Localized Generalization Error Model and Its Application to Architecture Selection for Radial Basis Function Neural Network

Daniel S. Yeung, *Fellow, IEEE*, Wing W. Y. Ng, *Member, IEEE*, Defeng Wang, Eric C. C. Tsang, *Member, IEEE*, and Xi-Zhao Wang, *Senior Member, IEEE*

***Abstract*—The generalization error bounds found by current error models using the number of effective parameters of a classifier and the number of training samples are usually very loose. These bounds are intended for the entire input space. However, support vector machine (SVM), radial basis function neural network (RBFNN), and multilayer perceptron neural network (MLPNN) are local learning machines for solving problems and treat unseen samples near the training samples to be more important. In this paper, we propose a localized generalization error model which bounds from above the generalization error within a neighborhood of the training samples using stochastic sensitivity measure. It is then used to develop an architecture selection technique for a classifier with maximal coverage of unseen samples by specifying a generalization error threshold. Experiments using 17 University of California at Irvine (UCI) data sets show that, in comparison with cross validation (CV), sequential learning, and two other ad hoc methods, our technique consistently yields the best testing classification accuracy with fewer hidden neurons and less training time.**

***Index Terms*—Localized generalization error, network architecture selection, radial basis function neural network (RBFNN), sensitivity measure.**

I. Introduction

FOR a pattern classification problem, one builds a classifier f_θ to approximate or mimic the unknown input–output mapping function F, where θ is the set of parameters selected from a domain Λ. In this paper, the mean square error (MSE) is used to measure the difference between f_θ and F. The MSE is widely applied in training real-value output classifiers like neural networks, which classify a given sample by thresholding the real-value classifier output. The behavior of a classifier trained by minimizing an MSE and the one trained by minimizing a classification error (0–1 loss function) are different. When two classifiers both yield the same, very small percentage of training classification error, the one which yields a larger training MSE would produce indecisive outputs which are more deviated from the target outputs; so a small change in the inputs may change the classification results [20]. This is not desirable and it indicates that the training classification error is not a good benchmark for the generalization capability of a classifier. Therefore, selecting a classifier using training classification error or its bound may not be appropriate.

The classification error for the entire input space is defined as

$$R_{\text{true}} = \int_T (f_\theta(\mathbf{x}) - F(\mathbf{x}))^2 p(\mathbf{x})\, d\mathbf{x} \tag{1}$$

where $\mathbf{x}$ denotes the input vector of a sample in the entire input space T, and $p(\mathbf{x})$ denotes the true unknown probability density function of the input $\mathbf{x}$.

Given a training data set D containing N training input–output pairs, $D = \{(\mathbf{x}_b, F(\mathbf{x}_b))\}_{b=1}^N$, a classifier f_θ could be constructed by minimizing the empirical risk (R_{emp}) over D, where

$$R_{\text{emp}} = \frac{1}{N}\sum_{b=1}^{N} (f_\theta(\mathbf{x}_b) - F(\mathbf{x}_b))^2. \tag{2}$$

The ultimate goal of solving a pattern classification problem is to find f_θ which is able to correctly classify future unseen samples [6], [10], [17]. The generalization error (R_{gen}) is defined as

$$R_{\text{gen}} = \int_{T\setminus D} (f_\theta(\mathbf{x}) - F(\mathbf{x}))^2 p(\mathbf{x})\, d\mathbf{x}. \tag{3}$$

Since both target outputs and distributions of the unseen samples are unknown, it is impossible to compute the R_{gen} directly. There are two major approaches to estimate the R_{gen}, namely, analytical model and cross validation (CV).

In general, analytical models could not distinguish trained classifiers having the same number of effective parameters but with different values of parameters. Thus, they yield loose error bounds [5]. Akaike information criterion (AIC) [1] only makes use of the number of effective parameters and the number of training samples. Network information criterion (NIC) [15] is an extension of AIC for application in regularized neural networks. It defines a classifier complexity by the trace of the

Manuscript received September 14, 2005; revised March 2, 2006 and December 24, 2006; accepted December 26, 2006. This paper was supported in part by the Hong Kong Research Grant Council under Project B-Q571, in part by the Hong Kong Polytechnic University Department of Computing Personal Research Account, and in part by the Hong Kong Polytechnic University Interfaculty under Research Grant GYD87.

D. S. Yeung and W. W. Y. Ng are with the Media and Life Science (MiLeS), Department of Computer Science and Technology, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China and with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong(e-mail: csdaniel@comp.polyu.edu.hk; wingng@ieee.org).

D. Wang and E. C. C. Tsang are with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong.

X.-Z. Wang is with the Machine Learning Center, Faculty of Mathematics and Computer Science, Hebei University, Baoding 071002, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.894058

second-order local derivatives of the network outputs with respect to connection weights. Unfortunately, current analytical models are only good for linear classifiers due to the singularity problem in their models [16]. A major problem with these models is the difficulty in estimating the number of effective parameters of the classifier and this could be solved by using the Vapnik–Chervonenkis (VC)-dimensions [17]. However, only loose bounds of VC-dimensions could be found for nonlinear classifiers and this puts a severe limitation on the applicability of analytical models to nonlinear classifiers, except for the support vector machine (SVM) [3], [4], [5].

Although CV uses true target outputs for unseen samples, it is time consuming for large data sets and, for k-fold CV and L choices of classifier parameters, kL classifiers must be trained. CV methods estimate the expected generalization error instead of its bound. Thus, they cannot guarantee the classifier finally constructed to have good generalization capability [5].

Many classifiers, e.g., SVM, multilayer perceptron neural networks (MLPNNs), and radial basis function neural networks (RBFNNs), are local learning machines. RBFNN, by its nature, learns the classification locally and every hidden neuron captures the local information of a particular region in the input space defined by the center and width of its Gaussian activation function [11]. A training sample located far away from a hidden neuron's center does not affect the learning of this hidden neuron [8]. MLPNN learns the decision boundaries for the classification problem in the input space using the location of the training samples. However, as pointed out in [2], the MLPNN output responses to unseen samples far away from the training samples are likely to be unreliable, so they proposed a learning algorithm which deactivates any MLPNN response to unseen samples much different from the training samples. This observation is further supported by the fact that, in many interesting industrial applications, such as aircraft detection in synthetic aperture radar (SAR) images, character and fingerprint recognitions [9], etc., the most significant unseen samples are expected to be similar to the training samples.

On the other hand, RBFNN is one of the most widely applied neural networks for pattern classification, with its performance primarily determined by its architecture selection. Reference [6] summarizes several training algorithms for RBFNN. For instance, a two-stage learning algorithm may be a quick way to train an RBFNN. The first, unsupervised stage is to select the center positions and widths for the RBF using self-organizing map or k-means clustering. The second, supervised stage computes the connection weights using least mean square method or pseudoinverse technique. Some proposed that all training samples may be selected as centers [6], [22]. In [18], a reformulated RBFNN was proposed which was trained by using a gradient-descent method for its parameters. In [26], the selection of centers is based on the separability of the data sets. The experimental results in [18] and [26] indicate that the choice of the number of hidden neurons indeed affect the generalization capability of the RBFNNs and an increase in the number of hidden neurons does not necessarily lead to a decrease in testing error. In [23] and [24], the optimal architecture was found by sequentially adding more hidden neurons to a small RBFNN and, in [25], genetic algorithm was used to search for the optimal number of hidden neurons, center position, and width, simultaneously. Thus, the selection of the number of hidden neurons affects the selection of RBFNN architecture and ad hoc choices or sequential search are the frequently used methods.

In this paper, we propose a localized generalization error model R_{SM} using the stochastic sensitivity measure (ST-SM), which bounds from above the generalization error for unseen samples within a predefined neighborhood of the training samples. In addition, an architecture selection method based on the R_{SM} is proposed to find the maximal coverage classifier with its R_{SM} bounded by a preselected threshold. RBFNN will be used to demonstrate the use of the R_{SM} and the architecture selection method.

We introduce the localized generalization error model and its corresponding architecture selection method in Sections II and III, respectively. In Section IV, experimental results of the architecture selection will be presented. We conclude this paper in Section V.

II. Localized Generalization Error Model

Two major concepts of R_{SM}, the Q-neighborhood and stochastic sensitivity measure, are introduced in Sections II-A and II-C, respectively. The derivation of the localized generalization error model is given in Section II-B and its characteristics are discussed in Section II-D. Section II-E discusses the method to compare two classifiers using the localized generalization error model.

