Self-Configuring Radial Basis Function Neural Networks for Chemical Pattern Recognition

Chuanhao Wan and Peter de B. Harrington*

Clippinger Laboratories, Ohio University Center for Intelligent Chemical Instrumentation, Ohio University, Athens, Ohio 45701-2979

Received April 9, 1999

Construction of radial basis function neural networks (RBFN) involves selection of radial basis function centroid, radius (width or scale), and number of radial basis function (RBF) units in the hidden layer. The *K*-means clustering algorithm is frequently used for selection of centroids and radii. However, with the *K*-means clustering algorithm, the number of RBF units is usually arbitrarily selected, which may lead to suboptimal performance of the neural network model. Besides, class membership and the related probability distribution are not considered. Linear averaging (L-A) was devised for selection of centroids and radii for the RBFs and computing the number of RBF units. The proposed method considers the class membership and localized probability density distribution of each class in the training sets. The parameters related to the network construction were investigated. The network was trained with the QuickProp algorithm (QP) or Singular Value Decomposition (SVD) algorithm and evaluated with the poly(chlorobiphenyl) (PCB) mass spectra and an Italian olive oil reference data set. The prediction accuracy of PCB data sets was better than 94%, and the prediction accuracy with Italian olive oil data sets achieved 100% with RMSEP as low as 2.4×10^{-3}. The training times were usually about a second on a personal computer. The performance of neural networks constructed from the linear-averaging method was observed to be better than that with the *K*-means algorithm with these data sets.

1. INTRODUCTION

Radial basis function neural networks (RBFNs)[1] have found increased applications across many disciplines. They have been used for image classification,[2] character recognition,[3] speech recognition,[4] and robot controls.[5] In analytical chemistry, they have been used for quantitative analysis of chemical measurements[6,7] and chemical system modeling.[8,9] The most common use of RBFNs is in the pattern recognition field. RBFNs have been used as classifiers for "electronic nose" detection of volatile organic compounds[10] and for complex mass spectra, such as adulterant detection[11] and whole-organism fingerprinting.[12,13]

Despite the numerous applications of the RBFNs, the optimal construction of RBFNs is still a fertile area of research.[14−16] A RBFN is characterized by a number of RBF units in the hidden layer, and each radial basis function unit has a characteristic centroid and radius. The critical steps involved in the construction of RBFNs are the selection of the number of radial basis functions in the hidden layer and the selection of the centroid and radius for each RBF unit. In some applications, each training object is used as the centroid of a RBF unit. The resultant network is a nongeneralized network in which the number of hidden units equals the number of training objects. This method is obviously not practical for some applications in which large data sets are used. Generalized radial basis function neural networks (GRBFN)[17] have much fewer RBF units in the hidden layer than training objects. One way to construct a GRBFN is to randomly sample a small portion of training objects as the radial basis centroids; however, the constructed network may not be stable or may not train at all. Other techniques, such as clustering,[18] integral wavelet transform,[19] and genetic algorithm,[20,21] have been investigated for constructing compact networks with more or less improvement in performance.

The standard *K*-means clustering[22,23] is commonly used to select the centroid and the radius for each RBF unit. However, no information about the number of RBF units can be obtained with this technique, and it is usually selected arbitrarily. Sutanto et al.[24] have proposed a mean-tracking clustering algorithm, in which a moving window is used to search the centroid of the RBF unit. This method has invoked many parameters (e.g. window sizes, initialization location, search directions) that are difficult to control when used with new data sets. In addition, their algorithm did not take the grouping of training objects into consideration, rather another algorithm has to be employed to group the training objects, which increases the complexity of net construction. Their work also did not consider the class membership of training objects; thus the probability distribution of different classes can overlap with one another. Bruzzone and Prieto[25] have realized that the class membership of training spectra might help to achieve better selection of the centroid and radius. However, with their algorithm, the number of clusters still had to be specified in advance as they have explicitly stated that, for the classes, the number of centroids is chosen according to both the number of training samples and the dimension of the input space. It can be seen that the selection of the number of clusters is in effect still quite arbitrary.

* Corresponding author. Fax: (740) 593-0148. Phone: (740) 593-2099. E-mail: harring@helios.phy.ohiou.edu.

10.1021/ci990306t CCC: $18.00
Published on Web 10/05/1999

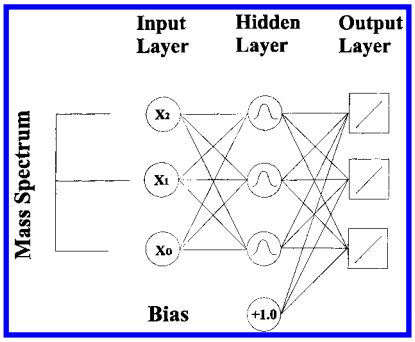

Figure 1. Architecture of radial basis function neural networks (RBFN).

On the basis of the work that has been pioneered by the aforementioned authors, a new clustering method for the selection of the number of RBF units, the centroid, and the radius has been devised. With the proposed clustering algorithms, generalized RBFN can be automatically constructed on the basis of the information from the training data without external user input.

2. RADIAL BASIS FUNCTION NEURAL NETWORKS

The most commonly used artificial neural network (ANN) models are radial basis function neural networks (RBFNs) and multilayer feed-forward neural networks (MLFNs).[26−29] For the MLFNs, such as the back-propagation neural networks (BNNs), the input to the forward layer is the linear summation of adjacent layers and usually has more than one hidden layer. In contrast to MLFN, radial basis function neural networks (RBFNs) have used a different projection approach and usually has only one hidden layer for nonlinear transformation, as shown by Figure 1.

