Probabilistic Random Forests: Predicting Data Point Specific Misclassification Probabilities

Markus Breitenbach
Department of Computer Science
University of Colorado
Boulder, CO 80309
breitenm@cs.colorado.edu

Rodney Nielsen
Department of Computer Science
University of Colorado
Boulder, CO 80309
rodney.nielsen@colorado.edu

Gregory Z. Grudic
Department of Computer Science
University of Colorado
Boulder, CO 80309
grudic@cs.colorado.edu

Abstract

Recently proposed classification algorithms give estimates or worst-case bounds for the probability of misclassification [Lanckriet *et al.*, 2002][L. Breiman, 2001]. These accuracy estimates are for *all* future predictions, even though some predictions are more likely to be correct than others. This paper introduces Probabilistic Random Forests (*PRF*), which is based on two existing algorithms, Minimax Probability Machine Classification and Random Forests, and gives data point dependent estimates of misclassification probabilities for binary classification. A PRF model outputs both a classification and a misclassification probability estimate for the data point. PRF makes it possible to assess the risk of misclassification, one prediction at a time, *without* detailed distribution assumptions or density estimation. Experiments show that PRFs give good estimates of the error probability for each classification.

1 Introduction

Classification has been extensively studied in the machine learning community, with the accuracy of most existing algorithms being determined using test data that the algorithm has never seen [1, 2, 3]. However, recent classification algorithms attempt to estimate accuracy *without* using the test data. For example, the binary Minimax Probability Machine Classification (MPMC) algorithm [4] computes a bound for the probability of misclassification, using only estimates of the covariance matrix and mean for each class, as obtained from the training data. Similarly, Random Forest (*RF*) algorithms [5], which learn an ensemble of CART tree classifiers using bagging and randomized feature selection. RF algorithms select examples from the training data (out-of-bag examples) to obtain good estimates of the true error rate of the final classifier.

However, both the MPMC bound and the RF algorithm misclassification probability estimate are independent of the data point, even though some classifications are more likely to be accurate than others. Other methods exist for estimating the *relative* accuracy of one prediction with respect to another, without using probability estimates. For example, in the case of Support Vector Machines one can use the real-valued output of the classification function as an estimate of misclassification-risk: the closer the value is to zero, the more likely it is that the classification could be wrong. This has been used to choose the classifier that is most-likely to be correct in one-against-all settings of binary classifiers used for multi-class problems [6, 7]. Similarly, Bayesian methods such as Relevance Vector Machines [8] and Gaussian Process Classification [9] can give variance estimates for each classification. Such existing methods yield estimates of accuracy that cannot easily be used to predict actual probabilities of errors for each classification. In general, in order to make specific probability estimates for each classification, existing algorithms require detailed probability assumptions or density modeling [10].

This paper introduces the Probabilistic Random Forests (*PRF*) algorithm, which gives an estimate of the probability of misclassification for each data point, without detailed probability distribution assumptions or resorting to density modelling. Using this probability estimate it is possible to assess how well the learned hypothesis models the data, which has many practical applications. For example, consider a classifier for diagnosing cancer. Cancer tests have a false-positive and a false-negative rate. A classifier that can give a good estimate of the misclassification probability for a particular patient would allow medical doctors to make more informed decisions about further testing and treatment protocols.

PRF is based on two existing algorithms: Minimax Probability Machine Classification [4] and Random Forests [5]. The MPMC algorithm computes a hyperplane that minimizes the maximum probability of misclassification using only estimates of the mean and covariance matrices of each class. The Random Forests algorithm builds an ensemble of CART tree classification predictors using bagging and randomized feature selection - RFs use out-of-bag training examples to estimate the overall error rate of the final classifier. Because a point being classified by a Random Forest is passed to each tree in the ensemble, we consider each point as generating a distribution defined by combining the output of each tree in the forest. PRF performs binary classification by first using Random Forests to generate *three* distributions for each prediction: one that represents the point currently being classified; another that represents the corresponding positive out-of-bag (*OOB*) training examples (i.e., those positive OOB examples that are classified by the same leaf nodes); and a third that represents the corresponding negative out-of-bag samples. Next, PRF classifies by using MPMC to calculate the *distance* between the distribution generated by the point being classified and the distributions of the corresponding positive and negative OOB training examples. If the distribution associated with the point currently being classified is closer to the negative OOB distribution than to the positive, then it is classified as being negative - otherwise it is classified as being positive. Finally, the relative distances to these distributions are used within the Bayes Theorem formulation to give an estimate of how likely it is that the classification is wrong.

