CTC: An Alternative to Extract Explanation from Bagging

Ibai Gurrutxaga, Jesús Mª Pérez, Olatz Arbelaitz, Javier Muguerza, José I. Martín, and Ander Ansuategi

Dept. of Computer Architecture and Technology, University of the Basque Country
M. Lardizabal, 1, 20018 Donostia, Spain
{i.gurrutxaga,txus.perez,
olatz.arbelaitz,j.muguerza,j.martin}@ehu.es
aansuategui001@ikasle.ehu.es
http://www.sc.ehu.es/aldapa

Abstract. Being aware of the importance of classifiers to be comprehensible when using machine learning to solve real world problems, bagging needs a way to be explained. This work compares Consolidated Tree's Construction (CTC) algorithm with the Combined Multiple Models (CMM) method proposed by Domingos when used to extract explanation of the classification made by bagging. The comparison has been done from two main points of view: accuracy, and quality of the provided explanation. From the experimental results we can conclude that it is recommendable the use of CTC rather than the use of CMM. From the accuracy point of view, the behaviour of CTC is nearer the behaviour of bagging than CMM's one. And, analysing the complexity of the obtained classifiers, we can say that Consolidated Trees (CT trees) will give simpler and, therefore, more comprehensible explanation than CMM classifiers. And besides, looking to the stability of the structure of the built trees, we could say that the explanation given by CT trees is steadier than the one given by CMM classifiers. As a consequence, the user of the classifier will feel more confident using CTC than using CMM.

1 Introduction

The main objective of machine learning techniques when used to solve real world problems is to automate knowledge acquisition for performing useful tasks. The most pursued objective is probably accurate prediction (error or guess), but there are real domains such as fraud detection, illness diagnosis, etc., where it is not enough to obtain the right classification and the users wish to gain insight into the domain [5]. To solve this kind of problems, the learner's output needs to be comprehensible. In other situations where comprehensibility is not necessary, it will also be an advantage for classifiers because it will help in processes of refinement.

In this context, classifiers can be divided in two main groups: classifiers with no comprehensible output and classifiers with comprehensible output. In the first group we can find artificial neural networks, support vector machines, multiple classifiers,

D. Borrajo, L. Castillo, and J.M. Corchado (Eds.): CAEPIA 2007, LNAI 4788, pp. 90–99, 2007.

etc., that, due to their complexity and structure do not provide an explanation to the classification. The second group includes classifiers that focus on representation, such as decision trees and rule sets. Comprehensible methods are usually very dependent on the training data. That is to say, classifiers induced from slightly different samples of the same data set are very different in accuracy and structure [6]. As Turney found when working on industrial applications of decision tree learning, not only to give an explanation but the stability of that explanation is of capital importance: "the engineers are disturbed when different batches of data from the same process result in radically different decision trees. The engineers lose confidence in the decision trees even when we can demonstrate that the trees have high predictive accuracy" [14].

Decision trees have been chosen as paradigm with comprehensive output in this work. Since in a decision tree the explanation is given by its structure, if we want to obtain a convincing explanation we need a way to build structurally steady trees with small complexity. Multiple classifiers such as bagging and boosting [1][2][4][7][13] reduce the error rate, but, even when the used weak classifiers are decision trees, a set of them needs to be combined to make a decision on the whole, and, as a consequence, comprehensibility disappears. Domingos explained it very clearly in [5]: "while a single decision tree can easily be understood by a human as long as it is not too large, fifty such trees, even if individually simple, exceed the capacity of even the most patient". Domingos proposes Combined Multiple Models (CMM) algorithm [5] to extract explanation from bagging or any other multiple classifier.

We have developed a methodology for building classification trees based on several subsamples, Consolidated Trees' Construction Algorithm (CTC), which is less sensitive to changes in the training set from a structural point of view. Therefore the classification is contributed with a more steady explanation. The aim of this work is to show that CTC algorithm can be used to extract explanation from bagging achieving better results than CMM from three points of view: accuracy, complexity of the built classifiers and stability in explanation.

The paper proceeds describing the two alternatives used to extract explanation from bagging, CTC and CMM, in Section 2. Details about the experimental methodology are described in Section 3. In Section 4 we present an analysis of the experimental results: comparison in accuracy, complexity and structural stability of CTC and CMM algorithms. Finally Section 5 is devoted to show the conclusions and further work.

