Data Classification With Radial Basis Function Networks Based on a Novel Kernel Density Estimation Algorithm

Yen-Jen Oyang, *Member, IEEE*, Shien-Ching Hwang, Yu-Yen Ou, *Student Member, IEEE*, Chien-Yu Chen, *Member, IEEE*, and Zhi-Wei Chen

***Abstract*—This paper presents a novel learning algorithm for efficient construction of the radial basis function (RBF) networks that can deliver the same level of accuracy as the support vector machines (SVMs) in data classification applications. The proposed learning algorithm works by constructing one RBF subnetwork to approximate the probability density function of each class of objects in the training data set. With respect to algorithm design, the main distinction of the proposed learning algorithm is the novel kernel density estimation algorithm that features an average time complexity of $O(n \log n)$, where n is the number of samples in the training data set. One important advantage of the proposed learning algorithm, in comparison with the SVM, is that the proposed learning algorithm generally takes far less time to construct a data classifier with an optimized parameter setting. This feature is of significance for many contemporary applications, in particular, for those applications in which new objects are continuously added into an already large database. Another desirable feature of the proposed learning algorithm is that the RBF networks constructed are capable of carrying out data classification with more than two classes of objects in one single run. In other words, unlike with the SVM, there is no need to resort to mechanisms such as one-against-one or one-against-all for handling datasets with more than two classes of objects. The comparison with SVM is of particular interest, because it has been shown in a number of recent studies that SVM generally are able to deliver higher classification accuracy than the other existing data classification algorithms. As the proposed learning algorithm is instance-based, the data reduction issue is also addressed in this paper. One interesting observation in this regard is that, for all three data sets used in data reduction experiments, the number of training samples remaining after a naïve data reduction mechanism is applied is quite close to the number of support vectors identified by the SVM software. This paper also compares the performance of the RBF networks constructed with the proposed learning algorithm and those constructed with a conventional cluster-based learning algorithm. The most interesting observation learned is that, with respect to data classification, the distributions of training samples near the boundaries between different classes of objects carry more crucial information than the distributions of samples in the inner parts of the clusters.**

Manuscript received December 24, 2002; revised April 4, 2004. This work was supported by the National Science Council, R.O.C., under Contract NSC 92-2213-E-002-095.

Y.-J. Oyang, S.-C. Hwang, and Y.-Y. Ou are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 106, Taiwan, R.O.C. (e-mail: yjoyang@csie.ntu.edu.tw; schwang@mars.csie.ntu.edu.tw; yien@csie.ntu.edu.tw.

C.-Y. Chen was with the Department of Computer Science and Information Engineering, National Taiwan University, R.O.C. She is now with the Graduate School of Biotechnology and Bioinformatics, Yuan Ze University, Chung-Li, Taiwan, 320, R.O.C. (e-mail: cychen@mars.csie.ntu.edu.tw).

Z.-W. is with the Advance Research Department, Quanta Computer Incorporated, Taipei, Taiwan, 231, R.O.C.

Digital Object Identifier 10.1109/TNN.2004.836229

***Index Terms*—Data classification, kernel density estimation, machine learning, neural network, radial basis function (RBF) network.**

I. Introduction

THE RADIAL basis function (RBF) network is a special type of neural networks with several distinctive features [18], [20], [24]. Since its first proposal, the RBF network has attracted a high degree of interest in research communities. A RBF network consists of three layers, namely the input layer, the hidden layer, and the output layer. The input layer broadcasts the coordinates of the input vector to each of the units in the hidden layer. Each unit in the hidden layer then produces an activation based on the associated RBF. Finally, each unit in the output layer computes a linear combination of the activations of the hidden units. How a RBF network reacts to a given input stimulus is completely determined by the activation functions associated with the hidden units and the weights associated with the links between the hidden layer and the output layer.

RBF networks have been exploited in many applications and quite a few learning algorithms have been proposed [2], [4], [5], [14], [16], [24], [26], [29], [30]. The problems that RBF networks have been applied to include function approximation, data classification, and data clustering. Depending on the problems that the learning algorithms are designed for, different optimization criteria may be employed.

One of the main applications that RBF networks have been applied to is data classification. However, latest development in data classification research has focused more on support vector machines (SVMs) [10] than on RBF networks, because several recent studies have reported that SVM generally are able to deliver higher classification accuracy than the other existing data classification algorithms [13], [15], [17]. Nevertheless, SVM suffer one serious drawback. That is, the time taken to carry out model selection could be unacceptably long for some contemporary applications, in particular, for those applications in which new objects are continuously added into an already large database. Therefore, how to expedite the model selection process has become a critical issue for SVM and has been addressed by a number of recent works [8], [11], [12], [19]. However, the approaches that have been proposed so far for expediting the model selection process of SVM all lead to lower prediction accuracy. Anyway, this is an issue that deserves continuous investigation. Another minor drawback of SVM is that mechanisms

such as one-against-one or one-against-all must be invoked to handle datasets with more than two classes of objects.

In this paper, a novel learning algorithm is proposed for efficient construction of the RBF networks that can deliver the same level of accuracy as SVM in data classification applications without suffering the drawbacks of SVM addressed above. In the RBF networks constructed with the proposed learning algorithm, each activation function associated with the hidden units is a spherical (or symmetrical) Gaussian function. In some works, the specific type of RBF networks with spherical Gaussian functions (SGFs) is referred to as the spherical Gaussian RBF network [31]. For simplicity, we will use SGF networks to refer to the RBF networks constructed with the learning algorithm proposed in this paper. With respect to algorithm design, the main distinction of the proposed learning algorithm is the novel kernel density estimation algorithm designed for efficient construction of the SGF networks. The main properties of the proposed learning algorithm are summarized as follows

1) The SGF networks constructed with the proposed learning algorithm generally deliver the same level of classification accuracy as the SVM.
2) The average time complexity for constructing an SGF network is bounded by $O(n \log n)$, where n is total number of training samples.
3) The average time complexity for classifying n' incoming objects is bounded by $O(n' \log n)$.
4) The SGF networks are capable of carrying out data classification with more than two classes of objects in one single run. That is, unlike with the SVM, there is no need to resort to mechanisms such as one-against-one or one-against-all for handling data sets with more than two classes of objects.

As the SGF networks constructed with the proposed learning algorithm are instance-based, the efficiency issue shared by almost all instance-based learning algorithms must be addressed. That is, a data reduction mechanism must be employed to remove redundant samples in the training data set in order to improve the efficiency of the instance-based classifiers. Experimental results reveal that the naïve data reduction mechanism employed in this paper is able to reduce the size of the training data set substantially with a slight impact on classification accuracy. One interesting observation is that, in the three data sets used in experiments, the number of training samples remaining after data reduction is applied and the number of support vectors identified by the SVM software are in the same order. In fact, in two out of the three cases reported in this paper, the difference is less than 15%. Since data reduction is a crucial issue for instance-based learning algorithms, further study on this issue should be conducted.

This paper also compares the performance of the SGF networks constructed with the proposed learning algorithm and the RBF networks constructed with a conventional cluster-based learning algorithm [16]. The most interesting observation learned is that, with respect to data classification, the distributions of samples near the boundaries between different classes of objects carry more crucial information than the distributions of samples in the inner parts of the clusters. Since the conventional cluster-based learning algorithm for RBF networks places one RBF at the center of a cluster, the distributions of samples near the boundaries between different classes of objects may not be accurately modeled. As a result, the RBF networks constructed with the conventional cluster-based learning algorithm in general are not able to deliver the same level of accuracy as those data classification algorithms such as SVM and the SGF networks that exploit the distributions of samples near the boundaries between different classes of objects.

Section II presents a review of the related works. Section III presents an overview of how data classification is conducted with the proposed learning algorithm. Section IV elaborates the novel kernel density estimation algorithm on which the proposed learning algorithm is based. Section V discusses the implementation issues and presents an analysis of time complexity. Section VI reports the experiments conducted to evaluate the performance of the proposed learning algorithm. Finally, concluding remarks are presented in Section VII.

