Improving Cooperative GP Ensemble with Clustering and Pruning for Pattern Classification

Gianluigi Folino
ICAR-CNR
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
folino@icar.cnr.it

Clara Pizzuti
ICAR-CNR
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
pizzuti@icar.cnr.it

Giandomenico Spezzano
ICAR-CNR
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
spezzano@icar.cnr.it

ABSTRACT

A boosting algorithm based on cellular genetic programming to build an ensemble of predictors is proposed. The method evolves a population of trees for a fixed number of rounds and, after each round, it chooses the predictors to include into the ensemble by applying a clustering algorithm to the population of classifiers. The method proposed runs on a distributed hybrid multi-island environment that combines the island and cellular models of parallel genetic programming. The large amount of memory required to store the ensemble makes the method heavy to deploy. The paper shows that by applying suitable pruning strategies it is possible to select a subset of the classifiers without increasing misclassification errors; indeed, up to 20% of pruning, ensemble accuracy increases. Experiments on several data sets shows that combining clustering and pruning enhances classification accuracy of the ensemble approach.

Categories and Subject Descriptors

H.2.8 [**Database Managment**]: Database Applications — *Data Mining*; I.2.2 [**Artificial Intelligence**]: Automatic Programming

General Terms

Algorithms

Keywords

Genetic Programming, Data Mining, Classification, Ensemble

1. INTRODUCTION

Ensemble learning algorithms have captured an increasing interest in the research community because of their capability of improving the classification accuracy of any single classifier. An ensemble of classifiers is constituted by a set of predictors that, instead of yielding their individual decisions to classify new examples, combine them together by adopting some strategy [5, 18, 6, 11, 4]. It has been pointed out that the boost in accuracy is tightly related with the diversity of the classifiers [11, 21]. Two classifiers are diverse if they make different incorrect predictions on new data points. Several approaches for building ensembles satisfying the diversity demand have been proposed. The *AdaBoost* algorithm introduced by Freund and Schapire [18] showed to be efficacious at generating different classifiers because it guides the underlying learning algorithm at focusing on harder examples by adaptively changing the distributions of the training set on the base of the performance of the previous classifiers.

The combination of Genetic Programming (GP) [20] and ensemble techniques has been receiving a lot of attention because of the improvements that GP obtains when enriched with these methods [19, 27, 22, 9, 15, 17]. In particular, $BoostCGPC$ (*Boost Cellular Genetic Programming Classifier*) [17] implements the $AdaBoost.M1$ boosting algorithm of Freund and Shapire [18] on a parallel computer by using the algorithm $CGPC$ (*Cellular Genetic Programming for data classification*) [14] as base classifier. Given a training set S of size N and the number P of processors used to run the algorithm, $BoostCGPC$ partitions the population of classifiers in P subpopulations, creates P subsets of tuples of size $n < N$ by uniformly sampling instances from S with replacement, and builds an ensemble of classification trees by choosing from each subpopulation the individual having the best fitness.

As already observed, one of the main problems of voting classification algorithms is the generation of many different predictors, thus a spontaneous observation that arises is that a population of predictors by itself constitutes an ensemble, thus why choose only one individual, though it is the one with the best fitness? Considering that the size of a population is generally not small, taking all the individuals of each subpopulation could not be a practical approach because of the resulting high number of predictors. A reasonable proposal could be to use a clustering algorithm [12] to group individuals in the population that are similar with respect to a similarity measure and then take the representatives of these clusters.

In this paper a distributed Boosting Cellular Genetic Programming Classifier endowed with a clustering strategy to build the ensemble of predictors is proposed. The algorithm, named $ClustBoostCGPC$ (*Clustering Boost Cellu-*

GECCO'06 July 8-12, 2006, Seattle, Wa, USA

lar Genetic Programming Classifier), instead of choosing, like *BoostCGPC*, from each subpopulation the individual having the best fitness, finds k groups of individuals similar with respect to a similarity measure and then takes the individual of each cluster having the best fitness. The algorithm runs on a distributed environment based on a hybrid multi-island model [2] that combines the island model with the cellular model. Each node of the network is considered as an island that contains a learning algorithm, based on cellular genetic programming, whose aim is to generate decision-tree predictors trained on the local data stored in the node. Every genetic program, however, though isolated, cooperates with the neighboring nodes by collaborating with the other learning components located on the network and takes advantage of the cellular model by exchanging the outermost individuals of the population.

The main drawback of the approach proposed is that the size of the ensemble increases as the number of clusters and the nodes of the network. Thus we could ask if it it possible to discard some of these predictors and still obtain comparable accuracy. The paper shows that by applying suitable pruning strategies it is possible to select a subset of the classifiers without increasing misclassification errors; indeed, up to 20% of pruning, ensemble accuracy increases. Experiments on several data sets shows that the combination of clustering and pruning enhances classification accuracy of the ensemble approach.

The paper is organized as follows. The next section reviews the Boosting algorithms of Freund and Schapire [26]. In Section 3 the algorithm *ClustBoostCGPC* and software architecture used to run it are described. Section 4 describes the pruning strategy adopted to reduce the size of the ensemble. In section 5, finally, the results of the method on some standard problems are presented.

