Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition Letters 29 (2008) 1049–1059
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Boosting recombined weak classifiers ☆

Juan J. Rodríguez *, Jesús Maudes

University of Burgos, Department of Civil Engineering, Escuela Politecnica Superior. c/Francisco de Vitoria s/n, 09006 Burgos, Spain

Available online 19 July 2007

Abstract

Boosting is a set of methods for the construction of classifier ensembles. The differential feature of these methods is that they allow to obtain a *strong* classifier from the combination of *weak* classifiers. Therefore, it is possible to use boosting methods with very simple base classifiers. One of the most simple classifiers are *decision stumps*, decision trees with only one decision node.

This work proposes a variant of the most well-known boosting method, AdaBoost. It is based on considering, as the base classifiers for boosting, not only the last weak classifier, but a classifier formed by the last r selected weak classifiers (r is a parameter of the method). If the weak classifiers are decision stumps, the combination of r weak classifiers is a decision tree.

The ensembles obtained with the variant are formed by the same number of decision stumps than the original AdaBoost. Hence, the original version and the variant produce classifiers with very similar sizes and computational complexities (for training and classification). The experimental study shows that the variant is clearly beneficial.

Keywords: Boosting; Classifier ensembles; Decision stumps

1. Introduction

Given one or more classification methods, one of the most natural ways of obtaining more accurate classifiers is the use of ensembles (Kuncheva, 2004). There are methods that combine classifiers obtained with different methods. This is the case for Stacking (Wolpert, 1992). Nevertheless, it is also possible to combine classifiers obtained from the same method. Combining identical classifiers is useless. Hence, it is necessary to use some approach that allows to obtain different classifiers from the same method.

In Bagging (Breiman, 1996), each classifier is obtained from a different data set. Each data set is a sample, with replacement, from the original training data. Normally, the size of the sample is the same than the size of the original data set. The sets are different because in a sample some training examples will be selected several times, while other examples will not appear in that sample.

In the Random Subspaces method (Ho, 1998), each classifier is built using all the training examples, but with a random subset of the attributes.

There are some ensemble methods that have been designed specifically for combining classifiers obtained with methods from a certain kind, normally decision trees. Random Forests (Breiman, 2001) are a variant of Bagging, using Random Trees as base classifiers. In this type of random trees, the selection of the attribute for a decision node is done using only a random subset of the attributes. Another type of random trees is presented in (Dietterich, 2000).

Other ensemble methods designed specifically for combining decision trees are (Li and Liu, 2003; Rodríguez et al., 2006; Rodríguez and Maudes, 2006). There are methods that extend the concept of decision trees, using a

☆ This work was partially supported by Spanish Ministry of Education and Culture, through Grant DPI2005-08498 and Junta Castilla y León VA088A05.
* Corresponding author. Fax: +34 947 25 89 10.
E-mail addresses: jjrodriguez@ubu.es (J.J. Rodríguez), jmaudes@ubu.es (J. Maudes).

0167-8655/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2007.06.019

unique structure that is equivalent to several decision trees. Among these methods are Option Trees (Buntine, 1993; Kohavi and Kunz, 1997) and Multitrees (Estruch et al., 2003).

One of the most successful ensembles methods is Boosting (Schapire, 1999, 2002). There are several variants, AdaBoost is the most well-known. These methods assign a weight to each example. Initially, all the examples have the same weight. In each iteration a new classifier, named *base* or *weak*, is constructed using the base learning method. The construction of the base classifier must take into account the weights distribution. Then, the weight of each example is adjusted, depending on the correctness of the prediction of the base classifier for that example. The final classification is obtained from a weighted vote of the base classifiers.

In order to say that an ensemble method is a boosting method, it must have the ability of generating a *strong* classifier from a *weak* method. It is said that a method is *weak* if it can guarantee that for a two classes data set the obtained classifier will have a training error smaller than 50% (Schapire, 1990).

One of the weakest classifiers are decision stumps, decision trees with only one decision. They are commonly used as base classifiers for Boosting (Schapire and Singer, 1998; Friedman et al., 2000; Reyzin and Schapire, 2006). A recent exhaustive comparison of several classification methods (Caruana and Niculescu-Mizil, 2006) includes results for boosted decision stumps.

This paper presents an approach for improving the results obtained with boosting and decision stumps. The idea is to combine several decision stumps in a not so weak tree.

The rest of the paper is organised as follows. The proposed variant of AdaBoost is presented in Section 2. Sections 3 and 4 are dedicated to the experimental validation. Finally, Section 5 concludes.

2. Combination of weak classifiers

2.1. AdaBoost

For the sake of self-containment, Fig. 1 shows the AdaBoost algorithm. Each example x_i is from a domain $\mathcal{X}$ and has an associated binary label y_i. In this method the binary labels are codified as $+1$ y -1.

AdaBoost associates a weight for each example. $D_t(x_i)$ is the weight for x_i in iteration t. The method generates a base classifier h_t, taking into account the weights distribution. A real value, α_t, is selected. It is the weight associated to h_t, and it depends on the training error of that classifier. Then, weights are readjusted.

2.2. Classifier example

Fig. 2 shows the classifier obtained with AdaBoost, using decision stumps as base classifiers. This classifier is for the *Sonar* data set (Blake and Merz, 1998). It has two classes, "rock" and "mine", and 60 numeric attributes.

The number of decision stumps in that classifier is five. Given that AdaBoost is an iterative method, these five decision stumps are also the five first weak classifiers constructed when using AdaBoost with more iterations.

The classifier is used in the following way. The example is classified by the five trees. Each tree has a numerical vote. The absolute value of the vote is given by the weight of the tree. The sign of the vote depends of the class predicted by the tree. The prediction of the ensemble is the sign of the sum of the votes of the trees.