A. Q-Neighborhood and Q-Union

For every sample $\mathbf{x}_b \in D$, one finds a set of samples $\mathbf{x}$ which fulfills $0 < |\Delta x_i| < Q \forall i = 1, \ldots, n$, where n denotes the number of input features, $\Delta\mathbf{x} = (\Delta x_1, \ldots, \Delta x_n)' = \mathbf{x} - \mathbf{x}_b$, and Q is a given real number. In pattern classification problem, one usually does not have any knowledge about the distribution of the true input space. Therefore, without any prior knowledge, every unseen sample has the same chance to appear; so, $\Delta\mathbf{x}$ may be considered as input perturbations which are random variables having zero mean uniform distributions

$$S_Q(\mathbf{x}_b) = \{\mathbf{x} \,|\, \mathbf{x} = \mathbf{x}_b + \Delta\mathbf{x}; |\Delta x_i| \le Q \qquad \forall i = 1, \ldots, n\}. \tag{4}$$

Then, $S_Q(\mathbf{x}_b)$ defines a Q-neighborhood of the training sample $\mathbf{x}_b$. Let S_Q be the union of all $S_Q(\mathbf{x}_b)$ and call it the Q-union. All samples in $S_Q(\mathbf{x}_b)$, except $\mathbf{x}_b$, are considered as unseen samples (i.e., $S_Q(\mathbf{x}_b)$ contains no training point other than $\mathbf{x}_b$).

For $0 \le Q_1 \le \cdots \le Q_k \le \infty$, the following relationship holds:

$$D \subseteq S_{Q_1} \subseteq \cdots \subseteq S_{Q_k} \subseteq T. \tag{5}$$

One should note that the shape of the Q-neighborhood is chosen to be a hypercube for ease of computation, but it could also be a hypersphere or other shapes. Moreover, in the localized generalization error model, the unseen samples could be selected from a distribution other than a uniform one. Only the

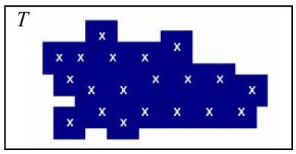

Fig. 1. Illustration of Q-union (S_Q) with 20 training samples. The Xs are training samples and any point in the shaded area is an unseen sample.

derivation of the ST-SM needs to be modified and the rest of the paper will remain the same.

B. Derivation of the Localized Generalization Error Bound (R^*_{SM})

Instead of finding a bound for the generalization error for unseen samples in the entire input space $T(R_{\mathrm{true}})$, we find a bound on R_{SM}, which is the error for unseen samples within S_Q only, i.e., the shaded area in Fig. 1. We ignore the generalization error for unseen samples which are located far away from training samples [R_{res} in (6)]. Note that R_{res} decreases when Q increases

$$R_{\mathrm{SM}}(Q)=R_{\mathrm{true}}-R_{\mathrm{res}}(Q)=\int_{S_Q}(f_\theta(\mathbf{x})-F(\mathbf{x}))^2p(\mathbf{x})\,d\mathbf{x}. \tag{6}$$

Let $\Delta y = f_\theta(\mathbf{x}) - f_\theta(\mathbf{x}_b)$, $\mathrm{err}_\theta(\mathbf{x}_b) = f_\theta(\mathbf{x}_b) - F(\mathbf{x}_b)$, then $R_{\mathrm{emp}} = (1/N)\sum_{b=1}^{N}(\mathrm{err}_\theta(\mathbf{x}_b))^2$, $E_{S_Q}((\Delta y)^2) = (1/N)\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}(\Delta y)^2 1/(2Q)^n d\mathbf{x}$, $\varepsilon = B\sqrt{\ln\eta/(-2N)}$, and A, B, and N be the difference between the maximum and minimum values of the target outputs, the maximum possible value of the MSE, and the number of training samples, respectively. In this paper, we assume the range of the desired output, i.e., the range of F, to be either known or assigned to a preselected value. Moreover, B is computable because the range of the network outputs will be known after the classifier is trained. In general, one expects that the error of unseen samples will be larger than the training error, so we assume that the average of errors of unseen samples in $S_Q(\mathbf{x}_b)$ will be larger than the training error of $\mathbf{x}_b$. By the Hoeffding's inequality [7], the average of the square errors of samples with the same population mean converges to the true mean with the following rate of convergence. With a probability of$(1-\eta)$, we have (7), as shown at the bottom of the page.

Both A and ε are constants for a given training data set when an upper bound of the classifier output values is preselected. The R^*_{SM} is an upper bound for the MSE of the trained classifier for unseen samples within the Q-union. This bound is better than those regression error bounds (based on AIC and VC-dimension) which are defined by using only the number of effective parameters and training samples while ignoring statistical characteristics of training data set such as their mean and variance. Moreover, those error bounds usually grow quickly with the increase of the number of effective parameters, e.g., number of hidden neurons in RBFNN and VC-dimension, while the R^*_{SM} grows much slower.

The term $E_{S_Q}((\Delta y)^2)$ will be discussed in Section II-C. Further discussion on the characteristics of the R^*_{SM} will be given in Section II-D.

C. Stochastic Sensitivity Measure for RBFNN

The output perturbation (Δy) measures the network output difference between the training sample $(\mathbf{x}_b \in D)$ and the unseen sample in its Q-neighborhood $((\mathbf{x}_b + \Delta\mathbf{x}) \in S_Q(\mathbf{x}_b))$. Thus, the ST-SM measures the expectation of the squares of network

$$\begin{aligned}
R_{\mathrm{SM}}(Q) &\leqslant \frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}(f_\theta(\mathbf{x})-F(\mathbf{x}))^2\frac{1}{(2Q)^n}\,d\mathbf{x}+\varepsilon\\
&= \frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}(f_\theta(\mathbf{x})-f_\theta(\mathbf{x}_b)+f_\theta(\mathbf{x}_b)-F(\mathbf{x}_b)+F(\mathbf{x}_b)-F(\mathbf{x}))^2\frac{1}{(2Q)^n}\,d\mathbf{x}+\varepsilon\\
&\leqslant \frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((\Delta y)^2)\frac{1}{(2Q)^n}\,d\mathbf{x}+\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((\mathrm{err}_\theta(\mathbf{x}_b))^2)\frac{1}{(2Q)^n}\,d\mathbf{x}\\
&\quad+\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((F(\mathbf{x}_b)-F(\mathbf{x}))^2)\frac{1}{(2Q)^n}\,d\mathbf{x}+\varepsilon\\
&\quad+2\sqrt{\left(\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((\Delta y)^2)\frac{1}{(2Q)^n}\,d\mathbf{x}\right)\left(\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((\mathrm{err}_\theta(\mathbf{x}_b))^2)\frac{1}{(2Q)^n}\,d\mathbf{x}\right)}\\
&\quad+2\sqrt{\left(\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((\mathrm{err}_\theta(\mathbf{x}_b))^2)\frac{1}{(2Q)^n}\,d\mathbf{x}\right)\left(\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((F(\mathbf{x}_b)-F(\mathbf{x}))^2)\frac{1}{(2Q)^n}\,d\mathbf{x}\right)}\\
&\quad+2\sqrt{\left(\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((\Delta y)^2)\frac{1}{(2Q)^n}\,d\mathbf{x}\right)\left(\frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)}((F(\mathbf{x}_b)-F(\mathbf{x}))^2)\frac{1}{(2Q)^n}\,d\mathbf{x}\right)}\\
&\leqslant (\sqrt{R_{\mathrm{emp}}}+\sqrt{E_{S_Q}((\Delta y)^2)}+A)^2+\varepsilon\\
&= R^*_{\mathrm{SM}}(Q)
\end{aligned} \tag{7}$$

output perturbations (Δy) between training samples in D and unseen samples in S_Q.