2. ENSEMBLE TECHNIQUES

Let $S = \{(x_i, y_i)|i = 1, \ldots, N\}$ be a training set where x_i, called example or tuple or instance, is an attribute vector with m attributes and y_i is the class label associated with x_i. A predictor (classifier), given a new example, has the task to predict the class label for it. Ensemble techniques [5, 25, 11, 4] build T predictors, each on a different training set, then combine them together to classify the test set. Boosting was introduced by Schapire [25] and Freund [26] for boosting the performance of any "weak" learning algorithm, i.e. an algorithm that "generates classifiers which need only be a little bit better than random guessing" [26]. The boosting algorithm, called *AdaBoost*, adaptively changes the distribution of the training set depending on how difficult each example is to classify. Given the number T of trials (rounds) to execute, T weighted training sets $S_1, S_2, \ldots, S_T$ are sequentially generated and T classifiers $C^1, \ldots, C^T$ are built to compute a weak hypothesis h_t. Let w_i^t denote the weight of the example x_i at trial t. At the beginning $w_i^1 = 1/n$ for each x_i. At each round $t = 1, \ldots, T$, a weak learner C^t, whose error ϵ^t is bounded to a value strictly less than $1/2$, is built and the weights of the next trial are obtained by multiplying the weight of the correctly classified examples by $\beta^t = \epsilon^t/(1 - \epsilon^t)$ and renormalizing the weights so that $\Sigma_i w_i^{t+1} = 1$. Thus "easy" examples get a lower weight, while "hard" examples, that tend to be misclassified, get higher weights. This induces AdaBoost to focus on examples that are hardest to classify. The boosted classifier gives the class label y that maximizes the sum of the weights of the weak hypotheses predicting that label, where the weight is defined as $ln(1/\beta^t)$. Freund and Schapire [26] showed theoretically that AdaBoost can decrease the error of any weak learning algorithm and introduced two versions of the method. In this paper we use AdaBoost.M1.

3. CLUSTBOOSTCGPC

In this section the description of the algorithm *ClustBoostCGPC* is given. The method builds an ensemble of classifiers by using, at each round of the boosting procedure, the algorithm *CGPC* (*Cellular Genetic Programming for data classification*) [14] to create a population of predictors and the clustering algorithm *k-means* to choose a fixed number k of predictors in the population. Before giving a detailed outline of the approach proposed, a brief review of the *CGPC* and *k-means* methods is provided.

3.1 The CGPC algorithm

Genetic programming can be used to inductively generate a GP classifier as a decision tree for the task of data classification [20]. Decision trees, in fact, can be interpreted as composition of functions where the function set is the set of attribute tests and the terminal set are the classes. The function set can be obtained by converting each attribute into an attribute-test function. For each attribute A, if $A_1, \ldots A_n$ are the possible values A can assume, the corresponding attribute-test function f_A has arity n and if the value of A is A_i then $f_A(A_1, \ldots A_n) = A_i$. When a tuple has to be evaluated, the function at the root of the tree tests the corresponding attribute and then executes the argument that outcomes from the test. If the argument is a terminal, then the class name for that tuple is returned, otherwise the new function is executed. The CGPC algorithm used for data classification is described in figure 1.

```
Let p_c, p_m be crossover and mutation probability
   for each point i in grid do in parallel
   generate a random individual h_i
   evaluate the fitness of h_i
end parallel for
while not MaxNumberOfGeneration do
   for each point i in grid do in parallel
      generate a random probability p
      if (p < p_c)
         select the cell j, in the neighborhood of i,
         such that h_j has the best fitness
         produce the offspring by crossing h_i and h_j
         evaluate the fitness of the offspring
         replace h_i with the best of the two offsprings
         evaluate the fitness of the new h_i
      else
      if ( p < p_m + p_c) then
         mutate the individual
         evaluate the fitness of the new h_i
      else
         copy the current individual in the population
      end if
      end if
end parallel for
end while
```

Figure 1: The algorithm CGPC.

CGPC adopts a cellular model of GP [24]. In the cellular model each individual has a spatial location, a small neighborhood and interacts only within its neighborhood. The main difference in a cellular GP, with respect to a *panmictic* algorithm, is its decentralized selection mechanism and the genetic operators (crossover, mutation) adopted.

CGPC generates a classifier as follows. At the beginning, for each cell, the fitness of each individual is evaluated. The fitness is the number of training examples classified in the correct class. Then, at each generation, every tree undergoes one of the genetic operators (reproduction, crossover, mutation) depending on the probability test. If crossover is applied, the mate of the current individual is selected as the neighbor having the best fitness, and the offspring is generated. The current tree is then replaced by the best of the two offsprings if the fitness of the latter is better than that of the former. After the execution of the number of generations defined by the user, the individual with the best fitness represents the classifier.

3.2 The k-means algorithm

The algorithm *k-means* [12] is a well known clustering method that partitions a set of objects into k groups so that the intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured with respect to the mean value of the objects in a cluster, which can be considered as the cluster's center. The algorithm first randomly selects k objects, and assigns the remaining objects to the most similar cluster, where similarity is computed as the distance between the object and the center of the cluster. After that, the new mean values of the clusters are computed and this process is repeated until the criterion function converges. Typically, the squared-error criterion is used, defined as $E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, m_i)^2$, where E is the sum of square error for all the objects, *dist* is a distance measure, generally the Euclidean distance, p is an object, and m_i is the mean of cluster C_i. Both p and m_i are multidimensional objects.

