Selecting Input Factors for Clusters of Gaussian Radial Basis Function Networks to Improve Market Clearing Price Prediction

Jau-Jia Guo, *Student Member, IEEE,* and Peter B. Luh, *Fellow, IEEE*

***Abstract*—In a deregulated power market, bidding decisions rely on good market clearing price prediction. One of the common forecasting methods is Gaussian radial basis function (GRBF) networks that approximate input–output relationships by building localized Gaussian functions (clusters). Currently, a cluster uses all the input factors. Certain input factors, however, may not be significant and should be deleted because they mislead local learning and result in poor predictions. Existing pruning methods for neural networks examine the significance of connections between neurons, and are not applicable to deleting center and standard deviation parameters in a GRBF network since those parameters bear no sense of significance of connection. In this paper, the inverses of standard deviations are found to capture a sense of connection, and based on this finding, a new training method to identify and eliminate unimportant input factors is developed. Numerical testing results from two classroom problems and from New England Market Clearing Price prediction show that the new training method leads to significantly improved prediction performance with a smaller number of network parameters.**

***Index Terms*—Market clearing price forecasting, neural networks, radial basis function networks.**

I. Introduction

In NEW England deregulated power markets, participants such as generation companies submit hourly supply offers (i.e., energy blocks and associated prices) to a nonprofit organization, Independent System Operator (ISO). ISO decides hourly market clearing prices (MCPs) based on supply offers and actual loads. Good prediction of MCPs can help a participant to make better effective bidding decisions in a competitive power market.

Among all existing forecasting methods, radial basis function (RBF) networks have been widely used [1]–[5], because they are capable of deducing hidden input–output relationships in data. For Gaussian RBF (GRBF) networks, the above capability is inherited from the property that GRBF networks decompose a function into localized Gaussian functions (clusters) through which local data features are represented [6], [7]. Such a local approximation approach has an advantage over the global approximation approach of multilayer perceptron (MLP) networks in terms of preventing local data features from fading away. However, local clusters in current GRBF networks use the same the input factors to approximate various data features. There are unimportant factors existing in some clusters. Unimportant factors in a cluster will generate unnecessary adjustable parameters which may result in the situation that a network misrepresents the underlying trend in data, and hence, has a small training error with a poor generalization capability. The above situation is the so-called over-fitting. Presently, the problem is being dealt with by controlling either the network complexity that is measured by the number of adjustable network parameters, or a network's effective complexity that aims to influence the estimate of network parameters [8]. Controlling approaches are then classified into regularization-based methods and cluster-driven methods as will be discussed.

Regularization-based methods were motivated by the observation that overfitting usually leads to an input–output mapping function with high curvature over regions where data are scattered [8]. Regularization-based methods thus impose the smoothness requirement on a mapping function by adding penalty terms related to function curvature to an error function [8]–[11]. Various forms of penalty terms such as curvature-driven smoothing or weight decay have been used [8], [11]. Another regularization-based technique that is different from using penalty terms is to add noise to input data during the training process [8]. Intuitively speaking, such a technique prevents a network from fitting an individual data point precisely to avoid overfitting. The strict mathematical proof can be shown by applying Taylor expansion to an error function [8], [12]. Parallel to regularization-based methods, cluster-driven methods control network complexity by sequentially adding or removing clusters. A sequential network growth algorithm provides a systematic way to add clusters responding to new data features [13]. The convergence rate of algorithm is enhanced by applying the extended Kalman filter (EKF) algorithm to adjust network parameters [14]. Further generalization improvement is obtained as strategies of cluster pruning are combined with network growth criteria [15].

The above methods indirectly deal with the over-fitting problem that is caused by irrelevant or insignificant input factors in clusters. The direct way to overcome the problem is to identify and eliminate insignificant input factors in each cluster. For existing neural network pruning algorithms such as *optimal brain damage* (OBD) and *optimal brain surgeon* (OBS), they examine the significance of connections (weights) between neurons [8], [11]. For a connection, the OBS approximately evaluates an error increase provided that such a connection is

Manuscript received October 7, 2002. This work was supported in part by National Science Foundation under Grant ECS-9726577 and in part by Select Energy, Inc. of the Northeast Utilities System.

The authors are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06279 USA (e-mail: luh@engr.uconn.edu).

Digital Object Identifier 10.1109/TPWRS.2003.811012

severed. A diagonal Hessian matrix is assumed in the error evaluation. The connections with error increases less than a subjectively selected threshold are then cut off. In contrast, the OBD uses a nondiagonal Hessian matrix to estimate an error increase. The adjustment to the remaining weights in order to minimize the error increase is also provided. However, the above approaches are not applicable to deleting center and standard deviation parameters related to input factors in clusters because those parameters have no sense of connection strength. The criterion to check whether a parameter has a sense of connection depends on whether the associated link can be disconnected if the parameter is set to zero. Center and standard deviation parameters do not meet the criterion. Up to date, few papers have addressed the pruning techniques at the cluster level. In this paper, the inverses of standard deviations are being used instead to identify insignificant input factors through the following rationale. The mean of a univariate Gaussian function can be viewed as a parameter shifting a univariate variable in the exponent, and the inverse of a standard deviation viewed as a coefficient weighting the shifted variable. The value of this coefficient being too small implies insignificance of the shifted variable. Conversely, the value of the coefficient being too large leads to the value of such a Gaussian function being close to zero, and thus, the Gaussian function becomes insignificant. Therefore, two kinds of insignificant input factors are identified through using the inverses of standard deviations.

