JINAV: Journal of Information and Visualization *ISSN: 2746-1440 (online)*
Vol. 1 No. 1 (2020)
https://doi.org/10.35877/454RI.jinav149

Weighted Bagging in Decision Trees: Data Mining

Yousef M. T. El Gimati

Statistics Department, Faculty of Science University of Benghazi, Libya, yousef.elgimati@uob.edu.ly

Abstract

The main focus of this paper is on the use of resampling techniques to construct predictive models from data and the goal is to identify the best possible model which can produce better predications. Bagging or Bootstrap aggregating is a general method for improving the performance of given learning algorithm by using a majority vote to combine multiple classifier outputs derived from a single classifier on a bootstrap resample version of a training set. A bootstrap sample is generated by random sample with replacement from the original training set. Inspired by the idea of bagging, we present an improved method based on a distance function in decision trees, called modified bagging (or weighted Bagging) in this study. The experimental results show modified bagging is superior to usual majority vote. These results are confirmed by both real data and artificial data sets with random noise. Modified bagged classifier performs significantly better than usual bagging on various tree levels for all sample sizes. An interesting observation is that the weighted bagging performs somewhat better than usual bagging with sumps.

Keywords: Resampling; bagging:, bootstrapping:, decision tree.

1. Introduction

A typical classification problem might involve a feature or input vector x and a response variable y, where $y \in \{1, 2...J\}$ labels the class, in which an algorithm is used on a given set of instances to produce a discrimination rule. The goal of classification algorithm is to be able to predict y from x with low prediction error on a separate data set, where the pair (x, y) follows some unknown distribution. A collection of instances with known values of all the attributes is referred to as a training set, $\ell = \{(x_i, y_i)\}, i = 1,..., n\}$. Attributes give numerical values which describe an observation, these values could be categorical or numerical. Ordered or numerical attributes can have either discrete or continuous values, while categorical attributes not having a natural order. The categories to which cases or instances are to be assigned are called class. Usually the class is a grouping of similar observation items. Based on the training set and a selected algorithm, a "classifier/model" $c(x)$ is devised. This classifier $c(x)$ is later used to predict the unknown value of the attribute for a new instance. Recall that classification problem involves predicating a class based on the attributes of a training sample. Methods for voting classification algorithms have succeeded in reducing classification error. A voting algorithms (final classifier) is composed of base classifiers (committee members), each of which makes its own classifications that are combined to create a single classification result of the whole committee, which leads to its name.

Two typical methods for voting classification approaches, bagging or **b**ootstrap **agg**regat**ing** (Breiman, 1996a) and boosting (Schapire, 1990) have been shown to be very successful in improving the accuracy of certain classifiers for both artificial and real data. Both methods resample training sets from the original data set and then a classifier is created from each training set. There are two major differences between bagging and boosting. Those that adaptively change the distribution of the training set based on the performance of previously created classifiers such as in the boosting algorithm and those that do not (change the distribution of the training set randomly) such as in bagging.

* Corresponding author.
E-mail address: yousef.elgimati@uob.edu.ly (Yousef M. T. El Gimati)

Another difference is boosting uses a function of the performance of a classifier as a weight for voting, while bagging uses equal weight voting. For example, in decision trees each tree is given a vote towards the classification of a test set, maintaining diversity in the ensemble by allowing each tree to contribute equally. Bagging does not measure each tree performance, whereas boosting does. With both bagging and boosting the error generally decreases when the size of committee instances increase.
The approach discussed in this paper will be confined to bagging technique which is actually yield better classifier performance – then learning one model over the entire training set. Bagging or Bootstrap aggregating is a method of randomly sampling with replacement from the original training set. This procedure is repeated *B* times (say *B=100*) and the resulting *B* classifiers are aggregated by a majority vote, which is in some sense equivalent to a "median" classifiers as described in Section 5, for one-dimension example. Here, the idea of bagging is extended to averaging the classifiers based on a *distance function* in decision tree, instead of a majority vote. It is therefore necessary to compare the averaging classifiers with a majority vote to ensure that a good classifier is optimal for providing a discrimination rule that minimizes the misclassification error rate on a separate test set.

1.1. Notations

It is an importance to introduce some notations to avoid ambiguity. However, some notation will be introduced at various places in this paper. Table 1 presents some notations used.

Table 1. Some notations to avoid ambiguity

Notations	Description
$\Re$ *or* Ω	Measurement space;
X	Measurement vector;
$y_i \in \{1, 2...J\}$	Class of observation i and $i = 1,2,...,n$;
ζ_j	Class label, where $j=\{1, 2,...,J\}$;
ℓ	Training set or data set;
N	Training set observations $i = 1,2,...,n$;
n_o	Test set observations $i = 1,2,..., n_o$;
$n_j, j=\{1, 2,...,J\}$	The number of observation belonging to class j;
ℓ^*	Bootstrap version of the training set ℓ;
$c(x)$	Single classifier;

1.2. Scenario of training and test sets

The classification problem is characterized by the following: one has two types of multivariate observations- the first, called trading samples, are those whose class identity (i.e., membership in a specific one of *J* given classes is known a priori), and the second type, referred to as test samples, consists of observations such as priori information is not available and which have to be assigned to one of the *J* classes.
A training set ℓ is used to construct a classifier. A classifier is a mapping or a function from the sample space to the labels. For simplicity, for each instance x a classifier $c(x)$ is equal one of the known classes *J*. the construction of a classifier $c(x)$ can have two purpose: (1) To predict the response variable corresponding to future measurement vectors as accurately as possible, (2) To understand the structural relationships between the response and the measured variables. The performance of a classifier on the training set which has been used both to construct and to evaluate the classifier will be overly optimistic. An obvious way to see how accurately a classifier performs is to test the classifier on a data set not used to construct $c(x)$ – this is usually referred to as the test set. The overall purpose of using training and test sets to construct and test the classifier is to provide an unbiased estimate of the error rate.

