Feature Selection and Multi-Class Classification Using a Rule Ensemble Method

Author
Institution
Address
City
email

Author
Institution
Address
City
email

Author
Institution
Address
City
email

ABSTRACT

Ensemble methods for supervised machine learning have become popular due to their ability to accurately predict class labels with groups of simple, lightweight "base learners." While ensembles offer computationally efficient models that have good predictive capability, they tend to be large and offer little insight into the patterns or structure in a dataset. In this study, we extend an ensemble technique that accurately predicts class labels and has the advantage of indicating which parameter constraints are most useful for predicting those labels. We develop a method for using this ranking to select features and remove less predictive attributes. We illustrate our method on a dataset containing images of potential supernovas, where we select 21 out of 39 features without reducing classification accuracy. We also extend the rule ensemble method to multi-class classification and compare it with the boosting and bagging ensemble methods on various classical multi-class datasets.

1. INTRODUCTION

Machine learning algorithms are popular tools for growing powerful models that can successfully predict class affiliations of unlabeled observations. These algorithms can attain high classification accuracy for datasets with complex behavior and from a wide variety of applications. A disadvantage of machine learning is that models can become overly complicated and, as a result, hard to interpret and expensive to evaluate for large datasets. Ideally we would like to generate models that are quick to build, cheap to evaluate, and that give users some insight into the data.

Ensemble methods build larger models by combining many elementary models, that are quick to build and referred to as base learners. The larger model captures more behavior than each base learner captures by itself and so collectively the base learners can model the population and accurately predict class labels [17]. Classical ensemble methods that use decision trees as base learners, such as bagging and boosting, have good predictive capability, but cannot offer much insight into the structure of a dataset [3, 12, 25].

In this paper, we extend an ensemble method that uses rules as base learners and was proposed by Friedman and Popescu [19] for binary classification and regression. This method is attractive, as it gives insight into the structure of a dataset, while most other ensemble methods cannot provide any insight. The Friedman and Popescu rule ensemble method uses the inner nodes from a set of decision trees as rules and combines the rules into a linear model, using weights from an L^1 penalized regression. The regression ranks rules in order of importance, and the L^1 penalty prunes rules of little utility, which gives insight into which attributes are more important.

First, we modify the Friedman-Popescu rule ensemble method by using several sophisticated techniques from image processing, statistics and mathematical optimization to calculate rule weights, rather than the constrained steepest descent method [18] that was originally employed [19]. The rule ranking we use successfully prunes more rules, which yields a smaller model. Because the techniques we employ have been developed in different fields, each frames the minimization problem slightly differently. To our knowledge, these variations of the rule ensemble scheme have not been investigated and evaluated in this context before.

Second, we define a voting scheme to remove less influential attributes. While Friedman and Popescu explored how the ranking could be used to study the importance of rules, we use the ranking to eliminate features from the dataset. We present an example of how we identify important attributes in a large scientific dataset by applying our techniques to a set of images of potential supernovas.

We finish by extending the rule ensemble method from its original binary capability to work with datasets that have multiple classes. We test this new multi-class method on classical machine learning datasets from the UC Irvine machine learning repository [14] and consider how the different weighting schemes perform in this general setting.

This research was supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231

2. OVERVIEW OF THE RULE ENSEMBLE METHOD

Consider a dataset of points $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i$ denotes the ith observation, with label y_i. Each of the observations, $\mathbf{x} \in \mathbb{R}^{\mathcal{K}}$, has $\mathcal{K}$ attributes or feature values that we measure for each observation. The matrix $\mathbf{X}$ will denote the set of all $\mathbf{x}_i$'s. The jth feature of the ith observation is the scalar x_{ij}. Our goal then is to be able to predict what class y a future unlabeled observation $\mathbf{x}$ belongs to. The rule ensemble method focuses specifically on the binary decision problem where y can be one of only two classes $\{-1, +1\}$. To classify observations we seek to construct a function $F(\mathbf{x})$ that maps an observation $\mathbf{x}$ to an output variable $\hat{y} = F(\mathbf{x})$ that predicts the true label y.

Define the risk of using any function that maps observations to labels as

$$R(F) = E_{\mathbf{x},y} L(y, F(\mathbf{x})), \tag{1}$$

where $E_{\mathbf{x},y}$ is the expectation operator. $L(y, \hat{y})$ is a chosen loss function that defines the cost of predicting a label $\hat{y}$ for an observation when the true label is y.

In the original method, only ramp loss was used, as it is particularly well suited to control for outliers in the binary classification problem [17, 19]. In this paper, we will also consider the use of the squared error loss function, as it has properties that are necessary to the alternative algorithms that we employ to weight rules. Within this framework we seek to find a function, that minimizes the risk. We approximate this optimal function with a function $\hat{F}(\mathbf{x})$ that minimizes the expected loss on the set of observed training data $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^N$. We assume that $\hat{F}(\mathbf{x})$ has the form of a linear combination of K base learners $\{f_k(\mathbf{x})\}_{k=1}^K$:

$$\hat{F}(\mathbf{x}; \mathbf{a}) = a_0 + \sum_{k=1}^{K} a_k f_k(\mathbf{x}). \tag{2}$$

The function $\hat{F}(\mathbf{x}; \mathbf{a})$ returns a real value. To get a prediction for the binary valued label, we take the sign of the approximation $\hat{y} = \text{sign}(\hat{F}(\mathbf{x}; \mathbf{a}))$. Ideally, we want the coefficients $\mathbf{a} = (a_1, a_2, \ldots, a_K)$ that minimize the risk (1). As an approximation, we use the coefficients $\mathbf{a}^*$ that minimize the risk over the given sample set of observations $\mathcal{S}$

$$\mathbf{a}^* = \underset{\{\mathbf{a}\}}{\text{argmin}}\ E_{\mathcal{S}} L(y, F(\mathbf{x}; \mathbf{a})), \tag{3}$$

$$= \underset{\{\mathbf{a}\}}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, a_0 + \sum_{k=1}^{K} a_k f_k(\mathbf{x}_i)\right). \tag{4}$$

The solution to equation (4) is prone to overfit the training data, so we include a penalty term. The penalty forces a sparse solution that eliminates less influential terms from the model, is more robust to training data, and allows for a more interpretable model. Here, we use the L^1 (lasso [30]) penalty and the solution $\hat{\mathbf{a}}$ to the penalized problem

$$\hat{\mathbf{a}} = \arg\min_{\{\mathbf{a}\}} \sum_{i=1}^{N} L\left(y_i, a_0 + \sum_{k=1}^{K} a_k f_k(\mathbf{x}_i)\right) + \lambda \sum_{k=1}^{K} |a_k|. \tag{5}$$

The impact of the penalty is controlled by the parameter $\lambda \geq 0$. This penalized problem has received a great deal of attention [13, 15, 20] and enables both estimation of the coefficients as well as coefficient selection.

