Soft Comput (2007) 11: 375–381
DOI 10.1007/s00500-006-0092-4

FOCUS

Wing W. Y. Ng · Daniel S. Yeung · Defeng Wang
Eric C. C. Tsang · Xi-Zhao Wang

Localized generalization error of Gaussian-based classifiers and visualization of decision boundaries

Published online: 20 April 2006

Abstract In pattern classification problem, one trains a classifier to recognize future unseen samples using a training dataset. Practically, one should not expect the trained classifier could correctly recognize samples dissimilar to the training dataset. Therefore, finding the generalization capability of a classifier for those unseen samples may not help in improving the classifiers accuracy. The localized generalization error model was proposed to bound above the generalization mean square error for those unseen samples similar to the training dataset only. This error model is derived based on the stochastic sensitivity measure(ST-SM)of the classifiers. We present the ST-SMS for various Gaussian based classifiers: radial basis function neural networks and support vector machine in this paper. At the end of this work, we compare the decision boundaries visualization using the training samples yielding the largest sensitivity measures and the one using support vectors in the input space.

Keywords Generalization error · Stochastic sensitivity measure · Support vector machine with gaussian kernel · Radial basis function neural network · Support vector · Most sensitive vector · Decision boundary visualization

1 Introduction

One trains a classifier to deal with a pattern classification problem. The training accuracy of this classifier indicates how good it learnt from the training dataset. However, how good will this classifier performs for the future unseen samples? Therefore, this is important to know the generalization capability of the classifier. Unfortunately, one does not have the knowledge of neither the real input–output mapping function nor the real distribution of the input features. In current literatures, analytical models and empirical cross-validations are the two most widely adopted techniques to estimate the generalization capability of the classifier. In this work, we focus on the classifiers with Gaussian kernel functions, i.e., radial basis function neural network (RBFNN) and support vector machine (SVM).

Empirical cross-validation splits the training dataset into K disjoint sets. K RBFNNs with M number of hidden neurons being trained. The ith RBFNN is trained using all the disjoint sets except the ith one and it is left for finding the validation error of this RBFNN. The average of all these K validation error is used to estimate the generalization error for RBFNN with M hidden neurons. The beauty of this method is that true target outputs are used in the validation sets. However, the RBFNN M hidden neurons may yield a high variance in the generalization error. That is, one does not know the upper bound of the generalization error for the final classifier that has been trained. This is because the variance of the validation error may be very large even though the average of them is very small [4].

On the other hand, analytical models are usually derived based on the bias–variance dilemma [3] and are functions of training error and number of effective parameters of the classifier. The number of effective parameters for linear classifier could be found accurately. However, for nonlinear classifiers, usually only a very loose upper bound could be found and VC-dimension is one of the bounds of it [2]. The upper bounds of the VC-dimensions of RBFNN (h(RBFNN)) and SVM (h(SVM)) are

$$h(RBFNN) < 4M \log_2 (24eMZ), \tag{1}$$

$$h(SVM) < \min (r/m, n) + 1, \tag{2}$$

where M, Z, r, m and n denote the number of hidden neurons, the maximum value of input, the radius of the hypersphere

W. W. Y. Ng (✉) · D. S. Yeung · D. Wang · E. C. C. Tsang
Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
E-mail: cswyng@comp.polyu.edu.hk
E-mail: csdaniel@comp.polyu.edu.hk
E-mail: csdfwang@comp.polyu.edu.hk
E-mail: csetsang@comp.polyu.edu.hk
Tel.: +852-27664901
Fax: +852-27740842

X.-Z. Wang
Machine Learning Center, Faculty of Mathematics and Computer Science, Hebei University, Baoding 071002, China
E-mail: wangxz@mail.hbu.edu.cn

which contain all the training samples in the kernel space, the width of the margin and the number of dimensions of the kernel space. One may notice that (1) grows quickly with respect to the M. Furthermore, with probability $1-\eta$ the generalization error bound for all unseen samples in terms of mean square error (MSE; R_{true}) is given by [8]

$$R_{\text{true}} \leq R_{\text{emp}} + (B\varepsilon/2)\left(1+\sqrt{1+\left(4R_{\text{emp}}/B\varepsilon\right)}\right), \tag{3}$$

and

$$\varepsilon \leq 4[\ln(2l/h)+1]/(l/h) + [-\ln(\eta/4)]/l, \tag{4}$$

where l, B, R_{emp}, η denote the number of training samples, the maximum value of the MSE, the training error and the confidence of the bound. One may find that the VC-dimension is only a function of the number of hidden neurons, but not their values. Therefore, two RBFNNs with the same number of hidden neurons yielding the same training error for the same training dataset will have the same generalization error bound. However, these two RBFNNs may not yield the same generalization capability in general.