The sensitivity measure (SM) of a neural network [12]–[14] gives a quantified data on the change of network outputs with respect to change of network inputs. Intuitively, it measures how sensitive the network output is to the input change. In [14], every input or weight is allowed to have its own mean and variance, and the input and weight perturbations are allowed to be arbitrary. Hence, the perturbed samples ($\mathbf{x}$ in Section II-A) can be considered as unseen samples around the training samples ($\mathbf{x}_b$). An analytical formula of the ST-SM for a Gaussian activation function RBFNN was developed in [12], which is independent of the number of training samples. We assume the inputs are independent and not identically distributed and weight perturbations are not considered in this paper; so, every input feature has its own expectation μ_{x_i} and variance $\sigma^2_{x_i}$. The input perturbation of the ith input feature is a random variable having a uniform distribution with zero mean and a variance $\sigma^2_{\Delta x_i}$. The centers and widths of the hidden neurons are constant and the connection weights are fixed beforehand. An RBFNN could be described as

$$f_\theta(\mathbf{x}) = \sum_{j=1}^{M} w_j \exp\left(\frac{\|\mathbf{x}-\mathbf{u}_j\|^2}{-2v_j^2}\right) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) \tag{8}$$

where M, $\mathbf{u}_j$, and v_j denote the number of hidden neurons, the center, and width of the jth RBFNN hidden neuron, respectively, and w_j denotes the connection weight between the jth hidden neuron and its corresponding output neuron. Let $\varphi_j = (w_j)^2 \exp\left(\left(\mathrm{Var}(s_j)/2v_j^4\right) - \left(E(s_j)/v_j^2\right)\right)$, $s_j = \|\mathbf{x}-\mathbf{u}_j\|^2$, $E(s_j) = \sum_{i=1}^{n}\left(\sigma^2_{x_i} + (\mu_{x_i} - u_{ji})^2\right)$, $\zeta_j = \varphi_j / v_j^4$

$$\mathrm{var}(s_j) = \sum_{i=1}^{n}\begin{pmatrix} E_D[(x_i-\mu_{x_i})^4] - \left(\sigma^2_{x_i}\right)^2 + 4\sigma^2_{x_i}(\mu_{x_i}-u_{ji})^2 \\ +4E_D\left[(x_i-\mu_{x_i})^3\right](\mu_{x_i}-u_{ji}) \end{pmatrix}$$

$$v_j = \varphi_j\left(\sum_{i=1}^{n}\left(\sigma^2_{x_i} + (\mu_{x_i}-u_{ji})^2\right)/v_j^4\right)$$

$p(\Delta\mathbf{x})$ denotes the probability density function of the input perturbations and $p(\Delta\mathbf{x}) = 1/(2Q)^n$, n is the number of input features and u_{ji} denotes the ith input feature of the jth center of the hidden RBF neuron ($\mathbf{u}_j = (u_{j1}, \ldots, u_{jn})'$). For uniformly distributed input perturbations, we have $\sigma^2_{\Delta x_i} = (2Q)^2/12 = Q^2/3$. Theoretically, we do not restrict the distribution of the input perturbations as long as the variance of the input perturbation ($\sigma^2_{\Delta x_i}$) is finite. However, uniform distribution is assumed here because without any prior knowledge on the distribution of unseen samples around the training samples, we assume that all of them have an equal chance of occurrence.

By the law of large numbers, when the number of input features is not too low, $\phi_j(\mathbf{x})$ would have a log-normal distribution; so, the RBFNN ST-SM is given by

$$\begin{aligned} &E_{S_Q}((\Delta y)^2) \\ &= \frac{1}{N}\sum_{b=1}^{N}\int_{S_Q(\mathbf{x}_b)} (f_\theta(\mathbf{x}_b+\Delta\mathbf{x}) - f_\theta(\mathbf{x}_b))^2 p(\Delta\mathbf{x}) d\Delta\mathbf{x} \\ &= \sum_{j=1}^{M}\left[\varphi_j\left(\exp\left(\left(\left(4\sum_{i=1}^{n}\sigma^2_{\Delta x_i}\left(\sigma^2_{x_i} + (\mu_{x_i}-u_{ji})^2 \right.\right.\right.\right.\right. \\ &\quad \left.\left.\left.\left. + 0.2\sigma^2_{\Delta x_i}\right)\right)/\left(2v_j^4\right) - \left(2\sum_{i=1}^{n}\sigma^2_{\Delta x_i}\right)/\left(2v_j^2\right)\right)\right. \\ &\quad - 2\exp\left(\left(\sum_{i=1}^{n}\sigma^2_{\Delta x_i}\left(\sigma^2_{x_i} + \left(\mu_{x_i-u_{ji}}\right)^2 \right.\right.\right. \\ &\quad \left.\left.\left. + 0.2\sigma^2_{\Delta x_i}\right)\right)/2v_j^4 - \left(\sum_{i=1}^{n}\sigma^2_{\Delta x_i}\right)/2v_j^2\right) + 1\Bigg)\Bigg] \\ &\approx \sum_{j=1}^{M}\varphi_j\left(\left(\sum_{i=1}^{n}\sigma^2_{\Delta x_i}\left(\sigma^2_{x_i} + (\mu_{x_i}-u_{ji})^2 \right.\right.\right. \\ &\quad \left.\left.\left. + 0.2\sigma^2_{\Delta x_i}\right)\right)/v_j^4\right) = \frac{1}{3}Q^2\sum_{j=1}^{M}v_j + \frac{0.2}{9}Q^4 n\sum_{j=1}^{M}\zeta_j. \end{aligned} \tag{9}$$

D. Characteristics of the R^*_{SM}

From (7), one may notice that the R^*_{SM} consists of three major components: training error (R_{emp}), ST-SM ($E_{S_Q}((\Delta y)^2)$), and the constants. The constants A and ε are preselected when the confidence of the bound $(1-\eta)$ and the training data set are fixed. Moreover, the constant B in ε could be preselected when the classifier type is selected by fixing the maximum classifier output bound. ε is generally very small for large N; so, they will not affect the result of comparisons of the generalization capability between classifiers. In contrast, if the classifier could not generalize the training samples, one may not expect the classifier to have good generalization capability to future unseen samples. Thus, the training error is one of the key components of the R^*_{SM}. Furthermore, the ST-SM term measures the output fluctuations of the classifier. A classifier having high output fluctuations yields high ST-SM because its output varies dramatically when the input value changes. Due to the classifier bias/variance dilemma, a classifier yielding a good generalization capability should minimize both terms or achieves a good balance between the two [19].

An interesting question is: Can R^*_{SM} be an effective mechanism for studying a classifier's bias/variance dilemma?

1) Limiting Cases of $R^*_{\mathrm{SM}}(Q)$: Obviously, when $Q \to \infty, S_Q \to T$, and $Q \to 0, S_Q \to D$. For $0 \le Q_1 \le \cdots \le Q_k \le (Q \to \infty)$, the relationship $D \subseteq S_{Q_1} \subseteq \cdots \subseteq S_{Q_k} \subseteq T$ holds. We further extend this relationship to

$$\begin{aligned} &R_{\mathrm{emp}}(\text{the limiting case of } R^*_{\mathrm{SM}} \text{ with } Q \to 0) \le R^*_{\mathrm{SM}}(Q_1) \\ &\qquad \le \cdots \le R^*_{\mathrm{SM}}(Q_k) \le R_{\mathrm{true}}(R^*_{\mathrm{SM}} \text{ with } Q \to \infty). \end{aligned} \tag{10}$$

This relationship shows that the limiting case of $R^*_{\mathrm{SM}}(Q)$ with $Q \to \infty$ bounds from above the R_{true}. For $Q \to \infty$

$$\begin{aligned} R_{\mathrm{true}} &= \int_T (f_\theta(\mathbf{x}) - F(\mathbf{x}))^2 p(\mathbf{x})\, d\mathbf{x} \\ &= \int_{S_Q} (f_\theta(\mathbf{x}) - F(\mathbf{x}))^2 p(\mathbf{x})\, d\mathbf{x} \\ &= R_{\mathrm{SM}}(Q) \le R^*_{\mathrm{SM}}(Q). \end{aligned} \tag{11}$$

Moreover, for the limiting case of $Q \to 0$, $R^*_{\mathrm{SM}}(Q)$ bounds R_{emp} from above. When $Q \to 0$, $E_{S_Q}((\Delta y)^2)$ vanishes, and we have

$$R_{\mathrm{emp}} \leq (\sqrt{R_{\mathrm{emp}}} + A)^2 + \varepsilon \leq R^*_{\mathrm{SM}}(Q). \tag{12}$$

2) R^*_{SM} for Other Classifiers: The R^*_{SM} as well as the R_{SM} could be defined for any classifier trained with MSE. Examples include feedforward neural networks like MLPNN, SVM, and recurrent neural networks such as Hopfield networks. The R^*_{SM} for other types of classifier could be defined by rederiving the ST-SM term for the particular type of classifier concerned.

3) Independence of Training Method: The R^*_{SM} is determined with no regard to the training methods being used. Only the parameters of the finally trained classifier are used in the model. Hence, the R^*_{SM} model could also be used to compare different training methods in terms of the generalization capability of the classifiers being built.