2 Two Alternatives to Extract Explanation from Bagging

The alternatives we are going to compare in this work, CTC and CMM, propose different strategies to combine the knowledge of the *m* classifiers used in bagging in a single one in order to maintain the explaining capacity of the final classifier.

2.1 CMM Algorithm

CMM proposes to recover the comprehensibility loss in bagging using the learning algorithm to model the produced data partitioning. The learning is done from randomly generated examples that are classified using the bagging. Finally these

examples will be used to build a classifier with comprehensible output. CMM is a general algorithm that can be used with different learners and ensemble methods but in this work we will use it to extract explanation from bagging when the selected learning algorithm is classification trees, specifically C4.5 release 8 of Quinlan [12]. The knowledge of this multi-classifier will be transmitted to CMM using it to artificially generate and label the examples that will be used to build it.

Algorithm 1 shows Domingo's CMM proposal adapted to the concrete implementation. *N_S* bootstrap samples are extracted from *S*, the original training set, and one C4.5 tree is built from each of them. *n* new examples are generated using the probability distribution implicit in the generated C4.5 trees (*n/N_S* examples from each component C4.5 tree). The corresponding class (*c*) is assigned to each example based on the class the bagging of all the generated C4.5 trees assigns them ($c = bagging_{M1,\ldots,MN\_S}(x)$). This way, the examples will be representative of the combination of basic classifiers. The CMM classifier will be the C4.5 tree built from the new sample obtained adding the *n* randomly generated examples to the original training set.

Algorithm 1. CMM Algorithm for bagging and C4.5

Inputs:
S training set
C4.5 classifier with comprehensible output
bagging procedure for combining models
N_S (*Number_Samples*) number of component models to generate
n number of new examples to generate

Procedure CMM (*S, C4.5, bagging, N_S, n*)

for $i := 1$ to *N_S*
  Let S^i be a bootstrap sample of *S*
  Let M^i be the model produced by applying *C4.5* to S^i
end for

for $j := 1$ to *n*
  Let *x* be the randomly generated example
  Let *c* be the class assigned to *x* by $bagging_{M1,\ldots,MN\_S}(x)$
  Let $T = T \cup \{(x,c)\}$
end for

  Let *M* be the model produced by applying *C4.5* to $S \cup T$

2.2 CTC Algorithm

CTC algorithm was created to solve a fraud detection problem where the class distribution needed to be changed and explanation was required. In this kind of problems, classification trees built from different subsamples are very different in structure and accuracy. CTC draws together the information of a set of subsamples building a single tree [10]. The structural consensus is achieved at each step of the tree's building process. The different subsamples are used to make proposals about

the feature that should be used to split in the current node, and, only one feature is selected. The repetition of this process in every node leads to the construction of a single tree. In order to make the CTC comparable to CMM, the split function used is the gain ratio criterion (the same used by Quinlan in C4.5 [12]). The iterative process is described in Algorithm 2.

CTC algorithm uses several subsamples to induce a single tree, therefore, if we want to use it to explain the classification made by bagging, the same subsamples used for bagging will be used to build the CT tree.

The algorithm starts extracting a set of subsamples (*N_S*) from the original training set. The subsamples are obtained based on the desired resampling technique. In this case the bootstrap samples used for bagging will be used. LS^i contains all the data partitions created from each subsample S^i. When the process starts, the only existing partitions are the bootstrap subsamples of bagging. The pair $(X,B)^i$ is the split proposal for the first data partition in LS^i. *X* is the feature selected to split and *B* indicates the proposed branches or criteria to divide the data in the current node. In the consolidation step, X_c and B_c are the feature and branches selected by a voting process among all the proposals. The process is repeated while LS^i is not empty. The Consolidated Tree's generation process finishes when, in the last subsample in all the partitions in LS^i, most of the proposals are not to split it, so, to become it a leaf node. When a node is consolidated as a leaf node, the a posteriori probabilities associated to it are calculated averaging the a posteriori obtained from the data partitions related to that node in all the subsamples. Once the consolidated tree has been built it works the same way a decision tree does.

