Applied Mathematics and Computation 239 (2014) 180–197

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

Towards an optimal support vector machine classifier using a parallel particle swarm optimization strategy

Hui ling Chen [a,c], Bo Yang [b,c], Su jing Wang [d], Gang Wang [b], Da you Liu [b,c,*], Huai zhong Li [a], Wen bin Liu [a]

[a] College of Physics and Electronic Information, Wenzhou University, Wenzhou, Zhejiang 325035, China
[b] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[c] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[d] State Key Laboratory of Brain and Cognitive Science, Institute of Psychology, Chinese Academy of Sciences, Beijing 100101, China

ARTICLE INFO

Keywords:
Support vector machines
Particle swarm optimization
Parallel computing
Feature selection

ABSTRACT

Proper parameter settings of support vector machine (SVM) and feature selection are of great importance to its efficiency and accuracy. In this paper, we propose a parallel time variant particle swarm optimization (TVPSO) algorithm to simultaneously perform the parameter optimization and feature selection for SVM, termed PTVPSO-SVM. It is implemented in a parallel environment using Parallel Virtual Machine (PVM). In the proposed method, a weighted function is adopted to design the objective function of PSO, which takes into account the average classification accuracy rates (ACC) of SVM, the number of support vectors (SVs) and the selected features simultaneously. Furthermore, mutation operators are introduced to overcome the problem of the premature convergence of PSO algorithm. In addition, an improved binary PSO algorithm is employed to enhance the performance of PSO algorithm in feature selection task. The performance of the proposed method is compared with that of other methods on a comprehensive set of 30 benchmark data sets. The empirical results demonstrate that the proposed method cannot only obtain much more appropriate model parameters, discriminative feature subset as well as smaller sets of SVs but also significantly reduce the computational time, giving high predictive accuracy.

1. Introduction

As one of primary machine learning techniques, SVM is based on the Vapnik–Chervonenkis theory and structural risk minimization principle [1–3]. It tries to find the tradeoff between minimizing the training set error and maximizing the margin, in order to achieve the best generalization ability and remain resistant to over fitting. Additionally, one major advantage of the SVM is the use of convex quadratic programming, which is able to provide global rather than local minima. Recently, many applications of the SVM have been found in a wide variety of fields, including handwritten digit recognition [4], face detection in images [5], text categorization [6], medical diagnosis [7] and so forth. Parameter setting plays a significant role in designing an effective SVM model. Since the linear kernel is a special case of RBF kernel, our investigation will primarily focus on Gaussian kernel to find out the optimal parameter values of RBF kernel function (i.e., C and γ). As shown by Keerthi

* Corresponding author at: College of Computer Science and Technology, Jilin University, Changchun 130012, China.
E-mail addresses: liudayou19420601@gmail.com, liudy@jlu.edu.cn (D.y. Liu).

http://dx.doi.org/10.1016/j.amc.2014.04.039
0096-3003/© 2014 Elsevier Inc. All rights reserved.

and Lin [8], the linear kernel with a penalty parameter C has the same performance as the RBF kernel with some parameters (C and γ). Other kernel parameters can also be tackled in the same way by using our developed method. The first parameter, penalty parameter C, determines the trade-off between the fitting error minimization and model complexity. The second parameter, gamma (γ) of the kernel function, defines the non-linear mapping from the input space to some high-dimensional feature space. In addition to the kernel parameter setting, choosing the optimal input feature subset will greatly influence the performance of the SVM as well. Excessively large feature vectors will significantly slow down the learning process as well as cause the SVM to overfit the training data. Feature selection addresses aforementioned problems by removing irrelevant, redundant and correlated features, and thus improving the accuracy of the classification model as well as decreasing the computational cost [9]. Both of parameter setting and feature selection are crucial because the feature selection influences the appropriate kernel parameters and vice versa [10], which suggests that they should be dealt with simultaneously.

The grid search method [11,12] and the gradient descent method [13–15] are two of the most common optimization methods for handling the parameter optimization for SVM. Grid search can lead to the highest classification accuracy in an interval through setting appropriate values for the upper bound, lower bound, and step of searching. However, this approach is a local search based that is vulnerable to local optimum. Additionally, to set an appropriate searching step is not an easy job. Apart from the grid search method, the gradient descent method is also often used to tackle the parameter optimization for SVM. Nevertheless, one disadvantage of gradient descent algorithm is that this algorithm is sensitive to initial parameters. If initial parameters are far from the optimal solution, it will be easily converged to a local optimum. Recently, metaheuristic-based search algorithms have been widely used to address optimization problems. These algorithms, such as the genetic algorithm (GA), artificial immune system (AIS), ant colony optimization (ACO), simulated annealing (SA), particle swarm optimization (PSO), are considered to have a better chance of finding global optimum solution than traditional methods.

A great deal of work on the model selection problem of SVM by employing metaheuristics has been done. In greater detail, in [16] an ACO based approach was proposed to simultaneously optimize the feature subset and the SVM kernel parameters. Its experimental results indicate that hybridized approach can correctly select the discriminating input features and achieve high classification accuracy. In [17], a multi-objective AIS based method was developed for parameter optimization in support vector machine. The fitness function of the method is based on the number of support vectors (SVs) and the classification accuracy (ACC) of SVM. However, it only deals with the parameter optimization, does not tackle the feature selection problem. In [18], GA was proposed to simultaneously optimize the feature subset and the SVM kernel parameters; the objective function is based on the number of features and the ACC of SVM. In [19], a PSO based approach was proposed for selecting features and setting kernel parameters. The objective function of the method depends on the number of features and the ACC of SVM. In [20,21], a PSO based method and a SA based method for parameter determination and feature selection were developed, respectively. While the objective functions of the two methods only depend on the ACC of SVM. In [22], a PSO based parameter optimization approach for least-squares support vector machine (LS-SVM) was proposed, and the objective function is only based on the ACC of LS-SVM. One disadvantage of the aforementioned methods is that their objectives do not consider the number of SVs. It will not only slow down the performance of testing stage, but also result in some irrelevant data being given as SVs.

In this paper, a novel method, namely PTVPSO-SVM, is proposed to overcome above-discussed shortages. In the proposed method, we take into account the ACC of SVM, the number of SVs and the number of features simultaneously in designing the objective function to exploit the maximum generalization capability of SVM. The three sub-objectives are integrated into one single objective function by linearly weighting. In order to further balance the local and global search in PSO, the adaptive control parameters (including TVAC and TVIW) are introduced which help the algorithm explore the search space more efficiently. Moreover, the mutation operation is adopted to avoid the premature convergence of the PSO algorithm. Most important of all, we have implemented our method in a parallel computing environment, which is able to significantly reduce computational time. To our knowledge, the work using a high performance technique to tackle the parameter optimization and feature selection for SVM was ever done by Huang and Dun [19]. They implemented a hybrid method in a heterogeneous computing environment using the web service technology. As we know, the load balancing of web service is not easy to design and cannot achieve excellent performance without the help of senior experts. Furthermore, they did not mention how to handle the concurrent requests of command, transmission and calculation from the clients to the server, and how to dispatch the tasks from the server to the clients according to the computational capability and load balancing of each client. In this paper we aim at implementing an easy and efficient scheme to reduce the computational time by using PVM[1] that makes easy load balancing and distribute computing. The feasibility and effectiveness of the proposed method are examined by comparing with the standard PSO-based and the grid search-based methods on a comprehensive set of 30 benchmark data sets from the UCI Machine Learning and StatLog databases. The numerical results and statistical analysis confirm the significant advantages of the proposed method over others. On one hand, the proposed method can not only obtain the appropriate parameter settings but also gain a subset of discriminative features, giving high generalization capability. On the other hand, it gives the minimum number of SVs and significantly reduces the computational time as well.

[1] http://www.csm.ornl.gov/pvm/.

The main contributions of this article are summarized as follows:

(a) A weighted function that simultaneously takes into consideration the ACC of SVM, the number of SVs and the selected features is developed to design the objective function. It enables the SVM classifier to give the minimum number of SVs and the most appropriate parameter setting, as well as the best discriminative subset of features, hence giving high generalization capability.
(b) The adaptive control parameters (including TVAC and TVIW) are introduced to further balance the local and global search in the PSO algorithm, and the mutation operation is employed to avoid the premature convergence of PSO algorithm.
(c) An efficient parallel implementation is presented to enhance the efficiency of the method with respect to computational time.

The remainder of this paper is organized as follows. Section 2 briefly describes SVM classification problems and PSO algorithms. Section 3 presents the details of implementations of the proposed PTVPSO-SVM method. The experimental results and discussion are presented in Section 4. Finally, Conclusions and future work are summarized in Section 5.

2. Theoretical backgrounds of related methodologies

2.1. SVM for classification

This section gives a brief description on SVM. For more details, one can refer to [3,4,23,24], which give a complete description of the SVM theory. Let us consider a binary classification task: $\{xi, yi\},\ i = 1,\ldots l,\ yi \in \{-1,1\},\ xi \in R^d$, where xi are data points and yi are corresponding labels. They are separated with a hyperplane given by $w^Tx + b = 0$, where w is a d-dimensional coefficient vector which is normal to the hyperplane and b is the offset from the origin. The linear SVM finds an optimal separating margin by solving the following optimization task:

$$\text{Minimize } g(w,\xi) = \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{n}\xi_i, \tag{1}$$

$$\text{Subject to}: y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i,\ \ \xi_i \geqslant 0, \tag{2}$$

where C is a penalty value, ξ_i is the positive slack variables. This primal problem can be reduced to the Lagrangian dual problem by introducing Lagrangian multipliers α_i. According to the Karush Kuhn–Tucker (KKT) condition, we can get the optimal solution α_i. If $\alpha_i > 0$, the corresponding data points are called SVs. Afterwards, we can get the optimal hyperplane parameters w and b. Then the linear discriminant function can be given by

$$g(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b\right). \tag{3}$$

In order to make the linear learning machine work well in non-linear cases, the original input space can be mapped into some higher-dimensional feature space via a mapping function ϕ. With this mapping, $\mathbf{x}_i^T\mathbf{x}$ in the input space can be represented as the form of $\phi(\mathbf{x}_i)^T\phi(\mathbf{x})$ in the feature space. The functional form of the mapping $\phi(\mathbf{x}_i)$ does not need to be known since it is implicitly defined by one selected kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Two most widely used kernels in SVM are the polynomial kernel and the Gaussian kernel (or Radial-Basis function, RBF), which are respectively defined as:

$$K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^p, \tag{4}$$

$$K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2), \tag{5}$$

where p is the polynomial order, and γ is the predefined parameter controlling the width of the Gaussian kernel. This investigation is going to consider the Gaussian kernel to find out the optimal parameter values of RBF kernel function (i.e., C and γ). Other kernel parameters can also be tackled in the same way by using our proposed method.

