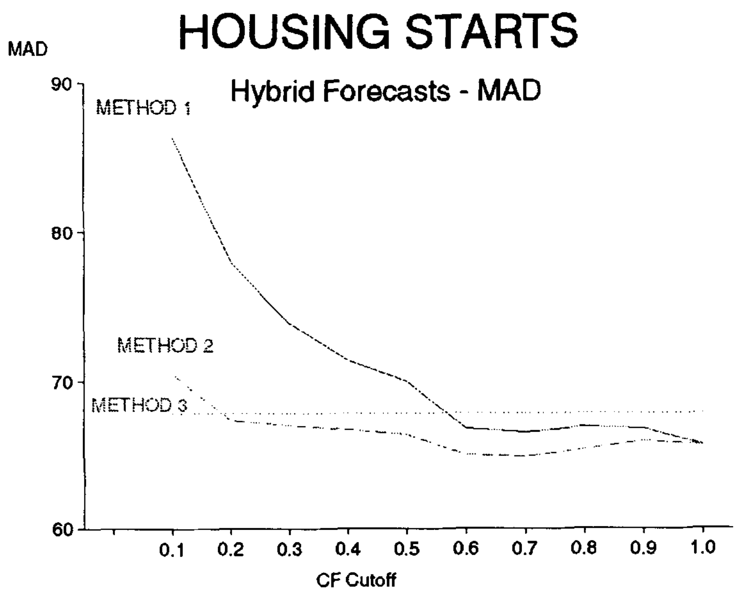


Fig. 13. RBF-UBJ hybrid – mean absolute deviation.

could improve its accuracy by generating a certainty factor along with its normal output. When output with a low certainty factors were removed, the accuracy of the network increased. The paper then applied the concept of RBF's and certainty factors to the problem of time series forecasting. The RBF network was trained to forecast future values from past values of a data set along with a certainty factor. This output was then combined with the future value forecasts from the Univariant Box-Jenkins method. The paper described three possible ways that the two forecasts could be combined into a single hybrid forecast by using the certainty factor value generated by the RBF network.

The experimental results show that the use of certainty factors does improve the overall accuracy of RBF neural network results. It was observed that after removing data points with low certainty, the average error of the network decreased. It was also observed that an optimal certainty cutoff point exists for each data set. At this cutoff point, the average error of the system is at its lowest point. As the cutoff is increased beyond the optimal point, the average error begins to increase slightly. This increase is due to the fact that at higher cutoff levels, so few points remain that noise error can have a greater impact on the average error. At the present, the optimal certainty factor cutoff must be determined through a trial and error approach. An algorithm or heuristic to find this point is the subject of current research.

The paper applied the concept of using RBF networks and certainty factors to time series forecasting and discussed three possible methods for combining the two models. Of the three models discussed, the most accurate appears to be Equal

Weighting Approach (Method 2) which uses the average of the RBF and UBJ forecasts when the certainty factor is above the cutoff value and only the UBJ forecast when the certainty factor is below the cutoff value. Method 2 was consistently better than Methods 1 or 3 and better than using RBF or UBJ forecasting alone. Method 2 does, however, require the user to determine the optimal certainty factor cutoff point which again must be found through a trial and error approach. Although the hybrid architecture for time series forecasting is capable of giving significantly better results, these results must still be weighed against the added costs in terms of the more complicated architecture. The architecture requires increased design time maintenance costs, since both the RBF and the UBJ models must be trained and monitored.

References

[1] B. Abraham and J. Ledolter, Statistical Methods for Forecasting (John Wiley & Sons, New York, NY, 1983).

[2] G.E.P. Box and G.M. Jenkins Time Series Analysis: Forecasting and Control (Holden-Day, San Francisco, CA, 1970).

[3] C. Bishop, Improving the generalization properties of radial basis function neural networks, Neural Computat. 3 (1991) 579–588.

[4] D.S. Broomhead and D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Syst. 2 (1988) 321–355.

[5] B.G. Buchanan and E.H. Shortliffe, Rule-based Expert Systems (Addison-Wesley, Reading, MA, 1985).

