ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT

Pattern Recognition III (IIII) III–III

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

FS_SFS: A novel feature selection method for support vector machines

Yi Liu, Yuan F. Zheng*

Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, USA

Received 8 October 2004; received in revised form 3 October 2005; accepted 3 October 2005

Abstract

In many pattern recognition applications, high-dimensional feature vectors impose a high computational cost as well as the risk of "overfitting". *Feature Selection* addresses the dimensionality reduction problem by determining a subset of available features which is most essential for classification. This paper presents a novel feature selection method named *filtered and supported sequential forward search* (FS_SFS) in the context of support vector machines (SVM). In comparison with conventional *wrapper* methods that employ the SFS strategy, FS_SFS has two important properties to reduce the time of computation. First, it dynamically maintains a subset of samples for the training of SVM. Because not all the available samples participate in the training process, the computational cost to obtain a single SVM classifier is decreased. Secondly, a new criterion, which takes into consideration both the discriminant ability of individual features and the correlation between them, is proposed to effectively filter out nonessential features. As a result, the total number of training is significantly reduced and the overfitting problem is alleviated. The proposed approach is tested on both synthetic and real data to demonstrate its effectiveness and efficiency.
© 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Feature selection; Sequential forward search (SFS); Support vector machines (SVM)

1. Introduction

Reduction of feature dimensionality is of considerable importance in machine learning. The reason for being so is twofold: to reduce the computational complexity and to improve the classifier's generalization ability. The first motivation is quite evident, since fewer features require less run time to train and to apply the classifier. The second motivation is low-dimensional representation reducing the risk of "overfitting". As a rule of thumb, a minimum of $10 \cdot d \cdot C$ training samples is required for a d-dimensional classification problem of C classes [1]. When it is impractical and even impossible to obtain the required number of training samples, the reduction of feature dimensionality helps decrease the size of the training samples and consequently improves the generalization performance of the classification algorithm.

Feature extraction and *feature selection* are two different approaches for the reduction of dimensionality. Feature extraction involves linear or nonlinear transformation from the original feature space to a new one of lower dimensionality. Although it does reduce the dimensionality of the vectors fed to the classifier, the number of features that must be measured remains the same. Feature selection, on the other hand, directly reduces the number of original features by selecting a subset of them that still retains sufficient information for classification. Feature selection techniques have been applied successfully in many applications, such as automated text categorization [2] and data visualization [3]. In general, feature selection approaches can be grouped into two categories: filter methods and wrapper methods [4]. Acquiring no feedback from classifiers, the filter methods estimate the classification performance by some indirect assessments, such as distance measures which reflect how well the classes separate from each other. The wrapper methods, on the contrary, are classifier-dependent. Based on the classification accuracy, the methods evaluate the "goodness" of the selected feature subset directly, which should intuitively yield better

* Corresponding author. Tel.: +1 614 292 8039; fax: +1 614 292 7596.
E-mail addresses: liuyi@ece.osu.edu (Y. Liu), zheng@ece.osu.edu (Y.F. Zheng).

0031-3203/$30.00 © 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2005.10.006

performance. As a matter of fact, many experimental results reported so far are in favor of the wrapper methods [4–6].

In spite of the good performance, the wrapper methods have limited applications due to the high computational complexity involved. This is true especially when the wrapper methods are applied to support vector machines (SVM), a state-of-art classifier that has found success in a variety of areas [7–11]. Given the fact that training SVM even only once needs a great deal of computation when the number of training samples is large, the integration of SVM and wrapper methods, which requires multiple times of SVM training, will be computationally infeasible. Even when some well-known suboptimal search strategies such as the sequential forward search (SFS) are used, the selection process is still quite costly. That calls for feature selection methods designed especially for SVM. Unfortunately there are just a few algorithms in the literature that have been proposed for feature selection in the context of SVM [12–16]. One possibility is to embed feature selection into the optimization process [12,13]. For example, Ref. [12] adds an extra term that penalizes the size of the selected feature subset to the standard cost function of SVM, and optimizes the new objective function to achieve feature selection. A similar idea is also employed in Ref. [13]. The major difference is that Ref. [13] introduces a binary vector whose elements indicate the presence and absence of the corresponding feature component, and then approximates the binary vector with a nonnegative real-valued vector σ so that the optimization can be performed efficiently via gradient descent. Then the features corresponding to the m largest valued elements of σ are selected. The benefit is that the optimization has to be done only once. Unfortunately the two methods evaluate the features on a individual basis and the correlation between them is ignored [15].

In the meantime, many researchers suggest to evaluate the importance of features by measuring the change of the cost function when a feature is removed, or equivalently when its weight is set to zero. In the case of SVM which has a quadratic cost function, the magnitude of weights w_i is the justified feature ranking criterion, and based on this criterion a SVM recursive feature elimination (SVM RFE) method is proposed in Ref. [14]. Unfortunately, this approach is limited to linear kernels. Some researchers propose to utilize the change of the discriminant function rather than the cost function itself for feature ranking [15,16]. Since for most kernels the SVM discriminant function is differentiable with respect to individual features, the algorithms become applicable to nonlinear kernels.

In this paper, we present a more efficient version of the wrapper/SFS method for SVM which is named filtered and supported SFS (FS_SFS). FS_SFS is designed especially for SVM and has the following properties to improve its efficiency over the conventional wrapper/SFS method:

(1) *FS_SFS combines the advantages of the filter and the wrapper methods.* By introducing a filtering process for each SFS iteration, FS_SFS reduces the number of features that has to be tested through the training of SVM. Then the pre-selected features are considered "informative", and are evaluated by the accuracy of classification as in the conventional wrapper method. In this way, we are able to reduce the unnecessary computation time spent on the testing of the "noninformative" features while maintaining the good performance delivered by the wrapper method.
(2) *FS_SFS introduces a new criterion that assesses features in a collective manner.* An effective filtering criterion is needed in FS_SFS since it is undesirable to discard many informative features through the filtering process. To address this problem, we develop a new criterion, which is computationally efficient and considers the discriminant ability of individual features as well as the correlation between them.
(3) *FS_SFS is specially designed for SVM classifier to improve the efficiency of the feature selection process.* During the feature search process, FS_SFS dynamically maintains an active set for training, which is a subset of the original training samples, as the candidates of the support vectors. Whenever the training of SVM is needed, only the samples in the subset are utilized. In this way, the efficiency of training a single SVM classifier is improved.

The rest of the paper is organized as follows. Section 2 gives a brief introduction of SVM and Section 3 explains FS_SFS in detail. Experimental results are given in Section 4 followed by conclusions and discussions in Section 5.

2. Support vector machines

SVM is a state-of-the-art learning machine which has been extensively used as a classification tool and has found a great deal of success in many applications. To facilitate the discussion, we give a very brief review of SVM in this section and refer the details to Refs. [17–19].

Consider N pairs of training samples:

$$\{X(1), Y(1)\}, \{X(2), Y(2)\}, \ldots, \{X(N), Y(N)\},$$

where

$$X(i) = [x_1(i)x_2(i)\ldots x_k(i)]^{\mathrm{T}}$$

is a k-dimensional feature vector representing the ith training sample, and $Y(i) \in \{-1, 1\}$ is the class label of $X(i)$.

A hyperplane in the feature space can be described as the equation $W \cdot X + b = 0$, where $W = [w_1 w_2 \ldots w_k]^{\mathrm{T}}$ and b is a scalar. The signed distance $d(i)$ from a point $X(i)$ in the feature space to the hyperplane is

$$d(i) = \frac{W \cdot X(i) + b}{\|W\|}.$$

When the training samples are linearly separable, SVM yields the optimal hyperplane that separates two classes with no training error, and maximizes the minimum value of $|d(i)|$. It is easy to find that the parameter pair (W, b) corresponding to the optimal hyperplane is the solution to the following optimization problem:

$$\begin{aligned} &\text{Minimize} \quad L(W) = \frac{1}{2}\|W\|^2 \\ &\text{Subject to} \quad Y(i)(W \cdot X(i) + b) \geqslant 1, \quad i = 1, \ldots, N. \end{aligned} \tag{1}$$

For linearly nonseparable cases, there is no such a hyperplane that is able to classify every training point correctly. However the optimization idea can be generalized by introducing the concept of *soft margin*. The new optimization problem thus becomes:

$$\begin{aligned} &\text{Minimize} \quad L(W, \xi) = \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{N}\xi(i) \\ &\text{Subject to} \quad Y(i)(W \cdot X(i) + b) \geqslant 1 - \xi(i), \\ &\qquad i = 1, \ldots N, \end{aligned} \tag{2}$$

where ξ_i are called slack variables which are related to the soft margin, and C is the tuning parameter used to balance the margin and the training error. Both optimization problems (1) and (2) can be solved by introducing the Lagrange multipliers $\alpha(i)$ that transform them to quadratic programming problems.

