Available online at www.sciencedirect.com
ScienceDirect
Journal of the Franklin Institute 344 (2007) 285–311
Journal of The Franklin Institute
www.elsevier.com/locate/jfranklin

Breast cancer detection from FNA using SVM with different parameter tuning systems and SOM–RBF classifier

Tingting Mu, Asoke K. Nandi*

Signal Processing and Communications Groups, Department of Electrical Engineering and Electronics, The University of Liverpool, Brownlow Hill, L69 3GJ, Liverpool, UK

Received 13 January 2006; received in revised form 12 September 2006; accepted 13 September 2006

Abstract

In this paper, we consider the benefits of applying support vector machines (SVMs), radial basis function (RBF) networks, and self-organizing maps (SOMs) for breast cancer detection. The Wisconsin diagnosis breast cancer (WDBC) dataset is used in the classification experiments; the dataset was generated from fine needle aspiration (FNA) samples through image processing. The 1-norm C-SVM (L_1-SVM), 2-norm C-SVM (L_2-SVM), and υ-SVM classifiers are applied, for which the grid search based on span error estimate (GSSEE), gradient descent based on validation error estimate (GDVEE), and gradient descent based on span error estimate (GDSEE) are developed to improve the detection accuracy. The gradient descent (GD) tuning method based on the span error estimate (SEE) is employed for the L_2-SVM classifier because of its reachable smooth nonlinearity. Such a GDSEE tuning system also has the advantage of saving available samples for the training procedure. The SOM–RBF classifier is developed to improve the performance of only the SOM learning procedure based on distance comparison, in which the RBF network is employed to process the clustering result obtained by the SOM. Experimental results demonstrate that SVM classifiers with the proposed automatic parameter tuning systems and the SOM–RBF classifier can be efficient tools for breast cancer detection, with the detection accuracy up to 98%.
© 2006 The Franklin Institute. Published by Elsevier Ltd. All rights reserved.

Keywords: Breast cancer detection; Support vector machines; Parameter tuning; Radial basis function networks; Self-organizing maps

*Corresponding author.
E-mail addresses: tingting.mu@liverpool.ac.uk (T. Mu), a.nandi@liverpool.ac.uk (A.K. Nandi).

0016-0032/$30.00 © 2006 The Franklin Institute. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.jfranklin.2006.09.005

1. Introduction

Worldwide, breast cancer is the most common form of cancer and the second most common cause of cancer deaths in females, which affects approximately 10% of all women at some stage of their life in the western world [1]. Breast cancer may be detected via a cautious study of clinical history, physical examination, and imaging with either mammography or ultrasound. However, definitive diagnosis of a breast mass can only be established through fine-needle aspiration (FNA) biopsy, core needle biopsy, or excisional biopsy [2]. Among these methods, FNA is the easiest and fastest method of obtaining a breast biopsy, and is effective for women who have fluid-filled cysts. FNA uses a needle smaller than those used for blood tests to remove fluid, cells, and small fragments of tissue for examination under a microscope. Research works on the Wisconsin diagnosis breast cancer (WDBC) data grew out of the desire of Dr. Wolberg to diagnose breast masses accurately based solely on FNA [3–5]. Previously, researchers from the University of Wisconsin, Madison, applied image processing techniques to derive the WDBC dataset directly from digital scans of FNA slides [3,4,6], and then employed machine learning techniques to differentiate benign from malignant samples [4,5,7], which could be the earliest study of machine learning application to breast cancer detection. Later, several works on computational intelligence were developed in the area of breast cancer, including the multilayer perceptron [8], radial basis function (RBF) networks [9], fuzzy classifiers [10,11], clustering algorithms [12], evolutionary computation [11,13,14], principal component analysis [15], and different kernel-based methods [16].

Support vector machines (SVMs) are highly successful in solving various nonlinear and non-separable problems in machine learning. In addition to the original C-SVM learning method proposed by Cortes and Vapnik [17] in 1995, the υ-SVM learning method was proposed by Schölkopf et al. [18] in 2000, which is closely related to the C-SVM but with a different optimization risk. In order to use SVM, one needs to decide the value of the regularization parameter (C for C-SVM and υ for υ-SVM) and the kernel function before training the SVM classifier. In recent years, the SVM has been widely applied to the area of breast cancer detection due to its powerful classification ability. Lee et al. [19] employed a linear SVM classifier to select features for breast cancer detection, and then evaluated the selected features using a general nonlinear SVM classifier with the RBF kernel. EI-Naqa et al. [20] proposed a successive enhancement learning scheme to train the SVM classifier for detection of microcalcification clusters in digital mammography. Land et al. [21] applied a recently designed SVM/generalized regression neural network (GRNN) Oracle hybrid to classify breast lesions, and evaluated its performance as an interpretive aid to radiologists. Among these previous SVM applications to breast cancer detection, it is found that few efforts have been made to study fine tuning of the SVM parameters. The SVM works by first embedding the data into a Euclidean space, by specifying the inner products between each pair of points in the transformed space using the kernel function, and then searching for linear relations among the data points, by maximizing the margin of separation between positive and negative data points in the transformed space. Thus, choosing a suitable kernel function is imperative to the success of the learning process, which is equivalent to selecting suitable kernel parameters for a parametric family of kernels (such as the RBF kernel). The regularization parameter also plays an important part in the SVM learning procedure, which controls the tradeoff between the complexity of an SVM and the number of non-separable points. In this study, different parameter tuning systems are developed for different SVM classifiers, such as the 1-norm C-SVM (L_1-SVM), 2-norm C-SVM (L_2-SVM), and υ-SVM to improve the detection accuracy of malignant masses for breast cancer diagnosis.

The SVM parameter tuning procedure includes the choice of the tuning criterion and the tuning method. As for the tuning criterion, several studies have been done on building different bounds for the error expectation for an SVM, such as the Jaakkola–Haussler bound [22], Opper–Winther bound [23], radius margin bound [24], and span bound [25]. Instead of using the above estimates of the generalized SVM performance, Xiong et al. [26] proposed a new measure of class separability in an empirical feature space. Ayat et al. [27] developed an estimate of the error probability of an SVM classifier. In fact, performing parameter tuning for an SVM based on a certain criterion is equivalent to an unconstrained optimization problem. Grid search (GS) [28], the gradient descent (GD) approach [26,27,29], and evolutionary methods [30,31] are popular optimization methods. In this study, our proposed GDSEE parameter tuning system using the GD approach based on the span error estimate (SEE), developed for the smoothed L_2-SVM classifier, not only efficiently improves the detection accuracy for the WDBC data, but also saves more of the data available for the training procedure.

Self-organizing maps (SOMs) are well-known clustering methods proposed by Kohonen [32], and have proven to be extremely useful for input data with high dimensionality and complexity. For breast cancer detection, classification can be performed based on the SOM clustering result by employing a distance comparison procedure, which is similar to the k-nearest neighbor (kNN) algorithm. However, it is observed in our experiments the detection accuracy of such a distance-comparison-based SOM is not satisfactory (see Table 5). The RBF networks view the design of a neural network as a curve-fitting problem in a high-dimensional space, on which early work was done in 1985 by Powell [33]. In this study, the RBF network is employed to process the SOM clustering result, which can be viewed as SOM–RBF-based dimensionality reduction processing step for a single-layer perceptron (SLP). The combination of the SOMs and RBF networks provides better performance than simply using the distance-comparison-based SOMs for the WDBC data.

This paper is organized as follows: Section 2 provides an explanation of the classification problem studied in this work. Section 3 gives a description of the theory of the SVM, SOM, RBF, GS, and GD approaches. The proposed SVM parameter tuning systems of grid search based on span error estimate (GSSEE), gradient descent based on validation error estimate (GDVEE), and gradient descent based on span error estimate (GDSEE), as well as the SOM–RBF classifier are explained in Section 4. Experimental results and comparative analysis are provided in Section 5. Section 6 concludes the work.

2. The classification problem

The classification problem addressed in this study is the detection of malignant breast tumors from a set of benign and malignant samples, called the WDBC dataset, which was obtained from the University of Wisconsin Hospitals, Madison, available at ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/. Features in this dataset were computed from digitized FNA samples [4,5,7], as follows.

After the FNA sample was taken from a breast mass, the material was mounted on a microscope slide and stained to highlight the cellular nuclei. A portion of well-differentiated cells was scanned using a digital camera. The image analysis software system Xcyt was used to isolate individual nuclei. An approximate boundary of each nucleus was provided as input and taken to convergence to the exact nuclear boundary, using a semi-automatic segmentation procedure called "snakes". Beginning with a user-defined approximate boundary as an initialization, the snake locates the actual boundary of the cell nucleus. In order to evaluate the size, shape and texture of each cell nuclei, 10 characteristics were derived, including the radius, perimeter, area, compactness, smoothness, concavity, concave points, symmetry, fractal dimension, and texture.

(1) *Radius* is computed by averaging the length of radial line segments, which are lines from the center of mass of the boundary to each of the boundary points.
(2) *Perimeter* is measured as the sum of the distances between consecutive boundary points.
(3) *Area* is measured by counting the number of pixels on the interior of the boundary and adding one-half of the pixels on the perimeter, to correct for the error caused by digitization.
(4) *Compactness* combines the perimeter and area to give a measure of the compactness of the cell, calculated as

$$\frac{perimeter^2}{area}.$$

This dimensionless number is minimized for a circle and increases with the irregularity of the boundary.
(5) *Smoothness* is quantified by measuring the difference between the length of each radial line and the mean length of the two radial lines surrounding it. If this number is small relative to the distance between consecutive boundary points, then the contour is smooth in that region. To avoid the numerical instability associated with small divisors, the following equation is used to calculate the smoothness:

$$\frac{\sum_{\text{points}} |r_i - (r_i + r_{i+1})/2|}{perimeter},$$

where r_i is the length of the line from the center of mass of the boundary to each boundary point.
(6) *Concavity* is captured by measuring the size of any indentations in the boundary of the cell nucleus.
(7) *Concave points* is similar to concavity, but counts only the number of boundary points lying on the concave regions of the boundary, rather than the magnitude of such concavities.
(8) *Symmetry* is measured by finding the relative difference in length between pairs of line segments perpendicular to the major axis of the contour of the cell nucleus. The major axis is determined by finding the longest chord, which passes from a boundary point through the center of the nucleus. The segment pairs are then drawn at regular intervals. To avoid numerically unstable results due to extremely small segments, the sums are again divided, rather than summing the quotients,

$$symmetry = \frac{\sum_i |left_i - right_i|}{\sum_i (left_i + right_i)},$$

where $left_i$ and $right_i$ denote the lengths of perpendicular segments on the left and right of the major axis, respectively.

