Applied Mathematics and Computation 219 (2013) 9914–9924

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Rotation Forests for regression

Carlos Pardo, José F. Diez-Pastor, César García-Osorio, Juan J. Rodríguez *

University of Burgos, Spain

ARTICLE INFO

Keywords:
Rotation Forest
Bagging
Random Subspaces
Boosting
Ensembles
Regression

ABSTRACT

Rotation Forest, originally proposed for the combination of classifiers, has shown itself to be very competitive, when compared with other ensemble construction methods. In this paper, the performance of Rotation Forest for combining regressors is investigated using a broad range of datasets, 61 in total, which vary in size from 13 to more than 40,000 instances, and from 2 to 60 attributes, with both numeric and nominal attributes. Rotation Forest has favourable results when compared with Bagging, Random Subspaces, Iterated Bagging and AdaBoost.R2, according to average ranks and a scoring matrix. Diversity error diagrams are used to analyse the behaviour of the ensemble methods.

1. Introduction

In the field of data mining, the task of classification is to predict the class of a given instance correctly. An instance is a set of related data values one of which (the class) we want to predict. One important research line in the field of machine learning is the development of methods for combining or grouping classifiers. These methods are known as multiclassifiers or ensembles. There is a general consensus that these methods are the best technique for dealing with the most difficult problems [1]. The underlying idea of an ensemble is to combine several classifiers, to obtain a classifier whose performance is better than each of the individual classifiers in the ensemble. Ensembles are made up of a number of classifiers trained to ensure a degree of diversity between them (they fail on different instances which is what makes the ensemble more robust).

Whereas the value to be predicted in certain classification problems (the class) belongs to a discrete set, in regression problems the value to be predicted is numeric, rather than discrete. For these types of problems, regressors rather than classifiers are combined. There is a growing interest in applying the techniques of building ensembles to the combination of regressors.

In this paper, we shall use the term "model" to refer to the base method, either a classifier or a regressor, that is combined in the ensemble. It is possible to combine models obtained with different methods (heterogeneous ensembles), but also to generate the models in the ensemble using a single method (homogeneous ensembles). This work considers the latter type of ensembles. In these ensemble methods, in order to obtain different models from the same method, the training dataset is usually transformed in some way.

One of the simplest ensemble methods is Bagging [2]. Each model in the ensemble is constructed using a random sample with replacement of the original training data. Usually, the size of the sample is the same as for the original data, but certain training objects appear several times in the sample while others are not included. For regression, the prediction given by the ensemble is the mean average of the ensemble model predictions.

In Random Subpaces [3], the transformation applied to the dataset is the deletion of certain features: the models are trained in random subspaces.

Iterated Bagging [4] is an ensemble method for regression. This ensemble is formed by several Bagging sub-ensembles. The first model is constructed as usual with Bagging. The following models are also constructed with Bagging, but the

* Corresponding author.
E-mail addresses: cpardo@ubu.es (C. Pardo), jfdpastor@ubu.es (J.F. Diez-Pastor), cgosorio@ubu.es (C. García-Osorio), jjrodriguez@ubu.es (J.J. Rodríguez).

0096-3003/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2013.03.139

outputs to be predicted are not the outputs in the original data. The Iterated Bagging ensemble prediction is the sum (not the average) of the Bagging sub-ensembles. The idea is that the subsequent Bagging sub-ensemble will compensate the errors of the current ensemble. Hence, the outputs used in its training are the differences between the prediction given by the current ensemble and the original outputs.

An additional detail of Iterated Bagging is that for estimating the predictions given by the current ensemble, only the "out of the bag" examples are used. An example is out of the bag, if it was not in the sample that was used to construct a particular model. Hence, only the models that were not trained with this example are used, in order to estimate the prediction for a given training object. In this way, optimistic predictions are avoided.

AdaBoost [5] is one of the most successful ensemble methods for classification. Some variants have been designed for regression, such as AdaBoost.R2 [6]. In this method, the training objects have a weight, initially the same for all the objects. A model is constructed taking these weights into account. The models also have a weight that is determined by their performance. After a model has been constructed, the examples are reweighted, so that greater weights are attached to those with larger errors, so that they will be more important in the next iteration for the construction of the next model. The prediction given by the ensemble is a weighted median of the prediction of the base models.

Rotation Forest [7,8] is an ensemble method, originally proposed for classification purposes. It is based on constructing each classifier with features obtained by rotating subspaces of the original dataset. One difference with other ensemble methods (e.g., Bagging, Random Subspaces) is that no information is lost from the dataset used to construct each base classifier, which contains the same information as in the original dataset. No instances are discarded in the rotated dataset, such that all the information on the features is in the rotated dataset, although is represented using new features. This method is suitable for those datasets that are affected by rotations, such as decision trees [9].

The performance of Rotation Forest for regression has previously been studied in [10]. In that work, it was found that AdaBoost.R2 generally outperformed Rotation Forest. This work presents different results, as Rotation Forest is generally the best method. The main cause for this apparent contradiction might be the number of data sets under consideration: only 5 in [10] and 61 in this study, which is all together more exhaustive in that respect. In addition to using Random Subspaces and Iterated Bagging in the comparison, further developments in this work are the selection among some configurations of the same method using an internal cross validation and the use of diversity error diagrams to analyse the behaviour of the ensemble methods.

The rest of the paper is organized as follows: the Rotation Forest method is described in Section 2, Section 3 presents the experiments and, finally, Section 4 summarises our findings.

2. Rotation Forest

Rotation Forest [7] is an ensemble method initially proposed for classification, but the adaptation for regression is immediate. Algorithm 1 shows the pseudocode of Rotation Forest adapted for regression.