II. Related Works

As mentioned earlier, there have been quite a few learning algorithms proposed for RBF networks. The learning algorithm determines the number of units in the hidden layer, the activation functions associated with the hidden units, and the weights associated with the links between the hidden and output layers. Learning algorithms designed for different applications may employ different optimization criteria. The general mathematical form of the output units in a RBF network is as follows:

$$\widehat{f}_j(\boldsymbol{x}) = \sum_{i=1}^{h} w_{i,j} r_i(\boldsymbol{x})$$

where $\widehat{f}_j$ is the function corresponding to the jth output unit and is a linear combination of h RBFs $r_1, r_2, \ldots, r_h$. Basically, there are two categories of learning algorithms proposed for RBF networks [5], [24], [26]. The first category of learning algorithms simply places one RBF at each sample [25]. On the other hand, the second category of learning algorithms attempts to reduce the number of hidden units in the network, or equivalently the number of RBFs in the linear function above [9], [16], [21]–[23]. One primary motivation behind the design of the second category of algorithms is to improve the efficiency of the learning process, as the conventional approaches employed to figure out the optimal parameter settings for an RBF network involve computing the inverse of a matrix with dimension equal to the number of hidden units in the network.

As mentioned earlier, one of the main applications of RBF networks is data classification. Most learning algorithms proposed for constructing RBF network-based classifiers conduct a clustering analysis on the training data set and allocate one hidden unit for each cluster [5], [14], [16], [22]. Algorithms differ by the clustering algorithm employed and how the parameters of the RBF network are set. The cluster-based approaches effectively improve the efficiency of the learning algorithm and

reduce the complexity of the RBF network constructed. However, because the cluster-based approaches typically place one RBF at the center of each cluster, the distributions of training samples near the boundaries between different classes of objects may not be accurately modeled. As the experimental results presented in Section VI of this paper reveals, with respect to data classification, the distributions of samples near the boundaries between different classes of objects carry more crucial information than the distributions of samples in the inner parts of the clusters. As a result, the RBF networks constructed with a conventional cluster-based learning algorithm generally are not able to deliver the same level of accuracy as those data classification algorithms such as SVM and the SGF networks that exploit the distributions of samples near the boundaries between different classes of objects.

In this paper, a novel learning algorithm for constructing SGF networks is presented. The mathematical treatment presented in this paper for the derivation of the learning algorithm is different from our previous work [27]. Nevertheless, both treatments exploit essentially the same idea and result in the same equations.

III. Overview of Data Classification With the Proposed Learning Algorithm

This section presents an overview of how data classification is conducted with the SGF networks constructed with the proposed learning algorithm. The details of the learning algorithms will be elaborated in the next section.

Assume that the objects of concern are distributed in an m-dimensional vector space and let f_j denote the probability density function that corresponds to the distribution of class-j objects in the m-dimensional vector space. The proposed learning algorithm constructs one SGF subnetwork for approximating the probability density function of one class of objects in the training data set. In the construction of the SGF network, the learning algorithm places one SGF at each training sample. The general form of the SGF network-based function approximators is as follows:

$$\hat{f}_j(\boldsymbol{v}) = \sum_{\boldsymbol{s_i} \in S_j} w_i \exp\left(-\frac{\|\boldsymbol{v} - \boldsymbol{s_i}\|^2}{2\sigma_i^2}\right) \tag{1}$$

where

$\hat{f}_j$	the SGF network-based function approximator for class-j training samples;
$\boldsymbol{v}$	a vector in the m-dimensional vector space;
S_j	the set of class-j training samples;
$\|\boldsymbol{v} - \boldsymbol{s_i}\|$	the distance between vectors $\boldsymbol{v}$ and $\boldsymbol{s_i}$;
w_i and σ_i	parameters to be set by the learning algorithm.

With the SGF network-based function approximators, a new object located at $\boldsymbol{v}$ with an unknown class is predicted to belong to the class that gives the maximum value of the likelihood functions defined in the following:

$$L_j(\boldsymbol{v}) = \frac{|S_j|}{|S|}\hat{f}_j(\boldsymbol{v})$$

where S_j is the set of class-j training samples and S is the set of training samples of all classes.

The essential issue of the learning algorithm is to construct the SGF network-based function approximators. In the next section, the novel kernel density estimation algorithm designed for efficient construction of the SGF network will be presented. For the time being, let us address how to estimate the value of the probability density function at a training sample. Assume that the sampling density is sufficiently high. Then, by the law of large numbers in statistics [28], we can estimate the value of the probability density function $f_j(\cdot)$ at a class-j sample $\boldsymbol{s_i}$ as follows:

$$f_j(\boldsymbol{s_i}) \cong \frac{(k_1+1)}{|S_j|} \cdot \left[\frac{R(\boldsymbol{s_i})^m \pi^{\frac{m}{2}}}{\Gamma\left(\frac{m}{2}+1\right)}\right]^{-1} \tag{2}$$

where

$R(\boldsymbol{s_i})$	the maximum distance between $\boldsymbol{s_i}$ and its k_1 nearest training samples of the same class;
$(R(\boldsymbol{s_i})^m \pi^{m/2})/$ $(\Gamma((m/2)+1))$	the volume of a hypersphere with radius $R(\boldsymbol{s_i})$ in an m-dimensional vector space;
$\Gamma(\cdot)$	the Gamma function [1];
k_1	a parameter to be set either through cross validation or by the user.

In (2), $R(\boldsymbol{s_i})$ is determined by one single training sample and therefore could be unreliable, if the data set is noisy. In our implementation, we use $\overline{R}(\boldsymbol{s_i})$ defined in the following to replace $R(\boldsymbol{s_i})$ in (2)

$$\overline{R}(\boldsymbol{s_i}) = \frac{m+1}{m}\left(\frac{1}{k_1}\sum_{h=1}^{k_1}\|\hat{\boldsymbol{s}}_{\boldsymbol{h}} - \boldsymbol{s_i}\|\right)$$

where $\hat{\boldsymbol{s}}_{\boldsymbol{1}}, \hat{\boldsymbol{s}}_{\boldsymbol{2}}, \ldots, \hat{\boldsymbol{s}}_{\boldsymbol{k_1}}$ are the k_1 nearest training samples of the same class as $\boldsymbol{s_i}$. The basis of employing $\overline{R}(\boldsymbol{s_i})$ is elaborated in Appendix A.

IV. Proposed Kernel Density Estimation Algorithm

This section elaborates the efficient kernel density estimation algorithm for construction of the SGF network-based function approximators. In fact, the proposed kernel density estimation algorithm is derived with some ideal assumptions. Therefore, some sort of adaptations must be employed, if the target data set does not conform to these assumptions. In this section, we will first focus on the derivation of the kernel density estimation algorithm, provided that these ideal assumptions are valid. The adaptations employed in this paper will be addressed later.

Assume that we now want to derive an approximate probability density function with the set of class-j training samples. The proposed kernel density estimation algorithm places one SGF at each sample as shown in (1). The challenge now is how to figure out the optimal w_i and σ_i values of each Gaussian function. For a training sample $\boldsymbol{s_i}$, the kernel density estimation algorithm first conducts a mathematical analysis on a synthesized data set. The synthesized data set is derived from two ideal assumptions and serves as an analogy of the distribution of class-j training samples in the proximity of $\boldsymbol{s_i}$. The first assumption is that the sampling density in the proximity of $\boldsymbol{s_i}$ is sufficiently high and, as a result, the variation of the probability

density function $f_j(\cdot)$ at $\boldsymbol{s_i}$ and the neighboring class-j samples approaches 0. The second assumption is that $\boldsymbol{s_i}$ and the neighboring class-j samples are evenly spaced by a distance determined by the value of the probability density function at $\boldsymbol{s_i}$. Fig. 1 shows an example of the synthesized data set for a training sample in a two-dimensional (2-D) vector space. The details of the model are elaborated in the following.