2.1 Base Learners - Rules

The base learners f_k in equation (2) can take on a variety of forms to create different ensemble methods. In ensemble methods such as bagging, random forests, and boosting, decision trees are used as base learners [7]. For the rule ensemble method, simple rules r_k of the form

$$r_k(\mathbf{x}_i; \mathbf{p}_k) = \prod_j I(x_{ij} \in p_{kj}), \tag{6}$$

where $I(x_{ij} \in p_{kj})$ is an indicator function, are used as base learners or terms in the linear model. Each p_{kj} is a constraint that the kth rule assigns to the jth attribute. For convenience we will denote $\mathbf{p}_k = (p_{k1}, \ldots)$ to be the vector of parameter constraints that an observation must meet to have the kth rule to evaluate to 1.

The rules are generated by computing parameter sets $\{\mathbf{p}_k\}_{k=1}^K$ by growing decision trees. Each internal node of the decision trees (not the root nodes) takes the form of a simple rule defined in equation (6). The decision trees are grown with gradient boosting and on randomly selected subsets of data, to avoid regrowing overlapping rules and to ensure a wide variety of rules. Further details on how to generate a diverse set of rules can be found either in Friedman's and Popescu's paper [19] or our technical report [10].

2.2 Weighting Rules

To approximate the coefficients $\hat{\mathbf{a}}$ defined in equation (5), we test a variety of algorithms that come from a wide range of fields. Here we give a brief overview of each method and describe how it encourages a sparse solution.

2.2.1 PATHBUILD

The method that Friedman and Popescu suggested to use to approximate coefficients is a constrained gradient descent method that we will refer to as PATHBUILD [18]. The method does not solve (5) explicitly, but rather initializes all coefficients to zero and only progresses in the direction of rules that have a large effect on the predictive capability of the model. At each iteration ℓ the subset of coefficients that get increased at iteration $\ell + 1$ is defined by

$$\{k : |g_k(\mathbf{X}; \mathbf{a}^\ell)| \geq \tau * ||g_k(\mathbf{X}; \mathbf{a}^\ell)||_\infty\}.$$

Here $g_k(\mathbf{X}; \mathbf{a}^\ell)$ is the kth component of the gradient of the risk (equation (1)) evaluated with $F(\mathbf{x})$ on the entire dataset $\mathbf{X}$ at the ℓth iteration of the descent. The coefficients in this set have components of the gradient that have a magnitude greater than some fraction $\tau \in [0, 1]$ of the absolute value of the largest gradient component. The set of directions the method advances in can change at every iteration. By not advancing in directions that have little change in the risk function, coefficients for rules that have little effect are prevented from "stepping" off zero and thus kept out of the model. Lower values of τ include more rules in the model. The largest model results when $\tau = 0$, which causes a basic gradient descent method and unpenalized regression.

2.2.2 GLMNET

The GLMNET package approximates a solution to the least squared error regression subject to an elastic net penalty,

$$\min_{\{\mathbf{a}\}} \frac{1}{N} ||F(\mathbf{X};\mathbf{a}) - \mathbf{y}||_2 + \lambda P_\alpha(\mathbf{a}), \quad (7)$$

with a coordinate-wise gradient descent method [15]. The elastic net is defined as

$$P_\alpha(x) = \alpha||\mathbf{a}||_1 + (1-\alpha)||\mathbf{a}||_2^2$$

for $\alpha \in [0,1]$. We set $\alpha = 1$ and get the same problem as in equation (5), but with the L^2 norm as the loss function instead of ramp loss. Using $\alpha = 1$, we solve for the rule weights $\mathbf{a}$. The coordinate-wise gradient descent method starts with the null solution, which corresponds to solving equation (7) with $\lambda = \infty$ and is similar to PATHBUILD. Then GLMNET cycles over the coefficients and uses partial residuals and a soft-thresholding operator to update each coefficient one by one [16]. GLMNET has modifications that allow parameters to be updated at the same time as neighboring parameters. As the coefficients are updated, λ is decreased exponentially and a set of coefficients is calculated at each increment of the path $\lambda = \infty$ to $\lambda = \lambda_{min}$. Each increment the previous solution as a "warm start" to approximate the next solution.

2.2.3 SPGL1

The SPGL1 (sparse projected-gradient $l1$) package [31] solves for the coefficients $\mathbf{a}$ in

$$\min_{\{\mathbf{a}\}} ||F(\mathbf{X};\mathbf{a}) - \mathbf{y}||_2 \text{ subject to } ||\mathbf{a}||_1 <= \sigma, \quad (8)$$

which is an equivalent formulation to the problem in equation (5) using the L^2 norm as the loss function. At each iteration of the algorithm, a convex optimization problem is constructed, whose solution yields derivative information that can be used by a Newton-based root-finding algorithm [32]. Each iteration of the SPGL1 method has an outer/inner iteration structure, where each outer iteration first computes an approximation to σ. The inner iteration then uses a spectral gradient-projection method to approximately minimize a least-squares problem with an explicit one-norm constraint specified by σ. Some advantages of the SPGL1 method are that only matrix-vector operations are required and numerical experience has shown that it scales well to large problems.

2.2.4 FPC

The FPC package (fixed point continuation method) [20] approximates the solution $\mathbf{a}$ by solving

$$\min_{\{\mathbf{a}\}} ||\mathbf{a}||_1 + \frac{\mu}{2} * ||F(\mathbf{X};\mathbf{a}) - \mathbf{y}||_2^2. \quad (9)$$

This problem formulation seeks to minimize the weighted sum of the norm of the coefficients and the error of the solution, the left and right terms respectively. The sparsity of $\mathbf{a}$ is controlled by the size of the weighting parameter μ. Increasing μ places more importance on minimizing the error, and reduces the ratio of the penalty to the error. Equation (9) is a reformulation of problem (5) with the lasso penalty, and is referred to as a basis pursuit problem in signal processing. The relation of the two problems can clearly be seen if, for any λ value, μ is chosen to be

$$\mu = \frac{2}{N\lambda}$$

and equation (9) is multiplied by λ. FPC was developed for compressing signals by extracting the central components of the signal. FPC uses the reformulations of the optimality conditions for the l_2 to declare a shrinkage operator s_ν, where ν is a shrinkage parameter that has both an effect on the speed of convergence and how many non-zero entries $\mathbf{a}^*$ has. The operator s_ν acts on a supplied initial value $\mathbf{a}^0$ (which we chose to be the null solution) and finds our solution $\mathbf{a}^*$ through a fixed point iteration.

