IMPROVING A RULE INDUCTION SYSTEM USING GENETIC ALGORITHMS

Haleh Vafaie
(George Mason University)

Kenneth De Jong
(George Mason University)

Abstract

The field of automatic image recognition presents a variety of difficult classification problems involving the identification of important scene components in the presence of noise, changing lighting conditions, and shifting view points. This chapter describes part of a larger effort to apply machine learning techniques to such problems in an attempt to automatically generate and improve the classification rules required for various recognition tasks. The immediate problem attacked is that of texture recognition in the presence of noise and changing lighting conditions. In this context standard rule induction systems like AQ15 produce sets of classification rules which are not necessarily optimal with respect to: 1) the need to minimize the number of features actually used for classification, and 2) the need to achieve high recognition rates with noisy data. This chapter describes one of several multistrategy approaches being explored to improve the usefulness of machine learning techniques for such problems. The approach described here involves the use of genetic algorithms as a "front end" to traditional rule induction systems in order to identify and select the best subset of features to be used by the rule induction system. The proposed approach has been implemented and tested on difficult texture classification problems. The results are encouraging and indicate significant improvements can be obtained from a multistrategy approach in this domain.

1. INTRODUCTION

In recent years there has been a significant increase in research on automatic image recognition in more realistic contexts involving noise, changing lighting conditions, and shifting viewpoints. The corresponding increase in difficulty in designing effective classification procedures for the important components of these more complex recognition problems has led to an interest in machine techniques as a possible strategy for automatically producing classification rules. This chapter describes part of a larger effort to apply machine learning techniques to such problems in an attempt to generate and improve the classification rules required for various recognition tasks. The immediate problem attacked is that of texture recognition in the context of noise and changing lighting conditions. In this context standard rule induction systems like AQ15 produce sets of classification rules which are not necessarily optimal in two respects. First, there is a need to minimize the number of features actually used for classification, since each feature used adds to the design and manufacturing costs as well as the running time of a recognition system. At the same time there is a need to achieve high recognition rates in the presence of noise and changing environmental conditions.

This chapter describes one of several multistrategy approaches being explored to improve the usefulness of machine learning techniques for such problems (see, for example, [Bala, 1992; chapter 18 of this book]). The approach described here involves the use of genetic algorithms as a "front end" to traditional rule induction systems in order to identify and select the best subset of features to be used by the rule induction system. The proposed approach has been implemented and tested on difficult texture classification problems. The results are encouraging and indicate significant advantages to a multistrategy approach in this domain.

2. FEATURE SELECTION

Since each feature used as part of a classification procedure can increase the cost and running time of a recognition system, there is strong motivation within the image processing community 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 high recognition rates under difficult conditions. This has led to the development of a variety of techniques within the image processing community for finding an "optimal" subset of features from a larger set of possible features. These feature selection strategies fall into two main categories.

The first approach selects features independent of their effect on classification performance. This generally involves transforming the original features according to procedures such as those presented by Karhunen-Loeve or Fisher [Kittler 75; Tou,67] to form a new set of features with lower dimensionality than the original one. The difficulty here is in identifying an appropriate set of transformations so that the smaller set of features preserve most of the information provided by the original data and are more reliable because of the removal of redundant and noisy features [Dom 89].

The second approach directly selects a subset *d* of the available *m* features in such a way as to not significantly degrade the performance of the classifier system [Ichino 84a]. Many researchers have adopted this method and have created their own variations on this approach. For example, Blanz, Tou and Heydorn, and Watanabe, after ordering the features, use methods such as, simply take the first *d* features or a branch and bound technique to select a subset of these features [Dom 89]. The main issue for this approach is how to account for dependencies between features when ordering them initially and selecting an effective subset in a later step.

A related strategy involves simply selecting feature subsets and evaluating their effectiveness. This process then requires a ϶criterion function and a ϶search procedure

[Foroutan 85]. The evaluation of feature set effectiveness has been studied by many researchers. Their solutions vary considerably and include using a Bayesian estimate of the probability of error, the Whitney and Stern estimate of probability of error based on a k-nearest neighbor bound [Ichino 84b], or a measure based on statistical separability [Dom 89].

