Getting Even More Out of Ensemble Selection

Quan Sun
Department of Computer Science
The University of Waikato
Hamilton, New Zealand
qs12@cs.waikato.ac.nz

ABSTRACT

Ensemble Selection uses forward stepwise selection from a library of models to build a classifier ensemble that is optimised to a given performance metric. The algorithm has received a lot of attention over time due to its excellent predictive performance in many data mining competitions, such as the KDD cup and the Netflix data mining competition. In this paper, we present two methods that were used in our winning solution for the UCSD FICO 2010 data mining contest. Our empirical and theoretical results show that they can also be used for improving Ensemble Selection's performance in general. The proposed methods were implemented using the WEKA machine learning package and evaluated on a variety of real world data sets. The results indicate that, by an appropriate application of each method, Ensemble Selection's predictive performance can be further improved and its ensemble building cost can be significantly reduced.

1. INTRODUCTION

The idea of ensemble learning is to build a predictive model by combining multiple base models. Usually ensemble methods can be used for improving prediction performance. In [9], the problem of classifier combination is considered in the context of two main fusion scenarios: fusion of opinions based on identical and on distinct representations. The author states that "in both cases (identical and distinct representations), the expert fusion involves the computation of a linear or nonlinear function of the a posteriori class probabilities estimated by the individual experts". Therefore, ensemble methods can be seen as methods for calculating optimal weight for each base model in terms of certain goals, such as classification accuracy. For a more detailed review of recent development of ensemble-based classifiers, the reader is referred to [1, 11].

Ensemble Selection is a method for constructing ensembles from a library of base models [4]. The underlying procedure of Ensemble Selection is simple. Firstly, base models are built by using different machine learning algorithms, regardless of the parameter settings. Then, a construction strategy, such as forward stepwise selection, is used to add to the ensemble the models that improve its evaluation score. In [3, 4], the authors showed a set of methods that can be employed to boost Ensemble Selection's performance, including selection with replacement, sorted ensemble initialization, bagged ensemble selection, and probability calibration. In this paper we carry on the research and propose two methods that can be used to improve Ensemble Selection's performance in general. The authors of [4] have given excellent experimental results showing Ensemble Selection can be optimised toward many common evaluation metrics, such as accuracy, root mean squared error, mean cross-entropy, lift, precision/recall, F-score and ROC Area (AUC). Hence, the methods proposed in this paper may be extended to those metrics. For simplicity, we use AUC as our performance metric for all experiments.

The rest of this paper is organized as follows. Information about data sets used in this paper is given in Section 2. In Section 3, we show the "ReTrainFull" strategy that can be used to improve Ensemble Selection's predictive performance. In Section 4, we propose a method for Ensemble Selection that dramatically reduces the computation time spent on the model library construction stage but still keeping Ensemble Selection's good performance. Discussion and future work are given in Section 5.

2. DATA SETS AND EVALUATION SETUP

Experiments in this paper are based on ten data sets. All of them are real world data sets which can be downloaded from the UCI machine learning repository [6], the UCSD FICO contest website[1] and the KDD Cup 2009 website[2]. These data sets were selected because they come from very different research and industrial areas, including social sciences, games, life sciences, physical sciences, marketing and E-commerce. Table 1 shows the basic properties of these data sets.

To simplify and speed up the experiments, all five multiclass data sets were converted to binary problems by keeping only the two largest classes in 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

[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

our experiments. Table 1 (in the rightmost column) shows the properties of the final data sets.

For each individual experiment, 100 WEKA [8] classification algorithms are used for building the model library, including: Naive Bayes, logistic regression with different ridge values, AdaBoostM1 [7] with decision stump, bagging with WEKA's REPTree, random forests [2] with different numbers of random attributes, LibSVM [5] with different kernels, J48 decision trees [10], and random trees with different parameters.

3. ENSEMBLE SELECTION WITH MODELS RETRAINED ON FULL TRAINING SET

The default construction strategy of Ensemble Selection uses the hillclimb set to guide base model weighting. In [3, 4], the authors have mentioned that once the base models to be used in the ensemble are selected, the hillclimb set can be put back in the training set, and then the base model with a nonzero weight can be retrained based on the full training set. Since the authors did not give results for examining if this strategy improves the ensemble performance in general, we therefore continued the research in this direction. In this section, we compare three ensemble strategies:

The **ES-Def** strategy, is the original Ensemble Selection algorithm with the default ensemble construction method (as shown in Step 1 to Step 6, in Figure 1). The **ES-BestModel** strategy is a strategy in which only the best model trained with any of the base classifiers is included in the final ensemble. The **ES-ReTrainFull** strategy employs the original Ensemble Selection algorithm with the base models retrained on the full training set if a base model's weight is greater than zero (Step 1 to Step 8, Figure 1).