Training Phase

Input: Training data set X ($N \times n$ matrix) with a set $\mathbf{F}$ of n features; output values Y ($N \times 1$); ensemble size L; number of subsets K

Output: Ensemble of regressors $D_1, D_2 \dots D_L$

for $i \leftarrow 1$ *to* L **do**

  Prepare the rotation matrix R_i^a:

  begin

    Randomly split $\mathbf{F}$ into K subsets $\mathbf{F}_{i,j}$ (for $j = 1 \dots K$)

    for $j \leftarrow 1$ *to* K **do**

      $X_{i,j} \leftarrow$ submatrix ($N \times K$) of X for the features in $\mathbf{F}_{i,j}$

      $X'_{i,j} \leftarrow$ bootstrap sample from $X_{i,j}$

      $C_{i,j} \leftarrow$ rotation matrix from $\texttt{PCA}(X'_{i,j})$

    $R_i \leftarrow$ arrangement of the $C_{i,j}$ matrices in a single rotation matrix // see Equation 1

    $P_i \leftarrow$ permutation matrix, matching the order of the features in $\mathbf{F}$ // see Equation 2

    $R_i^a \leftarrow P_i\, R_i$ // Rearrangement of R_i

  $D_i \leftarrow \texttt{BuildRegressor}(X R_i^a, Y)$

Prediction Phase

Input: Data point $\mathbf{x}$; ensemble $D_1, D_2 \dots D_L$

Output: Real value r

$r \leftarrow \frac{1}{L} \sum_{i=1}^{L} D_i(\mathbf{x} R_i^a)$

Let $\mathbf{x} = [x_1, \ldots, x_n]$ be a data point with n attributes or features, let X be a training dataset with N objects or instances, and let $Y = [y_1, \ldots, y_N]^\top$ be a vector with its outputs, where $y_i \in \mathbb{R}$. Let F be the feature set. The ensemble size, L, is a parameter of the method. Another parameter is the number of feature subsets K (alternatively, the parameter could be the size M of each subset).

Each regressor D_i in the ensemble is constructed with a transformed dataset. For each dataset, a rotation matrix R_i^a is constructed (lines 5–13 in the algorithm). Firstly, the set of features is split into K subsets (line 7).

For each feature subset $\mathbf{F}_{i,j}$ (lines 8–11), $X_{i,j}$ is the dataset X, having only the features in $\mathbf{F}_{i,j}$ (line 9). In the Rotation Forest method for classification, the instances of a proper subset of the classes are removed from $X_{i,j}$. For regression, this step is omitted because there are no classes in regression problems. Then, a bootstrap sample from $X_{i,j}$ is taken, obtaining the dataset $X\prime_{i,j}$.

Principal Component Analysis (PCA) is applied to $X\prime_{i,j}$ to obtain a rotation matrix $C_{i,j}$. Each principal component is in a column of this matrix. The size of this matrix is $M \times M_{i,j}$. M is the size of $\mathbf{F}_{i,j}$ and $M_{i,j} \leqslant M$. $M_{i,j}$ will usually be M, because from M features we can calculate up to M components (or fewer, if some of the eigenvalues were zero).

The $C_{i,j}$ matrices are organised in a single "block-diagonal" rotation matrix R_i

$$R_i = \begin{bmatrix} C_{i,1} & [\mathbf{0}] & \ldots & [\mathbf{0}] \\ [\mathbf{0}] & C_{i,2} & \ldots & [\mathbf{0}] \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{0}] & [\mathbf{0}] & \ldots & C_{i,K} \end{bmatrix}, \tag{1}$$

The matrix R_i cannot be used for direct rotation of the dataset X because the features are not in the expected order. For K groups of M features this order is:

$$\mathbf{F}_{i,1,1}, \mathbf{F}_{i,1,2} \ldots \mathbf{F}_{i,1,M}, \quad \mathbf{F}_{i,2,1}, \ldots \mathbf{F}_{i,2,M}, \ \ldots \ \mathbf{F}_{i,K,1}, \ldots \mathbf{F}_{i,K,M}.$$

Let P_i be a $n \times n$ permutation matrix: there is one 1 in each row and one 1 in each column, and 0s elsewhere. In order to obtain the desired order, for each $\mathbf{F}_{i,j,k}$ ($j = 1 \ldots K, k = 1 \ldots M, \mathbf{F}_{i,j,k} \in \{1 \ldots n\}$):

$$P_i[\mathbf{F}_{i,j,k}, (j-1)K + k + 1] = 1. \tag{2}$$

Then, XP_i is the dataset with the features reordered. Let $R_i^a = P_i R_i$ be the rearranged rotation matrix. The regressor D_i is trained with XR_i^a and Y.

In the prediction phase a prediction is sought for an object $\mathbf{x}$, the object is rotated with the corresponding R_i^a matrix, for each regressor D_i in the ensemble, and the prediction given by that regressor will be $D_i(\mathbf{x}R_i^a)$. The prediction of the ensemble is the average of the predictions of the regressors in the ensemble.[1]

3. Experiments

3.1. Data sets

Table 1 shows the characteristics of the 61 data sets used in the experiments. They are available in the format used by Weka,[2] of which 30 were collected by Luís Torgo.[3]

3.2. Settings

Weka [11] was used for the experiments. It provides all the considered methods, with the exception of Iterated Bagging and AdaBoost.R2 which were implemented for Weka.

The number of models in the ensembles was set to 100 and the methods were evaluated according to the *Root of Mean Squared Error* (RMSE).

The results were obtained using 5×2-fold *cross validation* [12]. The data were randomly split into two halves: one for training and the other for testing. Then the roles were exchanged. This process was repeated 5 times. The reported results represent the average of these 10 values.

