Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 195 (2008) 618–629
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

An empirical study of using Rotation Forest to improve regressors

Chun-Xia Zhang [a,*], Jiang-She Zhang [a], Guan-Wei Wang [b]

[a] *Faculty of Science, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China*
[b] *School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China*

Abstract

This paper investigates the performance of Rotation Forest ensemble method in improving the generalization ability of a base predictor for solving regression problems through conducting experiments on several benchmark data sets, which is also compared with that of Bagging, Random Forest, Adaboost.R2, and a single regression tree. The sensitivity of Rotation Forest to the choice of parameters included in it is also studied. On the considered regression data sets, Adaboost.R2 is seen to generally outperform Rotation Forest and both of them are better than Random Forest and a single tree. With respect to Bagging and Rotation Forest, it seems that there is not a clear winner between them. Furthermore, pruning the tree seems to have some bad effect on the performance of all the considered methods.

Keywords: Rotation Forest; Adaboost.R2; Bagging; Random Forest; Principal component analysis

1. Introduction

Ensemble or committee machines [9,31] currently have received much attention in the machine learning community and the constituent members of a committee machine are generally termed as base predictors. In a committee machine, an ensemble of base predictors is firstly generated by means of applying a base learning algorithm to different distributions of the training data, and then the predictions from each ensemble member are combined suitably to predict a new example. Two popular ensemble machine generation methods are Bagging [5,7] and Boosting [14,15,20,22,28,29]. The experiments conducted by many researchers [2,7,10,19,23,27,32] have demonstrated that the ensemble machines constructed by Bagging and Boosting are generally much more accurate than their constituent members when the base learning algorithm is taken to be an instable algorithm. Here an instable learning algorithm refers to that small permutations in its training data or in construction can lead to large changes in the constructed predictor [5]. The base learning algorithm commonly used in building ensemble machines are neural networks [21] and decision trees [4].

* Corresponding author.
E-mail addresses: cxzhang@mail.xjtu.edu.cn (C.-X. Zhang), jszhang@mail.xjtu.edu.cn (J.-S. Zhang), waldowgw@163.com (G.-W. Wang).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2007.05.010

Although Bagging and Boosting both combine the outputs from different predictors that are trained on the permutated training sets, they differ substantially in the ways to permutate the training data and to combine the predictions coming from their constituent members. Bagging takes different bootstrap samples [13] from the original training set and trains a predictor on each sample to build its constituent members, which can be generated in parallel. Boosting is a sequential algorithm in which each new base predictor is built by taking into account the performance of the previously generated predictors. Initially, a base predictor is constructed by applying the given base learning algorithm to the training data set with equal weights assigned to each training instance. In the subsequent iterations, the training data with weights updated according to the performance of the previously built base predictors are provided as the input of the base learning algorithm. Furthermore, the final decision of Bagging is constructed as combining the predictions of each base predictor with equal weights whereas that of Boosting is formed by a weighted voting scheme: the weight of each base predictor is determined by its performance on the training set used to build it.

The boosting algorithm [28] was originally developed for solving binary classification problems and Freund and Schapire [15] extended it to a multi-class case, which they called Adaboost.M1 and Adaboost.M2. So far there are many different versions of boosting algorithm for solving classification problems [7,10,19,22,23,29,32] and they have formed a family of methods, the most prominent member of which is Adaboost [15].

With respect to the ensemble methods for solving regression problems, Freund and Schapire [15] extended Adaboost.M2 to Adaboost.R, which reduces regression problems to the corresponding classification ones. Breiman proposed arc-gv (that is, arcing game values) [6] and Random Forest [8] algorithms for handling regression problems. Drucker [11] developed Adaboost.R2 algorithm which is an ad hoc modification of Adaboost.R and some promising results with Adaboost.R2 to solve regression problems were also obtained. Avnimelech and Intrator [1] extended the boosting algorithm to deal with regression problems by introducing the notations of weak and strong learning and an appropriate equivalence theorem between them. Ridgeway et al. [25] suggested mapping the regression problem into a classification one, applying their boosted Naïve Bayes classifier, and transforming the resulting classifier back as a regressor. Shrestha and Solomatine [30] proposed Adaboost.RT which firstly employs a pre-set relative error threshold value to demarcate predictions to be correct and incorrect, and the following steps are same as those in Adaboost that solves binary classification problems. Furthermore, many researchers [12,17,19,18,26] have viewed boosting as a gradient machine that optimizes some sort of a loss function. It should be mentioned, however, that there is not a single boosting algorithm that has been found to be a clear winner in solving regression problems.

Rodríguez et al. [27] recently proposed Rotation Forest, a new classifier ensemble method, based on principal component analysis (PCA) and showed that this method performs much better than other several ensemble methods on some benchmark classification data sets available from the UCI repository [3]. Its main idea is to simultaneously encourage diversity and individual accuracy within the ensemble: diversity is promoted by using PCA to do feature extraction for each base classifier and accuracy is sought by keeping all principal components and also using the whole data set to train each base classifier.

