

Optimal Ensemble Construction via Meta-Evolutionary Ensembles[★]

YongSeog Kim^a, W. Nick Street^b, Filippo Menczer^c

^a*Business Information Systems, Utah State University, Logan, UT 84322, USA*

^b*Management Sciences, University of Iowa, Iowa City, IA 52242, USA*

^c*School of Informatics, Indiana University, Bloomington, IN 47406, USA*

Abstract

In this paper we propose a meta-evolutionary approach to improve on the performance of individual classifiers. In the proposed system, individual classifiers evolve, competing to correctly classify test points, and are given extra rewards for getting difficult points right. Ensembles consisting of multiple classifiers also compete for member classifiers, and are rewarded based on their predictive performance. In this way we aim to build small-sized optimal ensembles rather than form large-sized ensembles of individually-optimized classifiers. Experimental results on 15 data sets suggest that our algorithms can generate ensembles that are more effective than single classifiers and traditional ensemble methods.

Key words: Optimal ensemble, evolutionary ensemble, feature selection, neural networks, diversity of ensemble, ensemble size.

1 Introduction

In recent years, a great deal of interest in the data mining community has been generated by ensemble classifiers. These are predictive models that combine the predictions of a collection of individual classifiers, such as decision trees or artificial neural networks. Popular methods such as Boosting, Bagging and Stacking differ in the ways that individual predictors are constructed, and in

[★] Corresponding author: YongSeog Kim, Business Information Systems, Utah State University, Logan, UT 84322. Tel: 1-435-797-2271, Fax: 1-435-797-2351.

Email addresses: yong.kim@usu.edu (YongSeog Kim),
nick-street@uiowa.edu (W. Nick Street), fil@indiana.edu (Filippo Menczer).

how their votes are combined. However, they have all demonstrated consistent—in some cases, remarkable—improvements in predictive accuracy over individual classifiers.

Much of the power of these methods comes from the diversity of the component classifiers. Intuitively, gathering a collection of problem solvers is only valuable if they are both accurate and diverse in their solutions. For instance, Boosting explicitly rewards a component classifier for correctly predicting difficult points, and is grounded by theoretical results that prove its effectiveness. The necessary diversity can be obtained in many ways, such as using different learning algorithms for the base classifiers, sampling the training examples, or projecting the examples onto different feature subspaces. However, little attention has been paid to the idea of creating an *optimal* collection of classifiers, or indeed, what the idea of “optimality” might even mean in such a context.

We propose to directly optimize ensembles by creating an two-level evolutionary environment. The various ensembles in this environment compete directly with one another, being judged on their estimated predictive performance. In addition, the underlying classifiers also compete with each other, being rewarded for correctly predicting the training examples. This reward is greater if the point in question is difficult, i.e., if it has been incorrectly classified by most of the other classifiers in the ensemble. We use feature selection as the mechanism for individual diversity.

In this paper, we demonstrate the feasibility of such a model and show that the predictive accuracy obtained is better than a single classifier. Our model not only maintains higher or comparable predictive accuracy, but also builds ensembles smaller than traditional ensemble methods. In particular, our model provides the framework to answer how ensembles can best be constructed through the evolutionary process. Finally, we examine the relationship between ensemble characteristics, such as classifier diversity and ensemble size, and the predictive accuracy of the ensemble.

The remainder of this paper is organized as follows. In Section 2 we review ensemble methods and ensemble feature selection algorithms. In Section 3 we present our bi-level approach to the ensemble feature construction, Meta-Evolutionary Ensembles (MEE) in detail. Section 4 presents and analyzes our experimental results. Section 5 addresses the directions of future research and concludes the paper.

2 Ensemble methods and feature selection

2.1 Ensemble methods

Recently many researchers have combined the predictions of multiple classifiers to produce a better classifier, an ensemble, and often reported improved performance [1–3]. Bagging [4] and Boosting [5,6] are the most popular methods for creating accurate ensembles. Bagging is a bootstrap ensemble method that trains each classifier on a randomly drawn training set. Each classifier’s training set consists of the same number of examples randomly drawn from the original training set, with the probability of drawing any given example being equal. Samples are drawn with replacement, so that some examples may be selected multiple times while others may not be selected at all. As a result, each classifier could return a higher test set error than a classifier using all of the data. However, when these classifiers are combined (typically by voting), the resulting ensemble produces lower test set error than a single classifier. The diversity among individual classifiers compensates for the increase in error rate of any individual classifier and improves prediction performance.

Boosting [5] produces a series of classifiers, with each training set based on the performance of the previous classifiers. New classifiers are constructed to better predict examples for which the current ensemble’s performance is poor. This is accomplished using adaptive resampling, i.e., examples that are incorrectly predicted by previous classifiers are sampled more frequently, or alternately given a higher cost of misclassification. Boosting can be implemented in two different ways, Arcing [7] and AdaBoosting [5]. In Arcing, the classifiers’ votes are weighted equally, while AdaBoost weights the predictions based on the classifiers’ training error.

The effectiveness of Bagging and Boosting can be explained based on the bias-variance decomposition of classification error [2]. Bagging and Boosting are known to reduce errors by reducing the variance term [7]. According to [5], Boosting also reduces errors in the bias term by focusing on the misclassified examples. It is noted that Boosting’s effectiveness depends more on the data set than on the component learning algorithms, and it is often more accurate than Bagging. However, Boosting, unlike Bagging, can create ensembles that are much less accurate than a single classifier. In particular, Bagging performs much better than Boosting on noisy data sets because Boosting can easily overfit data by focusing more on the misclassified examples [8]. In most cases, the improved performance of an ensemble is largely obtained by combining the first few classifiers [9].