1.2.1. Under –fitting and over-fitting issues

From a statistical perspective, two concepts related to bias and variance are the concepts of under-fitting and over-fitting to the training set (Breiman, 1996b). We would like to construct our model; which gives a good bias-variance tradeoff in such a way as to minimize the generalization error. Under-fitting can be described as a situation of large bias, because the model is not complex enough. In contrast, over-fitting refers to a situation in which the model adapts itself too closely to the training set leads to high variance. For example, in decision tree, too few terminal nodes result in a poor error generalization leads to large bias due to the lack of complexity (number terminal nodes). On the other hand, with too many terminal nodes, we get a poor error generalization, i.e. high variance.
The choice is critical: if the complexity of the tree is too small or too large then tree will tend to under-fit or over-fit the data. In order to overcome this problem, the tree has to be large enough not to introduce under-fitting and then a pruning procedure is often utilized to obtain a right sized tree that does not over-fit the data. Thus, bagging can be applied which is a variance reduction technique in likely the variance among a set of independent classifiers can be reduced by voting or averaging classifiers.

1.3. Pruning procedure

The main purpose of pruning a tree is to produce an optimal tree that does not over-fit the data. Almost all methods of pruning perform a post-order traversal of the tree, replacing a sub-tree by a single leaf node when the deviance or the error rate of the leaf replacing the sub-tree is lower than that of the sub-tree. A cost-complexity measure (Breiman et. al., 1984) is the one of the most popular and appealing pruning methods. We need is to investigate various effects of the degree of pruning which can be done by using cross-validation strategy.

1.3.1. Cross-validation for choosing tree complexity

Dividing the original data set into V mutually exclusive subsets, each of which will serve as an independent test for trees grown on training sets composed of $(V - 1)$ remaining subsets. We can predict on that set, and compute the deviance versus a number for the pruned tree. For example, if $V = 10$, we can then use 9 subsets to grow the tree and test it on the tenth. This can be done in 10 ways and we can average the results. The average is done over 10 trees for fixed number for the pruned tree and not fixed tree size.

To illustrate, Fig. 1 shows a classification tree after pruning using the deviance measure of impurity for the *iris* data (Fisher, 1936). The data consist of *50* samples from each of three different species of the flower. Observations on four variables (*sepal length & width and petal length & width*) are given for each sample. The Fig. 1 shows a pruned version with cost-complexity determined by *10-fold* cross-validation and this gives misclassification error rate *4%* with three terminal nodes. The figure shows the tree splits first on *petal. length*< 2.45 at the root, which gives a pure node at left child of 50 observations, all classified to class *setosa*, while the right child, *petal.length*$\geqslant 2.45$, with *100* observation from the remaining two classes, which need further sub-division. The second split, based on *petal.width*< 1.75, classified *49* observations to class *verycolour*, with *5* observations misclassified, whereas *petal.width* > 1.75 classifies *45* observations to class *virginica*, with *1* observation misclassified. Note that only two variables were used in the final tree.

Fig. 2 shows that another way of looking at the tree structure procedure is a recursive partitioning of the data into rectangles, with each rectangle corresponding to the terminal node or leaf in the tree.

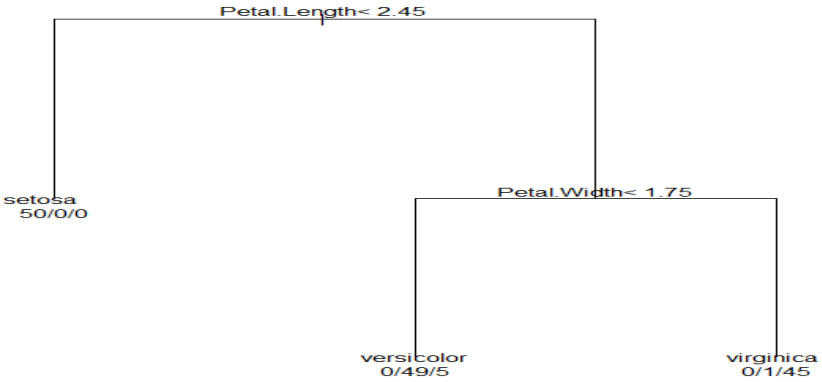

Fig. 1. The label under each terminal node gives the class label *of setosa, versicolor* and *viginica* respectively in the node. The tree was pruned with 10-fold cross-validation.

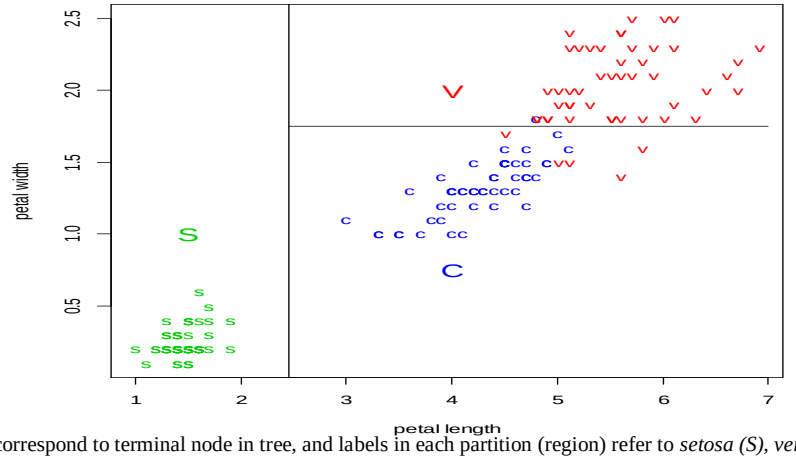

Fig. 2. Partition plot correspond to terminal node in tree, and labels in each partition (region) refer to *setosa (S), versicoloure (C)* and *virginica (V).*