As far as we know, there are not yet any published results of this method for regression and it is still not known whether Rotation Forest is also successful in improving the generalization ability of a base predictor for solving regression problems. Therefore, we investigate this problem in this paper. On several benchmark regression data sets, the performance of Rotation Forest is examined by some experiments through respectively adopting a pruned or a non-pruned regression tree as the base learning algorithm, and compared with that of Bagging, Random Forest and Adaboost.R2, and with that of a single regression tree. The sensitivity of Rotation Forest to the choice of parameters included in it is also studied. The results show that number of attributes that each subset contains has some influence on the performance of Rotation Forest whereas the ensemble size has trivial effect as long as it is chosen to be not too small. On the considered regression data sets, Adaboost.R2 generally outperforms Rotation Forest and both of them are better than Random Forest and a single tree. There is not a clear winner between Bagging and Rotation Forest. Furthermore, it seems that pruning the tree has some bad effect on the performance of all the considered methods.

The remainder of the paper is organized as follows. The detailed algorithm about how to solve regression problems using the Rotation Forest ensemble method is described in Section 2. Section 3 presents the experimental studies and Section 4 offers the conclusions of this paper.

2. Rotation Forest regressor ensemble method

Before we embark on employing Rotation Forest to solve regression problems, we firstly introduce some notations in order to facilitate the following descriptions. Consider a training set consisting of N labeled instances $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ in which each instance $(\mathbf{x}_i, y_i)$ is described by n input attributes and an output attribute, that is, $\mathbf{x} \in \mathbb{R}^n$ and $y \in \mathbb{R}$. The goal is to use the information only from $\mathscr{L}$ to construct predictors that have good generalization ability, namely, perform well on previously unseen data. Let X be an $N \times n$ matrix consisting of the values of n input attributes for each training instance and Y be an N-dimensional column vector containing the outputs of each training instance in $\mathscr{L}$, which means that $\mathscr{L}$ can be expressed as concatenating X and Y horizontally, that is, $\mathscr{L} = [X\ Y]$. Denote by $C_1, C_2, \ldots, C_T$ the regressors included in the ensemble machine and by $F = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$, the attribute set composed of n input attributes.

Similar to most other ensemble generation methods, number of base regressors, namely T, that are used to construct the ensemble should be specified in advance. Furthermore, another parameter K which specifies the number of subsets that the attribute set F should be split into needs also to be given. Each regressor C_i $(i = 1, 2, \ldots, T)$ constituting the ensemble is built by applying a given base learning algorithm to different training sets. In order to construct the training set for each regressor C_i, the following steps are necessary:

Step 1: Randomly split F into K subsets. In fact, we can also instead fix the number of input attributes, say M, in each subset. If n cannot be divided by M, let the remainder attributes constitute a subset. For simplicity of notations, suppose that M is a factor of n so that F is decomposed into K subsets.

Step 2: Denote by $F_{i,j}$ the jth attribute subset used for training the regressor C_i. For each subset $F_{i,j}$ $(j = 1, 2, \ldots, K)$, select the corresponding data in matrix X to constitute a new matrix, say $X_{i,j}$, and then draw a bootstrap sample $X'_{i,j}$ (with sample size generally smaller than that of $X_{i,j}$) from $X_{i,j}$. Use the matrix $X'_{i,j}$ to do PCA and store the coefficients of all computed principal components into a new matrix $D_{i,j}$ (an $M \times M$ matrix).

Step 3: Organize each $D_{i,j}$ $(j = 1, 2, \ldots, K)$ into a block diagonal matrix R_i whose jth diagonal element is $D_{i,j}$, and then rearrange the rows of R_i so that the order of them corresponds to that of the original attributes in F. If denote by R_i^a the obtained rotation matrix, the training set for regressor C_i is $[XR_i^a\ Y]$.

Fig. 1 presents the pseudocode of how to apply the Rotation Forest ensemble method to improve regressors.

As pointed out in [21,24,27], for an ensemble machine to achieve better generalization ability than a single predictor, it is critical that the ensemble machine consists of highly accurate members while at the same time disagree as much as possible. In the above algorithm, it should be noted that for each subset $F_{i,j}$ $(i = 1,2, \ldots, T;$ $j = 1, 2, \ldots, K)$, PCA is only applied on a subset of the training data set $X'_{i,j}$ instead of on the whole set $X_{i,j}$. The aim to do so is twofold: one is to avoid obtaining identical principal component coefficients when the same attribute subset is selected for different regressors and the other is to encourage the diversity of regressors. On the other hand, all the computed principal components are kept and the whole training set transformed through multiplying the rotation matrix is used to train each regressor in an attempt to seek high accuracy of each ensemble member.

When the algorithm described in Fig. 1 is applied to solve a regression problem, the number of subsets K or the number of attributes contained in each subset $F_{i,j}$, that is, M, should be specified in advance. Note that once one of these two parameters is given, the other is then determined. In order to simplify the following discussions, the value of M will always be firstly given in the subsequent conducted experiments.