In practice, a classifier trained using training dataset should not be expected to working well for unseen samples which are totally different from the training dataset. For those unseen samples, the classifier outputs are usually unreliable and the classification results are expected to be bad for those unseen samples [9]. In [5], a localized generalization error model was proposed. Only the unseen samples which are similar to the training dataset are considered. Their MSE for a particularly trained classifier is bounded above by this localized generalization error model. Importantly, classifiers with the same number of effective parameters but different values of the parameters will produce different values using this error model.

The major component of the localized generalization error model is the stochastic sensitivity measure (ST-SM) of the trained classifier. In this paper, the novel ST-SM for SVM with Gaussian kernel is presented. Therefore, the localized generalization error model is extended from RBFNN [5] to SVM. The selection of the number of hidden neurons is a significant research issue in RBFNN. Therefore, an architecture selection method was proposed in [5] and experimental results will also be presented in this work. Furthermore, we visualize the decision boundaries of classifiers using the training samples yielding the large ST-SM. This visualization is compared with the one using support vectors in input space.

Section 2 introduces the localized generalization error model. The ST-SMs for RBFNN and SVM are presented in Sect. 3. The experimental results of pattern classification using RBFNN trained using the localized generalization error are given in Sect. 4. Sect. 5 presents the decision boundaries visualization using both the support vectors and the most sensitive vectors. We conclude this work in Sect. 6.

2 Localized generalization error model

We bound above the generalization error for unseen samples which are similar to the training dataset only [5]. We denote this set of unseen samples by the Q-union and this is the union of all Q-neighborhoods. In particular, the bth Q-neighborhood is defined as the set of all unseen samples which sup-norm from the bth training sample ($\mathbf{X}^{(b)}$) is smaller than Q

$$S_Q^{(b)} = \{\mathbf{X} : |\Delta x_i| \leq Q \quad \forall i = 1, 2, \ldots, N\}, \tag{5}$$

where $\Delta\mathbf{X} = \mathbf{X} - \mathbf{X}^{(b)} = (\Delta x_1, \Delta x_2, \ldots, \Delta x_N)'$, N denotes the number of input features.

One may notice that the Q value is an arbitrary real number and thus this could be very small or very large. When the Q value approaches zero, the Q-union approaches the training dataset. In contrast, when the Q value approaches infinity, the Q-union approaches the entire input space. The localized generalization error (R_{SM}) is defined as:

$$\int_{S_Q} (f_\theta(\mathbf{X}) - F(\mathbf{X}))^2 p(\mathbf{X}) \mathrm{d}\mathbf{X}, \tag{6}$$

where $p(\mathbf{X})$, $f_{\boldsymbol{\theta}}$ and F denote the probability density function of the inputs, the classifier with parameter set $\boldsymbol{\theta}$ and the true unknown input–output mapping function. In this work, the MSE is of particular interest because it is widely adopted in neural network training. Moreover, both neural network and SVM are real-valued output classifiers, if one minimizes only the classification error, many classification decisions will be indecisive [1]. For example, we have a classifier outputting 0 and 1 for class 1 and 2 decisions. If only the classification error is minimized, the classifier outputs are most likely to be around the 0.5 and just fall to the correct region slightly. In contrast, one could quantitatively evaluate classifiers trained by minimizing the MSE. For the same training classification accuracy, the classifier yielding smaller MSE could be better than the one yielding larger MSE. This is because the one yielding smaller MSE is supposed to be better in approximating the unknown true input–output mapping function. With probability $1-\eta$, we have [5]:

$$\begin{aligned} R_{\text{SM}}(Q) &\leq \frac{1}{l}\sum_{b=1}^{l} \int_{S_Q^{(b)}} (f_\theta(\mathbf{X}) - F(\mathbf{X}))^2 \left(1/(2Q)^N\right) \mathrm{d}\vec{X} \\ &\quad + B\sqrt{\ln\,\eta/(-2l)} \\ &\leq \left(\sqrt{R_{\text{emp}}} + \sqrt{E\left((\Delta y)^2\right)} + A\right)^2 \\ &\quad + B\sqrt{\ln\,\eta/(-2l)} \\ &= R^*_{\text{SM}}(Q), \end{aligned} \tag{7}$$

where $E((\Delta y)^2)$, A and $1/(2Q)^N$ denote the stochastic sensitivity of the classifier (ST-SM), the maximum difference between all the known $F(\mathbf{X})$ and the probability density of the unseen samples in the Q-neighborhood, respectively.

