Acknowledgments

This work was supported in part by Michigan State University's Center for Microbial Ecology and the Beijing Natural Science Foundation of China.

References:

[1] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From Data Mining to Knowledge Discovery: An Overview, *Advances in Knowledge Discovery and Data Mining*, 1996, p1-34.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison Wesley, 1989.

[3] S. W. Wilson, D. E. Goldberg, ACritical Review of Classifier, *Proc. Third Inter. Conf. Genetic Algorithms and their Application* (ICGA), 1989,

[4] S. Smith, A Learning System Based on Genetic Algorithms, *Ph.D Dissertation* 1980 Computer Science, University of Pittsburgh.

[5 K. De Jong, Genetic Algorithms: A 10 Years Perspectives, *Proc. First International Conf. Parallel Problem Solving from Nature*, 1990, 169-177. P244.

[6] Anil K. Jain, *Artificial Neural Networks and Statistical Pattern Recognition.* Editor preface, 1991, Elsevier Science Publishers B.V.

[7] W. Siedlecki and J. Sklansky, On Automatic Feature Selection Internat. *Journal of Pattern Recognition and Artificial Intelligence*. Vol 2, No.2 1988, 197-220.

[8] W. Siedlecki and J. Sklansky, A Note on Genetic Algorithms for Large-Scale Feature Selection, P*attern Recognition Letters*, 10 1989, 335-347.

[9] W.F. Punch, E.D. Goodman, Min Pei, et al. Further Research on Feature Selection and Classification Using Genetic Algorithms. *Proc. Fifth Inter. Conf. Genetic Algorithms and their Applications (ICGA)*, 1993, p.557.

[10] M. Pei, E.D. Goodman, W.F. Punch, Y. Ding, Genetic Algorithms For Classification and Feature Extraction. 1995 *Annual Meeting, Classification Society of North America*, June 1995.

[11] S.Wilson. Classifier Systems and the Animate Problem. *Machine Learning Journal* 2: 199-2 28, Kluwer Academic Publishers, Boston, 1987.

[12] Bonelli, Pierre and Parodi, Alexandre (1991). An Efficient Classifier System and its Experimental Comparison With Two Representative Learning Methods on three Medical Domains. *Proc. Fourth Inter. Conf. Genetic Algorithms and their Application (ICGA)*, 1991, p. 288.

[13] R. L. Riolo. Modeling Simple Human Category Learning with a Classifier System. *Proc. Fourth Inter. Conf. Genetic Algorithms and their Application (ICGA)*, 1991, p. 324.

[14] J. Oliver, Discovering Individual Rules: An application of Genetic Algorithms, *Proc. Fifth Inter. Conf. Genetic Algorithms and their Application (ICGA)*, 1993, p.216.

[15] T. A. Sedbrook, H. Wright, R. Wright, Application of a Genetic Classifier for Patient Triage, *Proc. Fourth Inter. Conf. Genetic Algorithms and their Application (ICGA)*, 1991, p. 334.

[16] G.E. Liepins, L.A. Wang, Classifier System Learning of Boolean Concepts, *Proc. Fourth Inter. Conf. Genetic Algorithms and their Application (ICGA)*, 1991, p. 318.

ticularly important to biological researchers, as it can be used to derive new hypotheses and research efforts. In particular, for our biological examples, it was shown that HDPDCS can indeed distinguish between different bacterial communities on the basis of which substrates are being used -- i.e., 2,4-D degraders -- based on a standardized test that measures substrate uptake. This may help to distinguish what substrates are available in each of the communities, and may have ramifications in determining which bacteria are suitable for bioremediation of particular environments. In practical problems, one can derive several different patterns which includes different numbers of features to characterize the same classes.

Finally, the errors of classification in each class are not uniform across the data sets examined. This information may also provide some hints for research in the domain being studied.

There are some problems of classifiability of high dimensionality data sets noted for the GA/RULE approach. The quality of the data sets -- i.e., whether or not they have overlapping features in some classes, or lack sufficient information for discrimination among the classes -- affects pattern discovery and classification accuracy. The quantity of training samples given, or the ratio of training sample size to dimensionality, are crucial factors in the design of a pattern discovery and classification system. The sample (training) dataset needs to be representative, and it must also be large enough to allow for effective training. Otherwise, it will allow 'false' patterns and rules to be induced, based on a few examples which do not span the space of possible class membership.