The rest of paper is organized as follows. The fundamental of GRBF networks is briefly given in Section II. The detail of using the inverses of standard deviations, and a new two-step training method to preserve significant input factors in each cluster are presented in Section III. Testing results in Section IV show that the presented method leads to a GRBF network with a significantly improved prediction performance and a smaller number of network parameters.

II. Gaussian Radial Basis Function Networks

A standard Gaussian radial basis function network consists of three layers of neurons as depicted in Fig. 1. The network in Fig. 1 has N_x input factors, $N\phi$ clusters, and N_z output neurons. The detailed description of the network structure can be found in many references, e.g., [8] and [11]. Given an input vector x, the output ϕ_{n_ϕ} of the n_ϕth cluster is

$$\phi_{n_\phi} = e^{-(1/2)\left(x-c_{n_\phi}\right)^T \Sigma_{n_\phi}^{-1}\left(x-c_{n_\phi}\right)}, \quad n_\phi = 1, \ldots, N_\phi \quad (1)$$

where c_{n_ϕ} is a mean (center) vector, and $\sum_{n_\phi}$ is an $N_x \times N_x$ positive definite, symmetric covariance matrix determining the receptive area of the cluster. The matrix $\sum_{n_\phi}$ is selected to be a diagonal matrix in the paper because conventional data reduction techniques can be applied to provide a set of uncorrelated input factors to neural networks. As $\sum_{n_\phi}$ takes the form of $\operatorname{diag}\left(\sigma_{n_\phi 1}^2 \ \ldots \ \sigma_{n_\phi nx}^2 \ \ldots \sigma_{n_\phi N_x}^2\right)$ where $\sigma_{n_\phi n_x}$ as a standard deviation (width) parameter is nonnegative, (1) becomes

$$\phi_{n_\phi} = e^{-(1/2)\sum_{n_x=1}^{N_x}\left[\left(x_{n_x}-c_{n_\phi n_x}\right)/\sigma_{n_\phi n_x}\right]^2}, \quad n_\phi = 1, \ldots, N_\phi, \quad (2)$$

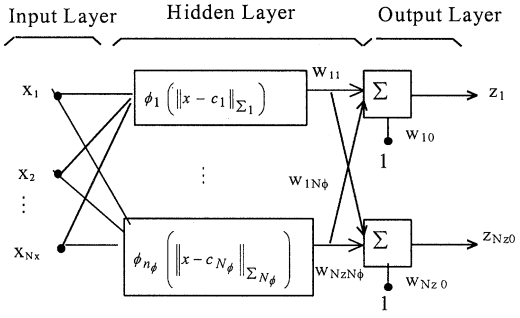

Fig. 1. Architecture of a standard GRBF network.

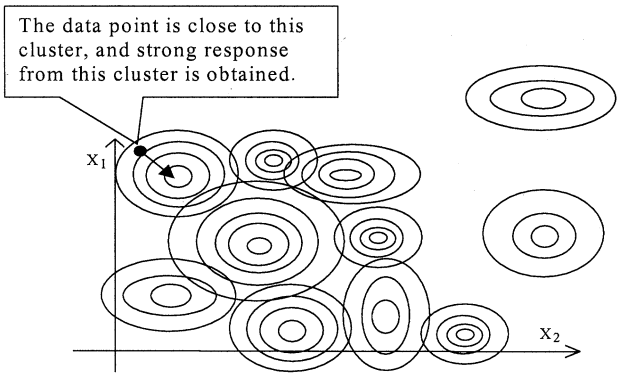

Fig. 2. Level curves of clusters representing local data features.

where x_{n_x} and $c_{n_\phi n_x}$ are components of vectors x and c_{n_ϕ}, respectively. Fig. 2 shows how clusters represent local data and respond to a new input in a two-dimensional (2-D) input space.

For the n_zth output neuron, its output is a linear combination of all cluster outputs plus a bias term, that is

$$z_{n_z} = \sum_{n_\phi=0}^{N_\phi} w_{n_z n_\phi} \phi_{n_\phi}, \quad n_z = 1, \ldots, N_z \quad (3)$$

where $w_{n_z 0}$ is a bias term with its associated ϕ_0 equal to 1, $w_{n_z n_\phi}$ $(n_\phi > 0)$ is the weight linking the n_ϕth cluster and the n_zth output neuron. In this paper, $c_{n_\phi n_x}$, $\sigma_{n_\phi n_x}$, and $w_{n_z n_\phi}$ are three types of adjustable network parameters.

