SLAC-PUB-12247

SEPARATING SIGNAL FROM BACKGROUND USING ENSEMBLES OF RULES

JEROME H. FRIEDMAN

Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94305
E-mail: jhf@stanford.edu

Machine learning has emerged as a important tool for separating signal events from associated background in high energy particle physics experiments. This paper describes a new machine learning method based on ensembles of rules. Each rule consists of a conjuction of a small number of simple statements ("cuts") concerning the values of individual input variables. These rule ensembles produce predictive accuracy comparable to the best methods. However their principal advantage lies in interpretation. Because of its simple form, each rule is easy to understand, as is its influence on the predictive model. Similarly, the degree of relevance of each of the respective input variables can be assessed. Graphical representations are presented that can be used to ascertain the dependence of the model jointly on the variables used for prediction.

1. Introduction

Predictive learning is a common application in data mining, machine learning and pattern recognition. The purpose is to predict the unknown value of an attribute y of a system under study, using the known joint values of other attributes $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ associated with that system. The prediction takes the form $\hat{y} = F(\mathbf{x})$, where the function $F(\mathbf{x})$ maps a set of joint values of the "input" variables $\mathbf{x}$ to a value $\hat{y}$ for the "output" variable y. The goal is to produce an accurate mapping. Lack of accuracy is defined by the prediction "risk"

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

where $L(y, \hat{y})$ represents a loss or cost for predicting a value $\hat{y}$ when the actual value is y, and the expected (average) value is over the joint distribution of all variables $(\mathbf{x}, y)$ for the data to be predicted.

As an example consider the problem of separating signal from background events in a high energy particle physics experiment. Here the outcome attribute y for each event has one of two values $y \in \{signal, background\}$. The attributes $\mathbf{x}$ used for prediction are the variables measured from each event, perhaps augmented with various quantities constructed from these measurements. The prediction $\hat{y}$ also realizes one of the two values $\hat{y} \in \{signal, background\}$. A natural loss function for this two–class classification problem would be

$$L(y, \hat{y}) = \begin{cases} L_S & \text{if } \; y = signal \;\&\; \hat{y} = background \\ L_B & \text{if } \; y = background \;\&\; \hat{y} = signal \end{cases} \tag{2}$$

with $L(y, \hat{y}) = 0$ for correct predictions. Here L_S and L_B are the respective user specified costs for misclassifying signal and background events for the particular problem. The goal is to construct a mapping function $F(\mathbf{x})$ that given (2) minimizes the prediction risk (1).

Although the loss function (2) characterizes the actual goal, it cannot be directly used to construct classification functions $F(\mathbf{x})$ with most machine learning procedures. The problem is that with this loss criterion the associated risk (1) is not a continuous function of the parameters associated with the predicting function $F(\mathbf{x})$. This excludes the application of numerical optimization techniques in the search for a good solution, requiring instead far more costly combinatorial optimization methods.

In order to apply numerical optimization techniques one must approximate the discrete loss (2) with a smooth continuous one that produces the same solution, at least in the limit of infinite amount of data. For finite data sets the hope is that the solutions will be similar enough to be useful. One scores the signal events with the numerical value $y = 1$ and the background with $y = -1$. In this case the predicting function $F(\mathbf{x})$ produces a numerical score that estimates a monotone function of the probability that $y = 1$ (signal event) given the joint values of the predictor variables $\mathbf{x}$; that is, $F(\mathbf{x}) = m(\Pr[y = 1 \,|\, \mathbf{x}])$ where $m(\eta)$ is a monotonically increasing function of its argument η. Classification is accomplished by thresholding this score at an appropriate value t

$$\begin{aligned} F(\mathbf{x}) \geq t &\Rightarrow signal \\ F(\mathbf{x}) < t &\Rightarrow background. \end{aligned} \tag{3}$$

The value chosen for the threshold t is the one

Contributed to PHYSTATO5: Statistical Problems in Particle Physics, Astrophysics and Cosmology,
12-15 September 2005, Oxford, England, United Kingdom

Work supported in part by the US Department of Energy contract DE-AC02-76SF00515

that minimizes the prediction risk (1) using (2) and thereby depends on the values chosen for L_S and L_B.

Within this framework a variety of smooth surrogate loss functions have been proposed in the statistics and machine learning literatures. A commonly used criterion is squared–error loss $L(y, \hat{y}) = (y - \hat{y})^2$. In this case the predicting score function approximates $F(\mathbf{x}) = 2 \cdot \Pr[y = 1 \,|\, \mathbf{x}] - 1$. Other popular choices include $L(y, \hat{y}) = \log(1 + e^{-y \cdot \hat{y}})$ used by logistic regression in statistics, and $L(y, \hat{y}) = e^{-y \cdot \hat{y}}$ used by the AdaBoost boosting procedure (Freund and Schapire 1996) from machine learning. For these latter two loss functions the numerical score estimates the log–odds

$$F(\mathbf{x}) = \log \frac{\Pr[y = 1 \,|\, \mathbf{x}]}{1 - \Pr[y = 1 \,|\, \mathbf{x}]}.$$

Given a particular smooth surrogate $L(y, \hat{y})$, the optimal mapping ("target") function $F^*(\mathbf{x})$ is defined as the one that minimizes the prediction risk (1) over all possible functions

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{\mathbf{x}y} L(y, F(\mathbf{x})). \tag{4}$$

This optimal predicting function is unknown because the distribution of the joint values of the variables $(\mathbf{x}, y)$, $p(\mathbf{x}, y)$, is unknown.

