Using Bayesian Networks for Selecting Classifiers in GP Ensembles

C. De Stefano[a], G. Folino[b], F. Fontanella[a,*], A. Scotto di Freca[a]

[a] *Dipartimento di Ingegneria Elettrica e dell'Informazione*
Università di Cassino e del Lazio meridionale
Via G. Di Biasio 43, 03043 Cassino (FR) – Italy
[b] *ICAR-CNR Istituto di Calcolo e Reti ad Alte Prestazioni*
Via P. Bucci, 87036 Rende (CS)–Italy

Abstract

Ensemble techniques have been widely used to improve classification performance also in the case of GP-based systems. These techniques should improve classification accuracy by using voting strategies to combine the responses of different classifiers. However, even reducing the number of classifiers composing the ensemble, by selecting only those appropriately "diverse" according to a given measure, gives no guarantee of obtaining significant improvements in both classification accuracy and generalization capacity. This paper presents a novel approach for combining GP-based ensembles by means of a Bayesian Network. The proposed system is able to learn and combine decision tree ensembles effectively by using two different strategies: in the first, decision tree ensembles are learned by means of a boosted GP algorithm; in the second, the responses of the ensemble are combined using a Bayesian network, which also implements a selection strategy to reduce the number of classifiers. Experiments on several data sets show that the approach obtains comparable or better accuracy with respect to other methods proposed in the literature, considerably reducing the number of classifiers used. In addition, a comparison with similar approaches, confirmed the goodness of our method and its superiority with respect to other selection techniques based on diversity.

1. Introduction

Ensemble techniques have been taken into account [2, 8, 25] in the last few years in order to further improve the performance of classification algorithms. They try to combine the responses provided by several experts effectively, in order to improve the overall classification accuracy [28]. Such techniques rely on: i) a diversification heuristic, used to extract sufficiently diverse classifiers; ii) a voting mechanism, to combine the responses provided by the learned classifiers. If the classifiers are sufficiently diverse, i.e. they make uncorrelated errors, then the majority vote rule tends to the Bayesian error as the number of classifiers increases [28].

Ensemble techniques have also been used to enhance the performance of classification systems in which decision trees are learned by means of Genetic Programming (GP). Examples of GP–based approaches using ensemble techniques can be found in [6, 14, 18, 24, 31]. Moreover, in [6], [14] and [24], ensembles of decision trees are evolved, and the diversity among the ensemble members is obtained by using bagging or boosting techniques. According to these approaches, an ensemble can be obtained by evolving each decision tree with reference to a different subset of the original data. Instead, in the the bagging approach [5], different subsets (called *bags*), the same size as the original training set, are obtained by applying a random sampling with replacement. Then the ensemble is built by training each classifier on a different *bag*. Finally, the responses provided by these classifiers are combined by means of the majority vote rule: an unknown sample is assigned the class label that has the highest occurrence among those provided by the whole set of classifiers. While, in the boosting approach [16], the classifier ensemble is trained by means of a stepwise procedure. At each step, a new classifier is trained by choosing the training set samples from the original training set, according to a suitable probability distribution. This distribution is adaptively changed at each step in such a way that samples misclassified in the previous steps have a better chance of being chosen. Eventually, classifier responses are combined by the weighted majority vote rule, where the weight associated with a classifier takes into account its overall accuracy on the training data.

In [13] a novel GP–based classification system, named Cellular GP for data Classification (CGPC), is presented. In the CGPC approach, individuals interact according to a cellular automata inspired model, whose goal is to enable a fine-grained parallel implementation of GP. In this model, each individual has a spatial location on a low-dimensional grid and interacts only with other individuals within a small neighborhood. In [14], an extension of CGPC, called *BoostCGPC*, is presented. It is based on two different ensemble techniques: the Breiman's bagging

*Corresponding author

algorithm [5] and the AdaBoost.M2 boosting algorithm [16]. Despite the significant performance improvements classifier ensembles can provide, their major drawback is that, usually, it is necessary to combine a large number of classifiers in order to obtain a marked error reduction. This implies large memory requirements and slow classification speeds. In fact, these aspects can be critical in some applications [32, 36], but this problem can be solved by selecting a fraction of the classifiers from the original ensemble. This reduction, often called "ensemble pruning" in the literature, has other potential benefits. In particular, an appropriate subset of complementary classifiers can perform better than the whole ensemble [32, 43, 44, 33, 4]. When the cardinality L of the whole ensemble is high, the problem of finding the optimal subset of classifiers becomes computationally intractable because of the resulting exponential growth of the search space, made of all the 2^L possible subsets. It is worth mentioning that several heuristic algorithms can be found in the literature for finding near optimal solutions. Examples of such heuristics are: Genetic algorithms (GAs) [43, 44] and semidefinite programming [42].

In order to be successful, any ensemble learning strategy should ensure that the classifiers making up the ensemble are suitably diverse, so as to avoid correlated errors [28]. In fact, as the ensemble size increases, it could happen that a correct classification provided by some classifiers is overturned by the convergence of other classifiers on the same wrong decision. If the ensemble classifiers are not sufficiently diverse, this event is much more likely and can reduce the obtainable performance, regardless of any combination strategy. Classifier diversity for bagging and boosting are experimentally investigated in [26, 27]. The results have shown that these techniques do not ensure obtaining sufficiently diverse classifiers. As regards boosting, in [26] it is observed that whereas highly diverse classifiers are obtained at the first steps, as the boosting process proceeds, classifier diversity strongly decreases.
More recently, AdaBoost has also been applied for generating more training subsets, used to learn an ensemble of Extreme Learning Machine (ELM) classifiers [40]. This approach tries to overcome some of the drawbacks in traditional gradient-based learning algorithm, and can also alleviate instability and over-fitting problems of ELM. The diversity issue has been also considered in the unsupervised learning of ensembles of clusters [41].

In a previous work [10], the classifier combination problem was reformulated as a pattern recognition one, in which the pattern is represented by the set of class labels provided by the classifiers when classifying a sample. Following this approach, a Bayesian network (BN) [35] was learned in order to estimate the conditional probability of each class, given the set of labels provided by the classifiers for each sample of a training set. Here, we have used Bayesian Networks because they provide a natural and compact way to encode joint probability distributions through graphical models, and allow probabilistic relationships among variables to be derived by using effective learning algorithms for both the graphical structure and its parameters. In particular, the joint probability among variables is modelled through the structure of a Direct Acyclic Graph (DAG), whose nodes are the variables while the arcs are their statistical dependencies. In this way, the conditional probability of each class, given the set of responses provided by the classifiers, can be directly derived by the DAG structure applying the Bayes Rule. Thus, the combining rule is automatically provided by the learned BN. Moreover, this approach makes it possible to identify redundant classifiers, i.e. classifiers whose outputs do not influence the output of the combiner: the behavior of these classifiers is very similar to that of other classifiers in the ensemble. For this reason, they may be discarded without affecting the overall performance of the combiner, thus overcoming the main drawback of the combining methods discussed above. In [11] the learning of the BN is performed by means of an evolutionary algorithm using a direct encoding scheme of the BN structure (DAG). This encoding scheme is based on a specifically devised data structure, called *Multilist*, which allows an easy and effective implementation of the genetic operators. Indeed, the rationale behind the choice of an evolutionary approach is that of trying to solve one of the main drawbacks of standard learning algorithms, such as the k2 one, which adopt a greedy search strategy and thus are prone to be trapped in local optima. On the contrary, evolutionary algorithms allow us to effectively explore complex high dimensional search space.

This paper presents a new classification system that merges the two aforementioned approaches. The goal is to build a high performance classification system, able to deal with large data sets but selecting only a reduced number of classifiers. For this purpose, we built a two–module system that combines the BoostCGPC algorithm [14], which produces a high performing ensemble of decision tree classifiers, with the BN-based approach to classifier combination [11]. In particular, the BN module evaluates classifiers diversity by estimating the statistical dependencies of the responses they provide. This ability is used to select the minimum number of classifiers, among those provided by the BoostCGPC module, required to effectively classify the data at hand. Moreover, the responses provided by the selected classifiers are effectively combined by means of a rule inferred by the BN module. In this way the proposed system exploits the advantages provided by both techniques and allows us to greatly reduce the number of classifiers in the ensemble. Finally, as regards the evolutionary learning of the BN, the mutation operator has been reformulated in order to ensure that any permutation on the elements of the main list, leaves unchanged the connection topology of the multi-list. This property is very important to manage the trade-off between the exploration and the exploitation ability of the evolutionary algorithm.

In order to assess the effectiveness of the proposed system, several experiments were performed. More specif-

ically, seven data sets, of different sizes, number of attributes and classes were considered. Two kinds of comparison were performed: in the former the results of our approach were compared with those achieved by using different combination strategies; in the latter, our results were compared with those obtained by combining subsets of classifiers selected according to different selection strategies. Moreover, a diversity analysis of the selected classifiers was carried out taking into account two diversity measures. A genotypic one, which compares the structures of the related decision trees, and a phenotypic one that considers the responses provided by the classifiers to be combined.

The remainder of the paper is organized as follows: Section 2 reviews some previous works done on ensemble pruning. Section 3 details the system architecture; in Section 4 several diversity measures, included those used in the experimental findings, are described; Section 5 illustrates the experimental results; finally Section 6 deals with discussion and some concluding remarks.