By introducing the kernel function, the decision function can be expressed as follows:

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

For the multi-class classifier, many strategies have been proposed [25]. The most popular ones are one-against-all and one-against-one strategies. In general, both strategies can give similar results in terms of classification accuracy. In this paper, we will consider the one-against-one strategy [12]. Briefly, classification of new instances for one-against-one case is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with most votes determines the instance classification.

2.2. Particle swarm optimization (PSO)

PSO is inspired by the social behavior of organisms such as bird flocking and fish schooling, which was first developed by Kennedy and Eberhart [26,27]. The algorithm seeks to explore the search space by a population of individuals or particles. Each particle represents a single solution with a velocity which is dynamically adjusted according to its own experience and that of its neighboring companions. And the population of particles is updated based on each particle's previous best performance and the best particle in the population. In this way, PSO combines local search with global search for balancing the exploration and exploitation. Considering a d-dimensional search space, the ith particle is represented as $\vec{X}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$, and its according velocity is represented as $\vec{V}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,d})$. The best previous position of the ith particle that gives the best fitness value is represented as $\vec{P}_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,d})$. The best particle among all the particles in the population is represented as $\vec{P}_g = (p_{g,1}, p_{g,2}, \ldots, p_{g,d})$. In every iteration, each particle updates its position and velocity according to the two best values.

2.2.1. PSO with adaptive control parameters

In order to reduce the dependence of the search process on the hard bounds of the velocity, the concept of an inertia weight wt was introduced in the PSO algorithm [28]. The velocity and position are updated as follows:

$$v_{i,j}^{n+1} = wt \times v_{i,j}^{n} + c_1 \times r_1 \left(p_{i,j}^{n} - x_{i,j}^{n}\right) + c_2 \times r_2 \left(p_{g,j}^{n} - x_{i,j}^{n}\right), \tag{7}$$

$$x_{i,j}^{n+1} = x_{i,j}^{n} + v_{i,j}^{n+1}, \quad j = 1, 2, \ldots, d, \tag{8}$$

where c_1 and c_2 are two acceleration coefficients, which define the magnitude of the influences on the particles velocity in the directions of the local and the global optima, respectively. To better balance the search space between the global exploration and local exploitation, time-varying acceleration coefficients (TVAC) have been introduced in [29]. This concept will be adopted in this study to ensure the better search capability for the solutions. The core idea of TVAC is that c_1 decreases from its initial value of c_{1i} to c_{1f}, while c_2 increases from c_{2i} to c_{2f} using the following equations as in [29]. TVAC can be mathematically represented as follows:

$$c_1 = (c_{1f} - c_{1i})\frac{t}{t_{\max}} + c_{1i}, \tag{9}$$

$$c_2 = (c_{2f} - c_{2i})\frac{t}{t_{\max}} + c_{2i}, \tag{10}$$

where c_{1f}, c_{1i}, c_{2f} and c_{2i} are constants, t is the current iteration of the algorithm, and $t_{\max}$ is the maximum number of iterations.

In addition, r_1 and r_2 in Eq. (7) are random numbers, generated uniformly in the range $[0, 1]$. The velocity $v_{i,j}$ is restricted to the range $[-v_{\max}, v_{\max}]$, in order to prevent the particles from flying out of the solution space. The inertia weight wt, which is used to balance the global exploration and local exploitation, a large inertia weight facilitates the global search, while a small inertia weight facilitates the local search. In order to reduce the weight over the iterations allowing the algorithm to exploit some specific areas, the inertia weight wt is updated according to the following equation:

$$wt = wt_{\min} + (wt_{\max} - wt_{\min})\frac{(t_{\max} - t)}{t_{\max}}, \tag{11}$$

where $wt_{\max}$, $wt_{\min}$ are the predefined maximum and minimum values of the inertia weight wt, t is the current iteration of the algorithm and $t_{\max}$ is the maximum number of iterations. Usually the value of wt is varied between 0.9 and 0.4. Eq. (11) is also known as the time-varying inertia weight (TVIW) [28], which has been shown to significantly improve the performance of PSO [30], since TVIW makes PSO have more global search ability at the beginning of the run and have more local search ability near the end of the run. In order to further improve the performance, wt is modified to be a nonlinearly decreasing inertia weight in this study, which is defined as follow:

$$wt = wt_{\min} + (wt_{\max} - wt_{\min})\left\{\frac{(t_{\max} - t)^n}{(t_{\max})^n}\right\}, \tag{12}$$

where n is the linearly adaptation parameter, the PSO algorithm can achieve the best balance between the global and local search through changing the different value of n.

2.2.2. Discrete binary PSO

PSO was originally introduced as an optimization technique for continuous space. In order to extend the application to discrete spaces, Kennedy and Eberhart [31] proposed a discrete binary version of PSO where a particle moves in a state space restricted to zero and one on each dimension, in terms of the changes in probabilities that a bit will be in one state or the other. If the velocity is high it is more likely to choose 1, and lower values favor choosing 0. A sigmoid function is applied to transform the velocity from continuous space to probability space:

$$sig(v_{i,j}) = \frac{1}{1+\exp(-v_{i,j})}, \quad j = 1,2,\ldots,d. \tag{13}$$

The velocity update Eq. (7) keeps unchanged except that $x_{i,j}$, $p_{i,j}$ and $p_{g,j} \in \{0, 1\}$, and in order to ensure that bit can transfer between 1 and 0 with a positive probability $v_{\max}$ was introduced to limit $v_{i,j}$. The new particle position is updated using the following rule:

$$x_{i,j}^{n+1} = \begin{cases} 1, & \text{if } rnd < sig(v_{i,j}) \\ 0, & \text{if } rnd \geqslant sig(v_{i,j}) \end{cases}, \quad j = 1,2,\ldots,d, \tag{14}$$

where $sig(v_{i,j})$ is calculated according to Eq. (13), and rnd is a uniform random number in the range [0,1].

2.2.3. Modified discrete binary PSO

The traditional discrete binary PSO can achieve good results in solving the combinatorial optimization problem, but its effect remains to be further improved. The feature selection refers to choosing subset of features from the set of original features, which is a kind of combinatorial optimization problem. According to the information sharing mechanism of PSO, Shen [32] proposed a modified binary PSO for feature selection. This new discrete binary PSO is further modified in this study to adapt to the optimization of the feature selection task. The modified discrete binary PSO is defined as:

$$\text{If } \left(0 \leqslant v_{ij} \leqslant \frac{(1-a)}{2}\right), \quad \text{then } x_{ij}(new) = x_{ij}(old), \quad j = 1,2,\ldots,d, \tag{15}$$

$$\text{If}\left(\frac{(1-a)}{2} < v_{ij} \leqslant \frac{(1+a)}{2}\right), \quad \text{then } x_{ij}(new) = p_{ij}, \quad j = 1,2,\ldots,d, \tag{16}$$

$$\text{If } \left(\frac{(1+a)}{2} < v_{ij} \leqslant 1\right), \quad \text{then } x_{ij}(new) = p_{gj}, \quad j = 1,2,\ldots,d, \tag{17}$$

where parameter a is a random value in the range of (0,1), which plays the role of balancing the global and local search. The larger the value of parameter *a is*, the bigger the chance of escaping the local minima is. x_{ij} is the current position of a particle, which is composed of the total features of the data, and are assigned 0 or 1, respectively, indicates whether the corresponding feature is selected or not. p_{ij} is the personal optimal value of each particle and p_{gj} is the global optimal value among all particles. v_{ij} is the velocity whose value is randomly distributed in the range of [0,1].

Eq. (15) is similar to the first term of the right hand side of Eq. (7), which is the velocity of the previous iteration, without the first term, the particle velocity is only determined by the best particle positions found so far (i.e., both personal and global best positions). Eqs. (16) and (17) are similar to the second and third terms of the right hand side of Eq. (7), which take into account the particle's individual experience and the interaction between the particles, respectively. Without the latter two terms, the particle will keep on flying along the same direction until it reaches the boundary of the search space. It can be seen as a behavior which tries to explore new search areas.

3. The proposed PTVPSO-SVM method

In this section, we describe the proposed PTVPSO-SVM method, which combines the parameter optimization with the feature selection together, in order to achieve the highest performance. The proposed approach is comprised of two stages as shown in Fig. 1. In the first stage, both SVM parameter optimization and feature selection are dynamically conducted by implementing PSO algorithm simultaneously. In the second stage, SVM model performs the classification tasks using the optimal parameter pair and feature subset via cross validation (CV) analysis. PTVPSO-SVM takes into consideration three fitness values for parameter optimization and feature selection. The first one is the ACC of SVM, the second one is the number of SVs and the last one is the number of selected features. In this way, the PTVPSO-SVM can not only achieve high classification accuracy, but also obtain good capability of generalization. Here, we first describe the method based on the serial PSO algorithm, termed TVPSO-SVM, and then implement it in a parallel way.

3.1. The TVPSO-SVM method based on the serial PSO algorithm

The TVPSO-SVM method was constructed through the following main steps.

Step 1: Encode the particle with $n+2$ dimensions. The first two dimensions are C and γ which are continuous values. The remaining n dimensions is Boolean features mask, which is represented by discrete value, '1' indicates the feature is selected, and '0' represents the feature is discarded.
Step 2: Initialize the individuals of the population with random numbers. Meanwhile, specify the PSO parameters such as the lower and upper bounds of the velocity, the size of particles and the number of iterations.
Step 3: Train the SVM model with the selected feature subset in Step 2.