3. Experimental studies

An experiment was set up in this section to examine the performance of Rotation Forest and compare it with that of Bagging [5], Adaboost.R2 [11], and Random Forest [8]. The base learning algorithm, namely a single regression tree, was also included for comparison. Drucker [11] adopted a pruned tree as the base learning algorithm for the ensemble methods that he compared, while the experiments conducted by Bauer and Kohavi [2] showed that pruning the tree is not beneficial to Bagging, therefore we respectively adopted

Training Phase

Given

$\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} = [X\ Y]$ where X is an $N \times n$ matrix containing the input attribute values and Y is an N-dimensional column vector containing the outputs of each training instance.

- T: number of regressors that constitute the ensemble.
- K: number of attribute subsets (or M: number of input attributes contained in each subset).
- $\mathscr{W}$: a base learning algorithm.

For $i = 1, 2, \cdots, T$

- Calculate the rotation matrix R_i^a for the ith regressor C_i
  1. Randomly split F into K subsets $F_{i,j}$ $(j = 1, \cdots, K)$.
  2. For $j = 1, 2, \cdots, K$
     (a) Select the columns of X that correspond to the attributes in $F_{i,j}$ to compose a new matrix $X_{i,j}$.
     (b) Draw a bootstrap sample $X'_{i,j}$ (with sample size smaller than that of $X_{i,j}$) from $X_{i,j}$.
     (c) Apply PCA on $X'_{i,j}$ to obtain a matrix $D_{i,j}$ whose kth column consists of the coefficients of the kth principal component.
  3. EndFor
  4. Arrange the matrices $D_{i,j}$ $(j = 1, 2, \cdots, K)$ into a block diagonal matrix R_i.
  5. Construct the rotation matrix R_i^a by rearranging the rows of R_i in order to match the order of attributes in F.
- Provide $[XR_i^a\ Y]$ as the input of $\mathscr{W}$ to build a regressor C_i .

EndFor

Predicting Phase

- For a given data point $\mathbf{x}$, let $C_i(\mathbf{x}R_i^a)$ be the value predicted by the regressor C_i, then the prediction of $\mathbf{x}$ can be calculated as

$$C^*(\mathbf{x}) = \frac{1}{T}\sum_{i=1}^{T} C_i(\mathbf{x}R_i^a).$$

Fig. 1. Pseudocode of applying Rotation Forest to improve regressors.

a non-pruned and a pruned regression tree as the base learning algorithm for the ensemble methods we considered here. The following experiments were all conducted using the *Stats* package in Matlab software with version 7.1. Table 1 shows some characteristics of the benchmark regression data sets utilized in our empirical study.

Table 1
Summary of the used data sets

Data set	# Train	# Prune	# Test	# Attribute	
				Continuous	Discrete
Friedman #1	200	40	5000	10	0
Friedman #2	200	40	5000	4	0
Friedman #3	200	40	5000	4	0
Boston Housing	401	80	25	12	1
Servo	133	16	18	0	4

The first three out of these data sets are synthetic. They originally appeared in the MARS paper [16] and are also described by Breiman [5] and Drucker [11]. The Boston Housing and Servo data are available from the UCI repository [3]. When adopting a pruned regression tree as the base learning algorithm to construct the ensembles, the examples used for training, pruning and testing were respectively taken to be 200, 40 and 5000 for the three synthetic data sets. While the division of data for Boston Housing was taken to be same as in the research work done by Drucker [11], that is, the data were randomly split into 401 instances for training, 80 instances for pruning and the remaining 25 ones for test. With respect to Servo data set, we randomly left out 10% of the examples as a test set, another 10% as a pruning set and grew the tree on the remaining 80% of the original data set. In the experiments conducted using a non-pruned regression tree as the base learning algorithm, the combination of training and pruning examples was used for training.

Because the one discrete attribute in Boston Housing data is already numeric, it was directly used in the experiments. With respect to the discrete attributes contained in Servo data, we used the method described in [27] to convert them into binary ones, that is, each categorical attribute was replaced by s binary ones encoded numerically as 0 and 1, where s is the number of possible categories of the original attribute. After doing this, there are totally 19 input attributes in Servo data set and these transformed binary attributes were utilized in the following experiments.

3.1. Effect of parameter M

Since the performance of Rotation Forest may be influenced by the parameter M included in it and the authors of [27] have also pointed out that the sensitivity of this algorithm to the parameter associated with it remains to be studied. In this subsection, we investigate this problem through running some experiments on the data sets given in Table 1.

For each data set, we took the value of M to be 1 to n with increment 1 and the ratio of the sample size of $X'_{i,j}$ to that of $X_{i,j}$, say f, to be 0.75 to conduct experiments. In fact, we have run some extra experiments to see how the performance of Rotation Forest varies with the value of f and found that the variation is little. A non-pruned and a pruned regression tree were respectively adopted as the base learning algorithm and the performance of Rotation Forest is evaluated with the root of mean squared error (RMSE) computed on the test data set. For each value of M, the number of base predictors, namely T, was set to be 50 to construct Rotation Forest, and then the test RMSE was computed. For each combination of the data set, the value of M and the base learning algorithm, the test RMSE was averaged over 100 trials through randomly generating training, pruning and testing instances for three synthetic sets and randomly split the original data set into three sets for training, pruning and testing with respect to the other two real-world data sets. Fig. 2 plots the averaged test RMSE versus the value of M used in the construction of Rotation Forest when respectively taking a non-pruned regression tree (left plots) and a pruned regression tree (right plots) as the base learning algorithm.

