Bagging Ensemble Selection

Quan Sun and Bernhard Pfahringer

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

Abstract. Ensemble selection has recently appeared as a popular ensemble learning method, not only because its implementation is fairly straightforward, but also due to its excellent predictive performance on practical problems. The method has been highlighted in winning solutions of many data mining competitions, such as the Netflix competition, the KDD Cup 2009 and 2010, the UCSD FICO contest 2010, and a number of data mining competitions on the Kaggle platform. In this paper we present a novel variant: bagging ensemble selection. Three variations of the proposed algorithm are compared to the original ensemble selection algorithm and other ensemble algorithms. Experiments with ten real world problems from diverse domains demonstrate the benefit of the bagging ensemble selection algorithm.

1 Introduction

The problem of constructing an ensemble of classifiers from a library of base classifiers has always been of interest to the data mining community. Usually, compared with individual classifiers, ensemble methods are more accurate and stable. We here reproduce the mathematical expression used in [8] to illustrate the idea of ensemble learning: let x be an instance and $m_i, i = 1...k$, a set of base classifiers that output probability distributions $m_i(x, c_j)$ for each class label $c_j, j = 1...n$. The output of the final classifier ensemble $y(x)$ for instance x can be expressed as:

$$y(x) = \arg\max_{c_j} \sum_{i=1}^{k} w_i m_i(x, c_j), \tag{1}$$

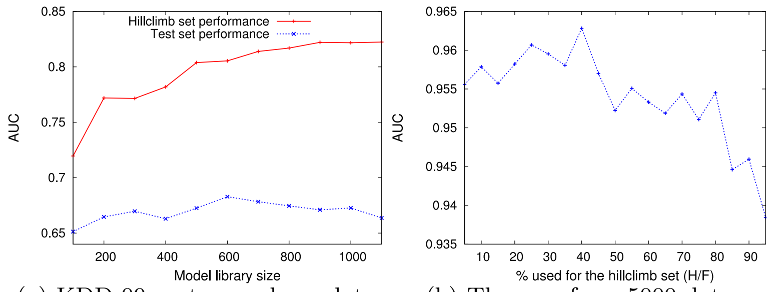
where w_i is the weight of base classifier m_i. In this particular form, ensemble learning strategies can be seen as methods for calculating optimal weights for each base classifier in terms of a classification goal. Since the mid-90's, many ensemble methods have been proposed. For a more detailed review of recent developments please refer to [2, 9].

Before introducing the new methods, we briefly review bagging (bootstrap aggregating) [3] and the ensemble selection algorithm proposed in [5]. Bagging is based on the instability of base classifiers, which can be exploited to improve the predictive performance of such unstable base classifiers. The basic idea is that, given a training set T of size n and a classifier A, bagging generates m new training sets with replacement, T_i, each of size $n' \leq n$. Then, bagging applies A to each T_i to build m models. The final output of bagging is based on simple voting [2].

Ensemble selection is a method for constructing ensembles from a library of base classifiers [5]. Firstly, base models are built using many different machine learning algorithms. Then a construction strategy such as forward stepwise selection, guided by some scoring function, extracts a well performing subset of all models. The simple forward model selection based procedure proposed in [5] 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 to the 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 ensemble selection is that it can be optimised for many common performance metrics or a combination of metrics. For variants of the ensemble selection algorithm, the reader is referred to [4, 5]. In the next section, we will describe the proposed bagging ensemble selection algorithms and explain the motivation of combining bagging and ensemble selection.

2 Bagging Ensemble Selection

Based on the data sets and comparison results from [5], the simple forward model selection based ensemble selection algorithm is superior to many other well-known ensemble learning algorithms, such as stacking with linear regression at the meta-level, bagging decision trees, and boosting decision stumps. However, sometimes ensemble selection overfits the hillclimbing set, reducing the performance of the final ensemble. Figure 1(a) shows the hillclimb and test



(a) KDD 09 customer churn data (b) The waveform-5000 data

Fig. 1. Ensemble selection hillclimb and test set learning curves

set learning curves of running ensemble selection on a data set. The red curve is the hillclimb set performance and the blue curve is the test set performance. It demonstrates that as the number of models in the model library increases, the performance (in terms of AUC) of ensemble selection on the hillclimb set gradually increases. However, the corresponding performance on the test set does not always increase; it may reach a peak (local or global) and then gradually decline. Also, as indicated in [5], for certain data sets, the root-mean-squared-error metric sometimes can decline very quickly. To overcome this problem, the authors of [5] proposed three additions to the simple forward selection procedure to reduce the chance of hillclimb set overfitting. The proposed additions are: (1) selection with replacement, where each individual classifier can be selected multiple times, which means some classifiers get larger weights than others; (2) sorted ensemble initialization, where instead of starting with an empty ensemble, models in the library are sorted by their performance, and the best N models are put into the initial ensemble; (3) "bagged" ensemble selection, where K groups (bags) of models are randomly selected from the model library, and ensemble selection is done inside each bag; the final ensemble is the union of the subsets selected for each of the bags. All three procedures also introduce additional parameters to the simple ensemble selection algorithm.

