Bagging Ensemble Selection for Regression

Quan Sun and Bernhard Pfahringer

Department of Computer Science
The University of Waikato
Hamilton, New Zealand
{qs12,bernhard}@cs.waikato.ac.nz

Abstract. Bagging ensemble selection (BES) is a relatively new ensemble learning strategy. The strategy can be seen as an ensemble of the ensemble selection from libraries of models (ES) strategy. Previous experimental results on binary classification problems have shown that using random trees as base classifiers, BES-OOB (the most successful variant of BES) is competitive with (and in many cases, superior to) other ensemble learning strategies, for instance, the original ES algorithm, stacking with linear regression, random forests or boosting. Motivated by the promising results in classification, this paper examines the predictive performance of the BES-OOB strategy for regression problems. Our results show that the BES-OOB strategy outperforms Stochastic Gradient Boosting and Bagging when using regression trees as the base learners. Our results also suggest that the advantage of using a diverse model library becomes clear when the model library size is relatively large. We also present encouraging results indicating that the non-negative least squares algorithm is a viable approach for pruning an ensemble of ensembles.

1 Introduction

The problem of constructing an ensemble from a library of base learners has always been of interest to the data mining community. Usually, compared with individual learners, ensemble strategies are more accurate and stable. In a typical regression setting, a given training set D consists of m instances, such as $D = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_m}, y_m)\}$, where $\mathbf{x_i}$ is an instance and y_i is a target, the task is to learn an approximate function $f : X \to \mathbb{R}$ of the true function f^0 from D. Let $f_j, j = 1...k$, be a set of base regression learners that output predictions $f_j(\mathbf{x_i})$. The output of a simple regression ensemble $F(\mathbf{x_i})$ for instance $\mathbf{x_i}$ can be expressed as:

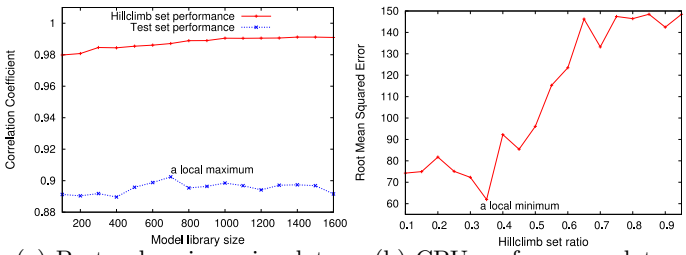
$$F(\mathbf{x_i}) = \sum_{j=1}^{k} w_j f_j(\mathbf{x_i}), \tag{1}$$

where w_j is the weight of base learner f_j. In this particular form, ensemble learning strategies can be seen as methods for calculating optimal weights for each base learner in terms of a regression goal. Since the mid-70s, many ensemble strategies have been proposed. We first review a few state-of-the-art ensemble

strategies for regression. Gradient Boosting [9] is a classical ensemble learning algorithm. It produces an ensemble of base learners (e.g., decision trees) based on a stage-wise procedure to optimise an arbitrary differentiable loss function. Stochastic Gradient Boosting [8] is an extension of the Gradient Boosting algorithm, where at each iteration, a base learner trains on a subset of the training set drawn at random without replacement. Bagging (bootstrap aggregating) [2] is based on the instability of base learners, which can be exploited to improve the predictive performance of such unstable base learners. The basic idea is that, given a training set T of size n and a learner A, bagging generates m new training sets with replacement, T_i. Then, bagging applies A to each T_i to build m models. The final output of bagging is based on simple averaging [2]. For instance, in a regression setting using Eq. 1, the weight w_j for f_j is $\frac{1}{k}$. MultiBoosting [16] is an ensemble algorithm designed to reduce both variance and bias simultaneously, in which Boosting is used as the base learner for Bagging. For a more detailed review of recent developments on ensemble learning strategies please refer to [14, 19]. Next, we discuss the motivations for proposing and studying the bagging ensemble selection (BES) strategy.

Before introducing the BES strategy, we briefly review the ensemble selection (ES) algorithm proposed in [6]. ES is a method for constructing ensembles from a library of base learners. Firstly, base models are built using many different machine-learning algorithms. Then a construction strategy such as forward step-wise selection, guided by some scoring function, extracts a well performing subset of all models. The simple forward step-wise model selection based procedure proposed in [6] works as follows: (1) start with an empty ensemble; (2) add to the ensemble the model in the library that maximizes the ensemble's performance using some given error metric on a hillclimb set; (3) repeat Step 2 until all models have been examined; (4) return that subset of models that yields maximum performance on the hillclimb set. One advantage of ES is that it can be optimised for many common performance metrics or even a combination of metrics. To exemplify this ability, a number of different metrics including mean absolute error, correlation coefficients, root mean squared error, and relative root squared error, will be used to present results in the graphs and tables of this paper. For variants of the ES algorithm, the reader is referred to [5, 6].

