Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 2739621, 10 pages
http://dx.doi.org/10.1155/2016/2739621

Research Article

A Parallel Genetic Algorithm Based Feature Selection and Parameter Optimization for Support Vector Machine

Zhi Chen, Tao Lin, Ningjiu Tang, and Xin Xia

College of Computer Science, Sichuan University, Chengdu, Sichuan 610065, China

Correspondence should be addressed to Tao Lin; sculintao@gmail.com

Received 27 October 2015; Revised 30 May 2016; Accepted 8 June 2016

Academic Editor: Tomàs Margalef

The extensive applications of support vector machines (SVMs) require efficient method of constructing a SVM classifier with high classification ability. The performance of SVM crucially depends on whether optimal feature subset and parameter of SVM can be efficiently obtained. In this paper, a coarse-grained parallel genetic algorithm (CGPGA) is used to simultaneously optimize the feature subset and parameters for SVM. The distributed topology and migration policy of CGPGA can help find optimal feature subset and parameters for SVM in significantly shorter time, so as to increase the quality of solution found. In addition, a new fitness function, which combines the classification accuracy obtained from bootstrap method, the number of chosen features, and the number of support vectors, is proposed to lead the search of CGPGA to the direction of optimal generalization error. Experiment results on 12 benchmark datasets show that our proposed approach outperforms genetic algorithm (GA) based method and grid search method in terms of classification accuracy, number of chosen features, number of support vectors, and running time.

1. Introduction

The overwhelming amount of data that is currently available in any field provides great opportunities for researchers to obtain knowledge that is impossible to obtain before. However, the enormous amount of data also requires the ability of efficiently extracting the essential knowledge from existing data and generalizing the obtained knowledge to the future unseen new data. Support vector machines (SVMs), proposed by Vapnik [1], have become the references for many classification problems because of their flexibility, computational efficiency, and capability of handling high dimensional data. Despite all the promising results that SVMs provided, it is still a challenge to efficiently construct a SVM classifier which can provide accurate prediction on the unseen new samples. This so-called generalization ability crucially depends on two tasks, namely, feature selection and parameter optimization [2–4].

Feature selection is used to identify a subset of available features which is most essential for classification. Feature selection is important for a variety of reasons, including generalization performance, computational efficiency, feature interpretability, and learning convergence [5–7]. Classification problems typically involve a number of features. However, not all of these features are equally important for a specific task. By extracting the essential information from a given dataset while using the smallest number of features, one can save significant computation time and build classifiers that have better generalization ability.

Along with feature selection, parameter optimization is another key factor that affects the generalization ability of SVMs. Proper parameter setting can not only improve the classification ability of a learned SVM model, but also lead to an efficient classification on the unseen new samples. The parameters that need to be optimized include the error penalty parameter C and the kernel function parameter, such as parameter γ for the Gaussian kernel function. The performance of a SVM largely depends on the choice of parameter. Thus, the selection of parameter is an important research topic in the study of SVMs [8–13].

Both feature selection result and parameter setting have significant impact on the accuracy and efficiency of SVMs. Besides, the choice of feature selection and the setting

of parameter are influenced by each other, and independently performing these two tasks might result in a loss of classification ability [2, 4]. Motivated by these views, the trend in recent years is to turn these two tasks into a multiobjective optimization problem so that global search algorithms, such as genetic algorithm (GA) [2, 14, 15], particle swarm optimization (PSO) [3], and ant colony optimization (ACO) [4], can be used to jointly perform these two tasks. However, jointly performing these two tasks results in a largely expanded solution space, and it requires strong search ability to find optimal feature subset and parameter for SVMs. Besides, given the fact that training SVM even only once needs a great deal of computations, it will be computationally infeasible to apply these global search algorithms into practical use, when the number of training samples increases.

The aim of this paper is to present an efficient and effective method of constructing SVM classifier, so that SVMs can be applied into wider range of practical use and provide promising results. In this paper, a coarse-grained parallel genetic algorithm (CGPGA) is used to jointly select feature subset and optimize parameters for SVMs. The key idea of CGPGA is to divide the whole GA population into several separate subpopulations, and each subpopulation can search the whole solution space in parallel way. After every certain number of generations, best individual of each subpopulation will migrate to other subpopulations. The distributed topology and the migration policy can significantly accelerate the process of feature selection and parameter optimization, so as to increase classification accuracy of SVM.

Another key issue addressed in this paper is the design of a proper fitness function which can be used to assess the true generalization ability of learned SVM and direct the search of CGPGA to the direction of optimal generalization error. An essential part in model selection process (i.e., choosing one classifier over another) is to evaluate the performance of classifiers and choose the best one. However, the classifier derived from the training data is often overoptimistic, due to overspecialization of the learning algorithm to the data [16]. In this paper, a new fitness function, which combines classification accuracy obtained from k-fold bootstrap method, the number of chosen features, and the number of support vectors, is proposed to measure the generalization ability of learned SVM. Experiments on 12 benchmark datasets show that our proposed method not only achieves higher classification accuracy, smaller feature subset, and smaller number of support vectors, but also takes significantly shorter processing time.