4. Conclusion and Future Research

In this paper we have shown that genetic algorithms can be a good data mining tool, and play an important role in pattern discovery and classification for high dimensionality and multiclass data. The basic approach is a classification -- feedback -- pattern evaluation technique. This is an automated pattern search method which utilizes feedback information from the classifier being designed to change the decision space. The GA/RULE does this by modifying the classifier pattern of the decision rule using inductive learning and evolution to improve the performance of the classifier. The results achieved by this approach indicate potential application in many other fields. We plan to go on to test various data sets with different types of features, design appropriate encodings. We also plan to apply parallel genetic algorithms to discover several disjunctive patterns on different nodes (island) and use coevolution to increase the quality of discovery patterns. The final goal is to put this approach into practical KDD systems.

probability 0.3 for a 0, 0.3 for a 1, and 0.4 for a #. The population size was 200, 40% of individuals (random) survived without change from each generation, elitism preserved the best individual, the crossover rate was 0.6, the mutation rate was 0.005, and fitness-proportional roulette wheel selection was used.

In several runs, the HDPDCS system discovered a reasonable classification rule and corresponding patterns for each data set. The HDPDCS outperformed a standard KNN method in every case, and its performance approached that of the hybrid GA/KNN method in most cases. It is most important to note, however, that the computation time required for the hybrid GA/KNN method running under similar circumstances was about 14 days (if run on a single SPARC I processor), whereas the HDBPCS required only 3 hours, although on a somewhat faster processor. The average accuracy results of six runs and the numbers of valid features after selection are shown in Table 1 below.

The useful patterns we discovered can be listed for every class of each different data set. Here we show the three patterns for *P. putida* isolates (104) of Rhizosphere data in Table 2. In the original data set each data sample has 96 features. After learning, we found patterns for three classes using only 8 features.

		KNN Correctness rate	GA/KNN Correctness rate	HDPDCS Correctness rate	Number of features	Number of valid features
Rhizosphere data	Training	71.00%	82.00%	80.00%	96	23
	Test	68.00%	79.90%	70.00%		
P.fluorescens	Training			86.80%	96	13
P. putida	Training			87.37%	96	8
2,4-D data	Training	93.36%	99.17%	98.61%	96	22
	Test	91.70%	98.00%	96.00%		

Table 1: Classification results using HDPDCS system

Feature index	Feature	9	21	35	41	52	66	81	95
Class 1	rhizosphere	0	0	0	0	1	1	0	#
Class 2	non-rhizosphere	1	#	#	1	0	1	#	1
Class 3	crop residue	0	1	1	0	0	0	1	0

Table 2: Discovery Patterns for *P. putida* data using HDBPCS system

Information about the patterns of the classes such as that shown in Table 2 is par-

After we delete all redundant features, we can easily pick out the useful patterns for the various classes. These patterns provide information useful to domain experts for evaluating and interpreting the characteristics of the classes, which constitutes "knowledge" about them. In the end, near-optimal patterns can be drawn from the best rule which keeps the classification performance at a certain level, and this makes knowledge available for use in tasks of prediction and description.

3.6 A Real-World Problem -- Biological Pattern Discovery and Classification.

We explored pattern discovery and classification of several real-world datasets exemplifying high-dimensionality biological data. Researchers in the Center for Microbial Ecology (CME) of Michigan State University have sampled various agricultural environments for study. Their first experiments used the Biolog© test as the discriminator. Biolog consists of a plate of 96 wells, with a different substrate in each well. These substrates (various sugars, amino acids and other nutrients) are assimilated by some microbes and not by others. If the microbial sample processes the substrate in the well, that well changes color, which can be recorded photometrically. Thus large numbers of samples can be processed and characterized based on the substrates they can assimilate. Each microbial sample described was tested on the 96 features provided by Biolog (for some experiments, extra taxonomic data were also used); the value of each feature is either 0 or 1.

