Meta-Evolutionary Ensembles

YongSeog Kim YONG-S-KIM@UIOWA.EDU
W. Nick Street NICK-STREET@UIOWA.EDU
Filippo Menczer FILIPPO-MENCZER@UIOWA.EDU
Management Sciences Department, University of Iowa, Iowa City, IA 52242 USA

Abstract

Ensemble methods have shown the potential to improve on the performance of individual classifiers as long as the members of the ensamble are sufficiently diverse. Individual classifiers have been trained for example on selected subsets of the records or on projections of the feature space to produce diversity. The resulting ensembles reflect *a priori* decisions about how to allocate records or features across classifiers. In this paper we propose a meta-evolutionary approach in which both individual classifiers and groups adapt. Ensembles compete for member classifiers, and are rewarded based on their predictive performance. Individual classifiers also evolve, competing to correctly classify test points, and are given extra rewards for getting difficult points right. In this way we aim to optimize ensembles rather than form ensembles of individually-optimized classifiers. Our preliminary results on a small number of data sets suggest that this approach can generate ensembles that are more effective than single classifiers and competetive with traditional ensemble methods. In the long run, this approach will provide new insight into how ensembles should be optimally constructed.

1. Introduction

In recent years, a great deal of interest in the machine learning 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 method such as Boosting, Bagging and Stacking differ in the ways that individual predictors are constructed, and in how their votes are combined. However, they have all demonstrated consistent – in some cases, remarkable – improvements in predictive accuracy over standard methods.

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, and as good as traditional ensemble methods. In the long term, this work is a step toward a clearer understanding of why ensembles are effective, and how they can best be constructed.

The remainder of this paper is organized as follows. In Section 2 we review ensemble methods and feature selection algorithms, both separately and in combination. 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 (Breiman, 1996b; Bauer & Kohavi, 1999; Wolpert, 1992). Bagging (Breiman, 1996b) and Boosting (Freund & Schapire, 1996; Schapire, 1990) 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 (Freund & Schapire, 1996) 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 (Breiman, 1996a) and AdaBoosting (Freund & Schapire, 1996). 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 (Bauer & Kohavi, 1999). Bagging and Boosting are known to reduce errors by reducing the variance term (Breiman, 1996a). According to (Freund & Schapire, 1996), 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 (Dietterich, 2000). In most cases, the improved performance of an ensemble is largely obtained by combining the first few classifiers (Opitz & Maclin, 1999).

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 (Kittler, 1986), nonlinear optimization (Bradley et al., 1998), and genetic algorithms (Yang & Honavar, 1998).

Our approach is based on the wrapper model (Kohavi & John, 1997) 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 (Menczer et al., 2000; Kim et al., 2000).

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 (Opitz, 1999). The diversity among component classifiers of ensemble has been proved critical to attain higher generalization accuracy (Krogh & Vedelsby, 1995; Hashem, 1997; Opitz & Shavlik, 1996). An ensemble generalizes well by combining many accurate component classifiers that make errors on different parts of data. According to (Hansen & Salamon, 1990), the expected error for an example i, $Error^i$, can be reduced to zero by adding infinite number of classifiers if $Error^i < 0.5$ and component classifiers are independent in the production of errors. 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 resampling training examples. However, ensemble methods typically use the complete set of features to train component classifiers. Recently several attempts have been made to incorporate the diversity in feature dimension into ensemble methods. The Random Subspace Method (RSM) in (Ho, 1998b; Ho, 1998a) 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 (Ho, 1998b; Ho, 1998a).

A more sophisticated way to select a subset of features for ensembles was proposed in (Guerra-Salcedo & Whitley, 1999). 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.

Genetic Ensemble Feature Selection (GEFS) (Opitz, 1999) 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. Longer chromosomes, however, make GEFS computationally expensive in terms of memory usage (Guerra-Salcedo & Whitley, 1999). 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 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 this 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