The above parameters need to be initialized prior to training. Parameters $c_{n_\phi n_x}$ and $\sigma_{n_\phi n_x}$ can be initialized by a clustering algorithm such as the K-mean algorithm [8], [11]. Training is then to learn the underlying input–output relationship (mapping) from a given set of training data $D = \{(x^k, t^k) | k = 1, \ldots, K\}$ consisting of K pairs of an input x^k and the corresponding target output t^k. The sequential mode of network training is conducted by adjusting $c_{n_\phi n_x}$, $\sigma_{n_\phi n_x}$, and $w_{n_\phi n_x}$ to minimize one error function at a time and cycling through K data points, that is

$$E = \sum_{n_z=1}^{N_z} \left(z_{n_z}(x^k) - t_{n_z}^k\right)^2, \quad k = 1, \ldots, K. \quad (4)$$

The above network structure and training method are commonly used in current GRBF networks. As mentioned in the introduction, using the same the input factors in all clusters will

produce unimportant factors in some clusters. Unimportant factors generate unnecessary parameters, resulting in the overfitting situation and poor generalization for new input data. Therefore, a procedure capable of pruning unimportant input factors in each cluster is needed. The intuitive answer to the need is to use data reduction techniques such as *independent component analysis* (ICA) and *principle component analysis* (PCA). For neural networks, data reduction techniques play a role of data preprocessing to provide a set of uncorrelated input factors by transforming an original set. However, the dimensionality of this uncorrelated set less than that of the original set is not guaranteed. Furthermore, a GRBF network is a collection of local mapping that relates input data with output data through clusters. Clusters at different locations need not use the same input factors for various local mapping. (Example 1 and 3 in Section IV illustrate this feature.) Therefore, what is needed here is the technique to prune insignificant factors cluster by cluster based on local mapping after input factors are entered to networks. The ICA or PCA is to provide a set of input factors that are entered to networks, and hence, it can be a complementary tool but is not the answer to the need.

III. Analysis of a Cluster

A. Neuron Connections

The key idea of our new training method is to identify and prune insignificant input factors within each cluster. As explained in the Introduction, existing pruning methods for neural networks cannot be applied to delete center or standard deviation parameters. To have parameters resemble the significance of connections, (2) is rewritten as

$$\phi_{n_\phi} = e^{-(1/2)\sum_{n_x=1}^{N_x}\left[\sigma_{n_\phi n_x}^{-1}\left(x_{n_x}-c_{n_\phi n_x}\right)\right]^2}. \tag{5}$$

The center parameter $c_{n_\phi n_x}$ in the exponent of (5) can be viewed as a shifting parameter, and the inverse of the standard deviation parameter $\sigma_{n_\phi n_x}^{-1}$ viewed as a weighting coefficient for the connection between the n_xth input factor and the n_ϕth cluster. Within this context, it can be argued that if $\sigma_{n_\phi n_x}^{-1}$ is too small, such a connection is insignificant. Conversely, if $\sigma_{n_\phi n_x}^{-1}$ is too large, the Gaussian function $\phi_{n\phi}$ is close to zero for most x_{n_x}, rendering ϕ_{n_ϕ} nonfunctional. The above reasoning suggests that we prune input factors within cluster $n_\phi {n_x}^{-1}$ with small $\sigma_{n_\phi n_x}^{-1}$, and in the process, we want to keep $\sigma_{n_\phi n_x}^{-1}$ from being too large. The detail of the above idea will be presented in Sections III-B and C.

B. Analysis on Significance of Neuron Connections

To identify insignificant input factors in a cluster by using $\sigma_{n_\phi n_x}^{-1}$, the output of a cluster in (5) is expressed as the product of univariate exponential functions, that is

$$\begin{aligned}\phi_{n_\phi} =& e^{-(1/2)\left[\sigma_{n_\phi 1}^{-1}\left(x_1-c_{n_\phi 1}\right)\right]^2}\cdots \\ & e^{-(1/2)\left[\sigma_{n_\phi n_x}^{-1}\left(x_{n_x}-c_{n_\phi n_x}\right)\right]^2}\cdots \\ & e^{-(1/2)\left[\sigma_{n_\phi N_x}^{-1}\left(x_{N_x}-c_{n_\phi N_x}\right)\right]^2}.\end{aligned} \tag{6}$$

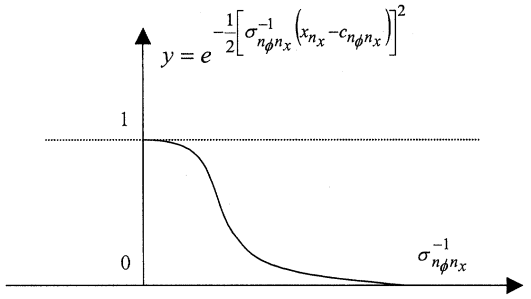

Fig. 3. Plot of y versus $\sigma_{n_\phi n_x}^{-1}$ (>0).

These univariate exponential functions correspond to different input factors but share the same form. Let us analyze one such function $y = e^{-(1/2)\left[\sigma_{n_\phi n_x}^{-1}\left(x_{n_x}-c_{n_\phi n_x}\right)\right]^2}$. To visualize how y is affected by $\sigma_{n_\phi n_x}^{-1}$, the term $\left(x_{n_x}-c_{n_\phi n_x}\right)$ in the exponent of y is first fixed. Fig. 3 shows the relationship between y and nonnegative $\sigma_{n_\phi n_x}^{-1}$.

As it can be seen from Fig. 3, when the value of $\sigma_{n_\phi n_x}^{-1}$ is very small, y is close to one; such a function can be taken out without affecting ϕ_{n_ϕ} in (6). When $\sigma_{n_\phi n_x}^{-1}$ is very large, y is close to zero, implying that ϕ_{n_ϕ} does not contribute to any network output and is nonfunctional.