2.3 Extension to Multi-class Classification

The rule ensemble method is designed for binary classification problems, but many datasets contain multiple classes that one needs to identify. To be applicable to classification in general, we extend the rule ensemble to multi-class problems. The regression performed to assemble the rules in the ensemble hinder the rule ensemble from being extended to problems where the classes are not ordered. To identify multiple classes, we base a method on the one-versus-all (OVA) classification technique, which has successfully been used to extend many binary methods into multi-class algorithms [22, 29]. Other methods for extending binary classification algorithms exist; however, many are more expensive than OVA, yet provide no more utility [27].

Generally for a problem with J classes, OVA classification performs J binary tests, where the jth test checks if an observation is or is not a member of the jth class. For the rule ensemble method we use a modified form of OVA classification. We perform J binary tests and each test returns a real valued prediction F_j. In the original method the label was predicted to be the sign of the real valued prediction. In the case of multiple classes, we avoid having multiple positive entries in the label vector $\hat{\mathbf{y}}$ by taking the prediction to be the class j^*, where F_{j^*} is greater than any other class label prediction. Choosing the largest label prediction is sensible, since the more confident the algorithm is that the observation is in a certain class, the closer to 1 the label prediction will be.

This section has provided a brief introduction to the origin of the ensemble method that was introduced by Friedman and Popescu [17, 19] and that is used in this study. Other papers provide more details on the algorithms we use to compute the coefficients [15, 18, 20, 32].

3. DATASETS AND METHODS FOR EXPERIMENTS

To test the behavior of the rule ensemble method on a binary classification problem, we used a dataset of images taken by a telescope [6, 24, 26], the goal being to identify potential supernovas. The initial data had three images for each observation. Those images were processed to yield 39 statistics for each observation that described the distribution and color of individual pixels within the original three images. These statistics became the attributes for the dataset and the observations were labeled with +1, -1 if they were or were not, respectively, an image of a supernova-like object. The dataset contains a total of 5,000 positive and 19,988 negative observations.

For a testing procedure on the binary supernova data, we

Set	Name	Attributes	Observations	Classes
1	breast-w	9	699	2
2	glass	9	214	7
3	ion	34	351	2
4	iris	4	150	3
5	pendigits	16	10992	10
6	phoneme	5	5404	2
7	pima	8	768	2
8	sonar	60	208	2
9	vehicle	18	846	4
10	waveform	21	5000	3

Table 1: Description of UC Irvine datasets used for multi-class problems.

randomly select 2,500 positive observations and 2,500 negative observations for a training set, and then use the remaining data for the testing set. We use this ratio of positive to negative observations based on previous work that was done in this application [1]. This selection process is repeated 10 times for cross-validation.

We assess the utility of the rule ensemble for multi-class problems on 10 datasets from the UC Irvine Machine Learning Data Repository [14] with five 2-fold cross-validation tests. We compare the accuracy of the rule ensemble with the classical bagging and boosting methods using OpenDT [2]. The datasets are briefly described in Table 1 and are taken from a wide variety of applications. The UC Irvine datasets and this testing procedure are chosen because they are standard for testing machine learning methods, including decision tree ensemble methods [3, 12, 25].

False positive and false negative error rates are used to assess the accuracy of the methods in addition to the overall error rate. The false positive rate is the ratio of observations misclassified as positive to the total number of negative observations in the test set, while the false negative rate is the ratio of observations misclassified as negative to the number of positive observations in the test set. The overall error rate is the ratio of observations misclassified to the total number of observations in the test set.

Experiments using the rule ensemble method were run using Matlab^{TM}7.10 on a MacBook Pro with a 2.66 GHz Intel Core i7 processor.

4. BINARY CLASSIFICATION RESULTS

To study the behavior of the rule ensemble method with a variety of weighting schemes, we first consider supernova dataset, which has binary class labels.

4.1 Rule Ensemble with PATHBUILD

Effect of Using the τ Threshold as Penalty The effect of the variable τ, that controls how many directions are updated at each iteration of PATHBUILD in the thresholded gradient descent method is shown in Figure 1. An increase in τ causes a higher threshold that results in fewer terms being included in each iteration of the coefficient finding method. Models built with a larger value of τ are also less accurate and have higher variance in the error rate. Within a certain range, decreasing τ further does not offer much

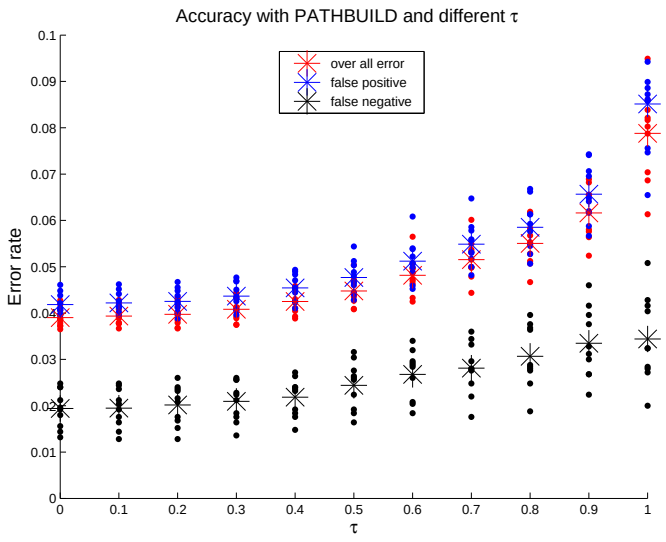

Figure 1: Error rate increases as τ increases and restricts the number of coordinates that PATHBUILD advances in at each iteration. This experiment was run with each tree having an average of 20 terminal nodes and 600 maximum rules.

increase in the predictive capability of the model. In this example, we see that when τ is between 0 and 0.4 there isn't a large increase in error rate, so using a weaker threshold of $\tau = 0.4$ will not significantly compromise the accuracy of our model. This is a good result, as a larger threshold decreases the computational expense of each iteration of the gradient descent method. The result that $\tau = 0.4$ produces similar error rates to using $\tau = 0$ means that we can get the same accuracy with fewer terms, less computation, and thus in less time.

4.2 Rule Ensemble with GLMNET

Here we use the rules generated in the previous experiment with GLMNET to build models using the coefficients that are generated at each step of the path $\lambda \in [\lambda_{min}, \infty]$. Figure 2 shows how the accuracy of the method changes as the weight of the penalty used to find the coefficients changes. The solution with GLMNET when λ is small results in slightly less error than the solution with PATHBUILD when τ is small. The variance in the error rates from solutions found with PATHBUILD is less than the variance of error rates from solutions found with GLMNET. Both solutions yield false positive rates that are more than twice as large as the false negative rates; this is probably a result of the ratio of positive to negative observations in the test set being small. The error rate slowly decreases as λ decreases, but then the error rate stabilizes when λ is very small, $\lambda < 0.01$. It is interesting that the variance in error rates of the solutions is relatively constant as λ changes.

