

# Application of radial basis function neural network model for short-term load forecasting

D.K. Ranaweera  
N.E. Hubele  
A.D. Papalexopoulos

*Indexing terms: Artificial neural network, Confidence intervals, Radial basis function, System load forecasting*

**Abstract:** A description and original application of a type of neural network, called the radial basis function network (RBFN), to the short-term system load forecasting (SLF) problem is presented. The predictive capability of the RBFN models and their ability to produce accurate measures that can be used to estimate confidence intervals for the short-term forecasts are illustrated, and an indication of the reliability of the calculations is given. Performance results are given for daily peak and total load forecasts for one year using data from a large-scale power system. A comparison between results from the RBFN model and the Back-propagation neural network are also presented.

## 1 Introduction

Short-term system load forecasting (SLF) is an essential function in electric power system operations and planning. Forecasts are needed for a variety of utility activities such as generation scheduling, scheduling of fuel purchases, maintenance scheduling and security analysis. Significant forecasting errors can lead to either excessively conservative scheduling or excessively risky scheduling, that can induce heavy economic penalties. Large savings can be achieved if accurate load forecasts are used to support these activities. The time horizon of the forecast depends on the way the forecast will be used. For example, when providing inputs to a unit commitment program, the time horizon could be a 24-hour period. For utility activities such as fuel purchase or maintenance scheduling, it could be a 168 hour-forecast.

A wide variety of techniques for short-term load forecasting have been reported. These techniques typically use two basic models: peak-load models and load-shape static or dynamic models [1-5]. These models vary in complexity, data requirements, flexibility, and ability to meet user specifications. These specifications usually require that the SLF programs be reliable during all seasons, during periods of unusual weather conditions,

e.g., warm fronts in the winter and cold fronts in the summer, and also respond accurately and consistently to system changes.

Extensive practical experience with these conventional statistical methods has confirmed their theoretical limitations that inhibit further performance improvements and their inability to meet some or all of the above requirements. These limitations are:

(i) the nonlinear relationships of the input and output variables are difficult to capture

(ii) the collinearity problem of the input variables limits the number of these inputs that can be used in the model

(iii) the models are not very flexible in the face of rapid system load changes.

However, recent progress [6-9] in the application of artificial neural network (ANN) technology to load forecasting appears to indicate that it is possible to use this technology to overcome these limitations. The type of neural networks applied to load forecasting can be classified as supervised learning-based networks [6, 8] and unsupervised learning-based networks [9]. The multi-layer perceptron is an example of supervised learning neural network, and the Kohonen network is an example of an unsupervised learning network. The inherent flexibility of the ANN technology is somewhat superseded by its apparent inability to provide: first, an insight into the nature of the problem being solved; and secondly, a complete set of systematic design procedures. The proper selection of network topology, learning rules and parameters, activation functions and threshold represent some of the obstacles to the practical use of supervised learning networks. Even for the Kohonen network, it is necessary to include a second neural network based on supervised learning to obtain the final forecast [9]. An additional factor that casts doubts on the practical usefulness of the ANN technology is the lack of procedures to determine:

first, whether or not the ANN model is being applied in a domain of the input variables where training data was available (extrapolation); and secondly, the accuracy of the predictions for given values of the input variables (local goodness of fit). Since currently proposed ANN models, as applied to a load-forecasting problem, do not flag mathematically when they are extrapolating from the training data, users may inadvertently use an invalid forecast as a result of extrapolation. This is because load predictions should be suspect when extrapolating beyond the range of the original training data. Due to nonlinearities and multiple correlated inputs, it is difficult to recognize when the ANN models are extrapolating. Furthermore, ANN models may have local areas of poor

---

© IEE, 1995  
Paper 1602C (P9, PI0), first received 22nd February and in revised form 31st August 1994  
D.K. Ranaweera and N.F. Hubele are with the Arizona State University, Tempe, Arizona, USA  
A.D. Papalexopoulos is with the Pacific Gas and Electric Company, San Francisco, California, USA

performance even when they are not extrapolating. The reason is that the training examples and the noise are usually nonuniformly distributed over the input domain. Classical techniques for training ANN models use only global measures of goodness of fit which usually fail to identify and determine local regions of poor fit. A local goodness of fit can be expressed as a confidence interval for each ANN output as a function of the input variables. The determination of confidence intervals for each load forecast has been one of the main unmet needs of SLF users who want to apply ANN technology to solve for SLF problems.