Experimental results in [6, 15] show that in the classification setting, the simple ES strategy sometimes overfits the hillclimb set, reducing its predictive performance. Our preliminary experimental results for employing ES on regression problems also identified a similar phenomenon. Figure 1 (a) shows an example of the hillclimb set overfitting problem of ES on the *Boston housing price* data. The red curve (top) is the hillclimb set performance; the blue curve (bottom) is the test set performance. We can see that as the size of the model library increases, the hillclimb set performance also improves gradually. However, the test set performance does not always improve. In this case, the local optimal performance is achieved when the model library size is about 700. Another practical issue is that users have to estimate the optimal hillclimb set ratio for a given data set. Figure 1 (b) shows an example on the *CPU* data based on cross-validation



(a) Boston housing-price data (b) CPU performance data

Fig. 1. Examples of the hillclimb set overfitting and the hillclimb set ratio problems of the Ensemble Selection strategy

based performance estimation. We can see that the local optimal performance is achieved when the hillclimb ratio is about 0.35; after that, the performance starts to drop. Although we could use cross-validation to estimate the hillclimb set ratio, this would substantially increase the practical training cost of ES. To overcome these problems and improve the predictive performance of the ES strategy, three BES strategies have been proposed in [15]. Experimental results show that, under the classification setting, the BES-OOB strategy is the most successful variant in terms of predictive performance. In this paper, we focus on examining the predictive performance of the BES-OOB strategy for regression problems.

2 The BES-OOB Strategy

The **BES-OOB** strategy uses the full bootstrap sample as the build set for model generation, and the respective out-of-bag sample as the hillclimb set for ensemble construction. The bootstrap sample is expected to contain about $1-1/e \approx 63.2\%$ of the unique examples of the training set [1, 2]. Therefore the hillclimb set (out-of-bag sample) is expected to have about $1/e \approx 36.8\%$ unique examples of the training set for each bagging iteration. Figure 2 shows the pseudocode for training the BES-OOB strategy. Please note that the term "classifier" in the pseudocode is used to refer to both classification and regression classifiers. An advantage of BES-OOB is that the user does not need to choose the size of the hillclimb set as in the ES algorithm. Bagging and ES are both generic strategies for supervised learning. When they are used for the regression setting, we can simply use regression algorithms as the base classifiers. BES-OOB combines the two strategies, therefore it is also a generic ensemble learning strategy for both classification and regression.

In the original ES algorithm, Step 6 in Figure 2 uses the forward step-wise selection method for ensemble construction. For the details of the method, we refer readers to [6]. In this paper, we also use the same method. Our emphasis here is to show that any "greedy" (active set) ensemble construction method that

BES-OOB(S, E, T)

S is the training set
E is the Ensemble Selection classifier
T is the number of bootstrap samples

```
1: H ← empty ensemble
2: for i ← 1 to T {
3:     S_b ← bootstrap sample from S
4:     S_oob ← out of bag sample
5:     train base classifiers in E on S_b
6:     E_i ← do ensemble selection based on
                 base classifiers' performance on S_oob
7:     add E_i to H
8: }
9: return H
```

Fig. 2. Pseudocode of the BES-OOB algorithm.

utilizes the out-of-bag sample for performance estimation can be used in Step 6 of BES-OOB. Although in recent years, the ES algorithm has been highlighted in winning solutions of many data mining competitions [15], there is no theoretical work on explicitly examining the convergence property of the ES algorithm. Here we attempt to give a brief discussion on the theoretical aspect. If we see the forward step-wise method used in ES as a "greedy" feature selection algorithm, then the predictions of each base regression classifier can be seen as the "feature" values. Based on the theoretical work on forward feature selection [4], when certain conditions are met, such as sufficient conditions for convex optimization, use of squared error loss and in absence of noise, the theoretical convergence rate of the forward step-wise method used in ES is sublinear at $m^{-\frac{1}{2}}$, where m is the number of base classifiers. A comprehensive theoretical analysis of the ES algorithm is beyond the scope of this paper and we leave it for future research.