2. The non-parametric bootstrap technique

In general, statistical methods can be classified as parametric or non-parametric though sometimes the distinction is less clear. The former involves the use of particular mathematical model. When no such mathematical model is used, the statistical analysis is non-parametric and uses only the fact that the random variables are independent and identically distributed (or *i.i.d.*). For simplicity, suppose that we have observed a data sample $\ell = (x_1, x_2,..., x_n)$, the outcomes of independent and identically distributed random variables $X_1, X_2,..., X_n$, possibly vector-valued, whose common cumulative distribution function (cdf) is F, where F is completely unspecified. A popular tool is non-parametric analysis is the bootstrap which gives inference about a population characteristic by using resampled data sets (Efron and Tibhirani, 1993).

2.1. Bootstrap technique: Background

An important role is played in non-parametric analysis by the empirical distribution which puts equal probability mass at each sample value x_i, $i = 1,2,..., n$. The corresponding estimate of F is the empirical distribution function of data ℓ and defined as

$$\hat{F} = \frac{\text{Number of } x_i \le x}{n} \qquad (1)$$

Note that, this is the cdf of a distribution that assigns probability $\frac{1}{n}$ to each of n samples values $xi = 1,2,..., n$ and are a random sample from some distribution F,

$$\ell = (x_1, x_2, ..., x_n) \overset{iid}{\sim} F \qquad (2)$$

A bootstrap sample can be defined to be a random sample of size *m* drawn from *F*,

$$\ell^* = (x_1^*, x_2^*, ..., x_m^*) \overset{iid}{\sim} \hat{F} \qquad (3)$$

The asterisk indicates that ℓ^* is not the actual data set ℓ, but rather a randomized, or resampled (with replacement), version of the actual data set ℓ. Many sample statistics can be thought of in terms of properties of the empirical distribution function F. For example, the sample means $\bar{x}^* = \frac{1}{m}\sum_{i=1}^{m} x_i$. Thus, we can think of the bootstrap procedure as a computational way for obtaining a good approximation to the numerical value of a parameter estimate. The bootstrap technique can readily be implemented as a computer program. First, we need a random number device to select integers each of which equals any value between *1* and *n* with probability $\frac{1}{n}$. Then, the bootstrap sample consists of the corresponding members of data set ℓ.

$$x_1^* = x_{i1},\ x_2^* = x_{i2}, ..., x_m^* = x_{in} \qquad (4)$$

where, *i = 1,2,..., n* is any integer number between 1 and *n*. the bootstrap sample consists of elements of the original data set ℓ, some elements appearing zero times, some elements appearing once, etc. second, the bootstrap algorithm works by drawing many independent bootstrap samples, evaluating the corresponding bootstrap summarizes and estimating the parameter by the empirical sample of the replications, i.e.

1. Select *b, b = 1, ..., B* independent bootstrap samples $\ell^*_1, \ell^*_2, \ell^*_3 ... \ell^*_B$, each consisting of *m* data values drawn with replacement according to Eq. (3);
2. Evaluate the bootstrap sample parameter corresponding to each bootstrap sample;
3. Estimate the parameter by the bootstrap sample of the *B* replications.

Having presented the underlying logic behind the bootstrap technique, we are in a position to introduce a modified version of the non-parametric bootstrap technique in the context of the prediction of the error rate for decision trees.

2.2. The bootstrap: Estimating the error rate

A set of data (training data set) ℓ consisting of $\ell = \{\ell_1, \ell_2, ..., \ell_n\}$ is given which is previously classified data. A labeled instance is a pair $\ell_i = \{(x_i, y_i), i = 1, 2... n,$ where x_i represents an attribute vector and y_i are either class label in the set $\{1,2, ..., J\}$ (classification) or are numerical (regression) associated with x_i for given instance ℓ_i . here, we are mainly concerned with the situation where y_i is binary. Suppose $\ell_i = \{(x_i, y_i)$, in the training set ℓ are a random sample from some distribution *F* (Davison and Hinkley, 1997), such that

$$\ell = (\ell_1, \ell_2, ..., \ell_n) \overset{iid}{\sim} F \qquad (5)$$

Let $\hat{F}$ be the empirical distribution putting $\frac{1}{n}$ mass on each observed case

$$\ell^* = (\ell^*_1, \ell^*_2, ..., \ell^*_m) \overset{iid}{\sim} \hat{F} \qquad (6)$$

Suppose ℓ^* be a random sample of size m taken *iid* with replacement from $\hat{F}$ where $l^{*b} = \{(x_i^*, y_i^*)$ is a single random observation. This procedure fits the model in question on a set of bootstrap samples, $\boldsymbol{l}^{*b}$, *(b = 1,..., B)*. If $C_{l^{*b}}$ is the predicted value at from the model fitted to the b^{th} bootstrap sample, our estimate is defined as

$$\hat{Err}_1^{boot} = \frac{1}{B}\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n} L[y_i, c_{l^{*b}}(x_i)] \qquad (7)$$