## 2 Radial basis function network (RBFN) model

The RBFN model consists of three layers; the input, hidden and output layers. The nodes within each layer are fully connected to the previous layer, as shown in Fig. 1. The input variables are each assigned to a node in the input layer and pass directly to the hidden layer without weights. The hidden nodes or units contain the radial basis functions (RBF), also called transfer functions, and are analogous to the sigmoid functions commonly used in the back-propagation network models. They are represented by the bell-shaped curve in the hidden nodes shown in Fig. 1.

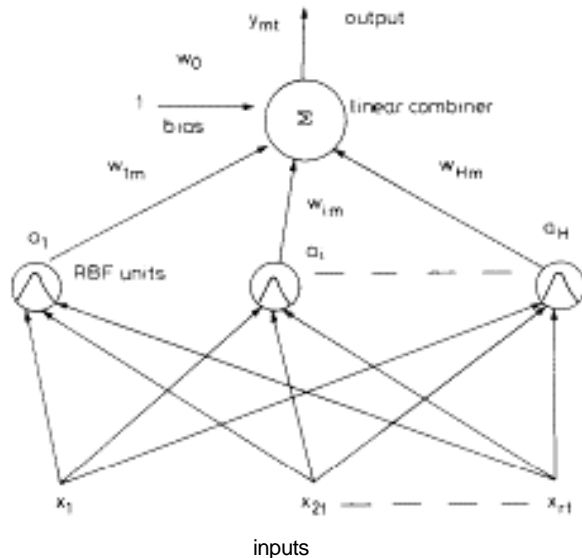


Fig. 1 Radial basis function network model for SLF

The RBF is similar to the Gaussian density function which is defined by a ‘centre’ position and a ‘width’ parameter. The Gaussian function gives the highest output when the incoming variables are closest to the centre position and decreases monotonically as the distance from the centre increases. The width of the RBF unit controls the rate of decrease; for example, a small width gives a rapidly decreasing function and a large value gives a slowly decreasing function.

Consider an observation used to train the model to have  $r$  inputs variables such as previous load values, forecasted weather readings, etc. Because all inputs are connected to each hidden node, each hidden node has an  $r$ -dimensional centre, but only one width value is used to scale all  $r$ -dimensions. The selection of the values of these centres and widths will be discussed in the Appendix. Let  $X_i$  be the incoming vector with components,  $x_{1i}, x_{2i}, \dots, x_{ri}$ . The output of the  $i$ th unit,  $a_i(X_i)$ , in the hidden layer

for the above input pattern is equal to

$$a_i(X_i) = \exp \left( - \sum_{j=1}^r \left[ \frac{x_{jt} - \hat{x}_{jt}}{\sigma_i} \right]^2 \right) \quad (1)$$

where

- $\hat{x}_{jt}$  = centre of  $i$ th RBF unit for input variable  $j$
- $\sigma_i$  = width of  $i$ th RBF unit
- $x_{jt}$  =  $j$ th variable of input pattern

The connections between the hidden units and the output units are weighted sums as shown in Fig. 1. The output value ( $y_{mt}$ ) of the  $m$ th output node is equal to the summation of the weighted outputs of the hidden units and the biasing term of the output node, and is described by eqn. 2.

$$y_{mt} = \sum_{i=1}^H w_{im} a_i(X_i) + w_0 \quad (2)$$

where

- $H$  = number of hidden layer nodes (RBF functions)
- $y_{mt}$  = output value of  $m$ th node in output layer for the  $t$ th incoming pattern
- $w_{im}$  = weight between  $i$ th RBF unit and  $m$ th output node
- $w_0$  = biasing term at  $m$ th output node

