Feature Selection: Evaluation, Application, and Small Sample Performance

Article *in* IEEE Transactions on Pattern Analysis and Machine Intelligence · March 1997

DOI: 10.1109/34.574797 · Source: IEEE Xplore

2 authors:

Arjun Jain
Max Planck Institute for Informatics
174 PUBLICATIONS 26,777 CITATIONS
SEE PROFILE

Douglas E. Zongker
Google Inc.
17 PUBLICATIONS 2,891 CITATIONS
SEE PROFILE

Feature Selection: Evaluation, Application, and Small Sample Performance

Anil Jain
Department of Computer Science
Michigan State University
East Lansing, Michigan, USA

Douglas Zongker
Department of Computer Science
Michigan State University
East Lansing, Michigan, USA

{jain,zongker}@cps.msu.edu

Abstract

A large number of algorithms have been proposed for feature subset selection. Our experimental results show that the sequential forward floating selection (SFFS) algorithm, proposed by Pudil *et al.*, dominates the other algorithms tested. We study the problem of choosing an optimal feature set for land use classification based on SAR satellite images using four different texture models. Pooling features derived from different texture models, followed by a feature selection results in a substantial improvement in the classification accuracy. We also illustrate the dangers of using feature selection in small sample size situations.

Keywords: feature selection, curse of dimensionality, genetic algorithm, node pruning, texture models, SAR image classification.

1 Introduction

The problem of feature selection is defined as follows: given a set of candidate features, select a subset that performs the best under some classification system. This procedure can reduce not only the cost of recognition by reducing the number of features that need to be collected, but in some cases it can also provide a better classification accuracy due to finite sample size effects [5]. The term *feature selection* is taken to refer to algorithms that output a subset of the input feature set. More general methods that create new features based on transformations or combinations of the original feature set are termed *feature extraction* algorithms. This paper is concerened primarily with the former group.

There has been a resurgence of interest in applying feature selection methods due to the large numbers of features encountered in the following types of problems:

1. Applications where data taken by multiple sensors are fused. Jain and Vailaya [6], for instance, have merged both color and shape features to provide an improved retrieval accuracy for a trademark image database.

2. Integration of multiple models, where the parameters from different mathematical models are pooled for the purpose of classification, such as Solberg and Jain [16].

3. Data mining applications, where the goal is to recover the hidden relationships among a large number of features, as in Punch et al. [11].

The goal of this paper is to illustrate the value of feature selection in combining features from different data models, and to demonstrate the potential difficulties of performing feature selection in small sample size situations, due to the curse of dimensionality. We present a taxonomy of feature selection algorithms. Several well-known and some recently proposed feature selection algorithms have been implemented and tested. Based on these results, the sequential forward floating selection (SFFS) method introduced in [10] has been found to be extremely powerful. We have applied this method to a large data set, created by pooling features from four different texture models, in order to classify SAR satellite images. Feature selection results on this dataset demonstrate (i) the existence of the curse of dimensionality, and (ii) that combining features from different texture models leads to a better classification accuracy than the performance of individual models. The curse of dimensionality phenomenon is further investigated by performing feature selection

on synthetic data sets of various sizes drawn from two Gaussian distributions [17], and evaluating the quality of the selected subset versus the known optimal subset.

2 Feature Selection Algorithms

Let Y be the original set of features, with cardinality n. Let d represent the desired number of features in the selected subset X, $X \subseteq Y$. Let the feature selection criterion function for the set X be represented by $J(X)$. Without any loss of generality, let us consider a higher value of J to indicate a better feature subset. Since we are maximizing $J(\cdot)$, one possible criterion function is $(1 - p_e)$, where p_e denotes the probability of error. The use of probability of error as a criterion function makes feature selection dependent on the specific classifier used and the size of the training and test data sets. Formally, the problem of feature selection is to find a subset $X \subseteq Y$ such that $|X| = d$ and

$$J(X) = \max_{Z \subseteq Y, |Z|=d} J(Z).$$

An exhaustive approach to this problem would require examining all $\binom{n}{d}$ possible d-subsets of the feature set Y. The number of possibilities grows exponentially, making exhaustive search impractical for even moderate values of n. Cover and Van Campenhout [1] showed that no nonexhaustive sequential feature selection procedure can be guaranteed to produce the optimal subset. They further showed that any ordering of the error probabilities of each of the 2^n feature subsets is possible.