The hidden layer of RBFN consists of a number of RBF units (n_h) and bias (b) unit. Each RBF unit employs a radial basis function as a nonlinear transfer function to operate on the input data, which are usually the extracted features or the full spectrum intensities. A radial basis function has a bell-shaped response surface, which is characterized by a centroid location ($\mathbf{c}_i$) and radius (r_j). Rather than using the weighted summation, the RBFN functions by measuring the Euclidean distance between input vector ($\mathbf{x}$) and the basis function centroid ($\mathbf{c}_j$), i.e., $||\mathbf{x}-\mathbf{c}_j||$, and performs the nonlinear transformation with radial basis function in the hidden layer as given in eq 1.

$$h_j(x) = \mathrm{e}^{-||x-c_j||^2/2r_j^2} \tag{1}$$

In which a Gaussian radial basis function is assumed, h_j is the notation for the output of the jth RBF unit. For the jth RBF unit, $\mathbf{c}_j$ and r_j are the centroid and radius (width or scale). The operation of the output layer can be linear or sigmoidal. It has been mathematically proven that a single hidden layer of the RBFN may be sufficient to model any nonlinear relationships in the training data, so a linear operation is often used in the output layer, which is given as eq 2.

$$y_k(x) = \sum_{m=1}^{n_h} w_{km} h_m(x) + b_k \tag{2}$$

In which y_k is the kth output unit for the input vector $\mathbf{x}$, w_{km} is the weight connection between the kth output unit and the mth RBF unit and b_k is the bias.

The weight w_{km} was updated with two learning algorithms in this paper. The first one was the QuickProp algorithm,[30] which assumes that the weight error has a quadratic surface and can be adjusted according to the first derivative of the error function. For this work, the learning rate (α) was set to 0.01, and a maximum growth factor (μ) of 1.0 was used. The shrink factor was set to $1/(1+\mu)$. The second learning algorithm used was the SVD algorithm. One major advantage of SVD solution is the high speed of the singular value decomposition (SVD) of large input matrices. The parameter that affects the prediction with SVD is the number of SVD factors used. The SVD learning algorithm can be illustrated with the following notations. Considering the number of objects to be classified is *ns*, the number of RBF units is *ny*, and there are a number of *nc* classes to be classified. The output encoding for each object is a binary vector, for which each component corresponds to a class. The target class is coded by a value of unity, and the other values are zero. For each training object, there are *nh* output values from the hidden layer. The matrix of output from the hidden layer for the *ns* training objects has *nh*+1 rows (an additional row of unity values is used to model bias) and *ns* columns. For each of **X** output unit there are *nh*+1 weight connections, and the weight matrix is referred as **W**, which has *ny* rows and *nh*+1 columns. The output matrix for the training set is referred to as **Y**, which has *ny* rows and *ns* columns. The mapping between the hidden layer and the output layer can be modeled with eq 3.

$$\mathbf{Y} = \mathbf{WX} \tag{3}$$

With eq 3, the weight matrix **W** can be computed by the singular value decomposition of the **X** matrix as given by eq 4.

$$\mathbf{X} = \mathbf{USV}^{\mathrm{T}} \tag{4}$$

In which **U** and **V** span the column and row space for the **X** matrix and **S** is the singular value matrix for **X**. By considering only the singular values (e.g., the first *q* largest singular values) that exceed a proportion of the variance, the pseudoinverse can be exploited to compute the weight matrix as given by eq 5.

$$\mathbf{W} = \mathbf{YVS}^{-1}\mathbf{U}^{\mathrm{T}} \tag{5}$$

The performance was compared between networks with the QuickProp algorithm and SVD learning in terms of their prediction accuracy. The overall prediction accuracy was evaluated with the percentage of correctly classified spectra with respect to the total prediction set. The prediction accuracy for each class was evaluated with the percentage of correctly classified spectra in the total spectra of that class. The root mean squared error of calibration (RMSEC) was calculated according to eq 6. RMSEP calculation was based on the same equation, but based on test data sets rather than

$$\mathrm{RMSEC} = \left[\frac{\sum_{i=1}^{ns}\sum_{j=1}^{ny}(y_{ij}-\hat{y}_{ij})^2}{(ns)(ny)}\right]^{1/2} \tag{6}$$

training data sets. Standard deviation was usually based on five runs. Training terminated after 250 000 iterations or until the given training error (RMSEC) was achieved. The effect of the training error was evaluated from 0.3 to 0.01. For the SVD algorithm, the training is controlled by the selection of the number of SVD factors used.

The architecture of RBFN is very similar to that of probabilistic neural networks (PNN).[31] Both RBFNs and PNNs are closely related to kernel discriminant analysis. However, PNN usually requires larger memory storage space and trains slower, because during the training phase of PNN, all the training objects are copied to the hidden layer for comparison with the new pattern in the prediction step.

3. CLUSTERING ALGORITHMS FOR RBFN CONSTRUCTION

3.1. *K*-Means Clustering Algorithm. The clustering method is a popular method for selection of centroids and radii of radial basis function units. The commonly used radial basis function is a Gaussian basis function. For the Gaussian basis functions, the critical parameters are the location of its centroid ($\mathbf{c}_j$) and the magnitude of its radius (r_i); *K*-means is a standard clustering method for selection of the RBF centroid and radius, which is illustrated in the following context. Considering a set of *ns* training spectra ($\mathbf{x}_i$), with the *K*-means algorithm, the number of clusters (*K*) is decided before clustering, and then the training data space is partitioned into *K* sets $\{s_1, s_2, ..., s_K\}$. The centroids are selected to minimize the error value given by eq 7.