All of the above mentioned techniques involve the assumption that some *a priori* information (such as a probability density function) about the data set is available, even though in practice such information is generally unavailable [Foroutan 85]. Also, the search procedures used in these techniques to select a subset of the given features plays an important role in the success of the approach. Exhaustively trying all the subsets is computationally prohibitive when there are a large number of features. Non-exhaustive search procedures such as sequential backward elimination and sequential forward selection pose many problems [Kittler 78]. These search techniques do not allow for back tracking; therefore, after a selection has been made it is impossible to make any revisions to the search. Also, in order to avoid a combinatorial explosion in search time, these search procedures generally do not take into consideration any inter-dependencies that may exist between the given features when choosing a subset.

There has been very little research in the machine learning community on optimal feature selection. A literature search revealed only one unpublished report by Forsburg [Forsburg 76] in which a technique based on an adaptive random search algorithm of Lbov [Lbov 65] was implemented for, but not adequately tested on, the problem of feature selection. In the published literature, the machine learning community has only attacked the problem of optimal feature selection indirectly in that the traditional biases for simplicity (Occam's razor) lead to efficient induction procedures that produce individual rules (trees) containing only a few features to be evaluated. However, each rule (tree) can and frequently does use a different set of features, resulting in much larger cumulative features sets than those typically acceptable for image classification problems. This problem is magnified by the tendency of traditional machine learning algorithms to overfit the training data, particularly in the context of noisy data, resulting in the need for a variety of ad hoc truncating (pruning) procedures for simplifying the induced rules (trees).

The conclusion of these observations is that there is a significant opportunity for improving the usefulness of traditional machine learning techniques for automatically generating useful classification procedures for image recognition problems if an effective means were available for finding feature subsets which are "optimal" from the point of view of size and performance. Since genetic algorithms are best known for their ability to efficiently search large spaces about which little is known and since they are relatively insensitive to noise, they seem to be an excellent choice as a feature selection strategy for use with a more traditional rule (tree) induction system. In the following sections this approach is described in more detail by illustrating the design, implementation, and testing of such a multistrategy system in which AQ15 is used as the rule induction procedure and the problem setting is that of texture classification.

3. A MULTISTRATEGY APPROACH

The overall architecture of the multistrategy system is given in Figure 1. It is assumed that an initial set of features will be provided as input as well as a training set in the form of feature vectors extracted from actual images and representing positive and negative examples of the various classes for which rules are to be induced. A genetic algorithm (GA) is used to explore the space of all subsets of the given feature set. Each of the selected feature subsets is evaluated (its fitness measured) by invoking AQ15 with the correspondingly reduced feature space and training set, and measuring the recognition rate of the rules produced. The best feature subset found is then output as the recommended set of features to be used in the actual design of the recognition system. Each of these components is described in more detail in the following sections.