In order to apply the *k-means* algorithm to a population of trees, it is necessary to specify the concept of distance between two individuals. To this end, each classification tree h is represented by a couple (f, e), where f is its fitness value and e is its distance from the empty tree Φ, considered as the origin tree. The metric adopted to measure the structural distance between two genetic trees is that introduced by Ekárt and Németh [13]. The distance between two trees h_1 and h_2 is calculated in three steps: (1) h_1 and h_2 are overlapped at the root node and the process is applied recursively starting from the leftmost subtrees. (2) For each pair of nodes at matching positions, the difference of their codes (possibly raised to an exponent) is computed. (3) The differences computed in the previous step are combined in a weighted sum. Formally, the distance of two trees h_1 and h_2 with roots R_1 and R_2 is defined as:

$$dist(h_1, h_2) = d(R_1, R_2) + \frac{1}{H} \sum_{i=1}^{m} dist(child_i(R_1), child_i(R_2))$$

where: $d(R_1, R_2) = (|c(R_1) - c(R_2)|)^z$, c is a numeric code assigned to each node of the tree, $child_i(Y)$ is the i^{th} of the m possible children of a generic node Y, if $i \leq m$, or the empty tree otherwise. The constant H is used to give different weights to nodes belonging to different levels and z is a constant such that $z \in \mathcal{N}$. Since the nodes of a classification tree are labelled with the attribute's names of the training examples, each attribute is coded with a number and this number is used as the code associated with each node labelled with that attribute. When computing the distance between a tree and the empty tree, $dist(h, \Phi)$ gives simply a weighted sum of the codes associated with the attributes appearing in the tree.

Once for each tree the couple (f, e) has been computed, since both f and e are numbers, the k-means algorithm employs the Euclidean distance to the tree population by using this two dimensional representation.

3.3 The ClustBoostCGPC algorithm to build GP ensemble

ClustBoostCGPC is a new cooperative ensemble learning algorithm for constructing GP ensembles. The idea is to incorporate different GP classifiers, each trained on different parts or aspects of the training set, so that the ensemble can better learn from the whole training data. *ClustBoostCGPC* applies the boosting technique in a distributed hybrid multi-island model of parallel GP and uses a clustering-based selective algorithm to maintain the diversity of the ensemble by choosing in each population the most accurate predictors of each group.

Our approach aims to emphasize the cooperation among the individuals of the population (classifiers) using a hybrid model of parallel GP that combines the multi-island and cellular models of GP to enhance accuracy and to reduce performance fluctuation of programs produced by GP.

The multi-island model is based on subpopulations created by dividing the original population into disjunctive subsets of individuals, usually of the same size. Each subpopulation is assigned to an island and a standard (panmictic) GP algorithm is executed on it. Occasionally, migration process between subpopulations is carried out after a fixed number of generations. The hybrid model modifies the multi-island model by substituting the standard GP algorithm with a cellular GP algorithm [16]. In our model we use the *CGPC* algorithm in each island. Each island operates in parallel on a subset of the tuples of the whole training set. The training and combination of the individual classifiers are carried together in the same learning process by a cooperative approach. Our model is based on the coevolution of different subpopulations of classifiers and a migration process that transfers asynchronously individuals among subpopulations.

In order to improve the prediction accuracy achieved by an ensemble, we need to ensure accuracy of classifiers and diversity among them. Although GP does not require any change in a training data to generate individuals of different behaviors, in [17] it is showed that GP enhanced with a boosting technique improves both the prediction accuracy and the running time with respect to the standard GP. *ClustBoostCGPC* combines the boosting method and the distributed hybrid model of GP to iteratively build an ensemble of classification trees through a fixed number of round.

The selection, at each round, of classifiers satisfying both high diversity and accuracy requirements, is a difficult optimization task. To this end in *ClustBoostCGPC* we applied a method that gradually achieves diversity and accuracy. First, we employ the k-means clustering algorithm to divide all individuals of each subpopulation into groups (clusters) according to similarity of the classifiers. Then, the most

accurate individual in each group is selected.

A more formal description the algorithm, in pseudo-code, is shown in figure 2.

Let a network of P nodes be given, each having a training set S_j of size n_j. At the beginning, for every node N_j, $j = 1, \ldots, P$, a subpopulation Q_j is initialized with random individuals and the weights of the training instances are set to $1/n_j$. Each subpopulation Q_j is evolved for t generations and trained on its local training set S_j by running a copy of the $CGPC$ algorithm (figure 1). After t generations, the evolved population of trees is clustered by using the *k-means* algorithm [12] and k groups of individuals are determined. For each group, the tree having the best fitness is chosen as representative of the cluster and output as the hypothesis computed. Then the k individuals of each subpopulation are exchanged among the P nodes and constitute the ensemble of predictors used to determine the weights of the examples for the next round.

