Shared Ensemble Learning using Multi-trees ⋆

V. Estruch C. Ferri J. Hernández-Orallo M.J. Ramírez-Quintana

DSIC, UPV, Camino de Vera s/n, 46020 Valencia, Spain.
{vestruch,cferri,jorallo,mramirez}@dsic.upv.es

Abstract. Decision tree learning is a machine learning technique that allows accurate and comprehensible models to be generated. Accuracy can be improved by ensemble methods which combine the predictions of a set of different trees. However, a large amount of resources is necessary to generate the ensemble. In this paper, we introduce a new ensemble method that minimises the usage of resources by sharing the common parts of the components of the ensemble. For this purpose, we learn a decision multi-tree instead of a decision tree. We call this new approach shared ensembles. The use of a multi-tree produces an exponential number of hypotheses to be combined, which provides better results than *boosting/bagging*. We performed several experiments, showing that the technique allows us to obtain accurate models and improves the use of resources with respect to classical ensemble methods.
Keywords: Decision-tree learning, Decision support systems, Boosting, Machine Learning, Hypothesis Combination, Randomisation.

1 Introduction

From the different machine learning approaches which are currently being applied with successful results, decision tree learning [16] is considered to be a paradigm with an optimal trade-off between the quality and the comprehensibility of the models learned.

A method that has recently been exploited to improve the accuracy of simple classifiers consists in the combination of a set of hypotheses (or ensemble) [3]. Well-known techniques for generating and combining hypotheses are boosting [8, 18], bagging [1, 18], randomisation [4], stacking [19] and windowing [17]. Although accuracy is significantly increased, "the large amount of memory required to store the hypotheses can make ensemble methods hard to deploy in applications"[12]. One way to partially overcome this limitation could be to share the common parts of the components of the ensemble.

In a previous work [5], we presented an algorithm for the induction of decision trees which is able to obtain more than one solution. To do this, once a node has been selected for splitting, the other possible splits at this point are suspended

⋆ This work has been partially supported by CICYT under grant TIC2001-2705-C03-01, Generalitat Valenciana under grant GV00-092-14, and Acción Integrada Hispano-Austriaca HU2001-19.

and stored until a new solution is required. This way, the search space is a multi-tree rather than a tree which is traversed thus producing an increasing number of solutions as the execution time increases. Since each new solution is built following a suspended node at an arbitrary place in the multi-tree, our method differs from other approaches such as the *boosting* method or the *bagging* method [1, 8, 18] which induce a new decision tree for each solution. Therefore, a multi-tree is not a forest [10] because a multi-tree shares the common parts of different trees (*shared ensemble*), whereas a forest is just a collection of trees.

Other works have attempted to generate a forest of 'different' trees, either semantically/vertically (by changing the weights of examples, e.g. boosting [8, 18], or the sample, e.g. bagging [1]) or syntactically/horizontally (by selecting attribute samples for each tree). Specifically, this latter approach has been presented independently by [9, 10], under the name pseudo-randomly selected feature subspaces, and by [20], under the name stochastic attribute selection committees. In both cases, the idea is to pseudo-randomly select a subset of attributes, to learn a first classifier, and then, to select another subset of attributes and learn a second classifier, and so on. Next, the elements from the set of decision tree classifiers (the forest) are combined. A related technique has been presented by Breiman in [2].

The main aim of both the horizontal and vertical approaches is to obtain better accuracy in the combination. There have also been attempts to combine horizontal and vertical approaches, such as the work from [21]. In [4], a randomised method has been introduced in the construction of the tree (random split criterion) and has been shown to be competitive w.r.t. boosting and bagging.

In this paper, we focus on the combination of hypotheses from the multi-tree approach in order to obtain accurate models. The use of this structure allows more hypotheses to be combined than in other combination methods using the same resources. Several hypothesis-fusion strategies are defined and evaluated experimentally. We also include a comparison between our approach and some well-known ensemble methods.