2.2.1. Computational procedure

The actual calculation of the estimation error rate each bootstrap sample is a straightforward process. One approach is used an independent test set $l_0 = \{(x_{0i}, y_{0i})$, $i = 1, \ldots, n_0\}$ consists of n_0 observations $l_{0i} = (x_{i0}, y_{i0})$ from the training set, so that our estimate is defined as

$$\hat{Err}_2^{boot} = \frac{1}{B}\frac{1}{n_0}\sum_{b=1}^{B}\sum_{i=1}^{n} L[y_{0i}, c_{l^{*b}}(x_{0i})] \qquad (8)$$

Where L is denoted a suitably chosen loss function for measuring errors between predicated value and the actual response. Here, we are particularly interested in the quadratic or *0/1* loss, because we are mainly concerned with the situation of y_i is binary. This can be done for the artificial example, because the underlying concept is known. In real world data sets (the underlying concept is not known), we generated of size m, as in Eq. (6), from the original training set of size n as in Eq. (5), where $m < n$. A tree is grown using l^{*b}, ($b = 1,\ldots, B$) and tested $(l \setminus l^{*b}) = \{(x_i', y_i')$, $i=1, \ldots, n'(b)\}$ that an observation in l but not in l^{*b}, our estimate is

$$\hat{Err}_3^{boot} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{n'(b)}\sum_{i=1}^{n'(b)} L[y_i', c_{l^{*b}}(x_i')] \qquad (9)$$

From the test set $\ell_{oi} = (x_{oi}, y_{oi})$ from the test set ℓ_o. ordinary bootstrap training sets ℓ^* are generated as in Eq. (6). Therefore, ℓ^* is a random draw of size m with replacement from Eq. (5). Many such samples are independently drawn, say $\ell^*_1, \ell^*_2, \ldots, \ell^*_B$ which bootstrap samples are use as training set individually. The procedure can then be described as follows:

Data is given by $l = \{(x_i, y_i)$, $i = 1, 2 \ldots n\}$, $x_i \in \Re$ and $y_i \in [-1,1]$

For b: 1 to B (B a number of bootstrap samples of size m)

1. Generate ℓ^{*b} from ℓ (*i.i.d.* sample with replacement according to Eq. 5)
2. For the sample ℓ^{*b} a tree grown.
3. Test the tree on the test set $l_0 = \{(x_{0i}, y_{0i})$, $i = 1, \ldots, n_0\}$
4. Evaluate the error rate from each bootstrap sample.

Estimate the error rate by the bootstrap samples of the B replications according to Eq. (8)

3. Bagging: An overview

Breiman's paper (1996a), is about improving the performance of a learning algorithm by using bagging which has been shown to improve classifier accuracy. Bagging is a general method for improving the performance of a given learning algorithm by using a majority vote to combine multiple classifier outputs derived from a single classifier on a bootstrap resample version of training set. A bootstrap sample is generated by taking a random sample with replacement from the original training set. According to this paper, bagging exploits the instability in the classifiers, for example in decision tree classification each bootstrap tree will typically produce different feature than the original and might have a different number of terminal node trees. The bagging algorithm differentiates between trees during the voting process. The procedure can be described as the process by which a learning algorithm generates B bootstrap samples and then a classifier c_b *(x), b = 1,2 ..., B* is built from these samples. A final classifier c_{bag} *(x)* is determined by a majority vote among the base classifiers where ties and broken arbitrarily.

3.1. The general performance

The objective of using bagging in the classification problem in general and in decision tree classification in particular is to improve classification accuracy over a single classification tree. Breiman (1996a) claims that bagging can only improve the predictive accuracy when the training set can cause significant changes in the prediction constructed. The most important aspect is that the variance is a component of the classification error, which can be reduced by aggregation. However, the bias may be larger for the aggregated classifier than for the un-aggregated. Thus, the performance of bagging depends much on the training set (Hastie et. al., 2008). As the training set changes the classifiers c_b *(x), b = 1,2,..., B* can differ markedly from each other and referred as to "unstable" classifier, for example, decision tree and the performance of the aggregated classifier becomes better. In contrast, a stable classifier such as linear discriminant analysis (LDA) does not change much over replicates of the training set.

3.2. Mechanics of the bagging algorithm

In bagging, bootstrapping and aggregating techniques are implemented in the following way: input a training set ℓ which consists of *n* observations $\ell_i = (x_i, y_i)$ with x_i being the feature vector and y_i being the response, taking values in *{1,2, ..., J}*. Draw bootstrap samples $\ell^{*1}, \ell^{*2}, ..., \ell^{*B}$ each of size *m* with replacement from the original training set ℓ.

For *b*: 1 to *B* (*B* a number of bootstrap samples)

1. ℓ^{*b} = bootstrap sample of size m from ℓ (*i.i.d* sample with replacement);
2. Compute the classifier c_b on each bootstrap ℓ^{*b};
3. Output a classifier $\{c_b\}$.

Each observation in ℓ or in a test set is then classified using a majority vote of $c_1, ..., c_b$ classifiers. A final decision rule: $c_{\text{bag}}(x) = \text{argmax}_{y_i \in \{-1,1\}} \sum_{b=1}^{B} \delta_{sign(c_b),y}$ where $\delta_{sign(c_b),y}=1$ if $\delta_{sign(c_b)} = y$ and 0 if $\delta_{sign(c_b)} \neq y$ is the Kronecker delta, *y* is true class label of classifier.

4. Examples

To illustrate how the algorithm works, we utilize two different settings of real and simulated data sets. The bagging is a method for improving the performance of a given learning algorithm, for instance decision tree to produce more accurate results than the conventional decision tree classification. The algorithm works by feeding perturbed versions of the training set to the base classification algorithm and combing the resulting rules by a majority vote.

