Pruning GP–Based classifier Ensembles by Bayesian Networks

C. De Stefano[1], G. Folino[2], F. Fontanella[1] and A. Scotto di Freca[1]

[1] Università di Cassino e del Lazio Meridionale – Italy
{destefano,fontanella,a.scotto}@unicas.it
[2] ICAR-CNR Istituto di Calcolo e Reti ad Alte Prestazioni –Italy
folino@icar.cnr.it

Abstract. Classifier ensemble techniques are effectively used to combine the responses provided by a set of classifiers. Classifier ensembles improve the performance of single classifier systems, even if a large number of classifiers is often required. This implies large memory requirements and slow speeds of classification, making their use critical in some applications. This problem can be reduced by selecting a fraction of the classifiers from the original ensemble. In this work, it is presented an ensemble-based framework that copes with large datasets, however selecting a small number of classifiers composing the ensemble. The framework is based on two modules: an ensemble-based Genetic Programming (GP) system, which produces a high performing ensemble of decision tree classifiers, and a Bayesian Network (BN) approach to perform classifier selection. The proposed system exploits the advantages provided by both techniques and allows to strongly reduce the number of classifiers in the ensemble. Experimental results compare the system with well-known techniques both in the field of GP and BN and show the effectiveness of the devised approach. In addition, a comparison with a pareto optimal strategy of pruning has been performed.

1 Introduction

In the last two decades, classifier ensemble techniques have shown to be a viable alternative to using a single classifier [10]. Such techniques try to effectively combine the responses provided by a set of classifiers, that have been properly trained in such a way that they are "diverse", i.e. they make uncorrelated errors. The responses are usually combined by means of a voting mechanism, which labels an unknown sample by assigning it the class label which has the highest occurrence among those provided by the whole set of classifiers. Ensemble techniques have been also used for improving GP–based classification systems [2, 6, 9]. In [2] and [6], ensembles of decision trees are evolved, and the diversity among the ensemble members is obtained using techniques like bagging and boosting. Both such approaches are meta-algorithms that aggregate multiple classifiers, or hypotheses, generated by the same learning algorithm trained on different distributions of training data. As concerns the combining rules, bagging uses

the majority vote, while boosting adopts the weighted majority vote, where the weight associated to a classifier is computed on the basis of its overall accuracy on the training data. In [6], a novel GP–based classification system, called *BoostCGPC*, based on the AdaBoost.M2 boosting algorithm [8], has been presented. It is based on a model of the population, in which 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. The experimental results presented in [6] showed that boostCGPC represents an effective classification algorithm able to deal with large data sets.

As mentioned above, classifier ensembles may improve the performance of single classifier systems, but often a large number of classifiers is required. This implies large memory requirements and slow speeds of classification, making their use critical in some applications. This problem can be solved by selecting a fraction of the classifiers from the original ensemble. Such reduction, often denoted as "ensemble pruning" in the literature, can perform even better than the whole ensemble if a subset of complementary classifiers is selected [12, 1]. When the cardinality N of the whole ensemble is high, the problem of finding the optimal sub-ensemble becomes computationally intractable because of the resulting exponential growth of the search space. Several heuristic algorithms have been proposed in the literature for finding near optimal solutions [12].

In a previous work [4] the above problem has been faced by reformulating the classifier combination problem 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. According to this approach, for each training sample, the combiner estimates the conditional probability of each class, given the set of labels provided by the ensemble classifiers. In this way, it is possible to automatically derive the combining rule through the estimation of the conditional probability of each class. Moreover, it is also possible to identify redundant classifiers, i.e. classifiers whose outputs do not influence the output of the combiner. In fact, if the behavior of such classifiers is very similar to that of other classifiers in the ensemble, then they may be discarded without affecting the overall performance of the combiner. In such a way the main drawback of the combining methods discussed above can be overcame. In [5] a Bayesian Network (BN) [11] has been used to automatically infer the joint probability distributions between the outputs of the classifiers and the class label. The BN learning has been performed by means of an evolutionary algorithm using a direct encoding scheme of the BN structure.