In this experiment, reported mean AUC scores and standard deviations were calculated based on 66% (training) versus 34% (testing) split evaluation from five independent runs. To get a clear picture of the result, we used 19 different setups for the hillclimb set; the size of the hillclimb set ranged from 5%, 10%, 15%,..., to 95% of the training set. Figure 2 shows the test set learning curves of the three strategies based on 570 individual experiments (3 algorithms, 10 data sets, 19 different hillclimb set ratios per data set).

From Figure 2, we can see a clear pattern showing that the performance of ES-Def and ES-BestModel is related to the size of the hillclimb set (hillclimb set ratio). For data sets Chess-94, Connect-4-95, Covtype-98, Localization-10 and Poker 07, the performance of ES-Def and ES-BestModel declines as the hillclimb set ratio increases. For data sets KDD-09, Magic-07, MiniBooNe-10 and UCDS-10, as the hillclimb set ratio increases, the performance of ES-Def and ES-BestModel firstly increases and then starts to drop after passing a peak. For all these four data sets, the performance peak occurs when the hillclimb ratio is less than 40%. A small hillclimb set ratio, for example, using 10% or fewer, usually yields good performance, such as in the Chess-94, Connect-4-95, Covtype-98, Localization-10 and Poker-07 data sets.

Also, we can see that in most cases, ES-Def outperforms ES-BestModel. However, for the MiniBooNe-10 data set, the ES-BestModel strategy outperforms the other strategies when hillclimb ratio is less than 15%.

The ES-ReTrainFull strategy clearly outperforms ES-Def and ES-BestModel on Chess-94, Connect-4-95, Covtype-98, Localization-10 and Poker-07. An interesting pattern is that the performance of the ES-ReTrainFull strategy is relatively stable when the hillclimb set ratio is less than 80%. This pattern inspired us to develop methods that can be used to reduce ES-ReTrainFull's training cost, which we will discuss in the next section. Another noticeable result is that, for the Magic-07 data set, the performance of ES-ReTrainFull reaches a peak when the hillclimb set ratio is 65%.

To sum up, we conclude that, when an appropriate hillclimb set ratio is used (for example, between 20% and 40%), the ES-ReTrainFull strategy does improve the original Ensemble Selection algorithm's performance in general.

4. SPEEDING UP ENSEMBLE SELECTION BY BUILDING MODEL LIBRARY ON A SUBSET OF THE BUILD SET

Building the model library for Ensemble Selection sometimes can be highly time consuming for certain machine learning algorithms. In our experiments, for the UCSD-10 data set, a single run could take 20 hours for building the model library on a computer with a 2.8ghz CPU. In the previous section, we have seen that the performance of the ES-ReTrainFull strategy is relatively stable when the hillclimb set ratio is less then 80%. However, we know that as the hillclimb set

```
EnsembleSelectionWithReTrainFull($A, L, F, E, r$)
 $A$ are the algorithms for building the model library
 $L$ is the model library
 $F$ is the training set
 $E$ is the ensemble
 $r$ is the hillclimb set ratio

 1. $T \leftarrow SampleWithoutReplacement(F, r)$ // $T$ is the build set
 2. $H \leftarrow F - T$ // $H$ is the hillclimb set
 3. $E \leftarrow$ empty ensemble
 4. $L \leftarrow BuildLibrary(A, T)$ // build model library on the build set $T$
 5. Assign weight to each base model, such as forward stepwise selection guided
    by performance on the hillclimb set $H$
 6. $E \leftarrow$ the subset of base models with weight greater than zero
 7. $F \leftarrow T \bigcup H$ // $F$ is the "full" training set
 8. for each method $M$ in $E$ {
      if ($M's$ weight $> 0$) {
         Rebuild $M$ on $F$
      } else { $M \leftarrow null$ // remove $M$ from $L$, release the memory }
    }
 Return $E \leftarrow$ the subset of base models with greater than zero weight
```

Figure 1: Pseudocode of the Ensemble Selection algorithm with the ReTrainFull strategy

ratio increases, the actual size of the build set (training instances that are used for building the model library) gets smaller and smaller. Thus, the pattern probably implies that ES-ReTrainFull's performance is not strongly affected by how much data is used for the build set; or at least, it is not as critical as the effects of an increase in the hillclimb ratio were on the original Ensemble Selection algorithm. If this is true, then we could speed up the ES-ReTrainFull strategy by building the base model library on a subset of the "full" build set. In this way, the training time of each base learner for the model library building stage can be dramatically reduced, especially for learners with training complexity that is worse than linear in terms of the runtime cost and size of the training set.

