Evolutionary Ensembles: Combining Learning Agents using Genetic Algorithms

Jared Sylvester and Nitesh V. Chawla
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
{jsylvest, nchawla}@nd.edu

Abstract

Ensembles of classifiers are often used to achieve accuracy greater than any single classifier. The predictions of these classifiers are typically combined together by uniform or weighted voting. In this paper, we approach the ensembles construction under a multi-agent framework. Each individual agent is capable of learning from data, and the agents can either be homogenous (same learning algorithm) or heterogeneous (different learning algorithm). These learning agents are combined by a meta-agent that utilizes evolutionary algorithm, using the accuracy as fitness score, for discovering the weights for each individual agent. The weights are indicative the best searched combination (or collaboration) of the set of agents. Experimental results show that this approach is a valid model for ensemble building when compared to the best individual agent and a simple plurality vote of the agents.

Introduction

With the mushrooming of various domains requiring learning expertise, it is becoming relevant to understand the data complexity and the best corresponding learning algorithm or classifier in a reasonable time. A particular classifier might be more suited for a particular domain or task (Wolpert & Macready 1997). Typically, a benchmark study is conducted over a validation set, and the best classifier is chosen. However, this can be limiting as it is very possible for more than one algorithm to be optimal for that particular dataset, or different algorithms being equally accurate but making different kinds of errors. It can then become pertinent to combine the outputs of multiple classifier in an ensemble framework (Freund & Schapire 1996; Breiman 1996; 1999; Dietterich 2000b; 2000a; Woods, Kegelmeyer, & Bowyer 1997; Chawla *et al.* 2004). The output predictions of the classifiers in the ensembles are combined through voting, which can be either uniform or weighted. In addition, ensembles can be either heterogeneous or homogeneous (Tsoumakas, Katakis, & Vlahavas 2004). Homogeneous ensembles combine the multiple instances of classifiers generated with the same algorithm, while heterogeneous ensembles are composed of classifiers built from different algorithms.

In this paper, we categorize each classifier as a learning agent. Bradshaw (Bradshaw 1997) has defined various characteristics desirable of an agent. We present a discussion of those in relation to the learning agents:

- *Reactivity*: Each agent is given a contribtion weight for the final vote or prediction. This weight can essentially be considered as the reaction weight from each agent for an incoming test example. Moreover, in a distributed learning framework, each agent can potentially wait on trigger factors such as data availability or decision making requirement.
- *Autonomy*: Each agent acts completely independent of the other and is proactive. The learning agents can be homogeneous or heterogeneous, but do not require any communication or sharing of information during the training stage. Each agent's construction is completely independent of the rest.
- *Collaborative behavior*: The meta-agent (genetic algorithm) controls the collaboration of the agents by assigning each one of them weights based on their combined performance on the validation set. These weights are indicative of the level of collaboration offered by each individual agent. Higher the weight, more the contribution towards the final prediction. These weights are not individually optimized, but are reflective of the optimal combination or collaboration of the agents.
- *Inferential capability*: Each learning agent has an explicit model of self after the learning process, but not of the other learning agents. The meta-agent acts as the controller and combiner of the individual inferential capabilities.

The agents are then combined by a meta-agent (that we'll call EVEN) in an evolutionary learning framework. The meta-agent looks for the best combination weights for the individual agents based on a validation set. This paper addresses both homogeneous and heterogeneous fusion of agents. For the combination to be more accurate than its members it is important for the members to be diverse (Kuncheva & Whitaker 2003; Dietterich 2000b; Kuncheva 2005). Two classifiers are diverse if they disagree on their errors. Our goal is to construct a multi-agent framework that utilizes the best combination of the available agents. This combination should ideally be better than using all the agents uniformly, and even the best individual agent. We used genetic algorithm based procedure, EVEN (EVolutionary ENsemble), to weight the votes of each agent in an ensemble. EVEN begins with random weights assigned to each available agent. After multiple generations, these weights will be refined to reflect the relative worth of each agent in the classification scheme. One can also implement agents' selection under this framework by setting a threshold on weights.

