Fast and Accurate Feature Selection Using Hybrid Genetic Strategies

César Guerra-Salcedo
Department of
Computer Science
Colorado State University
Fort Collins, CO 80523
guerra@cs.colostate.edu

Stephen Chen
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
chens@ri.cmu.edu

Darrell Whitley
Department of
Computer Science
Colorado State University
Fort Collins, CO 80523
whitley@cs.colostate.edu

Stephen Smith
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
sfs@cs.cmu.edu

Abstract- When dealing with object classification, each object is defined by a set of features (characteristics) that classify the object to a particular class. The problem is how to choose the best subset of characteristics that provide an accurate classification. Previous research has shown that Decision tables are as accurate as C4.5 for classification purposes. Two different genetic search techniques, CHC and CF/RSC, are applied to this problem. Results shows that CF/RSC and Decision tables are a very good combination when dealing with large feature spaces. Results also suggest that CHC is better when used for problems with noise added to the features.

1 Introduction

Feature subset selection is defined as a process of selecting a subset of features, *d*, out of the larger set of *D* features which maximize the classification performance of a given procedure over all possible subsets [10].

In a typical problem of case-based classification, instances (objects) need to be classified according to similar characteristics. The characteristics describing an instance could be from very different domains. For example, in a cloud classification problem [1] [2] [5] [6] there are 204 different characteristics to describe a cloud. These characteristics come from spectral, textural and physical measures from each sample area [5] [6].

Searching for an accurate subset of features is a difficult search problem. Search spaces to be explored could be very large, as in the cloud classification problem in which there are 2^{204} possible features combinations. Search strategies such as ***Hill-climbing*** and ***Best-first search*** [16], among others, have been used to find subsets of features with high predictive accuracy.

Case-based classification requires the use of a classifier for the instances. For this research we use decision tables as classifiers for all the classification problems. Empirically decision tables have shown to be as accurate as C4.5 [16] for case-based classification tasks. In Section two we offered an empirical comparison of decision tables with C4.5 for an image classification problem. The learninng curve of decision tables suggest that they are more accurate than C4.5.

In this paper we address the problem of feature subset selection using two genetic-based search algorithms. One is a genetic algorithm called CHC, and the other is a bit climber called Common Features/Random Sample Climbing (CF/RSC). GAs are search strategies based on the principles of natural selection. In a GA a $population$ of possible solutions called $chromosomes$ is maintained. Chromosomes are selected (each according to its $fitness$), recombined, and mutated to evolve a new population. The process is repeated until a 'stopping condition' has been achieved for the most-fit individual in the population or when a certain number of generations have been produced.

On the other hand, a simple bit climber is a local search procedure. It starts with a randomly generated solution (sometimes with zero features selected) or population of solutions. The bit climber flips every bit (one at a time) in the solution according to a certain permutation. If a better solution is found, it is kept and the procedure continues until no improvements can be found.

The results obtained as part of this research show significant improvement in accuracy for case-based classification problems when compared with other techniques. Experiments have been conducted for feature-subset accuracy analysis for the feature subsets obtained by our method. The results show that is almost unlikely to obtain the same feature subsets randomly.

As a final remark on our research, the bit climber has been successfully applied for this type of problems with better results than traditional approaches. CF/RSC was able to obtain very good feature subsets using less than 50% of time employed by CHC.

The paper is organized as follows. First a background review of the material involved in the research is presented. Second we present a description of the datasets for the experiments. The experimental set up and results are detailed in section 4. In section 5 a brief discussion is presented.

2 Background Review

In a supervised learning task, a training set of labeled fixed-length features vectors is given; the task is to induce a classification model. This model is used to predict the class label of subsequent instances or previously unseen cases. The information known about the class is inherent to the features presented to the classifier and determines the accuracy of it.

Practical machine learning algorithms such as ID3 [20] [18] and C4.5 [20], are known to perform poorly (degrade in accuracy) when presented with many features that are not necessary for predicting the desired output [14].

Optimal feature extraction has been studied for years [2] as a central topic of research in the Machine Learning Community. The selection of relevant features, as well as the elimination of irrelevant ones, play a central role in machine learning applications. Reducing the set of features considered for a specific task could improve the accuracy of a prediction or the speed of processing input data for specific manipulation.