With the machine learning approach one has a data base of previously solved cases $T = \{\mathbf{x}_i, y_i, w_i\}_1^N$, called a training sample, containing known signal and background events. Here $\mathbf{x}_i$ represents the measurement variables associated with the ith event. Each signal event is assigned the value $y_i = 1$ and the background events are assigned $y_i = -1$. Each event also has a weight w_i that depends on its type; signal events are assigned weights

$$w_i = L_S \ \pi_S \,/\, N_S$$

where L_S is the cost for misclassifying a signal event (2), π_S is the fraction of signal events in *future* data to be predicted, and N_S is the total number of signal events in the training data T. Each background event receives a weight

$$w_i = L_B \ \pi_B \,/\, N_B$$

where L_B, π_B, and N_B are the corresponding quantities for the background. With this weighting the classification threshold (3) that minimizes prediction risk is $t = 0$.

These weighted training data are presumed to represent a random sample drawn from the distribution of future data to be predicted. A machine learning procedure is then applied to these training data to derive an approximation $F(\mathbf{x})$ to $F^*(\mathbf{x})$ (4). This approximation will be used to score and then classify (3) future events given only their measured variables $\mathbf{x}$. The extent to which this $F(\mathbf{x})$ so derived provides a useful approximation to $F^*(\mathbf{x})$ will depend on the nature of $F^*(\mathbf{x})$, the training sample size N, and the particular machine learning procedure employed. Different procedures are appropriate for different target functions and/or different sample sizes.

2. Ensemble learning

Learning ensembles have emerged as being among the most powerful machine learning methods (see Breiman 1996 & 2001, Freund and Schapire 1996, Friedman 2001). Their structural model takes the form

$$F(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m f_m(\mathbf{x}) \tag{5}$$

where M is the size of the ensemble and each ensemble member ("base learner") $f_m(\mathbf{x})$ is a different function of the input variables $\mathbf{x}$ derived from the training data. Ensemble predictions $F(\mathbf{x})$ are taken to be a linear combination of the predictions of each of the ensemble members, with $\{a_m\}_0^M$ being the corresponding parameters specifying the particular linear combination. Ensemble methods differ in choice of particular base learners (function class), how they are derived from the data, and the prescription for obtaining the linear combination parameters $\{a_m\}_0^M$.

All popular ensemble methods use variants of the following generic procedure to generate the base learners used in (5). Each base learner is taken to be a simple function of the predictor variables characterized by a set of parameters $\mathbf{p} = (p_1, p_2, \cdots)$. That is,

$$f_m(\mathbf{x}) = f(\mathbf{x}; \mathbf{p}_m) \tag{6}$$

where $\mathbf{p}_m$ represents a specific set of joint parameter values indexing a specific function $f_m(\mathbf{x})$ from the parameterized class $f(\mathbf{x}; \mathbf{p})$. Particular choices for such parameterized function classes are discussed below. Given a function class the individual members

of the ensemble are generated using the prescription presented in Algorithm 1.

Algorithm 1

Ensemble generation

1 $F_0(\mathbf{x}) = 0$
2 For $m = 1$ to M {
3 $\mathbf{p}_m =$
$\arg\min_{\mathbf{p}} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + f(\mathbf{x}_i; \mathbf{p}))$
4 $f_m(\mathbf{x}) = f(\mathbf{x}; \mathbf{p}_m)$
5 $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot f_m(\mathbf{x})$
6 }
7 ensemble $= \{f_m(\mathbf{x})\}_1^M$

In line 3, $S_m(\eta)$ represents a different subsample of size $\eta < N$ randomly drawn without replacement from the original training data, $S_m(\eta) \subset \{\mathbf{x}_i, y_i\}_1^N$. As discussed in Friedman and Popescu 2003, smaller values of η encourage increased dispersion (less correlation) among the ensemble members $\{f_m(\mathbf{x})\}_1^M$ by training them on more diverse subsamples. Smaller values also reduce computation by a factor of N/η.

At each step m, the "memory" function

$$F_{m-1}(\mathbf{x}) = F_0(\mathbf{x}) + \nu \cdot \sum_{k=1}^{m-1} f_k(\mathbf{x})$$

contains partial information concerning the previously induced ensemble members $\{f_k(\mathbf{x})\}_1^{m-1}$ as controlled by the value of the "shrinkage" parameter $0 \leq \nu \leq 1$. At one extreme, setting $\nu = 0$ causes each base learner $f_m(\mathbf{x})$ to be generated without reference to those previously induced, whereas the other extreme $\nu = 1$ maximizes their influence. Intermediate values $0 < \nu < 1$ vary the degree to which previously chosen base learners effect the generation of each successive one in the sequence.