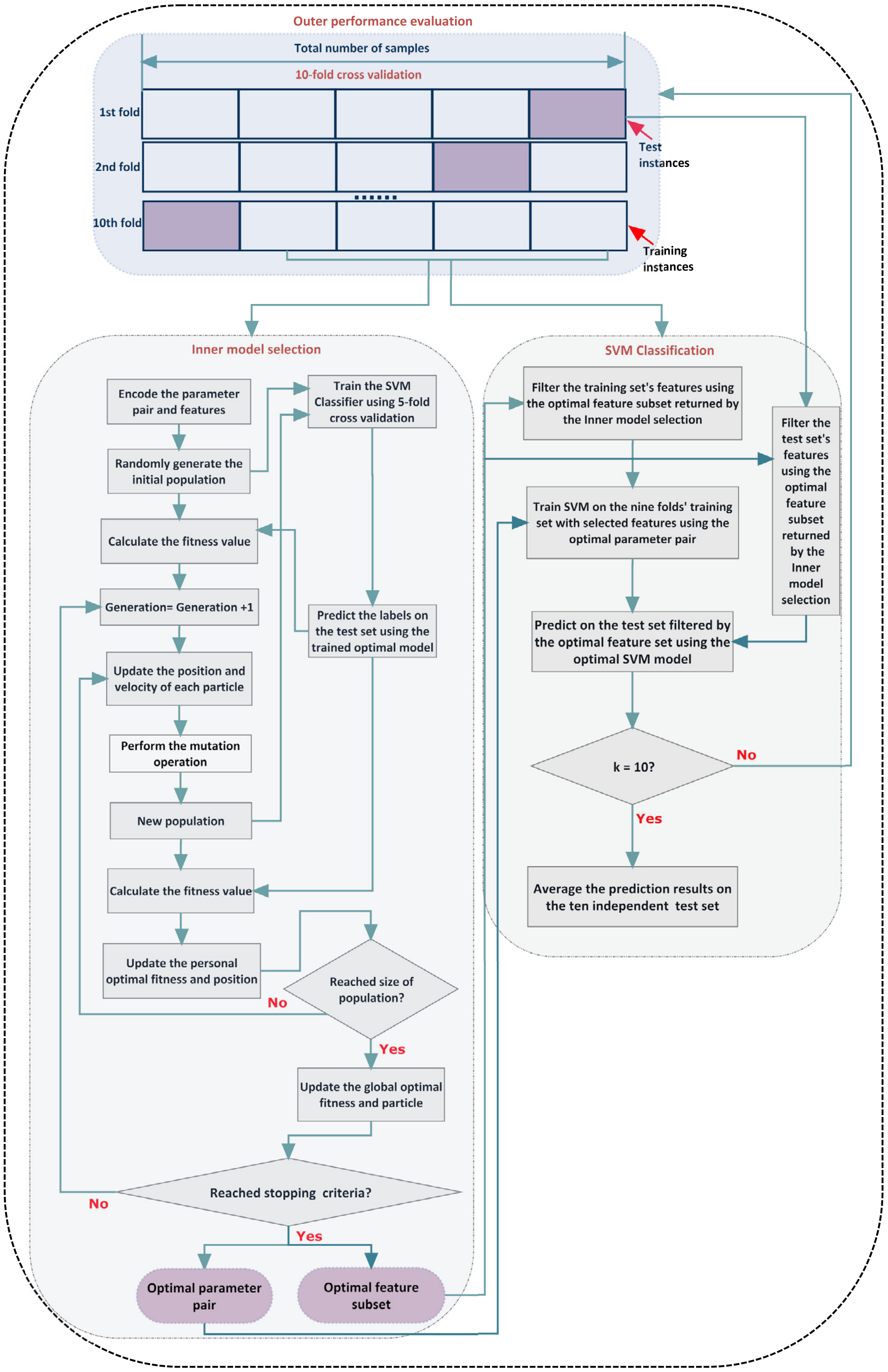
Outer performance evaluation
Total number of samples
10-fold cross validation
1st fold
2nd fold
10th fold
Test instances
Training instances
Inner model selection
Encode the parameter pair and features
Randomly generate the initial population
Calculate the fitness value
Generation= Generation +1
Update the position and velocity of each particle
Perform the mutation operation
New population
Calculate the fitness value
Update the personal optimal fitness and position
Train the SVM Classifier using 5-fold cross validation
Predict the labels on the test set using the trained optimal model
Reached size of population?
No
Yes
Update the global optimal fitness and particle
Reached stopping criteria?
No
Yes
Optimal parameter pair
Optimal feature subset
SVM Classification
Filter the training set's features using the optimal feature subset returned by the Inner model selection
Train SVM on the nine folds' training set with selected features using the optimal parameter pair
Predict on the test set filtered by the optimal feature set using the optimal SVM model
Filter the test set's features using the optimal feature subset returned by the Inner model selection
k = 10?
No
Yes
Average the prediction results on the ten independent test set

Fig. 1. Overall procedure of the proposed PTVPSO-SVM method.

Step 4: The particle with high classification accuracy and the small number of selected features can produce a high fitness value. In addition, the particle with smaller number of SVs can achieve higher classification accuracy, since the number of SVs is proportional to the generalization error of the SVM classifier [3]. Thus, we take all of them into account to design the objective function. The fitness value is calculated according to the following function:

$$\begin{cases} f_1 = avgacc = \frac{\Sigma_{i=1}^{K} Test\_Accuracy_i}{K} \\ f_2 = (1 - \frac{nsv}{m}) \\ f_3 = \left(1 - \frac{\sum_{j=1}^{n} ft_i}{n}\right) \\ f = \alpha \times f_1 + \beta \times f_2 + \lambda \times f_3, \end{cases} \tag{18}$$

where variable *avgacc* in the first sub-objective function f_1 represents the average test accuracy achieved by the SVM classifier via *K*-fold CV, where *K* = 5. Note that here the 5-fold CV is employed to do the model selection that is different from the outer loop of 10-fold CV, which is used to do the performance estimation. *nsv* and *m* in the second sub-objective function f_2 indicates the number of SVs and training data, respectively. In the third sub-objective function f_3, ft_i is the value of feature mask ('1' represents that feature is selected and '0' indicates that feature is discarded), *n* is the total number of features. The weighted summation of three sub-objective functions is selected as the final objective function. In *f*, variable α is the weight for SVM classification accuracy, β indicates the weight for the number of SVs, and λ represents the weight for the selected features. The weight can be adjusted to a proper value depends on the importance of the sub-objective function. Eq. (18) means that the ACC, the number of SVs and the size of feature subset have different significance to the classification performance. According to our preliminary experiments, the classification performance is more depend on ACC and number of SVs than selected features, so the value α and β are selected as bigger than that of λ. Generally, the weight is set to be constant value. We empirically found that the linearly increasing/decreasing function can further improve the classification performance over most data sets. Thus, we define the weight as the linearly increasing/decreasing function varying along with the iterations. They are taken the form of $\alpha = (\alpha_1 - \alpha_2)\frac{t}{t_{\max}} + \alpha_2$, $\beta = (\beta_1 - \beta_2)\frac{t}{t_{\max}} + \beta_2$, $\lambda = (\lambda_1 - \lambda_2)\frac{t}{t_{\max}} + \lambda_2$ respectively, where $\alpha_1 + \beta_1 + \lambda_1 = 1$, $\alpha_2 + \beta_2 + \lambda_2 = 1$. After the fitness value was obtained, the global optimal fitness was saved as *gfit*, personal optimal fitness as *pfit*, global optimal particle as *gbest* and personal optimal particle as *pbest*.

Step 5: Increase the number of iteration.

Step 6: Increase the number of population. Update the position and velocity of *C* and γ in each particle according to Eqs. (7) and (8), and the features in each particle according to Eq. (7) and Eqs. (15)–(17).

Step 7: In order to explore further the search space for the PSO algorithm, the mutation strategy is adopted for the continuous type of dimensions, which represent *C* and γ respectively. The following mutation operator is adopted, which has been also used in PSO for function optimization problems in [33]. The mathematical form of the mutation operator takes the following form:

$$x_k = \begin{cases} x_k + \Delta(t, UB - x_k) & \text{if } rand = 0 \\ x_k - \Delta(t, x_k - LB) & \text{if } rand = 1, \end{cases} \tag{19}$$

where x_k denotes one variable of the first two dimension in each particle, i.e., *C* or γ, while x'_k denotes the value after the mutation operation. The variable *rand* denotes the random value 0 or 1. *UB* denotes the upper bound of the variable x_k, while *LB* denotes the lower bound. The function Δ is given by

$$\Delta(t, y) = y * (1 - r^{(1-\frac{t}{t_{\max}})^b}), \tag{20}$$

where *r* is a random number in the range $[0, 1]$, *t* is the current iteration of the algorithm and t_{max} is the maximum number of iterations. The parameter *b* is a system parameter which determines the degree of dependence of mutation on the number of the iteration.

Step 8: Train the SVM model with the selected feature subset in Step 6 and calculate the fitness value of each particle according to Eq. (18).

Step 9: Update the personal optimal fitness (*pfit*) and personal optimal position (*pbest*) by comparing the current fitness value with the *pfit* stored in the memory. If the current fitness is dominated by the *pfit* stored in the memory, then keep the *pfit* and *pbest* in the memory; otherwise, replace the *pfit* and *pbest* in the memory with the current fitness value and particle position.

Step 10: If the size of the population is reached, then go to Step 11. Otherwise, go to Step 6.

Step 11: Update the global optimal fitness (*gfit*) and global optimal particle (*gbest*) by comparing the *gfit* with the optimal *pfit* from the whole population, If the current optimal *pfit* is dominated by the *gfit* stored in the memory, then keep the *gfit* and *gbest* in the memory; otherwise, replace the *gfit* and *gbest* in the memory with the current optimal *pfit* and the optimal *pbest* from the whole population.

Step 12: If the stopping criteria are satisfied, then go to Step 13. Otherwise, go to Step 5. The termination criteria are that the iteration number reaches the maximum number of iterations or the value of *gfit* does not improve after 100 consecutive iterations.

Step 13: Get the optimal (C, γ) and the feature subset from the best particle (*gbest*).