1) Sample $\boldsymbol{s_i}$ is located at the origin and the neighboring class-j samples are located at $(h_1\delta_i, h_2\delta_i, \ldots, h_m\delta_i)$, where $h_1, h_2, \ldots, h_m$ are integers and δ_i is the average distance between two adjacent class-j training samples in the proximity of $\boldsymbol{s_i}$. How δ_i is determined will be addressed later on.
2) The values of the probability density function $f_j(\cdot)$ at all the samples in the synthesized data set, including $\boldsymbol{s_i}$, are all equal to $f_j(\boldsymbol{s_i})$. The value of $f_j(\boldsymbol{s_i})$ is estimated based on (2) in Section III.

The proposed kernel density estimation algorithm begins with an analysis on the synthesized data set to figure out the values of w_i and σ_i that make function $g_i(\cdot)$ defined in the following virtually a constant function equal to $f_j(\boldsymbol{s_i})$:

$$g_i(\boldsymbol{x}) = w_i \left[\sum_{h_1=-\infty}^{\infty} \cdots \sum_{h_m=-\infty}^{\infty} \exp\left(-\frac{\|\boldsymbol{x} - (h_1\delta_i, \ldots, h_m\delta_i)\|^2}{2\sigma_i^2} \right) \right] \cong f_j(\boldsymbol{s_i}). \tag{3}$$

In other words, the objective is to make $g_i(\boldsymbol{x})$ a good approximator of $f_j(\boldsymbol{x})$ in the proximity of $\boldsymbol{s_i}$.

Let

$$q(y) = \sum_{h=-\infty}^{\infty} \exp\left(-\frac{(y - h\delta_i)^2}{2\sigma_i^2} \right) \tag{4}$$

where y is a real number. Then, we have

$$w_i \cdot \left\{ \underset{y \in \boldsymbol{R}}{\text{Minimum}} [q(y)] \right\}^m \le g_i(\boldsymbol{x}) \le w_i \cdot \left\{ \underset{y \in \boldsymbol{R}}{\text{Maximum}} [q(y)] \right\}^m \tag{5}$$

since

$$\sum_{h_1=-\infty}^{\infty} \sum_{h_2=-\infty}^{\infty} \cdots \sum_{h_m=-\infty}^{\infty} \exp\left(-\frac{\|\boldsymbol{x} - (h_1\delta_i, h_2\delta_i, \ldots, h_m\delta_i)\|^2}{2\sigma_i^2} \right) = \left[\sum_{h_1=-\infty}^{\infty} \exp\left(-\frac{(x_1 - h_1\delta_i)^2}{2\sigma_i^2} \right) \right] \cdot \left[\sum_{h_2=-\infty}^{\infty} \exp\left(-\frac{(x_2 - h_2\delta_i)^2}{2\sigma_i^2} \right) \right] \cdots \left[\sum_{h_m=-\infty}^{\infty} \exp\left(-\frac{(x_m - h_m\delta_i)^2}{2\sigma_i^2} \right) \right]$$

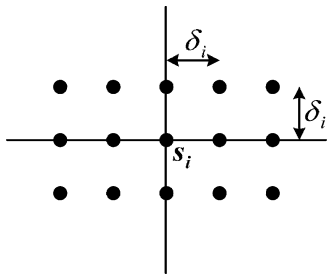


Fig. 1. Example of the synthesized data set for a training sample in a 2-D vector space.

TABLE I
BOUNDS OF FUNCTION $q(y)$ DEFINED IN (4) WITH ALTERNATIVE σ_i/δ_i RATIOS

$\beta = \sigma_i/\delta_i$	Bounds of $q(y)$
0.5	$1.253314144 \pm 1.80 \times 10^{-2}$
1.0	$2.506628275 \pm 1.34 \times 10^{-8}$
1.5	$3.759942412 \pm 2.94 \times 10^{-11}$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$. It is shown in Appendix B that, if $\sigma_i = \delta_i$, then $q(y)$ is bound by $2.506\,628\,274\,5 \pm 1.34 \times 10^{-8}$. Therefore, with $\sigma_i = \delta_i$, $g_i(\boldsymbol{x})$ defined in (3) is virtually a constant function. In fact, we can apply basically the same procedure presented in Appendix B to find the upper bounds and lower bounds of $q(y)$ with alternative σ_i/δ_i ratios. As Table I reveals, the bounds of $q(y)$ become tighter, if $\beta = \sigma_i/\delta_i$ is set to a larger value. However, the tightness of the bounds of $q(y)$ is not the only concern with respect to choosing the appropriate β value. We will discuss another effect to consider later.

As it has been shown that, with an appropriate σ_i/δ_i ratio, $g_i(\boldsymbol{x})$ defined in (3) is virtually a constant function, the next thing to do is to figure out the appropriate value of w_i that makes (3) satisfied. We have

$$\begin{aligned} g_i(\boldsymbol{s_i}) &= g_i(0, \ldots, 0) \\ &= w_i \left(\sum_{h_1=-\infty}^{\infty} \sum_{h_2=-\infty}^{\infty} \cdots \sum_{h_m=-\infty}^{\infty} \exp\left(-\frac{\left(h_1^2 + h_2^2 + \cdots + h_m^2\right)\delta_i^2}{2\sigma_i^2} \right) \right) \\ &= w_i \left(\sum_{h=-\infty}^{\infty} \exp\left(-\frac{h^2}{2\beta^2} \right) \right)^m \end{aligned} \tag{6}$$

where $\beta = \sigma_i/\delta_i$. Therefore, we need to set w_i as follows, in order to make $g_i(\boldsymbol{x})$ a good approximator of $f_j(\boldsymbol{x})$ in the proximity of $\boldsymbol{s_i}$:

$$w_i \left(\sum_{h=-\infty}^{\infty} \exp\left(-\frac{h^2}{2\beta^2} \right) \right)^m = f_j(\boldsymbol{s_i}).$$

If we employ (2) to estimate the value of $f_j(\boldsymbol{s_i})$, then we have

$$w_i = \frac{(k_1 + 1) \cdot \Gamma\left(\frac{m}{2} + 1\right)}{\lambda^m \cdot |S_j| \cdot \overline{R}(\boldsymbol{s_i})^m \cdot \pi^{\frac{m}{2}}} \quad \text{where}$$
$$\lambda = \sum_{h=-\infty}^{\infty} \exp\left(-\frac{h^2}{2\beta^2} \right). \tag{7}$$

So far, we have figured out that if we employ an appropriate ratio of $\beta = \sigma_i/\delta_i$ and set w_i according to (7), we can make $g_i(\boldsymbol{x})$ a good approximator of $f_j(\boldsymbol{x})$ in the proximity of $\boldsymbol{s_i}$. The only remaining issue is to derive a closed form of σ_i. In this paper, δ_i is set to the average distance between two adjacent class-j training samples in the proximity of sample $\boldsymbol{s_i}$. In an m-dimensional vector space, the number of uniformly distributed samples, N, in a hypercube with volume V can be computed by $N \cong (V/\alpha^m)$, where α is the spacing between two adjacent samples. Accordingly, we set

$$\delta_i = \frac{\overline{R}(\boldsymbol{s_i})\sqrt{\pi}}{\sqrt[m]{(k_1+1)\Gamma\left(\frac{m}{2}+1\right)}} \tag{8}$$

where $\overline{R}(\boldsymbol{s_i}) = (m+1/m)((1/k_1)\sum_{h=1}^{k_1} ||\hat{\boldsymbol{s}}_{\boldsymbol{h}} - \boldsymbol{s_i}||)$ as defined in Section III.

