Neurocomputing 73 (2010) 2089–2096

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Tuning SVM parameters by using a hybrid CLPSO–BFGS algorithm

Shutao Li *, Mingkui Tan

College of Electrical and Information Engineering, Hunan University, Changsha 410082, China

ARTICLE INFO

Article history:
Received 16 March 2009
Received in revised form
16 February 2010
Accepted 21 February 2010
Communicated by D. Xu
Available online 25 March 2010

Keywords:
Support vector machine
CLPSO–BFGS method
Model selection

ABSTRACT

Parameter settings of support vector machine (SVM) have a great influence on its performance. Grid search combining with cross-validation and numerical methods by minimizing some generalization error bounds are two usually adopted methods to tune the multiple parameters in SVM. However, the grid search is often time-consuming, especially when dealing with multiple parameters while the numerical methods are very sensitive to the initial value of the parameters. In this paper, we present a hybrid strategy to combine a comprehensive learning particle swarm optimizer (CLPSO) with Broyden–Fletcher–Goldfarb–Shanno (BFGS) method for effectively tuning the SVM parameters based on the generalization bounds. Rather than locating a single local optimum, the hybrid method can identify multiple local optima of the generalization bounds, which can greatly improve the stability of the parameter settings. The experimental results show that the proposed method can efficiently tune the parameters of both L1-SVM and L2-SVM and achieve competitive performance compared with other optimized classifiers.

1. Introduction

Support vector machine (SVM) is a newly developed classification technique, which aims at solving the classification problem by maximizing the margin between two opposite classes [1,2]. One of the important tricks of SVM is the introduction of kernels, which enable SVM to have the ability of dealing with infinite or nonlinear features in a high dimensional feature space. Besides SVM, kernel trick has also been widely used in many other machine learning algorithms, such as principal component analysis (PCA), linear discriminant analysis (LDA) and marginal fisher analysis (MFA) etc., resulting in many powerful kernel-based learning machines [3–5]. With kernels, the linear techniques can be easily extended to handle nonlinear problems, which have shown great improvements compared with the linear methods in many real world applications, such as image recognition, information retrieval and manifold data analysis [6–8]. However, the performance of kernel methods strictly depends on their hyperparameters, especially the kernel parameters that directly control the nonlinear mapping of the features. Therefore, the tuning of parameters, known also as the model selection, plays an important role in kernel methods. In this paper, we mainly concentrate on the model selection of SVM, which has gained great attentions in the last several years. A simple and direct way is to use the grid search on the log-scale of the parameters in combination with the cross-validation procedure on each candidate parameter vector [9,10]. This is an usually used parameter tuning method in other kernel methods [6,8]. However, the exhaustive grid search over the parameter space may result in a large number of model trainings and unacceptably long run time if there are multiple parameters. Another approach is to minimize some generalization bounds, such as the leave-one-out (*LOO*) error bounds, using numerical optimization methods [11–14]. The numerical methods are generally more efficient than grid search. However, owing to the non-convexity of the generalization bounds, these methods may get stuck into local optimum and cause instabilities [9,13]. In other words, the tuning of parameters cannot be fully solved by the numerical optimization methods, which are known for their fast convergence rate but high sensitivity to the initial point. Recently, some global stochastic optimization techniques, such as genetic algorithm (GA), particle swarm optimization (PSO) and simulated annealing (SA) algorithm have been adopted to tune the SVM parameters for their better global search abilities [15–18]. These methods, although can find the global solution in a high probability, are limited by the facts that they usually suffer from the problem of premature convergence, the slow convergence rate and the convergence to a single point.

In this paper, we propose a CLPSO–BFGS method to adjust the parameters of SVM based on two informed *LOO* bounds to address the above problems. CLPSO is an improved version of PSO with better global search abilities [19]. However, it still suffers from the problem of slow convergence rate and the convergence to a single solution, which makes it unreliable in real world applications if the local optimum is not good enough. On the other hand,

* Corresponding author.
E-mail addresses: shutao_li@yahoo.com.cn (S. Li), tanmingkui@gmail.com (M. Tan).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.02.013

although sensitive to the starting point, the numerical optimization methods usually possess very fast convergence rate to reach a local optimum. Hence, in this paper we propose to combine the global search ability of CLPSO and the local search ability of BFGS to tune the SVM parameters. Different from the existing methods, the hybrid method aims at effectively computing multiple global optima of the multimodal functions and therefore can find a group of candidate parameters for SVM. Finally, a better parameter pair can be selected from these candidate parameters.

The remainder of this paper is organized into five sections. Section 2 gives some preliminary studies about the informed *LOO* bounds of SVM. The proposed method will be detailed in Section 3. The experiments are described in Section 4. The discussions and conclusions of this paper are finally presented in Section 5.

2. Preliminary studies

2.1. SVM formulations

For a two-class classification problem, SVM finds a hyperplane that maximizes the distance between the hyperplane and the nearest sample of each class in the feature space [1]. There are mainly two types of SVM for classification in practice, namely L1-SVM and L2-SVM. Given l samples $x_i \in R^m$, $i=1,\ldots,l$ with labels $y_i \in \{\pm 1\}$, the SVM formulation with 1-norm of the slack variables, named as L1-SVM, is as follows:

$$\begin{aligned} &\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\|\xi\|_1 \\ &s.t \quad y_i(\mathbf{w}\Phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \quad \xi_i \geq 0 \ \ \forall i \end{aligned} \tag{1}$$

where ξ_i denotes the slack variables, $\xi=[\xi_1, \xi_2, \ldots, \xi_l]'$ and C adjusts the training errors and the term $1/2\|\mathbf{w}\|^2$. Its dual form is to solve the following quadratic optimization problem:

$$\begin{aligned} &\max \quad W(\boldsymbol{\alpha}) = \mathbf{e}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} \\ &s.t. \quad 0 \leq \alpha_i \leq C, i = 1,\ldots,l \\ &\qquad \mathbf{y}^T\boldsymbol{\alpha} = 0, \end{aligned} \tag{2}$$

where $\mathbf{e}$ is the vector of all ones, $\boldsymbol{\alpha}$ represents the Lagrange multipliers and $\mathbf{Q}$ is a $l \times l$ symmetric matrix with $\mathbf{Q}=y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. $K(\mathbf{x}_i, \mathbf{x}_j)$ denotes the inner product of $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ and Φ is a nonlinear function that maps the input vector to a higher feature space. In practice, $K(\mathbf{x}_i, \mathbf{x}_j)$ is calculated through a kernel function instead. Gaussian RBF kernel is one of the mostly used kernels and defined as follows:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{3}$$

where σ is the kernel width parameter.

