Genetic Approach to Feature Selection for Ensemble Creation

César Guerra-Salcedo
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
(970) 491-1943
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

Ensembles of classifiers have been shown to be very effective for case-based classification tasks. The vast majority of ensemble construction algorithms use the complete set of features available in the problem domain for the ensemble creation. Recent work on randomly selected subspaces for ensemble construction has been shown to improve the accuracy of the ensemble considerably. In this paper we focus our attention on feature selection for ensemble creation using a genetic search approach. We compare boosting and bagging techniques using three approaches for feature selection for ensemble construction. Our genetic-based method produces more reliable ensembles and up to 80% in memory reduction on the datasets employed.

1 INTRODUCTION

The ability to extract interesting information from large datasets is becoming increasingly important. Accurate classification of data is a key issue for its correct utilization. Classification systems deal with methods for assigning a set of input objects to a set of decision classes. The objects are described by a set of characteristics (such as color, temperature, melting point, etc.). The classification task is carried out by assigning the input instances to a class or classes commonly described by a set of features. There exists an implicit function $F(features) = class$ that maps each object to a class based on the features present in the object. Automatic classification systems become an important topic when talking about data mining and knowledge acquisition for expert systems. Classifiers ensembles have been shown to be very effective in combining the prediction of multiple classifiers to produce a more accurate composite classifier [Quinlan, 1996],[Dietterich, 1998]. Traditionally the only difference between one classifier from the rest of the ensemble has been the cases employed for its training. Several methods for ensemble construction have been proposed, among them are Bagging [Breiman, 1994] and Boosting [Freund and Schapire, 1996]. The main difference between them is the way they choose the cases to be used as training examples. Bagging selects its cases resampling them (with replacement) from the original training set. On the other hand, boosting employs a more sophisticated weighting mechanism. Both bagging and boosting make use of the complete set of features available in the problem domain for each single classifier construction.

Optimal feature extraction 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 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 [Bankert, 1994]. 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 [John et al., 1994]. In other words, the goal is to find a set of relevant features, which when presented to the classifier, maximizes its performance (accuracy).

Recently Tin Kam Ho [Ho, 1998b], [Ho, 1998a] has presented a method for systematic construction of classifiers ensembles based on feature selection. Her

method relies on the fact that combining multiple classifiers constructed using randomly selected features can achieve better performance in classification than using the complete set of features for the ensemble creation. Our principal motivation for this research is to find accurate subsets of features that could be suitable for ensemble creation. The main goal is to have improvements in accuracy as well as savings in storage space. Storing classifiers using the complete set of features is very costly in terms of memory usage.

In this paper we compare Ho's idea of randomly selecting subspaces to construct ensembles of table-based classifiers with the accuracy of ensembles constructed using features previously selected with a genetic search engine. Also we compared traditional boosting and bagging methods (using the complete set of features) with Ho's method and our method. Over the set of experiments our method show better performance and much better use of the storage space.

The paper is organized as follows. First a background review of the material involved in the research is presented. The experimental set up and results are detailed in section 3. In section 4 a brief discussion is presented.

2 BACKGROUND REVIEW

Many researchers have investigated techniques of combining the predictions of multiple classifiers to produce an ensemble of them. Most of the time the ensemble is more accurate than a single classifier. Two methods for ensemble construction have been widely used in recent years, Bagging [Dietterich, 1998] [Breiman, 1994] and Boosting [Dietterich, 1998],[Maclin and Opitz, 1996] (particularly AdaBoost.M1 [Freund and Schapire, 1996]).

2.1 BAGGING

Bagging (**B**ootstrap **agg**regat**ing** [Breiman, 1994]) constructs ensembles of T classifiers from bootstrap samples of N instances. N is the number of instances available in the training data file, each instance belongs to one of Cl classes. Bagging creates an ensemble of classifiers by training each individual classifier C_i $(1 \leq i \leq T)$ with a random redistribution of the training set (resampling with replacement). Each particular classifier is trained with N elements resampled from the train file. Given the fact that Bagging uses resampling with replacement, many instances may be repeated and many may be left out. When the ensemble is used for classification of an instance x, each single classifier votes on the possible class for x. The class with the most votes is used as the class predicted by the ensemble (ties are solved arbitrarily). Given the fact that in a Bagged ensemble all the classifiers are independent, bagging can generate each C_i in an ensemble in parallel.

