Genetic Search for Feature Subset Selection: A Comparison Between CHC and GENESIS

César Guerra-Salcedo
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
(970) 491-8130
guerra@cs.colostate.edu

Darrell Whitley
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
(970) 491-5373
whitley@cs.colostate.edu

ABSTRACT

Classification problems map an object to a particular class using a set of available features. The problem is how to choose the best subset of features that provide an accurate classification. Genetic algorithms may provide a novel yet powerful search method to automate feature subset selection on a classification problem. Two different types of genetic algorithms are compared on this domain. One of them, GENESIS, is based on the traditional simple genetic algorithm. The second, CHC, uses truncation selection, an adaptive uniform crossover operator and restarts. CHC and GENESIS have been used several times for different types of problems and demonstrate very good performance. Our results indicate that CHC yields better results on two applications. We also introduce a novel classifier, Euclidean Decision Tables.

1 Introduction

The amount of data collected by industry and research laboratories is rapidly growing as storage technologies and data collection methods improve. Accurate classification of data is a key issue for its correct utilization. Classification systems assign a set of input objects to a set of decision classes. The objects are described by a set of features (such as color, temperature, melting point, etc.). There exists an implicit function $F(features) = class$ that maps each object to a class based on the features present in the object.

In case-based classification, instances (objects) need to be classified according to similar features. The features describing an instance could be from very different domains. For example, in a cloud classification problem [Aha and Bankert, 1995] [Bankert, 1994] there are 204 different features to describe a cloud. These features come from spectral, textural and physical measures from each sample area [Bankert, 1994].

When classifying objects with a subset of features, the main goal is to obtain an accurate subset of features that allows classification within an acceptable margin of error. Feature subset selection is defined as a process of selecting a subset of features, *d*, out of the larger set of *D* features. The goal is to maximize the classification performance of a given procedure over all possible subsets.

Search spaces for subset feature selection can be large. In the cloud classification problem there are 2^{204} possible features combinations. Search strategies, such as ***Hill-climbing*** and ***Best-first search*** [Kohavi, 1995b], have been used to find subsets of features with high predictive accuracy.

In this paper we address the problem of feature subset selection using Genetic Algorithms (GAs) [Goldberg, 1989]. GAs have been successfully used in a variety of optimization problems, and are best known for being robust search techniques that can search extremely large spaces and obtain globally competitive solutions. They identify and exploit non-linear interactions of the features. Several researchers have used GAs for feature subset selection problems [Punch et al., 1993] [Bala et al., 1995] [Vafaie and Jong, 1994] [Turney, 1997] [Whitley et al., 1997]. Most of them use a classifier (inducer) as a fitness evaluation function. [1] Traditionally 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. For example Punch et al. [Punch et al., 1993] use a modified version of K-nearest neighbor as the classifier. Some other authors [Bala et al., 1995] [Vafaie and Jong, 1994] [Turney, 1997] use a decision tree generator [2] as the classifier in the evaluation function.

The outline of the paper is as follows. First, we present a

[1]Except in [Whitley et al., 1997] in which features are selected for a pattern matching problem without the use of a classifier.

[2]Or a modified version of it as in [Turney, 1997].

background section in which we introduce the main concepts related to this research. Second, we present a description of the data sets employed for this research. Third, we explain the set of experiments conducted as well as the results produced by them. Finally, conclusions and future research are presented.

2 Background Review

Practical machine learning algorithms such as ID3 [Quinlan, 1993] [Mitchell, 1997] and C4.5 [Quinlan, 1993], are known to degrade in accuracy when presented with many features that are not necessary for predicting the desired output [John et al., 1994].

Feature subset selection has been studied for years [Aha and Bankert, 1995] 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 prediction or the speed of processing input data.

A classifier is a system that classifies instances based primarily on 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. C4.5 is commonly used as a classifier, but it has the disadvantage of being computationally expensive. We therefore explore the use of Decision Tables as classifiers.