[6] S. Chen, C.F.N. Cowan and P.M. Grant, Orthogonal least squares learning algorithm for radial basis function network, IEEE Trans Neural Networks (2) (March 1991) 302–309.

[7] S. Chen, S.A. Billings, C.F.N. Cowan and P.M. Grant, Non-linear systems identification using radial basis functions, Int. J. Systems Sci. 21 (12) (1990) 2513–2539.

[8] CITIBASE: Citibank economic database [machine-readable magnetic data file]; 1946–1993; New York, Citibank, N.A. 1978.

[9] V. Kadirkamanathan, M. Niranjan and F. Fallside, Sequential adaption of radial basis function neural networks and its application to time-series prediction, Neural Information Processing Systems 3, R.P. Lippman, J.E. Moody and D.S. Tourefzky, eds. (Morgan Kaufmann, 1991).

[10] A.S. Lapedes and R. Farber, Non-linear signal processing using neural networks: Prediction and system modelling, Technical report, Los Alamos National Laboratory; Los Alamos, New Mexico 87545, 1987.

[11] Y. Lee, Handwritten digit recognition using K nearest-neighbor, radial-basis function, and back-propagation neural networks, Neural Computat. 3 (3) (1991) 440–449.

[12] J.-A. Leonard, M.A. Kramer and L.-H. Unger, Using radial basis functions to approximate a function and its error bounds, IEEE Trans. Neural Networks (4) (July 1992) 624–627.

[13] J.A. Leonard, M.-A. Kramer and L.-H. Unger, A neural network architecture that computes its own reliability, Comput. Chem. Eng., 16 (9) (1992) 819–835.

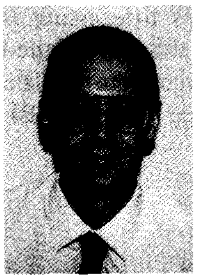
[14] Minitab Inc. Minitab Reference Manual Release 7, (Minitab Inc., State College, PA, 1989).

[15] J. Moody and C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Computat. 1 (1989) 281–294.

[16] D.E. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1: Foundations (MIT Press, Cambridge, MA, 1986).

[17] T.S. Rao and M.M. Gabr, An Introduction to Bispectral Analysis and Bilinear Time Series Models (Springer-Verlag, New York, NY, 1984).

[18] H. Tong, Threshold models in non-linear time series analysis, Lecture Notes in Statistics (Springer-Verlag, New York, NY, 1983).
[19] A.S. Weigand, B.A. Huberman and D.E. Rumelhart, Predicting the future: A connectionist approach, Int. J. Neural Syst. 1, (1990) 193–210.
[20] Y.-f. Wong, How gaussian radial basis functions work, Proc. Int. Joint Conf. Neural Networks, Vol. 2 (July 1991) 133–138.



Donald K. Wedding is a Ph.D. student at The University of Toledo. He is currently doing research on improving the accuracy of RBF neural networks. He received a B.S. in electrical engineering in 1987 and his M.S. in engineering science in 1988. After graduating, he worked in the defense industry as a software engineer for three years. In 1991, he received a second M.S. degree, this time in business management with a specialization in finance and information systems. Mr. Wedding was awarded his P.E. license in 1993 and expects to graduate with his Ph.D. in Engineering in 1995.

Krzysztof J. Cios received an M.S. degree in electrical engineering and a Ph.D. in computer science, both from Technical University (AGH), Krakow, Poland, and an M.B.A. degree from The University of Toledo.

He is an Associate Professor of Electrical Engineering at The University of Toledo, Toledo, OH, USA. His research interests are in the area of neuro-fuzzy systems, machine learning, and pattern recognition. His research was funded, among others, by the National Science Foundation, NASA, and the American Heart Association. He has published extensively in journals and in conference proceedings. Dr. Cios consults for several US companies. He is a senior member of the IEEE and a member of Sigma Xi. He is on editorial boards of the IEEE Transactions on Fuzzy Systems, Neurocomputing, and the Handbook of Neural Computation.