On the Optimality of Probability Estimation by Random Decision Trees

Wei Fan
IBM T.J. Watson Research
19 Skyline Drive, Hawthorne, NY 10532
weifan@us.ibm.com

Abstract

Random decision tree is an ensemble of decision trees. The feature at any node of a tree in the ensemble is chosen randomly from remaining features. A chosen discrete feature on a decision path cannot be chosen again. Continuous feature can be chosen multiple times, however, with a different splitting value each time. During classification, each tree outputs raw posterior probability. The probabilities from each tree in the ensemble are averaged as the final posterior probability estimate. Although remarkably simple and somehow counter-intuitive, random decision tree has been shown to be highly accurate under 0-1 loss and cost-sensitive loss functions. Preliminary explanation of its high accuracy is due to the "error-tolerance" property of probabilistic decision making. Our study has shown that the actual reason for random tree's superior performance is due to its optimal approximation to each example's true probability to be a member of a given class.

Introduction

Given an unknown target function $y = F(\mathbf{x})$ and a set of examples of this target function $\{(\mathbf{x}, y)\}$, a classification algorithm constructs an inductive model that approximates the unknown target function. Each example $\mathbf{x}$ is a feature vector of discrete and continuous values such as age, income, education, and salary. y is drawn from a discrete set of values such as {*fraud*, *nonfraud*}. A classification tree or decision tree is a directed single-rooted acyclic graph (sr-DGA) ordered feature tests. Each internal node of a decision tree is a feature test. Prediction is made at leaf nodes. Decision trees classify examples by sorting them down the tree from the root to some leaf node. Each non-leaf node in the tree specifies a test of some feature of that example. For symbolic or discrete features, each branch descending from the node specifies to one of the possible values of this feature. For continuous values, one branch corresponds to instances with feature value $\geq$ the threshold and another one $<$ the threshold. Different instances are classified by different paths starting at the root of the tree and ending at a leaf. Some instances, e.g., with missing attribute values etc., may be split among multiple paths. w is the weight of an instance $\mathbf{x}$; it is set to 1.0 initially or other real numbers proportional to the probability that $\mathbf{x}$ is sampled. When $\mathbf{x}$ splits among multiple paths, the weight is split among different paths usually proportional to the probability of that path. If $\mathbf{x}$ has a missing value at an attribute test and the attribute has value 'A' with probability 0.9 and value 'B' with probability 0.1 in the training data, $\mathbf{x}$ will be classified by both path 'A'and path 'B' with weights $0.9w$ and $0.1w$ respectively. Since every path is unique and every possible split is disjoint, the sum of all weights at every leaf node is the sum of the weight of every instance. A leaf is a collection of examples that may not be classified any further. Ideally, they may all have one single class, in which case, there is no utility for further classification. In many cases, they may still have different class labels. They may not be classified any further because either additional feature tests cannot classify better or the number of examples are so small that fails a given statistical significance test. In these cases, the prediction at this leaf node is the majority class or the class label with the most number of occurrences. Since each path from the root to a leaf is unique, a decision tree shatters the instance place into multiple leaves.

The performance of a decision tree is measured by some "loss function" specifically designed for different applications. Given a loss function $L(t, y)$ where t is the true label and y is the predicted label, an optimal decision tree is one that minimizes the average loss $L(t, y)$ for all examples, weighted by their probability. Typical examples of loss functions in data mining are 0-1 loss and cost-sensitive loss. For 0-1 loss, $L(t, y) = 0$ if $t = y$, otherwise $L(t, y) = 1$. For cost-sensitive loss, $L(t, y) = c(\mathbf{x}, t)$ if $t = y$, otherwise $L(t, y) = w(\mathbf{x}, y, t)$. In general, when correctly predicted, $L(t, y)$ is only related to $\mathbf{x}$ and its true label t. When misclassified, $L(t, y)$ is related to the example as well as its true label and the prediction. If the problem is not ill-defined, we expect $c(\mathbf{x}, t) \leq w(\mathbf{x}, t, y)$. For many problems, t is nondeterministic, i.e., if $\mathbf{x}$ is sampled repeatedly, different values of t may be given. This is due to various reasons, such as noise in the data, inadequate feature set, insufficient feature precision or stochastic nature of the problem (i.e., for the same $\mathbf{x}$, $F(\mathbf{x})$ returns different value at different time). It is usually difficult to know or measure apriori if a problem is deterministic. Without prior knowledge, it is hard to distinguish noise from stochastic nature of the problem. The