The training error could be found after we trained the classifier and the constant A could be determined after we fixed the training dataset. The last term vanishes when the number of training samples approaches infinity. The major component of the R^*_{SM} is the ST-SM which is defined as the expectation of the squared differences between training and unseen samples within the Q-union. This is a general concept which applicable to any classifier with real-valued output. In this work, we show the ST-SMs for two widely adopted Gaussian-based classifiers: RBFNN and SVM. The localized generalization error model could be applied to RBFNN (SVM) by plugging in the ST-SM for RBFNN (SVM) into Eq. (7).

2.1 Stochastic sensitivity measure of RBFNN

The functional form of the RBFNN using Gaussian activation function is defined as

$$f_\theta(\mathbf{X}) = \sum_{j=1}^{M} w_j \exp\left(\frac{\|\mathbf{X}-\mathbf{U}_j\|^2}{-v_j^2}\right) + \beta, \tag{8}$$

where β, M, w_j, v_j and $\mathbf{U}_j$ denote the bias term, the number of hidden neurons, connection weights between the jth hidden neuron and the output, the width and center values of the jth hidden neuron, respectively. We do not have the knowledge about the distribution of the unseen samples and thus uniform distribution is applied in the computation of the ST-SM. Such that, all of the unseen samples have the same chance to occur in future in our model. Furthermore, we make no assumption on the distribution of the inputs. By the central limit theorem, the sum of random variables tends to a normal distribution. That is, the inputs to the hidden neurons ($s_j = \|\mathbf{X}^{(b)} - \mathbf{U}_j\|^2$ and $s_j^* = \|\mathbf{X}^{(b)} + \Delta\mathbf{X} - \mathbf{U}_j\|^2$) could be assumed to have normal distributions and therefore the output of the hidden neurons have log-normal distributions. By using these assumptions, we have the ST-SM for RBFNN to be defined as [5]

$$\begin{aligned}
&E\left((\Delta y)^2\right)\\
&= E\left(\left[\sum_{j=1}^{M} w_j \exp\left(\frac{s_j^*}{-2v_j^2}\right) - \sum_{j=1}^{M} w_j \exp\left(\frac{s_j}{-2v_j^2}\right)\right]^2\right)\\
&= \sum_{j=1}^{M}\left[\varphi_j\left(\begin{array}{l}\exp\left(\dfrac{4\sum_{i=1}^{N}\sigma_{\Delta x_i}^2\left(\sigma_{x_i}^2+\left(\mu_{x_i}-u_{ji}\right)^2+0.2\sigma_{\Delta x_i}^2\right)}{2v_j^4} - \dfrac{2\sum_{i=1}^{N}\sigma_{\Delta x_i}^2}{2v_j^2}\right)\\ -2\exp\left(\dfrac{\sum_{i=1}^{N}\sigma_{\Delta x_i}^2\left(\sigma_{x_i}^2+\left(\mu_{x_i}-u_{ji}\right)^2+0.2\sigma_{\Delta x_i}^2\right)}{2v_j^4} - \dfrac{\sum_{i=1}^{N}\sigma_{\Delta x_i}^2}{2v_j^2}\right)+1\end{array}\right)\right]\\
&\approx \sum_{j=1}^{M}\varphi_j\left(\frac{\sum_{i=1}^{N}\sigma_{\Delta x_i}^2\left(\sigma_{x_i}^2+\left(\mu_{x_i}-u_{ji}\right)^2+0.2\sigma_{\Delta x_i}^2\right)}{v_j^4}\right)\\
&= \sum_{j=1}^{M}\varphi_j\left(\frac{\sum_{i=1}^{N}\frac{Q^2}{3}\left(\sigma_{x_i}^2+\left(\mu_{x_i}-u_{ji}\right)^2+0.2\frac{Q^2}{3}\right)}{v_j^4}\right)\\
&= \frac{1}{3}Q^2\sum_{j=1}^{M}\upsilon_j + \frac{0.2}{9}Q^4N\sum_{j=1}^{M}\zeta_j,
\end{aligned} \tag{9}$$

$$\text{where } \varphi_j = \left(w_j\right)^2\exp\left(\left(\mathrm{Var}\left(s_j\right)\Big/ 2v_j^4\right) - \left(E\left(s_j\right)\Big/ v_j^2\right)\right),$$

$$E\left(s_j\right) = \sum_{i=1}^{N}\left(\sigma_{x_i}^2+\left(\mu_{x_i}-u_{ji}\right)^2\right), \zeta_j = \varphi_j/v_j^4,$$

$$\begin{aligned}
Var\left(s_j\right) = \sum_{i=1}^{n}\Big(&E\left[\left(x_i-\mu_{x_i}\right)^4\right] - \left(\sigma_{x_i}^2\right)^2 + 4E\left[\left(x_i-\mu_{x_i}\right)^3\right]\\
&\times\left(\mu_{x_i}+u_{ji}\right) + 4\sigma_{x_i}^2\left(\mu_{x_i}+u_{ji}\right)^2\Big),
\end{aligned}$$