Reducing the number of features used for classification is desirable for a number of reasons [5]. There can be redundant or irrelevant information among the whole set of features; there can also be estimation errors in the system parameters used to measure the features. The problem of feature subset selection involves finding a "good" set of features under some objective function [14]. In other words, the goal is to find a set of relevant features, which when presented to the classifier, maximizes its performance (accuracy).

For this type of application, traditionally each chromosome in the population represents a possible subset of features that is presented to the inducer. The fitness of the chromosome is based on the accuracy of the evolved subset of features to predict class values for unseen cases. Different fitness functions for this task have been studied. In Bala et al. [3] [4], Vafaie et al. [26] [25] and Turney [24] a decision tree generator is used, in [12] a variant of a decision table is used and in [19] the authors used a modified version of K-nearest neighbor. In every case the fitness function is an inducer that classifies cases according to the features presented in a particular chromosome.

2.1 Classification and Classifiers

A classifier is a system that classifies instances based primarily in trained data from which the classifier infers a classification rule. A set of unseen cases are used to test the accuracy of the classification rule.

2.1.1 Decision Table Majority: DTM

Decision tables have been largely used as a tool for expert systems [27] and as a method of classification [16]. Kohavi [15] presents a different view of a decision table making them even simpler. He established a majority-class approach for unlisted cases. If an unseen case is not present in the table, the majority class of the table is returned as the class for the unseen case. Kohavi has called this approach $Decision\ Table\ Majority$ or DTM. A DTM has two components.

1. A set of features that are included in the table.
2. A sample consisting of labeled instances (each entry) from the space defined by the features selected.

Given an unlabeled instance, the DTM classifier searches for an exact match among the samples in the table. If no match is found the most common class of the elements stored in the table is returned as the class for the instance to be labeled. Otherwise, the most common class of all matching instances is returned.

An example of DTM for an artificial 2-class learning task is presented in Tables 1 and 2. Table 1 shows the task before selecting any features and in Table 2 the features have already been selected. Note the differences in feature size as well as the examples considered for each class.

	A_1	A_2	A_3	A_4	Class
E_1	1	1	0	0	+
E_2	1	1	0	1	+
E_3	1	1	1	1	+
E_4	1	0	0	0	-
E_5	0	1	0	1	-
E_6	1	1	1	0	+
E_7	1	1	0	1	+
E_8	1	1	0	1	+
E_9	1	0	1	0	-
E_10	0	1	0	1	-

Table 1: Training examples for an artificial 2-class learning task with four Boolean attributes.

	A_1	A_2	Class	Examples
$Entry_1$	1	1	+	E_1,E_2,E_3,E_6,E_7,E_8
$Entry_2$	1	0	-	E_4,E_9
$Entry_3$	0	1	-	E_5,$E10$

Table 2: Simple Decision Table for feature subset $\{A_1, A_2\}$ for the examples given in Table 1. For row $Entry_1$ the class is + applying 2. The global default class would be +, as there are 6 examples for this class versus only 4 for the other one.

When comparing DTM vs. C4.5, Kohavi shows two important results [15]. The average accuracy of DTM on the real-world datasets tested was equivalent to C4.5 and the average accuracy of DTM on artificial datasets tested was higher than C4.5. These results are based on experiments using datasets taken from the UCI repository [7]. Kohavi ([16] page 140) states that

> Decision tables can be improved in some ways. The weakest point of the hypothesis space is the use of the training set's majority label when a perfect match is not found. This can be replaced with something more sensible, such as finding a match on fewer features.

2.1.2 Euclidean Decision Tables

We propose the use of Euclidean Decision Tables (EDT) that are based on DTMs and nearest-neighbor classifiers. The way decision tables are used in the EDT algorithm differ slightly from those used in DTM algorithm [16]. The main

characteristic of EDTs is that they use a Euclidean distance measure as the measure between an unseen case and a case presented in the table. They also are constructed using a hashed-based table in which instances with same values are grouped and a majority-class approach is used. The algorithm for creating and using EDTs is presented next.