2. Related works

As mentioned in the introduction, the larger the number of classifiers included in the pool, the greater the memory requirements and the computational time needed for combining their results. For this reason the use of a large number of experts can be critical in some applications and it explains why in the last few years, many researchers focused their attention on ensemble pruning techniques. In [4], the initial ensemble is pruned by a sequential backward selection procedure that attempts to maximize the accuracy of the ensemble by eliminating those classifiers whose effect on the ensemble performance is either negligible or even damaging. In [44], a GA-based approach for pruning decision tree ensembles is presented. In this work, each individual represents an ensemble and uses a binary chromosome for encoding the presence or absence of a classifier in such an ensemble. The fitness function of each individual is computed by evaluating the performance that the corresponding ensemble obtains on a validation data set. In [23], this approach is used for pruning an ensemble obtained by using the AdaBoost algorithm.
Clustering techniques have been also used to prune ensembles. For example, in [19], the classifiers included in the ensemble were clustered by using the pairwise diversity as the distance measure. Only the classifier with the highest accuracy is selected, under the hypothesis that members of the same cluster tend to perform similarly for each cluster. More recently, Meynet and Thiran [34] proposed an information theory approach to assess the performance of an ensemble. This measure is employed for the selection of a subset of classifiers from the whole ensemble, initially obtained by using the AdaBoost algorithm.
Even mathematical optimization tools have been employed to solve the ensemble pruning problem. In [42], ensemble pruning is formulated as a semidefinite programming problem defined in terms of a matrix G, whose element G_{ij} represents the number of common errors between the i–th and j–th classifiers. Note that in this approach the size of the pruned ensemble needs to be fixed a-priori. In [39], Tsoumakas et al. presented an ensemble pruning approach called "selective fusion", which combines the outputs of the selected classifiers using a weighted voting strategy. The selection of the optimal classifier subset is approached as a multiple comparisons problem, which is solved by applying statistical tests to detect significant differences in cross-validation based estimates of the prediction errors.

Also in the Evolutionary Computation community, researchers have used different approaches to increase diversity among ensemble classifiers. In [18] and [31], for example, diversity among classifiers is pursued by means of a technique denoted as negative correlation learning [30]. This learning is based on a fitness function that scores single individuals by taking into account the responses given by the other individuals in the population. In such a way, during the evolution process, individuals are driven to learn different parts of the training data. Note that both approaches also use a selection strategy in order to reduce the size of the ensemble. In [31] the individuals in the population are clustered using the *k–means* algorithm. Then the best individual of each cluster is chosen to make up the final ensemble. In [18] the selection strategy is accomplished by using a greedy heuristic that chooses the most diverse classifiers, among those available in the ensemble. Finally, both approaches use the majority vote rule to combine the responses provided by the ensemble classifiers.

3. The Architecture of the System

The proposed system aims to classify large datasets exploiting the advantages of both an ensemble based GP classifier, and a Bayesian Network-based combining technique. In practice, the system comprises two main modules: the first one (denoted as ensemble module) is composed of an ensemble of decision tree classifiers (experts) built by means of the BoostCGPC algorithm. Whereas the second one (denoted as combining module) uses a Bayesian Network-(BN)-based combiner to effectively combine the responses of a subset of classifiers, suitably chosen from the whole ensemble provided by the first module.

The system architecture is modular. Thus, each module can possibly be substituted by a different one performing the same task. Therefore, for instance, the first module could be replaced by any other system able to produce an ensemble of classifiers.

In our case, the choice of these modules is due to their characteristics, as it has been discussed in the introduction. In particular, the BoostCGPC algorithm is able to deal with large datasets, and obtains great accuracy by using only a small subset of the training data, at a much lower computational cost [14]. Moreover, in [15], a comparison with the state-of-the-art classification algorithms,

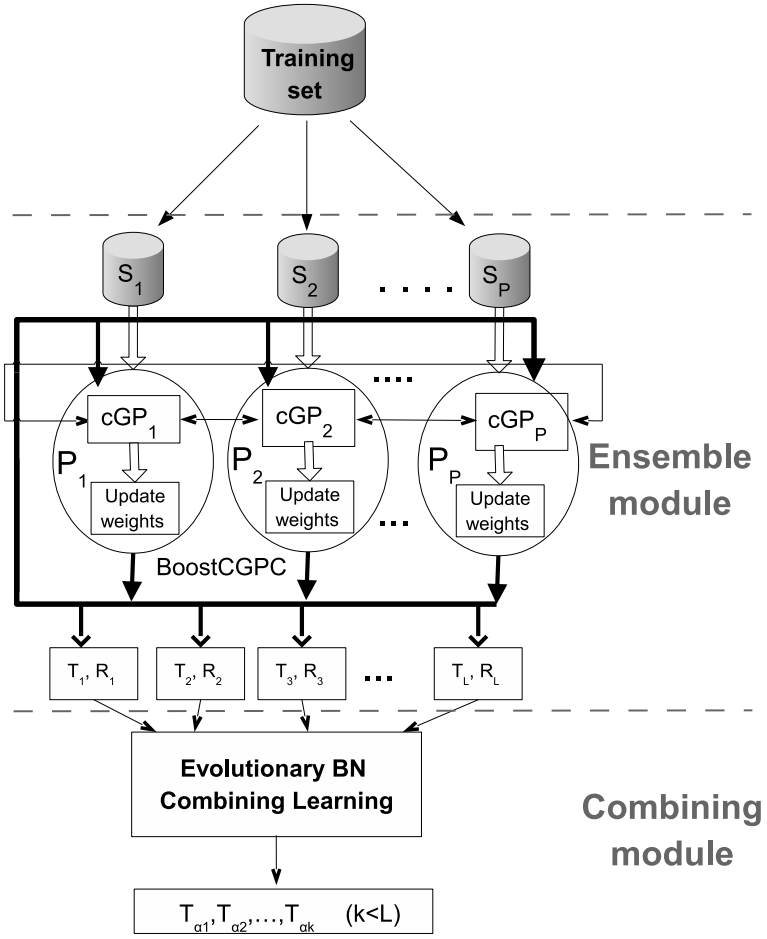


Figure 1: The learning of the system.

such as C4.5 and SVM, demonstrated that the BoostCGPC algorithm improves both the prediction accuracy and the execution time. Finally, another advantage of the GP–based ensemble module is that it can run on parallel/distributed architectures, permitting time-saving in the most expensive phase of the training process. Actually, the classifier typically runs on p processors/nodes, named $P_1, P_2, \ldots, P_p$, with one instance of the Genetic Programming (GP) algorithm for each processor/node. The dataset is also partitioned in p parts. Note that p can be chosen differently if it is more convenient to place more processes on a single node of the parallel/distributed architecture. The first module uses an algorithm based on the boosting technique and is described more in detail in Subsection 3.1.

The learning procedure of the whole system, shown in Fig. 1 consists of two distinct phases. In the first phase, the ensemble module produces an ensemble of L classifiers $T_1, T_2, \ldots, T_L$ trained by using a training set of labeled samples described by their feature values. At the end of this phase, the set of responses $R = \{R_1, R_2, \ldots, R_L\}$ is also produced, where each R_i contains the responses provided by the $i-th$ classifier for all the training set samples. The second phase concerns the learning of the combining module, which is performed by using a specifically devised evolutionary algorithm. The effect of this learning is twofold. On the one hand, the learned BN effects a combination rule: given the set of responses provided by the ensemble for an input sample, the probabilities of assigning these samples to each of the class to be discriminated are computed, and the class with the highest probability is chosen. On the other hand the BN, modeling the statistical dependencies among the responses provided by the classifiers and the actual class of the input samples, also allows us to adopt a classifier selection strategy (see Section 3.2). The final result of the learning procedure just described is a set of classifiers $T_{\alpha 1}, T_{\alpha 2}, \ldots, T_{\alpha k}$ classifiers with $k << L$ and $1 \leq \alpha_i \leq L$.

Once the whole system has been learned, the classification of an unknown samples is performed by using a two–step procedure (see Fig. 2): (i) feature values describing the unknown sample are provided to each of the classifiers selected during the learning procedure; (ii) then, their responses are given in input to the combining module, which assigns the sample to the most likely class[1].

In the following sub-sections, we will illustrate the Boost-

[1]Note that the second step does not require any further computa-

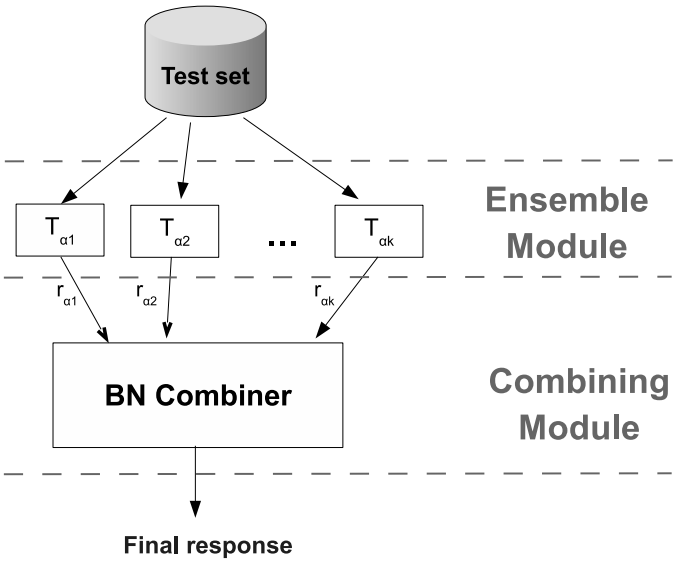


Figure 2: The testing phase of the system.

CGPC algorithm for producing the ensemble of decision tree classifiers, as well as the BN approach for combining their responses, and the evolutionary algorithm implemented for learning the structure of the BN.

3.1. The Ensemble Module

The *Boost Cellular Genetic Programming Classifier* [14] algorithm builds GP ensembles using a hybrid variation of the classical distributed island model of GP. GP ensembles offer several advantages over a monolithic GP, i.e. the possibility of coping with very large data sets, more simple and understandable models, robustness and obviously the advantages related to a distributed implementation.

