DOI: http://dx.doi.org/10.1109/TSMCC.2004.833295

Using All Data to Generate Decision Tree Ensembles

Gonzalo Martínez-Muñoz and Alberto Suárez

***Abstract*— The present work develops a new method to generate ensembles of classifiers that uses all available data to construct every individual classifier. The base algorithm, presented in [1], builds a decision tree in an iterative manner: The training data is divided into two subsets. In each iteration, one subset is used to grow the decision tree, starting from the decision tree produced by the previous iteration. This fully grown tree is then pruned by using the other subset. The roles of the data subsets are interchanged in every iteration. This process converges to a final tree that is stable with respect to the combined growing and pruning steps. To generate a variety of classifiers for the ensemble we randomly create the subsets needed by the iterative tree construction algorithm. The method exhibits good performance in several standard datasets at a low computational cost.**

***Index Terms*— Patern recognition, classification ensembles, bagging, decision trees.**

I. Introduction

THE development of ensembles of classifiers is a topic of great activity in the field of supervised learning [2]–[9]. This flurry of activity has been spurred by the performance improvements that can be obtained using this simple technique. An ensemble of classifiers categorizes new examples by pooling the decisions made by its components. The individual decisions of the base classifiers are combined by either weighted or unweighted voting to obtain the final decision of the ensemble. An ensemble can be much more accurate than any of the classifiers of which it is composed [6]. Obviously, this accuracy improvement can only be achieved if the single classifiers are sufficiently diverse: Pooling together the results of identical classifiers would not lead to any improvement; in fact the ensemble would always produce the same classification as a single classifier. The key point when creating ensembles is to use the available data to obtain classifiers with uncorrelated errors [6].

Several techniques have been proposed to achieve this goal [6]:

1) Manipulating the input features: This technique deletes features of the input data before constructing every individual classifiers. The selection of the features to delete should be done very carefully as the final performance of the ensemble could be seriously affected.
2) Manipulating the output targets [10], [11]: Every individual classifier is built using a different random class relabeling. This method generates a new 2 class problem from the original problem by assigning half of the original labels to a new A class and the other half of the labels to a new B class. Then, the classifiers are trained on these new problems. This method has the limitation that can be only used with problems having many classes.
3) Injecting randomness: This technique introduces a certain degree of randomness into the base learning algorithm in such a way that two executions with the same data produce two different classifiers. The learning algorithm performance is reduced to obtain a diversity of classifiers that can be combined into an ensemble. One example of this technique is choosing randomly between the k best tests that can be made in a decision tree node [12].
4) Subsampling the training examples by generating different views of the data to build every individual classifier. Methods as boosting [9], bagging [4] and wagging [2] fall into this group of techniques. Bagging [4], one of the most widespread methods, constructs each individual classifier using a random sample of N training examples drawn with replacement from the original N-sized training set (bootstrap sample). Each sample contains on average 63.2% of the original training set and the rest of the sample are repeated examples.

In general, all these techniques deteriorate the performance of the single classifiers for the sake of obtaining a diversity of classifiers that, when used as a committee, perform better than the single classifiers.

This paper presents an ensemble generation method that does not reduce the efficiency of the base algorithm. It is based on the intrinsic variability of the Iterative Growing and Pruning Algorithm (IGPA), a tree construction method designed by Gelfand et al. [1]. IGPA generates decision trees by dividing the training data in two disjoint groups of approximately equal size and class distribution. IGPA uses iteratively one data group to grow the tree and the other one to prune it, interchanging the roles of the groups on each iteration. This algorithm has the property that different subdivisions of the data may generate different trees, even though the same original training data is being used. This fact, together with the good performance of the individual classifier should lead to an improvement in the efficiency of the ensemble.