The remainder of this paper is organized as follows. A brief introduction to the SVM is given in Section 2. Section 3 introduces basic concept of parallel genetic algorithms. Section 4 gives a detailed description of our proposed approach. The results of our evaluation are given in Section 5. Section 6 concludes this paper.

2. Support Vector Machines

2.1. Linear SVM. First, we briefly describe the SVM formulation. SVM is designed for binary-classification problems. Given the training data (x_i, y_i), $i = 1, \ldots, l$, $x_i \in R^n$ and $y_i \in \{+1, -1\}$, where R^n is the input space, x_i is the sample vector, and y_i is the class label of x_i. A hyperplane in the feature space can be described as $w^T x + b = 0$, where w is normal to the hyperplane and b is a scalar. The distance $D(i)$ from a point x_i in the feature space to the hyperplane is

$$D(i) = \frac{w^T x_i + b}{\|w\|}, \quad i = 1, \ldots, l. \tag{1}$$

When the training samples are linearly separable, the SVM finds an optimal separating hyperplane that maximizes the minimum value of $D(i)$, by solving the following optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i\left(w^T x_i + b\right) \geq 1, \quad i = 1, \ldots, l. \end{aligned} \tag{2}$$

For linearly nonseparable case, there is no such a hyperplane that is able to classify every training sample correctly. In order to relax the separable case to nonseparable one, the slack variable ξ_i is introduced into the optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \end{aligned} \tag{3}$$

where parameter C is the tuning parameter used to balance the margin and the training error. Optimization problem (3) can be solved by introducing the Lagrange multipliers α_i that transform it to dual form:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j x_i x_j \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l \\ & \sum_{i=1}^{l}\alpha_i y_i = 0. \end{aligned} \tag{4}$$

In the classification phase, a sample x in the feature space is assigned a label y according to the following equation:

$$y = \operatorname{sign}\left(\sum_{x_i:\mathrm{SV}}\alpha_i y_i x_i x + b\right). \tag{5}$$

2.2. Kernel. When linear SVM cannot provide satisfactory performance, nonlinear SVM is suggested. The basic idea is to map x_i by a nonlinearly mapping function $f(x_i)$ to a higher dimensional feature space, in which the data are sparse and possibly more separable. Based on the observation that only the inner product of two vectors is needed in (4) and (5), the mapping is often not explicitly given. Instead, a kernel function $K(x_i, x_j) = f(x_i) \cdot f(x_j)$ is incorporated to simplify the computation of the inner product value. The kernel function $K(x_i, x_j) = f(x_i) \cdot f(x_j)$ gives the inner

product value of x_i and x_j in the feature space. Choosing a kernel function is therefore choosing a feature space and the decision function (5) becomes

$$y = \operatorname{sign}\left(\sum_{x_i:\mathrm{SV}} \alpha_i y_i K(x_i, x) + b\right). \quad (6)$$

Among a variety of kernel functions available, the generally used kernel functions include

$$\begin{aligned} &\text{Linear kernel: } K(x_i, x_j) = x_i \cdot x_j, \\ &\text{Polynomial kernel: } K(x_i, x_j) = \left(1 + x_i \cdot x_j\right)^d, \\ &\text{Gaussian kernel: } K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right). \end{aligned} \quad (7)$$

3. Parallel Genetic Algorithms

GAs are stochastic search algorithm based on principles of natural selection and recombination. They attempt to find optimal solution to the problem at hand by manipulating a population of candidate solutions. The population is evaluated and the best solutions are selected to reproduce and mate to form the next generation. After a number of generations, good traits dominate the population, resulting in an increase in the quality of the solutions. In most cases, GAs are efficient enough to find acceptable solutions. However, while being applied to more complex problems, they suffer the risk of premature convergence to local optima [17] and large increase in the time required to find adequate solutions.

There have been multiple efforts [18, 19] to make GAs faster, and one of the most promising choices is to use parallel implementations. The basic idea behind most PGAs is to divide the whole population into several subpopulations and evolve all the subpopulations simultaneously using multiple processors. A PGA basically consists of various GAs, and each processes a part of population or independent subpopulation, with or without communication between them. Therefore, PGAs can increase the diversity of population and significantly reduce computation time.

There are three main types of PGAs: (1) master-slave type, (2) fine-grained type, and (3) coarse-grained type [18]. A master-slave PGA acts like GA and does not affect the behavior of the algorithm. This model uses a single global population and the fitness evaluation is distributed among available processors or cores. Since, in this type of PGAs, selection and crossover consider the entire population, it is also known as global PGA. As for the fine-grained algorithm, the population is divided into a large number of very small subpopulations, which are maintained by different processors. In ideal case, each processor will be allocated only one individual. This method is rarely utilized; due to that it strictly requires too many processors and high communication cost for each generation.

The coarse-grained type is also known as distributed GA or island model, which divides the whole population into a few large subpopulations. Genetic operators are carried out within the subpopulation. After several generations, individuals from different subpopulations will be exchanged and form the new subpopulations for further evolution. The exchange process is named as "migration," which is the essential part inside the CGPGA that could diversify the population and prevent the premature convergence. The schematic of CGPGA is given in Figure 1.