2.2 BOOSTING

Boosting is a method for ensemble construction that instead of drawing a succession of independent bootstrap samples from the original cases (as bagging does), boosting maintains a weight for each instance. Boosting generates classifiers sequentially, changing the weights for each instance after a single classifier is created. Like bagging, boosting selects a training set of size N for classifier i in the ensemble. Given an integer T specifying the number of trials, T weighted training sets $S_1, S_2, ..S_T$ are generated in sequence and T classifiers $C_1, C_2, .., C_T$ are built. Initially the probability of picking an instance is $1/N$ and is recalculated after every trial. Boosting gives more weight for trial i to instances misclassified in trial $i-1$ $(i \geq 1)$. The probabilities for the next trial are generated by multiplying the probabilities of C_i's incorrectly classified instances by the factor $\beta_i = (1-\epsilon_i)/\epsilon_i$ and then renormalizing these probabilities so that their sum equals 1; ϵ_i is the error of the i single classifier when tested on the original training set. The boosted classifier C^* is obtained by summing the votes of the classifiers $C_1..C_T$, where the vote of the classifier C_i is worth $\log(1/\beta^i)$ units.

2.3 RANDOM SUBSPACE METHOD

The random subspace method for ensemble construction [Ho, 1998b], [Ho, 1998a] was originally conceived for tree-based classifiers. The method relies on a pseudorandom procedure that selects a subset of features (a random subspace of features) from the feature space. All samples in the dataset are projected to this subspace and a decision tree is constructed using the projected training examples. For every feature space of dimension n there are 2^n possible selections that can be made. With each selection a decision tree can be constructed. Ho suggested to construct the trees forming an ensemble using a random selection of $n/2$ features from the complete set of features [Ho, 1998b], [Ho, 1998a]. The method was originally conceived for tree-based ensembles with very good results. Ho was able to find more accurate tree-based ensembles than ensembles constructed using the complete set of features. However, in Ho's research neither bagging nor boosting were employed for ensembles constructed with the subspace method. The ensemble is supposed to be created by independently

constructing each classifier using the complete set of training instances and a particular random subset of features. To classify an unseen case x, each classifier votes on the class for x. The class with the most votes is the class predicted by the ensemble. For this research we apply the ideas of random subspace selection to the construction of ensembles whose base-classifier is a table. Table-based classifiers are widely used, among them are classification algorithms such as k-means clusters [Dillon and Goldstein, 1984], decision tables [Kohavi, 1995], nearest-neighbor classifiers [Punch et al., 1993]. Ho empirically demonstrated that the random subspace method for ensemble construction is better than using the complete set of features for the ensemble. Ho compares her method with bagged and boosted ensembles constructed using the complete set of features. In this research we obtain similar results using Ho's method for table-based classifiers. Also, we extended Ho's work by using her method for "bagged" and "boosted" ensembles.

2.4 FEATURE SELECTION USING GENETIC ALGORITHMS

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 [Aha and Bankert, 1994] [Aha and Bankert, 1995] [Bankert, 1994] [Bankert and Aha, 1996] there are 204 different characteristics to describe a cloud. These characteristics come from spectral, textural and physical measures from each sample area [Bankert, 1994] [Bankert and Aha, 1996].

Searching for an accurate subset of features is a difficult search problem. Search spaces to be explored could be very large. In a cloud classification problem in which each cloud is defined by 204 features there are 2^{204} possible features combinations. Search strategies such as Hill-climbing and Best-first search [Kohavi, 1995], among others, have been used to find subsets of features with high predictive 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. [Bala et al., 1995] [Bala et al., 1996], Vafaie et al. [Vafaie and Jong, 1994] [Vafaie and Imam, 1994] and Turney [Turney, 1997] a decision tree generator is used, in [Guerra-Salcedo and Whitley, 1998] a variant of a decision table is used and in [Punch et al., 1993] 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. We do not know of any application involving GA's and ensemble creation; in all of the references cited before the final product is a single classifier.

Guerra and Whitley show a comparison between GENESIS and CHC as genetic engines for feature selection problems [Guerra-Salcedo and Whitley, 1998]. Part of their results empirically demonstrated that CHC was better search algorithm for feature selection problems than GENESIS.

For the experiments reported here we combine the outputs of several runs of a GA-inducer system in one ensemble of classifier. The GA used for our experiments is an implementation of Eshelman's CHC [Eshelman, 1991].