Several popular ensemble methods represent special cases of Algorithm 1. A "bagged" ensemble (Breiman 1996) is obtained by using squared–error loss, $L(y, \hat{y}) = (y - \hat{y})^2$, and setting $\nu = 0$, and $\eta = N/2$ or equivalently choosing S_m (line 3) to be a bootstrap sample (Friedman and Hall 1999). Random forests (Breiman 2001) introduce increased ensemble dispersion by additionally randomizing the algorithm ("arg min", line 3) used to solve for the ensemble members (large decision trees). In both cases the coefficients in (5) are set to $a_0 = \bar{y}$, $\{a_m = 1/M\}_1^M$ so that predictions are a simple average of those of the ensemble members. AdaBoost (Freund and Schapire 1996) uses exponential loss, $L(y, \hat{y}) = \exp(-y \cdot \hat{y})$ for $y \in \{-1, 1\}$, and is equivalent to setting $\nu = 1$ and $\eta = N$ in Algorithm 1. Predictions are taken to be the sign of the final memory function $F_M(\mathbf{x})$. MART (Friedman 2001) allows a variety of loss criteria $L(y, \hat{y})$ for arbitrary y, and in default mode sets $\nu = 0.1$ and $\eta = N/2$. Predictions are given by $F_M(\mathbf{x})$.

Friedman and Popescu 2003 experimented with a variety of joint (ν, η) values for generating ensembles of small decision trees, followed by a regularized regression to estimate the linear combination parameters $\{a_j\}_0^M$ (5). Given a set of base learners $\{f_m(\mathbf{x})\}_1^M$ the parameters of the linear combination are obtained by a regularized linear regression on the training data $\{\mathbf{x}_i, y_i, w_i\}_1^N$

$$\{\hat{a}_m\}_0^M = \arg\min_{\{a_m\}_0^M} \sum_{i=1}^N w_i \, L\left(y_i \, , a_0 + \sum_{m=1}^M a_m f_m(\mathbf{x}_i)\right) + \lambda \cdot \sum_{m=1}^M |\, a_m \,|. \quad (7)$$

The first term in (7) measures the prediction risk (1) on the training sample, and the second (regularization) term penalizes large values for the coefficients of the base learners. The influence of this penalty is regulated by the value of $\lambda \geq 0$. It is well known that for this ("lasso") penalty, larger values of λ produce more overall shrinkage as well as increased dispersion among the values $\{|\, \hat{a}_m \,|\}_1^M$, often with many being set to zero (see Tibshirani 1996, Donoho *et al.* 1995). Its value is taken to be that which minimizes an estimate of future prediction risk (1) based on a separate sample not used in training, or by full (multi–fold) cross–validation. Fast algorithms for solving (7) for all values of $\lambda \geq 0$, using a variety of loss functions $L(y, \hat{y})$, are presented in Friedman and Popescu 2004. Empirical results presented in Friedman and Popescu 2003 indicated that small but nonzero values of ν ($\nu \simeq 0.01$) performed best in this context. Results were seen to be fairly insensitive to the value chosen for η provided it was small ($\eta \lesssim N/2$) and grew less rapidly than the total sample size N ($\eta \sim \sqrt{N}$) as N becomes large ($N \gtrsim 500$).

Although in principle most of these procedures can be used with other base learners, they have almost exclusively been applied with decision trees (Breiman, *et al* 1983, Quinlan 1993). This is due

to the attractive properties of trees in data mining applications, and the existence of fast algorithms for inducing decision tree ensembles.

3. Rule based ensembles

The base learners considered here are simple rules. Let S_j be the set of all possible values for input variable x_j, $x_j \in S_j$, and s_{jm} be a specified subset of those values, $s_{jm} \subseteq S_j$. Then each base learner takes the form of a conjunctive rule

$$r_m(\mathbf{x}) = \prod_{j=1}^{n} I(x_j \in s_{jm}) \tag{8}$$

where $I(\delta)$ is an indicator of the truth of its logical argument; $I(\delta) = 1$ if δ is true and $I(\delta) = 0$ if δ is false. Each such base learner (8) assumes two values $r_m(\mathbf{x}) \in \{0, 1\}$. It is nonzero when all of the input variables realize values that are simultaneously within their respective subsets $\{x_j \in s_{jm}\}_1^n$. For variables that assume orderable values the subsets are taken to be contiguous intervals

$$s_{jm} = (t_{jm}, u_{jm}]$$

defined by a lower and upper limit, $t_{jm} < x_j \leq u_{jm}$. For categorical variables assuming unorderable values (names) the subsets are explicitly enumerated. Such rules (8) can be regarded as parameterized functions of $\mathbf{x}$ (6) where the parameters $\mathbf{p}_m$ are the quantities that define the respective subsets $\{s_{jm}\}$.

Note that for the case in which the subset of values s_{jm} (real or categorical) appearing in a factor of (8) is in fact the entire set $s_{jm} = S_j$, the corresponding factor can be omitted from the product. In this case the rule can be expressed in the simpler form

$$r_m(\mathbf{x}) = \prod_{s_{jm} \neq S_j} I(x_j \in s_{jm}). \tag{9}$$

The particular input variables x_j for which $s_{jm} \neq S_j$ are said to be those that "define" the rule $r_m(\mathbf{x})$. As an example, the rule

$$r_m(\mathbf{x}) = \begin{cases} I(18 \leq \text{age} < 34) \\ \cdot I(\text{marital status} \in \{\text{single}, \\ \qquad \text{living together–not married}\}) \\ \cdot I(\text{householder status} = \text{rent}) \end{cases}$$

is defined by three variables, and a nonzero value increases the odds of frequenting bars and night clubs. In high energy physics applications each rule (9) can be interpreted as an intersection of "cuts" on the variables that define the rule.