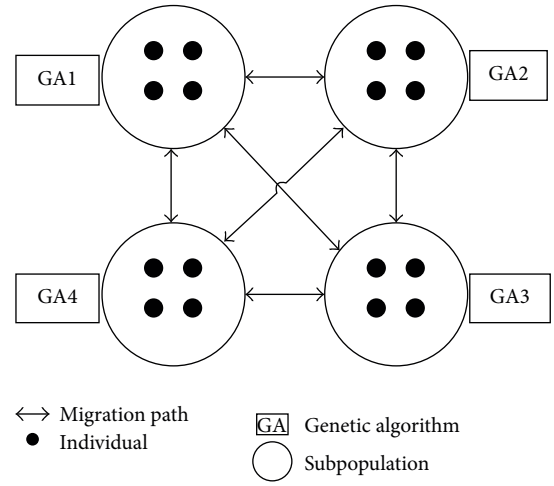

Figure 1: The schematic of coarse-grained type PGA.

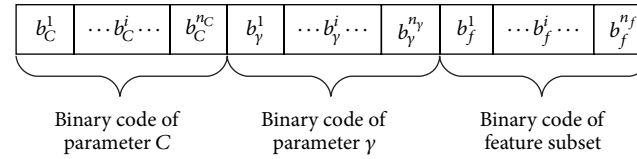

Figure 2: The chromosome comprises three parts, parameters C and γ and feature subset.

In this paper, CGPGA is applied to simultaneously select feature subset and optimize parameters of SVM. The whole population is divided into several subpopulations. Each of these subpopulations will take an independent evolution, and different evolutionary strategies will be applied on them. After every certain number of generations, the best individual will migrate to every other subpopulation and replace the worst one.

4. Method

The chromosome design, fitness function, and system architecture of the proposed CGPGA are described as follows.

4.1. Chromosome Design. The design of chromosome is an important step for the proposed CGPGA method. In this step, Gaussian kernel is chosen as kernel function of SVM classifier. Each chromosome comprises three parts, parameters C and γ and feature subset. Binary code is used to present the chromosome. Figure 2 shows the binary chromosome representation of our design.

In Figure 2, $b_C^1 \sim b_C^{n_C}$ represents the binary code of parameter C, $b_\gamma^1 \sim b_\gamma^{n_\gamma}$ represents the binary code of

parameter γ, n_C is the number of bits representing parameter C, and n_γ is the number of bits representing parameter γ. Note that the selection of n_C and n_γ is according to the computational precision. Besides, the binary code representing the genotype of the parameters (C, γ) should be transformed into phenotype by

$$p = \min_p + \frac{\max_p - \min_p}{2^n - 1} \times d, \tag{8}$$

where p is phenotype (true value) of bit string, $\min_p$ is minimum value of the parameter, $\max_p$ is maximum value of the parameter, d is decimal value of bit string, and n is length of bit string.

In the coding of feature subset, "1" indicates that the feature is selected and "0" indicates that the feature is not selected. n_f represents the number of features in the original dataset.

4.2. Fitness Function. Fitness function is an essential part of CGPGA. It evaluates the performance of each individual in the population and predicts which one has the best generalization ability. The classification accuracy obtained from k-fold bootstrap method, the number of selected features, and the number of support vectors in a SVM model are used to construct a fitness function. All these measurements have been proven to be good indicators of good generalization ability [5, 20, 21]. A high fitness value will be assigned to the individual with high classification accuracy, small number of chosen features, and small number of support vectors. The fitness function is

$$\begin{aligned} \text{fitness} &= W_A \times \text{Accuracy} + W_F f + W_V v, \\ f &= 1 - \frac{\left(\sum_{i=1}^{n_f} F_i\right)}{n_f}, \\ v &= 1 - \frac{\left(\sum_{i=1}^{l} V_i\right)}{l}, \end{aligned} \tag{9}$$

where W_A is classification accuracy weight, Accuracy is average prediction accuracy of 5-time bootstrap, W_F is weight of feature score, f is score of chosen feature subset, and for F_i "1" indicates that ith feature is selected. "0" indicates that ith feature is not selected. n_f is the number of original features, W_V is weight of support vector value and is set to 0.05 in our experiment, v is score of support vectors number, for V_i "1" indicates that ith sample is a support vector, and l is the number of samples in training set.

4.3. System Architectures of the Proposed CGPGA-SVM. System architecture of CGPGA-based feature selection and parameters optimization method is shown in Figure 3. Main steps are described as follows.

(1) Input Dataset. Input dataset includes all the labeled samples. It will be randomly split into a training set and a testing set using bootstrap method. Training set is used to construct the SVM model while testing set is used to test the generalization ability of learned SVM.

(2) Preprocess the Data. Data preprocess is important for a variety of reasons. It can avoid attributes in greater numeric ranges dominating those in smaller numeric ranges and increase SVM accuracy [2]. Each feature of the dataset can be linearly scaled to the range [0, 1] by

$$v' = \frac{v - \min}{\max - \min}, \tag{10}$$

where v is the original value, v' is the scaled value, max is upper bound of the feature value, and min is the low bound.

(3) Initialize the Population. Generally, the original population is randomly generated. However, in our experience, it is useful to randomly generate the genotype of parameters C and γ but select all the features. This will make the first generation of CGPGA run like a grid search procedure.