In the classification phase, a point $\tilde{X}$ in the feature space is assigned a label $\tilde{Y}$ according to the following equation:

$$\begin{aligned} \tilde{Y} &= \text{sgn}[W \cdot \tilde{X} + b] \\ &= \text{sgn}\left[\sum_{i=1}^{N}\alpha(i)Y(i)(X(i) \cdot \tilde{X}) + b\right]. \end{aligned} \tag{3}$$

For the applications where linear SVM dose not produce satisfactory performance, nonlinear SVM is suggested. The basic idea of nonlinear SVM is to map X by a nonlinearly mapping $\Phi(X)$ to a much higher dimensional space, in which the optimal hyperplane is found. Based on the observation that only the inner product of two vectors is needed to solve the quadratic programming problem, one can define the nonlinear mapping implicitly by introducing the so-called kernel function, which computes the inner product of vectors $\Phi(X(i))$ and $\Phi(X(j))$. Among a variety of kernel functions available, the radial basis and the polynomial function are often chosen for many applications:

- Radial basis function (RBF)

$$K(X_1, X_2) = \exp\left(-\frac{\|X_1 - X_2\|}{\sigma^2}\right),$$

where σ is the parameter controlling the width of the kernel.

- Polynomial function

$$K(X_1, X_2) = (X_1 \cdot X_2 + 1)^d,$$

where d is the degree of the polynomial.

Accordingly, the class label of an unseen point $\tilde{X}$ is given by

$$\begin{aligned} \tilde{Y} &= \text{sgn}[W \cdot \tilde{X} + b] \\ &= \text{sgn}\left[\sum_{i=1}^{N}\alpha(i)Y(i)K(X(i), \tilde{X}) + b\right]. \end{aligned} \tag{4}$$

A noteworthy feature of SVM is that it is based on the *structural risk minimization* induction principle, which provides a guaranteed bounded risk value even when the number of the training set is small. Nevertheless, as shown in Ref. [13], SVM may perform poorly when there are many irrelevant features and for such a situation, feature selection is a remedy.

3. FS_SFS: filtered and supported sequential forward search

3.1. Problem statement of feature selection

Consider the binary classification scenario, which has input vectors denoted as $X \in R^k$ and their corresponding class labels denoted as $Y \in \{1, -1\}$. Let

$$F = \{f_1, f_2, \ldots, f_k\} \tag{5}$$

be the set of all features under examination, and let

$$\begin{aligned} S &= \{(X(l), Y(l)) \,|\, l = 1, 2, \ldots, N\} \\ &= \{[x_1(l)\ x_2(l)\ \ldots\ x_k(l)]^{\mathrm{T}}, Y(l) \,|\, l = 1, \ldots, N\} \end{aligned} \tag{6}$$

denotes the training set containing N training pairs, where $x_i(l)$ is the numerical value of feature f_i for the lth training sample.

The goal of feature selection is to find a minimal set of features $F_s = \{f_{s_1}, f_{s_2}, \ldots, f_{s_d}\}$ to represent the input vector X in a lower dimensional feature space as

$$X_s = [x_{s_1}\ x_{s_2}\ \ldots\ x_{s_d}], \tag{7}$$

where $d < k$, while the classifier obtained in the low-dimensional representation still yields the acceptable classification accuracy.

3.2. Review of FS_SFS

As mentioned earlier, feature selection approaches can be categorized into two classes: the filter and the wrapper methods [4], whose pipelines are shown in Fig. 1(a) and (b), respectively. Guided by the feedback from the classifier, the wrapper method selects features in a more consistent way than the filter methods. The better performance

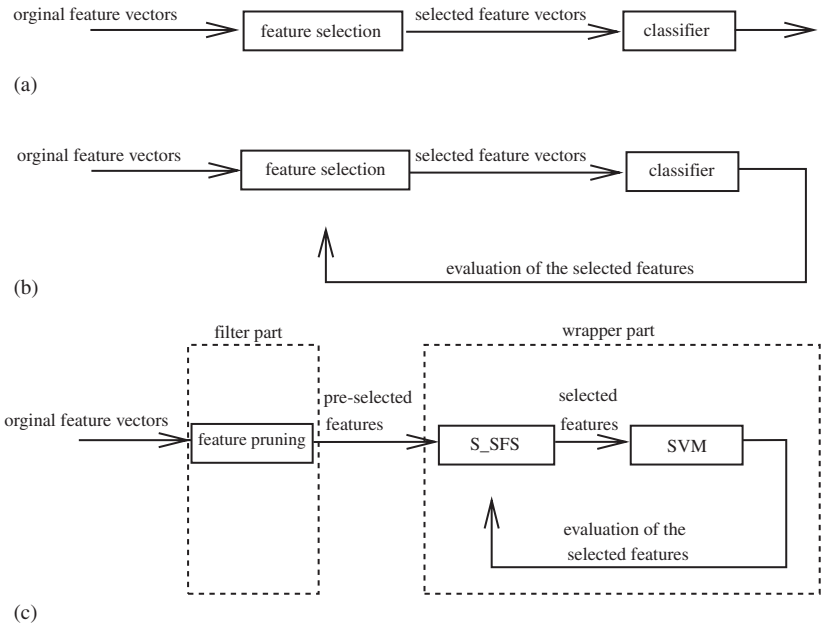


Fig. 1. The comparison among (a) filter methods, (b) wrapper methods, and (c) the proposed methods for feature selection.

of the former is achieved, however, at the cost of much more computation. Fig. 1(c) gives the outline of the proposed method which combines the advantage of both the filter and the wrapper methods. The filtering part, acting in the generic way similar to a filter method, ranks features without involving the classifier. The features with relatively high ranks are considered as "informative" feature candidates and then are re-studied by the wrapper part that further investigates their contributions to a specific classifier. This combinational framework delivers as good a performance as the conventional wrapper method, but is computationally simpler.

With the framework determined, feature selection is reduced to a problem of searching for the optimal subset [20]. Many search strategies have been proposed [21–23], from which we adopt a suboptimal search method called SFS [23] for its simplicity and effectiveness. Starting with an empty set, SFS iteratively selects one feature at a time and adds it to the current feature set. The feature added is the one which gives the smallest value according to a certain criterion comparing to adding the other remaining features. Different approaches have different criteria, such as class separability [24], classification errors [25], change of the boundary [26], and etc. The criterion employed in our algorithm is the objective function of SVM formulated in Eq. (2).

The FS_SFS method will be presented in detail in the following three subsections. The isolated filter and the wrapper parts, which are named Filtered_SFS (F_SFS) and Supported_SFS (S_SFS), respectively, are presented in Sections 3.3 and 3.4. Then they are then integrated as FS_SFS in Section 3.5.

3.3. F_SFS: filtered_SFS using a new criterion

Recall that the goal of the filter part is to discard some "noninformative" features to reduce the computational burden of the wrapper part. To serve this purpose, the filter part needs to meet two major requirements:

- the criterion for filtering must be simple so as not to introduce much computation;
- the process of filtering should lose as few informative features as possible.

Class separability is a classical criterion of filtering available in the literature. It involves calculating the normalized distance between classes and then eliminating the features that yield low separability values. The criterion is computationally simple and thus satisfies the first requirement. However, it has a major drawback, i.e., the criterion implicitly assumes the features to be orthogonal and overlooks the correlation between them. Consequently, those correlated features that individually separate the classes well but collectively provided redundant information might be retained, which violates the second requirement.