optimal decision y_* for $\mathbf{x}$ is the label that minimizes the expected loss $E_t(L(t, y_*))$ for a given example $\mathbf{x}$ when $\mathbf{x}$ is sampled repeatedly and different t's may be given. For 0-1 loss function, the optimal prediction is the most likely label or the label that appears the most often when $\mathbf{x}$ is sampled repeatedly. For cost-sensitive loss, the optimal prediction is the one that minimizes the empirical risk.

To choose the optimal decision, *a posteriori* probability is usually required. In decision tree, assume that n_c is the number or weight of examples with class label c at a leaf node, and n is the total number or weight of examples at the leaf. The raw *a posteriori* probability can be estimated as

$$P(c|\mathbf{x}) = \frac{n_c}{n} \quad (1)$$

It is usually NP-hard to find the most accurate decision tree on a given training data. Most of the practical techniques is to use "greedy" to approximate the simplest model with various heuristics. For decision trees, typical heuristics include information gain (Quinlan 1993), gini index (Mehta, Agrawal, & Rissanen 1996), Kearns-Mansour criterion (Kearns & Mansour 1996) among others. After the model is completely constructed, it is then further simplified via pruning with different techniques such as MDL-based pruning (Mehta, Rissanen, & Agrawal 1995), reduced error pruning (Quinlan 1993), cost-based pruning (Bradford *et al.* 1998), among others. Although no longer NP-hard, most of these techniques still require significant amount of computation as well as storage. Most algorithms require multiple scans of the training data and require data to be held in main memory.

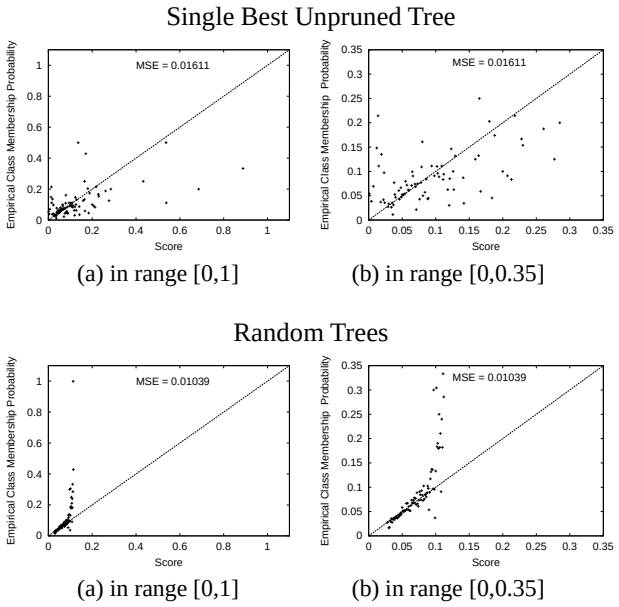
Random decision tree is proposed in (Fan *et al.* Nov 2003). It is an ensemble of decision trees trained randomly. When training each tree, the splitting feature at each node is chosen randomly from any "remaining" features. A chosen discrete feature on a particular decision path (starting from the root of the tree to the current node) cannot be chosen again since it is useless to test the same discrete feature more than once, i.e., each split path will have the same discrete feature value. However, continuous features can be chosen multiple times, each time with a different randomly chosen spliting value. During classification, each tree outputs raw posterior probability. The probabilities from each tree in the ensemble are averaged as the final posterior probability estimate. Although remarkably simple and somehow counter-intuitive, random decision tree has been shown to be highly accurate under both 0-1 loss and cost-sensitive loss functions. Preliminary explanation provided in (Fan *et al.* Nov 2003) is due to the "error-tolerance" property of probabilistic decision making. Borrowing from (Fan *et al.* Nov 2003), assuming 0-1 loss function and a binary problem. The probability threshold to predict $\mathbf{x}$ to be a member of class y is 0.5, i.e., we predict $\mathbf{x}$ to be a member of class y iff $P(y|\mathbf{x}) > 0.5$. Under this situation, an estimated probability of 0.99 and 0.51 will have exactly the same effect, predicting $\mathbf{x}$ to be a member of y, since both 0.99 and 0.51 are higher than the decision threshold of 0.5.