(4) Decide the Topology and Migrating Strategy. The important characteristics of CGPGA are the use of a few relatively large subpopulations and migration. By dividing the whole population into several separate subpopulations, one can apply different searching strategies (i.e., different crossover rates and mutation rates) to different subpopulations.

Specifically, the whole population will be divided into 2 subpopulations, and each subpopulation has 60 individuals. After every 10 generations, the best individual in each subpopulation will be sent to other subpopulations and replace the worst individual. The purpose of this constant communication is to ensure a good mixing of individuals.

(5) Apply Genetic Operation. Genetic operations, such as selection, crossover, and mutation, will be applied to generate better solutions. However, in a CGPGA, genetic operations will be carried out within the subpopulation, which means different subpopulations will take different crossover rates and mutation rates.

(6) Get the Parameters. This step refers to converting each parameter from its genotype into phenotype. The converting of parameters can be done by (8).

(7) Select Feature Subset. According to the binary code of feature set in each chromosome, related features can be chosen and unrelated features will be discarded. After training dataset and testing dataset discard unrelated features, they can be used to construct the SVM model and test the generalization ability of learned SVM.

(8) Evaluate the Individuals Using Bootstrap. Each individual in the population refers to a certain pair of parameters (C, γ) and a certain choice of feature selection. To obtain a reliable performance estimate of this individual, bootstrap method will be used k times. During each phase of bootstrap, 50% of the samples in the input dataset will be randomly chosen as the training set, while the rest of samples will be chosen as the testing set. Training set is used to construct the SVM model

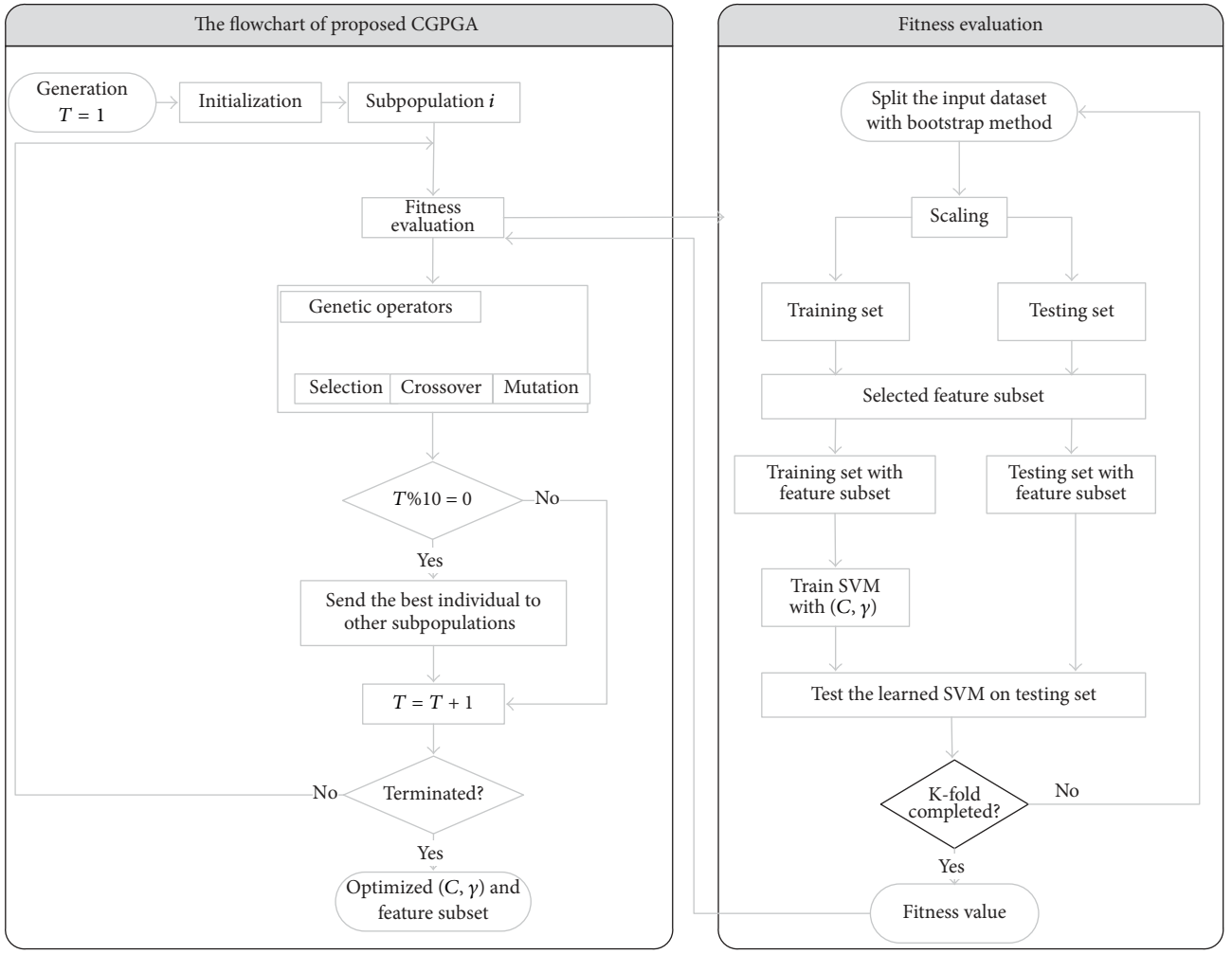

Figure 3: System architecture of the proposed method.

with the chosen parameters and feature subset, and testing set is used to predict the classification ability of learned SVM. The k classification accuracies from the k-time bootstrap then can be averaged to produce a single classification accuracy. After obtaining the classification accuracy, the fitness value can be calculated by (9). It must be mentioned that the evaluation of an individual is independent from the rest of the population, and there is no need to communicate during this phase. Thus, the evaluation of individuals is parallelized by assigning a fraction of the population to each of the available processors.