Pseudocode for the Meta-Evolutionary Ensembles (MEE) algorithm is shown in Figure 1, and a graphical depiction of the energy allocation scheme is shown if Figure 3. Each agent (candidate solution) 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 follows the commonality-based crossover framework (Chen et al., 1999). It 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 0.25. 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

```
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 ($class(r) == prediction(r, a')$)
        $count_{g,r} + +$
        $\Delta E = E^{g,r}_{envt} / \min(5, prevCount_{g,r})$
        $E^{g,r}_{envt} = E^{g,r}_{envt} - \Delta E$
        $E_a = E_a + \Delta E$
      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 $g$
    $accu_g = computeEnsembleAccuracy(g, Data_{test})$
  endfor
  for each ensemble $g$
    $E^g_{envt} = E^{tot}_{envt} / size(Pop^{i+1})$
    $E^g_{envt} * = size(g) \cdot 2 \cdot (inverseRankByAccuracy(g) + 1)/(G + 1)$
    for each record $r$ in $Data_{test}$
      $E^{g,r}_{envt} = E^g_{envt} / size(Data_{test})$
    endfor
  endfor
  $i = i + 1$
endwhile
```

Figure 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.

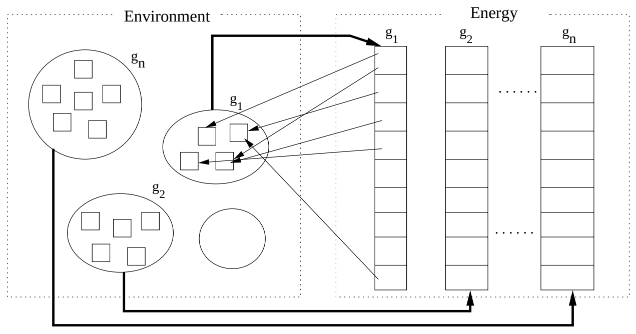

Figure 2. Graphical depiction of energy allocation in the MEE algorithm. Individual classifiers (small boxes in the environment) receive energy by correctly classifying test points. Energy for each ensemble is replenished between generations based on the accuracy of the ensemble. Ensembles with higher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

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

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

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 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 that was just evaluated 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 population size is independent of the reproduction threshold; θ only affects the energy stored by the population at steady-state.

The environment for each ensemble is replenished with energy based on its predictive 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 accuracy, 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. Note that the total replenishment energy that enters the system at each iteration is fixed and is independent of the population size.

4. Experimental Results

We tested the performance of MEE on several publicly available data sets (Blake & Merz, 1998). We show the characteristics of our data sets in Table 1. For comparison purposes we chose several data sets that were also used in (Opitz, 1999).

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 $\sqrt{inputs}$. 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, both for computational reasons and to maintain consistency with published expeirments using GEFS. The values for the various ELSA parameters are: $\Pr(mutation) = 1.0$, $\Pr(crossover) = 0.8$, $E_{cost} = 0.2$, $\theta = 0.3$, and $T = 30$. The value of E_{envt}^{tot} 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 5-fold cross-validation experiments. For each 5-fold cross-validation the original data set is first partitioned into 5 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 four sets. Within the training algorithm, each ANN is trained on two-thirds of the training set and tested on the remining third for

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

			Features		Neural Network	
Dataset	Records	Classes	Continuous	Discrete	Inputs	Epochs
diabetes	768	2	8	-	8	30
heart-cleveland	303	2	8	5	13	40
hepatitis	150	2	6	13	32	60
house-votes-84	435	2	-	16	16	40
ionosphere	351	2	34	-	34	40
iris	159	3	4	-	7	80
sonar	2080	2	60	-	60	100
wine	178	3	13	-	13	30

energy allocation purposes.

Experimental results are shown 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 of Table 2, a comparison is considered a tie if the intervals defined by one standard error of the mean overlap. On the data sets tested, MEE shows consistent improvement over a single neural network.

We also include the results of Bagging, AdaBoost, and GEFS from (Opitz, 1999) for indirect comparison. In these comparisons, we did 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 (Opitz, 1999). Our algorithm demonstrates competitive performance to that of the other algorithms. We note that such comparisons are inevitably inexact, since subtle methodological differences can cause variations in estimated accuracy. For instance, the results in (Opitz, 1999) are based on five 10-fold cross validation experiments, which generally result in higher estimated accuracy than those from 5-fold cross validation.[1]

It is also possible that the larger ensemble size used in GEFS contributes to improved accuracy. However, our preliminary results indicate that accuracy improvements flatten out at an ensemble size of approximately 15-25, seeming to confirm the results in (Opitz & Maclin, 1999). In fact, we expect that by optimizing the ensemble construction process, MEE will in general achieve comparable accuracy to other methods using fewer individuals.