3 Experiments

In this section, we conduct a series of experiments and statistical tests to examine the performance of the BES-OOB ensemble strategy for regression.

3.1 Comparison to Other Ensemble Strategies

Firstly we compare BES-OOB to three state-of-the-art ensemble strategies for regression: Stochastic Gradient Boosting (SGB) [8], standard Bagging (BG) [2] and an ensemble of Bagging and Stochastic Gradient Boosting, denoted by BSGB. BSGB can be seen as a variant of the MultiBoosting algorithm [16], in which SGB is used as a base learner for Bagging. The experiments are based on 42 regression data sets from UCI repository[1] and StatLib[2]. We use 10 times 10-fold

[1] http://archive.ics.uci.edu/ml
[2] http://lib.stat.cmu.edu

Dataset	BES-OOB	SGB		BG		BSGB	
quake	0.12	0.06	•	0.12		0.12	
cholesterol	0.19	0.07	•	0.18		0.19	
detroit	0.22	0.03	•	0.24		0.24	
breastTumor	0.22	0.16	•	0.22		0.22	
meta	0.38	0.14	•	0.36	•	0.38	
veteran	0.42	0.26	•	0.42		0.42	
schlvote	0.45	0.10	•	0.46		0.43	
sensory	0.53	0.48	•	0.53		0.52	
longley	0.54	0.43	•	0.54	•	0.50	•
strike	0.55	0.41	•	0.55		0.55	
kidney	0.55	0.38	•	0.53	•	0.58	○
baskball	0.55	0.43	•	0.55		0.56	
newton-hema	0.57	0.54		0.58	○	0.59	○
pbc	0.57	0.52	•	0.56	•	0.58	○
stanford	0.60	0.43	•	0.63		0.63	
sleep	0.64	0.52	•	0.64		0.62	
hungarian	0.65	0.63	•	0.65	•	0.66	○
winequality-red	0.66	0.61	•	0.66	•	0.68	○
echoMonths	0.67	0.69	○	0.69	○	0.68	
winequality-white	0.68	0.62	•	0.67	•	0.70	○
cleveland	0.69	0.63	•	0.69		0.70	○
pollution	0.75	0.51	•	0.73	•	0.75	
vineyard	0.76	0.67	•	0.75	•	0.76	
lowbwt	0.79	0.78	•	0.79		0.79	
elusage	0.82	0.81		0.82		0.84	○
vinnie	0.86	0.85	•	0.86	○	0.86	○
bolts	0.86	0.83		0.83	•	0.86	
gascons	0.88	0.76	•	0.84		0.83	
cloud	0.91	0.84	•	0.90	•	0.91	
autoMpg	0.91	0.92		0.91	•	0.93	○
servo	0.92	0.91		0.91	•	0.93	○
pwLinear	0.92	0.92		0.92		0.93	○
housing	0.92	0.90	•	0.91	•	0.93	○
boston	0.92	0.91	•	0.92	•	0.93	○
socmob	0.92	0.92		0.91	•	0.94	○
autoHorse	0.93	0.91	•	0.90	•	0.93	
autoPrice	0.93	0.92	•	0.93	•	0.94	○
cpu	0.97	0.93	•	0.96	•	0.98	
strikes	0.98	0.97	•	0.96	•	0.98	•
fishcatch	0.98	0.96	•	0.97	•	0.97	•
visualizing-galaxy	0.99	0.98	•	0.98	•	0.99	○
bodyfat	0.99	0.98	•	0.98	•	0.98	•

• ○, BES-OOB is significantly better or worse

	BES-OOB against		
	SGB	BG	BSGB
win/tie/loss	34/7/1	23/16/3	4/21/17

Table 1. Estimated correlation coefficients of BES-OOB, SGB, BG, and BSGB; and Win/tie/loss counts of *paired t-test*.