4) Time Complexity: The ST-SM has a time complexity of $O(Mn)$. The computational complexities of both the ST-SM and the R^*_{SM} are low and they do not depend on the number of training samples (N). However, similar to all other architecture selection methods, R^*_{SM} requires a trained classifier and the time for architecture selections is dominated by the classifier training time. Therefore, the proposed architecture selection method may not have a large advantage in terms of speed when being compared with other architecture selection methods, except the two CV-based methods.

5) Limitations of the Localized Generalization Error Model: The major limitation of the localized generalization error model is that it requires a function (i.e., f_θ) for the derivation of the ST-SM. Classifiers such as rule-based system and decision tree may not be able to make use of this concept. It will be a challenging task to find way to determine the R^*_{SM} for these classifiers.

Another limitation of the present localized generalization error model is due to the assumption that unseen samples are uniformly distributed. This assumption is reasonable when there is no *a priori* knowledge on the true distribution of the input space, and hence, every sample may have the same probability of occurrence. One would need to rederive a new R^*_{SM} when a different distribution of the input space is assumed. We also remark that even though the distribution of the unseen samples is known, their true target outputs are still unknown, and hence, it will be difficult to judge how good the bound is. On the other hand, if both input and Q-union distributions are the same, the localized generalization error model is expected to be a good estimation for the generalization error for the unseen samples, but this needs further investigation.

6) R^*_{SM} and Regularization: From (9), one may notice that the connection weight magnitude (w_j^2 in the φ_j) is directly proportional to the ST-SM, and thus, the R^*_{SM}. This provides a theoretical justification that, by controlling the magnitude of the connection weights between hidden and output layers, one could reduce the generalization error which is essential to the regularization of neural network learning [6].

7) Predicting Unseen Samples Outside the Q-Union: In practice, some unseen samples may be located outside the Q-union. This may be due to the fact that the Q value is too small, and thus, the Q-union covers only very few unseen samples. However, expanding the Q value will lead to a larger R^*_{SM}, because more dissimilar unseen samples are included in the Q-union, and a classifier with very large R^*_{SM} upper bound may not be meaningful. Furthermore, the R^*_{SM} bounds from above the MSE of the unseen samples and the MSE is an average of the errors. Thus, the R^*_{SM} bound may still work well even though some of the unseen samples are located outside the Q-union. This is supported by the experimental results presented in a Section IV-B. When splitting the training and testing data sets, naturally some unseen testing samples may fall outside the Q-union. However, our experiments show that the generalization capability of the RBFNNs selected by using the R^*_{SM} is still the best in terms of testing accuracy when compared with other methods.

On the other hand, if there is a large portion of unseen samples located outside the Q-union, i.e., dissimilar to the training samples, one may consider revising the training data set to include more such samples and retrain the classifier. As mentioned before, classifiers may not be expected to classify unseen samples that are totally different from the training set. We will describe an experiment for this scenario in Section IV-C.

E. Comparing Two Classifiers Using the R^*_{SM}

One way to compare two classifiers is to fix the $R^*_{\mathrm{SM}}(Q)$ value and compare the difference between their Q values. The other way is to fix the Q value and compare the $R^*_{\mathrm{SM}}(Q)$ of the two classifiers.

Assume there are two classifiers, f_1 and f_2. There exists a Q_1 for f_1 yielding $R^*_{\mathrm{SM}}(Q_1) = a$ and a Q_2 for f_2 yielding $R^*_{\mathrm{SM}}(Q_2) = a$. If $Q_1 < Q_2$, then f_2 has a better generalization capability because f_2 covers more unseen samples but still has the same generalization error upper bound. In other words, the architecture selection could be done by searching the functional space $\theta \in \Lambda$ and seek f_θ with the largest Q producing $R^*_{\mathrm{SM}}(Q) = a$. This is an important property of $R^*_{\mathrm{SM}}(Q)$ and we will make use of it to find the optimal classifier with the largest coverage in Section III.

On the other hand, one could compare the two classifiers, f_1 and f_2, based on their $R^*_{\mathrm{SM}}(Q)$ computed using the same Q value. The classifier with lower $R^*_{\mathrm{SM}}(Q)$ is expected to have a better generalization capability.

All other comparison methods by not fixing either Q or $R^*_{\mathrm{SM}}(Q)$ may not be meaningful.

III. Architecture Selection Using R^*_{SM} With Selected R^*_{SM} – MC^{2}SG

The selection of the number of hidden neurons in the RBFNN is usually done by sequential learning [8], [23], [24], [29] or by ad hoc choice. The sequential learning technique only makes use of the training error to determine the number of hidden neurons, without any reference to the generalization capability. Moreover, [8] and [29] assume that the classifier does not have prior knowledge about the number of training samples while [23] and [24] do. For ease of comparison with other architecture selection method, we assume that the number of training samples in our experiments is known to the classifier. In this section, we propose a new technique based on R^*_{SM} to find the optimal number of hidden neurons which makes use of the generalization capability of the RBFNN.

For any given threshold a on generalization error bound (R^*_{SM}), the localized generalization error model allows us to find the best classifier by maximizing Q, assuming that the MSE of all samples within the Q-union is smaller than a. One can formulate the architecture selection problem as a maximal coverage classification problem with selected generalization error bound (MC^{2}SG), i.e.,

$$\max_{\theta\in\Lambda} Q \quad \text{subject to } R^*_{\mathrm{SM}}(Q) \le a. \tag{13}$$

In the RBFNN training algorithm presented in Section III-B, once the number of hidden neurons is fixed, the center positions and widths could be estimated by any automatic clustering algorithm such as k-means clustering, self-organizing map, or hierarchical clustering; so we only need to concentrate on the problem of determining the number of hidden neurons. This means that $\theta = \mathrm{M}$ and $\Lambda = \{1, 2, \ldots, \mathrm{N}\}$ because it is not reasonable to have the number of hidden neurons higher than the number of training samples.

Problem (13) is a 2-D optimization problem. The first dimension is the number of hidden neurons (θ) and the second dimension is the Q for a fixed θ. For every fixed θ and Q, we can determine $R^*_{\mathrm{SM}}(Q)$. These two parameters, θ and Q, are independent. Furthermore, by substituting (9) into the (7), with probability $(1-\eta)$, we have the R^*_{SM} for RBFNN as follows:

$$R^*_{\mathrm{SM}} \approx \left(\sqrt{\frac{1}{3}Q^2 \sum_{j-1}^{M} \upsilon_j + \frac{0.2}{9} Q^4 n \sum_{j=1}^{M} \zeta_j} + \sqrt{R_{\mathrm{emp}}} + A \right)^2 + \varepsilon. \tag{14}$$

Let $R^*_{\mathrm{SM}}(Q) = a$. For every θ, let the Q that satisfies $R^*_{\mathrm{SM}}(Q) = a$ be Q^*, where a is a constant real number $R^*_{\mathrm{SM}}(Q) = a$ existing because the second-order derivative of (14) is positive. We could solve (14) as follows:

$$Q^4 \frac{0.2}{3} n \sum_{j=1}^{M} \zeta_j + Q^2 \sum_{j=1}^{M} \upsilon_j - 3(\sqrt{a-\varepsilon} - \sqrt{R_{\mathrm{emp}}} - A)^2 = 0. \tag{15}$$

Equation (15) could be solved by the quadratic equation and two solutions will be found for Q^2. For $a \geqslant \varepsilon$ and ε being usually a very small constant when the number of samples is large, there will be one positive and one negative real solution for Q^2 because, in (15), the coefficients for the terms Q^4 and Q^2 are positive, but the constant term is negative. This means that there will be two real and two imaginary solutions for Q and let Q^* be the only positive real solution among the four. Note that Q is defined to be the width of the Q-neighborhood and as such it must be a nonnegative real number

$$h(M, Q^*) = \begin{cases} 0, & R_{\mathrm{emp}} \geqslant a \\ Q^*, & \text{else} \end{cases}. \tag{16}$$

For RBFNN architecture selection, (13) is equivalent to

$$\max_{M\in\Lambda} \quad h(M, Q^*). \tag{17}$$

A. Parameters for MC^{2}SG

In (7), the difference between the maximum and minimum values of target outputs (A) and the number of training samples (N) are fixed for a given training data set, and the maximum possible value of the MSE and the confidence level of the R^*_{SM} bound, namely, B and η, could also be selected before any classifier training.