- For any feature subset construct a Euclidean Decision Table by simply projecting all given training examples in the feature subset selected as header for the table.
- For all "after projection" identical examples count class frequencies and assign the majority class to every entry.
- When classifying new examples, look up the projected example in the decision table using the Euclidean distance measure. Return as the classification result, the appropriate majority class of the entry with the minimum Euclidean distance between the entry and the unseen case.

2.1.3 Learning : Comparison Between C4.5 and EDTs

In order to empirically justify the use of EDTs instead of C4.5 we ran different sets of experiments using the LandSat dataset assessing the accuracy of EDTs as classifiers vs. the accuracy of C4.5. We show that EDTs perform well even when the features are selected randomly from the complete set of features (i.e., no search engine is used). The following algorithm describes our approach.

- The feature space (36) was divided in 15 sets. EDT classifiers were constructed using the amount of features described in each set. A total of 270 classifiers were constructed for each particular number of features; features were randomly selected. The first set of classifiers corresponds to those constructed using 36 features, the second set to classifiers constructed using 33 features, the third using 30 features. From the fourth to the fifteenth sets of classifiers the number of features used were: 28, 25, 23, 20, 18, 16, 14, 11, 9, 7, 5 and 3.
- Each group of 270 classifiers was divided into 27 groups of 10 independent classifiers each. Each classifier was trained with randomly selected data instances. The first group corresponds to classifiers trained with 49 samples, the second group corresponds to classifiers trained with 69 samples, the third represents classifiers trained with 93 samples. From the fourth to the twenty seventh group the number of samples used to train each group were 147, 229, 309, 397, 499, 597, 697, 765, 819, 987, 1130, 1285, 1447, 1653, 1755, 1863, 2265, 2545, 2725, 3315, 3525, 3885, 4255 and 4419.
- Each classifier was tested using the set of 2000 unseen cases. The standard deviation and average accuracy are reported. A total number of 4050 (15 x 27 x 10) experiments were performed.

The same experiments were carried out using C4.5 instead of EDT and using exactly the same files as with EDT. The results are summarized in Figure 1. EDT seems to be better than C4.5 except for three and five features in which the difference between them is in the order of 2% and 0.03% respectively.

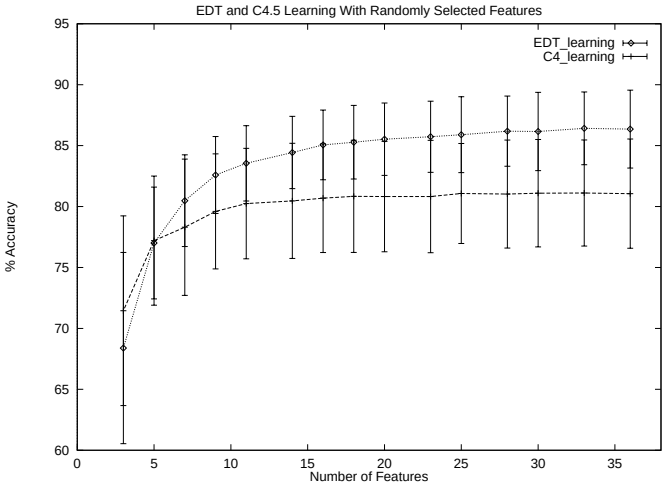

Figure 1: EDT and C4.5 Learning, no search engine procedure employed. 4050 experiments performed. Each point on the line is the average of 270 experiments (10 classifiers for each one of the 27 possible choices of training elements).

2.2 Genetic Algorithms Review

Genetic Algorithms (GAs) are stochastic search mechanisms based on natural selection concepts [13]. GAs have been applied to a wide variety of problems, including search problems [21], optimization problems [23], and in many problems in industry, economics, social science and drug design.

2.2.1 CHC

CHC [11] is a generational genetic search algorithm that uses truncation selection. The CHC algorithm randomly pairs parents, but only those string pairs which differ from each other by some number of bits (i.e., a mating threshold) are allowed to reproduce. The initial threshold is set to $l/4$, where l is the length of the string. When no offsprings are inserted into the new population during truncation selection, the threshold is reduced by 1. The crossover operator in CHC performs uniform crossover that randomly swaps exactly half of the bits that differ between the two parent strings.