$$E(c) = \sum_{m=1}^{K} \sum_{x_i \in S_n} ||x_i - c_m||^2 \tag{7}$$

With the *K*-means algorithm,[22] the only factor that is considered is the distance between centroid and training objects; class membership is not used during the calculation of the centroids and radii. Two problems arise with this algorithm: one is that there is no information derived about the number of clusters, and the number of clusters is arbitrarily selected. It is advantageous that the number of clusters is computed intelligently from the information provided in the training data without introducing external input. Another shortcoming of the *K*-means algorithm is that it is a mixture model, in which no class membership is considered. When comparing eq 1 to the normal density distribution equation, it can be seen that the nonlinear transformation of the Gaussian basis function actually models the probability density of the prediction spectrum at the basis function centroid. The linear summation of the hidden layer output models the probability of the training spectrum at the output unit. With the *K*-means clustering algorithm, each RBF unit in the hidden layer models the probability densities of different classes may overlap with each other. For example, the class B spectra may have a high density in the first RBF unit instead of having a high density in the second RBF unit. In this case, additional RBF units have to be added, and the weights have to be trained to compensate each other to derive a high probability for each class in the respective output units. By considering the class membership of the training spectra, for a single modal spectral distribution, it is possible to use as few as the number of classes to be classified to construct a compact RBFN. For the network constructed in this way, each class of spectra has a high-density distribution in only one hidden and output unit. For multimodal distribution, additional units may be needed to represent the probability distribution of each class. The advantage of this kind of network design is that the mechanism of the network can be better understood in terms of probability distribution theory, and the network is expected to be relatively compact. When taking into account the class-membership information, a theoretical solution with a compact RBF network for classification of three classes is illustrated in Figure 2, which is the same architecture as that of probabilistic neural networks. In Figure 2, linear neurons are assumed in the output layer. The network with three RBF units is the most compact network solution for three classes, unless the classes are correlated with each other. Without considering the class membership, more RBF units may be needed to resolve the overlapping density distributions. One more advantage of the proposed algorithm over *K*-means is that, by taking the class membership into account, the number of clusters can be computed according to the probability distribution of the class.

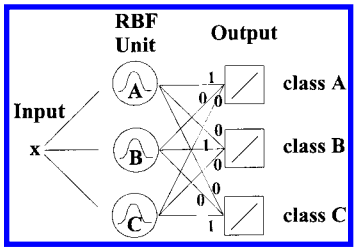

Figure 2. Theoretical compact model for classification with RBFN.

3.2. Linear-Averaging Clustering Algorithm. With the above considerations, the linear-averaging (L-A) algorithm was developed for computing centroids, radii, and the number of RBF units from the training data. Grouping of individual training object is related to the distance between the training object and the closest cluster of the same class membership. If it is beyond a certain distance, a new cluster is initialized. The grouping (clustering) parameter is controlled by a term of λ (in step 3 below). Considering ν classes in the training data, the L-A algorithm can be described as follows:

(1) For the ν classes, calculate the mean vector ($\bar{\mathbf{x}}$) and the average distance ($\bar{d}$) among the training spectra.

(2) Initialize the number of clusters (k) with ν, and initialize each cluster centroid with mean vector ($\bar{\mathbf{x}}$), initialize the radius with half of the mean distance ($\bar{d}/2$), and initialize the number of spectra (t) in the cluster with 1.

(3) For spectrum $\mathbf{x}_i$ (i is from 1 to *ns*), find its class membership and the smallest distances ($d_{\min}$) between the spectrum $\mathbf{x}_i$ and the clusters that are of the same class. If $d_{\min}$ is less than $\lambda\bar{d}$, group the spectrum in the nearest cluster of the same class and update the centroids and radii with geometric mean with the following computations:

$$\mathbf{c}^{n+1} = \frac{t \cdot \mathbf{c}^n + \mathbf{x}_i}{t+1} \tag{8}$$

$$\bar{d}^{n+1} = \frac{t \cdot \bar{d}^n + d_{\min}}{t+1} \tag{9}$$

Otherwise initialize a new cluster, and increase the number of clusters by 1.

Table 1. Italian Olive Oil Data

source region	objects	class index	source region	objects	class index
Calabria	56	1	East-Liguria	50	4
South-Apulia	206	2	West-Liguria	50	5
Inland-Sardin	65	3	Umbia	51	6

(4) Use obtained centroids and radius to construct the RBFN with the same number of RBFN units as the number of clusters.

It can be seen that with the L-A algorithm, only a single iteration is needed to construct RBFN, and the centroids, the radius, and the number of clusters are computed purely on the basis of information from the training data. The performance of L-A configured networks with respect to chemical pattern recognition will be evaluated with the poly(chlorobiphenyl) (PCB) data set and the olive oil data set.

4. EXPERIMENTAL SECTION

4.1.1. Poly(chlorobiphenyl) Data Sets. These data sets were used in previous work[32] and were used in this study to evaluate the network performance in terms of prediction accuracy, training speed, and network complexity. The data sets were electron ionization mass spectra for PCB compounds with chlorine atoms ranging from 1 to 9. The networks were trained to classify the spectra on the basis of the number of chlorines in the compound. The training set has 161 training objects; the test set has 120 objects. The report of neural network performance was based on the statistics of five runs with random weight initialization.

4.1.2. Italian Olive Oil Data Sets. Italian olive oil data sets were obtained from a public domain ftp site (ftp://ftp.clarkson.edu/pub/hopkepk/Chemdata/). This data set has been widely used as a standard reference data set for pattern recognition methods and has been well-characterized.[33,34] With this data set, the source regions of olive oils were to be classified on the basis of the content of eight fatty acids. However, only those with classes with more than 50 objects were used in this paper to alleviate composition imbalance in the training and test data sets. The composition of the data was given in Table 1. The prediction set was prepared by sampling 20, 40, 20, 20, 20, and 20% of spectra from each class; the remaining objects were used in the training set. Five pairs of such data sets were prepared with random withdrawing. The number of objects in the training and test set of each pair was 341 and 137, respectively.