Algorithm 2. CTC Algorithm

Generate *N_S bootstrap* samples (S^i) from *S*
CurrentNode := *RootNode*
for $i := 1$ to *N_S*
  $LS^i := \{S^i\}$
end for
repeat
  for $i := 1$ to *N_S*
    $CurrentS^i := First(LS^i)$
    $LS^i := LS^i - CurrentS^i$
    Induce the best split $(X,B)^i$ for $CurrentS^i$
  end for
  Obtain the consolidated pair (X_c,B_c) based on $(X,B)^I$, $1 \le i \le N\_S$
  if $(X_c,B_c) \neq Not\_Split$
    Split *CurrentNode* based on (X_c,B_c)
    for $i := 1$ to *N_S*
      Divide $CurrentS^i$ based on (X_c,B_c) to obtain *n* subsamples $\{S_1^i, \ldots, S_n^i\}$
      $LS^i := \{S_1^i, \ldots, S_n^i\} \cup LS^i$
    end for
  else consolidate *CurrentNode* as a leaf
  end if
CurrentNode := *NextNode*
until $\forall i$, LS^i is empty

3 Experimental Methodology

Eleven databases of real applications from the UCI Repository benchmark [9] have been used for the experimentation: *Breast-W, Iris, Heart-C, Glass, Segment, Voting, Lymph, Hepatitis, Hypo, Soybean-L,* and *KDDcup99*. For *KDDcup99*, in order to reduce the experimentation cost, we have used a stratified sample of 4,941 examples where the number of classes has been reduced to two (attack / not attack). The used domains have a wide range of characteristics: the number of patterns goes from 148 to 4,941; the number of features from 4 to 41; and the number of classes from 2 to 15.

The validation methodology used in this experimentation has been to execute 5 times a 10-fold stratified cross validation [8]. In each of the folds of the cross-validation we have obtained 200 bootstrap samples. These subsamples have been used to explore the effect of the *N_S* parameter (12 values: 3, 5, 10, 20, 30, 40, 50, 75, 100, 125, 150 and 200) in the particular implementation of bagging. So in each one of the 50 folds of the cross validations 12 bagging classifiers have been built. We have selected for each database the value of *N_S* that minimizes error rate. Once this parameter has been fixed it has been used to build CT trees and CMM classifiers and compare them from two points of view: accuracy and quality of the explanation. The quality of explanation has been evaluated based on complexity and stability of the given explanation. Complexity has been measured as the number of internal nodes of the tree, and, the stability in explanation as structural stability of the tress which has been measured by *Common* parameter (number of identical nodes —level, variable and division— among two trees). *Common* has been normalized in respect to the complexity so that the parsimony principle was taken into account. We will call this measure *%Common* and it will quantify the identical fraction of two or more trees [11].

In both cases, error and explanation, an analysis of the statistically significant differences has been done based on the recent work of Demšar [3].

For building CMM classifiers the number of randomly generated examples (*n*) needs to be fixed. Taking into account the process used to generate examples (*n/N_S* examples are generated from each component C4.5 tree) and that the number of component C4.5 trees goes from 3 to 200, this number needs to be large enough to generate a minimum set of examples from each one of the C4.5 trees and, as the original sample is added to these examples to build the CMM, it also needs not to be too small compared to it. Domingos generated 1,000 artificial examples but the databases used for the experimentation were smaller than the ones used in our experimentation. As a consequence, the number has been fixed to max (1,000; (*NPT* * 1.5)) being *NPT* the number of patterns of the training set.

4 Experimental Results

CTC and CMM algorithms have been compared from three points of view: error, complexity, and structural stability (measured based on *Common* and *%Common*). From a practical point of view, the complexity quantifies how simple the given explanation is, *Common* and *%Common* quantify structural stability of the trees, whereas the error would quantify the “quality” of the explanation given by the tree. Evidently

an improvement in comprehensibility must be supported with a reasonable error rate. As a consequence, we will start the comparison from the accuracy point of view.

4.1 Discriminating Capacity

In the first step, the effect of *N_S* in bagging has been analyzed so that for each database the best number of samples could be selected. Table 1 shows the obtained results. We can observe that even if the use of several basic classifiers provides bagging with stability, the results are not exactly the same for different values of *N_S*. Minimum error rates for each database are marked in bold. It can be observed that the smallest average error is achieved when *N_S* is 100. The values of *N_S* obtaining best results have been selected to build CTC and CMM classifiers.

