PERGAMON

Pattern Recognition 35 (2002) 701–711

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Feature selection using tabu search method

Hongbin Zhang*[,1], Guangyu Sun

Computer Institute of Beijing Polytechnic University, West San Huan North Road 56, 6#9 Beijing 100044, People's Republic of China

Received 9 February 1999; received in revised form 6 July 2000; accepted 5 December 2000

Abstract

Selecting an optimal subset from original large feature set in the design of pattern classifier is an important and difficult problem. In this paper, we use tabu search to solve this feature selection problem and compare it with classic algorithms, such as sequential methods, branch and bound method, etc., and most other suboptimal methods proposed recently, such as genetic algorithm and sequential forward (backward) floating search methods. Based on the results of experiments, tabu search is shown to be a promising tool for feature selection in respect of the quality of obtained feature subset and computation efficiency. The effects of parameters in tabu search are also analyzed by experiments. © 2001 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Feature selection; Tabu search; Pattern classifier; Search methods; Curse of dimensionality

1. Introduction

The problem of feature selection is important in designing pattern classifier, whose goal is to acquire an efficient subset so as to reduce the dimension of the feature set. When the number of initial features becomes too large, not only the cost of collecting features increases, but the performance of designed classifier will not be guaranteed in the case of small sample size. From the standpoint of Bayesian decision rules there are no "bad" features, i.e., the performance of a Bayes classifier cannot be improved by eliminating a feature. But in practice, the assumptions in designing Bayes classifier are usually not satisfied. As a consequence, for a given amount of samples, reducing the number of features may result in improving a non-ideal classifier's performance.

In applications, there are two forms of feature selection, each form addresses a specific objective and leads to a distinct type of optimization. The first form is to find an optimal subset having a predefined number of features and yield the lowest error rate of a classifier. This is a non-constrained combinatorial optimal problem. Another form of feature selection is aimed at seeking the smallest subset of features for which error rate is below a given threshold. This is a constrained combinatorial optimal problem, in which the error rate serves as a constraint and the smallest subset of features is the primary search criterion.

Let $\boldsymbol{F} = \{f_1, f_2, \ldots, f_n\}$ be the initial set of features with cardinality n. $\boldsymbol{S}$ is a subset of $\boldsymbol{F}$. $|\boldsymbol{S}|$ represents subset size, i.e., the number of inclusive features in the subset. Let m denote the required subset size in the 1st-form of feature selection problem and t the tolerable threshold in the 2nd-form of feature selection problem. Let $error(\boldsymbol{S})$ be the relevant error rate or some other measures of performance such as class separability (e.g., Mahalanobis distance), when $\boldsymbol{S}$ is used to design the classifier. The two forms of feature selection problem can be represented as follows:

1st-form of feature selection:
$$\min \quad J(\boldsymbol{S}) = error(\mathbf{S})$$
$$\text{s.t.} \quad \boldsymbol{S} \subset \boldsymbol{F},\ |\boldsymbol{S}| = m, \quad m < n.$$
2nd-form of feature selection:
$$\min \quad J(\boldsymbol{S}) = |\boldsymbol{S}|$$
$$\text{s.t.} \quad \boldsymbol{S} \subseteq F, \quad error(\boldsymbol{S}) < t.$$

[1] This work was supported by the National Natural Science Foundation of China (69675007) and Beijing Municipal Natural Science Foundation (4972008).

* Corresponding author. Tel.: + 86-10 68907155.
E-mail address: zhb@public.bta.net.cn (H. Zhang).

PII: S0031-3203(01)00046-2

Theoretically, both forms of feature selection are NP-hard problems. So the optimal solution cannot be guaranteed to be acquired except for doing exhaustive search in the solution space [1]. But exhaustive search is feasible only for small n. For a bigger n, the explosive computational cost makes the exhaustive search impracticable. The branch and bound search scheme (BB) introduced by Narendra and Fukunaga is more efficient by means of pruning a lot of "infeasible" solutions [2]. On the assumption that the objective function is monotonic, this algorithm is potentially capable of examining all feasible solutions so that it is an optimal method. Unfortunately, monotonic condition is seldom satisfied. Moreover, even if 99.9% computational cost saving is done by BB, the computational complexity remains exponential. Thus, BB or its improved version—relaxed BB (by introducing the concept of approximate monotonicity) [3]—are still unavailable for a large n.

The need of trade-off between the optimality and efficiency of algorithms for NP-problems was recognized early, and the main stream of feature selection research was thus directed toward suboptimal but efficient and robust methods. The preliminary works on feature selection was started in the early 60s. The approaches of feature selection at that time were based on probabilistic measures of class separability and on entropies. In some methods the independence of features was assumed and the features were selected on the basis of their individual merits. Such methods ignore the interactions among features. As a result, the selected subsets are not satisfactory. As was pointed out by Cover, the best two independent features do not have to be the two best [4]. The sequential forward selection method (SFS), sequential backward selection method (SBS), and their generalized versions GSFS, GSBS invented later belong to greedy algorithms in essence. These algorithms begin with a feature subset and sequentially add or remove features until some termination criterion is met. But these methods suffer from the so-called "nesting effect". It means that the features discarded cannot be re-selected, and the features selected cannot be removed later. Since these algorithms do not examine all possible feature subsets, they are not guaranteed to produce the optimal result. The plus l take away r (PTA) methods was proposed to prevent the "nesting" effect. But there is no theoretical guidance to determine the appropriate value of l and r. Max–min (MM) method proposed by Backer and Shipper is a computationally efficient method [5]. It only evaluates the individual and pairwise merits of features. The experiments made by many researchers show that this method gives the poorest results [6,7]. This confirms that it is not possible to select a feature subset in a high-dimensional space based on only two-dimensional information measures without a substantial information loss.