2.1 Decision Tables

Decision tables were originally conceived as a tool for representing managerial knowledge and needs. It is a detailed, logically-organized tabular structure with two major sections; conditions and actions. The condition may be thought of as 'If' statements, while the actions may be thought of as 'Then' statements. For each combination of conditions, we will take the actions specified.

2.1.1 A Different View of a Decision Table

Decision tables have been largely used as a tool for expert systems and as a method of classification [Kohavi, 1995b]. Kohavi [Kohavi, 1995a] 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 schema which is a set of features that are included in the table.
2. A body consisting of labeled instances from the space defined by the features in the schema.

Given an unlabeled instance, the DTM classifier searches for an exact match in the table using only features in the schema. If no match is found the majority class of the DTM is returned as the class for the instance to be labeled. Otherwise, the majority class of all matching instances is returned. An example of DTM is presented in Tables 1 and 2.

	A_1	A_2	A_3	A_4	Class
Ex_1	1	1	0	0	+
Ex_2	1	1	0	1	-
Ex_3	1	1	1	1	+
Ex_4	1	0	0	0	-
Ex_5	0	1	0	1	-
Ex_6	1	1	1	0	+
Ex_7	1	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	+	Ex_1,Ex_2,Ex_3,Ex_6,Ex_7
$Entry_2$	1	0	-	Ex_4
$Entry_3$	0	1	-	Ex_5

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 $+$. There are five cases with the same values for A_1 and A_2; four of them belong to class $+$ and only one (Ex_2) to class $-$. The global default class would be +, as there are 4 examples for this class versus 3 for the other one.***

When comparing DTMs vs. C4.5, Kohavi shows two important results [Kohavi, 1995a]. 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 UC Irvine repository.[3] Dr. Kohavi ([Kohavi, 1995b] page 140) states, "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".

We have modified the DTM algorithm to improve on its weakest point. We incorporate a Euclidean distance measure in the decision table lookup when an unseen case is tested. To distinguish our approach from its predecessors, we have called it ***Euclidean Decision Tables*** (EDT).

2.1.2 Euclidean Decision Tables

As stated, Euclidean decision tables 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 [Kohavi, 1995b]. The principal differences are in its creation and utilization. EDTs are something easy to implement, yet powerful enough for machine learning

[3]http://www.ics.uci.edu/ mlearn/MLRepository.html

applications. The algorithm for creating and using EDTs is presented next.

- For any feature subset construct a Euclidean Decision Table by simply projecting[4] all given training examples in the feature subset selected as header for the table.
- For all examples selected with identical feature values, 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. To determine the classification, find the majority class entry with the minimum Euclidean distance between the entry and the unseen case.

2.2 GENESIS

GENESIS [Grefenstette, 1984] is a genetic algorithm that has been used as a search engine by many researchers [Turney, 1997] [Bala et al., 1997] [Bala et al., 1995] because of its simplicity and its availability. GENESIS is a public domain software based on a simple GA. GENESIS uses crossover and mutation according to user-provided probabilities for searching the string space. GENESIS also allows different types of crossover and tools for Graying and deGraying the search space.

2.3 CHC

CHC [Eshelman, 1991] 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 and 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 individual 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.

3 Datasets

It is important to define a feature subset selection problem using a database of cases to induce the classifier. A set of unseen cases is useful to test the classifier. In this research we have been using two different datasets for our experiments. The datasets were chosen for this research because of their size both in terms of the number of cases and their length in number of 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 are presented in Table 3. Previous work related to feature subset selection employing this dataset was done by Bala et al. [Bala et al., 1995]. Bala and co-workers used GENESIS as the search engine and C4.5 as the classifier and evaluation function.