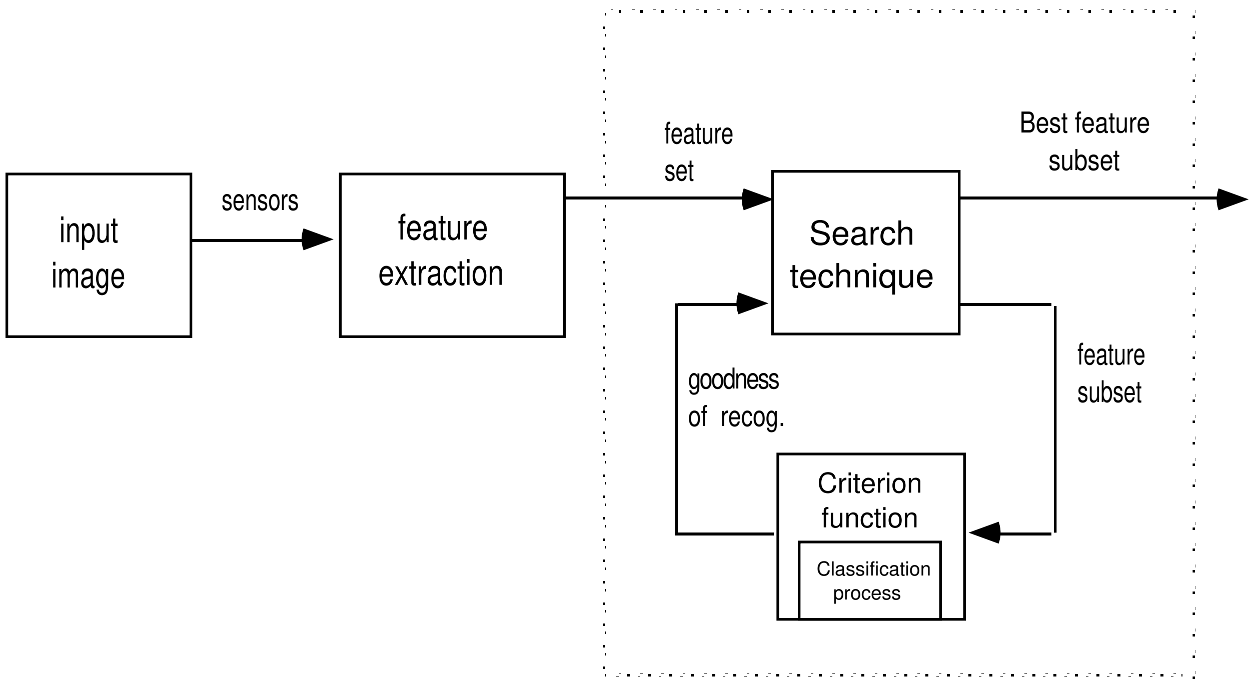

Figure 1: Block diagram of the adaptive feature selection process

3.1 Using Genetic Algorithms as the Search Procedure

Genetic algorithms (GAs) are adaptive search techniques initially introduced by Holland [Holland 75]. Genetic algorithms derive their name from the fact that their operations are similar to the mechanics of population genetics models of natural systems.

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". The result is an evolving 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 [De Jong 75]. 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 [De Jong 88].

The main issues in applying GAs to any problem are selecting an appropriate representation and an adequate evaluation function. In the following sections both of these issues are discussed in more detail for the problem of feature selection.

3.2 Representation Issues

The first step in applying GAs to the problem of feature selection is to map the search space into a representation suitable for genetic search. Since the main interest is in representing the space of all possible subsets of the given feature set, the simplest form of representation is to consider each feature in the candidate feature set as a binary gene. Then, each individual consists of fixed-length binary string representing some subset of the given feature set. An individual of length l corresponds to a l-dimensional binary feature vector X, where each bit represents the elimination or inclusion of the associated feature. For example, $x_i = 0$

represents elimination and $x_i = 1$ indicates inclusion of the *i*th feature. Hence, a feature set with five features can be represented as <x_1 x_2 x_3 x_4 x_5>. Then, an individual of the form <11111> indicates inclusion of all the features, and <11010> represents the subset where the third and the fifth features are eliminated.

The advantage to 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.

3.3 Evaluation procedures

Choosing an appropriate evaluation procedure is an essential step for successful application of GAs to any problem domain. Evaluation proceduress provide GAs with feed-back about the fitness of each individual in the population. GAs then use this feed-back to bias the search process so as to provide an improvement in the population€s average fitness. For the feature selection problem, the performance of a feature subset is measured by applying the evaluation procedure presented in Figure 2.

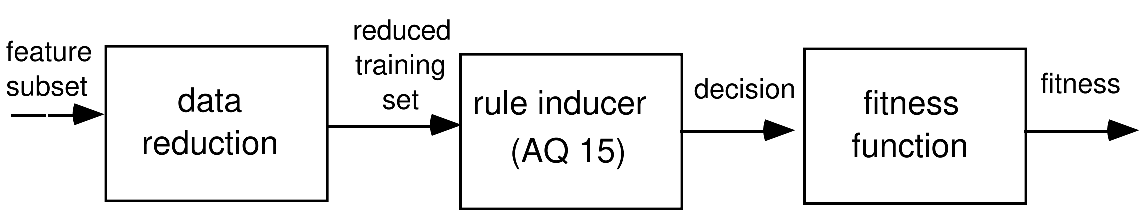

Figure 2: Feature set evaluation procedure

The evaluation procedure as shown is divided into three main steps. After a feature subset is selected, the initial training data, consisting of the entire set of feature vectors and class assignments corresponding to examples from each of the given classes, is reduced. This is done by removing the values for features that are not in the selected subset of feature vectors.

The second step is to apply a rule induction process (AQ15) to the new reduced training data to generate the decision rules for each of the given classes in the training data. The last step is to evaluate the rules produced by the AQ algorithm with respect to their classification performance on the test data. Each of these steps will be discussed in more detail.

3.3.1 The AQ algorithm

The AQ algorithm is a rule induction technique used to produce a complete and consistent description of classes of examples [Michalski 86, 83]. A class description is formed by a collection of disjuncts of decision rules describing all the training examples given for that particular class. A decision rule is simply a set of conjuncts of allowable tests of feature values.