In recent years, there have been some developments in the problem of feature selection. Siedlecki and Sklansky introduced the use of genetic algorithm (GA) for this problem and obtained good results [8,9]. But the premature phenomena in GA seem to be difficult to avoid and it needs to carefully select and test the parameters in GA. Pudil et al. improved the sequential methods by introducing sequential forward floating selection (SFFS) and sequential backward floating selection (SBFS) [10]. These two methods can be understood as plus 1-minus x and minus 1-plus x, where x is dynamically changed according to the backtrack effect. SFFS and SBFS avoid the problem of predefining l and r in PTA, and have a mechanics to control the depth of backtrack. According to the experimental results of Jain and Zongker [7], SFFS and SBFS achieve results comparable to the optimal algorithm (BB) but are faster than BB.

To conclude, despite some progress has been obtained, the available feature selection techniques for large feature set are not yet completely satisfactory. They are either computationally feasible but far from optimal, or they are optimal or almost optimal but cannot cope with the computational complexity of feature selection problems of realistic size. Recently, there has been a resurgence of interest in applying feature selection methods due to the application needs. In the application of information fusion of multiple sensors' data, integration of multiple models and data mining, the number of features is usually quite large. In some cases, it may be over 100 [7]. It is necessary to research more powerful methods for feature selection, which should give very good results and should be computationally more efficient. In Ref. [11], Sklansky stated that when the initial set contains 20 or more features and the selected subset contains 10 or fewer features, the problem of selecting a best or near-best subset can be quite difficult because we must search a large—often astronomically large—space of subsets of candidate features, and determine a figure of merit for an optimum or near-optimum classifier operating on each subset in this space. He referred to such problems as large-scale feature selection. Nowadays along with the upgrading of computer performance, the feature selection containing about 20 features has become easier, and the feature selection problem including 20–49 features has changed to so-called "medium-sized" problem. But for a classifier based on personal computer, the feature selection including a few dozens or over 100 of features is still a difficult and challenging problem.

It has been argued that since feature selection is typically done in an off-line manner, the execution time of a particular algorithm is of much less important than its ultimate classification. While this is generally true for feature sets of appropriate size. However, if the number of features reaches a few dozen or even more, the execution time becomes extremely important as it may be impractical to run some algorithms even once on such large data sets.

In a recent paper [12], Kudo and Sklansky carried out a comparative study of algorithms for large-scale feature selection (where the number of features is over 50). Unfortunately, their comparative study does not include tabu search method. In this paper, we introduce the use of tabu search method for feature selection, and compare the performance of tabu search with some other algorithms. In the following section, a brief description of tabu search is presented. In Section 3, experimental results of tabu search are shown, providing also comparisons to other methods. The parameters in tabu search will be discussed in Section 4, and a short conclusion follows in Section 5.

2. Tabu search

The tabu search, proposed by Glover [13,14], is a meta heuristic method that can be used to solve combinatorial optimization problem. It has received widespread attention recently. Its flexible control framework and several spectacular successes in solving NP-hard problems caused rapid growth in its application. It differs from the local search technique in the sense that tabu search allows moving to a new solution which makes the objective function worse in the hope that it will not trap in local optimal solutions. Tabu search uses a short-term memory, called tabu list, to record and guide the process of the search. In addition to the tabu list, we can also use a long-term memories and other prior information about the solutions to improve the intensification and/or diversification of the search.

The tabu search scheme can be outlined as follows: start with an initial (current) solution $\boldsymbol{x}$, called a configuration, evaluate the criterion function for that solution. Then, follow a certain set of candidate moves, called the neighborhood $N(\boldsymbol{x})$ of the current solution $\boldsymbol{x}$. If the best of these moves is not tabu (i.e., not in the tabu list) or if the best is tabu, but satisfies the aspiration criterion, which will be explained below, then pick that move and consider it to be the new current solution. Repeat the procedure for a certain number of iterations. On termination, the best solution obtained so far is the solution of the tabu search. Note that the solution that is picked at certain iteration is put in the tabu list (TL) so that it is not allowed to be reversed in the next l iterations, i.e., this solution is tabu. The l is the size of TL. When the length of tabu list reaches that size, then the first solution on the TL is freed from being tabu and the new solution enters that list. The process continues. The TL acts as a short-term memory. By recording the history of searches, tabu search can control the direction of the following searches. The aspiration criterion could reflect the value of the criterion function, i.e., if the tabu solution results in a value of the criterion function that is better than the best known so far, then the aspiration criterion is satisfied and the tabu restriction is relieved, and move to this solution is allowed.

Let $\boldsymbol{S}_b$ denote the best solution obtained so far, TL the tabu list. The algorithmic description of the tabu search can be summarized as follows:

(1) **Initialize.** Generate an initial solution $\boldsymbol{x}$. And let $\boldsymbol{S}_b = \boldsymbol{x}$. $k = 1$, $TL = \Phi$.

(2) **Generate candidate set.** Randomly pick out a certain number of solutions from the neighborhood of $\boldsymbol{x}$ to form the candidate set $N(\boldsymbol{x})$.

(3) **Move.** (a) If $N(\boldsymbol{x}) = \Phi$, Go back to step 2 to regenerate the candidate set. Otherwise, find out the best solution $\boldsymbol{y}$ in $N(\boldsymbol{x})$. (b) If $\boldsymbol{y} \in TL$, i.e., it is tabu, and $\boldsymbol{y}$ does not satisfy the aspiration criterion, let $N(\boldsymbol{x}) = N(\boldsymbol{x}) - \{\boldsymbol{y}\}$. Then go to 3(a). Otherwise, let $\boldsymbol{x} = \boldsymbol{y}$. And let $\boldsymbol{S}_b = \boldsymbol{y}$ if $\boldsymbol{y}$ is better than $\boldsymbol{S}_b$.