(9) *Fractal dimension* is approximated using the "coastline approximation" described by Mandelbrot [34]. The perimeter of the nucleus is measured using increasingly larger "rulers". As the ruler size increases, the precision of the measurement decreases, and the observed perimeter decreases. Plotting these values on a log–log scale and measuring the downward slope gives the negative of an approximation to the fractal dimension.

(10) *Texture* is measured by finding the variance of the gray-scale intensities in the component pixels.

The mean value, standard error, and the extreme (largest or "worst") value of each characteristic were computed for each image, which resulted in 30 features of 569 images, yielding a database of 569×30 samples representing 357 benign and 212 malignant cases.

Based on the features described above, the detection of malignant breast tumors can be viewed as a binary classification problem. Given a set of l labeled training samples $z = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l} \in (\chi \times Y)$, where χ is the n-dimensional real feature space ($n = 30$) with a binary label space $Y = \{1, -1\}$, $y_i \in Y$ is the label assigned to the sample $\boldsymbol{x}_i \in \chi$, the purpose is to seek a function $\psi : \chi \longrightarrow Y$ that best predicts the label for an input sample. To perform the label prediction task, different machine learning algorithms are applied, including the SVMs, RBF networks, and SOMs.

3. Theory

3.1. Support vector machines (SVMs)

The original idea of the SVM is to construct a hyperplane as the decision surface in such a way that the margin of separation between the positive and negative samples is maximized in an appropriate feature space. As most of the pattern recognition problems to be solved in practice are of nonlinear nature, kernel functions are employed to perform the nonlinear mapping, which computes the inner product matrix, the so-called kernel matrix, on pairs of samples in the transformed feature space. For kernel-based methods, the kernel matrix acts as a bottleneck. All the information available must be extracted from the kernel matrix. Different kernel functions have been designed based on their closure properties [35], among which the RBF kernel is commonly used, given by

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right), \tag{1}$$

where σ is the RBF kernel width set by the user. In the transformed kernel feature space κ, with a nonlinear mapping $\boldsymbol{\phi} : \chi^n \to \kappa$, the separating function $f(\boldsymbol{x})$ has the following form

$$f(\boldsymbol{x}) = \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{\omega})\boldsymbol{\phi}(\boldsymbol{x}) + b = K(\boldsymbol{\omega}, \boldsymbol{x}) + b, \tag{2}$$

where $\boldsymbol{\omega}$ is the vector of weights and b is the bias. In the SVM learning procedure, $f(\boldsymbol{x})$ is first obtained from the training result; the label for an input sample is then estimated by the step function sgn(x):

$$A_z(\boldsymbol{x}) = \mathrm{sgn}(f(\boldsymbol{x})), \tag{3}$$

where sgn(x) is 1 when $x \geqslant 0$, and -1 otherwise. This estimate A_z of ψ is attained by minimizing a regularized empirical risk R_{reg}, of which different choices lead to different SVM classifiers.

3.1.1. Hard-margin SVM

The hard-margin SVM is used for clearly separable cases, of which the regularized optimization risk expected to be minimized is

$$R_{\text{reg}} = \frac{1}{2}\|\boldsymbol{\omega}\|_{\kappa}, \tag{4}$$

subject to

$$y_i(K(\boldsymbol{\omega}, \boldsymbol{x}_i) + b) \geqslant 1, \quad i = 1, 2, \ldots, l.$$

This is equivalent to maximizing

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{5}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0,$$

$$\alpha_i \geqslant 0, \quad i = 1, 2, \ldots, l.$$

3.1.2. C-SVM learning

The C-SVM is a soft-margin SVM for non-separable cases, including the L_2-SVM and L_1-SVM, which introduces the margin slack vector $\boldsymbol{\xi}$ to allow for the possibility of samples violating the following inequality:

$$y_i(K(\boldsymbol{\omega}, \boldsymbol{x}_i) + b) \geqslant 1, \quad i = 1, 2, \ldots, l,$$

with the soft-margin loss

$$c(\boldsymbol{x}, y, f) = \begin{cases} 0, & yf(\boldsymbol{x}) \geqslant 1, \\ 1 - yf(\boldsymbol{x}) & \text{otherwise.} \end{cases} \tag{6}$$

The L_1-SVM attempts to minimize the regularized optimization risk by involving the 1-norm of the margin slack vector $\boldsymbol{\xi}$ as

$$R_{\text{reg}} = \frac{1}{2}\left(\|\boldsymbol{\omega}\|_{\kappa} + C\sum_{i=1}^{l} \xi_i \right), \tag{7}$$

subject to

$$y_i(K(\boldsymbol{\omega}, \boldsymbol{x}_i) + b) \geqslant 1 - \xi_i, \quad i = 1, 2, \ldots, l,$$

$$\xi_i \geqslant 0, \quad i = 1, 2, \ldots, l.$$

This is equivalent to maximizing

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{8}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0,$$

$$0 \leqslant \alpha_i \leqslant C, \quad i = 1, 2, \ldots, l,$$

where C is the positive regularization parameter set by the user.

The L_2-SVM attempts to minimize the regularized optimization risk by involving the 2-norm of the margin slack vector $\boldsymbol{\xi}$ as

$$R_{\text{reg}} = \frac{1}{2}\left(\|\boldsymbol{\omega}\|_{\kappa} + C\sum_{i=1}^{l} \xi_i^2 \right), \tag{9}$$

subject to

$$y_i(K(\boldsymbol{\omega}, \boldsymbol{x}_i) + b) \geqslant 1 - \xi_i, \quad i = 1, 2, \ldots, l.$$

This is equivalent to maximizing

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \left(K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{C}\delta_{ij} \right), \tag{10}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0,$$

$$\alpha_i \geqslant 0, \quad i = 1, 2, \ldots, l,$$

where C is the regularization parameter set by the user, and δ_{ij} is the Kronecker δ, which is 1 when $i = j$, and 0 otherwise. In fact, the L_2-SVM can be viewed as a special case of the hard-margin SVM with the modified kernel matrix

$$\mathbf{K}' = \mathbf{K} + \frac{1}{C}I,$$

where $\mathbf{K}$ is the original kernel matrix, and I is the identity matrix.

3.1.3. υ-SVM learning

The υ-SVM is also a soft-margin SVM for non-separable cases, which employs the same margin slack vector $\boldsymbol{\xi}$ as in the C-SVM learning procedure, but a different soft-margin loss, given by

$$c(\boldsymbol{x}, y, f) = \begin{cases} 0, & yf(\boldsymbol{x}) \geqslant \rho, \\ \rho - yf(\boldsymbol{x}) & \text{otherwise}, \end{cases} \tag{11}$$

where ρ denotes the margin width varying through positive values. Thus, the C-SVM can be viewed as a special case of the υ-SVM with the margin width ρ equal to 1. The υ-SVM procedure attempts to minimize the following regularized optimization risk:

$$R_{\text{reg}} = \frac{1}{2}\|\boldsymbol{\omega}\|_{\kappa} + \frac{1}{l}\sum_{i=1}^{l}\xi_i - \upsilon\rho, \tag{12}$$

subject to

$$y_i(K(\boldsymbol{\omega}, \boldsymbol{x}_i) + b) \geqslant \rho - \xi_i, \quad i = 1, 2, \ldots, l,$$

$$\xi_i \geqslant 0, \quad i = 1, 2, \ldots, l,$$

$$\rho \geqslant 0.$$

This is equivalent to maximizing

$$Q(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{13}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0,$$

$$\sum_{i=1}^{l} \alpha_i = \upsilon,$$

$$0 \leqslant \alpha_i \leqslant \frac{1}{l}, \quad i = 1, 2, \ldots, l,$$

where υ is the regularization parameter varying through $[0, 1]$, set by the user.

3.1.4. Decision of the separating hyperplane

The concept of support vectors (SVs) is derived from the Karush–Kuhn–Tucker (KKT) condition. Letting $\boldsymbol{\alpha}^0$ denote the optimal solution of $Q(\boldsymbol{\alpha})$, SVs are defined as the training samples with non-zero α_i^0. Margin SVs are defined as the training samples with α_i^0 not equal to zero, but less than C for the L_2-SVM, or less than υ for the υ-SVM, which are distributed along the margin (see Fig. 1). Non-margin SVs are defined as the training samples with α_i^0 equal to C for the L_2-SVM, or equal to υ for the υ-SVM, which are distributed either inside the margin but on the correct side of the decision surface, or on the wrong side of the decision surface (see Fig. 1).

For the C-SVM and υ-SVM learning methods, the optimal weights $\boldsymbol{\phi}(\boldsymbol{\omega}^0)$ and bias b^0 of the separating hyperplane in the transformed kernel feature space can be derived from

$$\boldsymbol{\phi}(\boldsymbol{\omega}^0) = \sum_{i=1}^{l}\alpha_i^0 y_i \boldsymbol{\phi}(\boldsymbol{x}_i), \tag{14}$$

$$b^0 = -\frac{1}{2S}\sum_{\boldsymbol{x}\in S_+ \bigcup S_-}\sum_{i=1}^{l}\alpha_i^0 y_i K(\boldsymbol{x}, \boldsymbol{x}_i). \tag{15}$$

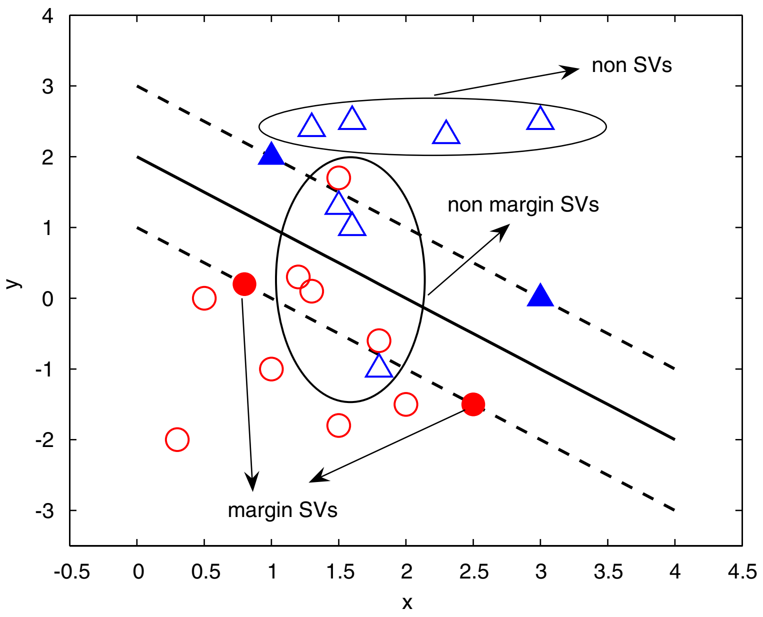

Fig. 1. Illustration of support vectors for linear, non-separable patterns.