(9) Termination Criteria. In our approach, if the generation number reaches generation 100 or the highest fitness value of the whole population does not improve during the last 30 generations, process will stop.

5. Experiments

5.1. Experiment Settings. The used platform is Intel Core CPU i7-3770 (3.4 GHz, 4 cores), 4 G RAM, Windows 7 operating system. The development environment is MATLAB (2012a). The software of SVM is Libsvm (3.1) [22]. In our experiment, the search range of parameter C is set to [0.01, 2048], while the search range of parameter γ is set to [0.0001, 10]. The whole population will be divided into 2 subpopulations, and each subpopulation has 60 individuals. After every 10 generations, the best individual in each subpopulation will be sent to other subpopulations and replace the worst individual. The crossover rate and mutation rate of first subpopulation are 0.8 and 0.05. The crossover rate and mutation rate of second subpopulation are 0.7 and 0.02.

To evaluate the classification ability of the proposed approach in different classification tasks, 12 real world datasets from the UCI database [23] have been adopted. Their number of classes, number of samples, and number of original features are shown in Table 1.

In order to show the effectiveness of our proposed method, we conduct several comparisons between our proposed method and two other methods, including grid search method and GA-SVM [2]. Specifically, grid search is a widely used method of parameter optimization. In most cases, grid search can get a satisfactory result. GA-SVM, proposed by Huang and Wang [2], is the most widely used feature selection and parameter optimization method for SVMs. It

Table 1: Datasets from UCI repository.

ID	Name	Number of classes	Number of instances	Number of features
1	Ionosphere	2	351	34
2	Breast cancer	2	683	10
3	Australia	2	690	14
4	Diabetes	2	768	8
5	Vehicle	4	846	18
6	Vowel	11	990	13
7	Car	4	1,728	6
8	Splice	2	3,175	60
9	DNA	3	3,186	180
10	WaveForm	3	5,000	40
11	Svmguide1	2	7,089	4
12	Mushrooms	2	8,124	112

can deal with feature selection and parameter optimization simultaneously and provide promising results.

To guarantee the result obtained by proposed method is valid, we adopt k-fold cross validation. The dataset will be partitioned into k independent subsets randomly, and the size of each subset is approximately equal. The k-fold cross validation process is then repeated k times, with each of k subsets being used exactly once as the testing dataset, while the remaining $(k - 1)$ datasets are used as a training set. The training set will be used as the input dataset of our proposed approach, and the performance of obtained parameters and feature subset will be tested on the testing set. The k results from the folds then can be averaged to produce a single estimation. In our experiment, k is set to 10. The evaluation procedure of our proposed approach using a 10-fold cross validation is shown in Figure 4. Take the Australian dataset as an example; the best pairs of parameters (C, γ), the classification accuracy, the number of chosen features, and the number of support vectors for each fold obtained by our proposed approach and grid search method are shown in Table 4.

5.2. Classification Results. Table 2 gives the comparison of our proposed approach CGPGA-SVM and GA-SVM [2]. Tenfold cross validation is used to estimate the classification accuracy of each approach. The obtained classification accuracy is illustrated with the form of "average ± standard deviation." As shown in Table 2, our proposed approach achieves higher classification accuracies on 11 datasets, except on "Svmguide1."

Table 3 gives the experiment results of our proposed approach and grid search. Tenfold cross validation is used to estimate the classification accuracy of each approach. The obtained classification accuracy is illustrated with the form of "average ± standard deviation." As we can see, the proposed approach produces smaller feature number, and grid search uses all the original features. Besides, the proposed approach achieves higher classification accuracy. To validate if this higher classification accuracy actually indicates stronger classification ability, we used nonparametric Wilcoxon-signed-rank test for all the 12 datasets. As shown in Table 4, the p values for Splice, Svmguide1, and Mushrooms are larger than the prescribed statistical significance level of 0.05, but all other p values are smaller than the significance level of 0.05. Generally, compared with the grid search, the proposed approach shows higher classification ability and produces smaller feature number.

Table 2: Comparisons between CGPGA-SVM and GA-SVM.

