Improving Random Forests

Marko Robnik-Šikonja

University of Ljubljana,
Faculty of Computer and Information Science,
Tržaška 25, 1001 Ljubljana, Slovenia
tel.: +386 1 4768459, fax: +386 1 4768498
Marko.Robnik@fri.uni-lj.si

Abstract. Random forests are one of the most successful ensemble methods which exhibits performance on the level of boosting and support vector machines. The method is fast, robust to noise, does not overfit and offers possibilities for explanation and visualization of its output. We investigate some possibilities to increase strength or decrease correlation of individual trees in the forest. Using several attribute evaluation measures instead of just one gives promising results. On the other hand replacement of ordinary voting with voting weighted with margin achieved on most similar instances gives improvements which are statistically highly significant over several data sets.

1 Introduction

Several authors have noted that constructing ensembles of base learners can significantly improve the performance of learning. Bagging [1], boosting [6], random forests [2] and their variants are the most popular examples of this methodology. Boosting and random forests are comparable and sometimes better than state-of-the-art methods in classification and regression [10].

The success of ensemble methods is usually explained with the margin and correlation of base classifiers [14, 2]. To have a good ensemble one needs base classifiers which are diverse (in a sense that they predict differently), yet accurate. The voting mechanism which operates on the top of base learners then ensures highly accurate predictions of the ensemble.

The AdaBoost algorithm constructs a series of base learners by weighting their training set of examples according to the correctness of the prediction. Correctly predicted examples have their weights decreased, and incorrect prediction results in the increased weight of an instance. In this way the subsequent base learners receive effectively different learning sets and gradually focus on the most problematic instances. Usually tree based models are used as base learners.

Random forests construct a series of tree-based learners. Each base learner receives different training set of n instances which are drawn independently with replacement from the learning set of n instances. The bootstrap replication of training instances is not the only source of randomness. In each node of the tree the splitting attribute is selected from a randomly chosen sample of attributes. Random forests are computationally effective and offer good prediction performance. They are proven not to overfit,

J.-F. Boulicaut et al. (Eds.): ECML 2004, LNAI 3201, pp. 359–370, 2004.

and are less sensitive to noisy data compared to boosting. As the training sets of individual trees are constructed by bootstrap replication, there is on average $1/e \approx 36.8\%$ of instances not taking part in construction of the tree[1]. These instances, called out-of-bag instances are the source of data for useful internal estimates of error, strength and correlation. Breiman's homepage offers several tools for exploiting the power of these estimates and random forests.

In spite of apparent success of random forests methodology, we believe there is room for improvement. This paper describes some successful and some unsuccessful attempts to do so.

Individually, each of the base learners is a poor predictor. The random selection of attributes makes individual trees rather weak. Our first aim was to strengthen individual trees, without sacrificing variety between them or, alternatively increase variance without sacrificing strength. We partly succeeded in the first goal by using ReliefF algorithm for attribute estimation and in the second by using several different attribute evaluation measures for split selection. Another improvements stems from the voting mechanism. Not all trees are equally successful in labeling all instances. We use internal estimates to identify instances most similar to the one we wish to label and then weight the votes of the trees with the strength they demonstrate on these near instances. Improvements are demonstrated on several classification data sets.

Throughout the paper we use the notation where each learning instance is represented by an ordered pair $(\mathbf{x}, y)$, where each vector of attributes $\mathbf{x}$ consists of individual attributes A_i, $i = 1,...,a$, (a is the number of attributes) and is labeled with the target value y_j, $j = 1,...,c$ (c is the number of class values). The correct class is denoted as y, without index. Each discrete attribute A_i has values v_1 through v_{m_i} (m_i is the number of values of attribute A_i). We write $p(v_{i,k})$ for the probability that the attribute A_i has value v_k, $p(y_j)$ is the probability of the class y_j, and $p(y_j|v_{i,k})$ is the probability of the class y_j conditioned by the attribute A_i having the value v_k.

The paper is organized into 6 sections. Section 2 contains some background on random forests, describes how strength and correlation are measured and shortly presents the databases and methodology we used to evaluate improvements. In Section 3 we propose using several attribute evaluation measures as split selectors to decrease correlation of trees in the forest. In Section 4 we analyze weighted voting. Section 5 contains evaluation of the proposed improvements on fresh data sets not used during development and Section 6 concludes the work.