No mutation is applied during the recombination phase of the CHC algorithm. When no offspring can be inserted into the population of a succeeding generation and the mating threshold has reached a value of 0, CHC infuses new diversity into the population via a form of restart known as cataclysmic mutation. Cataclysmic mutation uses the best indi-

vidual in the population as a template to re-initialize the population. The new population includes one copy of the template string; the remainder of the population is generated by mutating some percentage of bits (e.g 35%) in the template string.

2.3 Random Bit Climbing

A random bit climber is a hill climber based on a binary representation of the search space [9] [22]. Usually it begins with a randomly generated possible solution and starts by changing one bit at a time. If an improvement if found, it is accepted. In Random Bit Climbing (RBC) a random permutation is generated to determine the order in which bits flips are tested. After flipping every bit in the initial solution string, a new random sequence is chosen for testing the bits and the bit climber again checks every bit for an improvement. If the bit climber has tested every bit and no improvement is found, a local optimum has been reached. For the experiments reported in the paper we use a hybrid bit climber called Common Features/Random Sample Climbing (CF/RSC).

2.3.1 Common Features/Random Sample Climbing CF/RSC

CF/RSC is a two stage algorithm. It uses a population of individuals (usually small, e.g., 10 individuals.) Common Features (CF) is exploited when recombining two individuals in the population. Random sample climbing (RSC) is applied as a local improvement operator used to heuristically improve the individuals in the population. In generation 0, all individuals in the population are initialized to strings of all zeroes.

Random sample climbing (RSC) is applied to each individudal in the population. For each individual, n bits are randomly mutated. Usually n is a small number less than 10. This is done k times, thus generating k new solutions for each member of the population. In the experiments used here, $k = 30$. If the best solution found among the k new solutions is better than the original member of the population from which it was generated, then the new solutin replaces the original member of the population.

In subsequent generations, recombination is applied. By the commonality hypothesis [8], common features are believed to be the most important for classification. Thus, when two strings are recombined, commonly selected features in the two parents are passed on to offspring. All other features are set to zero. Then RSC is applied to the entire population.

3 Datasets

It is important to characterize a feature subset selection problem using a database of cases to construct the classifier. A set of unseen cases are useful to test the classifier. In this research we have focus our attention on two types of datasets, real datasets and artificially generated datasets. Real datasets are datasets in which cases belong to a real-world problem (for example cloud classification or satellite-imagery classification). On the other hand, artificially generated datasets are datasets in which cases are generated using a test case generator. For this research three real-world datasets were employed and one artificially generated classification problem. The real-world classification problems are: satellite classification dataset (LandSat), a DNA classification dataset and a Cloud classification dataset. On the other hand, the artificially generated classification problem rely on a LED identification problem. LED cases are artificially generated using a test case generator. All datasets were chosen for this research because of their size in terms of number of cases, their length in terms of number of features and the interaction between features. A brief description of each dataset follows.

3.1 LandSat Images dataset

The first data set is the LandSat dataset. The LandSat dataset consists of 4435 train cases and 2000 test cases. Each case represents a satellite image with 36 features. Each feature has been discretized, its values ranging from 0 to 255. The data can be categorized in six different classes. The classes and the distribution of available data is presented in Table 3. Previous results with this dataset, and the use of genetic algorithms as a search space engine, can be found in Bala et al.[4] [3].

Class Name	Learning Set
Red Soil	1072 (24.17%)
Cotton Crop	479 (10.80%)
Gray Soil	961 (21.67%)
Damp Gray Soil	415 (09.36%)
Soil w/ Veget. Stubb.	470 (10.60%)
Very Damp Gray Soil	1038 (23.40%)

Class Name	Unseen Cases
Red Soil	461 (23.05%)
Cotton Crop	224 (11.20%)
Gray Soil	397 (19.85%)
Damp Gray Soil	211 (10.55%)
Soil w/ Veget. Stubb.	237 (11.85%)
Very Damp Gray Soil	470 (23.50%)

Table 3: Classes and Data Distribution of the LandSat Dataset. The learning set represents the number and percentage of the total of cases for the learning data file. Unseen cases represents the number and percentage of the total of cases for the test data file.