From a different perspective, when $\sigma_{n_\phi n_x}^{-1}$ is very small, that is, $\sigma_{n_\phi n_x}$ is very large, it implies a wide and flat receptive field that contains no valuable information. When $\sigma_{n_\phi n_x}^{-1}$ is very large, that is, $\sigma_{n_\phi n_x}$ is very small, it suggests a sharp and narrow receptive field that cannot capture information beyond a very small neighborhood around the function's center.

C. Two-Step Training Method

As analyzed in Section III-B, the value of $\sigma_{n_\phi n_x}^{-1}$ is not desired to be too small or too large. However, to draw the line between normal values versus too small or large values is rather subjective. A new two-stage training method is developed to resolve this difficulty. The procedure first restricts $\sigma_{n_\phi n_x}^{-1}$ from being too large by having a penalty term on $\sigma_{n_\phi n_x}^{-1}$. Hypothesis testing is then performed to examine whether $\sigma_{n_\phi n_x}^{-1}$ is too small to be statistically significant.

To implement the above idea, the error function in (4) is augmented with weighted penalty terms on $\left\{\sigma_{n_\phi n_x}^{-1}\right\}$ to regulate them from being too large, that is

$$E' = \sum_{n_z=1}^{N_z}\left(z_{n_z}\left(x^k\right)-t_{n_z}^k\right)^2 + v\left(\sum_{n_\phi=1}^{N_\phi}\sum_{n_x=1}^{N_x}\left(\sigma_{n_\phi n_x}^{-1}\right)^2\right) \tag{7}$$

where the second term in (7) is the sum of squared $\sigma_{n_\phi n_x}^{-1}$ from all clusters, and ν is a weighting coefficient. As ν is zero, a neural network is completely determined by the squared error term. On the contrary, as ν goes infinite, the weighted penalty terms dominantly determine a network. Bounded by this range, the value of ν is empirically selected to reduce the training error as much as possible. By minimizing (7), the value of $\sigma_{n_\phi n_x}^{-1}$ tends to be not too large.

Second, input factors with very small values of $\sigma_{n_\phi n_x}^{-1}$ need to be identified and pruned in clusters after minimization. A

simple way is to set a threshold. If the value of $\sigma^{-1}_{n_\phi n_x}$ is less than the threshold, the associated input factor is eliminated. However, selecting such a threshold is rather subjective. Statistical hypothesis testing is therefore adopted with the following two hypotheses

H_0: the mean of $\sigma^{-1}_{n_\phi n_x}$ equals zero;

H_1: the mean of $\sigma^{-1}_{n_\phi n_x}$ does not equal zero.

The required statistics for the above hypothesis testing are the sample mean and sample variance of $\sigma^{-1}_{n_\phi n_x}$ that are obtained from R runs of Monte Carlo simulation by perturbing input or output of the training data set D. If R is not large and the sample variance is used in the testing, then the sample mean of $\sigma^{-1}_{n_\phi n_x}$ has a student's t distribution [16].

For each simulation run, a zero-mean Gaussian noise is added to the input or output of the training data set D. Through minimizing E' in (7), the estimate of $\left\{\sigma^{-1}_{n_\phi n_x}\right\}$ can be obtained. The sample mean $\overline{\sigma^{-1}_{n_\phi n_x}}$ and sample variance $S_{\sigma^{-1}_{n_\phi n_x}}$ of $\sigma^{-1}_{n_\phi n_x}$ for R Monte Carlo runs are obtained by

$$\overline{\sigma^{-1}_{n_\phi n_x}} = \frac{1}{R}\sum_{r=1}^{R}\sigma^{-1}_{n_\phi n_x}(r) \tag{8}$$

and

$$S_{\sigma^{-1}_{n_\phi n_x}} = \frac{1}{(R-1)}\sum_{r=1}^{R}\left(\sigma^{-1}_{n_\phi n_x}(r) - \overline{\sigma^{-1}_{n_\phi n_x}}\right)^2. \tag{9}$$

With the above statistics, hypothesis H_0 is accepted with probability $(1-\alpha)$ if and only if the following holds:

$$t = \frac{\overline{\sigma^{-1}_{n_\phi n_x}}}{\sqrt{\frac{1}{R}S_{\sigma^{-1}_{n_\phi n_x}}}} \le t_\alpha \tag{10}$$

where t_α is the value that the probability of t not greater than t_α is $(1-\alpha)$ in a student's t distribution with $(R-1)$ degrees of freedom [16]. The acceptance of H_0 implies that the input factor associated with this $\sigma^{-1}_{n_\phi n_x}$ is statistically insignificant; otherwise, the input factor is statistically significant. Once an input factor in a cluster is considered insignificant, it is eliminated from the cluster by setting the value of the associated $\sigma^{-1}_{n_\phi n_x}$ to zero.

As mentioned previously, few papers have addressed the pruning techniques at the cluster level. In [17], input factors of clusters are examined and pruned one by one with the criterion that the mean square error is not increased. Two distinct differences between the techniques in [17] and this paper are as follows. First, the technique in [17] performs exhaustive search for the input factors to be pruned. The technique in this paper uses the result of hypotheses testing with statistical meanings. Second, clusters with common variances are used in [17] such that the remaining input factors in clusters are treated equally. In contrast, clusters with covariance matrices are used in this paper. Because of the reciprocals of standard deviations, the remaining input factors are treated differently to reflect various effects on a cluster output.