2 Random Forests

Random forests as used by Breiman [2] consist of ensemble of K classifiers $h_1(\mathbf{x})$, $h_2(\mathbf{x}), \ldots, h_K(\mathbf{x})$. Each classifier votes for one of the classes and an instance being classified is labeled with the winning class. We denote the joint classifier as $h(\mathbf{x})$. Each training set of n instances is drawn at random with replacement from the training set of n instances. With this sampling called bootstrap replication, on average 36.8% of training instances are not used for building each tree. These out-of-bag instances come handy for computing an internal estimate of the strength and correlation of the forest.

[1] Constant $e \approx 2.718$ stands for the base of the natural logarithms.

Denote the set of out-of-bag instances for classifier h_k as O_k. Let $Q(\mathbf{x}, y_j)$ be the out-of-bag proportion of votes for class y_j at input $\mathbf{x}$ and an estimate of $P(h(\mathbf{x}) = y_j)$:

$$Q(\mathbf{x}, y_j) = \frac{\sum_{k=1}^{K} I(h_k(\mathbf{x}) = y_j; (\mathbf{x}, y) \in O_k)}{\sum_{k=1}^{K} I(h_k(\mathbf{x}); (\mathbf{x}, y) \in O_k)} \tag{1}$$

where $I(\cdot)$ is the indicator function. The margin function measures the extent to which the average vote for the right class y exceeds the average vote for any other class:

$$mr(\mathbf{x}, y) = P(h(\mathbf{x}) = y) - \max_{\substack{j=1 \\ j \neq y}}^{c} P(h(\mathbf{x}) = y_j) \tag{2}$$

It is estimated with $Q(\mathbf{x}, y)$ and $Q(\mathbf{x}, y_j)$. Strength is defined as the expected margin, and is computed as the average over the training set:

$$s = \frac{1}{n} \sum_{i=1}^{n} \left(Q(\mathbf{x}_i, y) - \max_{\substack{j=1 \\ j \neq y}}^{c} Q(\mathbf{x}_i, y_j) \right) \tag{3}$$

The average correlation is computed as the variance of the margin over the square of standard deviation of the forest:

$$\overline{\rho} = \frac{var(mr)}{sd(h())^2} = \frac{\frac{1}{n} \sum_{i=1}^{n} \left(Q(\mathbf{x}_i, y) - \max_{\substack{j=1 \\ j \neq y}}^{c} Q(\mathbf{x}_i, y_j) \right)^2 - s^2}{\left(\frac{1}{K} \sum_{t=1}^{K} \sqrt{p_k + \hat{p}_k + (p_k - \hat{p}_k)^2} \right)^2} \tag{4}$$

where

$$p_k = \frac{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}) = y)}{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}))}$$

is an out-of-bag estimate of $P(h_k(\mathbf{x}) = y)$ and

$$\hat{p}_k = \frac{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}) = \hat{y}_j)}{\sum_{(\mathbf{x}_i, y) \in O_k} I(h_k(\mathbf{x}))}$$

is an out-of-bag estimate of $P(h_k(\mathbf{x}) = \hat{y}_j)$ and

$$\hat{y}_j = \arg \max_{\substack{j=1 \\ j \neq y}}^{c} Q(\mathbf{x}, y_j)$$

is estimated for every instance $\mathbf{x}$ in the training set with $Q(\mathbf{x}, y_j)$.

Breiman [2] uses unpruned decision trees as base classifiers and introduces additional randomness into the trees. Namely, in each interior node of each tree a subset of r attributes is randomly selected and evaluated with the Gini index heuristics. The attribute with the highest Gini index is chosen as split in that node.

We have implemented the random forests methodology in the framework of our learning system[2]. All presented tests were executed with the recommended values of the parameters for the forests (number of trees $K = 100$, number of randomly selected attributes in each node $r = \sqrt{a}$, tree building is stopped when the number of instances in a node is 5 or less).