Finally, with (7) and (8) incorporated into (1), we have the following approximate probability density function for class-j training samples:

$$\begin{aligned}\hat{f}_j(\boldsymbol{v}) &= \sum_{\boldsymbol{s_i}\in S_j} w_i \exp\left(-\frac{||\boldsymbol{v}-\boldsymbol{s_i}||^2}{2\sigma_i^2}\right) \\ &= \sum_{\boldsymbol{s_i}\in S_j} \frac{(k_1+1)\cdot\Gamma\left(\frac{m}{2}+1\right)}{\lambda^m\cdot|S_j|\cdot\overline{R}(\boldsymbol{s_i})^m\cdot\pi^{\frac{m}{2}}} \exp\left(-\frac{||\boldsymbol{v}-\boldsymbol{s_i}||^2}{2\sigma_i^2}\right) \\ &= \frac{1}{|S_j|}\sum_{\boldsymbol{s_i}\in S_j}\left(\frac{\beta}{\lambda\cdot\sigma_i}\right)^m \exp\left(-\frac{||\boldsymbol{v}-\boldsymbol{s_i}||^2}{2\sigma_i^2}\right)\end{aligned} \tag{9}$$

where

$\boldsymbol{v}$
: a vector in the m-dimensional vector space;

S_j
: the set of class-j training samples;

$\sigma_i = \beta\delta_i = \beta(\overline{R}(\boldsymbol{s_i})\sqrt{\pi}/\sqrt[m]{(k_1+1)\Gamma((m/2)+1)})$

$\lambda = \sum_{h=-\infty}^{\infty}\exp(-h^2/2\beta^2)$.

In our study, we have observed that, as long as $\beta \geq 0.5$, then we have $\lambda = \sum_{h=-\infty}^{\infty}\exp(-h^2/2\beta^2) \cong \sqrt{2\pi}\cdot\beta$. If this approximation is adopted, then we can further simplify (9) and obtain

$$\hat{f}_j(\boldsymbol{v}) = \frac{1}{|S_j|}\sum_{\boldsymbol{s_i}\in S_j}\left(\frac{1}{\sqrt{2\pi}\cdot\sigma_i}\right)^m \exp\left(-\frac{||\boldsymbol{v}-\boldsymbol{s_i}||^2}{2\sigma_i^2}\right). \tag{10}$$

V. Implementation Issues and Analysis of Time Complexity

This section discusses the issues concerning implementation of the novel kernel density estimation algorithm proposed in the previous section and presents an analysis of time complexity. Fig. 2 summarizes the discussion so far by showing the detailed steps taken to create an SGF network-based data classifier and how the SGF network works. In procedure **make_classifier** presented in Fig. 2, it is assumed that the optimal values of the three parameters listed in Table II have been determined through cross validation. In the later part of this section, we will examine the cross validation issue.

With respect to the pseudocodes presented in Fig. 2, there are several practical issues to address. The first issue concerns the two ideal assumptions on which the derivation of (9) and (10) is based, i.e., the assumptions from which Fig. 1 is derived. If the target data set does not conform to these assumptions, then some sort of adaptations must be employed. The practice employed in this paper is to incorporate parameter $\hat{m}$ in Table II. In (2), (9), and (10), parameter m is supposed to be set to the number of attributes of the objects in the data set. However, because the local distributions of the training samples may not spread in all dimensions and some attributes may even be correlated, we replace m in these equations by $\hat{m}$, which is to be set through cross validation. In fact, the process conducted to figure out the optimal value of $\hat{m}$ also serves to tune w_i and σ_i, as we also replace m in (7) and (8) by $\hat{m}$.

Another parameter in Table II and Fig. 2 that needs to address is k_2. Since the influence of a Gaussian function decreases exponentially as the distance increases, when computing the values of the approximate probability density functions at a given vector $\boldsymbol{v}$ according to (9) or (10), we only need to include a limited number of nearest training samples of $\boldsymbol{v}$. The number of nearest training samples to be included can be determined through cross validation and is denoted by k_2.

There is one more practical issue to address. In earlier discussion, we mentioned that there is another aspect to consider in selecting the $\beta = \sigma_i/\delta_i$ ratio, in addition to the tightness of the bounds of function $q(y)$ defined in (4). If we examine (9) and (10), we will find that the value of the approximation function at a sample $\boldsymbol{s_i}$, i.e., $\hat{f}_j(\boldsymbol{s_i})$, is actually a weighted average of the estimated sample densities at $\boldsymbol{s_i}$ and at its nearby samples of the same class. Therefore, a smoothing effect will result. A larger $\beta = \sigma_i/\delta_i$ ratio implies that the smoothing effect will be more significant. Therefore, it is of interest to investigate the effect of β. Our experience suggests that, as long as β is set to a value within [0.6, 2], the value of β has no significant effect on classification accuracy. Therefore, β is not included in Table II.

As far as the time complexities of the algorithms presented in Fig. 2 are concerned, there are two separate issues. The first issue concerns the time taken to create an SGF network with n training samples and the second issue concerns the time taken to classify n' objects with the SGF network. In both issues, we need to identify the nearest neighbors of a sample. In our implementation, the kd-tree structure is employed [3], which is a data structure widely used to search for the nearest neighbors. With this practice, the average time complexity for constructing a kd-tree with n training samples is $O(n\log n)$. In procedure **make_classifier** presented in Fig. 2, we need to construct c kd-trees, if the training data set contains c classes of samples. Therefore, the average time complexity of this task is bounded by $O(cn\log n)$. Then, we need to identify the k_1 nearest neighbors for each of the n training samples and the average time complexity of this task is bounded by $O(k_1 n\log n)$. As the two tasks addressed above dominate the time complexity of procedure **make_classifier**, the overall time complexity for the procedure is $O(cn\log n + k_1 n\log n)$, or $O(n\log n)$, if both c and k_1 are regarded as constants.

In procedure **predict** presented in Fig. 2, the time complexity for classifying an incoming object is dominated by the work to identify k_2 nearest training samples of the incoming object. Therefore, the average time complexity for classifying

Procedure make_classifier
Input: a set of training samples $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$;
parameter values of k_1 and $\hat{m}$ listed in Table 2; parameter value of β.
Output: an SGF network.
Begin
for each class of training samples {
let S_j be the set of class-j training samples and construct a kd-tree for S_j;
for each $\boldsymbol{s}_i \in S_j$ {
let $\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \ldots, \hat{\boldsymbol{s}}_{k_1}$ be the k_1 nearest training samples of the same class as $\boldsymbol{s}_i$;
compute $\overline{R}(\boldsymbol{s}_i) = \frac{\hat{m}+1}{\hat{m}}\left(\frac{1}{k_1}\sum_{h=1}^{k_1} \| \hat{\boldsymbol{s}}_h - \boldsymbol{s}_i \|\right)$;
compute $\sigma_i = \beta \frac{\overline{R}(\boldsymbol{s}_i)\sqrt{\pi}}{\sqrt[\hat{m}]{(k_1+1)\Gamma(\frac{\hat{m}}{2}+1)}}$;
}
compute the approximate value of $\lambda = \sum_{h=-\infty}^{\infty} \exp\left(-\frac{h^2}{2\beta^2}\right)$;
construct an SGF sub-network with the following output function:
$$\hat{f}_j(\boldsymbol{v}) = \frac{1}{|S_j|}\sum_{\boldsymbol{s}_i \in S_j}\left(\frac{\beta}{\lambda \cdot \sigma_i}\right)^{\hat{m}} \exp\left(-\frac{\| \boldsymbol{v} - \boldsymbol{s}_i \|^2}{2\sigma_i^2}\right);$$
}
end

Procedure predict
Input: an SGF network constructed with Procedure **make_classifier**; parameter value of k_2 isted in Table 2; and an input object with coordinate $\boldsymbol{v}$;
Output: a prediction of the class of the input object;
Begin
let $\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \ldots, \hat{\boldsymbol{s}}_{k_2}$ be the k_2 nearest samples of $\boldsymbol{v}$ in the training data set;
max = 0;
for each SGF sub-network corresponding to one class of training samples {
let T_j be the subset of $\{\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \ldots, \hat{\boldsymbol{s}}_{k_2}\}$ that consists of class-j training samples;
compute the approximate value of $L_j(\boldsymbol{v}) = \frac{|S_j|}{|S|}\hat{f}_j(\boldsymbol{v})$ with
$$\hat{f}_j(\boldsymbol{v}) \cong \frac{1}{|S_j|}\sum_{\boldsymbol{s}_i \in T_j}\left(\frac{\beta}{\lambda \cdot \sigma_i}\right)^{\hat{m}} \exp\left(-\frac{\| \boldsymbol{v} - \boldsymbol{s}_i \|^2}{2\sigma_i^2}\right);$$
if ($L_j(\boldsymbol{v}) > max$) then {
class = j;
max = $L_j(v)$;
}
}
return (*class*);
end

Fig. 2. Pseudocodes of the proposed learning algorithm and the SGF network-based classifier.