Using the Biolog test suite to generate data, two test sets were used in showing the effectiveness of our approach.

Rhizosphere Data Set

Soil samples were taken from three environments found in agriculture:

1) near the roots of a crop (rhizosphere), 2) away from the influence of the roots (non-rhizosphere), or 3) in a fallow field (crop residue). There are 300 samples in total, 100 samples per area. The data set contains *P. fluorescens* isolates (196 isolates) and *P. putida* isolates (104). These two isolates represent two different species of bacteria which are phylogenetically closely related.

2,4-D Data Set

Soil samples were collected from a site that is contaminated with 2,4-D (dichlorophenoxyacetic acid), a pesticide. There were 3 classes, based on three genetically similar microbial isolates which show the ability to degrade 2,4-D. There were a total of 232 samples.

We applied the HDPDCS to the two datasets described above. The GA parameters used were those that gave the best results over a number of runs. The initial populations were created randomly, with the value at each vector position generated using

• **Genetic Operators**

Crossover was standard one-point crossover. Mutation was standard single bit-modification. Operations were performed only on the condition part of the classifiers/rules. Elitism was used to keep the best solution in the population, and mating was generational, not steady-state. Offspring replaced their parents.

• **Discovering Useful Patterns**

To keep the complexity of the useful patterns and rules generated to a minimum, an optimization step was applied to remove redundant features

..

$$\begin{bmatrix} y_{11}, & y_{12}, & \dots & y_{1N} \\ y_{21}, & y_{21}, & \dots & y_{2N} \\ \dots & \dots & \dots & \dots \\ y_{k1} & y_{k2} & \dots & y_{kN} \end{bmatrix} \Rightarrow \omega$$

For each pattern of different classes of a rule, we arrange the class-feature vectors as a matrix above; then for the binary case, we determine whether every class-feature vector has the same value (1 or 0) or a "don't care" (#) at any feature position (column); for the nominal case, we determine whether every class-feature vector has the same string at any feature position. If either of these situations occurs, the number of the n_don'tcare variable is incremented, as this feature is not contributing to the discrimination of classes; e.g., in the binary case

for any column i,

if $\forall(y_{mi}, y_{ni})\ \{(y_{mi} = y_{ni})$ or $(y_{mi} = \#)$ or $(y_{ni} = \#)\}$,

where m, n =1, 2, . . ., k, $m \neq n$,

then the *i*th feature is redundant for classification

else valid.

For the case of numerical features, the situation is more complicated. We might, for example, set as a criterion for matching that every class-feature vector have the same interval (i. e., lower and upper bounds) at any feature position, but that appears to be overly restrictive. Instead, one might, for example, require for a match only that the upper bounds are within epsilon of each other, for some given positive epsilon, and that the lower bounds are similarly within epsilon of each other. For other cases we can do same thing to remove redundant features.

It is convenient to assume that the discriminant function $g_j(x)$ uses scalar values and that the pattern $\bar{x}$ belongs to the *j*th class if and only if

$g_j(x) > g_i(x)$, for *j*, $i = 1, 2, \ldots, k$, $i \neq j$

This is equivalent to

$R(x) = \omega_j$, when $g_j(x) = \max \{g_i(x)\}$ $i = 1, 2, \ldots, k$,

The decision in N-dimensional feature space between pattern classes ω_j and ω_i is defined by the equation

$g_j(x) - g_i(x) = 0$

The above equation represents the decision surface. When a pattern $\bar{x}$ falls on the boundary of this surface, it is ambiguous as to which classification to make. It would be advantageous to allow the classifier to withhold making a decision and reject the pattern. However, in the results described here, the classification of the first pattern to achieve the maximum number of matches is arbitrarily assigned.

3.5 Evolution of the Rule Classifier and Pattern Discovery Using a GA

Running HDPDCS yields a small set of "best" classification rules which classify the training set more accurately than other rule sets. By examining those "good" rules, one can determine those features which are most useful for classification. In order to find the interesting patterns of each class as simply as possible, we must keep the number of features used to a minimum and provide a mechanism by which one can foster not only accuracy of classification, but also minimum cardinality of the feature set used. The problem then becomes a multiobjective or multicriterion optimization problem. One way to address this is to add another term to the fitness function and to choose different proportions of wild cards # (in the binary and nominal value cases) in the rule classifiers of the initial population.