A taxonomy of available feature selection algorithms into broad categories is presented in Figure 1. We first divide methods into those based on statistical pattern recognition (SPR) classification techniques, and those using artificial neural networks (ANN). The SPR category is then split into those guaranteed to find the optimal solution and those that may result in a suboptimal feature set. The suboptimal methods are further divided into those that store just one "current" feature subset and make modifications to it, versus those that maintain a population of subsets. Another distinction is made between algorithms that are deterministic, producing the same subset on a given problem every time, and those that have a random element which could produce different subsets on every run. Some representative feature selection algorithms are listed beneath each leaf node in the tree.

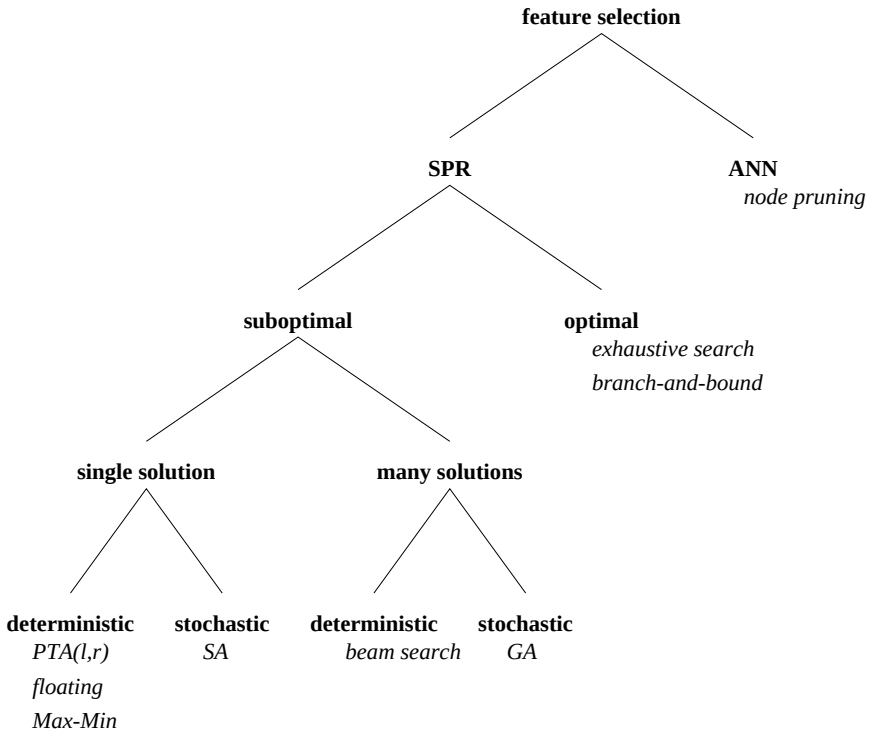

Figure 1: A taxonomy of feature selection algorithms.

2.1 Deterministic, Single-Solution Methods

The first group of methods begin with a single solution (a feature subset) and iteratively add or remove features until some termination criterion is met. These are also referred to as "sequential" methods. These are the most commonly used methods for performing feature selection. They can be divided into two categories, those that start with the empty set and add features (the "bottom-up," or "forward" methods) and those that start with the full set and delete features (the "top-down," or "backward" methods). Note that since they don't examine all possible subsets, these algorithms are not guaranteed to produce the optimal result. Kittler [7] gives a comparative study of these algorithms and the optimal branch-and-bound algorithm using a synthetic two-class Gaussian data set. Pudil *et al.* [10] update this study by introducing the two "floating" selection methods, SFFS and SFBS.

2.2 Deterministic, Multiple-Solution Methods

Siedlecki and Sklansky [14] have discussed performing a best-first search in the space of feature subsets, as well as a restricted version of this, called "beam search." Both these methods maintain a queue of possible solutions.

These are examples of methods that treat the space of subsets as a graph, called a "feature selection lattice," (where each node represents a subset, and an edge represents the containment relationship) and then apply any one of a number of standard graph-searching algorithms. Since this approach does not appear to be widespread in the literature, we have not included these methods in our evaluation.

2.3 Stochastic, Multiple-Solution Methods

Siedlecki and Sklansky [15] introduced the use of genetic algorithms (GA) for feature selection. In a GA approach, a given feature subset is represented as a binary string (a "chromosome") of length n, with a zero or one in position i denoting the absence or presence of feature i in the set. Note that n is the total number of available features. A population of chromosomes is maintained. Each chromosome is evaluated to determine its "fitness," which determines how likely the chromosome is to survive and breed into the next generation. New chromosomes are created from old chromosomes by the processes of (i) crossover, where parts of two different parent chromosomes are mixed to create offsprings,

and (ii) mutation, where the bits of a single parent are randomly perturbed to create a child.