In this section, we attempt to answer this question: If its model library is built on a subset, for example, 40% of the build set, would the ES-ReTrainFull strategy still outperform or be competitive to the original Ensemble Selection algorithm? To answer this question, we decided to compare three different setups for the ES-ReTrainFull strategy. The three setups are: ReTrainFull-100, which is the ES-ReTrainFull strategy used for the experiments in the last section; ReTrainFull-40 and ReTrainFull-60, where only a random 40% and 60% of the build set is used for building the model library, respectively.

Figure 3 shows the test set learning curves of the three different setups for the ES-ReTrainFull strategy, and the original Ensemble Selection algorithm (ES-Def) based on 760 individual experiments (4 algorithms, 10 data sets, 19 different hillclimb set ratios 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. We can see that when the hillclimb set ratio is greater than 50%, ReTrainFull-based strategies clearly outperform ES-Def, except on the MiniBooNe-10 data set, where the performance curves of all algorithms are not very stable. When the hillclimb set ratio is less than 50%, the results are mixed. For the ReTrainFull-based strategies, the general pattern is that the more training data a strategy uses, the better performance it gives. However, on the Chess-94, Localization-10 and Poker-07 data sets, all ReTrainFull-based strategies gave similar performance. Also, on these three data sets, ReTrainFull-based strategies, including the ReTrainFull-40 strategy, clearly outperform ES-Def when the hillclimb set ratio is between 20% and 40%. When the hillclimb set ratio is between 5% and 20%, the performance of all four strategies is very close.

Figure 4 shows the runtime curves of the four strategies on the Adult-96 and the Chess-94 data sets. We use these two figures here as an illustration because they are typical; the patterns are similar for the other eight data sets.

For ES-Def, we can see that as the hillclimb set ratio increases, the training time of ES-Def decreases. This is because the actual size of the build set gets smaller and smaller. Previous results have shown that the performance of ES-Def is strongly affected by the hillclimb ratio. For ReTrainFull-based strategies, we can see that in general ReTrainFull-40 is faster than ReTrainFull-60, and ReTrainFull-60 is faster than ReTrainFull-100. Also, the ReTrainFull-100 strategy always has a greater training cost than the ES-Def strategy, because some base models are retrained for the ReTrainFull strategy. However, an interesting pattern shows that the ReTrainFull-40 and the ReTrainFull-60 strategies are faster than ES-Def only when the hillclimb ratio is less than a certain value. For the model library and the ten data sets we have examined, the ReTrainFull-40 strategy is faster than ES-Def when the hillclimb ratio is less than 40%. Next, we attempt to analyse the training cost of ES-Def and ReTrainFull-based strategies, and to see in which situations, ReTrainFull-based strategies, such as ReTrainFull-40 are faster than ES-Def.