Matlab and C code implementing the PRM algorithm can be downloaded from: http://ucsu.colorado.edu/~breitenm/.

2 Probabilistic Random Forests

2.1 Random Forests revisited

Random Forests [5] build an ensemble of CART tree classification predictors using bagging. In addition to using bagging, each node of the trees only considers a small subset of features for the split, which enables the algorithm to build classifiers for high dimensional data very quickly. The trees are not pruned and can, therefore, get very large. The accuracy of these predictors is due to the minimization of the correlation between the classifiers, while maximizing the strength. Strength is a measure for the ability of a tree to classify data points correctly. Internal estimates, which are obtained using the out-of-bag examples, can give estimates for the performance on unseen data as well as estimates for the strength of each tree, and the correlation between trees. The classification of unseen points is done by voting. Due to the large number of simple classifiers and the minimized correlation between the classifiers the error converges toward error rates comparable to Ada-boost [1]. There are variants that split on a random feature or pick the best out of several random feature subsets by comparing how well the subsets perform on the out-of-bag examples. Another variant of Random Forests uses random linear combinations of features in the selected feature subset, i.e., the features are randomly weighted with uniformly distributed random numbers on the interval $[-1, 1]$.

2.2 Mimimax Probability Machine Classification

Binary Minimax Probability Machine Classification (MPMC) is formulated as finding a hyperplane that minimizes the maximum probability of misclassification [4]. Specifically, given two random vectors in d dimensional space, $\mathbf{u} = (u_1, ..., u_d) \in \Re^d$ symbolizing one class and $\mathbf{v} = (v_1, ..., v_d) \in \Re^d$ symbolizing the other class, drawn from two probability distributions characterized by mean and covariance matrices, $(\bar{\mathbf{u}}, \Sigma_\mathbf{u})$ and $(\bar{\mathbf{v}}, \Sigma_\mathbf{v})$, MPMC finds a hyperplane, $\mathbf{a}^T\mathbf{z} = b$ (where $\mathbf{a} = (a_1, ..., a_d) \in \Re^d$, $\mathbf{z} = (z_1, ..., z_d) \in \Re^d$, $b \in \Re$), such that

$$\max_{\Omega, \mathbf{a} \neq \mathbf{0}, b} \Omega \;\; s.t. \inf_{\mathbf{u} \sim (\bar{\mathbf{u}}, \Sigma_\mathbf{u})} Pr\{\mathbf{a}^T\mathbf{u} \geq b\} \geq \Omega \wedge \inf_{\mathbf{v} \sim (\bar{\mathbf{v}}, \Sigma_\mathbf{v})} Pr\{\mathbf{a}^T\mathbf{v} \leq b\} \geq \Omega \tag{1}$$

where the maximum probability of misclassification is given by $(1 - \Omega)$. The optimal hyperplane is found by solving the following optimization problem (see **Theorem 2** in [4]): find a minimum $m \in \Re$ and $\mathbf{a} \in \Re^d$ as defined below

$$m = \min_{\mathbf{a}} \left(\sqrt{\mathbf{a}^T\Sigma_\mathbf{u}\,\mathbf{a}} + \sqrt{\mathbf{a}^T\Sigma_\mathbf{v}\,\mathbf{a}}\right) \quad s.t. \;\; \mathbf{a}^T(\bar{\mathbf{u}} - \bar{\mathbf{v}}) = 1 \tag{2}$$