2.4 Optimal Methods

The branch-and-bound (BB) feature selection algorithm, proposed by Narendra and Fukunaga [9], can be used to find the optimal subset of features much more quickly than exhaustive search. One drawback is that the branch-and-bound procedure requires the feature selection criterion function to be monotone, i.e.:

$$J(A \cup B) \geq J(A) \qquad \forall A,B \subseteq Y. \tag{1}$$

This means that the addition of new features to a feature subset can never decrease the value of the criterion function. We know from the curse of dimensionality phenomenon that in small sample size situations this may not be true. Also, the branch-and-bound method is still impractical for problems with very large feature sets, because the worst case complexity of this algorithm is exponential. For a detailed explanation of the algorithm, the reader is referred to Narendra and Fukunaga [9].

Hamamoto *et al.* [4] give results showing that the branch and bound procedure works well even in cases where the feature selection criterion is nonmonotonic. Yu and Yuan [19] present a modification of the Narendra and Fukunaga's branch and bound algorithm, called BAB$^+$, and show, both analytically and experimentally, that it outperforms the original algorithm. Their modification essentially recognizes all "string-structure subtrees" (those subtrees that consist of a single path from the root to a terminal node) and immediately skips the search forward to the appropriate terminal node, thus saving intermediate evaluations.

2.5 Node Pruning

Mao *et al.* [8] use a multilayer feedforward network with a backpropagation learning algorithm for pattern classification [13]. They define a "node saliency" measure and present an algorithm for pruning the least salient nodes to reduce the complexity of the network after it has been trained. The pruning of input nodes is equivalent to removing the corresponding features from the feature set. The node-pruning (NP) method simultaneously

develops both the optimal feature set and the optimum classifier.

The squared-error cost function is used in training the network. The saliency of a node is defined as the sum of the increase in error, over all the training patterns, as a result of removing that node. Mao *et al.* [8] approximate the node saliency with a second-order expansion and then compute that value by finding the appropriate derivatives in a backpropagation fashion. While computing the saliency directly from the definition (i.e., by removing each node from the network in turn and evaluating it over all the test data) is impractical for a large network, this backpropagation method makes computing saliency values practical, as it requires only one pass through the training data (versus one pass per node).

The node pruning-based feature selection methodology first trains a network, and then removes the least salient node (input or hidden). The reduced network is trained again, followed by removal of the least salient node. This procedure is repeated until the desired tradeoff between classification error and size of the network is achieved.

3 Experimental Results

We have reproduced the results of Kittler [7] and Pudil *et al.* [10], comparing the feature selection algorithms in terms of classification error and run time on a 20-dimensional, 2-class data set. The two class-conditional densities were Gaussian, with mean vectors μ_1 and μ_2 and a common covariance matrix Σ used in [7, 10]. The criterion function for assessing the "goodness" of a feature subset was the Mahalanobis distance $((\mu_1 - \mu_2)^t \Sigma^{-1} (\mu_1 - \mu_2))$ between the two class means. Under Gaussian class-conditional densities, the probability of error is inversely proportional to the Mahalanobis distance [2]. Maximum likelihood estimates of the covariance matrix and mean vectors were computed from the data. A total of fifteen feature selection algorithms, listed in Table 1, were evaluated and compared.

Execution times reported are processor ticks (0.01 second) spent in user space on a SUN SPARCserver 1000. Ten randomly generated data sets, each with 1,000 patterns per class, were tested and the averages of the runs are reported. Figure 2 shows the results for some of the algorithms compared.