For the υ-SVM learning method, the optimal value of ρ^0 can be derived from

$$\rho^0 = -\frac{1}{2S}\left[\sum_{x\in S_+}\sum_{i=1}^{l}\alpha_i^0 y_i K(\boldsymbol{x},\boldsymbol{x}_i) - \sum_{x\in S_-}\sum_{i=1}^{l}\alpha_i^0 y_i K(\boldsymbol{x},\boldsymbol{x}_i)\right], \tag{16}$$

where S_+ and S_- are two sets of SVs with the same size as that of S, but different labels of -1 and 1, respectively. The optimal values of the slack vector $\boldsymbol{\xi}^0$ can be derived from the KKT conditions:

for the L_1-SVM,

$$\xi_i^0 = \max\{0, 1 - y_i f(\boldsymbol{x}_i)\}, \quad i = 1, 2, \ldots, l, \tag{17}$$

and for the υ-SVM,

$$\xi_i^0 = \max\{0, \rho^0 - y_i f(\boldsymbol{x}_i)\}, \quad i = 1, 2, \ldots, l. \tag{18}$$

3.1.5. *Span error estimate (SEE)*

For descriptive convenience, $V(\boldsymbol{\alpha})$, $M(\boldsymbol{\alpha})$, and $N(\boldsymbol{\alpha})$ are used to denote all the SVs, margin SVs, and non-margin SVs, respectively, in the solution $\boldsymbol{\alpha}$; $V_p(\boldsymbol{\alpha})$, $M_p(\boldsymbol{\alpha})$, and $N_p(\boldsymbol{\alpha})$ are used to denote all the SVs, margin SVs, and non-margin SVs, without the pth SV, respectively, in the solution $\boldsymbol{\alpha}$; $|\cdot|$ is used to denote the number of elements in the input set.

The SEE of an SVM classifier is calculated based on the concept of the span, which is defined as the C-span, and υ-span, respectively, for the C-SVM and υ-SVM learning methods.

Definition 1. The C-span of the pth SV for the C-SVM learning method is defined as [36]

$$S^2_{C,p} = \min\{\|\boldsymbol{x}_p - \hat{\boldsymbol{x}}_p\|^2_\kappa | \hat{\boldsymbol{x}}_p \in \Lambda^C_p\},$$

where

$$\Lambda^C_p = \left\{ \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i \boldsymbol{x}_i \middle| \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i = 1, \forall i \in M_p(\boldsymbol{\alpha}^0) : 0 \leqslant \alpha^0_i + \lambda_i y_i y_p \alpha^0_p \leqslant C \right\}.$$

Definition 2. The υ-span of the pth SV for the υ-SVM learning method is defined as [36]

$$S^2_{\upsilon,p} = \min\{\|\boldsymbol{x}_p - \hat{\boldsymbol{x}}_p\|^2_\kappa | \hat{\boldsymbol{x}}_p \in \Lambda^\upsilon_p\},$$

where

$$\Lambda^\upsilon_p = \left\{ \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i \boldsymbol{x}_i \middle| \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i = 1, \right.$$
$$\left. \sum_{i \in M_p(\boldsymbol{\alpha}^0)} y_i y_p \lambda_i = 1, \forall i \in M_p(\boldsymbol{\alpha}^0) : 0 \leqslant \alpha^0_i + \lambda_i y_i y_p \alpha^0_p \leqslant \frac{1}{l} \right\}.$$

Under the assumption that the sets of margin and non-margin SVs remain the same during the leave-one-out (LOO) procedure, the definitions of Λ^C_p and Λ^υ_p are simplified as

$$\Lambda^C_p = \left\{ \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i \boldsymbol{x}_i \middle| \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i = 1 \right\}, \tag{19}$$

$$\Lambda^\upsilon_p = \left\{ \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i \boldsymbol{x}_i \middle| \sum_{i \in M_p(\boldsymbol{\alpha}^0)} \lambda_i = 1, \sum_{i \in M_p(\boldsymbol{\alpha}^0)} y_i y_p \lambda_i = 1 \right\}, \tag{20}$$

where the removed constraints are viewed as true by default.

Consequently, the SEE for the C-SVM and υ-SVM learning methods are calculated based on the following two theorems.

Theorem 1. *If $M_p(\boldsymbol{\alpha}) \neq 0$, the C-span in Definition* 1 *exists. The SEE for the C-SVM learning procedure is given by* [36]

$$E^C_S = \frac{1}{l} |L^C(\boldsymbol{\alpha}^0)|,$$

where

$$L^C(\boldsymbol{\alpha}^0) = V(\boldsymbol{\alpha}^0) \cap \{p : S^2_{C,p} \alpha^0_p \geqslant y_p(K(\boldsymbol{\omega}^0, \boldsymbol{x}_p) + b^0)\}.$$

Theorem 2. *If the sets of the margin and non-margin SVs, besides the pth SV $\boldsymbol{x}_p$ being left out, stay the same for every SV $\boldsymbol{x}_p$, the υ-span in Definition 2 exists. The SEE for the υ-SVM learning procedure is given by* [36]

$$E_S^{\upsilon} = \frac{1}{l}|L^{\upsilon}(\boldsymbol{\alpha}^0)|,$$

where

$$L^{\upsilon}(\boldsymbol{\alpha}^0) = V(\boldsymbol{\alpha}^0) \cap \left\{ p : S_{\upsilon,p}^2 \alpha_p^0 \geqslant \frac{\rho^0}{\beta} - \xi_p^0, \beta \geqslant 1 \right\}.$$

For the hard-margin SVM and L_2-SVM, $L^C(\boldsymbol{\alpha}^0)$ can be simplified as

$$L^C(\boldsymbol{\alpha}^0) = V(\boldsymbol{\alpha}^0) \cap \{p : S_{C,p}^2 \alpha_p^0 \geqslant 1\}. \tag{21}$$

The span of the pth SV defined above can also be calculated using the matrix inversion, given by

$$S_p^2 = \frac{1}{(\tilde{\mathbf{K}}^{-1})_{pp}}, \tag{22}$$

where, for the hard-margin SVM and L_2-SVM,

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{K}_V & \boldsymbol{e} \\ \boldsymbol{e}^{\mathrm{T}} & 0 \end{pmatrix}, \tag{23}$$

for the L_1-SVM

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{K}_M & \boldsymbol{e} \\ \boldsymbol{e}^{\mathrm{T}} & 0 \end{pmatrix}, \tag{24}$$

for the υ-SVM

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{K}_M & \boldsymbol{y}_M & \boldsymbol{e} \\ \boldsymbol{y}_M^{\mathrm{T}} & 0 & 0 \\ \boldsymbol{e}^{\mathrm{T}} & 0 & 0 \end{pmatrix}, \tag{25}$$

and $\boldsymbol{y}_M$ is a column vector of labels of all the margin SVs, $\boldsymbol{e}$ is a column vector with all elements equal to 1, $\mathbf{K}_V$ denotes the kernel matrix of all the SVs, and $\mathbf{K}_M$ denotes the kernel matrix of all the margin SVs.

3.2. *Self-organizing maps (SOMs)*

SOMs map a multi-dimensional dataset onto a 1D- or 2D-lattice. Higher-dimensional lattices can also be used, but are not common. The SOMs discussed in this paper consist of two layers of neurons: a 1D input layer and a 2D competitive layer organized as a 2D-lattice of neurons. Each neuron in the competitive layer holds a weight vector $\boldsymbol{\omega}_i$, which has the same dimensionality as the input space. After the initialization of the competitive layer, the input samples are presented to all competitive neurons in parallel. Assuming that there are m neurons in the competitive layer, the best-matching neuron is chosen as the winner, of which the index is denoted by $i(\boldsymbol{x})$ for an input sample $\boldsymbol{x}$, and satisfies

$$i(\boldsymbol{x}) = \arg\min\{\|\boldsymbol{x} - \boldsymbol{\omega}_j\| | j = 1, 2, \ldots, m\}. \tag{26}$$

All the other neurons in the competitive layer, lying inside the topological neighborhood of the winning neuron $i(\boldsymbol{x})$, are adjusted based on the following rule at the kth iteration:

$$\boldsymbol{\omega}_j(k+1) = \boldsymbol{\omega}_j(k) + \eta(k) h_{j,i(x)}(k)(\boldsymbol{x} - \boldsymbol{\omega}_j(k)), \tag{27}$$

and

$$\eta(k) = \eta_0 \exp\left(-\frac{k}{\tau_1}\right),$$

$$h_{j,i(x)}(k) = \exp\left(-\frac{d_{j,i(x)}^2}{2\sigma(k)^2}\right),$$

$$\sigma(k) = \sigma_0 \exp\left(-\frac{n}{\tau_2}\right).$$

Here, $d_{j,i(x)}$ denotes the lateral distance between the winning neuron $i(\boldsymbol{x})$ and the jth excited neuron in the 2D-lattice; η_0, σ_0, τ_1, and τ_2 are training parameters set by the user.

3.3. RBF networks

RBF networks involve three layers, of which the structure is shown in Fig. 2.

(1) *The input layer* is made up of sensory units that connect the network to the environment.
(2) *The hidden layer* acts as a nonlinear transformation from the input space to the hidden space $\boldsymbol{\phi} : \chi \to \chi'$, similar to a kernel function.
(3) *The output layer* supplies the response of the network to the input pattern, which is to seek for an optimal separating function $f : \chi' \to Y$ in the transformed hidden space.