In a K-class classification problem, one may select $F(\mathbf{x}) \in \{(k_1, k_2, \ldots k_K)\}$ where $k_i \in \{0, 1\}$ and $\sum_{i=1}^{K} k_i = 1$. $k_i = 1$ if the sample belongs to the ith class, and one minimizes the MSE of all the RBFNN outputs simultaneously. All the $F(\mathbf{x})$, $f_\theta(\mathbf{x})$, and R^*_{SM} are vectors, and thus, the sum of R^*_{SM}s of all the K RBFNN outputs are minimized in the MC^{2}SG. One may notice that the minimization of the sum of R^*_{SM}s of all the outputs is equivalent to the minimization of the average of them. However, the average of R^*_{SM}s may provide a better interpretation and its range is not affected by the value K.

The determination of the constant a is made according to the classifier's output schemes for classification. For instance, if class outputs are different by one, then a may be selected as 0.25 because a sample is misclassified if the square of its deviation from the target output is larger than 0.25. From (16), one may notice that the smaller a is, the larger number of hidden neurons will be selected by the MC^{2}SG because the Q value of those RBFNNs will be zero if its training error is larger than the a value and training error decreases as the of number of hidden neurons increases. On the other hand, if the a value is selected to be larger than 0.25, the effect to the architecture selection will be insignificant. Experimental results show that those RBFNNs yielding training error larger than 0.25 will not yield good generalization capability, i.e., poor testing accuracies.

B. RBFNN Architecture Selection Algorithm for MC^{2}SG

The solution of (17) is realized using the following selection algorithm. The MC^{2}SG is independent of any training algorithm of the RBFNN. However, one of the fast RBFNN training methods is employed in this paper for all of the experiments [11]. Steps 2) and 3) are the unsupervised learning to find the centers and widths of the RBFs in the hidden neurons and Step 4) finds the least-square solution of the connection weights by making use of the linear relationship, shown in (8), between the hidden neuron outputs and RBFNN outputs. Other training algorithms could be adopted in Steps 2)–4) in the following architecture selection algorithm.

The architecture selection algorithm of the MC^{2}SG is as follows.

1) Start with $M = 1$ (M denotes the number of hidden neurons).
2) Perform k-means clustering algorithm to find the centers for the M hidden neurons.
3) For each of the M RBF hidden neurons, select its width value to be the distance between the center of itself and the one of the nearest hidden neuron.
4) Compute the connection weights using a pseudoinverse method.
5) Compute the Q-value for the current RBFNN using (16).
6) If the stopping criterion is not fulfilled, $M = M + 1$ and go to Step 2).

The stopping criterion could be selected as "M is equal to the number of training samples" and this will allow the MC^{2}SG to search for all possible number of hidden neurons. However,

it is computationally prohibited for large data set and we will discuss heuristic stopping criterion in Section III-C. Moreover, constructive approach is employed here because it is more efficient to start the search with one hidden neuron, and add one hidden neuron for each iteration.

C. Heuristic Method to Reduce the Computational Time for MC^{2}SG

Same as the other methods, $h(M, Q^*)$ is generally not differentiable with respect to M (not a smooth function). One must try out all possible M values in order to find the optimal solution. Our experimental results show that $h(M, Q^*)$ drops to zero when the classifier becomes too complex, i.e., M is too large, and heuristically, an early stop could be made to reduce the number of classifier trainings when Q approaches zero. In our experiments, we stop the search when the Q values drop below a threshold. In fact, Q does not increase significantly after it drops below 10% of the maximum value of Q being found, and thus, it is used as the threshold to speed up the MC^{2}SG.

IV. Experiments on 17 Benchmarking UCI Data Sets

The experimental setup is explained in Section IV-A and we present and discuss the experimental results in Section IV-B. Section IV-C shows the special case when the training samples are drawn with bias.

A. Experimental Setup

In this section, we compare the MC^{2}SG with well-known architecture selection methods [5], [6]:CV, sequential learning, and two ad hoc methods. Steps 2)–4) of the architecture selection algorithm presented in Section III-B are used to train the RBFNNs for all architecture selection methods. Every data set is divided into two parts, training and testing, and each consists of 50% of the samples. This is repeated ten times to generate ten independent runs for each data set. Table II shows the average classification accuracies on the testing sets for these ten runs. The testing data set is treated as future unseen samples. All inputs are scaled to $[0, 1]$ to eliminate the effect of large values. Seventeen real-world data sets from the UCI machine learning repository Table I are used. A wide range of data sets is selected from image classification, medical informatics, network security, and financial applications. The "DAPRA99 DoS" consists of normal and denial of service (DoS) types of samples from the 10% training data set of DAPRA99 data set [21]. The multiple feature data set consists of large number of features while DAPRA99 DoS, waveform, mushroom, and optical digit data sets consist of large number of samples. In addition, small data sets, e.g., iris, wine, and thyroid gland, and medium data sets are selected to demonstrate that the proposed method MC^{2}SG can be applied to any variation of classification problems in numbers of samples, features, and classes. Moreover, in the experiments, a sample is considered to be incorrect if its error is larger than 0.5 (a squared error of 0.25), and therefore, we will use $a = 0.25$ as the threshold value for the R^*_{SM} in MC^{2}SG.

As suggested in [5], fivefolds (5-CV) and tenfolds (10-CV) CVs are used in our experiments. In a C-fold CV, the training data set is divided into C disjoint partitions and one of them is used as the validation set. C classifiers are trained using different partitions and the average of these C validation errors is used as the CV error. The number of hidden neurons which yields the lowest CV error will be selected. The "sequen_MSE" and "sequen_01" methods are to add hidden neurons until, respectively, the training MSE < 0.025 and the classification error is minimized. The ad hoc method denoted by "N" uses every training sample as a center of RBFNN hidden neurons. The other ad hoc method "SQRT(N)" selects the number of hidden neurons to be equal to the square root of the number of training samples.

TABLE I
Seventeen Benchmarking Data Sets

Datasets	# Features	# Samples	# Classes
Thyroid Gland	5	215	3
Japanese Credit Approval	15	688	2
Wine Recognition	13	178	3
Breast Cancer	10	699	2
Hepatitis	18	155	2
Iris	4	150	3
Sonar Target	60	208	2
Ionosphere	34	351	2
Heart Disease	13	135	2
Mushroom	22	8,124	2
Waveform	21	5,000	3
Pima Diabetes	8	768	2
Multiple Feature	649	2,000	10
Optical Digit	64	5,620	10
German Credit Approval	24	1,000	2
Car Evaluation	6	1,728	4
DARPA 99 DoS	41	494,020	2

B. Experimental Results and Analysis

Experimental results in Tables II and III show that the MC^{2}SG performs best among the methods consistently in terms of highest average classification accuracy for unseen samples, smaller average number of hidden neurons, and fast in training times without regarding to the numbers of training samples, features, and classes of the data sets. However, we note that the differences in terms of average testing accuracies are not very big, therefore, one-tailed McNemar tests [27], [28] were performed to examine the statistical significance of the improvements made by the MC^{2}SG. Table IV shows that the RBFNNs selected by MC^{2}SG are statistically significantly better than those RBFNNs selected by other methods at 0.05 level of significance (i.e., McNemar test value larger than 2.71). Furthermore, the statistical significances shown in Table IV indicate that the proposed method outperforms other method more significantly when the number of training samples is large. It is also interesting to observe that from Tables II and IV, none of the other six methods performs the second best consistently in all of the experiments while the MC^{2}SG consistently performs best for all 17 data sets.