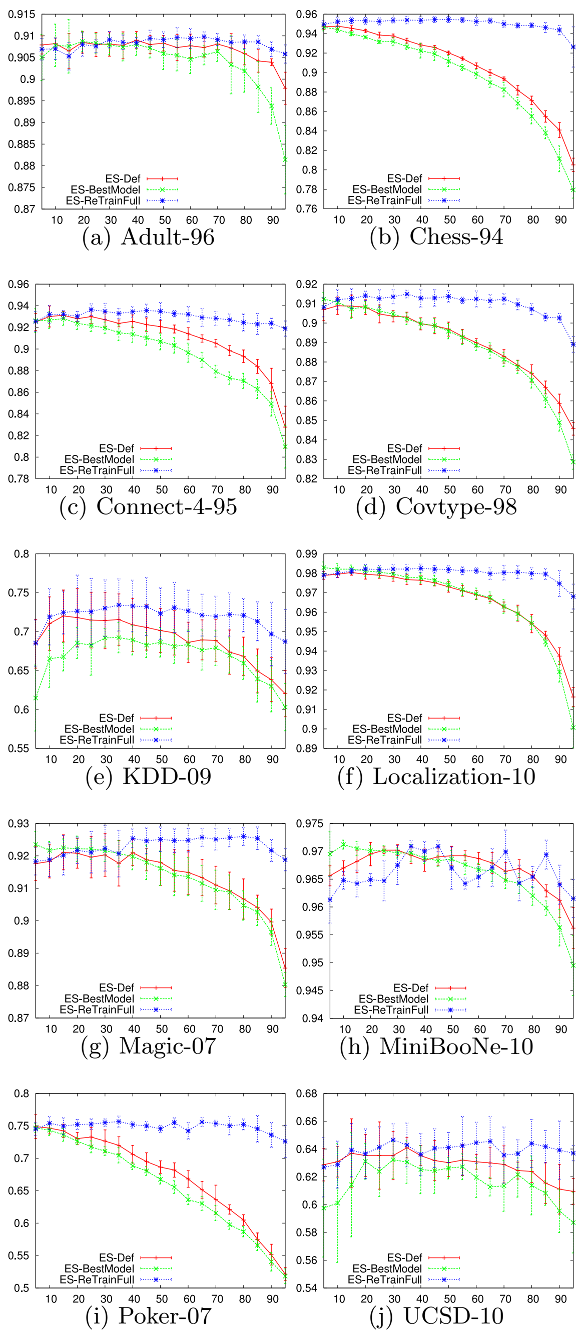


Figure 2: Learning curves of ES-Def, ES-BestModel and ES-ReTrainFull. *X*-axis is the hillclimb set ratio; *y*-axis is the AUC value

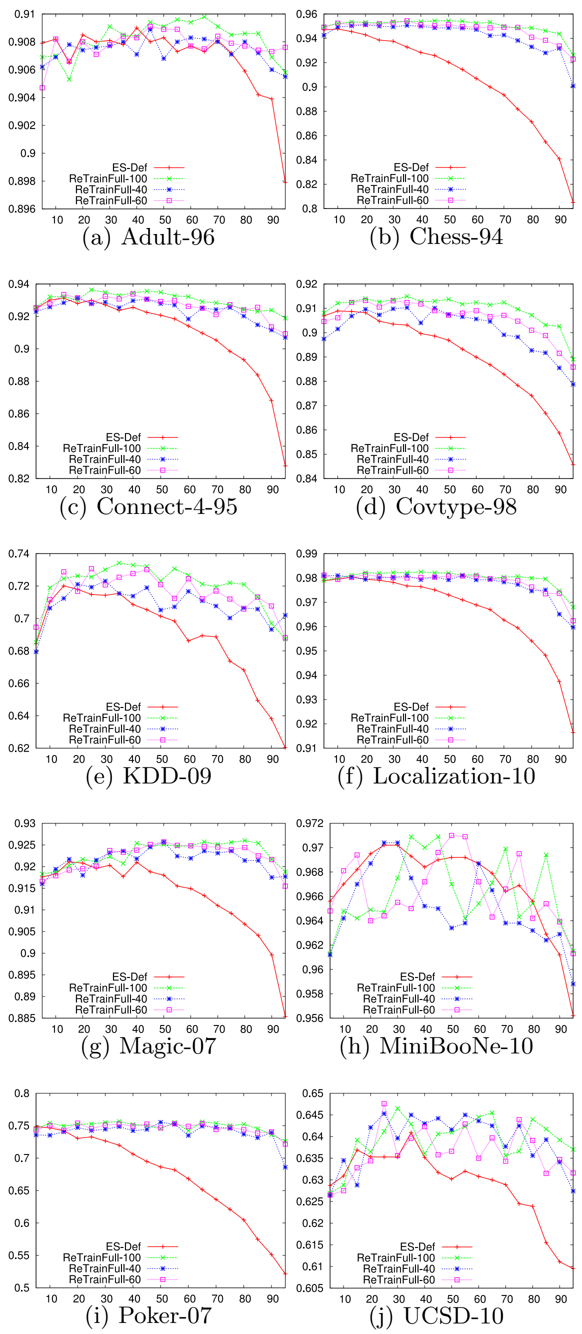
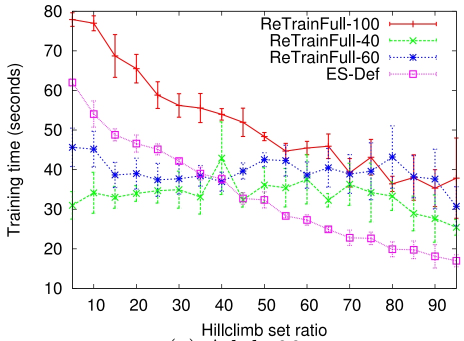
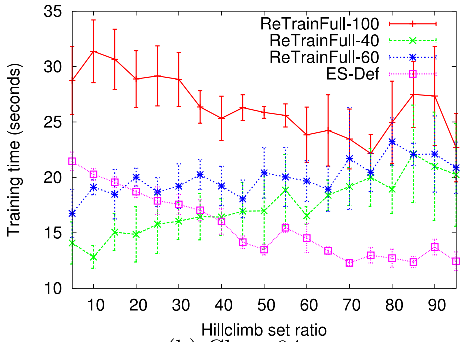