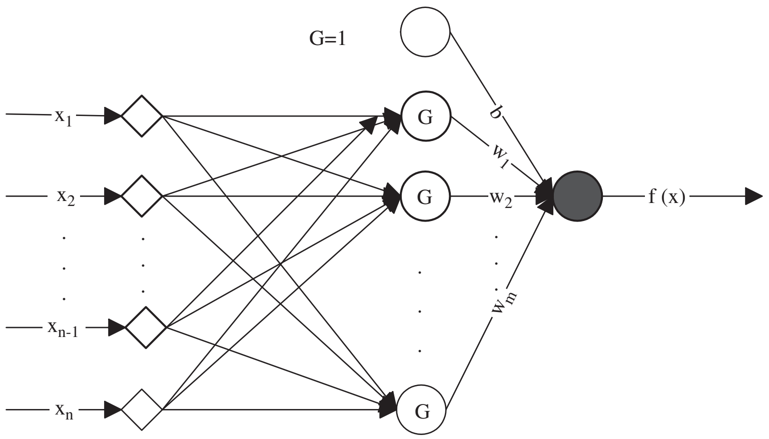

Fig. 2. The overall structure of RBF networks.

The nonlinear transformation $\boldsymbol{\phi} : \chi \to \chi'$ is made up of m real-valued functions

$$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x})]^{\mathrm{T}},$$

where m is the number of neurons in the hidden layer. Given m different points $\{\boldsymbol{t}_i\}_{i=1}^m$, the real-valued functions have the form

$$\phi_i(\boldsymbol{x}) = G(\|\boldsymbol{x} - \boldsymbol{t}_i\|), \quad i = 1, 2 \ldots, m, \tag{28}$$

where $G(\cdot)$ is known as the RBF, and $\{\boldsymbol{t}_i\}_{i=1}^m$ are taken to be the centers of the RBFs. There are different choices of the RBF, among which the Gaussian function is commonly used, as given by

$$G(\boldsymbol{x}, \boldsymbol{t}_i) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{t}_i\|^2}{2\sigma^2}\right). \tag{29}$$

The RBF learning procedure seeks a separating function

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \omega_i G(\|\boldsymbol{x} - \boldsymbol{t}_i\|) + b, \tag{30}$$

by minimizing the cost function

$$\varepsilon(f) = \sum_{i=1}^{l} (y_i - f) + \lambda \|Df\|^2, \tag{31}$$

where D is a linear differential operator. Letting $\boldsymbol{\varpi} = [\omega_1, \omega_2, \ldots, \omega_m, b]^{\mathrm{T}}$, as the regularization parameter λ approaches zero, the optimal value of $\boldsymbol{\varpi}^0$ converges to the pseudo-inverse solution to the over-determined least-squares data-fitting problem for $m < l$, as given by [37]

$$\boldsymbol{\varpi}^0 = (P^{\mathrm{T}} P)^{-1} P^{\mathrm{T}} \boldsymbol{y}, \tag{32}$$

where $\boldsymbol{y}$ is a column vector of labels of all the training samples, and

$$P = \begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1m} & 1 \\ G_{21} & G_{22} & \cdots & G_{2m} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ G_{l1} & G_{l2} & \cdots & G_{lm} & 1 \end{pmatrix},$$

$$G_{ij} = G(\boldsymbol{x}_i, \boldsymbol{t}_j); \quad i = 1, 2, \ldots, l; \; j = 1, 2, \ldots, m.$$

3.4. Grid search (GS)

GS is the most common and reliable optimization method for the simple model selection problem. To tune SVM parameters, GS is executed by varying the SVM parameters through a wide range of values with a fixed step size, assessing the performance of every parameter combination based on a certain criterion, and finding the best setting in the end. Because of its rapidly increased computational cost as the number of parameters increases, GS is only suitable for the adjustment of a small number of parameters, such as the cases in this paper, where only two parameters for each SVM classifier need to be optimized.

3.5. GD approach

The GD approach is a popular derivative-based optimization method, which performs much faster than GS. For the GD-based SVM parameter tuning problem, the step performed in each iteration is proportionally decided by the gradient of a certain optimization objective $O(\boldsymbol{\theta})$ with respect to the SVM parameter vector $\boldsymbol{\theta}$, instead of the fixed step size used in GS. The GD approach is executed by first initializing the SVM parameter vector to some value $\boldsymbol{\theta}_0$, secondly training the L_2-SVM classifier with $\boldsymbol{\theta}_k$ at the kth iteration, and thirdly updating each element of $\boldsymbol{\theta}_k$, so that $O(\boldsymbol{\theta}_k)$ is minimized based on the rule

$$\theta_{k+1,p} = \theta_{k,p} - \eta_p \exp\left(-\frac{k}{\tau}\right)\frac{\partial O(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}}, \quad p = 1, 2, \ldots, |\boldsymbol{\theta}_k|, \tag{33}$$

where $|\boldsymbol{\theta}_k|$ denotes the number of elements in the vector $\boldsymbol{\theta}_k$, and η_p and τ are used to control the convergence speed (set by the user). However, the GD approach only works for differentiable systems.

4. Methodology

4.1. SVM parameter tuning system

The performance of an SVM with a particular dataset depends on the training procedure of the hyperplane weights and bias, as well as the tuning process of the regularization and kernel parameters $\boldsymbol{\theta}$ (see Table 1). The regularization parameter controls the tradeoff between the complexity of the SVM classifier and the number of non-separable points. The kernel parameter decides the nonlinear mapping into an appropriate feature space. The hyperplane weights and bias can be decided by minimizing the regularized empirical risk R_{reg} using quadratic programming. However, the problem of SVM parameter selection is still of much interest. Generally, an SVM parameter tuning system is composed of two blocks, of which one is for criterion generation (the generalized error estimate in this study), and the other is a rapid and accurate tuning process. If there are enough training data, it is possible to split a part of the training data for validation, and obtain a measure known as the validation error estimate (VEE), which is unbiased as the number of validation samples increases. However, if there are not enough training data, one would like to employ all of the available data for training, and generate the error estimate completely on the training set without using any validation set. In this work, three SVM

Table 1
General information of the SVM parameters for different SVM classifiers with RBF kernel

Classifiers	Para.	Kernel para.	Reg. para.
L_1-SVM	$\boldsymbol{\theta} = [\sigma, C]^{\mathrm{T}}$	σ	C
L_2-SVM	$\boldsymbol{\theta} = [\sigma, C]^{\mathrm{T}}$	σ	C
υ-SVM	$\boldsymbol{\theta} = [\sigma, \upsilon]^{\mathrm{T}}$	σ	υ

Para. = Parameters. Reg. = Regularization.

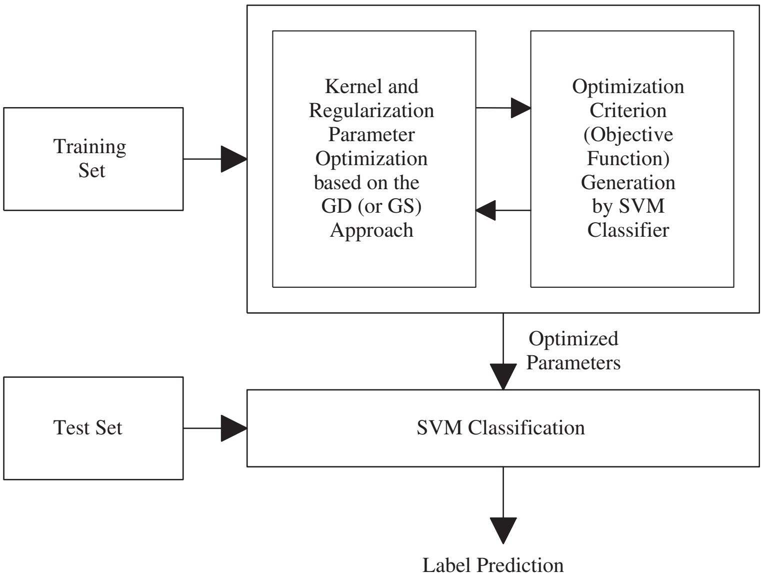
Training Set
Kernel and Regularization Parameter Optimization based on the GD (or GS) Approach
Optimization Criterion (Objective Function) Generation by SVM Classifier
Optimized Parameters
Test Set
SVM Classification
Label Prediction

Fig. 3. The overall structure of the SVM parameter tuning system.

tuning systems are developed based on different combinations of the VEE/SEE criteria and the GS/GD approaches, including the GSSEE, GDVEE, and GDSEE tuning systems. The overall structure of the proposed SVM parameter tuning systems is shown in Fig. 3. The GSSEE tuning system is developed for all the SVM learning methods, such as the L_1-SVM, L_2-SVM, and υ-SVM, whereas the GDVEE and GDSEE tuning systems are only developed for the L_2-SVM, as it is difficult to calculate $\partial\alpha_i^0/\partial C$ and $\partial\alpha_i^0/\partial\upsilon$ analytically for the L_1-SVM and υ-SVM classifiers, which are necessary in the derivative-based optimization procedures.

4.1.1. GSSEE tuning system

The GSSEE tuning system combines the GS approach and the SEE criterion, of which the structure is shown in Fig. 3. In GSSEE tuning, the SVM classifier is trained for various parameter settings, roughly from a large range with a large fixed step to a smaller range with a smaller fixed step. The SEE is calculated for each parameter setting. Parameters with the smallest SEE are finally selected. Based on Theorems 1 and 2 in Section 3.1.5, the objective function $O(\boldsymbol{\theta})$ for tuning of the SVM parameters is calculated as follows:

for the L_1-SVM,

$$O(\boldsymbol{\theta}) = \frac{1}{l}\sum_{p=1}^{s} \mathrm{sgn}\left(S_{C,p}^2(\boldsymbol{\theta})\alpha_p^0(\boldsymbol{\theta}) - y_p\left(K\left(\sum_{j=1}^{l}\alpha_j^0 y_j \boldsymbol{x}_j, \boldsymbol{x}_p\right) + b^0(\boldsymbol{\theta})\right)\right), \tag{34}$$

for the L_2-SVM,

$$O(\boldsymbol{\theta}) = \frac{1}{l}\sum_{p=1}^{s} \mathrm{sgn}(S_{C,p}^2(\boldsymbol{\theta})\alpha_p^0(\boldsymbol{\theta}) - 1), \tag{35}$$

for the υ-SVM,

$$O(\boldsymbol{\theta}) = \frac{1}{l}\sum_{p=1}^{s} \operatorname{sgn}\left(S_{\upsilon,p}^{2}(\boldsymbol{\theta})\alpha_p^0(\boldsymbol{\theta}) + \xi_p^0(\boldsymbol{\theta}) - \frac{\rho^0(\boldsymbol{\theta})}{\beta} \right), \tag{36}$$

where s denotes the number of SVs for an SVM classifier.