IV. Results and Insights

In this section, three examples tested on a computer with an AMD-850-MHz Duron processor are presented to show the key features of a GRBF network that uses the new training method. The first example is a function-learning problem to demonstrate the prediction improvement from the elimination of insignificant input factors in clusters. The second example shows GRBF networks' capability of preserving local data features. The third example is a practical application of predicting average on-peak-hour MCPs for New England power markets. In three examples, the number of Monte Carlo simulation runs is 10 with target output perturbed by a zero-mean Gaussian noise, and the t_α corresponding to 95% confidence from the table of a student's t distribution is about 2.27. The data were all normalized to be in $[0,1]$.

TABLE I
Results of Two GRBF Networks

	A: Actual function y		B: Improved GRBF		C: Conventional GRBF	
	f_1	f_2	f_1	f_2	f_1	f_2
C_1*	4	12	3.92	12.1	4.09	11.93
C_2	4	NA	3.96	NA	4.01	4.19
C_3	NA	NA	NA	NA	6.59	6.12
σ_1*	2	3	2.09	3.11	1.89	2.64
σ_2	5	NA	5.2	∞	4.74	14.63
σ_3	NA	NA	∞	∞	58.92	24.27
			Improved GRBF		Conventional GRBF	
Training	MAE		0.014		0.013	
	MAPE		0.70%		0.69%	
	Time		113 seconds		27 seconds	
Prediction	MAE		0.017		0.027	
	MAPE		0.72%		0.81%	

* Symbols c_i and σ_i represent the center and standard deviation of a Gaussian function, respectively.

Example 1: The improved and conventional GRBF networks were applied to learn a nonlinear function comprised of a linear combination of two Gaussian functions plus a constant term

$$y = 2f_1 + f_2 + 1$$

where

$$f_1 = e^{-(1/2)\left[((x_1-4)/2)^2+((x_2-4)/5)^2\right]}$$

and

$$f_2 = e^{-(1/2)[(x_1-12)/3]^2}.$$

Network input factors are x_1, x_2, and x_3. An irrelevant factor x_3 that is not used at all in f_1 and f_2 serves as an insignificant factor. There are 500 data points uniformly sampled in $[0,16]$ for x_1, $[0,8]$ for x_2, and $[0,12]$ for x_3. Out of these 500 data points, 416 entries were uniformly picked to form a training set. The remaining 84 entries constitute a prediction set. Since the entries in training and prediction sets were uniformly sampled and picked, network training and prediction errors are expected to be similar if a network does not overfit the training data.

To underline the pruning effect, both networks use two clusters. The results are summarized in Table I. The actual centers and variances of Gaussian functions f_1 and f_2 are in the column A. The column B lists the corresponding values of the improved

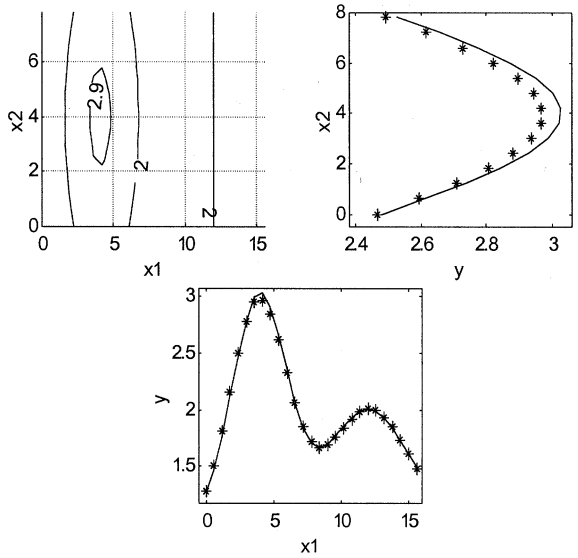

Fig. 4. (a) shows a 2-D contour plot of 2-chustered y. (b) and (c) show 1-D function plots of y (the solid curve) and the improved GRBF network (stars) along x_1 while x_2 is at 4 and along x_2 while x_1 is at 4, respectively.

GRBF network, and the column C contains the result of the conventional GRBF network. From the table, it is clear that the improved network eliminates x_3 from function f_1, and x_2 as well as x_3 from f_2 because the values of associated σ's are infinite. Furthermore, the mean absolute error (MAE) and mean absolute percentage error (MAPE) of prediction for the improved network are close to the training MAE and MAPE, respectively. However, for the conventional network, its prediction MAE is about twice as large as its training MAE, indicating the occurrence of overfitting training data. As for training times, the conventional network requires 27 s while the improved one needs 113 s.

Fig. 4 shows function y and the learning result of the improved GRBF network. The contour of y is plotted in (a) to illustrate two clusters located at $(x_1, x_2) = (4, 4)$ and $x_1 = 12$. Figures (b) and (c) are 1-D function plots along x_1 and x_2. As it can be seen from (b) and (c), the learning result of the improved network is matched to y except the region around $(x_1, x_2) = (4, 4)$.