cross-validation to estimate the performance of each strategy. Then, several statistical significance tests are conducted, including the non-parametric *Friedman-test* and the *Bonferroni-Dunn test* as described in [7]. This approach utilises the ranking information of each learner in comparison, which is suitable for comparing multiple learners on multiple data sets. The total numbers of win, tie and loss for the *paired t-test* (with significance level 0.05) are also recorded. To fairly compare the four strategies, REPTree (a CART-like regression tree) [10] is used as the base learner. The ensemble size is set to 1,500 for all these strategies. For SGB, the shrinkage parameter is set to 0.5 and the subsample size parameter is set to 50%. For BES-OOB, the number of base learners per "bag" is set to 30, and the number of bagging iterations is set to 50. Also, BES-OOB is set to optimise the correlation coefficient metric. For BSGB, the number of base learners for SGB (shrinkage is set to 0.5; subsample size is 50%) is set to 30, and the number of bagging iterations is set to 50. Table 1 presents the *paired t-test* results. Correlation coefficient scores are reported. Figure 3 is the graphical representation of the *Friedman-test* for the four strategies. We can see that both BES-OOB and BSGB significantly outperform BG and SGB, and BG significantly outperforms SGB. There is no significant difference between BES-OOB and BSGB's performance over the 42 data sets.

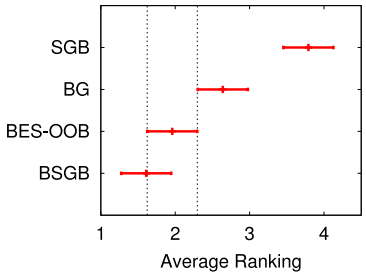


Fig. 3. Visualization of the *Friedman-test* results for BES-OOS, SGB, BG, and BSGB with REPTree as base learners over 42 data sets. The middle point of each bar indicates the average rankings, and the bars indicate the critical values of the *Bonferroni-Dunn test* (two-tailed test at significance level 0.05). Strategies having non-overlapped bars are significantly different.

3.2 Diverse Model Libraries

In the previous experiments, we have been testing on a single type base learner. However, one distinguishing feature of BES-OOB is that it can use different types of base learners. In this section, BES-OOB with a diverse model library consisting of three types of base learners (REPTree, SVM regression and M5P model tree [13]), denoted by BES-diverse, is compared to BES-OOB with only one of the three base learners, denoted by BES-reptree, BES-svm, and BES-m5p, respectively. Three different model library sizes are tested: 3, 30 and 300. The experimental setup is as follows: the number of bagging iterations for all BES strategies in comparison is set to 30; for BES-diverse, when the model library size is 3, only one of each type of base learners is used; when the model library size is 30, 10 of each type of base learners are used; so 100 of each type of the base learners are used for a model library size of 300. The correlation coefficient is set as the goal metric for all strategies.

Diversity is one of the key factors for ensemble learning. To simplify the procedure for generating diverse base learners, we adopt the "random subspace" idea [3] for each base learner in the library. That is, each base learner trains on a random subset (33% is used for all experiments) of the original variables. For REPTree, Weka default parameters are used, and we also randomly set its random seed; for SVM regression, we use the LibSVM default parameters for epsilon-SVM regression and RBF kernel, except the gamma value is randomly set to be between 0 and 1. We use the Weka default parameters for M5P model tree. Table 2 shows the *paired t-test* results of BES-OOB-diverse against BES-OOB-reptree, BES-OOB-svm and BES-OOB-m5p under three different model library sizes: 3, 30, and 300, respectively. Please note that the number of bagging iterations is set to 30. Therefore, the numbers of base learners that are allowed to be built for the three model library sizes are: 90, 900, and 9,000, respectively.

Figure 4 shows the average *Friedman-test* rankings of each strategy under the three model library sizes. We can see that the ranking of BES-OOB-diverse

Model library size = 3			
	BES-OOB-diverse against		
	A1	A2	A3
win/tie/loss	4/20/18	32/8/2	5/13/24

Model library size = 30			
	BES-OOB-diverse against		
	A1	A2	A3
win/tie/loss	19/15/8	34/6/2	9/20/13

Model library size = 300			
	BES-OOB-diverse against		
	A1	A2	A3
win/tie/loss	5/37/0	14/28/0	4/38/0

Table 2. Win/tie/loss counts of *paired t-test* for BES-OOB-diverse against BES-OOB-reptree (A1), BES-OOB-svm (A2) and BES-OOB-m5p(A3).

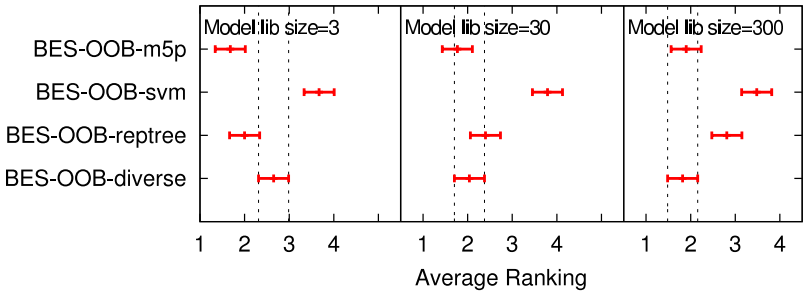


Fig. 4. Friedman average rankings under different model library sizes.

improves (from the third to the second, and finally to the first) when the model library size increases. The result implies that the advantage of using a diverse model library becomes clear when the model library size is relatively large. To the best of our knowledge, this is a novel result in the regression setting.