4.1.2. GDVEE tuning system

The GDVEE tuning system combines the GD approach and the VEE criterion, of which the structure is shown in Fig. 3. This tuning system is developed only for differentiable SVM classifiers. If k-fold cross-validation is employed, letting U denote the number of folds, the objective function $O(\boldsymbol{\theta})$ for tuning of the SVM parameters is calculated by averaging the validation errors over U groups of training and validation data, where each group's VEE is given by

$$T(\boldsymbol{\theta}) = \frac{1}{V}\sum_{(\mathbf{x}_i', y_i') \in z'} \varphi\left(-y_i' \left(K\left(\sum_{j=1}^{l} \alpha_j^0 y_j \mathbf{x}_j, \mathbf{x}_i' \right) + b^0(\boldsymbol{\theta}) \right) \right), \tag{37}$$

where z' is the set of V labeled validation samples with no intersection with the training samples in the space $\chi \times Y$, and φ is the sigmoid function used to smooth the validation error instead of the step function $\operatorname{sgn}(\cdot)$, given as

$$\varphi(x) = \frac{1}{1 + e^{-Ax}}, \tag{38}$$

where A is selected based on experimental results. Then, the derivatives of the k-fold cross-validation error $O(\boldsymbol{\theta}_k)$ with respect to the pth element of the SVM parameter $\boldsymbol{\theta}$ at the kth iteration are given by

$$\frac{\partial O(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}} = \frac{1}{U}\sum_{i=1}^{U} \frac{\partial T_i(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}}, \quad p = 1, 2, \ldots, |\boldsymbol{\theta}_k|. \tag{39}$$

Letting $\boldsymbol{\alpha}^* = [\boldsymbol{\alpha}^0, b^0]^{\mathrm{T}}$, the derivative of $T_i(\boldsymbol{\theta}_k)$ with respect to the pth element of $\boldsymbol{\theta}$ at the kth iteration is calculated by

$$\frac{\partial T_i(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}} = \left.\frac{\partial T_i(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}}\right|_{\boldsymbol{\alpha}^* \text{fixed}} + \frac{\partial T_i(\boldsymbol{\theta}_k)}{\partial \boldsymbol{\alpha}^*(\boldsymbol{\theta}_k)} \frac{\partial \boldsymbol{\alpha}^*(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}}, \quad p = 1, 2, \ldots, |\boldsymbol{\theta}_k|, \tag{40}$$

and

$$\frac{\partial \boldsymbol{\alpha}^*(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}} = -H^{-1} \frac{\partial H}{\partial \theta_{k,p}} H^{-1} (1 \cdots 1, 0)^{\mathrm{T}}, \tag{41}$$

$$H = \begin{pmatrix} \mathbf{K}_V & \mathbf{y}_V \\ \mathbf{y}_V^{\mathrm{T}} & 0 \end{pmatrix}, \tag{42}$$

where $\mathbf{y}_V$ is a column vector of the labels of all of the SVs.

4.1.3. GDSEE tuning system

The GDSEE tuning system combines the GD approach and the SEE criterion, of which the structure is shown in Fig. 3. This system is also developed for differentiable SVM classifiers. Different from the SEE criterion employed in the GSSEE tuning system, the sigmoid function, given in Eq. (38), is used to smooth the original SEE criterion derived from the step function to get

$$O(\boldsymbol{\theta}) = \frac{1}{l}\sum_{p=1}^{s} \varphi(S_{C,p}^2(\boldsymbol{\theta})\alpha_p^0(\boldsymbol{\theta}) - 1). \tag{43}$$

Thus, the derivative of the SEE criterion with respect to the pth element of the SVM parameter $\boldsymbol{\theta}$ at the kth iteration is calculated by

$$\frac{\partial O(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}} = \frac{1}{l}\sum_{p=1}^{s} \varphi'(S_{C,p}^2(\boldsymbol{\theta}_k)\alpha_p^0(\boldsymbol{\theta}_k) - 1)\left(\alpha_p^0(\boldsymbol{\theta}_k)\frac{\partial S_{C,p}^2(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}} + S_{C,p}^2(\boldsymbol{\theta}_k)\frac{\partial \alpha_p^0(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}}\right), \tag{44}$$

and

$$\frac{\partial S_{C,p}^2(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}} = S_{C,p}^4\left(\tilde{\mathbf{K}}(\boldsymbol{\theta}_k)^{-1}\frac{\partial \tilde{\mathbf{K}}(\boldsymbol{\theta}_k)}{\partial \theta_{k,p}}\tilde{\mathbf{K}}(\boldsymbol{\theta}_k)^{-1}\right)_{pp}, \quad p = 1, 2, \ldots, s, \tag{45}$$

where the matrix $\tilde{\mathbf{K}}(\boldsymbol{\theta}_k)$ is described in Eqs. (23)–(25), and $\partial\alpha_p^0(\boldsymbol{\theta}_k)/\partial\theta_{k,p}$ can be calculated in the same way as shown in Eq. (41).

4.2. SOM– RBF classifier

In the SOM–RBF classifier, of which the overall structure is shown in Fig. 4, the RBF network is employed to process the SOM clustering result on the same training data. In the SOM–RBF classifier, the RBF centers are set as the weight vectors of neurons from the competitive layer of a trained SOM, which are

$$\boldsymbol{t}_i = \boldsymbol{\omega}_i, \quad i = 1, 2, \ldots, m. \tag{46}$$

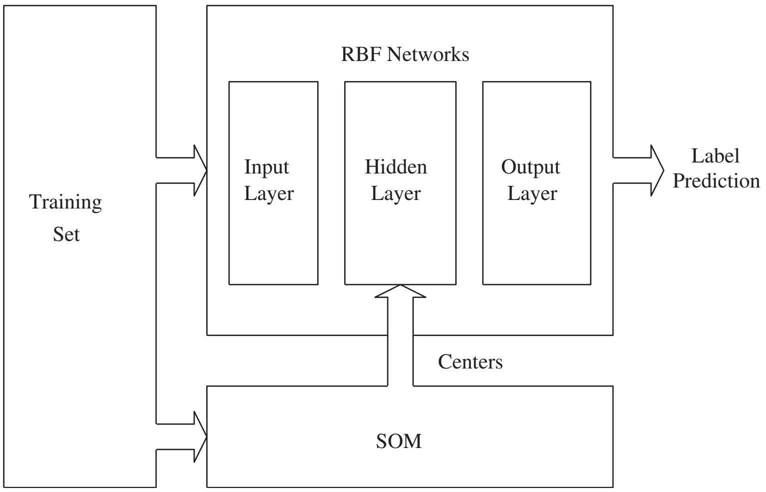

Fig. 4. The overall structure of the SOM–RBF classifier.

In this study, for the RBF networks, we employ the same width for each Gaussian function of Eq. (29), given as

$$\sigma = \frac{d_{\max}}{\sqrt{2m}}, \tag{47}$$

where $d_{\max}$ is the maximum distance between the chosen centers. For a 2D-lattice SOM with a size of $a \times b$, the number of RBF centers m is decided by the SOM topology:

$$m = a \times b.$$

The optimal setting of the SOM size $a \times b$ is selected based on the VEE criterion in this paper.

The SOM–RBF classifier is composed of two learning blocks: the first block learns a nonlinear transformation with a set of RBFs by using the unsupervised learning method of SOM; the second block learns a separating hyperplane in the transformed feature space by minimizing the cost function given in Eq. (31), which is similar to the SLP learning method. In fact, the first block of the SOM–RBF classifier is equivalent to a dimensionality reduction process, which reduces the dimensionality of the input feature space from n to m, corresponding to the second hyperplane-seeking block.

5. Experiments and analysis

In our experiments, the L_1-SVM, L_2-SVM, and υ-SVM classifiers with the different automatic parameter tuning systems of GSSEE, GDVEE and GDSEE, as well as the SOM–RBF classifier were applied to improve the detection accuracy of malignant tumors based on the features generated by FNA (WDBC dataset). All the results are summarized in terms of classification accuracy in percentage based on 10-fold cross-validation. The L_1-SVM and L_2-SVM classifiers were trained with the "SVM and kernel methods Matlab toolbox" [38], available at http://asi.insa-rouen.fr/%7Earakotom/toolbox/index. The υ-SVM classifier was trained with the quadratic optimization functions in the above mentioned toolbox and the Matlab optimization toolbox. The SOMs were trained with the MATLAB neural network toolbox.

The 10-fold cross-validation procedure was executed by randomly dividing the available set of 569 samples into 10 subsets, each with nearly the same size of 57 samples. Ten sets of trials were developed for every classification method. In each trial, one subset was used for testing and the remaining nine subsets for training, so that every subset was used as test data once. Finally, the mean value of the 10 test performance accuracies for the 10 trials was used to represent the generalized performance of the proposed methods.

5.1. SVM diagnosis with the GSSEE tuning system

In the first experiment, to perform diagnosis with the WDBC dataset, the L_1-SVM, L_2-SVM, and υ-SVM classifiers with the RBF kernel were trained separately for various parameter settings. The kernel and regularization parameters were selected using the GS approach based on the SEE criterion. For each SVM classifier, 10 parameter settings with the smallest SEE were adopted for each of the 10 training groups, which were then used to train the remaining nine groups of training sets, respectively, to calculate the corresponding SEE. The final parameters with the smallest mean value of the 10 training groups' SEE were selected. Detection accuracies with the selected parameters are recorded

Table 2
Performance comparison in percentage accuracy for different types of SVM classifiers using the GSSEE parameter tuning system based on 10-fold cross-validation

Classifiers	Kernel para.	Reg. para.	Accuracy in %
L_1-SVM	$\log_{10}\sigma = 0.6$	$\log_{10} C = 0.8$	98.2
L_2-SVM	$\log_{10}\sigma = 1.0$	$\log_{10} C = 1.3$	98.4
υ-SVM	$\log_{10}\sigma = 1.4$	$\log_{10} \upsilon = -0.9$	98.2

Para. = Parameters. Reg. = Regularization.