Genetic algorithms have been used for feature selection in a wrapper mode (Opitz 1999; Guerra-Salcedo & Whitley 1999; Yang & Honavar 1997). They explore the space of possible feature subsets, and identify the best performing of all (on a validation set). Kim et. al (Kim, Street, & Menczer 2002) proposed meta-evolutionary ensembles that simultaneously considered multiple ensembles and allowed each component classifier to move in the final best-fit ensemble. Thus, an ensemble essentially evolved from ensembles. However, they only used a single neural network classifier. Menczer et al. (Menczer, Street, & Degeratu 2000) present a local selection methodology for evolving heterogeneous neural agents. Local selection is defined as a scheme that minimizes the interactions among members. Del Castillo and Serrano (Castillo & Serrano 2004) propose a multi-strategy method for text categorization using evolutionary algorithms.

EVolutionary ENsembles: EVEN

We developed an evolutionary framework for combining both heterogeneous and homogeneous learning agents. We implemented this within an *agent* architecture, wherein each individual classifier or learning algorithm is encoded as a learning agent. The agents adhere to the characteristics defined in the Introduction. The meta-agent, as a genetic algorithm, searches the space of these individual agents and identifies the best set of weights based on the performance on a validation set. These weighted agents are then used for predictions on a testing set, and the accuracies are recorded. EVEN was implemented using the GAlib provided by (Wall 1998).

Genetic Algorithms

Genetic algorithms are computational models inspired by evolution of biological populations. They provide a useful approximation for multiple parameter optimizations (Forrest 1996). The initial model developed by Holland in 1975 has been greatly extended and altered, but the basics of genetic algorithms remain the same (Holland 1992). A "population" of potential solutions is maintained. Each solution is encoded as a "chromosome," each of which can be evaluated to determine the solution's "fitness." Each succeeding generation is populated by the most fit members of the previous generation and their "offspring." Offsprings are created with crossover and mutation operators. Crossover combines the genetic information of two solutions to create new children, echoing reproduction in the natural world. Mutation alters the genes of an individual to introduce more diversity into the population, allowing more solution space to be searched. Though the initial population is often seeded with randomly generated solutions, and thus usually performs poorly, performance improves greatly over subsequent generations.

EVEN

The first step in any ensemble system is to build the component members. Weka, an open source software package that implements numerous machine learning algorithms in Java, was used for this task (Witten & Frank 2000). Learning agents were constructed on the training sets, and then evaluated first on the validation data and then on the testing data, as shown in Figure 1. Each agent generated its corresponding prediction on a validation set for each dataset considered in this paper. EVEN then customized the collaboration of each learning agent based on the performance on the validation set. That is, all EVEN knows of the classifier is the predictions it made; EVEN does not store any information about the decision making process of the underlying agents. Once the weights are learned for each individual agent, their predictions on the unseen testing set can be appropriately weighted. Finally, EVEN weights the testing set prediction of each agent by its weight chosen from the validation set. As we will see in the Results section, the weights chosen for different types of agents change for different datasets. Thereby further attesting to the fact that the there is no single optimal learning algorithm for different domains or tasks.

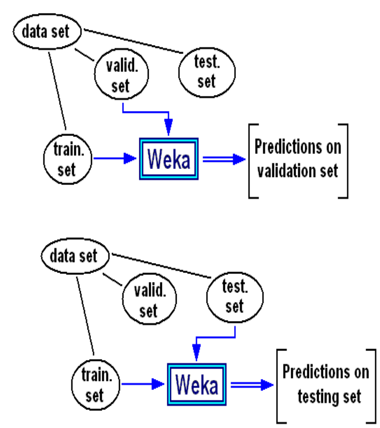

Figure 1: Generation of validation and testing set predictions using Weka. The validation set predictions are used by EVEN to generate weights for each member agent.

The fitness function used in EVEN is classification accuracy. Genetic algorithms optimize to their fitness functions, so it would be possible to swap in a different fitness function such as F-measure or ROC in place of accuracy. Each individual in EVEN's population is encoded as a real-valued, fixed-length array chromosome. Though Holland's initial algorithms used binary strings, real-valued chromosomes generally outperform bit-string encodings for problems which are naturally real-valued (Reeves & Rowe 2003). The array is equal in length to the number of classifiers being combined. Each value in the array corresponds to the weight of a classifier's vote in determining the final classification, and is normalized between zero and one. The population is initially random. The predictions on the validation set are generated for each member agent. EVEN through its generations evolves a set of weights that is the best-fit for the performance on the validation set. Thus, each generation is essentially a different combination in the agents' space that results in a better fitness score (accuracy). Figure 2 summarizes the procedure.