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

- The max-min algorithm, while very fast, gives poor results compared to the other

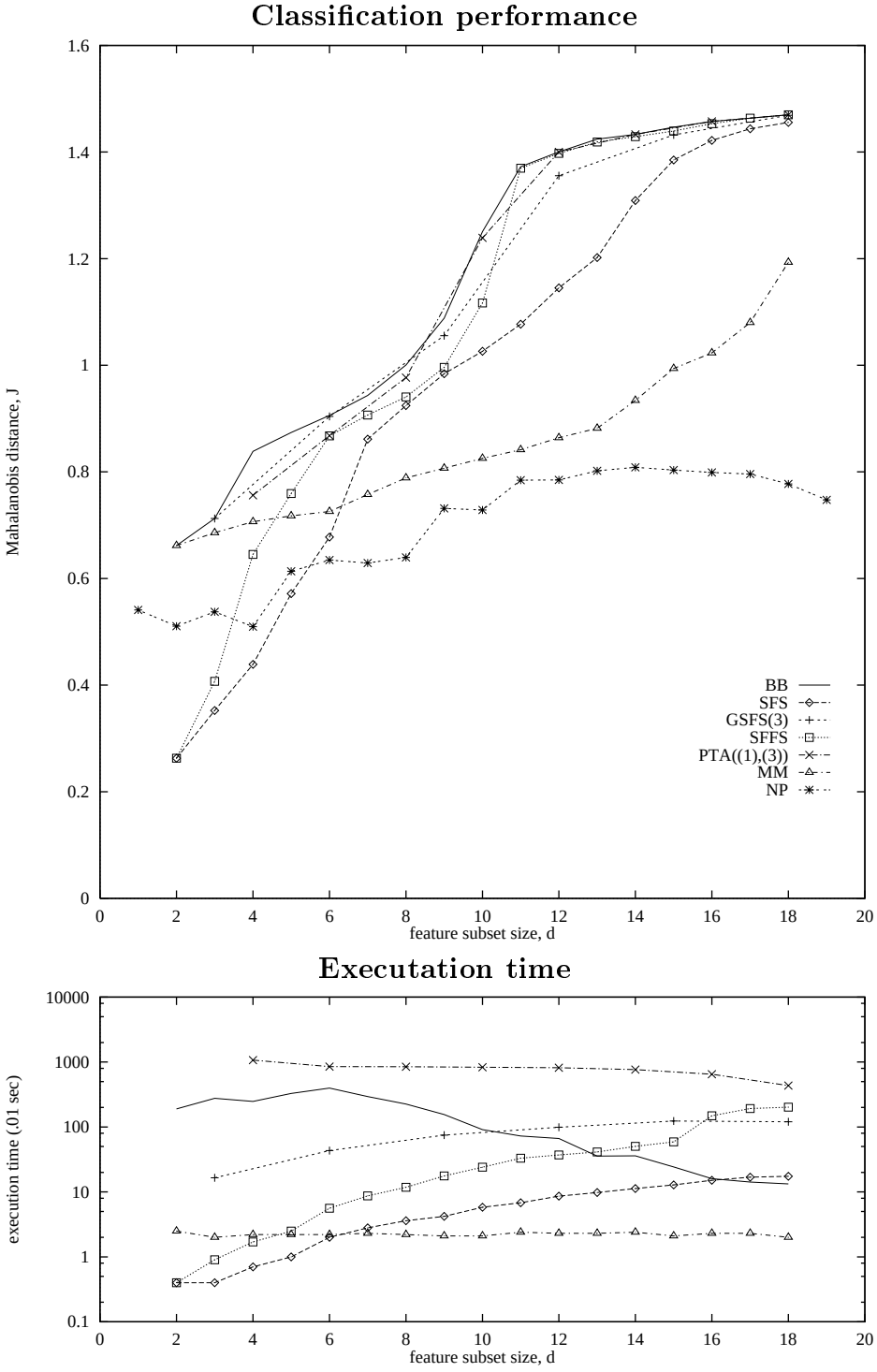

Figure 2: Performance and execution times of selected algorithms on synthethic 2-class Gaussian data set.

SFS	SBS	GSFS(2)	GSBS(2)
GSFS(3)	GSBS(3)	SFFS	SFBS
PTA((1), (2))	PTA((1), (3))	PTA((2), (3))	
BB	MM	GA	NP

Table 1: Feature selection algorithms used in experimental evaluation.

algorithms. It gives better subsets than SFS and SBS for small d, presumably because, unlike those algorithms, it is initialized by choosing the best possible pair. This initial advantage is quickly lost as the value of d increases.

- The SFS and SBS algorithms have comparable performance. Both the algorithms suffered performance hits on this data due to the nature of the generated data, which was engineered to show *nesting* problems. (For instance, the optimal 3-subset is not contained in the optimal 4-subset, the optimal 6-subset is not contained in the optimal 7-subset, etc.) The forward method is faster than its backward counterpart. This is to be expected, as the forward method starts with small subsets and enlarges them while the backward method starts with large subsets and shrinks them. It is computationally more expensive to determine the criterion value for large subsets than for small subsets. This is also true of the generalized methods (GSFS and GSBS).