2.4.1 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 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 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.

2.5 TABLE-BASED CLASSIFIERS EMPLOYED

For this research we employed two table-based classifiers, Euclidean Decision Tables (EDT) as proposed

by Guerra [Guerra-Salcedo and Whitley, 1998] and a modified version of a k-means classifier (KMA). 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 using the feature subset as criteria for projection.
- 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.

A modified version of k-means algorithm used for our experiments is based on the description of the k-means clustering algorithm presented in [Dillon and Goldstein, 1984]. In its original version k-means clustering assumes N individuals with j features each and p different classes. K clusters are defined over the complete set of cases (each case belongs to a particular class p). The elements are distributed among the K clusters according to their attribute values using a Euclidean distance measure. The procedure for clustering is as follows. Search for a partition with small error component E by moving individuals from one cluster to another until no transfer of an individual results in a reduction in E. At the end each cluster has j attributes. Each attribute represents the mean of the values of all the cases considered in the cluster for that particular attribute. Using this approach, cases belonging to different classes could be part of the same cluster. In order to compute the class of the cluster a majority-class membership approach is used. Our modify version allows us to define the number of elements e that make up of each cluster. Clusters are created on a single-class basis. The number of clusters belonging to a particular class depend on the number of elements for that class and e. This modification allows us to avoid elements from different classes being mixed in one cluster.

Table 1: Dataset employed for the experiments. In the DNA dataset the attributes values are 0 or 1. In the Segment dataset the attributes values are floats. In the LandSat dataset the attribute values are integers.

Dataset	Features	Classes	Train Size	Test Size
LandSat	36	6	4435	2000
DNA	180	39	2000	1186
Segment	19	7	210	2100

3 EXPERIMENTAL SETUP

A series of experiments were carried out using publicly available datasets provided by the Project Statlog [1] and by UCI machine learning repository [C. Blake and Merz, 1998]. Table 1 shows the datasets employed for this research.

3.1 ENSEMBLE RELATED SETUPS

Our main objective is to compare the accuracy of ensembles constructed using three different methods for feature selection: first, features selected using a genetic algorithm; second, features selected using the random subspace method; third, ensembles constructed using the complete set of features available. For each method four ensemble creation schemes were used.

- Simple ensemble creation in which an ensemble is formed by classifiers and trained with the complete set of training elements. For this approach a majority-class voting scheme is used for class prediction (in tables referenced as Normal).
- Bagged ensemble creation using the Bagging algorithm described in previous sections.
- Two versions of Adaboost.M1 that we called Adaboost.M1.1 and Adaboost.M1.2 (Boost. 1 and Boost. 2 on the tables).

3.1.1 Adaboost.M1.1

In boosting the initial weight $w_0(x_i)$ of an instance i is $1/N$ (for all i, N is the train-file size) and it is recalculated after every trial. The weights are calculated using the following: if a class for a test case y_i $(1 \leq i \leq N)$ is predicted correctly

$$w_{t+1}(y_i) = w_t(y_i)/(2(1-\epsilon_i)) \quad (1)$$

otherwise

$$w_{t+1}(y_i) = w_t(y_i)/(2\epsilon_i) \quad (2)$$

where ϵ_i is the error of the i single classifier when tested on the original training set. The number of

[1] ftp.ncc.up.pt: pub/statlog/datasets

times an instance x_i $(1 \leq i \leq N)$ has to be present in a particular classifier C_{t+1} represented by $\eta_{t+1}(x_i)$ is based on $w_t(x_i)$ and N. For Adaboost.M1.1 we calculate $\eta_{t+1}(x_i)$ as

$$\eta_{t+1}(x_i) = \lfloor N \cdot w_t(x_i) \rfloor \quad (3)$$

If $A = \sum_{\forall i} \lfloor N * w_t(x_i) \rfloor < N$, $N - A$ elements are still needed to complete N elements for classifier C_{t+1}. Those elements are picked one by one by first sorting the instances according to their $N * w_t(x_i)$ values and then selecting one of each $N - A$ sorted instances. For each instance selected, its value $\eta_{t+1}(x_i)$ is increased by one.