It can be observed in Fig. 2 that, the averaged test RMSE decreases firstly, then it gradually exhibits a minimum and eventually rises as the value of M grows except for Friedman #2 and Friedman #3 data sets on which the test RMSE decreases monotonically with M. Through comparing the left and right plots, namely, the plots obtained with adopting respectively a non-pruned regression tree and a pruned regression tree as the

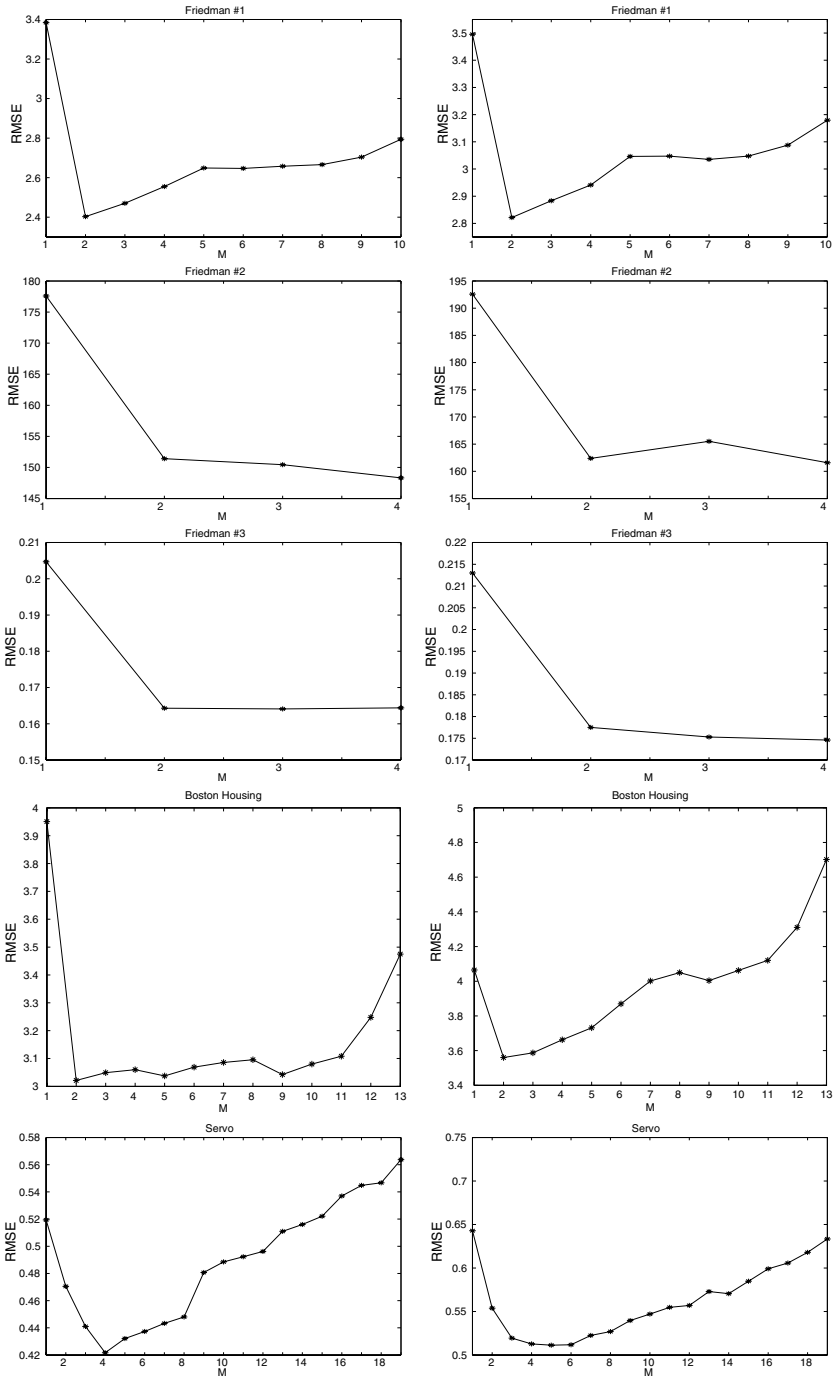

Fig. 2. The averaged test RMSE versus the value of parameter M when using non-pruned trees (left plots) and pruned trees (right plots) to construct Rotation Forest.

base learning algorithm, we can find that there is not much difference between them since the shape of the plots obtained for each data set is very similar. However, it should be recalled that the scales labeled on the vertical axes of the two plots for each data set are different. Furthermore, it seems that the performance of Rotation Forest can reach asymptotically optimal with $M = 2$ for the first four data sets and $M = 4$ for Servo data set. In the following experiments, these values of M were always used.