- The floating methods show results comparable to the optimal algorithm (BB) despite being, for the most part, faster than the branch-and-bound algorithm. The SFFS method lags behind for low d, probably because the algorithm satisfied its termination condition before being able to "float" up and down exploring various subsets. The lack of an explicitly specified termination condition may be the cause for this, since the original paper on floating methods [10] shows near-optimal results for all values of d. A change in termination criterion (such as requiring a minimum number of iterations) could produce better results.

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 importance than its ultimate classification performance. While this is generally true for feature sets of moderate size,

some recent applications (e.g., integration of multiple data models and data mining) have focused on performing feature selection on data sets with hundreds of features. In such cases execution time becomes extremely important as it may be impractical to run some algorithms even *once* on such large data sets. For instance, on a 500-feature data set, the GSFS(3) algorithm, which gives near-optimal results in [10], would require over 20 million subset evaluations for the first step.

The genetic algorithm proposed by Siedlecki and Sklansky was also implemented and tested. Unfortunately, this algorithm has several parameters for which no guidance is available on how to specify their values. Using a population size of 100 for fifteen generations, with a probability of mutation $p_m = 0.02$, we tried several different settings of the feasibility threshold (ranging from 0.1 to 0.4) and the tolerance margin (ranging from 0.05 to 1.0). The best result was obtained with threshold $t = 0.1$ and margin $m = 0.05$: an 8-element subset giving a recognition rate of 78.9%. The GA seems to display a tendency towards premature convergence—most runs reached their peak by the seventh or eighth generation and failed to make further improvements after that.

It is difficult to compare the GA method with the sequential methods. Unlike the sequential methods, this method does not attempt to find the best subset of a specified size—its search space encompasses all the subsets. It is hard to get the algorithm to find the overall best subset since the chromosome score is so heavily influenced by the subset size.

Siedlecki and Sklansky compared the GA approach with sequential search (forward and backward), and with a nonoptimal variation of branch and bound (Foroutan-Sklansky BB search) which is able to work with a nonmonotonic criterion. On a synthetic 24-dimensional data set as well as on a real 30-dimensional data set, the GA outperformed these other feature selection methods (in terms of both classification performance and computational effort).

Ferri *et al.* [3] compared SFS, SFFS, and the genetic algorithm methods on data sets with up to 360 dimensions. Their results show that SFFS gives good performance even on very high-dimensional problems. They show that the performance of GA, while comparable to SFFS on medium-sized problems (around 20–30 dimensions), degrades as the dimensionality increases.

4 Selection of Texture Features

We have applied feature selection for the purpose of land use classification using SAR (Synthetic Aperture Radar) images. Some of the SAR images used in our experiments are given in Figure 3. Solberg and Jain [16] have used texture features computed from SAR images to classify each pixel into one of five classes. A total of 18 features per pattern (pixel) were computed from four different texture models: local statistics, gray level co-ocurrence matrices (GLCM), fractal features, and a lognormal random field model (MAR). These 18 features are listed in Table 2. Our goal is to determine whether the classification error can be reduced by applying feature selection to this set of 18 features derived from four different texture models. A similar feature selection study for 2D shape features was reported by You and Jain [18].

#	**feature**	**model**
1	mean	local statistics
2	θ_1	MAR
3	θ_2	MAR
4	θ_3	MAR
5	σ (variance)	MAR
6	mean (logarithmic)	MAR
7	angular second moment	GLCM
8	contrast	GLCM
9	inverse difference moment	GLCM
10	entropy	GLCM
11	inertia	GLCM
12	cluster shade	GLCM
13	power-to-mean ratio	local statistics
14	skewness	local statistics
15	kurtosis	local statistics
16	contrast (from Skriver, 1987)	local statistics
17	lacunarity	fractal
18	dimension	fractal

Table 2: Set of 18 texture features from four different models.