Figure 3: Learning curves of ES-Def and the three different setups of the ES-ReTrainFull strategy. *X*-axis is the hillclimb set ratio; *y*-axis is the AUC value



(a) Adult-96



(b) Chess-94

Figure 4: Two typical runtime curves based on 19 different hillclimb set ratio setups

The main training cost of the Ensemble Selection algorithm (including ES-Def and ReTrainFull-based strategies) consists of two parts:

$$Cost_{ES} = Cost_{BuildModelLibrary} + Cost_{Hillclimbing}. \quad (1)$$

Compared with $Cost_{BuildModelLibrary}$, $Cost_{Hillclimbing}$ is very small, so small that we can safely ignore it for our calculation. Then we have:

$$Cost_{ES} \approx Cost_{BuildModelLibrary}. \quad (2)$$

Assume the training complexity function of $Cost_{BuildModelLibrary}$ (cost for building the model library) is $f(n)$, where n is the number of the build set instances, $n > 0$. Then, the training cost for ES-Def can be expressed as: $Cost_{ES-Def} \approx f(n)$; and the training cost for a ReTrainFull-based strategy can be expressed as: $Cost_{ReTrainFull} \approx f(nk) + f'(n)$, where $0 < k < 1$, is the percentage of the build set that is used for training the model library, and $f'(n)$ is the cost for base model retraining. Also, for simplicity, we assume $f'(n) <= f(n)$.

If we hope ReTrainFull-based strategies, for instance, the ReTrainFull-40 strategy, is faster than ES-Def in terms of training complexity on the same data set, then, we need to find out the situations that satisfy $f(nk) + f'(n) < f(n)$. What those situations are depends on the function form of $f(n)$.

Assume we have a model library with m base models of the same type. When $f(n)$ is linear, then the training cost for ES-Def is mn; the training cost for a ReTrainFull-based strategy is $mkn+zmn$, where again k is the percentage of the build set that is used for training, and z is the percentage of models in the model library that are also in the final ensemble. Therefore, to compare the training cost of ES-Def and ReTrainFull-based strategies, we actually compare mn and $mkn + zmn$. We can set $m = 1$, then, we need to compare only n and $kn + zn$, from which we can see that in the linear case $f(nk) + f'(n) < f(n)$ when $k + z < 1$.

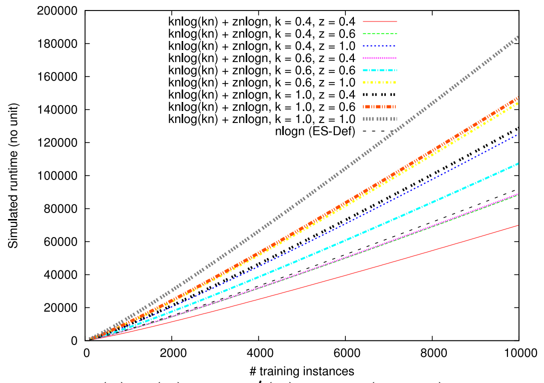
What if $f(n)$ is superlinear? For example, the m base models are all random trees and we know that the training complexity of a random tree is $O(nlogn)$. Thus, in this case, the training cost for ES-Def is $mnlogn$; the training cost for a ReTrainFull-based strategy is $mknlog(kn) + zmnlogn$. Therefore, to compare the training cost of ES-Def and ReTrainFull-based strategies, we actually compare $mnlogn$ and $mknlog(kn) + zmnlogn$. We can again set $m = 1$. Then, we need to compare only $nlogn$ and $knlog(kn) + znlogn$. We skip the derivation here, but we can easily find out that $knlog(kn) + znlogn = (k+z)nlogn + nklogk$. Since k is a percentage less than 1, we have $nklogk < 0$. Then we have $(k + z)nlogn + nklogk < nlogn$, when $k + z \leq 1$. Figure 5 (a) shows a simulation for setting k and z to different values. In the simulation $f(n)$ and $f'(n)$ are assumed to be $O(nlogn)$. We can see that there is one case in which the training cost for ReTrainFull is clearly lower than that for ES-Def: $k = 0.4$ and $z = 0.4$, which satisfies $k + z < 1$. Two cases, where $k = 0.4$, $z = 0.6$ and $k = 0.6$, $z = 0.4$, are slightly lower than ES-Def, which satisfies $k + z = 1$.