3.1. *Rule generation*

One way to attempt to generate a rule ensemble is to let the base learner $f(\mathbf{x}; \mathbf{p})$ appearing in Algorithm 1 take the form of a rule (8) and then try to solve the optimization problem on line 3 for the respective variable subsets $\{s_{jm}\}$. Such a (combinatorial) optimization is generally infeasible for more that a few predictor variables although fast approximate algorithms might be derived. The approach used here is to view a decision tree as defining a collection of rules and take advantage of existing fast algorithms for producing decision tree ensembles. That is, decision trees are used as the base learner $f(\mathbf{x}; \mathbf{p})$ in Algorithm 1. Each node (interior and terminal) of each resulting tree $f_m(\mathbf{x})$ produces a rule of the form (9).

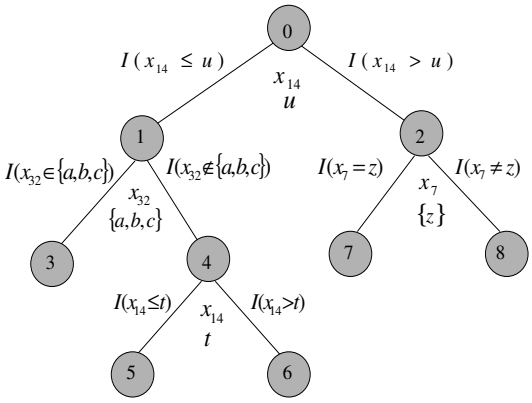

Fig. 1. A typical decision tree with five terminal nodes as described in the text.

This is illustrated in Fig. 1 which shows a typical decision tree with five terminal nodes that could result from using a decision tree algorithm in conjunction with Algorithm 1. Associated with each interior node is one of the input variables x_j. For variables that realize orderable values a particular value of that variable ("split point") is also associated with the node. For variables that assume unorderable categorical values, a specified subset of those values replaces the split point. For the tree displayed in Fig. 1

nodes 0 and 4 are associated with orderable variable x_{14} with split points u and t respectively, node 1 is associated with categorical variable variable x_{32} with subset values $\{a, b, c\}$, and node 2 is associated with categorical variable x_7 with the single value $\{z\}$.

Each edge of the tree connecting a "parent" node to one of its two "daughter" nodes represents a factor in (9) contributing to the rules corresponding to all descendent nodes of the parent. These factors are shown in Fig. 1 for each such edge. The rule corresponding to any node in the tree is given by the product of the factors associated with all of the edges on the path from the root to that node. Note that there is no rule corresponding to the root node. As examples, in Fig. 1 the rules corresponding to nodes 1, 4, 6, and 7 are respectively:

$$r_1(\mathbf{x}) = I(x_{14} \leq u)$$
$$r_4(\mathbf{x}) = I(x_{14} \leq u) \cdot I(x_{32} \notin \{a, b, c\})$$
$$r_6(\mathbf{x}) = I(t < x_{14} \leq u) \cdot I(x_{32} \notin \{a, b, c\})$$
$$r_7(\mathbf{x}) = I(x_{14} > u) \cdot I(x_7 = z).$$

3.2. *Rule fitting*

The collection of all such rules derived from all of the trees $\{f_m(\mathbf{x})\}_1^M$ produced by Algorithm 1 constitute the rule ensemble $\{r_k(\mathbf{x})\}_1^K$. The total number of rules is

$$K = \sum_{m=1}^{M} 2(t_m - 1) \tag{10}$$

where t_m is the number of terminal nodes for the mth tree. The predictive model is

$$F(\mathbf{x}) = \hat{a}_0 + \sum_{k=1}^{K} \hat{a}_k r_k(\mathbf{x}) \tag{11}$$

with

$$\{\hat{a}_k\}_0^K = \arg\min_{\{a_k\}_0^K} \sum_{i=1}^{N} w_i \, L\left(y_i \, , a_0 + \sum_{k=1}^{K} a_k r_k(\mathbf{x}_i)\right) + \lambda \cdot \sum_{k=1}^{K} |\, a_k \,|. \tag{12}$$

Fast algorithms for solving (12) for all values of $\lambda \geq 0$, and procedures for choosing a value for λ, are discussed in Friedman and Popescu 2004.

4. Rule based interpretation

The most important aspect of any predictive function $F(\mathbf{x})$ is its accuracy on future data as reflected by its prediction risk (1). Results from Friedman and Popescu 2005 suggest that rule based ensembles (11) (12) provide accuracy competitive with the best methods. However, accuracy is not the only desirable property of a predictive model. Often it is useful to be able to interpret the model to gain an understanding of how the respective input variables $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ are being used to formulate predictions. This information can be used to perform "sanity checks" to see if the model is consistent with one's *a priori* domain knowledge, and to gain an *a posteriori* understanding of the system that produced the data. Such information can be used to refine the model to improve its properties.