Given m and $\mathbf{a}$, the offset b of the MPMC hyperplane is uniquely given by:

$$b = \mathbf{a}^T\bar{\mathbf{u}} - \frac{\sqrt{\mathbf{a}^T\Sigma_\mathbf{u}\,\mathbf{a}}}{m} = \mathbf{a}^T\bar{\mathbf{v}} + \frac{\sqrt{\mathbf{a}^T\Sigma_\mathbf{v}\,\mathbf{a}}}{m} \tag{3}$$

and the probability bound Ω has the form:

$$\Omega = \frac{1}{m^2 + 1} \tag{4}$$

In this paper, we use two versions of this MPMC theory. The first is for the one dimensional case, $\mathbf{u} = u_1 \in \Re$, $\mathbf{v} = v_1 \in \Re$, and $\mathbf{z} = z_1 \in \Re$ (i.e., $d = 1$). Solving equation (2) for this one dimensional case leads to the following optimal linear MPMC boundary $a_1 z_1 = b$:

$$a_1 = \frac{1}{\bar{u}_1 - \bar{v}_1} \tag{5}$$

where b is given in (3).

The second version is for the two dimensional case $\mathbf{u} = (u_1, u_2) \in \Re$, $\mathbf{v} = (v_1, v_2) \in \Re$, and $\mathbf{z} = (z_1, z_2) \in \Re$ (i.e., $d = 2$). Equation (2) has a closed form solution in two dimensions as well. This involves finding the roots of a fourth order polynomial [11]; however, as we are not interested in the optimal boundary in two dimensional space (see description of algorithm in Section 2.3), instead of solving the problem in (2), we solve the following computationally simpler problem:

$$m = \min_{\mathbf{a}} (\mathbf{a}^T\Sigma_\mathbf{u}\,\mathbf{a} + \mathbf{a}^T\Sigma_\mathbf{v}\,\mathbf{a}) \quad s.t. \;\; \mathbf{a}^T(\bar{\mathbf{u}} - \bar{\mathbf{v}}) = 1 \tag{6}$$

Substituting $a_2 = (1 - a_1 d_1)/d_2$ into m, setting the result to zero and solving for a_1 gives:

$$\begin{array}{l} a_1 = \frac{\Sigma_{\mathbf{u}_{12}} d_2 - d_1 \Sigma_{\mathbf{u}_{22}} - d_1 \Sigma_{\mathbf{v}_{22}} + \Sigma_{\mathbf{v}_{12}} d_2}{2 d_1 \Sigma_{\mathbf{u}_{12}} d_2 - d_1^2 \Sigma_{\mathbf{u}_{22}} - \Sigma_{\mathbf{u}_{11}} d_2^2 - \Sigma_{\mathbf{v}_{11}} d_2^2 + 2 d_1 \Sigma_{\mathbf{v}_{12}} d_2 - d_1^2 \Sigma_{\mathbf{v}_{22}}} \\ a_2 = (1 - a_1 d_1)/d_2 \end{array} \tag{7}$$

where $d_1 = \bar{u_1} - \bar{v_1}$, $d_2 = \bar{u_2} - \bar{v_2}$, and

$$\sum\nolimits_\mathbf{u} = \left[\begin{array}{cc} \sum_{\mathbf{u}_{11}} & \sum_{\mathbf{u}_{12}} \\ \sum_{\mathbf{u}_{12}} & \sum_{\mathbf{u}_{22}} \end{array}\right] \quad \sum\nolimits_\mathbf{v} = \left[\begin{array}{cc} \sum_{\mathbf{v}_{11}} & \sum_{\mathbf{v}_{12}} \\ \sum_{\mathbf{v}_{12}} & \sum_{\mathbf{v}_{22}} \end{array}\right]$$

Note that it is possible that the hyperplane does not exist, if e.g., $d_2 = 0$.

2.3 Probabilistic Random Forests: Algorithm Description

We consider the standard binary classification problem. The training data exists in some n dimensional feature space, with the two classes being symbolized by $\mathbf{x} \in \Re^n$ for one class, and $\mathbf{y} \in \Re^n$ for the other class. We assume the training data has the form $\mathbf{x}_1, ..., \mathbf{x}_{N_x}$ and $\mathbf{y}_1, ..., \mathbf{y}_{N_y}$, where N_x is the number of training examples from the $\mathbf{x}$ class, and N_y is the number of examples from the $\mathbf{y}$ class.