In L2-SVM, it uses the 2-norm of the slack variables ξ_i in the objective function. Thus it formulates as follows:

$$\begin{aligned} &\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\|\xi\|_2^2 \\ &s.t. \quad y_i(\mathbf{w}\Phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \quad \xi_i \geq 0 \ \ \forall i \end{aligned} \tag{4}$$

By introducing the Lagrange multipliers $\boldsymbol{\alpha}$, one can obtain its dual form as follows:

$$\begin{aligned} &\max \quad \mathbf{e}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\left(\mathbf{Q} + \frac{\mathbf{I}}{C}\right)\boldsymbol{\alpha} \\ &s.t. \quad \alpha_i \geq 0, i = 1,\ldots,l \\ &\qquad \mathbf{y}^T\boldsymbol{\alpha} = 0. \end{aligned} \tag{5}$$

where $\mathbf{I}$ is an identity matrix. Given a new test sample $\mathbf{x}$ to be classified, for both types of SVM, its label can be predicted according to the following decision function:

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{6}$$

where b is a biased value.

2.2. Generalization error bounds of SVM

In the numerical model selection methods, the tuning of parameters is usually done by minimizing an estimate of a generalization error such as the leave-one-out (*LOO*) error or the k-fold cross-validation error. In general, *LOO* procedure gives an almost unbiased estimate of the expected generalization error of a learning method [20]. However, it suffers from the disadvantage of its high computational cost [20]. In *LOO* procedure, one instance is left out in turn for testing, and the training and testing will be repeated l times. Since a non-support vector can be correctly classified by the remaining training samples when it is omitted (i.e. non-support vector $\mathbf{x}_i$ does not change the decision function of [6] for its $\alpha_i=0$), a coarse estimate of the *LOO* generalization error rate can be approximated as follows:

$$LOO \ \ Err \leq \frac{1}{l} nSV \tag{7}$$

where nSV is the number of support vectors (SVs) and l is the number of training vectors [2]. This bound, although simple in computation, is too loose to the real *LOO* generalization errors and cannot be directly applied in SVM model selection. Some tighter bounds have been proposed by researchers in recent years. For the hard margin SVM with no bias b, Vapnik proposed that the *LOO* error is bounded by the following equation [1]:

$$LOO \ \ Err \leq \frac{D^2}{l\rho^2} = \frac{R^2\|\mathbf{w}\|^2}{l} \tag{8}$$

where $\rho = 2/\|\mathbf{w}\|$ is the margin between the two decision hyperplanes and $D=2R$ is the diameter of the smallest ball containing all training samples. The hard margin SVM is defined as follows:

$$\begin{aligned} &\min \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &s.t. \quad y_i(\mathbf{w}\Phi(\mathbf{x}_i) - b) \geq 1, \quad i = 1,\ldots,l \end{aligned} \tag{9}$$

where R^2 is the objective value of the following one-class SVM optimization problem:

$$\begin{aligned} &\max \quad \sum_i \beta_i K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i,j} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &s.t. \quad \beta_i \geq 0, i = 1,\ldots,l \\ &\qquad \mathbf{e}^T\boldsymbol{\beta} = 1 \end{aligned} \tag{10}$$

By using the span idea, Vapnik and Chapelle further extended the bound (8) to the general separable problems where the bias b is given [21]. For separable case (hard margin SVM without training errors) with bias, they proved that the *LOO* error rate on training data is bounded by:

$$LOO \ \ Err \leq \frac{1}{l}\frac{SD}{\rho} = \frac{SD\|\mathbf{w}\|^2}{4l} \tag{11}$$

where S is the span of all support vectors (see details about S in [21]). From the lemma 2 in [21], we have $S \leq D_{sv} \leq D$, where D_{sv} is the diameter of the smallest sphere containing the support vectors of the first category (those with $0 < \alpha_i < C$). Note that it is very difficult to calculate S. Then for hard margin SVM, we can

relax (11) and obtain a new *LOO* bound, shown as follows:

$$LOO\ \ Err \le \frac{D^2\|\mathbf{w}\|^2}{l} = \frac{R^2\|\mathbf{w}\|^2}{l} \tag{12}$$

Note that L2-SVM can be reduced to a hard margin classifier simply by replacing $\mathbf{w}$, the ith sample $\Phi(\mathbf{x}_i)$ and the kernel function $K(\mathbf{x}_i,\mathbf{x}_j)$ with $\tilde{\mathbf{w}} = \begin{pmatrix} \mathbf{w} \\ \xi\sqrt{C} \end{pmatrix}$, $\Phi(\mathbf{x}_i) = \begin{pmatrix} \Phi(\mathbf{x}_i) \\ \frac{1}{\sqrt{C}y_i}\mathbf{e}_i \end{pmatrix}$ and $\tilde{K}(\mathbf{x}_i,\mathbf{x}_j) = \left(K(\mathbf{x}_i,\mathbf{x}_j) + \frac{\delta_{ij}}{C}\right)$, where $\mathbf{e}_i$ is the m-dimensional vector with all zeros except that the ith component is equal to one, $\delta_{ij}=1$ if $i=j$ and $\delta_{ij}=0$ otherwise. Then the radius-margin *LOO* error bound of (12) for hard margin SVM can be extended to the non-separable L2-SVM. That is, for L2-SVM, the following bound holds [12]

$$LOO\ \ Err \le \frac{\tilde{R}^2\|\tilde{\mathbf{w}}\|^2}{l} \tag{13}$$

where $\tilde{R}^2$ can be obtained by solving an alternative optimization problem of (10) by replacing the term $K(\mathbf{x}_i,\mathbf{x}_j)$ with $\tilde{K}(\mathbf{x}_i,\mathbf{x}_j) = \left(K(\mathbf{x}_i,\mathbf{x}_j) + \frac{\delta_{ij}}{C}\right)$. Obviously, we can see that bound (8) and bound (13) are very similar to each other. Bound (13) has been proved to be very effective for choosing the parameters for L2-SVM [11,12] but not so good for L1-SVM [22]. For L1-SVM, recently several *LOO* bounds have been proposed. Based on the span idea, Chuang et al. proposed a differentiable *LOO* error bound [6], referred as the modified radius-margin bound:

$$LOO\ \ Err \le \frac{1}{l}\left[\left(R^2 + \frac{\Delta}{C}\right)(\|\mathbf{w}\|^2 + 2C\sum_{i=1}^{l}\xi_i)\right] \tag{14}$$

where Δ is a positive constant close to 1. The differentiability and the usefulness of this bound were discussed in [13]. This bound gives a better approximation to the *LOO* error rate of L1-SVM than other existing bounds but very sensitive to the initial point [13]. If a right starting point is given, from Chuang's study, bound (14) can be adopted to find near optimal parameters for L1-SVM. But if the starting point is inappropriate, although a local optimum to the bound can be identified, the generalization ability of SVM cannot be guaranteed [13]. Therefore, the selection of the starting point becomes very important. Unfortunately, it is still an open problem to find a tight differentiable *LOO* error bound for L1-SVM.