TABLE II
PARAMETERS TO BE SET THROUGH CROSS VALIDATION FOR THE SGF NETWORK

k_1	The parameter in equation (2).
k_2	The number of nearest training samples included in evaluating the values of the approximate probability density functions at an input vector according to equation (9) or (10).
$\hat{m}$	The parameter that substitutes for m in equations (2) and (7)-(10).

one object is bounded by $O(k_2 \log n)$ and the overall time complexity for classifying n' incoming objects is bounded by $O(k_2 n' \log n)$ or $O(n' \log n)$, if k_2 is treated as a constant.

In the discussion above, it is assumed that the optimal values for the parameters listed in Table II have been determined through cross validation, before procedure **make_classifier** is invoked. If a k-fold cross validation process is conducted [32], then for each possible combination of parameter values we need to construct one SGF network-based on a subset of training samples. Then, we need to invoke procedure **predict** to figure out how good this particular combination of parameter values is. Based on the analysis of time complexity presented above, it is apparent that the average time complexity of the cross validation process is bounded by $O(n \log n)$, if the number of possible combinations of parameter values is regarded as a constant.

VI. EXPERIMENTAL RESULTS AND DISCUSSIONS

The experiments reported in this section have been conducted to evaluate the performance of the SGF networks constructed with the learning algorithm proposed in this paper,

TABLE III
BENCHMARK DATA SETS USED IN THE EXPERIMENTS. (A) THE THREE LARGER DATA SETS. (B) THE SIX SMALLER DATA SETS

(A).

Data set	# of training samples	# of testing samples
satimage	4435	2000
letter	15000	5000
shuttle	43500	14500

(B).

Data set	# of samples
Iris1	150
Wine	178
Vowel	528
Segment	2310
Glass	214
Vehicle	846

in comparison with the alternative data classification algorithms. The experiments focus on the following three issues: classification accuracy, execution efficiency, and the effect of data reduction. The alternative data classification algorithms involved in the comparison include SVM, K nearest neighbor (KNN) [32], and the conventional cluster-based learning algorithm proposed in [16] for RBF networks. The learning algorithm proposed in [16] conducts clustering analysis on the training data set and allocates one hidden unit for each cluster of training samples. For simplicity, in the following discussion, we will use the conventional RBF network to refer to the data classifier constructed with the learning algorithm proposed in [16] and the SGF network to refer to the data classifier constructed with the learning algorithm proposed in this paper. In these experiments, the SVM software used is LIBSVM [7] with the radial basis kernel and the one-against-one practice has been adopted for the SVM, if the data set contains more than two classes of objects.

Table III lists main characteristics of the nine benchmark data sets used in the experiments. All these data sets are from the UCI repository [6]. The collection of benchmark data sets is the same as that used in [15], except that **DNA** is not included. **DNA** is not included, because it contains categorical data and an extension of the proposed learning algorithm is yet to be developed for handling categorical data sets. Among the nine data sets, three of them are considered as the larger ones, as each contains more than 5000 samples with separate training and testing subsets. The remaining six data sets are considered as the smaller ones and there are no separate training and testing subsets in these six smaller data sets. Accordingly, different evaluation practices have been employed for the smaller data sets and for the larger data sets. For the three larger data sets, tenfold cross validation has been conducted on the training set to determine the optimal parameter values to be used in the testing phase. On the other hand, for the six smaller data sets, the evaluation practice employed in [15] has been adopted. With this practice, tenfold cross validation has been conducted on the entire data set and the best result is reported. Therefore, the results reported with this practice just reveal the maximum accuracy that can be achieved, provided that a perfect cross validation mechanism is available to identify the optimal parameter values.

In these experiments, β in (9) has been set to 0.7. Our observation in this regard is that, as long as β is set to a value within [0.6, 2], then the value of β has no significant effect on classification accuracy. On the other hand, parameters α and β in the conventional RBF network proposed in [16] have been set to the heuristic values suggested by the authors, i.e., 1.05 and 5, respectively.

Table IV compares the accuracy delivered by alternative classification algorithms with the three larger benchmark data sets. As Table IV shows, the SGF network and the SVM basically deliver the same level of accuracy, which the KNN and the conventional RBF network are generally not able to match. Table V lists the experimental results with the six smaller data sets. Table V shows that the SGF network and the SVM basically deliver the same level of accuracy for four out of these six data sets. The two exceptions are **glass** and **vehicle**. The results with these two data sets suggest that both the SGF network and the SVM have some blind spots and, therefore, may not be able to perform as well as the other in some cases. The experimental results presented in Table V also show that the SGF network and the SVM generally deliver a higher level of accuracy than the KNN and the conventional RBF network.

In the experiments that have been reported so far, no data reduction is performed in the construction of the SGF network. As the learning algorithm proposed in this paper places one SGF at each training sample, removal of redundant training samples means that the SGF network constructed will contain fewer hidden units and will operate more efficiently. Table VI presents the effect of applying a naïve data reduction algorithm to the three larger data sets. The naïve data reduction algorithm examines the training samples one by one in an arbitrary order. If the training sample being examined and all of its ten nearest neighbors in the remaining training data set belong to the same class, then the training sample being examined is considered as redundant and will be deleted. With this practice, training samples located near the boundaries between different classes of objects will be retained, while training samples located far away from the boundaries will be deleted. As shown in Table VI, the naïve data reduction algorithm is able to reduce the number of training samples in the **shuttle** data set substantially, with less than 2% of training samples left. On the other hand, the reduction rates for **satimage** and **letter** are not as substantial. It is apparent that the reduction rate is determined by the characteristics of the data set. Table VI also reveals that applying the naïve data reduction mechanism will lead to slightly lower classification accuracy. Since the data reduction mechanism employed in this paper is a naïve one, there is room for improvement with respect to both reduction rate and impact on classification accuracy. This is a subject under investigation.