$$\upsilon_j = \varphi_j\left(\sum_{i=1}^{N}\left(\sigma_{x_i}^2+\left(\mu_{x_i}-u_{ji}\right)^2\right)\Big/ v_j^4\right), \mu_i \text{ and } \sigma_{x_i}^2$$

denote the mean and variance of the ith input feature, u_{ji} denotes the ith input feature of the jth center and $\sigma_{\Delta x_i}^2$ denotes the variance of the Δx_i. Furthermore, we assumed Δx_i has a uniform distribution, therefore $\sigma_{\Delta x_i}^2 = (2Q)^2/12 = Q^2/3$.

One may notice that the RBFNN ST-SM increases whenever the complexity of the RBFNN output increases. For example, if the RBFNN outputs fluctuate a lot for a small change in the input values, the RBFNN ST-SM will be large. On the other hand, the RBFNN ST-SM will be zero if the output of the RBFNN does not change for any change in the input values. One may observe from Eq. (7) that even though the ST-SM is zero, the generalization error will still be large if the training error is high. Therefore, a RBFNN with good generalization capability should yield small values in both training error and ST-SM. This indicates that the RBFNN learns the training dataset well and has good consistency in classifying the samples.

2.2 Stochastic sensitivity measure of SVM

The SVM training minimizes the structural risk rather than the empirical risk (training error) as RBFNN does. However, the localized generalization error model (R^*_{SM}) works with a trained classifier and is transparent to the training algorithm of the classifier. Therefore, we could apply the R^*_{SM} to the SVM as long as we concern the MSE of the SVM. As mentioned before, the generalization error bound for entire input space may not be helpful in evaluation of classifiers. For example, when one trains a classifier to distinguish animal images, this is helpless to evaluate this classifier using images of building [6]. The functional form of the SVM using Gaussian kernel function is defined as [8].

$$f_\theta(\mathbf{X}) = \sum_{j=1}^{M} \alpha_j y_j \exp\left(\frac{\left\|\mathbf{X}-\mathbf{X}_j\right\|^2}{-c}\right) + \beta, \tag{11}$$

where $\mathbf{X}_j$, α_j, y_j and c denote the jth support vector, the Lagrange multiplier for the jth support vector, the target output of the jth training sample, the width of the Gaussian kernel, respectively. In Eq. (10), M denotes the number of support vectors. One may notice that the functional forms of RBFNN and SVM with Gaussian kernel are the same after changing some of the variables. Furthermore, the SVM with Gaussian kernel was applied to find the centers for RBFNN in [7]. This was proved that the SVM outperforms the RBFNN for binary classification problem in [7], however the experimental results in [6], showed that this may not be true for multiclass problems. The ST-SM for the SVM with Gaussian kernel is defined as

$$E\left((\Delta y)^2\right) = \sum_{j=1}^{M} \left(\alpha_j y_j\right)^2 \exp\left(\frac{2\mathrm{Var}\left(k_j\right)}{c^2} - \frac{2E\left(k_j\right)}{c}\right) \times \left(\frac{4\sum_{i=1}^{N} \frac{Q^2}{3}\left(\sigma_{x_i}^2 + \left(\mu_{x_i} - x_i^{(j)}\right)^2 + 0.2\frac{Q^2}{3}\right)}{c^2}\right), \tag{12}$$

where $k_j = \left\|\mathbf{X}-\mathbf{X}^{(j)}\right\|^2$, $\mathrm{Var}\left(k_j\right) = \sum_{i=1}^{N}\left(E_D\left[\left(x_i-\mu_{x_i}\right)^4\right] - \left(\sigma_{x_i}^2\right)^2 + 4E_D\left[\left(x_i-\mu x_{x_i}\right)^3\right]\left(\mu_{x_i} - x_i^{(j)}\right) + 4\sigma_{x_i}^2\left(\mu_{x_i} - x_i^{(j)}\right)\right)$, $E\left(k_j\right) = \sum_{i=1}^{N}\left(\sigma_{x_i}^2 + \left(\mu_{x_i} - x_i^{(j)}\right)^2\right)$ and $x_i^{(j)}$ denotes the ith feature value of the jth support vector.