- **Fitness Function**

  Each rule's fitness is based on the classification of the known samples. The form of the fitness function is as follows:

  Fitness = CorrectPats/TotPats + $\alpha *$ **n_don'tcare/TotPats.**

  where TotPats is the number of total training samples to be examined and CorrectPats is the number of training samples correctly classified by the rule, and n_don'tcare is the number of invalid features (see below) for classification. The constant α is used to tune the two terms of the fitness function, and its value is determined on a problem-specific basis. This fitness function could be used to guide the GA to search out best rule sets as a multiobjective optimization problem, if the GA also searched the space of possible α's.

$$g_j(x) = [W]\left[\sum_{i=1}^{N} m(x_i, y_{ji}) = \begin{cases} 1 & (x_i \ni y_{ji}) \\ 0 & (x_i \notin y_{ji}) \end{cases}\right]$$

For numerical type of feature:

$$g_j(x) = [W]\left[\sum_{i=1}^{N} m(x_i, y_{ji}) = \begin{cases} 1 & (y_{jilower} \le x_i \le y_{jiupper}) \\ 0 & (x_i \notin [y_{jiupper}, y_{jilower}]) \end{cases}\right]$$

where

$$W = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & w_N \end{bmatrix}, \quad \text{or } \overline{w}^* = [w_1, w_2, \ldots, w_N], \qquad \overline{x} = [x_1, x_2, \cdots, x_N]$$

- $w_i \in \{0,1\}$, $1 \le i \le N$.

  When we use only binary values for $\overline{w}^*$, we interpret it as follows: if the *i*th weight component is one, then the *i*th feature was preserved in feature space; otherwise, the feature was discarded from the feature set. Thus we perform feature selection to define an optimal subset pattern and (potentially) reduce the dimensionality of the original data pattern.

- $w_i \in [a, b]$, such as $w_i \in [0.0, 10.0]$, $1 \le i \le N$.

  In this case, we do feature extraction by allowing the values of the weight components to range over some values [a, b], such as 0.0 to 10.0 (presuming that the features are first "normalized" to some standard range). That amounts to searching for a relative weighting of features that gives optimal performance on classification of the known samples. Thus we are selecting for features in a linearly transformed space. Those weight components that move towards 0 indicate that their corresponding features are not important for the discrimination task. Essentially, those features "drop out" of the feature space and are not considered. Any feature that moves towards the maximum weight indicates that the classification process is sensitive to changes in that feature. That feature's dimension is elongated, which probably reflects that an increased separation between the classes gave better resolution. The resulting weights indicate the usefulness of a particular feature, sometimes called its discriminatory power.

$\bar{x} = [x_1, x_2, \cdots, x_i, \cdots, x_N]$. $\quad i = 1, 2, \cdots, N,$

where $x_i \in \{0,1\}$, or nominal type, or numerical type in a defined range;

and N is the number of features

3.4 Evaluation of the Rules (Classifiers)

Each rule in the population is evaluated by applying it to classify the training data set. In this procedure, every class-feature vector of the rule's condition is compared with each training vector at every feature position (i.e. every component). For a binary value, a 0 in the class-feature vector matches a 0 in the training vector, a 1 matches 1 and a # don't-care character matches either 0 or 1. For nominal types, the value matches if the value is present in the enumerative set represented by the string of the feature. For numerical types, the value matches if the value is within the interval of the feature represented by the pair of strings (lower and upper bounds). Thus, for a training set with three classes, the training vector would be compared with the three class-feature vectors of each rule. The number of matching features in each pattern of class-feature vector of the rule's condition is counted, and the pattern of the rule's class-feature vector with the highest number of matches determines the classifier's message -- the class of the sample (handling of ties is described below). Since the class of each training sample is already known, this classification can then be judged correct or not. Based on the accuracy of classification, the decision rule can be directly rewarded or punished. Based on this rule "strength," the GA can evolve new rules.