(4) **Output.** If termination condition is satisfied, stop and output the $\boldsymbol{S}_b$. Otherwise, let $TL = TL \cup \{\boldsymbol{x}\}$. (Add the new solution to the tail of TL. And if the length of TL exceeds a predefined size, remove the head item of the list.). Let $k = k + 1$ and go back to step 2.

The above paragraph has outlined the basic steps of the tabu search for solving combinatorial optimization problems. For more details on the tabu search as well as a list of successful application, the reader is encouraged to refer to Glover [13–15]. In the next section, we develop a new method for feature selection based on tabu search technique.

3. Application of tabu search to feature selection

In this section, we present our tabu search-based algorithms for the two forms of feature selection problem, compare the performance of tabu search with that of other algorithms, such as classical sequential methods, branch and bound method, genetic method and sequential floating methods by experiments.

3.1. Feature selection based on misclassification error rate

The 2nd-form feature selection stated in Section 1 is a constrained optimization problem. Generally, a constrained optimization problem can be converted into a relevant non-constrained optimization problem by transforming the constraint into a penalty function. The optimization problem of the 2nd-form feature selection can be converted into the following forms:

$$\text{Min} \quad J(\boldsymbol{S}) = |\boldsymbol{S}| + p(e(\boldsymbol{S}))$$

$$\text{s.t.} \quad \boldsymbol{S} \subseteq \boldsymbol{F},$$

where $p(e(\boldsymbol{S}))$ is the penalty function. We adopt the penalty function used by Siedlicki and Sklansky

[9], i.e.,

$$p(e) = \frac{\exp((e-t)/m) - 1}{\exp(1) - 1},$$

where e is the error rate, t the tolerant error rate, and m a scale factor. This penalty function is monotonic with respect to e. $p(e)$ is negative when $e < t$, and $p(t) = 0$, $p(t+m) = 1$. As e approaches to zero, $p(e)$ slowly approaches its minimal value:

$$p(0) = \frac{\exp(-t/m) - 1}{\exp(1) - 1} > -\frac{1}{\exp(1) - 1}.$$

For greater value of e than $t + m$, the penalty function quickly rises toward infinity. According to the above properties, this penalty function is suitable for the 2nd-form feature selection problem. In the following experiments, we prefer to use $t = 0.20$ and $m = 0.01$ after several different settings were tested.

In an ideal experimental situation, it would be desirable to test tabu search-based feature selection procedure on a group of real data sets derived from various applications, because that would give the results of such testing credibility and statistical validity. However, for practical reasons any extensive testing involving real data is prohibitive. Therefore, we follow the path used by Siedlecki and Sklansky [9]. To simulate the conditions encountered by feature selection in practice, they replaced the true error rate function with a model of the classifier's error rate which is a function of the feature selection vector, accounting for various interactions among the features and various defects of the classifier training procedures, but without assuming any forms of underlying distributions in the data. The error rate model is as follows:

$$e = e_0 + \sum_{i_1,\dots,i_k} w_{i_1,\dots,i_k} \bar{\alpha}_{i_1} \cdots \bar{\alpha}_{i_k} + \varepsilon + \sum_{j=1}^{s} v_j \hat{\alpha}_{i_1} \cdots \hat{\alpha}_{i_d},$$

where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ denotes a feature selection vector which represents a feature subset in a n-dimension feature space. $\alpha_i = 1$ means that the ith feature belongs to the subset and $\alpha_i = 0$ means that the ith feature is not included in the subset. e_0 is a constant representing base error rate. $w_{i_1,\dots,i_k}$ is a value reflecting the error for eliminating features $i_1, \dots, i_k$. ε is used to simulate the training defects, which has a normal distribution $N(0, \varepsilon_0)$. The last item in the above model is called local defect, which is associated with specific subsets of features. In this item $\hat{\alpha}_i$ can stand for α_i, $\bar{\alpha}_i$, or a "don't care case". For more details refer to Ref. [9].

For the 2nd-form feature selection problem, we implemented tabu search as follows:

(1) Feature selection vector is represented by a 0/1 bit string. Initial solution is randomly generated.
(2) The neighborhood of a solution vector $\boldsymbol{x}$ is a set of solutions, which are generated through adding or deleting a feature on $\boldsymbol{x}$.
(3) Use all or randomly selected part of the neighborhood of the current solution as candidate solutions.
(4) Termination condition is a predefined number of iterations.

We implemented and tested the tabu search described above to solve a 30-dimension feature selection problem based on the above error rate model. The experimental results are shown in Table 1. Results of other algorithms such as SFS, SBS, GSFS, GSBS, PTA, SFFS, SBFS, GA, etc. are also shown in the same table for comparison.

The GA used in the experiments is implemented as follows:

(1) Randomly generate a certain number of chromosomes to construct an initial population set.
(2) The calculation method of fitness function is the same as that used in Ref. [9], that is

$$f(x_i) = (1 + \varepsilon) \max_{x_j \in \Pi} J(x_j) - J(x_i),$$

where x_i is a feature selection vector, Π is the population of feature selection vectors, ε is a small positive constant which assures that $\min f(x_i) > 0$, i.e., even the least fit solution is given a chance to reproduce.
(3) Crossover. Select chromosomes with a probability proportional to their fitness value, respectively, to build-up a mating set. Each chromosome can be selected more than once. Pair the chromosomes in the mating set stochastically, then exchange their genetic information at a randomly selected point called crossover point. The two produced offspring are accepted with a predefined probability called crossover rate.
(4) Mutation. Change at random each bit in each offspring with another predefined probability called mutation rate.
(5) Select the best chromosomes of the population size among the old population and the offspring to form the new population of the next generation.