BoostCGPC adopts the AdaBoost.M2 version of the well-known boosting algorithm introduced by Schapire and Freund for "boosting" the performance of any weak learner, i.e. an algorithm that "generates classifiers which need only be a little better than random guessing". Therefore, in this section, first the original boosting algorithm is described, then BoostCGPC and the distributed variant of boosting used are illustrated.

The boosting algorithm 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 re-normalizing the weights so that $\Sigma_i w_i^{t+1} = 1$. In this way, it focuses on examples that are hardest to classify, as "easy" examples get a lower weight, while "hard" examples, that tend to be misclassified, get higher weights. 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 $log(1/\beta^t)$. The final classifier h_f is defined as follows:

$$h_f = arg\ max\ \left(\sum_t^T log(\frac{1}{\beta^t}) h_t(x, y) \right) \quad (1)$$

In practice, BoostCGPC is an ensemble-based algorithm, in which each GP classifier forming the ensemble is built using a cellular GP algorithm (cGP), enhanced with the boosting technique, which runs on each node (see Fig. 1). cGP runs for T rounds; for every round it generates a classifier per node, exchanges it with the other nodes, and updates the weights of the samples for the next round, according to the boosting algorithm. In fact, the selection rule, the replacement rule and the asynchronous migration strategy are specified in the cGP algorithm. Each node generates the GP classifier by running for a fixed number of generations. During the boosting rounds, each classifier maintains the local vector of the weights that directly reflect the prediction accuracy. At each boosting round the hypotheses generated by each classifier are exchanged among all the processors in order to produce the ensemble of predictors. In this way each node maintains the entire ensemble and it can use it to recalculate the new vector of weights. After the execution of the fixed number of boosting rounds, the classifiers are updated.

The distributed variant of AdaBoost.M2 used in BoostCGPC is illustrated in the pseudocode reported in Fig. 3

tion with respect to the Majority Voting rule. In fact, it only needs to read tables storing class probabilities.

and described shortly in the following (we refer the reader to [14] for a detailed description of the algorithm).

Given the training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, containing N samples, where $\mathbf{x}_i$ represents the feature vector of the i-th sample and y_i the corresponding label, and given the number p of processors to be used to run the algorithm, we partition the population of classifiers in p subpopulations, one for each processor, and draw p sets of samples of size $n < N$, by uniformly sampling instances from S with replacement. Each subpopulation is evolved for g generations and trained on its local sample by running *cGP*. After g generations, the individual with the best fitness is selected for participating to vote. In fact the p individuals, one for each subpopulation, with the best fitness are exchanged among the p subpopulations and constitute the ensemble of predictors that will determine the weights of the examples for the next round. After the execution of the fixed number T of boosting rounds, the overall classifiers composing the ensemble (their number is $p \cdot T$), collected during the different rounds, are used to evaluate the accuracy of the classification algorithm.

3.2. The Combining Module

As mentioned in the introduction, the problem of combining the responses provided by a set of classifiers can be handled by estimating the conditional probability of each class given the set of labels provided by the classifiers. This problem may be effectively solved by using a Bayesian Network (BN). A BN is a probabilistic graphical model that allows the representation of a joint probability distribution of a set of random variables through a Direct Acyclic Graph (DAG) [35]. The nodes of the graph correspond to variables, while the arcs characterize the statistical dependencies among them. An arrow from node i to node j has the meaning that j is conditionally dependent on i, and we can refer to i as a *parent* of j. Therefore, in a BN, the i–th node e_i is associated with a conditional probability function $p(e_i|pa_{e_i})$, where pa_{e_i} indicates the set of nodes which are parents of e_i. Such function quantifies the effect that the parents have on that node.
Once the statistical dependencies among variables have been estimated and encoded in the DAG structure, the joint probability of the set of variables to be represented variables $E = \{e_1, \ldots, e_L\}$ can be described as:

$$p\,(e_1, \ldots, e_L) = \prod_{e_i \in E} p(e_i|pa_{e_i}) \tag{2}$$

In the classifier ensemble framework, this property can be used to infer the true class c of an unknown sample when the responses of the ensemble classifiers are known. In fact, suppose the ensemble consists of L classifiers, then the true class c and the L classifier responses can be modeled as a set of $(L+1)$ variables $\{c, e_1, \ldots, e_L\}$, and the Eq. (2) allows the description of their joint probability as:

$$p\,(c, e_1, \ldots, e_L) = p\,(\,c\,|\,pa_c\,) \prod_{e_i \in E} p\,(\,e_i\,|\,pa_{e_i}\,) \tag{3}$$

The node c may be the parent of one or more of the nodes of the DAG. Therefore, it may be useful to divide the set of DAG nodes that are not parent of c into two groups: the first, denoted E_c, contains the nodes having node c among their parents, and the second, denoted $E_{\overline{c}}$, the remaining ones. With this assumption, Eq. (3) can be rewritten as:

$$p(c, e_1, \ldots, e_L) = p(c|pa_c) \prod_{e_i \in E_c} p(e_i|pa_{e_i}) \prod_{e_i \in E_{\overline{c}}} p(e_i|pa_{e_i}) \tag{4}$$

This property allows a BN to recognize a given sample considering only the responses provided by the classifiers included in the Markov blanket of the class node c. In fact, given the set of responses concerning a sample, the BN can be used to estimate the conditional probability $p(c|e_1, \ldots, e_L)$. As a consequence, the BN module is able to evaluate the most probable class $\widehat{c}$ of an unknown input sample, given the responses provided by the first module classifiers, as follows:

$$\widehat{c} = \arg\max_{c \in C} p\,(c|e_1, ..., e_L) \tag{5}$$

where C is the set of classes. Considering the definition of conditional probability the above equation can be rewritten as follows:

$$\widehat{c} = \arg\max_{c \in C} \frac{p\,(c, e_1, ..., e_L)}{p\,(e_1, ..., e_L)} \tag{6}$$

omitting the terms not depending on the variable c, the above equation can be rearranged as:

$$\widehat{c} = \arg\max_{c \in C} p\,(c, e_1, ..., e_L). \tag{7}$$

which involves only the joint probabilities $p\,(c, e_1, ..., e_L)$. According to Eq. (4), and discarding the term that does not depend on c, Eq. (7) assumes the form:

$$\widehat{c} = \arg\max_{c \in C} p(\,c,\, e_1,\, \ldots,\, e_L\,) = \\ = \arg\max_{c \in C} p\,(\,c\,|\,pa_c\,) \prod_{e_i \in E_c} p\,(\,e_i\,|\,pa_{e_i}\,) \tag{8}$$

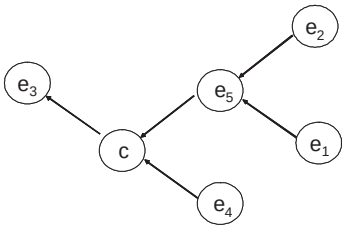


Figure 4: An example of a BN. The sets $\mathbf{pa_{e_5}} = \{\mathbf{e_1}, \mathbf{e_2}\}$, $\mathbf{pa_c} = \{\mathbf{e_4}, \mathbf{e_5}\}$ and $\mathbf{pa_{e_3}} = \{\mathbf{c}\}$ respectively represents the parent sets of the nodes e_5, c and e_3. The DAG structure induces the factorization of the joint probability $\mathbf{p(c, e_1, e_2, e_3, e_4, e_5)} = \mathbf{p(e_3|c)p(c|e_4, e_5)p(e_5|e_1, e_2)p(e_1)p(e_2)p(e_4)}$. Note that in this case $\mathbf{E_c} = \{\mathbf{e_3}\}$ and $\mathbf{E_{\overline{c}}} = \{\mathbf{e_4}, \mathbf{e_5}\}$.

Given $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, $x_i \in X$
with labels $y_i \in Y = \{1, 2, \ldots, K\}$, and a population Q of size q
Let $B = \{(i, y), i \in \{1, 2, \ldots, K\}, y \neq y_i\}$
For j = 1, 2, ..., p (for each processor in parallel)
Draw a sample S_j with size n for processor j
Initialize the weights $w^1_{i,y} = \frac{1}{|B|}$ for $i = 1, \ldots, n, y \in Y$,
where n is the number of training examples on each processor j.
Initialize the subpopulation Q_i, for $i = 1, \ldots, p$
with random individuals
end parallel for
For t = 1,2,3, ..., T (for each round)
For j = 1, 2, ..., p (for each processor in parallel)
Train *cGP* on the sample S_j using a weighted
fitness according to the distribution w^t
Compute a weak hypothesis $h_{j,t} : X \times Y \to [0, 1]$
Exchange the hypotheses $h_{j,t}$ among the p processors
Compute the error $\epsilon^t_j = \frac{1}{2} \sum_{(i,y) in B} w^t_{i,y} \cdot (1 - h_{j,t}(x_i, y_i) + h_{j,t}(x_i, y))$
if $\epsilon^t_j \geq 1/2$ **break loop**
Set $\beta^t_j = \epsilon^t_j / (1 - \epsilon^t_j)$,
Update the weights $w^t : w^{t+1}_{i,y} = \frac{w^t_{i,y}}{Z_t} \cdot \beta^{(\frac{1}{2}) \cdot (1 + h_{j,t}(x_i, y_i) - h_{j,t}(x_i, y))}$
where Z_t is a normalization constant $w^t_{i,y}$ will be a distribution)
end parallel for
end for t
output the hypothesis :
$h_f = arg\ max\ (\sum_j^p \sum_t^T log(\frac{1}{\beta^t_j}) h_{j,t}(x, y))$

Figure 3: The algorithm parallel *BoostCGPC* version AdaBoost.M2

An example of this rule is shown in Fig. 4. In this case the above equation becomes $\widehat{c} = \arg\max_{c \in C} p(e_3|c)p(c|e_4, e_5)$. Then the BN considers only the responses of the experts e_3, e_4 and e_5, while the experts e_1 and e_2 are not taken into account. Thus, this BN–based approach to combining allows a reduced set of relevant experts to be detected, namely the ones included in the Markov blanket of node c, whose responses are actually used by the combiner to provide the final output, while the other experts, which does not add information to the choice of $\widehat{c}$, are discarded.