2.3. Proposed method

In order to improve the accuracy of a classifier obtained with AdaBoost and a learning method, there are two direct approaches:

Given $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/(m)$
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t
- Get base classifier $h_t : \mathcal{X} \to \mathbb{R}$
- Choose $\alpha_t \in \mathbb{R}$
- Update:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Fig. 1. AdaBoost (Schapire, 1999, 2002).

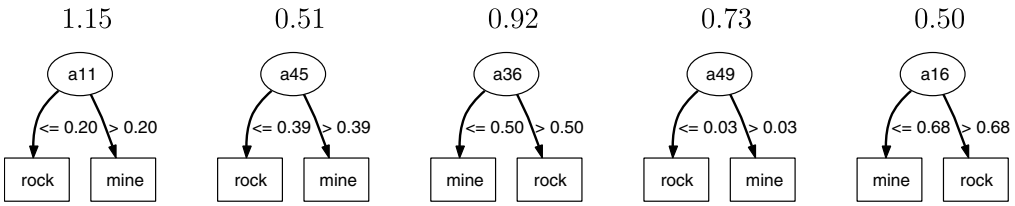

Fig. 2. Example of a classifier ensemble obtained with AdaBoost. The weight of each decision stump appears above the tree.

- To add more weak classifiers.
- To use more complex base classifiers.

These approaches have the following drawbacks: the construction of the classifier is slower, the use of the classifier is also slower, the classifier needs more memory.

Therefore, it would be interesting to have the possibility of obtaining more accurate classifiers without increasing neither the computational complexity nor the memory requirements.

The method proposed in this work is based on combining several weak classifiers in a not so weak classifier. For instance, if the first three classifiers from Fig. 2 are combined, the result is the tree from Fig. 3. In order to determine the class associated to each leaf of the tree, it is necessary to pass each training example through the tree. The class associated to a leaf is the class of the majority of the examples in the leaf.

In the proposed variant, each time a new decision stump is constructed, a tree is obtained from that decision stump and the decision stumps from previous iterations. The method has a parameter, r, the level of reuse. It is the number of classifiers from the former iterations that are going to be used. For instance, if $r = 2$ and the number of boosting iterations is five, the ensemble would be formed by five trees. The first one would have only one decision (and two leaves), the second would have two decisions (and four leaves) and the rest would have three decisions (and eight leaves).

Fig. 4 shows the proposed variant of AdaBoost. The only difference is that instead of using directly the classifier h_t, it is combined with the r previous classifiers. The resulting classifier is called h_t^r.

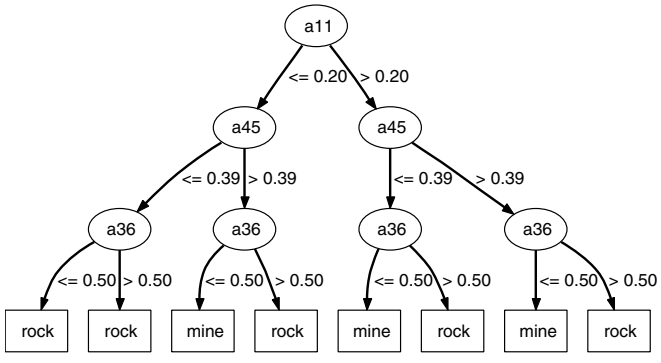

Fig. 3. A decision tree obtained from the combination of three decision stumps.

2.4. Some implementation details

It is not necessary to store explicitly the combined trees, because they can be obtained from the decision stumps and the reuse level r. For each tree, it is only necessary to associate which class is predicted in each leaf. For two classes problems, a bit per leaf is enough. Given a reuse level r, the number of leaves in each tree will be 2^{r+1}. For instance, if $r = 2$ only a byte per tree is necessary. For the tree in Fig. 3 the byte is 00101010 (or 11010101 if the correspondence between classes and digits were the opposite).

The computational cost of the classification of an example is nearly the same for the original version and the reuse version. In both cases, the example is classified by the same number of decision stumps. In the reuse version it is necessary to determine which leaf of the current tree is associated to the example. It is only necessary to keep in memory the binary classifications given by the $r + 1$ decision stumps, thus an integer variable with $r + 1$ bits is enough. This variable will be initialized to 0. Each time a decision stump classifies the instance, the value of the variable is shifted one bit to the left (that is, it is multiplied by 2). With the shift, the most significant bit is lost (if the variable has more than $r + 1$ bits, it is necessary to put this bit to 0). Then, the least significant bit is set to the classification given by the last decision stump (that is, the variable is incremented by the binary output of the decision stump).

For instance, if $r = 2$, the value of the variable with the outputs of the previous decision stumps is 110 and the current decision stump classifies the instance as 1, the new value of the variable would be 101. The leaf associated to the instance would be the number 5 (the leaves are numerated from 0 to 7).

In the construction of a classifier with AdaBoost, it is necessary to classify the training examples with the current classifiers. The weights of each training example are readjusted depending on the correctness of the classification of the last classifiers. Hence, in the reuse variant of AdaBoost, it is necessary to classify each training instance with every base classifier. This classification can be done with the method described above. Each training instance will have an associated integer of $r + 1$ bits with the outputs of the last decision stumps. The value of this integer indicates the leaf associated to the instance in the current tree.

Given $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/(m)$
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t
- Get base classifier $h_t : \mathcal{X} \to \mathbb{R}$
- Get combined classifier h_t^r from $h_t, h_{t-1}, \ldots h_{\max(t-r,1)}$
- Choose $\alpha_t \in \mathbb{R}$
- Update

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t^r(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t^r(x)\right)$$

Fig. 4. AdaBoost variant with classifiers reuse.

For each tree, the class and $r+1$ bits of each training example are used to determine the label of each leaf. A frequency matrix is used, its dimensions are $2^{r+1} \times 2$. The element (i,j) of the matrix stores the sum of the weights of the examples in the leaf i that are from the class j. This matrix can be updated while classifying by the last decision stump each training example. The class assigned to the leaf i is the value of j that maximizes the value of the matrix at (i,j).