4.2. Data Preprocessing. For both the PCB data sets and the Italian olive oil data sets, each object was normalized so that the intensity of the largest variable was unity. For PCB mass spectra, variables from 50 *m*/*z* to 550 *m*/*z* were used as in a previous study.[31] The normalized intensities of these variables (50 *m*/*z* to 550 *m*/*z*) were used as input to the neural networks with an input vector size of 500. For Italian olive oil data sets, the normalized fatty acid content for eight acids was used as input to the neural networks with an input vector size of 8. The biggest acid content among eight acids was unity after normalization.

4.3. Software and Hardware. All computations were run on a single processor 200-MHz Intel Pentium Pro Windows NT 4.0 workstation equipped with 64MB of RAM. Computer programs were written in C++ and compiled with a Borland 5.02 C++ compiler. All calculations used single-precision (32 bit) floating point arithmetic.

Table 2. Effects of λ on the Constructed Neural Networks with QuickProp Learning and PCB Data

λ	prediction $\pm$ std	RMSEP	layer size	time (min)	iterations
1.0	17 ± 3	2971.12	130	86.62	250 000
2.0	92 ± 1	0.24	71	1.17	6 646
3.0	89 ± 3	0.28	30	0.017	49
4.0	94 ± 3	0.21	17	0.015	35
5.0	94 ± 3	0.21	15	0.017	35
6.0	95 ± 4	0.20	12	0.017	35
7.0	93 ± 4	0.19	9	0.017	36
8.0	93 ± 4	0.20	9	0.012	36
9.0	94 ± 3	0.20	9	0.017	36
10.0	93 ± 4	0.20	9	0.017	36

5. RESULTS AND DISCUSSION

5.1. Evaluation of Linear-Averaging Configured Networks with PCB Data and QuickProp Learning. The linear-averaging algorithm can be used to automatically compute the centroid and radius for each RBF unit in the hidden layer. The number of hidden RBF units are computed by this algorithm. The grouping of the training object was controlled by parameter λ in step 3 of the algorithm, which also controlled the network configurations. The optimal selection of parameter λ may be affected by the specific training data and learning algorithm. Because of the popularity of QuickProp learning algorithm, the study of λ was carried out with QuickProp training of PCB data sets.

Prediction accuracy and its standard deviation, RMSEP, hidden layer size, and training times with respect to the changes of λ are given in Table 2. Very low prediction accuracy was observed with λ of 1.0, and networks never converged. The network training kept oscillating until the maximum of 250 000 iterations was reached in 86.62 min. With $\lambda \geq 2.0$, the overall prediction accuracy was usually better than 92%. With the use of a larger λ, the size of the hidden RBF layer tended to decrease until there was only one cluster for each class. The most dramatic decrease of hidden layer size was observed for the λ between 1.0 to 4.0. The RMSEP decreased for λ in the range from 1.0 to 4.0 and then essentially leveled off. For training times, the networks usually trained within 1 s except with λ of 2.0, with which the network training took 1.37 min to converge.

From Table 2, one may tend to draw a conclusion that a larger λ would be preferable. However, when looking at the prediction for each individual class (Figure 2), one could disclose some information that was not available from the overall prediction accuracy. The prediction accuracy for each individual class with λ of 2.0 are given in Figure 3, while the prediction accuracy with λ of 6.0 is given in Figure 4. It can be seen that, with λ of 6.0, the prediction for the first class (the PCBs with only one chlorine) was bad (0% prediction), and the predictions for the last class (the PCBs with nine chlorines) was very low. The high accuracy for other classes contributed to the overall good performance of the network. With λ of 2.0, the prediction for all classes was reasonably good. So if taking the prediction for every class into consideration, λ of 2.0 should be preferred. The same conclusion was reached with prediction plots with other values of λ.

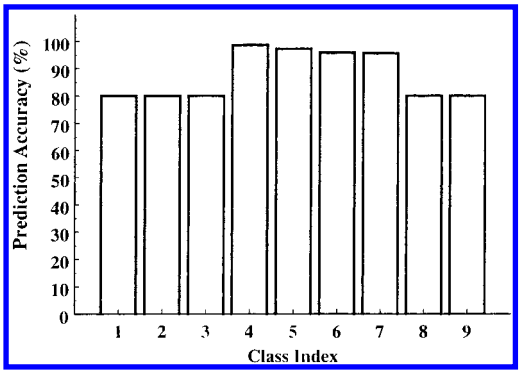

Figure 3. Prediction for each individual class with $\lambda = 2.0$.

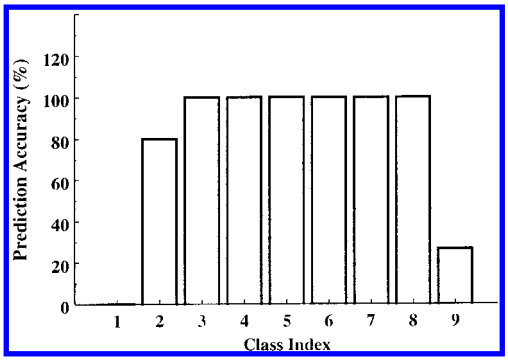

Figure 4. Prediction of individual class with $\lambda = 6.0$.