One could find the localized generalization error bound for the SVM with Gaussian kernel by substituting Eq. (11) into Eq. (7). The SVM ST-SM shows the expectation of the squared output differences between training and unseen samples within the Q-union. By comparing Eqs. (3) and (7), one may notice that the ST-SM serves as an alternative metric to the VC-dimensions for measuring the complexity of the SVM. By using the R^*_{SM}, we evaluate a generalization error bound which is different from the conventional one adopted in SVM. The generalization error in SVM evaluated via VC-dimension is a bound for the entire input space, while the R^*_{SM} evaluates the generalization error for SVM in a neighborhood of the training samples in the input space. Furthermore, this may be not very meaningful to discuss the neighborhood in the kernel space because it is an unknown and extremely high dimensional space. The VC-dimension of SVM with Gaussian kernel easily reaches infinity. Remarkably, the R^*_{SM} works in the input space directly and this may be more meaningful to know how much difference between unseen and training samples that could be allowed for a given threshold of generalization error upper bound.

Although the SVM has a sound theoretical foundation, its training requires ad-hoc selection of parameters, e.g. the type of kernel function and the width of the Gaussian kernels. In practice, cross-validation is adopted to repeatedly search for the optimal parameters within a given search space and either the validation error or training error is used as the objective function. The proposed localized generalization error model for SVM could serve as an alternative objective function for one to minimize during the search of the optimal parameters for SVM. In Sect. 3, one will find that the RBFNN found by minimizing the R^*_{SM} outperforms the one minimizing cross-validation error, in terms of both higher testing accuracies and faster training. On the other hand, a more efficient way may be combining the localized generalization error model with the support vector selection and margin optimization in the training of SVM. However, more works are needed to build the training algorithm and related theoretical works, so we may leave it as our future work.

2.3 Selection of the optimal number of hidden neurons for RBFNN

The selection of the optimal number of hidden neurons is an open problem in RBFNN training. In [5], we proposed a method using the R^*_{SM} to select the optimal number of hidden neurons for RBFNN. The maximum coverage classifier with selected generalization (MC^{2}SG) maximizes the Q value for a given threshold of the R^*_{SM}. For two RBFNNs with different number of hidden neurons f_1 and f_2, f_1 yields $R^*_{\mathrm{SM}} = a$ using Q_1 while f_2 yields $R^*_{\mathrm{SM}} = a$ using Q_2. If $Q_1 < Q_2$, by the definition of Q-union, classifier f_2 cover more unseen samples with the same generalization error when comparing with the classifier f_1. Thus, the classifier f_2 yields a better generalization capability.

However, the selection of the number of support vectors is well established in SVM training and thus this may not be helpful to use the R^*_{SM} for it. On the other hand, we compare the RBFNN trained using MC^{2}SGwith standard SVM in Sect. 3. In Sect. 4, we compare the support vectors with sensitive-most vectors (SMV) in decision boundaries visualization. The SMVs are the set of training samples which yield large R^*_{SM} values.

Table 1 Average classification accuracy (%) for testing datasets over ten independent runs

Datasets Methods	MC^{2}SG	5-CV	10-CV	Squen_MSE	Squen_01	SQRT(l)	l	SVM
Breast cancer	97.29	96.99	96.92	97.10	96.43	97.26	93.18	96.62
Heart disease	83.24	82.41	81.37	79.17	60.37	82.59	83.43	84.05
Pima diabetes	79.40	75.86	75.55	72.68	54.37	77.53	32.94	76.69

Table 2 Total training time over ten independent runs

Datasets Methods	MC^{2}SG	5-CV	10-CV	Squen_MSE	Squen_01	SQRT(l)	l	SVM
Breast cancer	1	3360	10127	1	13	1	18	20
Heart disease	2	101	314	36	29	1	2	5
Pima diabetes	13	2806	8241	707	1012	1	25	51

Table 3 Average number of hidden neurons and support vectors over ten independent runs

Datasets Methods	MC^{2}SG	5-CV	10-CV	Squen_MSE	Squen_01	SQRT(l)	l	SVM
Breast cancer	2.1	25.1	27.8	2.0	42.8	19.0	350.0	76.8
Heart disease	8.8	11.3	16.5	52.5	51.8	12.0	135.0	104.2
Pima diabetes	22.0	54.0	57.2	275.2	291.6	20.0	384.0	236.6

3 Experimental results for pattern classification problems

Three real world datasets from the UCI machine learning repository are used to compare the classification accuracies between the RBFNN trained using MC^{2}SG and SVM with Gaussian kernel. All of them are binary classification problems. Ten independent runs are performed for each of the datasets and 50% of samples are selected as the training dataset in each run. The SVM is trained with parameter optimized using cross-validations. Moreover, the experimental results are also compared with six widely adopted architecture selection techniques for RBFNN: two cross-validation methods, two sequential learning methods and two ad-hoc choices of the number of hidden neurons. As suggested in [4], five-folds (5-CV) and ten-folds (10-CV) cross-validations are used in the experiments. 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 l is the number of training samples which uses every training sample as a center of RBFNN hidden neuron. The ad-hoc method "SQRT(l)" method is to select the number of hidden neurons so that it is equal to the square root of the number of training samples.