Table VII compares the number of training samples remaining after data reduction is applied, the number of clusters identified by the conventional RBF network algorithm, and the number of support vectors identified by the SVM in the benchmark data sets. There are several interesting observations. First, for **satimage** and **letter**, the number of training samples

TABLE IV
COMPARISON OF CLASSIFICATION ACCURACY WITH THE THREE LARGER DATA SETS

Data sets	Data classification algorithms				
	SGF network	SVM	KNN with $k = 1$	KNN with $k = 3$	Conventional RBF network
1. satimage	92.30 ($k_1 = 6$, $k_2 = 26$, $\hat{m} = 1$)	91.30	88.80	90.65	90.25
2. letter	97.12 ($k_1 = 28$, $k_2 = 28$, $\hat{m} = 2$)	97.98	95.68	95.16	91.16
3. shuttle	99.94 ($k_1 = 18$, $k_2 = 1$, $\hat{m} = 3$)	99.92	99.94	99.91	97.34
Avg. 1-3	96.45	96.40	94.81	95.24	92.92

TABLE V
COMPARISON OF CLASSIFICATION ACCURACY WITH THE SIX SMALLER DATA SETS

Data sets	Data classification algorithms				
	SGF network	SVM	KNN with $k = 1$	KNN with $k = 3$	Conventional RBF network
1. iris	97.33 ($k_1 = 24$, $k_2 = 14$, $\hat{m} = 5$)	97.33	96.00	95.33	95.33
2. wine	99.44 ($k_1 = 3$, $k_2 = 16$, $\hat{m} = 1$)	99.44	95.52	96.07	98.89
3. vowel	99.62 ($k_1 = 15$, $k_2 = 1$, $\hat{m} = 1$)	99.05	99.62	97.35	93.37
4. segment	97.27 ($k_1 = 25$, $k_2 = 1$, $\hat{m} = 1$)	97.40	97.27	96.14	94.98
Avg. 1- 4	98.42	98.31	97.10	96.22	95.64
5. glass	75.74 ($k_1 = 9$, $k_2 = 3$, $\hat{m} = 2$)	71.50	72.01	72.01	69.16
6. vehicle	73.53 ($k_1 = 13$, $k_2 = 8$, $\hat{m} = 2$)	86.64	69.73	71.39	78.25
Avg. 1- 6	90.49	91.89	88.36	88.05	88.33

TABLE VI
EFFECTS OF APPLYING A NAÏVE DATA REDUCTION MECHANISM

	Satimage	letter	shuttle
# of training samples in the original data set	4435	15000	43500
# of training samples after data reduction is applied	1815	7794	627
% of training samples remaining	40.92%	51.96%	1.44%
Classification accuracy with the SGF network after data reduction is applied	92.15%	96.18%	99.32%
Degradation of accuracy due to data reduction	–0.15%	–0.94%	–0.62%

TABLE VII
COMPARISON OF THE NUMBER OF TRAINING SAMPLES REMAINING AFTER DATA REDUCTION IS APPLIED, THE NUMBER OF SUPPORT VECTORS IDENTIFIED BY THE SVM SOFTWARE, AND THE NUMBER OF CLUSTERS IDENTIFIED BY THE CONVENTIONAL RBF NETWORK ALGORITHM

	# of training samples after data reduction is applied	# of support vectors identified by LIBSVM	# of clusters identified by the conventional RBF network algorithm
satimage	1815	1689	322
letter	7794	8931	462
shuttle	627	287	45

remaining after data reduction is applied and the number of support vectors identified by the SVM are almost equal. For **shuttle**, though the difference is larger, the two numbers are still in the same order. On the other hand, the number of clusters identified by the conventional RBF network algorithm is consistently much smaller than the number of training samples remaining after data reduction is applied and the number of support vectors identified by the SVM. Our interpretation of these observations is that both the SVM and the naïve date reduction mechanism employed here attempt to identify the training samples that are located near the boundaries between different classes of objects. Therefore, the numbers with these two algorithms presented in Table VII are almost equal or at least in the same order. On the other hand, since multiple samples are needed to precisely describe the boundary of a cluster, the number of training samples remaining after data reduction is applied and the number of support vectors identified by the SVM are in general much larger than the number of clusters identified by the conventional RBF network algorithm. The results reported in Table VII along with the results presented in Table IV and Table V also suggest that, with respect to data classification, the distributions of samples near the boundaries between different classes of objects carry more crucial information than the distributions of samples in

TABLE VIII
COMPARISON OF EXECUTION TIMES IN SECONDS

		SGF network without data reduction	SGF network with data reduction	SVM	Conventional RBF network
make_ classifier	satimage	676	303	64644	136
	letter	2842	1990	387096	712
	shuttle	98540	773	467955	2595
predict	satimage	21.30	7.40	11.53	0.63
	letter	128.60	51.74	94.91	2.15
	shuttle	996.10	5.85	2.13	0.48

the inner part of the clusters. Since the conventional RBF network incorporates one RBF located at the geometric center of a cluster to model the distribution of the training samples inside the cluster, the accuracy delivered by the conventional RBF network is generally lower than that delivered by the SGF network and the SVM.

Table VIII compares the execution times of the SGF network, the SVM, and the conventional RBF network with the three larger data sets presented in Table III. In Table VIII, the total times taken to construct classifiers based on the given training data sets are listed in the rows marked by **make_classifier**. On the other hand, the times taken by alternative classifiers to predict the classes of the testing samples are listed in the rows marked by **predict**.

As Table VIII reveals, the time taken to construct an SVM classifier with the model-selection process employed in [7] is substantially higher than the time taken to construct an SGF network or a conventional RBF network. A detailed analysis reveals that it is the model-selection process that dominates the time taken to construct an SVM classifier. These results imply that the time taken to construct an SVM classifier with optimized parameter setting could be unacceptably long for some contemporary applications, in particular, for those applications in which new objects are continuously added into an already large database.

The results in Table VIII also imply that, in dealing with those data sets such as **satimage** and **letter** that do not contain a high percentage of redundant training samples, the SGF networks are favorable over the SVM. In such cases, the SGF networks enjoy substantially higher efficiency than the SVM in the **make_classifier** phase and are able to deliver the same level performance as the SVM in terms of both classification accuracy and the execution time in the **predict** phase. On the other hand, if the data set contains a high percentage of redundant training samples such as **shuttle**, then data reduction must be applied for the SGF network or its efficiency in the **predict** phase would suffer. With data reduction employed, the execution time of the SGF network in the **predict** phase then is comparable with that of the SVM. As the incorporation of the naïve data reduction mechanism may lead to slightly lower classification accuracy, it is of interest to develop advanced data reduction mechanisms.

Table VIII also shows that the conventional RBF networks generally enjoy higher efficiency in comparison with the SGF networks with data reduction and the SVM in the **predict** phase. This phenomenon is due to the fact shown in Table VII that the number of clusters identified by the conventional RBF network algorithm for a data set is generally smaller than the number of training samples employed to construct the SGF network after data reduction and the number of support vectors identified by the SVM algorithm. Nevertheless, as mentioned earlier, because the conventional cluster-based learning algorithm for RBF networks places one RBF at the center of each cluster, the distributions of the objects in the data set may not be accurately modeled. As a result, the conventional RBF networks in general are not able to deliver the same level of accuracy as the SVM and the SGF networks, which exploit the distributions of training samples near the boundaries between different classes of objects.

VII. CONCLUSION

In this paper, a novel learning algorithm for constructing SGF network-based data classifiers is proposed. With respect to algorithm design, the main distinction of the proposed learning algorithm is the novel kernel density estimation algorithm designed for efficient construction of the SGF networks. The experiments presented in this paper reveal that the SGF networks constructed with the proposed learning algorithm generally achieve the same level of classification accuracy as SVM. One important advantage of the proposed learning algorithm, in comparison with the SVM, is that the time taken to construct an SGF network with optimized parameter setting is normally much less that time taken to construct an SVM classifier. Another desirable feature of the SGF networks is that they can carry out data classification with more than two classes of objects in one single run. In other words, it does not need to resort to mechanisms such as one-against-one or one-against-all for handling datasets with more than two classes of objects. The other main properties of the proposed learning algorithm are:

1) the average time complexity for constructing an SGF network is bounded by $O(n \log n)$, where n is total number of training samples;
2) the average time complexity for classifying n' incoming objects is bounded by $O(n' \log n)$.

As the SGF networks constructed with the proposed learning algorithm are instance-based, this paper also addresses the efficiency issue shared by almost all instance-based learning algorithms. Experimental results reveal that the naïve data reduction mechanism employed in this paper is able to reduce the size of the training data set substantially with a slight impact on classification accuracy. One interesting observation in this regard is that, for all three data sets used in data reduction experiments, the numbers of training samples remaining after data reduction is applied are quite close to the numbers of support vectors identified by the SVM software.