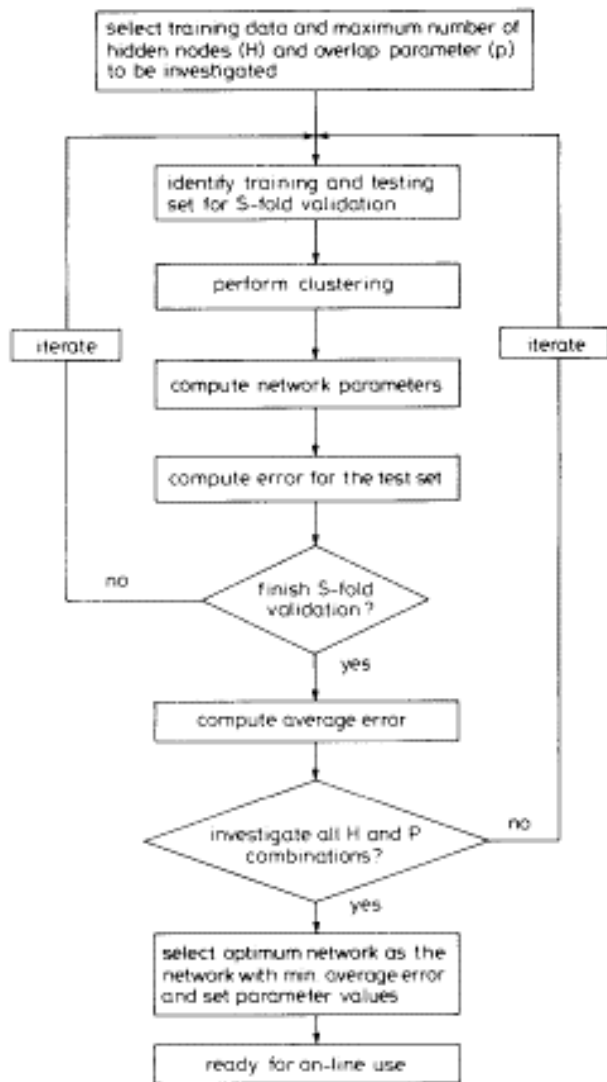
The parameters of the RBF units are determined in three steps of the training activity. First, the unit centres are determined by some form of clustering algorithm. Then, the widths are determined by a nearest-neighbour method. Finally, weights connecting the RBF units and the output units are calculated using multiple linear regression techniques. The techniques for the calculation of the parameters of the RBF units are presented in the Appendix.

The design parameters, i.e., the number of RBF units in the hidden layer ( $H$ ) and the value of the overlap parameter ( $p$ ) for the nearest neighbour (described in the Appendix) are chosen by the model builder to attain the optimal network structure for better performance. The parameters can be easily determined using a method called the  $S$ -fold cross-validation procedure (SFCV) [11]. In this procedure the training data set is first randomly divided into  $S$  equal-sized data sets. Then, for a given  $H$  and  $p$ , RBFN is trained using  $S - 1$  data sets and the remaining subset is used to test the network's generalization ability. The mean square difference between the target outputs and the predicted outputs is the error associated with the testing subset. This procedure is repeated  $S$  times, using different  $S - 1$  subsets for training and a different subset for testing at each time. The mean square error over all the testing subsets is the error estimate for the given RBFN network. This procedure is repeated several times with different values of  $H$  and  $p$  to obtain the ‘optimum’ network structure with the minimum mean square error. (Clearly, the infinity set of combinations of  $H$  and  $p$  is not completely exploited, such that, the network may not be optimal in the global sense.)

The reliability measures, i.e., the extrapolation index to check the extrapolation and confidence intervals, can easily be included in the RBFN network as described in Reference 10. Extrapolation is defined as any local region of the input space with little or no training data to support a model forecast. Extrapolation is thus related to the estimation of local training data ‘density’. It works as follows: the network is trained, and the estimates of data

computed data densities, a threshold value is defined to identify the extrapolation. The threshold value can be

**Fig. 2** Flowchart of RBFN implementation and optimization of SLF



the minimum value of the data densities computed for the training set or a value close to it. Then, a new input observation or testing case generates a forecast and an extrapolation index. If this index is below the threshold, it indicates that the model is extrapolating and the forecast is unreliable. This concept will be demonstrated using actual data in Section 3. The computational details for the extrapolation index are given in the Appendix.