Table 1 presents the average testing accuracy of ten independent runs for all the methods. One may notice that the RBFNNs trained using MC^{2}SG outperform the RBFNNs trained using other methods. The sequential learning method Squen_01 performs worse than ad-hoc choice of SQRT(l). This reinforces our thinking that minimizing the classification error only may not be a good choice for real-valued output classifiers. Some of the training samples may have a classification decisions which are just passing the decision threshold. The ad-hoc choice of using all training samples as RBFNN's center performs worst in breast cancer and pima diabetes. This is because the RBFNNs have too large complexity and try to memorize the training samples rather than learning the input–output mapping function. On the other hand, the performance of the RBFNNs trained using MC^{2}SG is worse than SVM in the experiments of the heart disease dataset by 0.8% on average. On the other hand, the MC^{2}SG performs better than SVM in the experiments using the other two datasets by 0.4 and 1.8%, respectively. The experimental results show that the RBFNN trained using MC^{2}SGis comparable to SVM even in binary classification problems.

The total running times for MC^{2}SGfor ten independent runs in experiments for all the three datasets are the second fastest among all the methods (Table 2). The ad-hoc choice of selecting the number of hidden neurons to be the square root of the number of training samples is fast because no selection of parameter is needed. Selecting the number of hidden neurons using cross-validation methods is very time consuming and infeasible for large datasets.

Table 3 shows the average number of hidden neurons selected by various methods and the SVM column shows the number of support vectors for the Gaussian kernel SVM. One may notice that the numbers of support vectors are large in all the experiments. On the other hand, the MC^{2}SG methods finds RBFNNs with fewer number of hidden neurons, but higher testing accuracies. The large numbers of support vectors indicate that the samples in the datasets have many overlapping between two classes. The SVMs need to project the training samples to a much higher dimensional feature space where the samples from different class are linearly separable.

4 Decision boundaries visualization

Every training sample contributes to the R^*_{SM}. For a given Q value, one could evaluate the generalization error for unseen

samples around the bth training sample ($\mathbf{X}^{(b)}$) by feeding in the $\mathbf{X}^{(b)}$ into the ST-SM equation instead of the whole dataset. That is, the mean values of the training dataset are replaced by the feature values of $\mathbf{X}^{(b)}$ and all the variance of inputs equal to zero. In Eq. (7), both the constants and training error are fixed after we trained the classifier, The $R^*_{\mathrm{SM}}$of a training sample depends only on the ST-SM. Therefore, we denote the set of training samples which yields large ST-SM by SMV. The sample yielding a large value in ST-SM indicates that the classifier outputs around this sample fluctuate a lot. One may expect this sample located near to one of the decision boundaries in the input space of the classifier.

On the other hand, support vectors (SV) of SVM are defined to be the set of training samples which allocated nearest to the decision boundary between two classes. This is based on the assumption in SVM that the samples from the two classes are linearly separable in the projected kernel space.

One may notice that both SVs and SMVs are the sets of training samples which allocated nearest to the decision boundaries of the classifier. However, the SVs are found in the kernel space which is projected by the kernel functions and usually has a dimension much larger than the original input space. In contrast, the SMVs are found in the original input space. More importantly, the SMV does not require the assumption of linearly separability because the SMVs are those samples which yields large ST-SM. In [7] and the following figures show that the SVs for SVM using Gaussian kernel surround the clusters rather than locating on the decision boundaries when we plot them in the input space. In contrast, the SMVs visualize the decision boundaries between classes.

We use two three-class problems from the UCI machine learning repository to demonstrate the differences between the SVs and SMVs. We plot only two features for each dataset to facilitate the visualization. These features are selected if they visually separating the classes. The UCI IRIS dataset consists of three classes and one of the class is linearly separable. Figure 1 shows the distribution of the samples in various classes with the third and fourth features. For the SVMs, the one-against-all method is adopted to deal with the multiclass problem.

From Fig. 2, one may observe that the SMVs describe the decision boundaries very well. However, the SVs in Fig. 3 surround the samples in each class rather than describing the decision boundaries in the input space.

Figure 4 describes the distribution of the samples in the 7th and 12th features of the UCI wine dataset. One may find that this dataset is more complicated when compared with the UCI IRIS dataset. The SMVs, again, excellently describe the decision boundaries in Fig. 5. In contrast to the SMVs, the SVs in Fig. 6 seem to be randomly distributing all around the input space.