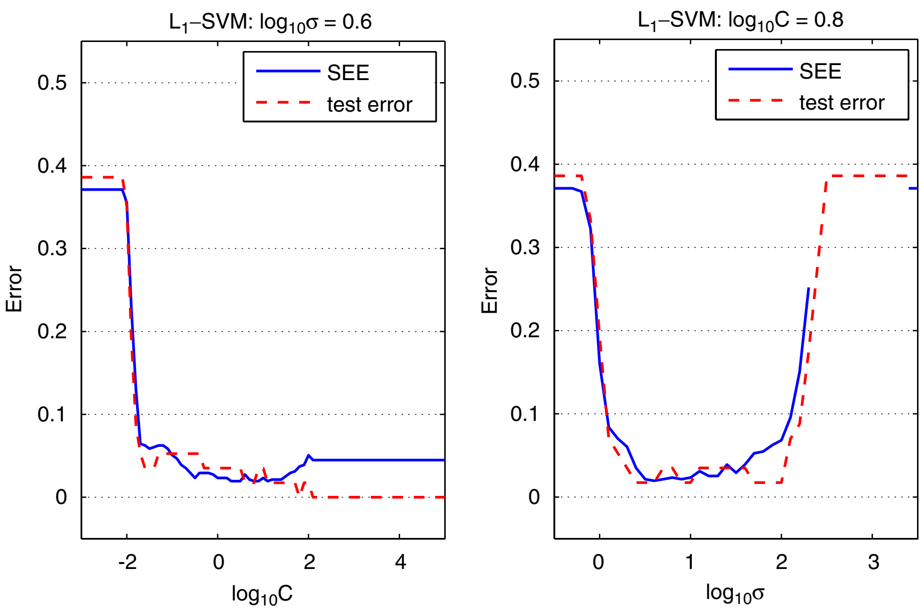

Fig. 5. Comparison of the SEE criterion and test error versus different values of one SVM parameter, with the other fixed, for the L_1-SVM classifier.

in Table 2 for the GSSEE based SVM classifiers, from which it can be seen that all the three SVM classifiers offer similar performance of over 98%.

To verify whether the appropriate kernel and regularization parameters were derived, we compared the SEE criterion with the test error versus different values of one parameter, with the other parameter fixed at the selected value for the L_1-SVM, L_2-SVM, and υ-SVM classifiers, on the same training and test group (see Figs. 5–7). It can be seen that the SEE predicts well the values of the true test error as well as its variation, which yields appropriate parameter selections for all three SVM classifiers. It can also be seen that the performance of an SVM classifier is more sensitive to changes in the kernel parameter than in the regularization parameter. Distributions of the span performance estimate $(1 - SEE)$ and the test performance versus different values of both the kernel and regularization parameters are plotted for the L_1-SVM, L_2-SVM, and υ-SVM with the same training group in Figs. 8–10. For the L_2-SVM, it is clearly seen that there is a large triangular zone in the parameter space where high performance can be reached (see Fig. 8). An SVM classifier

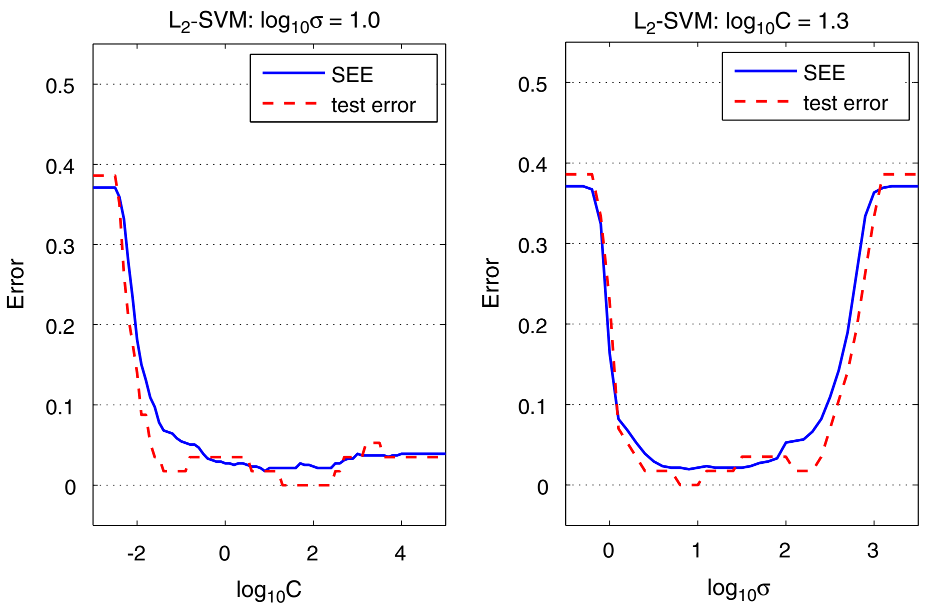

Fig. 6. Comparison of the SEE criterion and test error versus different values of one SVM parameter, with the other fixed, for the L_2-SVM classifier.

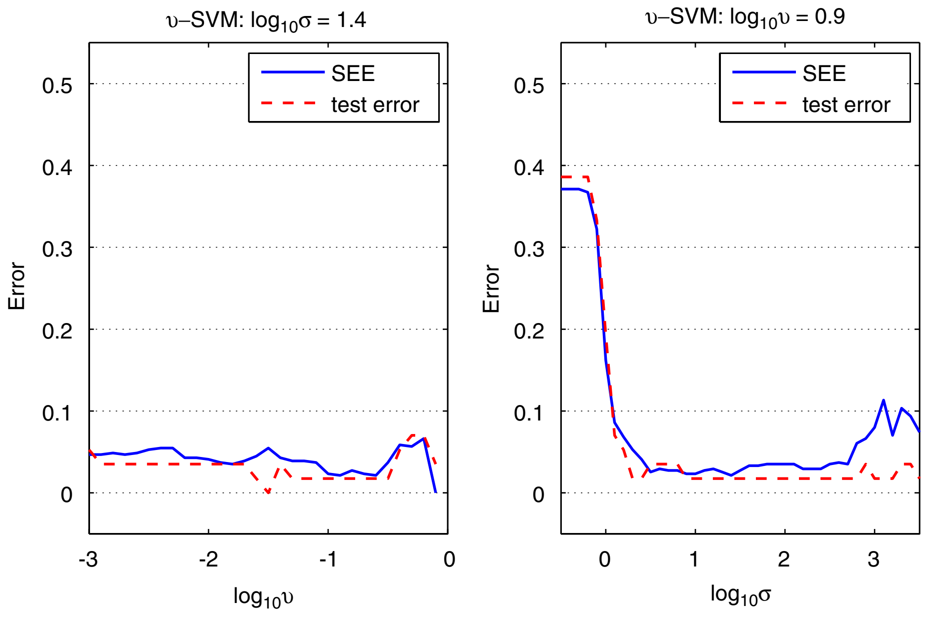

Fig. 7. Comparison of the SEE criterion and test error versus different values of one SVM parameter, with the other fixed, for the υ-SVM classifier.

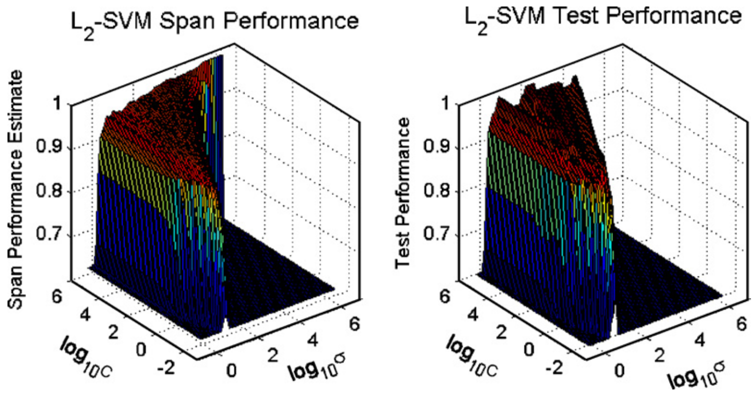

Fig. 8. Comparison of the span and test performance versus different values of the kernel and regularization parameters for the L_2-SVM classifier.

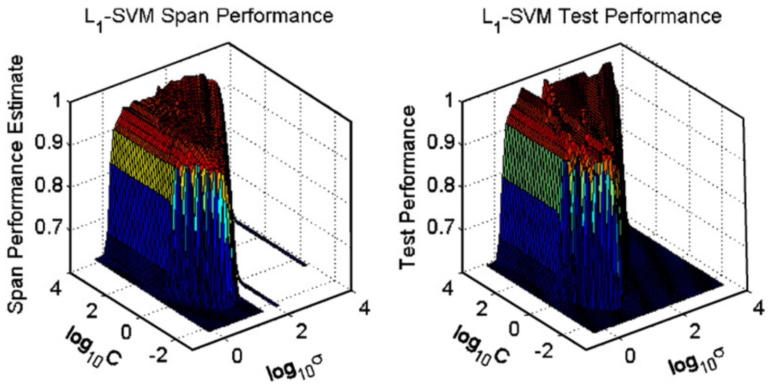

Fig. 9. Comparison of the span and test performance versus different values of the kernel and regularization parameters for the L_1-SVM classifier.