2.3. BFGS method for SVM parameter tuning

In SVM model selection, we need to find an optimum (global or local optimum) regarding the generalization bounds. BFGS method is the commonly used method [11–14,22] and the RBF kernel is the mostly used kernel for its many good properties and simplicity in computation. If RBF kernel is used, there are only two parameters, C and σ^2, need to be tuned. As suggested by Chapelle et al. [11], a variable transformation is often adopted in parameter tuning. That is, the optimization is done in a logarithmic space $(\ln C, \ln\sigma^2)$ rather than directly searching in the (C, σ^2) space, and $(0, 0)$ is often adopted as the starting point. The experiments of Keerthi et al. shows the usefulness of the *LOO* bound criteria and the associated implementations [13,14]. However, considering that there is no guarantee of the convexity of the *LOO* bounds, the BFGS based parameter tuning methods may collapse if an inappropriate starting point is chosen. Simply, this problem can be addressed by a grid search strategy. That is to say, we can choose a series of starting points in the parameter space and then implement BFGS method from each starting point. Obviously, the computation cost of this strategy will increase exponentially as the number of parameters increases.

3. CLPSO–BFGS for tuning SVM parameters

As discussed previously, one drawback of the gradient-based optimization methods is their sensitivity to the starting point when dealing with non-convex problems and thus only can ensure to converge to a local optimum. Considering that there is no guarantee of the convexity of the *LOO* bounds, if an inappropriate initial point is adopted, the gradient-based methods may converge to a local optimum, which cannot give a right approximation to the good parameters. In other words, the numerical methods may occupy high computation efficiency but lack of stabilities. Therefore, it is valuable to use some global optimization methods to find the global (or near global) optimum. In addition, if there are multiple local solutions to the *LOO* bounds, it is also valuable to identify these multiple optima to provide more choices for the final parameter selection. However, finding multiple optima of a function is much more challenging and is practically impossible by using deterministic optimization methods so far. Recently, the stochastic global optimization techniques, such as GA, PSO and SA have been adopted to tackle with the non-convex problems. Among them, the PSO is a relatively new technique and has gained rapid popularity in many communities.

PSO is first proposed by Kennedy et al by modeling the social behaviors of birds flocking [23,24]. As a population based method, PSO has considered a lot about the cooperation and competition among particles. but little about the particles' individual search ability. In the last several years, many improved variants have been proposed. Among these new variants, the comprehensive learning particle swarm optimizer (CLPSO) showed state-of-art global search ability on many complex problems [19]. Different from the standard PSO, the CLPSO uses all other particles' historical best information to update a particle's velocity and thus explores a larger search space than other PSO optimizers. The basic evolution mechanism of a single particle in CLPSO is described as follows:

$$V_{id} = w\,V_{id} + c_1\,r_1(p_{id}^{fi} - x_{id}) \tag{15}$$

$$x_{id} = x_{id} + V_{id} \tag{16}$$

where V_{id} and x_{id} represents the velocity and position of particle i in d dimension, respectively, w is the inertia weight, c_1 is the accelerate constant, r_1 is a random number in ranges of [0, 1], and fi defines which particles' *pbest* the particle i should follow. p_{id}^{fi} can be the corresponding dimension of any particle's *pbest* including its own in d dimension. Although CLPSO has greatly improved the global search ability, it still cannot satisfactorily solve the two bottlenecks simultaneously, namely the premature convergence and the slow convergence rate [25,26]. Another drawback of CLPSO as well as most PSOs is that it is difficult for them to find multiple optima of the multimodal problems, due to an intrinsic restriction that all particles must converge to only one point at the final step [27]. In our research, we address these problems by supposing that the global and local searches in PSO can be treated separately. The basic idea is, once a particle enters an optimality region, it will quickly reach the local optimum (candidate of the global optimum). In this sense, the only role of PSO is to guide the particles to find the optimality region while the local search is implemented through numerical optimization methods, such as BFGS method. Because the parameter space in SVM is usually a box-constrained space, a modified box-constrained BFGS method is dynamically interleaved into the main iterations of PSO algorithms (named as the context PSO) to implement the local search process. Since the local search of PSO is performed by BFGS method, the local convergence rate of the context PSO is no longer so important while better global search ability is preferred. Taking

all things into consideration, the CLPSO is adopted as the context PSO in our research for its good global search ability [19].

Note that the local search is only effective when at least one of the particles enters the optimality region that contains the global optimum. Hence in this paper, we define a local diversity index (LDI) to indicate whether or not the population enters into an optimality region in a high probability. Suppose x_0 is the particle with the best fitness value, x_{01} and x_{02} are the two nearest particles to x_0, LDI is measured by $L(S)$ as follows:

$$L(S)=\frac{\sum_{i=1}^{2}\|x_{oi}-x_o\|}{2\sqrt{\sum_{k=1}^{Dim}(ub^k-lb^k)^2}} \tag{17}$$

where S denotes the population, ub^k and lb^k, respectively, denote the upper and lower bounds, Dim is the dimensionality of the problem. Obviously, $L(S)\le 1$ holds. We suppose that the population is in an optimality region with high probability if $L(S)$ is smaller than a predefined value L_0 and the local search will be performed using the particle with the best fitness value as the initial solution.