The paper is organised as follows. In Section 2, we introduce the multi-tree structure. Section 3 discusses different ways to combine the components of a shared ensemble. Section 4 presents several experiments showing that the approach effectively generates accurate models. Finally, Section 5 summarises and presents some future work.

2 Multi-tree structure

In this section, we present the multi-tree structure, and we discuss the different criteria required to construct it.

The construction of decision trees is performed in two different steps [17]:

- **Tree Construction:** The entire decision tree is constructed in this phase. The process is driven by a splitting criterion that selects the best split. The selected split is applied to generate new branches, and the rest of the splits are discarded. The algorithm stops when the examples that fall into a branch belong to the same class.

– **Pruning**. This phase consists in the removal of useless parts of the tree. There are two options: *pre-pruning*, when the process is performed during the construction of the tree, or *post-pruning*, when the pruning is performed by analysing the leaves once the tree has been built.

Thus, decision trees are built in an eager way, which makes it possible for the quick construction of a model. However, it may produce bad models because of bad decisions.

In [5], we have defined a new structure in which the rejected splits are not removed, but stored as *suspended* nodes. The further exploration of these nodes after the first solution is built permits the extraction of new models from this structure. For this reason, we call it a decision multi-tree, rather than a decision tree. Since each new model is obtained by continuing the construction of the multi-tree, these models share their common parts. A decision multi-tree can also be seen as an AND/OR tree [13, 15], if one considers the split nodes as being OR-nodes and considers the nodes generated by an exploited OR-node as being AND-nodes.

To populate a multi-tree, we need to specify a criterion that selects one of the suspended nodes. In [6], we presented and evaluated some possible criteria, such as topmost, bottom, or random. Our experimental results showed that random is a trade-off between speed and quality.

Once the multi-tree has been built, we can use it for two different purposes: to select one or n comprehensible models (decision trees) according to a selection criterion (Occam, MDL, ...), or to use the multi-tree as an ensemble whose components can be combined. In this paper, we address the latter.

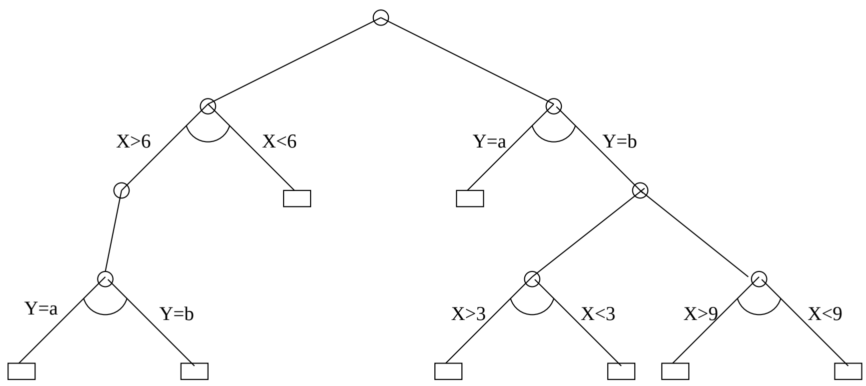

Fig. 1. Selection of a single decision tree from the multi-tree structure.

Figure 1 shows a decision multi-tree. OR-nodes are represented with an arc and leaves are represented by rectangles. Three different models are exhibited in this multi-tree since two suspended nodes have been exploited.

The decision multi-tree approach presents some interesting features. First, the number of solutions grows exponentially wrt. the number of suspended OR-nodes that are exploited. Secondly, the solutions share some of their parts. The percentage of the shared quantity depends on the depth of the suspended OR-node that is exploited. Exploring deep suspended nodes in the bottom areas of the multi-tree causes the generation of models that share many of their conditions; therefore they could be very similar. However, the exploration of OR-nodes in the top positions of the multi-tree produces solutions which are different enough.