Note, however, that ensemble models are more complex for human to under-

stand. Ensemble models are also more expensive in terms of computing times and require more memory than individual classifiers.

2.2 Feature subset selection

Feature selection is defined as the process of choosing a subset of the original predictive variables by eliminating redundant and uninformative ones. In many cases this can reduce overfitting and lead to better generalization. Most feature selection research has focused on heuristic search approaches, such as sequential search [10], nonlinear optimization [11], and genetic algorithms [12].

Our approach is based on the wrapper model [13] of feature selection, which requires two components: a search algorithm that explores the combinatorial space of feature subsets, and one or more criterion functions that evaluate the quality of each subset based directly on the predictive model. In this work, we use artificial neural networks (ANNs) as an induction algorithm to evaluate the quality of the selected feature subsets. As a search algorithm, we turn to evolutionary algorithms (EAs) to intelligently search the space of possible feature subsets. An EA is a parallel and global search algorithm that works with a population of solutions to simultaneously evaluate many points in the search space. Standard EAs often converge prematurely to local optima and employ computationally expensive global selection mechanisms. We instead use a new evolutionary algorithm that maintains diversity by employing a *local* selection scheme. This evolutionary local selection algorithm (ELSA) has been successfully applied to multi-objective optimization problems, such as feature selection in both supervised and unsupervised learning [14–16].

We employ feature selection not only to increase the prediction accuracy of an individual classifier but also to promote diversity among component classifiers in an ensemble [17]. The diversity among component classifiers of ensemble has been proven critical to attaining higher generalization accuracy [18–20]. An ensemble generalizes well by combining many accurate component classifiers that make errors on different parts of data. Ensemble feature selection is based on the notion that different feature subsets among component classifiers of an ensemble can provide the necessary diversity. It is similar to the notion that different training samples among component classifiers provide the necessary diversity in ordinary ensemble methods.

2.3 Ensemble feature selection algorithms

The improved performance of ordinary ensemble methods comes primarily from the diversity caused by re-sampling training examples. However, ensem-

ble methods typically use the complete set of features to train component classifiers. Further, ensemble construction based on re-sampling is not recommended when the size of data sets is large and data examples are relatively homogeneous. This is mainly because re-sampling of homogeneous records may not boost the diversity among classifiers and the large size of re-sampling may take most of precious resources.

Recently several attempts have been made to incorporate the diversity in feature dimension into ensemble methods. The Random Subspace Method (RSM) in [21,22] was one early algorithm that constructed an ensemble by varying the feature subset. RSM used C4.5 as a base classifier and randomly chose half of the original features to build each classifier. Each classifier tree was constructed after all the training examples were projected to the subspace of selected features. The predictions were combined by simple majority voting. In comparative experiments, RSM demonstrated better performance on four public data sets than a single tree classifier with all the features and examples, and also outperformed Bagging and Boosting on the full-dimensional data sets [21].

A more sophisticated way to select a subset of features for ensembles was proposed in [23]. They used a genetic algorithm (GA) to explore the space of all possible feature subsets. Their experiments paired four different ensemble methods, including Bagging and AdaBoost, with three different feature selection algorithms: complete, random, and genetic search. Using two table-based classification methods, ensembles constructed using features selected by the GA showed the best performance, followed by RSM. In [24], a new entropy measure of the outputs of the component classifiers was used to explicitly measure the ensemble diversity and to produce good feature subsets for ensemble using hill-climbing search.

Genetic Ensemble Feature Selection (GEFS) [17] also used a GA to search for possible feature subsets. Component classifiers (ANNs) in GEFS were explicitly evaluated in terms of both generalization accuracy and diversity. GEFS starts with an initial population of classifiers built using up to $2 \cdot D$ features, where D is the complete feature dimension. Using a variable feature subset size promotes diversity among the classifiers and allows some features to be selected more than once. Crossover and mutation operators search for new feature subsets, and new candidate classifiers are built for each of the new feature sets. Finally, GEFS prunes the population to the 100 most-fit members and majority voting is applied to determine the ensemble prediction. GEFS produces a good initial population, and in most cases produces better results the longer it runs. GEFS reported better estimated generalization than Bagging and AdaBoost on about two-thirds of 21 data sets tested.

However, longer chromosomes that can consider up to $2 \cdot D$ features make

GEFS computationally very expensive in terms of memory usage [23]. Further, GEFS evaluates each classifier after combining two objectives in a subjective manner using $fitness = accuracy + \lambda diversity$, where *diversity* is the average difference between the prediction of component classifiers and the ensemble. Since there is no obvious way to set the value of λ , GEFS dynamically adjusts the parameter based on the discrete derivatives of the ensemble error, the average population error and the average diversity within the ensemble.

Although all these methods reported improved performance using feature selection for ensemble construction ensemble, they have one common limitation in methodology: only one ensemble is considered. In this paper, we propose a new algorithm for ensemble feature selection, Meta-Evolutionary Ensembles (MEE), that considers multiple ensembles simultaneously and allows each component classifiers to move into the best-fit ensemble. Genetic operators change the ensemble membership of the individual classifiers, allowing the size and membership of the ensembles to change over time. By having the various ensembles compete for limited resources, we can optimize their predictive performance.