Given a network constituted by P nodes,
each having a data set S_j of size n_j
For j = 1, 2, ..., P (for each island in parallel)
  Initialize the weights $w_i^1 = \frac{1}{n_j}$ for $i = 1, \ldots, n_j$,
  where n_j is the number of training examples on each node j.
  Initialize the subpopulation Q_j, for $j = 1, \ldots, P$
  with random individuals
end parallel for
For t = 1,2,3, ..., T
  For j = 1, 2, ..., P (for each island in parallel)
  Train $CGPC$ on the partition S_j using a weighted
  fitness according to the distribution w^t
  Run the k-means algorithm to compute
  k weak hypotheses $h_{j_1,t}....h_{j_k,t} : X \rightarrow Y$
  Exchange the hypotheses $h_{j_c,t}$ $c = 1, \ldots, k$
    among the P nodes
  let $D_{j_c} = 1$ if arg max $h_{j_c,t}(x_i) \neq y_i$
    0 otherwise
  Compute the error $\epsilon_{j_c}^t = \sum_{i=1}^n w_i^t D_{i_c}$
  Set $\beta_{j_c}^t = \epsilon_{j_c}^t / (1 - \epsilon_{j_c}^t)$,
  Compute $avg\_beta_j^t = \frac{\sum_{c=1}^k \beta_{j_c}^t}{k}$
  Update the weights $w_i^{t+1} = avg\_beta_j^t \times w_i^t$ if $h_{j,t}(x_i) = y_i$
end parallel for
end for t
output the hypothesis :

$$h_f = arg\ max\ (\sum_{t=1}^T \sum_{j=1}^P \sum_{c=1}^k log(\frac{1}{\beta_{j_c}^t}) D_{j_c}^t)$$

where $D_{j_c}^t = 1$ if $h_{j_c,t}(x_i) = y_i$, 0 otherwise

Figure 2: The algorithm CLUSTBOOSTCGPC

We implemented *ClustBoostCGPC* using a distributed infrastructure and a distributed framework to run GP.

The software architecture of *ClustBoostCGPC* is illustrated in figure 3.

We used *dCAGE* (distributed Cellular Genetic Programming System) a distributed environment to run genetic programs by a multi-island model, which is an extension of [16]. *dCage* has been modified to support the hybrid variation of the classic multi-island model.

In the new implementation, to take advantage of the cellular model of GP, the islands are evolved independently using the $CGPC$ algorithm, and the outermost individuals are asynchronously exchanged. The training sets assigned to each islands can be thought as portions of the overall data set. *dCAGE* distributes the evolutionary processes (islands)

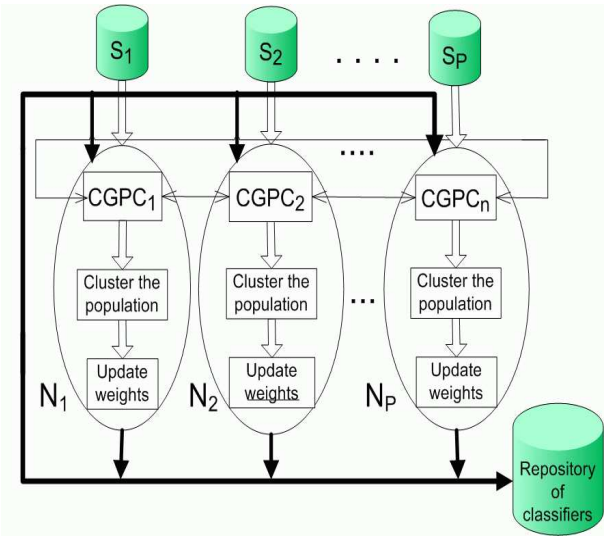

Figure 3: Software architecture of CustBoostCGPC.

that implement the classification models over the network nodes using a configuration file that contains the configuration of the distributed system. *dCAGE* implements the hybrid model as a collection of cooperative autonomous islands running on the various hosts within an heterogeneous network that works as a peer-to-peer system. The MPI (*Message Passed Interface*) library is used to make cooperate the islands. Each island, employed as a peer, is identical to each other. At each round, a *collector* process collects the GP classifiers from the other nodes, handling the fusion of the results on behalf of the other peers, and redistributes the GP ensemble for future predictions to all the network nodes.

We assume that each training set $S_i, i = 1, \ldots, P$ resides on a different node. The size of each subpopulation $Q_i, i = 1, \ldots, P$ present on a node, must be greater than a threshold determined from the granularity supported by the processor. Each node, using a training set S_i and a subpopulation Q_i implements a classifier process $CGPC_i$ as learning algorithm and generates a population of classifiers. Communication among the nodes is local and asynchronous. The configuration of the structure of the processors is based on a ring topology and a classifier process is assigned to each. During the boosting rounds, each classifier process maintains the local vector of the weights that directly reflect the prediction accuracy on that site. At every boosting round the hypotheses generated by each of these classifiers ($CGPC_i$ in Figure 3) are clustered employing the standard k-means algorithm. Then, the most accurate classifier in each group is selected to be included in the ensemble of predictors.

Next, the ensemble built so far is broadcasted to each classifier process to locally recalculate the new vector of the weights and a copy of the ensemble is stored in a repository. After the execution of the fixed number T of boosting rounds, the classifiers stored in the repository are used to evaluate the accuracy of the classification algorithm.