Dataset	CGPGA-SVM	GA-SVM
Ionosphere	**98.85 ± 2.01**	98.03 ± 2.64
Breast cancer	**98.53 ± 1.16**	98.39 ± 1.75
Australia	**90.13 ± 2.39**	87.10 ± 2.64
Diabetes	**81.76 ± 3.37**	79.04 ± 2.44
Vehicle	**86.05 ± 3.54**	83.69 ± 2.64
Vowel	**99.29 ± 0.68**	98.58 ± 1.44
Car	**99.83 ± 0.39**	99.36 ± 0.51
Splice	**92.66 ± 1.43**	89.07 ± 1.81
DNA	**96.79 ± 1.31**	95.83 ± 1.01
WaveForm	**87.81 ± 1.60**	82.60 ± 2.56
Svmguide1	96.66 ± 0.76	**96.69 ± 0.98**
Mushrooms	**100.0 ± 0.00**	99.96 ± 0.06

5.3. Number of Support Vectors. The time taken for a SVM to compute the class label of a new pattern is proportional to the number of support vectors. A large number of support vectors indicate a slow classification on the new pattern. A small number of support vectors can extend the application of SVM to a wider field where classification has to be done in great speed. We compared the number of support vectors produced by CGPGA-SVM, GA-SVM, and grid search method. Since 10-fold cross validation was proposed in our experiment, 10 SVM models will be constructed for each of the datasets. We averaged the number of support vectors in 10 SVM models and compared them in Figure 5. Two conclusions can be drawn from Figure 5. (1) In most cases, our proposed method is capable of producing SVM with less support vectors than GA-SVM and grid search method (except on dataset of Svmguide1). (2) The number of support vectors does not scale

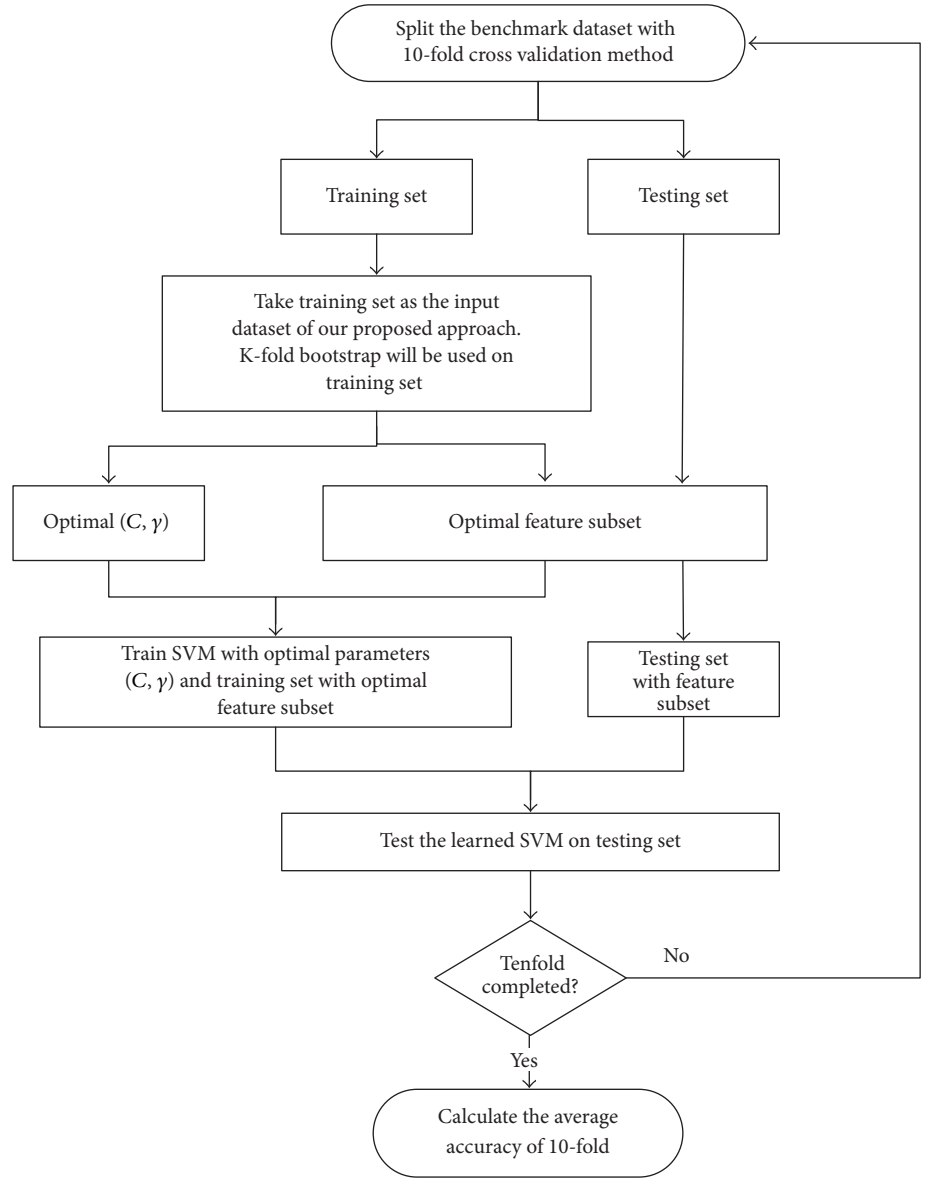

Figure 4: The procedure of experiment on the benchmark dataset using our proposed approach.