We report results for one SAR image (the October 17 image from [16]), containing approximately 22,000 pixels. This data was split equally to form "independent" training

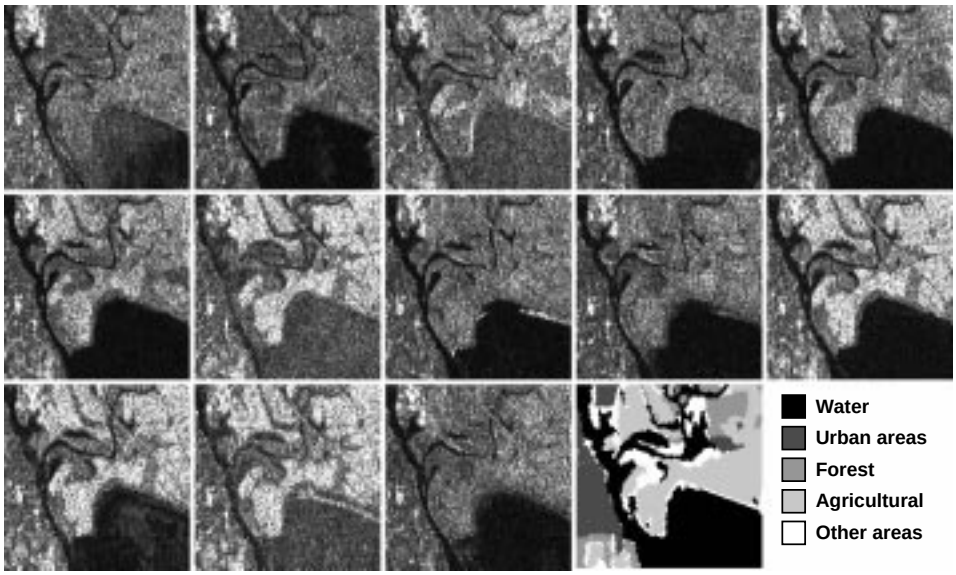

Figure 3: Sample SAR images, with corresponding ground truth image at lower right.

and test sets.

The k-Nearest Neighbor (KNN) recognition rate was used as the feature selection criterion. Both 1NN and 3NN classifiers were used. Based on its consistently high performance for the synthetic data in Section 3, we chose to apply the SFFS method to the texture data set. The results of these runs are shown in Figure 4. Feature selection results based on KNN classifiers have the following behavior: as more features are added, there is a relatively smooth rise in the recognition rate, which then peaks and eventually falls, demonstrating the curse of dimensionality.

Table 3 gives, for each KNN run, the best recognition rate achieved and the optimal number of features. The feature selection process is not just using the features derived from a single texture model but is utilizing features from different models to provide a better performance. For instance, in every case, the five-feature subset selected contains features from at least three different texture models. The best individual texture model for this data set was the MAR model with a classification accuracy of 68.8% [16]. Pooling features from four different texture models and then applying feature selection increased the classification accuracy of a 1NN classifier to 89.3%.

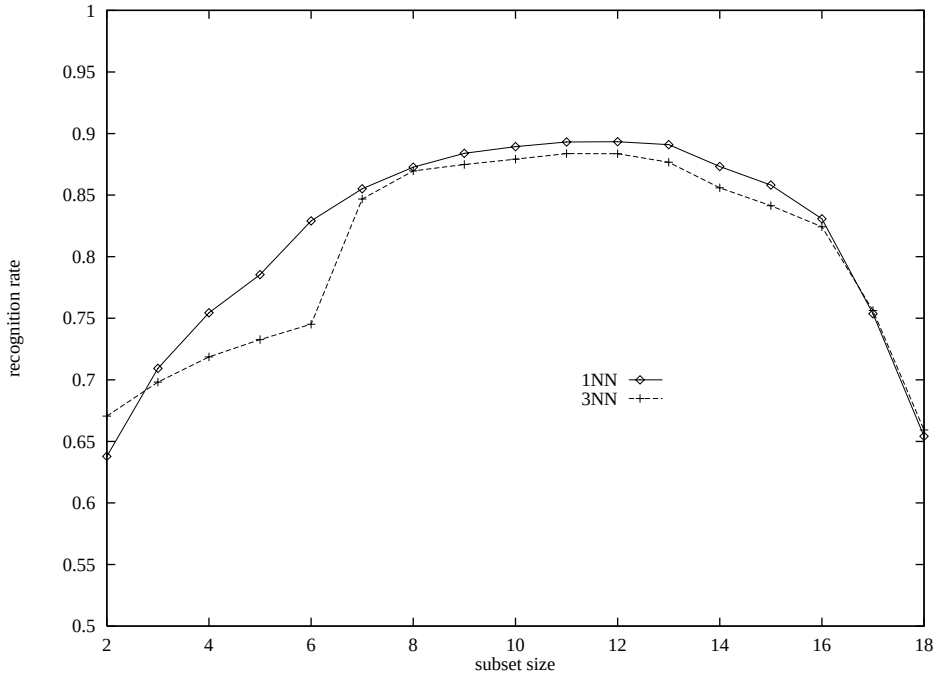

Figure 4: Recognition rates of SFFS method on texture features.