3.2. Comparison with other methods

In this subsection, we compare the performance of Rotation Forest with that of several other ensemble methods, that is, Bagging, Random Forest and Adaboost.R2. The results obtained with the base learning algorithm, namely a single regression tree, were also included into the comparison. Firstly, the dependence of the performance of the ensemble methods on the number of base regressors is investigated. Secondly, how well the considered methods perform on the data sets given in Table 1 is studied.

In order to study how the performance of the considered several ensemble methods changes with the number of base regressors included into the ensembles, we respectively took 50 non-pruned and pruned regression trees to construct each ensemble. As for Adaboost.R2, Drucker [11] have proposed three loss functions (that is, linear, square and exponential) that can be used and here we employed the linear loss function to conduct the following experiments since the difference between the results computed with different loss functions is not very large. In the process of building an ensemble with each method, the test RMSE was registered at every time that a tree was added into it. For each data set, this process was repeated 100 runs through randomly generating or splitting the experimental data. Fig. 3 displays the dependence of the test RMSE averaged over 100 runs on the number of non-pruned trees (left plots) and pruned trees (right plots) that were used to construct the ensembles.

From these plots, the following observations can be made:

- All ensembles are similar in their behavior, i.e., the test RMSEs of them decrease monotonically as the number of base regressors in the ensemble grows. This disease gradually levels off at an asymptotically constant.
- Rotation Forest generally performs worse than Adaboost.R2 except for Friedman #1 data set, on which they performs almost equally well when adopting a non-pruned regression tree as the base learning algorithm.
- The performance of Rotation Forest is better than that of Random Forest in all the cases besides the Friedman #3 data set, where the former is slightly worse than the latter if a non-pruned regression tree is used to build the ensembles.
- There is not a clear winner between Bagging and Rotation Forest, that is, Rotation Forest performs sometimes better and sometimes worse.
- Pruning the tree seems to have some undue effect on the performance of the ensemble methods because the scales labeled on the vertical axes of the right plots are larger than those of left plots. Furthermore, pruning the tree also slightly changes the ranking of the ensemble methods on each data set. For example, the order of the ensemble methods with respect to the averaged test RMSE on Friedman #2 data set from best to worst is Bagging, Adaboost.R2, Rotation Forest and Random Forest in the left plot, whereas the order of Bagging and Adaboost.R2 is reversed in the right plot.

It should be mentioned that in practical applications, the number of base regressors included into the ensembles should be picked to be not too large if the prediction speed and storage needs are taken into account. It can be seen in Fig. 3 that, the performance of all ensemble methods begins to level off when the value of T lies in the vicinity of 10. Therefore, we took the value of T to be 10 to conduct some more experiments to study the performance of the ensemble methods when the ensemble size (that is, the number of base regressors used to build an ensemble) is relatively small.

For each combination of the data set, the base learning algorithm and the ensemble construction method, we took 10 regression trees to construct an ensemble whose performance was then evaluated with RMSE computed on the test set. With respect to the three synthetic data sets, the trial was repeated 100 times through randomly generating data whereas for the Boston Housing and Servo data sets, the trial was run 100 times through randomly splitting the data into three sets used for training, pruning and testing. Then the average and standard deviation of these 100 test RMSEs were calculated. Tables 2 and 3 give the detailed experimental results. As for Adaboost.R2, the results computed with linear, square and exponential loss functions were all listed here for a complete comparison.

For reference, we also display the averages and the standard deviations of test RMSEs calculated with a single regression tree as well. In order to see whether Rotation Forest is significantly better or worse than other