3.2 Cloud Classification dataset

The second data set is a cloud dataset[1] used for developing an automated system for cloud classification [1] [2] [5] [6]. There are 1633 cases (no training and test distinction), each case represents a cloud with 204 features on a continuous-based range belonging to one of 10 different cloud types. The

[1] Provided by Richard Bankert from the Naval Research Laboratory.

classes and the distribution of available data is presented in Table 4.

Class Name	Data Available
Cirrus	212 (12.98%)
Cirrocumulus	72 (04.40%)
Cirrostratus	166 (10.16%)
Altostratus	154 (09.43%)
Nimbostratus	59 (03.61%)
Stratocumulus	251 (15.37%)
Stratus	225 (13.77%)
Cumulus	123 (07.53%)
Cumulonimbus	149 (09.12%)
Clear	222 (13.59%)

Table 4: Classes and Data Distribution of the Cloud Dataset. The data represents the number and percentage of the total number of cases in the data file

3.3 DNA dataset

The third data set is a DNA dataset. The dataset represents Primate splice-junction gene sequences (DNA). There are 2000 training cases, 1186 test cases, and 180 binary features for each case. Three different classes exist in this dataset. The task is to recognize exon/intron boundaries referred as EI sites (25%); intron/exon boundaries referred to as IE sites (25%); or neither (50%).

Splice junctions are points on a DNA sequence at which "superfluous" DNA is removed during protein creation. The IE borders are referred to as "acceptors" and the EI borders are "donors". The instances were taken the UCI repository. The features provide a window of 60 nucleotides, each represented as a three binary indicator features that represent the value A,C,G or T, thus giving 180 features. The classification is the middle point of the window, thus providing 30 nucleotides at each side of the junction.

3.4 LED generation problem

The idea is to identify a LED representation of a digit (a seven segment representation) in each instance of the dataset. All the instances are artificially generated. This is a Binary-based dataset in which a one represents a segment on and zero represents the segment is off. Every instance in the dataset may be constructed by significant features and spurious features. A significant feature is such that it represents a segment in a valid LED representation. A spurious feature does not play a role in the representation and its randomly generated. In order to make the classification problem harder, the use of spurious features and the possibility of noise in the significant features have been added.

There is another alternative form of representing digits using only five segments instead of seven. Both seven-segment and five-segment representations are depicted in Figure 2.

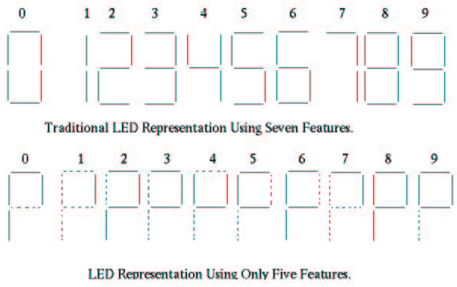

Figure 2: Seven and Five segment representations of a digit. Dashed lines represent unused features

There are several options in the generator program that we exploit for our experiments. The different problems generated for this research were the following:

1. A problem with 24 features in which the first 7 features are significant and the rest are spurious. There is no noise added.
2. A problem with 100 features in which the first 7 features are significant and the rest are spurious. There is no noise added.
3. A problem with 30 features in which the significant features are not contiguous. Features corresponding to a LED representation are separated for three spurious features. No noise was added to the problem.
4. A problem with 70 features in which three replications of the same valid LED number are found. significant features for each replication are separated from each other by four features. The four features are either valid or spurious. There is no noise added.
5. Same as problem 4 but with 10% of noise added to each valid feature.

4 Preliminary Results

In this section we present the preliminary results of our research. We have been using CHC and CF/RSC as genetic search engines and EDT as inducer. All the work done so far is based on the data sets described in the previous section.

4.1 Experimental Setup. Real-world Datasets

For all experiments the following setup applied: For CHC a population size of 50 and a total number of function evaluations of 15000 were used. For CF/RSC the initial population has size 10, the pool of random solutions has 30 elements, five climbings for element in the pool are allowed and 10 generations were used (15000 trials total).