The schematic of the proposed method is shown in Fig. 1. Here, a particle P_i is a row vector constructed by the parameter C and the kernel width parameter σ^2. The modified BFGS method is interleaved into the main iterations of CLPSO algorithm and will be started when $L<L_0$ is satisfied. The CLPSO iterations of the proposed method will be stopped if a maximum number *mFvals* of fitness evaluations are reached. Consider that the condition $L(S)<L_0$ may not satisfy in the main iterations of CLPSO, the local search will be performed after all PSO iterations if no local search has ever been performed. The whole optimization process can be also terminated after one local search is performed. In this case, only one local solution will be determined. Finally, once a particle is chosen to perform the local search, it will be randomly repulsed to an arbitrary location for a new global search. Therefore, the particles in the hybrid strategy will never gather to a single point and the premature convergence can be avoided.

For L2-SVM, bound (13) is tight enough to approximate the *LOO* errors and good enough for finding the optimal parameters. Therefore, we directly use this bound for the objective function of the CLPSO algorithm for L2-SVM. However, for L1-SVM, the looseness of the modified *LOO* error bound (14) may cause bias to the true generalization ability of L1-SVM. In order to find the right optimality region with high generalization ability, we use an alternative metric, named as the sum of the error rate (*SER*), as the fitness function to guide the particles to find the optimality region instead of directly using the *LOO* error bound. As (7) suggested, the number of support vectors can be a rough bound to the *LOO* errors. Then in the *SER* metric, we consider both values of the modified radius-margin bound (14) and the number of support vectors. Considering that the *LOO* error bound value may be larger than 1, we scale it by $\tan h$ function and the sum of the error rate can be simply defined as follows:

$$SER=\tan h(f)+\tan h\left(\frac{nSV}{l}\right) \tag{18}$$

where *nSV* represents the corresponding number of SVs and l stands for the number of training samples. *SER* makes a balance between the bound value and the number of SVs and can provide a better guidance to the particles to find the optimality region. An optimal SVM classifier will be trained using the obtained parameters and then adopted to predict the testing dataset. As mentioned previously, the hybrid method can find multiple local optima for the objective function. When each local search is completed, a local solution will be stored. Considering that the *LOO* errors are also bounded by the number of support vectors, in the BFGS course, the parameter vector with the minimum number of support vectors is also recorded.

4. Experimental results and discussions

4.1. Parameter settings

LIBSVM is to implement the SVMs [28]. Parameter settings of the hybrid method are listed in Table 1. In the experiments, the swarm size *ps* is set to 5. ε denotes the termination tolerances of BFGS method discussed in Section 3 and L_0 is the predefined value

Step 1: Initialize the population.

Stochastically assign $S=[P_1,...,P_{ps}]^{\mathrm{T}}$, velocities $V=[V_1,...,V_{ps}]^{\mathrm{T}}$, solution set $T=\phi$, and step index $k=0$.

Step 2: Evaluate the fitness of all particles.

If $L(S)$ is greater than the predefined value L_0, directly evaluate the fitness of all particles and go to step 4.

Step 3: Do local search.

Choose the particle with the best fitness value as the starting point and implement the local search using BFGS method. For those particles without local search, directly calculate their fitness values.

Step 4: Update the status of the particles.

Update the p_{best} and p_g. Update the particles' velocities and locations through the dynamic equations of CLPSO. Set $k=k+1$.

Step 5: Check termination conditions.

If the number of the fitness evaluations larger than $mFvals$, go to step 6. Or else go to step 2. Here, $mFvals$ is the predefined maximum number of fitness evaluations.

Step 6: Final parameter selection.

If no local search is performed in the iterations of CLPSO, do local search with p_g. Choose the best parameters for SVM and train an optimal classifier using the selected parameters.

Fig. 1. Schematic of the CLPSO–BFGS for SVM parameter tuning.

Table 1
Parameter settings for CLPSO–BFGS method.

ε	ps	L_o	$mFvals$
1.00e−4	5	0.05	150

of $L(S)$. The maximum number of fitness evaluations $mFvals$ is set to 150. Other parameters of CLPSO are kept default in [19]. Among the newly introduced parameters, L_0 plays an important role for the performance of the proposed method. We suppose that $L(S)$ should be smaller than 0.1, which can ensure an enough evolution of the population. On the other hand, if L_0 is too small and $L(S) < L_0$ cannot reach, the local search will be started after all the PSO iterations and only one local optimum can be identified in this case.

4.2. Derivative calculation of the LOO bounds

In BFGS method, the gradients of the *LOO* bounds should be calculated. Here, only the final results are presented. The details of derivative calculation of these two bounds can be seen in [12,13]. Denote bound (14) for L1-SVM and bound (13) for L2-SVM as f_1 and f_2, respectively, the derivatives can be calculated as follows:

$$\frac{\partial f_1}{\partial C} = \frac{1}{l}\left[2\left(R^2+\frac{\Delta}{C}\right)\sum \xi_i - 2obj\frac{\Delta}{C^2}\right] \tag{19}$$

$$\frac{\partial f_1}{\partial \sigma^2} = -\frac{1}{l}\left[\left(R^2+\frac{\Delta}{C}\right)\sum_{i,j}\alpha_i\alpha_j y_i y_j \frac{\partial K}{\partial \sigma^2} + 2obj\sum_{i,j}\beta_i\beta_j\frac{\partial K}{\partial \sigma^2}\right] \tag{20}$$

$$\frac{\partial f_2}{\partial C} = \frac{1}{l}\left[\frac{\sum_{i=1}^{l}\alpha_i^2}{C^2}R^2 - \|\tilde{\mathbf{w}}\|^2\frac{\sum_i \beta_i(1-\beta_i)}{C^2}\right] \tag{21}$$

$$\frac{\partial f_2}{\partial \sigma^2} = -\frac{1}{l}\left[R^2\sum_{i,j}\alpha_i\alpha_j y_i y_j\frac{\partial K}{\partial \sigma^2} + \|\tilde{\mathbf{w}}\|^2\sum_{i,j}\beta_i\beta_j\frac{\partial K}{\partial \sigma^2}\right] \tag{22}$$

$$\frac{\partial K}{\partial \sigma^2} = \frac{\partial K(x_i,x_j)}{\partial \sigma^2} = K(x_i,x_j)\frac{\|x_i-x_j\|^2}{2\sigma^4} \tag{23}$$

where *obj* in (19) and (20) denote the objective value of problem (1).