Table 1. Error values for bagging in 11 databases and different values of *N_S*

	Bagging											
N_S	**03**	**05**	**10**	**20**	**30**	**40**	**50**	**75**	**100**	**125**	**150**	**200**
Breast-w	5.61	5.24	5.35	4.64	4.84	4.78	4.78	4.75	4.64	**4.61**	4.64	4.70
Iris	5.87	6.40	5.87	6.53	6.13	5.87	5.73	5.47	5.47	**5.33**	5.47	5.33
Heartc	24.34	23.41	21.91	21.58	21.97	21.50	21.44	20.91	**20.39**	20.53	20.53	20.46
Glass	29.68	27.47	26.45	24.59	24.69	24.37	23.70	23.62	**23.13**	24.18	23.71	23.85
Segment	3.75	3.18	2.80	2.64	2.57	2.48	2.47	**2.38**	2.40	2.42	2.39	2.43
Voting	4.05	3.82	3.73	3.50	3.59	3.54	3.50	3.59	3.45	3.46	**3.41**	3.50
Lymph	21.10	20.77	20.29	19.38	18.85	19.56	19.41	19.07	18.82	19.17	19.17	**18.60**
Hepatitis	20.11	18.05	17.80	17.88	17.89	18.36	17.37	16.96	16.97	17.08	**16.61**	16.83
Hypo	0.82	**0.75**	0.78	0.77	0.75	0.76	0.76	0.76	0.76	0.75	0.76	0.75
Soybean_large	13.24	11.24	10.41	10.00	9.17	9.38	9.45	9.24	**9.03**	9.10	9.10	9.38
kddcup	0.37	0.31	0.36	0.33	0.32	0.32	0.30	0.32	0.30	0.28	0.28	**0.28**
Average	11.72	10.97	10.52	10.17	10.07	10.08	9.90	9.73	**9.58**	9.72	9.64	9.65

Before starting with the comparison of CTC and CMM we compare in Table 2 error rates achieved for both algorithms, C4.5 (as base classifier) and bagging, so that we situate CTC and CMM in respect to them. Results in Table 2 show that as we expected bagging is the algorithm achieving the smallest error rates, whereas the largest ones are achieved with C4.5. CTC and CMM are situated among these two algorithms, being the error of CMM slightly smaller in average.

The multiple test proposed by Demšar in [3] has been used to deeper analyse the differences among the four algorithms. With this aim, we need to rank each algorithm for every database. Average rank values are: bagging (1.36), CTC (2.50), C4.5 (3.05) and CMM (3.09). Even if in results in Table 2 we could see that in average CMM achieves smaller error rate than CTC, this was an effect of the average. Rank values show that, if instead of analysing absolute values we analyse the rank, CTC is in second position, whereas CMM is in the 4th one, even behind C4.5.

Table 2. Error values for C4.5, bagging, CTC and CMM in 11 databases. *N_S* fixed based on best results obtained for bagging.

	C4.5	N_S	Bagging	CMM	CTC
Breast-w	5.63	125	4.60	5.26	5.60
Iris	5.75	125	5.33	5.34	4.14
Heartc	23.96	100	20.39	22.85	23.42
Glass	31.55	100	23.13	28.16	29.85
Segment	3.24	75	2.38	3.27	3.33
Voting	3.41	150	3.41	3.69	3.36
Lymph	20.44	200	18.60	20.77	20.19
Hepatitis	20.29	150	16.61	18.51	20.95
Hypo	0.71	5	0.75	0.78	0.73
Soybean_large	11.02	100	9.03	11.57	10.67
kddcup	0.46	200	0.28	0.50	0.46
Average	**11.50**		**9.50**	**10.97**	**11.16**

Next step is to analyse whether significant differences among the 4 algorithms exist using Friedman test [3]. The critical value (α=0.05) is 2.9223 and the achieved value has been F_F= 6.3293**.** As a consequence significant differences exist. And we need to use a post-hoc test. We will use Nemenyi test [3] to compare all classifiers to each other and Bonferroni-Dunn test [3] to compare all classifiers with a control classifier (C4.5 in our work). Nemenyi test (for 4 algorithms, 11 databases and α=0.05) says that if the difference among average ranks is smaller than 1.4142 there is not significant difference among the compared algorithms, whereas the critical value for Bonferroni-Dunn is 1.3179.