Class Name	Learning Set	Unseen Cases
Red Soil	1072 (24.17%)	461 (23.05%)
Cotton Crop	479 (10.80%)	224 (11.20%)
Gray Soil	961 (21.67%)	397 (19.85%)
Damp Gray Soil	415 (09.36%)	211 (10.55%)
Soil, Vegetation Stubble	470 (10.60%)	237 (11.85%)
Very Damp Gray Soil	1038 (23.40%)	470 (23.50%)

Table 3 ***Classes and Data Distribution of the LandSat Dataset.***

3.2 Cloud Classification dataset

The second dataset, provided by Richard Bankert from the Naval Research Laboratory, presents cloud features and classifications [Aha and Bankert, 1995] [Bankert, 1994]. There are 1633 cases with no predefined training and test distinction. Thus, for each training run 500 samples were selected for training (building the classifer) and 400 for testing and as the basis for the evaluation function. The samples were regenerated randomly on each run, so that there were no samples kept in reserve for final validation. We thus used a different form of validation for this problem, similar to cross validation.

Each sample represented a cloud with 204 features on a continuous-based range belonging to one of 10 different cloud types. The 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.***

[4]A hash table often is used to reduce computation time.

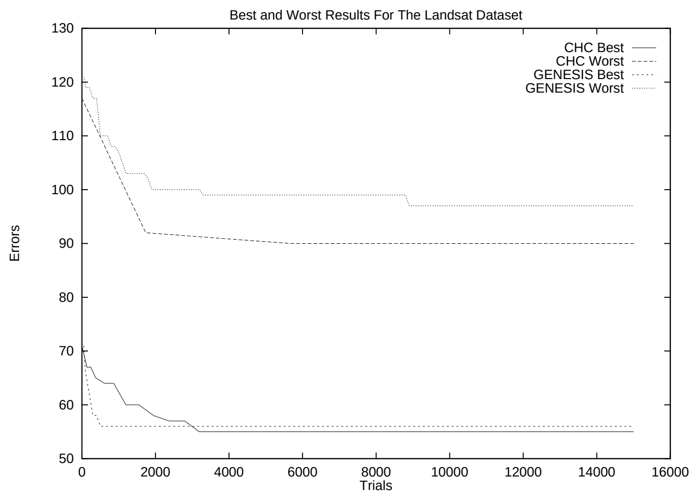

Figure 1 ***Best and worst runs for each algorithm using the LandSat dataset. The Y axis shows misclassification errors while the X axis represents the number of function evaluations.***

4 Preliminary Results

The central goal of this research is to apply and compare evolutionary methods like genetic algorithms to feature subset selection problems in order to improve on existing classifiers.

As stated in previous sections, genetic algorithms have been used for the problem of feature subset selection with encouraging results. In this section we present the preliminary results of our research using a "wrapper" approaches to feature subset selection. The wrapper approach uses the genetic algorithm as a search engine "wrapped" with a classifier. The classifier is "the objective function" and evaluates a subset of features evolved by the GA. The fitness of the chromosome is interpreted as "how good are the features presented in the chromosome to classify unseen cases". This is the most commonly used approach when applying GAs to feature subset selection problems[Bala et al., 1995]. For the experiments reported in this chapter, the classifier used was EDT. CHC and GENESIS were used as search engines and EDT as the evaluation function.

For both genetic algorithms, standard default parameter settings from the literature were used. The total number of function evaluations (trials) were fixed at 15000 for both genetic algorithms. For each algorithm the number of elements used as population was fixed to 50. Each chromosome in the GA represented a possible subset of features. The goal was to find a subset of features that minimized the misclassification errors associated with an EDT classifier. In order to make the comparison fair, each algorithm was ran for 10 runs. For each run, a random datasets of samples was used; the datasets used were the same for both algorithms.

4.1 The LandSat Dataset Results

Previous work done with the LandSat dataset and GAs is reported by Bala et al.[Bala et al., 1995]. They obtained an average accuracy rate of 83.1% on 2000 unseen cases with an average of 17 features selected out of 36 using C4.5 as the classifier. In some ways, this is a toy problem give the search space is only 2^{36}.