4.1. Examples setup

The first setting is using a real data set, the total data set is randomly divided into a training set of 300 observations and a test set of 345 observations as described in Section 4.2. The second setting is using simulated a training set of 300 noisy observations with a sinusoidal boundary as described in Section 4.3. Further, we simulated an independent set of 2500 observations as test set by the same way as training set. The tree is grown on the training set and evaluated in the test set without any pruning (full tree), with default pruning and several pruned tree levels using **R-tree** for full tree and using **rpart** for the rest of pruned trees (Crawley, 2014).

A bootstrap sample ℓ^{*b}, *b=1,2..., B* is selected from the original training set ℓ and a tree is grown using ℓ^{*b}. this procedure is repeated 50 times giving tree classifiers $c_1,..., c_{50}$. Bagging then uses all the predicted classes i.e., the combined classifier predicts the class most frequently predicted by the sub-classifiers built from the bootstrap samples.

4.2. Real data example

Our first example employs a real data set collected by EL Ojali in 1994. The data set consists of 510 anæmics. Libyan infant's in their first year of life and 135 normal healthy infants of their same age group as control. The data is randomly divided into training set of 300 observations and the rest of the data of 345 observations as test set. Fourteen predicator variables are available with two classes as response variable given by disease class and control class.

The tree is grown on the training set of 300 observations as exhibited in the LHS of Fig. 3 with misclassification error rate of 7%. The RHS of the same figure shows the tree after pruning (maximum possible pruning) has taken place with an error rate of 14%. Note that, no further pruning is possible for the tree on the RHS of Figure 3 as beyond that level all observations go to one class which is not legitimate tree. Our aim is to improve the performance of the tree algorithm through a "bagged" tree with low generalization, predication error of misclassification rate on independent test set.

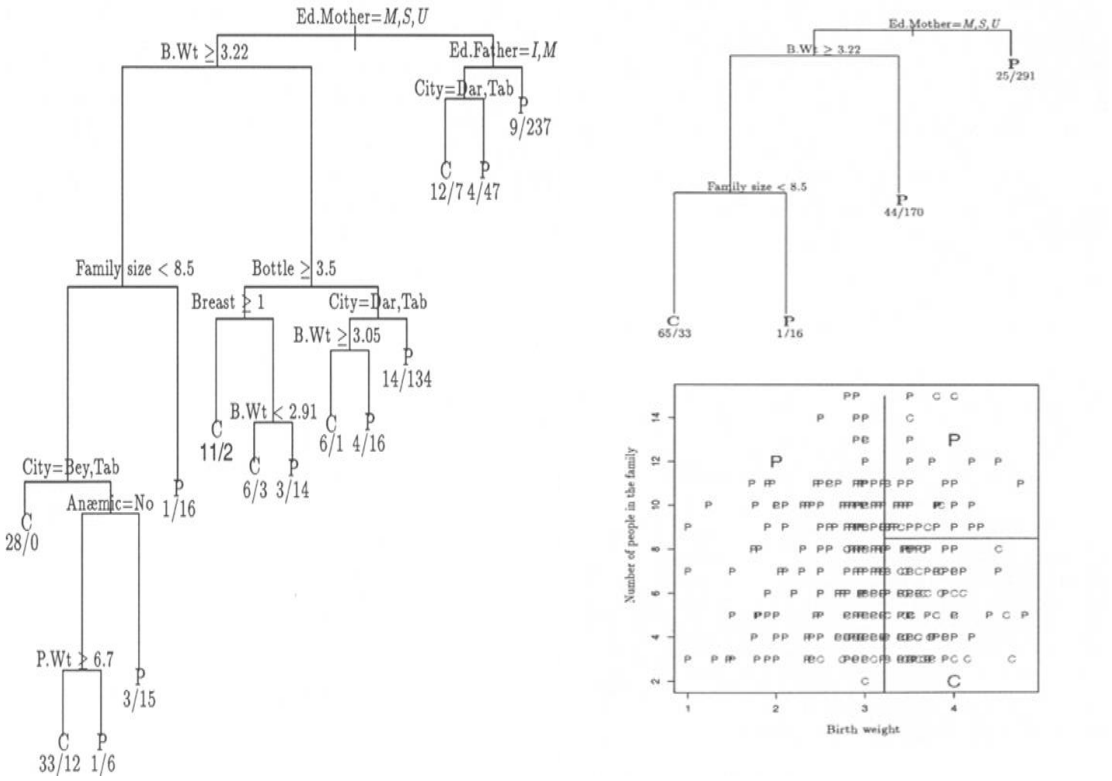

Fig. 3. A decision tree with two classes. Patient (p) and control (C) of Libyan infants. The LHS panel is based on a full tree, while the RHS panel is based on with the strongest possible a pruned tree and partition plot (below) correspond to terminal node in tree.

4.2.1. Results from bagging of real data set

As rule, a pruned tree may produce a higher misclassification error than a full tree. A full tree or large tree may have low misclassification error, but it could be introduced over-fitting. To be able to explore the effects of the pruning procedure of the bagging having low generalization error, various tree levels are used to produce combination of classifiers.

Results from bagging of full and different pruned tree levels with different training sizes are represented in Table 2. These results show that the bagged error estimates on un-pruned trees over the three training sizes produced an average error decrease of about 50.6%, while error decrease of about 47.0% for default pruning, eight and six terminal nodes. The bagged test errors for four terminal nodes produced an average error decrease of about 46.5%, which nearly equal to the other pruned trees.

Table 2: comparing un-bagged and bagged test errors for full trees and on different pruned tree levels.