For the studies reported here AQ15 was used as the rule induction component. This system (for more detail see [Michalski 86]) requires a set of parameters, feature descriptions, and a training set as input. It then uses the given parameters to direct the AQ algorithm in its process of searching for a complete and consistent set of class descriptions. The following is a simplified example of decision rules produced by AQ15 for a particular class in which instances were represented by 9 integer-values features:

$< x_1, x_2, \ldots, x_9 >$ is a member of the class if:

[x_1 = 3 to 7] and [x_4 = 4 to 9] and [x_8 =1 to 10] (total:9, unique 6)

or

[x_1 = 6 to 8] and [x_5 = 4 or 5 or 9] (total:5, unique 3)

or

[x_2 = 4] and [x_3 = 3 to 9] and [x_7 = 2 to 6] and [x_9 = 1 to 8] (total:2, unique 1)

This class is described by three decision rules, where each covers a total of 9, 5, and 2 training examples respectively. The term эunique refers to the number of positive training examples that were uniquely covered by each rule. Notice that, although each individual rule involves the testing of only a few features, the entire rule set involves 8 of the 9 features provided.

3.3.2 Fitness function

In order to use genetic algorithms as the search procedure, it is necessary to define a fitness function which properly assesses the decision rules generated by the AQ algorithm. The fitness function must be able to discriminate between correct and incorrect classification of examples, given the AQ created rules. Finding an appropriate function is not a trivial task, due to the noisy nature of most image data. The procedure as depicted in Figure 3 was developed to best achieve this goal.

The fitness function takes as input a set of feature or attribute definitions, a set of decision rules created by the AQ algorithm, and a collection of testing examples defining the feature values for each example. The fitness function then evaluates the AQ generated rules on the testing examples as follows.

For every testing example a match score (which will be described in more detail) is evaluated for each of the classification rules generated by the AQ algorithm, in order to find the rule(s) with the highest or best match. At the end of this process, if there is more than one rule having the highest match, one rule will be selected based on the chosen conflict resolution process (explained later). This rule then represents the classification for the given testing example. If this is the appropriate classification, then the testing example has been recognized correctly. After all the testing examples have been classified, the overall fitness function is evaluated by adding the weighted sum of the match score of all of the correct recognitions and