3.3 Pruning an Ensemble of ES Ensembles

Until now, we have been using the standard output aggregation method for BES-OOB. That is, the final prediction of BES-OOB is simply the average of all individual ES learners. The final ensemble size of BES-OOB is therefore the number of bagging iterations. In this section, we consider methods for ensemble pruning. Usually, there are two main reasons for doing ensemble pruning. The first is to reduce the prediction cost (e.g., runtime or memory requirements) without sacrificing too much predictive performance. The second is to obtain a more accurate model. Based on the theoretical work of [20] in the study of neural networks, we know that theoretically "many could be better than all". This implies that the performance of an optimal subset of base learners may outperform the population average. Since the default BES-OOB strategy uses simple averaging, appropriate ensemble pruning may improve BES-OOB's performance in terms of both accuracy and prediction cost. We compare simple averaging to two pruning methods: pruning with the cocktail ensemble (CE) algorithm, and pruning with the stacking strategy using the non-negative least-squares (NNLS)

BES-OOB-CE(S, E, T)
S is the training set
E is the Ensemble Selection (ES) learner
T is the number of bootstrap samples

1: $H \leftarrow$ BES-OOB(S,E,T) // an ensemble of T ES ensembles
2: $f_1^c \leftarrow$ the ensemble in H with the smallest out-of-bag estimate of error
3: $e_{min} = +\infty$
4: for $i \leftarrow 2$ to T
5: $f_i \leftarrow null$
6: for each $f \in H$
7: $e \leftarrow$ estimated error of combing f and f_{i-1}^c
8: if $e < e_{min}$ then $f_i \leftarrow f$ and $e_{min} \leftarrow e$
9: }
10: if f_i is *null* then $f_N^c \leftarrow f_{i-1}^c$ and *break*
11: $f_i^c \leftarrow p_i f_{i-1}^c + (1 - p_i) f_i$, where p_i is obtained by Eq.2 for the mean squared error
12: }
13: return f_N^c

Fig. 5. Pseudocode of the CE method for BES-OOB ensemble pruning.

algorithm as the meta-level learner. The three methods in this comparison are denoted by BES-OOB-avg, BES-OOB-ce, and BES-OOB-nnls, respectively. Next, we briefly introduce the BES-OOB-ce and the BES-OOB-nnls methods.

Cocktail ensemble (CE) [18], is a novel method of ensemble learning. One reason for using CE as an ensemble pruning method for BES-OOB is that the authors explicitly mentioned that the method is proposed for learning ensemble of ensembles. Since combination of multiple ensembles (equivalent to finding the optimal weights for each base learner) is an NP-hard problem [11], the authors of [18] proposed using the pair-wise combination for multiple ensembles. In addition, CE has an appealing mathematical foundation, which we will briefly discuss here. For a full account of the method, we refer readers to [18]. The basic idea is that, given two ensembles f_1 and f_2, a linear ensemble of ensembles f_1 and f_2 can be expressed as:

$$f^c = pf_1 + (1-p)f_2, \quad wrt \quad p \in [0,1]$$

where p is the weight for f_1 and $1-p$ is the weight for f_2. Then, the optimal weight of f_1 is:

$$p^* = \frac{E_2 - E_1}{2\triangle} + 0.5, \tag{2}$$

where E_1 and E_2 are the generalization errors of f_1 and f_2, and $\triangle = \mathbb{E}_x[(f_1 - f_2)^2]$ is the squared output difference of the two ensembles. Here E_1, E_2 and $\triangle$ can be estimated from data (in BES-OOB, we use the out-of-bag sample). Figure 5 shows the pseudocode for the CE method, which has been adapted for BES-OOB pruning.

Stacking, or stacked generalisation [17], is a popular ensemble learning strategy, where the weights of the base classifiers are the regression coefficients of the meta-level regressor. Usually linear regression (LR) is used at the meta-level.