Here we propose a new criterion which is able to yield a more compact feature subset. In this new criterion, the correlation between features as well as the class separability of the individual features are taken into consideration. Also it retains the advantage of simple computation.

Suppose we have a feature combination $F_s = \{f_{n_1}, f_{n_2}, \ldots, f_{s_d}\}$, and we can calculate a score for each individual feature f_i. This score, which is denoted as R_{i,F_s}, is the measure of the importance of that particular feature f_i such

that, the higher the score the more important the feature is. The evaluation of the score for a given feature subset F_s takes the following steps:

(1) *Determining the discriminant ability of the feature*: The discriminant ability of feature f_i is described by the class separability as

$$D_i = \frac{|m_1^i - m_2^i|}{std_1^i + std_2^i}, \tag{8}$$

where m_1^i and std_1^i (m_2^i and std_2^i) are the mean and standard deviation of the samples belonging to class 1 (-1), when only feature f_i is considered. It can be seen from Eq. (8) that the further the two classes are separated from each other using f_i, the larger D_i would be, and therefore the better discriminant ability that feature f_i has.

(2) *Determining the correlation between f_i and F_s*: First we define the correlation coefficient $\rho_{i,j}$ between two features f_i and f_j as

$$\rho_{i,j} = \prod_{c=1}^{2} \rho_{i,j}^{(c)} = \prod_{c=1}^{2} \frac{\mathrm{cov}(S_c(f_i), S_c(f_j))}{\sqrt{\mathrm{var}(S_c(f_i)) \cdot \mathrm{var}(S_c(f_i))}}, \tag{9}$$

where $S_c(f_i) = \{x_{f_i}(l) \mid Y(l) = c\}$ is the training vectors that are represented by feature f_i and labeled as class c. Based on $\rho_{i,j}$, we further define the correlation coefficient between a single feature f_i and a feature set F_s as

$$\rho_{i,F_s} = \max_{f_j \in F_s} |\rho_{i,j}|. \tag{10}$$

A high value of ρ_{i,F_s} indicates that f_i is highly correlated with certain feature $f_j \in F_s$, and therefore it carries redundant information.

(3) *Calculating the score of the feature*: It is desirable to select the features that can individually separate the classes well and has small correlation with the features in the subset which has been obtained so far. Thus the final score assigned to f_i is defined as

$$R_{i,F_s} = \frac{D_i}{\max\{D_l\}} - |\rho_{i,F_s}|, \tag{11}$$

where D_i is normalized such that it is in the same range as $|\rho_{i,F_s}|$.

It is worth noting that R_{i,F_s} is dependent not only on f_i but also on F_s. As a result, the score of f_i usually changes during the SFS process.

3.4. S_SFS: supported_SFS in the context of SVM

Supported SFS is basically a variation of the SFS algorithm that is specially tailored to SVM to expedite the feature searching process. Recall that in SVM, there is a special group of training samples named "support vectors", whose corresponding coefficients $\alpha(i)$ in Eq. (3) are nonzeros. In other words, training samples other than support vectors have no contribution to determine the decision boundary. Since the number of support vectors is relatively small, we could train SVM just by using the support vectors. Following this idea, we propose the supported SFS algorithm, which dynamically maintains an *active subset* as the candidates of the support vectors, and trains SVM using this reduced subset rather than the entire training set. In this way, we are able to find the boundary with less computational cost.

The procedure of S_SFS is described as follows. The first step is to choose the best single features among the k possible choices. To do so, we train SVM k times, each of which uses all the training samples available but with only one feature f_i. Mathematically the initial feature combination set is

$$F_1^i = f_i,\ f_i \in F, \tag{12}$$

and the active training set V_1^i, which is the entire training set, is

$$V_1^i = \{1, 2, \ldots, N\}. \tag{13}$$

Although, every training sample in S is involved in this initial training task, the computational complexity is not high because the input vector is just one-dimensional (1-D). After the training, each single-feature combination F_1^i is associated with a value M_1^i, which is the minimum of the objective function, and a group of support vectors v_i. The feature that yields the smallest M_1^i

$$j = \arg\min_{i \in \{1,2,\ldots,N\}} M_1^i \tag{14}$$

is chosen as the best one. Thus we obtain the initial feature combination $F_1 = \{f_j\}$ and its active training set $V_1 = \{v_j\}$.

At step n, we have already obtained the feature combination F_n that contains n features, and the active training set V_n. To add one more feature into the feature combination set, we test each remaining feature f_i one by one and construct the corresponding active training set for every new feature combination as follows:

$$F_{n+1}^i = F_n \cup \{f_i\} \quad \text{for } f_i \in F_n^{av},$$
$$V_{n+1}^i = V_n \cup \{v_i\}, \tag{15}$$

where $F_n^{av} = \{f_r \mid f_r \in F \text{ and } f_r \notin F_n\}$ is the collection of the available features to be selected from.

For each F_{n+1}^i, we train SVM just by using the samples in V_{n+1}^i. The resulting minimum of the objective functions and the collection of the support vectors are denoted as M_{n+1}^i and SV_{n+1}^i, respectively. Then the feature that yields the

combination with the least M_{n+1}^i

$$j = \arg\min_{f_i \in F_n^{av}} M_{n+1}^i \tag{16}$$

is selected, and accordingly the new feature combination F_{n+1}, and new active training set V_{n+1} are obtained as follows:

$$F_{n+1} = F_{n+1}^j,$$
$$V_{n+1} = SV_{n+1}^j. \tag{17}$$

The SFS process continues until no significant reduction of M_n^j is found or the desired number of features has been obtained.

3.5. FS_SFS: the integration of F_SFS and S_SFS

The integration of F_SFS and S_SFS is quite straightforward for which the basic idea is to discard the features with low scores according to the criterion discussed in Section 3.3, so as to reduce the number of features which S_SFS has to evaluate.

Assuming we are at step n of SFS with F_n and V_n available, FS_SFS works as follows:

(1) calculate the score R_{i,F_n} for each remaining feature f_i;
(2) select K_n highest scored features to construct F_n^{av};
(3) determine the next feature to be added using Eqs. (15) and (16);
(4) update the active training set using Eq. (17).

K_n determines how many features we want to keep after the filtering at step n. One extreme case is that K_n is equal to the number of all remaining features. In this scenario, the filter part does not contribute at all and evidently FS_SFS is reduced to S_SFS. Similarly, if K_n is equal to 1, the wrapper method is unnecessary and FS_SFS becomes F_SFS. K_n is usually chosen between the two extremes and thus works as a tuning parameter to balance between the performance and the complexity of the algorithm.

4. Experimental results

In the experiments, we apply the proposed feature selection method to both synthetic and real-world data. First, we design a synthetic data set to test whether the support vectors are effectively chosen by the active training set. Then we adopt the data set in Ref. [13] to compare the performance of FS_SFS and other three algorithms, which demonstrates that FS_SFS is more capable of selecting a small number of features, when most of the available features are irrelevant. Finally, we employ 10 data sets which are from the widely used University of California, Irvine (UCI) repository of machine learning [27] to test the capability of FS_SFS on real-world problems. For all the experiments, the optimization of SVM is achieved by SVMTorch [28].

4.1. Results on synthetic data 1

Three experiments are carried out on a synthetic data set. For each experiment we use N vectors $X = (x_1, x_2, \ldots, x_k)$ from two classes (class 1 or class -1) in a k-dimensional data space. The components x_i are *independent* Gaussian variables whose distributions are designed as follows:

$$p(x_i) = \begin{cases} \dfrac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\dfrac{x_i - 1}{2\sigma_i^2}\right) & \text{if } X \text{ belongs to class 1,} \\ \dfrac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\dfrac{x_i + 1}{2\sigma_i^2}\right) & \text{if } X \text{ belongs to class } -1, \end{cases} \tag{18}$$

where $\sigma_i = 0.5 \times 2^{(i-1)}$ and $i = 1, 2, \ldots, k$.

