Meta-Evolutionary Ensembles

YongSeog Kim
Management Sciences Dept.
University of Iowa
Iowa City, IA 52242 USA
yong-s-kim@uiowa.edu

W. Nick Street
Management Sciences Dept.
University of Iowa
Iowa City, IA 52242 USA
nick-street@uiowa.edu

Filippo Menczer
Management Sciences Dept.
University of Iowa
Iowa City, IA 52242 USA
filippo-menczer@uiowa.edu

Abstract- Ensemble methods have shown the potential to improve on the performance of individual classifiers as long as the members of the ensemble are sufficiently diverse. In this paper we propose a meta-evolutionary approach in which both individual classifiers and ensembles 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 build small-sized optimal ensembles rather than form large-sized ensembles of individually-optimized classifiers. We use neural networks as individual classifiers, and feature selection to promote the diversity among classifiers in the same ensemble. Experimental results on 17 data sets suggest that our algorithms can generate ensembles that are more effective than single classifiers and traditional ensemble methods. We also use the evolutionary framework to explore the role of ensemble characteristics such as size and diversity in creating accurate ensembles.

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. 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. We examine the relationship between classifier diversity and ensembles size to and the predictive accuracy of the ensemble. Finally, we compare the effectiveness of two different classifiers, neural networks and decision trees, in the framework of ensemble feature selection.

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 Feature selection for Ensembles

Recently many researchers have combined the predictions of multiple classifiers to produce a better classifier, an ensemble, and often reported improved performance [4, 1]. Bagging [2] and Boosting [7] are the most popular methods for creating accurate ensembles. Bagging 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. By combining multiple classifiers by voting, Bagging can produce lower test set error than a single classifier because the diversity among individual classifiers compensates for the increase in error rate of any individual classifier.

Boosting [7] 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 [3] and AdaBoosting [7]. 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 comes primarily from the diversity caused by re-sampling training examples while using 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 [10, 9] 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 with some success.

A more sophisticated way to select a subset of features for ensembles was proposed in [8]. 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 [6], 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) [11] also used a GA to search for possible feature subsets. GEFS starts with an initial population of classifiers built using up to $2 \cdot D$ features, where D is the complete feature dimension. It is possible for some features to be selected more than once in GEFS and crossover and mutation operators are used to search for new feature subsets. Using 100 most-fit members with majority voting scheme, 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 [8]. 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.

However, all these methods consider only one ensemble. 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, we also use a local selection mechanism in which classifiers compete with each other only if they belong to the same ensemble.

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 in 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 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 en-

```
initialize population of agents, each with energy θ/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} ++
            ΔE = E_envt^{g,r} / min(5, prevCount_{g,r})
            E_envt^{g,r} = E_envt^{g,r} − ΔE
            E_a = E_a + ΔE
         endif
      endfor
      E_a = E_a − E_cost
      if (E_a > θ)
         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
```

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.

ergy 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 $E^{g,r}_{envt}$ 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

$$\Delta E = \frac{E^{g,r}_{envt}}{\min(5, prevCount_{g,r})}. \quad (1)$$

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 ($E_{cost} < \theta$).

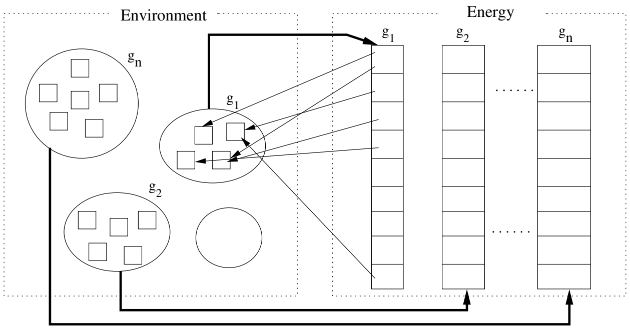

Figure 2: Graphical depiction of energy allocation in the MEE algorithm. Individual classifiers (small boxes in the environment) receive energy by correctly classifying test points. Energy for each ensemble is replenished between generations based on the accuracy of the ensemble. Ensembles with higher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

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 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 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 Experimental results of MEE/ANN

We tested the performance of MEE combined with neural networks on several data sets that were used in [11]. 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 $\sqrt{inputs}$. 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: $\Pr(mutation)$ = 1.0, $\Pr(crossover)$ = 0.8, E_{cost} = 0.2, θ = 0.3, and T = 30. The value of E_{envt}^{tot} = 30 is chosen to maintain a population size around 100 classifier agents.

Experimental results are shown in Table 1. 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. 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. 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 1, a comparison is considered a tie if the intervals defined by one standard error [1] 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 [11] 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 [11]. In terms of predictive accuracy, our algorithm demonstrates better or equal performance compared to single neural networks with the complete set of features and the traditional ensembles, Bagging and Boosting. However, MEE shows slightly worse performance compared to GEFS. 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 such as Glass and Labor data.

From the perspective of computational complexity, our 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.

4.2 Guidelines toward optimized ensemble construction

In this section, we use MEE to examine ensemble characteristics and provide some guidelines for building optimal ensembles. We expect that by optimizing the ensemble construction process, MEE will in general achieve comparable accuracy to other methods using fewer individuals. We use data collected from the first fold of the first cross-validation routine for the following analyses.

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 the number of classifiers in the ensemble increases. For example, our result in Figure 4.2 indicates that accuracy improvements flatten out at an ensemble size of approximately 15-25, seeming to confirm the results in [12].