For evaluation of improvements we use the UCI data sets [11] from Breiman's paper and two additional parity problems: parity2 with two parity attributes, and parity3 with three parity attributes. Each of these parity problems contains also 10 random binary attributes. The characteristics of data sets are collected in Table 1 which contains information on the number of instances (n), number of classes (c), number of attributes (a), number of numeric (num) and nominal (nom) attributes, and the percentage of missing values (miss). All problems except three larger ones were evaluated with 10-fold cross-validation. For three larger problems we used predefined fixed splits to training and testing sets; for letters 15000 instances for learning, 5000 for testing, for sat 4435 for learning, 2000 for testing, and for zip 7291 for learning, 2007 for testing.

Table 1. Characteristics of the problems used during the development process.

name	n	c	a	num	nom	miss
breast-cancer	699	2	9	9	0	0.25
bupa	345	2	6	6	0	0
diabetes	768	2	8	8	0	0
ecoli	336	8	7	7	0	0
german-numeric	1000	2	24	24	0	0
glass	214	7	9	9	0	0
ionosphere	351	2	34	34	0	0
letter	20000	26	16	16	0	0
parity2	200	2	12	0	12	0
parity3	200	2	13	0	13	0
sat	6435	6	36	36	0	0
segmentation	2310	7	19	19	0	0
sonar	208	2	60	60	0	0
soybean	683	19	35	0	35	9.78
vehicle	846	4	18	18	0	0
vote	435	2	16	0	16	5.63
vowel	990	11	11	10	1	0
zip	9298	10	256	256	0	0

For comparison of classifiers we use Wilcoxon signed-rank test, a non-parametric equivalent of paired t-test. Its advantage is that it does not require normal distributions

[2] All algorithms and programs are available at http://lkm.fri.uni-lj.si/rmarko/

or homogenity of variance. The price we pay for this is lower power of the test, so we risk that some differences will not be recognized as significant. Details of the tests can be found in [15], and a discussion of their use in machine learning context in [4]. We compare the classification accuracy and the area under the ROC curve (AUC). For problems with more than two classes the AUC extension proposed by [7] is used. This extension evaluates separation of pairs of classes, which could be misleading in certain situations, e.g., in some folds we have noticed cases with perfect AUC score (1.0) but less than 100% classification accuracy.

Throughout the paper and our work with random forests we have used these data sets several times to evaluate improvements, which could lead to overfitting. To avoid this danger we introduce another set of learning problems, not used during the development, in Section 5 and test the final methods on them.

Our first aim was to increase strength or reduce correlation of the forests. We tackled this challenge with attribute evaluation.

3 Attribute Evaluation

Breiman uses Gini index as the feature evaluation measure in random forests. Gini index is fast but has some drawbacks compared to other heuristics (see [9] for detailed analysis), in particular it cannot detect strong conditional dependencies among attributes. If the problem is described with attributes where such dependencies arise (e.g., XOR type of attributes) the evaluation will be incorrect and resulting performance of classifier could be poor. The reason for this deficiency of Gini index is that it measures the impurity of the class value distribution before and after the split on evaluated attribute. In this way it assumes the conditional (upon the class) independence of attributes, evaluates each attribute separately and does not take the context of other attributes into account. The same behavior is typical for all impurity based measures (e.g., Gain ratio, DKM, MDL, j-measure). In problems which possibly involve much feature interactions these measures are not appropriate. The measure which solves this problem for classification problems is ReliefF [13].

My first idea of using ReliefF in random forests was to evaluate attributes in the preprocessing step, and use the quality estimates as weights in the process of selecting subsample of attributes in each interior node of the tree. While this approach worked well for artificial problems with highly dependent attributes (parity problems) it on average increased the correlation between the trees and resulted in decreased performance. Another idea was to replace Gini as the sole attribute evaluation measure with several others and thereby decrease correlation but retain strength. This indeed was the case and we describe the exact procedure below.

The problem of attribute evaluation has received much attention in the machine learning and there are several measures for estimating attributes' quality. In classification problems the most popular are e.g., Gini index [3], Gain ratio [12], ReliefF [8], MDL [9], and DKM [5]. DKM is suitable only for two class problems, so we did not use it in our study.

