Robust Feature Selection Algorithms

Haleh Vafaie and Kenneth De Jong

Center for Artificial Intelligence
George Mason University
Fairfax, VA 22030

Abstract

Selecting a set of features which is optimal for a given task is a problem which plays an important role in a wide variety of contexts including pattern recognition, adaptive control, and machine learning. Our experience with traditional feature selection algorithms in the domain of machine learning lead to an appreciation for their computational efficiency and a concern for their brittleness. This paper describes an alternate approach to feature selection which uses genetic algorithms as the primary search component. Results are presented which suggest that genetic algorithms can be used to increase the robustness of feature selection algorithms without a significant decrease in computational efficiency.

1. Introduction

Finding an optimal set of features from a large set of candidate features is a problem which occurs in many contexts. In modeling one would like to identify and exploit minimal models of the phenomena under study. In control theory, minimizing the number of control parameters is an important part of the design process. In image understanding and machine learning, finding a minimal set of features necessary for recognition and classification is a key part of designing efficient and implementable systems.

In recent years there has been a significant increase in the size and complexity of the problems undertaken in these and other areas. Along with this has come an increase in the size and complexity of the corresponding feature selection problems, resulting in renewed efforts to automate the process of feature selection, and in some cases, the additional requirement that feature selection occur "on line" because of dynamically changing environments.

The work presented here was motivated by our experiences in using conventional feature selection algorithms for difficult machine learning problems involving texture recognition. In our case there can easily be several hundred candidate features and complex interactions among the features. On such problems conventional features selection algorithms run fast, but produce results which vary dramatically in quality from quite good to poor. As a consequence, we have explored the use of genetic algorithms (GAs) as a means for improving the robustness of such algorithms without sacrificing too much in speed. We present our initial results in the following sections.

2. Feature Selection Techniques

Since each feature used can increase the cost and running time of a system, there is strong motivation to design and implement systems with small feature sets. At the same time there is a potentially opposing need to include a sufficient set of features to achieve acceptably high performance. This has led to the development of a variety of search techniques for finding an "optimal" subset of features from a larger set of possible features. Exhaustively trying all the subsets is computationally prohibitive when there are a large number of features. There are two main approaches to avoiding the combinatorial explosion as the number of candidate features grows. The first involves developing problem specific strategies (heuristics) which use domain knowledge to prune the feature space to a manageable size [5]. The second approach is to use generic heuristics (primarily hill-climbing algorithms) when domain knowledge is costly to exploit or unavailable [8].

In the case of texture recognition, we found ourselves in the second camp: lots of possible features, but little in the way of domain knowledge to assist the search process. We adopted a feature selection algorithm from the literature which involved the basic components illustrated in Figure 1. The search procedure is a classic "greedy" hill-climbing algorithm, sequential backward selection (SBS), which removes one feature at a time until no improvement in the criterion function is obtained. The criterion function is problem specific and can vary from simple performance measures to complex multistage evaluation procedures.

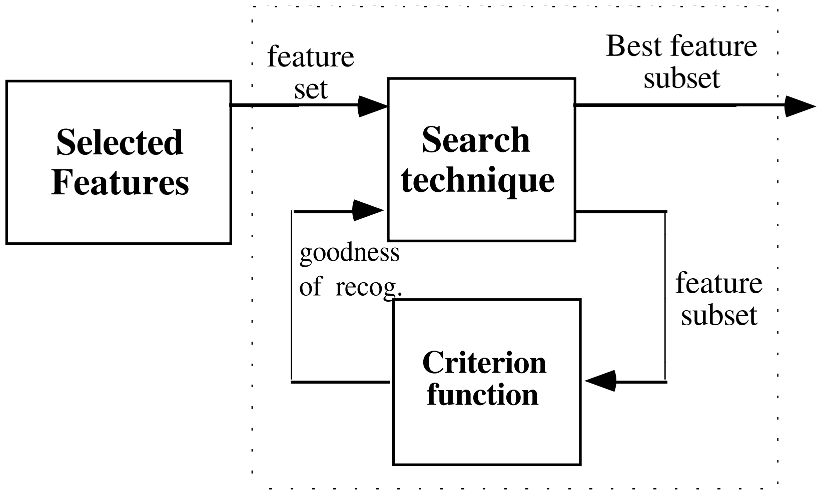


Figure 1: Block diagram of the feature selection process

Our experience with the SBS approach suggests that it is very fast, but also quite brittle in the sense that the quality of the results varies widely across data sets. In the following sections we explore in more detail the cause of the brittleness, and develop a more robust approach by replacing the given search procedure with a genetic algorithm.