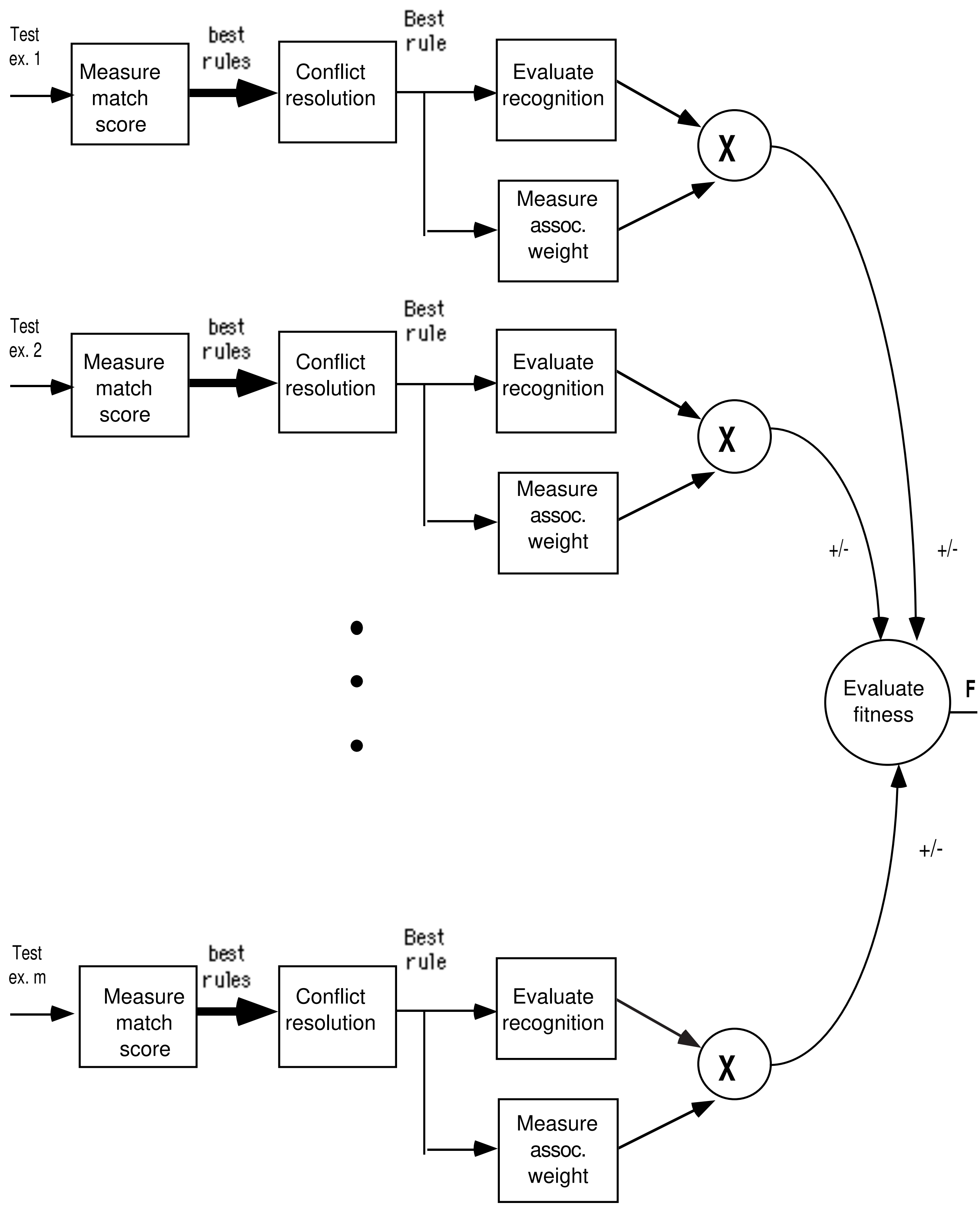

Figure 3: Block diagram for the fitness function **F**

subtracting the weighted sum of the match score of all of the incorrect recognitions. More formally:

$$\mathbf{F} = \sum_{i=1}^{n} S_i * W_i - \sum_{j=n+1}^{m} S_j * W_j$$

where :

S_i is the best match score evaluated for the test example *i,*

n is the number of testing examples that were classified correctly,

m is the total number of testing examples, and

W_i is the weight allocated to the test example *i.*

The range of the value of **F** is dependent on the number of testing events and their weights. In order to normalize and scale the fitness function **F** to a value acceptable for GAs, the following operations were performed:

$$\text{Fitness} = 100 - [(\mathbf{F} / TW) * 100]$$

where:

TW= total weighted testing examples

$$= \sum_{i=1}^{m} W_i$$

As indicated in the above equations, after the value of **F** was normalized to the range [-100, 100], the subtraction was performed in order to ensure that the final evaluation is always positive (the most convenient form of fitness for GAs). Hence, fitness values for feature subsets fall in the range [0, 200]) with lower values representing better classification performance. For example, a fitness value of zero is achieved when all the testing examples are completely and correctly recognized ($S_i = 1$). Selecting four equally weighted classes, each containing 100 testing examples (m = 400, W_i = 1 for every i), results in

$$\mathbf{F} = \sum_{1}^{400} 1*1 = 400$$

and

$$\text{Fitness} = 100 - [(400/400) * 100] = 0.$$

There are two main factors that define how the fitness function evaluates the performance of the decision rules on the testing examples: 1) the method used to calculate the degree to which rules match an example, and 2) the procedure for assigning "importance" weights to each correct/incorrect classification.

Match score (S):

The match between a rule and a testing example may be treated as strict or partial. In strict matching, the degree of consonance between a condition and a feature value of a testing example is treated as a boolean value. That is, it evaluates to zero if the example's feature value is not within the condition€s specified range, and is one otherwise.

A rule€s overall match score is calculated as the minimum of the degree of match for each of the conditions in that rule. Therefore, for each condition in a given rule, if all the conditions match the example value, the match score is 1, otherwise it is 0.

$$S_S = \min (S_i)$$

With partial matching, the degree of match is expressed as a real number in the range [0, 1] and is computed as follows. For each condition in a given rule, if the feature in that condition has a linear domain, then the following distance measure is used as its match score:

$$S = (1 - |a_j - a_k| / n)$$

where:

a_j is the condition€s closest feature value to the testing example's value a_k,

a_k is the testing example€s feature value , and

n is the total number of values given for the feature.