II. Learning algorithm

A. Base algorithm

The base classifier in the ensemble is a decision tree built by applying the Iterative Growing and Pruning Algorithm (IGPA) [1]. The input of the learning algorithm is the set of training data, which consists of a collection of N_{train} labeled examples $L = \{(\boldsymbol{x_n}, y_n), n = 1, 2, ..., N_{train}\}$. Each training example $(\boldsymbol{x_n}, y_n)$ is characterized by the feature vector $\boldsymbol{x_n}$ and the class label y_n. The goal is to predict the class label y given a feature vector $\boldsymbol{x}$ using the knowledge contained in the training data set L. To construct an IGPA classification tree, the training set L is randomly divided into two subsets, $L^{(1)}$ and $L^{(2)}$, of approximately equal size and class distribution. Then, the algorithm makes use of one subset to fully grow the tree and of the other one to prune it. The growing and pruning sequence is repeated with the roles of the subsets reversed at each iteration: first, a tree T_0 is grown using $L^{(1)}$. Then, the fully-grown tree T_0 is reduced to its optimal size T_0^* with respect to subset $L^{(2)}$ using a pruning procedure that returns the pruned tree with the smallest resubstitution error for a given data set (in this iteration $L^{(2)}$). Once the first pruning is completed, the roles of the data subsets are interchanged and a new tree T_1 is generated by growing new leaves off the terminal nodes of T_0^* using the $L^{(2)}$ subset. Then T_1 is pruned to its optimal size with respect to $L^{(1)}$. In the successive iterations the roles of the growing and pruning subsets are interchanged until two consecutive pruned trees are of equal size. It can be proved that this sequence converges [1].

The details of the growing and pruning phases are:

1) Tree growing phase: For the tree growing phase the CART algorithm is used [13]. This algorithm generates a tree in a recursive way. Starting from a terminal node, t, that corresponds to a region of the attribute space $U(t)$, two children nodes are created by means of a boolean test on the attributes. This test divides the original region, $U(t)$, into two disjoint regions, $U(t_R)$ and $U(t_L)$ (corresponding to the children nodes t_R and t_L), such that $U(t) \equiv U(t_R) \bigcup U(t_L)$ and $\emptyset \equiv U(t_R) \bigcap U(t_L)$. The recursive process continues, by splitting the regions defined by the nodes t_R and t_L, until some stopping criteria is fulfilled.

The two main points in the growing phase are: (1) how to chose the splits and (2) when to stop the recursive process. In the CART algorithm the growing phase stops when any of the following conditions is fulfilled for a terminal node t:

1) All the training examples falling in the region $U(t)$ belong to the same class.
2) The number of training examples falling in the region $U(t)$ is less or equal than a parametrically specified minimum.
3) There is no split such that there are at least 1 example falling in $U(t_R)$ and in $U(t_L)$.

Next, we look at the problem of selecting the split at each node. The main point here is to find the split that better separates the different classes. In the CART algorithm this is done by measuring the value of an impurity function after and before the split, s, and by choosing the split that maximize this variation:

$$\Delta i(s,t) = i(t) - (i(t_L)p_L + i(t_R)p_R)$$

where p_L and p_R are the fraction of the examples falling in $U(t)$ that go to the regions defined by $U(t_R)$ and $U(t_L)$ respectively. As impurity function CART uses the Gini criterion given by:

$$i(t) = \sum_{i \neq j} p(i|t)p(j|t)$$

where $p(i|t)$ is the fraction of data in node t that belongs to class i.

2) Tree pruning phase: For the pruning phase we use a simple and fast algorithm proposed in [1]. This pruning method works in a bottom-up manner, i.e. each node of the tree is processed only after its children have been processed. The tree branch grown form inner node t_i is pruned only if the error of the pruned tree is lower or equal to the error of the unpruned tree. The true error is estimated as the resubstitution error on a dataset that is independent of the data used to grow the decision tree.

B. Ensemble algorithm

The efficiency of bagging in reducing the generalization error seems to be high when the base classifier has low bias and exhibits large variability [3]. The IGPA procedure is unstable with respect to how the training examples are grouped. Therefore, the desired variability can be obtained in IGPA by random grouping of the data. This mechanism can not be exploited in other tree construction algorithms, such as CART and C4.5 [14], which are not affected by the way training data is ordered or grouped. Ensemble methods using C4.5 or CART as base algorithms induce variability in the classifiers by introducing a perturbation that is extrinsic to the data (a bootstrap resampling in bagging, an adaptive or random weighting in boosting and wagging, respectively, a randomization of outputs [10], etc.). This perturbation generally deteriorates the performance of the individual classifiers.

In this work, we take advantage of the aforementioned instability and obtain a diversity of classification trees by running the IGP algorithm with different random subdivisions of the training data into the two subsets $L^{(1)}$, $L^{(2)}$. In this manner, the variability in the classifiers in IGPA-ensembles is intrinsic to the tree construction algorithm and not imposed in an *ad-hoc* manner. Furthermore variability is not achieved at the expense of decreasing the performance of the individual decision trees.