The second reliability measure, the confidence interval, determines the accuracy of the ANN model when the network is not extrapolating beyond its training set. The confidence interval provides the SLF user with an indication of the goodness of the fit for a given forecast. If an RBFN model has more than one output node, confidence intervals may be computed for each of the outputs of an RBFN. To simplify the following discussion, only one output is considered. The accuracy of the forecast can be evaluated by the width of the confidence interval on the prediction. This width is a function of the error estimates obtained during the training of the network and the outputs of the hidden nodes for the inputted test pattern. As discussed earlier, the SFCV method allows the user to estimate the variance by withholding training

patterns and using them as test patterns. The mean squared error of these test patterns provides an estimate of the variance of the model. The details for computing the confidence intervals are given in the Appendix. The half-width of the confidence interval, is added and subtracted from the network forecast to provide the SLF user with a 'minimum' and 'maximum' for the forecast. For example, if the half-width is 500 MW of the forecast, then the forecast may be reported as a value, plus or minus 500 MW.

For RBFN models, all the parameters that are necessary for implementing confidence intervals and the extrapolation index are calculated as an integral part of the training of the RBFN. These reliability measures are implemented as additional output nodes to the underlying RBFN using information available from the training procedure. A flowchart of the procedure is shown in Fig. 2.

### 3 Numerical results

This Section presents some of the comparisons between the RBFN model and the more commonly used Back-propagation (BPN) model. Results from the calculation of the extrapolation index and the confidence intervals are also presented. The results were obtained using Pacific Gas and Electric Company's (PG&E) load data for 1986. Load data for 1985 were used to train the networks. The following holidays were excluded from both the training data set and the testing data set to ensure that the observations were free from any irregular load patterns: New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving Day and Christmas Day. The input data was normalized such that the mean and the standard deviation are equal to zero and one, respectively, to avoid biasing towards any of the input variables as a consequence of magnitude differences. The RBFN model has two separate modules. One module forecasts peak load for the next day and the other module forecasts total load (energy) for the next day. Each module has seven different neural networks to generate forecasts for each day of the week. The inputs to each of the neural network are: (a) forecasting of peak load for day  $t$ : peak load of the previous day ( $t - 1$ ) and forecast minimum and maximum temperatures for the day  $t$ ; and (b) forecasting of total load for day  $t$ : total load of the previous day ( $t - 1$ ) and forecast minimum and maximum temperatures for the day  $t$ . Experiments with temperature values of day ( $t - 1$ ) as additional inputs, have also been carried out, but there was no significant improvement in the forecasting accuracy.

Forecast maximum and minimum temperatures at six different locations in California were used in the model. Thus, the input layer of each neural network has 13 nodes. The output layer has three nodes, one for the forecast load, one for the confidence interval in the forecast, and one for the extrapolation index of the forecast.

To find the optimal network structure (i.e., design parameters  $H$  and  $r$ ), a set of RBFN models for each of the days was trained using the procedure described earlier in this paper. In these simulations, the number of RBF units varied from 2 to 14 and the overlap parameter ( $r$ ) varied from 2 to 10. Based on the cross-validation results, it was found that the neural networks perform very well with  $H = 10$  and  $r = 8$ .

A back-propagation model (BPN) has also been developed to compare the performance of the RBFN model. The training and the testing of the BPN model were performed with the training and testing data set mentioned earlier in the paper. The input layer of the BPN model

has 13 nodes and the output layer has one node. The BPN model has only one hidden layer. The network was trained with a variety of hidden nodes to find the optimal one. The training was terminated when the training set error (for the 1985 data set) continued to decline and the test set error (for the 1986 data set) began to increase. This balancing act prevented the overfitting of the training data set. Based on the training results, the BPN model, for any day and for both peak and total load forecasts, showed superior performance when the number of hidden nodes was equal to 10. Both the RBFN and BPN models used for the simulations were coded in FORTRAN and run on a VAX 8600 machine. The average CPU time to train the RBFN model with 13 inputs and 50 training patterns was less than 0.11 s. However, the BPN model took more than an hour of CPU time to train the same data set. Both RBFN and BPN models can make a prediction within less than 5 ms of CPU time.