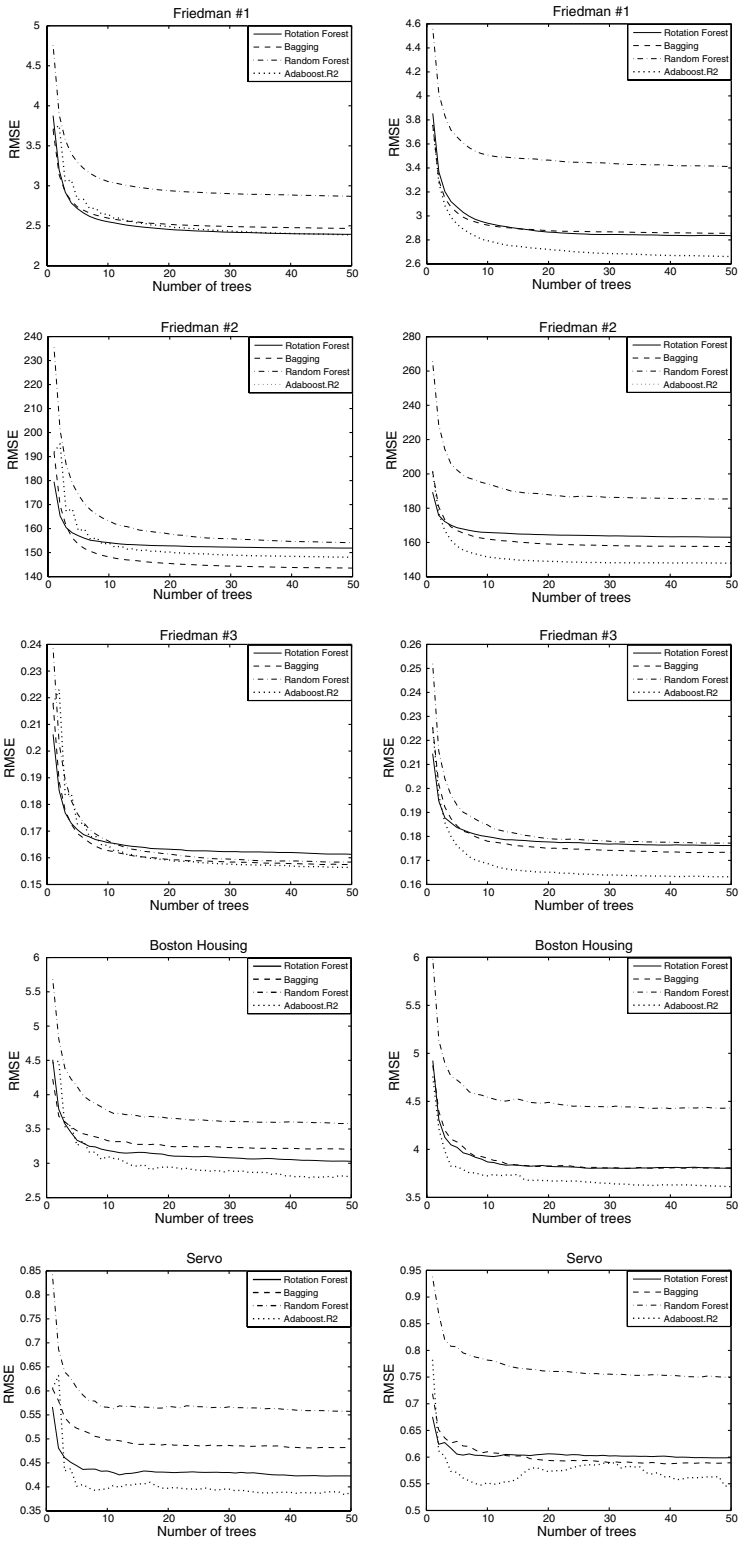

Fig. 3. The dependence of the averaged test RMSE on the number of non-pruned trees (left plots) and pruned trees (right plots) included an ensemble.

Table 2
Comparison of performance of different methods computed with non-pruned regression trees

Data set	Rotation Forest	Single tree	Bagging	Random Forest	Adaboost.R2		
					Linear	Square	Exponential
Friedman #1	2.547 ± 0.107	3.396 ± 0.122 •	2.574 ± 0.081 •	3.063 ± 0.144 •	2.647 ± 0.081 •	2.675 ± 0.083 •	2.651 ± 0.079 •
Friedman #2 ($\times 10^2$)	1.523 ± 0.046	1.786 ± 0.077 •	1.486 ± 0.036	1.628 ± 0.067 •	1.551 ± 0.046 •	1.567 ± 0.051 •	1.544 ± 0.047 •
Friedman #3	0.167 ± 0.009	0.201 ± 0.010 •	0.163 ± 0.008 ○	0.168 ± 0.008	0.164 ± 0.009 ○	0.164 ± 0.009 ○	0.164 ± 0.008 ○
Boston Housing	3.115 ± 0.851	4.205 ± 1.473 •	3.225 ± 0.931	3.443 ± 0.973 •	3.121 ± 0.960	3.095 ± 0.908	3.093 ± 0.948
Servo	0.464 ± 0.244	0.577 ± 0.372 •	0.537 ± 0.398	0.730 ± 0.276 •	0.366 ± 0.197 ○	0.398 ± 0.206 ○	0.365 ± 0.194 ○

"○": Rotation Forest is significantly worse.
"•": Rotation Forest is significantly better.
Significance level $\alpha = 0.05$.

Table 3
Comparison of performance of different methods computed with pruned regression trees

Data set	Rotation Forest	Single tree	Bagging	Random Forest	Adaboost.R2		
					Linear	Square	Exponential
Friedman #1	2.928 ± 0.131	3.481 ± 0.143 •	2.931 ± 0.122	3.563 ± 0.149 •	2.876 ± 0.091 ○	2.876 ± 0.114 ○	2.892 ± 0.109 ○
Friedman #2 ($\times 10^2$)	1.644 ± 0.062	1.935 ± 0.121 •	1.625 ± 0.082 ○	1.948 ± 0.110 •	1.570 ± 0.053 ○	1.586 ± 0.061 ○	1.578 ± 0.053 ○
Friedman #3	0.180 ± 0.009	0.213 ± 0.012 •	0.177 ± 0.010 ○	0.186 ± 0.011 •	0.174 ± 0.009 ○	0.174 ± 0.009 ○	0.174 ± 0.010 ○
Boston Housing	3.894 ± 0.978	4.311 ± 1.468 •	3.833 ± 1.058	4.982 ± 1.435 •	3.423 ± 0.732 ○	3.494 ± 0.821 ○	3.389 ± 0.723 ○
Servo	0.531 ± 0.258	0.690 ± 0.384 •	0.616 ± 0.334 •	0.817 ± 0.267 •	0.473 ± 0.226 ○	0.477 ± 0.257	0.462 ± 0.207 ○