3.3. Methods and options

Either pruned *(P)* or unpruned *(U)* Regression Trees were used as the base models and the pruning method was *Reduced Error Pruning* (REP) [13].

[1] For classification, the probability assigned to a class was the average of the probabilities assigned by all the classifiers in the ensemble.

[2] http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html.

[3] http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html

Table 1
Datasets used in the experiments.

Dataset	Examples	Numeric	Nominal
2d-planes	40768	10	0
abalone	4177	7	1
ailerons	13750	40	0
auto-horse	205	17	8
auto-mpg	398	4	3
auto-price	159	15	0
auto93	93	16	6
bank-32nh	8192	32	0
bank-8FM	8192	8	0
baskball	96	4	0
bodyfat	252	14	0
bolts	40	7	0
breast-tumor	286	1	8
cal-housing	20640	8	0
cholesterol	303	6	7
cleveland	303	6	7
cloud	108	4	2
cpu	209	6	1
cpu-act	8192	21	0
cpu-small	8192	12	0
delta-ailerons	7129	5	0
delta-elevators	9517	6	0
detroit	13	13	0
diabetes-numeric	43	2	0
echo-months	130	6	3
elevators	16599	18	0
elusage	55	1	1
fishcatch	158	5	2
friedman	40768	10	0
fruitfly	125	2	2
gascons	27	4	0
house-16H	22784	16	0
house-8L	22784	8	0
housing	506	12	1
hungarian	294	6	7
kin8nm	8192	8	0
longley	16	6	0
lowbwt	189	2	7
machine-cpu	209	6	0
mbagrade	61	1	1
meta	528	19	2
mv	40768	7	3
pbc	418	10	8
pharynx	195	1	10
pole	15000	48	0
pollution	60	15	0
puma32H	8192	32	0
puma8NH	8192	8	0
pw-linear	200	10	0
pyrimidines	74	27	0
quake	2178	3	0
schlvote	38	4	1
sensory	576	0	11
servo	167	0	4
sleep	62	7	0
stock	950	9	0
strike	625	5	1
triazines	186	60	0
veteran	137	3	4
vineyard	52	3	0
wisconsin	194	32	0

Six families of ensemble methods were considered. For some families, several configurations were used. The families and their parameter values were:

1. Rotation Forest. The feature subset size was 3, the default value in Weka.
2. Bagging. Two sample sizes were considered: 100% and 75% of the original training size.
3. Random Subspaces. Two subspace sizes were considered: 50% and 75% of the original space size.

4. Iterated Bagging. Two configurations were considered: 10×10 and 5×20. For the former, Bagging was iterated 10 times, in each Bagging iteration 10 trees were constructed. For the latter, Bagging was iterated 5 times, in each Bagging iteration 20 trees were constructed.
5. AdaBoost.R2-W. This version used reweighting [5]. The method used for constructing regression trees takes into account instance weights. There are three proposed loss functions for AdaBoost.R2: linear, square and exponential.
6. AdaBoost.R2-S. This version applies resampling [5]. The regression trees were constructed without taking into account instance weights. Nevertheless, the tree was constructed using a sample of the training data and this sample is obtained using the instance weights as probabilities.

Moreover, trees can be constructed with or without pruning. All the previous configurations were combined with these two options. Hence, there are 2 configurations for Rotation Forest, 4 configurations for Bagging, Random Subspaces and Iterated Bagging and 6 configurations for AdaBoost.R2-W and AdaBoost.R2-S.

The best configuration was selected for each family and data set using an internal 5-fold cross validation. For each configuration, the method was trained with four folds and tested with the remaining fifth fold and the process was repeated until all the folds had been used once for testing; the average of the 5 accuracy measurements were then used to select the best configuration, in order to build the model that would provide the predictions for the external cross validation. This selection was done for each of the 5×2 experiments carried out for estimating the performance of a method.

3.4. Results

3.4.1. Families results

In this section, the results for the different families are presented. Table 2 shows the results obtained with the different families for the data sets under consideration. Rotation Forest has the best results for 28 of the 61 data sets. The method that had the second-best results was AdaBoost-S, with 16 data sets.

For each data set, Rotation Forest was compared with the other methods using the *corrected resampled t-test statistic* [14] (significance level: 0.05). The significant differences are marked with the symbols $\oplus$ and $\ominus$.

Table 3 summarizes the results. It shows the average ranks [15] for the six families. For each data set, the methods are sorted according to their performance. The best method has rank 1, the second has rank 2, ...and the last one has rank 6. If several methods have the same result they are assigned an average value (e.g., a rank 1.5 if two methods have the best result). Average ranks are obtained, for each family, as the mean value from all the data sets. Rotation Forest is the family with the best average rank.

Table 3 also shows the number of data sets for which Rotation Forest has better and worse results than the other methods, and the number of times these are significant differences. From among all the methods under consideration, Rotation Forest has a favourable balance of wins and losses.

According to a *Sign Test* [15] (significance level: 0.05), for 61 data sets there is a significant difference if one method is better than the other for 39 data sets or more. With the sole exception of Bagging, Rotation Forest is better than the other alternatives in at least 39 data sets.

Average ranks and the number of wins and losses do not take the magnitude into account of the differences between the methods. Given the RMSE of two methods, i and j for one data set k, the *Quantitative Score* [16,10] is defined as

$$QS_{i,j,k} = \frac{RMSE_{k,i} - RMSE_{k,j}}{\max(RMSE_{k,i}, RMSE_{k,j})}.$$

Fig. 1 shows the Quantitative Scores (expressed in %) for the comparisons between Rotation Forest and the other families. In these graphs there are 61 bars (one for each data set). The data sets are sorted according to the Quantitative Score. The number of bars with a positive sign indicates the number of data sets for which Rotation Forest is better than the other method. The bar lengths indicate the percentage improvement (or degradation).