3. Experimental validation

The description of the used data sets appears in Table 1. They are from the UCI Repository (Blake and Merz, 1998). For each data set and considered method, 10-fold cross validation was done 10 times.

In AdaBoost, the base learner must take into account the distribution of weights of the examples. This can be done in two ways (Freund and Scapire, 1997). In the reweighting version, the base learner is trained with the weighted training data. In the resampling version, the base learner is trained with a sample of the training data. The latter is necessary when the base learner is not able to handle weighted examples. In this work, the two versions are considered.

For the number of iterations of the boosting algorithm, three values were considered: 10, 25 and 100. With the respect to the reuse level, three variants were considered: AdaBoost (the version without reuse), AdaBoost-$r1$ (reuse version with $r = 1$) and AdaBoost-$r2$. There is a reason for not considering higher reuse levels. Successful ensembles are obtained from accurate but diverse classifiers (Kuncheva, 2005). Clearly, augmenting the reuse level reduces the diversity of the classifiers.

Other methods, such as AdaBoost with full trees, were not considered because it was desired to compare methods with a similar computational complexity. In the reuse variant of AdaBoost the number of classifiers and their complexity is the same than for the variant without reuse.

Table 1
Data sets

Data set	Classes	Examples	Attributes	
			Discrete	Continuous
Anneal	6	898	32	6
Audiology	24	226	69	0
Autos	7	205	10	16
Balance-scale	3	625	0	4
Breast-cancer	2	286	10	0
Cleveland-14-heart	2	303	7	6
Credit-rating	2	690	9	6
German-credit	2	1000	13	7
Glass	7	214	0	9
Heart-statlog	2	270	0	13
Hepatitis	2	155	13	6
Horse-colic	2	368	16	7
Hungarian-14-heart	2	294	7	6
Hypothyroid	4	3772	22	7
Ionosphere	2	351	0	34
Iris	3	150	0	4
Labor	2	57	8	8
Letter	26	20,000	0	16
Lymphography	4	148	15	3
Mushroom	2	8124	22	0
Pendigits	10	10,992	0	16
Pima-diabetes	2	768	0	8
Primary-tumor	22	239	17	0
Segment	7	2310	0	19
Sonar	2	208	0	60
Soybean	19	683	35	0
Splice	3	3190	60	0
Vehicle	4	846	0	18
Vote	2	435	16	0
Vowel-context	11	990	2	10
Vowel-nocontext	11	990	0	10
Waveform	3	5000	0	40
Wisconsin-breast-cancer	2	699	0	9
Zoo	7	101	16	2

Given a number of iterations, the number of decision nodes is the same for the 3 considered variants. The number of decision nodes is an indicator of the complexity of the classifier because in the training phase it is necessary to select that number of decision nodes and in the classification phase it is necessary to evaluate all the decision nodes.

The presented method is for two classes data sets. For multiclass problem the method named "one vs. all" (Rifkin and Klautau, 2004), also named "one vs the rest", was used. For each class a classifier is constructed, it discriminates between the examples of that class and the examples of the other classes. For data sets with more than two classes, the number of decision stumps in the classifier will be the number of classes multiplied by the number of boosting iterations.

Table 2 shows the accuracies obtained for the different reuse variants in every data set when using the reweighting version of AdaBoost. The results for the resampling version are in Table 3.

Tables 4 and 5 indicate how many times one method is better than the others. For all the number of iterations considered AdaBoost-*r*2 is clearly better than AdaBoost and AdaBoost-*r*1.

Figs. 5 and 6 show the relationship between the results from AdaBoost-*r*2 and AdaBoost. Two relations are considered, the accuracy difference and the accuracy rate. In these graphs, the relations from the different data sets are plotted in ascending order. As in the previous tables, the figures show that AdaBoost-*r*2 is better than AdaBoost more times than the opposite. Moreover, the graphs also show that when AdaBoost-*r*2 is better than AdaBoost, the differences are more important than when AdaBoost-*r*2 is worse than AdaBoost.

Tables 6 and 7 also show comparisons of the different methods. Nevertheless, these tables do not compare directly the averages of the accuracies. Instead, they show the results of the *corrected resampled t-test statistic* (Nadeau and Bengio, 2003) from the 100 results (from 10-fold cross validation, 10 times) for each method and data set (significance level: 0.05). There is a clear advantage for the versions with reuse.

Table 2
Accuracies for the different methods and data sets, for the reweighting version of AdaBoost