The mean absolute percentage error (MAPE) was used to compare the performance of the RBFN and the BPN models. It is defined as follows:

$$MAPE = (1/N') \sum_{i=1}^{N'} [ | \text{forecast load} - \text{actual load} | \times 100 ] / \text{actual load}$$

where  $N'$  is the number of patterns in the data set used to evaluate the forecasting capability of the model. Table 1

**Table 1: MAPE for forecasting of peak and total load for 1986**

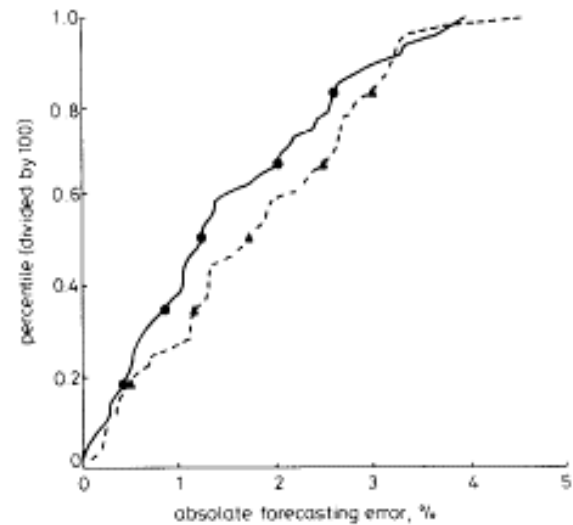
Day	RBFN		BPN	
	Peak	Total	Peak	Total
Monday	1.82	1.54	2.44	1.79
Tuesday	2.22	1.71	2.53	1.91
Wednesday	2.58	2.08	2.53	1.99
Thursday	2.18	1.50	2.11	2.10
Friday	1.89	1.56	3.78	1.71
Saturday	1.96	2.17	3.94	2.21
Sunday	2.94	2.01	3.56	2.11
All day types	2.23	1.58	2.98	1.97

contains the detailed results produced by the two ANN models (RBFN and BPN) for the seven day types, beginning with Monday. The first and second column for each model represent the MAPE for the peak load and the total load forecasts, respectively.

As can be seen from Table 1, the RBFN model almost consistently outperforms the BPN model for most of the day types for both peak and total load. For the Friday day type, for example, the MAPE peak load error for the whole year of 1986 was reduced from 3.78 to 1.89. The cumulative absolute percentage error distribution curve for the total load forecasting of the Thursday day type is given in Fig. 3. This curve also illustrates that the RBFN model performs better than the BPN model.

As previously mentioned, the RBFN model can also provide, at no additional computational cost, reliability measures such as an extrapolation index and a confidence interval for the forecast. Table 2 illustrates the peak load forecast, the actual peak load, and the confidence interval for the forecasts for February 1986. For example, the forecast for Day 1 is 11 263 MW with a lower confidence limit of, 11263.31 -450.94 = 10812.37 MW and an upper confidence limit of 11263.31 + 450.94 = 11714.25 MW, while the actual peak load is 11178 MW, within the confidence interval. Using this type of calculation, it can be seen from Table 2

that the confidence intervals consistently contained the actual peak loads. While some of the confidence intervals may be considered to be wide, the availability of this



**Fig. 3** Cumulative absolute percentage error distribution for forecasting total load for Thursdays in 1986

--▲-- BPN  
—●— RBFN

**Table 2: Actual and RBFN peak load predictions, with half-width of 90% confidence interval (CI) for February 1986**

Day	Actual load	Predicted load	Half-width of CI
	(MW)	(MW)	(MW)
1	11178	11263.31	450.94
2	10695	10153.58	581.13
3	12097	11758.06	616.79
4	12161	11893.97	613.09
5	12210	11841.14	529.44
6	12260	12467.07	547.72
7	12215	12125.24	484.14
8	10427	10359.47	499.95
9	11405	11287.24	540.82
10	12488	12436.33	630.02
11	12527	12324.79	619.98
12	12383	12072.38	532.26
13	12315	12239.48	546.19
14	11303	11426.42	504.26
15	10545	10268.72	459.43
16	10386	10529.78	496.02
17	11835	11810.62	620.54
18	12054	12124.41	618.38
19	12257	12142.08	522.95
20	12264	12573.45	547.91
21	11366	11548.37	487.24
22	10487	10448.10	479.71
23	9986	10328.06	513.89
24	11840	11734.05	616.74
25	11960	11657.73	609.30
26	12229	11844.89	522.95
27	12472	12573.53	545.95
28	11360	11353.72	498.19

information significantly differentiates the RBFN model from all other neural-network-based load forecasting models. The confidence intervals could play an important role in providing information to the operations analysts and dispatchers for decisions such as scheduling, maintenance and dispatching.