From the LandSat dataset 30 different datasets were generated randomly. Each dataset consists of 400 training cases and 700 test cases to be used as input for the classifier. For each feature selection vector an EDT classifer was constructed using the 400 available training samples. The accuracy of the classifier when tested on the 700 test cases was used as the evaluation for the chromosome representing the feature selection vector.

At the end of the experiments the best run (the one with the lowest number of misclassification errors) was selected and a classifier was constructed using the set of available train elements and tested using the 2000 unseen cases. The results are shown in Table 5. The plot of the best and worst runs from each algorithm is presented in Figure 1. For this dataset the algorithm with better performance was CHC. CHC was able to find better sets of features for all the runs in the experiment on testing data.

When the subsets of features yielding the best performance in conjunction with EDT were tested on unseen cases, there were 240 errors (an 88% accuracy rate) for the CHC solution and 242 errors for the (an 87.9% accuracy rate) for the GENESIS solution. The error rates are nearly identical. These compare very well with the accuracy rate of 83.1% reported by Bala et al. [Bala et al., 1995]. These results suggest that EDTs are as good as C4.5 in terms of providing a viable evaluation function for finding useful subsets of features.

On the best run, CHC found a solution using 16 features, while the best GENESIS solution used 21 features. On the other hand, the worst GENESIS solution used 18 features. A small set of features is desirable when feature subset selection problems are investigated [Kohavi, 1995b].

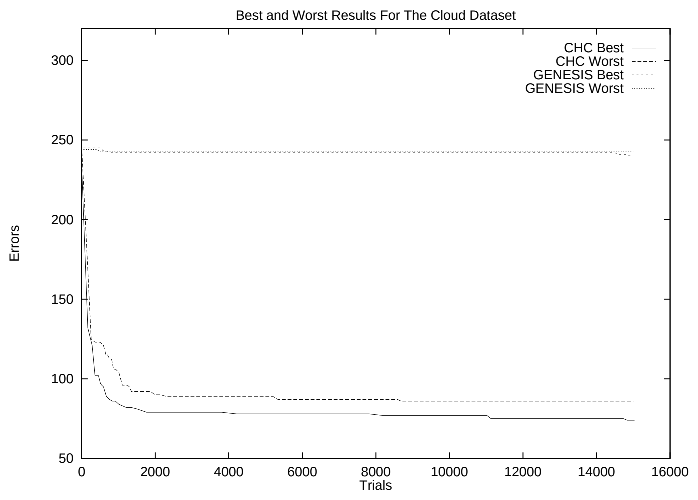

Figure 2 ***Plot of the average number of errors from the best and worst runs for each algorithm using the Cloud dataset. The Y axis shows misclassification errors while X axis represents the number of function evaluations.***

Algorithm	Best Run	Worst Run	Average	Test Errors
CHC	55 errors 16 feat.	90 errors 18 feat.	66.3 errors 17.4 feat.	240
GENESIS	56 errors 21 feat.	97 errors 18 feat.	72.3 errors 20.7 feat.	242

Table 5 ***Results for the comparison between CHC and GENESIS for the LandSat dataset. Best Run column represents the best results reported for the GA on the data used for that run. Average column represents the average after 30 independent runs. The last column represents the number of errors after running the classifier with the features reported in the best run and the unseen cases from the original dataset.***

4.2 The Cloud Dataset Results

In this dataset there are 2^{204} different combinations for each chromosome. Previous work has been done with this data set [Aha and Bankert, 1995] [Bankert, 1994], but we do not know of any application involving genetic algorithms and decision tables for the same dataset. Each feature selection vector was evaluated by constructing an EDT classifier using 400 training elements. The accuracy of the classifier when tested on 500 test cases was used in order to evaluate the chromosome. For each run of the genetic algorithm, the 400 training samples and 500 evaluation samples were randomly selected from the total 1,633 samples available.