III. Empirical results

In order to test the performance of the proposed ensemble of IGP trees, several real data sets from the UCI repository [15] are used: German Credit, Pima Indian Diabetes, Breast Cancer Wisconsin, and Sonar. To analyze the performance of the method as a function on the size of the training data we carry out a more detailed study in the waveform dataset, a synthetic data set proposed by [13]. In order to avoid spurious effects, data sets with no missing values were chosen. Table I shows the characteristics of the selected data sets. Column 2 and 3 give the number of examples for training and testing respectively. Column 4 shows the number of classes and column 5 the number of attributes of the problem.

The performance of the method is evaluated using CART-based bagging as a reference. C4.5-based bagging could also have been used. However, CART and C4.5 are very similar decision tree construction algorithms, and their accuracy is comparable in many empirical datasets [16]. Furthermore,

TABLE I

CHARACTERISTICS OF THE USED DATASETS

Dataset	**Train**	**Test**	**Classes**	**Attribs**
Waveform	300	5000	3	21
German Credit	600	400	2	24
Pima Indian Diabetes	500	268	2	8
Breast Cancer Wisconsin	500	199	2	9
Sonar	120	88	2	60

IGPA is based on the growing heuristics of the CART algorithm, which makes CART-based bagging preferable as a benchmark.

Two types of experiment were made: First we measure the performance of the algorithm with respect to the number of classifiers: A collection of 101 classifiers are generated in each run of the experiment; the results are then analyzed sequentially in order to obtain the error values for these classifiers grouped in ensembles whose size ranges from 1 to 101 classifiers. A second batch of experiments is designed to analyze the dependency of performance on how many training examples are used. Additionally, we also report the error performance of the individual classifiers, CART and IGPA. This last measure is used as a reference error for the ensembles and to ascertain whether the ensembles of classifiers perform better than the simple classifiers. Note that the error of a CART tree is generally lower than that of a bagging ensemble with a single classifier. This is due to the fact that the CART tree is grown using all the elements in the training set, while the resampling process with replacement used to generate the decision trees in the ensemble selects on average only 63.2% different instances of the original training set (the rest of the sample are repeated examples). This is not the case in the IGPA ensemble: Since the variability in the tree construction is intrinsic to the algorithm, both the individual tree and the ensemble tree use the same amount of data.

The protocol for the experiments is as follows: For each dataset in Table I (i) N random training sets with sizes as specified in Table I are generated ; (ii) each algorithm is run N times, one for each training set. In this way the different algorithms work under the same conditions and can be compared in a fair way. The two-sided paired Student's t-test is used to determine whether the differences in performance are statistically significant: The Student's t-test gives the probability (p-value) of two populations having the same mean. A p-value below 0.05 is normally considered as a statistically relevant difference [17].

TABLE II

AVERAGE ERROR IN % FOR THE INDIVIDUAL CLASSIFIERS. STANDARD DEVIATION BETWEEN PARENTHESES

	CART	IGPA	Paired t-test
Waveform	30.1(2.0)	30.6(1.7)	0.31
German	**27.0(2.0)**	28.3(2.1)	0.0061
Pima	25.9(2.5)	2.63(2.5)	0.38
Breast	5.90(1.8)	5.61(1.6)	0.35
Sonar	30.1(4.0)	30.5(5.2)	0.65

Table II displays the results for the CART and IGPA individual classifiers. Figures 1 to 6 present the mean of the ensemble classification error over 50 executions (N=50) with the standard deviation shown between parentheses. The last column in II gives the values of two-sided paired Student's t-test. Statistically important differences are highlighted (i.e. for values of the Student's t-test under 0.05). Notice that, in this case, the whole training set is being used to generate the CART tree. Therefore, the CART tree error rate is lower than that of $C = 1$ CART-bagging (see first column of Table III).

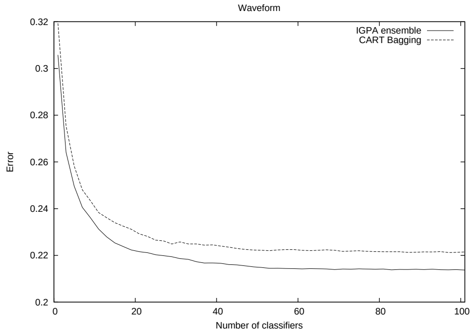

Fig. 1. Error evolution with respect the number of classifiers for the Waveform data set.