4.3 Rule Ensemble with SPGL1

The results using SPGL1 are shown in Figure 3. The accuracy of the SPGL1 solution increases when σ increases. The error rates are similar to those found by PATHBUILD and GLMNET, but slightly higher than GLMNET even when σ is

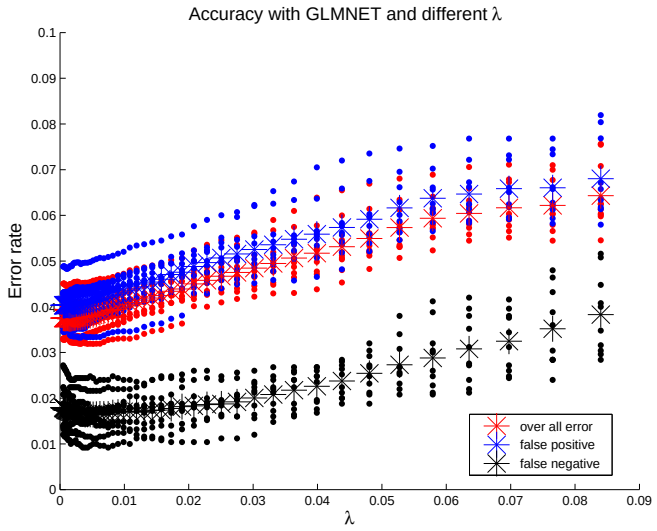

Figure 2: Error rate decreases when GLMNET is used to solve for the coefficients and the constraint parameter λ decreases.

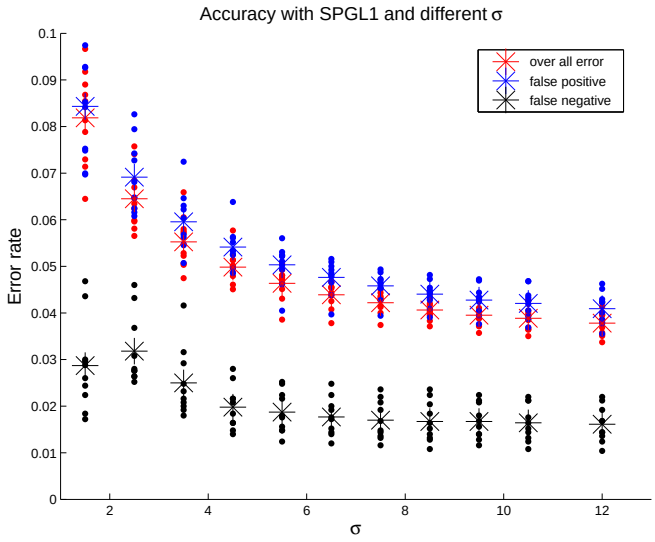

Figure 3: Error rate decreases when SPGL1 is used to solve for the coefficients and the constraint parameter σ increases.

large.

4.4 Rule Ensemble with FPC

The results of solutions generated by FPC are shown in Figure 4. They are roughly as accurate as the solutions generated with the previous solvers. FPC also has an explicit display of the thresholding as seen in Figure 5; the norm of the coefficients increases dramatically then asymptotically approaches a certain value. The asymptotic behavior is caused by the threshold constricting the coefficients and essentially preventing another coefficient from stepping off of zero. The thresholding is also seen in the error rate decreases as the weight on the mean squared error is increased, but stabilizes once the training set is reasonably fit. The value of μ where the error stabilizes is the value needed to build the model, but unfortunately it is not clear how to choose this value of μ *a priori*. The need for a selection of the penalty parameter is one of the difficulties that FPC, SPGL1, and GLMNET have. PATHBUILD shares a similar problem with the need to selection the gradient descent constriction parameter τ.

4.5 Identifying Important Attributes Via Rule Importance

Figure 4 shows that the rule ensemble method is quite successful at correctly classifying observations when all of the attributes are used to generate rules and build the model. Attributes have variable importance in the model and we suspect that not all of the 39 attributes in the full dataset are needed to model and correctly predict class labels. We want to use the rule ensemble method to select only the attributes that are important and save the expense of considering the other less important variables.

The importance of a rule is indicated by the magnitude of the coefficient for that rule. The larger a coefficient is in magnitude, the more important the corresponding rule is, as that rule will have a larger contribution to the model. To

Rule r_k	$\lvert a_k \rvert$
$x_2 \geq$ -0.315 & $x_{18} \geq$ 0.047	0.1045
$x_{29} <$-0.251	0.0725
$x_{23} \geq$ -0.606	0.0317
$x_1 <$ -0.324	0.0274
$x_{12} \geq$ 0.260	0.0193

Table 2: Example of ordering rules by importance. These are the five rules with greatest importance in the first model as selected by FPC with $\mu = 0.25$.

sift out the most important attributes, we look at which rules FPC considered important at different values of μ. Rules are ordered by the magnitude of their corresponding coefficient and if a rule is one of the 20 most important in a solution generated with a certain μ (we considered 13 values of μ), then that rule receives a vote. An example of ordering the rules is in Table 2 where the 5 most important rules from one test with a given μ are ordered. Figure 6 shows for how many values of μ each rule was considered to be one of the 20 most important; this indicates that certain rules are important in solutions with all values of μ tried, while others are considered important only when certain μ are used. This process is continued for 5 different cross-validation sets, which yields 5 sets of rules that were in the top 20 most important rules for at least one value of μ. The sets of rules are decomposed into sets of the attributes that were used to make up the rules in each set. Then we let the 5 repetitions vote on which attributes are needed to make the most influential rules and keep only the attributes that are in the set of important attributes for at least 3 out of the 5 repetitions. This set of attributes forms a smaller subset of the total attributes available in the initial dataset; it is the subset attributes that are used in at least one of the most important rules in at least 3 of the 5 repetitions.