Let us now discuss the use of classifier rules as pattern discriminators. A given data sample $\bar{x}$ of unknown class normally can be assigned to any of the k classes. Thus, we have k possible decisions. Let $R(\bar{x})$ be one of the above rule classifiers -- i.e., a function of $\bar{x}$ that tell us which decision to make for every possible data pattern $\bar{x}$. For example, $R(\bar{x}) = \omega_j$ denotes the decision to assign pattern $\bar{x}$ to class ω_j. During the classification step, class membership needs to be determined based on the comparison of k discrimination functions $g_1(\bar{x})$, $g_2(\bar{x})$, ... , $g_k(\bar{x})$, or, say, by k matching functions $m(x_i, y_{ji})$ computed for the input pattern under consideration. A discrimination function or matching function for two different cases is defined as follows:

For binary type of feature:

$$g_j(x) = [W]\left[\sum_{i=1}^{N} m(x_i, y_{ji}) = \begin{cases} 1 & (x_i = y_{ji}) or (y_{ji} = \#) \\ 0 & (x_i \neq y_{ji}) and (y_{ji} \neq \#) \end{cases}\right]$$

For nominal type of feature:

of examiner as a classification predicate for the input data pattern. All class-feature vectors are put together to determine which features form useful patterns when used in this rule's decision for classification. The <message> part of the classifier is a class variable ω which indicates into which class the rule classifier places an input data element based on matching against the pattern of each feature vector. A classifier rule therefore takes the following form:

<classifier> : : = <condition> : <message>

The specific form of a rule for a class-feature vector, consisting of one pattern of N elements, is:

$$(y_{11}, ..., y_{1i}, ..., y_{1N}), \ ... , (y_{j1}, ..., y_{ji}, ..., y_{jN}), \ ... , \ (y_{k1}, ..., y_{ki}, \ ... y_{kN}) : \omega ,$$

where i = 1, 2, · · · , N; j = 1, 2, · · · , k, and

where ω is a variable whose value can be one of the k classes.

Class-feature vectors consisting of multiple patterns of N elements in disjunctive normal form as described as follows:

$$(y_{11}, ..., y_{1i}, ..., y_{1N})_1, \ ... (y_{11}, ..., y_{1i}, ..., y_{1N})_q,$$
$$(y_{j1}, \ ..., y_{ji}, ..., y_{jN})_1, \ \ ... (y_{j1}, \ ..., y_{ji}, ..., y_{jN})_r,$$
$$... ,$$
$$(y_{k1}, ..., y_{ki}, ... y_{kN})_1, \ \ ... \ (y_{k1}, ..., y_{ki}, ... y_{kN})_1 : \omega ,$$

The <condition> is a simple pattern matching device. For the binary and nominal types of features, the alphabet of the class-feature vector consists of 0, 1, and a don't-care character; i.e., $y_{ji} \in \{0, 1, \#\}$.

When the value of a feature is of nominal type, each feature y_{ji} is a fixed-length binary string which enumerates the nominal values in order. For example, for the set of colors = {white, yellow, black}, a three-bit string feature in turn represents white, yellow, and black as present or not by 0 or 1. When the value of a feature is a numerical type, each feature y_{ji} is a pair of numerical values represented by a binary string whose length depends on the desired resolution. The pair of values $y_{ji} = (a, b)$ represents the upper and lower bounds, respectively, of an interval of the feature on the pattern.

3.3 Training Data

The training data consists of a set X of *training vectors* $\bar{x}$ each of size N (again, N indicating the number of features being used), and each with a known classification label:

The results of this process are two: 1) "good" rules are created for classifying "unknown" data, and 2) domain researchers are shown the interesting patterns and those features which are important for the classification, based on the features used by the rules.

Our HDPDCS, like most machine learning systems, performs supervised learning. The characteristics of the HDPDCS genetic algorithm are described in the following sections and in Fig. 2.