Since our goal is to prune an ensemble, simply using LR would not reduce the ensemble size. Here, we propose using stacking with the NNLS algorithm. To the best of our knowledge, this is the first time that stacking with NNLS is considered as an ensemble pruning approach. Eq. 5 shows the basic form of the NNLS optimisation problem.

$$\min_{w \geq 0} \|Xw - y\|_2^2. \tag{3}$$

Here, X is the data matrix, w is the regression coefficient vector, and y is the target matrix. We can see that it is the same as the linear least-squares regression form, but with extra constraints on the values of the coefficient vector. For our experiment, we use the NNLS algorithm proposed in [12]. The BES-OOB ensemble strategy constructs an ensemble of ES ensembles. Each individual ES is trained on a corresponding bootstrap sample, and its ensemble selection is guided by its performance on the out-of-bag sample. The basic steps of using stacking with NNLS for BES-OOB pruning are as follows: Suppose S is the training set, and H is an ensemble of ES ensembles (same as line 1 in Figure 5). A meta-dataset can be constructed by using the predictions of each ES in H on S. The targets in S are used as the targets of the meta-dataset. Then, we use NNLS to build a model on the meta-dataset. The NNLS regression coefficients are used as the weights for each ES. Therefore, the final ensemble consists only of ES ensembles with greater than zero weight.

The experimental setup is as follows: the number of bagging iterations is set to 30 for all three methods (BES-OOB-avg/ce/nnls). For each bagging iteration, one REPTree-based ES learner is trained. The number of trees used for each ES is 10. As in the previous experiment, each REPTree is built using a random 33% of the original attributes. So in total 300 REPTree learners are built for each of the three methods in the comparison. At the individual bagging iteration level, all three BES-OOB methods are set to optimise the mean squared error (MSE) metric. At the pruning level, BES-OOB-ce is also set to optimise the MSE metric based on Eq. 2. Also, based on Eq. 5, we know that BES-OOB-nnls optimises square error by default. In total 42 data sets are used for this experiment.

Table 3 shows the *corrected paired t-test* results. The reported root relative squared errors are estimated from 10 times 10-fold cross-validation. The final ensemble sizes, and the final number of trees, are also reported. Figure 6 shows the *Friedman-test* results. Based on the *corrected paired t-test* results, we can see that both of the two pruning methods, BES-OOB-ce and BES-OOB-nnls, show competitive predictive performance compared to BES-OOB-avg, but with smaller final ensemble sizes and final number of trees. There are no significant *t-test*-based performance differences between BES-OOB-avg and the two pruning methods on most of the 42 data sets (39 for BES-OOB-ce; 37 for BES-OOB-nnls) in this experiment. Based on the *Friedman-test* and the *Bonferroni-test*, we can see that the performance of BES-OOB-avg and BES-OOB-nnls has no significant differences over the 42 data sets.

The final ensemble size for BES-OOB-avg is 30 (equal to the number of bagging iterations). Over the 42 data sets, the average final ensemble size of