Classifier	**Recognition rate (%)**	**Optimal number of features**
1NN	89.3	12
3NN	88.4	11

Table 3: Best classification accuracy achieved by the SFFS feature selection method.

5 Effect of Training Set Size on Feature Selection

How reliable are the feature selection results in the presence of small amounts of training data? In the case where Mahalanobis distance is used as the criterion, the error arising from estimating the covariance matrix can lead the feature selection process astray, producing inferior results (relative to the true distributions) on independent test data even if the selected subset is optimal for the given training data [12]. We have also seen this effect on the texture data set in the previous section. This phenomenon, which is related to the *curse of dimensionality*, is highlighted by running the feature selection algorithm on varying amounts of training data drawn from known class conditional densities. Trunk [17] used the following two-class example to illustrate the existence of the curse of dimensionality. The two class-conditional densities are given below:

$$p(\mathbf{x}|\omega_1) \sim N(\boldsymbol{\mu}, I) \qquad p(\mathbf{x}|\omega_2) \sim N(-\boldsymbol{\mu}, I) \tag{2}$$

where

$$\boldsymbol{\mu} = \left[\begin{array}{cccc} \frac{1}{\sqrt{1}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \cdots \end{array} \right]^t \tag{3}$$

and I denotes the $n \times n$ identity matrix. Trunk showed the following results:

1. If the mean vector $\boldsymbol{\mu}$ is known, then the probability of error $p_e(n)$ monotonically decreases as the number of features n is increased.

2. If the mean vector is unknown then, for a given training sample size used to estimate $\boldsymbol{\mu}$, the estimated probability of error $\hat{p}_e(n)$ shows a peaking behavior, i.e. $\lim_{n\to\infty} \hat{p}_e(n) = 1/2$.

The class-conditional densities in Eqs. (2) and (3) are useful to investigate the quality of the feature subsets generated by various feature selection methods because for any n and d, the optimal d-feature subset of the given n features is known for the true distribution: it is the first d features.

Data sets of various size, ranging from 10 to 5,000 training patterns per class, were generated from the two 20-dimensional distributions (Eqs. (2) and (3)). For each training set size, five data sets were generated, and the results averaged. The feature selection quality for a training set was calculated by taking the number of commonalities in the resulting feature subset when compared with the optimal subset of the true distribution:

features that were included in both subsets, and features that were excluded from both subsets. This count was divided by the number of dimensions, and that value was averaged over values of d from 1 to 19 inclusive to give a final quality value for the feature set. Note that this value is *not* a measure of the classification error, but a measure of the difference between the subset produced by a feature selection method and the ideal feature subset. The average quality for different training set sizes for the branch and bound and SFS methods is shown in Figure 5.