3 Shared Ensemble Combination

In this section, we address the question of how to combine different solutions in a multi-tree. Given several classifiers that assign a probability to each prediction (also known as soft classifiers), there are several combination methods or fusion strategies. Let us denote by $p_k(c_j|x)$ an estimate of the posterior probability that classifier k assigns class c_j for example x. In decision tree learning, the $p_k(c_j|x)$ depends on the leaf node where each x falls. More precisely, these probabilities depend on the *proportion* of training examples of each class that have fallen into each leaf node during training. The reliability of each leaf usually depends on the *cardinality* of the leaf.

Let us define a *class vector* $v_{k,j}(x)$ as the vector of training cases that fall in each node k for each class j. For leaf nodes, the values would be the training cases of each class that have fallen into the leaf. To propagate these vectors upwards to internal nodes, we must clarify how to propagate through AND and OR nodes. This is done for each new unlabelled example we want to make a prediction for. For the OR-nodes, the answer is clear: an example can only fall through one of its children. Hence, the vector would be the vector of the child where the example falls. AND-nodes, however, must do a fusion whenever different alternate vectors occur. This is an important difference in shared ensembles: fusion points are distributed all over the multi-tree structure. Following [11], we have considered several fusion strategies that convert m class vectors into one combined vector Ω_j:

- **sum**: $\Omega_j = \sum_{k=1}^{m} v_{k,j}$
- **arithmetic mean**: $\Omega_j = \sum_{k=1}^{m} \frac{v_{k,j}}{m}$
- **product:** $\Omega_j = \prod_{k=1}^{m} v_{k,j}$
- **geometric mean:** $\Omega_j = \sqrt[m]{\prod_{k=1}^{m} v_{k,j}}$
- **maximum:** $\Omega_j = \mathtt{max}_k(v_{k,j})$
- **minimum:** $\Omega_j = \mathtt{min}_k(v_{k,j})$

There have been some studies to determine which strategy is better. In particular, [11] concludes that, for two-class problems, minimum and maximum are the best strategies, followed by average (arithmetic mean).

In addition, we have devised some transformations to be done to the original vectors at the leaves before its propagation:

- **good loser** : $v'_{k,j}(x) = \sum_j v_{k,j}(x)$ if $j = \texttt{argmax}(v_{k,j}(x))$ and 0 otherwise
- **bad loser:** $v'_{k,j}(x) = v_{k,j}(x)$ if $j = \texttt{argmax}(v_{k,j}(x))$ and 0 otherwise.
- **majority:** $v'_{k,j}(x) = 1$ if $j=$ $\texttt{argmax}(v_{k,j}(x))$ and 0 otherwise.
- **difference**: $v'_{k,j}(x) = v_{k,j}(x) - \sum_{i \neq j} v_{k,j}(x)$

For example, the following table shows the results of applying the transformations to two vectors.

Original	Good loser	Bad loser	Majority	Difference
{ 40, 10, 30 }	{ 80, 0, 0 }	{ 40, 0, 0}	{ 1, 0, 0 }	{ 0, -60, -20 }
{ 7, 2, 10 }	{ 0, 0, 19 }	{ 0, 0, 10 }	{ 0, 0, 1 }	{ -5, -15, 1 }

In the next section, we show an experimental evaluation of these fusion and transformation methods for problems with more than two classes.

4 Experiments

In this section, we present an experimental evaluation of our approach, as it is implemented in the SMILES system [7]. SMILES is a multi-purpose machine learning system which (among many other features) includes the implementation of a multiple decision tree learner.

For the experiments, we used GainRatio [17] as splitting criterion and we chose a random method [6] for populating the shared ensemble (after a solution is found, a suspended OR-node is woken at random). Pruning is not enabled.

The experiments were performed in a Pentium III-800Mhz with 180MB of memory running Linux 2.4.2. We have used several datasets from the UCI dataset repository [14]. Table 1 shows the dataset name, the size in number of examples, the number of classes and the number of nominal and numerical attributes.