3. Sources of Brittleness

The SBS algorithm is a search procedure that starts with the complete set of features, and discards one feature at a time until the desired number of features have been reached [4]. At a particular stage, there are m features remaining. To determine which (if any) feature to remove next, each of the m features are evaluated as a candidate for removal by temporarily removing it and computing the effects via the criterion function. The feature who's removal leads to the largest improvement (lowest decrease in the criterion function) is removed permanently. The process is then repeated for the remaining m-1 features until no improvements are obtained.

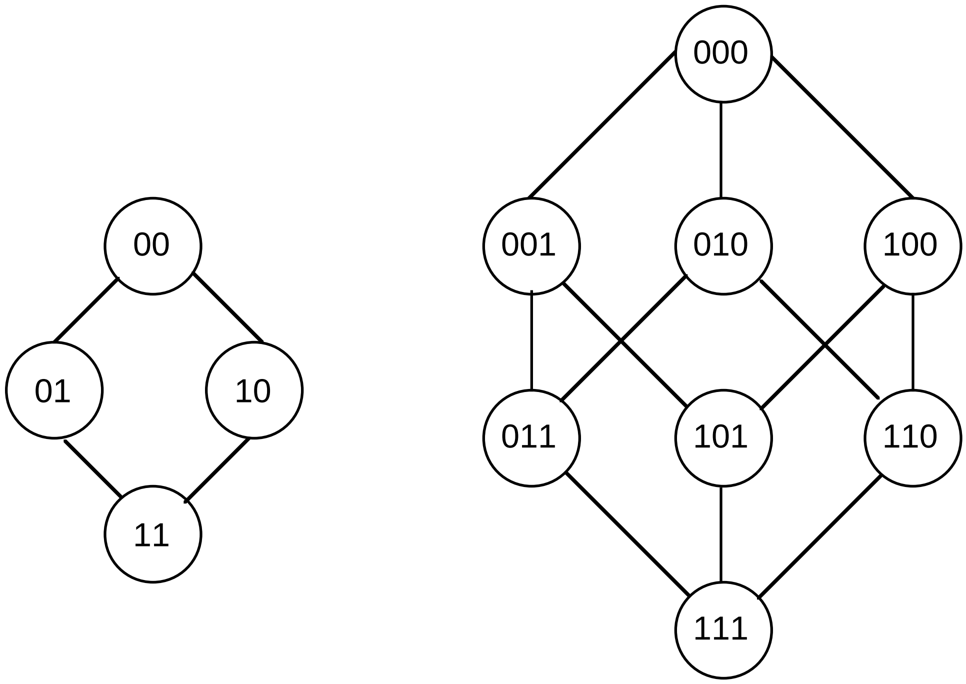
An insightful way to visualize this process is to represent the entire feature set as a binary string of N boolean variables, each representing the presence or absence of the ith feature. As illustrated in Figure 2, the SBS algorithm starts with the string of all ones, and moves up the lattice one level at a time by selecting the node at the next level which results in maximal improvement.

Notice that for problems with N > 2, committing to a particular node generally makes other higher level nodes in the lattice inaccessible. For example, committing to 011 makes 100 inaccessible at the next level. This is both the source of the power and the brittleness of the algorithm. Its running time is clearly no worse than N^2 in terms of the number of different feature sets evaluated before termination. It achieves this, however, by ignoring any possible interactions among the features. Hence, it is possible for no single feature to yield an improvement (thus terminating the search) while the simultaneous removal of 2 or 3 features might result in significant improvements. Similarly, a member of the optimal feature set can be removed early in the search process to obtain some initial improvement, but then be unavailable for later more significant improvements. In addition, if the criterion function contains any imprecision or noise, such greedy algorithms become even less reliable.