3.3. The Evolutionary Learning of the BN

Using a BN for combining classifier responses requires that both the network structure and the parameters of the probability distribution, be learned from a training set of examples. The aim of structural learning is to find the statistical dependencies relations among the variables (graph nodes). It can be seen as an optimization problem, which requires the definition of a search strategy in the graph space to maximize a scoring function evaluating the effectiveness of candidate solutions. A typical scoring function is the posterior probability of the structure given the training data. Let us assume that a set of N labeled samples is available. The training set used for structural learning is obtained by considering, for each of the N samples, both the labels provided by the combining classifiers and the "true" label. Denoting with S^h the DAG of a candidate BN and with D the training set having N elements, the scoring function to be maximized is the likelihood of D given the structure S^h: $score(S^h) = p(D|S^h)$. According to the chain rule property of random variables in a DAG, this score can be factorized in the following way[2]:

$$score(S^h) = \prod_{i=1}^{L+1} localscore(i) \tag{9}$$

where the $localscore(i)$ measures how much the variable e_i is statistically dependent on the set of its parent nodes pa_{e_i}, according to the examples of the training set D [21]. It is worth noticing that any change in S^h requires that only the local scores of the nodes affected by that change need to be updated for computing the new $Score(S^h)$. The function $localscore(i)$ is computed as follows:

$$localscore(i) = \prod_{m=1}^{S_{pa_{e_i}}} \frac{\Gamma(\alpha_{im})}{\Gamma(\alpha_{im} + N_{im})}$$

[2]Note that if L is the number of considered classifiers, the DAG to be searched for consists of $L + 1$ nodes, where the $(L + 1)^{th}$ node represents the true class c.

$$\prod_{k=1}^{K} \frac{\Gamma(\alpha_{imk} + N_{imk})}{\Gamma(\alpha_{imk})}. \quad (10)$$

where $\Gamma(.)$ is the Euler Gamma function and $S_{pa_{e_i}}$ is the total number of states of pa_{e_i}. Considering that in our case each expert has K possible states, each corresponding to one of the classes in C, if the expert e_i has q parents, then $S_{pa_{e_i}} = K^q$. This implies that, for each response provided by the expert e_i, a vector of q terms representing the answers of the parent nodes of e_i must be analyzed. The term N_{imk} represents how many times pa_{e_i} is in the state m and the expert e_i is in the state k. The term N_{im}, instead, represents how many times pa_{e_i} is in the state m independently from the state of the expert e_i. The terms α_{im} and α_{imk} are normalization factors, whose values were determined following the uninformative assignment suggested by Buntine [22]:

$$\alpha_{im} = N'/S_{pa_{e_i}} \qquad \alpha_{imk} = N'/(K \cdot S_{pa_{e_i}})$$

where N' represents the equivalent sample size. In our case, we have experimentally set $N' = 2\,K$. With these assumptions, the considered scoring function is called BDeu metric.

Once the DAG structure S_h has been determined, the parameters of the conditional probability distributions are computed from D.

As regards the computational complexity of the function $localscore(i)$, its evaluation requires to compute the number of occurrences in D of each possible state of e_i associated to each possible state of pa_{e_i}. Thus, denoting with q the maximum number of parents that an expert may have, the computational complexity of $localscore(i)$ can be expressed as $\mathcal{O}(NKS_{pa_{e_i}}) = \mathcal{O}(NK^{q+1})$. In order to limit the exponential growth of the complexity of this function score, we experimentally set to three the maximum number of parents that a node may have.
Under these assumptions, the computational complexity of the function $score(S^h)$ (Eq. 9) is, in the worst case, $\mathcal{O}(LNK^4)$.

The exhaustive search for the BN structure which maximizes the scoring function is a NP-hard problem. For this reason, greedy algorithms are used to search for suboptimal solutions by maximizing a local scoring function at each step, which takes into account only the local topology of the DAG. To overcome this problem, we use an alternative approach in which the structure of the BN is learned by means of an Evolutionary algorithm, using a direct encoding scheme. In fact, the algorithm is based on a specifically devised data structure for encoding DAG, called *multilist* (ML), which consists of two basic lists. The first one, called *main list*, contains all the nodes of the DAG arranged in such a way that source nodes occupy the first positions, and sink nodes, the last ones. Moreover, nodes with both incoming and outgoing arcs are inserted in the *main list* after their parents. A second list called *sublist* is associated with each node of the *main list*, representing the outgoing connections among that node and

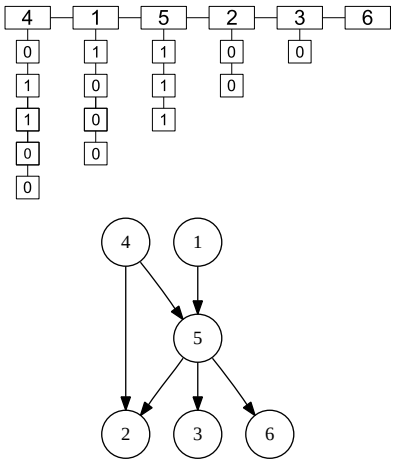


Figure 5: A multilist (top) and the encoded DAG (bottom).
sottomissione

the other nodes in the DAG. More specifically, if s_i is the *sublist* associated with the $i-th$ element of the *main list*, then it contains information about the outgoing arcs possibly connecting the $i-th$ element and the other elements following it in the *main list*, ordered according to the position of such elements. Since an arc may be present or not, each element of a *sublist* contains a binary information: 1 if the arc exists, 0 otherwise (see figure 5). Note that the above definition ensures that an ML intrinsically represents a DAG structure, then the application of the genetic operators always produces valid offspring. This property makes the proposed evolutionary BN learning more efficient, because it allows our evolutionary algorithm to avoid the time consuming task of checking the acyclicity property every time a genetic operator is applied.

Two mutation operators were defined. They can modify an ML in two different ways: the m mutation changes an ML by swapping two elements of the main list, whereas the s mutation adds and/or deletes one or more arcs in a sub list.
The m–mutation performs a permutation on the elements of the main list, but leaves unchanged the connection topology of the ML. This mutation consists of two steps:

(i) randomly pick two elements in the main list and swap their positions.
(ii) modify sublist elements in such a way to restore the connection topology as it was before the step (i).

It is worth noticing that the m–mutation generates a new ordering of the variables, which modifies the directions of the existing arcs in the DAG, but preserves dependencies between variables. If we consider the DAG in figure 5, for instance, the swap between the second and the fourth node in the main list changes only the directions of the arcs connecting the couples of nodes $(1, 5)$ and $(5, 2)$. This operator is applied according to a predefined probability value p_m.
The s–mutation, instead, modifies the values of the *sublist*

elements. For each element of the sublists, p_s represents the probability of changing its value from 0 to 1, or vice versa. Thus the effect of this operator is that of adding or deleting arcs in the DAG. This operation is applied with probability p_s. The values of the evolutionary parameters p_m and p_s were determined by performing a set of preliminary experiments. As regards the probability p_m, the best values are in the range $\{0.8, \ldots, 1.0\}$, while the value chosen for the probability p_s is equal to $1/N_s$, where N_s is total number of elements in the sublists of an individual. This value depends on the number L of nodes in the DAG to be learned, since N_s is equal to $L(L-1)/2$. Thus this probability value is such that, on the average, only one sublist element is modified when it is applied. It is worth noticing that variations in the value of p_m slightly influence the obtainable results. On the contrary, higher values for p_s significantly reduce the performance of the system. Further details about ML data structure and the genetic operators can be found in [11].

The evolutionary algorithm for the BN learning starts by randomly generating an initial population of P individuals (multilists). Afterwards, the fitness of each individual is evaluated by computing the scoring function (Eq. 9). At each generation, the best e individuals are selected and copied in the new population in order to achieve an elitist strategy. Then, the tournament is used to select $(P-e)$ individuals and the m and s mutation operators are applied to each selected individual according to the probabilities p_m and p_s, respectively. Finally, these individuals are added to the new population. This process is repeated for n_g generations.
Note that the algorithm is able to find good solutions in terms of both node ordering (by using the m-mutation), and connection topology (by using the s-mutation). In fact, a different node ordering may be obtained by applying the m-mutation, which modifies the father-child relationship between couples of nodes, while the statistical dependencies among classifiers may be varied by applying the (s-mutation), which modifies the connection topology. The computational complexity of the proposed evolutionary learning algorithm depends on five parameters: the population size P, the number of generations n_g, the number L of involved classifiers, the number K of classes and the number N of elements making up the training set D. The first two terms, P and n_g, affect the total number of fitness evaluations (equal to Pn_g). The computational complexity of both the mutation operators (m and s mutation) depends on L and, in the worst case, is equal to $\mathcal{O}(L^2)$. Finally, we have used as fitness the scoring function (Eq. 9), whose computational complexity is, in the worst case, $\mathcal{O}(LNK^4)$, as discussed above.

4. Diversity Measures and Correlation

Performing a diversity analysis and exploring the correlation with the accuracy of an ensemble would constitute a good step forward for trying to understand the performance of our system. In addition, other motivations also require this kind of analysis. In fact, an important property of an ensemble-based classifiers is the ability to achieve good values of accuracy by using only a very limited number of classifiers. However, the approach described in this paper needs a considerable overhead to select the trees composing the final ensemble. Actually, if we verified a strong correlation among diversity and accuracy, an alternative approach to the trees selection could be based on diversity measures, i.e. fixing a threshold of diversity and choosing all the classifiers under this threshold or removing the most similar ones. To this aim, in this paper two metrics were adopted in order to analyze the relation between the diversity of the selected trees and the classification error. The first metric considers the genotypic diversity in the ensemble, whereas the second takes into account the phenotypic diversity.