Tree size	Data size	Ordinary tree test error	Bagged test error (%) Initial	Final	Decrease
Full Tree	125	13.6	21.5	10.4**	51.6
	200	13.6	21.2	10.7**	49.5
	300	13.6	21.7	10.7**	50.7
Default pruned	125	14.2	22.9	11.9**	48.0
	200	14.2	24.1	11.6**	51.9
	300	14.2	19.6	11.6**	41.0
8-node trees	125	13.0	23.0	11.6*	49.6
	200	13.0	24.1	11.8*	51.0
	300	13.0	19.4	11.3*	41.8
6-node trees	125	16.2	23.0	11.0***	52.2
	200	16.2	21.2	12.7***	40.1
	300	16.2	21.5	11.0***	48.8
4-node trees	125	20.0	22.0	11.6***	47.3
	200	20.0	30.7	12.5***	59.3
	300	20.0	19.0	12.5***	34.2

Significant codes: '***' high significant, '**' significant at 1%, '*' significant at 5%.

4.3. Simulation example with random noise

In an attempt to differentiate performance, the simulated example involves a fairly complex decision boundary. The two input features for this example are randomly drawn from a two dimensional uniform distribution on [0,3x]×[-1,1]. The desired output indicates one of two classes, represented by values ± 1, which is defined by $\sin(x_1) < x_2$ as a classifier. A random noise is added to a training set of 300 observations by randomly perturbing the output y to describe precisely the random mechanism, let us define u as random uniform on [0,1], and v (= 0.05) then the class label y is changed if $u < v$.

Note that, an independent test set consisted of 2500 observations generated in the same way as the training set with same amount of random noise added. The plot in Fig. 5 shows sine wave plot with the training set of 300 noisy observations classified into two classes. A full tree is grown and tested on the test as exhibited in Fig. 6 at top-left with 12.0% misclassification error as well as the partition regions at top-right. Moreover, the tree is pruned up to stump (as single split tree with only two terminal nodes) with about 28.1% misclassification error as shown on the lower-left of Fig. 6 as well as the partition decision at the lower-right in the same figure.

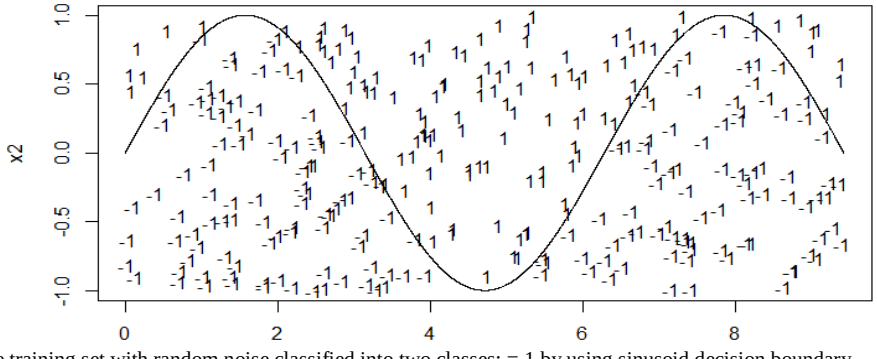

Fig. 5. Plot of the training set with random noise classified into two classes: = 1 by using sinusoid decision boundary

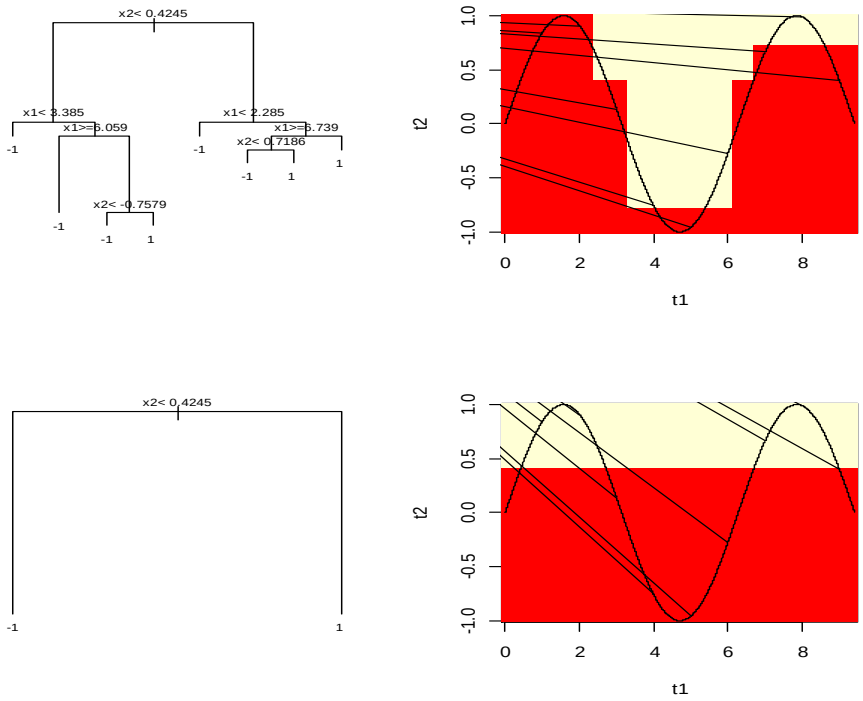

Fig. 6. The top-LHS panel shows the full decision tree and the top-RHS panel shows the partition decision tree. The lower-LHS panel shows the stump, while the lower-LHS panel displays the partition into 2 regions.

4.3.1. Results from bagging of simulation example with noise

As described in section 4.3, the training and test errors are estimated on different pruned tree levels and these estimated are compared with the un-bagged test error. The results are represented in Table 3 for resulting of ordinary tree test errors and bagged test error rate of full tree and different pruned tree levels with different training sizes. These results show that the bagged error estimates produced an average error decrease of 48.3% for default pruning, 12.0% for six and 19.7% for terminal nodes. The bagged test error rates for stumps shows that training sizes 125 and 200 gave better improvement than size 300, especially at the first 15 iterations. However, these result show that bagging did not perform well with stumps.