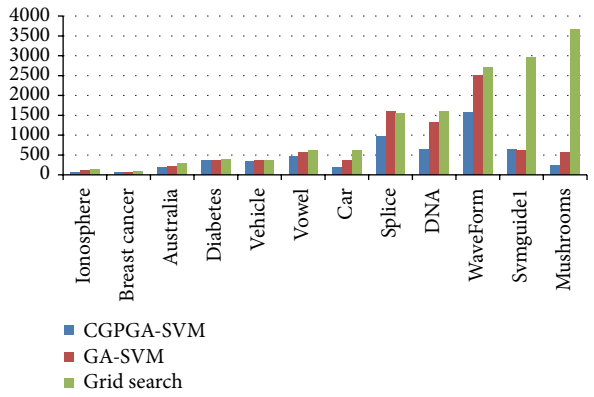

Figure 5: Average number of support vectors obtained from 10-fold cross validation.

with the number of training samples. In fact, the proportion of support vectors in model produced by CGPGA-SVM was maintained at a low level. This could enable the practitioners to apply the SVM to wider fields where classification has to be done in great speed, for example, online applications.

5.4. Computational Efficiency. A serious limitation of global search methods is that they involve high computational complexity. By using parallelization strategy, our proposed approach can significantly reduce the running time, while getting an enough adequate solution. We run our proposed approach and GA-SVM on all 6 datasets and recorded the average time involved in one generation of evolution. To get a fair enough comparison, both our approach and GA-SVM have 120 individuals in their populations. Table 5 gives the

Table 3: Comparisons between CGPGA-SVM and grid search.

Dataset	CGPGA-SVM		Grid search		p values
	Tenfold cross validation accuracy (%)	Number of selected features	Tenfold cross validation accuracy (%)	Number of selected features	
Ionosphere	**98.85 ± 2.01**	13.8 ± 3.46	93.16 ± 3.00	34.0 ± 0.00	0.005*
Breast cancer	**98.53 ± 1.16**	2.60 ± 0.84	96.77 ± 1.81	10.0 ± 0.00	0.039*
Australia	**90.13 ± 2.39**	5.10 ± 1.60	85.51 ± 4.46	14.0 ± 0.00	0.011*
Diabetes	**81.76 ± 3.37**	3.90 ± 0.99	76.44 ± 3.74	8.00 ± 0.00	0.014*
Vehicle	**86.05 ± 3.54**	10.3 ± 1.34	78.60 ± 3.29	18.0 ± 0.00	0.005*
Vowel	**99.29 ± 0.68**	6.70 ± 0.48	98.88 ± 1.00	13.0 ± 0.00	0.037*
Car	**99.83 ± 0.39**	6.00 ± 0.00	99.48 ± 0.51	6.00 ± 0.00	0.046*
Splice	**92.66 ± 1.43**	26.7 ± 3.53	92.03 ± 1.52	60.0 ± 0.00	0.386
DNA	**96.79 ± 1.31**	87.1 ± 4.53	96.05 ± 1.09	180. ± 0.00	0.005*
WaveForm	**87.81 ± 1.60**	21.1 ± 2.69	86.68 ± 1.47	40.0 ± 0.00	0.037*
Svmguide1	**96.66 ± 0.76**	2.40 ± 0.52	96.32 ± 0.71	4.00 ± 0.00	0.594
Mushrooms	**100.0 ± 0.00**	42.5 ± 3.54	99.98 ± 0.04	112. ± 0.00	0.317

∗ indicates significance at 0.005 level.

Table 4: Experiment results for Australian dataset using our proposed approach and grid search.

Fold number	Our proposed approach					Grid search method				
	C	γ	Number of features	Number of support vectors	Accuracy (%)	C	γ	Number of features	Number of support vectors	Accuracy
1	368.08299	1.48664	5	173	85.5072	32	0.125	14	202	82.6087
2	827.07180	1.65084	6	206	91.3043	0.5	2.0	14	324	89.8551
3	529.60439	0.32159	7	203	89.8551	0.5	1.0	14	261	86.7647
4	1845.7247	0.95494	3	198	94.2857	0.125	0.03125	14	499	81.1594
5	115.07262	4.63776	6	215	92.8571	512	0.0625	14	193	86.9565
6	347.31311	0.29031	7	197	89.8551	1024	0.03125	14	197	89.8551
7	379.01391	6.52218	3	214	89.7059	0.5	2.0	14	319	88.2353
8	131.80564	4.06696	5	204	90.0000	128	0.125	14	189	76.8116
9	584.01293	5.78718	3	229	88.2353	32	0.125	14	205	82.8571
10	74.464588	3.41797	6	188	89.7059	0.125	0.015625	14	504	90.0000
Avg			5.1	202.7	90.1312			14	289.3	85.5104

results. As we can see, on a common used 4-core CPU, our approach takes significant shorter time than GA-SVM.

6. Limitations and Conclusions

The overwhelming amount of data that is currently available in any field poses new challenges for machine learning techniques. To extract essential knowledge from these enormous data and generalize obtained knowledge to the future unseen new data, two problems must be efficiently addressed for SVM, namely, feature selection and parameter optimization. The number of input features in a classifier should be limited without losing its predictive power. With a smaller feature set, the classification decision is more easily explained and can be made in shorter time. Parameter optimization is another important factor that affects the generalization ability of SVMs. With a proper setting of parameters, the classification accuracy on the unseen new patterns can be ensured.

This work investigated a hybrid CGPGA-based model that hybridized the coarse-grained parallel genetic algorithm and support vector machines to maintain the classification accuracy with a small and suitable feature subset. The distributed topology and migration policy of CGPGA enable us to search the solution space with different search strategies in parallel way, thereby providing strong search ability and high efficiency.