"○": Rotation Forest is significantly worse.
"•": Rotation Forest is significantly better.
Significance level $\alpha = 0.05$.

methods, a one-tailed paired t test was performed and the results for which a significant difference with Rotation Forest was found are marked with a bullet or a open circle next to them. A bullet next to a result indicates that Rotation Forest is significantly better than the respective method for the corresponding data set. An open circle next to a result denotes that Rotation Forest performs significantly worse than the corresponding method.

As seen in Tables 2 and 3, Rotation Forest performs significantly better than a single tree and Random Forest except for Friedman #3 data set on which the difference between Rotation Forest and Random Forest is not significant when the base learning algorithm is a non-pruned tree. When adopting a pruned tree as the base learning algorithm, Adaboost.R2 is seen to significantly outperform Rotation Forest in almost all cases besides Servo data set on which Adaboost.R2 using the square loss function is only a little better than Rotation Forest. If the base learning algorithm is taken to be a non-pruned tree, Rotation Forest performs significantly better than Adaboost.R2 on Friedman #1 and Friedman #2 data sets and the case is reversed on Friedman #3 and Servo data sets, namely, the former is significantly worse than the latter. On Boston Housing data set, Adaboost.R2 with square and exponential loss functions seems to be better than Rotation Forest but the difference is not significant. With respect to the advantage of Rotation Forest over Bagging, the statistically significant differences are favorable in two cases, unfavorable in three cases and not significant in five cases.

Table 4
Scoring matrix for different methods with non-pruned tree (values are expressed in %)

Method	Single tree	Bagging	Random Forest	Adaboost.R2			Rotation Forest	Total
				Linear	Square	Exponential		
Single tree	0	−18.03	−6.45	−23.19	−21.86	−23.42	−20.43	−113.38
Bagging	18.03	0	12.09	−5.50	−4.07	−5.77	−2.65	12.13
Random Forest	6.45	−12.09	0	−15.98	−14.88	−16.23	−13.97	−66.70
Adaboost.R2 (Linear)	23.19	5.50	15.98	0	1.85	−0.29	3.43	49.67
Adaboost.R2 (Square)	21.86	4.07	14.88	−1.85	0	−2.14	1.81	38.63
Adaboost.R2 (Exponential)	23.42	5.77	16.23	0.29	2.14	0	3.71	51.57
Rotation Forest	20.43	2.65	13.97	−3.43	−1.81	−3.71	0	28.09

Table 5
Scoring matrix for different methods with pruned tree (values are expressed in %)

Method	Single tree	Bagging	Random Forest	Adaboost.R2			Rotation Forest	Total
				Linear	Square	Exponential		
Single tree	0	−14.11	3.86	−21.32	−20.71	−21.62	−15.83	−89.72
Bagging	14.11	0	17.36	−8.17	−7.48	−8.50	−1.90	5.42
Random Forest	−3.86	−17.36	0	−23.71	−23.16	−23.94	−18.70	−110.73
Adaboost.R2 (Linear)	21.32	8.17	23.71	0	0.78	−0.45	6.53	60.05
Adaboost.R2 (Square)	20.71	7.48	23.16	−0.78	0	−1.22	5.82	55.17
Adaboost.R2 (Exponential)	21.62	8.50	23.94	0.45	1.22	0	6.91	62.64
Rotation Forest	15.83	1.90	18.70	−6.53	−5.82	−6.91	0	17.18

In order to further analyze the relative performance of these prediction methods over the considered data sets, some quantitative measure is required and we calculated the so-called scoring matrices [30] in Tables 4 and 5.

The scoring matrix gives the average relative performance (expressed in %) of one procedure over another procedure for the considered data sets. The element of the scoring matrix $SM_{i,j}$ stands for the average performance of the ith method (labeled in row) over the jth method (labeled in column) and it is calculated as

$$SM_{i,j} = \frac{1}{N}\sum_{k=1}^{N}\frac{RMSE_{k,j} - RMSE_{k,i}}{\max(RMSE_{k,i}, RMSE_{k,j})},$$

where N is the number of the considered data sets.

It can be observed in Table 4 that the order of the methods ranked by scoring from highest to lowest are Adaboost.R2 (exponential), Adaboost.R2 (linear), Adaboost.R2 (square), Rotation Forest, Bagging, Random Forest and Single tree when the base learning algorithm is a non-pruned regression tree. In Table 5, the order of scores computed for each method hardly changes except that the order of Random Forest and Single tree is altered, which means that Random Forest performs even worse than a single tree when a pruned tree is adopted as the base learning algorithm.