	Size: 10			Size: 25			Size: 100		
	AdaBoost	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost	AdaBoost-*r*1	AdaBoost-*r*2
Anneal	96.49	97.10	**98.25**	98.25	98.90	**99.22**	99.12	99.39	**99.51**
Audiology	**75.05**	73.74	74.09	77.08	77.33	**77.43**	**80.15**	78.87	78.07
Autos	70.50	69.25	**73.04**	73.80	75.27	**77.70**	81.08	81.69	**81.84**
Balance-scale	86.19	**86.54**	86.28	91.35	91.43	**91.65**	92.14	92.69	**92.80**
Breast-cancer	71.93	71.44	**75.13**	72.57	71.87	**73.48**	72.25	71.70	**73.13**
Cleveland-14	**83.28**	82.68	82.31	**82.85**	81.82	81.39	**82.35**	81.02	79.73
Credit-rating	84.80	84.80	**85.42**	85.71	85.58	**86.38**	**86.14**	85.97	85.84
German-credit	**71.27**	71.25	**71.27**	72.60	**72.69**	72.35	**74.63**	74.30	74.51
Glass	**69.60**	69.00	69.50	71.63	72.95	**73.08**	73.70	74.40	**75.68**
Heart-statlog	**81.59**	81.52	81.22	**81.81**	80.70	80.67	**81.56**	80.56	79.26
Hepatitis	80.10	**81.66**	81.13	81.63	81.33	**83.03**	**82.49**	82.48	81.88
Horse-colic	81.92	**81.93**	80.81	**82.12**	81.39	80.81	81.98	81.55	**82.26**
Hungarian-14	80.71	**81.06**	79.97	81.10	**81.36**	80.47	**81.44**	80.71	79.73
Hypothyroid	99.10	99.08	**99.33**	99.20	99.44	**99.57**	99.58	99.62	**99.63**
Ionosphere	90.89	91.05	**91.45**	**92.34**	92.34	92.34	92.63	92.37	**92.82**
Iris	**94.67**	94.00	**94.67**	**94.67**	94.33	94.53	94.00	93.80	**94.20**
Labor	**84.30**	83.33	81.70	**86.43**	83.17	85.17	**87.43**	83.70	84.80
Letter	62.65	66.12	**67.15**	69.21	72.86	**75.96**	76.22	79.86	**83.72**
Lymphography	80.42	80.09	**81.43**	82.62	82.37	**83.37**	83.37	83.33	**83.92**
Mushroom	96.29	97.87	**97.95**	98.73	99.79	**99.93**	99.90	**100.00**	**100.00**
Pendigits	85.82	88.46	**90.43**	92.05	94.24	**95.86**	95.89	97.49	**98.40**
Pima-diabetes	**74.93**	74.37	74.59	75.37	75.22	**75.57**	75.45	75.27	**75.54**
Primary-tumor	**46.38**	46.10	43.89	**46.70**	46.00	45.52	45.52	**45.96**	45.64
Segment	93.50	93.87	**94.17**	94.66	95.66	**96.24**	96.52	97.31	**97.62**
Sonar	75.65	77.31	**77.71**	81.20	81.15	**81.46**	**84.86**	84.42	84.29
Soybean	88.22	89.87	**89.00**	92.19	**92.49**	90.83	**93.75**	92.86	90.45
Splice	92.26	93.56	**94.03**	94.97	94.91	**95.16**	95.66	95.63	**95.83**
Vehicle	66.93	68.50	**69.01**	71.24	72.53	**73.49**	75.19	76.06	**77.07**
Vote	**95.68**	95.30	95.24	**95.65**	95.51	95.61	**95.65**	95.60	95.42
Vowel-context	58.78	62.40	**65.11**	69.15	74.55	**79.00**	80.83	85.03	**88.81**
Vowel-nocontext	58.98	62.35	**64.95**	68.95	74.42	**78.58**	80.36	84.24	**87.44**
Waveform	80.27	80.69	**80.82**	83.59	83.69	**84.00**	**84.92**	84.89	84.82
Wisconsin-breast	95.08	**95.19**	95.15	95.22	95.40	**95.59**	95.72	95.85	**95.91**
Zoo	96.34	**97.63**	96.63	96.42	**96.73**	96.25	**96.22**	95.14	94.45

Three ensemble sizes are considered: 10, 25 and 100 decision stumps. For each data set and ensemble size, the best result is marked in bold.

Table 3
Accuracies for the different methods and data sets, for the resampling version of AdaBoost

	Size: 10			Size: 25			Size: 100		
	AdaBoost	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost	AdaBoost-*r*1	AdaBoost-*r*2
Anneal	96.68	97.03	**97.97**	98.24	98.63	**99.11**	99.09	99.32	**99.47**
Audiology	**75.67**	73.68	74.23	77.42	77.19	**77.86**	78.56	78.88	**79.09**
Autos	68.93	71.35	**72.82**	73.39	75.20	**77.51**	78.50	80.79	**81.98**
Balance-scale	84.88	85.84	**85.66**	91.20	91.12	**91.44**	92.18	**92.86**	92.67
Breast-cancer	72.83	71.87	**73.23**	**72.24**	71.73	71.90	**72.20**	71.83	71.18
Cleveland-14	82.18	**82.45**	82.15	**82.61**	82.37	82.05	**82.35**	81.59	80.90
Credit-rating	85.04	84.91	**85.23**	85.46	85.62	**85.70**	85.54	**85.83**	85.52
German-credit	**71.68**	71.27	71.16	**73.08**	72.89	72.84	73.50	73.96	**74.74**
Glass	69.44	69.91	**70.76**	71.63	72.69	**73.75**	73.04	74.32	**76.07**
Heart-statlog	**82.00**	81.63	81.04	**82.41**	81.33	81.19	**82.33**	80.74	80.30
Hepatitis	81.45	81.25	**81.64**	**82.41**	82.33	82.01	**82.59**	81.31	81.97
Horse-colic	**82.09**	81.69	81.66	81.39	81.12	**81.74**	81.69	81.91	**82.26**
Hungarian-14	80.87	80.75	**80.96**	**80.82**	80.31	80.47	**80.89**	**80.89**	80.17
Hypothyroid	99.00	99.10	**99.23**	99.20	99.29	**99.58**	99.62	99.63	**99.65**
Ionosphere	89.80	90.46	**91.09**	91.55	91.63	**91.69**	92.97	92.68	**93.14**
Iris	**94.87**	94.73	94.47	**94.73**	94.60	94.13	**94.60**	94.33	94.27
Labor	85.50	**85.80**	84.17	85.90	**86.93**	**86.93**	87.00	87.10	**87.40**
Letter	62.67	65.13	**66.23**	69.44	71.67	**75.00**	76.42	78.25	**82.94**
Lymphography	81.80	**81.98**	80.84	82.97	83.11	**83.31**	82.96	83.54	**84.05**
Mushroom	96.83	97.66	**98.51**	98.91	99.71	**99.93**	99.91	**100.00**	**100.00**
Pendigits	85.79	87.54	**89.49**	91.86	93.59	**95.44**	95.94	97.21	**98.22**
Pima-diabetes	**74.77**	74.43	74.60	**75.69**	75.31	75.49	**75.92**	75.21	75.26
Primary-tumor	45.91	**45.93**	45.17	**46.70**	46.14	45.96	**46.43**	46.02	46.05
Segment	93.26	93.68	**94.38**	94.50	95.21	**96.19**	96.52	97.31	**97.81**
Sonar	74.57	75.78	**76.92**	**80.56**	80.38	80.45	83.98	83.31	**84.29**
Soybean	**89.08**	88.70	89.02	**92.58**	92.05	90.76	**93.76**	92.90	91.10
Splice	92.25	93.44	**93.74**	94.88	94.88	**94.99**	95.57	95.71	**95.75**
Vehicle	67.17	68.17	**69.14**	70.70	72.65	**73.25**	73.45	75.87	**77.05**
Vote	**95.47**	95.03	94.80	95.42	95.47	**95.31**	95.42	95.86	**95.90**
Vowel-context	58.58	59.23	**63.43**	69.59	71.90	**76.59**	80.17	83.03	**88.36**
Vowel-nocontext	58.57	59.23	**63.22**	69.44	72.00	**76.68**	79.87	82.76	**87.58**
Waveform	80.13	80.09	**80.30**	83.34	83.39	**83.80**	84.48	84.54	**84.88**
Wisconsin-breast	95.01	**95.15**	95.07	95.21	95.47	**95.52**	95.52	95.74	**95.92**
Zoo	95.34	**95.65**	95.14	**95.63**	95.35	94.35	**96.11**	94.25	93.85