Our study has shown that the actual reason for random tree's superior performance is due to its optimal approximation to each example's true probability to be a member of a given class. On the reliablity curve, the probability output by random tree matches the true probability closely throughout the range (between 0 and 1). However, the best single decision tree (one trained with best spliting criteria such as information gain) matches well only at points close to either 0 or 1. In the range around 0.5, single best decision tree either significantly over-estimates or under-estimates.

Random Decision Tree

The "canonical" form of random decision tree first builds the structure of N random decision trees without the data. It then updates the statistics of each node by scanning the training examples one by one. The feature set is provided to construct the structure of the tree. At each node of the tree, a remaining feature is completely randomly chosen despite of the training data. A discrete feature can be used only once in a decision path starting from the root of the tree till the current node. Continuous features can be discretized and treated as discrete features. Another approach is to pick a random dividing point (i.e., $<$ and $\geq$ the dividing value) each time that this continuous feature is chosen. Since a different dividing point is picked whenever a continuous feature is chosen, it can be used multiple times in a decision path. After the tree structures are finalized, the dataset is scanned only once to update the statistics of each node in every tree. These statistics are simply to track the number of examples belonging to each class that are "classified" by each node. These statistics are used to compute the posteriori probability (Eq. 1). Empty nodes are removed from the random trees. To classify an example, the *a posteriori* probability output from multiple trees are averaged as the final probability estimate. In order to make a decision, a loss function is required in order to minimize the expected loss when the same example is drawn from the universe repeatedly. When the number of features are small, generate empty trees are realistic. However, when the number of features are a large, generated empty trees before scanning the data can lead to astronomical amount of memory usage. In our implementation, we do not generate a node unless some example in the training datasets will actually go there. In other words, we do not generate empty nodes. On average, we find that a random tree is about 2 to 3 times bigger than a single best tree when stored in the Linux file system.

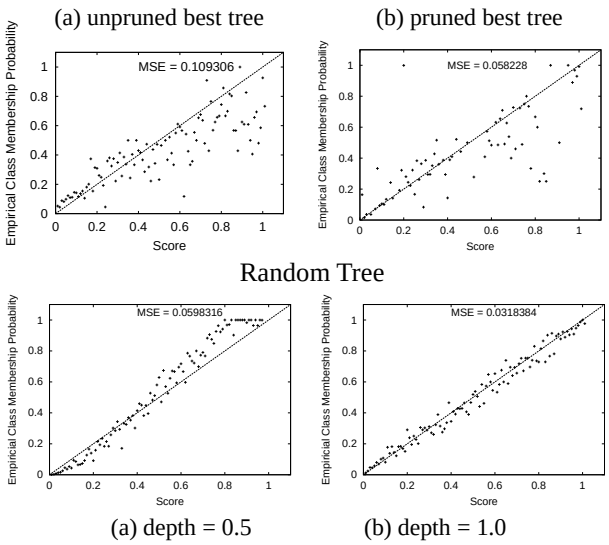
According to (Fan *et al.* Nov 2003), the minimal number of decision trees are 30. That is when the t-distribution is nearly identical to normal distribution. When the distribution of the data is extremely skewed, however, it usually requires a sample size of about 50. They have also discussed the depth of the tree. One heuristic suggested in (Fan *et al.* Nov 2003) is to go about half of the number of features, since that is when the combinations to sample half out of n items maximizes in order to create diversity. One restriction of the random decision tree as well as other state-of-the-art decision tree learners (such as c4.5 and dti) is that they all have a "single feature information gain bias". In other words, the data must have features with information gain individually but not "collectively." An example to show "single feature information gain bias" can be found in (Fan *et al.* Nov 2003).