Furthermore, there is one more issue: how much data should be used for the hillclimb set? Figure 1(b) shows a typical test set learning curve for running ensemble selection with hillclimb sets of varying sizes. Assume the training set is F, and the hillclimb set H is a subset of F. Here, the x-axis shows the ratio H/F and indicates the percentage of F that is used for the hillclimb set. Based on the learning curve, we can see that the performance of ensemble selection is not stable, and is related to how much data is used for H. In the figure, there is a performance peak at $x = 40\%$, but performance starts to drop from $x = 50\%$. Different data sets may have different optimal ratios, which usually can be found only by using cross-validation. Therefore, this parameter indirectly increases the complexity of ensemble selection. Based on these observations, we propose a new ensemble learning algorithm called bagging ensemble selection: if we view the simple forward ensemble selection algorithm as an unstable base classifier, then we can apply the bagging idea to construct an ensemble of simple ensemble selection classifiers, which should be more robust than an individual ensemble selection classifier. In addition, the respective out-of-bag samples can be used as the hillclimb set. Specifically we will use the following three variations of bagging ensemble selection.

The **BaggingES-Simple** algorithm is the straightforward application of bagging to ensemble selection, with ensemble selection being the base classifier inside bagging. In this algorithm, the amount of data used for the hillclimb set is still a user-specified parameter (with a default of 30%). Each bootstrap sample is split into a train and a hillclimbing set according to this parameter.

The **BaggingES-OOB** algorithm uses the full bootstrap sample for model generation, and the respective out-of-bag instances as the hillclimb set for selection. The bootstrap sample is expected to contain about $1 - 1/e \approx 63.2\%$ of the unique examples of the training set [1, 3]. Therefore the hillclimb set (out-of-bag sample) is expected to have about $1/e \approx 36.8\%$ unique examples of the train-

```
Inputs:
  Training set $S$; Ensemble Selection classifier $E$; Integer $T$ (number of bootstrap samples)

Basic procedure:
  for $i = 1$ to $T$ {
      $S_b$ = bootstrap sample from $S$ (i.i.d. sample with replacement)
      $S_{oob}$ = out of bag sample
      train base classifiers (can be a diverse model library) in $E$ on $S_b$
      $E_i$ = do ensemble selection based on base classifiers' performance on $S_{oob}$
  }
```

Fig. 2. Pseudocode of the BaggingES-OOB algorithm

ing set for each bagging iteration. An advantage of BaggingES-OOB is that the user does not need to choose the size of the hillclimb set. Figure 2 shows the pseudocode for training the BaggingES-OOB ensemble.

The **BaggingES-OOB-EX** algorithm is an extreme case of BaggingES-OOB, where in each bagging iteration only the single best classifier (in terms of performance on the hillclimb set) is selected. Therefore, if the number of bagging iterations is set M, then the final ensemble size will be exactly M as well.

3 Experimental Results

We experiment with ten classification problems. All of them are real world data sets which can be downloaded from the UCI repository [6], the UCSD FICO data mining contest website[1] and the KDD Cup 2009 website[2]. These data sets were selected because they are large enough, and they come from very different research and industrial areas. Table 1 shows the basic properties of these data sets. To make experiments possible for large model libraries, selecting from thousands of base classifiers, all five multiclass data sets were converted to binary problems by keeping only the two largest classes each. After this conversion to binary problems, for data sets that are larger than 10,000 instances, a subset of 10,000 instances is randomly selected for our experiments. Table 1 (in the rightmost column) shows the basic properties of the final data sets.