Table II shows that, except for the German data set, there are no large differences between both methods, although CART trees have slightly lower error performance in 4 out of the 5 data sets.

Table III presents the classification errors obtained with ensembles of decision trees that use univariate splits. The three sections of the table present the results for an ensemble of 1, 11 and 101 classifiers, respectively. Every section is divided into three columns: the first column displays the results for the CART-based bagging algorithm. The second column shows the results for the IGPA-ensemble obtained by randomly subdividing the training data into two groups of approximately equal size and equal class distribution. The third column shows the p-values for the two-sided paired Student's t-test. Again statistically significant results are highlighted ($p - value < 0.05$).

Each experiment was run 50 times (N=50) for both algorithms. The classification error reported is an average over these 50 executions, with the standard deviation around this average displayed between parentheses in the table. The generated ensembles were analyzed sequentially to obtain the classification errors for ensembles from 1 to 101 classifiers. Figures 1 to 5 show the error with respect to the number of classifiers for the selected problems.

Table III shows that for all the data sets the proposed method yields better or equivalent results to CART-based bagging. In fact, from the 5 analyzed data sets IGPA ensemble performs clearly better in the waveform, breast and German data sets and also in sonar though with a smaller difference. In the Pima data set the results are equivalent for both algorithms. Another important fact is that, for data sets depicted in figures 1, 2 and 4, the improvement is accomplished using few classifiers (C=11) and is maintained when adding more trees to the ensemble.

This generally better performance of the proposed method

TABLE III

AVERAGE ERROR IN % FOR ENSEMBLES OF 1, 11 AND 101 CLASSIFIERS. STANDARD DEVIATION BETWEEN PARENTHESES

	C=1			C=11			C=101		
	CART Bagging	IGPA ensemble	Paired t-test	CART Bagging	IGPA ensemble	Paired t-test	CART Bagging	IGPA ensemble	Paired t-test
Waveform	31.9(1.7)	**30.6(1.7)**	0.0043	23.8(2.1)	**23.1(1.6)**	0.0042	22.2(2.2)	**21.4(1.9)**	2.8E-7
German	28.5(1.9)	28.3(2.1)	0.64	26.3(1.6)	**25.0(1.9)**	6.6E-6	25.9(1.7)	**24.4(1.7)**	2.3E-12
Pima	26.6(2.3)	26.3(2.5)	0.50	25.1(2.2)	24.7(2.2)	0.088	24.9(1.9)	24.7(2.3)	0.33
Breast	6.78(2.2)	**5.61(1.6)**	3.6E-4	5.12(1.6)	**4.53(1.4)**	2.7E-3	4.65(1.4)	**4.25(1.3)**	1.6E-3
Sonar	32.0(4.9)	30.5(5.2)	0.14	27.1(4.5)	**25.9(4.5)**	0.046	26.1(4.1)	**25.3(4.4)**	0.023

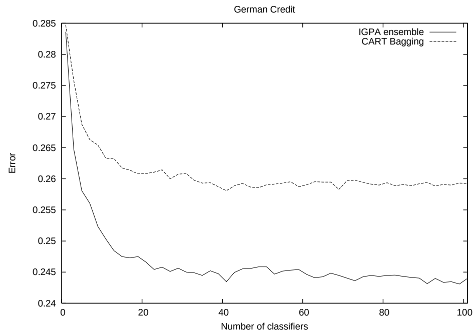

Fig. 2. Error evolution with respect the number of classifiers for the German Credit data set.

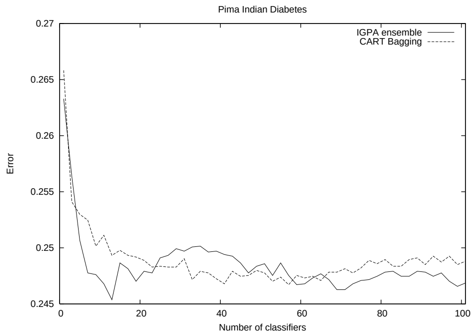

Fig. 3. Error evolution with respect the number of classifiers for the Pima Indian Diabetes data set.