Most ensemble as well as other machine learning methods produce "black–box" models. They are represented in an incomprehensible form making it difficult to impossible to understand how the input variables are being used for prediction. One of the primary benefits that distinguish rule based ensembles is the ability to intepret the resulting model to gain such information.

Rules of the form (9) represent easily understandable functions of the input variables $\mathbf{x}$. Although a large number of such rules participate in the initial ensemble, the fitting procedure (12) generally sets the vast majority ($\sim$ 80% to 90%) of the corresponding coefficient estimates $\{\hat{a}_k\}_1^K$ to zero and their corresponding rules are not used for prediction. As noted above, this selection property is a well known aspect of the lasso penalty in (12). The remaining rules will have varying coefficient values depending on their estimated predictive relevance. The most relevant rules can then be examined for interpretation.

A commonly used measure of relevance or importance I_k of any predictor in a linear model such as (11) is the absolute value of the coefficient of the corresponding standardized predictor. For rules this becomes

$$I_k = |\, \hat{a}_k \,| \cdot \sqrt{s_k(1 - s_k)} \tag{13}$$

where s_k is the rule support

$$s_k = \sum_{i=1}^{N} w_i \, r_k(\mathbf{x}_i) \Big/ \sum_{i=1}^{N} w_i. \tag{14}$$

Those rules with the largest values for (13) are the most influential for prediction based on the predic-

tive equation (11). These can then be selected and examined for interpretation.

4.1. *Input variable importance*

In predictive learning a descriptive statistic that is almost always of interest is the relative importance or relevance of the respective input variables $(x_1, x_2, \cdots, x_n)$ to the predictive model; that is, which of the variables are most influential in making predictions and which in retrospect need not have been included. For the models (11) considered here, the most relevant input variables are those that preferentially define the most influential rules appearing in the model. Input variables that frequently appear in important rules are judged to be more relevant than those that tend to appear only in less influential rules.

This concept can be captured by a measure of importance J_j of input variable x_j

$$J_j = \sum_{x_j \in r_k} I_k / m_k. \tag{15}$$

This measure sums the importances (13) of those rules (9) that contain x_j ($x_j \in r_k$) each divided by the total number of input variables m_k that define the rule. In this sense the input variables that define a rule equally share its importance, and rules with more variables do not receive exaggerated influence by virtue of appearing in multiple input variable importance measures. The distribution of $\{J_j\}_1^n$ (15) can be examined to ascertain the relative influence of each of the respective input variables on the model's predictions. Illustrations are provided in the example below.

4.2. *Partial dependence functions*

Visualization is one of the most powerful interpretational tools. Graphical renderings of the value of $F(\mathbf{x})$ as a function of its arguments provides a comprehensive summary of its dependence on the joint values of the input variables. Unfortunately, such visualization is limited to low dimensional arguments. Viewing functions of higher dimensional arguments is more difficult. It is therefore useful to be able to view the partial dependence of the approximation $F(\mathbf{x})$ on selected small subsets of the input variables. Although a collection of such plots can seldom provide a comprehensive depiction of the approximation, it can often produce helpful clues.

Let $\mathbf{z}_l$ be a chosen "target" subset, of size l, of the input variables $\mathbf{x}$

$$\mathbf{z}_l = \{z_1, \cdots, z_l\} \subset \{x_1, \cdots, x_n\},$$

and $\mathbf{z}_{\setminus l}$ be the complement subset

$$\mathbf{z}_{\setminus l} \cup \mathbf{z}_l = \mathbf{x}.$$

The approximation $F(\mathbf{x})$ in principle depends on variables in both subsets

$$F(\mathbf{x}) = F(\mathbf{z}_l\,,\, \mathbf{z}_{\setminus l}).$$

If one conditions on specific values for the variables in $\mathbf{z}_{\setminus l}$, then $F(\mathbf{x})$ can be considered as a function only of the variables in the chosen subset $\mathbf{z}_l$

$$F_{\mathbf{z}_{\setminus l}}(\mathbf{z}_l) = F(\mathbf{z}_l \,|\, \mathbf{z}_{\setminus l}). \tag{16}$$

In general, the functional form of $F_{\mathbf{z}_{\setminus l}}(\mathbf{z}_l)$ will depend on the particular values chosen for $\mathbf{z}_{\setminus l}$. If however, this dependence is not too strong then the averaged function

$$\bar{F}_l(\mathbf{z}_l) = E_{\mathbf{z}_{\setminus l}}[F(\mathbf{x})] = \int F(\mathbf{z}_l\,,\, \mathbf{z}_{\setminus l})\, p_{\setminus l}(\mathbf{z}_{\setminus l})\, d\mathbf{z}_{\setminus l} \tag{17}$$

can represent a useful summary of the "partial dependence" of $F(\mathbf{x})$ on the chosen variable subset $\mathbf{z}_l$ (Friedman 2001). Here $p_{\setminus l}(\mathbf{z}_{\setminus l})$ is the marginal probability density of $\mathbf{z}_{\setminus l}$

$$p_{\setminus l}(\mathbf{z}_{\setminus l}) = \int p(\mathbf{x})\, d\mathbf{z}_l, \tag{18}$$

where $p(\mathbf{x})$ is the joint probability density of all of the inputs $\mathbf{x}$. This complement marginal density (18) can be estimated from the training data, so that (17) becomes

$$\bar{F}_l(\mathbf{z}_l) = \sum_{i=1}^{N} w_i\, F(\mathbf{z}_l\,,\, \mathbf{z}_{i\setminus l}) \Big/ \sum_{i=1}^{N} w_i. \tag{19}$$

where $\mathbf{z}_{i\setminus l}$ are the data values of $\mathbf{z}_{\setminus l}$.

