Comparison of Algorithms that Select Features for Pattern Classifiers

Mineichi Kudo [a,1,2] and Jack Sklansky [b,2]

[a] *Division of Systems and Information Engineering, Graduate School of Engineering, Hokkaido University, Sapporo 060-8628, Japan*

[b] *Department of Electrical Engineering, University of California, Irvine, California 92697, USA.*

Abstract

A comparative study of algorithms for large-scale feature selection (where the number of features is over 50) is carried out. In the study, the goodness of a feature subset is measured by leave-one-out correct-classification rate of a nearest-neighbor (1-NN) classifier and many practical problems are used. A unified way is given to compare algorithms having dissimilar objectives. Based on results of many experiments, we give guidelines for the use of feature selection algorithms. Especially, it is shown that sequential floating search methods are suitable for small- and medium-scale problems and genetic algorithms are suitable for large-scale problems.

Keywords: Feature selection, Monotonicity, Genetic algorithms, Leave-one-out method, k-nearest neighbor method

1 Introduction

Feature selection in the design of pattern classifiers has three goals : (1) to reduce the cost of extracting features, (2) to improve the classification accuracy, and (3) to improve the reliability of the estimate of performance. In the design of segmenters of medical images or of remotely sensed aerial images, the initial set of candidate features ofthen consists of over one hundred features. Such a large number of features often includes many *garbage* features. Such features

[1] Corresponding author. e-mail:mine@main.eng.hokudai.ac.jp

[2] This work was carried out within the Japan-U.S. Cooperative Science Program of the U.S. National Science Foundation (NSF) and the Japan Society for the Promotion of Science (JSPS). The authors thank NSF and JSPS for their financial support. Part of this support was provided by NSF grant No. IRI-9123720.

are not only useless in classification, but sometimes degrade the performance of a classifier designed on the basis of a finite number of training samples. In such a case, removing the garbage features can improve the classification accuracy.

The choice of an algorithm for selecting features from an initial set Y depends on $|Y|$, the number of features in Y. We say that the feature selection problem is small-scale, medium-scale, or large-scale if $|Y|$ belongs to $[0, 19]$, $[20, 49]$, or $[50, \infty]$, respectively.

A large number of algorithms have been proposed for feature selection and some comparative studies have been carried out [1–5]. However, many comparative studies do not treat large-scale problems or nonmonotonic problems where an addition of a feature can degrade the classification accuracy. Some studies adopt a monotonic criterion like Mahalanobis distance, but this monotonicity is satisfied only when the optimal Bayes classifier is used – an impractical hypothesis. Pudil *et al.* [4] included nonmonotonic criteria in comparison of large-scale problems, but they compared only a few algorithms.

In comparing feature selection algorithms, we place them in three groups according to what is optimized. In one group, the algorithms find the feature subset of a specified dimensionality in which the classes of data are most discriminable. In a second group, the algorithms find the smallest feature dimensionality for which the discriminability exceeds a specified value. In a third group, the algorithms find a compromise between a small subset of features and class discriminability.

To ensure that our conclusions and recommendations are realistic, we tested the feature selection algorithms on real, rather than synthetic data. These data included mammograms, synthetic aperture radar images, numerical shape descriptors of mushrooms, and shape descriptors of motor vehicles. Our estimates of discriminability were based mainly on leave-one-out correct-classification rate of nearest-neighbor (1-NN) classifiers.

We developed a unified method of comparing selection algorithms with different objectives. We describe this method in Section 3.

2 Algorithms

Here we describe our terminology.

Feature Sets: The initial feature set is denoted by Y, $|Y| = n$ and a selected feature subset by X. Unless mentioned otherwise, we find the best X of size m. X_i denotes

a feature subset of size i.

Criterion: A criterion function $J(X)$ evaluates the goodness of subset X on the basis of the ability of a classifier to discriminate the classes in the feature space represented by X. A larger value of J indicates a better feature subset.

We list the algorithms treated in this paper in Table 1, along with their time complexities, the objective types and the search types. Every algorithm is classified into one of the three objective types. For Objective Type A, the algorithm finds the subset of a given size for which J is a maximum. For Objective Type B, the algorithm finds the smallest subset for which J is not less than a specified value, J_0. For Objective Type C, the algorithm finds a compromise between Objective Types A and B. This compromise is found by minimizing a penalty function of $|X|$ and J.

Table 1
Types and time complexity of feature selection algorithms. The objective types are (A) to find the best subset of a given size, (B) to find the smallest subset satisfying a condition, (C) to find a subset whose combined size and error rate are optimal. The search types are (S) Sequential or (P) Parallel.

Algorithm	Time Complexity	Objective Type	Search Type
SFS, SBS	$\Theta(n^2)$	A	S
GSFS(g), GSBS(g)	$\Theta(n^{g+1})$	A	S
PTA(l, r)	$\Theta(n^2)$	A	S
GPTA(l, r)	$\Theta(n^{\max\{l+1, r+1\}})$	A	S
SFFS, SBFS	$O(2^n)$	A	S
BAB, BAB$^+$,BAB^{++}	$O(2^n)$	A	S
RBAB, RBABM	$O(2^n)$	B	S
GA	$\Theta(1)$ ($\Theta(n)$)	C	P
PARA	$\Theta(n)$ ($\Theta(n^2)$)	C	P

In Table 1, $\Theta(\cdot)$ denotes a tight estimate of complexity (exact except for a multiplicative constant) and $O(\cdot)$ denotes an estimate of complexity for which only an upper bound is known. Time complexities under a typical setting of parameters are shown in parentheses. These time complexities are only a clue when we use these algorithms. In many practical situations, some algorithms are carried out faster than their estimates and others are not. For example, BAB is the fastest in some problems in spite of its time complexity of $O(2^n)$ and GA consumes time in proportion to the number of generations and also to the population size.

We summarize these algorithms below. Some algorithms are improved at some points in this paper. Such points and the new algorithms are marked by '†'.

The first six algorithms are found in Reference (1).

SFS,SBS:

GSFS(g) SFS selects the best single feature and then the best pair including the best

GSBS(g)

single, and so on. SBS is the backward version. These algorithms are generalized to GSFS(g) and GSBS(g) in such a way that g features are evaluated at the same time and the best g-feature subset is chosen for addition or deletion in the algorithms.

PTA(l, r): Plus-l take-away-r algorithm. Go forward l stages (by adding l features

GPTA(l, r)

by SFS) and go backward r stages (by deleting r features by SBS) and repeat this process. In the generalized algorithm (GPTA(l, r)), GSFS(l) and GSBS(r) are used instead of SFS and SBS in inclusion and exclusion stage, respectively.

SFFS,SBFS: The floating version of PTA(l, r). Unlike PTA(l, r), SFFS can backtrack unlimitedly as long as the backtrack finds a better feature subset than the feature subset obtained so far at the same size (Pudil, Novovičová and Kittler [3]). SBFS is the backward version.

BAB,BAB$^+$:

BAB^{++}†

BAB(s) The branch and bound methods. BAB is the original algorithm proposed

BAB$^+$(s)

BAB^{++}(s)†

by Narendra and Fukunaga [6] and BAB$^+$ is an improved algorithm by Yu and Yuan [7]. Both methods give the optimal solution, as long as the criterion function J is monotonic. BAB(s) is a faster but suboptimal version of BAB [6] (also BAB$^+$(s) for BAB$^+$ and BAB^{++}(s) for BAB^{++}). In a feature subset X of size $t(\geq m)$ under search, BAB uses the best criterion value obtained so far at size m for cutting the branches below X, while BAB(s) uses the best criterion value obtained so far at size $t-s$ (if $t-s \leq m$ then it behaves like BAB), where s is a *look-ahead* parameter. In this study, we use $s = 0, 1, 2$. In these algorithms, the efficiency depends on how fast they find a better threshold for cutting branches. Therefore, in BAB^{++}†, we improved BAB$^+$ so that they can take an initial threshold θ_0 externally before staring the algorithm. This initial threshold θ_0 is useful until the algorithm finds a better threshold than it. How to give θ_0 is described in Section 4.4.1.

RBAB
RBABM†
: The relaxed branch and bound methods. These algorithms can find the optimal solution even if the monotonicity of J is damaged a little. Unlike BAB, RBAB (Foroutan and Sklansky [8]) aims to find the smallest subset for which the criterion value is not under a given threshold θ and the search is carried out for subsets with the criterion values over or equal to $\theta - \delta (\delta > 0)$, where δ is called a *margin*. RBABM† does not use margin δ anymore. Instead, RBABM cuts branches below X only when both X and a parent of X are under θ.

GA: The genetic algorithm. Many studies has been done on GA for feature selection (for example, see [9,10]). In GA, a feature subset is represented by a binary string with length n, called a *chromosome*, with a zero or one in position i denoting the absence or presence of feature i. Each chromosome is evaluated in its fitness through an optimization function in order to survive to the next generation. A population of chromosomes is maintained and evolved by two operators of crossover and mutation. This algorithm can be regarded as a parallel and randomized algorithm. The initial population is arbitrary. So, we discuss how to choose the initial population†.

PARA†: A parallel algorithm. This algorithm maintains a population of N feature subsets as the same as GA but it updates the population only by local hill-climbing, that is, the population for next generation is made of N best feature subsets of all unvisited immediate above supersets and immediate below subsets of the current N subsets.

2.1 Practical Implementation of GA and PARA

Here, we describe in detail the implementation of GA and PARA.

The following function has been often used in GA [11]:

$$O(X) = -|X| - \frac{\exp\left((\theta - J(X))/\alpha\right) - 1}{\exp(1) - 1}, \tag{1}$$

where α and θ are constants specified according to the problem (usually $\alpha = 0.01 J_{\max}$, where $J_{\max}$ is an estimated upper bound of J). This optimization function requires the answer subset X to satisfy $J(X) \geq \theta$ first and then to be as small as possible, because the exponential function penalizes heavily for subsets not satisfying this first requirement. In this study, however, we take more straight forward optimization functions in order to make GA be comparable with Objective Type A and Objective Type B algorithms.