Table 3: Comparing un-bagged and bagged test errors for full trees and on different pruned tree levels.

Tree size	Data size	Ordinary tree test error	Bagged test error (%)		
			Initial	Final	Decrease
Full Tree	125	12.0	22.9	10.8***	52.8
	200	12.0	16.9	8.3***	50.9
	300	12.0	14.3	8.4***	41.3
Default pruned	125	13.0	16.1	13.8**	14.3
	200	13.0	14.0	12.5**	10.7
	300	13.0	13.5	11.1**	17.7
8-node trees	125	13.0	16.1	14.9*	7.5
	200	13.0	14.0	12.9*	7.9
	300	13.0	13.5	10.8**	20.0
6-node trees	125	14.0	16.1	12.3***	21.4
	200	14.0	14.0	11.9***	19.2
	300	14.0	13.5	*11.7***	18.4
4-node trees	125	28.1	28.2	27.1*	3.9
	200	28.1	28.1	26.7*	5.0
	300	28.1	28.1	28.0	0.3

Significant codes: '***' high significant, '**' significant at 1%, '*' significant at 5%.

4.4. Significance tests for classifications

In the performance between different classifiers made by two or more methods, we may decide that one method is more accurate than the others. The comparison between methods is usually much more accurate than the standard error suggests. In our examples, we tested the significant different between un-bagged and bagged test error by using McNemar's test (Fleiss, 1981), in which n_A and n_B be the number of test errors made by un-bagged method (*A*) and bagged method (*B*) and vice versa. Then McNemar's test is given as

$$\frac{|n_A - n_B| - 1}{\sqrt{n_A + n_B}} \sim N(0,1)$$

The results of these methods are represented in Table 2 of the real data and Table 3 of the simulated data for full trees and different pruned tree levels. It is observed from Table 2 that bagged method has statistically significant results, for example, in full trees of training size of 125 observations, method *B* with bagged error rate 10.4% is significant at 1% better than method *A* with 13.6% un-bagged test error. Furthermore, it can be noticed from Table 3 that the bagged method has also highly statistically significant results, such as in full trees of training size of 125 observations, method *B* with bagged test error rate 10.8% is highly significant better than method *A* with 12.0% un-bagged test error rates.

5. Modified bagging in decision trees

The simplest method of generating multiple classifiers from a training set ℓ is bagging. For this method classifiers are constructed from bootstrap samples of size *m* with replacement from the original training set ℓ. In the context of aggregation of preferences, the classical way to combine multiple classifiers to construct a single classifier is via a simple voting strategy of corresponding output values of the classifiers. For example, in decision trees the majority vote is used in which all trees have the same weight. The majority vote (final classifier) is composed of the sub-classifiers, each of which make its own classifications that are combined composed of the sub-classifiers, each of which make its own classifications that are combined to construct a classification rule result of the whole committee. The majority rule in some sense equivalent to a "**median**" procedure. To illustrate this let us consider the classification example with a two-class problem: and C_2 as class labels. To simplify this example, we assume that x is one-dimension vector of *n* training examples. Also, suppose we have classifier c_m(x), m=1, 2…, M so that is $x < c_m$ (x) then C_1 otherwise C_2. For given a new observation x^* to C_1 if [#m=1,…,M c_m(x) > x*], as majority vote. The majority vote is usually constituted by fifty percent plus one of the classes that has the power to make decisions binding upon the whole. Herein, the majority criterion is equivalent to the median since [#m=1,…,M c_m(x) > $\tilde{c}_m$] =$M/2$, so that if $x^* < c_m$ then x^* is classified to C_1 otherwise to C_2, where C_m is the median of $c_m(x)$, m=1,…, M.
However, the evaluation of moments, such as the variance is an important property to compare the averaging classifier with the median classifier to ensure that a good classification rule has low variance for providing minimum error rate. Let us denote $\tilde{\mu}$ is the median of the population, standard asymptotic theory reveals that $(4n[\, f(\tilde{\mu})^2)^{-1}$ as an estimator of var($\tilde{\mu}$), if we knew the density function. For example, in the case of normal distribution, we have $\tilde{\mu}$ =μ, so that $f(\tilde{\mu}) = f(\mu) = \frac{1}{\sigma\sqrt{2\pi}}$, the variance of the median. In other words, the mean is subject to smaller chance fluctuation than the median.

In this work, we modify usual bagging to improve accuracy by using mean distance between two different class representations. For this purpose, we use so called a *distance function*. This function takes as arguments *x*'*s* and returns a number which indicates how far apart the corresponding labels are of each other.

5.1. Distance: Background

The most popular choice is *Euclidean distance*, defined by

$$d(x_i,x_j)= \sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+....+(x_{ip}-x_{jp})^2} \quad (10)$$

Another well-known metric is the Manhattan distance or city block, defined by

$$d(x_i, x_j) =| x_{i1}-x_{j1} |+|x_{i2}-x_{j2} |+...+|x_{ip} - x_{jp}| \quad (11)$$

Formula (10) and (11) satisfy the following constraints of a distance function:

1. $d(x_i,x_j) = 0$, if and only if $x_i = x_j$
2. $d(x_i,x_j) = d(x_j, x_i)$
3. $d(x_i,x_j) + d(x_j, x_i) \geq d(x_i, x_h)$, for all objects x_i, x_j and x_h

The Manhattan metric is based on the distance that people have to walk between two points in city block. The Euclidean, metric is based on the usual geometrical distance. The metrics presented here can be used in our work for computing the distance between two objects in two dimensions. In this work, the Manhattan distance can be used for full trees and different pruned tree levels, because the classes are portrayed in perpendicular direction. However, the two metrics are equivalent when the tree has a single split.