Data set	Root Relative Squared Error			Ensemble Size			Number of Trees		
	nnls	ce	avg	nnls	ce	avg	nnls	ce	avg
autoHorse	34.60	36.87	34.78	5.7	5.6	30	22.0	22.2	118.8
autoMpg	37.53	38.48	37.58	8.7	4.6	30	38.4	21.3	138.6
autoPrice	34.99	38.52	35.91	6.2	4.3	30	23.3	17.2	114.1
baskball	84.46	85.26	83.69	4.9	4.2	30	15.0	12.5	98.2
bodyfat	14.79 ∘	**27.53**	17.13	5.3	3.4	30	13.3	9.7	75.2
bolts	31.39	36.12	35.50	5.4	4.5	30	14.7	14.4	91.0
boston	41.85	44.99	43.02	7.4	5.3	30	28.7	23.4	124.9
breastTumor	97.40	97.75	96.50	4.5	4.1	30	15.9	15.7	104.2
cholesterol	101.31	100.47	99.15	3.7	3.9	30	13.6	15.0	108.1
cleveland	74.15	75.01	73.78	5.9	4.0	30	22.0	17.0	119.0
cloud	44.80	47.19	44.09	5.6	5.3	30	20.3	19.5	109.5
cpu	19.46 ∘	**29.41**	22.10	4.8	4.1	30	21.3	16.7	115.0
detroit	151.11	283.02	151.73	3.6	19.3	30	11.5	61.9	94.2
echoMonths	72.99	71.40	70.55	3.8	3.8	30	11.7	10.9	89.3
elusage	49.91	49.52	48.76	6.0	7.2	30	16.0	21.0	83.2
fishcatch	19.84	24.40	21.24	7.5	3.7	30	26.9	13.0	106.7
gascons	25.50	30.09	25.24	5.2	9.1	30	19.5	35.8	115.2
housing	41.57	45.03	43.16	7.3	5.2	30	28.5	21.2	123.7
hungarian	74.99	75.10	74.36	4.7	5.2	30	15.3	17.8	102.2
kidney	78.38	82.08	81.29	4.6	5.2	30	12.9	16.0	91.2
longley	47.35	61.70	49.97	5.0	12.5	30	16.4	39.8	102.1
lowbwt	63.02	64.00	61.62	4.4	3.9	30	13.8	11.6	92.3
meta	149.15	147.21	112.30	1.6	3.9	30	4.8	14.3	94.2
newton-hema	85.56	85.20	82.78	3.9	5.4	30	13.2	17.4	97.1
pbc	84.45	85.07	84.24	4.9	6.1	30	18.2	24.5	116.6
pollution	71.70	77.01	72.06	5.8	4.6	30	18.8	16.2	101.5
pwLinear	• **48.92**	55.53	53.93	5.0	3.8	30	14.0	12.9	96.8
quake	100.00	99.78	99.54	3.5	6.2	30	18.0	28.3	132.6
schlvote	84.32 ∘	**110.16**	79.01	2.9	12.6	30	8.4	36.4	84.8
sensory	• **84.82**	86.99	86.97	5.4	5.8	30	16.7	20.7	103.7
servo	37.66	44.59	43.44	3.9	7.0	30	10.9	23.3	94.0
sleep	82.01	80.57	77.49	4.3	4.5	30	11.9	13.3	90.0
socmob	• **37.24**	41.23	39.81	6.5	4.4	30	19.9	15.7	100.0
stanford	92.16	95.82	88.37	4.1	6.9	30	12.6	21.8	91.7
strike	88.44	87.79	80.79	3.3	4.8	30	11.8	16.9	111.8
strikes	• **0.41**	8.95	4.42	1.4	10.0	30	2.9	21.9	66.0
veteran	97.46	93.98	91.51	3.3	4.3	30	10.3	13.5	94.8
vineyard	62.82	63.97	62.57	4.5	8.1	30	14.2	24.1	95.1
vinnie	• **50.89**	52.39	51.30	7.7	4.5	30	23.5	13.3	88.4
visualizing-galaxy	15.60	16.31	15.65	10.5	5.7	30	42.7	23.6	126.0
winequality-red	77.08	77.98	77.39	7.4	9.8	30	40.2	57.8	166.1
winequality-white	76.13	76.75	76.39	9.0	15.4	30	63.4	106.1	215.1
Average	63.53	69.79	62.65	5.2	6.2	30	19.0	23.2	106.7
(win/tie/loss); avg vs. nnls: 0/37/5; avg vs. ce: 3/39/0;									

• ∘, BES-OOB-avg is significantly worse or better, respectively; at significance level 0.05

Table 3. The Root Relative Squared Error values, the ensemble sizes, and the number of trees in the final ensemble for BES-OOB-avg, BES-OOB-ce and BES-OOB-nnls, over 42 data sets.

BES-OOB-ce is 6.2, corresponding to a 70% reduction in terms of ensemble size; the average final ensemble size of BES-OOB-nnls is 5.2, corresponding to a 83% reduction in terms of ensemble size. The average final number of trees of BES-OOB-avg is 106.7, which is about 36% of the total 300 trees. The average final number of trees of BES-OOB-ce is 23.2, corresponding to a 78% ((106.7 - 23.2)/106.7) reduction in terms of number of trees; the average final number of trees of BES-OOB-nnls is 19.0, corresponding to a 82% reduction in terms of number of trees. Figure 7 shows the boxplot visualization for the ensemble sizes and the numbers of trees of BES-OOB-avg, BES-OOB-ce and BES-OOB-nnls. Notably, the BES-OOB-nnls method significantly outperforms the BES-OOB-avg method on 5 data sets (about 12% of the 42 data sets). This is a significant empirical result indicating that BES-OOB-nnls not only works well for ensemble pruning, but also could be used for further improving the predictive performance of the BES-OOB strategy.