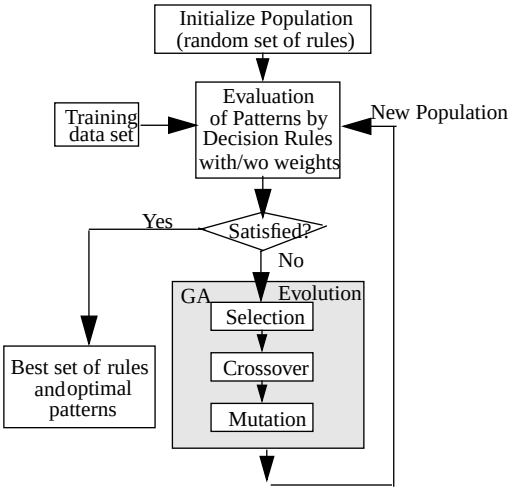

Figure 2: The structure of the GA/RULE approach

The objective of the GA/RULE approach is to find the optimal transformation that yields both the lowest classification error and patterns with smallest number of features. The basic tenet of this approach is to utilize a feedback linkage between pattern evaluation and classification, but in a way much different from GA/KNN [8]. GA/RULE directly manipulates a rule representation which is used for classification and evolving patterns.

3.2 Rule (Chromosome) Structure

Suppose there are k possible pattern classes $\omega_1, \omega_2, \cdots, \omega_k$. A classifier is a production rule which is used to make a decision assigning a pattern x to one of a set of classes ω_j, j = 1, 2, $\cdots$, k. The <condition> part of the rule in HDPDCS is a string which consists of k *class-feature vectors*, where k is the number of classes. Each class-feature vector consists of one pattern of N elements or multiple patterns of N elements in disjunctive normal form, where N is the number of features being used for the classification of that class k. Each pattern of a class-feature vector plays the role

ture space is selected or transformed into a new feature space with fewer features which (potentially) offers better separation of the pattern classes, which, in turn, facilitates discovery of nearly optimal patterns and improves the performance of the decision-making classifier. The criterion for optimality of the patterns of different classes is usually the probability of misclassification.

This GA data mining approach includes both the steps of selection, cleaning, transformation, projection of data, and pattern extraction -- two major steps in the whole KDD multi-step process [1].

3. GA/RULE Approach

One of great challenge for data mining from high dimensionality data is the computational cost. Algorithms are typically required to meet some acceptable computational efficiency limitations. In our previous research, the computational cost of the GA/KNN method [9] [10] we have used is very high and requires parallel or distributed processing to be attractive for large, high-dimensionality problems. We have explored whether we could decrease computational cost without sacrificing performance by directly generating rules -- i.e,. using a genetic algorithm combined with a production (rule) system. This inductive learning method is simpler to implement than the previous GA/KNN hybrid system, and requires substantially fewer computation cycles to achieve answers of similar quality. The system was developed for application to both classification and pattern discovery by feature extraction of high dimensionality feature patterns. Thus, we named our system the HDPDCS (High Dimensionality Pattern Discovery and Classification System) and first have applied it to two previous biological data sets (containing two species) with encouraging results (the original form was called HDBPCS) [10]. The accuracy of this method is significantly better than the classical KNN method, and the algorithm is significantly faster than our previous hybrid method for the classification of binary feature patterns. Our work builds on that of Wilson[11], Bonneli[12], Riolo[13], Oliver[14], Sedbrook et. al[15], and Liepins[16], all of whom studied systems for inductive learning of classification.

3.1 General Structure of the GA/RULE Approach

First, the binary feature pattern discovery and classification problem was selected for study because it is both a common problem and easy to represent in the HDPDCS. Then, general pattern discovery and classification were studied. As with other inductive learning approaches, this algorithm uses a collection of labeled “training” data to drive learning. What is characteristic of the approach is the “evolution” of classifiers/rules to classify the training data, and subsequent testing of the accuracy of those classifiers on data not used in classifier training.

cess.

In the following section we introduce the basic concept of this approach, then apply this approach first to a binary data set, showing the effectiveness of the approach on biological data. In further ongoing research, it is being extended to handle variety of feature types that include binary, nominal, numerical and structural features.