Evaluation of the best solutions at the end of the run are different for this data set compared to the LandSat data. In this case, there is no reserve of unseen cases. At the end of the experiments the best run (the one with the lowest number of misclassification errors) was selected and tested using 30 independently created datasets. Each dataset consists of 60% elements of the total for training and 40% elements out of the total for test. The results are shown in Table 6. The plot of the average for the best and worst runs from each algorithm is presented in Figure 2.

For this dataset the algorithm with better performance was clearly CHC. CHC was able to find much better sets of features for all the runs in the experiment. The difference between the number of features found by CHC in its best run and by GENESIS in its best run is 13, but the difference in misclassification errors is huge. The accuracy rate for CHC is 77.6% while the accuracy rate for GENESIS is only 33.7%. The difference in classification errors after evaluating the best run on 30 independent datasets is almost 3 times better for the set of features found by CHC.

Algorithm	Best Run	Worst Run	Average	Test Errors
CHC	74 err. 99 feat.	86 err. 93 feat.	67.2 err. 96.4 feat.	146.2
GENESIS	240 err. 103 feat.	243 err. 99 feat.	242.2 err. 105.2 feat.	433.4

Table 6 ***Results for the comparison between CHC and GENESIS for the Cloud dataset. Best Run column represents the best results reported for the GA on the data used for that run. Err stands for errors and feat stands for features. Average column represents the average after 30 independent runs. The last column represents the average number of errors when the best set of features found is tested using 30 independent datasets.***

5 Conclusion and Future Work

Although there are some genetic-based systems for feature subset selection available, there are substantial differences between them. Two major results outcome from this research.

First, a comparative study between systems for feature subset selection based on GAs with fixed-length chromosome representation suggests that at least for this two real world data sets, CHC is better than GENESIS. Second, a new approach for feature subset selection based on CHC and Euclidean decision tables is proposed. The results for this system are very promising. We are in the process of extending this research by analyzing and comparing the use of a different classifier (C4.5) using additional datasets.

The way we posed the evaluation function for this research was very simple. There are other ways to pose the evaluation function, especially for large problems like the Cloud dataset. An interesting extension to this work is to use penalty functions to discourage the used of a large number of features.

Acknowledgments

César Guerra-Salcedo is a visiting researcher at Colorado State University supported by CONACyT under registro No. 68813 and by ITESM. This work was also supported by NSF grant 9503366.

References

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

[Bala et al., 1997] Bala, J., Jong, K. A. D., Huang, J., Vafaie, H., and Wechsler, H. (1997). Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, 4(3).

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

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

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

[Goldberg, 1989] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

[Grefenstette, 1984] Grefenstette, J. J. (1984). GENESIS: A System for Using Genetic Search Procedures. In *Proceedings of a Conference on Intelligent Systems and Machines*, pages 161–165.

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

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

[Kohavi, 1995b] Kohavi, R. (1995b). *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Stanford University.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. Mc. Graw Hill.

[Punch et al., 1993] Punch, W., Goodman, E., Pei, M., Chia-Shun, L., Hovland, P., and Enbody, R. (1993). Further Research on Feature Selection and Classification Using Genetic Algorithms. In Forrest, S., editor, *Proc. of the 5th Int'l. Conf. on GAs*, pages 557–564. Morgan Kaufmann.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

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

[Vafaie and Jong, 1994] Vafaie, H. and Jong, K. A. D. (1994). Improving a Rule Learning System Using Genetic Algorithms. In *Machine Learning: A Multistrategy Approach*, pages 453–470. Morgan Kaufmann.

[Whitley et al., 1997] Whitley, D., Beveridge, J. R., Guerra-Salcedo, C., and Graves, C. (1997). Messy Genetic Algorithms for Subset Feature Selection. In *Proceedings of the 7th International Conference on Genetic Algorithms*. Morgan Kaufmann.