3.2. Parallel implementation of the TVPSO-SVM method (PTVPSO-SVM)

When dealing with practical problems, especially those with large scales, the evolutionary-based methods such as PSO and GA will cost a lot of computational time. There is an urgent need to improve the performance using high-performance computing techniques. Consequently, we attempt to implement TVPSO-SVM in parallel using PVM to speed up the search and optimization process.

The parallel architecture of PTVPSO-SVM is implemented through a heterogeneous environment consisting of one scheduling server, multiple scheduling workstations and several workstations, as shown in Fig. 2. The scheduling server analyzes the total number of tasks from the obtained problems and the current load of the scheduling workstations, and builds a task list comprising a set of sub-task lists as well. All of the scheduling workstations are managed by a scheduling server. The jobs are dynamically dispatched from the sub-tasks to the computing nodes according to the load balancing of computing nodes, which are managed by the current scheduling workstation. The sub-task list consists of a job list that is a set of jobs for each computing node. In the proposed parallel method, PVM, a software package enables users to exploit their existing computer hardware to solve much larger problems at minimal additional cost. The flowchart of the PTVPSO-SVM is presented in Fig. 3.

The pseudo codes of the PTVPSO-SVM are described in detail as follows:

```
Pseudo-code for scheduling server procedure
pro_Scheduling_Server
Begin
    fn_Encoding(DataSource) //Encoding C, $\gamma$ and features
    DataSource = fn_Init(Data) //Create initial particle with feasible random numbers
    fn_TrainSVM(DataFeatrue) //Train the SVM model with the selected feature subset
    Fitness = fn_CalculateFitness(DataSource) //Calculate fitness value
While (Termination)
    task_list = fn_BuildTaskList() //Build task-lists
  While (TaskNumber == 0)
      fn_Dispatch(task_list) //Dispatch tasks to scheduling workstations
    TaskNumber = TaskNumber − 1
  End While
    DataSource = fn_MergeDFSW() //Merge data from scheduling workstations
    Gfit = fn_UpdateGOF(DataSource) //Update the global optimal fitness (gfit)
    Gbest = fn_UpdateGOP(DataSource) //Update the global optimal particle (gbest)
  End While
  Result = fn_GetOptimal(DataSource) //Get the optimal C, $\gamma$ and feature subset from gbest
End
```

```
Pseudo-code for scheduling workstation procedure
pro_Scheduling_Workstation
Begin
    //Wait for receiving a new sub-task, a result from a node or a termination signal
  While (Waiting (Flag))
    If Flag == Task_ptr
      While (TaskNumber == 0)
          Dispatch (task_list) //Dispatch tasks to nodes
        TaskNumber = TaskNumber − 1
      End While
    End If
    If Flag == Result_ptr
      Results = fn_ReceiveResult()
    If Flag == Termination_ptr
      Break;
    End If
  End While
  DataSource = fn_MergeDFN(Results) //Merge data from nodes
End
```

```
Pseudo-code for nodes procedure
pro_Nodes
Begin
    While (Waiting (Flag)) //Wait for receiving a new Job
    If Flag == Job_ptr
        Position = fn_UpdataPosition(DataSource) //Update the position of each particle
        Velocity = fn_UpdataVelocity(DataSource) // Update the velocity of each particle
        fn_Mutation(DataSource) //Perform the mutation operation
        Model = fn_Train(DataSource) //Train SVM model
        Fitness = fn_Calculate_Fitness(DataSource) //Calculate the fitness value
        Pfit = fn_UpdataPFit(DataSource) //Update the personal optimal fitness (pfit)
        Pbest = fn_UpdataPbest(DataSource) //Update the personal optimal position (pbest)
    End if
    If Flag == Termination_ptr
      Break;
    End if
  End While
      fn_SendToWorkstation(Results) //Send results to scheduling workstation
End
```

4. Experimental designs

4.1. Data description

In order to evaluate the performance of the proposed method in different classification tasks, a comprehensive set of 30 benchmark data sets taken from the UCI Machine Learning and StatLog databases were tried in this investigation. Those data sets have 3–100 attributes and 2–11 classes, and thus covering a wide range of conditions. The detail characteristics of these data sets are presented in Table 1.

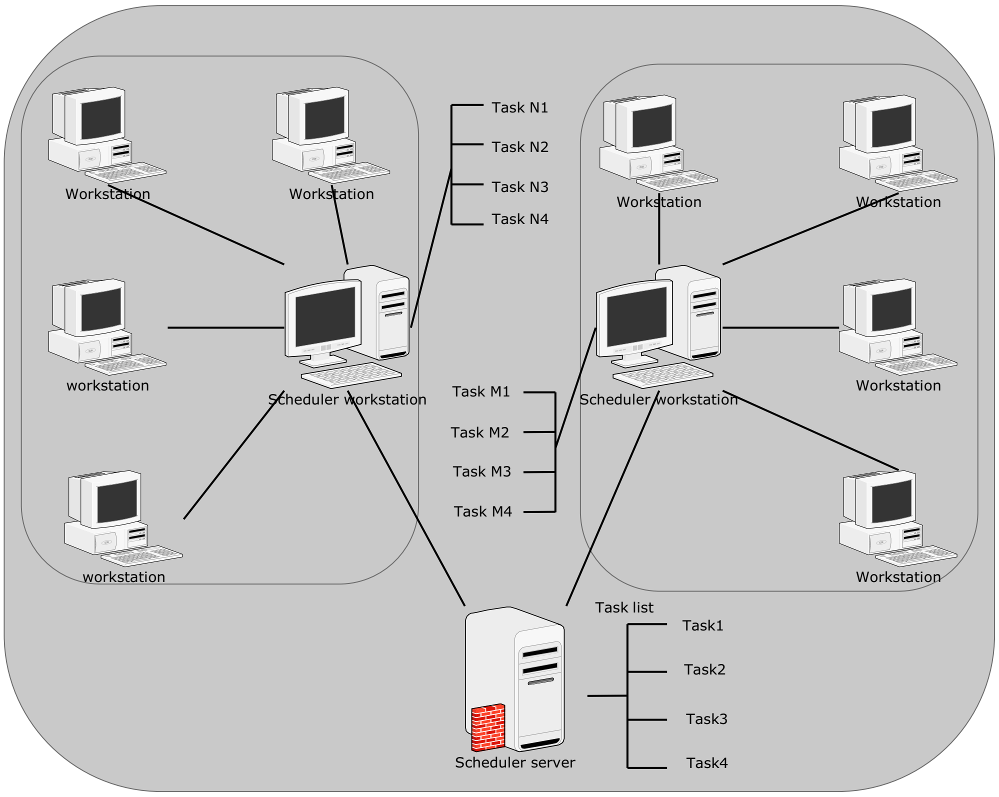

Fig. 2. The parallel architecture of the PTVPSO-SVM.

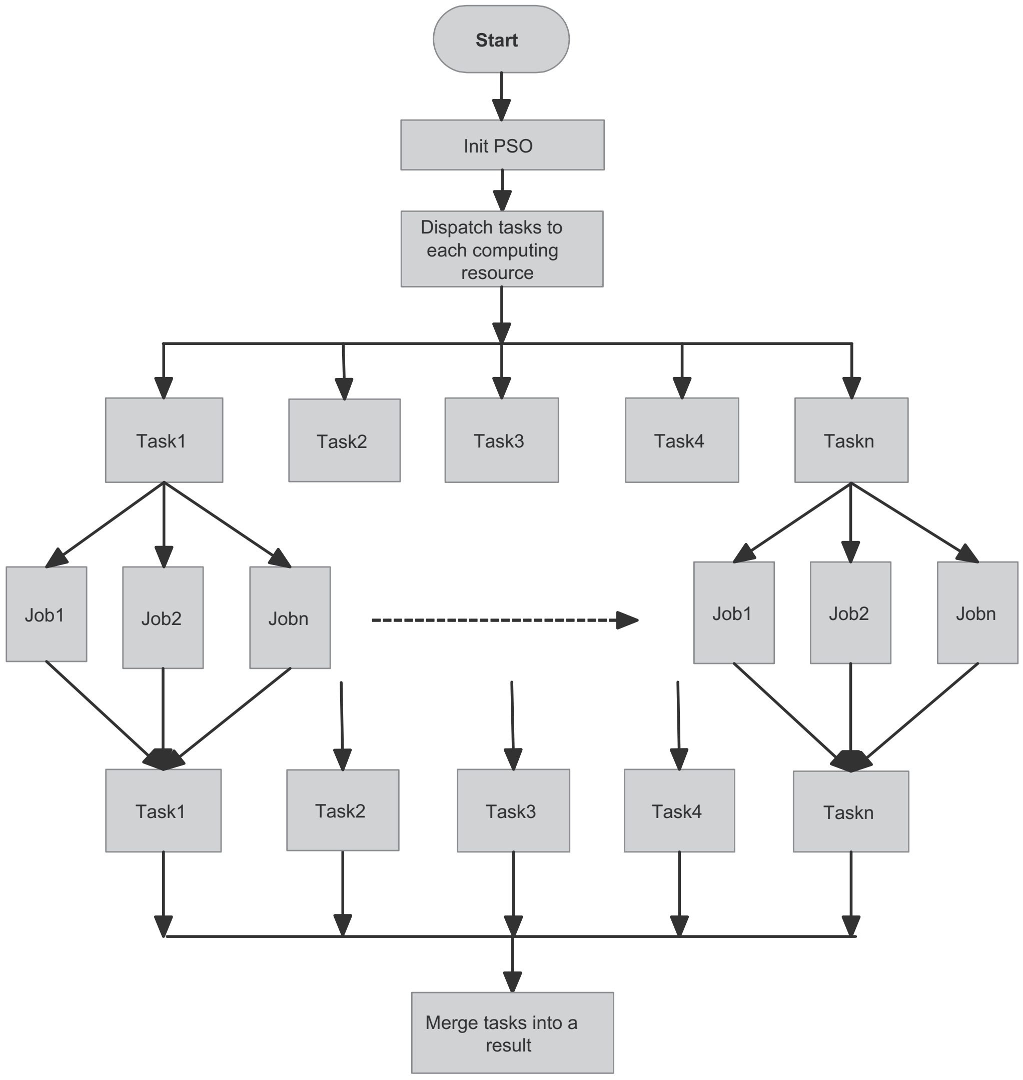

Fig. 3. The flowchart of the PTVPSO-SVM.