The Probabilistic Random Forest algorithm differs from standard Random Forests in two ways. First, the split at each node of the tree in the forest is defined by the following hyperplane:

$$a_1 K\left(\mathbf{f}_x, \mathbf{f}\right) + a_1 K\left(\mathbf{f}_y, \mathbf{f}\right) - b \begin{cases} \geq 0 \Rightarrow \text{right child node} \\ < 0 \Rightarrow \text{left child node} \end{cases} \tag{8}$$

where vector $\mathbf{f}$ contains a randomly chosen subset of the n possible feature values for the point being classified; $\mathbf{f}_x$ and $\mathbf{f}_y$ are the corresponding feature values from a randomly chosen training examples in the $\mathbf{x}$ and $\mathbf{y}$ classes, respectively; $K\left(\mathbf{f}_\gamma, \mathbf{f}\right) = \mathbf{f}_\gamma{}^T \mathbf{f}$ with $\gamma \in \{\mathbf{x}, \mathbf{y}\}$ is a linear kernel; and a_1, a_2 and b are obtained by setting $z_1 = K\left(\mathbf{f}_x, \mathbf{f}\right)$, $z_2 = K\left(\mathbf{f}_y, \mathbf{f}\right)$ and evaluating equations (7) and (3) using the data at the tree node to estimate the means and covariance matrices of the two classes (which are defined by $(\bar{\mathbf{u}}, \Sigma_{\mathbf{u}})$ and $(\bar{\mathbf{v}}, \Sigma_{\mathbf{v}})$ in the previous section). The key characteristic that we use from this formulation is that each point being evaluated by a tree t_i (for $i = 1, ...k$, where k is the number of trees in the forest) has a real valued number associated with it at the terminal node as follows:

$$\beta_i = a_1 K\left(\mathbf{f}_x, \mathbf{f}\right) + a_1 K\left(\mathbf{f}_y, \mathbf{f}\right) - b \tag{9}$$

These values $\beta_1, ..., \beta_k$ are used for both classification and estimating the probability of classification error as described below.

The second key difference between PRFs and standard Random Forests is that each terminal node in each tree in the forest has at least two out-of-bag examples from each class, which are *NOT* used to construct the tree. The idea is to sample at least one value for each class from each tree of the ensemble when classifying data. Using two, however, guarantees that we have additional values to adjust the predictions as described later on. Therefore, we stop the construction of each tree when adding further nodes would violate this condition. These out-of-bag examples are mapped to β values using equation 9. Therefore, given an instance $\mathbf{r} \in \Re^n$ which is to be classified, we can obtain three sets of real valued outputs: $\beta = \beta_1, ..., \beta_k$ corresponding to the terminal node outputs for the point $\mathbf{r}$ being classified; $\beta^x = \beta_1^x, \beta_2^x, ...$ which are the outputs of all the $\mathbf{x}$ class out-of-bag examples that are classified by the same terminal nodes as $\mathbf{r}$; $\beta^y = \beta_1^y, \beta_2^y, ...$ which are the corresponding outputs of all the $\mathbf{y}$ class out-of-bag examples. Given these three distributions, we classify the point $\mathbf{r}$ using Bayes Rule as follows:

$$\Pr\left(\mathbf{x}' \left| \beta\right.\right) = \frac{\Pr\left(\mathbf{x}'\right) \Pr\left(\beta \left| \mathbf{x}'\right.\right)}{\Pr\left(\mathbf{y}'\right) \Pr\left(\beta \left| \mathbf{y}'\right.\right) + \Pr\left(\mathbf{x}'\right) \Pr\left(\beta \left| \mathbf{x}'\right.\right)} \tag{10}$$

where $\Pr(\mathbf{x}'|\beta)$ is the probability that the point is *NOT* in class $\mathbf{x}$ given the set β; $\Pr\left(\mathbf{x}'\right)$ is the probability of not being in class $\mathbf{x}$ (i.e., the probability of class $\mathbf{y}$); $\Pr(\mathbf{y}')$ is the probability of class $\mathbf{x}$; $\Pr(\beta|\mathbf{x}')$ is the probability of generating the set β assuming the class is $\mathbf{y}$ (i.e., not $\mathbf{x}$); and $\Pr(\beta|\mathbf{y}')$ is the probability of generating the set β assuming the class is $\mathbf{x}$.