Clearly, any attempt at improving the situation by looking at all second order and possibly third order feature interactions quickly becomes computationally prohibitive. An alternative which seemed quite hopeful was the use of genetic algorithms which are best known for their ability to efficiently search large spaces about which little is known, and which are relatively insensitive to noise. In the next section this approach is described in more detail.

4. Feature Selection Using Genetic Algorithms

Genetic algorithms (GAs), a form of inductive learning strategy, are adaptive search techniques initially introduced by Holland [7]. Genetic algorithms derive their name from the fact that their operations are similar to the mechanics of genetic models of natural systems.



a) Two feature lattice b) Three feature lattice

Figure 2: Feature Set Lattices

Genetic algorithms typically maintain a constant-sized population of individuals which represent samples of the space to be searched. Each individual is evaluated on the basis of its overall fitness with respect to the given application domain. New individuals (samples of the search space) are produced by selecting high performing individuals to produce "offspring" which retain many of the features of their "parents". This eventually leads to a population that has improved fitness with respect to the given goal.

New individuals (offspring) for the next generation are formed by using two main genetic operators, crossover and mutation. Crossover operates by randomly selecting a point in the two selected parents gene structures and exchanging the remaining segments of the parents to create new offspring. Therefore, crossover combines the features of two individuals to create two similar offspring. Mutation operates by randomly changing one or more components of a selected individual. It acts as a population perturbation operator and is a means for inserting new information into the population. This operator prevents any stagnation that might occur during the search process.

Genetic algorithms have demonstrated substantial improvement over a variety of random and local search methods [2]. This is accomplished by their ability to exploit accumulating information about an initially unknown search space in order to bias subsequent search into promising subspaces. Since GAs are basically a domain independent search technique, they are ideal for applications where domain knowledge and theory is difficult or impossible to provide [3].

The main issues in applying GAs to any problem are selecting an appropriate representation and an adequate evaluation function. Since GAs require the same kind of evaluation function as the SBS algorithm, it can be used without modification. The natural representation for the feature selection problem is precisely the one described earlier, name a binary string of length N representing the presence or absence of each of the N possible features. The advantage of this representation is that the classical GA's operators as described before (binary mutation and crossover) can easily be applied to this representation without any modification. This eliminates the need for designing new genetic operators, or making any other changes to the standard form of genetic algorithms.

5. Experimental Results

In performing the experiments reported here, the SBS algorithm was used as described above. For the GA approach, GENESIS [6], a general purpose genetic algorithm program, was used with the standard parameter settings recommended in [2]: a population size=50, a mutation rate= 0.001, and a crossover rate=0.6. The results presented here show the average performance of ten runs.

5.1 Artificial Problem Sets

The first set of experiments involved constructing a family of artificial problems to test our hypotheses about the sources of brittleness of the greedy feature selection algorithm and to compare its performance with the GA approach. The basic idea involved constructing families of functions in which the degree of interaction among features could be controlled so as to make the problem either easier or harder for the greedy algorithm.

As indicated earlier in Figure 2, three features (bits) are the minimal size problem in which non-linear interactions can cause problems. We can construct a function with the desired properties by thinking of it as a function of 3 boolean variables (one for each feature) and writing out all the first, second and third order terms:

$$\begin{aligned} f(x_0,x_1,x_2) = {} & a*x_0 + b*x_1 + c*x_2 \\ & + d*x_0*x_1 + e*x_0*x_2 + f*x_1*x_2 \\ & + g*x_0*x_1*x_2 \end{aligned} \quad (1)$$

In general, not all the terms are needed to produce sufficient interaction to cause problems. In this example, we seek a function which (referring back to Figure 2) assigns the optimal value to 100, but causes the SBS algorithm to select 011 on the first round. This can be easily achieved in a variety of ways. For example, we can let b=c and d=e=0. Then (1) simplifies to:

$$\begin{aligned} f(x_0,x_1,x_2) = {} & a*x_0 + b*x_1 + b*x_2 \\ & + f*x_1*x_2 + g*x_0*x_1*x_2 \end{aligned} \quad (2)$$