In order to avoid costly global selection (i.e., selection of agents for next generation by sorting and comparing all evaluated solutions based on accuracy values) common to most GAs, we use a local selection mechanism in which classifiers compete with each other only if they belong to the same ensemble. Using ANNs as the base classifier and EA for feature selection, we evaluate and reward each classifier based on two different criteria, accuracy and diversity. A classifier that correctly predicts data examples that other classifiers in the same ensemble misclassify contributes more to the accuracy of the ensemble to which it belongs. We imagine that some limited “energy” is evenly distributed among the examples in the data set. Each classifier is rewarded with some portion of the energy if it correctly predicts an example. The more classifiers that correctly classify a specific example, the less energy is rewarded to each, encouraging them to correctly predict the more difficult examples. The predictive accuracy of each ensemble determines the total amount of energy to be replenished at each generation. Finally, we select the ensemble with the highest accuracy as our final classification model.

3 Meta-Evolutionary Ensembles

3.1 Theoretical motivation

In this section, we formulate optimal ensemble construction as an optimization problem, and motivate why we take a gradient approach based on a generic

algorithm. For notational simplicity, we take a two-class classification problem, where class the label for a data point i , y_i for all $i \in \{1, 2, \dots, N\}$, is either 0 or 1. Note that our discussion can be easily extended for multiple-class classification cases without modifying the main structure. Let's also assume that there is an ensemble e that consists of K individual classifiers, C_j , where $j \in \{1, 2, \dots, K\}$. Then, we can represent the predicted class label for a data point i by a classifier C_j , \bar{y}_i^j , as follows:

$$\bar{y}_i^j = \begin{cases} 1 & \text{if classifier } C_j \text{ predicts data } i \text{ to be class 1} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Since the predicted class label for a data point i by an ensemble e , \hat{y}_i^e , is a weighted sum of \bar{y}_i^j , we represent it as follows:

$$\hat{y}_i^e = \begin{cases} 1 & \text{if } \sum_{j=1}^K w_j \cdot \bar{y}_i^j > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where w_j represents the weight allocated to classifier C_j , and can have the same value for all classifiers as in Bagging or be dependent on \bar{y}_i^j .

Optimal ensemble construction problem is to find an ensemble with the lowest classification error out of m ensembles, and can be formulated as follows:

Find an ensemble e such that $error^e \leq error^k$

where

$$\begin{aligned} error^e &= \sum_{i=1}^N (\hat{y}_i^e - y_i)^2, \quad e \in \{1, 2, \dots, m\} \\ error^k &= \sum_{i=1}^N (\hat{y}_i^k - y_i)^2, \quad \forall k \neq e \in \{1, 2, \dots, m\} \end{aligned} \quad (3)$$

Note that each ensemble can consist of a different number of classifiers. Though the optimal ensemble construction problem can be formulated as in Equation 3, it is a difficult task to find the global solution by exploring the search space exhaustively. For example, when we build a classifier based on a subset of features out of D features as in our approach, there are 2^D different ways of building a classifier. Since an ensemble can consist of any number of classifiers, there are 2^{2^D} different ways of building an ensemble. Therefore, if we consider m ensemble, the total number of candidate solutions we should evaluate through exhaustive algorithms will be $m \cdot 2^{2^D}$. Even with a small number of features, this exponential search space cannot be searched exhaustively.

Therefore, we take a gradient approach to search solutions space efficiently, and genetic algorithms have been known very successful for exploring large search space.

3.2 Algorithm detail

Pseudocode for the Meta-Evolutionary Ensembles (MEE) algorithm is shown in Figure 1, and a graphical depiction of the energy allocation scheme is shown in Figure 2. Each agent (candidate classifier) in the population is first initialized with randomly selected features, a random ensemble assignment, and an initial reservoir of energy. The representation of an agent consists of $D + \log_2(G)$ bits. D bits correspond to the selected features (1 if a feature is selected, 0 otherwise). The remaining bits are a binary representation of the ensemble index, where G is the maximum number of ensembles. Mutation and crossover operators are used to explore the search space. A mutation operator randomly selects one bit of an agent and mutates it. Our crossover operator takes two agents, a parent a and a random mate, and scans through the bits of the two agents. If a difference is found, the value of the bit in a is flipped with a probability of 0.5. In this process, the mate contributes only to construct the offspring’s bit string, which inherits all the common features of the parents.

In each iteration of the algorithm, an agent explores a candidate solution (classifier) similar to itself, obtained via crossover and mutation. The agent’s bit string is parsed to get a feature subset J . An ANN is then trained on the projection of the data set onto J , and returns the predicted class labels for the test examples. The agent collects ΔE from each example it correctly classifies, and is taxed once with E_{cost} . The net energy intake of an agent is determined by its fitness. This is a function of how well the candidate solution performs with respect to the classification task. But the energy also depends on the state of the environment. We have an energy source for each ensemble, divided into bins corresponding to each data point. For ensemble g and record index r in the test data, the environment keeps track of energy $E_{envt}^{g,r}$ and the number of agents in ensemble g , $count_{g,r}$ that correctly predict record r . The energy received by an agent for each correctly classified record r is given by