We can estimate $\Pr(\mathbf{x}')$ and $\Pr(\mathbf{y}')$ simply by counting how many times the two classes occur in the entire training set. To estimate $\Pr(\beta|\mathbf{x}')$ we use the sets β^x and β within the theoretical framework of the one dimensional MPMC theory defined in equation (5) as follows. Let $\bar{u}_1 = \bar{\beta}$ be the mean of the β values, and $\bar{v}_1 = \bar{\beta}^x$ be the mean of the β^x values, with corresponding variances $\Sigma_{\mathbf{u}} = V(u_1) = V(\beta)$ and $\Sigma_{\mathbf{v}} = V(v_1) = V(\beta^x)$. The estimates of means are plugged into equation (5) to obtain a_1, which is used in equation (2) along with the variance estimates to obtain the optimal m. This m is plugged into equation (4) to obtain a bound, which we call $\Omega^{x'}$, on the *separability* of the sets β^x and β. By separability, we mean that if a point belongs to one set, what is the minimum probability of classifying it correctly. If the two sets are statistically identical, points in one set cannot be separated from points in the other set (i.e., $\Omega^{x'} = 0$). If they are completely different statistically, then points in one set are completely separable from points in the other set (i.e., $\Omega^{x'} = 1$). Therefore, we can substitute $\Omega^{x'}$ as an analogue for $\Pr(\beta|\mathbf{x}')$, the probability that the set β was generated given that the points evaluated were from the class $\mathbf{x}'$. A similar calculation obtains $\Omega^{y'}$ as the analogue for $\Pr(\beta|\mathbf{y}')$ by using the set β^y instead of β^x. We call these analogues, because the following theorem shows that $\Pr(\beta|\mathbf{x}')$ is monotonic in $\Omega^{x'}$, but not equal.

Theorem: *Assume an infinite set of trees in the random forest, with the correlation, measured with respect to the outputs β_i (see equation (9)) from the terminal node of each tree, between any two trees being in the range $(-1, 1)$. Assume that each tree has a finite nonzero strength. Then $\Pr(\beta|\mathbf{x}')$ monotonically increases as $\Omega^{x'}$ increases and $\Pr(\beta|\mathbf{y}')$ monotonically increases as $\Omega^{y'}$ increases*

Proof Sketch: Because the strength of each tree is nonzero, and given that the trees have finite correlations in the range $(-1, 1)$, then from Theorems 1.2 and 2.3 in [5], it must be the case that the mean of the values β^x, for class $\mathbf{x}$, cannot equal the mean of the values β^y, for class $\mathbf{y}$. Therefore, there is a nonzero probability that points in $\mathbf{y}'$ can be distinguished from points in $\mathbf{x}'$ by the MPMC (see **Theorem 2** in [4]). Because out-of-bag points are not used in the construction of a tree, they are unbiased or statistically identical to a group of points the tree has never seen. Therefore, when a point from the $\mathbf{x}'$ class is classified, the associated values β have a higher probability of being like $\beta^{x'}$ than they do of being like β^x. Therefore, as $\Omega^{x'}$ decreases, $\Pr(\beta|\mathbf{x}')$ must also decrease. If it did not, then as the two distributions move closer to one another, the probability of them being generated according to the same process would decrease - leading to a contradiction. The same argument applies for points belonging to the $\mathbf{y}'$ class. This completes the proof sketch.