Since there are many sources of randomness, we performed the experiments by averaging 10 results of a 10-fold cross-validation. This makes a total of 100 runs for each pair composed of a method and a dataset.

4.1 Evaluation of fusion and vector transformation techniques

Table 2 shows the mean accuracy and the standard deviation using the different fusion techniques introduced in Section 3 for each dataset. We summarise the results with the geometric means for each technique.

The techniques studied are *sum*, *product*, *maximum*, *minimum*, and *arithmetic mean*, all of which use the original vectors. In the table, we do not include the experiments with *geometric mean* because they are equivalent to the results of *product.* The multi-tree was generated by exploring 100 suspended OR-nodes, thus giving thousands of possible hypotheses (with much less required memory than 100 non-shared hypotheses). According to the experiments, the best fusion

#	Dataset	Size	Classes	Nom.Attr.	Num.Attr.
1	Balance-scale	625	3	0	4
2	Cars	1728	4	5	0
3	Dermatology	358	6	33	1
4	Ecoli	336	8	0	7
5	Iris	150	3	0	4
6	House-votes	435	2	16	0
7	Monks1	566	2	6	0
8	Monks2	601	2	6	0
9	Monks3	554	2	6	0
10	New-thyroid	215	3	0	5
11	Post-operative	87	3	7	1
12	Soybean-small	35	4	35	0
13	Tae	151	3	2	3
14	Tic-tac	958	2	8	0
15	Wine	178	3	0	13

Table 1. Information about datasets used in the experiments.

technique was *maximum*. Thus, we will use this fusion method to study the effect of applying the transformations on the vector.

Table 3 illustrates the results for accuracy using the *original* vector and the *good loser*, *bad loser*, *majority* and *difference* transformations. According to these experiments, all transformations get very similar results, except from *majority*. We will use the combination *max + difference* in the following experiments.

4.2 Influence of the size of the multi-tree

Let us study the influence of the size of the multi-tree, varying from 1 to 1,000 explored OR-nodes. Table 4 shows the accuracy obtained using the shared ensembles depending on the number of OR-nodes opened. The results indicate that the greater the population of the multi-tree the better the results of the combination are.

4.3 Comparison with other ensemble methods

Figure 2 presents an accuracy comparison among our method (*multi-tree*), the *boosting* method and the *bagging* method (*multi-tree* and *bagging* without pruning) for the mean of all the 15 datasets, depending on the number of iterations. We have employed the Weka[1] implementation of these two ensemble methods.

Although our method initially obtains lower results with few iterations, with a higher number of iterations it surpasses the other two systems. Probably the slow increase of accuracy in the multi-tree method is due to the random selection of the OR-nodes to be explored.

[1] http://www.cs.waikato.ac.nz/~ml/weka/

#	Arit. Acc.	Arit. Dev.	Sum. Acc.	Sum. Dev.	Prod. Acc.	Prod. Dev.	Max. Acc.	Max. Dev.	Min. Acc.	Min. Dev.
1	80.69	5.01	81.24	4.66	76.61	5.04	83.02	4.76	76.61	5.04
2	91.22	2.25	91.25	2.26	83.38	3.65	90.90	2.09	83.38	3.65
3	94.17	4.06	94.34	3.87	89.06	5.19	94.00	4.05	89.06	5.19
4	80.09	6.26	79.91	6.13	76.97	7.14	80.09	6.11	76.97	7.14
5	95.63	3.19	95.77	3.18	93.28	3.71	95.93	2.81	93.28	3.71
6	94.53	5.39	94.20	5.66	94.00	5.34	94.47	5.45	94.40	5.34
7	99.67	1.30	99.71	1.18	81.00	8.60	99.89	0.51	81.00	8.60
8	73.35	5.86	73.73	5.82	74.53	5.25	77.15	5.88	74.53	5.25
9	97.87	2.00	97.91	1.80	97.58	2.45	97.62	1.93	97.58	2.45
10	94.52	4.25	93.76	5.10	92.05	5.71	92.57	5.43	92.05	5.71
11	62.50	16.76	63.25	16.93	61.63	17.61	67.13	14.61	61.63	17.61
12	97.50	8.33	97.50	9.06	97.75	8.02	94.75	11.94	97.75	8.02
13	63.60	12.59	64.33	11.74	62.00	12.26	63.93	12.03	62.00	12.26
14	81.73	3.82	82.04	3.78	78.93	3.73	82.68	3.97	78.93	3.73
15	94.06	6.00	93.88	6.42	91.47	7.11	92.53	6.99	91.47	7.11
Geomean	85.83	4.72	85.99	4.71	82.53	5.93	86.40	4.52	82.55	5.93