The following conclusions can be drawn based on the empirical results:

(1) The computation cost of the SFS or SBS algorithms is much lower than that of other algorithms. However, the obtained feature subset is poor compared to other algorithms' results. *GSFS*(n) and *GSBS*(n) give more attractive results while n increases. Meanwhile, the computation cost increases exponentially to n. Moreover, the nesting effect remains in *GSFS*(n) and *GSBS*(n) as in SFS and SBS.
(2) The PTA method outperforms the SFS and SBS methods, but the results are still far from satisfactory.

Table 1
Performance of various algorithms on a 30-dimension data set[a]

Algorithm	Experiment times	Criterion function			Error rate (mean)	Number of selected features	Computation cost[b]
		Best	Mean	Worst			
SFS	1		13.89		0.214	12	465
GSFS(2)	1		13.76		0.195	14	2600
GSFS(3)	1		13.16		0.211	12	12,255
GSFS(4)	1		12.74		0.208	12	54,280
SBS	1		14.02		0.200	14	465
GSBS(2)	1		13.96		0.199	14	2600
GSBS(3)	1		13.52		0.182	14	12,255
GSBS(4)	1		12.93		0.199	13	54,280
PTA(2,1)	1		12.98		0.200	13	1333
PTA(3,1)	1		13.89		0.214	12	885
PTA(3,2)	1		12.98		0.200	13	2139
PTA(1,2)	1		13.03		0.210	12	1333
PTA(1,3)	1		13.03		0.210	12	885
PTA(2,3)	1		12.95		0.199	13	2139
SFFS	1		12.98		0.200	13	3063
SBFS	1		12.75		0.195	13	3345
GA[c]	20	12.41	12.88	13.24	0.198	$12 \sim 13$	3000
Tabu[d]	20	12.22	12.69	13.11	0.201	$12 \sim 13$	1800

[a]We tested a number of settings of parameters in GA and tabu search. The results listed above are comparatively good ones. In the Section 4 of this paper, the relationship between the parameters and the performance of tabu search will be discussed.
[b]The computation cost is measured by the number of evaluation times of the criterion function, since the cost of designing a classifier and estimating the relevant error rate is far more than that of a particular feature selection algorithm.
[c]The parameters in GA: population 60, crossover rate 100%, mutation rate 5%, generations 50.
[d]The parameters in tabu search: tabu list length 30, candidate set size 30, iteration 60.

(3) The floating methods SFFS and SBFS yield rather good results. Additionally, the efficiency of these algorithms is rather high. Note that the computation cost varies in different particular problem. However, each backtrack will contribute to give more attractive feature subset in certain level. The defect of these sequential floating methods is that they are still likely to trap into a local optimal solution even if the criterion function is monotonic and the scale of the problem is quite small. Fig. 1 shows an example of this defect, where SFFS will search along the path: $0001 \rightarrow 0011 \rightarrow 0111 \rightarrow 1111$ (shown by bold arrows), which is the same as the result of SFS. But "0011" is merely the third one of the best solutions with subset size 2.
(4) Under the condition of approximately equal computational cost with SFFS and SBFS, running GA 20 times, the best result is superior to SFFS and SBFS, and the average value of the criterion functions is near to that of SFFS and SBFS.
(5) During 20 times of running, the tabu search algorithm obtained the best solution among all algorithms tested (the criterion function value is 12.22). The average criterion function value is better as well. And the computational cost is much lower than that of GA, SFFS and SBFS.

Fig. 2 shows the variations of the criterion function value, the number of selected features and error rate in one run of tabu search. From the curves, it can be seen that the tabu search rapidly converged to the feasible/infeasible region border, where the size of feature vectors is between 13 and 15, error rate between 0.18 and 0.22, and intensification search are concentrated on the region where the optimal solution most likely exists. Therefore, tabu search is more efficient and shows good performance.

In order to test the performance of tabu search for large-scale feature selection, we also implemented the proposed tabu search method on a 60- and a 100-dimension feature selection problems based on the above error rate model. The experimental results are shown in Tables 2 and 3, respectively.

From Tables 2 and 3, we see that for 60- and 100-dimension feature selection problems, tabu search also obtained satisfactory results. During a few runs, tabu search method generally has a high possibility to find better solutions that cannot be found by other selection algorithms.

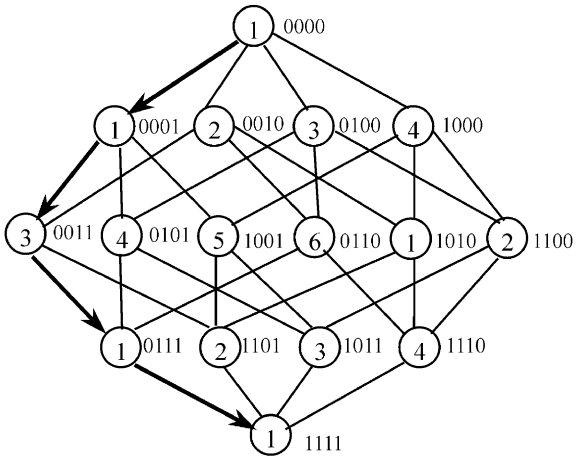

Fig. 1. An example showing that SFFS traps into local optimal solution. Each node represents a solution vector, and the 0/1 bit strings represent the solutions. Each digit within the circles represents the rank of the error rate in the same subset size level. SFFS will search along the path emphasized by bold arrows. SBFS's search path is the reverse of that of SFFS.