The three experiments deal with the 2-D, 3-D and 10-D data, respectively. The values of N and k in each experiment are

(1) 2-D case: $N = 100$ and $k = 2$;
(2) 3-D case: $N = 100$ and $k = 3$;
(3) 10-D case: $N = 250$ and $k = 10$.

Fig. 2 shows the effectiveness of the FS_SFS algorithm in estimating the support vectors in the 2-D scenario. By training SVM, using only x_1, we obtain the support vectors associated with x_1, which are denoted as v_1 and circled in Fig. 2(a). Similarly, we obtain v_2 (Fig. 2b). Then as discussed in Section 3, FS_SFS trains SVM by using only samples $V = v_1 \cup v_2$. As one can see from Figs. 2(c) and (d), FS_SFS yields exactly the same support vectors as the standard SVM training method, which involves all the original training samples.

We also test FS_SFS for the 3-D case, and Fig. 3 shows how the active training set changes when more and more features are added to the candidate feature set F. Again, FS_SFS and the standard SVM methods generate the same support vectors.

The third experiment is carried out on the 10-D feature space, and K_n is set to

$$K_n = \left\lfloor \frac{|F_n|}{2} \right\rfloor, \tag{19}$$

where $|F_n|$ denotes the number of features in the feature set F_n. In other words, half of the available features are discarded at every SFS iteration step. According to Eq. (18), the samples are generated in such a way that if $i < j$, the variance of feature x_i is larger than that of x_j, and therefore x_i has more discriminant ability than x_j. For that reason, we expect x_i to be selected before x_j. For the convenience of display, we assign a feature x_i a point as the following:

$$\text{Point}(x_i) = 11 - \text{pos}(x_i), \tag{20}$$

where $\text{pos}(x_i)$ is the order of x_i selected. For example, if x_i is the number one feature selected, its point would be 10.

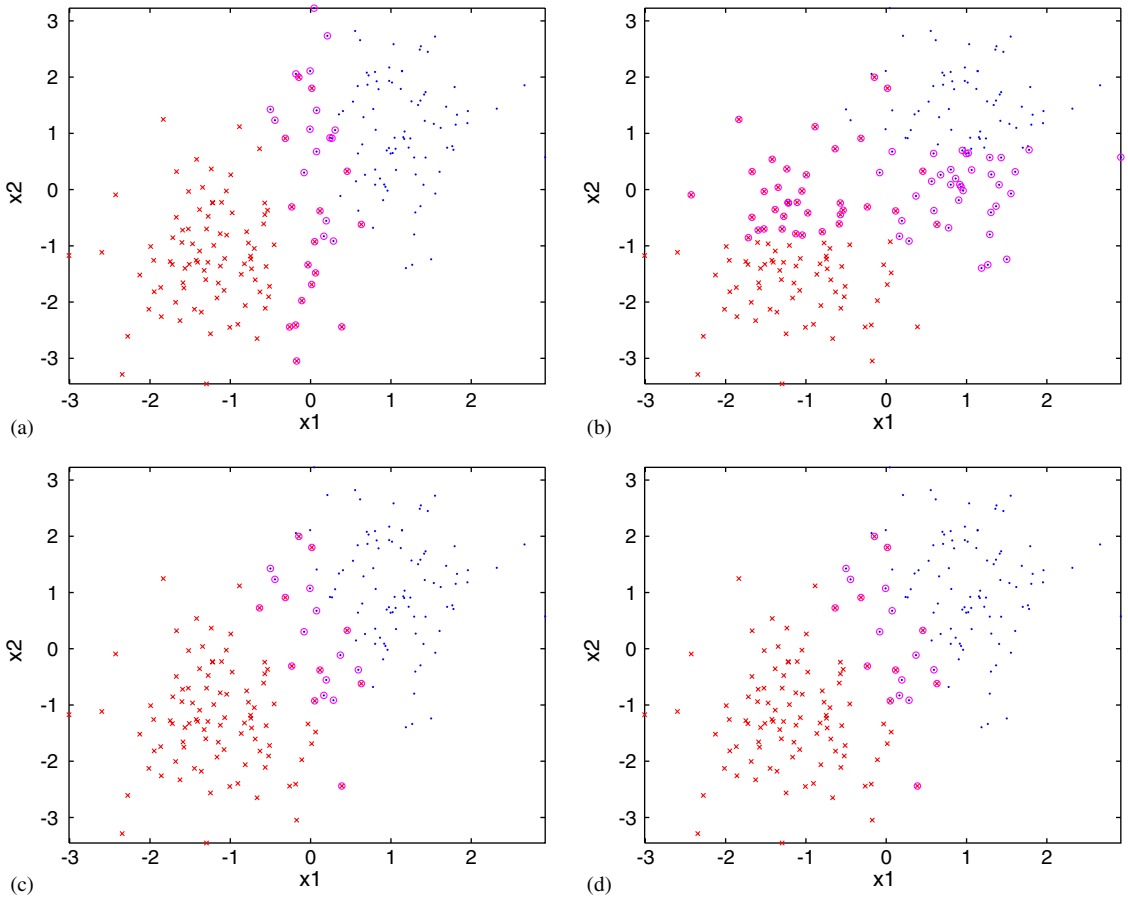


Fig. 2. The active training set of the 2-D case, which are circled, maintained by S_SFS. (a) v_1, which is the support vectors obtained by considering only feature x_1. (b) v_2, which is the support vectors obtained by considering only feature x_2. (c) The support vectors obtained by training SVM on $V = v_1 \cup v_2$. (d) The support vectors obtained by using all the training samples.

Fig. 4(a) gives the ideal point of x_i. Figs. 4(b) and (c) show the actual points of the features, which are averaged over 100 trials, when SFS and FS_SFS are applied, respectively. Here by notation SFS, we mean the wrapper methods using the SFS strategy. As one can see, FS_SFS is able to achieve similar results as SFS but with a lower computational cost.

Different values of the parameter K_n are tried for the 10-D case, and the classification accuracy and the run time obtained for the case are listed in Table 1. Not to our surprise, with the increasing of K_n the accuracy of the classification increases but the selection process takes longer time, which confirms that the performance and complexity of the algorithm can be balanced by tuning K_n.

4.2. Results on synthetic data 2

In the experiment described above, all the features to be selected from, are more or less relevant. In order to test the performance in the presence of a large number of irrelevant features, we adopt the artificial data designed in Ref. [13], in which only six out of total 202 available features are useful for the classification. The data set is constructed such that the class labels are evenly distributed as $P\{y=1\} = P\{y=-1\} = 0.5$. The feature vectors X are 202-D which are sampled according to the probability distribution function (pdf) shown in the following equation such that only the first six are relevant, while the rest of them is just noise.

$$P\{x_i \mid y\} = \begin{cases} 0.7 \cdot y \cdot N(i,1) + 0.3 \cdot N(0,1) & \text{if } 1 \leqslant i \leqslant 3, \\ 0.3 \cdot y \cdot N(i-3,1) + 0.7 \cdot N(0,1) & \text{if } 4 \leqslant i \leqslant 6, \\ N(0,1) & \text{if } 7 \leqslant i \leqslant 202. \end{cases} \tag{21}$$

FS_SFS is applied to this data set to select the best two features, and Fig. 5 gives the average classification errors on 500 testing samples versus various training set sizes. The performance of the standard SVM as well as three other feature selection algorithms, which are DFPA [15], SVM BFE [14], and the classical filter method using class separability as the filtering criterion, respectively, are also presented for the purpose of comparison.