Some of these data sets contain missing values. The missing categorical attributes are replaced by the mode of the attributes and the missing continuous ones are replaced by the mean of the attributes. In addition, all the categorical attributes were changed to attributes with integer values to enable the chosen algorithms to handle them. For all considered problems the input attributes are first scaled so that they lie in a suitable range. Usually, the data could be normalized by scaling them into the interval of $[-1, 1]$ according to the Eq. (21), where x is the original value, x' is the scaled value, $\max_a$ is the maximum value of feature a, and min_a is the minimum value of feature a.

$$x' = \left(\frac{x - \min_a}{\max_a - \min_a}\right) \times 2 - 1. \tag{21}$$

In order to guarantee the valid results, the k-fold CV presented by Salzberg [34] was used to evaluate the classification accuracy. This study set k as 10, i.e., the data was divided into ten subsets. For each time, one of the ten subsets is used as the test set and the other nine subsets are put together to form a training set. Then the average error across all ten trials is computed. The advantage of this method is that all of the test sets are independent and the reliability of the results could be improved. In order to ensure the same class distribution in the subset, the data is split via stratified sampling in which the sample proportion in each data subset is the same as that in the population. It is worth noting that the test data used in the test stage is isolated from the training data used in the training stage, namely the optimal (C, γ) and feature subset are obtained from the training dataset, and then the test dataset is used to obtain the average CV accuracy in the testing stage, thus preventing it from obtaining the over-estimate the accuracy.

Note that only one repetition of the 10-fold CV will not generate enough classification accuracies for comparison. Because of the arbitrariness of partition of the data set, the predicted accuracy of a model at each iteration is not necessarily the same.

Table 1
Description of data sets used in the experiments.

No.	Data sets	# of classes	# of instances	# of features	Miss.
1	Wisconsin breast cancer (Wisconsin)	2	699	9	Yes
2	Australian credit (Australian)	2	690	14	Yes
3	Germen credit (German)	2	1000	24	No
4	Pima Indians diabetes (Pima)	2	768	8	No
5	Hepatitis	2	155	19	Yes
6	Statlog heart (Heart)	2	270	13	No
7	Cleveland heart (Cleveland)	2	303	13	Yes
8	Bupa liver disorders (Bupa)	2	345	6	No
9	Wine	3	178	13	No
10	Image segmentation (Segment)	7	2310	19	No
11	Vowel recognition (Vowel)	11	990 (528/462)	10	No
12	Ionosphere	2	351	34	No
13	Sonar	2	208	60	No
14	Wisconsin diagnostic breast cancer (WDBC)	2	569	30	No
15	Teaching assistant evaluation (Teaching)	3	151	5	No
16	Parkinsons	2	195	22	No
17	Thyroid	3	215	5	No
18	Blood transfusion (Blood)	2	748	4	No
19	Glass	6	214	9	No
20	Iris	3	150	4	No
21	Vehicle silhouettes (Vehicle)	4	846	17	No
22	Haberman survival (Haberman)	2	306	3	No
23	Wisconsin prognostic breast cancer (WPBC)	2	198	34	Yes
24	Hill-valley	2	1212 (606/606)	100	No
25	Mammographic-masses (Mammo)	2	961	5	Yes
26	Contraceptive method choice (CMC)	3	1473	9	No
27	Landsat satellite (Landsat)	6	6435 (4435/2000)	36	No
28	Yeast	10	1484	8	No
29	Zoo	7	101	17	No
30	Monks1	2	556 (124/432)	7	No

Miss.' means 'missing value'. In (A/B), A represents the training set, B denotes the test set.

To evaluate accurately the performance of the data sets, the 10-fold CV will be repeated 10 times and then obtained results will be averaged. Note that Vowel, Hill-Valley, Landsat Satellite and Monks1 data sets have pre-defined training/test splits. Thus, except these data sets, all of the experimental results are averaged over the 10 runs of 10-fold CV.

4.2. Experimental scheme

The proposed experimental framework is articulated around the following three main experiments.

(1) The first experiment aims at assessing the effectiveness of the PTVPSO-SVM method in the whole original hyper dimensional feature space. For comparison purpose, we implemented the standard PSO algorithm based method (PSO-SVM) as in [20] and the gird search based method (GRID-SVM).
(2) In the second experiment, it is plan to assess the capability of the PTVPSO-SVM with feature selection to enhance further the performance of the SVM classifier by using the time variant PSO approach.
(3) The third experimental part is designed to evaluate the capability of the proposed parallel TVPSO-SVM method with respect to the CPU time and the number of SVs.

4.3. Experimental setup

The proposed PTVPSO-SVM method is implemented using Microsoft visual C++ 2008, PVM and LIBSVM. For SVM, LIBSVM implementation was utilized, which is originally developed by Chang and Lin [12]. We implemented the parallel PSO from scratch. The empirical experiment was conducted on the platform with 8 workstations, 2 scheduler workstations and 1 scheduler server, each with quad-core Intel Xeon 2.0 GHz CPU and 4 GB RAM.

The detail parameter setting for PTVPSO-SVM is set as follows. The number of the iterations and particles is set to 250 and 8, respectively. The searching ranges for C and γ are as follows: $C \in [2^{(-10)}, 2^{(15)}]$ and $\gamma \in [2^{(-15)}, 2^{(10)}]$. v_{max} is set about 60% of the dynamic range of the variable on each dimension for the continuous type of dimensions. For the discrete type particle for feature selection, $[-v_{max}, v_{max}]$ was set as [0,1]. As suggested in [29], c_{1i}, c_{1f}, c_{2i} and c_{2f} were set as follows: $c_{1i} = 2.5$, $c_{1f} = 0.5$, $c_{2i} = 0.5$, $c_{2f} = 2.5$. According to our preliminary experiment, wt_{max} and wt_{min} were set to 0.9 and 0.4, the system parameter b in mutation operator to 5, the parameter a in the discrete binary PSO to 0.5, the parameter n in the inertia weight wt to 2. Additionally, the parameters of α_1, α_2, β_1, β_2, λ_1 *and* λ_2 are taken as $\alpha_1 = 0.3$, $\alpha_2 = 0.6$, $\beta_1 = 0.7$, $\beta_2 = 0.3$, $\lambda_1 = 0$, $\lambda_2 = 0.1$, respectively. In view of some difficult classification tasks, we gave LIBSVM a cache size of 640 M.

For PSO-SVM, here we use the same setting as adopted in [20], i.e., the acceleration coefficients c_1 and c_2 were set to 2, the number of the iterations and particles were set to 50 and 6 when not considering feature selection, while 250 and 8 when considering feature selection. The searching ranges for C and γ are as follows: $C \in [0.01, 35{,}000]$ and $\gamma \in [0.0001, 32]$. The velocity range for feature selection, namely $[-v_{max}, v_{max}]$ was set as $[-6,6]$. The objective function was designed the same one as in [20], i.e., the ACC was used to design the fitness of each particle. And the same feature selection scheme in [20] is adopted for PSO-SVM, namely, if the value of a variable is less than or equal to 0.5, then its corresponding feature is not chosen. Conversely, if the value of a variable is greater than 0.5, its corresponding feature will be chosen. For GRID-SVM, the range of the related parameters C and γ were varied between $C = \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma = \{2^{-15}, 2^{-13}, \ldots, 2^{1}\}$, and the grid search technique [35] is employed using 5-fold CV to find out the optimal parameter values of RBF kernel function.

5. Experimental results and discussion

5.1. Experiment I

As mentioned before, in this experiment we attempted to assess the effectiveness of the PTVPSO-SVM without feature selection. Table 2 summarizes the results of the ACC for the 30 data sets using PTVPSO-SVM without feature selection, PSO-SVM without feature selection and GRID-SVM. It should be noted that we adopted the same experimental setup for PSO-SVM as used in [20], but we cannot get that high accuracy on some data sets as reported in [20]. Thus, here we report the results achieved in our own experiment environment for comparison. It can be observed that, the classification accuracies achieved by the developed method is much better than those of PSO-SVM and RID-SVM in all of the data sets.

In order to verify the effectiveness of the proposed method, a paired t-test on the ACC is employed for all of the data sets. As shown in Table 2, the developed PTVPSO-SVM performs significantly better than both PSO-SVM and GRID-SVM in almost all cases examined; only the Bupa, Wine and WDBC data sets did not exhibit the significant improvement between PTVPSO-SVM and PSO-SVM. It reveals that the developed method can obtain more appropriate parameters and yield higher ACC in comparison with two other methods. The better performance of the proposed method can be attributed to all features, i.e., adaptive control parameters (including TVIW and TVAC), mutation operators, modified binary PSO algorithm and consideration of the three sub-objectives (ACC, number of SVs and selected features) in the objective function.

In order to further examine the superior performance of the developed method to that of PSO-SVM and GRID-SVM, we conducte a detailed comparison study among the three methods on the Wisconsin and German data sets. For the sake of

Table 2
Comparison with two existing methods in terms of ACC on all data sets.