Example 2: The training and prediction sets of Example 1 were used in this example. Ten data points that were uniformly sampled in $[19, 21]$, for x_1, [7], [8] for x_2, and $[0, 12]$ for x_3 to represent a new data feature are around the center of a new function

$$f_3 = e^{-(1/2)(x_1-20)^2}.$$

and those points are far from any entry in the above training and prediction sets. To show the capability of the improved GRBF network in preventing local data features from fading away, five points out of ten new data points were added to the beginning of the training set, and the remaining five points were added to the end of the prediction set. Thus, a new nonlinear function to be learned is

$$y' = y + f_3$$

TABLE II
RESULTS OF IMPROVED GRBF AND MLP NETWORKS

		A: Improved GRBF	B: MLP
Training	MAE	0.014	0.028
	MAPE	0.71%	1.42%
	Time	115 seconds	161 seconds
Prediction	MAE	0.02	0.036
	MAPE	0.97%	1.87%

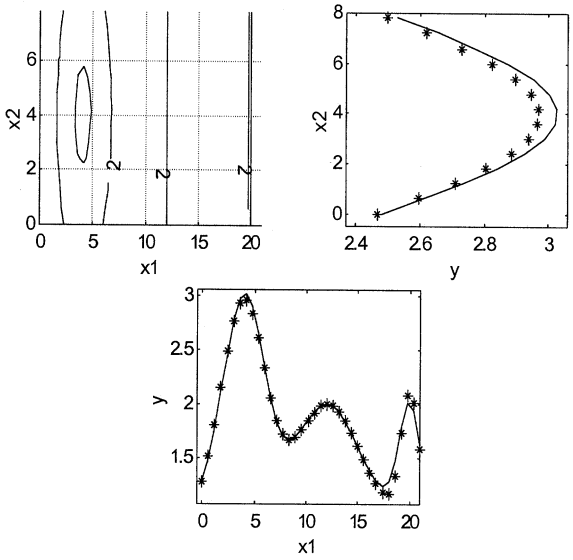

Fig. 5. (a) shows a 2-D contour plot of 3-clustered y'. (b) and (c) show 1-D function plots of y' (the solid curve) and the improved GRBF network (stars) along x_1 while x_2 is at 4 and along x_2 while x_1 is at 4, respectively.

The improved GRBF network and an MLP network ([8] and [11]) were applied to learn this three-clustered function.

The summary of results is tabulated in Table II. The column B shows that the MAE and MAPE of the MLP network with nine hidden neurons are 0.028 and 1.42%, respectively. In addition, the training time is 161 seconds. Actually, the MLP network can reach 0.86% of the training MAPE without new data points. From the testing cases where the number of hidden neurons varies, one common thing found in those cases is that MLP networks have a difficulty in learning the characteristic of scarce new data and a slow convergence. This is mainly because MLP networks are global approximators, and tend to capture the relationship that the majority of data contain.

The prediction MAE and MAPE of the improved GRBF network with three clusters are 0.02 and 0.97% in the column A. Compared to the MAE and MAPE of the improved GRBF network in Example 1, those of the improved GRBF network are larger. The reason can be found in Fig. 5 that has a contour plot of y', and 1-D function plots of y' and the learning result of the improved GRBF network. From (b) and (c), they clearly show that learning in the regions around $(x_1, x_2) = (4, 4)$ and $x_1 = 20$ is not perfect and hence leads to larger prediction errors for data points in those regions. The region $x_1 = 20$ is where scarce new data points are located.

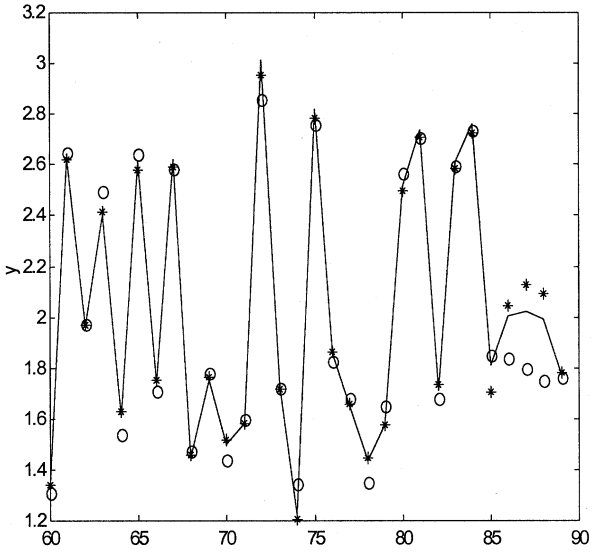

Fig. 6. Plot of the last 30 predictions of MLP network (circles) and improved GRBF network (stars) versus function y' (the solid curve).

Fig. 6 is a graph of predictions of both networks, and two things are clearly shown. First, the improved GRBF has larger errors on the last several predictions that are for new data points. As explained earlier, the reason is that the improved network did learn but not perfectly learn the relationship of new data points. Second, the MLP network has a difficulty in predicting for scarce new data points, which is shown from the fact that its predictions are not matched to the curve of y'.

Checking training errors of individual data points indicates that the MLP network did not learn at all for scarce new data points during training. Compared to the MLP network, the improved GRBF network obviously tries to preserve the feature of scarce data points by one of clusters even after a long period of updating and prediction.