This results in the following assignments of "value" to all possible subsets of the 3 features:

x_0,x_1,x_2	$f(x_0,x_1,x_2)$
000	0
100	a (global optimum)
001	b
010	b
011	2b+f (local optimum)
101	a+b
110	a+b
111	a+2b+f+g

This can be further simplified by letting m=f+2b. Our goal can then be achieved by requiring:

$a > 0$, $a > b$, $a > m$ (global optimum at 100)

and

$m > a+b$, $m > a+m+g$ (local optimum at 011)

The constraints $a > m$ and $m > a+b$ imply that $b < 0$. If we arbitrarily let b=-4, then $m = 2b+f = f-8$. For simplicity we can set f=9 resulting in m=1, and assign g=-5 yielding:

$$\begin{aligned} f(x_0,x_1,x_2) = {} & \boldsymbol{a}*x_0 - 4*x_1 - 4*x_2 \\ & + 9*x_1*x_2 - 5*x_0*x_1*x_2 \end{aligned} \quad (3)$$

and

x_0,x_1,x_2	$f(x_0,x_1,x_2)$
000	0
100	***a*** (global optimum)
001	-4
010	-4
011	1 (local optimum)
101	***a***-4
110	***a***-4
111	***a***-4

By varying the value of ***a*** > 0, we can test our hypothesis about the source of brittleness of the SBS algorithm. While 0 < ***a*** < 1, the feature set 011 is the global optimum and should be easily found. When 1 < ***a*** < 5, the feature set 011 is the local optimum and the global optimum 100 should never be found. When ***a*** >= 5, the feature set 111 is a local optimum, and the global optimum should never be found.

In the first experiment, we added to this minimal 3-feature dependency problem an additional 27 features which had no effect on the criterion function, but served to increase the size of the search space significantly (2^{30}). Figure 3 presents the results for varying ***a*** between 0 and 10. The behavior of the SBS algorithm was as expected, finding the global optimum only when a<1. For this simple case, the GA approach consistently found the global optimum.

To make things more difficult, we changed the problem slightly by replicating the 3-feature dependency 3 times and then included 21 additional bits as before to create a total search space of 2^{30}. Figure 4 shows the results of applying both algorithms to this problem with ***a*** again ranging from 0 to 10. In this case the GA did not always find the global optimum (it's a heuristic as well), but again significantly outperformed SBS when ***a*** > 1.