Except ReliefF all these measures are impurity based, meaning that they measure impurity of the class value distribution. They are fast, demanding $O(n \cdot a)$ steps for the

evaluation of a attributes. We included also Myopic ReliefF, which contrary to ReliefF, is also impurity based and possesses some interesting biases [9]. The general form of all impurity based measures is:

$$I(A_i) = i(y) - \sum_{j=1}^{m_i} p(v_{i,j}) i(y|v_{i,j}) \ ,$$

where $i(y)$ is the impurity of class values before the split, and $i(y|v_{i,k})$ is the impurity of class values after the split on $A_i = v_{k,j}$. By subtracting the weighted impurity of the splits from the impurity of unpartitioned instances we measure gain in the purity of class values resulting from the split. Larger values of $I(A_i)$ imply pure splits and therefore good attributes. We cannot directly apply these measures to numerical attributes, but we can use any of the number of discretization techniques first and then evaluate discretized attributes, or, as in our case where the binary trees are built, we simply take the maximum purity gain over all possible splits of the numeric attribute.

We briefly present measures we used, first the ones based on impurity followed by ReliefF.

Gini index is used in CART learning system [3] and because of its simplicity it is also the measure chosen by Breiman for random forests.

$$Gini(A_i) = -\sum_{i=1}^{c} p(y_i)^2 + \sum_{j=1}^{m_i} p(v_{i,j}) \sum_{i=1}^{c} p(y_i|v_{i,j})^2 \tag{5}$$

Gain ratio [12] is implemented in C4.5 program and is the most often used impurity based measure. It is defined as

$$GR(A_i) = \frac{\sum_{i=1}^{c} p(y_i) \log p(y_i) - \sum_{j=1}^{m_i} \sum_{i=1}^{c} p(y_i|v_{i,j}) \log p(y_i|v_{i,j})}{\sum_{j=1}^{m_i} p(v_{i,j}) \log p(v_{i,j})} \ . \tag{6}$$

Its gain part tries to maximize the difference of entropy (which serves as the impurity function) before and after the split. To prevent excessive bias towards multiple small splits the gain is normalized with the attribute's entropy.

MDL is based on the Minimum Description Length principle and measures the quality of attributes as their ability to compress the data. The difference in coding length before and after the value of the attribute is revealed corresponds to the difference in impurity. Kononenko [9] has shown empirically that this criterion has the most appropriate bias concerning multi-valued attributes among a number of other impurity-based measures. It is defined as:

$$MDL(A_i) = \frac{1}{n} \left(\log_2 \binom{n}{n_{1.}, \ldots, n_{c.}} - \sum_{j=1}^{m_i} \log_2 \binom{n_{.j}}{n_{1j}, \ldots, n_{cj}} + \log_2 \binom{n+c+1}{c-1} - \sum_{j=1}^{m_i} \log_2 \binom{n_{.j}+c-1}{c-1} \right) \tag{7}$$

Here n is the number of training instances, $n_{i.}$ the number of training instances from class i, $n_{.j}$ the number of instances with j-th value of given attribute, and n_{ij} the number of instances from class i with j-th value of the attribute.

Myopic ReliefF [9, 13] is a byproduct of the analysis of ReliefF' behavior with large number of near instances. It is strongly correlated to Gini index but has additional normalization against multi-valued attributes and in case of multi-class problems:

$$MR(A_i) = \frac{\left(\sum_{j=1}^{m_i} p(v_{i,j})^2 \sum_{i=1}^{c} p(y_i|v_{i,j})^2\right) - \left(\sum_{j=1}^{m_i} p(v_{i,j})^2\right) \sum_{i=1}^{c} p(y_i)^2}{\left(\sum_{i=1}^{c} p(y_i)^2\right)\left(1 - \sum_{i=1}^{c} p(y_i)^2\right)} \tag{8}$$

ReliefF evaluates partitioning power of attributes according to how well their values distinguish between similar instances. An attribute is given a high score if its values separate similar observations with different class and do not separate similar instances with the same class values. ReliefF samples the instance space, computes the differences between predictions and values of the attributes and forms a statistical measure for the proximity of the probability densities of the attribute and the class. Its quality estimates can be explained as the the proportion of the explained class values. Assigned quality evaluations are in the range $[-1, 1]$. The computational complexity for evaluation of a attributes is $O(m \cdot n \cdot a)$, where m is the number of iterations. Details of the algorithm and its analysis can be found in [13]. We use it throughout this paper with its default settings, except for the number of iterations m, which we set to the $\log n$ to ensure fast execution needed for large number of evaluations in the trees. Nevertheless ReliefF is slower than impurity functions as it has to find nearest neighbors in $O(n \cdot a)$ steps even when the sample of attributes contains less than a attributes.