As one can see, the presence of a large number of irrelevant features does hurt the performance, which again

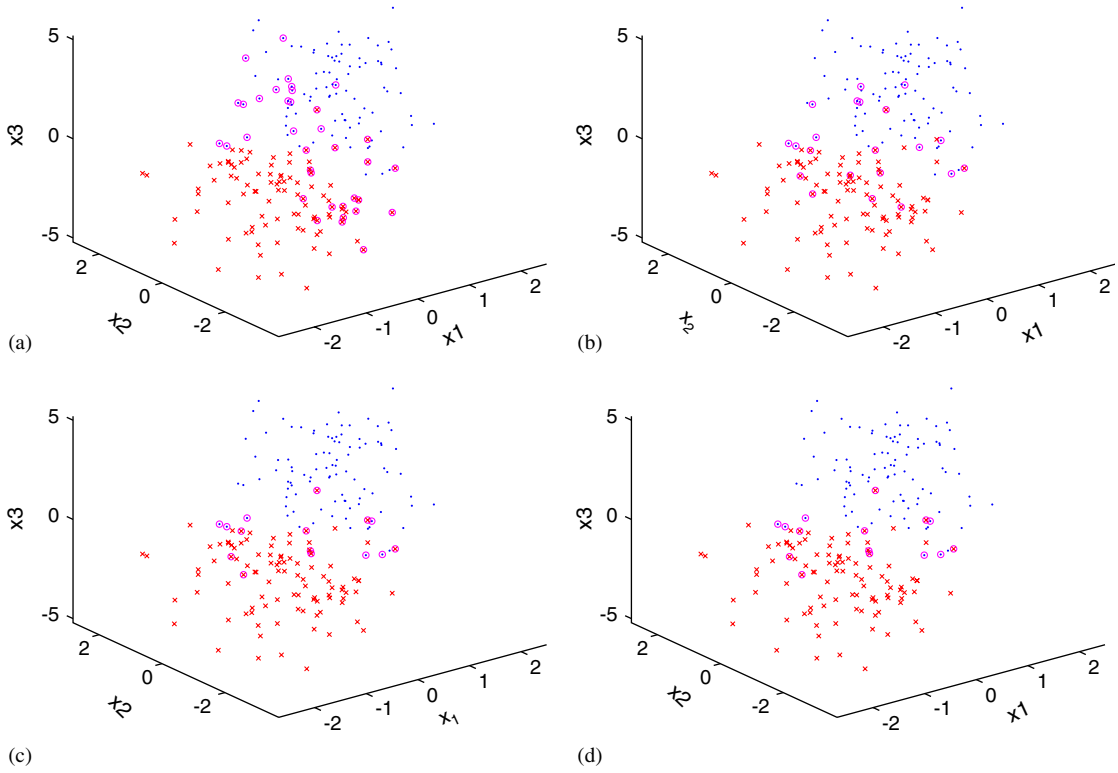


Fig. 3. The active training set of the 3-D case, which are circled, maintained by S_SFS. (a) The support vectors obtained when $F=\{x_1\}$. (b) The support vectors obtained when $F=\{x_1, x_2\}$. (c) The support vectors obtained when $F=\{x_1, x_2, x_3\}$. (d) The support vectors obtained by using all the training samples.

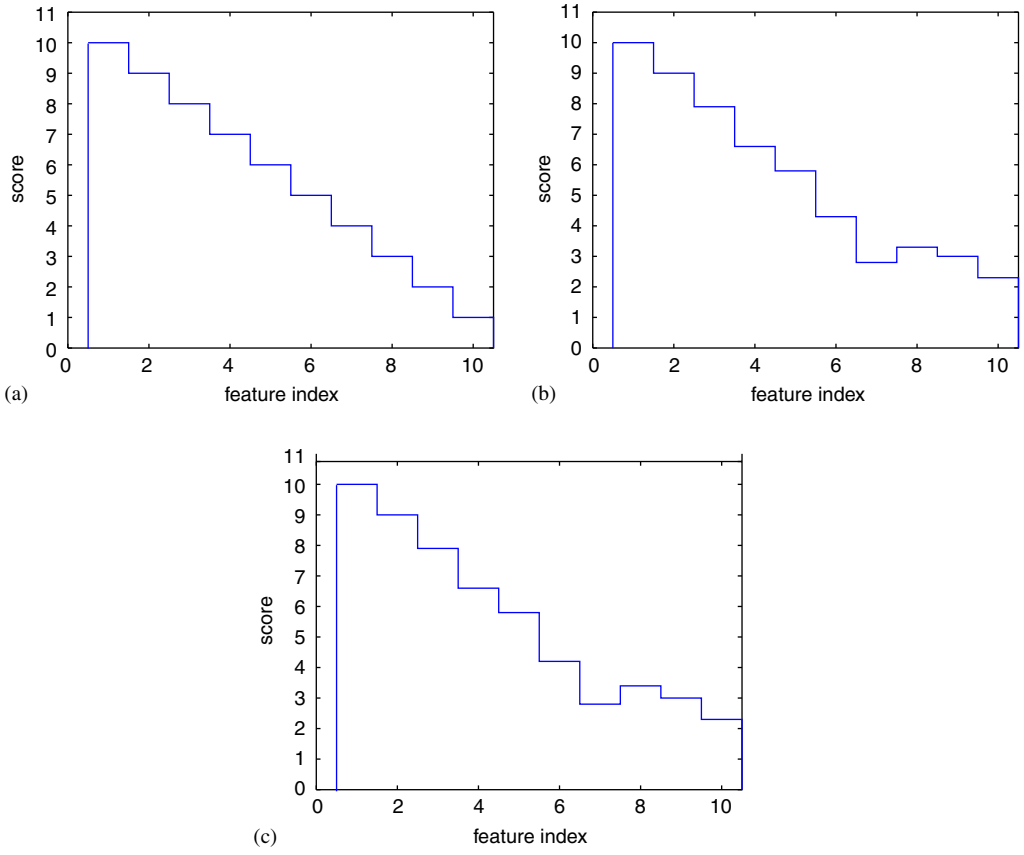


Fig. 4. The points of feature components x_i. (a) The ideal points. (b) The points obtained by using SFS. (c) The points obtained by using FS_SFS.

Table 1
Comparison of the classification accuracy and the run time with different values of K_n

	Classification accuracy		Run time (s)
	Training (%)	Testing (%)	
$K_n = \lfloor \frac{\lvert F_n \rvert}{4} \rfloor$	99.3	94.0	11.1
$K_n = \lfloor \frac{\lvert F_n \rvert}{2} \rfloor$	99.6	96.1	14.6
$K_n = \lfloor \frac{3\lvert F_n \rvert}{4} \rfloor$	99.4	96.9	16.5
$K_n = \lvert F_n \rvert$	99.8	97.2	22.7

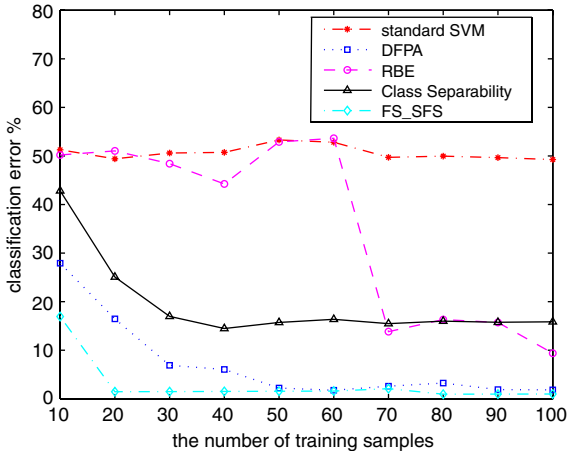


Fig. 5. A comparison of feature selection methods on a synthetic linear problem.

demonstrates the importance of feature selection. FS_SFS outperforms the other algorithms, especially when the training size is small. As more and more training samples are used, SFS_SFS performs marginally better than the method of DFPA [15].