Example 3: The MLP, conventional GRBF, and improved GRBF networks were applied to predict average on-peak-hour MCP's (i.e., the average of MCP's from hour eight to hour 23) for New England power markets. Forecasting average on-peak-hour MCPs is critical because power energy for daily on-peak hours often is transacted in the form of 16-h energy blocks. The training period is from May 1, 1999 to June 30, 2000 and the prediction period from July 1, 2000 to June 30, 2001. According to the best prediction results obtained, the conventional GRBF network uses 23 input factors and six clusters, and the MLP network uses 45 input factors and eight hidden neurons [18]. To demonstrate the effectiveness of the new training method, the improved GRBF network also uses six clusters and the same 23 input factors that the conventional network uses. Therefore, tunable parameters are 23 pairs of center and width parameters in each cluster, six weighting coefficients connecting six clusters to an output neuron, and one bias term. The network training is stopped as either of the following criteria is met. One criterion is that the given number of iterations is reached, and the other is that the decrease of training MAPE is less than 0.01%.

The results are summarized in three tables. Table III has the number of parameters used in each network, and the input factors used in improved GRBF clusters before and after pruning are listed in Table IV. The prediction performance of networks is in Table V.

TABLE III
COMPARISON OF THE NUMBER OF PARAMETERS

	Before training	After training
Improved GRBF	283	243
Conventional GRBF	283	283
MLP	377	377

TABLE IV
COMPARISON OF INPUT FACTORS BEFORE AND AFTER PRUNING

	Cluster index						
	Before training	After training					
Data type	1~6	1	2	3	4	5	6
Week day	t		t	t	t	t	t
Average L	t, t-2,t-7	t	t	t	t-2,	t, t-2	
Max L	t, t-2,t-7	t	t	t		t-2	
Min L	t, t-2,t-7	t	t	t		t	
S	t, t-1,t-7	t, t-1, t-7	t, t-1, t-7	t,t-1, t-7	t-7	t, t-1, t-7	
G	t, t-1,t-7	t, t-1, t-7	t, t-1, t-7		t,t-1, t-7		t, t-1, t-7
O	t, t-1,t-7	t-1	t, t-1, t-7	t, t-7	t, t-1, t-7	t-7	t, t-1, t-7
Max T1	t			t	t		
Min T1	t				t		
Max T2	t		t	t	t		
Min T2	t		t		t		

* L: on-peak-hour load; S: Surplus; G: Gas; O: Oil

* T1: Temperature at Boston; T2: Temperature at Hartford

Table III shows that the improved network uses only 243 parameters after training. In contrast, the conventional network uses 283 parameters. The number of network parameters is reduced by more than 14%. For the MLP network, its number of parameters is 55% more than that of the improved network. Table IV records the input factors used in the improved GRBF clusters before and after pruning. For instance, cluster 1 after pruning has no following input factors: week day index, temperatures, 2 and 7-day-lagged load (i.e., $t-2$ and $t-7$), and current as well as 7-day-lagged oil prices (i.e., t and $t-7$). Obviously, clusters use different combinations of input factors, and none of them use the same set of input factors.

The benefit of elimination of unimportant input factors can be seen from Table V. The prediction summary of the MLP network is in the column A with 12.2% of the overall MAPE, and the results of the conventional and improved GRBF networks are in the column B with 17.5% and in the column C with 11.9% of the overall MAPE, respectively. The overall MAPE of the improved network is better than that of the conventional network by 32% in terms of percentages. Even compared to the MLP network with a high number of parameters, the overall predic-

TABLE V
PREDICTION PERFORMANCE OF NEURAL NETWORKS

	A: MLP		B: Conventional GRBF		C: Improved GRBF		D: GRBF with an added cluster	
Month	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
Jul-00	4.79	10.4%	3.97	8.7%	3.50	8.0%	3.50	8.0%
Aug-00	7.15	14.1%	8.03	15.5%	5.12	10.3%	5.12	10.3%
Sep-00	5.63	10.9%	5.80	11.0%	4.47	8.6%	4.47	8.6%
Oct-00	4.46	7.9%	7.99	13.8%	5.20	9.1%	5.20	9.1%
Nov-00	3.94	7.4%	3.82	6.9%	3.60	6.8%	3.60	6.8%
Dec-00	8.90	13.2%	14.57	20.5%	7.24	10.7%	7.24	10.7%
Jan-01	7.49	11.9%	9.50	13.6%	7.03	11.0%	7.03	11.0%
Feb-01	4.75	10.9%	11.69	26.8%	7.18	16.3%	5.04	11.4%
Mar-01	8.30	13.8%	8.88	16.1%	7.56	12.2%	7.58	12.4%
Apr-01	5.36	13.9%	9.20	24.7%	5.73	14.6%	6.16	15.9%
May-01	8.61	18.6%	11.58	27.7%	8.68	19.6%	8.99	20.3%
Jun-01	5.31	13.5%	9.11	24.9%	6.00	16.0%	4.73	12.5%
Overall	6.25	12.2%	8.67	17.5%	5.94	11.9%	5.72	11.4%

tion performance of the improved network is still better than that of the MLP network. The only expense for the improved GRBF network is that it requires more training time than the conventional one due to performing the Monte Carlo simulation. Thanks to the advance of CPU speeds, the training time of the improved network takes only about 5 min. Once network training is finished, the improved network makes predictions as quickly as the conventional one. Except the training time and the use of hypotheses testing, there is not much difference between the conventional and improved GRBF networks in terms of the algorithm complexity, since one uses the standard deviations and the other uses the inverses of them.