Three ensemble sizes are considered: 10, 25 and 100 decision stumps. For each data set and ensemble size, the best result is marked in bold.

Table 4
Comparison of results for the reweighting version of AdaBoost

	Size: 10		Size: 25		Size: 100	
	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2
AdaBoost	19/1/14	22/1/11	19/0/15	22/0/12	14/0/20	21/0/13
AdaBoost-*r*1	–	22/0/12	–	26/0/8	–	21/1/12

Each item in the table shows the number of wins, ties and losses of the method of the column with respect to the method of the row.

Table 5
Comparison of results for the resampling version of AdaBoost

	Size: 10		Size: 25		Size: 100	
	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2
AdaBoost	21/0/13	21/0/13	19/0/15	21/0/13	22/0/12	23/0/11
AdaBoost-*r*1	–	22/0/12	–	24/0/10	–	25/0/9

Each item in the table shows the number of wins, ties and losses of the method of the column with respect to the method of the row.

Instead of comparing the average accuracies obtained from the 10×10-fold cross validations, it is also possible to compare two methods according to the number of folds where one method is better than the other. Fig. 7 shows graphs of the differences between the number of folds with one method better than the other and the opposite. In this

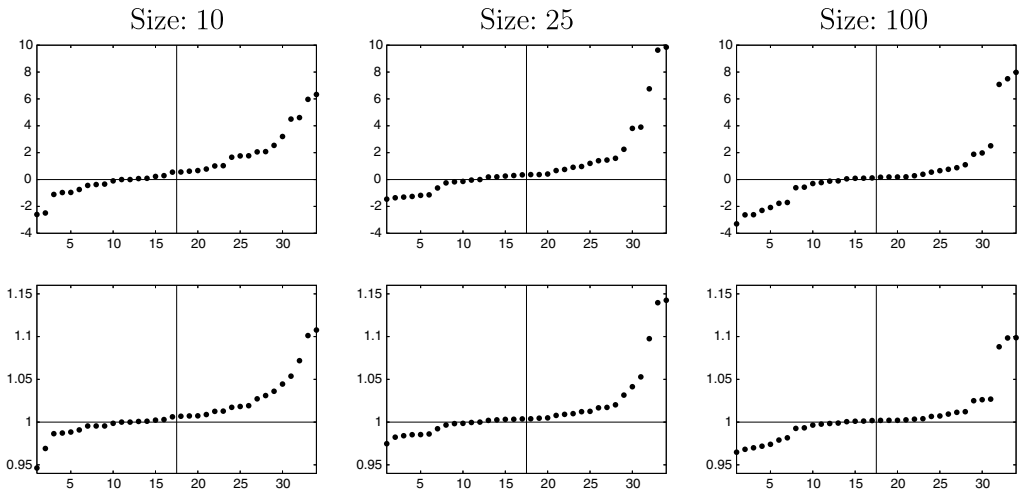

Fig. 5. Relationship, for the different data sets, between the accuracies of AdaBoost-*r*2 and AdaBoost (reweighting version). Top: difference of accuracies. Bottom: rate between accuracies.

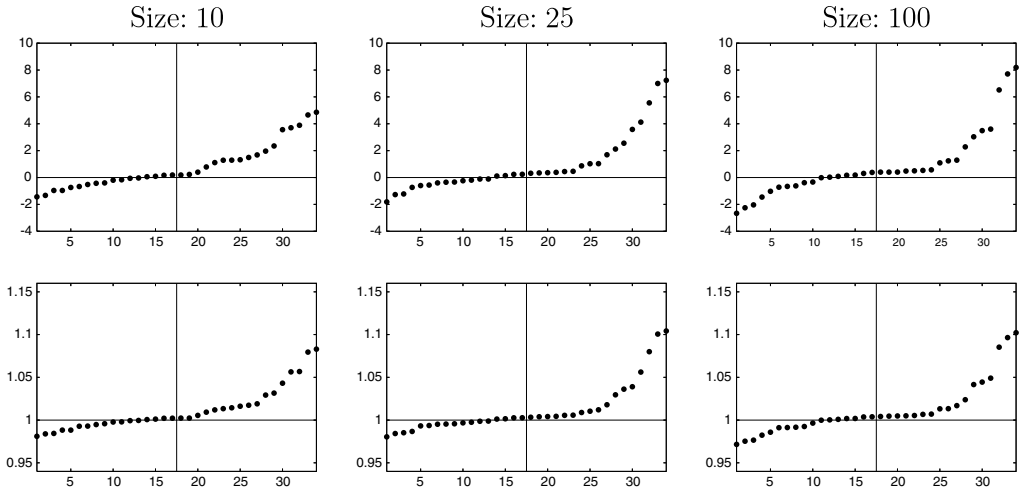

Fig. 6. Relationship, for the different data sets, between the accuracies of AdaBoost-*r*2 and AdaBoost (resampling version). Top: difference of accuracies. Bottom: rate between accuracies.