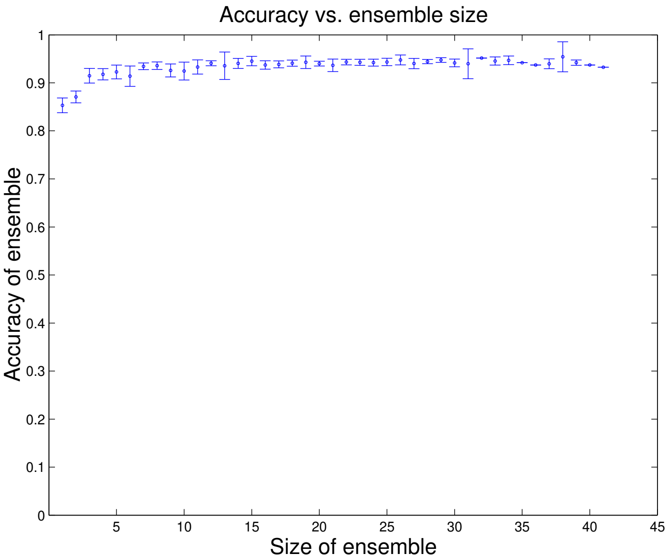

Figure 3: The relationship between the predictive accuracy and ensemble size with 95% confidence interval on the Soybean data. We observed similar patterns on other data sets.

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:

$$\alpha \oplus \beta = \begin{cases} 0 & \text{if } \alpha = \beta \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

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

$$diversity^e = \frac{\sum_{i=1}^{g} \sum_{j=1}^{N} (pred_j^i \oplus pred_j^e)}{g \cdot N} \tag{3}$$

where N is the number of records in the test data and $pred_j^i$ and $pred_j^e$ represent the predicted class label for record j by classifier i and ensemble e respectively. The larger the value of $diversity^e$, the more diverse ensemble is.

We show the relationship between the predictive accuracy and ensemble diversity in Figure 4.2. This show the expected the positive relationship between accuracy and diversity. However, our results show that too much diversity among classifiers can deteriorate ensemble performance, as the final decision made by ensemble becomes a random guess.

5 Conclusions

In this paper, we propose a new two-level ensemble construction algorithm, Meta-Evolutionary Ensembles (MEE) that uses

[1] In our experiments, standard error is computed as standard deviation / $\sqrt{iter}$ where $iter = 5$

Table 1: Experimental results of MEE/ANN

	Single Net					MEE		
Dataset	Avg.	S.D.	Bagging	AdaBoost	GEFS	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
glass	57.1	2.69	66.9	68.9	69.6	61.1	1.73	100
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
hypo	93.8	0.09	93.8	93.8	94.1	93.9	0.06	50
ionosphere	89.3	0.85	90.8	91.7	94.6	93.5	0.81	100
iris	95.9	1.10	96.0	96.1	96.7	96.5	0.73	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
segment	92.3	0.97	94.6	96.7	96.4	93.2	0.28	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
soybean	92.0	0.92	93.1	93.7	94.1	93.8	0.19	50
vehicle	74.7	0.48	79.3	80.3	81.0	76.4	1.12	50
Win-loss-tie	15-0-2		7-4-6	9-6-2	4-7-6			

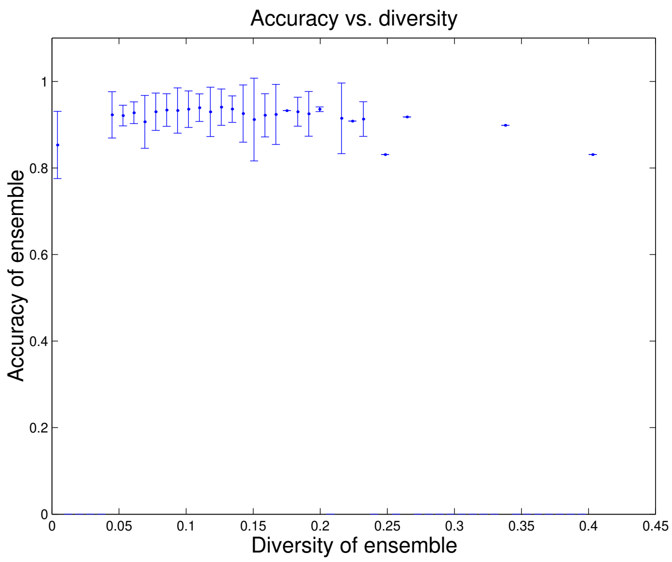

Figure 4: The relationship between the predictive accuracy and ensemble diversity with 95% confidence interval on the Soybean data. Similar patterns were observed on other data sets.

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 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. Further, 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.

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 [5] 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 [13] for large-scale predictive tasks.

Acknowledgements

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

Bibliography

[1] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and

variants. *Machine Learning*, 36:105–139, 1999.

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

[3] L. Breiman. Bias, variance, and Arching classifiers. Technical Report 460, University of California, Department of Statistics, Berkeley, California, 1996.

[4] L. Breiman. Stacked regression. *Machine Learning*, 24(1):49–64, 1996.

[5] L. Breiman. Random forests–Random features. Technical Report 567, University of California, Department of Statistics, Berkeley, California, 1999.

[6] P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. Technical Report TCD-CS-2000-02, Trinity College Dublin, Department of Computer Science, 2000.

[7] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. of 13th Int'l Conf. on Machine Learning*, pages 148–156, Bari, Italy, 1996.

[8] C. Guerra-Salcedo and D. Whitley. Genetic approach to feature selection for ensemble creation. In *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 236–243. Morgan Kaufmann, 1999.

[9] T. K. Ho. C4.5 decision forests. In *Proc. of 14th Int'l Conf. on Pattern Recognition*, pages 545–549, 1998.

[10] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

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

[12] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[13] W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proc. of 7th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining (KDD-01)*, 2001. 377–382.