For the supernova dataset, the smaller subset of attributes included only 21 of the 39 original attributes. Tests were

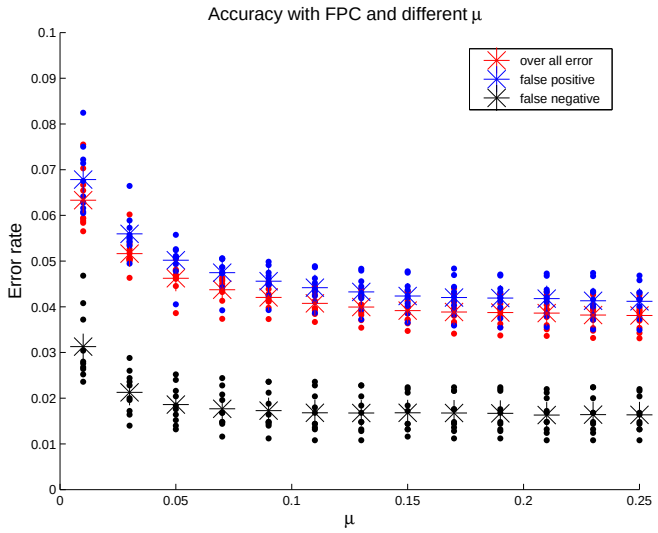

Figure 4: Error rate decreases when FPC is used and the weight on the risk or mean squared error is increased (μ increased).

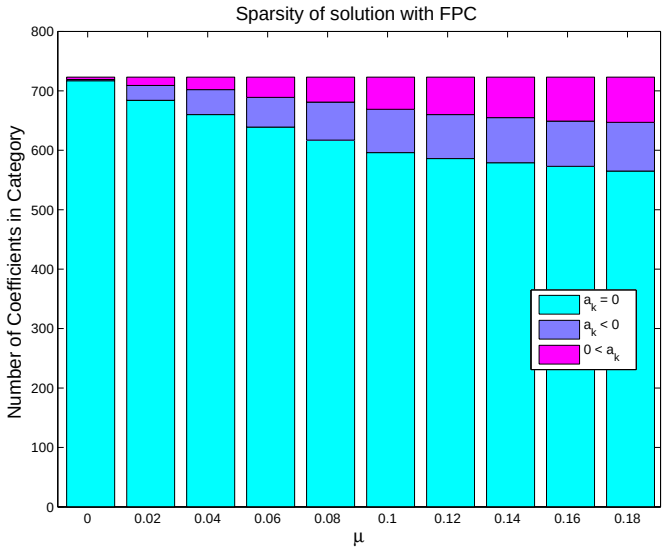

Figure 5: The sparsity of the solution is indicated by the number of coefficients that are equal to zero. As μ is increased, more terms are included in the model. The sparsity of the solution stops decreasing when μ is large. Here 78% of the coefficients are trivial when $\mu = 0.19$.

repeated using the same cross-validation sets and method parameters as were used in Figure 4, but using only the smaller subset of 21 attributes to train on rather than all 39 attributes. Figure 7 compares the error rate of the method when 21 attributes were used with the error rate of the method when all 39 attributes were used. The results show that the accuracy of the method improves when we reduce the number of attributes used in the model. The method successfully ranks rules and identifies more important attributes. The method loses accuracy when the less important features are included; in essence, the extra attributes act as noise. After the method identifies these attributes as less important and we remove them, the method is able to return an even more accurate model and the insight of which attributes are not adding predictive capability to the model. Garnering better accuracy with fewer attributes may allow the extra attributes to be excluded from the data collection, which will save time in collecting data, save space in storing data, and allow an overall better analysis.

5. RESULTS ON MULTI-CLASS PROBLEMS

Here we use the method described in section 2.3 to extend the rule ensemble to multi-class problems. To consider the overall utility of the method for classification problems, we compare the rule ensemble method on multi-class problems with two other common multi-class ensemble methods: bagging and boosting. We let bagging use 1000 trees and boosting use 50 trees in five 2-fold cross validations. Both bagging and boosting employ random forests for growing trees. These testing parameters are the same that were used in a previous comparison of ensemble methods [3]. Label predictions from ensembles of trees can be made by taking a vote of each tree's prediction or by averaging the predictions from all the trees. Results for both methods are presented. Minimal tuning of tree size, number of rules used, and constraint parameter τ was done to use the rule ensemble method on each dataset.

5.1 Comparison with Bagging in OVA Classification on Vehicle Dataset

Figure 8 compares using the rule ensemble and bagging on the vehicle dataset. Bagging here is used with binary trees in an OVA classification scheme rather than in its standard form, which uses multi-class decision trees. The error at predicting affinity to each class is in Figure 8 and shows that the rule ensemble beats bagging for the majority of the classes. Figure 8 also shows the varying level of success that the ensemble techniques had at predicting each class, which ensemble was better for a class was not consistent for all classes in a dataset. Some classes are easier to identify than others (e.g. "opel" is easier to distinguish than van).

5.2 Results on Multi-class Datasets

The results of the multiple class tests are given in Figures 9-10. The rule ensemble was much stronger than both tree ensembles if averaging of each tree's label prediction was used for classification. However, Figure 9 shows that if the trees voted on which class label is best, then the rule ensemble was better on some datasets, but not others. Voting was better at label prediction than averaging base learner predictions, but neither boosting nor bagging provided a universal win over the rule ensemble, as can be seen in Figure 9. What is not apparent in Figures 9-10 is that the rule ensemble was a much better predictor for binary labels than the tree ensembles. This result is apparent in Figure 8 where nearly every individual class is better predicted by the rule ensemble method. Figure 10 shows the accuracy of the rule ensemble method with different coefficient solvers. Some datasets are easier to classify (larger percent of data correctly classified) while others, such as the #2 dataset glass, were more difficult to classify for all the methods. No solver was universally better for finding the rule weights. Each solver returned a

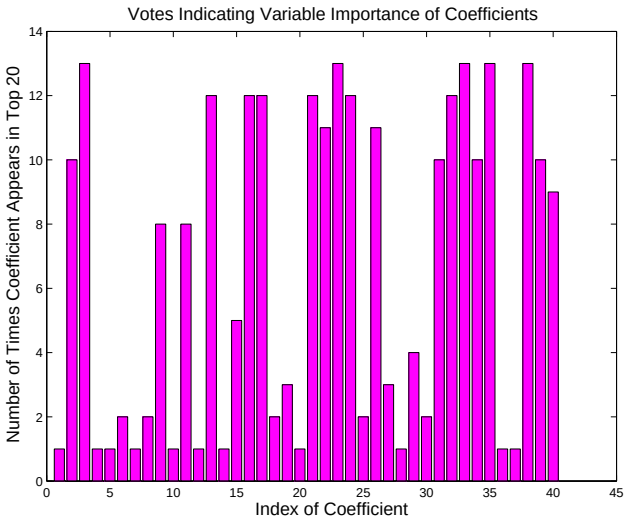

Figure 6: This histogram shows how many times a rule was one of the top 20 most important rules in a solution. Solutions were generated at each of 13 different values of μ, as shown in Figure 4. Rules that received 13 votes were one of 20 most influential rules for every value of μ tried. Only rules that were in the top 20 most influential for at least one solution are shown. The attributes used to compose these rules were used to find a smaller subset of attributes to train on for the results in Figure 7.