First we estimate an upper bound $J_{\max}$ and a lower bound $J_{\min}$ in the criterion function through a preliminary feature selection described later. The following

optimization function is in accord with Objective Type A algorithms:

$$O_A(X) = \begin{cases} J(X) - \epsilon|X| \ (|X| \leq m) \\ J_{\min} - \epsilon|X| \quad (|X| > m) \end{cases} \tag{2}$$

where $\epsilon = \frac{\beta}{n}(J_{\max} - J_{\min})$ ($\beta = 0.01$ in this paper) and is introduced in order to make the second term work only when two subsets have almost the same criterion values.

The following optimization function is in accord with Objective Type B algorithms:

$$O_B(X) = \begin{cases} -|X| + (J(X) - J_{\min})/(J_{\max} - J_{\min} + \epsilon) \ (J(X) \geq \theta) \\ -n + (J(X) - J_{\min})/(J_{\max} - J_{\min} + \epsilon) \quad (J(X) < \theta) \end{cases}, \tag{3}$$

where ϵ is an arbitrary small positive constant and θ is a threshold. It is noted that $0 \leq (J(X) - J_{\min})/(J_{\max} - J_{\min} + \epsilon) < 1$ and thus a superset X' of X never takes a larger criterion value as long as both X' and X hold $J(X'), J(X) \geq \theta$.

We use both O_A and O_B in GA and PARA for comparison with Objective Type A and Objective Type B algorithms. In addition we use $O_C = J(X)$ in order to find the optimal solution in J.

In GA, a difficulty is in the setting of parameters. GA has mainly four parameters to be set: the population size N, the maximum number of generations T, the probability of crossover p_c, and the probability of mutation p_m. In addition, there is arbitrariness in an initial population of chromosomes. We determined these values on the basis of the results of many experiments using artificial data. In this study, as in most problems, we set $N = 2n$ and $T = 50$. The resulting complexity of GA is $\Theta(n)$. We use mainly two sets of values of (p_c, p_m): $(0.8, 0.1)$ and $(0.6, 0.4)$. In addition, we use the following two types of initial populations of chromosomes: (P1) $2n$ extreme feature subsets consisting of n distinct 1-feature subsets and n distinct $(n-1)$-feature subsets and (P2) $2n$ feature subsets in which the number of features is in $[m-2, m+2]$ and all features appear as evenly as possible, where m is the desired number of features and n is the original number of features. A setting (P1) is denoted by $\{1, n-1\}$ and (P2) by $[m-2, m+2]$. Thus GA is specified by a six-tuple of (objective type (O_A, O_B or O_C), N, T, initial population type ($\{1, n-1\}$ or $[m-2, m+2]$), p_c, p_m).

PARA has also the two parameters of population size and the maximum number of generations. We used $N = 2n$ and $T = 50$, as in GA. PARA is specified by a triplet of (objective type (O_A, O_B or O_C), N, initial population type ($\{1, n-1\}$ or $[m-2, m+2]$)). Some parameters are omitted if these values

are obvious from the context. PARA is carried out until its population is not better than its previous population.

2.2 Criterion Function and Monotonicity

In feature selection, the selection of the criterion function $J(X)$ is very important. If we know which classifier will be used in the problem under consideration, the best criterion is, in general, the correct recognition rate of the classifier for infinitely many samples in feature space X. However, it is very difficult to estimate the correct recognition rate (or the error rate) of a classifier on the basis of a limited number of training samples. This is one reason why the previous comparative studies on feature selection used Mahalanobis distance, which gives an upper bound of the Bayes error rate with *a priori* probabilities of classes. Such a parametric estimation is not practical in many problems. In this study we estimate the correct recognition rate by the *leave-one-out technique*, where a training sample is used as a test sample, a classifier is constructed by the remaining training samples, and the test sample is classified by the constructed classifier. This procedure is repeated until all training samples are classified and the results are summed up. If the number of the training samples is large enough, we use the *d-fold cross validation technique* where each of d nonoverlapping similar-sized subsets is used in place of a training sample in the leave-one-out technique. In this technique we usually use the nearest neighbor classifier (1-NN) because of its good asymptotic property in large training samples. It is known that $\epsilon_{NN} < 2 \cdot \epsilon_{Bayes}$ when the number of training sample is sufficiently large. (We define ϵ_{NN} as the error rate of the 1-NN classifier, and ϵ_{Bayes} as the error rate of the Bayes optimum classifier.) Another reason for using 1-NN is its ease of implementation with the leave-one-out technique.

If we use near-Bayes-optimal classifiers, the correct recognition rate is likely to be nearly monotonic. To exploit this property, Foroutan and Sklansky [8] introduced the concept of *approximate monotonicity* and proposed a *relaxed branch and bound method*, RBAB. RBAB chooses the smallest feature subset from feasible solutions of feature subsets for which J exceeds a threshold θ. The search is continued for feature subsets for which J exceeds $\theta - \delta (\delta > 0)$, where δ is called a *margin*. Thus, within a range of margin δ, J is permitted to violate its monotonicity and RBAB finds the optimal solution. However, there still remains the problem of determining the value of δ.

We propose another version of the branch and bound method, called RBABM. A criterion function J is said to be *k-monotonic* if $X' \subset X, |X| - |X'| \geq k \Rightarrow J(X') \leq J(X)$ for every X', X. BAB does the optimal search only when J is 1-monotonic and RBABM does so even when J is 2-monotonic. That is,

RBABM permits a violation of monotonicity between parents and children, but requires monotonicity between grandparents and grandchildren.

3 Preliminary Feature Selection

The result of an algorithm with a sequential search can be represented by a curve, called a *criterion curve*, connecting points of $(m, J(X_m))$ obtained by the algorithm at size $m(m = 1, \cdots, n)$. Many comparative studies [2,3,12] compared algorithms by the criterion curves. However, our concern is in only a part where the criterion value does not degrade so much. In addition, it is difficult to compare algorithms with different objective types in criterion curves.

To cope with these difficulties, we propose preliminary feature selection. This approach was inspired by Sklansky and Siedlecki [11]. With an algorithm with a low time complexity like SFS and SBS, we execute a preliminary feature selection in order to get a criterion curve. Then we classify the problem under consideration into one of three cases: a monotonic case, an approximate monotonic case or a nonmonotonic case. In addition, the values of θ and δ for RBAB and RBABM are determined from the criterion curve(s).

If the problem is judged as monotonic or approximate monotonic, we determine a parameter $\alpha(= 1\%, 5\%)$ showing the degree of degradation and find the point that is degraded with α as compared with the maximum criterion value $J_{\max}$ (Fig. 1(a),(b)). From this α-degradation point, we determine a criterion value J_α as a threshold and the corresponding number of features m_α as a desired number of features.

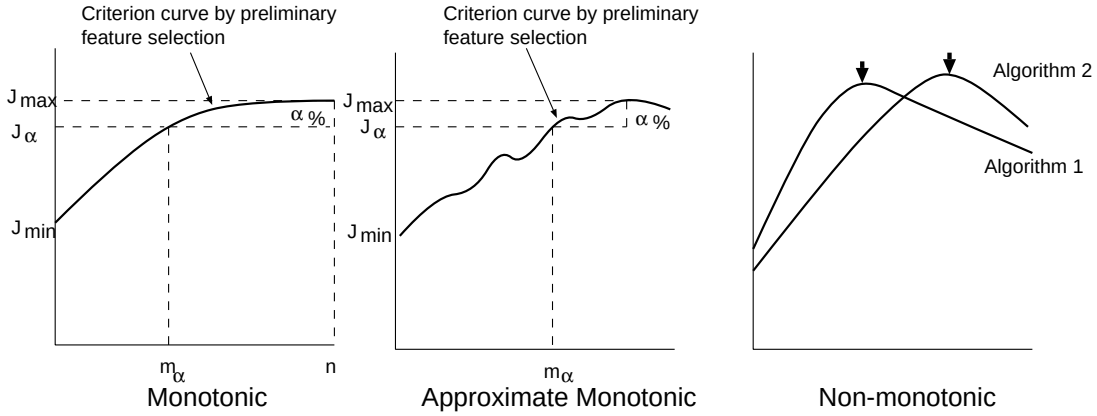


Fig. 1. Classification of problems

The value of m_α is passed to Type-A algorithms and the value of J_α is passed to Type-B algorithms as threshold θ. For Type-C algorithms, both values

are passed to their optimization functions O_A and O_B. In addition, an upper bound $J_{\max}$ and a lower bound $J_{\min}$ in J are read from the criterion curve and used in O_A and O_B.

If the problem is judged as nonmonotonic, we use only Type-A and Type-C algorithms. Type-A algorithms are carried out so as to draw their criterion curves in such a way that an algorithm with a forward search is required to find the best $(n-1)$-feature subset and an algorithm with a backward search is required to find the best 1-feature subset. In the process of this search, we can have a criterion curve. Then we choose the optimal subset in the criterion curve as the answer (Fig. 1(c)). For Type-C algorithms, we use $O_C = J(X)$ as the optimization function for obtaining the optimal subset in the criterion.

4 Experimental Results

4.1 Datasets

We examined eight different problems consisting of one synthetic data and five real data (Table. 2). Some datasets indicated by (small) are datasets in which some features are preselected by a method SUB that is a revised version of the method of Kudo and Shimbo [13]. SUB was devised for *classifier-independent feature selection* [14]. Unlike the classifier-specific algorithms discussed in this paper, classifier-independent feature selection algorithms do not need a criterion function J and aim to select a feature subset for which any kind of classifier is improved in its discriminability compared with when the initial feature set is used. A comparative study of classifier-independent algorithms is shown in [15].