Ensemble Selection is not restricted by the type of base models used for the model library. We found that the empirical training cost $f(n)$ of the model library used in our experiments, is between $O(n^{1.2})$ and $O(n^{1.4})$. However, the cost could be much higher when the model library is large, say thousands of models of diverse types. Figure 5 (b) shows a simulation for the case that $f'(n) < f(n)$. In this simulation, $f(n)$ is set to $n^{1.5}$ and $f'(n)$ is set to $n^{1.3}$. The figure shows that when k is less than 60%, the training cost for ReTrainFull is much lower than for ES-Def.

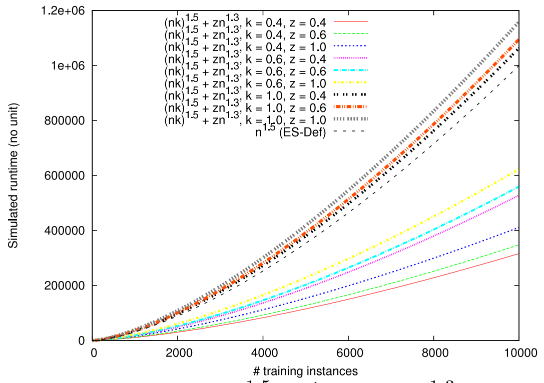
Based on the above empirical and theoretical analysis, we conclude that building the model library on a subset of the build set (we suggest 40%) and then retraining the base model that has a nonzero weight on the full build set, is a procedure that not only reduces the training cost of the original Ensemble Selection algorithm, but also keeps (sometimes improves) Ensemble Selection's predictive performance. The more that training the model library costs in terms of the function form of its training complexity, the greater the performance gain the new strategy, such as the ReTrainFull-40 strategy, will give.

5. FUTURE WORK AND CONCLUSIONS

There are many other aspects of Ensemble Selection that could be investigated: for instance, how can we find out the optimal hillclimb set ratio for a given data set? In future



(a) $f(n)$ and $f'(n)$ are $O(nlogn)$



(b) $f(n)$ is $O(n^{1.5})$, $f'(n)$ is $O(n^{1.3})$

Figure 5: Training complexity simulation for ES-Def and ReTrainFull-based strategies

research, we will try to tackle this problem by extending concepts and techniques of learning curve approximation. The original authors of Ensemble Selection pointed out that the hillclimb set is likely to be overfitted when the base model library is large, say 2000 models [4]. Therefore, finding methods that can be used to avoid overfitting is another interesting direction that we will explore. In addition, we will compare Ensemble Selection to other ensemble algorithms on large data sets.

Ensemble Selection has many good features: it can be optimised to any evaluation metric; its implementation is relatively simple, and Ensemble Selection usually yields good predictive performance on practical problems. In this paper, we presented two methods that can be used to improve Ensemble Selection's performance. Based on the experimental results, we conclude that by an appropriate application of each method, Ensemble Selection's predictive performance can be further improved and its ensemble building cost can be significantly reduced, especially when the training complexity of the model library is worse than linear. In future work, we will continue our research on Ensemble Selection and report our findings in successive papers.

6. REFERENCES

[1] Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Application to Data Mining. Springer (2009)

[2] Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)

[3] 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)

[4] 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)

[5] Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`

[6] Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`

[7] Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning. pp. 148–156 (1996)

[8] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: An update. SIGKDD Explorations 11(1) (2009)

[9] Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)

[10] Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA (1993)

[11] Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. 33, 1–39 (2010)