Table 2. Comparison between fusion techniques.

Nevertheless, the major advantage of the method can be appreciated by looking at the consumption of resources. Figure 3 shows the average training time depending on the number of iterations (1-300) for the three methods. Note that, as expected, the time increase of *bagging* is linear. *Boosting* behaves better with high values because the algorithm implemented in Weka trickily stops the learning if it does not detect a significant increase in accuracy. Finally, SMILES presents a sub-linear increase of required time due to the sharing of common components of the multi-tree structure.

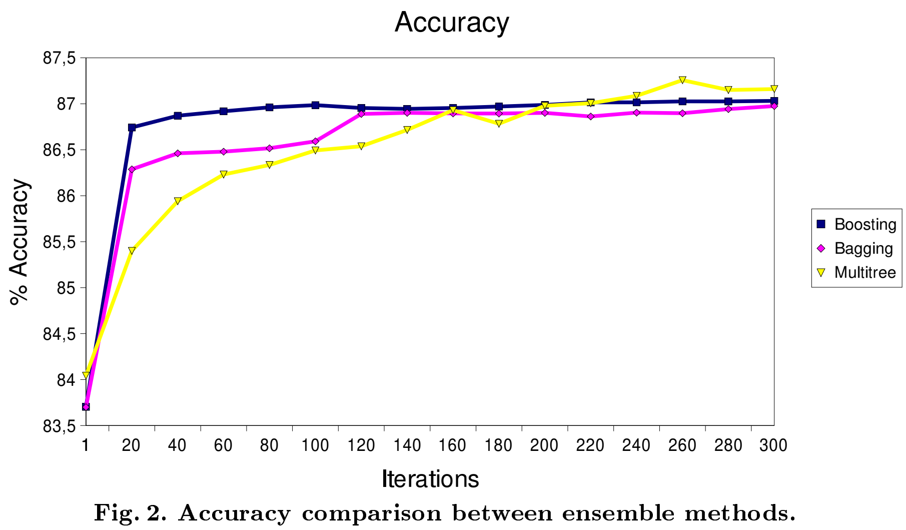

Fig. 2. Accuracy comparison between ensemble methods.

	Max + Orig		Max + Good		Max + Bad		Max + Majo.		Max + Diff.	
#	Acc.	Dev	Acc.	Dev	Acc.	Dev	Acc.	Dev	Acc.	Dev
1	83.02	4.76	83.02	4.76	83.02	4.76	67.84	6.61	83.02	4.76
2	90.90	2.09	90.90	2.09	90.90	2.09	81.48	3.22	90.90	2.09
3	94.00	4.05	94.00	4.05	94.00	4.05	79.97	7.98	94.00	4.05
4	80.09	6.11	80.09	6.11	80.09	6.11	78.21	6.07	80.09	6.11
5	95.93	2.81	95.93	2.81	95.93	2.81	89.44	4.84	95.93	2.81
6	94.47	5.45	94.47	5.45	94.47	5.45	91.47	6.90	94.47	5.45
7	99.89	0.51	99.89	0.51	99.89	0.51	77.58	6.29	99.89	0.51
8	77.15	5.88	77.15	5.88	77.15	5.88	83.42	5.06	77.15	5.88
9	97.62	1.93	97.62	1.93	97.62	1.93	90.40	4.02	97.62	1.93
10	92.57	5.43	92.57	5.43	92.57	5.43	89.14	6.74	92.57	5.43
11	67.13	14.61	67.13	14.61	67.13	14.61	68.25	15.33	67.00	14.60
12	94.75	11.94	94.75	11.94	94.75	11.94	50.75	28.08	94.75	11.94
13	63.93	12.03	63.87	12.14	63.93	12.03	60.93	11.45	65.13	12.53
14	82.68	3.97	82.68	3.97	82.68	3.97	68.26	4.35	82.68	3.97
15	92.53	6.99	92.53	6.99	92.53	6.99	78.41	11.25	92.53	6.99
Gmean	86.40	4.52	86.39	4.53	86.40	4.52	76.11	7.19	86.49	4.54