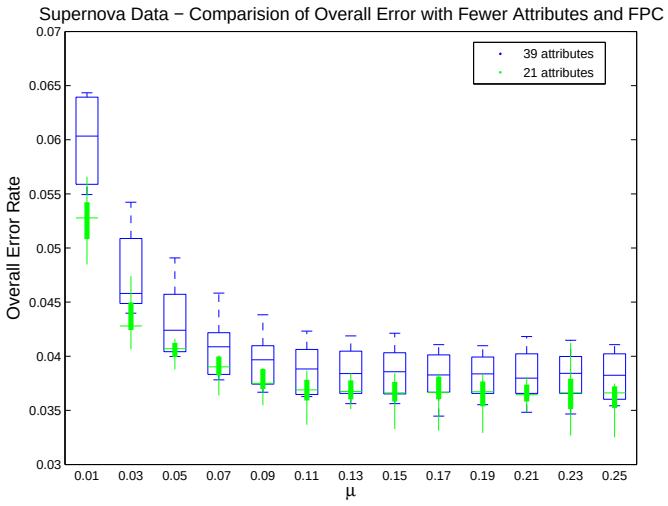

Figure 7: Comparison of overall error rate when fewer attributes are used. The preliminary tests used all 39 attributes in the dataset. The subsequent tests used only the subset of 21 attributes that were used to construct the most important rules in the preliminary tests. Using the restricted set of attributes gives a lower error rate indicating that the rule ensemble method successfully identified important attributes in the dataset.

sparse solution, but there was little consistency in which solver returned the sparest solution. Some solvers returned the sparest solution on one dataset, but not on another. The only general trend was that PATHBUILD usually returned the least sparse solution. The constraint parameter τ could be adjusted to return a solution that had a comparably few number of rules to the other solvers, but at the expense of accuracy. PATHBUILD had to include many more rules in the final model to yield a model that had comparable accuracy to models that had coefficients generated with one of the other solvers. It is not yet clear how to choose between the other solvers for a given dataset.

6. PREVIOUS WORK

Rule ensembles have received a lot of attention, partially due to their successes demonstrated in several applications. The original Friedman and Popescu method was used in both a study of chemical mutagenicity [23] and a search for super symmetric particles in data from the Large Hadron Collider [9]. The version of the rule ensemble proposed by Friedman and Popescu has also been included in larger machine learning tools [21].

Various rule generation schemes that focus on how to quickly grow or generate non-overlapping rules have been proposed. SLIPPER [8] and LRI [33] are two methods that build an ensemble of boosted rules and incorporate a re-weighting of rules. A more recent method proposed by Rückert and Kramer [28] uses a regularized regression to combine rules into a classifier. This method is very different from the SLIPPER, LRI, and the rule ensemble considered here as the initial rules are simply defined, without reference to the labels, and then the regression method is used to select and re-weight the predefined rules. Another method proposed generates individual rules by greedily minimizing the negative log likelihood [11] and has been used on multi-class problems, but does not offer the interpretability of rule weights. Another group proposed to generate simple classifiers as rules [4], but not use any weighting to combine the rules into a linear model. This method was modified to accommodate missing data [5], but was not applied to multi-class dataset and still did not use sophisticated rule weighting.

The advantage of the rule ensemble used here is that it generates rules that reflect behavior in the dataset and then uses a regularized regression to eliminate excess rules and indicate more important rules and features. The difference between the previous work and our work here is that all of the previous proposed methods have looked at how to generate rules, while we focus on how to combine them into a model. Regression methods for calculating appropriate weights have been developed, but they have not been studied in this application of rule ensembles. Further, individual applications of Friedman's and Popescu's method have not been able to apply the method to multi-class methods. As far as we can determine, this paper is the first time that this more modern rule ensemble method has been tested on multi-class datasets.

Both the rule generation phase and the rule weighting phase are important to the outcome of the model. There has been a lot of focus on how to build rules, but here we look at different methods for weighting the rules and see how sen-

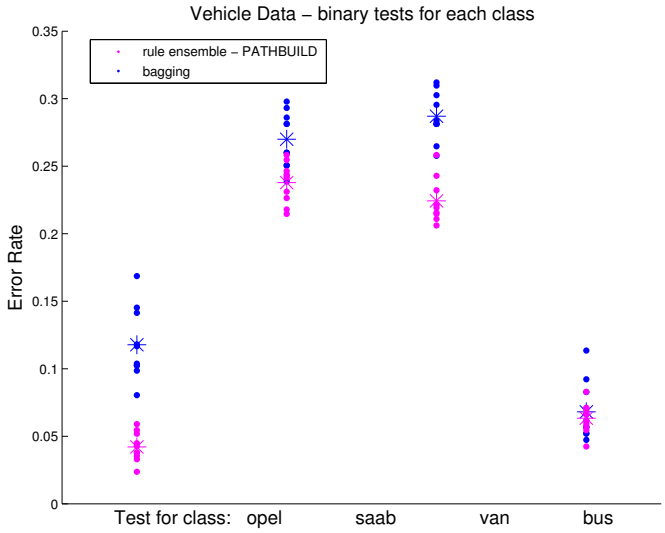

Figure 8: Binary tests on each class of vehicle data. Figure shows accuracy when using bagging in an OVA classification method rather than with multi-class decision trees.

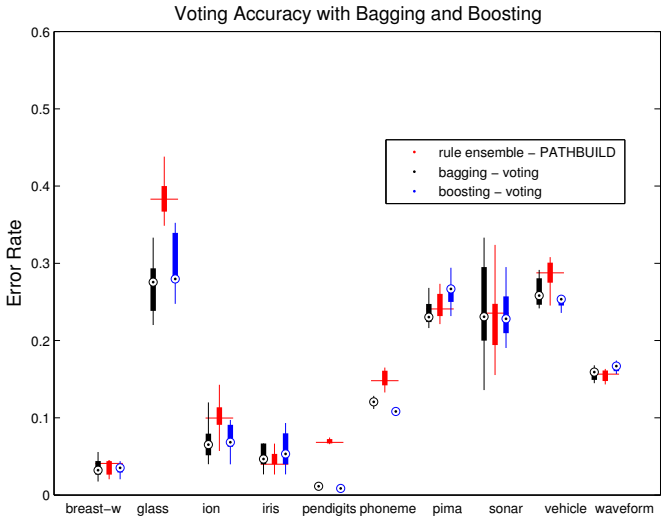

Figure 9: Comparison of the misclassification rate of the rule ensemble method with that of bagging and boosting ensemble methods when voting is used.

sitive the final model is to the method used for rule weighting. Finally, we look at a different feature selection scheme than was suggested [19] and empirically show that this rule ensemble can be sued to reduce the number of features in a dataset without adversely affecting the accuracy of the model. While previous studies have used the rule ranking to interpret the importance of features [19], [23], [9], we use the rule ranking to actually select features and reduce the dimension of the dataset.