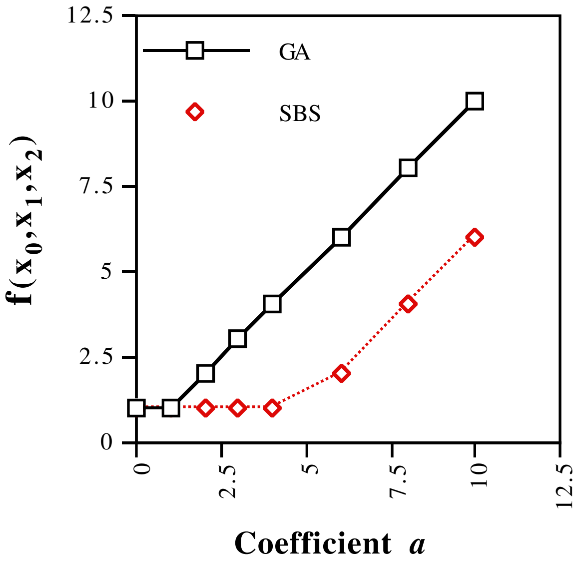


Figure 3: The comparison results of varying *a*

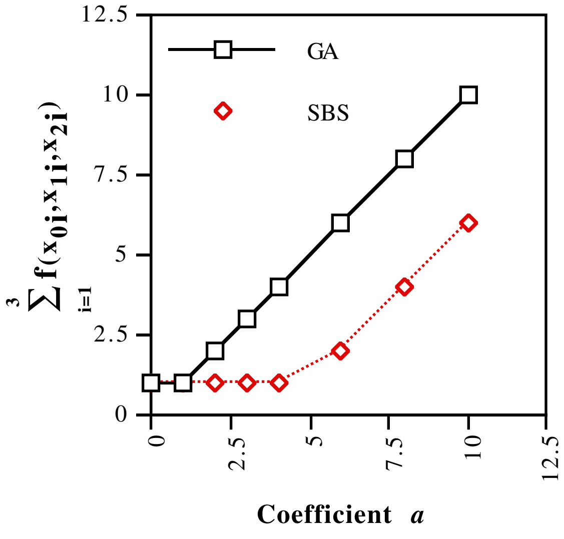


Figure 4: The comparison results of varying *a*

To see more clearly the effects of increasing the number of 3-feature dependencies would have on the two algorithms, we fixed $a = 4$ and varied the number of 3-feature dependencies from 1 to 10, in each case filling in as before with additional extraneous bits to keep the search space at 2^{30}. The results are shown in Figure 5 and indicate quite clearly that the advantage of a GA approach increases with the number of such dependencies.

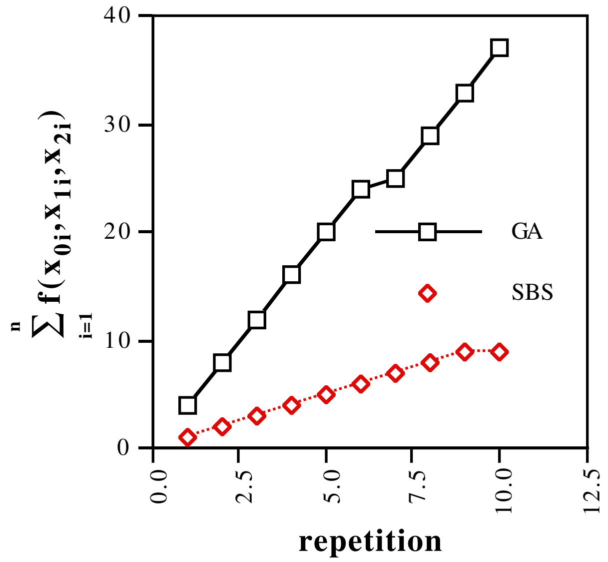


Figure 5: Varying the number of repetitionsof 3-feature dependencies

As a final experiment in this artificial domain, we constructed **m**-feature dependencies by overlapping 3-feature problems (i.e., shared features). Figure 6 presents the results of this experiment. Again, as expected, the GA approach performs significantly better as the number of dependencies grows.

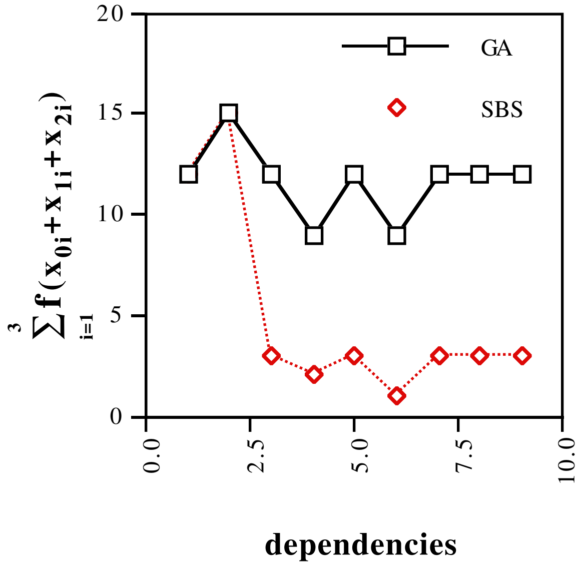


Figure 6: Varying the number of interdependencies

These initial results served to clarify our understanding of the two algorithms and provided insight into the kinds of behavior seen in earlier experiments using real data, namely, that the high variance in the performance of the greedy algorithm was due to feature interdependencies, and that a GA approach would be more robust in general, but less efficient when there were few or no such interactions. In the following section we describe two classes of experiments that support this view.