$$\Delta E = \frac{E_{envt}^{g,r}}{\min(5, prevCount_{g,r})}. \quad (4)$$

An agent receives greater reward for correctly predicting an example that most in its ensemble get wrong. The min function ensures that for a given point there is enough energy to reward at least 5 agents in the new generation. Candidate solutions receive energy only inasmuch as the environment has sufficient resources; if these are depleted, no benefits are available until

```

initialize population of agents, each with energy  $\theta/2$ 
while there are alive agents in  $Pop^i$  and  $i < T$ 
  for each ensemble  $g$ 
    for each record  $r$  in  $Data_{test}$ 
       $prevCount_{g,r} = count_{g,r}$ 
       $count_{g,r} = 0$ 
    endfor
  endfor
  for each agent  $a$  in  $Pop^i$ 
     $a' = mutate(crossover(a, randomMate))$ 
     $g = group(a)$ 
    train( $a$ )
    for each record  $r$  in  $Data_{test}$ 
      if ( $class(r) == prediction(r, a)$ )
         $count_{g,r} ++$ 
         $\Delta E = E_{envt}^{g,r} / \min(5, prevCount_{g,r})$ 
         $E_{envt}^{g,r} = E_{envt}^{g,r} - \Delta E$ 
         $E_a = E_a + \Delta E$ 
      endif
    endfor
     $E_a = E_a - E_{cost}$ 
    if ( $E_a > \theta$ )
      insert  $a, a'$  into  $Pop^{i+1}$ 
       $E_{a'} = E_a / 2$ 
       $E_a = E_a - E_{a'}$ 
    else if ( $E_a > 0$ )
      insert  $a$  into  $Pop^{i+1}$ 
    endif
  endfor
  for each ensemble  $g$ 
    replenish energy based on predictive accuracy
  endfor
   $i = i + 1$ 
endwhile

```

Fig. 1. Pseudo-code of Meta-Evolutionary Ensembles (MEE) algorithm. In each iteration, the environmental energy for each pair of an ensemble g and a test example r is replenished based on the predictive accuracy of g . The main loop calls agents in random order and agents are rewarded based on their accuracy on each test record r , normalized by the number of other agents that correctly classify r in the same ensemble.

the environmental resources are replenished. Thus an agent is rewarded with energy for its high fitness values, but also has an interest in finding unpopulated niches, where more energy is available. The result is a natural bias toward diverse solutions in the population. E_{cost} for any action is a constant ($E_{cost} < \theta$).

In the selection part of the algorithm, an agent compares its current energy level with a constant reproduction threshold θ . If its energy is higher than θ , the agent reproduces: the agent and its mutated clone become part of the new population, with the offspring receiving half of its parent's energy. If the energy level of an agent is positive but lower than θ , only that agent joins the new population.

The environment for each ensemble is replenished with energy based on its pre-

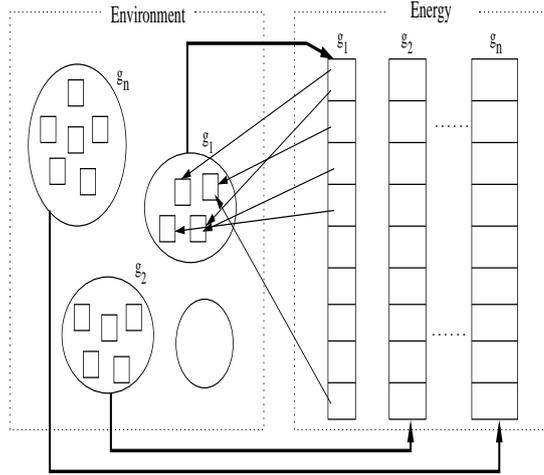


Fig. 2. Graphical depiction of energy allocation in the MEE algorithm. Individual classifiers (small boxes in the environment) receive energy by correctly classifying test points. Energy for each ensemble is replenished between generations based on the accuracy of the ensemble. Ensembles with higher accuracy have their energy bins replenished with more energy per classifier, as indicated by the varying widths of the bins.

dictive accuracy, as determined by majority voting with equal weight among base classifiers. We sort the ensembles in ascending order of estimated accuracy and apportion energy in linear proportion to that ranking, so that the most accurate ensemble is replenished with the greatest amount of energy per base classifier. Since the total amount of energy replenished also depends on the number of agents in each ensemble, it is possible that an ensemble with lower accuracy can be replenished with more energy in total than an ensemble with higher accuracy.

4 Experimental results

4.1 Data sets

We test the performance of MEE combined with neural networks on several data sets that are publically available [25] and were used in [17]. We show the characteristics of our data sets in Table 1.

In our experiments, the weights and biases of the neural networks are initialized randomly between 0.5 and -0.5, and the number of hidden node is determined heuristically as \sqrt{inputs} . For example, the number of hidden nodes of models for both “credita” and “creditg” is set to seven. This way the structure of ANNs is dynamically adjusted depending on the number of input nodes to

Table 1
 Summary of the data sets used in the computational experiments.

Dataset	Records	Classes	Features		Neural Network
			Cont.	Disc.	Inputs
credita	690	2	6	9	47
creditg	1000	2	7	13	63
diabetes	768	2	8	-	8
cleveland	303	2	8	5	13
hepatitis	155	2	6	13	32
votes-84	435	2	-	16	16
ionosphere	351	2	34	-	34
krvskp	3196	2	-	36	40
labor	57	2	8	8	29
sick	3772	2	6	21	31
sonar	208	2	60	-	60
iris	150	3	4	-	4
hypo	3772	4	6	21	31
segment	2310	7	19	-	19
soybean	683	19	-	35	84

reduce computational burden. The other parameters for the neural networks include a learning rate of 0.1 and a momentum rate of 0.9. The number of training epochs was kept small (50) for computational reasons. The values for the various parameters are: $\Pr(\textit{mutation}) = 1.0$, $\Pr(\textit{crossover}) = 0.8$, $E_{cost} = 0.2$, $\theta = 0.3$, and $T = 30$. The value of $E_{envt}^{tot} = 30$ is chosen to maintain a population size around 100 classifier agents.