We posed each experiment as a minimization problem, trying to find the subset of features that minimized the number of misclassification errors of the classifier constructed.

From the LandSat training file (4435 cases) 700 training cases and 500 test cases were chosen randomly to generate 30 independent datasets. The cases in the test file are different than the cases in the training file. The features were selected from the 700 available training elements and a classifier was constructed. The accuracy of the classifier when tested on the 500 test cases was used as the evaluation for the chromosome. At the end, a classifier was constructed using the features depicted by the final chromosome and trained with the original train set and tested with the original test set.

From the Cloud file (1633 cases) 30 sets of 400 training cases and 500 test cases were generated randomly. Instances in the test file are different than the cases in the training file. Each feature selection vector was evaluated by constructing an EDT classifier using 400 training elements and tested on 500 test cases in order to evaluate each chromosome. At the end, a classifier was constructed with the features depicted by each chromosome and tested using 10-fold Cross Validation.

From the DNA training file (2000 cases) 700 training cases and 500 test cases were chosen randomly to generate 30 independent datasets. Each feature selection vector was evaluated by constructing an EDT classifier. At the end, a classifier was constructed with the features depicted by each chromosome using the original training file and tested using the original testing file. The accuracy results are summarized in table 5.

Problem Type	Algorithm Used	Best Acc.	Worst Acc.	Avg	Avg. Number Features
LandSat	CHC	88.9%	86.4%	87.56%	12.62
	CF/RSC	89.8%	84.4%	87.6%	12.8
DNA	CHC	92.5%	85.6%	89.38%	11.24
	CF/RSC	93.8%	88.0%	91.2%	8.5
Cloud	CHC	80.8%	78.0%	79.3%	42.13
	CF/RSC	84.4%	78.4%	80.9%	14.0

Table 5: Accuracy results for CHC and CF/RSC on different real datasets. The results represent the accuracy of a classifier constructed with the features depicted in the final chromosome, trained with the original train set and tested with the original test set (except in Cloud dataset in which 10 fold CV were used).

CF/RSC found better subsets for each real-problem. The idea of starting with a solution with all zeroes and add features is a good approach. CHC uses a crossover operator called HUX (Half Uniform Crossover) that randomly swaps exactly half of the bits that differ between the two strings. HUX seems to be not very suitable for this particular problem specially when features are sparsed.

4.2 Experimental Setup. Artificial Datasets

For each of the five different problems two files (train and test) with 1250 elements each were generated (to be used as original training and testing files). From the training file 400 training instances and 400 test instances were chosen randomly to generate 30 independent datasets. The features were selected from the 400 available training cases and a classifier was constructed. The accuracy of the classifier when tested on the 400 test cases was used as the evaluation for the chromosome. At the end, a classifier was constructed using the features depicted by the final chromosome and trained with the original train set and tested with the original test set. The setups for both CHC and CF/RSC are the same as the ones described in previous section. The results are summarized in Table 6.

Problem Type	Algorithm Used	Best Acc.	Worst Acc.	Avg	Avg. Number Features
Problem 1	CHC	100%	100%	100%	5.0
	CF/RSC	100%	100%	100%	5.0
Problem 2	CHC	100%	100%	100%	6.8
	CF/RSC	100%	100%	100%	5.0
Problem 3	CHC	100%	100%	100%	5.0
	CF/RSC	100%	100%	100%	5.0
Problem 4	CHC	100%	100%	100%	5.0
	CF/RSC	100%	100%	100%	5.0
Problem 5	CHC	88.2%	84.6%	86.4%	24.2
	CF/RSC	87.5%	79.8%	84.1%	17.9

Table 6: Accuracy results for CHC and CF/RSC on different real datasets. The results represent the accuracy of a classifier constructed with the features depicted in the final chromosome, trained with the original train set and tested with the original test set. The problem number corresponds to the problem numbering in the explanation of the experiments.

In this type of problems the accuracy of solutions obtained by CHC and by CF/RSC were almost the same except in problems two and five. In problem two there are 100 features involved and the optimal solution has only five bits. The main difference in the accuracy seems to be HUX again. CHC is trapped with a seven-bits solution and its hard for it to jump to a better solution (five-bits). In problem five, the correct features are highly related with each other, CHC seems to take advantage of this fact. CF/RSC random flipping does not produce very good solutions to combine with.