5.2 Realistic Problem Sets

The first example is based on texture images that were randomly selected from Brodatz album of textures [1]. These images are depicted in Figure 7. One hundred feature vectors, each containing 18 features were then randomly extracted from an arbitrary selected area of 30 by 30 pixels from each of the chosen textures.The goal of this experiment was to find an optimal set of features to be used by AQ15 in order to induce texture classification rules (for a detailed description, see [9]). In order to obtain precise measurements of the classification accuracy associated with a particular feature set, AQ15 had to be run to produce the rules, and then the rules had to be tested for accuracy. Since this can be a computationally expensive process for large data sets, a heuristic evaluation function taken from the feature selection literature was used instead. The heuristic involves selecting feature subsets which maximally separate classes using a Euclidean distance measure [4]. More specifically, the function to be maximized is:

$$J(e) = 1/2 \sum_{i=1}^{c} P_i \sum_{j=1}^{c} P_j \, 1/n_i n_j \sum_{k=1}^{n_i} \sum_{l=1}^{n_j} \partial(e_{ik}, e_{jl})$$

where:

- c denotes the number of classes
- e_{ik}, e_{jl} are testing examples from class i and j respectively
- $\partial(e_{ik}, e_{jl})$ represents the Euclidean distance between two elements
- n_i, n_j denote the number of training examples for the class i and j respectively
- P_i, P_j are the probabilities for the class i and j respectively

Figure 8 shows the results of a typical experiment on this data. J(e) induced relatively little interaction among the features, so both algorithms find the optimal subset, but SBS requires fewer trials (executions of the given criterion function).

Unfortunately, the heuristic evaluation function is not all that good, and so the improvement of the actual recognition rate of the AQ15-produced rules was not strongly correlated with improvements in J(e).

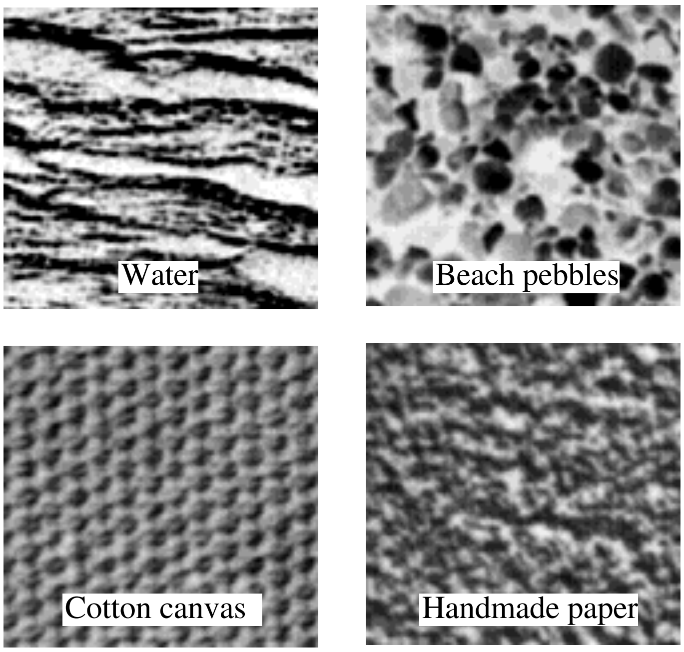


Figure 7: The texture images used in this set of experiments.

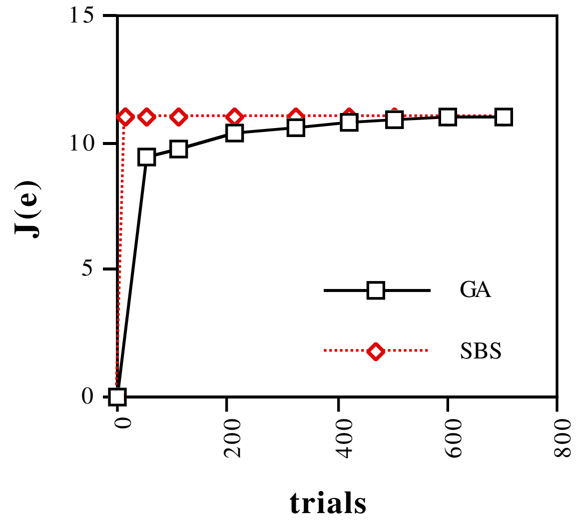


Figure 8: The comparison results of performance over time

In our second example we used a smaller data set (breast cancer data) and used the more computationally expensive way to evaluate feature subsets: by running AQ15 for each feature set to be evaluated and measuring the classification accuracy of the rules produced on the test data. As one might expect, the combination of the rule induction process and the classification evaluation function produced rather strong interactions among the features. Figure 9 show the results of a typical experiment in this context. The robustness of the GA approach is quite evident here.