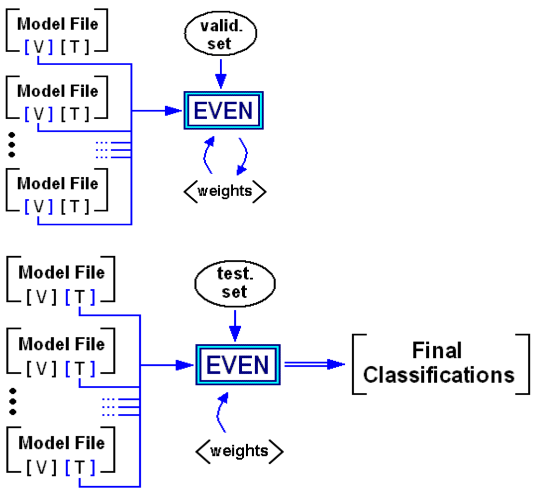

Figure 2: EVEN Procedure.

EVEN uses a steady state algorithm (i.e. some individuals are carried over to the next generation without modification, causing overlap between generations). Point mutations occur on a Gaussian distribution, meaning that after EVEN has decided a gene in a chromosome will be mutated its value is added to a random value selected from a Gaussian distribution to get the new value. The uniform crossover operator used for interbreeding, meaning that each gene of the child takes its value by randomly selecting which parent to copy from. That is, uniform crossover is similar to an n-point crossover, where there are n+1 genes on the chromosome.

EVEN is able to combine agents learned on any set of training examples, as long as the predictions are available for all instances in the validation and testing sets. This means that EVEN can combine both heterogeneous and homogeneous agents. This ability implies that EVEN can be easily applied to distributed learning of extremely large datasets. As we noted in the Introduction, each agent is completely autonomous so the approach can be easily parallelized or distributed, without requiring any communication or sharing of inference among the agents. This is a very desirable property for scalability and reduced computational complexity (Chawla *et al.* 2004).

Dataset	Classes	Train	Valid	Test
Satimage	6	4826	805	804
DNA	3	2389	399	398
Phoneme	2	3782	811	811
Pendigit	10	7694	1649	1649
Waveform	3	37500	6250	6250

Table 1: Datasets and the size of the training, validation and testing subsets.

Experiments

We ran various experiments to test our approach:

- Heterogenous agents: The following learning algorithms, as implemented in Weka, were utlized for this part of the experiments: ID3 decision trees , J48 decision trees (Weka implementation of C4.5), JRIP rule learner (Weka implementation of Ripper rule learning algorithm), NBTree (Naive Bayes tree), Nave Bayes, 1R classifier, logistic model trees, logistic regression, decision stumps and 1BK (k-nearest neighbor algorithm) (Witten & Frank 2000).
- Homogenous agents: The homogeneous agents were different instances of the same algorithm. These were constructed either by a) randomly perturbing the algorithm during learning (randomized trees) or b) by dividing a dataset into disjoint parts and learning a single agent on each part. Both led to a different learned function of the dataset, due to a random selection of data or decision tree split point.

In the subsequent subsections, we compare the performance of EVEN with the uniform voted ensemble and the single best member of the ensemble.

Datasets

We downloaded five datasets from the UCI repository: satimage, DNA, phoneme, waveform and pendigits (Blake & Merz 1998). Each dataset was randomly split into disjoint training, validation and testing sets. The datasets are summarized in Table 1. As evident, the datasets have varying characteristics.

Results

A number of different experiments were run with EVEN. The first set of experiments was to verify that EVEN was a legitimate framework for generating accurate combinations. Nine different algorithms implemented in Weka were used to generate the independent heterogeneous agents for these experiments. The second set of experiments had EVEN learn ensembles composed of randomized trees; that is each agent was a random instantiation of a decision tree algorithm. Another round of experiments was done with homogeneous classifiers which were trained on small subsets of

	Satimage	DNA	Phoneme	Pendigits
Best	88.57	94.97	90.99	99.33
Average	76.52	85.93	81.34	80.48
Uniform Vote	88.54	94.72	85.33	98.48
EVEN	89.94	94.97	89.77	99.21

Table 2: Accuracy of the best agent in the ensemble and the average of all agent, with the accuracies of a plurality vote and evolutionary ensemble.