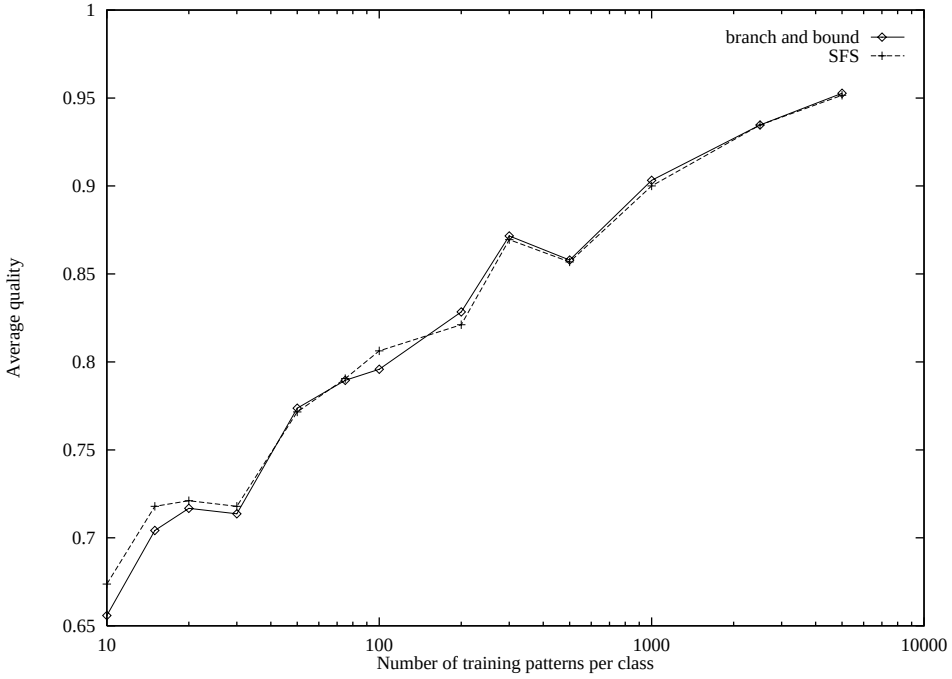

Figure 5: Quality of selected feature subsets as a function of the size of training data.

Since the true class-conditional densities are not at all deceptive with respect to feature selection—the features are all independent with identical variance, only the differences in feature means provide discriminatory information—the selected feature subset should not depend on the specific feature selection algorithm used; any feature selection algorithm should perform well on such a simple problem. Indeed, the performance of the SFS algorithm in Figure 5 closely matches that of the branch-and-bound algorithm. As expected, the quality of the selected feature subset for small training sets is poor, but improves as

the training set size increases. For example, with 20 patterns in the training set, one run of branch-and-bound selecting 10 features chose the subset $\{1, 2, 4, 7, 9, 12, 13, 14, 15, 18\}$ (the optimal 10-subset from the true distribution being $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$). With 2,500 patterns in the training set, one run of the branch-and-bound procedure selected the subset $\{1, 2, 3, 4, 5, 6, 7, 9, 10, 11\}$.

6 Summary

This paper illustrates the merits of various methods of feature selection. In particular, the results of Pudil *et al.* [10] demonstrating the quality of the floating search methods are replicated. The floating search methods show a great promise of being useful in situations where the branch-and-bound method can not be used, due to either the nonmonotonicity of the feature selection criterion or computational reasons.

Results on texture data show that feature selection is useful in utilizing feature derived from different texture models. We also show the pitfalls of using feature selection with limited training data. By using feature selection on a classification problem with known distributions and comparing the selected subsets (under finite sample size) with the true optimal subset, the quality of the selected subset can be quantified. Our experiments show the potential pitfalls of using feature selection on sparse data in a high dimensional space.

References

[1] T. M. Cover and J. M. Van Campenhout. On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7(9):657–661, September 1977.

[2] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[3] F. Ferri, P. Pudil, M. Hatef, and J. Kittler. Comparative study of techniques for large-scale feature selection. In E. Gelsema and L. Kanal, editors, *Pattern Recognition in Practice IV*, pages 403–413. Elsevier Science B.V., 1994.

[4] Y. Hamamoto, S. Uchimura, Y. Matsuura, T. Kanaoka, and S. Tomita. Evaluation of the branch and bound algorithm for feature selection. *Pattern Recognition Letters*, 11:453–456, July 1990.

[5] A. K. Jain and B. Chandrasekaran. Dimensionality and sample size considerations. In P. R. Krishnaiah and L. N. Kanal, editors, *Pattern Recognition Practice*, volume 2, chapter 39, pages 835–855. North-Holland, 1982.

[6] A. K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recogniton*, 29(8):1233–1244, August 1996.

[7] J. Kittler. Feature set search algorithms. In C. H. Chen, editor, *Pattern Recognition and Signal Processing*, pages 41–60. Sijthoff and Noordhoff, Alphen aan den Rijn, Netherlands, 1978.

[8] J. Mao, K. Mohiuddin, and A. K. Jain. Parsimonious network design and feature selection through node pruning. In *Proceedings of 12th ICPR, Jerusalem*, pages 622–624, 1994.

[9] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–922, September 1977.

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

[11] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody. Further research on feature selection and classification using genetic algorithms. In *Proc. 5th International Conference on Genetic Algorithms*, pages 557–564, 1993.

[12] S. J. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, March 1991.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8, pages 318–362. MIT Press, 1986.

[14] W. Siedlecki and J. Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2):197–220, 1988.

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

[16] A. H. S. Solberg and A. K. Jain. A study of the invariance properties of textural features in SAR images. In *Proc. IGARS Conference*, pages 670–672, Florence, Italy, July 1995.

[17] G. V. Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):306–307, July 1979.

[18] Z. You and A. K. Jain. Performance evaluation of shape matching via chord length distribution. *Computer Vision, Graphics and Image Processing*, 28:185–198, 1984.

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