During the training, it was also possible to compute the extrapolation index,  $r(x)$ , for the training set. To interpret  $r(x)$  in terms of an adequate amount of training data, it is necessary to set a threshold value. As suggested in Reference 10, the threshold for  $r(x)$  was set to be

equal to the minimum value of  $r(x)$  over the training set. The  $r(x)$  values for the training data set were scaled between 0 and 1 for easy interpretation as  $r_{\text{norm}} = [r(x) - r_{\text{min}}] / [r_{\text{max}} - r_{\text{min}}]$ . Here  $r_{\text{max}}$  and  $r_{\text{min}}$  are the maximum and minimum values of the  $r(x)$  over the training set, respectively. When there is insufficient training data in the vicinity of the test points, the model is extrapolating and the forecasts are considered to be unreliable.

To investigate the usefulness of this index, a testing data set was generated in which the actual temperatures for some of the days in the testing year (1986) were changed by 20°F. This modified data set was prepared by subtracting 20° from the actual temperatures on the Tuesday day type for the month of December and, by adding 200 to the actual temperatures on the Tuesday day type for July 1986. Table 3 shows the extrapolation

**Table 3: Extrapolation index  $r_{\text{norm}}$  for actual temperature and corrupted temperature**

Day	$r_{\text{norm}}$ for actual temp.	$r_{\text{norm}}$ for corrupted temp.
1st Tuesday in December	0.62728	-0.00865
2nd Tuesday in December	0.54351	-0.01419
3rd Tuesday in December	0.68739	-0.00208
4th Tuesday in December	0.72128	-0.02018
5th Tuesday in December	0.39072	-0.00959
1st Tuesday in July	0.18032	-0.02348
2nd Tuesday in July	0.23677	-0.02176
3rd Tuesday in July	0.15112	-0.08387
4th Tuesday in July	0.08387	-0.02370
5th Tuesday in July	0.27496	-0.02257

index  $P_{\text{norm}}$  for the correct temperature measurements and the corrupted temperature measurements using the peak load model.

As can be seen from Table 3, the RBFN network consistently indicates whether or not the model is extrapolating from the training data. In this case ( $P_{\text{norm}} < 0$ ) the model has not learned what output is associated with a given input. This information is very useful to the SLF users who often need an indication as to whether or not the model will be able to make a reliable prediction.

#### 4 Conclusions

The system load forecasting model is a critically important decision-support tool for operating the electric power system securely and economically. The performance of many scheduling and network functions depends, to a large extent, on the accuracy of the load forecasts. In this paper, a radial basis function network model and a back-propagation model were developed to provide peak and total load forecasts for the next day. Both models have been tested using a year of real data from the PG&E power system. The results strongly indicated that the RBFN model performs better than the BPN model. The RBFN model can also compute reliability measures which is an added advantage of the RBFN model. These measures provide confidence intervals for the forecasts and an extrapolation index to determine when the model is extrapolating beyond its original training data. The reliability measures are implemented as additional output nodes at no additional computational cost by using information available from the fitting of the target function. These reliability measures are very useful to the operators and provide a reasonable solution to an unmet need in the industry. Furthermore, the training time for

*IEE Proc-Gener. Transm. Distrib., Vol. 142, No. 1, January 1995*

the RBFN model is much less than that for the BPN model.