	Random Trees
Best	74.18
Average	68.84
Vote	83.89
EVEN	85.33

Table 3: Accuracy of the best agent (randomized decision tree) in the ensemble and the average of all agents, with the accuracies of a plurality vote and evolutionary ensemble. Agents were learned on the entire training portion of the waveform set.

the available data. Finally, a set of experiments was done to see if the weights assigned to agents in heterogeneous ensembles were correlated with their accuracy.

For all of the experiments, EVEN's accuracy is compared to the most accurate classifier in the ensemble, the average of all the models, and a simple, un-weighted vote of all models. EVEN was run for 1000 generations with a population size of 128. Each generation evaluated the combination of different learning agents and generated a set of weights. The set of weights that resulted in the best fitness score on the validation set was retained. These set of weights were then used for weighted predictions on the testing set. In some limited cases, a greater number of generations resulted in very slight increases validation accuracy, and at most negligible increases in accuracy on the test set. Larger populations did not have an affect on performance.

Heterogeneous Agents

We only used satimage, DNA, phoneme and pendigits datasets for these experiments. We did not use Waveform dataset as various individual agents invariably achieved almost 100% accuracy; so there was very little expected gain from the ensemble. Table 2 summarizes these results. For all datasets, EVEN outperformed the voted ensemble. As expected, both EVEN and uniformly voted ensemble achieve significantly higher performances than the average member of the ensemble. It is encouraging that EVEN is very comparable to the best agent's performance on the testing set[1]. However, it seems, at first, that EVEN should be able to learn to defer to the best classifier, and in doing so be able to at least equal its performance. Further experimentation would be necessary to confirm this, but it is likely that this does not occur because the best performing classifier on the testing set was not necessarily the best on the validation set which EVEN was exposed to. On the four datasets evaluated here, EVEN outperformed the best classifier once, tied once and was beaten twice, albeit by very narrow margins. We plan to add more datasets and agents in our framework to conduct a more complete analysis with statistical tests of significance.

Randomized Trees as Homogeneous Agents

The second method used to generate diverse agents was randomized trees (Dietterich 2000a). We hypothesized that each agent could be a different random instantiation of a learning algorithm such as decision trees; each learning agent randomly selecting differnet splitting points for decision tree construction. To further weaken each individual decision tree, we implemented a stopping criteria that resulted in a bushier and weaker tree as it preempted the growth of the tree. The results appear in Table 3. As can be seen, EVEN outperformed both the simple vote and the most accurate of the trees. EVEN showed an improvement of 16.5% over the average accuracy of its constituent learning agents, and an improvement of around 11% over the best classifier. EVEN overcomes the individual weaknesses of the agents, and identifies their best collaborative aspects.

[1]Note that the reported best accuracy is potentially a biased estimate, as the best agent is chosen directly on the testing set.

Homogeneous Agents on Disjoint Sets of Data

The third method divided the training data into disjoint subsests, and trained agents on each of those subsets. These agents were homogeneous because they used the same underlying learning algorithm. The purpose of this experiment was to mimic a distributed learning setting, such that the agents could be resident on different sites and develop a local model of the data. Note that the autonomy of the agents is completely maintained. Both OneR and J48 models were trained on each subset. The OneR classifiers provided an opportunity to investigate the effects of a EVEN on "weak learners", while the J48 classifiers provide a less hindered estimate of the system's capability. Waveform was used again for training the classifiers, although it was randomly divided into 25 partitions, with each model being trained on one partition. Waveform, being the largest dataset, allowed us to construct multiple disjoint partitions.

Again, EVEN outperformed both voting and the best classifier when learned on either OneR or J48 algorithm, as indicated in Table 4. Improvements over the average of the models were 18.6% for the OneR experiment and 12.6% for the C4.5 experiment; improvements over the best of the models were around 16.5% for the OneR experiment and 12% for the C4.5 experiment. Moreover, EVEN provided an improvement over the uniformly voted ensemble as well for both the experiments. EVEN has not been implemented as a distributed system, but it is easy to imagine such an architecture, where in each agent is responsible for a particular set or site of data. This would be especially good for large datasets which do not easily fit in the memory of a single computer or for datasets that are naturally distributed at different sites. In addition, no inter-processor communication would be required while training the agents. Each agent reports its final predictions to a meta-agent.

	One-R	C4.5
Best	57.88	81.46
Average	55.65	79.91
Vote	65.86	91.81
EVEN	74.27	92.54

Table 4: Accuracy of the best agent in the ensemble and the average of all agents, with the accuracies of a plurality vote and evolutionary ensemble. Agents were learned on disjoint subsets of the waveform dataset.