Table 3. Effects of Training Error (TE) on Prediction Accuracy and Iterations with PCB Data

TE	iterations	prediction ± std	RMSEP	time (min)
0.3	5 097	74 ± 5	0.38	0.81
0.2	6 646	92 ± 1	0.24	1.17
0.1	7 828	93 ± 0	0.17	1.25
0.05	55 766	95 ± 0	0.13	8.73
0.01	250 000	95 ± 0	0.12	35.00

While the λ parameter controls the clustering of the input data, two important parameters that are involved in the actual training are the desired recognition error and the number of iterations. As mentioned above, a maximum of 250 000 iterations was set for training, which is large enough for most cases. The termination of training is observed to be controlled by the recognition error with this number of maximum iterations in most cases, and the effects of recognition error on prediction accuracy and required iterations are given in Table 3. It can be seen from Table 3, with the decreasing recognition error, the required iteration increases. Substantial increases in training times were observed when the recognition error was below 0.1. Note that with a recognition error of 0.01, the training did not complete within 250 000 iterations, indicating that more iterations are needed to train the network to a recognition error of 0.01. The time for 250 000 iterations is about half of that with λ of 1.0 in Table 2, which is caused by the difference of the number of hidden units for the two networks (71 in Table 3 versus 130 in Table 2). The prediction accuracy is low with a training error of 0.3 due to insufficient training, while the prediction accuracy with a training error smaller than 0.3 is sufficiently good (>92%). The observation from Table 3 indicates that use of the training error of 0.2 is satisfactory in terms of prediction accuracy and training times.

5.2. Evaluation of Linear-Averaging Configured Networks with PCB Data and SVD Learning. With linear-averaging algorithm constructed networks and SVD learning of PCB data sets, the performance of the networks is given in Table 4. The prediction accuracy (100%) was surprisingly good for λ ranging from 1.0 to 10.0. It was observed that the RMSEP increased slightly from 0.028 to 0.049 when λ discretely increased from 1.0 to 10.0. The results were insensitive to the random weight initialization as evidenced by the standard deviation of 0.00 for both the prediction and recognition accuracy. One other point should be noted that with SVD training the training usually was less than a few seconds.

Table 4. Effects of λ on the RBFN Models of PCB Data with SVD Learning

λ	prediction ± std	RMSEP	time (min)	factors
1	100 ± 0	0.028	0.053	16
2	100 ± 0	0.028	0.027	18
3	100 ± 0	0.031	0.016	22
4	100 ± 0	0.037	0.016	16
5	100 ± 0	0.041	0.00	15
6	100 ± 0	0.046	0.00	10
7	100 ± 0	0.049	0.00	10
8	100 ± 0	0.049	0.016	10
9	100 ± 0	0.049	0.053	10
10	100 ± 0	0.049	0.00	10

Table 5. Effects of Threshold p on the Network Prediction with PCB Data

p	factor	prediction ± std
0.1	3	71 ± 0
0.01	9	98 ± 0
0.001	18	100 ± 0
0.0001	71	100 ± 0

Table 6. Effects of Threshold p on the Prediction of Individual PCB Class[a]

p	prediction for nine individual PCB classes								
0.1	0	0	0	100	100	100	89 ± 4	0	0
0.01	0	100	100	100	100	100	100	100	100
0.001	100	100	100	100	100	100	100	100	100
0.0001	100	100	100	100	100	100	100	100	100

[a] Each column corresponds to the number of Cl atoms in the PCB.

One important parameter with SVD training is the number of SVD factors used in the model. The number of SVD factors used for calculating the pseudo-inverse was controlled by a proportionality threshold p. Only factors that had singular values greater than p times the total of the singular values were used. For Table 4, p was empirically chosen to be 10^{-3}, which means that all the factors that had singular values larger than 0.1% of the sum of all the singular values were used in the SVD model. With the use of different number of SVD factors, different amounts of information were included in the SVD model, which may affect the prediction accuracy of neural networks. Table 4 revealed that λ also affected the number of SVD factors used in the SVD training. Though no linear relationship was observed between λ and the SVD factor, λ of 2.0 gave one of the lowest RMSEP values. Given λ of 2.0, the effects of the threshold term p on the overall prediction accuracy are shown in Table 5 with the prediction for each individual class given in Table 6. It was observed that with a p less than 10^{-3}, it was able to maintain 100% prediction accuracy, while, with a p of 0.1, several classes were completely misclassified. With a p

Table 7. SVD Training of L-A Constructed RBFN with Italian Olive Oil Data

set	recognition ± std	RMSEC	factor	time (min)
A	98 ± 1	0.096	42	0.15
B	97 ± 1	0.10	40	0.18
C	98 ± 0	0.11	42	0.15
D	97 ± 0	0.10	43	0.15
E	97 ± 1	0.10	41	0.18

Table 8. Prediction for Individual Class of Italian Olive Oil with Five Data Sets[a]

set	recognition for each class					
A	96	98	100	98	100	100
B	96	98	100	93	95	100
C	96	96	100	98	98	100
D	89	99	100	88	100	100
E	98	97	100	93	98	100

[a] Each column corresponds to class number.

Table 9. Hidden Layer Size and Clusters for Each Italian Olive Oil Class with Five Training Sets

set	class 1	class 2	class 3	class 4	class 5	class 6	total
A	41	61	18	17	24	27	188
B	42	85	27	25	20	1	200
C	37	92	19	24	11	17	200
D	17	90	19	13	25	23	187
E	42	87	16	26	28	1	200

of 0.01, all the classes except the first class were correctly classified. With the use of a smaller threshold p, more SVD factors were used in the model as expected, and a more accurate network prediction was achieved. It indicated that a p value less than 10^{-3} should be used to achieve 100% prediction accuracy with the PCB data sets.