#### 5 References

- GROSS, G., and GALIANA, F.D.: 'Short-term load forecasting', *Proc. IEEE*, 1987, pp. 1558-1573
- GUPTA, P.C.: 'A stochastic approach to peak demand forecasting in electric utility systems', *IEEE Trans.*, 1971, **PAS-90**, (2), pp. 824-832
- CHRISTAANSE, W.R.: 'Short-term load forecasting using general exponential smoothing', *IEEE Trans.*, 1971, **PAS-90**, (2), pp. 900-911
- HUBELE, N., and CHENG, C.: 'Identification of seasonal short-term load forecasting models using statistical decision functions', *IEEE Trans.*, 1990, **PWRS-5**, (1), pp. 40-45
- PAPALEXOPOULOS, AD., and HESTERBERG, T.C.: 'A regression based approach to short-term load forecasting', *IEEE Trans.*, 1990, **PWRS-5**, (4), pp. 1535-1547
- PARK, D., EL-SHARKAWI, M.E., MARKS, R., ATLAS, A., and DAMBORG, M.: 'Electric load forecasting using an artificial neural network', *IEEE Trans.*, 1991, **PWRS-6**, (2), pp. 442-449
- PAPALEXOPOULOS, A.D., HAO, SHANGYOU, and PENG, TM.: 'Short-term system load forecasting using an artificial neural network'. 2nd International Forum on Applications of Neural Networks to Power Systems, Yokohama, Japan, 1993, pp. 239-244
- PENG, TM., HUBELE, N.F., and KARADY, G.G.: 'Advancement in the application of neural networks for short-term load forecasting', *IEEE Trans.*, 1992, **PWRS-7**, (1), pp. 250-257
- BAUMANN, T., STRASSER, H., and LANDRICHTER, H.: 'Short-term load forecasting methods in comparison: Kohonen learning, Back-propagation learning, multiple regression analysis and Kalman filters'. 11th Power System Computation Conference, 1993, 1, pp. 445-451
- LEONARD, JA., KRAMER, MA., and UNGER, L.H.: 'A neural network architecture that predicts its own reliability', *Comput. in Chem. Engng*, 1992, **16**, (9), pp. 819-835
- WEISS, SM., and KULIKOWSKI, C.A.: 'Computer systems that learn' (Morgan Kaufmann, San Mateo, 1991)
- MOODY, J., and DARKEN, C.J.: 'Fast learning in networks of locally tuned processing units', *Neural Comput.*, 1989, **1**, pp. 281-294
- DUDA, R.O., and HART, P.E.: 'Pattern analysis and scene classification' (Wiley Interscience, New York, 1973)

#### 6 Appendix

##### 6.1 Computation of RBF parameters

**6.1.1. Calculation of the RBF unit centres:** Any clustering algorithm can be used to determine the RBF unit centres. Based on the prior work of Moody and Darken [12], the 'K-means' clustering algorithm is used herein. The K-means clustering algorithm finds a set of clusters each with  $r$ -dimensional centres from the given training data. The dimension of the centres is determined by the number of input variables or nodes of the input layer. The cluster centres become the centres of the RBF units. The number of clusters,  $H$ , is a design parameter and determines the number of RBF units, i.e., nodes in the hidden layer. The K-means clustering algorithm proceeds as follows

- Initialize the centre of each cluster to a different randomly selected training pattern
- Assign each training pattern to the nearest cluster. This can be accomplished by calculating the Euclidean distances between the training patterns and the cluster centres
- When all training patterns are assigned, calculate the average position for each cluster centre. They then become new cluster centres
- Repeat steps (ii) and (iii), until the cluster centres do not change during the subsequent iterations

**6.1.2 Calculation of the RBF unit widths.** When the RBF centres have been established, the width of each

RBF unit can be calculated. The width of any RBF unit is selected as the root mean square distance to the nearest  $p$  RBF units, where  $p$  is a design parameter for the RBF network. For the unit  $i$ , it is given by eqn. 3.

$$\mathbf{s}_i = \left[ (1/p) \sum \sum (\hat{x}_{ki} - \hat{x}_{kj})^2 \right]^{1/2} \quad (3)$$

where  $\hat{x}_{ki}$  and  $\hat{x}_{kj}$  are the  $k$ th entries of the centres of the  $i$ th and  $j$ th hidden units.

This part of the algorithm generates the necessary centres and widths for the RBFs. The design parameters,  $H$  and  $p$ , determine the transformed value of the inputs in the hidden layer of Fig. 1.