2. Basic Concept of the Approach

In the design of automatic pattern classifiers, feature selection and extraction are crucial to optimizing performance, and strongly affect classifier design. Ideally, the problems of feature selection and extraction on the one hand and classifier design on the other should never be considered independently. Yet, for practical considerations, most researchers make the simplifying assumption that the feature selection/extraction stage and the classification stage are independent. However, the ultimate goal is correct classification, and the intermediate step of feature extraction and dimensionality reduction is, in a sense, subservient to that goal, rather than being an end in itself. It would be better to couple pattern evaluation, including feature extraction, with effective classification techniques [6] [7] [8]. This then implies some sort of classification decision feedback mechanism to modify or adapt the feature patterns. Our research follows this direction.

We have developed a data mining approach based on genetic algorithms (GA's). The basic operation of this approach utilizes a feedback linkage between pattern evaluation and classification. That is, we carry out pattern discovery by feature extraction (with dimensionality reduction) and appropriate setting of thresholds simultaneously with classifier design, through "genetic learning and evolution," as shown in Figure 1.

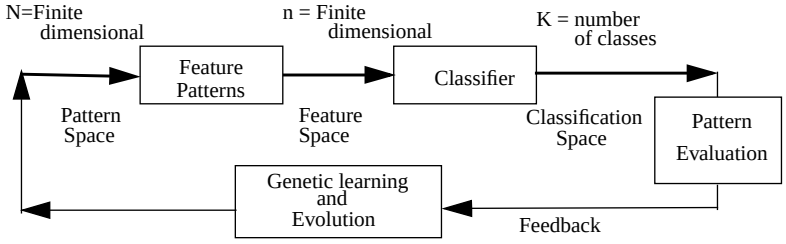

Figure 1. Pattern Discovery and Classifier with GA Feedback Learning System

The objective of this approach is to find interesting feature patterns of certain classes with a feature subset reduced from the original *N* features such that useful class discriminatory information is included and redundant class information and/or noise is excluded. We take the following general approach. The data's original fea-

PATTERN DISCOVERY FROM DATA USING GENETIC ALGORITHMS

M. PEI, E. D. GOODMAN
Case Center for Computer-Aided Engineering and Manufacturing

W. F. PUNCH, III
Intelligent Systems Laboratory, Department of Computer Science
Genetic Algorithms Research and Applications Group (GARAGe)
Michigan State University, 230 Engineering Building, East Lansing, MI 48824
e-mail: pei@egr.msu.edu

In this paper we summarize our research on pattern discovery and classification from high-dimensionality data sets using genetic algorithms. We have developed a GA-based approach utilizing a feedback linkage between pattern evaluation and classification. That is, we carry out pattern discovery by feature extraction (with dimensionality reduction) and appropriate setting of thresholds with classifier design simultaneously, through "genetic learning and evolution." This approach combines a GA with a production decision rule system. We first apply this approach on a binary data set, demonstrating the effectiveness of this approach on real biological data. In further ongoing research, it will be extended to handle a variety of feature types that include binary, nominal, numerical and structural features.

Keywords: genetic algorithms, classification, pattern discovery, feature extraction, production rules.

1. Introduction

The growing glut of data in the worlds of science, business and government create an urgent need for a new generation of automated and intelligent tools and techniques which can analyze, summarize, and extract "knowledge" from raw data [1]. Most of knowledge discovery or data mining tools and techniques are based on statistics, machine learning, pattern recognition and artificial neural networks. The great challenge for data mining comes from huge databases of noisy, high-dimensionality data. Genetic algorithms (GAs) are good candidates for attacking this challenge since GAs are very useful for extracting patterns in multiclass, high-dimensionality problems where heuristic knowledge is sparse or incomplete [2] [3].

The data mining approach we have developed is called the GA/RULE approach--a genetic algorithm combined with a production rule system. This approach is based on the general structure of a Pittsburgh-style classifier system [4] [5], and focuses on discovery of feature patterns by the classification rules. The inductive learning procedure of the GA evolves a rule classifier using a known "training" sample set. The result of training is a small set of "best" classification rules which classify the training set more accurately than other rule sets. By examining these "good" rules, one can discover the feature patterns which are most useful for classification. These patterns of interest, represented as rules, provide information useful to domain experts for evaluating and interpreting the extracted patterns to decide what constitutes "knowledge", then consolidating the knowledge, resolving conflicts with previously extracted knowledge, and making the knowledge available for use in the KDD pro-