The genotypic measure of diversity we adopted was first introduced in [12] to evaluate the structural diversity between two trees. Given two trees T_a and T_b with roots R_a and R_b, the adopted distance is defined as:

$$dist(T_a, T_b) = d(R_a, R_b) + \frac{1}{H}\sum_{i=1}^{m} dist(child_i(R_a), child_i(R_b))$$

where: $d(R_a, R_b) = (|c(R_a) - c(R_b)|)^2$, 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. Since the nodes of a classification tree are labeled 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 labeled with that attribute. In practice, the two trees T_a and T_b are overlapped at the root node and, recursively, for each pair of nodes at matching positions, the difference of their codes is computed and combined in a weighted sum.

We used a disagreement measure as phenotypic diversity measure, based on the *kappa statistics*, introduced in the previous subsection, and defined as follows. Given a dataset D containing N samples and two classifiers T_a and T_b such that $\mathcal{X} \to \mathcal{C}$, where $\mathcal{X}$ is the feature space and C the set of classes, it is possible to build a contingency matrix M. The element M_{ij} of M contains the number of samples for which $T_a(x) = i$ and $T_b(x) = j$, where $x \in D$ and $i, j \in C$.

Let

$$\Theta_1 = \frac{\sum_{i=1}^{|\mathcal{Y}|} M_{i,i}}{N}$$

be the probability that two classifiers agree, where N is the size of the training set and $\mid \mathcal{Y} \mid$ is the number of different classes. Let also

$$\Theta_2 = \sum_{i=1}^{|\mathcal{Y}|} \left(\frac{\sum_{j=1}^{|\mathcal{Y}|} M_{i,j}}{N} \frac{\sum_{j=1}^{|\mathcal{Y}|} M_{j,i}}{N}\right)$$

be the probability that two classifiers agree by chance, given the observed counts in the table. Then the κ measure of disagreement between classifiers T_a and Y_b is defined as

$$\kappa(T_a, T_b) = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}$$

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.

5. Experimental Analysis

The proposed approach was tested on seven real data sets: *Adult*, *Census*, *Covtype*, *Phoneme*, *PhotoObject*, *Satimage*, and *Segment*. The size and class distribution of these data sets are described in Table 1. They present different characteristics as regards the number and type (continuous and discrete) of attributes, the number of classes (two classes and multiple classes problems) and the number of samples. In particular, Adult and Census contain census data collected by the U.S. Census Bureau. CovType is a large data set comprising data representing forest cover type from cartographic variables, while Phoneme data set contains data distinguishing between nasal and oral vowels. Satimage data set was generated from Landsat multispectral scanner image data. As for Segment dataset, it contains data representing pixels randomly extracted from hand-segmented images. Finally, PhotoObject is a very large dataset containing astronomical data collected from optical images [1].

All the experiments were performed on a Linux cluster with 16 Itanium2 1.4GHz nodes, each having 2 GBytes of main memory and connected by a Myrinet high performance network. In each experiment, the BoostGCPC module uses standard GP parameters (prob. of crossover= 0.8, prob. of mutation=0.1, maximum depth=17, no parsimony factor) and a population of 100 individuals per node. The original training set was partitioned among 5 nodes and 10 rounds of boosting, with 100 generations for round, were performed to produce an ensemble of 50 classifiers. It is worth remembering that the algorithm produces a different classifier for each round on each node.

The above process was repeated 30 times, so as to obtain 30 ensembles (each one including 50 classifiers) for each experiment. In this way, all results were averaged over 30 runs. In each run, the BN learning was carried out by using the responses, on the whole training set, provided by the classifier ensemble in the BoostCGPC module. For experimental purposes, three BN combiners were generated, namely BN10, BN20 and BN50, by considering the responses of the first 10 classifiers, of the first 20 classifiers and of the whole set of 50 classifiers in the ensemble, respectively. The results on the test set were obtained by first submitting each sample to the classifier ensemble. Then, the responses were provided to the corresponding BN module, which produced the final output label.

In order to assess the goodness of the proposed system, using the parameters, the methodology and the datasets described above, we conducted a number of experiments, described in detail in the next subsections. First, we compared our approach with different state-of-the-art combination strategies. Then, in order to verify whether there is a correlation between the diversity of the experts and the accuracy of the results provided by the whole system, a diversity analysis of the selected classifiers was carried out. Finally, Evo-BN was compared with other selection strategies, to test the effectiveness of the ensemble pruning strategy.

5.1. Accuracy Analysis

The results achieved by our approach (hereafter Evo-BN) have been compared with those obtained by the techniques detailed below:

- Weighted majority vote rule (WMjV), which is the one used in the BoostCGPC approach for combining the ensemble responses (see equation 1).
- Majority vote rule (MjV). Given a set of responses related to an unknown sample, this rule counts the responses for each class and assigns the sample to the most voted class. If a tie occurs, the sample is rejected.
- C4.5 algorithm (C4.5). As mentioned in the introduction, we reformulated the classifier combination problem as a pattern recognition one. For this reason, we also compared our results with those obtained by a classifier system. We employed the C4.5 algorithm as classifier system, with the Gini index as quality measure for split calculation [37, 3]. We used the implementation provided by the WEKA tool [20].
  Decision trees were learned using the responses of the classifiers supplied by the first module of our system for each considered dataset, on the training data of that dataset. The learned trees were tested providing them the responses given by the just mentioned classifiers on the test set.
- Bayesian Networks (BN). In order to test the effectiveness of the evolutionary learning performed by the second module of the proposed system, we compared our results with those obtained by using three standard algorithms for learning Bayesian Networks, namely K2, PC and TAN. In all the experiments, the implementations of these algorithms provided by the WEKA tool were used. A brief description of K2, PC and TAN algorithms is reported in the following.
  - K2 (K2-BN). This algorithm uses a hill climbing technique to learn Bayesian Networks from data [9]. Since the results provided by the K2

Table 1: The data sets used in the experiments

datasets	#attr.	samples	classes	attr. types
Adult	14	48,842	2	categorical, continuous
Census	4	299,285	2	categorical, continuous
Covtype	54	581,012	7	integer, continuous
Phoneme	5	5,404	2	categorical, continuous
PhotoObject	7	750,000	2	continuous
Satimage	36	6,435	7	integer
Segment	19	2,310	6	continuous

algorithm depend on the assumed node ordering, for each of the thirty performed runs a different random ordering has been generated.

- PC algorithm (PC-BN). This algorithm starts with the fully connected undirected graph [38]. Then for every couple of variables (graph nodes) a conditional independence test is performed. Afterwards, the arcs connecting nodes which, according to the tests performed, are statistically independent, are removed. Finally, the DAG is obtained by orienting the remaining arcs in a way that does not produce any cycle.
- Tree Augmented Naive Bayesian Network (TAN-BN). This approach extends the naive Bayes technique. In practice, in the learned BN the class variable (the true class variable c in our case) has no parents and each attribute node (a classifier node e_i in our case) has as parent the class variable. Moreover, each attribute node is also connected to another (just one) attribute node [17]. The search in this reduced BN structure space is performed by finding a maximal weighted spanning tree in a graph, where the weight of each arc is represented by the value of mutual information of the variables connected by that arc [7].

Note that we adopted the same scoring function used by our system for evaluating the quality of the network structure. To perform a fair comparison, for each considered algorithm and for each dataset, we adopted the following procedure: BNs were learned on the responses provided by the set of classifiers supplied by the first module of our system on the training set; then the learned BNs were tested on the responses obtained on the test set by the just mentioned classifiers.

Comparison results are shown in Tables 2 and 3. In particular, Table 2 shows the results of our method and those of Weighted Majority vote, Majority vote and C4.5, while Table 3 reports the performance of the proposed system and those of the three BN learning algorithms previously mentioned (K2, PC and TAN).

In both tables, the second column shows the number of classifiers taken into account by the BN combiners (10, 20 and 50, respectively), while the error rate on the test set, the time (in seconds) needed to find solutions and the number of classifiers actually used, are reported for each comparing method. It is worth noticing that time and number of actually used classifiers have not been reported for Weighted Majority vote and Majority vote methods (see Table 2). In fact, for both these methods, the number of used classifiers coincides with the number of classifiers making up the ensemble (10, 20 or 50), while the time is not reported since they do not require any learning. In order to statistically validate the comparison results, we performed the non-parametric Wilcoxon rank-sum test [29] ($\alpha = 0.05$) over 30 runs. The values in bold in the error columns highlight for each dataset the results, which are significantly better with respect to the second best results (values starred in the table), according to the Wilcoxon test. As concerns the results which do not present a statistically significant difference, the best two results are both starred.

From Table 2 it can be seen that the proposed approach achieves better performance for twelve out of the twenty-one tested ensembles, while for the remaining ones the differences are not statistically significant. Furthermore, if we consider the best performance for each dataset, our system achieves the best performance for five out of the seven datasets. For instance, for the Covtype dataset, our system obtains an error rate of 31% using only, on average, 4.1 classifiers. The second best result for this dataset was achieved by the weighted majority vote rule, exhibiting an error rate of 32.51%, but using the responses provided by the whole set of 50 classifiers. A similar behavior can be also observed for the remaining datasets on which our system obtained better results (Adult, Census, Phoneme, Segment). It is worth remarking that all results achieved by the proposed approach were obtained by using less than six classifiers. More specifically, in almost all cases less than 5 classifiers were used, while for Census dataset with 20 classifiers and for PhotoObject dataset with 20 and 50 classifiers, 5.20, 5.10 and 5.35 classifiers, on average, were used.