We note that while it is generally a good idea to overfit the individual classifiers in an ensemble, we have not done so in the reported experiments, and may in fact be underfitting. As a preliminary test of the appropriateness of overfitting in the MEE framework, we have observed that on the Sonar data set, increasing the number of epochs by 40 resulted in significantly improved performance.

[1]This was confirmed with preliminary tests using single neural networks a few 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 preliminary experimental results indicate that this method is consistently better than a single classifier, and approximately equal in quality to Bagging, Boosting and GEFS.

We see this work as an initial step toward a better understanding of how and why ensemble methods achieve improved predictive accuracy. Then 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. 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 for large-scale predictive tasks.

Acknowledgements

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

Table 2. Experimental results, including win-loss-tie counts

	Single Net						MEE		
Dataset	Avg.	S.D.	S.E.	Bagging	AdaBoost	GEFS	Avg.	S.D.	S.E.
diabetes	76.5	0.5	0.2	77.2	76.7	77.0	77.1	0.4	0.2
heart-cleveland	81.1	1.5	0.7	83.0	78.9	83.9	83.6	1.0	0.4
hepatitis	81.8	0.5	0.2	82.2	80.3	83.3	85.0	1.5	0.7
house-votes-84	95.2	0.6	0.3	95.9	94.7	95.6	95.2	0.7	0.3
ionosphere	86.4	1.4	0.6	90.8	91.7	94.6	89.3	1.1	0.5
iris	95.4	0.9	0.4	96.0	96.1	96.7	96.0	0.8	0.4
sonar	81.0	1.5	0.7	83.2	87.0	82.2	83.4	0.6	0.3
wine	98.1	0.5	0.2	–	–	–	99.3	0.2	0.1
Single Net				4-3-0	2-2-3	4-0-3	6-0-2		
Bagging					3-4-0	4-3-0	1-2-4		
AdaBoost						6-1-0	2-2-3		
GEFS							2-1-4		

References

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*, 105–139.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Irvine, Department of Information and Computer Sciences.

Bradley, P. S., Mangasarian, O. L., & Street, W. N. (1998). Feature selection via mathematical programming. *INFORMS Journal on Computing*, *10*, 209–217.

Breiman, L. (1996a). *Bias, variance, and arcing classifiers* (Technical Report 460). University of California, Department of Statistics, Berkeley, California.

Breiman, L. (1996b). Stacked regression. *Machine Learning*, *24*, 49–64.

Chen, S., Guerra-Salcedo, C., & Smith, S. (1999). Non-standard crossover for a standard representation – commonality-based feature subset selection. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, *40*, 139–157.

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. *Proc. 13th Int'l Conf. on Machine Learning* (pp. 148–156). Bari, Italy.

Guerra-Salcedo, C., & Whitley, D. (1999). Genetic approach to feature selection for ensemble creation. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 236–243). Morgan Kaufmann.

Hansen, L., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 993–1001.

Hashem, S. (1997). Optimal linear combinations of neural networks. *Neural Networks*, *10*, 599–614.

Ho, T. (1998a). C4.5 decision forests. *Proc. 14th Int'l Conf. on Pattern Recognition* (pp. 545–549).

Ho, T. (1998b). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 832–844.

Kim, Y., Street, W. N., & Menczer, F. (2000). Feature selection in unsupervised learning via evolutionary search. *Proc. 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining (KDD-00)* (pp. 365–369).

Kittler, J. (1986). Feature selection and extraction. *Handbook of Pattern Recognition and Image Processing*. New York: Academic Press.

Kohavi, R., & John, G. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems* (pp. 231–238). Cambridge, MA: MIT Press.

Menczer, F., Degeratu, M., & Street, W. N. (2000). Efficient and scalable pareto optimization by evolutionary local selection algorithms. *Evolutionary Computation*, *8*, 223–247.

Opitz, D. (1999). Feature selection for ensembles. *16th National Conf. on Artificial Intelligence (AAAI)* (pp. 379–384). Orlando, FL.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, *11*, 169–198.

Opitz, D., & Shavlik, J. (1996). Actively searching for an effective neural-network ensemble. *Connection Science*, *8(3/4)*, 337–353.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.

Yang, J., & Honavar, V. (1998). Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applciations*, *13*, 44–49.