3.2. Feature selection based on Mahalanobis distance

In this section we use Mahalanobis distance as the objective function to solve the 1st-form feature selection problem. We have taken the same experiments as Kittler [6], Pudil [10] and Jain [7], in order to compare the performance of tabu search with that of sequential methods, sequential floating methods, GA, etc. The data set used is a 20-dimensional, 2-class set, and the 2-class conditional densities are Gaussian, with mean vectors μ_1 and μ_2, respectively and a common covariance matrix Σ. Under Gaussian class conditional densities, the probability of error is inversely proportional to the Mahalanobis distance. So the Mahalanobis distance can be used to assess the "goodness" of a feature subset. That is,

$$J_M = (\mu_1 - \mu_2)^{\mathrm{T}} \Sigma^{-1} (\mu_1 - \mu_2).$$

The Mahalanobis distance is monotonically with the feature subset, i.e.,

$$J_M(A \cup B) > J_M(A),$$

where J_M (A) is the Mahalanobis distance of the two classes under feature subset A. The monotonic relation between Mahalanobis distance and the probability of error can be represented as

$$J_M(A) > J_M(B) \Leftarrow \Rightarrow error(A) < error(B),$$

where $error(A)$, $error(B)$ denote the classifier's error rates using subset A and B, respectively.

Taking Mahalanobis distance as the criterion function, we use tabu search to solve the 1st-form feature selection problem, i.e., select an optimal feature subset of a certain number of features that yields the lowest error rate of a classifier. To make tabu search and GA suitable for this problem, we need some modification to them.

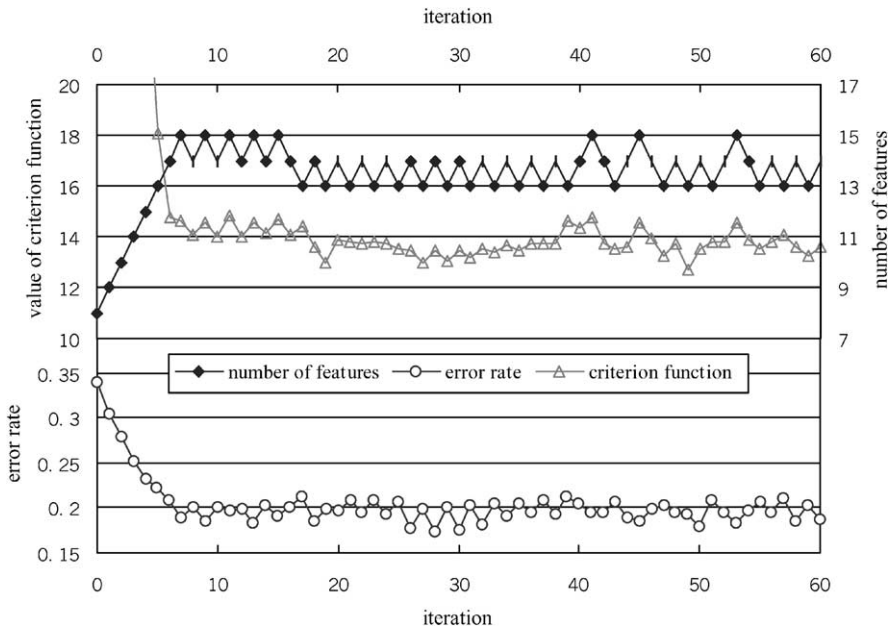

Fig. 2. Criterion function value, number of features selected and error rate in 1 run of tabu search.

Table 2
Performance of various algorithms on a 60-dimension data set

Algorithm	Experiment times	Criterion function			Error rate (mean)	Number of selected features	Computation cost[a]
		Best	Mean	Worst			
SFS	1		20.72		0.197	21	1830
GSFS(2)	1		18.90		0.199	19	19,375
SBS	1		20.52		0.193	21	1830
GSBS(2)	1		20.51		0.203	21	19,375
PTA(2,1)	1		18.58		0.195	19	37,820
PTA(1,2)	1		19.65		0.196	20	37,820
SFFS	1		18.03		0.200	18	11,225
SBFS	1		18.63		0.196	19	17,240
GA[b]	20	18.73	19.25	20.08	0.198	19 ∼ 20	7200
Tabu[c]	20	18.19	19.09	19.55	0.200	18 ∼ 20	7200

[a]The computation cost is measured by the number of evaluation times of the criterion function, since the cost of designing a classifier and estimating the relevant error rate is far more than that of a particular feature selection algorithm.
[b]The parameters in GA: population 120, crossover rate 100%, mutation rate 5%, generations 60 (without local search).
[c]The parameters in tabu search: tabu list length 120, candidate set size 60, iteration 120. Among 20 runs, the initial feature subsets are: one time is null set, one time the full feature set, and the others are randomly selected subsets.

Table 3
Performance of various algorithms on a 100-dimension data set

Algorithm	Experiment times	Criterion function			Error rate (mean)	Number of selected features	Computation cost[a]
		Best	Mean	Worst			
SFS	1		40.82		0.196	41	5050
GSFS(2)	1		38.97		0.199	39	87125
SBS	1		44.34		0.201	44	5050
GSBS(2)	1		38.85		0.197	39	87125
PTA(2,1)	1		38.89		0.200	39	171700
PTA(1,2)	1		39.05		0.200	40	171700
SFFS	1		38.58		0.197	39	31950
SBFS	1		38.49		0.196	39	43045
GA[b]	20	38.61	39.73	41.75	0.199	39 ∼ 42	20000
Tabu[c]	20	38.24	39.39	40.10	0.200	38 ∼ 40	20000

[a]The computation cost is measured by the number of evaluation times of the criterion function, since the cost of designing a classifier and estimating the relevant error rate is far more than that of a particular feature selection algorithm.
[b]The parameters in GA: population 200, crossover rate 100%, mutation rate 5%, generations 100 (without local search).
[c]The parameters in tabu search: tabu list length 200, candidate set size 100, iteration 200. Among 20 runs, the initial feature subsets are: one time is null set, one time the full feature set, and the others are randomly selected subsets.