Table 6
Comparison of the methods according to the corrected resampled *t*-test statistic, for the reweighting version

	Size: 10		Size: 25		Size: 100	
	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2
AdaBoost	6/28/0	7/27/0	7/27/0	8/26/0	6/28/0	6/27/1
AdaBoost-*r*1	–	4/30/0	–	5/29/0	–	4/29/1

Each item in the table shows the number of wins, ties and losses of the method of the column with respect to the method of the row.

Table 7
Comparison of the methods according to the corrected resampled *t*-test statistic, for the resampling version

	Size: 10		Size: 25		Size: 100	
	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2
AdaBoost	3/31/0	8/26/0	3/0/31	9/25/0	6/28/0	7/26/1
AdaBoost-*r*1	–	4/28/0	–	7/27/0	–	4/28/0

Each item in the table shows the number of wins, ties and losses of the method of the column with respect to the method of the row.

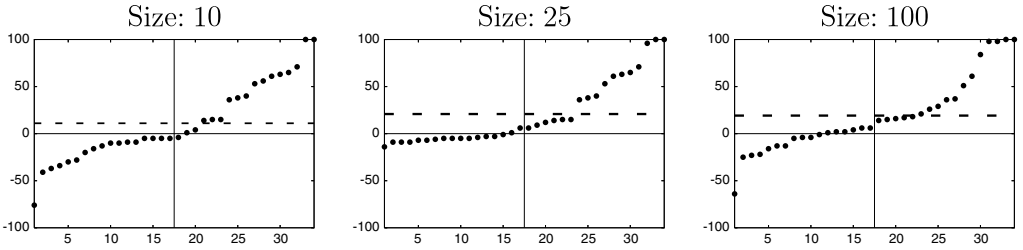

Fig. 7. Sorted differences between the number of wins (number of folds where AdaBoost-*r*2 is better than AdaBoost) and losses, for the resampling version. The dashed line marks the average value of the difference.

Table 8
Average ranks

	Reweighting			Resampling		
	Size: 10	Size: 25	Size: 100	Size: 10	Size: 25	Size: 100
AdaBoost	2.25	2.21	2.03	2.24	2.18	2.32
AdaBoost-*r*1	2.07	2.21	2.22	2.03	2.16	2.09
AdaBoost-*r*2	1.68	1.59	1.75	1.74	1.66	1.59

Table 9
Accuracies for the different methods and data sets, for the resampling version of AdaBoost

	AdaBoost	AdaBoost-*r*1	AdaBoost-*r*2
Anneal	**99.55**	**99.55**	99.44
Audiology	76.96	77.81	**78.66**
Autos	79.45	**83.33**	82.86
Balance-scale	92.16	**94.07**	93.11
Breast-cancer	**71.69**	69.27	69.99
Cleveland-14	**83.15**	79.53	79.20
Credit-rating	**85.80**	85.07	85.22
German-credit	73.90	**74.70**	74.20
Glass	69.18	**75.63**	74.65
Heart-statlog	**80.74**	77.41	75.19
Hepatitis	80.04	80.58	**82.00**
Horse-colic	83.15	82.33	**83.69**
Hungarian-14	**81.70**	80.63	79.28
Hypothyroid	99.55	99.63	**99.71**
Ionosphere	**92.90**	**92.90**	92.61
Iris	**94.00**	93.33	92.67
Labor	**86.33**	84.33	84.33
Letter	80.31	83.46	**86.81**
Lymphography	82.29	84.38	**84.43**
Mushroom	**100.00**	**100.00**	**100.00**
Pendigits	96.92	98.54	**99.11**
Pima-diabetes	**75.66**	75.27	74.10
Primary-tumor	**46.60**	46.31	43.07
Segment	97.32	98.14	**98.31**
Sonar	**86.52**	84.60	**86.52**
Soybean	**93.99**	93.41	90.62
Splice	95.58	95.71	**95.80**
Vehicle	75.29	**79.44**	78.97
Vote	95.64	**96.32**	95.64
Vowel-context	82.42	90.51	**93.03**
Vowel-nocontext	82.22	88.28	**91.82**
Waveform	84.48	**84.76**	84.68
Wisconsin-breast	95.28	**96.57**	95.85
Zoo	**96.00**	93.18	93.18

The number of iterations is 500. For each data set and ensemble size, the best result is marked in bold.

case, the advantage of AdaBoost-*r*2 over AdaBoost grows with the size of the ensemble.

Another way of comparing the results of different methods over different data sets is to use average ranks (Demšar, 2006). For each data set, the methods are sorted according to their performance. Each method will have a position in this rank. If several methods have the same result, they are assigned an average rank (e.g., if two methods have the best value, their position in the ranking will be 1.5). The average rank of a method is calculated from its ranks in all the considered data sets. Table 8 shows these average ranks for the six cases (two versions: reweighting and resampling, three sizes: 10, 25 and 100). The average rank of AdaBoost-*r*2 is always smaller than 1.8, while the average ranks of the other two versions are always greater than 2.