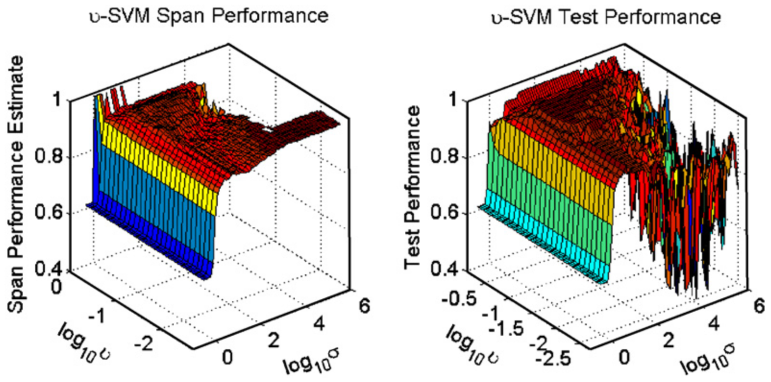

Fig. 10. Comparison of the span and test performance versus different values of the kernel and regularization parameters for the υ-SVM classifier.

with the parameters selected from the central area of the triangular region performs with more stability than with other parameter selections. It is also seen that, for the υ-SVM classifier, the SEE will not exist (due to the appearance of the singular matrices) as υ and σ increase, while the test error becomes noisy and sensitive as υ decreases and σ increases, where the SEE cannot truly predict such a noisy behavior (see Fig. 10). For the L_1-SVM classifier, as C decreases and σ increases, the SEE will not exist either, but can still predict the test error well, as long as it exists (see Fig. 9).

5.2. L_2-SVM diagnosis with the GDVEE and GDSEE tuning systems

In the second experiment, the GDVEE and GDSEE tuning systems were employed to tune the kernel and regularization parameters for the L_2-SVM classifier. For the GDVEE tuning system, the validation data were selected by first training an L_2-SVM classifier using all training samples for each group (with no restriction on the SVM parameter values, $C = 2.4$, and $\sigma = 6.3$), then sorting the training samples based on their values of the separating function $f(\boldsymbol{x}_i)$, and finally selecting the validation data by choosing every ninth sample from the sorted training samples (1:9:512). The sigmoid function given in Eq. (38) was employed to smooth the tuning criteria of VEE and SEE, of which the factor A was set as 10 for VEE and 15 for SEE, respectively. In Fig. 11, the smoothed SEE and VEE criteria

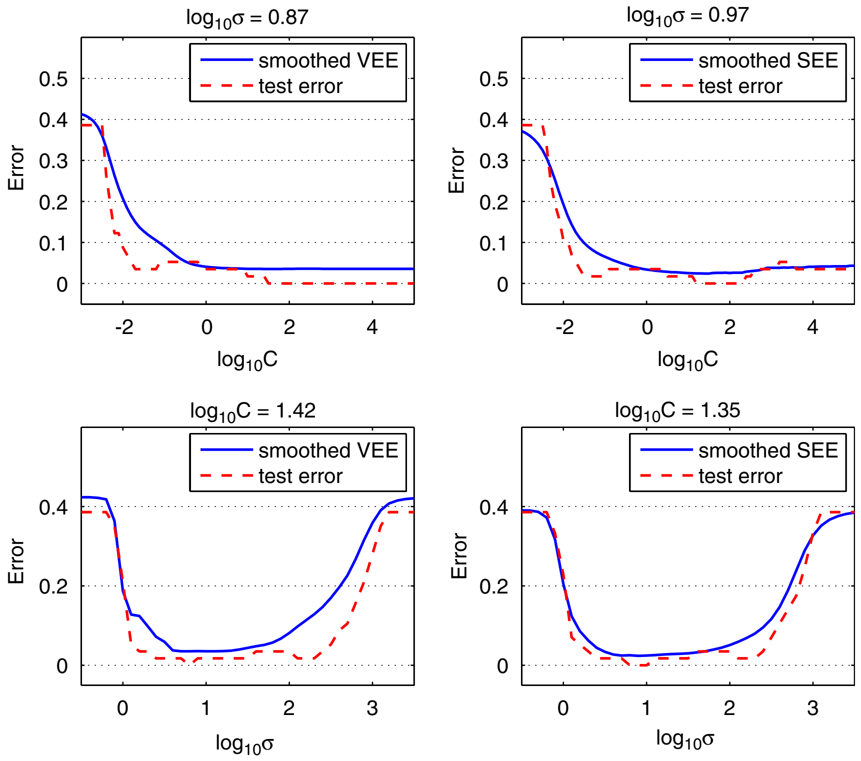

Fig. 11. Comparison of the smoothed VEE/SEE criteria and test error, versus different values of one L_2-SVM parameter, and with the other fixed.

are compared with the test error on the same training and test group, versus different values of one parameter with the other fixed at the selected values.

There are two different ways to calculate the derivatives of the smoothed VEE and SEE, of which one is using Eqs. (39) and (44), and the other is using the following approximation based on the definition of the derivative:

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x+\Delta x)-f(x)}{\Delta x}, \tag{48}$$

where Δx is close to 0. We show a comparison of the above two ways of computing the derivatives of VEE and SEE with respect to C and σ in Fig. 12, from which it is seen that the two are similar to each other. However, to calculate the derivatives using Eq. (48), one has to train the L_2-SVM classifier twice to get the values of $O(\boldsymbol{\theta})$ and $O(\boldsymbol{\theta}+\Delta\boldsymbol{\theta})$, which costs more time as compared with the method using Eqs. (39) and (44). Therefore, we employed Eqs. (39) and (44) to calculate the derivatives.

For GD tuning, the convergence-controlling parameter η was set as 1 for both C and σ. Different values of 500, 800, and 1000 were used for τ. The stopping condition was decided jointly by the iteration number, which was set as 300 in this experiment, and the value of $|O(\boldsymbol{\theta}_{k+1})-O(\boldsymbol{\theta}_k)|$ to overcome the problem of overfitting. Different from the GD approach, as implemented by Chapelle et al. [29], we repeated the GD iterative procedure three or four times, each time with a different initial setting of $\boldsymbol{\theta}$, which is equivalent to finding an optimal area by converging from different directions. The selected parameters

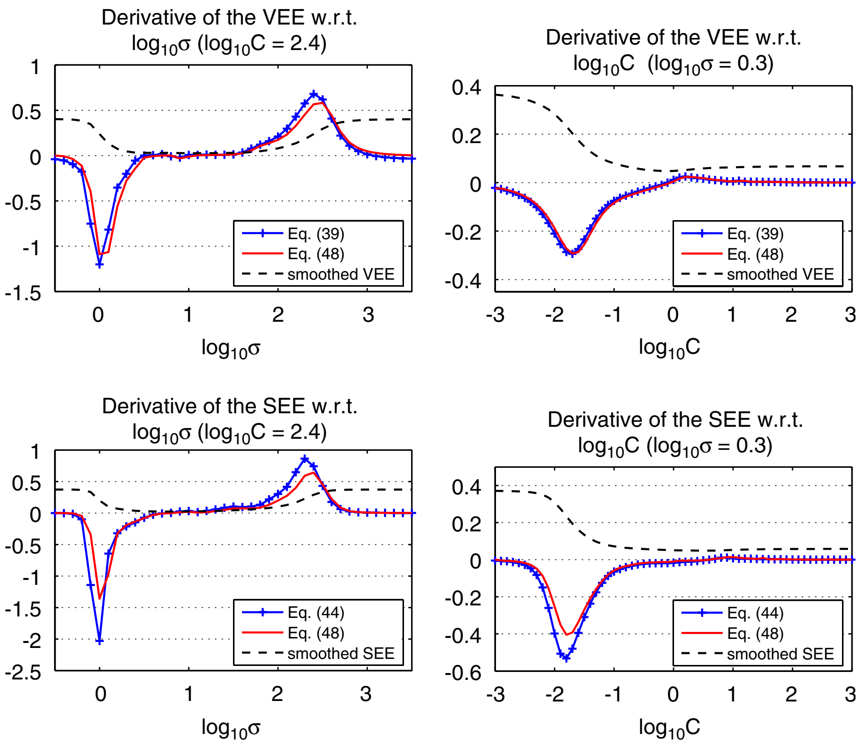

Fig. 12. Derivative comparison of the L_2-SVM classifier, computed by two different ways.

Table 3
Performance comparison in percentage accuracy using different parameter tuning systems for the L_2-SVM classifier based on 10-fold cross-validation

Tuning systems	$\log_{10}\sigma$	$\log_{10}C$	Accuracy (%)
GSSEE	1.0	1.3	98.4
GDVEE	0.87	1.42	98.1
GDSEE	0.97	1.35	98.6

Table 4
Confusion matrix in percentage accuracy of the L_2-SVM classifier with the GDSEE tuning system, where B. denotes benign and M. denotes malignant

Confusion matrix	B. (Predicted)	M. (Predicted)
B. (Actual)	99.4	0.6
M. (Actual)	2.7	97.3

and the corresponding L_2-SVM performance using the different parameter tuning systems of GSSEE, GDVEE, and GDSEE are recorded in Table 3, from which it is seen that the detection accuracy is 98.1% by using the GDVEE tuning system. Although more than 10% of the training samples were sacrificed for validation, this performance is still statistically close to the performance 98.4% reached by using the GSSEE tuning system (see Table 3). With the GDSEE tuning system, a satisfactory performance of 98.6% is reached for the L_2-SVM classifier (see Table 3), of which the confusion matrix is shown in Table 4 with the sensitivity equal to 99.4% and the specificity equal to 97.3%. The SEE has improved the detection performance in comparison with the VEE, with the advantage of making more data available for training. The GDSEE tuning system not only provides a more accurate method for parameter tuning, but also provides more information to train an L_2-SVM classifier.

5.3. SOM– RBF diagnosis

In this experiment, all the training, validation and test data were normalized before being provided as input to the SOM–RBF classifier, for which the 2-D hexagonal grid was employed. The optimal SOM size was selected from 3×2, 6×3, 6×4, 5×5, 7×7, 10×5, and 12×6 based on the average validation error over 10 groups of validation data. Different training epochs of 500, 800, 1000 were used. The final SOM size of 6×4 was selected for the WDBC data. The corresponding detection accuracy was 97.1 % based on 10-fold cross-validation (see the sixth row in Table 5). The SOM–RBF classifier not only improves the performance of the distance-comparison-based SOM satisfactorily, which is only 94.4% in accuracy (see the seventh row in Table 5), but also has a low standard deviation of 0.4%.