Ensemble selection is not restricted by the type of base classifiers used. Theoretically, any classifier can be used as a base classifier for ensemble selection. In this paper, the WEKA [7] implementation of the random tree classifier is used as the base classifier for all experiments. There are two reasons for focussing solely on random trees as base classifiers. The first one is simplicity: just by varying a single parameter, the random seed, we can obtain a large and relatively diverse model library. The second one is fair comparsion: most other ensemble methods

[1] The University of California, San Diego and FICO 2010 data mining contest, http://mil.ucsd.edu/

[2] The KDD Cup 2009, http://www.kddcup-orange.com/

Table 1. Data sets: basic characteristics

Data set with release year	#Insts	Atts:Classes	Class distribution (#Insts)
Adult 96	48,842	14:2	23% vs 77% (10,000)
Chess 94	28,056	6:18	48% vs 52% (8,747)
Connect-4 95	67,557	42:3	26% vs 74% (10,000)
Covtype 98	581,012	54:7	43% vs 57% (10,000)
KDD09 Customer Churn 09	50,000	190:2	8% vs 92% (10,000)
Localization Person Activity 10	164,860	8:11	37% vs 63% (10,000)
MAGIC Gamma Telescope 07	19,020	11:2	35% vs 65% (10,000)
MiniBooNE Particle 10	130,065	50:2	28% vs 72% (10,000)
Poker Hand 07	1,025,010	11:10	45% vs 55% (10,000)
UCSD FICO Contest 10	130,475	334:2	9% vs 91% (10,000)
Original data sets			Final binary data sets

are limited to uniform base classifiers. To speed up our experiments, parameter K of the random tree, the number of random attributes, is always set to 5, and the minimum number of instances at each leaf node is set to 50. In [5], the authors have shown that ensemble selection can be optimised to many common evaluation metrics. Bagging ensemble selection inherits this very useful feature; the goal metric is therefore a user-specified parameter. In this paper, the AUC (area under the ROC curve) metric is used for all experiments.

The following sections present two sets of results. One shows the results from comparing the three bagging ensemble selection algorithms to the simple forward ensemble selection algorithm (ES) and the ES++ algorithm, which is the improved version of ES with the three additions, as described in the introduction. This is followed by an analysis of the final ensemble sizes for these algorithms. The other set of results shows a comparison between bagging ensemble selection and other ensemble learning algorithms.

3.1 Comparison of Bagging Ensemble Selection Algorithms to the Forward Ensemble Selection Algorithms

In this experiment the following setup is used: the number of bags (bagging iterations) for BaggingES-Simple, BaggingES-OOB and BaggingES-OOB-EX is set to 50. For each data set, we run 10 experiments per algorithm, increasing the size of the model library per bag by 10 for each successive experiment: from 10 to 20, then to 30 and so on until 100 for the tenth experiment. For example, when the size of the model library is 100, then, in total, 5,000 base classifiers (random trees) are trained. Accordingly, we run 10 experiments on each data set for the ES algorithm and the ES++ algorithm (hillclimb ratio is set to 30% for both ES and ES++) that we want to compare. The size of the model library increases by 500 in each successive experiment, from a base 500 to 1,000, then 1,500 until it reaches 5,000 in the tenth experiment, which means all five algorithms in the comparison use the same number of base classifiers in each individual experiment. Also, for the ES++ algorithm, the number of subgroups is set to 50.

Figure 4 shows the test set learning curves of the ES algorithm, the ES++ algorithm, and the three bagging ensemble selection algorithms based on 500 individual experiments (5 algorithms, 10 data sets, 10 different model library sizes per data set). For each experiment, the algorithms are trained on 66% of the data set and evaluated on the other 34%. We repeated each experiment five times and the mean values were used for generating the figures and comparison. Based on Figure 4, we can see that ES and ES++ outperform bagging ensemble selection when the size of the model library is greater than 1,000 on the Adult-96 data set. For all other nine data sets, bagging ensemble selection, particularly BaggingES-OOB (blue curves) and BaggingES-OOB-EX (green curves), clearly outperform the ES algorithm and the ES++ algorithm. For data sets Chess-94, KDD-09 and Localization-10, BaggingES-OOB and BaggingES-OOB-EX gave similar performance.

An interesting pattern is that, for data sets Connect-4-95, Magic-07 and UCSD-10, the test performance of BaggingES-OOB-EX declines as the size of the model library increases. This is probably due to the fact that model diversity is more important for these data sets than for others. Thus, as the model library gets larger and larger, the best base classifier of each of the 50 bags of BaggingES-OOB-EX might become more similar to each other, thus losing model diversity.