These figures show that the SMVs could visualize the decision boundaries of the classifier in the input space. In a two-dimensional space, one could find the decision boundaries by human eye and draw the intuitive decision boundaries. However, when the number of features is larger than 3, this is very difficult to visualize it by human. The SMVs provide a systematic way to visualize the decision boundaries of the classifier. These decision boundaries indicate the region of the input space where the classifier changes its decisions.

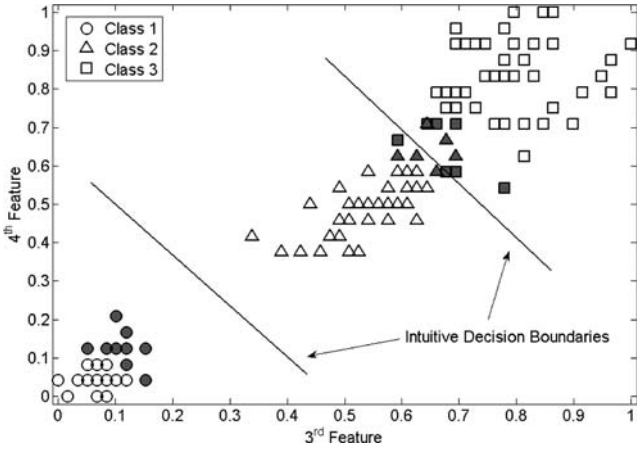

Fig. 2 Distribution of the SMVs (Shaded Points) for UCI IRIS dataset

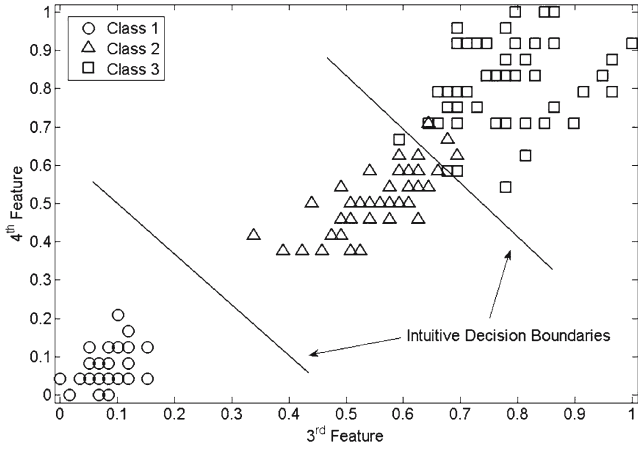

Fig. 1 Distribution of the samples for UCI IRIS dataset

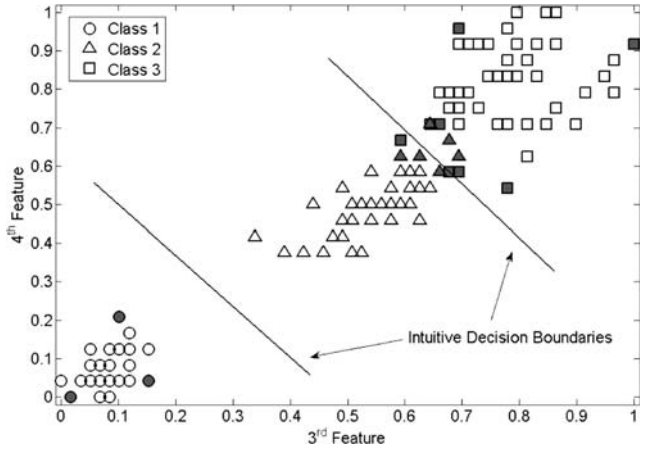

Fig. 3 Distribution of the SVs (Shaded Points) for UCI IRIS dataset

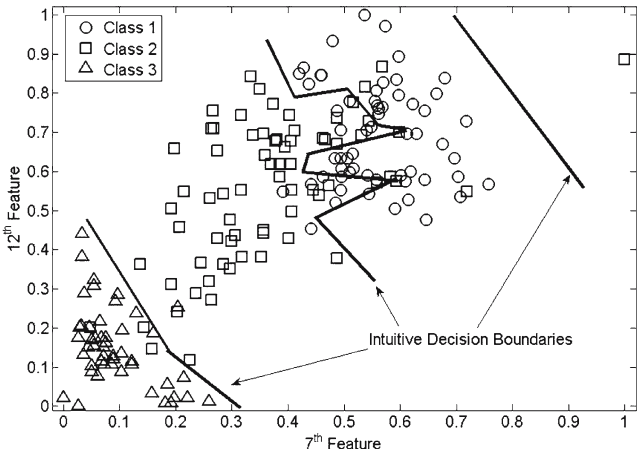

Fig. 4 Distribution of the samples for UCI wine dataset

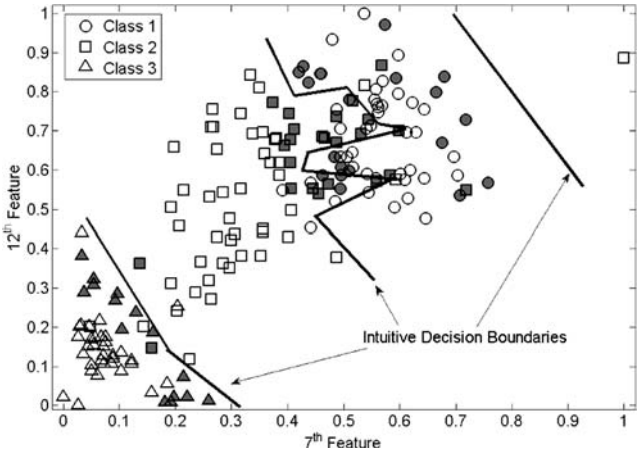

Fig. 5 Distribution of the SMVs (shaded points) for UCI wine dataset

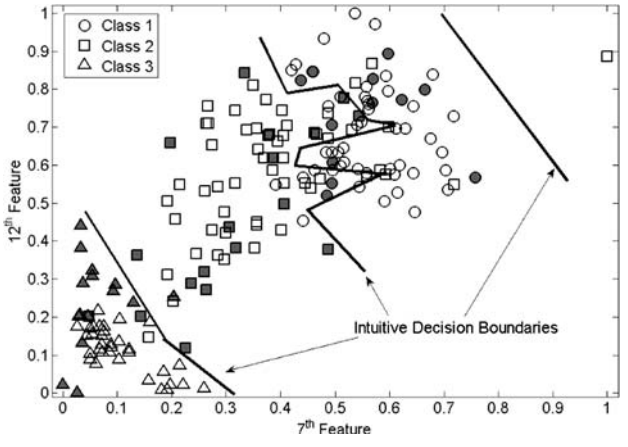

Fig. 6 Distribution of the SVs (shaded points) for UCI wine dataset

However, one may notice that the decision boundaries of the classifier may not be the same one to the decision boundaries in the real unknown input–output mapping function. Moreover, these decision boundaries may change when the parameters in the classifier change due to the change in decision made by the classifier changes.

The major drawback of using SMVs is that one needs to determine the number of SMVs. In the experiments described in this section, we select the one-third of samples which yield the largest ST-SM, i.e., R^*_{SM}, values. This is an important future work to find an automatic problem independent method to select the number of SMVs.

5 Conclusion

In this paper, we described the localized generalization error models for Gaussian-based classifiers. In particular, we demonstrated this model using the RBFNN and SVM with Gaussian kernel. Furthermore, this model could be easily extended to other Gaussian-based classifier by defining the corresponding ST-SM. Moreover, the localized generalization error model of RBFNN was adopted to select the optimal number of hidden neurons for RBFNN. The resulting RBFNNs were compared with standard SVMs and found comparable testing accuracies for binary classification problems.