From the Quantitative Scores a *Scoring Matrix* SM [16,10] is obtained by averaging the scores across all (N) the data sets.

$$SM_{i,j} = \sum_{k=1}^{N} \frac{1}{N} QS_{i,j,k}.$$

Table 4 shows this Scoring Matrix. Rotation Forest has a favourable score when compared with any other method. In the table, the last column shows the total value of each row, it is a performance measure for each method. Rotation Forest clearly shows the best result.

Although Tables 3 and 4 agree in the best method, Rotation Forest, they differ in the order of the other methods. They give different results because the scoring matrix takes into account the magnitude of the differences in the errors, while the average ranks only take into account the order of the methods.

3.4.2. Individual results

The previous results compared six families of methods. It is also possible to study the performance of the different methods without grouping them into families, thereby making it possible to ascertain which options give the best results.

Table 2
Results for the different methods and data sets. The best result for each data set is marked in bold.

Dataset	Rotation forest	Bagging		Random subspaces		Iterated bagging		AdaBoost R2-W		AdaBoost R2-S	
2dplanes	**1.004859**	1.013039	⊕	1.486609	⊕	1.026160	⊕	1.019278	⊕	1.036699	⊕
abalone	**2.118008**	2.181329	⊕	2.212120	⊕	2.192436	⊕	2.257864	⊕	2.224059	⊕
ailerons	**0.000163**	0.000166	⊕	0.000181	⊕	0.000166		0.000167	⊕	0.000166	⊕
auto-horse	**14.24495**	21.48258		17.21359		19.61038		19.03034		18.84850	
auto-mpg	**2.833731**	3.384313	⊕	3.169765	⊕	3.468779	⊕	3.302314		3.270033	
auto-price	**2544.270**	2811.593		2657.883		2805.742		2654.938		2715.811	
auto93	**5.996746**	9.184314	⊕	6.832310		9.118953		7.999304		8.289641	
bank-32nh	**0.085394**	0.086097		0.090950	⊕	0.086874		0.087752	⊕	0.086358	
bank-8FM	0.032778	0.032592		0.049778	⊕	**0.031695**		0.032575		0.032268	
baskball	**0.094110**	0.094570		0.099885		0.095404		0.096403		0.098632	
bodyfat	2.147761	1.605292		1.951744		**1.585451**		1.619724		1.591565	
bolts	**13.64687**	13.68913		14.48961		13.65187		14.25322		14.44445	
breast-tumor	10.18011	10.28000		**10.14816**		10.35958		10.34739		10.91544	
cal-housing	53113.98	51302.62	⊖	52227.55		50681.83	⊖	50064.89	⊖	**49666.49**	⊖
cholesterol	51.36594	**50.81945**		51.09384		51.42633		51.84079		52.26418	
cleveland	**0.896754**	0.945510		0.925044		0.954390		0.957223		0.935334	
cloud	0.601441	0.596108		0.583049		**0.555846**		0.601196		0.566442	
cpu	67.97903	70.99767		76.63036		67.12399		67.22130		**63.19950**	
cpu-act	**2.492983**	2.673092		2.695393		2.523115		2.501045		2.553252	
cpu-small	2.929343	3.084035		3.061580		2.973163		**2.928926**		2.969627	
delta-ailerons	0.000164	**0.000163**		0.000167	⊕	0.000165		0.000172	⊕	0.000168	
delta-elevators	**0.001428**	0.001437		0.001449	⊕	0.001443	⊕	0.001475	⊕	0.001486	⊕
detroit	70.87960	57.64301		69.86852		**56.41864**		66.53573		71.14636	
diabetes-numeric	0.687361	0.642223		0.711550		0.664124		0.697451		**0.640930**	
echo-months	11.80740	11.24551		11.36964		11.36834		**11.17899**		12.38867	
elevators	0.002648	0.002932	⊕	0.003226	⊕	**0.002604**		0.002758		0.002679	
elusage	**13.24999**	17.05742	⊕	13.83023		17.11714		16.67509		17.88310	
fishcatch	94.3579	122.4886		138.9906	⊕	134.9826		92.9600		**88.0561**	
friedman	1.436373	1.361611	⊖	1.750733	⊕	1.230336	⊖	**1.223865**	⊖	1.274957	⊖
fruitfly	**16.11356**	16.92420		16.72875		16.69582		16.68082		19.52100	
gascons	16.56246	**13.36005**		16.31507		14.79114		15.23908		13.93028	
house-16H	34229.92	33043.98		33924.82		33382.21		33372.85		**33002.01**	
house-8L	30579.55	**30245.43**		31476.34	⊕	30832.55		30578.16		30426.31	
housing	3.757041	3.934902		4.160726		3.970550		3.702022		**3.663753**	
hungarian	**0.363813**	0.372314		0.367038		0.377535		0.397682		0.382483	
kin8nm	**0.128836**	0.152289	⊕	0.173363	⊕	0.134293		0.141330	⊕	0.136040	⊕
longley	1724.547	1562.199		1715.160		1582.158		1879.681		**1552.271**	
lowbwt	454.6150	**451.1449**		460.2108		453.8834		455.2991		463.3115	
machine-cpu	81.93499	82.55282		85.97422		96.43588		80.76914		**75.22683**	
mbagrade	**0.320895**	0.333475		0.324957		0.330072		0.330939		0.340325	
meta	**697.6843**	719.3639		751.6000		726.2079		944.4525		803.5071	
mv	0.229403	0.183499	⊖	1.533746	⊕	0.135598	⊖	0.135096	⊖	**0.126266**	⊖
pbc	**880.6622**	923.2950		925.9117		938.2374	⊕	953.8157	⊕	951.5385	⊕
pharynx	**314.4732**	426.2455	⊕	370.6978	⊕	430.3829	⊕	449.7603	⊕	433.7802	⊕
pole	5.239366	5.454475		7.210854	⊕	5.318227		4.433465	⊖	**4.347380**	⊖
pollution	**47.24494**	50.43081		54.47696		52.37454		49.62775		51.06973	
puma32H	0.012832	0.007967	⊖	0.014567		0.008041	⊖	0.007974	⊖	**0.007936**	⊖
puma8NH	3.282299	**3.215566**	⊖	3.602815	⊕	3.234148		3.366146		3.294971	
pw-linear	1.869700	**1.860341**		2.049747		1.880415		1.911546		1.877279	
pyrimidines	0.122045	0.114359		0.120967		0.114620		0.114491		**0.104622**	
quake	0.188969	**0.188308**		0.188875		0.189023		0.202141		0.197127	
schlvote	**1130664**	1165550		1155455		1388034		1261144		1448975	
sensory	**0.720179**	0.735805		0.731524		0.740966		0.742897	⊕	0.749728	
servo	0.770093	0.799162		0.919120		0.837626		0.825762		**0.749353**	
sleep	**3.493071**	3.571728		3.685573		3.715490		3.804602		3.669260	
stock	**0.850263**	1.061872		1.005375	⊕	1.089851	⊕	0.931931		0.858515	
strike	516.7889	**495.2925**		499.8378		496.7877		527.9213		610.8574	
triazines	0.138358	**0.136369**		0.139854		0.136497		0.143053		0.137396	
veteran	147.1761	146.1992		148.0241		**146.1895**		154.8604		160.9588	
vineyard	3.130831	3.122334		3.187350		3.110877		3.003016		**2.965895**	
wisconsin	33.82990	**33.75127**		34.15153		33.94249		34.15160		34.57672	