5.4. Performance comparison

The performance accuracies for the WDBC data obtained in our experiments are compared with those obtained by the linear SVM classifier [14], the SVM classifier with the

Table 5
Performance comparison in percentage accuracy for different classification methods based on 10-fold cross-validation

Performance	Mean	Std. dev.	Max	Min
L_1-SVM/GSSEE (RBF)	98.2	1.2	100	96.5
υ-SVM/GSSEE (RBF)	98.2	1.5	100	96.5
L_2-SVM/GSSEE (RBF)	98.4	1.3	100	96.5
L_2-SVM/GDSEE (RBF)	98.6	1.1	100	96.5
L_2-SVM/GDVEE (RBF)	98.1	1.9	100	94.1
SOM–RBF	97.1	0.4	97.5	96.1
SOM	94.4	1.8	97.4	89.4
SVM (linear) [14]	94.0	N/A	95.0	N/A
SVM (RBF) [39]	97.7	2.5	N/A	N/A
Fuzzy [11]	95.8	3.1	N/A	N/A
ENN [40]	95.6	N/A	N/A	N/A

RBF kernel [39], the fuzzy classifier using an evolutionary scatter partition of the feature space [11], and the edited nearest-neighbor (ENN) with pure filtering [40] in Table 5. It is seen that all of our SVM classifiers offer higher performance of above 98% and lower standard deviation than the previously reported methods listed. In particular, the L_2-SVM with the GDSEE parameter tuning system offers the highest detection accuracy of 98.6%. The SOM–RBF classifier offers the lowest standard derivation 0.4%, indicating that it is a more stable classifier than the others evaluated.

6. Conclusion

In this study, the L_1-SVM, L_2-SVM, and υ-SVM classifiers with the GSSEE, GDVEE, and GDSEE parameter tuning systems, as well as the SOM–RBF classifier have been applied to improve the detection accuracy of malignant tumors in the WDBC dataset generated from FNA. Experimental results demonstrate the effectiveness of the SVM learning methods with different parameter tuning systems and the SOM–RBF classifier. In our experiments, the average performance of the SVM classifiers is over 98% (see Table 5). Tuning the SVM parameters based on the SEE criterion increases the available training data, which is especially useful when working with a small dataset. The GD-based tuning system for the L_2-SVM classifier can not only speed up the parameter tuning procedure, but also lead to more accurate classification. The SOM–RBF classifier improves the performance of the distance-comparison-based SOM learning procedure. In our experiments, the best performance of 98.6% was reached by using the L_2-SVM classifier with the GDSEE parameter tuning system (see Table 5). It is significant that the values of the standard deviation of the accuracies of our proposed methods are less than those of the other existing classifiers for the same purpose; of the classifiers tested, the SOM–RBF classifier has the lowest standard deviation (see Table 5).

Acknowledgments

T. Mu would like to acknowledge financial support from the Overseas Research Students Awards Scheme (ORSAS), UK, the Hsiang Su Coppin Memorial Scholarship

Fund which is a generous donation of funds to the University of Liverpool from the estate of the late Mrs. Hsiang Su Coppin, and the University of Liverpool, UK. We also thank Prof. Rangayyan, from Department of Electrical and Computer Engineering, University of Calgary, Canada, for his comments on the paper.

References

[1] L. Duijm, J.H. Groenewoud, F.H. Jansen, J. Fracheboud, M. Beek, H.J. de Koning, Mammography screening in the Netherlands: delay in the diagnosis of breast cancer after breast cancer screening, Br. J. Cancer 91 (2004) 1795–1799.

[2] Breast Cancer Diagnostic Algorithms for Primary Care Providers, Breast Expert Workgroup, third ed., Cancer Detection Section, California Department of Health Service, 2005.

[3] W.H. Wolberg, W.N. Street, O.L. Mangasarian, Breast cytology diagnosis via digital image analysis, Anal. Quant. Cytol. Histol. 15 (6) (1993) 396–404.

[4] W.H. Wolberg, W.N. Street, O.L. Mangasarian, Machine learning techniques to diagnose breast cancer from fine-needle aspirates, Cancer Lett. 77 (1994) 163–171.

[5] W.H. Wolberg, W.N. Street, O.L. Mangasarian, Image analysis and machine learning applied to breast cancer diagnosis and prognosis, Anal. Quant. Cytol. Histol. 17 (2) (1995) 77–87.

[6] W.N. Street, W.H. Wolberg, O.L. Mangasarian, Nuclear feature extraction for breast tumor diagnosis, in: Proceedings of IST/SPIE Symposium on Electronic Imaging: Science and Technology, vol. 1905, San Jose, CA, 1993, pp. 861–870.

[7] O.L. Mangasarian, Breast cancer diagnosis and prognosis via linear programming, Cancer Lett. 43 (4) (1995) 570–577.

[8] T.C.S.S. André, R.M. Rangayyan, Classification of breast masses in mammograms using neural networks with shape, edge sharpness, and texture features, J. Electron. Imaging 15 (1) (2006) 13019.

[9] R. Santo, R.D. Lopes, R.M. Rangayyan, Classification of mammographic masses using radial basis functions and simulated annealing with shape, acutance, and texture features, in: Proceedings of the IASTED Conference on Biomedical Engineering, Innsbruck, Austria, 2005, pp. 458–175.

[10] H. Cheng, M. Cui, Mass lesion detection with a fuzzy neural network, Pattern Recognition 37 (6) (2004) 1189–1200.

[11] S.Y. Ho, H.M. Chen, S.J. Ho, T.K. Chen, Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space, IEEE Trans. Syst. Man Cybern. B 34 (2) (2004) 1031–1044.

[12] M.K. Markey, J.Y. Lo, G.D. Tourassi, C.J. Floyd, Self-organizing map for cluster analysis of a breast cancer database, Artif. Intell. Med. 27 (2) (2003) 113–127.

[13] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas, E.L. Michalkiewicz, A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets, Artif. Intell. Med. 30 (1) (2004) 27–48.

[14] H. Guo, A.K. Nandi, Breast cancer diagnosis using genetic programming generated feature, in: Proceedings of the Machine Learning and Signal Processing Workshop, Mystic, CT, USA, 2005, pp. 215–220.

[15] G.M. Palmer, C. Zhu, T.M. Breslin, F. Xu, K.W. Gilchrist, N. Ramanujan, Comparison of multiexcitation fluorescence and diffuse reflectance spectroscopy for the diagnosis of breast cancer, IEEE Trans. Biomed. Eng. 50 (11) (2003) 1233–1242.

[16] L. Wei, Y. Yang, R.M. Nishikawa, Y. Jiang, A study on several machine-learning methods for classification of malignant and benign clustered microcalcifications, IEEE Trans. Med. Imaging 24 (3) (2005) 371–380.

[17] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[18] B. Schölkopf, A. Smola, R.C. Williamson, P. Bartlett, New support vector algorithms, Neural Comput. 12 (5) (2000) 1083–1121.

[19] Y. Lee, O.L. Mangasarian, W.H. Wolberg, Breast cancer survival and chemotherapy: a support vector machine analysis. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 55, (2000) pp. 1–10.

[20] I. El-Naqa, Y. Yang, M.N. Wernick, N.P. Galatsanos, R. Nishikawa, A support vector machine approach for detection of microcalcifications, IEEE Trans. Med. Imaging 21 (12) (2002) 1552–1563.

[21] W.H.J. Land, L. Wong, D.W. Mckee, T. Master, F.R. Anderson Breast cancer computer aided diagnosis (CAD) using a recently developed SVM/GRNN Oracle hybrid, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 5, 2003, pp. 4705– 4711.

[22] T.S. Jaakkola, D. Haussler, Probabilistic kernel regression models in: Proceedings of the Workshop on Artificial Intelligence and Statistics, vol. 7, Morgan Kaufmann, Los Altos, CA, 1999.
[23] M. Opper, O. Winther, Gaussian processes and SVM: mean field and leave-one-out, in: A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 2000, pp. 311–326.
[24] K. Chung, W. Kao, C. Sun, L. Wang, C. Lin, Radius margin bounds for support vector machines with the RBF kernel, Neural Comput. 15 (11) (2003) 2643–2682.
[25] V. Vapnik, O. Chapelle, Bounds on error expectation for support vector machines, in: A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 2000, pp. 261–280.
[26] H. Xiong, M.N.S. Swamy, M.O. Ahmad, Optimizing the kernel in the empirical feature space, IEEE Trans. Neural Networks 16 (2) (2005) 460–474.
[27] N.E. Ayat, M. Cheriet, C.Y. Suen, Automatic model selection for the optimization of SVM kernels, Pattern Recognition 38 (10) (2005) 1733–1744.
[28] C.W. Hsu, C.C. Chang, C.J. Lin, A practical guide to support vector machines, Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, 2003.
[29] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (2002) 131–159.
[30] F. Friedrichs, C. Igel, Evolutionary tuning of multiple SVM parameters, Neurocomputing 64 (2005) 107–117.
[31] D.S. Kim, H. Nguyen, J.S. Park, Genetic algorithm to improve SVM based network intrusion detection system, in: Proceedings of the 19th International Conference on Advanced Information Networking and Applications, 2005, 155–158.
[32] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.
[33] M.J.D. Powell, Radial basis functions for multivariable interpolation: a review, in: Proceedings of the IMA Conference on Algorithms for the Approximation of Functions and Data, 1985, pp. 143–167.
[34] B.B. Mandelbrot, The Fractal Geometry of Nature, W. H. Freeman and Company, New York, 1997 (Chapter 5).
[35] N.C.J. Shawe-Taylor, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, 2004.
[36] A. Gretton, R. Herbrich, O. Chapelle, B. Schölkopf, Estimating the leave-one-out error for classification learning with SVMs, Technical Report CUED/F-INFENG/TR.424, Cambridge University Engineering Department, 2001.
[37] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Syst. 2 (1988) 321–355.
[38] S. Canu, Y. Grandvalet, A. Rakotomam, SVM and Kernel Methods Matlab Toolbox, Perception Systems et Information, INSA de Rouen, Rouen, France, 2003.
[39] Y.I. Chang, S.C. Lin, Synergy of logistic regression and support vector machine in multiple-class classification, in: Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning, Exeter, UK, 2004, pp. 25–27.
[40] W. Lam, C. Keung, C.X. Ling, Learning good prototypes for classification using filtering and abstraction of instances, Pattern Recognition 35 (7) (2002) 1491–1506.