4. Additional experiments

4.1. Experiments with 500 iterations

In the previous section, the maximum number of considered iterations for the boosting algorithm was 100. It must be noted that the number of base classifiers in the ensemble is the number of iterations multiplied by the number of classes. For instance, for the "letter" data set, the ensemble is formed by $26 \times 100 = 2600$ base classifiers. This value is greater than the number of decision stumps combined in other works. For instance, Eibl and Pfeiffer (2005) and Li (2006) consider, respectively, ensembles of 2000 and 500 decision stumps.

Nevertheless, it could be possible that with more iterations, the results for classical AdaBoost were as good as the results of AdaBoost with reuse. Hence, this section

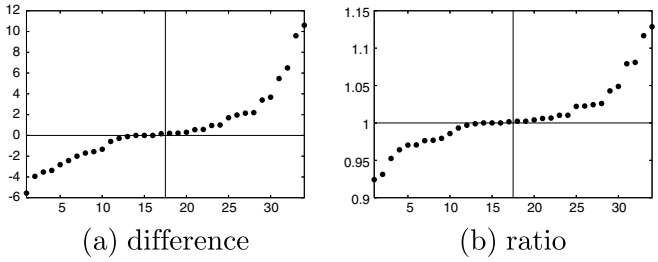

Fig. 8. Relationship, for the different data sets, between the accuracies of AdaBoost-*r*2 and AdaBoost (resampling version) when the ensemble size is 500.

Table 10
Comparison of results for ensembles of 500 decision stumps

	AdaBoost-*r*1	AdaBoost-*r*2
(a) *Wins/ties/losses*		
AdaBoost	19/2/13	19/2/13
AdaBoost-*r*1		14/3/17
(b) *Significant wins/ties/losses*		
AdaBoost	4/30/0	4/30/0
AdaBoost-*r*1		4/30/0

Each item in the table shows the number of wins, ties and losses of the method of the column with respect to the method of the row.

includes results for 500 iterations. Only the resampling version is considered, the results are for one 10-fold cross validation. Table 9 shows the accuracies of the different methods for each data set.

Fig. 8 shows the relationships between the accuracies of AdaBoost and AdaBoost-*r*2. The latter has better results than the former for the majority of the data sets. Table 10 shows the number of wins, ties and losses for each pair of methods. The direct comparison of accuracies indicates that the versions with reuse are still better than the version without reuse, although the differences are less important than when using less iterations. For this number of iterations, it is better to reuse only the previous classifier (AdaBoost-*r*1) than to reuse the two previous classifiers (AdaBoost-*r*2). When using the considered statistical test, the version without reuse is never better than the versions with reuse, while the two versions with reuse are significantly better than AdaBoost for four data sets.

The average ranks for AdaBoost, AdaBoost-*r*1 and AdaBoost-*r*2 are, respectively, 2.15, 1.88 and 1.97. With these number of iterations, it is better the *r*1 version than the *r*2 version, but both are better than the version without reuse.

4.2. More complex base classifiers

It is also possible to reuse more complex classifiers. This section reports results using decision trees with three decision nodes and four leaves. The reuse idea can be used in the same way. For two-classes data sets, the output of these trees is binary. For a reuse level r, $r+1$ classifiers are combined. There are 2^{r+1} possible output combinations. For each combination the class is determined according to the labels of the training examples that receive that classifications. Fig. 9 shows an example.

These results are for the resampling version of AdaBoost, from 10×10-fold cross validation. Table 11 shows the number of wins, ties and losses for each pair of meth-

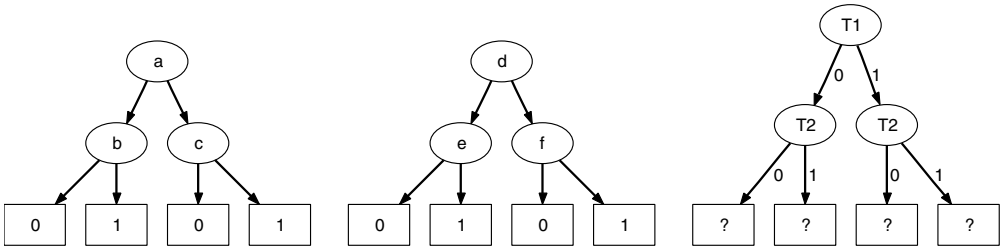

Fig. 9. Combination of two 4-leaves decision trees. Left and center: the trees T1 and T2. Right: the combined classifier. The leaves labels of the combined classifier depend on the training data. For instance, the label of the leftmost leaf would be the most frequent class for the examples that are classified as 0 by T1 and T2.

Table 11
Comparison of results for ensembles of four-leaves trees

	Size: 10		Size: 25		Size: 100	
	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2	AdaBoost-*r*1	AdaBoost-*r*2
AdaBoost	20/0/14	16/0/18	21/1/12	15/1/18	16/1/17	14/1/19
AdaBoost-*r*1	–	16/0/18	–	14/1/19	–	12/1/21
AdaBoost	*1/33/0*	*2/32/0*	*1/33/0*	*4/30/0*	*1/33/0*	*4/30/0*
AdaBoost-*r*1	–	*2/32/0*	–	*2/32/0*	–	*1/33/0*

Each item in the table shows the number of wins, ties and losses of the method of the column with respect to the method of the row. In italics, the number of significant wins, ties and losses.

Table 12
Average ranks when using four-leaves trees as base classifiers

	Size: 10	Size: 25	Size: 100
AdaBoost	2.06	2.09	1.91
AdaBoost-*r*1	1.90	1.79	1.88
AdaBoost-*r*2	2.04	2.12	2.21

ods. The direct comparison of the accuracies shows that a reuse level of 2 is too much, AdaBoost-*r*2 is the worst method. This can be caused by the excessive similarity between classifiers (i.e., two consecutive classifiers share six of the nine decision nodes, a classifier is formed by three 3-decision trees). AdaBoost-*r*1 is better than AdaBoost when the size is 10 and 25, but slightly worse for 100 (the difference is only one data set).