4.3. Benchmark datasets

Thirteen benchmark datasets were adopted to illustrate the performance of the proposed method [29]. These benchmark have been widely used to verify variant classifiers including RBF neural networks, Kernel Fisher Discriminant (KFD), and variants of Adaboost based on RBF neural networks [30,31]. General information of these datasets is listed in Table 2. Organization of the experiments is as follows: firstly, 4 of the 13 datasets, Splice, Image, Banana and Waveform, were chosen to show the performance of the proposed method as case study. Then, the performance of the proposed method was compared with GA, SA and CLPSO on the above four datasets. Finally, the optimal SVM classifiers obtained by the proposed method were also compared with other optimized classifiers reported in literature [30] on the 13 benchmark datasets.

Table 2
General information of the 13 datasets.

	# Training data	# Test data	# Attributes
Thyroid	140	75	5
Titanic	150	2051	3
Heart	170	100	13
Cancer	200	77	9
Banana	400	900	2
Ringnorm	400	7000	20
Twonorm	400	7000	20
Waveform	400	4600	21
Diabetis	468	300	8
Solar	666	400	9
German	700	300	20
Splice	1000	2175	60
Image	1300	1010	18

4.4. Experimental results

Combined with BFGS method, the hybrid method can find multiple local optima for the *LOO* bounds. In the first experiment, 4 benchmark datasets, Splice, Image, Banana and Waveform, were adopted to do case studies. Here, only the top five parameter pairs with the minimum test errors were recorded. The *LOO* error bound values f, the number of SVs and the *SER* values were also recorded. As suggested by Chapelle et al. [11], we also did the parameter settings in the logarithmic space ($\ln C$ and $\ln\sigma^2$). In our experiments, the search scope is set to $[-10,10]\times[-10,10]$. Table 3 and 4 respectively, shows the results on the four benchmark datasets using bound (14) for L1-SVM and bound (13) for L2-SVM. The last rows of each dataset in Table 3 and 4 are the results obtained by BFGS method from initial point (0, 0). We scaled the attribute values to $[-1, 1]$ for each dataset, as suggested by [10]. Here, for L1-SVM, only the top five parameter pairs with the least *SER* values were recorded while for L2-SVM, the five parameter pairs with the least *LOO* bound values were recorded.

From Table 3, the minimum error rate on Splice dataset is 0.0961 at point (6.157, 2.468) and the parameter pair with the least *SER* value also produces competitive results. Note that the error rate obtained by BFGS method from point (0, 0) is 0.3747, which implies that BFGS method may fail to find the optimal (or near optimal) parameters for L1-SVM from an inappropriate initial point. For Image dataset, the minimum error rate is 0.0218 with the least *SER* value. As to Banana dataset, the best result is obtained with least bound value and the parameter pair with the least *SER* value can also produce competitive results. In contrast, BFGS method from (0, 0) fails to find the near optimal parameters with error rate 0.5594 on the testing dataset. Without data scaling, as reported in [13], bound (14) shows competitive results on the above four datasets from initial solution (0, 0) compared with L2-SVM. However, if we scale the data to $[-1, 1]$, a common data preprocessing step in data classification, bound (14) does not show the same stable performance from (0, 0). Therefore, the gradient method from a single starting point may be feasible and efficient for some problems but not stable, making it unsafe to use.

Table 4 presents the results obtained by CLPSO–BFGS using bound (13) for L2-SVM. On all four datasets, the parameters with the least *LOO* bound values can provide competitive test error rates on the test datasets. Generally speaking, bound (13) for L2-SVM is much tighter than bound (14) for L1-SVM. From Table 4, there are also multiple optima for bound (13). Taking Splice dataset for example, there are three different parameter pairs with the same *LOO* bound values. However, we prefer to choose the parameter pair with the least number of SVs as the final

Table 3
Results of L1-SVM (%).

Datasets	$\ln C$	$\ln \sigma^2$	f	nSV	SER	Err
Splice						
CLPSO–BFGS	6.162	2.233	**0.6026**	860	1.2352	0.1021
	6.157	2.468	0.6196	783	1.2053	**0.0961**
	5.882	2.216	0.6035	866	1.2389	0.1011
	5.874	2.597	0.6479	**740**	**1.1994**	0.0966
	–	–	–	–	–	–
BFGS	10.000	0.824	0.9501	979	1.4925	0.3747
Image						
CLPSO–BFGS	9.313	−2.957	**0.3950**	779	0.9122	0.0337
	0.355	−0.854	0.6474	**348**	0.8314	0.0337
	8.530	−2.954	**0.3950**	782	0.9138	0.0337
	0.666	−1.702	0.5314	415	**0.7953**	**0.0218**
	9.236	−2.939	**0.3950**	784	0.9149	0.0337
BFGS	6.489	−2.963	0.3955	781	0.9137	0.0337
Banana						
CLPSO–BFGS	−0.644	−3.906	0.8479	166	1.0827	0.1133
	−0.433	−3.419	0.8763	148	**1.0586**	0.1108
	−0.818	−3.851	**0.8447**	167	1.0831	0.1118
	−0.605	−3.587	0.8543	158	1.0690	**0.1100**
	−0.296	−3.437	0.8925	**145**	1.0600	0.1114
BFGS	−4.968	−0.043	1.8320	366	1.6735	0.5594
Waveform						
CLPSO–BFGS	6.337	−0.587	0.5045	183	0.8937	**0.1085**
	6.342	−1.006	0.4439	261	0.9902	0.1126
	7.727	−0.551	0.5151	**175**	**0.8855**	0.1087
	7.728	−1.000	**0.4433**	266	0.9981	0.1120
	–	–	–	–	–	–
BFGS	4.708	−1.228	0.4657	311	1.0860	0.1148

Table 4
Results of L2-SVM on four datasets (%).