⊕, ⊖ statistically significant improvement or degradation.

Table 5 shows these average ranks. Pruned and unpruned trees are denoted with *(P)* and *(U)*, respectively. The loss functions of AdaBoost are denoted with the suffixes -L (linear), -Q (quadratic) and -E (exponential). Moreover, to make the comparison more complete some other baseline methods are also included: single trees, ensembles of trees obtained by randomization (only for pruned trees: a subset of the training data is randomly selected for the pruning phase; the trees predictions are combined by averaging), nearest neighbours (using one neighbour or selecting the optimal number of

Table 3
Average ranks; wins and losses.

Method	Rank	All		Significant	
Rotation Forest	2.6721	Wins	Losses	Wins	Losses
Bagging	3.0492	35	26	9	5
Iterated Bagging	3.4754	39	22	7	4
AdaBoost.R2-S	3.5410	40	21	6	5
AdaBoost.R2-W	3.8689	41	20	10	5
Random Subspaces	4.3934	48	13	17	0

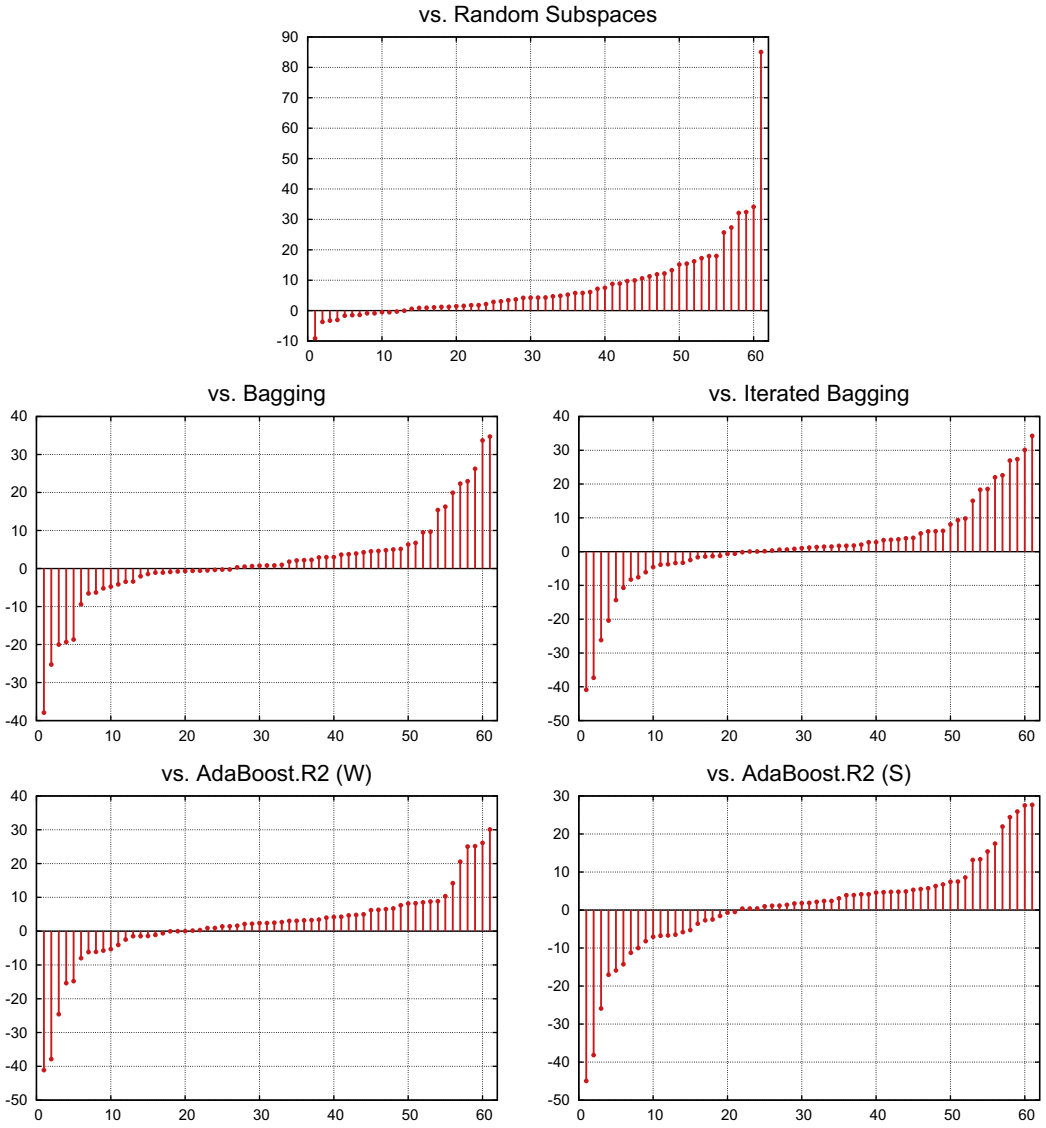

Fig. 1. Quantitative Scores.

neighbours with full cross validation) and linear regression (using all the features or only a selection of them). The best average rank is for Rotation Forest using trees without pruning.

3.4.3. Error evolution

The previous results were obtained using ensembles of 100 trees. The error depends on the ensemble size and different methods could be better for different ensemble sizes.

Table 4
Scoring matrix.

	Rotation forest	AdaBoost R2-S	Iterated bagging	Bagging	AdaBoost R2-W	Random subspaces	Total
Rotation Forest	0.0000	1.0771	1.7373	1.8133	1.8600	8.1671	14.6548
AdaBoost.R2-S	−1.0771	0.0000	0.7088	0.8853	0.8333	6.1555	7.5059
Iterated Bagging	−1.7373	−0.7088	0.0000	0.1301	0.0587	5.6440	3.3867
Bagging	−1.8133	−0.8853	−0.1301	0.0000	−0.1183	5.9714	3.0245
AdaBoost.R2-W	−1.8600	−0.8333	−0.0587	0.1183	0.0000	5.4909	2.8572
Random Subspaces	−8.1671	−6.1555	−5.6440	−5.9714	−5.4909	0.0000	−31.4291

Table 5
Average ranks.

Method	Rank
Rotation Forest *(U)*	8.64
Iterated Bagging 5x20 *(P)*	11.21
AdaBoostR2-E-S *(P)*	11.58
Bagging 75% *(U)*	11.68
Bagging 100% *(U)*	12.66