All computational results for MEE are based on the performance of the best ensemble and are averaged over five standard 10-fold cross-validation experiments. For each 10-fold cross-validation the original data set is first partitioned into 10 equal-sized sets, each maintaining the original class distribution. Each set is in turn used as an evaluation set while the classification system is trained on the other sets. Within the training algorithm, each ANN is trained on two-thirds of the training set and tested on the remaining third for energy allocation purposes.

4.2 Predictive accuracy and data characteristics

Experimental results of various models are summarized in Table 2. We present the performance of a single neural network using the complete set of features as a baseline algorithm. In the win-loss-tie results shown at the bottom, a comparison is considered a tie if the intervals defined by one standard error of the mean overlap. In our experiments, standard error is computed as standard deviation / \sqrt{iter} where $iter = 5$. We also include the results of Bagging, AdaBoost, and GEFS from [17] for *indirect* comparison. In these comparisons, we do not have access to the accuracy results of the individual runs. Therefore, a tie is conservatively defined as a test in which the one-standard-deviation interval of our test contained the point estimate of accuracy from [17].

In terms of predictive accuracy, our algorithm demonstrates superior performance compared to single neural networks with the complete set of features. As shown in the win-loss-tie summary, MEE shows significantly superior performance to single neural networks in 12 data sets and slightly better performance in the other data sets: diabetes, votes-84, and hypo. Compared to the traditional ensembles (Bagging and Boosting), MEE also shows superior performance. In comparison to Bagging, MEE demonstrates significantly better performance in five data sets but only marginally in the remaining data sets. Compared to Boosting, MEE performs significantly better in eight data sets but shows worse performance in four data sets. It is interesting that MEE performs worse performance in two data sets, labor and segment, compared to both ordinary ensemble methods.

Note also that MEE shows comparable performance compared to GEFS with a win-loss-tie score (4-5-6). However, we note that such comparisons are inevitably inexact, since subtle methodological differences can cause variations in estimated accuracy. For example, it is possible that the more complex structure of neural networks used in GEFS can learn more difficult patterns in data sets. We do not have implementation details enough to replicate the same results of GEFS. It is possible the training epochs in GEFS were empirically determined for each data set to optimize performance, while we minimized such an effort.

In addition to predictive accuracy, computational time is another popular measurement used to compare different algorithms. From the perspective of computational time, our MEE algorithm can be very slow compared to Bagging and Boosting. However, MEE can be very fast compared to GEFS because GEFS uses twice as many as input features as used in MEE. In addition, the larger number of hidden nodes and longer training epochs can make GEFS extremely slow.

Table 2
 Experimental results of MEE/ANN with a varying number of epochs

Dataset	Single Net		Bagging	AdaBoost	GEFS	MEE		
	Avg.	S.D.				Avg.	S.D.	Epochs
credita	84.3	0.30	86.2	84.3	86.8	86.4	0.52	40
creditg	71.7	0.43	75.8	74.7	75.2	75.6	0.78	50
diabetes	76.4	0.93	77.2	76.7	77.0	76.8	0.42	50
cleveland	80.7	1.83	83.0	78.9	83.9	83.3	1.54	50
hepatitis	81.5	0.21	82.2	80.3	83.3	84.9	0.65	40
votes-84	95.9	0.41	95.9	94.7	95.6	96.1	0.44	40
ionosphere	89.3	0.85	90.8	91.7	94.6	93.5	0.81	100
krvskp	98.8	0.63	99.2	99.7	99.3	99.3	0.10	50
labor	91.6	2.29	95.8	96.8	96.5	94.4	0.78	50
sick	95.2	0.47	94.3	95.5	96.5	99.3	0.03	50
sonar	80.5	2.03	83.2	87.0	82.2	85.2	1.57	100
iris	95.9	1.10	96.0	96.1	96.7	96.5	0.73	100
hypo	93.8	0.09	93.8	93.8	94.1	93.9	0.06	50
segment	92.3	0.97	94.6	96.7	96.4	93.2	0.28	50
soybean	92.0	0.92	93.1	93.7	94.1	93.8	0.19	50
Win-loss-tie	12-0-3		5-2-8	8-4-3	4-5-6			

In summary, in terms of predictive accuracy, MEE and GEFS is the best followed by traditional ensemble methods, and single neural network is the worst. However, single neural network classifier is the fastest followed by traditional ensemble methods and MEE, and GEFS is the slowest.

We also try to profile data sets that MEE relatively works better or worse to provide data analysts with guidelines on how to build construct ensembles, using either re-sampling records or selecting features. We first investigate if MEE performs worse on multi-class data sets. In general, if there are multiple concepts to learn, classifiers (either single or ensemble models) need a sufficient number of data points with detailed information from most of input features to learn multiple patterns. Therefore, classifiers with information from few projected variables will not perform well. Note that, among 15 data sets, there are four multi-class data sets (iris, hypo, segment, and soybean) while the remaining 11 data sets are bi-class data sets. Out of four multi-class data sets, MEE shows consistently worse performance on “segment” data compared to re-sampling based ensemble methods and GEFS, although it shows comparable performance on the other three data sets. Therefore, from the current study, we do not find any strong link for the relationship between MEE’s performance and the number of classes.

We also assume that MEE may not work well on the data sets with few input variables because there is no room to boost the diversity among classifiers when they are built on different sets of feature spaces. Table 2 confirms our conjecture by showing the trends that MEE performs worse on data sets with few input variables such as “iris”, “diabetes”, “segment”, and “labor”. However, it warrants further investigation because of few exceptions on other data sets.