In this paper we present a new classification system that exploits the advantages of the two aforementioned approaches. The goal is to build a high performance classification system that uses a small number of classifiers and is able to deal with large data sets. For this purpose, we built a two–module system that combines the BoostCGPC algorithm [6] with the BN based approach to classifier combination [5]. The proposed system allows us to strongly reduce the

number of classifiers in the ensemble. More specifically, such result is achieved by following two different approaches: the boostCGPC evolves diverse classifiers (decision trees) by means of a boosting technique; the BN module evaluates classifiers diversity by estimating the statistical dependencies of the responses they provide. Such estimate is used to select, among the classifiers provided by the BoostCGPC module, a small number of them. Moreover, the responses provided by the selected classifiers are effectively combined by means of a rule learned by the BN module.

The effectiveness of the proposed system, has been tested by performing several experiments. The obtained results have been compared with those obtained by the BoostCGPC approach [6] and with those achieved by using the K2 algorithm [4]. Moreover, the effectiveness of the devised approach as pruning strategy has been tested by comparing its results with those obtained by the Pareto optimal pruning strategy [10].

2 System Architecture

The proposed system consists of two main modules: the first one builds an ensemble of decision tree classifiers (experts) by means of the BoostCGPC algorithm. The second one uses a BN to implement the combining rule that produces the final output of the whole system. More specifically, unknown samples are recognized using a two–step procedure: (i) the feature values describing the unknown sample are provided to each of the ensemble classifiers built by the BoostCGPC module; (ii) the set of responses produced is given in input to the BN module. Such module labels the sample with the most likely class, among those of the problem at hand, given the responses collected by the first module[3]. Also the learning phase requires two steps. In the first step, the BoostCGPC module is trained using a data set containing labeled samples described by their feature values. This learning is carried out, by means of a boosting–based technique (described in Subsection 2.1). In the second step, the responses provided by the set of decision trees built in the first step are used to learn the BN of the second module (Subsection 2.2).

2.1 BoostCGPC Algorithm

The *Boost Cellular Genetic Programming Classifier* [6] 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 correlated with a distributed implementation.

[3] Note that the second step does not require any further computation with respect to the Majority Voting rule. In fact, it only needs to read tables storing class probabilities.

Each GP classifier forming the ensemble is built using a cellular GP algorithm (cGP), enhanced with the boosting technique, which runs on each node. 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. 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.

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 bit better than random guessing".

In practice, the original 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 T weak hypotheses 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 $\Sigma_i w_i^{t+1} = 1$. 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:

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

Given the training set $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and the number P of processors to use 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 k generations and trained on its local sample by running cGP.

After k generations, the individual with the best fitness is selected for participating to vote. In fact the P individuals of each subpopulation having 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, collected during the different rounds, are used to evaluate the accuracy of the classification algorithm.

2.2 The BN Module

As mentioned in the Introduction, the problem of combining the responses provided a set of classifiers can be handled by estimating the conditional probability of each class given the set of labels provided by the classifiers. Such problem may be effectively solved by using a Bayesian Network (BN). In particular, in [4], a BN has been used for combining the responses of more classifiers in a multi expert system.