Experiment results obtained from several real world datasets of UCI database showed promising performance in terms of 10-fold accuracy, the size of selected feature subset, the number of support vectors, and training time. They revealed that the proposed approach not only optimized SVMs' model parameters, but also correctly obtained the discriminating feature subset in an efficient way. Moreover,

Table 5: Comparisons of computational efficiency.

Dataset	Proposed approach(s)	GA-SVM (s)	Speedup
Ionosphere	**1.52**	6.54	4.30×
Breast cancer	**4.51**	18.61	4.13×
Australian	**15.98**	70.68	4.42×
Diabetes	**25.90**	97.43	3.76×
Vehicle	**12.50**	57.41	4.59×
Vowel	**12.40**	52.37	4.22×
Car	**64.39**	256.26	3.98×
Splice	**141.35**	580.97	4.11×
DNA	**253.59**	1108.20	4.37×
WaveForm	**202.05**	846.59	4.19×
Svmguide1	**480.78**	2077.00	4.32×
Mushrooms	**244.74**	996.11	4.07×

the proportion of support vectors in model produced by our method was maintained at a low level. This could result in faster classification on the unseen new pattern and extend the applications of SVM to wider fields where classification has to be done in great speed.

Despite all of the promising results, our proposed work also has its limitations. Training SVM is a computation-intensive task. Our proposed work does not reduce the number of SVM models constructed during the optimization procedure or the time needed for training one SVM. The acceleration is achieved by assigning multiple SVM learning processes to each of the available processors, so that the SVM learning processes can be done in parallel way and the computational resources can be used in cost-efficient way. When the problem becomes large enough, our proposed work may show its burden.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The research is partly supported by Science and Technology Supporting Program, Sichuan Province, China (nos. 2013GZX0138 and 2014GZ0154), and Scientific Research Foundation for Young Teachers, Sichuan University (no. 2015SCU11050).

References

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[2] C.-L. Huang and C.-J. Wang, "A GA-based feature selection and parameters optimizationfor support vector machines," *Expert Systems with Applications*, vol. 31, no. 2, pp. 231–240, 2006.

[3] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817–1824, 2008.

[4] C.-L. Huang, "ACO-based hybrid classification system with feature subset selection and model parameters optimization," *Neurocomputing*, vol. 73, no. 1–3, pp. 438–448, 2009.

[5] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[6] K. Z. Mao, "Feature subset selection for support vector machines through discriminative function pruning analysis," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 60–67, 2004.

[7] Y. Liu and Y. F. Zheng, "FS_SFS: a novel feature selection method for support vector machines," *Pattern Recognition*, vol. 39, no. 7, pp. 1333–1345, 2006.

[8] S. S. Keerthi, "Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1225–1229, 2002.

[9] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1–3, pp. 131–159, 2002.

[10] R. Debnath and H. Takahashi, "An efficient method for tuning kernel parameter of the support vector machine," in *Proceedings of the IEEE International Symposium on Communications and Information Technologies: smart Info-Media Systems (ISCIT '04)*, pp. 1023–1028, October 2004.

[11] K.-P. Wu and S.-D. Wang, "Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space," *Pattern Recognition*, vol. 42, no. 5, pp. 710–717, 2009.

[12] J. Sun, C. Zheng, X. Li, and Y. Zhou, "Analysis of the distance between two classes for tuning SVM hyperparameters," *IEEE Transactions on Neural Networks*, vol. 21, no. 2, pp. 305–318, 2010.

[13] M. Lázaro-Gredilla, V. Gómez-Verdejo, and E. Parrado-Hernández, "Low-cost model selection for SVMs using local features," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 6, pp. 1203–1211, 2012.

[14] Z. Wang, Y.-H. Shao, and T.-R. Wu, "A GA-based model selection for smooth twin parametric-margin support vector machine," *Pattern Recognition*, vol. 46, no. 8, pp. 2267–2277, 2013.

[15] M. Zhao, C. Fu, L. Ji, K. Tang, and M. Zhou, "Feature selection and parameter optimization for support vector machines: a new approach based on genetic algorithm with feature chromosomes," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5197–5204, 2011.

[16] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, Boston, Mass, USA, 2006.

[17] M. Rocha and J. Neves, "Preventing premature convergence to local optima in genetic algorithms via random offspring generation," in *Multiple Approaches to Intelligent Systems*, pp. 127–136, Springer, Berlin, Germany, 1999.

[18] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[19] H. Mühlenbein, "Parallel genetic algorithms, population genetics and combinatorial optimization," in *Parallelism, Learning, Evolution*, J. D. Becker, I. Eisele, and F. W. Mündemann, Eds., vol. 565 of *Lecture Notes in Computer Science*, pp. 398–406, Springer, Berlin, Germany, 1991.

[20] Y. Zhan and D. Shen, "Design efficient support vector machine for fast classification," *Pattern Recognition*, vol. 38, no. 1, pp. 157–161, 2005.

[21] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics Surveys*, vol. 4, pp. 40–79, 2010.

[22] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.

[23] A. Asuncion and D. Newman, *UCI Machine Learning Repository*, 2007.