Table 3. Comparison between vector transformation methods.

5 Conclusions

This work has presented a novel ensemble method. The main feature of this technique is the use of a structure called multi-tree that permits sharing common parts of the single components of the ensemble. For this reason, we call it *shared ensemble*.

Several combination methods or fusion strategies have been presented, as well as class vector transformation techniques. The effectiveness of these methods has also been examined by an experimental evaluation. We have also investigated the importance of the size of the multi-tree w.r.t. the quality of the results obtained.

Finally we have compared the new ensemble method with some well-known ensemble methods, namely *boosting* and *bagging*. The accuracy results for the new method are quite encouraging: although our results are initially worse than the other two methods, when the number of iterations is increased, the new approach equals and even excels the other methods. Nevertheless, as we have shown, it is in the use of resources where the shared ensembles offer an important advance. Our system is very appropiate for complex problems where other ensemble methods such as *boosting* or *bagging* require huge amounts of memory and time.

As future work, we propose the study of a new strategy for generating trees. This strategy would be different from the current random technique we have employed to explore OR-nodes, and would probably be based on the semantic discrepancy of classifiers. This technique would provide a way to improve the results of our ensemble method with few iterations.

	1		10		100		1000	
#	Acc.	Dev.	Acc.	Dev.	Acc.	Dev.	Acc.	Dev.
1	76.82	4.99	77.89	5.18	83.02	4.76	87.68	4.14
2	89.01	2.02	89.34	2.20	90.90	2.09	91.53	2.08
3	90.00	4.72	91.43	4.67	94.00	4.05	94.00	4.05
4	77.55	6.96	78.58	6.84	80.09	6.11	80.09	6.11
5	93.63	3.57	94.56	3.41	95.93	2.81	95.56	2.83
6	94.67	5.84	94.27	5.69	94.47	5.45	95.00	5.14
7	92.25	6.27	96.45	4.15	99.89	0.51	100.00	0.01
8	74.83	5.17	75.33	5.11	77.15	5.88	82.40	4.52
9	97.55	1.89	97.84	1.86	97.62	1.93	97.75	1.92
10	92.62	5.22	93.43	5.05	92.57	5.43	90.76	5.89
11	60.88	17.91	63.00	15.88	67.00	14.60	68.13	15.11
12	97.25	9.33	96.00	10.49	94.75	11.94	95.50	10.88
13	62.93	12.51	65.00	12.19	65.13	12.53	65.33	12.92
14	78.22	4.25	79.23	4.03	82.68	3.97	84.65	3.34
15	93.12	6.95	93.29	6.31	92.53	6.99	92.99	5.00
Gmean	83.88	5.52	84.91	5.30	86.49	4.54	87.47	4.47

Table 4. Influence of the size of the multi-tree.