3.1 Forests with Several Evaluation Measures

In Table 2 we compare performance of two random forests variants. The standard one which uses Gini index as the sole attribute evaluation heuristic, and the modified one which uses five attribute evaluation measures (each fifth of the trees is build with different heuristics: Gini index, Gain ratio, MDL, Myopic ReliefF, or ReliefF). We present results for classification accuracy and AUC. For accuracy the difference over all data sets is significant at 0.2 level using Wilcoxon signed-rank test, while for AUC the differences are not significant.

While some improvement has been achieved we cannot be satisfied. Our further investigation were in the area of classification with the forests.

4 Weighted Voting

A close investigation of the performance of individual trees on artificial data sets have shown that not all trees are equally responsible for incorrect classification of individual instances. This simple observation led to the idea that it would be useful to use only

Table 2. Random forests performance for sole Gini and 5 estimators.

	Gini index		5 estimators	
name	accuracy	AUC	accuracy	AUC
breast-cancer	0.966	0.992	0.967	0.991
bupa	0.734	0.760	0.716	0.777
diabetes	0.762	0.756	0.770	0.747
ecoli	0.869	0.899	0.872	0.897
german-numeric	0.750	0.638	0.750	0.601
glass	0.763	0.942	0.776	0.940
ionosphere	0.937	0.958	0.937	0.963
letter	0.957	0.999	0.963	0.999
parity2	0.820	0.912	0.865	0.941
parity3	0.575	0.644	0.635	0.681
sat	0.910	0.985	0.908	0.984
segmentation	0.982	0.999	0.981	0.999
sonar	0.817	0.903	0.793	0.891
vehicle	0.750	0.926	0.752	0.926
vote	0.957	0.990	0.956	0.990
vowel	0.979	0.999	0.977	0.999
zip	0.934	0.995	0.938	0.995

some selected trees in classification. The selection of trees was based on their performance on similar instances, but without success. Further refinement of this idea has led to weighted voting.

For each instance we want to classify with the forest, we first find some of its most similar instances. Similarity is measured with the forests as in [2]. For that matter we keep track of the similarity for all the training instances. When classifying an instance I with one of the trees, all the training instances from the leaf node where I is classified into have their similarity score increased by one. As we repeat the procedure for all the trees the frequency of co-occurrence forms a measure of instance similarity. The same procedure is used by Breiman to detect prototypes and outliers. We select t most similar training instances and classify them with each tree where they are in the out-of-bag set. For each tree in the forest we measure margin (see Eq. (2)) on these similar out-of-bag instances. The trees with negative average margin are left out of classification. For final classification we use weighted voting of the remaining trees where weights are average margins on the similar instances when they are in the out-of-bag set. The algorithm demands $O(n \cdot K)$ additional space for saving information about n training instances in the leaves of the K trees. While learning time remains unchanged, the classification takes on average t/e times more time to classify t instances with the trees where they are in the out-of-bag set (this happens on average in $1/e \approx 36.8\%$ of the trees).

A few tests have shown that algorithm is quite robust to the selection of t, so we have set $t = 30$ which works satisfactory, but no systematic study was performed. In the left-hand side of Table 3 we present the accuracy and AUC of the forests with weighted voting (Gini with wv). If we compare these results with ordinary voting (see Gini columns in Table 2), we see that weighted voting is mostly better and never worse than ordinary voting.

Table 3. Random forests performance for weighted voting and 5 estimators with weighted voting.