(a) in range [0,1] (b) in range [0,0.35]

(a) in range [0,1] (b) in range [0,0.35]

Figure 1: donation: reliability plot of best and random tree



(a) depth = 0.5 (b) depth = 1.0

Figure 2: adult: reliability plot of best and random tree

Experiment

Data Set

We have chosen three datasets in this study. The first one is the famous donation dataset that first appeared in KDD-CUP'98 competition. Suppose that the cost of requesting a charitable donation from an individual $\mathbf{x}$ is \$0.68, and the best estimate of the amount that $\mathbf{x}$ will donate is $Y(\mathbf{x})$. Its benefit matrix (converse of loss function) is:

	predict $donate$	predict $\neg donator$
actual $donate$	Y(x) - \$.0.68	0
actual $\neg donate$	-\$0.68	0

The accuracy is the total amount of received charity minus the cost of mailing. Assuming that $p(donate|\mathbf{x})$ is the estimated probability that $\mathbf{x}$ is a donor, we will solicit to $\mathbf{x}$ iff $p(donate|\mathbf{x}) \cdot Y(\mathbf{x}) > 0.68$. The data has already been divided into a training set and a test set. The training set consists of 95412 records for which it is known whether or not the person made a donation and how much the donation was. The test set contains 96367 records for which similar donation information was not published until after the KDD'98 competition.

The second data set is a credit card fraud detection problem. Assuming that there is an overhead $v \in \{\$60, \$70, \$80, \$90\}$ to dispute and investigate a fraud and $y(\mathbf{x})$ is the transaction amount, the following is the benefit matrix:

	predict $fraud$	predict $\neg fraud$
actual $fraud$	$y(x)$ - v	0
actual $\neg fraud$	-v	0

The accuracy is the sum of recovered frauds minus investigation costs. If $p(fraud|\mathbf{x})$ is the probability that $\mathbf{x}$ is a fraud, $fraud$ is the optimal decision iff $p(fraud|\mathbf{x}) \cdot y(x) > v$. The dataset was sampled from a one year period and contains a total of .5M transaction records. The features (20 in total) record the time of the transaction, merchant type, merchant location, and past payment and transaction history summary. We use data of the last month as test data (40038 examples) and data of previous months as training data (406009 examples).

The third dataset is the adult dataset from UCI repository. We use the natural split of training and test sets. The training set contains 32561 entries and the test set contains 16281 records. The feature set contains 14 features that describe the education, gender, country of origin, martial status, capital gain among others. Each data item indicates if an individual earns more than \$50K or less.

Decision Tree Learner

The experiments were based on C4.5 release 8 (Quinlan 1993). The single best tree is constructed by running C4.5 with all the default settings. It computes both the unpruned and pruned decision trees. The random decision tree algorithm is a modified version of C4.5 that does not compute information gain. It chooses remaining features randomly, i.e., discrete feature is chosen only once in a decision path and continuous feature can be chosen multiple times yet with a different splitting threshold. After a splitting feature is chosen, the data items with the same feature values are grouped. The procedure is run recursively until either the predefined tree depth is met or the splitted node becomes empty.

Results

Reliability Plot "Reliability plot" shows how reliable the "score" of a model is in estimating the empirical probability of an example $\mathbf{x}$ to be a member of a class y. To draw a reliability plot, for each unique score value predicted by the model, we count how many examples in the data have this same score (e.g.,N), and how many among them have class label y (e.g.,n). Then the empirical class membership probability is simply $\frac{n}{N}$. Most practical datasets are limited in