Table III shows that the CV method is very time consuming, and it requires 5 to 10 000 times longer to find the best RBFNN. The reason that less time is required by the MC^{2}SG than the two sequential learning methods is due to the early stopping of the

TABLE II
AVERAGE CLASSIFICATION ACCURACY (AND THEIR STANDARD DEVIATION IN BRACKETS) FOR TESTING DATA SET OVER TEN INDEPENDENT RUNS

DATASETS\METHODS	MC^{2}SG	5-CV	10-CV	SQUEN_MSE	SQUEN_01	SQRT(N)	N
Thyroid Gland	86.82%(1.49%)	80.65%(3.70%)	77.38%(6.03%)	84.30%(1.60%)	85.14%(2.08%)	86.54%(1.25%)	85.79%(1.54%)
Japanese Credit Approval	88.87%(1.06%)	87.06%(1.13%)	86.13%(1.10%)	79.77%(1.68%)	88.84%(1.62%)	88.34%(1.80%)	39.45%(7.02%)
Wine Recognition	93.18%(0.59)	90.06%(0.90%)	91.19%(1.20%)	91.14%(1.10%)	91.82%(2.61%)	90.57%(1.17%)	93.18%(1.10%)
Breast Cancer	97.29%(0.54%)	96.99%(0.93%)	96.92%(0.62%)	97.10%(0.73%)	96.43%(0.52%)	97.26%(0.54%)	66.85%(4.37%)
Hepatitis	82.99%(1.75%)	80.26%(2.07%)	79.35%(1.89%)	77.20%(1.94%)	77.66%(2.39%)	77.27%(3.47%)	74.42%(2.29%)
Iris	97.87%(0.77%)	96.53%(1.12%)	96.07%(1.13%)	96.27%(1.05%)	96.13%(0.93%)	97.77%(1.26%)	83.60%(6.34%)
Sonar Target	83.20%(0.82%)	80.49%(1.07%)	80.87%(0.61%)	81.17%(0.92%)	78.25%(0.94%)	76.50%(1.12%)	82.62%(1.04%)
Ionosphere	84.71%(1.07%)	83.29%(1.48%)	83.40%(1.30%)	82.06%(1.34%)	79.94%(1.30%)	84.11%(1.66%)	48.17%(5.96%)
Heart Disease	83.24%(0.47%)	82.41%(0.79%)	81.37%(0.68%)	79.17%(1.31%)	60.37%(1.25%)	82.59%(0.56%)	83.23%(0.95%)
Mushroom	99.95%(0.20%)	99.70%(0.02%)	99.94%(0.02%)	99.79%(0.20%)	99.95%(0.08%)	99.73%(0.20%)	N/A(N/A)
Waveform	73.48%(0.50%)	72.71%(0.74%)	72.89%(0.61%)	73.21%(0.77%)	73.32%(0.70%)	72.30%(0.56%)	N/A(N/A)
Pima Diabetes	79.40%(0.94%)	75.86%(0.86%)	75.55%(1.10%)	72.68%(3.32%)	54.37%(2.64%)	77.53%(1.14%)	32.94%(2.86%)
Multiple Feature	97.50%(0.50%)	97.50%(0.67%)	97.28%(0.87%)	96.80%(0.65%)	97.28%(0.66%)	92.53%(1.45%)	56.05%(2.80%)
Optical Digit	95.33%(0.43%)	94.80%(0.48%)	95.25%(0.39%)	95.01%(0.40%)	94.18%(0.43%)	91.82%(0.47%)	N/A(N/A)
German Credit Approval	76.64%(0.87%)	75.30%(1.55%)	75.28%(1.43%)	69.62%(1.50%)	42.99%(1.90%)	75.04%(1.28%)	34.49%(7.92%)
Car Evaluation	92.35%(0.51%)	90.92%(0.60%)	91.61%(0.38%)	92.33%(0.36%)	76.50%(0.88%)	84.22%(0.82%)	76.50%(0.88%)
DARPA 99 DoS	99.64%(0.01%)	99.59%(0.02%)	99.64%(0.01%)	99.30%(0.02%)	99.30%(0.02%)	N/A(N/A)	N/A(N/A)

TABLE III
AVERAGE NUMBER OF HIDDEN NEURONS AND TOTAL RUNNING TIME IN SECONDS OVER TEN INDEPENDENT RUNS (IN BRACKET)

DATASETS\METHODS	MC^{2}SG	5-CV	10-CV	SQUEN_MSE	SQUEN_01	SQRT(N)	N
Thyroid Gland	6.7 (1)	16.2 (40)	11.7 (221)	5.1 (1)	38.2 (8)	10.0 (1)	108.0 (1)
Japanese Credit Approval	13.9 (5)	11.0 (3366)	10.8 (4700)	36.1 (12)	44.0 (16)	19.0 (1)	344.0 (18)
Wine Recognition	7.2 (9)	33.7 (119)	38.7 (291)	36.1 (14)	24.3 (9)	9.0 (1)	90.0 (1)
Breast Cancer	2.1 (1)	25.1 (3360)	27.8 (10127)	2.0 (1)	42.8 (13)	19.0 (1)	350.0 (18)
Hepatitis	6.3 (10)	7.0 (67)	4.8 (168)	42.7 (10)	42.9 (10)	9.0 (1)	78.0 (1)
Iris	6.2 (2)	13.5 (58)	9.9 (134)	22.3 (6)	26.7 (6)	9.0 (1)	75.0 (1)
Sonar Target	22.9 (45)	32.4 (220)	47.1 (541)	55.7 (98)	48.5 (65)	10.0 (1)	105.0 (2)
Ionosphere	13.6 (24)	33.3 (1216)	26.6 (2799)	77.2 (57)	94.9 (94)	13.0 (1)	176.0 (4)
Heart Disease	8.8(2)	11.3 (101)	16.50 (314)	52.5 (36)	51.8 (29)	12.00 (1)	135.0 (2)
Mushroom	51.6 (348)	118.0 (3756)	105.00 (7997)	66.0 (461)	95.5 (738)	64.00 (22)	4062.0 (N/A)
Waveform	20.4 (26)	78.3 (1277)	35.70 (2838)	114.9 (489)	113.7 (489)	50.00 (10)	2500.0 (N/A)
Pima Diabetes	22.0 (13)	54.0 (2806)	57.20(8241)	275.2 (707)	291.6 (1012)	20.00 (1)	384.0 (25)
Multiple Feature	104.0 (1517)	340.0 (50470)	367.50(106269)	423.3 (15431)	356.7 (15671)	32.00 (57)	1000.0 (15661)
Optical Digit	92.9(329)	104.0 (22462)	117.0 (46630)	115.7 (382)	88.6 (282)	53.00 (28)	2811.0 (N/A)
German Credit Approval	10.5 (4)	40.4 (8219)	25.6 (17506)	214.7 (545)	236.5 (783)	23.0 (2)	500 (63)
Car Evaluation	94.8 (86)	154.0 (10240)	189.2 (22097)	215.0 (413)	865.0 (4630)	30.0 (2)	865.0 (180)
DARPA 99 DoS	84 (46932)	152 (711945)	84 (1479379)	198.0 (155008)	178.0 (143589)	497.0 (N/A)	247010 (N/A)

algorithm. However, the MC^{2}SG and the two sequential methods would still be at least five to ten times faster than the CV methods even if these methods try out all the possible number of hidden neurons.

A sequential learning method using best training MSE as a stopping criterion performs better than the one using training classification error. However, both methods will select a classifier with a larger number of hidden neurons. Moreover, they consider the training error only without regarding to the classifier complexity and generalization capability; thus, their performances fluctuate. For instance, in Table II, the experimental results for the Pima diabetes, heart disease, German credit approval, and car evaluation data sets show that the average testing accuracies of the RBFNNs selected based on the minimization of the training classification error are 54.37%, 60.37%, 42.99%, and 76.50%, respectively. In contrast, the average testing accuracies for those RBFNNs selected based on the training MSE are 72.68%, 79.17%, 69.62%, and 92.33%, respectively. These experimental results show that the training classification error may not be an appropriate criterion for architecture selection of RBFNNs trained using MSE. Moreover, the average testing accuracies of the RBFNNs selected based on the proposed generalization error model are 79.40%, 83.24%, 76.64%, and 92.35%, respectively. This indicates that the selection of RBFNN based on training performance may not be appropriate.