Partial dependence functions (19) can be used to help interpret models produced by any "black box" prediction method, such as neural networks, support vector machines, nearest–neighbors, radial basis functions, etc. They only require the value of $F(\mathbf{x})$ for specified values of $\mathbf{x}$. However, when there are a large number of predictor variables, it is very useful to have an measure of relevance (Section 4.1) to reduce the potentially large number variables, and variable combinations, to be considered.

5. Illustration

In this section we apply the RuleFit procedure to a signal/background separation problem from a high energy particle physics experiment and illustrate the various interpretational tools described in Section 4. The training data consists of 50000 Monte Carlo simulated events, half of which are signal and half are background. Details concerning the specific application and the nature of the 50 input variables are withheld at the request of the experimenters. An additional 23000 events were generated (half signal and half background) to evaluate performance. These latter ("test") events were not used to train the predictive model.

All parameters of the RuleFit procedure were set to their default values: $\nu = 0.01$ and $\eta = \min(N/2, 100 + 6\sqrt{N}) \simeq 1450$ events in Algorithm 1, four terminal nodes for each tree, and 3500 generated rules in the initial ensemble (585 trees). It is possible that performance could be improved by tuning some of these parameters for this specific application.

Applying RuleFit to the training data produced a model (11) with 410 rules having nonzero coefficients from (12). The corresponding error rate on the test data was 6.97%. Another measure of prediction quality, area under the ROC curve ("AUC"), was 0.977. Perfect prediction would have zero error rate and AUC = 1.

Figure 2 displays a graphical representation of prediction quality. The upper frame shows the distribution of the model scores $F(\mathbf{x})$ (11) for the 11500 signal events in the test sample; the lower frame shows the corresponding plot for the 11500 background events. One sees that signal events tend to have predominately higher scores than the background. Using a threshold of $t = 0$ (3) gives rise to the minimal error rate of 6.97%, with slightly more background being classified as signal than signal classified as background. Increasing the threshold value ($t > 0$) would reduce background errors leading to a purer sample of signal events at the expense of classifying more of the signal as background. Lowering the threshold ($t < 0$) would capture more of the signal at the expense of increased background contamination. In this context modifying the threshold can be viewed as changing the relative values of the misclassification costs L_S and L_B in (2).

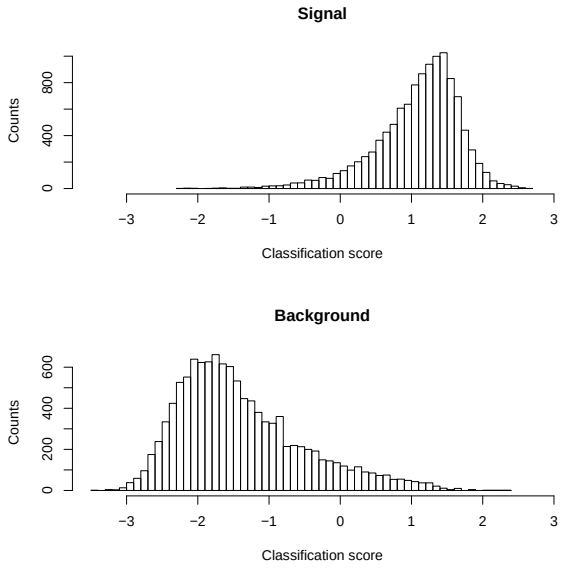

Fig. 2. Distribution of RuleFit prediction scores for signal (upper) and background (lower) test events.

This signal/background trade–off is more directly captured by the corresponding ROC curve shown in Fig. 3. Here the fraction of captured signal events (true positives) is plotted against the fraction of background contamination (false positives) as the threshold t is varied. One sees that permitting 5% background contamination allowed 90% of the signal events to be captured, whereas 10% background captures approximately 95% of the signal.

Table 1 illustrates some typical rules by displaying the five most important using (13). The first column shows the rules' relative importance normalized so that the maximum value over all rules is 100. The second column gives the coefficient $\hat{a}_k$ (11) of the corresponding rule $r_k(\mathbf{x})$. Positive coefficient values indicate that satisfying the rule ($r_k(\mathbf{x}) = 1$) increases the odds of being a signal event, where as negative values decrease the odds. The third column shows the rule's support (14). The last column shows the variables and cut values that define the corresponding rules. One sees that here all of these relatively important rules are fairly simple, typically involving two to three variables. Knowing the meaning of the variables for each of the rules could lead to insights concerning what aspects of the experiment lead to separating signal from background.

Figure 4 plots the relative importances (15) of

Table 1. The five most important rules for differentiating signal from background events.