The problem type is determined by the preliminary feature selection with SFS and SBS.

SAR: Synthetic aperture radar satellite image.
We tested a SAR image with 28561(=169x169) pixels. There are three classes corresponding to different landmarks (urban area, runway and agricultural area). A pixel is characterized by ten texture features. All pixels are labeled by hand. One percent of all pixels are extracted randomly in proportion to the population size of classes.

Vehicle: Vehicle data [16].
The task is to classify a given silhouette as one of four types of vehicle, using 18 features extracted from the silhouette. From the preliminary feature selection, we determined $\alpha = 5\%$ and $J_\alpha = 0.821, m_\alpha = 9$. For RBAB, $\delta = 0.005$ is used.

Table 2
Experimental data. n: # of features, M : # of classes, K: # of training samples per class, J: criterion function ((L): leave-one-out, (k-CV): k-fold cross validation).

Database	n	M	K	J	Problem type
SAR	10	3	131, 61, 93	(L) 1-NN	nonmonotonic
Vehicle	18	4	199-218	(9-CV) linear classifier	app. monotonic
Mammogram(small)	19	2	57 and 29	(L) weighted 5-NN	app. monotonic
Kittler	20	2	1000 each	Mahalanobis	monotonic
Mushroom(small)	29	2	500 each	(L) 1-NN	monotonic
Sonar(small)	40	2	111 and 97	(L) 1-NN	app. monotonic
Sonar(large)	60	2	111 and 97	(L) 1-NN	nonmonotonic
Mammogram(large)	65	2	57 and 29	(L) weighted 5-NN	nonmonotonic

Mammogram: (large) A mammogram database.

The database is a collection of 86 mammograms from 74 cases which are gathered from University of California, San Francisco (UCSF), the Mammographic Image Analysis Society (MIAS), and the University of California, Los Angels (UCLA). Chosen 65 features are of 18 features characterizing calcification (number, shape, size, etc.) and of 47 texture features (histogram statistics, Gabor wavelet response, edge intensity, etc.).

There are two classes of benign and malignant (57 and 29 samples, respectively). For this problem, we penalize the error of misclassifying malignant as benign heavily more than the reverse error. We, thus, calculated a criterion value by multiplying 10/11 and 1/11 to these two kinds of errors obtained by the leave-one-out method. In addition, to meet the 5-NN method to this special requirement, we weighted malignant by three such that a neighborhood of malignant is counted as three neighborhoods. In this modification, if two of five neighborhoods are malignants, the unknown sample is judged as malignant by $2 \times 3 > 3$.

Mammogram: (small) A mammogram database.

This data is the same data with Mammogram(large) data. The 19 features are preselected from original 65 features by SUB. We used $J_\alpha = 0.97$ and $m_\alpha = 13$ for $\alpha = 1\%$. For RBAB, $\delta = 0.06$ is used.

Kittler: Kittler's synthetic data used in many references [1,3,12].
There are two classes with 20 features. The two distributions of samples are normal distributions with a common covariance matrix Σ and different mean vectors μ_1 and μ_2. As the criterion function J, we used Mahalanobis

distance defined by $(\mu_1 - \mu_2)^t \Sigma^{-1} (\mu_1 - \mu_2)$ that is equivalent to the Bayes correct recognition rate in this problem. We used $J_\alpha = 4.75$ and $m_\alpha = 16$ for $\alpha = 5\%$.

Sonar(large): A sonar database [16].
The task is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock using 60 features of which each describes the energy within a particular frequency band, integrated over a certain period of time. The database consists of 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions and 97 patterns obtained from rocks under similar conditions.

Sonar(small): A sonar database.
This data is the same as above data. The 40 features are preselected from 60 by SUB. We used $J_\alpha = 0.9$ and $m_\alpha = 32$ for $\alpha = 1\%$.

Mushroom: (small)
A mushroom database [16].

The task is to assess the edibility of a large mushroom samples. There are two classes of edible and poisonous and with 4208 and 3916 samples, respectively. Taken 22 categorical features include cap-shape, odor, gill-color, etc. They were converted to 125 numeral features in such a way that a categorical feature with k possible category values was converted to k numeral features. For example, in a categorical feature of gill-size with "broad" or "narrow", "broad" is converted to two numeral features (1, 0) and "narrow" is to (0, 1). From these 125 original features, 29 features are preselected by SUB. We chose 1000 samples (500 each) randomly.

4.2 Evaluation Method

In this study, we used two kinds of graph in comparison of results: (G1) a graph of m vs. $J(X_m)$, (G2) a graph of the evaluation number of J vs. $J(X)/(J_u - J_u) - |X|/(n-1)$, where J_l and J_u are determined depending on problems in order to account the number of features in addition to the criterion value $J(X)$. In (G2), thus, the goodness is evaluated by lines with slope $(J_u - J_l)/(n-1)$.

4.3 Setting of GA and PARA

Used values of some parameters in GA and PARA are summarize in Tables 3 and 4.

Table 3
Parameters of GA. O: optimization function, N: population size, I :initial population type, T : maximum # of generations, (p_c, p_m): crossover probability and mutation probability, K : # of trials in each parameter set. '/' means "or".

Database	O	N	I	T	(p_c, p_m)	K
SAR	O_C	20	[2,6]/{1,9}	50	(0.6,0.4)/(0.8,0.1)	3
Vehicle	O_A/O_B	36	[8,12]/{1,17}	50	(0.6, 0.4)/(0.8,0.1)	1
Mammogram (small)	O_A/O_B	38	[12,16]/{1,18}	50	(0.6, 0.4)/ (0.8,0.1)	1
Kittler	O_A/O_B	40	[14,18]/{1,19}	50	(0.6, 0.4)/ (0.8,0.1) /(0.8,0.02)	1
Mushroom(small)	O_A/O_B	58	[1,5]/1,28	50	(0.6, 0.4)/ (0.8,0.1)	1
Sonar(small)	O_A/O_B	80	[30,34]/{1,39}	50	(0.6, 0.4)/ (0.8,0.1)	1
Sonar(large)	O_C	120	[18,22]/{1,59}	50	(0.6,0.4)/(0.8,0.1)	3
Mammogram(large)	O_C	130	[17,21]/{1,64}	50	(0.6, 0.4)/(0.8,0.1)	3

Table 4
Parameters of PARA. O: optimization function, N: population size, I :initial population type, K : # of trials in each parameter set. '/' means "or".

Database	O	N	I	K
SAR	O_C	20	[2,6]/{1,9}	3
Vehicle	O_A/O_B	36	[8,12]/{1,17}	3
Mammogram (small)	O_A/O_B	38	[12,16]/{1,18}	3
Kittler	O_A/O_B	40	[14,18]/{1,19}	1
Sonar(small)	O_A/O_B	80	[30,34]/{1,39}	1
Sonar(large)	O_C	10/120	[18,22]	3
	O_C	120	{1,59}	3
Mammogram(large)	O_C	10/130	[17,21]	3
	O_C	130	{1,64}	3

4.4 Results and Discussion

All results are shown in Figs. 2– 10. We summarize these results in Tables 5 and 6.

Next, we examine the results in detail on individual topics.

Table 5
Comments to Aid Choice of Algorithm

Algorithm	Comment
SFS, SBS:	are fast, but are low in performance. They are useful for the preliminary feature selection (see Section 3).
GSFS(g), GSBS(g):	are very time-consuming. Their performance is a little better than SFS and SBS. They are effective when the desired number of features is very small or very close to the number of initial features.
PTA(l, r):	is effective when we want to obtain better solutions than SFS and SBS in a moderate time.
GPTA(l, r):	shows almost the same high performance as SFFS and SBFS, but is much more time-consuming than SFFS and SBFS.
SFFS, SBFS:	are very effective for Objective Type A. Their computation time is admissible for small-scale and medium-scale problems. For Objective Type C in large-scale problems, they are somewhat inferior in performance to GA and consume much more time than GA.
BAB^{+},BAB^{++}:	are very effective for Objective Type A in small-scale and medium-scale monotone problems, but often require a lot of time.
$BAB^{+}(s)$,BAB^{++}(s):	are faster than the optimal BABs but are lower in performance. We can use these algorithms before the optimal BABs, increasing the value of s as $0, 1, 2, \ldots$
RBAB, RBABM:	are very effective for Objective Type B in monotone or approximate monotone problems. They can be used for small-scale and medium-scale problems, but sometimes require an excessive time. It may be useful to terminate the algorithms by a given number.
GA:	is very useful for Objective Type C in large-scale problems because of its low computation time. Almost all large-scale problems in the real world become nonmonotonic due to garbage features. For such problems, GA would be appropriate. GA can also available for Objective Types A and B, but takes much more time than the other selection algorithms in small-scale and medium-scale problems.
PARA:	is inferior in performance to GA and takes much more time than GA for Objective Type C. With a small size of population, it can be used for Objective C in large-scale problems. For Objectives A and B, it has a possibility to find better solutions than the other algorithms.

Table 6
Recommended Algorithms for Feature Selection

Objective Type	Scale of Initial Set of Features								
	Small ($n < 20$)			Medium ($20 \leq n < 50$)			Large ($50 \leq n \leq 100$)		Very Large ($n > 100$)
	Mono.	Approx. Mono.	Non Mono.	Mono.	Approx. Mono.	Non Mono.	Mono. or Approx. Mono.	Non Mono.	
A	BAB^{++}	SFFS SBFS	*	BAB^{++} BAB$^{++}(s)$	SFFS SBFS GA	*	SFFS SBFS GA	*	GA
B	RBAB RBABM		*	RBAB w. termination RBABM w. termination GA		*	GA	*	GA
C	*	*	GA SFFS SBFS	*	*	GA SFFS SBFS	*	GA	GA

*: rarely occurs

4.4.1 Efficiency of BAB algorithms

The branch and bound method, BAB, is equivalent to the exhaustive search method when the criterion function is monotonic, but too time-consuming. So, we compared several variants of the branch and bound methods on Kittler's data. First, we compared BAB, BAB^{+} and BAB^{++}. As an initial threshold θ_0 for BAB^{++}, we adopted the better one of solutions obtained by SFS and SBS at each size $m (m = 1, \ldots, n-1)$. The result is shown in Fig. 3. As comparison, the results of SFFS and SBFS are also shown. In Fig. 3, the result of BAB^{++} does not include the number of evaluation consumed by SFS and SBS, because they were carried out already in the preliminary feature selection.