4.3. Results on real-world data

The proposed algorithm is applied to ten real-world data sets [27] which are

(1) the BUPA Liver Disorders data set (BUPA Liver) which contains 354 instances with six features;
(2) the Wisconsin Breast Cancer data set (BCW) which contains 683 instances with nine features;
(3) the data of letters 'A' and 'B' from Letter Image Recognition data set (A–B-letter) which contains 1555 instances with 16 features;
(4) the Johns Hopkins University Ionosphere data set (Ionosphere) which contains 351 instances with 34 features;
(5) the Glass Identification data set (Glass) which contains 214 instances with nine features;
(6) the Heart Disease data set collected from Cleveland Clinic Foundation (Heart Disease) which contains 303 instances with 13 features;

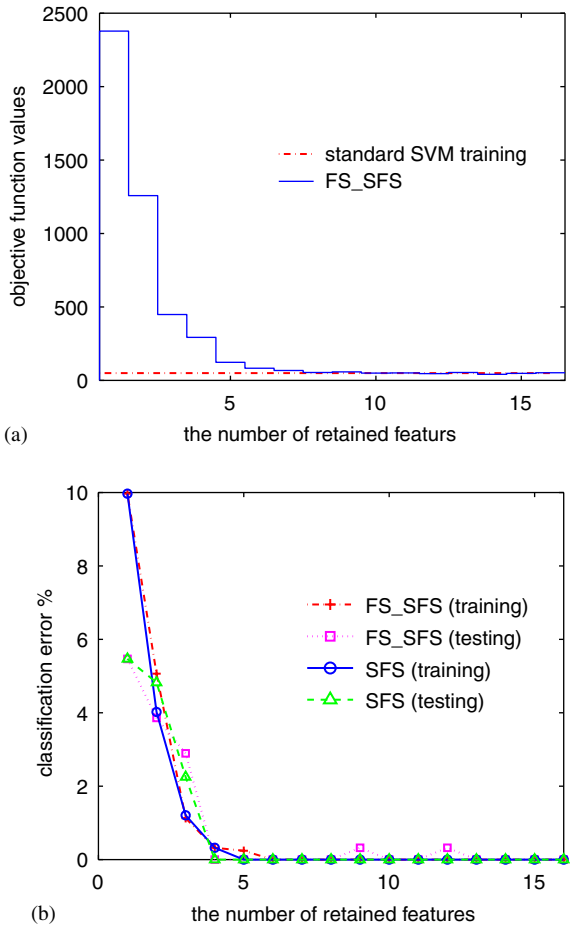


Fig. 6. The results of letter recognition (A and B). (a) The change of the value of the objective function versus the number of retained features. (b) The comparison of classification errors between FS_SFS and SFS.

(7) Pima Indians Diabetes data set (PI Diabetes) which contains 768 instances with eight features;
(8) Japanese Credit Screening data set (Credit Screening) which contains 690 instances with 15 features;
(9) Postoperative Patients data set (PO Patients) which contains 90 instances with eight features;
(10) Wisconsin Diagnostic Breast Cancer data set (WDBC) which contains 569 instances with 30 features.

For each data set we randomly set aside 20% instances as the testing samples and the rest as the training samples. Again we let $K_n = \lfloor |F_n|/2 \rfloor$. Fig. 6 shows the performance of FS_SFS on the A–B-letter data set. The solid line in Fig. 6(a) represents the value of the objective function obtained by FS_SFS. As one can see, it monotonically decreases with more features added and gradually converges to the dash–dot line, which depicts the value of the objective function when SVM is trained utilizing all the training samples and all the available features. Fig. 6(b) plots the classification errors yielded by FS_SFS and SFS, respectively, where they yield comparable errors over both the training

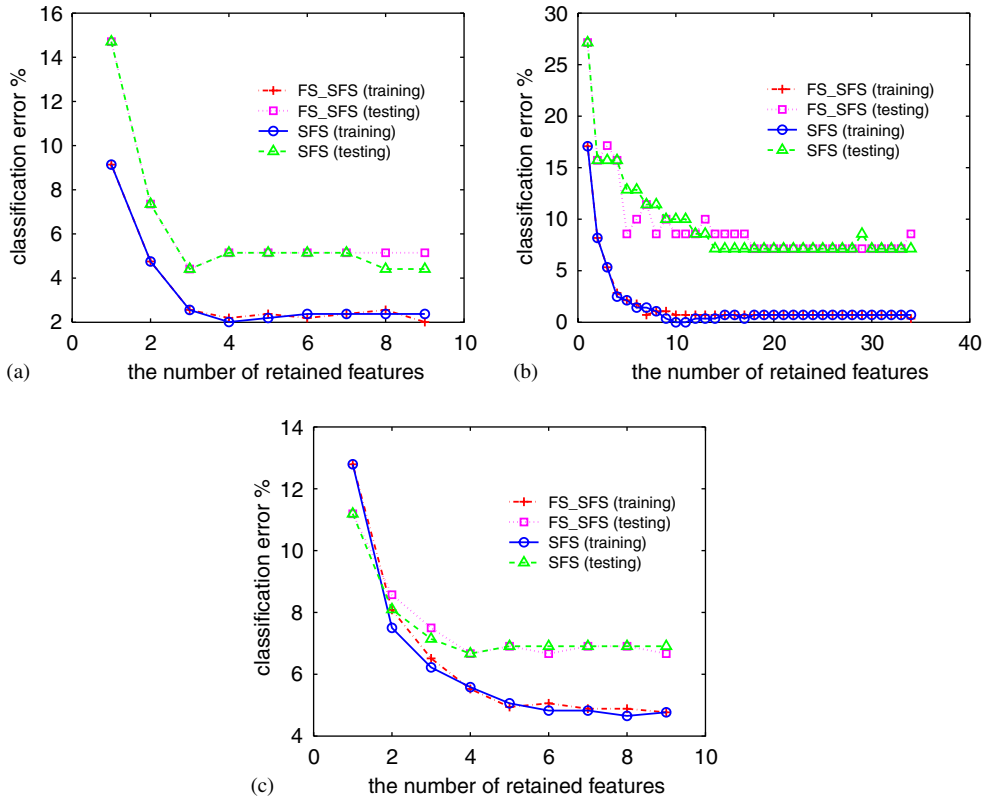


Fig. 7. The comparison of classification errors between FS_SFS and SFS. (a) The Wisconsin Breast Cancer data set. (b) The Johns Hopkins University Ionosphere data set. (c) The Glass Identification data set.

Table 2
Comparison of the average classification accuracy and the run time between FS_SFS and SFS over 20 trials

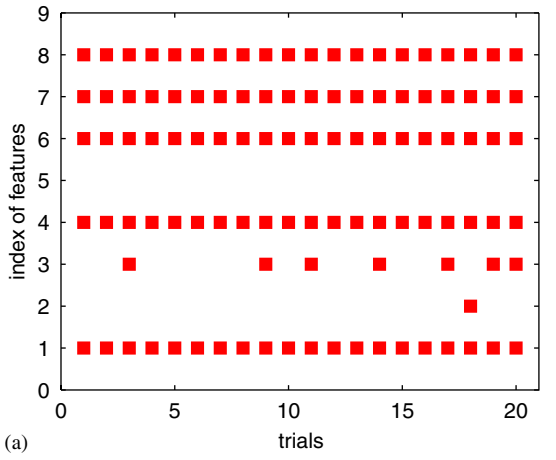
	Number of features		Classification accuracy				Run time (s)		
	Available	Selected	Training		Testing		FS_SFS	SFS	$\frac{\text{FS\_SFS}}{\text{SFS}}$ (%)
		(min, max)	FS_SFS (%)	SFS (%)	FS_SFS (%)	SFS (%)			
BUPA Liver	6	4.6 (4, 5)	78.7	78.5	70.2	71.7	4.31	6.08	70.9
BCW	9	5.4 (5, 6)	97.4	97.4	96.3	95.4	10.6	13.3	79.7
A–B Letter	16	6.2 (6, 7)	99.95	100	99.7	99.8	27.2	37.7	72.1
Ionosphere	34	10 (10, 10)	98.9	99.3	92.0	90.6	81.5	118.9	68.5
Glass	9	4 (4, 4)	94.1	94.2	93.8	93.3	21.6	27.6	78.1
Heart Disease	13	8.6 (7, 9)	93.3	93.3	84.8	84.8	19.7	33.9	58.1
PI Diabetes	8	4.2 (4, 5)	81.0	79.4	74.9	74.9	15.5	24.2	64.0
Credit Screening	15	6.6 (5, 7)	90.4	91.4	85.6	84.8	28.8	49.6	58.1
PO Patients	8	5 (4, 6)	73.1	73.1	71.8	71.8	9.1	12.0	75.8
WDBC	30	15 (15, 15)	99.3	99.9	92.9	92.4	7.9×10^3	1.3×10^4	62.1

The range of the number of features selected are given in the parentheses as (min, max). For BCW, Glass and Post-operative Patients data set, the linear SVM is employed. For the rest, nonlinear SVM is utilized and the radial basis function is adopted as the kernel function.

and the testing sets. The similar results are observed for the other nine data sets, and Fig. 7 shows the performance comparison on three data sets.