4. Conclusions

The experiments conducted in [27] demonstrated that Rotation Forest is a very successful ensemble classifier method and it generally performs much better than other ensemble classifiers constructed by Bagging, Adaboost and Random Forest when the ensemble size is small. However, it is not known whether Rotation Forest performs well in solving regression problems. This problem is investigated in this paper by running experiments on several benchmark regression data sets through adopting a non-pruned and a pruned regression tree respectively as the base learning algorithm. Meanwhile, the performance of Rotation Forest is

compared with that of other ensemble procedures Bagging, Random Forest and Adaboost.R2, and with that of a single regression tree. The sensitivity of Rotation Forest to the choice of parameters included in it is also studied. The results show that number of attributes that each subset contains has some influence and the ensemble size T has some but not large effect except that the value of T is too small. On the considered regression data sets, Adaboost.R2 generally outperforms Rotation Forest and both of them are better than Random Forest. There is not a clear winner between Rotation Forest and Bagging, that is, the former performs sometimes better and sometimes worse. Furthermore, it seems that pruning the tree has some undue effect on the performance of all methods.

Nevertheless, neural network is also a commonly used base learning algorithm in constructing ensemble machines and it is deserved to study how Rotation Forest will perform with it. Combining the techniques of Rotation Forest with those of other ensemble methods to build more powerful ensemble machines is also a promising work.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 10531030 and 60675013).

References

[1] R. Avnimelech, N. Intrator, Boosting regression estimators, Neural Comput. 11 (2) (1999) 499–520.
[2] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Mach. Learn. 36 (1/2) (1999) 105–139.
[3] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 1998. Available from: <http://http://www.ics.uci.edu/~mlearn/MLRepository.html>.
[4] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Chapman & Hall, New York, 1984.
[5] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[6] L. Breiman, Prediction games and arcing algorithms, Neural Comput. 11 (7) (1999) 1493–1518.
[7] L. Breiman, Arcing classifiers, Ann. Stat. 26 (3) (1998) 801–849.
[8] L. Breiman, Random Forests, Mach. Learn. 45 (1) (2001) 5–32.
[9] A. Chandra, X. Yao, Evolving hybrid ensembles of learning machines for better generalisation, Neurocomputing 69 (7–9) (2006) 686–700.
[10] T. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization, Mach. Learn. 40 (2) (2000) 139–157.
[11] H. Drucker, Improving regressors using boosting techniques, in: Proceedings of the 14th International Conference on Machine Learning, Morgan, Kaufmann, 1997, pp. 107–115.
[12] N. Duffy, D. Helmbold, Boosting methods for regression, Mach. Learn. 47 (2–3) (2002) 153–200.
[13] B. Efron, R. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, New York, 1993.
[14] Y. Freund, R. Schapire, Experiment with a new boosting algorithm, in: Proceedings of the 13th International Conference on Machine Learning, Bari, Italy, 1996, pp. 148–156.
[15] Y. Freund, R. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[16] J. Friedman, Multivariate adaptive regression splines, Ann. Stat. 19 (1) (1991) 1–141.
[17] J. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. 29 (5) (2001) 1189–1232.
[18] J. Friedman, Stochastic gradient boosting, Comput. Stat. Data Anal. 38 (4) (2002) 367–378.
[19] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: A statistical view of boosting, Ann. Stat. 28 (2) (2000) 337–407.
[20] V. Gómez-Verdejo, M. Ortega-Moral, J. Arenas-García, A.R. Figueiras-Vidal, Boosting by weighting critical and erronrous samples, Neurocomputing 69 (7–9) (2006) 679–685.
[21] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern. Anal. Mach. Intell. 12 (10) (1990) 993–1001.
[22] R. Meir, G. Rätsch, An introduction to boosting and leveraging, Lect. Notes Comput. Sci. 2600 (2003) 118–183.
[23] D. Optiz, R. Maclin, Popular ensemble methods: an empirical study, J. Art. Intel. Res. 11 (1999) 169–198.
[24] D.W. Optiz, J.W. Shavlik, Generating accurate and diverse members of a neural network ensemble, Neural Inform. Proc. Syst. 8 (1996) 535–541.
[25] G. Ridgeway, D. Madigan, T. Richardson, Boosting methodology for regression problems, in: Proceedings of Artificial Intelligence and Statistics, Fort Lauderdale, Florida, 1999, pp. 152–161.
[26] G. Ridgeway, The state of boosting, Comput. Sci. Stat. 31 (1999) 172–181.

[27] J.J. Rodríguez, L.I. Kuncheva, C.J. Alonso, Rotation Forest: a new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (10) (2006) 1619–1630.
[28] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (1990) 197–227.
[29] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Ann. Stat. 26 (5) (1998) 1651–1686.
[30] D.L. Shrestha, D.P. Solomatine, Experiments with Adaboost.RT, an improved boosting scheme for regression, Neural Comput. 18 (7) (2006) 1678–1710.
[31] V. Tresp, Committee machines, in: Yu Hen Hu, Jenq-Neng Hwang (Eds.), Handbook for neural network signal processing (2001).
[32] G.I. Webb, Multiboosting: a technique for combining boosting and wagging, Mach. Learn. 40 (2) (2000) 159–196.