BAB^{+} is very efficient as compared with the original BAB especially when the desired number of features m is relatively small. In addition, giving an initial threshold θ_0 to BAB^{+}, that is, BAB^{++}, contributes a little in efficiency (Fig. 3). Since such an initial threshold is useful until the algorithm finds a new better threshold, it would work well especially when there are many possible solutions, that is, when $\binom{n}{m}$ is large, thus, when n is large or m is close to $n/2$.

Next we examined the suboptimal version of BAB^{++}, BAB$^{++}(s)$ ($s = 0, 1, 2$). The look-ahead parameter s shows how depth the algorithm looks ahead. The larger value of s the algorithm uses, the closer it approaches the optimal solution. BAB$^{++}(n-m-1)$ is equivalent to BAB^{++}. The results are shown in Figs. 3(a) and (b). In the criterion, BAB$^{++}(1)$ and BAB$^{++}(2)$ are sufficiently close to the optimal curve and are faster than BAB^{++}.

In conclusion, in this problem, BAB^{+} and BAB^{++} outperforms other algorithms in our concerning area ($12 < m < 20$) in both performance and speed (Fig. 3). Also in vehicle data with 18 features and in mushroom(small) data with 29 features, BAB^{++} was comparable in speed with the other algorithms and finds near-optimal solutions, although some other algorithms found the same solutions (Figs. 8 and 9). Its suboptimal version BAB$^{++}(s)$ was inferior in performance to BAB^{++} but faster. One possible use of BAB$^{++}(s)$ is to carry out it in advance of BAB^{++}, increasing the value of s one by one from 0. Then, we can have some solutions in an acceptable time.

For approximate monotonic problems, RBAB and RBABM are expected to be effective. We had solutions only in mammogram(small) and vehicle data in a reasonable time. The results of mammogram(small) are shown in Fig. 6. From these figures, we can see that RBAB and RBABM succeeded to find smaller feature subsets than those of other algorithms, without a large degradation of performance. Especially RBAB could find the smallest subset with 6 features. This best solution cannot be obtained by SFFS and SBFS even if they are carried out in every number of features (Fig. 6(d)). RBAB worked well even in vehicle data, while GA also gave the same answer.

RBAB and RBABM require a tremendous number of evaluations (Figs. 6(c) and 8(b)). RBAB evaluated 147195 subsets (28 % of possible $2^{19}-1$ subsets) and RBABM evaluated 31143 (6 %) in mammogram(small) data, respectively, and RBAB evaluated 92380 subsets (35 % of possible $2^{18}-1$ subsets) in vehicle data. However, they found their answers already in the first 34586 and 6331 evaluations in mammogram(small) data, respectively, and 4709 evaluations in vehicle data (RBAB only). Therefore, a possible efficient use of RBAB and RBABM is to terminate them after a special number of evaluations.

4.4.2 Sequential Algorithms

In this section, we restrict our discussion to sequential algorithms. When using a sequential algorithm, both the forward and backward forms of the algorithm should be used at the same time. Usually backward algorithms are better than their counterparts. However, this depends on the specific problem (for example, SFFS is better than SBFS in sonar(large) and vehicle data, see Figs. 5(a) and 8(a)). In addition, it seems that forward algorithms are better than back-

ward algorithms in the case of m is very small (say, $m < 5$) and the inverse holds in the case m is near n (say, $n - m < 5$). However, this is not true. We can see two counterexamples in mushroom(small) data and sonar(small) data (Figs. 9 and 10). Therefore, both directions should be examined at the same time.

The sequential forward and backward floating search algorithms (SFFS and SBFS) are known to be effective for many problems [2–5]. Indeed, these algorithms found fairly good solutions in a moderate time in our experiments. However, it cannot be said that they are always better than the others. SBFS or SFFS sometimes failed to find good solutions (see Figs. 2, 6, 8 and 10), but at least one of them worked well in almost all datasets. On average, GPTA(1,2) was the best in performance as a single sequential algorithm, while GPTA(1,2) was very time-consuming. Overall, SFFS and SBFS found fairly good solutions in a shorter time than the other sequential algorithms, if both algorithms are used at the same time.

4.4.3 GA and PARA

We describe the sensitivity of GA with respect to some of its parameters.

(i) The values of p_c and p_m are crucial. We tested the three sets of these parameters $(0.6, 0.4), (0.8, 0.1)$ and $(0.8, 0.02)$, mainly the first two. In our experiments, pair (0.8, 0.1) worked best through all experiments, but this seems to depend on the problems. We recommend to use several sets of parameters and to use not too small a value of p_m ($p_m > 0.02$) to avoid being captured in local maxima.
(ii) We tested two kinds of initial populations, (P1) and (P2). The difference is not large. When (P1) is used, GA showed a tendency to find larger subsets as compared with when (P2) is used. We recommend a large value of p_m (say, $p_m \geq 0.1$) if (P1) is used.
(iii) Since GA is a randomized algorithm, it can produce different solutions among different trials for the same parameter set and the same initial population. Our results show that this variability in solutions is very small. Our GA produced almost the same solutions for different trials (see, Figs. 4(b), 5(b) and 7(a)).

Next, let us discuss the performance of GA and PARA.

(i) GAs with Objectives Types A and B work well for approximate monotonic problems (Figs. 6, 8, 9 and 10). GAs with Objective Type B are superior in finding smaller subsets with acceptable discriminability.
(ii) For large nonmonotonic problems, GA with Objective Type C works well in finding better subsets in the criterion, provided GA has with an appropriate set of parameters (Figs. 4 and 5). Some algorithms other

than GA succeeded in finding smaller subsets with comparable criterion values, but this seems because GA was adjusted in order to find the best-discriminability subset. From successes of GAs with Objective Types A and B, we believe that once an optimization function is modified to account both class disciminability and feature dimensionality (for example, as Eq. (1)), GA will be satisfactory.

(iii) GA takes much more time than the others in small-scale and medium-scale problems (Figs. 2, 6, 7 and 10), in spite of its complexity of $\Theta(n)$. This is because GAs needed many iterations (generations) for their convergence and sometimes reached the limit of fifty. The number of generations affected heavily their execution time in small-scale and medium-scale problems. On the contrary, in large-scale problems, this impacts on execution time and makes GA more efficient (Figs. 4(c) and 5(c)).

(iv) PARA worked well for monotonic and approximate monotonic problems (Figs. 2, 6 and 10), but not for nonmonotonic problems. This is because PARA does not have a mechanism to escape from local maxima such as GA has.

In conclusion, GA is suitable for large-scale problems because of its efficiency and effectiveness. In addition, GA is useful to find better answers for Objective Types A and B even in small-scale and medium-scale problems if the computation time is not an important consideration.

4.4.4 SFFS and SBFS vs. GA

Let us compare SFFS and SBFS with GA. There may be a problem in the termination condition of SFFS and SBFS. We terminated them when the number of features reaches a given size m for the first time. We knew after our experiments that $m + \delta m$ is recommended in Pudil *et al.* [4] to improve the performance. SFFS and SBFS try to find solutions along a vertical line (at a fixed m) and GA with Objective Type B tries to find solutions over a horizontal line (over a fixed criterion value). Therefore, in monotonic or approximate monotonic problems, it is still difficult to compare algorithms with different objective types in a completely common way. However, we observed that GA sometimes found better solutions that SFFS and SBFS could not find. This is confirmed by comparing solutions of GA with the criterion curves of SFFS and SBFS (Figs. 4(d), 5(d) and 6(d)). In mammogram(large) data, nine of twelve solutions of GA are better than or equal to those of SFFS and SBFS and the remaining three are worse (Fig. 4). Two of the three worse solutions correspond to the same pair of $(p_c, p_m) = (0.6, 0.4)$ (Fig. 4(d)). In sonar(large) data, five of twelve solutions of GA are better or equal to those of SFFS and SBFS and the remaining seven are worse (Fig. 5). Six of the seven worse solutions corresponds to the same pair of $(p_c, p_m) = (0.6, 0.4)$ (Fig. 5(d)). These results suggest that several distinct parameter sets of GA should be examined at the

same time. The time complexity of GA is $\Theta(n)$, while that of SFFS and SBFS is at least $\Theta(n^2)$. This difference increases as n increases. Then, even if we take into consideration some iterations of GA with some parameter sets, GA would be more effective than SFFS and SBFS for larger problems. Even if a particular GA takes much more time than SFFS and SBFS, the GA has a high possibility to find better solutions than the two algorithms. This is because SFFS and SBFS do not have a mechanism of jumping from a subset to another very different subset. They trace a sequence of subsets of which adjacent subsets are different by only one feature.

In addition, in medium-scale approximate monotonic problems, GAs with Objective Types A and B found several solutions which could not be found by SFFS and SBFS in mammogram(small) and sonar(small) data (for example, Fig. 6(d)).