For non-linear feature value domains, partial matching is defined to be identical to strict matching (i.e., the match score for a given condition is 1 when a match occurs, and is 0 otherwise). The total partial match score for a given rule is evaluated by averaging the match scores of all the conditions for the given rule.

$$S_p = \text{Average}\ (S_i)$$

To clarify this process, consider the following example of evaluating the match score for a testing example with the given set of rules. The feature set consists of three features: S_1, S_2, and S_3 with the following properties:

Feature	domain	Total number of Values
S_1	linear	10
S_2	nominal	4
S_3	linear	3

The rule set to be evaluated consists of:

Rule A: $[S_1 = 1 \text{ or } 2 \text{ or } 4]$

Rule B: $[S_1 = 1 \text{ or } 5]$ and $[S2 = 2]$

The testing example to be matched is:

S_1	S_2	S_3
4	1	0

This results in the following strict match scores:

Rule A => min(1, 1, 1) = 1

Rule B => min(0, 0, 1) = 0

and the following partial match scores:

Rule A => Ave(1, 1, 1) = (1+1+1) / 3 = 1.0
Rule B => Average((1 - | 5 - 4| / 10), 0, 1) = (.9 + 0 + 1) / 3 = 0.633

Conflict resolution:

For any testing example, after the appropriate match score is evaluated for all the rules generated by the AQ algorithm, a single rule representing the highest match score must be selected. However, there may be situations where more than one rule has the highest match score. This is when the conflict resolution process is used to select a single rule. This process is performed by selecting a rule with highest typicality, that is, a rule that covers the most training examples as specified by the total number of examples it covers (see example of section 3.3.1). However, there may be situations that all the selected rules have the same typicality. In this situation a rule is randomly selected as the one having the best match score.

In addition to the match score of a given testing example and the conflict resolution process, weights can be associated with the classes to be recognized and can thus play an important role in the evaluation of fitness.

weight (W):

This option is useful when encoding user or expert€s knowledge into the evaluation function. In the current implementation each class is given a weight according to its importance in the overall recognition process. Therefore, in order to measure the recognition rate accurately, appropriate individual weights need to be allocated to each of the testing examples. The weight of the recognized class is associated with a testing example if the given example is correctly recognized. For situations where an example is incorrectly recognized, the weight is the average of the weight of the correct class and the class that was recognized (incorrect class). Hence,

$$\text{Weight} = \begin{cases} W_c & \text{if } T_i \text{ was correctly recognized.} \\ (W_c+W_{in}) / 2 & \text{if } T_i \text{ was incorrectly recognized} \end{cases}$$

where :

W_c is the weight associated with the correctly recognized class,

W_{in} is the weight associated with the incorrectly recognized class, and

T_i is a testing example.

Consider the following illustrative example, consisting of only two classes (A and B). Suppose the weights assigned to the classes are:

class	weight
A	1
B	2

and the recognition results are:

example	correct class	recognized class
1	A	A
2	B	B
3	B	A
4	A	B

Then, the associated weight for each example is:

example	weight
1	1
2	2
3	1.5
4	1.5

In summary, the "fitness" of a particular subset of features is determined by running AQ on the correspondingly reduced training data sets, and then measuring the classification accuracy of the rules produced by AQ on a testing data set using the weighted evaluation function described above. This fitness value is the key element in biasing the GA-based adaptive search of the space of feature subsets towards even more "fit" subsets.

4. EXPERIMENTAL RESULTS

The experiments described in this section were performed using two existing systems, GENESIS, a genetic algorithm package [Grefenstette 84], and AQ15, an AQ-based rule induction system [Michalski 86]. Since each of these systems has a number of user-controlled parameters, the approach taken was to attempt to identify reasonable settings for these parameters and then hold them fixed for the entire set of experiments.

In the case of GENESIS the standard parameter settings recommended in [De Jong 75] were used (a population size=50, a mutation rate= 0.001, and a crossover rate=0.6) without any further experimentation. Preliminary experiments with AQ15 suggested that there were only a few parameters that significantly affected classification performance. Additional experiments were performed to find reasonable values for these parameters. The resulting settings (Mode=IC, Ambig=NEG, Trim=Spec, and Maxstar=13) were then used through out all the experiments in conjunction with the standard default setting for the LEF function (which included a preference for small numbers of features).

The proposed multistrategy approach was tested on four texture images randomly selected from Brodatz [Brodatz 66] album of textures. These images are depicted in Figure 4. Two 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. These feature vectors were divided equally between training examples used for the generation of decision rules, and testing examples used to measure the performance of the produced rules. In the experiments reported here, equal recognition weights (i.e., W=1) were assigned to all the texture classes.