In summary, the SGF network constructed with the proposed learning algorithm is favorable over the SVM in dealing with a data set that does not contain a high percentage of redundant training samples. In such case, the SGF network is able to deliver the same level of performance as the SVM in terms of both accuracy and the time taken in the prediction phase, while requiring substantially less time to construct the classifier. On the other hand, if the data set contains a high percentage of redundant training samples, then data reduction must be applied, or the execution time of the SGF network would suffer. As the incorporation of the naïve data reduction mechanism may lead to slightly lower classification accuracy, it is of interest to develop advanced data reduction mechanisms. This paper also compares the performance of the SGF networks constructed with the proposed learning algorithm and the RBF networks constructed with a conventional cluster-based learning algorithm. The most interesting observation learned is that, with respect to data classification, the distributions of training samples near the boundaries between different classes of objects carry more crucial information than the distributions of samples inside the clusters. As a result, the conventional RBF networks generally are not able to deliver the same level of accuracy as those learning algorithms such as SVM and the SGF networks that exploit the distributions of training samples near the boundaries between different classes of objects.

Based on the study presented in this paper, there are several issues that deserve further studies, in addition to the development of advanced data reduction mechanisms mentioned above. One issue is the extension of the proposed learning algorithm for handling categorical data sets. Another issue concerns why the SGF network fails to deliver comparable accuracy in the **vehicle** test case, what the blind spot is, and how improvements can be made. Finally, it is of interest to develop incremental version of the proposed learning algorithm to cope with the ever-growing contemporary databases.

APPENDIX A

Assume that $\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \ldots, \hat{\boldsymbol{s}}_{k_1}$ are the k_1 nearest training samples of $\boldsymbol{s_i}$ that belongs to the same class as $\boldsymbol{s_i}$. If k_1 is sufficiently large and the distribution of these k_1 samples in the vector space is uniform then we have

$$k_1 \approx \frac{\rho R(\boldsymbol{s_i})^m \pi^{\frac{m}{2}}}{\Gamma\left(\frac{m}{2}+1\right)}$$

where ρ is the local density of samples $\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \ldots, \hat{\boldsymbol{s}}_{k_1}$ in the proximity of $\boldsymbol{s_i}$. Furthermore, we have

$$\sum_{h=1}^{k_1} \|\hat{\boldsymbol{s}}_h - \boldsymbol{s_i}\| \approx \int_0^{R(s_i)} \rho \left(\frac{2r^{m-1}\pi^{\frac{m}{2}}}{\Gamma\left(\frac{m}{2}\right)} \right) r dr = \frac{2\rho R(\boldsymbol{s_i})^{m+1}\pi^{\frac{m}{2}}}{(m+1)\Gamma\left(\frac{m}{2}\right)}$$

where $(2r^{m-1}\pi^{m/2})/(\Gamma(m/2))$ is the surface area of a hypersphere with radius r in an m-dimensional vector space. Therefore, we have

$$R(\boldsymbol{s_i}) = \frac{m+1}{m} \cdot \frac{1}{k_1} \sum_{h=1}^{k_1} \|\hat{\boldsymbol{s}}_h - \boldsymbol{s_i}\|.$$

The right-hand side of the equation above is then employed in this paper to estimate $R(\boldsymbol{s_i})$.

APPENDIX B

Let $q(y) = \sum_{h=-\infty}^{\infty} \exp(-(y-h\delta)^2/2\sigma^2)$, where $\delta \in \boldsymbol{R}$ and $\sigma \in \boldsymbol{R}$ are two coefficients and $y \in \boldsymbol{R}$. We have

$$q'(y) = \left(-\frac{1}{\sigma^2}\right) \sum_{h=-\infty}^{\infty} (y-h\delta) \exp\left(-\frac{(y-h\delta)^2}{2\sigma^2}\right).$$

Since $q(y)$ is a symmetric and periodical function, if we want to find the global maximum and minimum values of $q(y)$, we only need to analyze $q(y)$ within interval $[0, \delta/2]$. Let $y_0 \in [0, (\delta/2))$ and $y_0 = (\delta/2)\cdot(j/n) + \varepsilon$, where $n \geq 1$ and $0 \leq j \leq n-1$ are integers, and $0 \leq \varepsilon < (\delta/2n)$. We have

$$q(y_0) = q\left(\frac{j\delta}{2n}\right) + \int_{\frac{j\delta}{2n}}^{\frac{j\delta}{2n}+\varepsilon} q'(t)dt.$$

Let us consider the special case with $\sigma = \delta$. Then, we have

$$q(y_0) = \sum_{h=-\infty}^{\infty} \left[\exp\left(-\frac{1}{2}\left(\frac{j}{2n}-h\right)^2\right) - \frac{1}{\sigma^2} \int_{\frac{j\delta}{2n}}^{\frac{j\delta}{2n}+\varepsilon} (t-h\delta) \exp\left(-\frac{(t-h\delta)^2}{2\sigma^2}\right) dt \right].$$

Let $r(h) = -(1/\sigma^2)\int_{j\delta/2n}^{(j\delta/2n)+\varepsilon}(t-h\delta)\exp(-(t-h\delta)^2/2\sigma^2)dt$. Since $(-1/\sigma^2)(t-h\delta)\exp(-(t-h\delta)^2/2\sigma^2)$ is a decreasing function for $t \in [(h-1)\delta, (h+1)\delta]$ and is an increasing function for $t \notin [(h-1)\delta, (h+1)\delta]$, we have

1)
$$r(0) \leq \varepsilon\left(\frac{-1}{\sigma^2}\right)\left(\frac{j\delta}{2n}\right)\exp\left[-\frac{1}{2\sigma^2}\left(\frac{j\delta}{2n}\right)^2\right] = \varepsilon\left(\frac{-1}{\sigma}\right)\left(\frac{j}{2n}\right)\exp\left[-\frac{1}{2}\left(\frac{j}{2n}\right)^2\right];$$

2)
$$r(1) \leq \varepsilon\left(\frac{-1}{\sigma^2}\right)\left(\frac{j\delta}{2n}-\delta\right)\exp\left[-\frac{1}{2\sigma^2}\left(\frac{j\delta}{2n}-\delta\right)^2\right] = \varepsilon\left(\frac{-1}{\sigma}\right)\left(\frac{j}{2n}-1\right)\exp\left[-\frac{1}{2}\left(\frac{j}{2n}-1\right)^2\right];$$

3) for $h \neq 0$ and $h \neq 1$,

$$r(h) \leq \varepsilon\left(\frac{-1}{\sigma^2}\right)\left(\frac{(j+1)\delta}{2n}-h\delta\right)\exp\left[-\frac{1}{2\sigma^2}\left(\frac{(j+1)\delta}{2n}-h\delta\right)^2\right] = \varepsilon\left(\frac{-1}{\sigma}\right)\left(\frac{(j+1)}{2n}-h\right)\exp\left[-\frac{1}{2}\left(\frac{(j+1)}{2n}-h\right)^2\right].$$

Therefore

$$q(y_0) = \sum_{h=-\infty}^{\infty} \left[\exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) + r(h) \right] \le \left[\sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) \right] + \varepsilon\theta$$

where

$$\theta = \left(\frac{-1}{\sigma}\right)\left(\frac{j}{2n}\right)\exp\left[-\frac{1}{2}\left(\frac{j}{2n}\right)^2\right] + \left(\frac{-1}{\sigma}\right)\left(\frac{j}{2n}-1\right)\exp\left[-\frac{1}{2}\left(\frac{j}{2n}-1\right)^2\right] + \left(\frac{-1}{\sigma}\right)\sum_{\substack{h=-\infty \\ h\neq 0,1}}^{\infty}\left(\frac{(j+1)}{2n}-h\right)\exp\left[-\frac{1}{2}\left(\frac{(j+1)}{2n}-h\right)^2\right].$$