On the other hand, according to the statistical test, AdaBoost-*r*2 is the best method. The version without reuse is never significantly better than the versions with reuse.

Table 12 shows the average ranks (Demšar, 2006) of the three considered methods. For all the considered sizes, AdaBoost-*r*1 is the best method.

5. Conclusion

Although it is possible to use boosting with strong base classifiers (e.g., decision trees, neural networks, etc.) the name of the method comes from is ability to obtain strong classifiers from weak methods. A weak method commonly used with boosting are decision stumps, decision trees with only one decision node and two leaves. This work presents a method that improves the results of AdaBoost when used with decision stumps. It is based on combining several decision stumps in a tree.

The computational complexity is very similar for the original AdaBoost and the proposed method. The number of decision stumps is the same in both versions. Although in the proposed variant a decision node will appear in several trees, it will be only evaluated once for each example.

The storage requirements of the two versions are the same, with the only exception that in the proposed variant 2^{r+1} bits are necessary for each decision stump (r is the number of reused classifiers). The maximum value for r considered in this work is 2, hence a byte for each decision stump is enough.

The experimental validation, over 34 data sets from the UCI repository shows the validity of the proposed method, when using decision stumps as base classifiers. When using four leaves decision trees as base classifiers, the advantage for the reuse version is not so clear, but the results are generally favourable for the reuse version with $r = 1$.

The present work has only considered small decision trees as base classifiers. The method could be used with any other classification method, although it will only be useful if the classification method does not generate strong classifiers. In future works, other weak classification methods will be considered.

Currently, the proposed method only considers two classes data sets. The original version of AdaBoost had the same limitation. In this work multiclass classifiers have been obtained constructing as many classifiers as classes. Nevertheless, it would be interesting to add the reuse ability to some of the multiclass variants of AdaBoost, such as AdaBoost.M2 (Freund and Scapire, 1997), AdaBoost.OC (Schapire, 1997), AdaBoost.ECC (Guruswami and Sahai, 1999), AdaBoost.ERP (Li, 2006) or BoostMA (Eibl and Pfeiffer, 2005). This idea of reusing weak classifiers could be also included in other versions of Boosting, such as AdaBoost$_{\mathbb{R}}$ (Nock and Nielsen, 2006).

References

Blake, C.L., Merz, C.J., 1998. UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Buntine, W., 1993. Learning classification trees. Artificial Intelligence Frontiers in Statistics. Chapman and Hall, London, pp. 182–201.

Caruana, R., Niculescu-Mizil, A., 2006. An empirical comparison of supervised learning algorithms. In: ICML 2006: 23rd International Conference on Machine Learning.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Dietterich, T.G., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Mach. Learn. 40 (2), 139–158.

Eibl, G., Pfeiffer, K.P., 2005. Multiclass-boosting for weak classifiers. J. Mach. Learn. Res. 6, 189–210.

Estruch, V., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M., 2003. Beam search extraction and forgetting strategies on shared ensembles. In: Multiple Classifier Systems, Fourth International Workshop, MCS 2003.

Freund, Y., Scapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139.

Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: A statistical view of boosting. Ann. Stat. 38 (2), 337–374.

Guruswami, V., Sahai, A., 1999. Multiclass learning, boosting, and error-correcting codes. In: Proceedings of the 12th Annual Conference on Computational Learning Theory (COLT 1999), ACM.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20 (8), 832–844.

Kohavi, R., Kunz, C., 1997. Option decision trees with majority votes. In: Machine Learning Proceedings of the 14th International Conference.

Kuncheva, L.I., 2004. Combining Pattern Classifiers: Methods and Algorithms. Wiley–Interscience.

Kuncheva, L., 2005. Diversity in multiple classifier systems (editorial). Inform. Fusion 6 (1), 3–4.

Li, L., 2006. Multiclass boosting with repartitioning. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006.

Li, J., Liu, H., 2003. Ensembles of cascading trees. In: Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03), pp. 585–588.

Nadeau, C., Bengio, Y., 2003. Inference for the generalization error. Mach. Learn. 52, 239–281.

Nock, R., Nielsen, F., 2006. A ℝeal generalization of discrete adaboost. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (Eds.), ECAI-06, Proceedings of the 17th European Conference on Artificial Intelligence, vol. 141 of Frontiers in Artificial Intelligence and Applications. IOS Press.

Reyzin, L., Schapire, R.E., 2006. How boosting the margin can also boost classifier complexity. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006.

Rifkin, R., Klautau, A., 2004. In defense of one-vs-all classification. J. Mach. Learn. Res. 5, 101–141.

Rodríguez, J.J., Maudes, J., 2006. Ensembles of grafted trees. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (Eds.), ECAI-06, Proceedings of the 17th European Conference on Artificial Intelligence, vol. 141 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 803–804.

Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J., 2006. Rotation forest: A new classifier ensemble method. IEEE Trans. Pattern Anal. Mach. Intell. 28 (10), 1619–1630 http://doi.ieeecomputersociety.org/10.1109/TPAMI.2006.2%11.

Schapire, R.E., 1990. The strength of weak learnability. Mach. Learn. 5 (2), 197–227.

Schapire, R.E., 1997. Using output codes to boost multiclass learning problems. In: Proceedings of the 14th International Conference on Machine Learning (ICML-97). pp. 313–321.

Schapire, R.E., 1999. A brief introduction to boosting. In: Dean, T. (Ed.), Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99). Morgan Kaufmann, pp. 1401–1406.

Schapire, R.E., 2002. The boosting approach to machine learning: An overview. In: MSRI Workshop on Nonlinear Estimation and Classification. http://www.cs.princeton.edu/schapire/papers/msri.ps.gz.

Schapire, R.E., Singer, Y., 1998. Improved boosting algorithms using confidence-rated predictions. In: Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT 1998), ACM. pp. 80–91.

Wolpert, D., 1992. Stacked generalization. Neural Network 5 (2), 241–260.