Tabu search's implementation is modified as follows.

(1) The initial solution remains being generated randomly, but it must have exactly the required number of features.
(2) The objective function value of a feature subset equals to the Mahalanobis distance for this feature subset.
(3) The neighborhood of a solution is generated by removing a feature meanwhile adding another feature randomly.

Other steps are the same as that in Section 3.1.
The modification to GA is as below:

(1) All the initial solutions should have the required subset size.
(2) The Mahalanobis distance is used as the fitness function of the feature subsets.
(3) The crossover operation is implemented by randomly choosing the required number of features from the union set of the two mating chromosomes.
(4) Mutation should be modified as adding a feature and eliminating another feature simultaneously.

Table 4
Comparison of various algorithms using Mahalanobis distance as criterion function

Algorithm	Experiment times	Output distance	Mahalanobis
		$m = 10$	$m = 12$
BB	1	38.16	67.23
SFS	1	29.72	42.86
GSFS(2)	1	29.72	47.30
GSFS(3)[a]	1	29.27	49.62
SBS	1	30.00	41.39
GSBS(2)	1	30.00	62.92
GSBS(3)	1	30.00	41.39
SFFS	1	38.16	54.33
SBFS	1	34.51	62.92
GA[b]	20	38.06[c]	56.48[d]
Tabu[e]	20	38.16[f]	58.22[g]

[a] *GSFS*(3) cannot get the solution for $m = 10$ directly. Here *GSFS*(3) is first used to obtain a solution for $m = 9$, then take a step of SFS to get the solution for $m = 10$. The situation of *GSBS*(3) is similar.

[b] The parameters of GA: population 40, crossover rate 100%, mutation rate 25%, generations 30.

[c] Mean of 20 runs' results: 18 times among the 20 runs get the optimal solution whose Mahalanobis distance value is 38.16, 1 time obtains Mahalanobis distance of 36.77, another run gets Mahalanobis distance of 37.55.

[d] Mean of 20 runs' results; the largest Mahalanobis distance achieved is 67.23, the smallest value is 43.87. 7 times of the runs get the optimal value.

[e] The parameters of tabu search: tabu list length 40, candidate set size 40, iterations 30.

[f] All of the 20 runs get the optimal solution whose Mahalanobis distance value is 38.16.

[g] Mean of 20 runs' results: 7 times among the runs get the optimal solution whose Mahalanobis distance is 67.23, 1 time of the runs outputs Mahalanobis distance of 62.92, 11 times of the runs output Mahalanobis distance of 52.59, another run outputs 52.44.

In an initial 20-dimensional feature space, we tested 11 algorithms for selecting $m = 10$ and 12 features. The experimental results are shown in Table 4. We also use exhaustive search to get the Mahalanobis distance of all feasible solutions. The distribution of the Mahalanobis distances of all the solutions that include exactly 12 features is shown in Fig. 3. The positions of several algorithms' results are pointed out in Fig. 3 as well. Note that the best and the worst results during 20 runs of TS and GA are also shown.

The following conclusions can be drawn from the empirical results:

(1) The SFS and SBS algorithm have comparable performance. But their results are far from the optimal solution obtained by BB.

(2) At $m = 10$, *GSFS*(2), *GSFS*(3), *GSBS*(2), *GSBS*(3) do not outperform SFS and SBS while their computation cost increases considerably. For $m = 12$, *GSBS*(3) get the same solution as SBS. *GSFS*(2) and *GSFS*(3) get little better solutions, but these remain far from the optimal solution. Only *GSBS*(2) achieves a near optimal solution for $m = 12$. We can also see that *GSFS*(*n*), *GSBS*(*n*) do not always get a better solution with increasing *n*. The performance even degrades in some case.

(3) The SFFS, SBFS algorithms obtained rather satisfactory results for both $m = 10$ and 12. Specially, SFFS obtained the optimal solution for $m = 10$.

(4) The GA algorithm performs well for $m = 10$. 18 times of the 20 runs obtained the optimal solution. In the worst case, it still obtained a near optimal solution. For $m = 12$, 7 times of the runs GA reach the global optimal solution. But in the worst case, the result is comparable to that of SFS.

(5) For $m = 10$, the tabu search algorithm obtained global optimal feature subset in all 20 runs. For $m = 12$, the probability that TS obtained the optimal feature subset at each run is approximately equal to that of GA using the mentioned parameters, under such setting the two algorithm have comparable computation cost (tabu search is a little more efficient). The tabu search algorithm is superior to GA in that the worst result obtained by TS is noticeably better than what is given out by GA, the mean of the resulted criterion function of tabu search is higher than that of GA.

4. The effect of parameters in tabu search

As mentioned above, we have experimented a number of different parameter settings in tabu search. In this section, the effect of these parameters on the performance of tabu search will be discussed, and the best values obtained for them by our extensive parametric study are also analyzed.

4.1. Tabu list size

Tabu list embodies the short-term memory function for tabu search. It forces the algorithm not to reverse to the last *l* (tabu list size) moves. Therefore, the size of tabu list determines how many solutions one would like the search not to reverse back. A large tabu list size allows more diversification and forces the new solution to be a father point. On the contrary, a small tabu list size makes the search more intensive within the neighborhood of the current solution. In practice, we need to use an appropriate size of tabu list.