A PRF model substitutes these analogues, $\Omega^{x'}$ for $\Pr(\mathbf{x}'|\beta)$ and $\Omega^{y'}$ for $\Pr(\mathbf{y}'|\beta)$, into equation (10) to approximate $\Pr(\mathbf{x}'|\beta)$ and $\Pr(\mathbf{y}'|\beta) = 1 - \Pr(\mathbf{x}'|\beta)$. These approximations are then used to estimate a classification and the corresponding probability of misclassification. If $\Pr(\mathbf{y}'|\beta) > \Pr(\mathbf{x}'|\beta)$, the point is classified as belonging to class $\mathbf{x}$ with the probability of misclassification being $\Pr(\mathbf{x}'|\beta)$. Otherwise, it is classified as belonging to class $\mathbf{y}$ with the probability of misclassification being $\Pr(\mathbf{y}'|\beta)$. We further use the out-of-bag examples to calibrate these first-order estimates of the misclassification probability into final error rate estimates as follows.

For each training example, r, visit each tree, t_i, in the ensemble where r was an out-of-bag example (approximately 33% of the trees). Perform the same steps as described earlier for classifying test examples. In this case, β is the set of outputs associated with our out-of-bag example, r, and β^x and β^y are the outputs from all other out-of-bag examples (excluding r) that were classified by the same terminal nodes as r. As noted previously, the process provides a classification for example r and an estimate for the misclassification probability. Now, we calculate the difference between the actual and predicted error rates for the out-of-bag examples. This provides a gross adjustment to the predicted error rate for test examples having similar predictions. In this implementation, we make the adjustment based on the mean values over prediction intervals. After classifying a set of test points, we group together points whose predicted misclassification probabilities fall in the same arbitrary range. Then, using these same ranges, we group the out-of-bag points based on their predictions. Finally, we adjust the predictions of test points by adding the difference between the mean actual error rate and mean predicted error rate for those out-of-bag points that fall in the same interval as the test point. The rationale for using this interval-based approach to adjusting predictions was two-fold. First, due to the unbiased nature of OOB examples, those points clasified by the same terminal nodes as a given test point statisticaly will have both similar predictions and similar error rates. Second, emperical evidence demonstrated that the error in our first-order approximations of the misclassification rate was similar for points with similar predictions. In the future, we will likely implement a method more directly analogous to leave-one-out validation.

Next we give a complete description of how the Probabilistic Random Forest is constructed.

Estimate $\Pr(\mathbf{x}')$ and $\Pr(\mathbf{y}')$ by counting how many times the two classes occur in the entire training set. The following steps are repeated for each tree in the ensemble:

1. Create a bag (i.e., a bootstrap sample - N samples taken randomly with replacement - of the training data) of positive and negative examples. The bags are created separately for each class.
2. Extract all of the examples from the learning set that are not part of the bag. These are the out-of-bag examples.
3. Grow a decision tree from the bag. At each node pick one point at random from class $\mathbf{x}$ and $\mathbf{y}$. Pick a random subset of features without replacement whose size is the square root of the dimensionality. Obtain the best split by determining the decision boundary as defined in equation (8) using the 2D-MPMC method (see section 2.2), with a kernel function K. Split the learning and out-of-bag examples according to the obtained decision boundary. Stop growing the tree, if (a) a child node would have less than two training or OOB examples from either class; or (b) the decision function does not further seperate the classes. If a node is a leaf in the tree, compute β^x and β^y, the real valued outputs of the decision function for each of the out-of-bag examples of class $\mathbf{x}$ and $\mathbf{y}$, and store them with that node.

Classifying a data point is straight forward: traverse each tree and obtain β as well as β^x and β^y from the respective terminal nodes of the trees. Compute the classification and the estimate of misclassification, including the adjustment based on the out-of-bag performance, as described earlier.