For 6 out of 10 model library sizes, the BaggingES-Simple algorithm outperforms all other algorithms on the UCSD-10 data set. The ES++ algorithm outperforms other algorithms on the UCSD-10 data set when model library sizes are 500 and 5,000, but had a relatively poor performance when model library size is 1,000. Again, we can see that, for Covtype-98, KDD-09, MiniBooNe-10 and UCSD-10, the learning curves of the ES algorithm are not very stable. Figure 3 (left panel) shows the histogram presentation of the performance in terms of the number of wins for each algorithm over the ten data sets. We can see that BaggingES-OOB and BaggingES-OOB-EX are the top two winners.

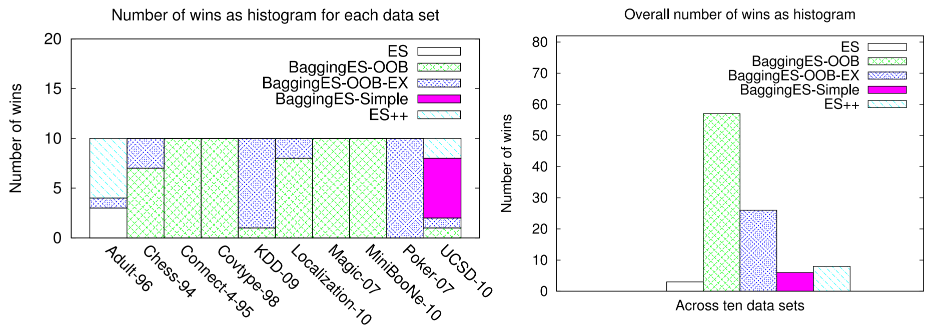


Fig. 3. Histogram presentation for counting number of wins for each algorithm

Next, we look at the final ensemble sizes of ES, ES++, BaggingES-OOB, BaggingES-OOB-EX and BaggingES-Simple. Figure 5 shows the relationship between model library size and the final ensemble size for these algorithms on the ten data sets. Please note that the final ensemble size of BaggingES-OOB-EX is always 50 because the number of bagging iterations is set to 50. Except for the BaggingES-OOB-EX algorithm, we can see that the final ensemble size of the other four ensemble algorithms increases linearly or sublinearly as the size of the model library increases (note that the y-axis is logarithmic). The final ensemble size of BaggingES-OOB, ES, and ES++ grows relatively faster than BaggingES-Simple's ensemble size. One possible reason is that in Bagging-OOB-Simple, the size of the build set (training set excluding the hillclimb set) is relatively small compared to BaggingES-OOB. Theoretically, for BaggingES-OOB, the hillclimb set (out-of-bag sample) has 36.8% unique instances of the training set, and the training set has 63.2% unique instances; however, BaggingES-Simple uses the bootstrap sample for both training and hillclimbing. For this experiment, the hillclimb ratio for BaggingES-Simple is set to 30%, thus its hillclimb set has fewer unique instances than BaggingES-OOB's hillclimb set. Therefore adding more base classifiers to BaggingES-Simple's model library may not necessarily improve the hillclimb performance since the hillclimb set might be too simple and the local hillclimb performance maximum could be achieved quickly.

Another interesting pattern is that ES has a much smaller ensemble size than BaggingES-OOB and BaggingES-Simple have. This could be because the local performance maximum of ES on the hillclimb set can be achieved more quickly compared to bagging ensemble selection. Again, adding more base classifiers to ES's model library may not necessarily improve the hillclimb performance.

Based on those observations, it seems that one reason for the good performance of BaggingES-OOB is that it usually has a larger final ensemble compared to all other algorithms. However, this does not imply that a larger final ensemble always yields better predictive performance. Refer to the learning curves in Figure 4, for data sets Chess-94, KDD-09 and Poker-07: BaggingES-OOB-EX's performance is competitive with BaggingES-OOB even though its final ensemble size is only 50. Therefore, whenever final ensemble size is crucial, for example, when an application requires fast real-time prediction, then the BaggingES-OOB-EX algorithm should be considered.

To sum up, we conclude that the advantage of the BaggingES-OOB algorithm and the BaggingES-OOB-EX algorithm over ES/ES++ is that their ensembles are evaluated on diverse hillclimb sets generated by the bagging procedure, and therefore are more robust and stable.