5.3. Evaluation of Linear-Averaging Configured Networks with Italian Olive Oil Data and SVD Learning. Statistics obtained from L-A configured RBFN for five sets of Italian olive oil are given in Table 7. The prediction for each individual class is given in Table 8. For all five data sets and with a threshold p of 10^{-3}, the networks were trained to a RMSEC around 0.10 and prediction accuracy was greater than 96%. About 41 SVD factors were used in the computation. It took about 10 s to train. Table 8 also discloses that the recognition for classes 3 and 6 has achieved 100% training, but, for classes 1 and 4 in set d, the recognition was less than 90%. The training objects for these two classes were not as representative as in other training sets. Table 9 gives the hidden layer size for each data set and the clusters formed for each class of training objects. The hidden layer size did not vary much even though the clusters formed for each class somewhat varied. This stability of size indicated that the L-A algorithm had good convergence in terms of the hidden layer.

With the L-A method configured neural networks, the prediction with five olive oil data sets are given in Table 10. The overall prediction accuracy for the five test sets was better than 94%, and the prediction for classes 3 and 6 was 100%. For a few cases, accuracy was not that good; it might be caused by the extrapolation in the training space. If more training objects were available, prediction might be improved.

5.4. Comparison of Performance with *K*-Means Algorithm. The *K*-means algorithm has been proven to perform

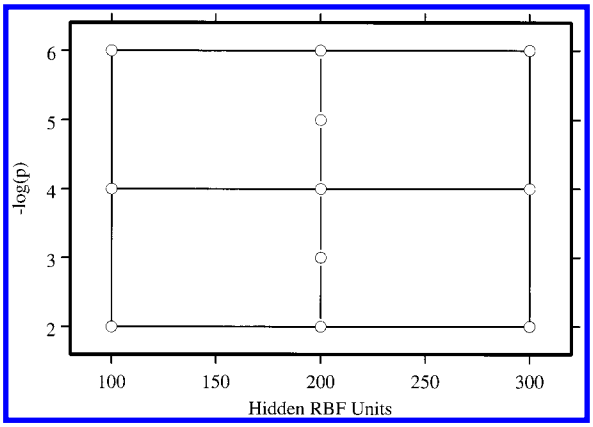

Figure 5. Central composite design for studying effects of hidden layer size and threshold p.

Table 10. Prediction with Five Sets of Olive Oil Data

set	prediction ± std	RMSEP	prediction for each olive oil class					
A	95 ± 2	0.12	78	96	100	100	78	100
B	96 ± 1	0.11	90	99	100	63	100	100
C	98 ± 1	0.098	100	100	100	75	100	100
D	97 ± 0	0.11	89	97	100	90	100	100
E	96 ± 1	0.11	100	97	100	89	80	100

reasonably well for most applications. However, the optimal number of RBF units was not automatically computed but obtained from trial and error. In contrast to the *K*-means algorithm, L-A can optimize the number of the RBF units for the RBFN with information from the training data. As a control method, the *K*-means algorithm was applied to classify the same Italian olive oil data sets with the number of hidden RBF units set to 50 and a threshold p of 10^{-3}. The prediction accuracy was unexpectedly bad, with a prediction accuracy of 38% and a RMSEC of 0.36. The low recognition was found to be caused by the fact that only the second class was correctly classified (100%), while other classes were completely misclassified (0% recognition). It was also noted that with p of 10^{-3}, only one SVD factor was used with olive oil data sets. Thus two factors may be responsible for the low prediction accuracy. One may be that too few hidden units were used, and a number of hidden units greater than 50 should be used with the *K*-means algorithm. Another factor may be that too few SVD factors were used, which was controlled by threshold p. To optimize these two factors for the *K*-means algorithm, central composite experimental design was employed to investigate the effects of these two variables, as illustrated in Figure 5. The experiments used a combination of a number of hidden units of 50, 200, and 300 and a p term of 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}.

The prediction results as a function of hidden layer size on the training and prediction of olive oil data are given in Table 11. The prediction accuracy with the *K*-means algorithm was worse than that with the L-A algorithm. The use of a larger hidden layer size did not improve the prediction accuracy substantially. Note that with the hidden layer changed from 50 to 300 units, the number of SVD factors used only changed by one. Thus the size of hidden layer did not affect the number of SVD factors used for model building, but the threshold p was a critical parameter.

Table 11 gives the effects of threshold p on the network performance with the *K*-means algorithm. It can be seen that,

Table 11. Effects of Hidden Layer Size and Tolerance *p* on the Network Performance with *K*-Means Algorithm and Olive Oil Data

hidden units	$-\log(p)$	factor	recognition ± std	RMSEC	prediction ± std	RMSEP
100	2	1	60 ± 4	0.32	58 ± 3	0.33
100	4	3	77 ± 2	0.27	76 ± 2	0.25
100	6	7	93 ± 3	0.2	92 ± 2	0.21
200	2	1	47 ± 2	0.33	48 ± 3	0.33
200	3	2	49 ± 16	0.31	50 ± 15	0.34
200	4	4	81 ± 2	0.31	81 ± 1	0.25
200	5	6	89 ± 5	0.23	87 ± 7	0.24
200	6	9	98 ± 1	0.21	95 ± 2	0.23
300	2	1	59 ± 5	0.25	58 ± 3	0.34
300	4	4	80 ± 3	0.26	80 ± 2	0.25
300	6	10	95 ± 3	0.22	95 ± 1	0.21

with the use of a smaller threshold *p*, more SVD factors were included in the model, and the prediction accuracy increased. For example, with *p* of 10^{-6}, the network was trained to a recognition accuracy of 98% and a prediction accuracy of 95%. However, when compared to the RMSEP of 0.066 obtained from the L-A algorithm results, which used the same number of SVD factors (nine-factor model), the RMSEP of 0.23 was obviously much worse. The observations with a different combination of parameters further proved that the L-A configured network generally worked better than its *K*-means counterpart using the same number of SVD factors for Italian olive oil data sets.