3 Experimental results

Figure (1) presents the key experimental findings of this work. Chiefly, that the algorithm accurately predicts the error rate for individual regions of the input space. The figure shows this by comparing the predictions to actual values within five intervals. Intervals were created by sorting all of the errors by their associated predictions and dividing at the midpoint between errors, such that there were an equal number of errors per interval. Diamonds show the test data error rates before adjustment, circles show the error rates on out-of-bag examples as described in section 2, and asterisks demonstrate the bottom line efficacy of the algorithm, which adjusts the first-order predictions according to performance on the out-of-bag data within the same prediction interval. A perfect prediction would be exactly along the dashed line. We can see in figure (1) that the adjusted predictions (asterisks) follow the actual error rate closely.

We evaluated our method on the same five datasets as in [4], and votes as in [5]. The real-world datasets were downloaded from UCI [12]. Rather than generating the data for twonorm, we downloaded it from [13]. The features in all datasets were scaled to be within $[-1, 1]$. Missing boolean attribute values in votes were replaced with 0.0, resulting in about 50% of the dataset having some essentially neutral noise. We used the square-root of the dataset's dimension for the size of the random feature set, as recommended in [14]. The decision at each branch in the trees was made using a linear kernel. Mean values are reported from 100 random partitions of the data holding out 10% for testing, with the

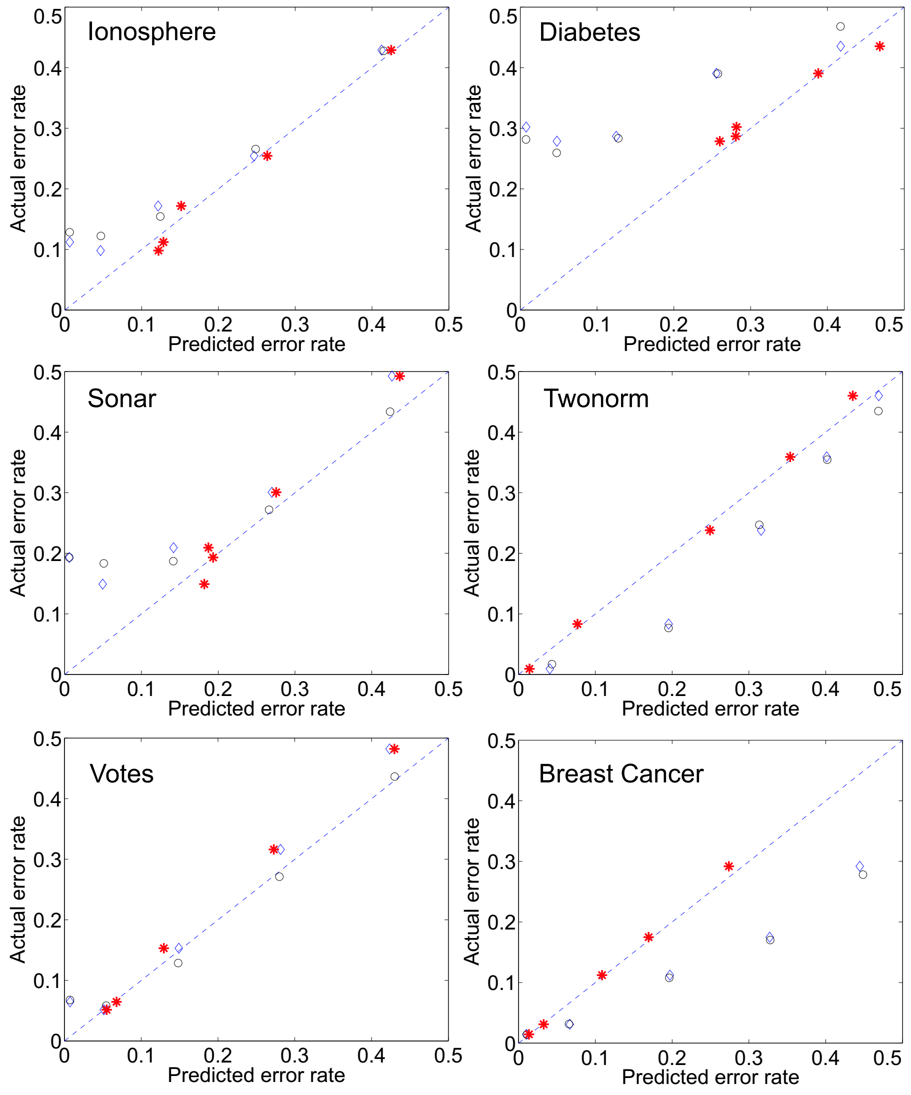

Figure 1: Diamonds show the test data error rates before adjustment, circles show the error rates on out-of-bag examples as described in section 2, and asterisks demonstrate the bottom line efficacy of the algorithm, which adjusts the first-order predictions according to performance on the out-of-bag data within the same prediction interval.