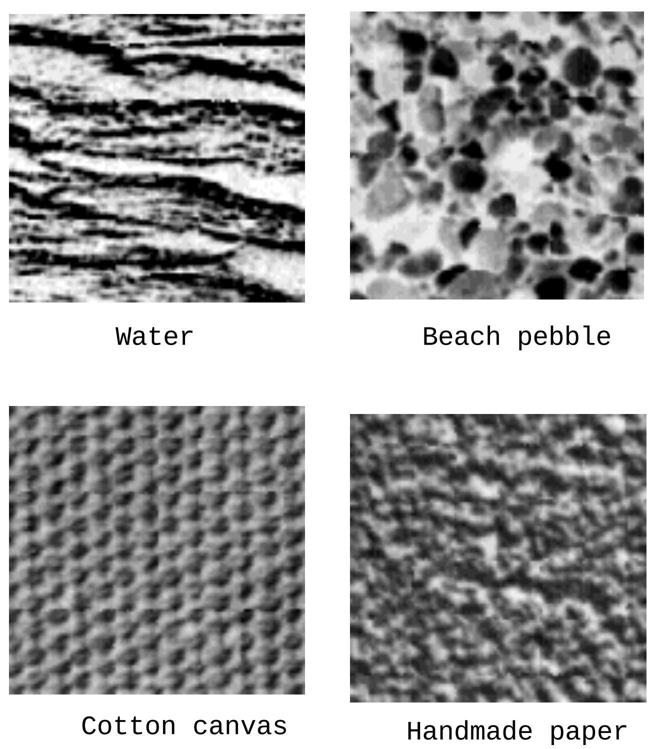

Figure 4: The texture images used in the experiments.

The experimental results are summarized in Figure 5 and Table 1 and provide rather encouraging support for the adopted multistrategy approach. Figure 5 shows the steady improvement in the fitness of the feature subsets being evaluated as a function of the number of generations of the genetic algorithm. This indicates very clearly that the performance of rule induction systems (as measured by recognition rates) can be improved in these domains by appropriate feature subset selection.

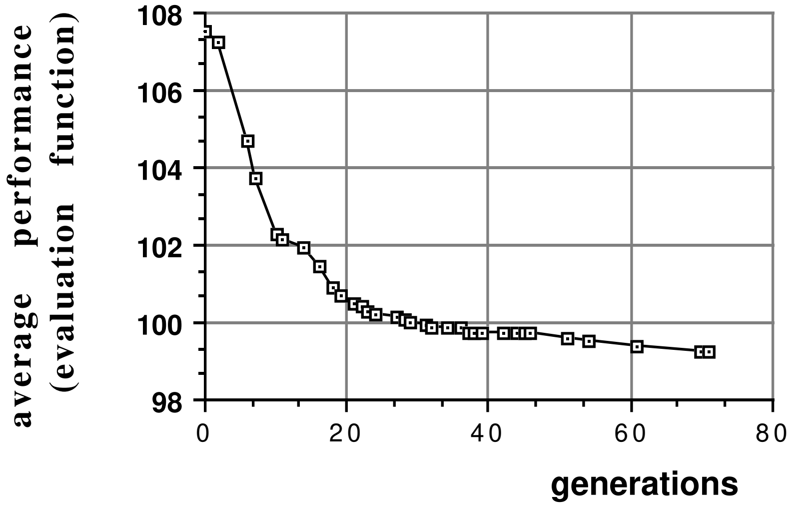

Figure 5: The improvement in feature set fitness over time

	Best performance [0,200]	Size of feature set	Learning time (sec)	complexity (#complexes/class)
AQ	111.5	18	5.2	301.5
GA-AQ	95.5	9	3.3	211.5

Table 1: Comparative results of the experiments

Table 1 suggests that there are other advantages as well when one compares the results of the multistrategy approach with the best performance obtainable using AQ15 alone. The result of the feature selection process was to reduce the initial feature set consisting of 18 elements to a subset of having only 9 elements for the best performing individual. This represented a 50% reduction in the number of features.

Another advantage of using this approach is that choosing the appropriate subset of features reduces the time required to perform rule induction on large data sets (which are typical in the image processing world). This is a direct result of feature selection process. The time required to generate decision rules grows rather quickly as a function of the number of features. Attempting rule learning on problems involving more than 50-100 features is generally computationally infeasible. Any significant reduction in the number of features used translates directly into reduced rule learning time, thus extending the applicability of these rule induction techniques to many problems that were previously computationally prohibitive.