Datasets	$\ln C$	$\ln \sigma^2$	f	nSV	SER	Error for L2-SVM
Splice						
CLPSO–BFGS	5.634	2.248	**0.6011**	871	1.2397	0.1016
	0.802	2.584	0.6117	**854**	**1.2314**	**0.0993**
	4.993	2.269	**0.6011**	863	1.2356	0.1011
	4.993	2.253	**0.6011**	869	1.2387	0.1016
	3.724	2.283	0.6013	862	1.2353	0.1007
BFGS	5.717	3.070	0.6011	868	1.2382	0.1016
Image						
CLPSO–BFGS	0.152	−1.753	0.3136	858	0.8821	0.0238
	0.097	−1.681	**0.3135**	853	0.8794	0.0238
	0.394	−1.866	0.3147	829	0.8681	**0.0208**
	0.272	−1.686	0.3142	819	0.8623	0.0228
	0.286	−1.677	0.3143	**813**	**0.8592**	0.0218
BFGS	0.155	−1.731	0.3135	850	0.8779	0.0238
Banana						
CLPSO–BFGS	−0.917	−3.668	0.4546	325	1.0966	**0.1137**
	−0.841	−3.778	0.4546	322	1.0925	0.1141
	−0.863	−3.759	**0.4545**	323	1.0938	0.1145
	−1.106	−3.669	0.4567	338	1.1159	0.1141
	−0.850	−3.685	0.4547	**321**	**1.0912**	**0.1137**
BFGS	−1.042	−3.763	0.4556	335	1.1110	0.1149
Waveform						
CLPSO–BFGS	−0.235	−0.139	0.3959	231	**0.8973**	0.1007
	−0.461	−0.048	**0.3946**	234	0.9016	**0.0983**
	−0.689	0.209	0.3966	229	0.9691	0.0985
	−0.553	−0.032	0.3948	238	0.9898	**0.0983**
	−0.467	−0.141	0.3951	241	0.9146	0.0993
BFGS	−0.532	2.404	0.3995	228	0.8949	0.1015

optimal parameters. From Table 4, we can see that with *SER* criterion, except for Image dataset, the CLPSO–BFGS can produce better results than BFGS method.

To show the complexity of the proposed method, we make a comparison between BFGS and CLPSO–BFGS on L2-SVM by counting the number of their fitness function evaluations, as shown in Table 5. In general, CLPSO–BFGS needs more computations than BFGS method (two or more times than BFGS method). However, if a wrong starting point is chosen, BFGS will also cost a lot of calculations to achieve to the local optimum. For example, as to Splice dataset, BFGS needs 139 fitness evaluations to find the local optimum from initial solution (0, 0), while CLPSO–BFGS can search for several local solutions within 150 fitness evaluations, namely 300 SVM trainings including the training of one-class SVM. Hence, CLPSO–BFGS holds a relatively high efficiency per each candidate optimal parameter in computation. The proposed method also needs fewer computations than grid search method. Specifically, if 5-cross-validation is adopted, then only 60 candidate parameter pairs can be tested using grid search method within 300 SVM trainings. Apparently, 60 candidate parameter pairs are usually not enough for real applications. For example, if we uniformly sample 20×20 points from the search space $[-10,10] \times [-10,10]$, it should need 2000 SVM trainings in total. What's more, the number of SVM trainings will increase exponentially as the number of parameters increases. However, the proposed method can be very effective when dealing with multiple parameters because of the introduction of local search. In addition, if only one parameter pair is needed, the optimization process can be terminated once a local search is finished, which can save a lot of computations but less stable. In conclusion, on the time complexities, the proposed method is a little higher than the pure gradient method but much less than the grid search method.

Table 5
Number of function evaluations on four datasets (only for L2-SVM).

Method	Splice	Image	Banana	Waveform
BFGS	139	25	31	29
CLPSO–BFGS	150	150	150	150

As mentioned in the introduction section, the stochastic optimization techniques can be directly adopted to optimize the *LOO* bounds. In the following experiment, the results obtained by CLPSO–BFGS were compared with those obtained by GA, PSO and SA methods. Here, only L2-SVM was studied. All experiments were run 30 times. The mean values and standard deviation of the error rates on different datasets are presented. In this paper, CLPSO was used to implement the PSO algorithm [32]. The adaptive simulated annealing (ASA) algorithm was adopted to implement the SA algorithm [33], and the GA toolbox developed Houck et al. was employed to implement the GA algorithm [34]. For convenience of comparison, all the algorithms mentioned above were completed when a total of 150 fitness evaluations were reached. The number of particles in CLPSO and the population size of GA are both set to 5. Other parameters of the aforementioned algorithms were kept default. The results were reported in Table 6. We can conclude from this table that CLPSO–BFGS method can achieve the best performance among the listed algorithms. The underlying reason is that, by adopting the BFGS method as the local search, the CLPSO–BFGS can achieve more accurate results than the traditional stochastic methods. What's more, the CLPSO–BFGS can find multiple local optima to the bounds, which can provide better results than a single local optimum.

Table 6
Error rates for L2-SVM (%).

	CLPSO	GA	ASA	CLPSO–BFGS
Splice	10.16 ± 0.00	10.16 ± 0.00	10.16 ± 0.00	$\mathbf{9.98 \pm 0.14}$
Image	5.44 ± 6.19	9.41 ± 0.00	3.34 ± 0.00	$\mathbf{2.41 \pm 0.35}$
Banana	17.87 ± 2.12	16.12 ± 0.00	11.61 ± 0.61	$\mathbf{11.21 \pm 0.25}$
Waveform	11.16 ± 0.00	11.46 ± 0.00	11.19 ± 0.00	$\mathbf{9.84 \pm 0.06}$

To further verify the proposed method, in the second experiment, the results obtained by the proposed method were compared with those published in literature [31] on 13 benchmark datasets that have been preprocessed by Rätsch. In Rätsch's preprocessing, the benchmark datasets were regenerated into 100 partitions (20 for image and splice datasets) with testing and training set (about 60:40%) [3,29]. An optimal classifier was obtained using the proposed method and tested on each of the datasets. The average testing error rates and the standard deviations were recorded, as shown in Table 6. In order to compare the algorithms, the best errors for every classification problem were bolded. The parameter v in v-SVM is estimated by Bayes risk p and the kernel parameter is estimated by running n-fold cross-validation [30].

From Table 7, we can see that, the CLPSO–BFGS shows improved performance compared with single BFGS method on two types of SVM. At first, as also mentioned in Table 3, BFGS method using bound (14) on Banana and Splice dataset may fail to find the optimal parameters for L1-SVM from starting point (0, 0). Maybe we can improve the performance of the BFGS by providing a suitable starting point, about which, however, we usually have little knowledge in advance, making it hard to implement. Secondly, generally speaking, bound (13) for L2-SVM shows better performance than bound (14) for L1-SVM, as shown in Table 7, which lies in that bound (13) is much tighter than bound (14) on approximating the *LOO* errors. Although BFGS method with bound (13) for L2-SVM shows good enough results on most of the benchmark dataset, we still argue its stability. For example, for Banana dataset, the BFGS method obtains an error rate of 41.16 ± 11.28 (%) from the starting point (0, 0). It indicates that bound (13) may be also not stable if an inappropriate initial solution is provided. By contrast, CLPSO–BFGS can achieve an error rate of 10.44 ± 0.46 (%), the best results among the listed optimized classifiers. Again, L2-SVM with parameters selected by CLPSO–BFGS also produces the best results on Thyroid and Waveform datasets. Finally, one can conclude from Table 7 that there is no classifier that can outperforms others on every dataset while the proposed method shows competitive performance compared with other optimized classifiers.