In earlier work [2,5], GA was compared with SFFS and SBFS. Ferri *et al.* [2] concluded that GA and SFFS are comparable in performance, but as the dimensionality increases the result of GA becomes worse than that of SFFS. Jain and Zongker [5] compared GA with SFFS on Kittler's data and reported a tendency of premature convergence on the use of GA. In contrast to these reports, our experiments show that in small-scale and medium-scale problems GA is better than SFFS and SBFS if GA is given extended time for training. As the dimensionality increases, GA becomes faster than SFFS and SBFS. Below we suggest several possible explanation of the difference between our observations and those of References (2) and (5).

The first possible explanation is a difference in the way of determining the training time. Ferri *et al.* stopped GA when GA reached the same number of evaluations as SFFS, but we stopped GA after 50 generations. Our training time determination produced a GA faster than SFFS and SBFS, while the subsets selected by GA were comparable in performance with those of SFFS and SBFS.

A second possible explanation is a difference in the values of parameters. There are many possible differences between Ferri *et al.* 's GAs and ours: the population size, the initial population, crossover rate and mutation rate. Ferri *et al.* did not reveal the values of their parameters in their paper [2]. We took the population size as twice the number of features so as to make GA adapt to an increase of the dimensionality. This adaptive population size may have brought better results than their GAs. The crossover and mutation rates also strongly affect the results, as described in the previous section.

A third possible explanation is a difference in the objective of selection. Ferri *et al.* used Objective Type B for their GA and the penalty function of Eq. (1) with a monotonic criterion. We used only Objective Type C for GA on large-

scale problems. For Objective Type C, GA seems to be more effective than SFFS and SBFS by virtue of its ability to examine a wider range of candidate solutions. For Objective Type A, SFFS or SBFS could yield a good feature subset stably. If both SFFS and SBFS are used simultaneously, the result would be better.

Jain and Zongker [5] compared GA and SFFS on Kittler's data. We also compared GA and SFFS on Kittler's data. In our experiments, half of the results were better than or equal to those of SFFS and SBFS. Almost all worse results were obtained when the mutation rate p_m was 0.02, which is the same situation as Jain and Zongker. This suggests the advisability of using a higher mutation rate. Our suggestion is $p_m = 0.1$

In summary, if GA is applied to the same problem two or three times with a few different sets of parameters, GA can be better than SFFS and SBFS in the following two points: 1) GA is controllable in the execution time, indeed we can terminate the generation whenever we want, and 2) the result of GA can be improved by repeating trials and by varying the values of parameters.

With regard to item (1), GA seems preferable for all large-scale problems in which $n \geq 100$. For example, to get the criterion curve of SFS we need 5050 evaluations for $n = 100$ and 45150 evaluations for $n = 300$. If we estimate the complexity of SFFS by $\Theta(n^{2.4})$, we need approximately 63000 evaluations for $n = 100$ and 880000 evaluations for $n = 300$. In our setting of GA, we need only 10000 evaluations for $n = 100$ and 30000 evaluations for $n = 300$.

The second item brings us a high possibility to find better answers at the expense of more time. In fact, in our two large-scale nonmonotonic problems, GA found the best feature subset in the criterion and feature subsets better or equal to those of SFFS and SBFS in more than half of several settings and trials. Thus, in more realistic situations where a time-consuming estimate of the correct recognition rate of a classifier is used and $n > 100$, GA becomes the only practical way to get reasonable feature subsets.

5 Conclusions

Our main conclusions are as follows:

(i) Our methodology permits comparisons of algorithms having diverse objectives. This methodology is based on the use of a criterion curve that provides a guideline of combinations of the number of features and the power of discrimination among the classes. This criterion curve is obtained by applying the simplest sequential search methods over the entire

range of the number of features. (See Section 3).

(ii) Preliminary feature selection using algorithms with a low time complexity is effective to capture a given problem (i.e. to estimate the principal design parameters) and to determine some parameters needed for algorithms in a subsequent refined feature selection. (See Section 3)

(iii) SFFS and SBFS give better solutions than the other sequential search algorithms in a reasonable time for small-scale and medium-scale problems. Both algorithms should be carried out at the same time (See Sections 4.4.2 and 4.4.4).

(iv) GA is suitable for large-scale problems and has a high possibility to find better solutions that cannot be found by the other selection algorithms. A few runs with different sets of parameters are recommended. GA is slower, but more effective than SFFS and SBFS in small-scale and medium-scale problems. (See Sections 4.4.2–4.4.4).

(v) BAB^{+} and BAB^{++} can work even in medium-scale problems. (See Section 4.4.1).

(vi) RBAB and RBABM are effective for Objective Type B in small-scale and medium-scale problems, but it consumes an excessive time in medium-scale and large-scale problems. Termination by a given number is recommended for medium-scale and large-scale problems. (See Sections 4.4.1).

Acknowledgements

The authors thank the UCI repository of machine learning databases that gave us many kinds of databases and to the Turing Institute, Glasgow, Scotland for the vehicle dataset.

References

[1] J. Kittler. Feature set search algorithms. Pattern Recognition and Signal Processing, *C. H. Chen, ed.*, pp. 41–60. Sijthoff and Noordhoff, Alphen aan den Rijn, Netherlands, (1978).

[2] F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler. Comparative study of techniques for large-scale feature selection. Pattern Recognition in Practice IV, *E. S. Gelsema and L. N. Kanal, eds.*, pp. 403–413. Elsevier Science B. V., (1994).

[3] P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, **15**, 1119–1125, (1994).

[4] P. Pudil, F. J. Ferri, J. Novovičová, and J. Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. *12th International Conference on Pattern Recognition*, pp. 279–283, (1994).

[5] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. Pattern Anal. Machine Intell.*, **19**, 153–157, (1997).

[6] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, **26**, 917–922, (1977).

[7] B. Yu and B. Yuan. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition*, **26**(6), 883–889, (1993).

[8] I. Foroutan and J. Sklansky. Feature selection for automatic classification of non-gaussian data. *IEEE Transactions on Systems Man, and Cybernetics*, **17**, 187–198, (1987).

[9] M. R. Vriesenga. *Genetic Selection and Neureal Modeling for Designing Pattern Classifier*. Doctor thesis, University of California, Irvine, (1995).

[10] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, **10**, 335–347, (1989).

[11] J. Sklansky and W. Siedlecki. Large-scale feature selection. Handbook of Pattern Recognition and Computer Vision, *L. F. Pau C. H. Chen and P. S. P. Wang, eds.*, chapter 1.3 pp. 61–123. World Scientific, (1993).

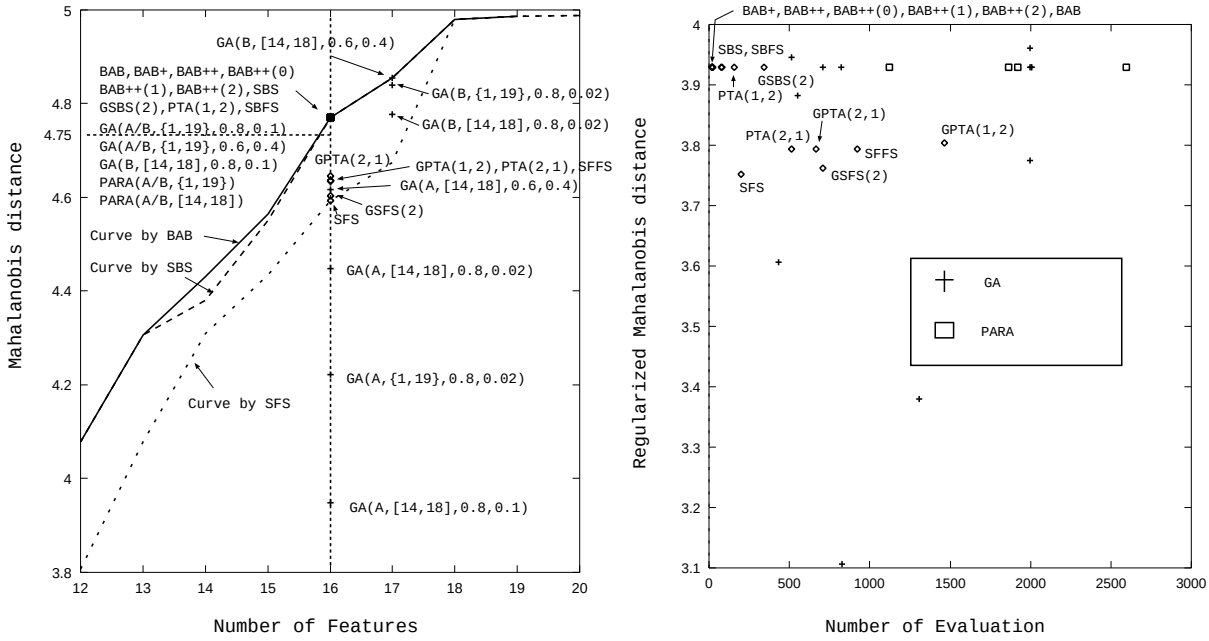
[12] D. Zongker and A. Jain. Algorithms for feature selection: An evaluation. *13th International Conference on Pattern Recognition*, volume 2, pp. 18–22, (1996).

[13] M. Kudo and M. Shimbo. Feature selection based on the structural indices of categories. *Pattern Recognition*, **26**, 891–901, (1993).

[14] H. J. Holz and M. H. Loew. Relative feature importance: A classifier-independent approach to feature selection. Pattern Recognition in Practice IV, *E. S. Gelsema and L. N. Kanal, eds.*, pp. 473–487. Amsterdam: Elsevier, (1994).

[15] M. Kudo and J. Sklansky. Classifier-independent feature selection for two-stage feature selection. Advances in Pattern Recognition, *A. Amin, D. Dori, P. Pudil, and H. Freeman, eds.*, volume 1451 of *Lecture Notes in Computer Science*, pp. 548–554. Springer, (1998).