7. CONCLUSIONS

We examined four different methods to find coefficients to assemble rules into a linear model. All 4 methods present the challenge of needing to select a constraint parameter that controls the sparsity/accuracy trade-off of the solution that they return. If each parameter is chosen correctly, then the methods are capable of producing coefficients that allow for similar accuracy in the model. The different approaches that the methods take for finding the coefficients do result in slightly different rankings of the rules. The difference in coefficients that each method considers important is shown in Figure 11. Ideally all solvers would select the same terms to be the most significant and would order the terms by importance the same way. Figure 11 shows that some rules that one method considers important are not considered to be important to another method. FPC and SPGL1 order coefficients similarly, which is indicated by SPGL1 giving a significant magnitude to coefficients that FPC also gives a significant magnitude to. GLMNET's and PATHBUILD's ordering share less similarity with FPC and SPGL1, as indicated by coefficients such as 9 and 18 that GLMNET and PATHBUILD give a significant magnitude to, but both FPC and SPGL1 give small values to. The difference in methods is also reflected in the sparsity of the solutions that they return. To achieve similar accuracy (taken here at 96% accuracy), PATHBUILD returns a solution with 40-50% of the coefficients non-zero while the other methods return much sparser solutions that have only 12-19% of the coefficients non-zero. In general, SPGL1 returned the sparsest solutions and PATHBUILD returned the least sparse solutions for models with similar error rates.

We also showed the utility of the rule ensemble method for identifying important attributes in a dataset containing images of potential supernovas. The rule ensemble method has the benefit over tree methods of providing insight into a dataset by returning weighted rules. Rules with large weights have a larger effect on the model and thus can be thought of as more important than other rules. We used the importance of such rules to alert us to the more significant features in the dataset by looking at which features the important rules are defined on. This technique allowed us to select 21 attributes out of the 39 available and reduce the error rate of the model by building models only on the reduced set of attributes. Traditional algorithms that use ensembles of decision trees, such as boosting and bagging, aren't able to provide this insight into the importance of certain variables of a dataset because they do not rank or weight of rules.

As a final step, we extended the rule ensemble method to multi-class problems and compared it with two well-known tree ensemble methods, namely boosting and bagging. We found that extending the rule ensemble to work on multi-class problem with an OVA-inspired technique, the rule ensemble method performed comparably to the tree methods on a set of 10 classical datasets. This result highlights the power of the rule ensemble method, as we had expected the tree ensemble methods to do better on multi-class problems. Tree ensembles can use multi-class decision trees, which provide what one would think is a more natural extension to multi-class problems than using the OVA method. However, the rule ensemble method returned comparable rates of accuracy on most datasets and even performed better on some of the datasets. The discrepancy between the tree en-

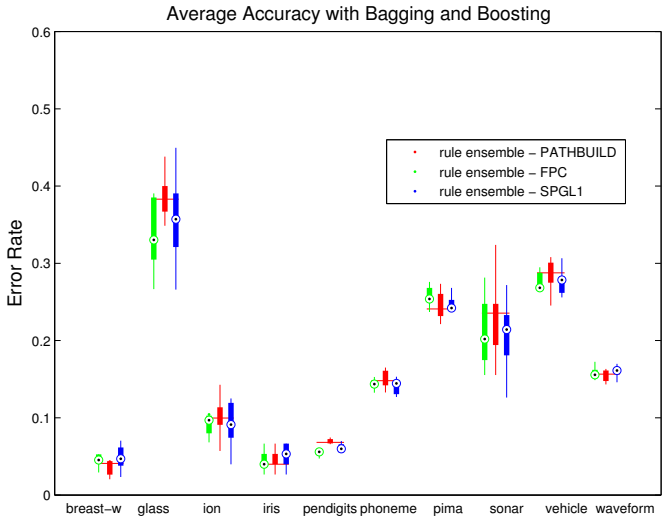

Figure 10: Comparison of the error rate on UCI datasets from models that were built with the rule ensemble method using different solvers to find the coefficients a.

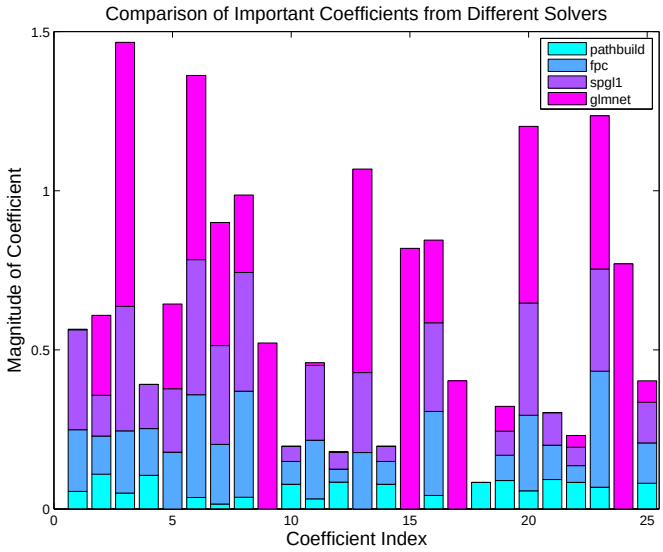

Figure 11: Length of bars indicate the magnitude of coefficients as calculated by different solvers. Only the coefficients with the 10 largest magnitudes from each solver are displayed. Coefficients plotted come from solutions that yielded similar error rates: $\tau = 0.4$, $\mu = .11$, $\sigma = 8.5$, $\lambda = 0.014$.

sembles with voting and the rule ensemble was larger on problems that had a relatively large number of labels, such as the pendigits dataset, which had the most labels out of all the datasets, than on datasets with fewer labels. To improve the accuracy of the rule ensemble on problems with multiple classes, we would like to try using multi-class decision trees to build the rules and then relabel the nodes for each binary problem. This technique might yield better rules as it would allow for differentiation between the classes in the rule building phase. Better rules would then allow for a clearer separation of binary labels in the regression phase. This technique would also make the training phase more efficient, as it would only require one set of rules to be constructed rather the as many sets of rules as there are classes.