The comparison between the number of classifiers selected by our method and that selected by C4.5 shows that, for ensembles including a large number of classifiers (50 in our experiments), the classifiers actually used by our method are much less than those used by C4.5. For

Table 2: Comparison results. Bold values represent the best statistically significant results, while starred values represent the second best results.

Dataset	#cl.	Evo-BN			WMjV	MjV	C4.5		
		err.	#sel.	time	err.	err.	err.	#sel.	time
Adult	**10**	**15.05**	3.10	33.84	17.38	17.35	17.04*	2.8	3.69
	20	**15.65**	3.15	200.46	17.11*	22.88	22.73	4.7	6.35
	50	**13.53**	4.40	3893	14.33*	15.31	16.82	35.0	21.85
Census	**10**	**4.85**	3.55	158	5.27*	5.28	5.33	3.05	9.12
	20	**4.87**	5.20	307	5.24*	5.3	2.94	5.3	30
	50	**4.20**	4.65	6430	5.08	8.87	4.97*	16.51	110
Covtype	**10**	**34.05**	3.75	576	35.95	35.83	35.63*	3.55	15
	20	33.29*	4.90	1753	34.72	36.96	33.93*	5.8	35.21
	50	**31.00**	4.10	7848	32.51*	34.92	33.07	9.9	111
Phoneme	**10**	18.92*	3.10	25.69	19.84	19.7	18.64*	9.84	1.14
	20	17.82*	3.10	130.59	18.37	19.89	17.84*	19.26	1.44
	50	**16.12**	3.50	1433	17.36*	17.5	17.72	37.6	1.90
PhotoObject	**10**	0.67*	3.85	374	0.68	0.68	0.67*	3.5	22.7
	20	0.67*	5.10	844	0.68	1.14	0.66*	4.8	73.7*
	50	0.65*	5.35	8900	0.66	5.07	0.64*	8.0	306.3
Satimage	**10**	21.51*	4.10	32	23.29	22.89	21.16*	10	1.45
	20	20.15*	3.75	159	22.17	21.26	19.64*	20	1.82
	50	19.67*	3.15	1763	21.63	20.36	19.79*	47.95	2.08
Segment	**10**	**12.48**	2.35	15.6	18.28	17.87	13.88*	8.74	1.10
	20	**11.50**	2.55	81.5	16.14	15.86	13.32*	17	1.15
	50	**11.46**	2.85	840	14.9	13.88	13.01*	30.63	1.34

smaller ensembles (containing 10 and 20 classifiers), there are only few cases in which the number of classifiers selected by C4.5 is slightly lower than that selected by ours. In almost all such cases, however, our method obtained better results.

As concerns Table 3, it shows that the proposed approach outperforms the other BN learning approaches for thirteen ensembles. Also in this case for the remaining cases the performance differences are not statistically significant. The results also show that our approach always selects much less classifiers than the other algorithms. In fact, these approaches are not able to reduce significantly the number of classifiers to be used, but tend to use all the classifier responses.

As for the learning times required by the compared approaches, from both tables it can be observed that in all the cases the proposed system needs much more time than the other ones. This is due to the fact that our algorithm evaluates much more solutions than those evaluated by the approaches taken into account for the comparison. It is worth noticing that this higher amount of resources is required only for the learning phase of the system. On the contrary, the time needed by our approach for classifying unknown samples is lower than that of the other approaches. This is because in our system the number of classifiers involved in classification phase is strongly reduced.

5.2. Diversity analysis

In this subsection, a diversity analysis of the ensembles generated by the BoostCGPC algorithm was conducted. The aim is to verify whether there is a correlation between the genotypic and/or the phenotypic diversity of the experts and the accuracy of the results provided by the whole system. More specifically, on the one hand we want to assess whether the trees selected by the BN module present diversity values greater than the average diversity of all the trees composing the ensemble; on the other hand, we want to verify whether this diversity has the effect of producing a lower error rate.

Therefore, we proceed as follows: given an ensemble, for every couple of trees, we compute two diversity measures (genotypic and phenotypic), as described in section 4. Afterwards, for each diversity measure, we compute two average values: $\overline{d_a}$ over all the couples of trees making up the ensemble; $\overline{d_s}$ over all the couples of trees selected by the BN module. Finally, for each diversity measure, the *Diversity ratio* $d_r = \overline{d_s}/\overline{d_a}$ is obtained. As concerns the genotypic diversity measure, values of $d_r > 1$ indicate that the trees selected by the BN present, on the average, a diversity greater than that of all the trees included in the ensemble. On the contrary, for the phenotypic diversity measure, the same behavior occurs for values of $d_r < 1$.

Figures 6 and 7 show the diversity ratio versus the test error for the different datasets. Figure 6 shows the genotypic diversity plots, while Figure 7 the phenotypic diver-

Table 3: Comparison results. Bold values represent the best statistically significant results, while starred values represent the second best results.

Dataset	#cl.	Evo-BN			K2			PC			TAN		
		err.	**#sel.**	**time**	**err.**	**#sel.**	**time**	**err.**	**#sel.**	**time**	**err.**	**#sel.**	**time**
Adult	**10**	**15.05**	3.10	33.84	17.34	3.35	3.00	18.04	10	10.80	17.15*	10	1.68
	20	**15.65**	3.15	200.46	17.14*	3.40	5.49	17.85	20	76.41	17.88	20	2.66
	50	**13.53**	4.40	3893	18.29	3.70	24.52	17.50*	43.40	1066	18	47.5	8.43
Census	**10**	**4.85**	3.55	158	5.42	3.95	5.96	5.41	9.95	34.80	5.39*	10	6.60
	20	**4.87**	5.20	307	5.40*	4.05	15.27	5.58	20	203.3	5.91	20	13.52
	50	**4.20**	4.65	6430	5.20	4.20	73.65	5.10*	48.40	3412	10.29	50	72.70
Covtype	**10**	**34.05**	3.75	576	38.37	3.55	11.34	35.68	9.80	98	35.64*	10	16.95
	20	33.29*	4.90	1753	37.00	4.15	28.89	33.95*	18.50	593	33.99*	20	38.28
	50	**31.00**	4.10	7848	33.94	3.85	361.00	32.45	48.40	5538	32.4*	47.5	126.1
Phoneme	**10**	18.92*	3.10	25.69	19.72	4.00	1.20	19.53*	9.10	6.84	20.05	9	1.43
	20	17.82*	3.10	130.59	19.35	5.00	1.52	18.67*	17.90	38.70	19.48	18	1.51
	50	**16.12**	3.50	1433	19.33	5.05	10.21	17.85*	46.70	560	19.41	50	2.02
PhotoObject	**10**	0.67*	3.85	374	0.66*	3.45	22.44	0.67	9.50	122	0.67	10	25.13
	20	0.67*	5.10	844	0.65*	3.70	64.53	0.76	18.95	763.56	0.7	20	62.09
	50	**0.65**	6.35	8900	0.96	2.85	772	0.82*	48.70	7615	0.87	50	290
Satimage	**10**	21.51*	4.10	32	22.06	4.55	1.45	21.41	7.50	8.90	21.12*	10	1.48
	20	20.15*	3.75	159	21.08	4.10	1.75	20.52	15.80	64.50	19.69*	20	1.66
	50	19.67*	3.15	1763	20.51	4.75	3.53	20.41	42.70	785	19.59*	50	2.50
Segment	**10**	**12.48**	2.35	135.92	14.43	2.55	1.09	13.64*	8.75	6.70	13.71	10	1.12
	20	**11.50**	2.55	311.11	13.30	2.75	1.38	12.88*	2.85	43.80	12.91	20	1.32
	50	**11.46**	2.85	841.47	12.87	3.25	2.25	12.87	46.60	543	12.81*	50	6.74

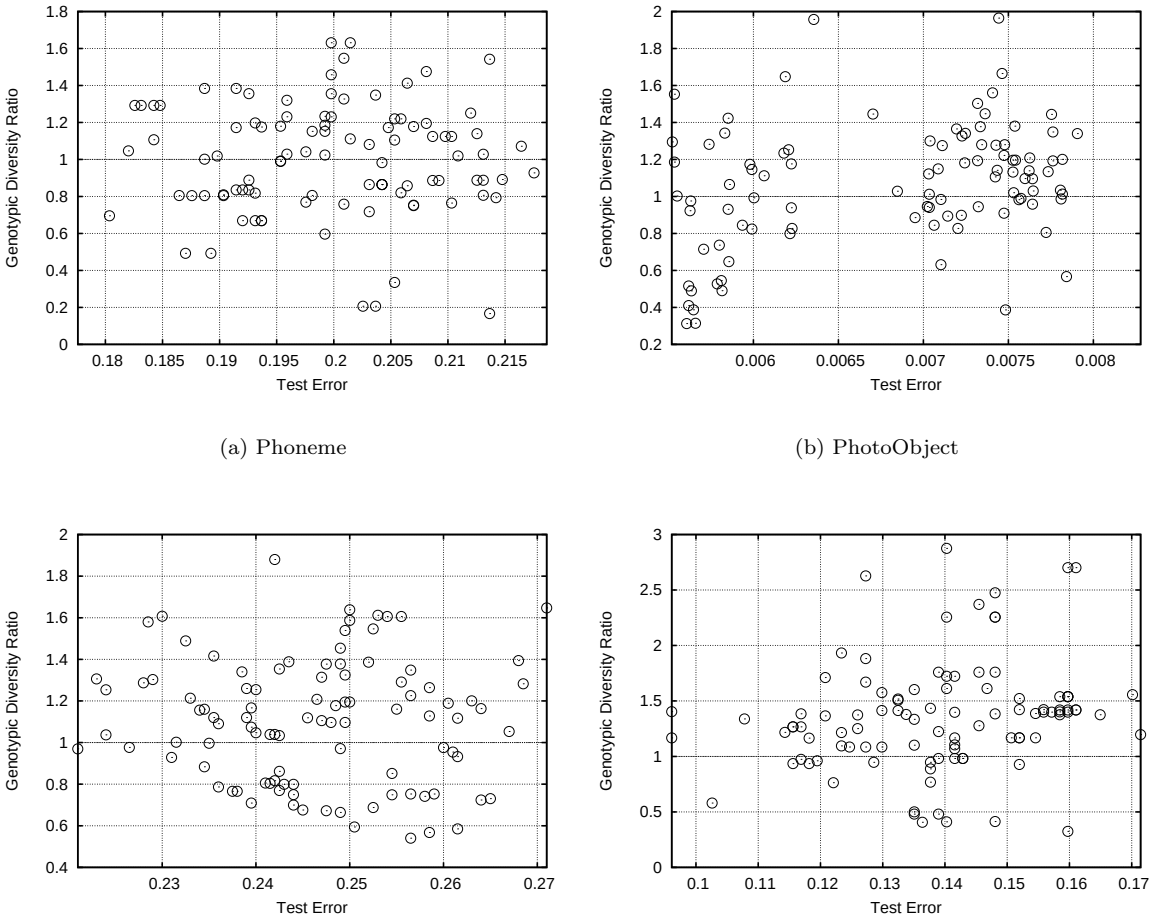
sity plots. A straight line ($d_r = 1$) separates the plotting areas. Each plot refers to one of datasets and the circles represent the results relative to one the 90 performed experiments (for each dataset we have executed 30 runs and for each run we have considered the results of BN10, BN20 and BN50 combiners). For the sake of conciseness, we have shown only the results obtained for *Phoneme*, *PhotoObject*, *Satimage* and *Segment* in the figures, since the remaining ones exhibit a very similar behavior.