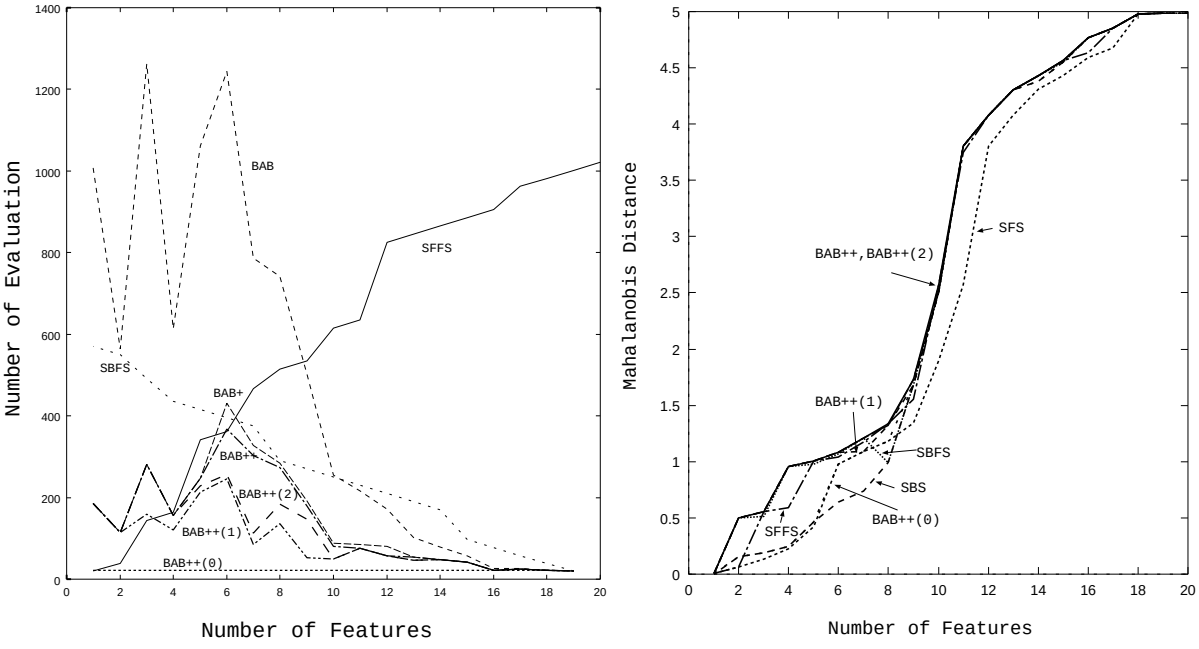
[16] P. M. Murphy and D. W. Aha. *UCI Repository of machine learning databases [Machine-readable data repository]*. University of California, Irivne, Department of Information amd Computation Science, (1996).



(a) $|X|$ vs. $J(X)$

(b) U vs. $J(X)/(5-4) - |X|/(20-1)$

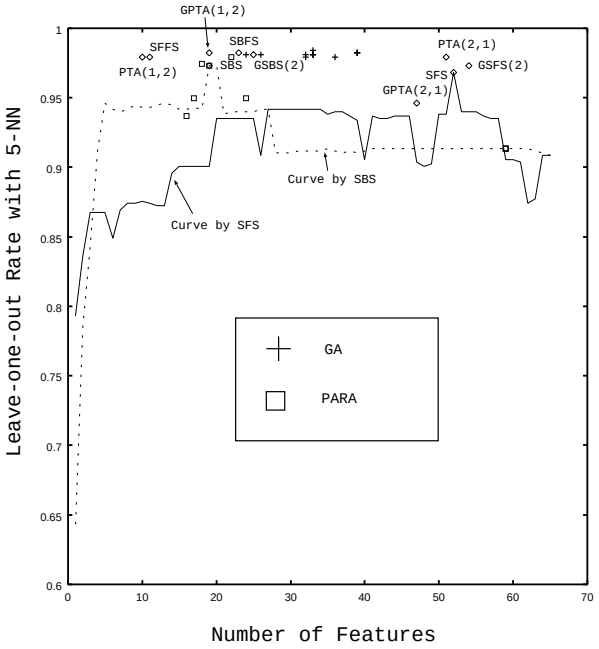
Fig. 2. Result of Kittler's data. $J_\alpha = 4.75$ and $m_\alpha = 16$ for α=5%. Here X is a selected feature subset, $J(X)$ is Mahalanobis distance, $|X|$ is the size of X, and U is the number of evaluation.



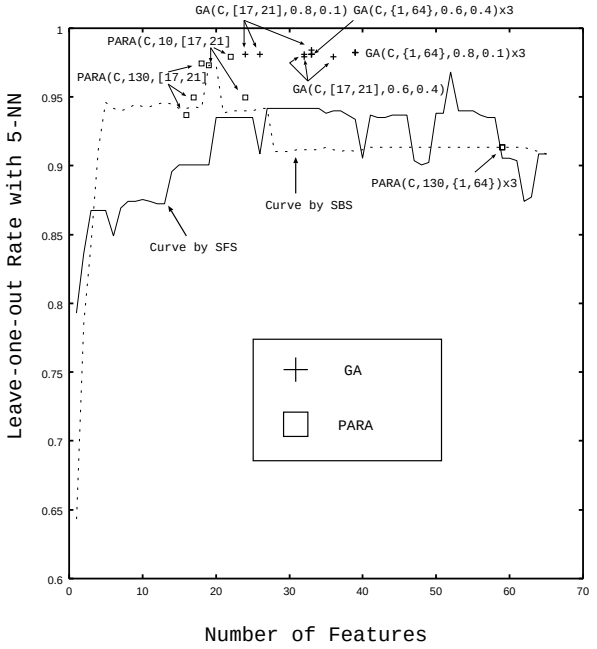
(a) $|X|$ vs. U

(b) $|X|$ vs. $J(X)$

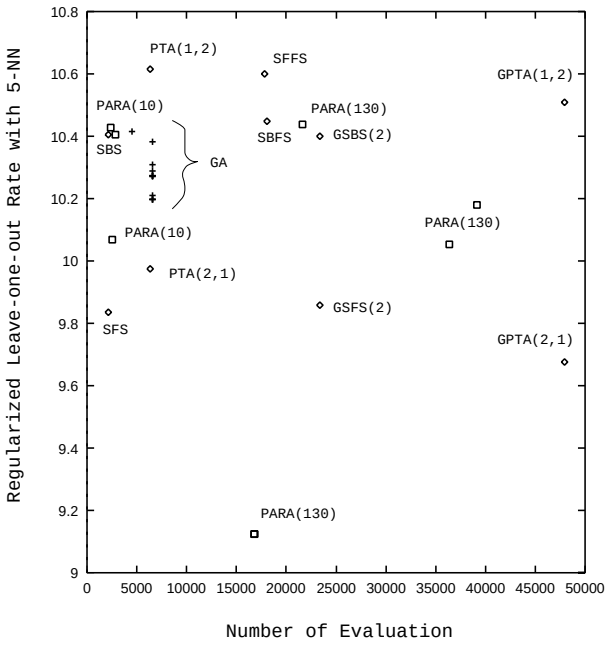
Fig. 3. BAB families Kittler's data.