Individual Agents' Accuracy and Assigned Weights

To establish the weights and accuracy trend among the agents, we ran 32 iterations of EVEN using the heterogenous mix of learning agents. Tables 5 to 8 report those results. We used the same datasets — satimage, dna, phoneme and pendigits — as in the original run. For each run of EVEN, the weight assigned to each of the agents was recorded. The average weights given to each agent, along with that agent's validation and testing accuracy, are in the Tables. Stronger agents were favored with higher weights, on average, but not in every execution. The rank-ordering of different agents based on their individual performances is not consistent across different datasets. In addition, there is a high variance in performance among the different learning agents for a dataset. As we mentioned, high performing agents didn't always get a high weight due to their lack of collaboration with the other agents.

This makes a very compelling case for EVEN, as it was able to identify a collaborative set of autonomous agents for each dataset, irrespective of their different inferential capabilities. Each agent has a different reactivity to each dataset, and to each other. Analysis into what properties of the learning algorithms might have influenced their weights is an area for future research. In addition, altering the genetic operators used may have an affect on accuracy and weight correlation. We would like to come up with the diversity measures of the best classifiers and best combined classifiers (chosen by EVEN) to shed more light on these results.

Classifier	Valid. Acc	Test. Acc	Avg.Wgt.
C4.5	86.70	84.97	0.2467
Ripper	85.57	86.33	0.2794
NBTree	80.72	83.11	0.0987
NB	78.11	79.75	0.0404
OneR	60.32	60.75	0.3475
LMT	86.82	88.20	0.3629
Log Reg	84.4500	85.4700	0.2321
Dec Stump	42.79	43.23	0.1530
1Bk	90.30	88.57	0.9457

Table 5: A Individual accuracy and average, maximum and minimum weights for Satimage dataset.

Classifier	Valid. Acc	Test. Acc	Avg. Wgt.
C4.5	91.48	93.4700	0.3921
Ripper	92.73	93.22	0.5138
NBTree	92.23	92.72	0.1839
NB	93.99	92.71	0.4816
OneR	66.17	61.81	0.8212
LMT	95.24	94.97	0.5658
Log Reg	95.24	94.97	0.7019
Dec Stump	64.41	61.31	0.4497
1Bk	75.69	79.90	0.1190
ID3	91.73	93.22	0.7092

Table 6: A Individual accuracy and average, maximum and minimum weights for DNA dataset.

Classifier	Valid. Acc	Test. Acc	Avg.Wgt.
C4.5	86.56	84.96	0.5893
Ripper	85.94	82.86	0.2028
NBTree	87.05	83.48	0.6671
NB	76.20	76.82	0.0800
OneR	74.59	75.34	0.0685
LMT	86.31	86.07	0.1197
Log Reg	73.61	75.46	0.1054
Dec Stump	76.57	76.08	0.0744
1Bk	90.75	90.99	0.9678

Table 7: Individual accuracy and average, maximum and minimum weights assigned for Phoneme dataset.

Conclusions and Future Work

An evolutionary approach was taken to building a combination of multi-agent framework. EVEN implemented an agent architecture such that each individual learning was reactive, autonomous, and independent. The meta-agent implemented the collaboration among the individual agents by evolving a set of weights for each agent based on the performance on the validation set. This resulted in a weighted prediction on the final testing set. The ensembles built by the system proved to be superior to the uniform voting of all the learning agents. The ensembles were about as accurate as the best classifier when multiple methods were used to generate the algorithms, and superior to the best classifier in all tests using a single type of classifier. The system was also shown to effectively combine classifiers trained on small subsets of the training data. Data is also presented that shows a correspondence between the weight assigned to an agent and it's accuracy. We believe it is important to look at distributed learning framework as part of an agent architecture to give certain decision making and independence to each learning algorithm.