As regards the genotypic diversity (Figure 6), it can be observed that for problems with more than two classes (Satimage and Segment) the circles tend to spread out the area above the straight line, thus exhibiting a greater diversity, while circles are more uniformly distributed for the two class problems. As for the phenotypic diversity (Figure 7), the behavior seems to be different and, for all the datasets, the average diversity of the selected trees is lower than that of entire ensemble.

To investigate further the dependency between the test error and the two considered diversity measures, we also used the *Pearson product-moment correlation coefficient* (PMCC). Then, we computed a t-test with a significance value of 0.05. According to this analysis, only for the genotypic diversity and for the cases of *Segment* and *PhotoObject* datasets, a slight correlation was detected. In conclusion, we found experimentally that the BN module does not necessarily select the subset of classifiers which are more diverse among them in the original ensemble, but rather tries to model the joint probability of the true class given the responses of the selected classifiers. In the next Section, we will also show that our method outperforms a selection method which uses a greedy strategy for selecting the classifiers according to a diversity measure.

5.3. Comparison with other selection strategies

In order to test the effectiveness of the ensemble pruning performed by our system, we compared its results with those obtained by two effective and widely used pruning strategies. The rationale behind the first considered strategy is that of selecting the most diverse classifiers according to a previously chosen diversity measure [32]. The pruned ensemble is created by progressively selecting classifier pairs with the lowest diversity until the desired number of classifiers, fixed a-priori, is reached. In our comparison we adopted the two diversity measures described in subsection 4 and considered two different values for the size of the selected ensemble: 5 and 10. This approach will be denoted by the terms *most diverse* in the following. The second classifier selection approach, whose detailed description is reported in [28], considers two quality measures for each couple of classifiers in the original ensemble: the average train error and a diversity measure. These couples of values, can be plotted in a two-dimensional space, where every pair of classifiers is represented by a dot. At this point we can imagine that the most desirable pairs of classifiers are those represented by the dots making the



(a) Phoneme

(b) PhotoObject

(c) Satimage

(d) Segment

Figure 6: Genotypic diversity ratio vs Test error

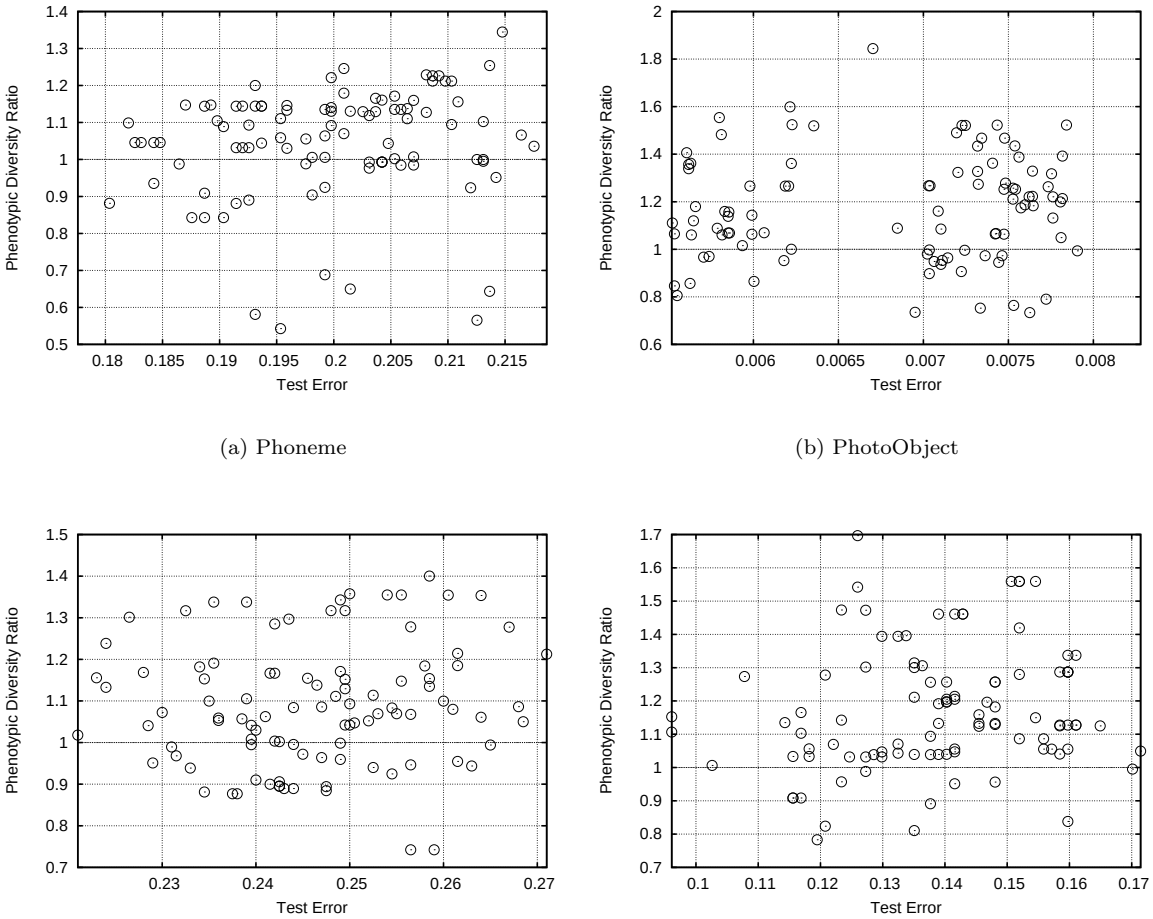
Pareto front of the whole set of classifier pairs. Note that the Pareto front contains all non-dominated points of the plot. A point i is non-dominated if and only if there is no other point j, so that j is better than i on both quality measures. Also in this case, we considered the two diversity measures detailed in Subsection 4. This method will be referred as *Pareto optimal* in the following. For both algorithms we have implemented our own versions.

The comparison results are shown in Table 4. For our approach and for the Pareto optimal one, the average test error and the number of selected classifiers are shown. As regards the most diverse approach, since the number of selected classifiers is fixed a priori, only the test error is reported. Also in this case the results were validated statistically by means of the non-parametric Wilcoxon rank-sum test ($\alpha = 0.05$) over 30 runs and the best statistically significant results are marked in bold, while starred values are the second best results.

For all the datasets, but PhotoObject (20 and 50), our method outperforms both the *pareto optimal* and the *most diverse* selection strategy, but always selects a significant lower number of classifiers. In addition, while our method and the Pareto optimal one generally improve their performance even in the case of a large number of classifiers in the original ensemble (case 20 and 50), the most diverse strategy behaves considerably worse as the number of classifiers increases, especially when the phenotypic diversity measure is used. Take a look for instance at the case of 50 classifiers for *Satimage* and *Segment* datasets or at the case of 20 and 50 classifiers for the *PhotoObject* dataset.

6. Conclusions

We presented a novel approach for improving the performance of derivation tree ensembles, learned by means of a boosted GP algorithm. Our basic idea is to combine the classifiers in the ensemble by using a BN to estimate the conditional probability of each class given the responses of these classifiers. This approach allows modeling explicitly the dependencies among the experts, trying to solve the problems derived by the error correlation that can occur even in boosting techniques. The learning of the Bayesian



(a) Phoneme

(b) PhotoObject

(c) Satimage

(d) Segment

Figure 7: Phenotypic diversity ratio vs Test error

networks was performed by means of an evolutionary algorithm. The system also implements a pruning strategy by exploiting the Markov blanket property. This property allows the true class node state to be predicted only considering the nodes of its Markov blanket.

The experimental results showed that the proposed system further improves the performance achieved by using the boosted GP algorithm, and uses only a small number of classifiers. The selection ability of our system has been investigated by performing further experiments, in which two different measures were used for evaluating the diversity of the selected experts. On the basis of these measures, we analyzed the pruned ensembles provided by the BN module in order to verify whether the performance of our system is correlated to the diversity of the classifiers included in these ensembles. The results showed that the BN module does not necessarily select the subset of classifiers that are more diverse among them in the original ensemble, but instead it tends to select the ensemble classifiers that make fewer errors and therefore are less diverse among them.