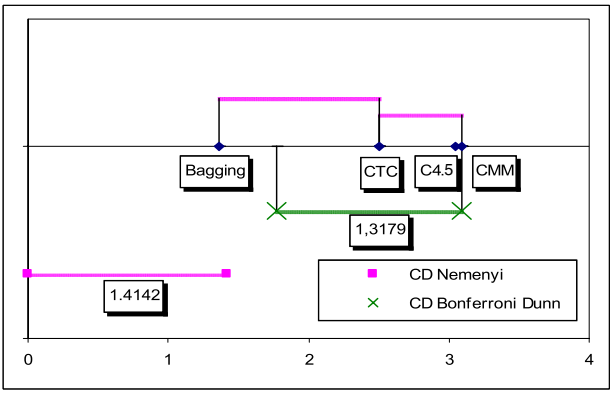

Fig. 1. Results for Nemenyi and Bonferroni-Dunn for C4.5, bagging, CTC and CMM

Figure 1 shows graphically results for these tests based on CD ((*Critical Difference*)) diagrams. In Nemenyi's test (upper line in the figure) two algorithms are connected by a line if no significant differences exist whereas for Bonferroni-Dunn test (lower line in the figure) there are significant differences with the control classifier only if the corresponding point is outside the line. Graphs show that based on both kind of tests, there are not significant differences among bagging and CTC (there are significant differences among bagging and the rest of the algorithms) but neither among CTC and the rest of the algorithms.

To make a deeper analysis of CTC and CMM we will make the statistical tests proposed by Demšar for two algorithms (Sign Test and Wilcoxon Signed-Ranks Test). We can use the signs obtained when calculating relative improvements of CTC with respect to CMM presented in Table 3 for the Sign Test. Results in Table 3 show that CTC has smaller error than CMM in 6 databases out of 11 (in bold) obtaining an average relative improvement of 2.28%. However, none of the two tests finds significant differences (α=0.05) in the set of 11 databases we have used (for the Sign test the statistic should be at least 9 and it should be smaller than 10 for Wilcoxon test). As a consequence we could say that even if in accuracy the behaviour of CTC and CMM is similar, it is slightly better for CTC.

Table 3. Relative improvement of CTC with respect to CMM for each database; average relative improvement and values of the statistics of the Sing Test and Wilcoxon Test

	CTC - CMM
Breast-w	6.46%
Iris	**-22.46%**
Heartc	2.52%
Glass	6.00%
Segment	1.89%
Voting	**-8.84%**
Lymph	**-2.81%**
Hepatitis	13.18%
Hypo	**-6.15%**
Soybean_large	**-7.73%**
kddcup	**-7.20%**
Average	**-2.28%**
Sign Test	6
Wilcoxon Test	31

4.2 Explaining Capacity

Comprehensibility or explaining capacity of classification trees and quality of the given explanation can be measured by complexity and stability (*common* & *%common*). These values are shown in Table 4. If we analyse the complexity, results show that in 10 databases out of 11, trees obtained with CTC are simpler than the ones obtained with CMM with a relative average improvement of 32.08%. Similar

behaviour is found when analysing the stability. In most of the databases *Common* and *%Common* are larger for CTC than for CMM (with average relative improvements of 36.33% and 118.41%). If we look to the statistics for the Sign Test and Wilcoxon Test (in the two last rows in Table 4) we find that there are significant differences in favour of CTC in the three parameters. So, we can state that from the explanation point of view CT classifiers are simpler and more stable than CMM classifiers.

Table 4. Explanation related values: Complexity, *Common* and *%Common* for CTC and CMM

	Complexity		**Common**		**%Common**	
	CTC	**CMM**	**CTC**	**CMM**	**CTC**	**CMM**
Breast-w	4.02	**3.52**	**2.71**	2.56	68.41	**74.10**
Iris	**3.42**	5.42	1.81	**2.87**	53.02	**53.15**
Heartc	**22.51**	34.54	**1.91**	1.55	**8.49**	4.47
Glass	**34.71**	54.14	**4.97**	3.56	**14.34**	6.58
Segment	**62.64**	100.24	**14.79**	12.95	**23.51**	12.94
Voting	**5.13**	6.74	**4.35**	3.11	**84.74**	46.35
Lymph	**12.33**	21.74	**4.12**	2.29	**33.28**	10.56
Hepatitis	**10.16**	28.30	**2.14**	1.57	**21.04**	5.57
Hypo	**5.98**	7.38	**2.83**	2.52	**48.44**	34.72
Soybean_large	**25.98**	39.36	**9.42**	4.52	**36.25**	11.48
kddcup	**15.80**	25.42	**5.42**	3.04	**34.27**	11.82
Average	**18.43**	29.71	**4.95**	3.69	**38.71**	24.70
Sign Test		**10**		**10**		**9**
Wilcoxon Test		**1**		**5**		**4**