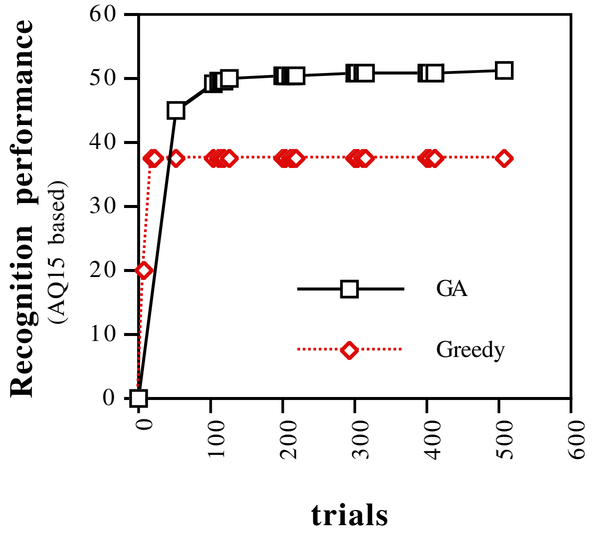


Figure 9: The comparison results of performance over time

6. Summary and Conclusions

The goal of the research reported here was to understand better the observed brittleness of traditional feature selection algorithms and to use that knowledge to develop more robust approaches. The results suggest that the source of the brittleness is a tendency to get trapped on local peaks caused by interdependencies among features. Extending these algorithms directly to avoid such local minima was viewed as computationally prohibitive. Rather, a fairly straightforward implementation of genetic algorithms proved quite effective in improving the robustness of feature selection over a range of problems without significant increases in computational complexity.

At the same time, it should be noted that the traditional approach is more efficient when the number of interacting features is small. An interesting open question is whether a multistrategy approach could be developed which could combine the two approaches since, in general, information about the degree of interactions is generally not available a priori.

Acknowledgments

This research was conducted in the Center for Artificial Intelligence at George Mason University. The Center's research are supported in part by Advanced Research Projects Agency under grant No. N00014-91-J-1854, administrated by the office of Naval Research, and under the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under grant No. N00014-91-J-1351, and in part by the National Science Foundation under grant No. IRI-9020266.

References

[1] Brodatz, P. "*A Photographic Album for Arts and Design*," Dover Publishing Co., Toronto, Canada, 1966.

[2] De Jong, K. "*Analysis of the behavior of a class of genetic adaptive systems*," Ph.D. Thesis, Department of Computer and Communications Sciences, University of Michigan, Ann Arbor, MI., 1975.

[3] De Jong, K. "Learning with Genetic Algorithms : An overview," *Machine Learning* Vol. 3, Kluwer Academic publishers, 1988.

[4] Devijver, P., and Kittler, J. "*PATTERN RECOGNITION: A STATISTICAL APPROACH,*" Prentice Hall, 1982.

[5] Dom, B., Niblack, W., and Sheinvald, J. "Feature selection with stochastic complexity," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Rosemont, IL., 1989.

[6] Grefenstette, John J. Technical Report CS-83-11, Computer Science Dept., Vanderbilt Univ., 1984.

[7] Holland, J. H.. "*Adaptation in Natural and Artificial Systems*," University of Michigan Press, Ann Arbor, MI., 1975.

[8] Kittler, J. " Feature set search algorithms," in *Pattern Recognition and Signal Processing*, C.H. Chen, Ed., Sijthoff and Noordhoff, The Netherlands, 1978.

[9] Vafaie, H., and De Jong, K.A., "Improving the performance of a Rule Induction System Using Genetic Algorithms," *Proceedings of the First International Workshop on MULTISTRATEGY LEARNING*, Harpers Ferry, W. Virginia, USA, 1991.