Table 1: Data sets used in the experiments

Dataset	Attr.	Tuples	Classes
Adult	14	48842	2
Census	41	299285	2
Covtype	54	581012	7
Mammography	10	11183	2
Phoneme	5	5404	2
Satimage	36	6435	6
Segment	19	2310	7

Table 2: Main parameters used in the experiments

Name	Value
max_depth_for_new_trees	6
max_depth_after_crossover	17
max_mutant_depth	2
grow_method	RAMPED
selection_method	GROW
crossover_func_pt_fraction	0.7
crossover_any_pt_fraction	0.1
fitness_prop_repro_fraction	0.1
parsimony_factor	0

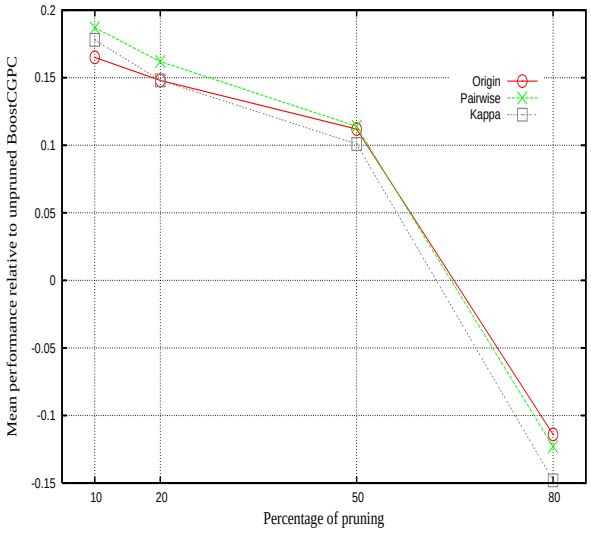

Figure 4: Mean performance of pruned ClustBoostCGPC relative to unpruned ClustBoostCGPC with different pruning percentages.

4. REDUCING THE SIZE OF THE ENSEMBLE

A drawback of the method proposed, and of the ensemble methods in general, is the large amount of memory required to maintain the classifiers. In our case, the size of the ensemble increases as the number of clusters and the number of nodes of the network. Thus we could ask if it it possible to discard some of the predictors generated and still obtain comparable accuracy. This approach is well known in the literature and it is called *pruning* [23] or *thinning* [3] the ensemble. Pruning the ensemble requires a strategy to choose the classifiers to remove. There is a general agreement that the predictors forming the ensemble have to be both diverse and accurate. A pruning policy thus identifies the most similar classifiers and removes them. The concept of similarity in this context plays a central role. In the Machine Learning community diversity means that the predictors have to make independent classification errors, i.e. they disagree with each other. A disagreement measure used in [23] is, for example, the κ statistics [1]. In the Genetic Programming community the concept of diversity is perceived in a different way [7, 8], in particular it reflects the structural diversity of the genetic programs in a generation [13]. In this paper we adopt different diversity measures to choose the trees to prune, we compare them and we show, in the experimental results, that the ensemble can be quite substantially pruned without increasing misclassification errors; indeed, up to 20% of pruning, ensemble accuracy increases. The first two diversity measures used are the pairwise distance between two trees (denoted *pairwise*), and the distance of a tree from the empty tree (denoted *origin*), introduced in subsection 3.2. The third measure is the κ statistics, defined as follows. Given two classifiers h_i and h_j, where $h_i, h_j : X \rightarrow Y$, consider the following $\mid Y \mid \times \mid Y \mid$ contingency table M. For elements $a, b \in Y$, define $M_{a,b}$ to contain the number of examples x in the training set for which $h_i(x) = a$ and $h_j(x) = b$. If h_i and h_j give identical classifications, all non-zero counts will appear along the diagonal. If h_i and h_j are very different, then there should be a large number of counts off the diagonal. Let $\Theta_1 = \frac{\sum_{a=1}^{|Y|} M_{a,a}}{N}$ be the probability that two classifiers agree, where N is the size of the training set and $\mid Y \mid$ is the number of different classes. Let also $\Theta_2 = \sum_{a=1}^{|Y|}(\frac{\sum_{b=1}^{|Y|} M_{a,b}}{N} \frac{\sum_{b=1}^{|Y|} M_{b,a}}{N})$ be the probability that two classifiers agree by chance, given the observed counts in the table. Then the κ measure of disagreement between classifiers h_i and h_j is defined as $\kappa(h_i, h_j) = \frac{\Theta 1 - \Theta 2}{1 - \Theta 2}$. A value of $\kappa = 0$ implies that $\Theta 1 = \Theta 2$ and the two classifiers are considered different. A value of $\kappa = 1$ implies that $\Theta 1 = 1$, which means that the two classifiers agree on each example.

Thus a pruning strategy first computes the κ, *origin*, and *pairwise* measures and then chooses the predictors to eliminate in the following way. If the *origin* measure is used, the predictors h_i are ordered in increasing order of $dist(h_i, \Phi)$ and eliminated by starting with that having the lowest value until the pruning percentage fixed has been reached.

If *pairwise* or κ measures are adopted, the pruning strategy choose pairs (h_i, h_j) of classifiers by starting with the pair having the lowest $dist(h_i, h_j)$ or $\kappa(h_i, h_j)$ value, considering them in increasing order of *dist* or κ until the pruning percentage fixed has been reached.