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) [11]. The nodes of the graph correspond to variables, while the arcs characterize the statistical dependencies among them. An arrow from a node i to a node j has the meaning that j is conditionally dependent on i, and we can refer to i as a *parent* of j. 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 represented variables $\{e_1, \ldots, e_L\}$ can be described as:

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

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

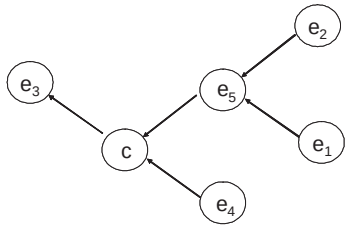

Fig. 1. An example of a BN. The sets $\mathbf{pa_{e_5}} = \{\mathbf{e_1}, \mathbf{e_2}\}$, $\mathbf{pa_c} = \{\mathbf{e_4}, \mathbf{e_5}\}$ and $\mathbf{pa_{e_3}} = \{\mathbf{c}\}$ respectively represents the parent sets of the nodes e_5, c and e_3. The DAG structure induces the factorization of the joint probability $\mathbf{p(c, e_1, e_2, e_3, e_4, e_5) = p(e_3|c)p(c|e_4, e_5)p(e_5|e_1, e_2)p(e_1)\ p(e_2)p(e_4)}$. In this case $\mathbf{E_c} = \{\mathbf{e_3}\}$, $\mathbf{E_{\overline{c}}} = \{\mathbf{e_4}, \mathbf{e_5}\}$.

are known, if we consider c as a variable in the joint probability of Eq. (2). 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 parent of one or more nodes of the DAG. Therefore, it may be useful to divide the set of DAG nodes that are not parent of c in two groups: the first one, denoted as E_c, contains the nodes having the node c among their parents, and the second one, denoted as $E_{\overline{c}}$, the remaining ones. With this assumption, Eq. (3) can be rewritten as:

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

As will be shown in the following section, this property allows a BN to recognize a given sample only considering the responses provided by classifiers represented by the nodes that are directly linked to the class node. For instance, the BN shown in Fig. 1 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 approach allows to detect a reduced set of relevant experts, namely the ones connected to node c, whose responses are actually used by the combiner to provide the final output, while the set $E_{\overline{c}}$ of experts, which do not add information to the choice of $\widehat{c}$, are discarded.

Using a BN for combining the responses of a set of classifiers requires that both the network structure, which determines the statistical dependencies among variables, and the parameters of the probability distributions be learned from a training set of examples. The structural learning is aimed at capturing the relation between the variables, and hence the structure of the DAG. It can be seen as an optimization problem which requires the definition of a search strategy in the space of graph structures, and a scoring function for evaluating the effectiveness of candidate solutions. A typical scoring functions is the posterior probability of the structure given the training data [3]. Once the DAG structure has been determined, the parameters of the conditional probability distributions are computed from training data.

The exhaustive search of 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 at each step a local scoring function 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. The algorithm is based on a specifically devised data structure for encoding DAG, called *multilist* (ML), which allows an effective and easy implementation of the genetic operators. Further details about ML data structure and the genetic operators can be found in [5].

3 Experimental Results

The proposed approach has been tested on five data sets: *Census*, *Segment*, *Adult*, *Phoneme* and *Covtype*. The size and class distribution of these data sets are described in Table 1. They present different characteristics in the number and type (continuous and nominal) of attributes, two classes versus multiple classes and number of samples. Each dataset has been divided, as usual, in a training set (2/3 of the original data) and in a test set (1/3 of the original data).

All the experiments have been 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. The BoostGCPC module used 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 for node. The original training set has been partitioned among 5 nodes and 10 rounds of boosting, with 100 generations for round, have been performed in order to produce 50 classifiers. It is worth to remember the algorithm produce a different classifier for each round on each node.

All results were obtained by averaging over 30 runs. For each run of the BoostCGPC module, a run of the BN module has been carried out. Each BN run has been performed by using the responses, on the whole training set, provided by the classifiers learned in the corresponding BoostCGPC run. The results on the test set has been obtained by first submitting each sample to the learned decision trees ensemble. Then the ensemble responses have been provided to the learned BN. Finally, the BN output label has been compared with the true one of that sample.