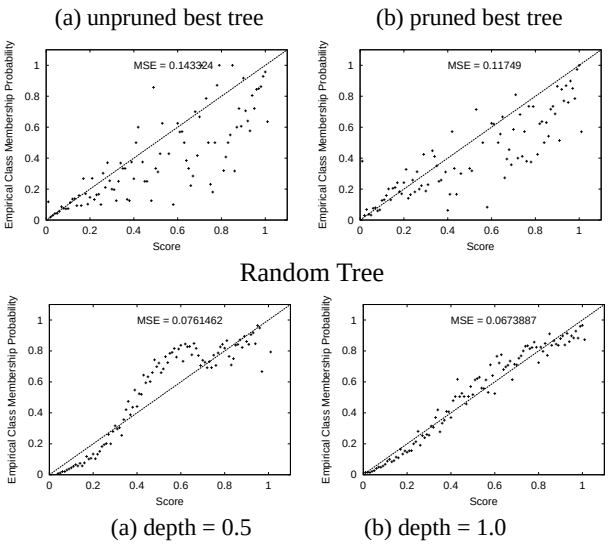


Figure 3: ccf: reliability plot of best and random tree

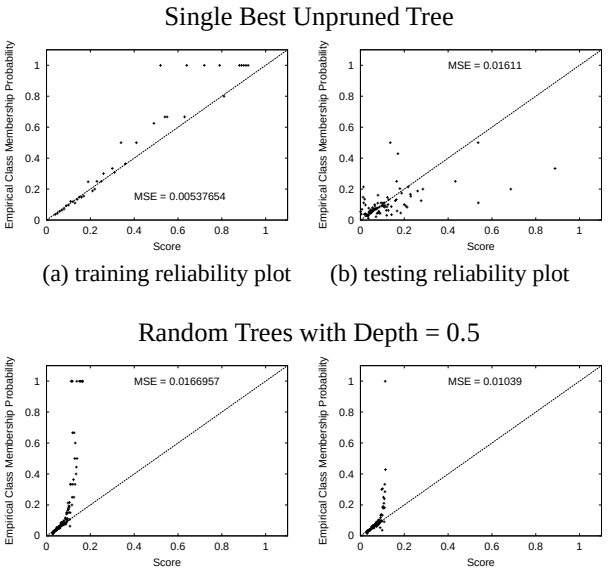


Figure 4: donation: reliability plot of best and random tree show overfitting of single best tree and "non-overfitting" by random tree

size; some scores may just cover a few number of examples and the empirical class membership probability can be extremely over or under estimated. To avoid this problem, we normally divide the range of the score into continuous bins and compute the empirical probability for examples falling into each bin.

The reliability plots for single best tree and random tree are shown in Figures 1 to 3. Figure 1 is for the donation dataset. There are 1000 bins of equal size; in other words, each bin covers a range of 1.0/1000 = 0.001. The x-axis is the predicted score by the model and the y-axis is the matching empirical class membership probability measured from the data. To visually compare the performance, we also draw the perfect straight line where every score exactly matches the true probability. The top two plots are for single best *unpruned decision tree*. Since the default pruning procedure of C4.5 deems every internal node of the unpruned decision tree as statistically insignificant, the pruned decision tree has just one useless node predicting everybody is a non-donor. Since most points are within the range between 0 and 0.35, we "enlarged" that area on a separate plot on the immediate right. As we can see clearly that the single best unpruned tree's score is like a "cloud" that scatters around the perfect matching line. However, the random decision tree (with depth of 0.5) matches the true probability very well. To summarize these results, we use *mean square error* or MSE to measure how closely the score matches the empirical probability. Assuming that n_j is the number of examples covered in bin j, s_j is the score or predicted probability and p_j is the empirical probability, then $MSE = \frac{\sqrt{n_j \cdot (s_j - p_j)^2}}{\sum n_j}$. The MSE for single best unpruned tree and random tree are labelled on the top of each reliability plot. The MSE for single best unpruned tree is 0.01611 while the MSE for random tree is 0.01039.

The reliability plots for the adult and credit card fraud data sets are shown in Figures 2 and 3 respectively (with bin size 100). As we can see clearly that both single best pruned and unpruned tree's posterior probability is far from the empirical class membership probability; the visual effect is that the empirical probability scatters widely around the score value, and this is particularly true in the middle range around 0.5. However, the random tree's probability estimates are very close to the true empirical probability throughout the whole range.