3.2 Comparison of Bagging Ensemble Selection Algorithms to Other Ensemble Learning Algorithms

In this experiment, we compare BaggingES-OOB (the most successful variant of the bagging ensemble selection based algorithms) to other popular ensemble learning methods. The following algorithms (WEKA [7] implementations) are evaluated: Voting with probability averaging, stacking with linear regression at

Table 2. Mean and standard deviation of the AUC performance of BaggingES-OOB and five other popular ensemble learning methods

Data set	BES-OOB	Voting	Stacking	AdaBst.M1	RandomFrst	ES++
Adult-96	0.905±0.001	0.902±0.002*	0.892±0.004*	0.783±0.008*	0.902±0.002*	0.906±0.002
Chess-94	0.875±0.004	0.859±0.003*	0.841±0.011*	0.971±0.002∘	0.862±0.004*	0.866±0.003*
Connt-4-95	0.918±0.006	0.911±0.006*	0.897±0.007*	0.905±0.005*	0.912±0.006*	0.916±0.005
Covtype-98	0.884±0.002	0.882±0.002*	0.875±0.004*	0.878±0.003*	0.882±0.002*	0.881±0.001*
KDD-09	0.678±0.029	0.678±0.027	0.656±0.031*	0.580±0.011*	0.675±0.029	0.669±0.029
Localiz-10	0.966±0.002	0.957±0.002*	0.940±0.006*	0.938±0.004*	0.960±0.002*	0.963±0.003*
Magic-07	0.920±0.004	0.916±0.004*	0.910±0.004*	0.868±0.005*	0.919±0.004*	0.913±0.002*
MiniB-10	0.964±0.002	0.963±0.002*	0.959±0.002*	0.928±0.006*	0.963±0.002*	0.963±0.001*
Poker-07	0.697±0.018	0.660±0.022*	0.620±0.041*	0.740±0.007∘	0.674±0.018*	0.671±0.020*
UCSD-10	0.649±0.011	0.648±0.008	0.612±0.016*	0.632±0.010*	0.646±0.008	0.646±0.007*
(win/tie/loss)		(0/2/8)	(0/0/10)	(2/0/8)	(0/2/8)	(0/3/7)

"*" BaggingES-OOB is significantly better, "∘" BaggingES-OOB is significantly worse, level of significance 0.05

the meta-level (Stacking), AdaBoostM1, and RandomForest. ES++ is also included for comparison. All ensemble algorithms use the random tree as the base classifier. The total number of base classifiers allowed to be trained for each ensemble algorithm is equal. For bagging ensemble selection the number of bags is set to 50, and the number of base classifiers of individual ensemble selection in each bag is set to 100; thus in total 5,000 base classifiers (random trees) are trained. For other ensemble algorithms, the number of base classifiers is set to 5,000. The training complexity of random tree is $O(nlogn)$, where n is the size of the training set. In this experiment, all ensemble algorithms train on the same number of random trees, therefore the training costs for the model library of each ensemble algorithm in this comparison are roughly the same.

Table 2 shows the performance of each algorithm on the ten data sets. Standard deviations and significant test results were calculated from five independent runs of 66% (training) versus 34% (testing) split validation. The results for which a significant difference with BaggingES-OOB was found, are marked with a "*" or "∘" next to them. An asterisk "*" next to a result indicates that BaggingES-OOB was significantly better than the respective method (column) for the respective data set (row). A circle "∘" next to a result indicates that BaggingES-OOB was significantly worse than the respective method. We can see that AdaBoost.M1 significantly outperforms BaggingES-OOB on the Chess-94 and the Poker-07 data sets. On the other eight data sets, BaggingES-OOB is competitive (7 ties) to or superior (41 significant wins) to all other ensemble algorithms.

4 Conclusions

Ensemble selection is a popular ensemble learning method. Over the past several years, ensemble selection has been empirically examined and has proven to be a very effective and accurate ensemble learning strategy. One disadvantage of ensemble selection is that it is unstable and sometimes overfits the hillclimb set. In this paper, to further improve ensemble selection we proposed using the bagging strategy, which utilises the unstable property, to reduce the variance of a single ensemble selection. Our experiments on ten real world problems show that the bagging ensemble selection, especially BaggingES-OOB, which uses the out-of-bag sample as the hillclimb set, yields a robust and more accurate classifier ensemble than the original ensemble selection.

When the underlying problem requires fast prediction, we suggest using BaggingES-OOB-EX instead, because the user can control the size of the final ensemble. In terms of predictive performance, bagging ensemble selection is also competitive (in many cases, superior) to other state-of-art ensemble learning algorithms, such as voting, random forest, stacking and boosting. Again, bagging ensemble selection is not restricted by the type of base classifiers.