Data sets	(1) PTVPSO-SVM without FS (%)	(2) PSO-SVM without FS (%)	(3) GRID-SVM (%)	p-value (1) vs. (2)	p-value (1) vs. (3)
Wisconsin	98.62	97.55	96.62	(<0.0)*	(<0.0)*
Australian	88.05	86.93	84.91	(<0.0)*	(<0.0)**
German	78.02	76.34	75.33	(<0.0)*	(<0.0)*
Pima	78.14	77.58	76.65	(<0.0)*	(<0.0)**
Hepatitis	86.01	84.67	81.10	(<0.0)*	(<0.0)*
Heart	86.75	85.24	82.81	(<0.0)*	(<0.0)*
Cleveland	87.21	86.55	83.44	(<0.0)*	(<0.0)**
Bupa	74.76	72.03	70.95	0.061	(<0.0)*
Wine	98.99	98.64	98.20	0.075	(<0.0)*
Segment	99.01	98.41	97.10	(<0.0)*	(<0.0)**
Vowel	65.22	64.18	62.77	(<0.0)*	(<0.0)*
Ionosphere	95.21	94.34	93.90	(<0.0)*	(<0.0)*
Sonar	92.66	91.27	88.98	(<0.0)*	(<0.0)**
WDBC	98.44	98.01	97.45	0.085	(<0.0)*
Teaching	62.86	60.77	59.68	(<0.0)*	(<0.0)**
Parkinson	96.56	95.13	93.39	(<0.0)*	(<0.0)**
Thyroid	98.03	97.25	95.71	(<0.0)**	(<0.0)**
Blood	80.53	79.98	78.32	(<0.0)*	(<0.0)*
Glass	74.66	72.45	69.98	(<0.0)*	(<0.0)**
Iris	98.26	97.33	95.67	(<0.0)*	(<0.0)*
Vehicle	87.67	84.98	84.27	(<0.0)**	(<0.0)**
Haberman	75.77	74.21	72.29	(<0.0)*	(<0.0)**
WPBC	81.22	79.33	77.48	(<0.0)*	(<0.0)*
Hill-valley	73.21	71.92	69.80	(<0.0)*	(<0.0)**
Mammo	86.44	83.56	82.75	(<0.0)**	(<0.0)**
CMC	58.66	56.67	53.85	(<0.0)*	(<0.0)*
Landsat	94.52	93.45	91.15	(<0.0)*	(<0.0)**
Yeast	65.87	63.42	61.23	(<0.0)*	(<0.0)*
Zoo	96.67	94.55	93.55	(<0.0)*	(<0.0)*
Monks1	84.98	83.64	80.56	(<0.0)*	(<0.0)**

FS means feature selection.
(<0.0)* and (<0.0)** mean the p-value is lower than the level of 0.05 and 0.005 respectively.

Table 3
Comparison with two existing methods in terms of ACC for the Wisconsin data set.

Fold	PTVPSO-SVM without FS			PSO-SVM without FS			GRID-SVM		
	C ($\times 10^3$)	γ ($\times 10^{-5}$)	ACC (%)	C ($\times 10^3$)	γ ($\times 10^{-5}$)	ACC (%)	C	γ	ACC (%)
#1	2.648	3.052	98.571	3.493	1.000	94.285	0.500	0.125	92.753
#2	3.817	3.052	100.00	4.462	1.000	100.000	0.125	0.125	98.571
#3	32.768	3.052	98.571	4.770	1.000	98.571	8.000	0.125	98.571
#4	5.367	3.052	96.418	13.168	1.000	95.714	0.125	0.125	95.714
#5	24.929	3.052	100.00	11.376	1.000	95.714	0.125	0.125	95.714
#6	0.821	3.052	98.571	15.864	1.000	97.142	0.125	0.125	100.000
#7	23.171	3.052	98.571	28.004	1.000	95.714	4.000	0.125	94.286
#8	10.442	3.052	100.000	14.422	1.000	100.000	2.000	0.125	98.571
#9	32.768	3.052	97.142	9.640	1.000	97.142	0.500	0.125	95.714
#10	3.852	3.052	98.571	14.125	1.000	98.571	0.500	0.125	94.286
Avg.	14.058	3.052	98.642	11.932	1.000	97.285	1.600	0.125	96.418
Dev.	12.929	0.000	1.190	7.228	0.000	1.958	2.565	0.000	2.375

FS means feature selection.

Table 4
Comparison with two existing methods in terms of ACC for the German data set.

Fold	PTVPSO-SVM without FS			PSO-SVM without FS			GRID-SVM		
	C ($\times 10^3$)	γ ($\times 10^{-5}$)	ACC (%)	C ($\times 10^3$)	γ ($\times 10^{-5}$)	ACC (%)	C ($\times 10^{3}$)	γ ($\times 10^{-4}$)	ACC (%)
#1	5.469	3.052	79.000	28.654	1.000	77.000	0.008	78.125	76.000
#2	26.093	3.052	80.000	31.677	1.000	75.000	0.008	312.500	71.000
#3	32.768	3.052	78.000	10.926	1.000	78.000	2.048	1.221	74.000
#4	16.306	3.052	77.000	11.508	1.000	77.000	0.128	4.882	73.000
#5	5.014	3.052	78.000	26.527	1.000	75.000	0.512	4.882	77.000
#6	32.768	3.052	78.000	31.982	1.000	76.000	8.192	0.305	75.000
#7	13.783	3.052	78.000	14.717	1.000	77.000	32.768	4.882	79.000
#8	30.556	3.052	80.000	6.596	1.000	78.000	0.002	312.500	74.000
#9	0.892	3.052	78.000	23.526	1.000	77.000	2.048	1.221	76.000
#10	21.163	3.052	78.000	35.000	1.000	78.000	8.192	312.500	79.000
Avg.	18.481	3.052	78.400	22.111	1.000	76.800	5.391	103.302	75.400
Dev.	12.046	0.000	0.966	10.287	0.000	1.135	10.144	146.224	2.547

FS means feature selection.

simplicity, here we show the detailed results using one time run of 10-fold CV. Tables 3 and 4 summarize the classification accuracy rates and the optimal pairs of (C, γ) obtained by three methods for each fold. As shown in the tables, the ACC of PTVPSO-SVM is 98.64% for the Wisconsin data set, while the ACC of PSO-SVM and GRID-SVM are 97.29% and 96.42% respectively. On the German data set, the PTVPSO-SVM achieved with the ACC of 78.40%, while PSO-SVM and GRID-SVM obtained with the ACC of 76.80% and 75.40% respectively. In order to identify any differences among the three methods in terms of the classification performance, we conducted a paired t-test among them on the two data sets. As indicated in Table 5, PTVPSO-SVM performs significantly superior to the other two methods at the prescribed statistical significance level of 5%. It reveals that the proposed PTVPSO-SVM can obtain much more appropriate parameter settings with high generalization capability.

5.2. Experiment II

In this experiment, we attempt to investigate whether or not feature selection may further improve the classification performance for SVM. Table 6 reports the ACC and the number of selected features achieved by PTVPSO-SVM and PSO-SVM using feature selection on the 30 data sets. Compared with the results obtained by the methods without feature selection as shown in Table 2, feature selection has enhanced further the classification accuracy for SVM on all the data sets as shown in Figs. 4 and 5. Furthermore, the feature selection does not select all features for use in the SVM classification model. Additionally, from Table 6, it can be observed that the classification accuracies achieved by the PTVPSO-SVM are consistently higher than those of PSO-SVM in all cases. Moreover, compared with the PSO-SVM with feature selection, the number of

Table 5
Results of the t-test for three methods in terms of classification accuracy.

Data sets	PTVPSO-SVM vs. PSO-SVM			PTVPSO-SVM vs. GRID-SVM		
	df	t-value	p-value	*df*	t-value	p-value
Wisconsin	9.000	2.387	0.041	9.000	3.040	0.014
German	9.000	3.207	0.011	9.000	3.105	0.013

Table 6
Results of PTVPSO-SVM and PSO-SVM using feature selection.

Data sets	# of original features	PTVPSO-SVM with FS		PSO-SVM with FS	
		ACC (%)	# of selected features	ACC (%)	# of selected features
Wisconsin	9	99.74	5.0	98.34	6.2
Australian	14	90.12	7.7	88.32	8.6
German	24	80.45	12.6	77.65	14.3
Pima	8	79.64	4.6	78.89	5.3
Hepatitis	19	89.43	7.1	86.12	8.0
Heart	13	88.65	7.9	87.04	8.3
Cleveland	13	90.34	6.2	88.21	8.2
Bupa	6	76.87	5.2	74.64	5.7
Wine	13	99.98	5.9	99.04	7.4
Segment	19	99.86	13.6	99.61	14.4
Vowel	13	68.42	6.3	65.68	7.0
Ionosphere	34	98.54	15.5	96.89	17.6
Sonar	60	95.36	25.3	93.57	32.4
WDBC	30	99.87	13.4	99.40	18.3
Teaching	5	65.46	2.3	63.24	3.2
Parkinson	22	98.56	5.6	96.76	8.2
Thyroid	5	99.63	2.5	99.01	3.2
Blood	4	84.73	2.1	81.32	2.6
Glass	9	78.55	3.2	75.87	4.1
Iris	4	99.32	1.2	98.69	1.8
Vehicle	17	90.27	7.3	87.37	9.2
Haberman	3	78.34	1.1	77.02	1.7
WPBC	34	84.45	14.3	82.55	19.2
Hill-Valley	100	75.65	39.2	73.72	48.3
Mammo	5	89.32	2.1	86.81	3.4
CMC	9	61.22	4.6	57.97	5.4
Landsat	36	97.98	12.3	95.65	18.8
Yeast	8	68.17	4.2	65.24	4.9
Zoo	17	98.97	7.3	96.75	9.6
Monks1	7	87.80	2.7	85.84	3.5

FS means feature selection.

Table 7
A comparison of PTVPSO-SVM and PSO-SVM using feature selection on the Wisconsin and German data sets (%).

Data sets	PTVPSO-SVM with FS	PSO-SVM with FS	Paired *t*-test *p*-value
Wisconsin	99.74 ± 0.000	98.34 ± 0.009	<0.01
German	80.45 ± 0.003	77.65 ± 0.005	<0.01

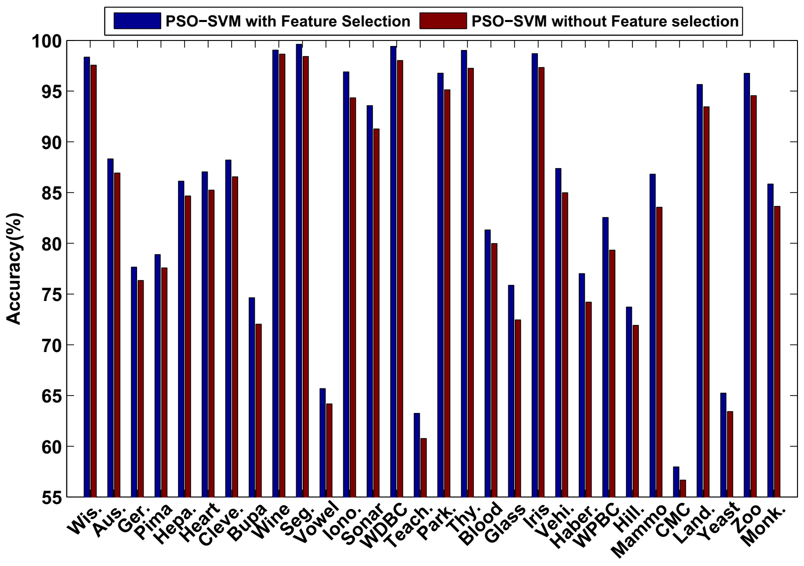

Fig. 4. Predictive accuracy of PSO-SVM with feature selection and PSO-SVM without feature selection.