4.3 Guidelines toward optimized ensemble construction

4.3.1 Ensemble size and predictive accuracy

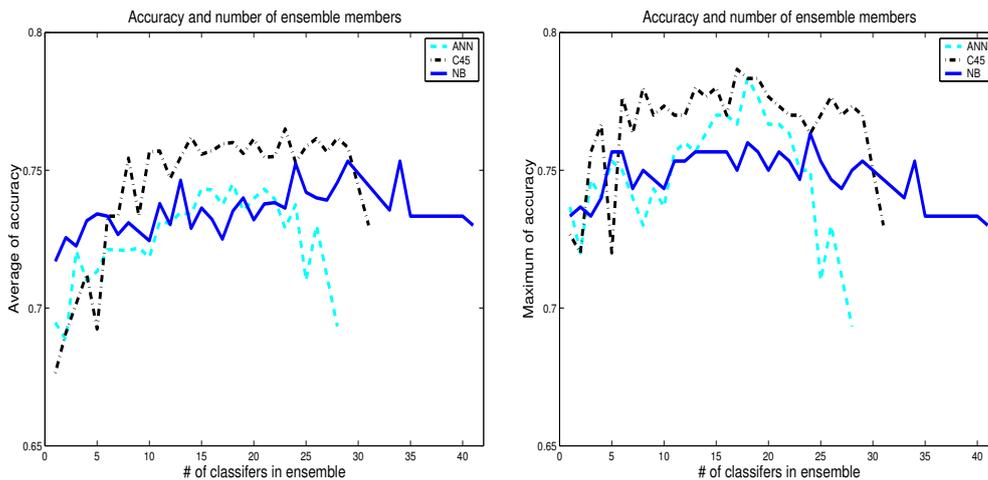
In this section, we use MEE to examine ensemble characteristics and provide data analysts with practical guidelines on how to construct optimal ensembles. We expect that by optimizing the process of ensemble construction, MEE will in general achieve superior or at least comparable accuracy to other methods using fewer individual classifiers. Note that we show the findings from only one data set, credit-g, due to the similar findings on few other data sets and the limitation of spaces. In particular, we use data collected from the first fold of the first cross-validation routine for the following analyses. Two additional classifiers, a Naive Bayesian [26] and C4.5 (a decision tree algorithm) [27] were adopted to study if different classifiers need different configurations for

building optimized ensembles in terms of ensemble size and diversity.

We first investigate whether the ensemble size is positively related with the predictive accuracy. It has been well-established that to a certain degree, the predictive accuracy of an ensemble improves as more classifiers built on different sets of records are included in the ensemble. It is our objective to investigate whether the same positive relationship exists when the classifiers are built on different sets of input feature spaces.

The Figure 3 shows relationship diagrams between the predictive accuracy and the size of ensemble for three different classifier. Note that the ensemble size is measured by the number of classifiers that belong to the ensemble. Two different accuracy measurements, average and maximum accuracy, are used. The average accuracy and maximum accuracy for a given ensemble size is computed by taking an average and the maximum value of accuracy values of all ensembles with the same size.

The Figure 3(a) shows a steady improvement of average accuracy of decision tree ensembles up to an ensemble size of 10 and the improvements flatten at an ensemble size of approximately 10-25, seeming to confirm the results in [9]. However, in contrast to the findings in [9], we found a negative relationship between two factors when ensemble consists of 25 or more classifiers. We partly attribute this finding to the fact that our algorithm in its nature is a gradient search algorithm and its results are dependent on initial populations and stochastic properties. In particular, we believe that ensembles consisting of 25 or more classifiers is not fully explored as other small-sized ensembles. This is confirmed that more than 90% of decision tree ensembles explored consist of 25 or less classifiers. Similar patterns are observed in Figure 3(b) for maximum accuracy of decision tree ensembles.



(a) Avg. accuracy and size

(b) Max. accuracy and size

Fig. 3. Relationship diagrams between ensemble size and accuracy

In contrast, ensembles of Naive Bayesian classifiers shows robust performance over various ensemble sizes. Note that predictive models of Naive Bayesian are more likely to be dependent on class distributions of data sets rather than on marginal variations of feature spaces compared to other classifiers. For example, predictive models of decision trees are very different depending on which input variables are chosen at the root and successive nodes of the tree. However, as shown in Figure 3(a) and (b), Naive Bayesian ensembles show robust performance.

The average and maximum accuracy of neural network ensembles show a steady increase (ensemble size is < 10), flattens out (ensemble size is between 10 and 20), and finally sharply decreases (ensemble size is > 20). In particular, the maximum accuracy peaks at an ensemble size of 18 and sharply decreases as more than 18 neural networks consist of ensemble.

Overall, ensembles of three different classifiers share the same positive relationship between ensemble size and accuracy. Further, decision tree and neural networks ensembles also show the negative relationship between these two factors. Note that it is not our main goal to compare predictive accuracy of three ensemble models and hence no attempts to compare three models (e.g., decision tree ensembles are more accurate than neural networks ensembles) are made.