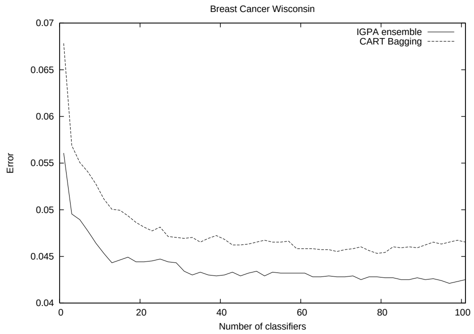

Fig. 4. Error evolution with respect the number of classifiers for the Breast Cancer Wisconsin data set.

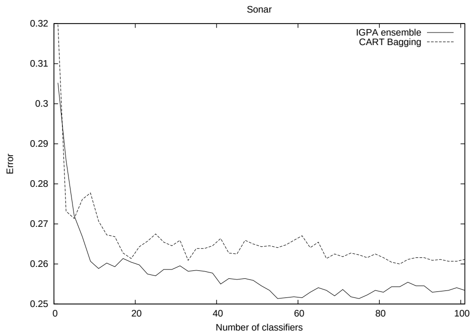

Fig. 5. Error evolution with respect the number of classifiers for the Sonar data set.

over CART bagging could be due to the fact that, in the IGPA-ensemble, every classifier is built making use of all training data instead of an average of 62.3% of the data as bagging does.

We have also measured the variation of the classification error with respect to the number of training examples in the synthetic waveform data set [13]. Table IV shows the mean error (averaged over 10 executions, with standard deviation between parentheses) and mean number of leaves of the generated trees for the IGPA-ensemble and for CART-bagging. The last column shows the result of the p-value of the paired Student's t-test. As before, statistically significant results are highlighted (p-value< 0.05). In each execution an ensemble of 101 trees is generated using the same training data for both algorithms. A graphical rendering of the results is presented in Figure 6. Observe that, as the size of the training data set increases, the error and tree size differences between both algorithms increase as well. Again, this could be due to the fact that our method uses the available data more efficiently. Indeed the obtained trees are bigger in the IGPA-ensemble - each tree is generated with the whole training set - and this fact gives more predictive power to each single tree. This may be the reason of the advantage of the IGPA-ensemble over CART-bagging for this data set.

The presented method is faster and more efficient than CART-based bagging (see Table V) . CART needs to build auxiliary trees to obtain the pruning parameters by cross validation (usually 10 trees) while the IGP algorithm builds just one tree for each member of the ensemble. Additionally, in the IGP algorithm the pruning and growing phases are only performed with half of the data, which implies a reduction of

TABLE IV

ERROR IN % AND TREE SIZE (NUMBER OF LEAVES) VARIATION WITH RESPECT TO THE TRAINING DATA SIZE FOR THE WAVEFORM DATA SET USING 101 CLASSIFIERS. STANDARD DEVIATION BETWEEN PARENTHESES

Size	CART-Bagging	$\|T\|$	IGPA-ensemble	$\|T\|$	Paired t-test
50	26.1 (2.0)	3.42	26.2 (2.8)	3.61	0.8288
100	24.2 (3.0)	4.64	24.0 (3.1)	5.13	0.6856
150	23.9 (2.7)	5.40	**23.1 (2.1)**	6.59	0.0320
200	23.9 (1.9)	6.30	23.0 (1.8)	8.05	0.0552
250	24.0 (2.9)	6.52	**23.2 (2.6)**	9.09	0.0302
300	22.6 (3.2)	7.90	**21.9 (2.8)**	11.0	0.0203
500	20.4 (1.0)	10.5	**19.8 (0.8)**	16.2	0.0422
750	21.4 (1.9)	12.6	**20.1 (0.9)**	21.7	0.0176
1000	20.3 (1.9)	15.5	**18.6 (1.1)**	28.3	0.0013

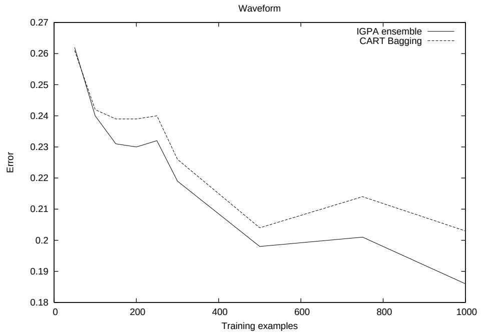

Fig. 6. Error variation with respect to the training data size for the waveform data set.

the computational cost. The IGP algorithm usually converges after few iterations (in our experiments 2 or 3 iterations and no more than 4 are sufficient to reach convergence).