When the value of the objective function does not decrease significantly, the feature selection process stops and the features that have not been selected at that point are deemed irrelevant to the classification problem. Throughout the experiments, FS_SFS and SFS always select the same number of features when the stop condition is satisfied. More detailed results are listed in Table 2, which evidentally shows that FS_SFS improves the efficiency of SFS without sacrificing the accuracy of either the selection or the classification.

FS_SFS also shows stability in the selection of features. An example is shown in Fig. 8(a). X and Y axes are the



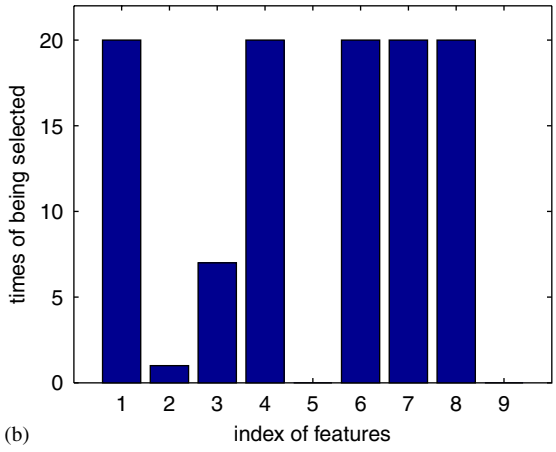


Fig. 8. The stability of FS_SFS on the BCW data set. (a) The selected features (squared) during trials. (b) Each bar shows how many times the corresponding features are selected over 20 trials.

indices of features and trials, respectively, and the selected features are highlighted by squares. For example, in the first trial five features (feature #1, #4, #6, #7 and #8) are selected, and in the fifth trial six features (feature #1, #3, #4, #6, #7 and #8) are selected. Over the 20 trails conducted, the results are stable: feature #1, #4, #6, #7 and #8 are always selected while feature #5 and #9 are not. The stability can be seen more clearly in Fig. 8(b) which displays the total times each feature has been selected over the 20 trials.

As discussed before, one novelty of FS_SFS is that the repeated training of SVM is conducted on the subsets of the original training sets. In order to concretely show how this strategy helps to reduce the computational cost, in Fig. 9 against the number of the selected features we plot the change of R_a, which is defined as

$$R_a = \frac{S_a}{S_t} = \frac{\#\text{ of the samples in the subsets}}{\#\text{ of the samples in the original training sets}}. \quad (22)$$

As one can see, although the value of R_a varies from data set to data set, it does not grow quickly during the selection process. Actually, it stays much less than 1 except for the first iteration, when the active training set V_1^i is the entire set (Eq. (13)) and therefore $R_a = 1$. It is also observed that R_a hardly decreases significantly as the iteration process continues and here we offer an explanation. According to Eq. (15), we know that

$$|V_{n+1}^i| = |V_n| + |v_i| - |V_n \cap v_i| \geqslant |v_i|, \quad (23)$$

where $|\cdot|$ denote the number of elements in the corresponding set. More specifically, the average size of the active training

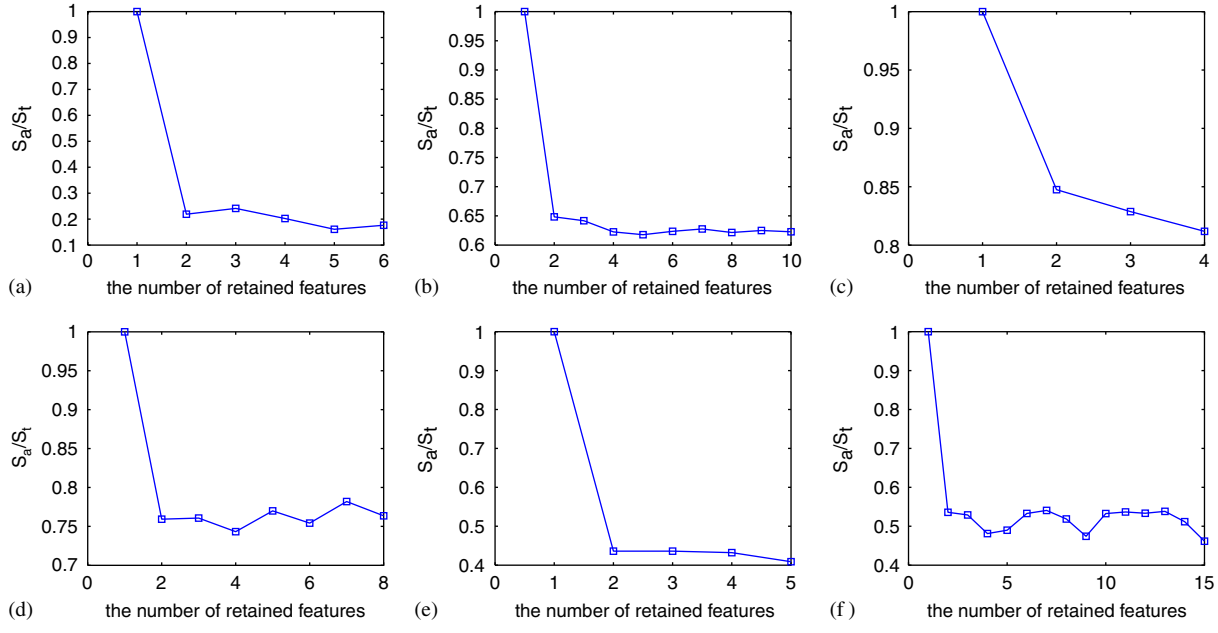


Fig. 9. The change of R_a, which is defined as the ratio of the size of the subset (used for training) to the size of the total training set, during the feature selection process for six data sets. (a) BCW, (b) Ionosphere, (c) PI Diabetes, (d) Heart Disease, (e) Glass, (f) WDBC.

Table 3
Comparison of the classification accuracy between FS_SFS, DFPA [15], SVM BFE (Ref. [14] for linear and Ref. [16] for nonlinear problems) and the filter method using class separability as the filtering criterion (boldface indicates the best performance)

	# of selected features	Classification accuracy							
		Training				Testing			
		FS_SFS (%)	DFPA (%)	RFE (%)	Filter (%)	FS_SFS (%)	DFPA (%)	RFE (%)	Filter (%)
BUPA Liver	5	**78.7**	77.2	77.3	68.6	**70.2**	67.1	69.7	61.7
BCW	5	**97.4**	97.1	97.1	96.7	**96.3**	**96.3**	95.9	96.2
A–B Letter	6	**99.95**	99.08	99.8	99.6	**99.7**	98.9	99.4	99.3
Ionosphere	10	**98.9**	96.1	98.0	94.1	92.0	92.6	**92.7**	92.0
Glass	5	**94.9**	91.3	94.7	94.1	**92.9**	91.9	87.1	90.5
Heart Disease	9	**91.9**	91.2	91.9	80.8	**78.3**	69.7	76.8	77.3
PI Diabetes	4	**79.3**	77.2	73.0	77.4	74.5	73.1	74.4	**75.7**
Credit Screening	7	**90.4**	68.2	68.0	67.1	**85.6**	70.0	68.8	69.7
PO Patients	5	**73.7**	70.4	65.1	61.4	**71.8**	67.1	54.1	61.2
WDBC	15	**99.3**	97.1	97.5	95.1	93.0	**95.4**	**95.4**	92.9

set is lower bounded by

$$\frac{\sum_{i\in F_n^{av}}|v_i|}{|F_n^{av}|}, \tag{24}$$

which prevents the decrease of R_a.