5.2. Mechanics of the modified bagging algorithm

The procedure of the process of modified bagging will perform the following steps. Given a training set ℓ which consists of n observations $\ell_i = (x_i, y_i)$ with x_i the feature vector in two dimensions and $y_i \in [-1,1]$. Draw bootstrap samples $\ell^{*1}, \ell^{*2}..., \ell^{*B}$ each of size m with replacement from the original training set ℓ. Let z be a *2D* array covering the sample values x, and define the distance between z_1 and z_2 as $d(z_1, z_2) =| z_{11} - z_{21}|+ |z_{12} - z_{22}|$.

For b: 1 to B (B a number of bootstrap samples)

1. Draw ℓ^*b = bootstrap sample of size m from ℓ (i.i.d. sample with replacement)
2. Compute the classifier $c_b(z)$ for $z \in \Re^2$
3. For each z^*, let $D(z^*) = \min [\min_z (d(z^*, z|c_b(z) \neq {}_(z^*), \min_z (d(z^*, z|z \text{ on edge of } Z)]$

$c_{bag}(z^*) = \text{sign}\left[\sum_{b=1}^{B} D \times c_b(z^*)\right]$, where sign(.)=±1 depending on the sign of its argument.

5.3. Simulated example

In order to evaluate the performance of modified bagging and compare it with the usual bagging, we simulated the same example setup that used in Section 4.3. For this example, we generate bootstrap *samples* $\ell^{*1}, \ell^{*2}..., \ell^{*50}$ of 125,200 and 300 observations from the training set each corresponding the test set. Experimental results and analysis are presented as follows:

5.3.1. Significant differences between modified and usual bagging

The statistical significant differences between the two methods are usually much more accurate than suggested by the test error. In this section we follow the same procedure as in Section 4.4. the results of these two methods are represented in Table 4 for full trees and different pruned tree levels with several training sizes. It can be seen that modified bagging has statistically significant results for reducing the test error for different pruned tree levels with all

training sizes this is the main motivations behind this work as it provides empirical evidence that model averaging distance can reduce the generalization error of the over- all classifier.

Table 4: Comparing bagged and modified bagged test errors for full trees and on different pruned tree levels.

Tree size	Data size	Usual bagged test error(%)	Modified bagged test error(%)	**Decrease**
Full Tree	125	10.8	9.9	8.3*
	200	8.3	7.7	7.2*
	300	8.4	7.6	9.5*
Default pruned	125	13.8	9.8	28.9***
	200	12.5	10.6	15.2***
	300	11.1	10.2	8.1*
8-node trees	125	14.9	11.3	24.2***
	200	12.9	10.6	17.8***
	300	10.8	10.2	5.6*
6-node trees	125	12.3	9.5	22.8***
	200	11.9	9.7	18.5***
	300	11.7	9.9	15.4***
4-node trees	125	27.1	24.1	11.1***
	200	26.7	25.0	7.4**
	300	28.0	26.5	5.3**

Significant codes: '***' high significant, '**' significant at 1%, '*' significant at 5%.

6. Conclusion

Combining several decision classifiers in a committee can provide an improvement over the use of a single. From our investigation of the bagged classifier for decision tree to improve the classification accuracy, one can see that bagging performs significantly better than a single decision in our analysis. These results are confirmed by both real data and artificial data sets with random noise. We also observed the increased sample size, the degree of improvement increased, while the degree of improvement decreased with stump. These observations suggested that bagging with large tree and a slight amount of pruned tree delivers better performance. The idea of bagging is extended to averaging the classifiers based on the Manhattan distance. In this paper, we focused on building decision trees, and experimental results and analysis show that weighted bagged classifier performs significantly better than usual bagging on various tree levels for all sample sizes. An interesting observation is that the weighted bagging performs somewhat better than usual bagging with sumps.

6.1. Potential research directions

In the work, our weighted bagging is only in the case of a two-dimensional predicator space, but what about the more practically interesting case of several variables? This we leave as an open research problem.

Acknowledgements

Firstly, I express my gratitude to Pro. Charles Taylor (Leeds University) for his scientific advice of this work, secondly, we would like to acknowledge the Research and Consulting Centre (RCC), University of Benghazi, Libya for funded this work.

References

Breiman, L. and Friedman, J. H. and Olshen, R. and Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, California.

Breiman, L. (1996a). Bagging predictors. *Machine Learning* **26**, 123-40.

Breiman, L. (1996b). Bais, variance and arcing classifier. *Technical report*, Statistics Department, University of California, Berkeley.

Crawley M. J. (2014). *The R Book*. John Wiley & Sons, Ltd.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Method and their Application*. Cambridge University Press.

Efron, B. and Tibhirani, R. (1993). *An Introduction to The Bootstrap*. Chapman & Hall, London.

El Ojali, (1994). *Pattern of anæmic among Libyan infants of northeastern Libya*. [Unpublished Master's dissertation]. Medicine Department, University of Garyounis.

Fisher, R. A. (1936). The use of multiple measurements in Taxonomic problems. Annals of Eugenics **7**, 179-80.

Fleiss, J. L. (1981). *Statistical methods for rates and proportions*. New York: Wiley.

Hastie, T. and Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning - Data Mining*, Inference, and Prediction. Springer Series in Statistics. New York.

Schapire, R. E. (1990). The strength of learnability. *Machine Learning* **5**, 197-227.