References

1. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
2. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
3. T. G Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.
4. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2):139–157, 2000.
5. C. Ferri, J. Hernández, and M.J. Ramírez. Induction of Decision Multi-trees using Levin Search. In *Int. Conf. on Computational Science, ICCS'02*, LNCS, 2002.
6. C. Ferri, J. Hernández, and M.J. Ramírez. Learning multiple and different hypotheses. Technical report, D.S.I.C., Universitat Politécnica de Valéncia, 2002.
7. C. Ferri, J. Hernández, and M.J. Ramírez. SMILES system, a multi-purpose learning system. `http://www.dsic.upv.es/~flip/smiles/`, 2002.
8. Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *the 13th Int. Conf. on Machine Learning (ICML'1996)*, pages 148–156, 1996.
9. Tim Kam Ho. Random decision forests. In *Proc. of the 3rd International Conference on Document Analysis and Recognition*, pages 278–282, 1995.
10. Tim Kam Ho. C4.5 decision forests. In *Proc. of 14th Intl. Conf. on Pattern Recognition, Brisbane, Australia*, pages 545–549, 1998.
11. Ludmila I. Kuncheva. A Theoretical Study on Six Classifier Fusion Strategies. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002.
12. Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *14th Int. Conf. on Machine Learning*, pages 211–218. Morgan Kaufmann, 1997.
13. N.J. Nilsson. *Artificial Intelligence: a new synthesis.* Morgan Kaufmann, 1998.
14. University of California. UCI Machine Learning Repository Content Summary. `http://www.ics.uci.edu/~mlearn/MLSummary.html`.

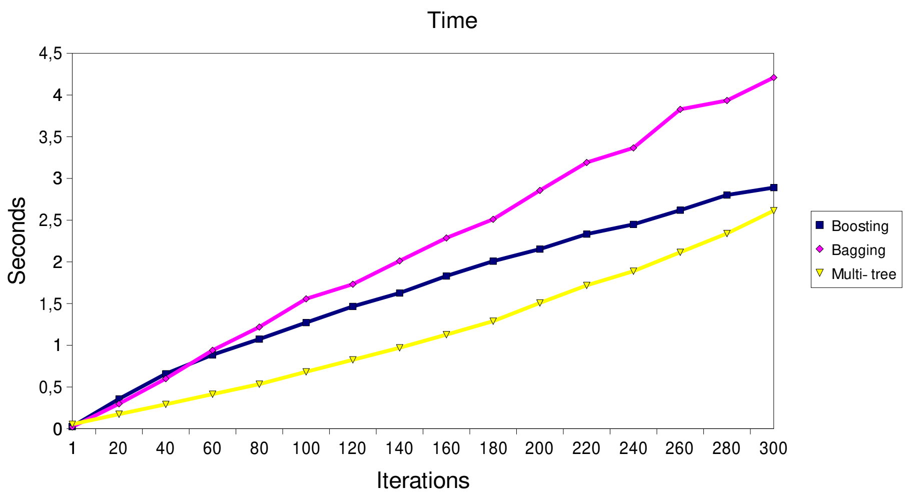

Fig. 3. Time comparison between ensemble methods.

15. J. Pearl. *Heuristics: Intelligence search strategies for computer problem solving.* Addison Wesley, 1985.
16. J. R. Quinlan. Induction of Decision Trees. In *Read. in Machine Learning*. M. Kaufmann, 1990.
17. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
18. J. R. Quinlan. Bagging, Boosting, and C4.5. In *Proc. of the 13th Nat. Conf. on A.I. and the 8th Innovative Applications of A.I. Conf.*, pages 725–730. AAAI/MIT Press, 1996.
19. David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.
20. Zijian Zheng and Geoffrey I. Webb. Stochastic attribute selection committees. In *Australian Joint Conference on Artificial Intelligence*, pages 321–332, 1998.
21. Zijian Zheng, Geoffrey I. Webb, and K.M. Ting. Integrating boosting and stochastic attribute selection committees for further improving the performance of decision tree learning. In *Proc. of 10th Int. Conf. on Tools with Artificial Intelligence (ICTAI-98), IEEE Computer Society Press*, pages 216–223, 1998.