On the other hand, we visualized the decision boundaries of classifiers in the input space using the localized generalization error model. This helps in better understanding the trained classifier. Furthermore, this is an interesting future research topic of applying the SMV to find a multiclass SVM to replace the multiple binary SVMs for solving multiclass classification problems.

Acknowledgements This research work is supported by the Hong Kong Research Grant Council under the Grant B-Q944 and the Hong Kong Polytechnic University Research Grant GT-891.

References

1. Anthony M, Bartlett PL (1999) Neural network learning: theoretical foundations. Cambridge University Press, Cambridge
2. Bartlett PL, Williamson RC (1996) The vapnik-chervonenkis dimension and pseudodimension of two-layer neural networks with discrete inputs. Neural Comput 8:653–656
3. Geman S, Bienenstock E (1992) Neural networks and the bias/variance dilemma. Neural Comput 4:1–58
4. Hastie T, Tibshirani R, Friedman J (2001) The element of statistical learning. Springer, Berlin Heidelberg Newyork
5. Ng WWY, Yeung DS, Wang D, Tsang ECC, Wang XZ (2005a) Localized generalization error and its application to RBFNN training. In Proceedings of international conference on machine learning and cybernetics:4667–4673
6. Ng WWY, Dorado A, Yeung DS, Pedrycz W and Izquierdo E (2005b) Image classification with the use of radial basis function neural networks and the minimization of localized generalization error. Pattern Recogn (in press)
7. Scholkopf B, Sung KK, Berges CJC, Girosi F, Niyogi P, Poggio T Vapnik V (1997) Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Trans. Signal Process 45:2758–2765
8. Vapnik V (1998) Statistical learning theory. Wiley-Interscience Publication, New York
9. Chakraborthy D and Pal NR (2003) A Novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning. IEEE Trans. on Neural Networks:1–14