4.3.2 Ensemble diversity and predictive accuracy

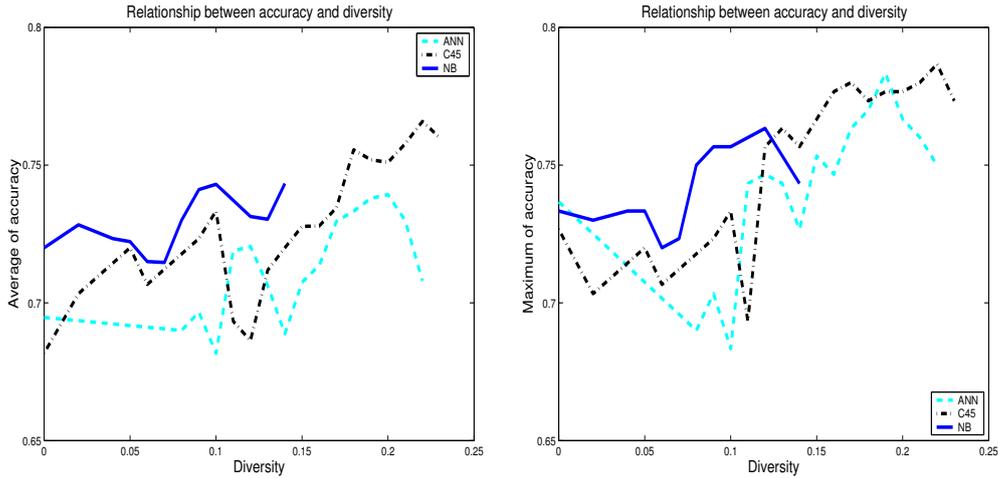
We also investigate whether the diversity among classifiers is positively related with the ensemble’s classification performance. In our experiment, the diversity of ensemble is measured based on the difference of predicted class between each classifier and the ensemble. We first define a new operator \oplus as follows:

$$\alpha \oplus \beta = \begin{cases} 0 & \text{if } \alpha = \beta \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

When an ensemble e consists of g classifiers, the diversity of ensemble e , $diversity^e$, is defined as follows:

$$diversity^e = \frac{\sum_{i=1}^K \sum_{j=1}^{N_2} (\bar{y}_j^i \oplus \hat{y}_j^e)}{K \cdot N_2} \quad (6)$$

where N_2 is the number of records in the test data and \bar{y}_j^i and \hat{y}_j^e represent the predicted class label for record j by classifier i and ensemble e respectively.



(a) Avg. accuracy and diversity (b) Max. accuracy and diversity

Fig. 4. Relationship diagrams between ensemble diversity and accuracy

The larger the value of $diversity^e$, the more diverse the ensemble is.

We show the relationship between the predictive accuracy and ensemble diversity in Figure 4(a) and (b). These two diagrams show the expected positive relationship between accuracy and diversity for ensembles. From homogeneous ensembles with $diversity^e < 0.1$, it is not easy to draw a conclusion about the positive relationship between ensemble diversity and accuracy. This is in particular true to ensembles of neural networks and Naive Bayesian models. However, the performance of ensembles with $diversity^e > 0.1$ improves as they become more diverse. Ensembles of Naive Bayesian models show relatively stable performance over various values of diversity although ensembles with diverse classifiers show slightly superior performance. Note also that ensembles of Naive Bayesian models show relatively uniform diversity compared to ensembles of decision trees and neural networks. We attribute this finding to the fact that Naive Bayesian models are heavily dependent on the overall distributions of records and hence individual Naive Bayesian models generate relatively uniform predictions as long as the distributions of records are not drastically different.

However, our results do not provide sufficient information to determine whether too much diversity among classifiers can deteriorate the performance of ensemble models. Too much diversity can negatively affect the ensemble performance as the final decision made by ensemble becomes a random guess. Ensembles of neural networks show a sudden drop of predictive accuracy after a certain point of diversity ($diversity^e > 0.2$). Decision tree ensembles also provide a partial support of this claim but it warrants further investigation using more data sets.

5 Conclusions

In this paper, we propose a new ensemble construction algorithm, Meta-Evolutionary Ensembles (MEE). This algorithm employs a novel two-level evolutionary search through the space of ensembles, using feature selection as the diversity mechanism. At the first level, individual classifiers compete against each other to correctly predict held-out examples. Classifiers are rewarded for predicting difficult points, relative to the other members of their respective ensembles. At the top level, the ensembles compete directly based on classification accuracy.

Our model has several nice properties. First of all, our experimental results indicate that this method shows very comparable classification accuracy while keeping the ensemble size small by optimizing it directly. The final solution shows consistently improved classification performance compared to a single classifier at the cost of computational complexity. Compared to the traditional ensembles (Bagging and Boosting) and GEFS, our resulting ensemble shows comparable performance while maintaining a smaller ensemble.

Our model also makes it possible to understand and analyze how and why ensemble methods achieve improved predictive accuracy. Our two-level evolutionary framework confirms that more diversity among classifiers can improve predictive accuracy. Up to a certain level, the ensemble size also has a positive effect on the ensemble performance. In our model, the fittest ensemble is the one that survives through direct competition based on classification accuracy. In this way, we optimize ensembles directly, rather than combining optimized classifiers into an ensemble.

Further, our framework is a meta-search algorithm, meaning that it is independent of classifier types and/or various mechanism to promote diversity among classifiers. For example, we use feature selection as the mechanisms for individual diversity in this study. However, our flexible framework enables us to use data sampling (or both) to promote diversity among classifiers as in traditional ensemble methods. Our preliminary experiments show no big difference in overall performance between two methods for promoting diversity.