5. Conclusions

The task of tuning SVM parameters is important for the SVM applications and can be fulfilled by minimizing a generalization error bound by using the gradient optimization methods, which are known for their fast local convergence speed but very sensitive to the initial point. In view of the non-convexity of the existing generalization bounds, a hybrid CLPSO–BFGS algorithm is proposed to perform SVM model selection based on two existing generalization error bounds. The CLPSO–BFGS maintains both the global search ability of CLPSO algorithm and the fast local convergence rate. The proposed strategy can also find multiple solutions to the non-convex *LOO* error bound, which provides more choices for the final parameter selection and makes the model selection more stable and reliable. The experimental

Table 7
Test errors of the classifiers (%).

	RBF-Net	Ada-Boost	LP-AdaB	QP-AdaB	AdaB-Reg	SVM Cross-val.	KFD	v-SVM	L1-SVMBFGS	L1-SVMCLPSO-BFGS	L2-SVMBFGS	L2-SVMCLPSO-BFGS
Thyroid	4.52 ± 2.12	4.40 ± 2.18	4.59 ± 2.22	4.35 ± 2.18	4.55 ± 2.19	4.80 ± 2.19	$\mathbf{4.20 \pm 2.07}$	4.71 ± 1.88	4.79 ± 2.33	4.55 ± 2.10	4.61 ± 2.29	$\mathbf{4.20 \pm 2.03}$
Titanic	23.26 ± 1.34	22.58 ± 1.18	23.98 ± 4.38	22.71 ± 1.05	22.64 ± 1.20	$\mathbf{22.42 \pm 1.02}$	23.25 ± 2.05	24.62 ± 6.83	23.65 ± 3.03	23.51 ± 2.92	23.82 ± 4.03	22.89 ± 1.15
Heart	17.55 ± 3.25	20.29 ± 3.44	17.49 ± 3.53	17.17 ± 3.44	16.47 ± 3.51	15.95 ± 3.26	16.14 ± 3.39	15.87 ± 3.13	20.36 ± 6.42	16.94 ± 3.71	16.49 ± 4.0	16.02 ± 3.26
Cancer	27.64 ± 4.71	30.36 ± 4.73	26.79 ± 6.08	25.91 ± 4.61	26.51 ± 4.47	26.04 ± 4.74	$\mathbf{24.77 \pm 4.63}$	26.75 ± 4.03	28.81 ± 4.56	28.75 ± 4.61	27.60 ± 4.71	26.03 ± 4.60
Banana	10.76 ± 0.42	12.26 ± 0.67	10.73 ± 0.43	10.90 ± 0.46	10.85 ± 0.42	11.53 ± 0.66	10.75 ± 0.45	10.48 ± 0.47	44.91 ± 4.06	11.65 ± 5.90	41.16 ± 11.28	$\mathbf{10.44 \pm 0.46}$
Ringnorm	1.70 ± 0.21	1.93 ± 0.24	2.24 ± 0.46	1.86 ± 0.22	1.58 ± 0.12	1.66 ± 0.12	$\mathbf{1.49 \pm 0.12}$	1.68 ± 0.12	1.69 ± 0.13	1.68 ± 0.13	1.69 ± 0.22	1.68 ± 0.12
Twonorm	2.85 ± 0.28	3.03 ± 0.28	3.17 ± 0.43	2.97 ± 0.26	2.70 ± 0.24	2.96 ± 0.23	$\mathbf{2.61 \pm 0.15}$	2.74 ± 0.21	3.17 ± 0.33	2.90 ± 0.27	2.72 ± 0.35	2.64 ± 0.20
Waveform	10.66 ± 1.08	10.84 ± 0.58	10.53 ± 1.02	10.07 ± 0.51	9.79 ± 0.81	9.88 ± 0.43	9.86 ± 0.44	10.26 ± 0.40	10.70 ± 0.74	9.78 ± 0.48	9.76 ± 0.47	$\mathbf{9.58 \pm 0.37}$
Diabetis	24.29 ± 1.88	26.47 ± 2.29	24.11 ± 1.90	25.39 ± 2.20	23.79 ± 1.80	25.53 ± 1.73	23.21 ± 1.63	$\mathbf{23.21 \pm 1.57}$	25.69 ± 4.10	25.36 ± 2.95	25.43 ± 3.57	23.50 ± 1.66
Solar	34.37 ± 1.95	35.7 ± 1.79	34.74 ± 2.00	36.22 ± 1.80	34.2 ± 2.18	32.43 ± 1.82	33.16 ± 1.72	$\mathbf{32.33 \pm 1.81}$	33.41 ± 1.94	33.15 ± 1.83	32.75 ± 3.76	33.33 ± 1.79
German	24.71 ± 2.38	27.45 ± 2.50	24.79 ± 2.22	25.25 ± 2.14	24.34 ± 2.08	$\mathbf{23.61 \pm 2.07}$	23.71 ± 2.20	23.64 ± 2.19	29.64 ± 2.04	29.26 ± 2.88	29.27 ± 3.76	24.41 ± 2.13
Splice	9.95 ± 0.78	10.14 ± 0.51	10.22 ± 1.59	10.11 ± 0.52	$\mathbf{9.50 \pm 0.65}$	10.88 ± 0.66	10.52 ± 0.64	11.12 ± 0.72	33.70 ± 10.16	10.99 ± 0.74	12.46 ± 6.77	10.84 ± 0.74
Image	3.32 ± 0.65	2.73 ± 0.66	2.76 ± 0.61	$\mathbf{2.67 \pm 0.61}$	$\mathbf{2.67 \pm 0.61}$	2.96 ± 0.60	4.76 ± 0.58	3.13 ± 0.63	3.87 ± 0.64	3.34 ± 0.71	3.33 ± 0.62	2.97 ± 0.45

results on benchmark datasets demonstrate that the proposed strategy can obtain competitive results compared with other optimized classifiers, and more stable performance than the gradient methods and less computationally expensive than the grid search method.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (No. 60871096 and 60835004), the Ph.D. Programs Foundation of Ministry of Education of China (No.200805320006), the Open Projects Program of National Laboratory of Pattern Recognition, China and the Key Project of Chinese Ministry of Education (2009–120).