Importance	Coefficient	Support	Rule
100	-0.16	0.45	$x_6 \leq 0.31$ & $x_{16} \leq 1117$ & $x_{32} \leq 1.31$
83	0.13	0.41	$0.025 \leq x_{14} < 0.53$ & $x_{27} <$82.4
82	0.22	0.093	-500 $\leq x_3 <$ 92.6 & $x_{21} \leq$ -0.022 & $x_{39} >$ 1.18
75	0.12	0.32	$x_1 \leq$ 5.2 & -500$\leq x_3 <$92.6 & $x_{21} >$ -0.022
73	-0.12	0.41	$x_1 >$4.37 & $x_{23} \leq$ 160.1 & $x_{32} \leq$ 1.41

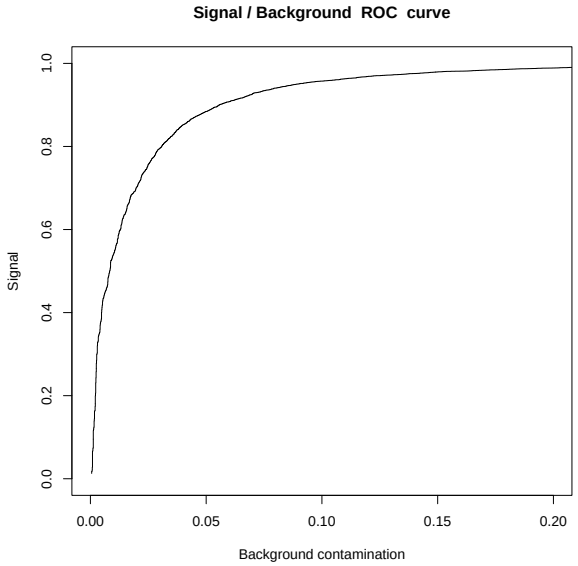

Fig. 3. ROC curve for RuleFit test predictions.

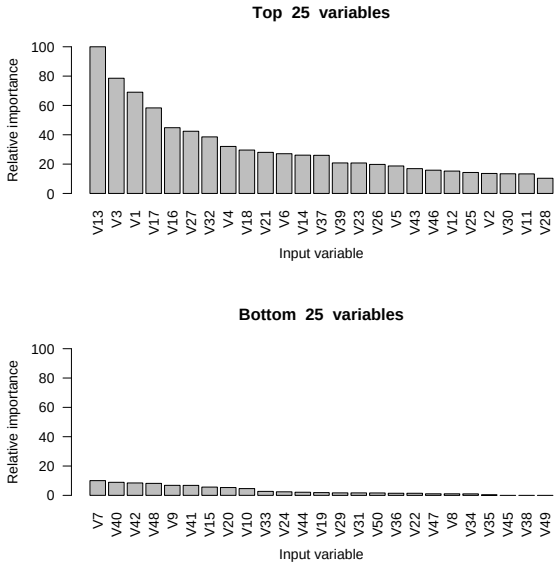

Fig. 4. Relative importances of the 50 input variables to the RuleFit predictive model.

each of the 50 input variables in (inverse) order of their importance values. Here some variables are clearly far more relevant than others to the predictive model (11). Knowing which variables are the important ones for separating signal from background can lead to insights concerning the experimental setup.

Table 2. Error rate and one minus area under the ROC curve for RuleFit models based on subsets of the most important predictor variables.

Variables	1-AUC	Error
50	0.0230	6.97
25	0.0232	7.06
20	0.0237	7.06
15	0.0264	7.60

This information can also be used to simplify the actual predictive model. This is illustrated in Table 2. Each row shows the test error rate (third column) and $1 - AUC$ (second column) for RuleFit models using subsets of the input variables. The first column shows the number of (most important – see Fig. 4) variables used out of the total of 50. One sees that training the model using only the 20 most important variables results in no significant decrease in model quality. Using only the top 15 variables degrades performance only by about 8%. Predictive models with fewer variables might be preferred if some of those variables deemed to be unimportant and thus expendable were especially difficult or expensive to measure.

Figure 5 shows plots of the single variable partial dependence of $F(\mathbf{x})$ (11) on the nine most important variables. One sees that, for example, the odds of being a signal event decrease monotonically with increasing values of the most important variable x_{13}. For the next most important variable x_3, predicted signal odds are lowest for $95 \lesssim x_3 \lesssim 170$ and