4.3 Evaluating the Accuracy of the Solutions

In order to test the accuracy of the subsets obtained by CHC and CF/RSC a statistical test was performed. Feature-subset accuracy analysis was performed for each subset obtained by both algorithms. The basic idea of feature-subset accuracy analysis was proposed by McFarland et al [17] in 1986. They developed a technique for cluster analysis called Cluster Significance Analysis or CSA. CSA is based on definition of the tightness of a cluster. The easiest way of computing tightness is by using the mean squared distance of a cluster. Suppose we have a cluster C_i with three elements and each element has two dimensions. The Mean Squared Distance (MSD) of C_i is

defined as :

$$MSD(C_i) = \sum_{\substack{i<>j \\ 1 \leq i,j \leq 3}} (x_i - x_j)^2 + (y_i - y_j)^2 \qquad (1)$$

MSD is calculated by taking the squared distance between each pair of points in the cluster. The significance of a cluster C_i composed of n elements, given the fact that there are m elements available ($m > n$) is calculated as follows. Compute $MSD(C_i)$, there are $\binom{m}{n}$ different clusters of n elements. Generate randomly M clusters of n elements each out of $\binom{m}{n}$ features and compute their MSDs. The number of groups (including C_i) that have MSDs equal to or less than $MSD(C_i)$ is designated as A. The probability p that a cluster at least as tight as C_i would have arisen by chance alone is given by $p = A/M$. This significance probability or p-value indicates the significance of the relationships between the elements in C_i.

Instead of comparing MSDs, we compared accuracies of classifiers. There are m features in a classification problem. Once that a subset of n features has been found by the genetic search, there are $\binom{m}{n}$ possible classifiers with n features. Using all the available training instances and the n features selected in the chromosome with the best performance a classifier Cl_0 was constructed. The accuracy $Acc(Cl_0)$ was computed using the original test file. After that, 100000 subsets of n features (in all cases $\binom{m}{n} > 100000$) were randomly generated. For each subset i a classifier Cl_i was constructed using all the available training instances and $Acc(Cl_i)$ was computed.

The number of classifiers (including Cl_0) that have accuracy equal to or better than $Acc(Cl_0)$ is designated as A. The probability p that a classifier at least as accurate as Cl_0 would have been created by chance alone is given by $p = A/100000$. This significance probability or p-value indicates the significance of the relationships between the features in Cl_0. Also, the p-value gives us an idea of how difficult it is to generate a particular subset of features randomly. For each different experiment, the p-values calculated using the features obtained by either CHC or CF/RSC were 0.00001.

5 Discussion

This paper presents research results applying genetic search methods to the problem of feature subset selection. Although some genetic-based systems for feature subset selection have been previously studied, the work presented produces much better results than previous works using the same approach. The paper also presents a novel bit climber technique called CF/RSC. The results for real-world classification problems using CF/RSC were better than the results obtained by CHC (so far known as one of the most efficient and effective genetic algorithms for feature subset selection problems). The system proposed in this research uses an accurate and easy-to-implement classifier called Euclidean Decision Tables.

Based on the results presented in this paper, it seems that CHC is a very powerful search algorithm for problems in which noise has been added.

We are also proposing a technique for evaluate the accuracy of a subset of features based on the probability of randomly generate such subset.

Acknowledgments

César Guerra-Salcedo is a visiting researcher at Colorado State University supported by CONACyT under registro No. 68813 and by ITESM.

Stephen Chen and Stephen Smith were sponsored in part by the Advanced Research Projects Agency and Rome Laboratory, Air Force Material Command, USAF, under grant numbers F30602-95-1-0018 and F30602-97-C-0227, and the CMU Robotics Institute. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Advanced Research Projects Agency and Rome Laboratory or the U.S. Government.

Bibliography

[1] David W. Aha and Richard L. Bankert. Feature Selection for Case-Based Classification of Cloud Types: An Empirical Comparison. In *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning*, 1994.

[2] David W. Aha and Richard L. Bankert. A Comparative evaluation of Sequential Feature Selection Algorithms. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 1–7, 1995.