Table 3 gives the performance comparison of FS_SFS, DFPA [15], SVM BFE (Ref. [14] for linear and Ref. [16] for nonlinear problems) and the filter method using class separability as the filtering criterion, respectively. Similar to what we have observed on the synthetic data 2, when the number of training samples is insufficient with respect to the number of features, which is the case for the Post-operative Patient data set (90 instances with eight features), FS_SFS achieves significantly higher classification accuracy than the other three approaches. Another special data set is the Credit Screening data set. The instances in this set exhibit a good mix of attributes—continuous, nominal with small numbers of values, and nominal with larger numbers of values, and in this case FS_SFS again shows major advantage. For the rest of the adopted data set where most of the features are relevant and the training samples are relatively ample, the performances of FS_SFS, DFPA and SVM BFE are close. Nevertheless, FS_FSF still yields the best results most of the time.

5. Conclusions

In this paper, we present a novel feature selection method in the context of SVM. Fundamentally the proposed method, which is named FS_SFS, is a more efficient version of a wrapper/SFS approach. FS_SFS introduces a feature pruning process into the wrapper part such that some "noninformative" features are filtered out and consequently the number of SVM training is reduced. To make the pruning process more effective, we develop a new feature ranking criterion to take into account the class separability of individual features as well as the correlation between features. Furthermore, during the SFS searching process, an active training set is maintained as the candidates of the support vectors. SVM training is thus performed over the reduced training set. In this way, the number of samples participating in a single optimization procedure decreases, and the training process is expedited. As SVM is becoming a popular learning machine for object classification, the contribution of this paper is important as it significantly reduces the time of training, which has been a bottleneck of many applications using SVM. Yet, the efficiency is achieved without sacrificing the effectiveness. The experimental results show that the proposed method delivers as good a performance, when the features are more or less relevant, and produces a more effective classifier when many irrelevant features are present.

References

[1] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (1) (2000) 4–37.
[2] Y. Yang, J.O. Pedersen, A comparative study on feature selection, in: Proceedings of ACM International Conference on Research and Development in Information Retrieval, 1999, pp. 42–49.
[3] J. Mao, A.K. Jain, Artificial neural networks for feature selection and multivariate data projection, IEEE Trans. Neural Networks 6 (2) (1995) 296–317.
[4] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. 97 (1997) 273–324.
[5] B.-H. Juang, S. Katagiri, Discriminative learning for minimum error classification, IEEE Trans. Signal Process. 40 (12) (1992) 3043–3054.
[6] H. Watanabe, T. Yamaguchi, S. Katagiri, Discriminative metric design for robust pattern recognition, IEEE Trans. Signal Process. 45 (11) (1997) 2655–2662.
[7] M. Pontil, A. Verri, Support vector machines for 3D object recognition, IEEE Trans. Pattern Anal. Mach. Intell. 20 (6) (1998) 637–646.
[8] A. Tefas, C. Kotropoulos, I. Pitas, Using support vector machines to enhance the performance of elastic graph matching for frontal face

authentication, IEEE Trans. Pattern Anal. Mach. Intell. 23 (7) (2001) 735–746.
[9] S. Tong, E. Change, Support vector machine active learning for image retrieval, in: Proceedings of ACM International Conference on Multimedia, 2001, pp. 107–118.
[10] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of International Conference on Machine Learning, 1999, pp. 200–209.
[11] Y. Liu, Y.F. Zheng, X. Shen, R. Ewing, A GM based multi-layer method for object tracking in video sequences, in: Proceedings of International Conference on Information Technology Research and Education, 2003, pp. 11–16.
[12] P.S. Bradley, O.L. Mangasarian, Feature selection via concave minimization and support vector machines, in: Proceedings of International Conference on Machine Learning, 1998, pp. 82–90.
[13] J. Weston, S. Mukherjee, O. Chapelle, M. Ponntil, T. Poggio, V. Vapnik, Feature selection for SVMs, Advances in Neural Information Processing Systems, vol. 13, MIT Press, Cambridge, MA, 2001.
[14] I. Guyon, J. Weston, S. Barnhill, V. Bapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (1–3) (2002) 389–422.
[15] K.Z. Mao, Feature subset selection for support vector machines through discriminative function pruning analysis, IEEE Trans. Syst. Man, Cybern. 34 (1) (2004) 60–67.
[16] T. Evgeniou, M. Pontil, C. Papageorgiou, T. Poggio, Image representation and feature selection for multimedia database search, IEEE Trans. Knowledge Data Eng. 15 (4) (2003) 911–920.
[17] V.N. Vapnik, An overview of statistical learning theory, IEEE Trans. Neural Networks 10 (5) (1999) 988–999.
[18] C. Cortes, V.N. Vapnik, Support vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[19] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, NY, 1995.
[20] K.Z. Mao, Fast orthogonal forward selection algorithm for feature subset selection, IEEE Trans. Neural Networks 13 (5) (2002) 1218–1224.
[21] P.M. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection, IEEE Trans. Comput. 26 (1977) 917–922.
[22] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, A.K. Jain, Dimensionality reduction using genetic algorithms, IEEE Trans. Evol. Comput. 4 (2) (2001) 164–171.
[23] T. Marill, D.M. Green, On the effectiveness of receptors in recognition systems, IEEE Trans. Inf. Theory 9 (1963) 11–17.
[24] P.A. Devijver, J. Kittler, Pattern recognition: a statistical approach, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[25] J.M. Park, J. Reed, Q. Zhou, Active feature selection in optic nerve data using support vector machine, in: Proceedings of International Joint Conference on Neural Networks, 2002, pp. 1178–1182.
[26] L. Hermes, J.M. Buhmann, Feature selection for support vector machines, in: Proceedings of International Conference on Pattern Recognition, 2000, pp. 712–715.
[27] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, CA, http://www.ics.uci.edu/mlearn/MLRepository.html, 1998.
[28] R. Collobert, S. Bengio, SVMTorch: support vector machines for large-scale regression problems, J. Mach. Learn. Res. 1 (2001) 143–160.

About the Author—YI LIU received the B.S. and M.S. degrees from the Department of Information Science and Electronics Engineering at Zhejiang University, Hangzhou, China, in 1997 and 2000, respectively. She is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering at The Ohio State University, Columbus, Ohio. Her research interests include machine learning, pattern recognition and their applications in the area of multimedia signal processing.

About the Author—YUAN F. ZHENG received B.S. degree from Tsinghua University, Beijing, China in 1970, and the MS and Ph.D. degrees in Electrical Engineering from The Ohio State University, Columbus, Ohio in 1980 and 1984, respectively.
From 1984 to 1989, he was with the Department of Electrical and Computer Engineering at Clemson University, Clemson, South Carolina. Since August 1989, he has been with The Ohio State University, where he is currently Winbigler Professor of Electrical and Computer Engineering. He is on leave at the Shanghai Jiao Tong University, Shanghai, China in the academic year 2004–2005 and continues his participation and collaboration there. His research interests include two aspects. One is in wavelet transform for image and video compression for internet and satellite communications. Current efforts focus on content-based compression, 3D wavelet transformation, and video object tracking. The other is in robotics which includes robots for biological applications, multiple robots coordination, legged robots, human-robot coordination, and personal robotics. He is currently on the Editorial Board of International Journal of Multimedia Tools and Applications, on the Editorial Board of Autonomous Robots, an associated editor of the International Journal of Intelligent Automation and Soft Computing, on the Editorial Board of International Journal of Intelligent Control and Systems, and on the Editorial Board of International Journal of Control, Automation, and Systems.
Professor Zheng was Vice-President for Technical Affairs of the IEEE Robotics and Automation Society from 1996 to 1999. He was an associate editor of the IEEE Transactions on Robotics and Automation between 1995 and 1997. He was the Program Chair of the 1999 IEEE International Conference on Robotics and Automation, held in Detroit, MI, on May 10–15, 1999. Professor Zheng received the Presidential Young Investigator Award from President Ronald Reagan in 1986.