References

[1] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[2] N. Cristianini, J. Shawe-Taylor, An Introduction to SVM, Cambridge University Press, Cambridge, 2000.
[3] K.R. Müller, et al., An introduction to kernel based method, IEEE Trans. Neural Networks 12 (2) (March 2001) 181–201.
[4] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (January 2007) 40–51.
[5] D. Xu, S. Chang, Video Event recognition using kernel methods with Multi-Level temporal alignment, IEEE Trans. Pattern Anal. Mach. Intell. 30 (11) (November 2008) 1985–1997.
[6] D. Xu, S. Lin, S. Yan, X. Tang, Rank-one projections with adaptive margin for face recognition, IEEE Trans. Syst. Man Cybern. Part B 37 (5) (October 2007) 1226–1236.
[7] D. Xu, S. Yan, D. Tao, S. Lin, H. Zhang, Marginal fisher analysis and its variants for human gait recognition and content based image retrieval, IEEE Trans. Image Process. 16 (11) (November 2007) 2811–2821.
[8] S. Yan, Y. Hu, D. Xu, B. Zhang, H. Zhang, Q. Cheng, Nonlinear discriminant analysis on embedded manifold, IEEE Trans. Circuits Syst. Video Technol. 17 (4) (April 2007) 468–477.
[9] H. Frohlich, A. Zell, Efficient parameter selection for support vector machines in classification and regression via model-based global optimization, In Proc. Int. Joint Conf. Neural Networks (2005) 1431–1436.
[10] C. W. Hsu, C. C. Chang, C. J. Lin, A practical guide to support vector classification, Available at 〈http://www.csie.ntu.edu.tw/~cjlin/libsvm/〉.
[11] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (1) (January. 2002) 131–159.
[12] S.S. Keerthi, Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms, IEEE Trans. Neural Networks 13 (5) (September. 2002) 1225–1229.
[13] K.M. Chung, W.C. Kao, C.L. Sun, L.L. Wang, C.J. Lin, Radius margin bounds for support vector machines with the RBF kernel, Neural Comput. 15 (11) (November. 2003) 2643–2681.
[14] S.S. Keerthi, V. Sindhwani, O. Chapelle, An efficient method for gradient-based adaptation of hyperparameters in svm models, NIPS 19 (2007) 673–680.
[15] A.C. Lorena, A.C.P.L.F. de Carvalho, Evolutionary tuning of SVM parameter values in multiclass problems, Neurocomputing 71 (2008) 3326–3334.
[16] H.J. Escalante, M. Montes, L.E. Sucar, Particle Swarm Model Selection, J. Mach. Learn. Res. 10 (2009) 405–440.
[17] F. Melgani, Y. Bazi, Classification of electrocardiogram signals with support vector machines and particle swarm optimization, IEEE Trans. Inf. Tech. Biomed. 12 (5) (2008) 667.
[18] S. Lin, Z. Lee, C. Chen, T. Tseng, Parameter determination of support vector machine and feature selection using simulated annealing approach, Appl. Soft Comput. 8 (4) (Sep. 2008) 1505–1512.
[19] J.J. Liang, A.K. Qin, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (June. 2006) 281–295.
[20] M.S.M. Lee, S.S. Keerthi, C.J. Ong, D. DeCoste, An efficient method for computing Leave-One-Out error in support vector machines with Gaussian kernels, IEEE Trans. Neural Networks 15 (May 2004) 750–757.
[21] V. Vapnik, O. Chapelle, Bounds on error expectation for support vector machine, in: A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, 1999.
[22] K. Duan, S.S. Keerthi, A.N. Poo, Evaluation of simple performance measures for tuning SVM hyperparameters, Neurocomputing 51 (2003) 41–59.
[23] R.C. Eberhart, J. Kenned, A new optimizer using particle swarm theory, In: Proceedings of the sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, October 04–06, 1995, pp. 39–43.
[24] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proc. IEEE Int. Conf. Neural Networks 1 (1995) 1942–1948.
[25] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (April 1997) 67–82.
[26] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler maybe better, IEEE Trans. Evol. Comput. 8 (3) (June 2004) 204–210.
[27] J. Seo, C. Im, C. Heo, J. Kim, H. Jung, C. Lee, Multimodal function optimization based on particle swarm optimization, IEEE Trans. Magn. 42 (4) (April 2006) 1095–1098.
[28] C. C. Chang, C. J. Lin, LIBSVM: a library for support vector machines, 2001. Software available at 〈http://www.csie.ntu.edu.tw/~cjlin/libsvm〉.
[29] 〈http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm〉.
[30] I. Steinwart, On the optimal parameter choice for v-support vector machines, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1274–1284.
[31] G. Rätsch, T. Onoda, K.R. Müller, Soft margins for AdaBoost, Mach. Learn. 42 (3) (2001) 287–320.
[32] 〈http://www.ntu.edu.sg/home/epnsugan〉.
[33] 〈http://alumnus.caltech.edu/~ingber/#ASA〉.
[34] 〈http://www.ise.ncsu.edu/mirage/GAToolBox/gaot/〉.

Shutao Li received his B.S., M.S., and Ph.D. degrees in electrical engineering from the Hunan University, in 1995, 1997, and 2001, respectively. He joined the College of Electrical and Information Engineering, Hunan University, in 2001. He was Research Associate in the Department of Computer Science, Hong Kong University of Science and Technology, from May 2001 to October 2001. From November 2002 to November 2003, he was a postdoctoral fellow at the Royal Holloway College, University of London, working with Prof. John Shawe-Taylor. During April 2005 to June 2005, he worked in the Department of Computer Science, Hong Kong University of Science and Technology as a visiting professor. Now, he is a full professor with the College of Electrical and Information Engineering, Hunan University. He has authored or coauthored more than 100 refereed papers. He has won two 2nd-Grade National Awards at Science and Technology Progress of China in 2004 and 2006. His professional interests are computational intelligence, information fusion, pattern recognition, and image processing. Dr. Li was a member of the IEEE, where he served as a member in Neural Networks Technical Committee from 2007–2008.

Mingkui Tan received the B.S and M.S degree in environmental science and electrical engineering from Hunan University, in 2006 and 2009, respectively. He is currently a Ph.D. student in school of computer engineering at Nanyang Technological University. His technical interests include kernel methods and machine learning.