If $\theta \ge 0$, then we have for any $0 \le \varepsilon < (\delta/2n)$

$$\left[\sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) \right] + \varepsilon\theta \le \left[\sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) \right] + \frac{\delta}{2n}\theta. \quad \text{(A.1)}$$

On the other hand, if $\theta < 0$, then we have for any $0 \le \varepsilon < (\delta/2n)$

$$\left[\sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) \right] + \varepsilon\theta \le \left[\sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) \right]. \quad \text{(A.2)}$$

Combining (A.1) and (A.2), we obtain, for all $y \in [0, (\delta/2)]$

$$q(y) \le \lim_{n\to\infty} \operatorname{Maximum}_{0\le j\le n-1} \left\{ \sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right), \sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) + \frac{\delta}{2n}\theta \right\}.$$

Similarly, we can show that

$$q(y) \ge \lim_{n\to\infty} \operatorname{Minimum}_{0\le j\le n-1} \left\{ \sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right), \sum_{h=-\infty}^{\infty} \exp\left(-\frac{1}{2}\left(\frac{j}{2n} - h \right)^2 \right) + \frac{\delta}{2n}\rho \right\}$$

where

$$\rho = \left(\frac{-1}{\sigma}\right)\left(\frac{j+1}{2n}\right)\exp\left[-\frac{1}{2}\left(\frac{j+1}{2n}\right)^2\right] + \left(\frac{-1}{\sigma}\right)\left(\frac{j+1}{2n}-1\right)\exp\left[-\frac{1}{2}\left(\frac{j+1}{2n}-1\right)^2\right] + \left(\frac{-1}{\sigma}\right)\sum_{\substack{h=-\infty \\ h\neq 0,1}}^{\infty}\left(\frac{j}{2n}-h\right)\exp\left[-\frac{1}{2}\left(\frac{j}{2n}-h\right)^2\right].$$

If we set $n = 100\,000$, then we have, with $\sigma = \delta$, $2.506\,628\,261 \le q(y) \le 2.506\,628\,288$, for $y \in [0, (\delta/2)]$.

REFERENCES

[1] E. Artin, *The Gamma Function*. New York: Holt, Rinehart Winston, 1964.

[2] F. Belloir, A. Fache, and A. Billat, "A general approach to construct RBF net-based classifier," in *Proc. 7th Eur. Symp. Artificial Neural Network*, 1999, pp. 399–404.

[3] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[4] M. Bianchini, P. Frasconi, and M. Gori, "Learning without local minima in radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 749–756, May 1995.

[5] C. M. Bishop, "Improving the generalization properties of radial basis function neural networks," *Neural Computat.*, vol. 3, no. 4, pp. 579–588, 1991.

[6] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," Dept. Inform. Comput. Sci., Univ. California, Irvine, CA, 1998.

[7] C. C. Chang and C. J. Lin. (2001) LIBSVM: A library for support vector machines. [Online] Available http://www.csie.ntu. edu.tw/~cjlin/libsvm

[8] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Mach. Learn.*, vol. 46, no. 1, pp. 131–159, 2002.

[9] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[10] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[11] D. Decoste and K. Wagstaff, "Alpha seeding for support vector machines," in *Proc. Int. Conf. Knowledge Discovery and Data Mining*, 2000, pp. 345–349.

[12] K. Duan, S. S. Keerthi, and A. N. Poo, "Evaluation of simple performance measures for tuning SVM hyperparameters," *Neurocomput.*, vol. 51, pp. 41–59, 2003.

[13] S. Dumais, J. Platt, and D. Heckerman, "Inductive learning algorithms and representations for text categorization," in *Proc. Int. Conf. Information and Knowledge Management*, 1998, pp. 148–154.

[14] G. W. Flake, "Square unit augmented, radially extended, multilayer perceptrons," *Neural Networks: Tricks of the Trade*, pp. 145–163, 1998.

[15] C. W. Hsu and C. J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

[16] Y. S. Hwang and S. Y. Bang, "An efficient method to construct a radial basis function neural network classifier," *Neural Netw.*, vol. 10, no. 8, pp. 1495–1503, 1997.

[17] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proc. Eur. Conf. Machine Learning*, 1998, pp. 137–142.

[18] V. Kecman, *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. Cambridge, MA: MIT Press, 2001.

[19] S. S. Keerthi, "Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1225–1229, Sep. 2002.

[20] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[21] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computat.*, vol. 1, no. 2, pp. 281–294, 1989.

[22] M. Musavi, W. Ahmed, K. Chan, K. Faris, and D. Hummels, "On the training of radial basis function classifiers," *Neural Netw.*, vol. 5, no. 4, pp. 595–603, 1992.

[23] M. J. L. Orr, "Regularization in the selection of radial basis function centres," *Neural Computat.*, vol. 7, no. 3, pp. 606–623, 1995.

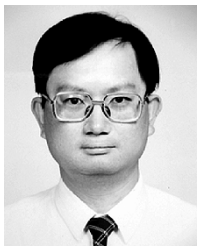
[24] M. J. L. Orr, "Introduction to radial basis function networks," Center Cognitive Sci., Univ. Edinburgh, Edinburgh, U.K., 1996.

[25] M. J. Orr, "Optimising the widths of radial basis function," in *Proc. 5th Brazilian Symp. Neural Networks*, 1998, pp. 26–29.

[26] M. J. Orr, J. Hallam, A. Murray, and T. Leonard, "Assessing rbf networks using delve," *Int. J. Neural Syst.*, vol. 10, no. 5, pp. 397–415, 2000.

[27] Y. J. Oyang, S. C. Hwang, Y. Y. Ou, C. Y. Chen, and Z. W. Chen, "A novel learning algorithm for data classification with radial basis function networks," in *Proc. 9th Int. Conf. Neural Information Processing*, 2002, pp. 1021–1026.

[28] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1991.
[29] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.
[30] M. J. Powell, "Radial basis functions for multivariable interpolation: a review," in *Algorithm for Approximation*, J. C. Mason and M. G. Cox, Eds. Oxford, U. K.: Oxford Univ. Press, 1987, pp. 143–167.
[31] B. Scholkopf, K. K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 1–8, Nov. 1997.
[32] I. H. Witten and E. Frank, *Data Mining*. San Mateo, CA: Morgan Kaufmann, 1999.



Yen-Jen Oyang (M'88) received the B.S. degree in information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1982, the M.S. degree in computer science from the California Institute of Technology, Pasadena, in 1984, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1988.

He is currently a Professor in the Department of Computer Science and Information Engineering, National Taiwan University. His research interests include data mining/machine learning and bioinformatics.

Shien-Ching Hwang received the B.S. degree in mathematics from Fu Jen Catholic University, Taiwan, R.O.C., in 1993, and the M.S. and Ph.D. degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1995 and 2000, respectively.

He is currently a Postdoctoral Researcher at the Department of Computer Science and Information Engineering, National Taiwan University. His research interests include machine learning, data mining, and bioinformatics.

Yu-Yen Ou (SM'03) is working towards the Ph.D. degree in computer science and information engineering department, the National Taiwan University, Taipei, Taiwan, R.O.C.

His research interests include machine learning, data mining, and bioinformatics.

Chien-Yu Chen (M'04) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1996, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1998, and the Ph.D. degree in computer science and information engineering from National Taiwan University in 2003.

She is currently an Assistant Professor in the Graduate School of Biotechnology and Bioinformatics, Yuan Ze University, Chung-Li, Taiwan, R.O.C. Her research interests include bioinformatics, data mining, and machine learning.

Zhi-Wei Chen received the B.S. degree from National Chiao Tung University, Taiwan, R.O.C., and the M.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1999 and 2002, respectively.

He is currently an Engineer in the Advance Research Department, Quanta Computer Incorporated, Taipei, Taiwan, R.O.C.