	Gini with wv		5est with wv	
name	accuracy	AUC	accuracy	AUC
breast-cancer	0.967	0.992	0.967	0.991
bupa	0.739	0.770	0.719	0.781
diabetes	0.770	0.759	0.773	0.750
ecoli	0.869	0.902	0.866	0.896
german-numeric	0.760	0.641	0.758	0.613
glass	0.795	0.944	0.781	0.947
ionosphere	0.940	0.960	0.940	0.964
letter	0.958	0.999	0.964	0.999
parity2	0.875	0.940	0.910	0.971
parity3	0.625	0.691	0.740	0.841
sat	0.910	0.985	0.908	0.984
segmentation	0.982	0.999	0.981	0.999
sonar	0.865	0.936	0.841	0.922
vehicle	0.755	0.927	0.746	0.927
vote	0.957	0.991	0.956	0.990
vowel	0.979	0.999	0.977	0.999
zip	0.934	0.995	0.939	0.995

We executed Wilcoxon signed-ranks test to establish significance of the difference with ordinary voting. The results for accuracy and AUC are significant even at 0.001 level!

An obvious thing to do is to combine the two successful improvements: several attribute estimators and weighted voting. We present these results in the right-hand side of Table 3. Some improvements are observed but also slight degradation of performance on others. The differences are significant only compared to plain random forests (0.1 for accuracy and 0.2 for AUC) or random forests with 5 estimators (0.1 for accuracy, non significant for AUC) while they are nonsignificant compared to weighted voting.

The results indicate that weighted voting is clearly and significantly beneficial, while the positive effect of multiple estimators is not so convincing. Next Section tries to clarify this issue.

5 Evaluation on Fresh Data Sets

To avoid the overfitting resulting from the persistent use of data sets described in Table 1, we introduce another set of 17 learning problems from UCI repository, not used during development. The criterion for choosing particular data sets is their availability in the format our system could read (or little effort to transform them into such format). Their characteristics are contained in Table 4.

We compare the performance of plain random forests with random forests with one or both of successful improvements: several attribute evaluation heuristics and voting weighted with the average margin on similar instances. The accuracy and AUC figures collected in Table 5 are calculated with 10-fold cross validation.

Table 4. Characteristics of the new data sets.

name	n	c	a	num	nom	miss
adult	5908	2	14	6	8	1.05
audiology	226	24	69	0	69	2.03
banding	138	2	29	19	10	6.85
credit aus	690	2	15	6	9	0.65
heart	270	2	13	7	6	0
hepatitis	155	2	19	6	13	5.67
house	91	2	16	0	16	4.53
iris	150	3	4	4	0	0
lymphography	148	4	18	0	18	0.04
mushroom	8124	2	22	0	22	1.39
post-operative	90	3	8	1	7	0.42
primary tumor	339	22	17	0	17	3.9
promoter	106	2	57	0	57	0
rheumatism	355	6	32	21	11	0.04
spambase	4601	2	57	57	0	0
splice-jxn	3190	3	60	0	60	0.03
yeast	1484	10	8	8	0	0

We can observe similar effects as before. The weighted voting causes strong boost in performance (significant at 0.001 level for accuracy and 0.01 for AUC), while several estimators show only a moderate nonsignificant improvement. The combination of both is significantly different to plain variant at 0.005 level for accuracy and at 0.1 level for AUC; it is different to several estimators variant at 0.1 level for accuracy and AUC. The overall best method is therefore weighted voting, while it looks that using several estimators is beneficial only for problems with strong conditional dependencies.

The t-test for this comparisons (and for the differences on the development data sets as well) would declare all the differences more significant.

6 Conclusions

We investigate possibilities to improve the performance of random forests. First we propose the use of several attribute evaluation measures for split selection in the process of building the trees. This procedure decreases the correlation between the trees and retains their strength which results in slight increase of the method's performance. The improvement is especially visible on data sets with highly dependent attributes and the reason for this is the use of ReliefF algorithm.

The most important improvement in the performance of random forests is achieved by changing the mechanism of voting. When classifying a new instance with ordinary voting each tree casts a vote for one of the classes and the forest predicts the winning class. We propose first to estimate the average margin of the trees on the instances most similar to the new instance and then, after discarding the trees with negative margin, weight the trees' votes with the margin. Evaluation on several data sets has shown that this approach significantly improves accuracy and AUC.