AdaBoostR2-C-S *(P)*	13.02
AdaBoostR2-L-S *(P)*	13.49
Iterated Bagging 5x20 *(U)*	14.09
Bagging 100% *(P)*	14.19
Iterated Bagging 10x10 *(P)*	14.40
AdaBoostR2-C-S *(U)*	14.87
Rotation Forest *(P)*	14.90
AdaBoostR2-L-S *(U)*	15.55
AdaBoostR2-C-W *(U)*	15.89
AdaBoostR2-E-S *(U)*	16.13
Bagging 75% *(P)*	16.22
AdaBoostR2-E-W *(P)*	16.76
Randomization *(P)*	17.39
AdaBoostR2-L-W *(P)*	17.68
Iterated Bagging 10x10 *(U)*	17.77
AdaBoostR2-C-W *(P)*	18.06
Random Subspaces 50% *(U)*	18.23
Random Subspaces 75% *(P)*	18.30
Linear Regression (all)	18.78
K-NN	18.80
Linear Regression (selection)	18.99
AdaBoostR2-L-W *(U)*	19.31
Random Subspaces 75% *(U)*	19.88
AdaBoostR2-E-W *(U)*	19.98
Random Subspaces 50% *(P)*	20.84
Tree *(P)*	25.94
1-NN	27.52
Tree *(U)*	28.23

Fig. 2 shows the average error across all the data sets for the four methods with top ranks in Table 5. The output variables for the different data sets have different magnitudes, so for obtaining these results the output variables were previously re-scaled to the interval [0,1]. With these transformed data sets 5×2-fold cross validation was used to estimate the errors. Rotation Forest had the smallest average error.

From among the four methods, Fig. 3 shows the percentage of data sets that had the best result for the ensemble size under consideration. Rotation Forest has the greater percentage, except for very small ensembles of less than 5 classifiers.

3.5. Diversity-error diagrams

Successful ensembles are formed by models with low errors, but that are diverse. These two objectives conflict with each other, because if the errors of two models are small, they cannot be very different. Several diversity measures has been proposed, in order to analyse the behaviour of ensemble methods [17].

One of the techniques used is diversity-error diagrams [18]. These are scatter plots in which there is a point for each pair of models. The horizontal axis represents the diversity between the two models; for classification, the κ (kappa) statistic is usually used. The vertical axis represents the average error of the two models.