3.1.2 Adaboost.M1.2

In our second version of AdaBoost.M1 the number of copies of x_i to be present at C_{t+1} are assigned according to a "Stochastic Universal Sampling" [Whitley, 1993]. Stochastic Universal Sampling is a method for computing the number of copies assigned to an individual in a genetic algorithm. As in Adaboost.M1.1, for each instance x_i, we calculate $N * w_t(x_i)$. The quantity $\kappa_i = N * w_t(x_i) - \eta_{t+1}(x_i)$ is kept for each instance. If $N - A > 0$ the classifier C_{t+1} still needs $N - A$ elements for training. Those elements are assigned using the first $N - A$ κ_i such that if $\kappa_i > 0.5$ and $rand(seed) > 1 - \kappa_i$ then $\eta_{t+1}(x_i) = \eta_{t+1}(x_i) + 1$.Where $rand(seed)$ is a randomly generated number in the range (0,1).

3.2 GA-CLASSIFIER SETUP

Our main focus is to apply genetic-based search to feature selection for ensemble creation. The idea is to find subsets of features to be used as input for each classifier in an ensemble. Several randomly independent experiments were carried out using this approach. We used datasets with features ranging from 19 up to 180. For all of the experiments the setup was similar.

Using each dataset original train file, 50 independent train and test files were randomly generated $((Tr_1, Ts_1), (Tr_2, Ts_2), .., (Tr_{50}, Ts_{50}))$. 50 different experiments using CHC as search engine were run using these files; experiment i used files (Tr_i, Ts_i). For a particular experiment i a chromosome represents a plausible feature selection vector. In order to evaluate a chromosome and obtain its fitness, a classifier C_i was constructed using Tr_i and tested using Ts_i. After 10000 trials the ga-based engine was stopped and the best individual was saved. At the end of each set of experiments (one for each dataset), 50 individuals were saved. Each one of those individuals was meant to be used as a feature template for a classifier in a particular ensemble. For an ensemble E_k (k representing the dataset employed for that ensemble) the classifier C_j^k uses the individual j in the set of individuals saved for the experiment corresponding to k. Figure 1 depicts graphically our method.

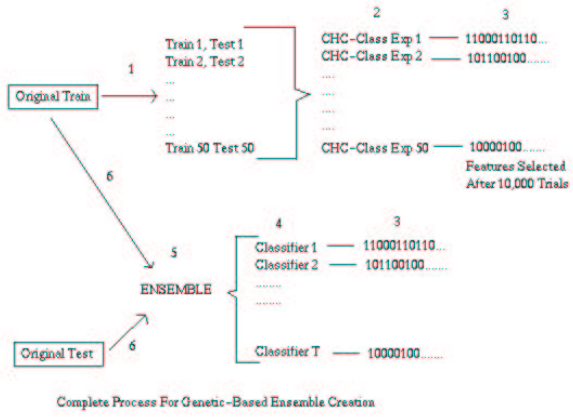

Figure 1: Genetic-based ensemble creation. Numbers represent the order of the operations to do.

4 RESULTS

An ensemble is a colection (aggregation) of several single classifiers. For this research all the classifiers that form an ensemble are of the same type, either EDT or KMA. The type of the classifiers forming an ensemble is refered as base classifier.

Two different sets of results were obtained from our experiments. First, a comparison between different approaches for constructing ensembles based on selection of features was carried out. Ensembles were constructed using the complete set of features, a randomly generated subset of features (random subspace method) and a subset of features obtained by genetic search. For each feature-selection method four different methods of ensemble creation were employed; Construction based on the complete set of instances, bagging, Adaboost.M1.1 and Adaboost.M1.2. Second, a comparison in memory usage based on the total number of features used for the different approaches was carried out as well.

For the random subspace method, 50 feature subsets were generated once for each dataset. The subsets were then used as input for the ensemble-constructor algorithm. The same 50 random-feature subsets were used for all the experiments involving a dataset.

Table 2: Accuracy using EDT as base classifier for the ensemble.

Feature Selection	Normal			Bagging			Boost. 1			Boost. 2		
Method	DNA	SAT	SEG	DNA	SAT	SEG	DNA	SAT	SEG	DNA	SAT	SEG
CHC-EDT	92.41	**91.20**	92.28	92.32	**91.10**	92.71	93.33	**91.20**	92.09	92.91	91.15	92.38
CHC-KMA	**93.59**	91.05	92.76	**93.84**	91.05	93.00	**93.92**	91.10	92.38	**93.84**	**91.45**	92.42
Random	87.52	91.00	**94.14**	88.02	91.00	**93.80**	87.52	91.15	**93.09**	87.94	90.95	**94.14**
All Feat.	75.88	89.45	87.38	76.55	89.50	87.33	75.88	89.45	87.38	74.03	88.00	85.42

Table 3: Accuracy using KMA as base classifier for the ensemble.