The above study indicates that the L-A algorithm can not only configure the network (i.e. select the number of hidden RBF units and compute the centroid and radius for each hidden RBF unit) but also yield an improved prediction compared with the *K*-means algorithm. The optimized networks configured with L-A have achieved higher prediction accuracy with the PCB data and the Italian olive oil data sets. It is expected that the linear-averaging algorithm can also be applied to other chemical pattern recognition problems.

The RBFN configured with the L-A algorithm has some similarities to fuzzy linear interpolation network (FLIN)[35] in that both employed clustering analysis in the hidden layer to extract information solely from the network input. However, FLIN used a Kohonen layer and a Grossberg layer while RBFN used a RBF hidden layer, and there was no algorithm developed to optimize the size of the hidden layer for FLIN. However, despite the satisfactory predictions demonstrated with PCB data sets and Italian olive oil data sets for the L-A algorithm configured RBFNs, it is worthwhile to point out some disadvantages of RBFNs. One is that RBFN usually requires more memory storage space than MLFNs because the centroids of hidden RBF units have the same dimensionality as that of input objects, whose dimensionalities are usually very high. Feature extraction can alleviate this problem. Memory requirement is also proportional to the number of hidden RBF units; thus compactness of the network is a critical factor in concern. Another point to be considered is that RBFN is essentially an implementation of the kernel density estimation method, optimal parametrization may not work well all the time, and extrapolation could happen which decreases the prediction accuracy. Despite the efforts to optimize the RBFN configuration as demonstrated in this paper, further study is still needed to address the above-mentioned disadvantages of RBFNs.

6. CONCLUSIONS

An effective method, linear averaging (L-A), has been presented to automatically configure RBF neural networks. The proposed algorithm can compute the number of radial basis functions, the centroid, and the radius on the basis of the information from the training data without external user input. The algorithm was based on the probability distribution of the training data, and the class memberships of the training object and cluster were considered during the clustering process. The performance of L-A configured neural networks for chemical pattern recognition was evaluated with PCB data sets and Italian olive oil data sets. With QuickProp learning, the prediction accuracy was usually above 92% with a small hidden layer size, and training times were usually not longer than 1 min. Learning with the SVD algorithm gave even much better prediction accuracy and lower RMSEP compared to the QuickProp algorithm. The RMSEP as low as 0.028 has been achieved with optimized networks. With the L-A configured networks, it was found out that the number of SVD factors, which was controlled by a parameter *p*, had significant effects on the prediction accuracy. A value of *p* smaller than 0.001 was generally required to achieve satisfactory prediction. Compared with the *K*-mean algorithm results on the same data sets, the L-A algorithm usually had a much lower RMSEP. The evaluation with PCB and Italian olive oil data sets indicates that L-A can not only automatically configure the network but also have overall better performance. The L-A algorithm may be applied to a variety of chemical pattern recognition problems for fast training and accurate classification.

ACKNOWLEDGMENT

Part of this work was presented to the 50th Pittsburgh Conference of the Analytical Chemistry and Spectroscopy Societies (PITTCON'99), Orlando, FL, Mar 7−12, 1999. Tricia Buxton and Susan Slagel are thanked for their proofreading and helpful suggestions. Reviewers are thanked for their valuable comments for this work.

REFERENCES AND NOTES

(1) Bishop, C. M. *Neural Networks for Pattern Recognition*; Clarendon Press: Oxford, U.K., 1995.

(2) Rollet, R.; Benie, G. B.; Li, W.; Wang, S. Image classification algorithm based on the RBF neural network and K-means. *Int. J. Remote Sens.* **1998**, *19*, 3003−3009.

(3) Garris, M. D.; Wilson, C. L.; Blue, J. L. Neural network-based systems for handprint OCR applications. *IEEE Trans. Image Process.* **1998**, *7*, 1097−1112.

(4) Phillips, W. J.; Tosuner, C.; Robertson, W. Speech Recognition Techniques Using RBF Networks. *Proceedings of the IEEE WESCANEX 95. Communications, Power, and Computing*; IEEE: New York, 1995; Vol. 1, pp 185−190.

(5) Salomon, R. The Application of Radial Basis Function Networks with Implicit Continuity Constraints. In *Artificial Neural Networks-ICANN'97*; Gerstner, W., Germond, A., Hasler, M., Nocoud, J. D., Eds.; Lecture Notes in Computer Science *1327*; 1997; pp 805−810.

(6) Fishbacher, C.; Jagemann, K. U.; Danzer, K.; Muller, U. A.; Papenkordt, L.; Schuler, J. Enhancing calibration models for noninvasive near-infrared spectroscopic blood glucose determination. *Fresenius J. Anal. Chem.* **1997**, *359*, 78−82.

(7) Goodacre, R.; Hammond, D.; Kell, D. B. Quantitative Analysis of the Adulteration of Orange Juice with Sucrose Using Pyrolysis Mass Spectrometry and Chemometrics. *J. Anal. Appl. Pyrol.* **1997**, *40−41*, 135−158.