[3] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler. Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification. In *14th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1995.

[4] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler. Visual Routine for Eye Detection Using Hybrid Genetic Architectures. In *14th Int. Joint Conf. on Artificial Intelligence (IJCAI)Proceedings of ICPR 96*, 1996.

[5] Richard L. Bankert. Cloud classification of avhrr imagery in maritime regions using a probabilistic neural network. *Applied Metheorology*, 33(8):909–918, 1994.

[6] Richard L. Bankert and David W. Aha. Improvement to a neural network cloud classifier. *Applied Metheorology*, 35(11):2036–2039, 1996.

[7] E. Keogh C. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[8] Stephen Chen and Stephen Smith. Experiments on Commonality in Sequencing Operators. In *Proceedings of the third annual Genetic Programming Conference*. Morgan Kaufmann, 1998.

[9] Lawrence Davis. Bit-Climbing, Representational Bias, and Test Suite Design. In L. Booker and R. Belew, editors, *Proc. of the 4th Int'l. Conf. on GAs*, pages 18–23. Morgan Kaufmann, 1991.

[10] P. Devijver and J. Kittler. *Pattern recognition: A Statistical Approach*. Prentice Hall, 1982.

[11] Larry Eshelman. The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In G. Rawlins, editor, *FOGA -1*, pages 265–283. Morgan Kaufmann, 1991.

[12] Cesar Guerra-Salcedo and Darrell Whitley. Genetic Search For Feature Subset Selection: A Comparison Between CHC and GENESIS. In *Proceedings of the third annual Genetic Programming Conference*. Morgan Kaufmann, 1998.

[13] John Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[14] G John, R Kohavi, and K Pfleger. Irrelevant Features and the Subset Selection Problem. In William W. Cohen and haym Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129. Morgan Kauffmann, 1994.

[15] Ron Kohavi. The Power of Decision Tables. In N. Lavrac and S. Wrobel, editors, *Proceedings of the European Conference on Machine Learning*, pages 174–189. Springer Verlag, 1995.

[16] Ron Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Stanford University, 1995.

[17] James McFarland and Daniel Gans. On the Significance of Clusters in the Graphical Display of Structure-Activity Data. *Journal of Medicinal Chemistry*, 29:505–514, 1986.

[18] Tom M. Mitchell. *Machine Learning*. Mc. Graw Hill, 1997.

[19] W.F. Punch, E.D. Goodman, Min Pei, Lai Chia-Shun, P. Hovland, and R. Enbody. Further Research on Feature Selection and Classification Using Genetic Algorithms. In Stephanie Forrest, editor, *Proc. of the 5th Int'l. Conf. on GAs*, pages 557–564. Morgan Kaufmann, 1993.

[20] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[21] S. Rana, D. Whitley, and R. Cogswell. Searching in the Presence of Noise. In H.M. Voigt, W. Ebeling, Ingo Rechenberg, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature, 4*, pages 198–207. Springer/Verlag, 1996.

[22] Soraya B. Rana and L. Darrell Whitley. Bit Representations With A Twist. In *Proceedings of the 7th International Conference on Genetic Algorithms*. Morgan Kaufmann, 1997.

[23] J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, and Rajarshi Das. A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In J. D. Schaffer, editor, *Proc. of the 3rd Int'l. Conf. on GAs*, pages 51–60. Morgan Kaufmann, 1989.

[24] Peter Turney. How to Shift Bias: Lessons from the Baldwin Effect. *Evolutionary Computation*, 4(3):271–295, 1997.

[25] Haleh Vafaie and Ibrahim Imam. Feature Selection Methods: Genetic Algorithms vs. Greedy-like Search. In *Proceedings of the International Conference on Fuzzy and Intelligent Control Systems*, 1994.

[26] Haleh Vafaie and Kenneth A. De Jong. Improving a Rule Learning System Using Genetic Algorithms. In *Machine Learning: A Multistrategy Approach*, pages 453–470. Morgan Kaufmann, 1994.

[27] J. Vanthienen and G. Wets. From Decision Tables to Expert System Shells. *Data and Knowledge Engineering*, 13(3):205–220, 1994.