The results achieved by our approach (hereafter BN-Boost-CGPC) have been compared with those obtained by the BoostCGPC approach, which uses the wighted majority rule (Eq. 1) for combining the ensemble responses. Moreover, in order to test the effectiveness of the evolutionary learning performed by the second module of the proposed system, we also compared our results with those obtained by a standard algorithm for learning Bayesian Networks, namely the K2 algorithm [4]. Such a algorithm uses a hill climbing technique to learn Bayesian Networks from data. With the aim of performing a fair comparison, we adopted for the K2 algorithm the same scoring function used by our system for evaluating the quality of the network structure. For each dataset, these BNs have been learned on the responses provided by the set of classifiers supplied by the first

Table 1. The data sets used in the experiments

datasets	**attr.**	**samples**	**classes**
Adult	14	48842	2
Census	4	299285	2
Phoneme	5	5404	2
Segment	36	2310	6
Covtype	54	581012	7

module of our system on the training set. The trained BNs have been tested on the responses given by the just mentioned classifiers, obtained on the test set.

Comparison results are shown in Tab. 2. The second column shows the cardinality of the ensembles taken into account (10, 20 and 50 classifiers). The ensembles made of 10 and 20 classifiers have been obtained by considering respectively the first 10 and 20 classifiers generated by the BoostCGPC algorithm. For each considered method, the table reports the training and test error. Column 5 contains the number of classifiers actually used by our approach, i.e. only the classifiers that are directly connected to the class label node in the DAG. Note that for the other methods such number has not been reported since it coincides with the number of classifier making up the ensemble (10, 20 or 50). In order to statistically validate the comparison results, we performed the two–tailed t–test($\alpha = 0.05$) over the 30 carried out runs. The values in bold in the test error columns highlight, for each dataset, the results which, according to the performed test, are significantly better with respect to the second best result (such results are starred). The proposed approach, for all considered datasets, achieves better performance than those obtained by the two methods used for the comparison. It is worth to remark that such results are always achieved by using only a small number of the available classifiers.

Table 2. Comparison results. Bold values represent the best statistically significant results, while starred values represent the second best results. The Columns with the headers tr and ts respectively contain the train and test errors.

Dataset	ens.	BN-BoostCGPC			BoostCGPC		K2-BN	
		tr	ts	# sel.	tr	ts	tr	ts
Adult	**10**	15.03	**15.05**	3.05	17.24	17.38	17.16	17.34*
	20	15.70	**15.65**	3.05	16.99	17.11*	17.55	17.83
	50	13.19	**13.53**	3.90	14.43	14.33*	17.66	18.29
Census	**10**	4.72	**4.85**	3.50	5.27	5.27*	5.45	5.42
	20	4.73	**4.87**	4.25	5.24	5.24*	5.00	5.40
	50	4.09	**4.20**	3.65	4.97	5.08*	4.50	5.20
Covtype	**10**	33.83	**34.05**	3.15	35.97	35.83*	37.23	38.37
	20	33.06	**33.29**	3.75	34.73	34.72*	36.55	37.00
	50	30.80	**31.00**	3.50	32.52	32.51*	33.04	33.94
Phoneme	**10**	17.97	**18.92**	3.05	19.14	19.84	19.52	19.72*
	20	17.10	**17.82**	3.86	17.92	18.37*	19.03	19.35
	50	15.81	**16.12**	3.21	16.85	17.36*	18.73	19.33
Segment	**10**	12.08	**12.48**	2.25	17.33	18.28	13.52	14.43*
	20	10.80	**11.50**	2.55	15.24	16.14	12.33	13.30*
	50	11.08	**11.46**	2.85	14.15	14.90	11.69	12.87*

In order to test the effectiveness of the ensemble pruning performed by our system, we compared its results with those obtained by the Pareto optimal pruning strategy [10]. Such approach considers, for each couple of classifiers in the