We also compared our results with those obtained by using two effective and widely used pruning strategies. The results showed that our method outperforms (or is not significantly different from) the pruning strategies compared and always selects a significantly lower number of classifiers.

Finally, the classifier selection ability of the proposed approach makes it suitable when both a high recognition rate and a low computational cost are required. In addition, while our algorithm maintains its effectiveness even when the original ensemble include a large number of classifiers, other selection strategies based on diversity analysis behave considerably worst in this case.

Future work will focus on investigating two aspects. First, classifier selection procedure, based for example on a divergence measure or similar, will be applied before learning the BN, in order to verify whether better performance of the whole system can be achieved. Second, different scoring functions (e.g. MDL, AIC, BIC) will be tried with the aim of improving the selection ability and/or the accuracy of the proposed system.

Table 4: Comparison results for the selection strategies. The acronym MD stands for Most Diverse; the number following it represents the size of the selected ensemble. The bold values represent the best statistically significant results, while starred values represent the second best results. The best two results which are not statistically different are both starred.

Dataset	ens.	Evo-BN		Pareto optimal				MD 5		MD 10	
				geno		pheno		geno	pheno	geno	pheno
		error	#sel.	error	#sel.	error	#sel.	error	error	error	error
Adult	**10**	**15.05**	3.05	17.25	4.95	17.24	5.20	17.25	17.18*	–	–
	20	**15.65**	3.05	17.68	6.75	16.99*	11.05	21.08	17.16	21.28	17.20
	50	**13.53**	3.90	17.15	9.75	16.99*	13.05	19.64	17.16	18.41	17.41
Cens	**10**	**4.85**	3.50	5.42	4.90	5.42	5.75	5.41*	5.41	–	-
	20	**4.87**	4.25	5.40*	7.05	5.40	10.40	7.94	10.56	8.66	8.90
	50	**4.20**	3.65	5.38*	9.65	5.39	16.60	6.31	12.20	7.90	9.44
Covtype	**10**	**34.05**	3.15	36.01*	5.10	36.01	6.55	36.01	35.94	–	-
	20	**33.29**	3.75	35.15*	7.65	35.38	9.65	36.80	36.15	36.60	36.01
	50	**32.00**	3.50	34.33*	9.35	34.71	14.70	35.80	36.18	36.75	35.98
Phoneme	**10**	**18.92**	3.05	20.29	5.50	20.09*	5.85	20.63	20.12	–	-
	20	**17.82**	3.86	20.04	6.70	19.59*	9.10	21.62	20.19	19.92	19.71
	50	**16.12**	3.21	19.79	8.90	19.51*	10.75	23.02	23.93	21.34	23.97
PhotoObject	**10**	**0.67**	3.55	0.70	5.10	0.69*	8.95	0.69	0.71	–	-
	20	0.67*	3.85	0.69	7.95	0.69*	13.05	2.78	0.73	2.28	0.71
	50	**0.68**	4.24	0.69	11.55	0.68*	22.85	2.51	0.76	2.45	0.72
Satimage	**10**	**21.51**	3.55	24.56	5.70	24.12*	5.70	25.07	24.76	–	-
	20	**20.15**	3.20	24.17	7.05	23.01*	9.80	25.94	24.91	24.01	23.28
	50	**19.67**	2.65	23.95	8.50	22.66*	14.25	28.19	32.90	26.46	26.97
Segment	**10**	**12.48**	2.25	30.14	5.90	30.74	5.40	30.08	17.43*	–	-
	20	**11.50**	2.55	29.11	7.85	28.38	10.10	30.58	17.93	29.05	16.25*
	50	**11.46**	2.85	28.52	9.20	27.59	14.45	32.24	20.95	29.42	20.35*

References

[1] The Sloan Digital Sky Survey (SDSS). available at: http://www.sdss.org/.

[2] Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics*, 1(1-4), 2010.

[3] R. Agrawal, M. Mehta, and J. Shafer. SPRINT: A scalable parallel classifier for data mining. In *Proc. of 22th International Conference on Very Large Data Bases (VLDB)*, volume 96, pages 544–555, 1996.

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

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

[6] E. Cantú-Paz and C. Kamath. Inducing Oblique Decision Trees with Evolutionary Algorithms. *IEEE Transaction on Evolutionary Computation*, 7(1):54–68, February 2003.

[7] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, May 1968.

[8] I. Christou, G. Gekas, and A. Kyrikou. A classifier ensemble approach to the tv-viewer profile adaptation problem. *International Journal of Machine Learning and Cybernetics*, 3(4):313–326, 2012.

[9] G. Cooper and E. Herskovits. A Bayesian method for constructing Bayesian belief networks from databases. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 86–94, 1991.

[10] C. De Stefano, C. D'Elia, A. Scotto di Freca, and A. Marcelli. Classifier Combination by Bayesian Networks for Handwriting Recognition. *Int. Journal of Pattern Rec. and Artif. Intell.*, 23(5):887–905, 2009.

[11] C. De Stefano, F. Fontanella, C. Marrocco, and A. Scotto di Freca. A Hybrid Evolutionary Algorithm for Bayesian Networks Learning: An Application to Classifier Combination. In *EvoApplications (1)*, pages 221–230, 2010.

[12] A. Ekárt and S. Z. Németh. Maintaining the Diversity of Genetic Programs. *Lecture Notes in Computer Science, EuroGP 2002*, 2278:162–171, 2002.

[13] G. Folino, C. Pizzuti, and G. Spezzano. A Cellular Genetic Programming Approach to Classification. In *Proc. Of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 1015–1020, Orlando, Florida, July 1999. Morgan Kaufmann.

[14] G. Folino, C. Pizzuti, and G. Spezzano. GP Ensembles for Large-Scale Data Classification. *IEEE Transaction on Evolutionary Computation*, 10(5):604–616, October 2006.

[15] G. Folino, C. Pizzuti, and G. Spezzano. Training Distributed GP Ensemble With a Selective Algorithm Based on Clustering and Pruning for Pattern Classification. *IEEE Trans. Evolutionary Computation*, 12(4):458–468, 2008.

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

[17] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163, Nov. 1997.

[18] C. Gagné, M. Sebag, M. Schoenauer, and M. Tomassini. Ensemble learning for free with evolutionary algorithms? In *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO 2007)*, pages 1782–1789, 2007.

[19] G. Giacinto and F. Roli. An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, 22(1):25–33, 2001.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, nov 2009.

[21] D. Heckerman. A tutorial on learning with Bayesian networks. Technical report, Learning in Graphical Models, 1995.

[22] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243, 1995.

[23] D. Hernández-Lobato, J. Hernández-Lobato, R. Ruiz-Torrubiano, and Á. Valle. Pruning adaptive boosting ensembles by means of a genetic algorithm. *Intelligent Data Engineering and Automated Learning–IDEAL 2006*, pages 322–329, 2006.

[24] H. Iba. Bagging, Boosting, and Bloating in Genetic Programming. In *Proc. Of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 1053–1060, Orlando, Florida, July 1999. Morgan Kaufmann.

[25] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239, 1998.

[26] L. Kuncheva and C. Shipp. An investigation into how AdaBoost affects classifier diversity. In *Proc 9th International Conference on Information Processing and Management of Uncertainty (IPMU02)*, 2002.

[27] L. Kuncheva, M. Skurichina, and R. P. W. Duin. An experimental study on diversity for bagging and boosting with linear classifiers. *Information Fusion*, 3(4):245–258, 2002.

[28] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[29] E. L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Springer, 2006.

[30] Y. Liu and X. Yao. Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29:716–725, 1999.

[31] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Trans. Evolutionary Computation*, 4(4):380–387, 2000.

[32] D. Margineantu and T. Dietterich. Pruning Adaptive Boosting. In *Proceedings of the International Conference on Machine Learning*, pages 211–218, 1997.

[33] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez. An Analysis of Ensemble Pruning Techniques Based on Ordered Aggregation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):245–259, 2009.

[34] J. Meynet and J. P. Thiran. Information Theoretic Combination of Classifiers with Application to AdaBoost. In M. Haindl, J. Kittler, and F. Roli, editors, *MCS 2007*, volume 4472 of *Lecture Notes in Computer Science*, pages 171–179. Springer, 2007.

[35] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[36] A. L. Prodromidis and S. J. Stolfo. Cost Complexity-Based Pruning of Ensemble Classifiers. *Knowledge and Information Systems*, 3(4):449–469, 2001.

[37] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, San Mateo, Calif., 1993.

[38] P. Spirtes, C. Glymour, and R. Schienes. *Causation prediction and search*. MIT Press, 2001.

[39] G. Tsoumakas, L. Angelis, and I. P. Vlahavas. Selective fusion of heterogeneous classifiers. *Intelligent Data Analalisys*, 9(6):511–525, 2005.

[40] J. H. Zhai, H. Y. Xu, and X. Z. Wang. Dynamic ensemble extreme learning machine based on sample entropy. *Soft Computing*, 16(9):1493–1502, 2012.

[41] L. Zhang, X.-H. Ling, J. W. Yang, X. Q. Wang, and F. Z. Li. Cascaded cluster ensembles. *International Journal of Machine Learning and Cybernetics*, 3(4):335–343, 2012.

[42] Y. Zhang, S. Burer, and W. Street. Ensemble pruning via semi-definite programming. *The Journal of Machine Learning Research*, 7:1315–1338, 2006.

[43] Z. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

[44] Z.-H. Zhou and W. Tang. Selective Ensemble of Decision Trees. In G. Wang, Q. Liu, Y. Yao, and A. Skowron, editors, *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, volume 2639 of *Lecture Notes in Computer Science*, pages 589–589. Springer Berlin–Heidelberg, 2003.