The rule ensemble method has the advantage over some other methods by being able to identify relationships and hierarchies between variables to a certain extent when building the decision trees. The rules in the decision trees get more complex the deeper the tree is grown and also are able to have limited support in the parameter space, so they only affect certain observations that fall in that space. By including more variables, complex rules can be seen as resembling discrete correlations, and the post-processing of the rules allows for overly simplified correlations (that precede more complex rules in depth) to be removed from the model. The post-processing also allows for overly complex rules to be pruned from the model. Thus some variable interactions can be captured by the rule ensemble method without any *a priori* assumption that they exist, as is needed in standard regression models, and excessive computation is not spent considering correlations that do not exist.

We do not compare the computational efficiency of the rule ensemble method with tree ensemble methods here, since it is currently written in Matlab^{TM}, while the tree ensemble methods used are written in C. However, we do not expect that the rule ensemble method will reduce the amount of time necessary for the training portion of the algorithm to run because it must perform the coefficient solving method in addition to the tree growing. If the rule ensemble method is able to prune a substantial number of repetitive or unnecessary rules, then it is likely to run substantially more quickly than the tree methods. Comparing the time efficiency of the rule ensemble with other tree methods and other machine learning techniques will be part of future work. We do not present the computational efficiency of the coefficient solving methods used in the rule ensemble method for the same reason. Each solver is written in a different programming language, and each will have to be implemented in the same language and level of optimization before a meaningful study can be performed.

Acknowledgements

We would like to thank Sean Peisert and Peter Nugent for their valuable comments and suggestions.

8. REFERENCES

[1] S. Bailey and C. Aragon et al. How to find more supernovae with less work: Object classification techniques for difference imaging. *The Astrophysical Journal*, 665:1246, 2007.

[2] R.E. Banfield. The OpenDT project. Technical report, University of South Florida, 2003. `http://opendt.sourceforge.net/`.

[3] R.E. Banfield, L.O. Hall, W.P. Kegelmeyer K.W. Bowyer, D. Bhadoria, and S. Eschrich. A comparison of ensemble creation techniques. *Lecture Notes in Computer Science*, 3077:223–32, 2004.

[4] J. Błaszczyński and K. Dembczyński et al. Ensembles of decision rules. *Foundations of Computing and Decision Sciences*, 31(3-4):221–232, 2006.

[5] J. Błaszczyński and et al. K. Dembczyński. Ensembles of decision rules for solving binary classification problems in the presence of missing values. In *Rough Sets and Current Trends in Computing*, pages 224–234. Springer, 2006.

[6] J.S. Bloom and J.W. Richards et al. Automating discovery and classification of transients and variable stars in the synoptic survey era. *Arxiv preprint arXiv:1106.5491*, 2011. `http://arxiv.org/abs/1106.5491`.

[7] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and regression trees*. Chapman & Hall/CRC, 1998.

[8] W.W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proceedings of the National Conference on Artificial Intelligence*, pages 335–342. JOHN WILEY & SONS LTD, 1999.

[9] J. Conrad and F. Tegenfeldt. Applying rule ensembles to the search for super-symmetry at the large hadron collider. *Journal of High Energy Physics*, 2006:040, 2006.

[10] O.A. DeMasi, J.C. Meza, and D.H. Bailey. Dimension reduction using rule ensemble machine learning methods: A numerical study of three ensemble methods. 2011. `http://crd.lbl.gov/~dhbailey/dhbpapers/Ensemble_TechReport.pdf`.

[11] K. Dembczyński, W. Kotłowski, and R. Słowiński. Maximum likelihood rule ensembles. In *Proceedings of the 25th international conference on Machine learning*, pages 224–231. ACM, 2008.

[12] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 4(2):139–57, 2000.

[13] D.L. Donoho, I.M. Johnstone, G. Kerkuacharian, and D. Picard. Wavelet shrinkage; asymptopia? (with discussion). *Journal Royal Statistical Society*, 57(2):201–37, 1995.

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

[15] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33, 2010.

[16] J.H. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals ofApplied Statistics*, 1(2):302–32, 2007.

[17] J.H. Friedman and B.E. Popescu. Importance sampled learning ensembles. Technical report, Department of Statistics Stanford University, 2003. `http://www-stat.stanford.edu/~jhf/ftp/isle.pdf`.

[18] J.H. Friedman and B.E. Popescu. Gradient directed regularization. Technical report, Department of Statistics Stanford University, 2004. `http://www-stat.stanford.edu/~jhf/ftp/pathlite.pdf`.

[19] J.H. Friedman and B.E. Popescu. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2(3):916–54, 2008.

[20] E.T. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for l_1-regularized minimization with applications to compressed sensing. Technical report, Rice University CAAM, 2007. `http://www.caam.rice.edu/~zhang/reports/tr0707.pdf`.

[21] A. Hoecker and P. Speckmayer et al. TMVA-toolkit for multivariate data analysis. *Arxiv preprint physics/0703039*, 2007.

[22] C.W. Hsu and C.J. Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–25, 2002.

[23] J.J. Langham and A.N. Jain. Accurate and interpretable computational modeling of chemical mutagenicity. *Journal of chemical information and modeling*, 48(9):1833–1839, 2008.

[24] N.M. Law and S.R. Kulkarni et al. The Palomar Transient Factory: System Overview, Performance, and First Results. *Publications of the Astronomical Society of the Pacific*, 121:1395–408, 2009. `http://adsabs.harvard.edu/abs/2009PASP..121.1395L`.

[25] J. Quinlan. Bagging, boosting, and C4.5. *Proceedings Thirteenth American Association for Artificial Intelligence National Conference on Artificial Intelligence*, pages 725–30, 1996.

[26] A. Rau and S.R. Kulkarni et al. Exploring the Optical Transient Sky with the Palomar Transient Factory. *Publications of the Astronomical Society of the Pacific*, 121:1334–51, 2009. `http://adsabs.harvard.edu/abs/2009PASP..121.1334R`.

[27] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–41, 2004.

[28] U. Rückert and S. Kramer. A statistical approach to rule learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 785–792. ACM, 2006.

[29] A.C. Tan, D. Gilbert, and Y. Deville. Multi-class protein fold classification using a new ensemble machine learning approach. *Genome informatics series*, pages 206–17, 2003.

[30] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–88, 1996.

[31] E. van den Berg and M. P. Friedlander. SPGL1: A solver for large-scale sparse reconstruction, 2007. `http://www.cs.ubc.ca/labs/scl/spgl1`.

[32] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008. `http://link.aip.org/link/?SCE/31/890`.

[33] S. Weiss, S.M. Weiss, and N. Indurkhya. Lightweight rule induction. 2000.