5 Conclusions

Being aware of the importance of classifiers to be comprehensible when using machine learning to solve real world problems, we propose in this work CTC algorithm as an alternative to the Combined Multiple Models (CMM) algorithm proposed by Domingos to extract explanation from bagging. We have compared both proposals from three points of view: accuracy, complexity of the built classifiers and stability in explanation.

From the experimental results we can conclude that it is recommendable the use of CTC rather than the use of CMM. From the accuracy point of view, the behaviour of both algorithms, CTC and CMM, is similar, although the behaviour of CTC is nearer to bagging's one than the behaviour of CMM. Based on Demšar proposal for statistical analysis we can say that there are not significant differences among bagging and CTC whereas these differences exist if we compare bagging with CMM or C4.5. After analysing the complexity of both kinds of trees, we can say that CT trees will give simpler and as a consequence more comprehensible explanation than CMM classifiers. We could say this explanation is 32.08% simpler. And besides, looking to how steady the structure of the built trees is, we could say that the explanation fraction maintained common in Consolidated Trees at least twice as big as the one

maintained CMM classifiers. As a consequence, the explanation given to bagging using CTC will be simpler and more stable than the one given by CMM.

There are many things that can be done in the future related to this work. Firstly the experimentation can be extended to more databases. The way the classification is made in CTC can be changed: it can be seen as a multiple classifier system that classifies the same way bagging does but with a single structure. Related to the measure of stability in explanation, other structural measures can be tried.

Acknowledgments

This work was partly funded by the Diputación Foral de Gipuzkoa and the E.U.

The *lymphograph*y domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

References

1. Bauer, E., Kohavi, R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. Machine Learning 36, 105–139 (1999)
2. Breiman, L.: Bagging Predictors. Machine Learning 24, 123–140 (1996)
3. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)
4. Dietterich, T.G.: An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. Machine Learning 40, 139–157 (2000)
5. Domingos, P.: Knowledge acquisition from examples via multiple models. In: Proc. 14th International Conf. on Machine Learning Nashville, TN, pp. 98–106 (1997)
6. Drummond, C., Holte, R.C.: Exploiting the Cost (In)sensitivity of Decision Tree Splitting Criteria. In: Proc. of the 17th Int. Conf. on Machine Learning, pp. 239–246 (2000)
7. Freund, Y., Schapire, R.E.: Experiments with a New Boosting Algorithm. In: Proceedings of the 13th International Conference on Machine Learning, pp. 148–156 (1996)
8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, Heidelberg (2001)
9. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. University of California, Department of Information and Computer Science, Irvine, CA (1998), http://www.ics.uci.edu/ mlearn/MLRepository.html
10. Pérez, J.M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I., Martín, J.I.: Combining multiple class distribution modified subsamples in a single tree. Pattern Recognition Letters 28(4), 414–422 (2007)
11. Pérez, J.M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I., Martín, J.I.: Consolidated Trees: an Analysis of Structural Convergence. In: Williams, G.J., Simoff, S.J. (eds.) Data Mining. LNCS (LNAI), vol. 3755, pp. 39–52. Springer, Heidelberg (2006)
12. Quinlan, J.R. (eds.): C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Mateo, California (1993)
13. Skurichina, M., Kuncheva, L.I., Duin, R.P.W.: Bagging and Boosting for the Nearest Mean Classifier: Effects of Sample Size on Diversity and Accuracy. In: Roli, F., Kittler, J. (eds.) MCS 2002. LNCS, vol. 2364, pp. 62–71. Springer, Heidelberg (2002)
14. Turney, P.: Bias and the quantification of stability. Machine Learning 20, 23–33 (1995)