5. EXPERIMENTAL RESULTS

In this section *ClustBoostCGPC* and *BoostCGPC* are compared on 7 data sets. Two data sets (*Census* and *Covtype*) are from the UCI KDD Archive[1], three (*Segment*, *Satimage*, and *Adult*) are taken from the UCI Machine Learning Repository [2], one (*Phoneme*) is from the ELENA project [3], and

[1]http://kdd.ics.uci.edu/
[2]http://www.ics.uci.edu/∼mlearn/MLRepository.html
[3]ftp.dice.ucl.ac.be in the directory pub/neural/ELENA/databases

Table 3: Comparison of the misclassification error of BoostCGPC and ClustBoostCGPC (5 and 10 clusters).

Dataset	BoostCGPC	ClustBoostCGPC (5 clusters)	ClustBoostCGPC (10 clusters)
Adult	17.231 ± 0.164	14.749 ± 0.163	14.641 ± 0.161
Census	6.120 ± 0.045	4.695 ± 0.042	4.681 ± 0.041
Covtype	33.374 ± 0.468	30.850 ± 0.453	30.044 ± 0.442
Mammography	2.159 ± 0.099	1.309 ± 0.093	1.304 ± 0.091
Phoneme	19.121 ± 0.475	16.968 ± 0.462	16.918 ± 0.448
Satimage	21.398 ± 0.573	20.220 ± 0.552	20.185 ± 0.543
Segment	13.452 ± 0.421	12.127 ± 0.418	12.114 ± 0.417

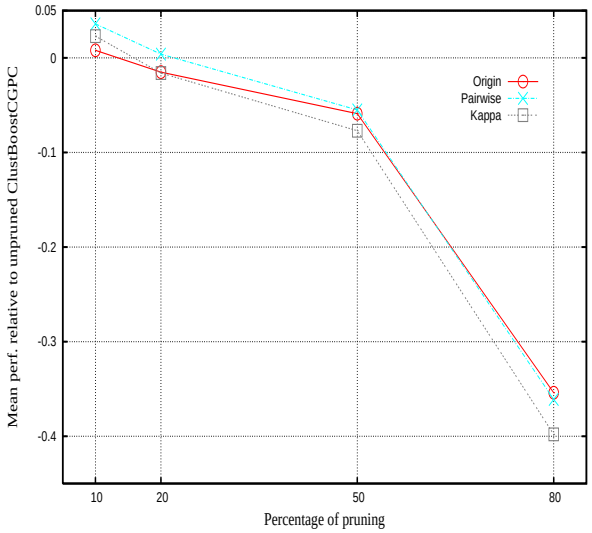

Figure 5: Mean performance of pruned ClustBoostCGPC relative to unpruned BoostCGPC with different pruning percentages.

one (*Mammography*) is a research data set used in [10]. The size and class distribution of these data sets are described in table 1.

The experiments were performed using a network composed by 10 1,133 Ghz Pentium III nodes having 2 Gbytes of Memory, interconnected over high-speed LAN connections.

All results were obtained by averaging 50 runs. In order to do a fair comparison between *ClustBoostCGPC*, and *BoostCGPC* we used a network of 10 nodes for both the algorithms, the number T of rounds was 10, population size 100 on each node, number of generations 100 (for a total number of generations 100×10=1000), and number of clusters fixed for *ClustBoostGPC* 5 and 10. Thus *BoostCGPC* generated 100 classifiers, while *ClustBoostCGPC*, in one run 500 predictors, and in the other run 1000 predictors. A main difference between *ClustBoostCGPC* and *BoostCGPC* regards the partitioning of the training sets on the nodes of the network. *ClustBoostCGPC* runs on a distributed environment where it is supposed that each node has its own data set. In order to simulate this kind of situation, each data set has been equally partitioned among the 10 nodes. Thus each node contains 1/10 of the training set. *BoostCGPC* runs on a parallel computer, thus according to the sequential *AdaBoost* approach, it creates 10 subsets of tuples of size 1/10 the overall training set by uniformly sampling instances with replacement. The parameters used for the experiments are shown in table 2.

The main objectives of the experiments have been to investigate the influence of the clustering approach on the accuracy when different number of clusters are chosen, and to analyze and compare the pruning strategies described in the previous section.

As regard the first objective, *ClustBoostCGPC* has been executed by fixing the number of clusters to 5 and 10, thus by using an ensemble of 500 an 1000 predictors, and compared with *BoostCGPC*, that uses an ensemble of 100 predictors. Table 3 shows the classification errors of *BoostCGPC* and *ClustBoostCGPC*. The table shows that for all the data sets the clustering strategy sensibly improves the accuracy of the method. For example, on the *Adult* data set *ClustBoostCGPC* (5 clusters) obtains an error of 14.749 instead of 17.231 of *BoostCGPC*. Thus the choice of the best five predictors in the clustered population instead of the best classification tree, at each boosting round, allows to obtain a much better result. However, as the table points out, augmenting the number of clusters is no more beneficial because the reduction of the misclassification error is minimal.

Using an ensemble of 500 predictors instead of 100 needs a larger amount of memory to store all the classifiers. Thus the improved accuracy is obtained at the cost of higher storage requirements. In the second set of experiments executed we show that the ensemble can be substantially pruned without decreasing performance. To this end we considered the ensemble of 500 predictors and we applied the pruning strategies described in the previous section.