We tested different tabu list lengths varied from 2 to 90 (at interval of 2) for the 2nd-form feature selection

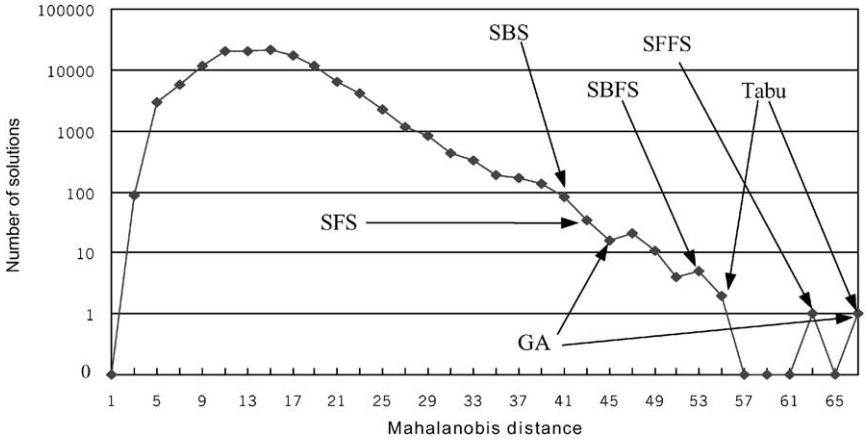

Fig. 3. The distribution of Mahalanobis distance of all the solutions that include 12 features and the positions of several algorithms' results. A point (x, y) in the curve indicates the number of solutions, whose Mahalanobis distance values are in the interval $[x-1, x+1]$.

Table 5
Comparison of different sizes of tabu list

Tabu list length	The number of tests which get the same solution as when tabu list size is 90	Criterion function (mean)
2	10	13.06
4	12	13.00
6	16	12.92
8	19	12.81
10	20	12.74
12	20	12.74
14	20	12.74
16 ~ 90	21	12.66

problem. The maximum number of iterations was set to 90. Each search process with different tabu list length started from a common initial solution. We repeated the test 21 times for each different initial solution. The results are listed in Table 5. Note that if the algorithm reverses to the best solution obtained previously, then stop the search since that would repeat the previous path.

From Table 5, we observed that as the tabu list size reaches a certain value, continuously increasing the tabu list size will not improve the performance further. In our experiments, the tabu search did not get better result when the tabu list size exceeds 16. Note that 16 is approximately the half of 30, the initial feature number.

The appropriate length of tabu list should be determined jointly with other parameters in tabu search. Additionally, the distribution of feasible solutions in particular problems should also be considered. Generally, it is suggested that the region of $[n/2, 3n]$ for tabu list size would be a good choice, where n is the number of initial features.

4.2. Candidate set size

This parameter controls the number of trial solutions to be generated from a current solution. We tested various values of the candidate set size for the 2nd-form problem of feature number 30, and tabu list size 30. The candidate set size was set to vary from 30 (whole neighborhood) down to 4. To make the computational cost approximately equal, as the candidate set size decreased, the number of iteration should increase properly. In one test, each search using different candidate set size started from the same initial solution, and we repeated this test 20 times with different randomly generated initial solutions. The results are given in Table 6.

From Table 6, we can see that the larger the candidate set size is, the better the mean objective function value is, because one has more choices to select from. Table 6 also shows that the larger the candidate set size is, the more frequently tabu list takes effect and some moves are tabu. This makes an intensification of the search. On the contrary, a small candidate set size makes the tabu list seldom take effect, and the search process becomes too arbitrary. This will decrease the performance of the tabu search.

4.3. Initial solution

We have tested the effect of different initial solutions on the resultant solution by parametric study. In the experiments the tabu list length was 30, the candidate set size 30, and the maximum number of iterations was 60. We compared the results obtained from random initial solutions with those from the better initial solutions, which are the results of other near optimal algorithms. The results are shown in Table 7.

Table 6
Comparison of different candidate set sizes

Candidate set size	Number of iterations	Output criterion function			Total times tabu list took effect (mean of 20 runs)	Mean of times tabu list took effect at each iteration
		Mean	Best	Worst		
30	60	12.69	12.22	13.11	33.8	0.554
15	120	12.73	12.33	13.05	32.6	0.271
8	225	12.87	12.12	13.65	30.3	0.134
4	450	13.13	12.61	13.76	33.6	0.074

Table 7
Comparison of different initial solutions

Initial solution	Experiment times	Criterion function of initial solution	Criterion function of resultant solution
Result of SFS	1	13.89	12.61
Result of SBS	1	14.02	12.12
Result of *PTA*(2,1)	1	12.98	12.77
Result of *PTA*(1,2)	1	13.03	12.36
Result of SFFS	1	12.98	12.77
Result of SBFS	1	12.75	12.75
Randomly generated	20	—	12.69 (mean)

From Table 7 we can see that the tabu search is not much sensitive to initial solutions in the experiments. When we took the near optimal solutions (the outputs of other algorithms) as the initial solutions, the average value of the criterion function is 12.56, a little better than that of the random initial solutions. In fact, when the initial solution is far from the optimal solution, the tabu search acts more like SFS or SBS at the beginning steps.

4.4. Neighborhood type

Recall that in Section 3.1 we have defined the neighborhood of a solution as those generated by alternating the status (inclusive/exclusive) of exactly one feature. We call such neighborhood as type A to distinguish from neighborhood type B defined as below.

A B-type neighborhood of a solution $\boldsymbol{x}$ is generated by reversing one or two features' states of $\boldsymbol{x}$. The B-type neighborhood can be divided into following five cases.