Table 1 suggests one more benefit of this multistrategy approach. If one measures the complexity of the generated rules in terms of the number of complexes (conditions) per class, the feature selection process also reduces the complexity of generated class descriptions.

5.0 SUMMARY AND CONCLUSIONS

The experimental results obtained indicate the potential power of applying a multistrategy approach to the problem of learning rules for classification of texture images. The reported results indicate that an adaptive feature selection strategy using genetic algorithms can yield a significant reduction in the number of features required for texture classification and simultaneously produce improvements in recognition rates of the rules produced by a rule induction system (AQ15). In addition, the reduction of the number of features improved the execution time required for rule induction substantially. This is an important step towards the application of machine learning techniques for automating the construction of classification systems for difficult image processing problems.

More testing is needed in order to substantiate the results presented here. The robustness and scaling up characteristics of the presented system must be assessed with additional experiments involving more texture classes, larger feature sets, and more testing and training examples.

There are a number of interesting directions one might pursue in extending the ideas presented here. One extension has already been explored and described by Bala et al.[Bala 92; chapter 18 of this book], in which further improvements in classification performance can be achieved by using genetic algorithms to refine the AQ generated rules.

Another straightforward direction would be to use GAs with some background knowledge. For example, including user and/or expert provided estimates of the relative importance of various features for texture classification could further improve speed and performance.

A much more provocative, but more difficult, extension currently under development involves the addition of some form of constructive induction. That is, a more complex set of features could be constructed by including combinations of the given features in the initial set. The feature selection process could then be applied to this set in order to find even more effective feature subsets.

ACKNOWLEDGEMENTS

This research was done in the Artificial Intelligence Center of George Mason University. The activities of the Center are supported in part by the Defense Advanced Research Projects Agency under grants administrated by the office of Naval Research, No. N00014-87-K-0874, and No. N00014-91-J-1854, in part by the Office of Naval Research under grant s No. N00014-88-K-0226, No. N00014-88-K-0397, No. N00014-90-

J-4059, and No. N00014-91-J-1351, and in part by the National Science Foundation under grant No. IRI-9020266.

The authors also wish to thank Dr. Pachowicz and Jerzy Bala for their informative comments.

References

Bala, J.W., De Jong, K., and Pachowicz, Peter ϶Learning Noise Tolerant Classification Procedures by Integration of Inductive Learning and Genetic Algorithms, *Machine Learning: A Multistrategy Approach*, Vol. IV, R.S. Michalski and G. Tecuci (Eds.), Morgan Kaufman Publishers, San Mateo, 1992.

Brodatz, P. ϶*A Photographic Album for Arts and Design*, Dover Publishing Co., Toronto, Canada, 1966.

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.

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

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.

Forsburg, S., эAQPLUS: An Adaptive Random Search Method for Selecting a Best Set of Attributes from a Large Collection of Candidates, Internal Report, Department of Computer Science, University of Illinios, Urbana, 1976.

Foroutan, I., and Sklansky, J. эFeature selection for piece wise linear classifiers, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1985.

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

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

Ichino, M., and Sklansky, J. эOptimum Feature selection by zero-one Integer Programming, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 14, No. 5, 1984a.

Ichino, M., and Sklansky, J. эFeature selection for linear classifiers, *Seventh International Conference on Pattern Recognition,* Montreal, Canada, Vol. 1, 1984b.

Kittler, J., Feature selection method based on the Karhunen-Loeve expansion, *Pattern Recognition Theory and Application*, Fu , K.S. and Whinston, A.B., Ed., Bondol, France, 1975.

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

Lbov, G.S. "Wybor effektiwnosti sistemy zabisimych priznakow," in a collection of IM SO AN USSR Computer Systems, Nowosibivsk, Vol. 19, 1965.

Michalski, R.S., Mozetic, I., Hong, J.R., and Lavrac, N.. эThe Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains, *AAAI*, 1986.

Michalski, R.S., and Stepp R. E.. эLearning from Observation: Conceptual clustering, *Machine Learning: An Artificial Intelligence Approach,* R. S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Tioga, Palo Alto, CA, 1983.

Tou, J.T., and Heyden, R.P., э Some approaches to optimal feature selection, *Computer and Information Sciences-II*, Tou, J.T. Ed., New York, Academic press, 1967.