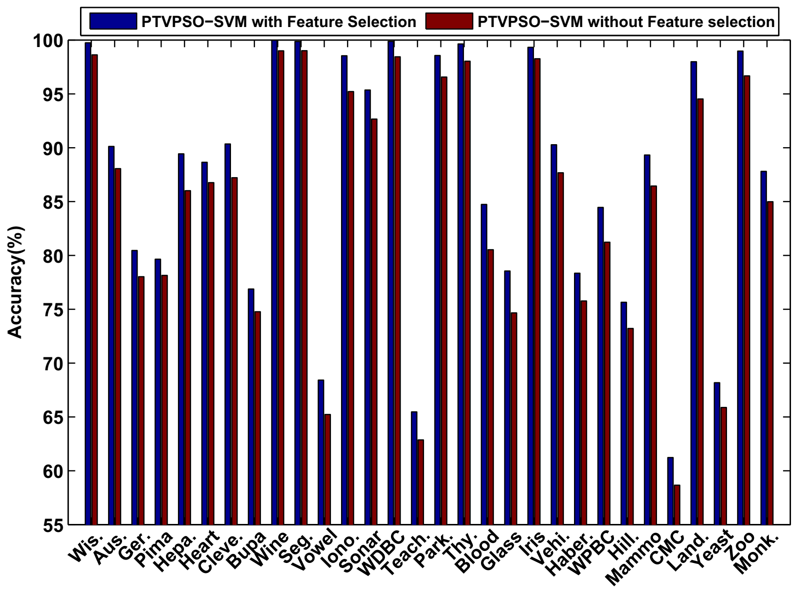

Fig. 5. Predictive accuracy of PTVPSO-SVM with feature selection and PTVPSO-SVM without feature selection.

Table 8
Feature selected for Wisconsin and German data sets by the PTVPSO-SVM method.

Fold	Selected features for Wisconsin	Selected features for German
#1	F_1 F_4 F_7 F_8	F_1 F_2 F_3 F_4 F_5 F_6 F_9 F_{11} F_{17} F_{20} F_{23} F_{24}
#2	F_1 F_2 F_3 F_4 F_9	F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_9 F_{13} F_{16} F_{18} F_{19} F_{22}
#3	F_1 F_3 F_4 F_6 F_9	F_1 F_2 F_3 F_6 F_8 F_9 F_{10} F_{16} F_{19} F_{20} F_{21} F_{22} F_{24}
#4	F_1 F_3 F_4 F_5 F_6 F_9	F_1 F_2 F_3 F_4 F_5 F_6 F_9 F_{10} F_{11} F_{12} F_{14} F_{17} F_{18} F_{20}
#5	F_1 F_4 F_6 F_7 F_9	F_1 F_2 F_3 F_4 F_5 F_8 F_{10} F_{15} F_{16} F_{17} F_{18} F_{22}
#6	F_1 F_3 F_6 F_9	F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_{10} F_{16} F_{17} F_{22}
#7	F_1 F_3 F_4 F_6 F_7	F_1 F_2 F_3 F_4 F_5 F_{10} F_{16} F_{17} F_{18} F_{19} F_{20} F_{22}
#8	F_1 F_3 F_6 F_8 F_9	F_1 F_2 F_3 F_5 F_6 F_7 F_8 F_9 F_{11} F_{12} F_{13} F_{18} F_{20} F_{21}
#9	F_1 F_3 F_4 F_6 F_8 F_9	F_1 F_2 F_4 F_5 F_6 F_9 F_{10} F_{15} F_{17} F_{18} F_{20} F_{21} F_{24}
#10	F_1 F_2 F_3 F_6 F_9	F_1 F_2 F_3 F_4 F_5 F_8 F_{12} F_{13} F_{16} F_{18} F_{21} F_{22}

Table 9
Frequency of selected features in one run 10-fold CV on the Wisconsin data set.

Feature#	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
Frequency	10	2	8	7	1	8	3	3	8

selected features obtained by PTVPSO-SVM is more appropriate (based on the classification accuracy rates). Thus, the developed PTVPSO-SVM method can find the better beneficial subset of features as compared to PSO-SVM. Table 9.

Take the Wisconsin data set and German data set for example. By using the feature selection, the classification accuracies obtained by the PTVPSO-SVM on these two data sets have been improved by 1.1% and 2.4% respectively. For the PTVPSO-SVM method, the average number of the selected features via 10 runs of 10-fold CV was about 5 and 13 for the two data sets respectively, while the average number of features selected by PSO-SVM was about 6 and 14 respectively. The superiority of the PTVPSO-SVM in terms of ACC is statistically significant as shown by paired *t*-test *p*-value reported in Table 7. From the table, it is interesting to find that the standard deviation of the feature subset acquired by PTVPSO-SVM is smaller than that of PSO-SVM, which indicates consistency and stability of the developed method.

To explore how many features and what features will be selected, we further conduct an experiment on the two data sets which are used in the previous experiment to investigate the detailed feature selection mechanism of the PSO algorithm. For the simplicity, here we show the results through one time run of 10-fold CV. The selected features of 10 folds for the Wisconsin data set and German data set are shown in Table 8. The original numbers of features of each data set are 9 and 24, respectively. As shown in the table, not all features are selected for classification after the feature selection on both data sets. Furthermore, feature selection has increased the classification accuracy, as demonstrated in Table 7. For the

Wisconsin data set, the average number of selected features by PTVPSO-SVM is 5.0, and the most important features are F_1, F_3, F_4, F_6 and F_9, which can be found in the frequency of selected features of 10-fold CV as shown in Table 9. For the German data set, the average number of selected features by PTVPSO-SVM is 12.6, and its most important features are F_1, F_2, F_3, F_4, F_5, F_6, F_9, F_{10}, F_{16}, F_{17}, F_{18}, F_{20} and F_{22}, which can be found in the frequency of the selected features of 10-fold CV as shown in Fig. 6.

5.3. Experiment III

As mentioned before, the main aim of this experiment is to evaluate the efficiency of the proposed method with respect to the computational time and the number of SVs. Table 10 summarizes the average results of the four methods (TVPSO-SVM,

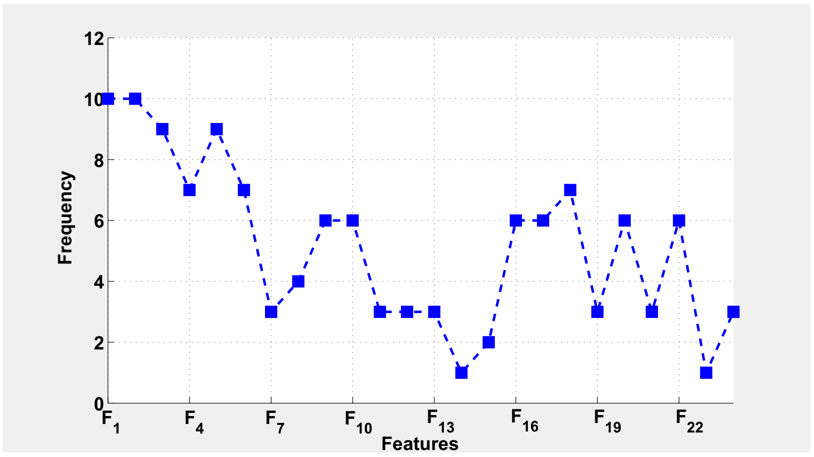

Fig. 6. The frequency of the selected features in one run 10-fold CV process on the German data set.

Table 10
The performance comparison of proposed methods with GRID-SVM.

Data sets	TVPSO-SVM		PTVPSO-SVM		PSO-SVM	GRID-SVM
	# of SVs	ACC (%)	# of SVs	ACC (%)	# of SVs	# of SVs
Wisconsin	54.31	99.71	54.28	99.74	72.43	84.70
Australian	305.82	90.12	305.81	90.12	327.33	344.28
German	465.52	80.46	465.54	80.45	474.51	488.90
Pima	369.35	79.66	369.33	79.64	376.40	388.82
Hepatitis	37.71	89.47	37.68	89.43	44.72	56.31
Heart	94.35	88.63	94.37	88.65	102.25	114.31
Cleveland	103.16	90.36	103.24	90.34	112.26	130.97
Bupa	178.32	76.83	178.29	76.87	195.43	210.08
Wine	41.27	99.97	41.33	99.98	61.43	76.02
Segment	289.72	99.85	289.65	99.86	318.50	335.00
Vowel	366.62	68.42	366.65	68.42	374.59	387.00
Ionosphere	89.34	98.55	89.41	98.54	97.92	111.55
Sonar	101.64	95.37	101.49	95.36	110.90	129.85
WDBC	47.23	99.89	47.21	99.87	54.31	71.33
Teaching	78.23	65.44	78.12	65.46	82.78	99.54
Parkinson	81.33	98.54	81.42	98.56	89.56	97.90
Thyroid	19.43	99.63	19.28	99.63	26.91	34.06
Blood	305.12	84.75	305.23	84.73	314.31	329.77
Glass	98.35	78.52	98.43	78.55	112.38	125.88
Iris	28.63	99.31	28.65	99.32	36.52	41.65
Vehicle	276.54	90.23	276.43	90.27	291.88	310.33
Haberman	118.76	78.36	118.81	78.34	127.75	146.39
WPBC	87.39	84.46	87.32	84.45	91.34	95.28
Hill-Valley	365.44	75.64	365.41	75.65	374.66	389.00
Mammo	328.37	89.30	328.34	89.32	342.45	355.60
CMC	976.65	61.24	976.61	61.22	995.55	1101.60
Landsat	1181.00	97.97	1182.00	97.98	1209.00	1303.00
Yeast	985.48	68.16	985.43	68.17	993.44	1066.50
Zoo	38.06	98.98	38.05	98.97	42.54	48.09
Monks1	46.00	87.81	46.00	87.80	51.00	57.00