In regression, several error measures can be considered, in this work RMSE was used:

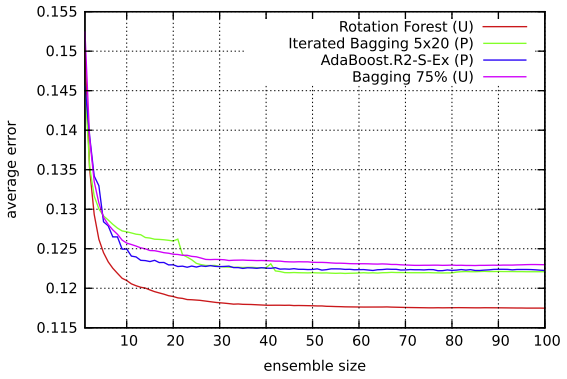

Fig. 2. Average error.

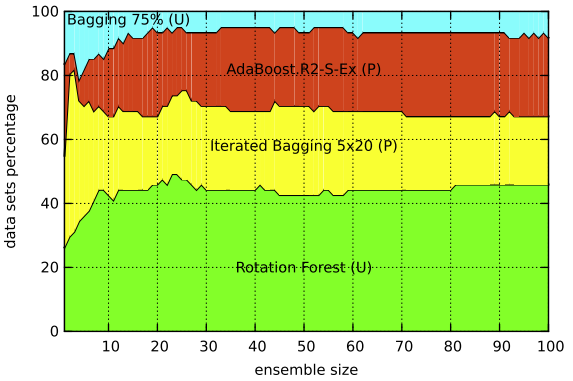

Fig. 3. Percentage diagram.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(a_i - p_i)^2}{n}},$$

where a_i represents the actual values and p_i are the predicted values.

The negative RMSE of one of the models with respect to the other was used for the measurement of diversity:

$$\text{negative}RMSE = -\sqrt{\sum_{i=1}^{n} \frac{(q_i - p_i)^2}{n}},$$

where p_i and q_i are the predictions of the two models. The reason for using the negative RMSE is because in these diagrams, when κ is used, smaller values indicate greater diversity.

The drawback of RMSE error and diversity measures is that their values are not bounded. We used the transformed data sets for these diagrams, in order to work around this problem: the outputs were previously rescaled to [0,1]. The data for these diagrams were also obtained using 5×2-fold cross validation.

Fig. 4 shows the diversity error diagrams for one data set and three of the best four methods according to Table 5. The other method, Iterated Bagging, was not included, because it is an ensemble of ensembles and with these diagrams it cannot be directly compared to the other methods. Only the members of the first sub-ensemble predict the original output variable, the predictions of the remaining members of the sub-ensembles compensate the errors of the previous sub-ensembles. Hence, it is not sensible to calculate the average error of diversity among the members of different sub-ensembles.

Diversity-error relative movement diagrams [19] are used to summarize these diagrams for all the data sets under consideration. Comparison of the diagrams generated by two methods is done by calculating the center of each diagram and drawing an arrow between each one. In relative movement diagrams, the arrows are shifted to start at the origin of the coordinates so the coordinates of the tips represent the differences between the errors and diversities of the two methods that were compared.

Fig. 5 shows the diversity-error relative movement diagrams. The numbers in the corners indicate how many arrows point in that direction. For instance, there are 24 arrows that point upwards to the left for the arrows from Bagging to Rota-

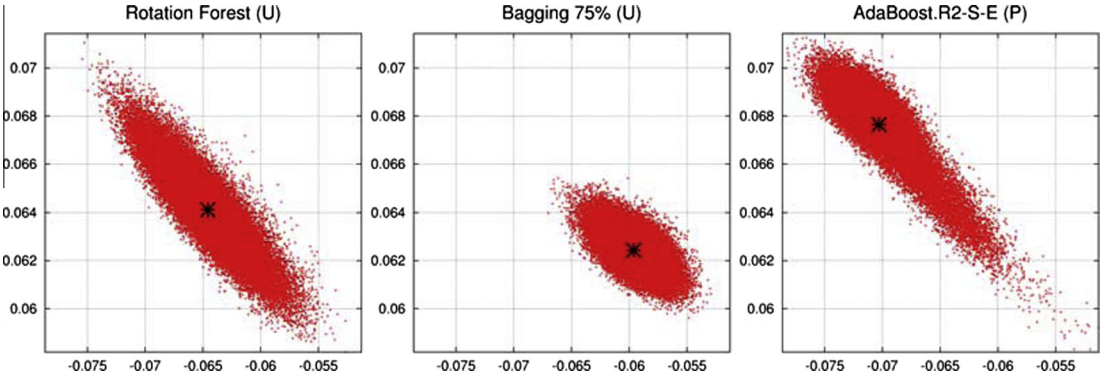

Fig. 4. Diversity error diagrams for the dataset ailerons. The symbol "∗" shows the average point.