There are many possible expansions to the system outlined above. One of the issues that would be worth addressing in future includes optimizing to other performance metrics, such as ROC. Also, it would be important to evaluate EVEN with respect to bagging, boosting and various other meta-learning schemes. Incorporating each agent's proba-

Classifier	Valid. Acc	Test. Acc	Avg.Wgt.
C4.5	96.24	96.06	0.3316
Ripper	95.88	96.30	0.4911
NBTree	94.48	93.94	0.3573
NB	86.11	86.69	0.1526
OneR	39.90	38.14	0.8390
LMT	98.06	98.18	0.5719
Log Reg	94.85	96.12	0.3385
Dec Stump	20.62	19.53	0.2198
1Bk	99.03	99.33	0.9117

Table 8: Individual accuracy and average, maximum and minimum weights for Pendigits dataset.

bility distributions for predictions generated, as opposed to the predictions alone, might increase performance. A more detailed analysis would also be needed to determine which of the genetic operators are best suited for EVEN. We would also like to be able to measure the diversity among the selected agents as an indicator of the improvement offered by EVEN (Kim, Street, & Menczer 2002). As mentioned, a distributed version of EVEN could be designed for building ensembles of models trained on small bites of data and compared to the performance of DIvote (Chawla *et al.* 2004). Finally, further analysis of the weights assigned to the ensembles of different classifier schemes could prove insightful. Another avenue of research would be to allow the meta-agents to adapt its rules to the changes in the domain, that is be adaptive to the environment.

Acknowledgements

The models used by EVEN were built in the Waikato Environment for Knowledge Analysis (Weka) developed at the University of Waikato (Witten & Frank 2000). EVEN was implemented with the GAlib library developed by Matthew Wall at MIT (Wall 1998). Thanks to the anonymous reviewers for their useful comments.

References

Blake, C., and Merz, C. 1998. UCI repository of machine learning databases.

Bradshaw, J. 1997. *Software Agents*. Cambridge, MA: The MIT Press.

Breiman, L. 1996. Bagging predictors. *Machine Learning* 24(2):123–140.

Breiman, L. 1999. Pasting bites together for prediction in large data sets. *Machine Learning* 36(1,2):85–103.

Castillo, M. D., and Serrano, J. 2004. A multistrategy approach for digital text categorization from imbalanced documents. *ACM SIGKDD Explorations Newsletter* 6:70–79.

Chawla, N. V.; Hall, L. O.; Bowyer, K. W.; and Kegelmeyer, W. P. 2004. Learning Ensembles From Bites: A Scalable and Accurate Approach. *Journal of Machine Learning Research* 5:421 – 451.

Dietterich, T. 2000a. An empirical comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. *Machine Learning* 40(2):139 – 157.

Dietterich, T. G. 2000b. Ensemble methods in machine learning. *Lecture Notes in Computer Science* 1857:1–15.

Forrest, S. 1996. Genetic algorithms. *ACM Computing Surveys* 28:77–80.

Freund, Y., and Schapire, R. 1996. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*.

Guerra-Salcedo, C., and Whitley, L. 1999. Genetic approach to feature selection for ensemble creation. In *International Conference on Genetic and Evolutionary Computation*, 236–243.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems*. Cambridge, MA: The MIT Press.

Kim, Y.; Street, N.; and Menczer, F. 2002. Meta-evolutionary ensembles. In *IEEE Intl. Joint Conf. on Neural Networks*, 2791–2796.

Kuncheva, L., and Whitaker, C. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning* 51:181–207.

Kuncheva, L. 2005. Diversity in multiple classifier systems. *Information Fusion* 6:2–3.

Menczer, F.; Street, W. N.; and Degeratu, M. 2000. Evolving heterogeneous neural agents by local selection. In Honavar, V.; Patel, M.; and Balakrishnan, K., eds., *Advances in the Evolutionary Synthesis of Neural Systems*. Cambridge, MA: MIT Press.

Opitz, D. 1999. Feature selection for ensembles. In *AAAI/IAAI*, 379–384.

Reeves, C., and Rowe, J. 2003. *Genetic Algorithms - Principles and Perspectives: A Guide to GA Theory*. MA: Kluwer.

Tsoumakas, G.; Katakis, I.; and Vlahavas, I. 2004. Effective voting of heterogeneous classifiers. *Lecture Notes in Computer Science* 3201:465–476.

Wall, M. 1998. Galib: A c++ library of genetic algorithm components. (version 2.4, revision b).

Witten, I. H., and Frank, E. 2000. *Data Mining: Practical machine learning tools with Java implementations*. San Francisco, CA: Morgan Kaufmann.

Wolpert, D. H., and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1):67–82.

Woods, K.; Kegelmeyer, W. P.; and Bowyer, K. W. 1997. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Anal. Machine Intelligence* 19:405–410.

Yang, J., and Honavar, V. 1997. Feature subset selection using A genetic algorithm. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, 380.