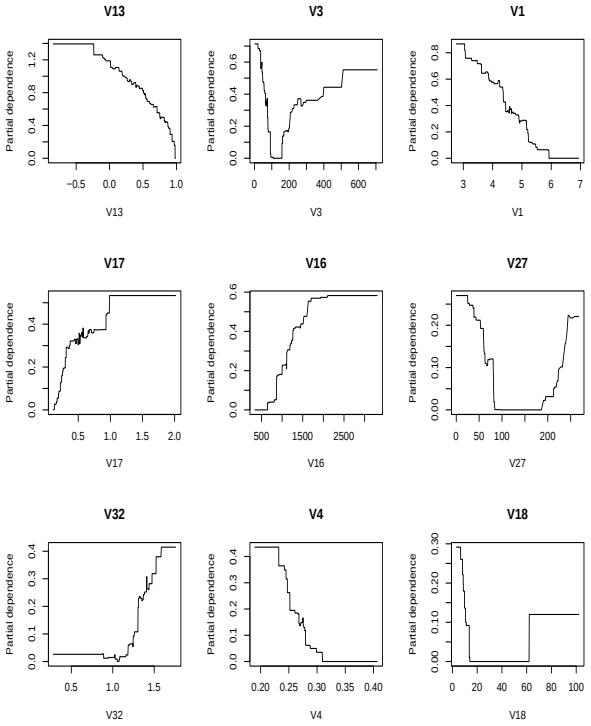

Fig. 5. Single–variable partial dependence plots of the odd of a signal event as a function of the nine most important predictor variables.

become higher for values outside this range. In general, examination of such partial dependences on the important variables provides information on how the values of the corresponding variables are being used for prediction.

More detailed information can be obtained from two–variable partial dependence plots. Figure 6 shows the partial dependence of $F(\mathbf{x})$ (11) on the joint values of selected variable pairs using several plotting formats. The upper left frame shows the partial dependence on (x_1, x_{13}) using a perspective mesh representation. One sees that signal odds increase as either of the two variables become larger. The upper right frame shows a contour plot of the partial dependence on (x_{17}, x_{13}). Here the signal odds are highest for $x_{17} \simeq 0.4$ and $x_{13} \simeq -0.4$, and decrease in all directions from that point. The lower two frames of Fig. 6 use a "heat map" to represent the respective two–variable partial dependence plots. Lowest values are colored red (darker) while higher values are (lighter) yellow, and the highest values (surrounded by yellow) are colored white. As an an example, one sees from the lower right frame that for large values of x_1 the odds of being a signal event are low and at most depend weakly on x_3, whereas for small values of x_1 the odds strongly depend on the value of x_3. This is an example of an interaction (correlation) effect between these two variables.

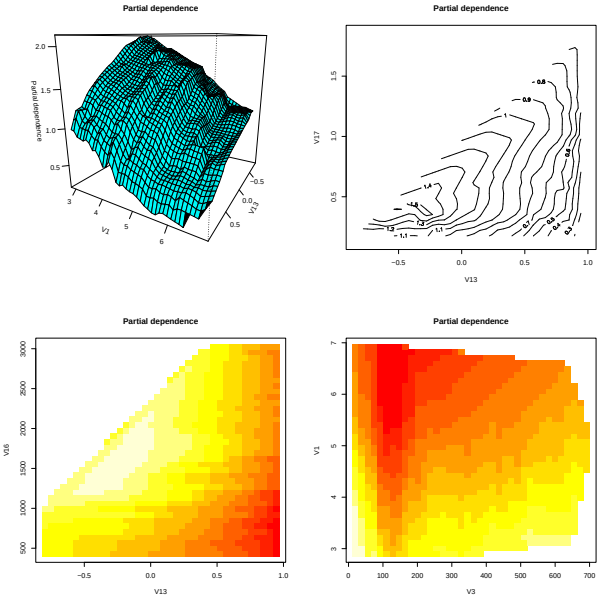

Figure 6: Two–variable partial dependence plots of the odds of a signal event as a function of the joint values of selected variable pairs. Upper left: perspective mesh plot, upper right: contour plot, lower: heat map representation.

6. Conclusion

This paper has outlined the RuleFit technique for predictive learning and illustrated some of its features on a signal/background separation problem in high energy particle physics. A more complete description of the procedure along with its other features can be found in Friedman and Popescu 2005. A software interface to the R statistical package can be obtained from http://www-stat.stanford.edu/~jhf/RuleFit.html.

Acknowledgments

This research was partially supported by the Department of Energy under contract DE-AC02-76SF00515 and the National Science Foundation under grant DMS-97-64431.

References

1. Breiman, L. (1996). Bagging Predictors. *Machine Learning* **26**: 123-140.
2. Breiman, L. (2001). Random Forests. *Machine Learning* **45**: 5-32.
3. Breiman, L., Friedman, J. H., Olshen, R., and Stone, C. (1983). *Classification and Regression Trees.* Wadsworth.
4. Donoho, D., Johnstone, I., Kerkyacharian, G. and Picard, D. (1995). Wavelet shrinkage; asymptotia? (with discussion). *J. Royal. Statist. Soc* **57**: 201-337.
5. Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kauffman, San Francisco, 148-156.
6. Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics* **29**: 1189-1232.
7. Friedman, J. H. and Hall, P. (1999). On bagging and nonlinear estimation. *Stanford University, Department of Statistics.* Technical Report.
8. Friedman, J. H., and Popescu, B. E. (2003). Importance Sampled Learning Ensembles *Stanford University, Department of Statistics.* Technical Report.
9. Friedman, J. H., and Popescu, B. E. (2004). Gradient directed regularization for linear regression and classification. *Stanford University, Department of Statistics.* Technical report.
10. Friedman, J. H., and Popescu, B. E. (2005). Predictive learning via rule ensembles. *Stanford University. Department of Statistics.* Technical report.
11. Quinlan, R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo.
12. Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc. B.* **58**: 267-288.