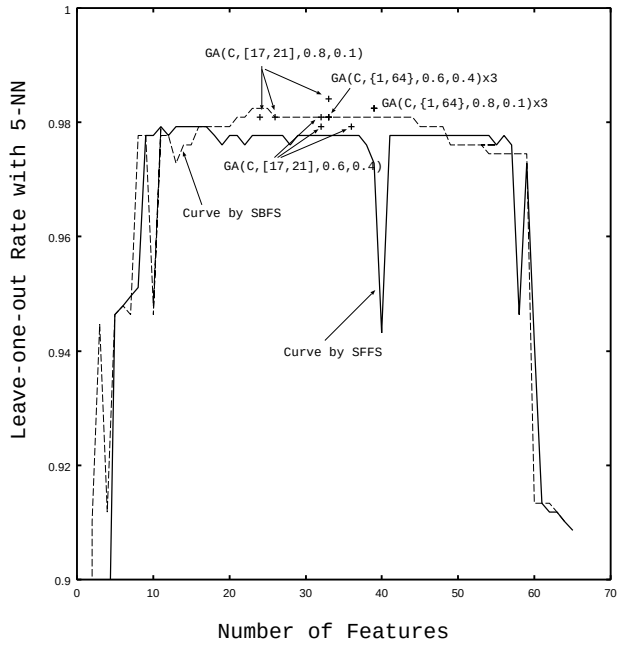
(a) $|X|$ vs. $J(X)$



(b) GA and PARA

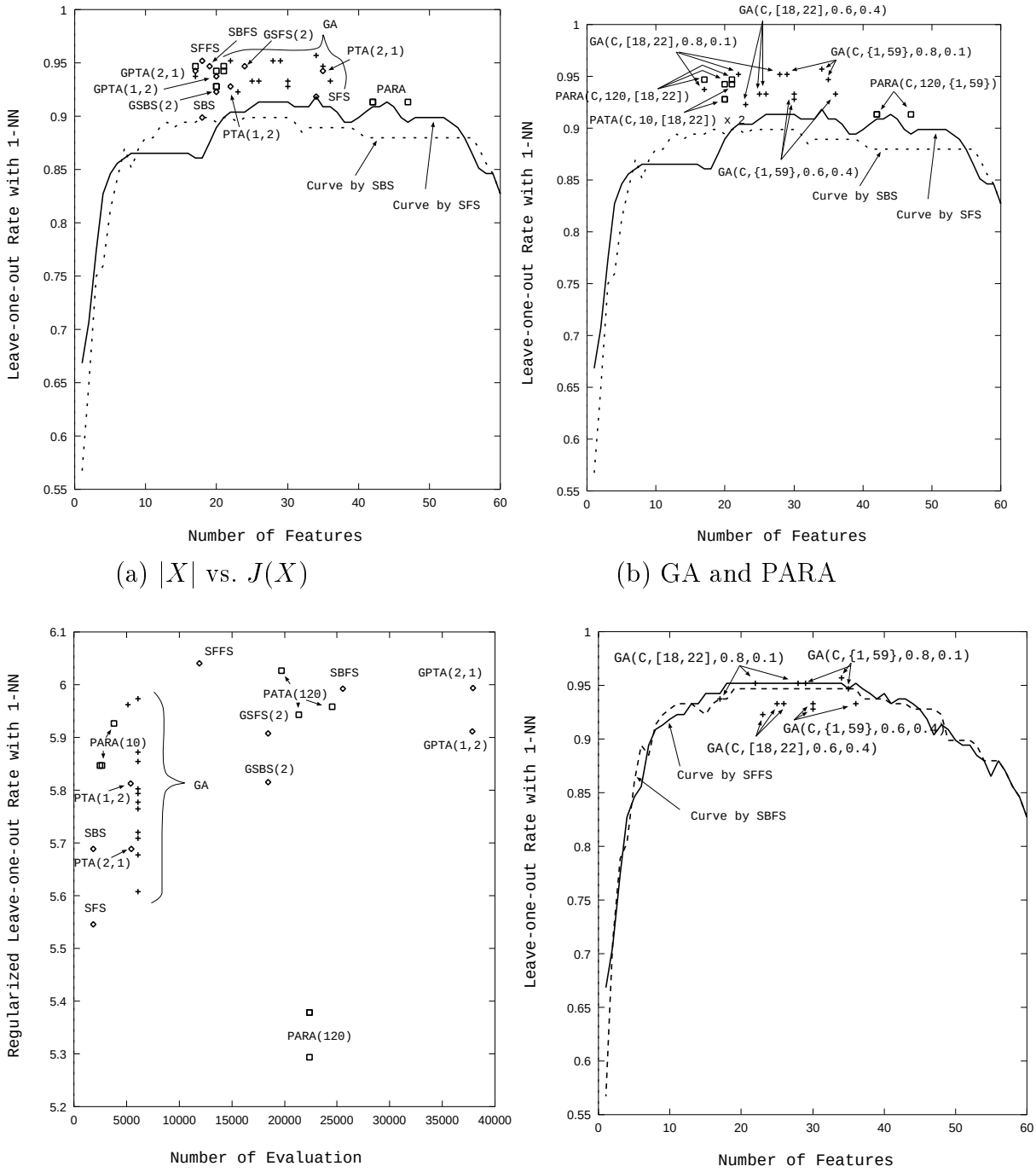


(c) U vs. $J(X)/(1-0.9) - |X|/(65-1)$



(d) GA vs. SFFS and SBFS

Fig. 4. Result graph of Mammogram(large) data. Here X is a selected feature subset, $J(X)$ is the leave-one-out correct recognition rate by a weighted 5-NN method, $|X|$ is the size of X, and U is the number of evaluation.



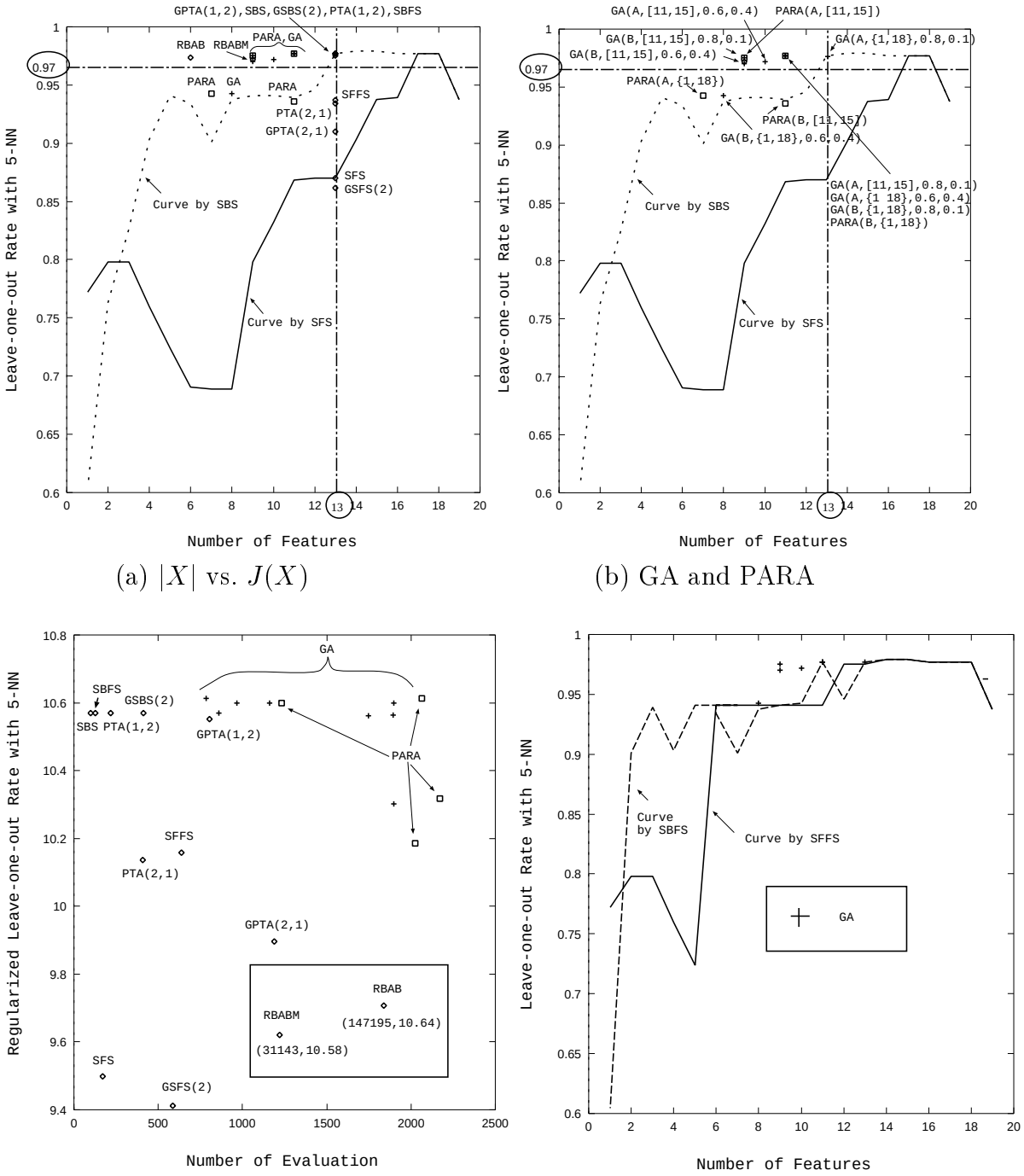
(a) $|X|$ vs. $J(X)$

(b) GA and PARA

(c) U vs. $J(X)/(1-0.85) - |X|/(60-1)$

(d) GA vs. SFFS and SBFS

Fig. 5. Result of Sonar(large) data. Here X is a selected feature subset, $J(X)$ is the leave-one-out correct recognition rate by a weighted 1-NN method, $|X|$ is the size of X, and U is the number of evaluation.