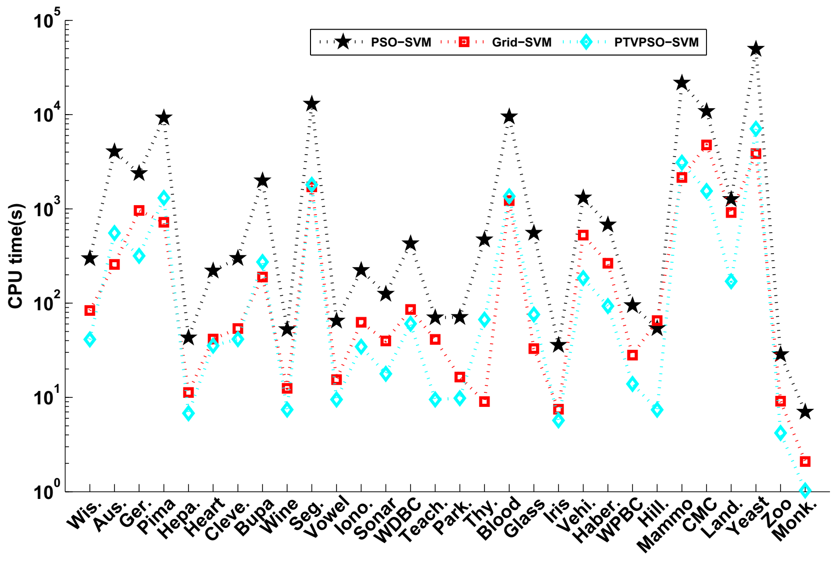

Fig. 7. The average CPU time cost of three methods on each data set.

PTVPSO-SVM, PSO-SVM and GRID-SVM) in terms of the number of SVs and prediction accuracy via 10 runs of 10-fold CV. From Table 10, it can be seen that the GRID-SVM generated much more number of SVs than both TVPSO-SVM and PTVPSO-SVM for all of the cases examined. It is one of the main reasons why the inferior performance acquired by the GRID-SVM as shown in Table 2. The explanation lies in the fact that the number of SVs is proportional to the generalization error of SVM [3]. Both TVPSO-SVM and PTVPSO-SVM almost generate the same number of SVs and prediction accuracy on all of the data sets, the slightly different results may be due to the inexact nature of the optimization process and the randomness of the data partitions. The approximately identical accuracy and number of SVs of the two methods have verified the correctness of the parallel design and implementation. Note that the number of SVs on most data sets is not integer, because they are the average of the 10 runs of 10-fold CV.

As for the cost of computational time, we have computed the running time spent by PTVPSO-SVM, PSO-SVM and Grid-SVM on all the data sets. Note that the average running time over 10 runs of 10-fold CV were reported here. As shown in Fig. 7, it can be seen that PTVPSO-SVM need much less CPU time when compared to PSO-SVM thanks to using high performance computing technique. In addition, the running time of PTVPSO-SVM is less than that of Grid-SVM in most of the data sets. It indicates that the parallel implementation has remarkably reduced the computational time, and it confirms the scalability of the developed method as well. It is worth noticing that there were only 8 computers used to act as the computation nodes in our experiment. With the increasing of the computation nodes, the running time can be further reduced.

6. Conclusions and future work

In this work we present PTVPSO-SVM, a parallel time variant PSO based parameter optimization and feature selection approach for SVM. The main novelty of this method lies in the adopted objective function which aims at maximizing the generalization capability of the SVM classifier. The weighted function simultaneously takes into consideration the classification performance of SVM and the number of SVs, as well as selected features, and the weight of each sub-objective function is defined as a linearly increasing/decreasing function along with the iterations. In addition, the proposed method is implemented in an efficient parallel environment which further enhances the performance of SVM classifier to a great extent in terms of computational time. Moreover, the developed method is adaptive in nature attributed to adaptive control parameters. It can explore larger search space by introducing the mutation operators that overcome the premature convergence of the PSO algorithm. Additionally, the modified binary PSO algorithm has further improved the performance in feature selection task. The evaluation on a comprehensive set of real-world problems reveals that the proposed method can not only achieve high prediction accuracy but also compute efficiently owing to the high performance computing technology. Moreover, compared with PSO-SVM and GRID-SVM, the main advantages of our developed PTVPSO-SVM method are that it can produce much more appropriate model parameters and discriminative feature subset, as well as fewer numbers of SVs.

Based on our empirical analysis, it can be safely concluded that, the developed PTVPSO-SVM method can serve as a promising alternative tool for parameter optimization and feature selection in SVM. More experiments on larger databases should be done to confirm the overall superiority of the proposed method. In addition, we plan to develop a more efficient method, which is optimized with the better scheduling algorithm on scheduling server and scheduling workstation. It should be noted that the recent trends in computer microprocessor development have shifted from a single powerful core to multi-core. Hence, we are going to develop a new method for multi-core computing nodes in order to increase the CPU processor load.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61303113, 61133011, 61373053, 61272018, 61202171, 61373166, 61379095. This work is also funded by the open project program of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University under Grant No. 93K172013K01, Zhejiang Provincial Natural Science Foundation of China under Grant Nos. R1110261, Y1100769, LQ13G010007 and LQ13F020011, the China Postdoctoral Science Foundation funded project (No. 2012M521251), and Wenzhou Major Science and Technology Project Grant (H20100048).

References

[1] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, United States, 1992, pp. 144–152.
[2] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[3] V. Vapnik, Statistical Learning Theory, Wiley, NY, 1998.
[4] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[5] E. Osuna, R. Freund, F. Girosit, Training support vector machines: an application to face detection, in: IEEE Conference on Computer Vision and Pattern Recognition 1997, Los Alamitos, CA, 1997, pp. 130–136.
[6] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the 10th European Conference on Machine Learning, 1998, pp. 137–142.
[7] H.L. Chen, B. Yang, J. Liu, D.Y. Liu, A support vector machine classifier with rough set based feature selection for breast cancer diagnosis, Expert Syst. Appl. 38 (7) (2011) 9014–9022.
[8] S. Keerthi, C. Lin, Asymptotic behaviors of support vector machines with Gaussian kernel, Neural Comput. 15 (7) (2003) 1667–1689.
[9] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (7–8) (2003) 1157–1182.
[10] H. Frohlich, O. Chapelle, B. Scholkopf, Feature selection for support vector machines by means of genetic algorithms, in: Proceedings of the 15th IEEE international conference on tools with artificial intelligence, Sacramento, CA, USA, 2003, pp. 142–148.
[11] H. Frohlich, A. Zell, Efficient parameter selection for support vector machines in classification and regression via model-based global optimization, in: IEEE International Joint Conference on Neural Networks, 2005, pp. 1431–1436.
[12] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines. Software available at <http://www.csie.ntu.edu.tw/cjlin/libsvm>, 2001.
[13] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (1) (2002) 131–159.
[14] C. Gold, P. Sollich, Model selection for support vector machine classification, Neurocomputing 55 (1–2) (2003) 221–249.
[15] S. Keerthi, Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms, IEEE Trans. Neural Networks 13 (5) (2002) 1225–1229.
[16] C.-L. Huang, ACO-based hybrid classification system with feature subset selection and model parameters optimization, Neurocomputing 73 (1–3) (2009) 438–448.
[17] A. Ilhan, K. Mehmet, A. Erhan, A multi-objective artificial immune algorithm for parameter optimization in support vector machine, Appl. Soft Comput. 11 (1) (2011) 120–129.
[18] C.-L. Huang, C.-J. Wang, A GA-based feature selection and parameters optimization for support vector machines, Expert Syst. Appl. 31 (2) (2006) 231–240.
[19] C.-L. Huang, J.-F. Dun, A distributed PSO-SVM hybrid system with feature selection and parameter optimization, Appl. Soft Comput. 8 (4) (2008) 1381–1391.
[20] S.-W. Lin, K.-C. Ying, S.-C. Chen, Z.-J. Lee, Particle swarm optimization for parameter determination and feature selection of support vector machines, Expert Syst. Appl. 35 (2008) 1817–1824.
[21] S.-W. Lin, K.-C. Ying, S.-C. Chen, Z.-J. Lee, Particle swarm optimization for parameter determination and feature selection of support vector machines, Expert Syst. Appl. 35 (4) (2008) 1817–1824.
[22] X.C. Guo, J.H. Yang, C.G. Wu, C.Y. Wang, Y.C. Liang, A novel LS-SVMs hyper-parameter selection based on particle swarm optimization, Neurocomputing 71 (16–18) (2008) 3211–3215.
[23] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods, Cambridge University Press, 2000.
[24] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Disc. 2 (2) (1998) 121–167.
[25] C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Networks 13 (2) (2002) 415–425.
[26] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the Proceedings of the IEEE International Conference on Neural Network, 1995, pp. 1942–1948.
[27] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth international Symposium on Micro Machine and Human Science, Nagoya, 1995, pp. 39–43.
[28] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: IEEE International Conference on Evolutionary Computation, Piscataway, NJ, 1998, pp. 69–73.
[29] A. Ratnaweera, S. Halgamuge, H. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[30] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation, WA D.C., USA, 1999, pp. 1945–1949.
[31] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of IEEE Conference on Systems, Man And Cybernetics, Orlando, FL, USA, 1997, pp. 4104–4108.
[32] Q. Shen, J.-H. Jiang, C.-X. Jiao, G.L. Shen, R.-Q. Yu, Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists, Eur. J. Pharm. Sci. 22 (2–3) (2004) 145–152.
[33] P.K. Tripathi, S. Bandyopadhyay, S.K. Pal, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, Inf. Sci. 177 (22) (2007) 5033–5049.
[34] S.L. Salzberg, On comparing classifiers: pitfalls to avoid and a recommended approach, Data Min. Knowl. Disc. 1 (1997) 317–328.
[35] C.W. Hsu, C.C. Chang, C.J. Lin, A practical guide to support vector classification, Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei. Available at <http://www.csie.ntu.edu.tw/cjlin/libsvm>, 2003.