(1) $0 \to 1$: adding a feature.
(2) $1 \to 0$: removing a feature.
(3) $00 \to 11$: adding two features simultaneously.
(4) $11 \to 00$: removing two features simultaneously.
(5) $10 \to 01$: adding a feature meanwhile removing another feature.

Note that an A-type neighborhood is included in the B-type neighborhood.

Suppose the original feature space be n-dimension. The size of A-type neighborhood is n, and B-type $n(n+1)/2$. So when using neighborhood of type B, the computation cost will increase tremendously if we try to maintain high-ratio access to the neighborhood. However, decreasing the access ratio will make the function of tabu list degrade and the control over search direction becomes weak. Extremely, when the neighborhood becomes the power set 2^n, the tabu search will become random search. Based on the above analysis, the conclusion is naturally that tabu search using B-type neighborhood with low access rate will not perform well. The following experiments on the 1st-form of feature selection verified this analysis.

Let the neighborhood type be B, tabu list size 30, and the number of iterations 60. At each step, the next

Table 8
Comparison of two types of neighborhood

Neighborhood type	Experiment times	Output criterion function			Times when tabu list take effect		
		Mean	Best	Worst	Mean	Most	Least
A	20	12.69	12.22	13.11	33.8	49	27
B	20	13.02	12.32	13.64	2.2	4	0

move is selected from 30 solutions, which are randomly picked out from the B-type neighborhood of the current solution. Repeat this experiment 20 times. The results are listed in Table 8. To compare expediently, we also list the results using neighborhood type A in Table 8.

From the experimental results it can be seen that the tabu list seldom takes effect when the neighborhood is B-type. In fact, in 2 runs the move is never tabu. It becomes no different from the random search, and this makes the performance of tabu search worse.

5. Conclusion

In this paper, we have used tabu search to solve the two forms of optimal feature selection problem. The experimental results are very encouraging and show that the tabu search not only has a high possibility to obtain the optimal or a close to the optimal solution, but also requires less computational time than the branch and bound method and most other currently used suboptimal methods.

The branch and bound method is optimal, but the optimality of the results is constrained by the monotonicity of the feature selection criterion function. Another drawback is that, for problems with more features, the branch and bound method is still impractical, and the size of features it can tackle depends on the computer's performance.

Based on our experimental results, the sequential floating methods SFFS and SBFS are efficient and usually can find fairly good solutions. But they also suffer from trapping into local optimal solutions. We have demonstrated an example of this problem.

In this paper, our experiments are done on synthetic data. It will be a meaningful work to test and compare the performance and efficiency of tabu search with GA and SFFS (SBFS) on real data sets.

References

[1] T.M. Cover, J.M. Van Campenhout, On the possible orderings in the measurement selection problem, IEEE Trans. Systems Man Cybern. 7 (9) (1977) 657–661.
[2] P.M. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection, IEEE Trans. Comput. 26 (9) (1977) 917–922.
[3] I. Foroutan, J. Sklansky, Feature selection for automatic classification of non-Gaussian data, IEEE Trans. Systems Man Cybernet. 17 (1987) 187–198.
[4] T.M. Cover, The best two independent measurements are not the two best, IEEE Trans. Systems Man Cybern. 4 (2) (1974) 116–117.
[5] E. Backer, J.A. Shipper, On the max–min approach for feature ordering and selection, Proceedings of the Seminar on Pattern Recognition, Liege, 1977.
[6] J. Kittler, Feature set search algorithms, in: C.H. Chen (Ed.), Pattern Recognition and Signal Processing, Sijthoff and Noordhoff, Alphen aan den Rijn, The Netherlands, 1978, pp. 41–60.
[7] A. Jain, D. Zongker, Feature selection: evaluation, application, and small sample performance, IEEE Trans. Pattern Anal Mach. Intell. 19 (2) (1997) 153–158.
[8] W. Siedlecki, J. Sklansky, On automatic feature selection, Int. J. Pattern Recognition Artif. Intell. 2 (2) (1988) 197–220.
[9] W. Siedlecki, J. Sklansky, A note on genetic algorithm for large-scale feature selection, Pattern Recognition Lett. 10 (11) (1989) 335–347.
[10] P. Pudil, Novovicova, J. Kittler, Floating search methods in feature selection. Pattern Recognition Lett. 15 (11) (1994) 1119–1125.
[11] J. Sklansky, W. Siedlecki, Large-scale feature selection, in: C.H. Chen, L.F. Pau, P.S.P. Wang (Eds.), Handbook of Pattern Recognition and Computer Vision, World Scientific, Singapore, 1993, pp. 61–123.
[12] M. Kudo, J. Sklansky, Comparison of algorithms that select features for pattern classifiers, Pattern Recognition 33 (1) (2000) 25–41.
[13] F. Glover, Tabu search I. ORSA J. Comput. 1 (1989) 190–206.
[14] F. Glover, Tabu search II. ORSA J. Comput. 2 (1989) 4–32.
[15] F. Glover, M. Laguna, Tabu search, in: R.C. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill, Berkshire.

About the Author—HONGBIN ZHANG received the B.S. degree in Automation in 1968, and the M.S. degree in Pattern Recognition and Intelligent System in 1981, both from Tsinghua University, China. From 1986 to 1989 he was an invited researcher in Department of Information Science of Kyoto University, Japan. From 1993 to 1994 he was a visiting scholar of RPI, USA. Since 1993 he has been a professor of the Computer Institute of Beijing Polytechnic University, China. His current research interests include pattern recognition, computer vision, image processing and neural networks.

About the Author—GUANGYU SUN received the B.S. degree in Geology from Peking University in 1992, and the M.S. degree in Pattern Recognition and Computer Vision from Beijing Polytechnic University in 1999. His current interests include pattern recognition and computer vision.