The ad hoc method N yields very high accuracy in some data sets, e.g., wine and breast cancer data sets, while it yields less than 50% testing accuracy in the experiments of the Japanese credit approval, German credit approval, and ionosphere data sets. While SQRT(N) performs better than the method N in many data sets, it does use more hidden neurons than the

TABLE IV
MCNEMAR TEST STATISTICS BETWEEN MC^{2}SG AND OTHER METHODS FOR TESTING DATA SET OVER TEN INDEPENDENT RUNS

DATASETS\METHODS	5-CV	10-CV	SQUEN_MSE	SQUEN_01	SQRT(N)	N
Thyroid Gland	40.09	75.00	14.40	7.36	1.29	6.00
Japanese Credit Approval	12.67	24.43	176.89	0.276	2.50	1399.40
Wine Recognition	9.12	3.14	3.45	2.79	4.69	0.01
Breast Cancer	6.33	6.90	5.79	16.33	0.05	999.34
Hepatitis	9.26	9.94	35.89	34.20	36.74	50.67
Iris	1.39	2.08	1.65	1.72	0.05	87.23
Sonar Target	12.67	12.12	3.80	15.45	31.39	1.16
Ionosphere	43.18	42.48	6.88	31.08	0.57	582.20
Heart Disease	2.11	6.32	21.78	212.78	1.91	9.00
Mushroom	101.15	2.78	62.23	0.67	86.17	N/A
Waveform	67.88	26.42	24.95	4.53	82.10	N/A
Pima Diabetes	77.84	79.57	153.12	676.73	39.92	1332.80
Multiple Feature	2.08	33.33	42.12	30.42	404.54	3815.60
Optical Digit	77.84	10.13	52.40	169.27	779.39	N/A
German Credit Approval	6.10	10.04	110.44	1215.70	1169	1699.80
Car Evaluation	75.558	21.24	0.73	1041.60	484.36	1041.60
DARPA 99 DoS	124.00	0.00	839.83	839.83	N/A	N/A

TABLE V
EXPERIMENTAL RESULTS FOR THE BIASED TRAINING DATA SET FOR THE UCI THYROID GLAND DATA SET

	MC^{2}SG	5-CV	10-CV	SQUEN_MSE	SQUEN_01	SQRT(N)	N
# Hidden Neurons	4	10	13	42	55	10	107
Average (Std Dev) Training Accuracy	85.98(1.22)	92.52(3.55)	93.45(3.48)	98.13(0.67)	100.00(0.00)	91.65(3.44)	100.00(0.00)
Average (Std Dev) Testing Accuracy	59.26(3.03)	39.82(3.79)	46.30(4.34)	52.58(2.47)	51.11(6.11)	37.89(6.10)	45.84(6.38)

MC^{2}SG for virtually the same level of classification accuracy. For example, from the breast cancer data set, the MC^{2}SG uses nine times less hidden neurons than the SQRT(N). Both ad hoc methods are fast in training time because they employ preselected number of hidden neurons; however, this number is usually too large. In addition, for data sets consisting of extremely large number of training samples, both ad hoc methods may select an unreasonable number of hidden neurons. For example, the method N selects 247 010 hidden neurons for the DAPRA99 DoS data set which is infeasible for training.

One may notice that the testing accuracies of some data sets, e.g., waveform, ionosphere, and German credit approval, are not very high. However, our aim of presenting these experimental results here is not to demonstrate the superiority of our method for all data sets. Our aim is to present an overall comparison of the performance of the RBFNNs constructed by our proposed method and that of other popular methods using the same training and testing data sets.

C. Experiments on a Biased Data Set

With a random splitting of training and testing data sets, the Q-union of the training samples usually covers most of the testing samples with a small Q value, i.e., the testing samples are similar to training samples. This would also be the case in real-world applications of pattern classifier that unseen samples may not be very dissimilar to the training ones.

In some cases when the training data set is sampled poorly, the classifier being trained using this data set would perform poorly for future unseen samples. However, the interesting question is: Will the MC^{2}SG still be working well in such cases? We performed a biased sampling using the thyroid gland data set because this data set consists of fewer features and thus is easier for visualization. Training samples are selected if its value in feature 1 is less than 0.57 such that the training and testing samples are approximately splitting in half and their distributions are shown in Fig. 3. Then, we swap the training and testing data sets to obtain the second run of this experiment. These processes are repeated for the other four features to obtain ten training and testing data set pairs. The results shown in Table V are the average over these ten trials. Fig. 2 shows the distributions of training and testing samples in a random, unbiased splitting, and the shaded area in Figs. 2 and 3 is the coverage of the Q-union. From Figs. 2 and 3, one may notice that most of the testing samples are covered by the Q-union in random splitting while a large portion of testing samples are located outside the Q-union in the biased splitting. From Table V, one may notice that the accuracies of the classifiers selected by all methods are lower than 60%; however the MC^{2}SG still outperforms other architecture selection methods. This indicates that the minimization of the local generalization error helps minimizing the generalization error even for unseen samples that deviated from the training samples a lot.

One may notice that the training error in such a biased training data set is misleading. Thus, CV methods which depend on resampling from the training samples may not provide good estimation of the generalization error. The two sequential learning methods depend totally on the training error and they perform worse than the MC^{2}SG; however, interestingly, they perform better than the CV methods. This may be due to the fact that CV methods make use of only a portion of the biased training

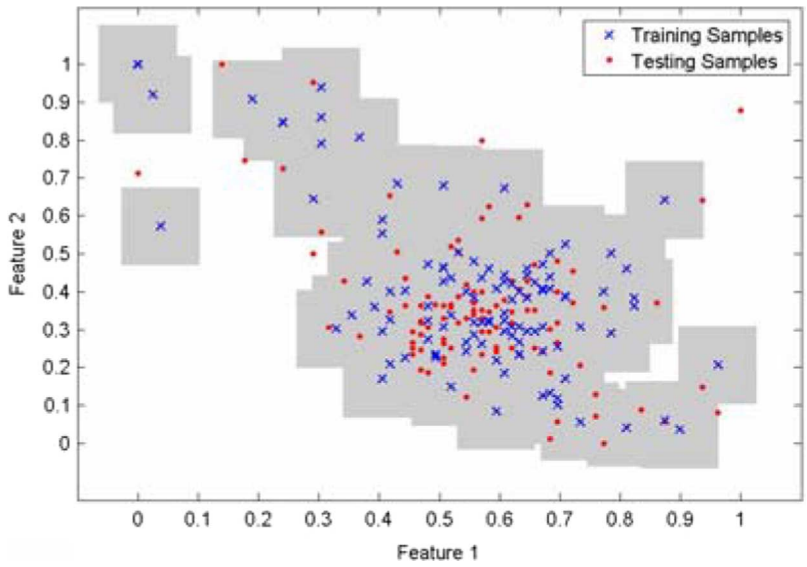

Fig. 2. Distributions of the training and testing samples of a random split for the UCI thyroid gland data set and shaded area is the coverage of the Q-union.

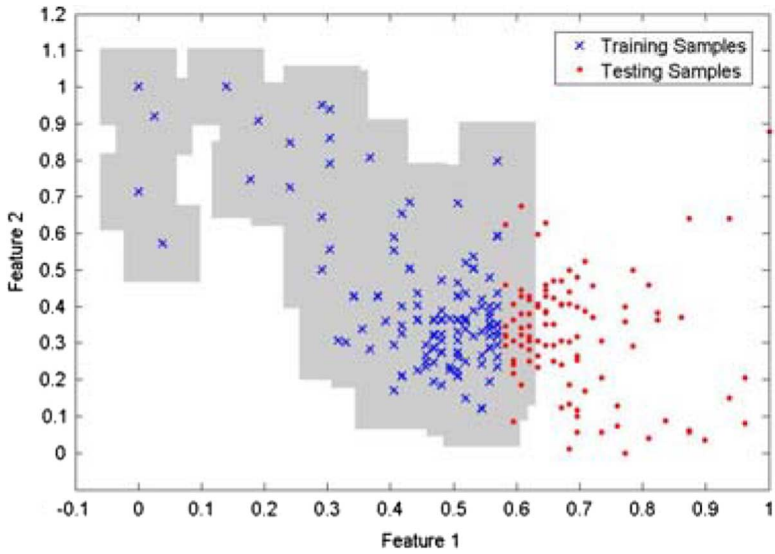

Fig. 3. Distributions of the training and testing samples of the biased split for the UCI thyroid gland data set and shaded area is the coverage of the Q-union.

samples while all other methods use all of the training samples; thus, the training samples used in both CV methods deviate a lot more than the original training data set from the testing samples. In real random splitting, this may not be the case. However, it is difficult to guarantee the quality of the training data set. In summary, experimental results show that the MC^{2}SG method finds the RBFNN yielding the best testing classification accuracy with fewer hidden neurons and uses less training time consistently in all 17 data sets.

V. Conclusion and Future Work

In this paper, we propose a new generalization error model based on the localized generalization error. R^*_{SM} bounds from above the generalization error for unseen samples within the Q-neighborhoods of the training samples. Moreover, an architecture selection method, namely MC^{2}SG based on the R^*_{SM}, is proposed to find the RBFNN classifier which has the largest coverage of unseen samples while its R^*_{SM} is still less than a preselected threshold. The R^*_{SM} was shown to be a generalization of R_{true} and the experimental results support our claim.

This paper has demonstrated the use of the MC^{2}SG to find the number of hidden neurons for an RBFNN, while the values of other parameters were found using existing methods. A possible extension of our result is to find the values of other RBFNN parameters, e.g., center positions, width, and connection weights, via an optimization of R^*_{SM} because these parameters are also coefficients of the R^*_{SM}. However, the tradeoff between the optimality of the solution and time complexity will be an important consideration.