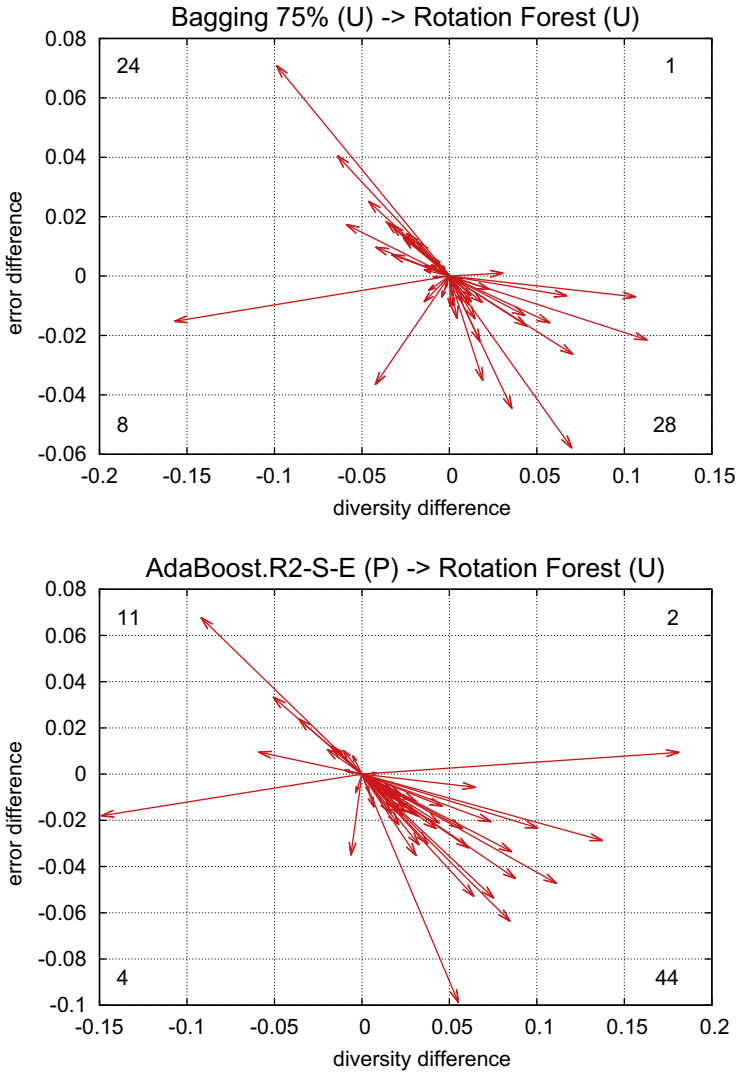

Fig. 5. Diversity-error relative movement diagrams.

tion Forest. If an arrow points upwards, it means that the members have more error in the second method. If the arrow points to the left it means that the members are more diverse for the second method.

No clear pattern emerges from the comparison between Bagging with Rotation Forest: 25 arrows point upwards, 36 downwards; 32 arrows point to the right and 29 to the left. Besides, improvements on one axis and worsening on the other (24 + 28) are much more commonly found than improvements of worse positions on both axes (8 + 1).

In the comparison of AdaBoost with Rotation Forest the situation is much clearer. The majority of the arrows point downwards to the right. This means that the models obtained with Rotation Forest are less diverse, but more accurate than the models obtained with AdaBoost.R2. For classification [7], Rotation Forest also had less diverse but more accurate classifiers than AdaBoost.

4. Conclusions

The performance of Rotation Forest for regression problems has been studied using 61 datasets. The ensembles obtained combine regression trees. Rotation Forest obtained favourable results when compared with Bagging, Iterated Bagging, Random Subspaces and AdaBoost.R2.

Diversity-error diagrams have been used to compare the behaviour of Rotation Forest with Bagging and AdaBoost.R2. There is no clear pattern when compared with Bagging. When compared with AdaBoost.R2, Rotation Forest usually has less diverse but more accurate members. In this case, the error improvement generally compensates the reduced diversity.

The Rotation Forest method has a parameter that is the feature number or group size. In the experiments, this value was set to 3 features per group and even better results could be obtained adjusting this parameter for each data set.

Acknowledgements

This work was supported by the Projects TIN2011-24046 and IPT-2011-1265-020000 of the Spanish Ministry of Economy and Competitiveness.

References

[1] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.
[2] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[3] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[4] L. Breiman, Using iterated bagging to debias regressions, in: 277, Machine Learning 45 (3) (2001) 261. <http://dx.doi.org/10.1023/A:1017934522171>.
[5] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.
[6] H. Drucker, Improving regressors using boosting techniques, in: ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 107–115. <http://portal.acm.org/citation.cfm?id=645526.657132>.
[7] J.J. Rodríguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: a new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10) (2006) 1619–1630. <http://doi.ieeecomputersociety.org/10.1109/TPAMI.2006.211>.
[8] L.I. Kuncheva, J.J. Rodríguez, An experimental study on rotation forest ensembles, in: 7th International Workshop on Multiple Classifier Systems, MCS 2007, LNCS, vol. 4472, Springer, 2007, pp. 459–468. <http://dx.doi.org/10.1007/978-3-540-72523-746>.
[9] J.R. Quinlan, C4.5: Programs for Machine Learning, Machine Learning, Morgan Kaufmann, San Mateo, California, 1993.
[10] C. Zhang, J. Zhang, G. Wang, An empirical study of using rotation forest to improve regressors, Applied Mathematics and Computation 195 (2) (2008) 618–629. <http://dx.doi.org/10.1016/j.amc.2007.05.010>.
[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD explorations 11 (1).
[12] T.G. Dietterich, Approximate statistical test for comparing supervised classification learning algorithms, Neural Computation 10 (7) (1998) 1895–1923.
[13] T. Elomaa, M. Kääriäinen, An analysis of reduced error pruning, Journal of Artificial Intelligence Research 15 (2001) 163–187.
[14] C. Nadeau, Y. Bengio, Inference for the generalization error, Machine Learning 52, pp. 239–281.
[15] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[16] D.L. Shrestha, D.P. Solomatine, Experiments with AdaBoost.RT, an improved boosting scheme for regression, Neural Computation 18 (7) (2006) 1678–1710. <http://dx.doi.org/10.1162/neco.2006.18.7.1678>.
[17] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles, Machine Learning 51 (2003) 181–207.
[18] D.D. Margineantu, T.G. Dietterich, Pruning adaptive boosting, in: Proceedings 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 211–218.
[19] J. Maudes, J.J. Rodríguez, C. García-Osorio, N. García-Pedrajas, Random feature weights for decision tree ensemble construction, Information Fusion 13 (1) (2012) 20–30. <http://dx.doi.org/10.1016/j.inffus.2010.11.004>.