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

Dataset	ens.	BN-Boost		Pareto optimal			
		error	#sel.	geno		pheno	
				error	#sel.	error	#sel.
Adult	**10**	**15,05**	3,05	17,25	4,95	$17,24^{\star}$	5,20
	20	**13,53**	3,90	17,15	9,75	$16,99^{\star}$	13,05
	50	**13,53**	3,90	17,15	9,75	$16,99^{\star}$	13,05
Cens	**10**	**4,85**	3,50	$5,42^{\star}$	4,90	5,42	5,75
	20	**4,87**	4,25	$5,40^{\star}$	7,05	5,40	10,40
	50	**4,20**	3,65	$5,38^{\star}$	9,65	5,39	16,60
Covtype	**10**	**34,05**	3,15	$36,01^{\star}$	5,10	36,01	6,55
	20	**33,29**	3,75	$35,15^{\star}$	7,65	35,38	9,65
	50	**32,00**	3,50	$34,33^{\star}$	9,35	34,71	14,70
Phoneme	**10**	**18,92**	3,05	20,29	5,50	$20,09^{\star}$	5,85
	20	**17,82**	3,86	20,04	6,70	$19,59^{\star}$	9,10
	50	**16,12**	3,21	19,79	8,90	$19,51^{\star}$	10,75
Segment	**10**	**12,48**	2,25	$30,14^{\star}$	5,90	30,74	5,40
	20	**11,50**	2,55	29,11	7,85	$28,38^{\star}$	10,10
	50	**11,46**	2,85	28,52	9,20	$27,59^{\star}$	14,45

original ensemble, two quality measures: the average train error and a pairwise 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 up the 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. As concerns the diversity measure, we taken into account two different measures, better described in [7]: a genotypic measure that evaluates the structural diversity between the two trees representing the couple of classifiers to be assessed; a phenotypic measure based on Kappa statistics, which gives a score of how much homogeneity there is in the responses provided by two classifiers. The comparison results are shown in Table 3. Also in this case the results have been statistically validate by means of the two–tailed t–test($\alpha = 0.05$) over the 30 carried out runs and the best statistically significant results are marked in bold. From the table it can be seen that for all the datasets, our method outperforms the Pareto optimal approach although it selects a significant minor number of classifiers.

4 Conclusions

We presented a new framework for improving the performance of classifier ensemble, by means of an effective pruning algorithm based on Bayesian networks.

The framework consists of two modules: an ensemble-based Genetic Programming system, which produces a high performing ensemble of decision tree classifiers, and a Bayesian Network approach to perform classifier selection.

The effectiveness of the proposed system has been tested by comparing the accuracy of the framework with those obtained by the BoostCGPC approach and with those achieved by using a Bayesian Network learned by using the K2 algorithm. In addition, in order to validate the effectiveness of the approach as pruning strategy, it has been compared with the Pareto optimal pruning strategy. For all the datasets, our method obtains better results than the other methods both in terms of accuracy and of the number of classifiers selected, confirming the goodness of its usage as a pruning strategy. Future works will include the comparison with other state-of-the-art methods and the exploration of the overhead in terms of execution time our method requires.

Acknowledgments

This research work has been partially funded by the MIUR project FRAME, PON01-02477.

References

1. R. Banfield, L. Hall, K. Bowyer, and W. Kegelmeyer. Ensembles diversity measures and their application to thinning. *Information Fusion*, 6:49–62, 2005.
2. E. Cantú-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, 7(1):54–68, February 2003.
3. G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.
4. 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.
5. 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.
6. G. Folino, C. Pizzuti, and G. Spezzano. Gp ensembles for large-scale data classification. *IEEE Trans. on Evolutionary Computation*, 10(5):604–616, October 2006.
7. 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.
8. Y. Freund and R. Shapire. In *Proceedings of the 13th Int. Conference on Machine Learning*.
9. C. Gagné, M. Sebag, M. Schoenauer, and M. Tomassini. Ensemble learning for free with evolutionary algorithms? In *GECCO*, pages 1782–1789, 2007.
10. L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
11. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
12. Z. Zhou and W. Tang. Selective ensemble of decision trees. *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 589–589, 2003.