TABLE V

MEAN EXECUTION TIME NEEDED TO BUILD 101-CLASSIFIERS ENSEMBLE FOR THE WAVEFORM DATA SET USING 300 TRAINING DATA

	CART-bagging	IGPA-ensemble
time(sec.)	538	59

IV. CONCLUSIONS

The IGPA-ensemble algorithm generates in a natural way a diversity of classifiers without inserting spurious randomness into the training data or in the learning procedure. Experiments on standard UCI data sets illustrate that an ensemble of trees constructed by randomizing the subsets used in the Iterative Growing and Pruning algorithm exhibits some improvement in classification error over tree ensembles constructed by bagging. This means that, in the IGPA-ensemble, a good diversity of classifiers is obtained although all the classifiers are constructed using the same training examples.

Moreover, it is observed that, when increasing the size of the training set, the improvement in the error rate given by the IGPA-ensemble with respect to the CART-based bagging is higher in the waveform synthetic data set. This improvement of performance seems to be correlated with an increase of the difference in size of the generated trees.

REFERENCES

[1] S. B. Gelfand, C. S. Ravishankar, and E. J. Delp, “An iterative growing and pruning algorithm for classification tree design,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13(2), pp. 138–150, 1991.

[2] E. Bauer and R. Kohavi, “An empirical comparison of voting classification algorithms: Bagging, boosting, and variants,” *Machine Learning*, vol. 36, no. 1-2, pp. 105–139, 1999.

[3] L. Breiman, “Arcing classifiers,” *The Annals of Statistics*, vol. 26(3), pp. 801–849, 1998.

[4] ——, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[5] T. G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.

[6] ——, “Machine-learning research: Four current directions,” *The AI Magazine*, vol. 18, no. 4, pp. 97–136, 1998.

[7] J. R. Quinlan, “Bagging, boosting, and c4.5,” in *Proceedings of the thirteenth National Conference on Artificial Intelligence*, Cambridge, MA, 1996, pp. 725–730.

[8] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, “Boosting the margin: A new explanation for the effectiveness of voting methods,” *The Annals of Statistics*, vol. 12(5), pp. 1651–1686, 1998.

[9] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” in *European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[10] T. G. Dietterich, “Solving multiclass learning problems via errorcorrecting output codes,” *Journal of Artificial Intelligence*, pp. 263–286, 1997.

[11] R. E. Schapire, “Using output codes to boost multiclass learning problems,” in *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 313–321.

[12] T. G. Dietterich and E. B. Kong, “Machine learning bias, statistical bias, and statistical variance of decision tree algorithms,” Oregon State University, Covallis, OR, Tech. Rep., 1995.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. New York: Chapman & Hall, 1984.

[14] J. R. Quinlan, *C4.5: programs for machine learning*. San Mateo (California): Morgan Kaufmann, 1993.

[15] C. L. Blake and C. J. Merz, “UCI repository of machine learning databases,” 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[16] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[17] S. M. Ross, *Introduction to probability and statistics for engineers and scientists*. John Wiley & Sons, 1987.

PLACE
PHOTO
HERE

Gonzalo Martínez-Muñoz received the University Degree in Physics and the Master in Computer Science from the Universidad Autónoma de Madrid (UAM), in 1995 and 2001 respectively. Currently, he is working toward the Ph.D. degree. From 1996 to 2002 he worked for Geosys SL, a Spanish company specialised in Geographical Information Systems and remote sensing, as software designer and developer in the framework of research and development projects. Currently, he is professor in the Escuela Politécnica Superior (UAM) of Object Oriented Programming and Experimental Algorithmics. His research interests include pattern recognition, decision trees, machine learning and genetic algorithms.

PLACE
PHOTO
HERE

Alberto Suárez received the degree of Licenciado in Chemistry from the Universidad Autónoma de Madrid (Spain), in 1988, and the Ph. D. degree in Physical Chemistry from the Massachusetts Institute of Technology (Cambridge, USA) in 1993. After holding postdoctoral positions at Stanford University (USA), the Université Libre de Bruxelles (Belgium), and the Katholieke Universiteit Leuven (Belgium), he is currently a professor in the Computer Science Department of the Universidad Autónoma de Madrid (Spain). He has worked on relaxation theory in condensed media, stochastic and thermodynamic theories of non-equilibrium systems, lattice-gas automata, and decision tree induction. His current research interests include machine learning, computational finance and information processing in the presence of noise.