Table 5. Random forests performance on fresh data sets.

	Gini		5 estimators		weighted voting		5est & wv	
name	accuracy	AUC	accuracy	AUC	accuracy	AUC	accuracy	AUC
adult	0.849	0.835	0.858	0.808	0.849	0.835	0.857	0.797
audiology	0.736	0.862	0.726	0.889	0.784	0.883	0.779	0.913
banding	0.738	0.606	0.716	0.553	0.796	0.687	0.760	0.677
credit aus	0.867	0.936	0.874	0.936	0.868	0.937	0.871	0.937
heart	0.830	0.886	0.837	0.890	0.833	0.889	0.848	0.897
hepatitis	0.833	0.717	0.858	0.735	0.858	0.755	0.851	0.769
house	0.956	0.988	0.956	0.988	0.956	0.988	0.956	0.988
iris	0.953	0.990	0.953	0.990	0.960	0.991	0.960	0.991
lymphography	0.817	0.895	0.824	0.884	0.824	0.909	0.857	0.916
mushroom	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
post-operative	0.656	0.669	0.700	0.669	0.678	0.689	0.711	0.669
primary tumor	0.440	0.740	0.439	0.723	0.460	0.722	0.481	0.701
promoter	0.914	0.943	0.885	0.955	0.923	0.961	0.953	0.991
rheumatism	0.698	0.668	0.690	0.682	0.701	0.680	0.690	0.692
spambase	0.952	0.985	0.953	0.984	0.952	0.985	0.952	0.984
splice-jxn	0.970	0.994	0.969	0.994	0.970	0.994	0.966	0.994
yeast	0.611	0.815	0.619	0.823	0.611	0.821	0.614	0.811

Systematical testing of the number of similar instances used in the weighting remains further work. The results indicate that there is a room for improvement when using several attribute evaluation measures, but a further study of which estimators to use and how to combine them is necessary. Also a verification of the performance of weighted voting in the regression context is needed.

Acknowledgements

I thank Leo Breiman for valuable insights into the inner workings of the method published at his web site and for invited talk he gave at ECML/PKDD 2003 which initiated my interest in random forests. This work was supported by Slovene Ministry of Education, Science and Sport through the research programme P2-0209.

Bibliography

[1] Leo Breiman. Bagging predictors. *Machine Learning Journal*, 26(2):123–140, 1996.

[2] Leo Breiman. Random forests. *Machine Learning Journal*, 45:5–32, 2001.

[3] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*. Wadsworth Inc., Belmont, California, 1984.

[4] Janez Demšar. Statistically correct comparison of classifiers over multiple datasets, 2004. (submitted).

[5] Thomas G. Dietterich, Michael Kerns, and Yishay Mansour. Applying the weak learning framework to understand and improve C4.5. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference (ICML'96)*, pages 96–103. Morgan Kaufmann, San Francisco, 1996.

[6] Yoav Freund and Robert E. Shapire. Experiments with a new boosting algorithm. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference (ICML'96)*. Morgan Kaufmann, 1996.
[7] David J. Hand and Robert J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning Journal*, 45:171–186, 2001.
[8] Igor Kononenko. Estimating attributes: analysis and extensions of Relief. In Luc De Raedt and Francesco Bergadano, editors, *Machine Learning: ECML-94*, pages 171–182. Springer Verlag, Berlin, 1994.
[9] Igor Kononenko. On biases in estimating multi-valued attributes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1034–1040. Morgan Kaufmann, 1995.
[10] David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. *Neurocomputing*, 55:169–186, 2003.
[11] Patrick M. Murphy and David W. Aha. UCI repository of machine learning databases, 1995. http://www.ics.uci.edu/ mlearn/MLRepository.html.
[12] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.
[13] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning Journal*, 53:23–69, 2003.
[14] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In Douglas H. Fisher, editor, *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*, pages 322–330. Morgan Kaufmann, 1997.
[15] Jerrold H. Zar. *Biostatistical Analysis (4th Edition)*. Prentice Hall, Englewood Clifs, New Jersey, 1998.