The next step is to compare this algorithm more rigorously to others on a larger collection of data sets, and perform any necessary performance tweaks on the EA energy allocation scheme. This new experiment is to verify the claim that Breiman [28] proposed through experiments on synthetic data sets. He claimed that there is relatively little room for other types of ensemble construction algorithm to obtain further improvement because his decision forest method performs at or near the Bayes optimal level. Along the way, we will examine the role of various characteristics of ensembles (size, diversity, etc.) and classifiers

(type, number of dimensions / data points, etc.). By giving the system as many degrees of freedom as possible and observing the characteristics that lead to successful ensembles, we can directly optimize these characteristics and translate the results to a more scalable architecture [29] for large-scale predictive tasks.

Another direction of future research is to investigate whether oversearching affects the classification performance of our model. Through the evolutionary process, MEE evaluates a number of ensemble models and selects one as the final solution. However, more extensive search can increase the probability of finding *fluke* rules that fit the data well but have low predictive accuracy [30]. We will investigate the relationship between the number of models and the predictive accuracy, particularly on data sets where MEE did not perform well.

Acknowledgements

This work was supported in part by NSF grant IIS-99-96044.

References

- [1] L. Breiman, Stacked regression, *Machine Learning* 24 (1) (1996) 49–64.
- [2] E. Bauer, R. Kohavi, An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants, *Machine Learning* 36 (1–2) (1999) 105–139.
- [3] D. H. Wolpert, Stacked generalization, *Neural Networks* 5 (2) (1992) 241–259.
- [4] L. Breiman, Bagging Predictors, *Machine Learning* 24 (2) (1996) 123–140.
- [5] Y. Freund, R. Schapire, Experiments with a New Boosting Algorithm, in: *Proc. of 13th Int’l Conf. on Machine Learning*, Bari, Italy, 1996, pp. 148–156.
- [6] R. E. Schapire, The strength of weak learnability, *Machine Learning* 5 (2) (1990) 197–227.
- [7] L. Breiman, Bias, variance, and Arcing classifiers, Tech. Rep. 460, University of California, Department of Statistics, Berkeley, California (1996).
- [8] T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting and randomization, *Machine Learning* 40 (2) (2000) 139–157.
- [9] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, *Journal of Artificial Intelligence Research* 11 (1999) 169–198.

- [10] J. Kittler, Feature selection and extraction, in: Y. Fu (Ed.), Handbook of Pattern Recognition and Image Processing, Academic Press, New York, 1986, pp. 203–217.
- [11] P. S. Bradley, O. L. Mangasarian, W. N. Street, Feature Selection via Mathematical Programming, *INFORMS Journal on Computing* 10 (2) (1998) 209–217.
- [12] J. Yang, V. Honavar, Feature Subset Selection Using a Genetic Algorithm, *IEEE Intelligent Systems and their Applications* 13 (2) (1998) 44–49.
- [13] R. Kohavi, G. H. John, Wrappers for feature subset selection, *Artificial Intelligence* 97 (1–2) (1997) 273–324.
- [14] F. Menczer, M. Degeratu, W. N. Street, Efficient and scalable Pareto optimization by evolutionary local selection algorithms, *Evolutionary Computation* 8 (2) (2000) 223–247.
- [15] Y. Kim, W. N. Street, F. Menczer, An ecological system for unsupervised model selection, *Intelligent Data Analysis* 6 (6) (2002) 2–17.
- [16] Y. Kim, W. N. Street, G. J. Russell, F. Menczer, Customer targeting: A neural network approach guided by genetic algorithms, *Management Science* 51 (2) (2005) 264–276.
- [17] D. Opitz, Feature selection for ensembles, in: 16th National Conf. on Artificial Intelligence (AAAI), Orlando, FL, 1999, pp. 379–384.
- [18] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: D. T. G. Tesauro, T. Leen (Eds.), *Advances in Neural Information Processing Systems*, Vol. 7, MIT Press, Cambridge, MA, 1995, pp. 231–238.
- [19] S. Hashem, Optimal linear combinations of neural networks, *Neural Networks* 10 (4) (1997) 599–614.
- [20] D. Opitz, J. Shavlik, Actively Searching for an Effective Neural-network Ensemble, *Connection Science* 8(3/4) (1996) 337–353.
- [21] T. K. Ho, The random subspace method for constructing decision forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (8) (1998) 832–844.
- [22] T. K. Ho, C4.5 decision forests, in: *Proc. of 14th Int’l Conf. on Pattern Recognition*, 1998, pp. 545–549.
- [23] C. Guerra-Salcedo, D. Whitley, Genetic approach to feature selection for ensemble creation, in: *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, 1999, pp. 236–243.
- [24] P. Cunningham, J. Carney, Diversity versus quality in classification ensembles based on feature selection, *Tech. Rep. TCD-CS-2000-02*, Trinity College Dublin, Department of Computer Science (2000).

- [25] C. L. Blake, C. J. Merz, UCI repository of machine learning databases [<http://www.ics.uci.edu/~mlearn/MLRepository.html>], university of California, Irvine, Department of Information and Computer Sciences. (1998).
- [26] B. Zadrozny, C. Elkan, Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers, in: Proc. of 18th Int'l Conf. on Machine Learning, 2001, pp. 609–616.
- [27] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
- [28] L. Breiman, Random forests–Random features, Tech. Rep. 567, University of California, Department of Statistics, Berkeley, California (1999).
- [29] W. N. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: Proc. of 7th Int'l Conf. on Knowledge Discovery & Data Mining (KDD-01), 2001, pp. 377–382.
- [30] J. R. Quinlan, R. M. Cameron-Jones, Oversearching and layered search in empirical learning, in: Proc. of 14th Int'l Joint Conf. on Artificial Intelligence, Morgan Kaufmann, 1995, pp. 1019–1024.