Dataset	PRF	MPMCL	MPMCG	SVML	SVMG	RF
Ionosphere	80.8 ± .7 %	85.4%	93.0%	87.8%	91.5%	92.9%
Sonar	81.0 ± .9 %	75.1%	89.8%	75.9%	86.7%	84.1%
Breast Cancer	95.9 ± .3 %	97.2%	97.3%	92.6%	98.5%	97.1%
Pima diabetes	69.9 ± .5 %	73.8%	74.6%	70.1%	75.3%	66.9%
Twonorm	96.3 ± .1 %	96.3%	97.2%	95.6%	97.4%	96.1%
Votes	90.2 ± .6 %					95.9%

Table 1: Test set accuracy of the Probabilistic Random Forest (PRF) compared to the linear Minimax Probability Machine (MPMCL), the Gaussian MPMC (MPMCG), the linear Support Vector Machine (SVML), the Gaussian SVM (SVMG) (results published in [4]) and Random Forests (RF) (results published in [5]).

exception of twonorm, where 300 random examples were used for training and 3000 for testing, per prior work [15]. The forest constructed for each partition consisted of 100 trees.

4 Conclusion

This paper proposes an algorithm (Probabilistic Random Forests, *PRM*) that estimates data point dependent misclassification probabilities. The experimental results show that the estimates are very close to the actual misclassification rates. The method works without making detailed distributional assumptions or density estimates, and uses only the training data to generate these probability estimates. The algorithm extends recent methods that either calculate bounds [4], or estimate [5] misclassification probabilities for all data, independent of the particular point being classified (but also using only the training data).

Future research will make more extensive use of kernel functions, since the current version of the algorithm only makes use of linear kernels. We plan to explore ways to automatically determine a good kernel for each node. Furthermore, we will examine how choosing from several of the parameters at each node (feature subset as well as the random points for the two classes), based on their performance on out-of-bag examples, affects the overall strength of the trees. We have identified and will implement techniques to increase the accuracy rate to more competitive values. Initial experiments indicate that PRMs can give state of the art classification accuracy, and excellent estimates of data point dependent misclassification probabilities.

References

[1] Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artificial Inteligence*, 14(5):771–780, 1999.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, , and C. J. Stone. *Classification and Regression Trees*. Wadsworth Inc., 1984.

[3] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[4] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. Technical report, Dept. of Comp. Sci. and IE, National Taiwan University, Taipei, 2001.

[7] L. Bottou, C. Cortes, J. Dcnkcr, H. Druckcr, I. Guyon, L. Jackcl, Y. lc Cun, U. Muller, E. Sackingcr, P. Simard, , and V. Vapnik. Comparison of classifier methods: A case study in handwriting digit recognition. *International Conference on Pattern Recognition*, pages 77–87, 1994.

[8] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning*, 1:211–244, 2001.

[9] Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*, 2002.

[10] T.W. Anderson and R. R. Bahadur. Classification into two multivariate normal distributions with different covariance matrices. *Annals of Mathematical Statistics*, pages 420–431, 1962.

[11] T. R. Strohmann, A. Belitski, D. M. DeCoste, and G. Z. Grudic. *Sparse Greedy MPM Classification.* submitted to NIPS, 2003.

[12] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[13] The delve project. http://www.cs.toronto.edu/ delve/data/twonorm/desc.html.

[14] Leo Breiman. Random forests readme file.

[15] Leo Breiman. Arcing classifiers. *The Annals of Statistics*, pages 801–849, 1998.