Table 4 reports the results of the different pruning strategies for all the data sets. The percentages of pruning experimented are 10%, 20%, 50%, and 80% of the ensemble. The table reports in the column named *Err* the misclassification error of the ensemble pruned of the percentage showed in the corresponding row, in the column *GainC* the relative gain in percentage of the pruned ensemble with respect to the complete ensemble generated by *ClustBoostCGPC*, and in the column *GainB* the relative gain in percentage of the pruned ensemble with respect to the ensemble generated by *BoostCGPC*. A positive value means that the misclassification error is diminished, while a negative one that it increased. The table clearly shows that up to 50% of pruning, for all the data sets, independently the pruning strategy used, the ensemble can be reduced and still have

Table 4: Error and gain of pruned ClustBoostCGPC with respect to unpruned ClustBoostCGPC and unpruned BoostCGPC.

		origin			*pairwise*			*kappa*		
		Err	*GainC*	*GainB*	*Err*	*GainC*	*GainB*	*Err*	*GainC*	*GainB*
Adult	10%	14.711	0.26%	14.62%	14.462	1.95%	16.07%	14.734	0.10%	14.49%
	20%	14.971	-1.51%	13.12%	14.742	0.05%	14.44%	15.042	-1.99%	12.70%
	50%	15.425	-4.58%	10.48%	16.032	-8.70%	6.96%	15.823	-7.28%	8.17%
	80%	19.281	-30.73%	-11.90%	22.469	-52.34%	-30.40%	23.856	-61.74%	-38.45%
Census	10%	4.582	2.40%	25.13%	4.289	8.64%	29.92%	4.453	5.15%	27.24%
	20%	4.744	-1.04%	22.49%	4.706	-0.24%	23.10%	4.768	-1.56%	22.09%
	50%	5.073	-8.06%	17.11%	4.944	-5.31%	19.22%	5.085	-8.32%	16.91%
	80%	7.901	-68.30%	-29.10%	7.542	-60.65%	-23.24%	7.988	-70.15%	-30.52%
Covtype	10%	30.679	0.55%	8.08%	30.128	2.34%	9.73%	30.203	2.10%	9.50%
	20%	30.904	-0.18%	7.40%	30.355	1.60%	9.05%	30.800	0.16%	7.71%
	50%	32.322	-4.77%	3.15%	30.952	-0.33%	7.26%	31.012	-0.53%	7.08%
	80%	34.188	-10.82%	-2.44%	33.681	-9.18%	-0.92%	33.955	-10.06%	-1.74%
Mamm.	10%	1.297	0.95%	39.93%	1.223	6.60%	43.35%	1.265	3.39%	41.41%
	20%	1.385	-5.77%	35.85%	1.307	0.18%	39.46%	1.407	-7.45%	34.83%
	50%	1.467	-12.04%	32.05%	1.421	-8.52%	34.18%	1.604	-22.50%	25.71%
	80%	2.302	-75.81%	-6.62%	2.208	-68.63%	-2.27%	2.420	-84.82%	-12.09%
Phoneme	10%	16.893	0.44%	11.65%	16.528	2.59%	13.56%	16.349	3.65%	14.50%
	20%	17.185	-1.28%	10.13%	16.892	0.45%	11.66%	17.002	-0.20%	11.08%
	50%	18.042	-6.33%	5.64%	18.312	-7.92%	4.23%	18.207	-7.30%	4.78%
	80%	19.207	-13.20%	-0.45%	20.209	-19.10%	-5.69%	19.389	-14.27%	-1.40%
Satimage	10%	20.184	0.18%	5.67%	20.003	1.07%	6.52%	20.010	1.04%	6.49%
	20%	20.240	-0.10%	5.41%	20.081	0.69%	6.15%	20.120	0.49%	5.97%
	50%	20.512	-1.44%	4.14%	20.958	-3.65%	2.06%	20.601	-1.88%	3.72%
	80%	22.845	-12.98%	-6.76%	22.696	-12.25%	-6.07%	22.904	-13.27%	-7.04%
Segment	10%	12.027	0.82%	10.59%	11.906	1.82%	11.49%	12.004	1.01%	10.76%
	20%	12.209	-0.68%	9.24%	12.140	-0.11%	9.75%	12.233	-0.87%	9.06%
	50%	12.638	-4.21%	6.05%	12.643	-4.25%	6.01%	12.901	-6.38%	4.10%
	80%	16.467	-35.79%	-22.41%	15.842	-30.63%	-17.77%	15.068	-24.25%	-12.01%

an error lower than *BoostCGPC* (see column *GainB*). For example, the error obtained with the ensemble generated by *ClustBoostCGPC* on the *Census* data set pruned of 50% with the *pairwise* measure is 4.944, while that generated by *BoostCGPC* is 6.12, thus using the former approach gives a gain of 19.22%. Furthermore, it is worth to note that pruning improves the performance of *ClustBoostCGPC* if 10% of the classifiers are eliminated. Indeed the *pairwise* strategy, for almost all the data sets, allows to prune up to 20% of predictors and still decrease the misclassification error of *ClustBoostCGPC*. Thus it seems that the structural diversity used in Genetic programming gives better results than the behavioral diversity employed in the Machine Learning community. Finally figures 4 and 5 shows the overall performances, averaged for all the data sets, in terms of the relative gain. In particular, figure 4 displays the relative performance of each pruning strategy computed as the difference between the corresponding misclassification error and that obtained by *BoostCGPC* divided by the gain, that is the difference in percentage points between these errors. A value of 0 means that the pruned ensemble obtains the same performance as *BoostCGPC*, while a value greater than 0 that the pruned ensemble performs better than *BoostCGPC*. Figure 5 shows the same performance results compared with respect to *ClustBoostCGPC*. These two figures summarizes the results of table 4 and clearly point out that the pairwise strategy behaves better than the two others.