**6.1.3 Computation of the weights between the hidden nodes and the output nodes:** When the centres and widths of the RBF units have been chosen, then the  $N$  training patterns or historical observations are processed through the hidden nodes (let the number of hidden nodes be  $H$ ) to generate a  $H \times N$  matrix, called 'A'. Let  $T$  be the  $M \times N$  desired output matrix for the training patterns and  $M$  be the number of output nodes. The objective is to find the weights that minimize the error between the actual output and the desired output of the network. Essentially, we are trying to minimize

$$\|T - WA\| \quad (4)$$

where  $W$  is the  $M \times H$  matrix of weights on the connections between the hidden and output nodes of the network. The selection of weights between the hidden layer and the output layer is computed by linear least-squares regression. The solution to equation 4 can be obtained using the pseudo-inverse of 'A' and is given by

$$W = TA^T (AA^T)^{-1} \quad (5)$$

## 6.2 Implementation of reliability measures

The computation details regarding the calculation of the extrapolation index and the confidence intervals are now presented.

**6.2.1 Computation of extrapolation index:** The following method is based on the well-known method of density estimation called Parzen windows [10, 13].

Define a probability density for a hidden unit as

$$\mathbf{r}_i = (n_i / N) / (p^{1/2} \mathbf{s}_i)^r \quad (6)$$

where  $N$  is the total number of training patterns and  $n_i$  is

$$n_i = \sum_{t=1}^N a_i(X_t) \quad (7)$$

That is,  $\mathbf{r}_i$  is the fraction of the number of training patterns with high activation values ( $n_i$ ) per set volume ( $(\pi^{1/2} \sigma_i)^r$ ) of a hidden unit  $i$ . While  $n_i$  may not be an integer, it is treated as such.

Depending on the location of the centres and the magnitude of the unit width, the hidden units can overlap.

The extrapolation index is a weighted average as defined in the following equation [10]

$$\mathbf{r}(X_t) = \left[ \sum_{i=1}^H a_i(X_t) \mathbf{r}_i \right] / \left[ 1 - \max_{i,r} (a_i(X_t)) + \sum_{i=1}^H a_i(X_t) \right] \quad (8)$$

where  $\max_{i,r} (a_i(X_t))$  represents the maximum activation of any hidden unit over all components of  $X_t$ .

Equation (8) is the extrapolation index. If the test pattern is very close to a hidden unit centre, its activation will be high and the corresponding extrapolation index will be high. If the test pattern is far away from all the hidden units, the corresponding extrapolation index will be low (probably below the threshold value) and will indicate a poor modeling capability for this pattern.

**6.2.2 Computation of confidence intervals:** In addition to the model's total variance, it is possible to compute the contribution of a hidden unit to the variance estimate. This is computed as the weighted average of the squared error of all the training patterns as given in the following equation

$$S_i^2 = \left[ \sum_{t=1}^N a_i(X_t) (E_t^2) \right] / (n_i - 1) \quad (9)$$

where each training pattern  $t$  contributes to the variance proportional to its activation level at that unit,  $a_i(X_t)$ .  $E_t^2$  is the squared error of the output of the optimum network for pattern  $t$  and  $N$  is the total number of training patterns. Again,  $n_i$  is defined in eqn. 7.

As was shown in Fig. 1, the outputs from the hidden nodes are added together to generate a network forecast. To produce a confidence interval on the network forecast, individual confidence intervals for each of the hidden nodes will be weighted. For example, the half-width of the 95% confidence interval for the hidden unit  $i$  is given by

$$CI_i = t_{95} (S_i) n_i^{-0.5} \quad (10)$$

where  $t_{95}$  is the critical value of the  $t$  distribution with  $(n_i - 1)$  degrees-of-freedom.

The half-width of the confidence interval for the network is then computed as a weighted average of the hidden unit half-widths. Similar to the extrapolation index, the weight is the ratio of the individual activation value of the input pattern for the hidden unit to the total activation. The half-width of the confidence interval on the network forecast for input pattern  $t$  is given by

$$CI(X_t) = \left[ \sum_{i=1}^H a_i(X_t) CI_i \right] / \left[ \sum_{i=1}^H a_i(X_t) \right] \quad (11)$$

The probability level of the confidence interval is set by the critical value used in eqn. 10.