Feature Selection	Normal			Bagging			Boost. 1			Boost. 2		
Method	DNA	SAT	SEG	DNA	SAT	SEG	DNA	SAT	SEG	DNA	SAT	SEG
CHC-EDT	91.48	90.35	91.52	90.69	**90.90**	92.33	93.25	90.70	87.00	94.26	90.90	92.38
CHC-KMA	**92.66**	90.65	**92.00**	92.66	90.80	**92.71**	**93.92**	90.70	91.04	**94.51**	91.30	92.92
Random	92.58	**90.85**	85.00	91.73	90.45	88.28	91.98	**91.10**	**92.85**	92.49	**91.45**	**93.38**
All Feat.	88.70	89.20	76.23	**93.25**	90.50	79.28	93.84	89.70	78.38	93.67	89.90	84.33

4.1 ACCURACY AND EFFECTIVENESS OF THE METHODS

The results presented in Table 2 and Table 3 are the average of 10 independent runs. Table 2 presents the results of ensemble creation algorithms using EDT as a constituing classifier for the ensembles. For Table 3 the base classifier for the ensembles was KMA. In both tables, the rows labeled CHC-EDT and CHC-KMA represents ensembles constructed using the features obtained by CHC using EDT as evaluation function and by CHC using KMA as evaluation function respectively.

In Table 2 the EDT-based ensembles created using the features obtained by the genetic search approach, either CHC-EDT or CHC-KMA, was best in eight out of 12 experiments (with two ties). Also, the features obtained by CHC-KMA seems to be more effective for EDT-based ensembles than the ones obtained by the CHC-EDT system itself. They were the best option in five out of the eight times when the genetic-search approach was better than the other feature selection methods.

The ensembles created using the random subspace method won in four competitions. This option seems to be the best option for datasets with small number of features as in the Segmentation dataset. For EDT-based ensembles and the datasets employed in this research, the use of the complete set of features for ensemble creation was least robust. This option presented the worst performance.

From the comparisons between bagging, Adaboost.M1.1, Adaboost.M1.2, and the use of the complete set of instances for EDT-based ensembles, bagging and Adaboost.M1.1 were the most successful methods. The use of the complete set of instances was effective only in three experiments and Adaboost.M1.2 in 2. These results are summarized in Table 4.

Table 4: Comparison between Bagging, Adaboost.M1.1, Adaboost.M1.2 and the use of the complete set of instances (depicted as Normal) for ensemble creation using EDT as base classifier

Algorithm	DNA	SAT	SEG
CHC-EDT	Boost.1	Boost.1/Normal	Bagging
CHC-KMA	Boost.1	Boost.2	Bagging
Random	Bagging	Boost.1	Boost.2/Normal
All Feat.	Bagging	Bagging	Boost.1/Normal

Table 5: Comparison between Bagging, Adaboost.M1.1, Adaboost.M1.2 and the use of the complete set of instances (depicted as Normal) for ensemble creation using KMA as base classifier

Algorithm	DNA	SAT	SEG
CHC-EDT	Boost.2	Boost.2/Bagging	Boost.2
CHC-KMA	Boost.2	Boost.2	Boost.2
Random	Boost.2	Boost.2	Boost.2
All Feat.	Boost.1	Bagging	Boost.2

On the other hand, using KMA as base classifier for ensembles, the genetic search approach was better six times, random subspace method five and using all features was best for one experiment. These results are summarized in Table 3. Once again the use of the complete set of features was not a robust alternative.

When comparing the different methods for ensemble creation, bagging, Adaboost.M1.1, Adaboost.M1.2 and the use of the complete set of instances for KMA-based ensembles. Adaboost.M1.2 were the most successful method in 10 out of 16 experiments. Bagging was better only in two experiments and Adaboost.M1.1 in one. The use of the complete set of instances was the worst approach. These results are summarized in Table 5.

On the other hand, when comparing the accuracy of EDT-based ensembles with the accuracy of KMA-based ensembles, EDT-based ensembles were more accurate (higher accuracy percentages). However, the

Table 6: Comparison Between EDT and KMA for ensemble creation using four different ensemble creation algorithms. The best accuracies as well as the system that produce them are depicted in parenthesis.