6. CONCLUSIONS

A distributed Boosting Cellular Genetic Programming Classifier, endowed with a clustering strategy to build the ensemble of predictors, and three pruning strategies to reduce the size of the ensemble, have been presented. The algorithm runs on a distributed environment based on a hybrid multi-island model. Experiments on several data sets indicate that ensemble accuracy can be increased at the cost of greater storage requirements. However, by applying pruning techniques similar or even better prediction accuracy can be achieved.

7. ACKNOWLEDGMENTS

This work has been partially supported by the research project MIUR-legge 297/1999.

8. REFERENCES

[1] A. Agresti. *Categorical Data Analysis.* John Wiley and Sons,Inc., 1990.

[2] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.

[3] R.E. Banfield, L.O. Hall, K.W. Bowyer, and W.P. Kegelmeyer. Ensembles diversity measures and their application to thinning. *Information Fusion*, 6:49–62, 2005.

[4] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, (36):105–139, 1999.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[6] Leo Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–824, 1998.

[7] Edmund Burke, Steven Gustafson, and Graham Kendall. A survey and analysis of diversity measures in genetic programming. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 716–723, New York, 2002. Morgan Kaufmann Publishers.

[8] Edmund Burke, Steven Gustafson, Graham Kendall, and Natalio Krasnogor. Advanced population diversity measures in genetic programming. In *Parallel Problem Solving from Nature - PPSN VII*, number 2439 in Lecture Notes in Computer Science, LNCS, page 341 ff., Granada, Spain, 2002. Springer-Verlag.

[9] E. Cantú-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transaction on Evolutionary Computation*, 7(1):54–68, February 2003.

[10] N. Chawla, T.E. Moore, W. Bowyer K, L.O. Hall, C. Springer, and P. Kegelmeyer. Bagging-like effects for decision trees and neural nets in protein secondary structure prediction. In *BIOKDD01: Workshop on Data mining in Bioinformatics (SIGKDD01)*, 2001.

[11] Thomas G. Dietterich. An experimental comparison of three methods for costructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, (40):139–157, 2000.

[12] R.C. Dubes and A.K. Jain. *Algorithms for Clustering Data*. MIT Press, Prentice Hall, 1988.

[13] Anikó Ekárt and Sandor Z. Németh. Maintaining the diversity of genetic programs. *Lecture Notes in Computer Science, EuroGP 2002*, 2278:162–171, 2002.

[14] G. Folino, C. Pizzuti, and G. Spezzano. A cellular genetic programming approach to classification. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO99*, pages 1015–1020, Orlando, Florida, July 1999. Morgan Kaufmann.

[15] G. Folino, C. Pizzuti, and G. Spezzano. Ensemble techniques for parallel genetic programming based classifiers. In E. Costa C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, editor, *Proceedings of the Sixth European Conference on Genetic Programming (EuroGP-2003)*, volume 2610 of *LNCS*, pages 59–69, Essex, UK, 2003. Springer Verlag.

[16] G. Folino, C. Pizzuti, and G. Spezzano. A scalable cellular implementation of parallel genetic programming. *IEEE Transaction on Evolutionary Computation*, 7(1):37–53, February 2003.

[17] G. Folino, C. Pizzuti, and G. Spezzano. Boosting technique for combining cellular gp classifiers. In M. Keijzer, U. O'Reilly, S.M: Lucas, E. Costa, and T. Soule, editors, *Proceedings of the Seventh European Conference on Genetic Programming (EuroGP-2004)*, volume 3003 of *LNCS*, pages 47–56, Coimbra, Portugal, 2004. Springer Verlag.

[18] Y. Freund and R. Scapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th Int. Conference on Machine Learning*, pages 148–156, 1996.

[19] Hitoshi Iba. Bagging, boosting, and bloating in genetic programming. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO99*, pages 1053–1060, Orlando, Florida, July 1999. Morgan Kaufmann.

[20] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[21] L.I. Kuncheva and C.J. Whitaker. Diversity measures in classifier ensembles. *Machine Learning*, (51):181–207, 2003.

[22] W.B. Langdon and B.F. Buxton. Genetic programming for combining classifiers. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO'2001*, pages 66–73, San Francisco, CA, July 2001. Morgan Kaufmann.

[23] D.D. Margineantu and T.G. Dietterich. Pruning adaptive boosting. In *Proceedings of the International Conference on Machine Learning*, pages 211–218, 1997.

[24] C.C. Pettey. Diffusion (cellular) models. In David B. Fogel Thomas Bäck and Zbigniew Michalewicz, editors, Handbook of Evolutionary Computation, pages C6.4:1–6. Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997.

[25] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[26] R. E. Schapire. Boosting a weak learning by maiority. *Information and Computation*, 121(2):256–285, 1996.

[27] Terence Soule. Voting teams: A cooperative approach to non-typical problems using genetic programming. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO99*, pages 916–922, Orlando, Florida, July 1999. Morgan Kaufmann.