Another future work could be the derivation of the R^*_{SM} for distribution of S_Q other than the uniform distribution. One could ultimately derive the R^*_{SM} by taking the distribution of S_Q as one of its parameters.

This paper could also serve as a theoretical foundation for further applications of the R^*_{SM} such as feature selection and active learning. It will also be interesting to investigate the derivation

of the ST-SM for other classifiers or using objective functions other than MSE for classifier training.

ACKNOWLEDGMENT

The authors would like to thank the six reviewers who provided very helpful comments.

REFERENCES

[1] H. Akaike, "A Bayesian analysis of the minimum AIC procedure," *Ann. Inst. Statist. Math.*, pp. 9–14, 1978.
[2] D. Chakraborty and N. R. Pal, "A novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 1–14, Jan. 2003.
[3] V. Cherkassky and F. Mulier, *Learning From Data*. New York: Wiley, 1998.
[4] V. Cherkassky, X. Shao, F. M. Mulier, and V. N. Vapnik, "Model complexity control for regression using VC generalization bounds," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1075–1089, Sep. 1999.
[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Element of Statistical Learning*. New York: Springer-Verlag, 2001.
[6] S. Haykin, *Neural Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1998.
[7] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Statist. Assoc.*, vol. 58, pp. 13–30, 1963.
[8] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 57–67, Jan. 2005.
[9] L. C. Jain and V. R. Venuri, Eds., *Industrial Applications of Neural Networks*. Boca Raton, FL: CRC Press, 1999.
[10] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[11] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, pp. 281–294, 1989.
[12] W. W. Y. Ng and D. S. Yeung, "Selection of weight quantisation accuracy for radial basis function neural network using stochastic sensitivity measure," *Inst. Electr. Eng. Electron. Lett.*, pp. 787–789, 2003.
[13] W. W. Y. Ng, D. S. Yeung, X.-Z. Wang, and I. Cloete, "A study of the difference between partial derivative and stochastic neural network sensitivity analysis for applications in supervised pattern classification problems," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2004, pp. 4283–4288.
[14] W. W. Y. Ng, D. S. Yeung, and I. Cloete, "Quantitative study on effect of center selection to RBFNN classification performance," in *IEEE Proc. Int. Conf. Syst., Man, Cybern.*, 2004, pp. 3692–3697.
[15] H. Park, N. Murata, and S.-I. Amari, "Improving generalization performance of natural gradient learning using optimized regularization by NIC," *Neural Comput.*, pp. 355–382, 2004.
[16] S. Watanabe, "Algebraic analysis for nonidentifiable learning machines," *Neural Comput.*, vol. 13, pp. 899–933, 2001.
[17] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[18] N. B. Karayiannis and M. M. Randolph-Gips, "On the construction and training of reformulated radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 835–846, Jul. 2003.
[19] S. Geman and E. Bienenstock, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, pp. 1–58, 1992.
[20] M. Anthony and P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*. Cambridge, U.K.: Cambridge Univ. Press, 1999.
[21] W. W. Y. Ng, R. K. C. Chang, and D. S. Yeung, "Dimensionality reduction for denial of service detection problems using RBFNN output sensitivity," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2003, pp. 1293–1298.
[22] Y.-J. Oyang, S.-C. Hwang, Y.-Y. Ou, C.-Y. Chen, and Z.-W. Chen, "Data classification with radial basis function networks based on a novel kernel density estimation algorithm," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 225–236, Jan. 2005.
[23] W. Kaminski and P. Strumillo, "Kernel orthonormalization in radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1177–1183, Sep. 1997.
[24] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 306–314, Mar. 2000.
[25] H. Leung, N. Dubash, and N. Xie, "Detection of small objects in clutter using a GA-RBF neural network," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 1, pp. 98–118, Jan. 2002.
[26] K. Z. Mao, "RBF neural network center selection based on fisher ratio class separability measure," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1211–1217, Sep. 2002.
[27] S. L. Salzberg, "On comparing classifiers: A critique of current research and methods," *Data Mining Knowl. Disc.*, pp. 317–327, 1997.
[28] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. London, U.K.: Chapman & Hall, 2004.
[29] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

Daniel S. Yeung (M'89–SM'99–F'04) received the Ph.D. degree in applied mathematics from Case Western Reserve University, Cleveland, OH, in 1974.

In the past, he has worked as an Assistant Professor of Mathematics and Computer Science at Rochester Institute of Technology, as a Research Scientist in the General Electric Corporate Research Center, and as a System Integration Engineer at TRW. He was the chairman of the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He leads a group of researchers in Hong Kong and China who are actively engaging in research works on computational intelligence and data mining. His current research interests include neural network sensitivity analysis, data mining, Chinese computing, and fuzzy systems.

Dr. Yeung was the President of IEEE Hong Kong Computer Chapter, an Associate Editor for both the IEEE TRANSACTIONS ON NEURAL NETWORKS and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS. He has been elected as President Elect for the IEEE Systems, Man, and Cybernetics Society. He served as a General Co-Chair of the 2002–2007 International Conference on Machine Learning and Cybernetics held annually in China, and a Keynote Speaker for the same Conference. His IEEE Fellow citation makes reference to his "contribution in the area of sensitivity analysis of neural networks and fuzzy expert systems."

Wing W. Y. Ng (S'01–M'06) received the B.Sc. degree in information technology and the Ph.D. degree in computer science from the Hong Kong Polytechnic University, Kowloon, Hong Kong, in 2001 and 2006, respectively.

In 2006, he joined the Media and Life Science (MiLeS) Computing Laboratory, Department of Computer Science and Technology, Shenzhen Graduate School, Harbin Institute of Technology, China, where he is currently an Assistant Professor. His major research interests include media computing, localized generalization error model, feature selection, active learning, and neural network sensitivity analysis.

Dr. Ng is the Co-Chair of Technical Committee on Computational Intelligence, IEEE Systems, Man and Cybernetics Society. He was also the Founder and Chairman of Hong Kong Polytechnic University Student Branch Chapter, IEEE Systems, Man and Cybernetics Society. He served as the Conference Secretary of the 2002–2007 International Conference on Machine Learning and Cybernetics held annually in China.

Defeng Wang (S'03) received the B.Eng. degree in computer application from Jilin University, Changchun, China, in 2000, the M.Eng. degree in computer application from Xidian University, Shaanxi, China, in 2003, and the Ph.D. degree from The Hong Kong Polytechnic University, Kowloon, Hong Kong, in 2006.

His recent research is focused on kernel methods, large margin learning, and medical image analysis.

Eric C. C. Tsang (M'04) received the B.Sc. degree in computer studies from the City University of Hong Kong, Hong Kong, in 1990 and the Ph.D. degree in computing from the Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1996.

He is an Assistant Professor at the Department of Computing, the Hong Kong Polytechnic University. His main research interests are in the area of fuzzy expert systems, fuzzy neural networks, machine learning, genetic algorithm, rough sets, fuzzy rough sets, fuzzy support vector machine, and multiple classifier system.

Xi-Zhao Wang (M'98–SM'02) received the B.Sc. and M.Sc. degrees in mathematics from Hebei University, Baoding, China, in 1983 and 1992, respectively, and the Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China, in 1998.

From 1983 to 1998, he worked as a Lecturer, an Associate Professor, and a Full Professor at the Department of Mathematics, Hebei University. From 1998 to 2001, he worked as a Research Fellow at the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong. Since 2001, he has been the Dean and Professor of the Faculty of Mathematics and Computer Science, Hebei University. His main research interests include inductive learning with fuzzy representation, fuzzy measures and integrals, neurofuzzy systems and genetic algorithms, feature extraction, multiclassifier fusion, and applications of machine learning. He has published over 60 international journal papers, completed over 20 funded research projects, and supervised over 30 doctorate or Masters degrees.

Prof. Xi-Zhao Wang is an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, an Associate Editor of International Journal of Information Sciences, Chair of IEEE Systems, Man, and Cybernetics Baoding Chapter; Chair of IEEE Systems, Man, and Cybernetics Technical Committee on Computational Intelligence, and an Executive Member of Chinese Association of Artificial Intelligence. He is the General Co-Chair of the 2002, 2003, 2004, 2005, 2006, and 2007 International Conference on Machine Learning and Cybernetics, cosponsored by IEEE Systems, Man, and Cybernetics Society.