Algorithm	DNA	SAT	SEG
Normal	EDT (93.59% CHC-KMA)	EDT (91.20% CHC-EDT)	EDT (**94.14% Random**)
Bagging	EDT (93.84% CHC-KMA)	EDT (91.10% CHC-EDT)	EDT (93.80% Random)
Boost. 1	EDT/KMA (93.92% CHC-KMA)	EDT (91.20% CHC-EDT)	EDT (93.09% Random)
Boost. 2	KMA (**94.51% CHC-KMA**)	EDT/KMA (**91.45% CHC-KMA/Random**)	EDT (**94.14% Random**)

results obtained by KMA-based ensembles are more uniform for all the approaches. EDT-based ensembles showed very poor performance when the complete set of features was used for the creation of the ensemble.

The best classifier turn out to be EDT. However, as mentioned above, the features obtained by the system CHC-KMA were very effective for the ensemble creation experiments. Table 6 shows the comparisons between methods and classifiers for each dataset.

4.2 EXPERIMENTS RELATED TO MEMORY USAGE

The second set of experiments carried out a comparison in the number of features selected by each approach. In a table-based classifier each feature is represented as a column. Reducing the number of features reduces the number of columns as well; fewer columns employed in a classifier represents less memory used for storage. Experiments involving memory usage for ensemble creation are very important. One of the caveats of using classifier ensembles is the enormous amount of memory used to store the ensemble.

An important advantage of the genetic search method is its ability to obtain small feature subsets. Smaller tables are easier to implement and to store. In our research we obtain smaller tables using the genetic search method. The comparison between the number of features (columns) obtained by the genetic-search method and the number of features obtained by the other methods is presented in Table 7. The percentage of savings in feature space for DNA are in the order of 88% compared with random subspace method and 93% compared to ensembles constructed using the whole set of features. For LandSat dataset the percentage of savings in feature space was 40% compared to random subspace method and 70% compared to the use of the complete set of features for the ensemble construction. For the Segmentation dataset the percentages of savings compared to random subspace method was 64% and 81% compared to ensembles created using all the available features.

Table 7: Average number of features used for the ensemble.

Algorithm	DNA	SAT	SEG
CHC-EDT	11.24 $\pm$ 2.13	12.6 $\pm$ 1.89	3.6 $\pm$ 0.728
CHC-KMA	16.26 $\pm$ 3.5	11.04 $\pm$ 2.16	5.9 $\pm$ 0.886
Random	90.00	18.00	10.00
All Feat.	180.00	36.00	19.00

5 DISCUSSION

Ensemble construction is a very important method for improving classifier accuracy. We are proposing a novel method for selecting features for ensemble construction. Our method has been empirically shown to be more accurate than other methods proposed elsewhere [Ho, 1998b], [Ho, 1998a]. The main advantage of our approach is the enormous percentage of savings in storage for table-based ensembles.

Another important contribution of this research is the modified boosting scheme (labeled in the results as Boost. 2) which has empirically shown to be more effective than traditional boosting.

Acknowledgments

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

References

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

[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., 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).*

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

[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.

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

[Breiman, 1994] Breiman, L. (1994). Bagging Predictors. Technical Report 421, Dept. of Statistics Technical Report 421, University of California, Berkeley, California.

[C. Blake and Merz, 1998] C. Blake, E. K. and Merz, C. (1998). UCI repository of machine learning databases.

[Dietterich, 1998] Dietterich, T. G. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning (submitted)*, 3:1–22.

[Dillon and Goldstein, 1984] Dillon, W. R. and Goldstein, M. (1984). *Multivariate Analysis Methods and Applications.* John Wiley and Sons.

[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.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In Saitta, L., editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann.

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

[Ho, 1998a] Ho, T. K. (1998a). C4.5 Decision Forest. In *Proceedings of the 14th International Conference on Pattern Recognition*, pages 605–609.

[Ho, 1998b] Ho, T. K. (1998b). The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20-8:832–844.

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

[Maclin and Opitz, 1996] Maclin, R. and Opitz, D. (1996). An Empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551. AAAI Press/MIT Press.

[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, 1996] Quinlan, J. R. (1996). Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press/MIT Press.

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

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

[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, 1993] Whitley, L. D. (1993). A Genetic Algorithm Tutorial. Technical Report Nb. CS-93-103, Colorado State University.