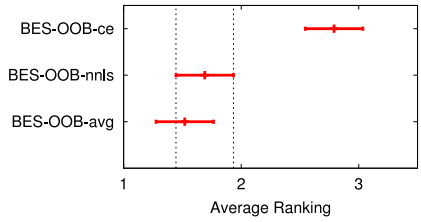
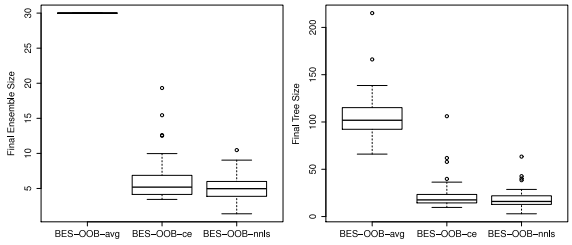


Fig. 6. The result of the *Friedman-test* over 42 data sets with two-tailed *Bonferroni-Dunn test* at significance level 0.05. Strategies having non-overlapped bars are significantly different.



(a) Final Ensemble Size (b) Number of trees

Fig. 7. The boxplot visualization for the final average ensemble sizes and the final average tree sizes.

4 Conclusions

Bagging ensemble selection using the out-of-bag sample for hillclimbing (BES-OOB), is a relatively new ensemble learning strategy. In this paper, we studied the predictive performance of BES-OOB in the regression setting. The main contributions of this paper are:

- Previous studies focused on using BES-OOB for classification problems only. In this paper, through a series of experiments and statistical tests, we have shown that, in the regression setting, the BES-OOB strategy is competitive to MultiBoosting, and is superior to Bagging and Stochastic Gradient Boosting when using CART-like regression trees as the base learners.
- We have shown that using a diverse model library could further boost BES-OOB's predictive performance when the model library size is relatively large.
- Our results also have shown that both the cocktail ensemble and the stacking with NNLS methods work well for BES-OOB ensemble pruning. Particularly, the latter method can be also used to improve the predictive performance of the BES-OOB strategy.

One reason for the good predictive performance of the BES-OOB strategy is that it can optimise a user-specified error metric directly in the base learner

selection stage. Out-of-bag samples seem to work well for ES's ensemble selection in practice. Another notable feature of BES-OOB is its simplicity and ease of implementation. The success of the BES-OOB ensemble strategy over a broad range of data sets examined in this study strongly suggests the applicability of the method to a wide range of problems.

References

1. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine learning 36(1-2), 105–139 (Jul 1999)
2. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
3. Bryll, R., Gutierrez-Osuna, R., Quek, F.: Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition 36(6), 1291–1302 (2003)
4. Buhlmann, P., van de Geer, S.: Statistics for High-Demensional Data: Methods, Theory and Applications. Springer (2011)
5. Caruana, R., Munson, A., Niculescu-Mizil, A.: Getting the most out of ensemble selection. In: ICDM '06 Proceedings of the Sixth International Conference on Data Mining (2006)
6. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: ICML '04 Proceedings of the twenty-first international conference on machine learning (2004)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (December 2006)
8. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics and Data Analysis 38, 367–378 (1999)
9. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics 29, 1189–1232 (2000)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: An update. SIGKDD Explorations 11(1) (2009)
11. Hernandez-Lobato, D., Martinez-Munoz, G., Suarez, A.: Pruning in ordered regression bagging ensembles. In: IJCNN '06 International Joint Conference on Neural Networks. pp. 1266 –1273 (2006)
12. Lawson, C., Hanson, R.: Solving Least-Squares Problems. Prentice-Hall (1974)
13. Quinlan, J.R.: Learning with continuous Classes. In: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence. pp. 343–348 (1992)
14. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33, 1–39 (2010)
15. Sun, Q., Pfahringer, B.: Bagging ensemble selection. In: Proceedings of the 24th Australasian Conference on Artificial Intelligence. pp. 251–260. Perth, Australia, Springer (2011)
16. Webb, G.I.: Multiboosting: A technique for combining boosting and wagging. Machine Learning 40(2) (2000)
17. Wolpert, D.H.: Stacked generalization. Neural Networks 5, 241–259 (1992)
18. Yu, Y., Zhou, Z.H., Ting, K.M.: Cocktail ensemble for regression. In: Seventh IEEE International Conference on Data Mining, 2007. ICDM 2007. pp. 721 –726 (2007)
19. Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. Boca Raton, FL: Chapman & Hall/CRC (2012)
20. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. Artificial Intelligence 137, 239–263 (May 2002)