We experimented with only one type of base classifier in this paper, but to get the best out of the algorithm, we suggest using a more diverse model library. The bagging ensemble selection idea can be easily generalised to regression problems, since bagging is applicable to both classification and regression. In future research, we will compare bagging ensemble selection to other ensemble methods for regression problems. The success of the proposed methods on the diverse data sets selected for the study strongly suggests the applicability of the bagging ensemble selection algorithm 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 1(38) (1998)
2. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Application to Data Mining. Springer (2009)
3. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
4. 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)
5. 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)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: An update. SIGKDD Explorations 11(1) (2009)
8. Partalas, I., Tsoumakas, G., Vlahavas, I.: An ensemble uncertainty aware measure for direct hill climbing ensemble pruning. Machine Learning 81(3) (2010)
9. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33, 1–39 (2010)

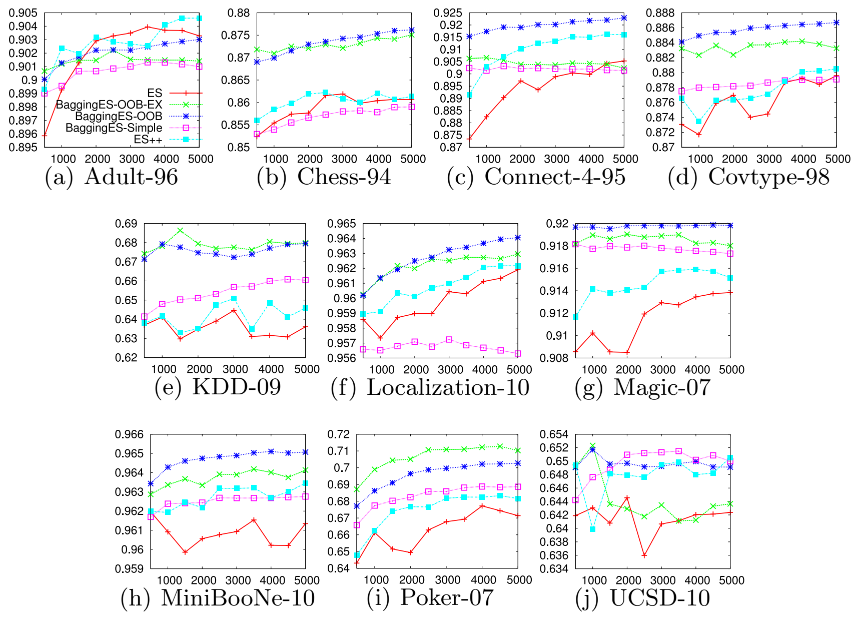


Fig. 4. Learning curves of ES, ES++ and the three bagging ensemble selection algorithms. X-axis is the model library size; y-axis is the AUC performance

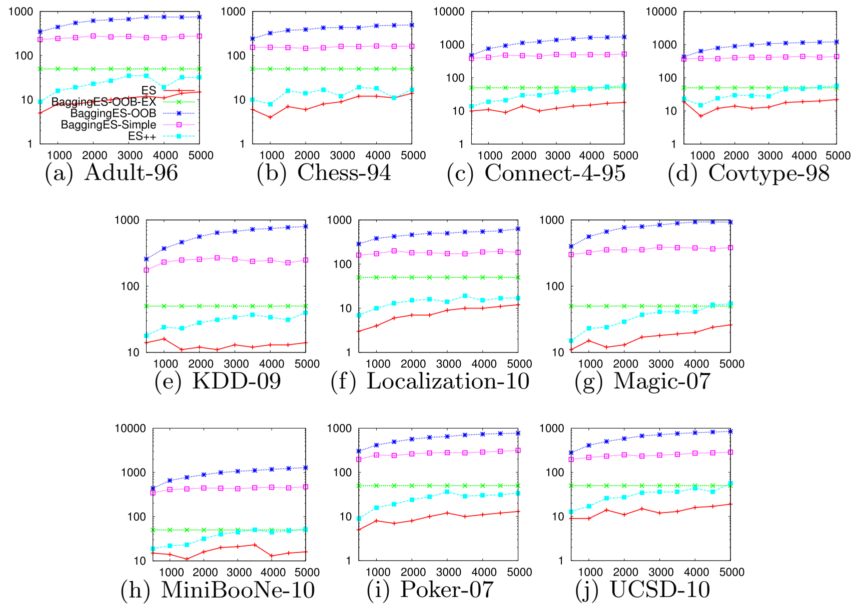


Fig. 5. Final ensemble sizes of ES, ES++ and the three bagging ES based algorithms. X-axis is the model library size; y-axis is the final ensemble size in logarithmic scale