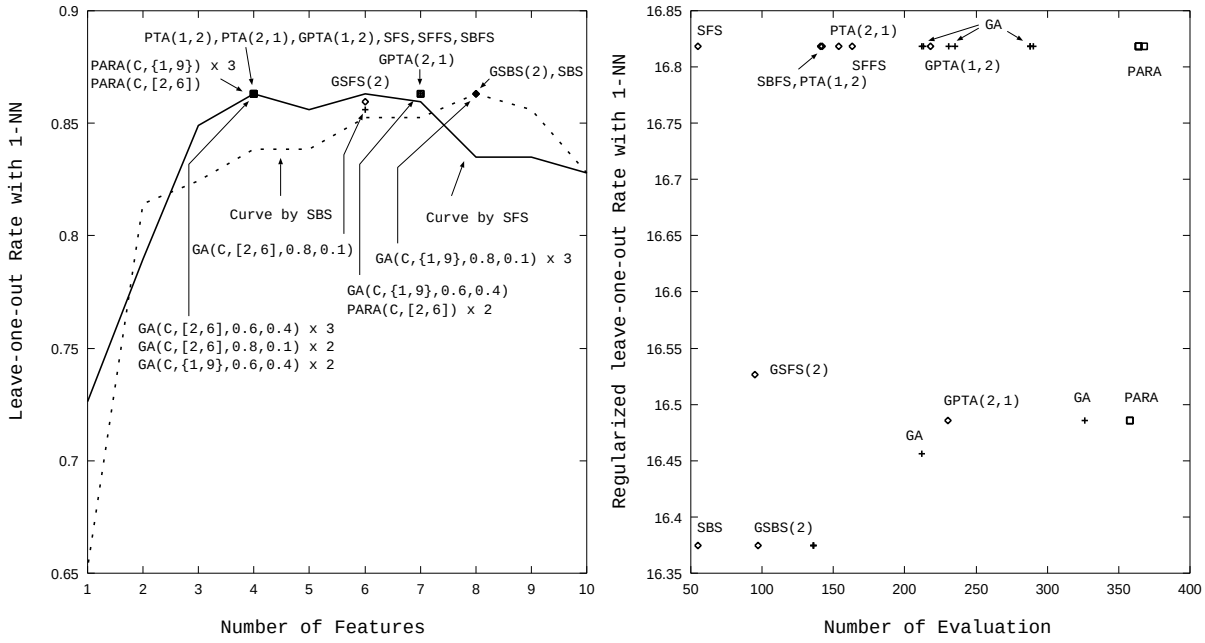
(a) $|X|$ vs. $J(X)$

(b) GA and PARA

(c) U vs. $J(X)/(1-0.9) - |X|/(65-1)$

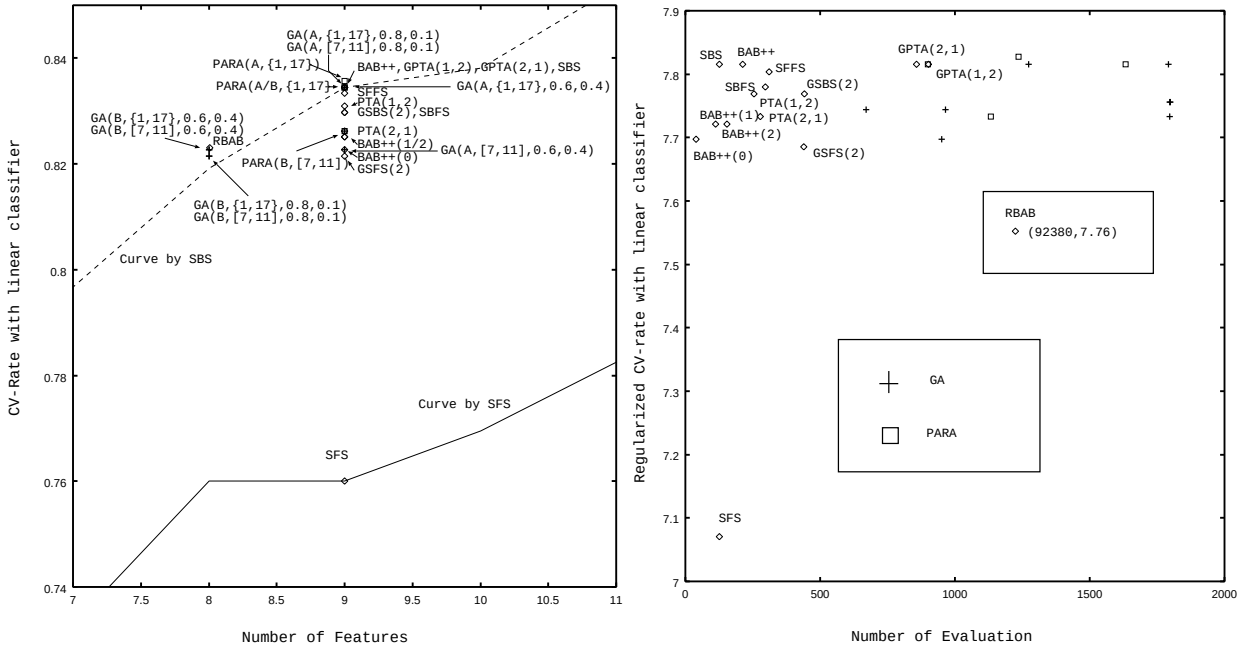
(d) GA vs. SFFS and SBFS

Fig. 6. Result of Mammogram(small) data. $J_\alpha = 0.97$ and $m_\alpha = 13$ for α=1%. Here X is a selected feature subset, $J(X)$ is the leave-one-out correct recognition rate by a weighted 5-NN method, $|X|$ is the size of X, and U is the number of evaluation.



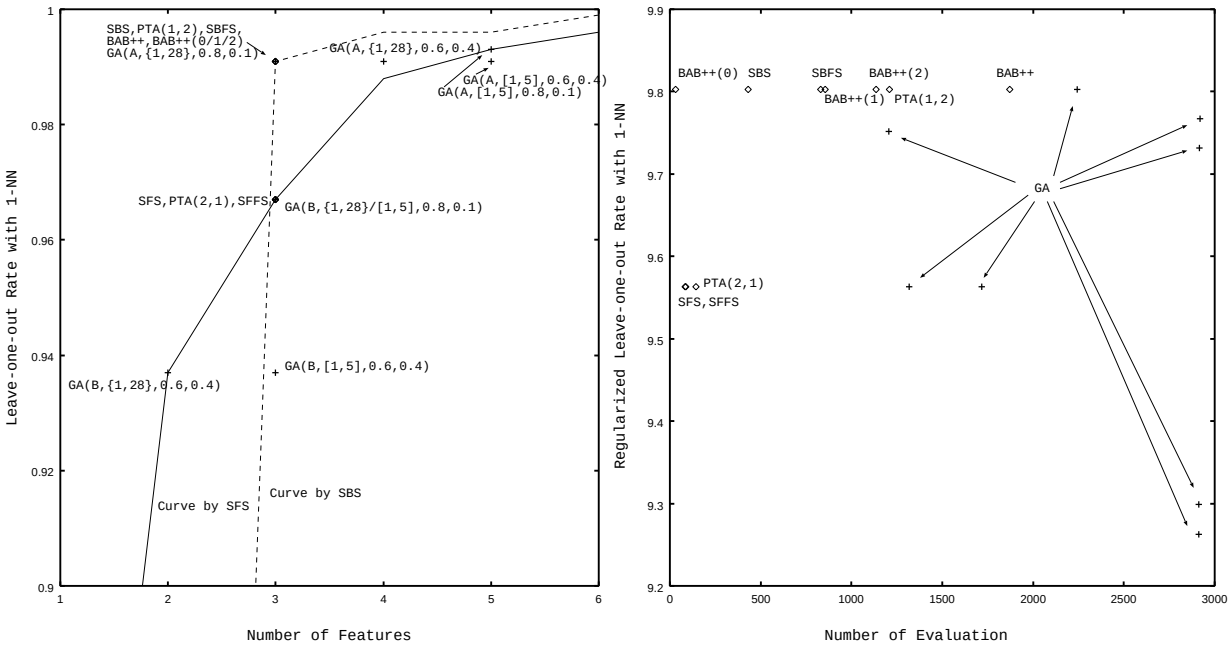
(a) $|X|$ vs. $J(X)$ (b) $J(X)/(0.9-0.85)-|X|/(10-1)$

Fig. 7. Result graphs of SAR data. Here X is a selected feature subset, $J(X)$ is the leave-one-out correct recognition rate by a weighted 5-NN method, $|X|$ is the size of X, and U is the number of evaluation.



(a) $|X|$ vs. $J(X)$ (b) U vs. $J(X)/(0.9-0.8)-|X|/(18-1)$

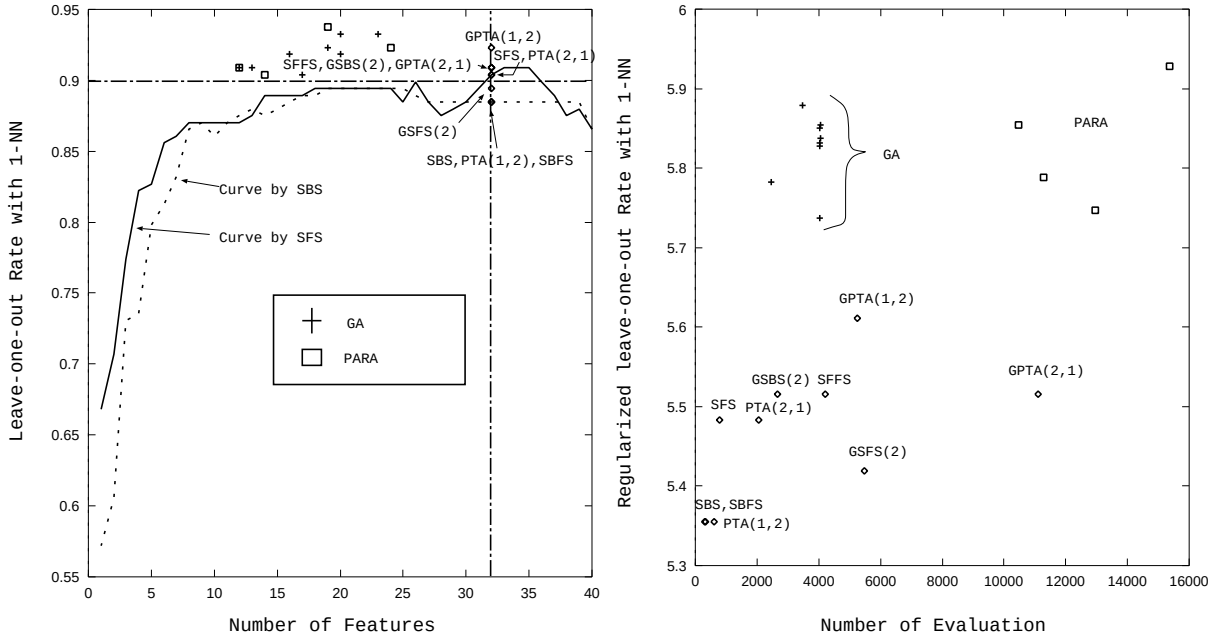
Fig. 8. Result graphs of Vehicle data. $J_\alpha = 0.821$ and $m_\alpha = 9$ for α=5%. Here X is a selected feature subset, $J(X)$ is the 9-fold cross validation correct recognition rate with a linear classifier, $|X|$ is the size of X, and U is the number of evaluation.



(a) $|X|$ vs. $J(X)$

(b) U vs. $J(X)/(1-0.9) - |X|/(29-1)$

Fig. 9. Result graphs of Mushroom(small) data. $J_\alpha = 0.989$ and $m_\alpha = 3$ for α=1%. Here X is a selected feature subset, $J(X)$ is the leave-one-out correct recognition rate with 1-NN, $|X|$ is the size of X, and U is the number of evaluation.



(a) $|X|$ vs. $J(X)$

(b) U vs. $J(X)/(1-0.85) - |X|/(60-1)$

Fig. 10. Result graphs of Sonar(small) data. $J_\alpha = 0.9$ and $m_\alpha = 32$ for $\alpha = 1\%$. Here X is a selected feature subset, $J(X)$ is the leave-one-out correct recognition rate with 1-NN, $|X|$ is the size of X, and U is the number of evaluation.