One more thing is worth being noted. From Table V, there are five months that the MLP network outperforms the improved GRBF network. Out of these five months, the only ones that the MAPE difference between the MLP and improved networks is more than 2% are February 2001 with 5.45%, and June 2001 with 2.52%. The reason why the improved GRBF network has higher errors in both months is that existing six clusters do not properly capture the characteristic of data points in the first half of February and part of data points in June. Consequently, the network overshot predictions. Therefore, a new cluster needs to be added to the improved network. The following is how a new cluster was added to the improved GRBF network in this example. Triggered by the event that larger prediction errors occurred on consecutive days around the beginning of February, a GRBF network with only one cluster was trained based on the data points of those days. The setting of the number of consecutive days is case-dependent, and is six in this example. The newly obtained cluster was then added to the existing improved GRBF network, and the whole network was retrained based on the data up to the beginning of February. This new network continues predictions for the remaining days in February and the remaining months. Because of the new cluster, the degree of overestimates is significantly reduced. As shown from the column D of Table IV, the prediction MAPE is reduced from 16.3% to 11.4% in February and from 16% to 12.5% in June. Beside the improvement in February and June, the overall MAPE is also improved to be 11.4% decreased from 11.9%.

V. CONCLUSIONS

Input factors that are not important to a cluster in a GRBF network should be deleted since they will mislead local learning and result in poor generalization. In this paper, the inverses of standard deviations are found to capture a sense of connection in clusters, and a new training method based on the inverses of standard deviations to identify and eliminate unimportant input factors is developed. Numerical testing shows that the method prunes unimportant input factors, and hence significantly improves prediction performance.

ACKNOWLEDGMENT

The authors would like to thank Mr. L. Zhang for providing testing results using an MLP network.

REFERENCES

[1] S. A. Billings and X. Hong, "Dual-orthogonal radial basis function networks for nonlinear time series prediction," *Neural Networks*, vol. 11, no. 3, pp. 479–493, Apr. 1998.
[2] N. E. Longinow, "Predicting pilot look-angle with a radial basis function network," *IEEE Trans. Syst., Man, and Cybern.*, vol. 24, pp. 1511–1518, Oct. 1998.
[3] R. W. Anderson, S. Das, and E. L. Keller, "Estimation of spatiotemporal neural activity using radial basis function networks," *J. Comput. Neuroscience*, vol. 5, no. 4, pp. 421–441, Dec. 1998.
[4] L. A. Venta, L. M. Salchenberger, and E. R. Venta, "Improved diagnosis of breast implant rupture with sonographic findings and artificial neural networks," *Academic-Radiology*, pp. 238–244, 1998.
[5] "Clustering based RBF (Radial basis Function) neural network model for short-term freeway traffic volume forecasting," in *Proc. Fifth Int. Conf. Applicat. Advanced Technol. Transportation Eng.*, B. Park, C. T. Hendrickson, and S. G. R. Editor, Eds., Newport Beach, CA, Apr. 26–29, 1998, pp. 280–287.
[6] E. J. Hartman, J. D. Keeler, and J. M. Kawalski, "Layered neural networks with Gaussian hidden units as universal approximators," *Neural Comput.*, vol. 2, no. 2, pp. 210–215, 1990.
[7] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, 1991.
[8] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford, 1995.
[9] ——, "Improving the generalization properties of radial basis function neural networks," *Neural Comput.*, vol. 3, pp. 579–588, 1991.
[10] A. N. Tikhonov and A. V. Goncharskyy, *Ill-Posed Problems in the Natural Sciences*. Moscow, Russia: Mir Publishers, 1987.
[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[12] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural Comput.*, vol. 7, pp. 108–116, 1995.
[13] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, pp. 213–225, 1991.
[14] V. Kadirkamanathan, "A function estimation approach to sequential learning with neural networks," *Neural Comput.*, vol. 5, pp. 954–975, 1993.
[15] Y. Lu, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Comput.*, vol. 9, pp. 461–478, 1997.
[16] C. F. Lee, *Statistics for Business and Financial Economics*. MA: Lexington, 1993.
[17] C. Andrew, M. Kubat, and G. Pfurtscheller, "Trimming the inputs of EBF networks," in *Proc. 3rd Europe. Symp. Artificial Neural Networks* , Brussels, Belgium, Apr. 19–21, 1995, pp. 291–296.

[18] L. Zhang and P. B. Luh, "Confidence regions for cascaded neural network prediction in power market," in *IEEE Power Eng. Soc. Winter Meeting*, vol. 2, 2001, pp. 533–538.

Jau-Jia Guo (S'00) received the B.S. degree from the Engineering Science Department at National Cheng-Kung University, Tainan, Taiwan, R.O.C., in 1993, and the M.S. degrees in Physics Department from the University of Connecticut, Storrs, in 1997, and the M.S. degree in Electrical and Computer Engineering (ECE) Department from the University of Connecticut in 2000. He is currently pursuing the Ph.D. degree in the ECE Department at the University of Connecticut.

Peter B. Luh (M'80–SM'91–F'95) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1973, the M.S. degree in Aeronautics and Astronautics Engineering from Massachusetts Institute of Technology (MIT), Cambridge, in 1977, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, in 1980.

Since 1980, he has been with the University of Connecticut, and now is a SNET Endowed Professor of Communications and Information Technologies with the Department of Electrical and Computer Engineering, and the Director of the Booth Research Center for Computer Applications. His major research interests include schedule generation and reconfiguration for manufacturing and power systems.

Dr. Luh is the Editor-in-Chief of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION. He also was an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.