(8) Sanchez, M. S.; Swierenga, H.; Sarabia, L. A.; Derks, E.; Buydens, L. Performance of Multi-Layer Feedforward and Radial Base Function

Neural Networks in Classification and Modeling. *Chemomet. Intell. Lab. Syst.* **1996**, *33*, 101−119.
(9) Tetteh, J.; Metcalfe, E.; Howells, S. L. Optimization of Radial Basis and Backpropagation Neural Networks for Modeling Auto-Ignition Temperature by Quantitative-Structure Property Relationships. *Chemomet. Intell. Lab. Syst.* **1996**, *32*, 177−199.
(10) Schaller, E.; Bosset, J. O.; Escher, F. "Electronic Noses" and Their Application to Food. *Food Sci. Technol. (London)* **1998**, *31*, 305−316.
(11) Goodacre, R. Use of Pyrolysis Mass Spectrometry with Supervised Learning for the Assessment of the Adulteration of Milk of Different Species. *Appl. Spectrosc.* **1997**, *51*, 1144−1153.
(12) Goodacre, R.; Timmins, E. M.; Burton, R.; Kaderbhai, N.; Woodward, A. M.; Kell, D. B.; Rooney, P. J. Rapid Identification of Urinary Tract Infection Bacteria Using Hyperspectral Whole-Organism Fingerprinting and Artificial Neural Networks. *Microbiology* (*Reading, U.K.*) **1998**, *144*, 1157−1170.
(13) Kang, S. G.; Kenyon, R. G. W.; Ward, A. C. Analysis of Different State in Streptomyces Albidoflavus SMF301 by the Combination of Pyrolysis Mass Spectrometry and Neural Networks. *J. Biotechnol.* **1998**, *62*, 1−10.
(14) Chakravarthy, S. V.; Ghosh, J. Scale-Based Clustering Using the Radial Basis Function Network. *IEEE Trans. Neural Networks* **1996**, *7*, 1250−1261.
(15) Kubat, M. Decision Trees Can Initialize Radial-Basis Function Networks. *IEEE Trans. Neural Networks* **1998**, *9*, 813−821.
(16) Zheng, G. L.; Billings, S. A. Radial Basis Function Network Configuration Using Mutual Information and the Orthogonal Least Square Algorithm. *Neural Networks* **1996**, *9*, 1619−1637.
(17) Haykin, S. *Neural Networks: A Comprehensive Foundation*; MacMillan: New York, 1994.
(18) Hwang, Y. S.; Bang, S. Y. An Efficient Method to Construct a Radial Basis Function, Neural Network Classifier. *Neural Networks* **1997**, *10*, 1495−1503.
(19) Mukherjee, S.; Nayar, S. K. Automatic Generation of RBF Networks Using Wavelets. *Pattern Recognit.* **1996**, *29*, 1369−1383.
(20) Billings, S. A.; Zheng, G. L. Radial Basis Function Network Configuration Using Genetic Algorithms. *Neural Networks* **1995**, *8*, 877−890.
(21) Li, K. Initializing of an RBF Network by a Genetic Algorithm. *Neurocomputing* **1997**, *14*, 273−288.
(22) Hartigan, J. The K-means Algorithm. In *Clustering Algorithms*; Hartigan, J. A., Ed.; Wiley: New York, 1975; pp 84−112.
(23) Rollet, R.; Benie, G. B.; Li, W.; Wang, S.; Boucher, J. M. Image Classification Algorithm Based on RBF Neural Network and K-means. *Int. J. Remote Sens.* **1998**, *19*, 3003−3009.
(24) Sutanto, E. L.; Mason, J. D.; Warwick, K. Mean-Tracking Clustering Algorithm for Radial Basis Function Center Selection. *Int. J. Control* **1997**, *67*, 961−977.
(25) Bruzzone, L.; Prieto, D. F. Supervised Training Technique for Radial Basis Function Neural Networks. *Electron. Lett.* **1998**, *34*, 1115−1116.
(26) Masters, T. In *Practical Neural Network Recipes in C++*; Masters, T., Ed.; Academic Press: New York, 1993.
(27) Tetko, I. V.; Villa, A. E. P.; Livingstone, D. J. Neural Network Studies. 2. Variable Selection. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 794−803.
(28) Tetko, I. V.; Villa, A. E. P.; Aksenova, T. I.; Zielinski, W. L.; Brower, J.; Collantes, E. R.; Welsh, W. J. Application of Pruning Algorithm To Optimize Artificial Neural Networks for Pharmaceutical Fingerprinting. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 676−684.
(29) Kovalishyn, V. V.; Tetko, I. V.; Luik, A. I.; Kholodovych, V. V.; Villa, A. E. P.; Livingstone, D. J. Neural Network-Studies. 3. Variable Selection in the Cascade-Correlation Learning Architecture. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 794−803.
(30) Fahlman, S. E. *An Empirical Study of Learning Speed in Back-Propagation Networks*; Technique Report, Report CMU-CS-88-162; Carnegie Mellon University: Pittsburgh, PA, September, 1988.
(31) Masters, T. In *Advanced Algorithms for Neural Networks A C++ Sourcebook*; Masters, T., Ed.; Wiley: New York, 1995; pp 157−222.
(32) Harrington, P. B. Temperature-Constrained Cascade Correlation Networks. *Anal. Chem.* **1998**, *70*, 1297−1306.
(33) Forina, M.; Tiscornia, E. Pattern Recognition Methods in the Prediction of Italian Olive Oil Origin by their Fatty Acid Content. *Ann. Chim.* **1982**, *72*, 143−155.
(34) Glover, D. M.; Hopke, P. K. Exploration of Multivariate Chemical Data by Projection Pursuit. *Chemomet. Intell. Lab. Syst.* **1992**, *16*, 45−59.
(35) Harrington, P. B.; Pack, B. W. FLIN: Fuzzy Linear Interpolating Network. *Anal. Chim. Acta* **1993**, *277*, 189−197.

CI990306T