Overfitting

One interesting observation we have found is that random decision tree doesn't overfit the training data at all. To study the effect of overfitting, we have run an additional test that "trains on the *test data* and then predicts on the same test data" (called "training reliability plot") and compare it with the previous results of "training on *train data* and test on the test data" (called "testing reliability plot"). The results on donation and adult datasets are shown in Figures 4 and 5. One important point is that the exact training reliability plot of a single best tree should be the one where every unique score falls exactly on the "perfect line". Since the whole 0-1 probability range is divided into equal sized bins and we have finite number of examples, there will be some small errors. Visually, some of the points may not be exactly on the "perfect line". However, this is sufficient for comparing single best and random tree because they use exactly the same bin size.

As we can see clearly from Figure 4 and 5, for the single best decision tree, there is a big difference between the training reliability plot and the testing reliability plot. The training reliability plot on the left becomes much less than

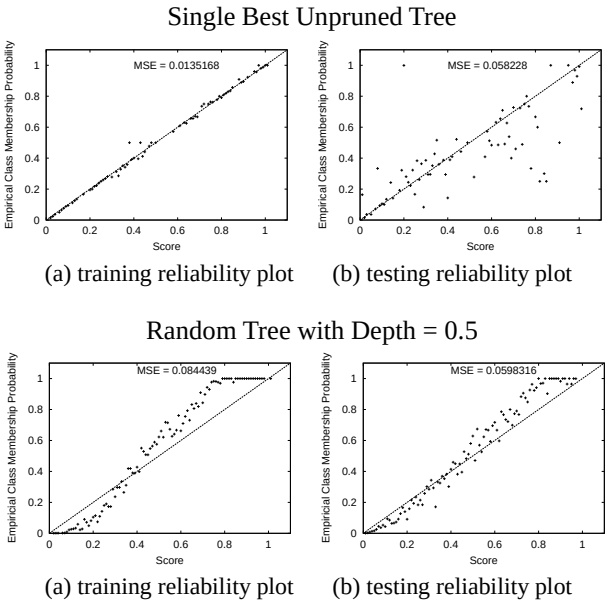


Figure 5: adult: reliability plot of best and random tree to show overfitting of single best tree and "non-overfitting" of random trees

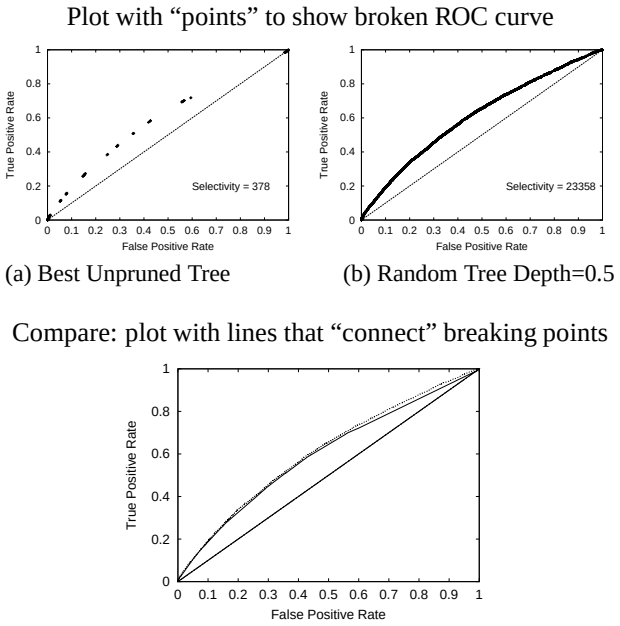


Figure 6: donation: exhaustive ROC plot of best and random tree

perfect when the same model is tested on a different dataset, as shown in the testing reliability plot on the right. In the testing reliability plots, we can see that the "points" are scattered widely around the perfect line. The MSE for training and testing on the same dataset is 0.005 [1], and MSE for training and testing on a different dataset is 0.016. On the other hand, the training and testing reliability plots for the random tree are nearly identical for both the donation and adult datasets.

Selectivity

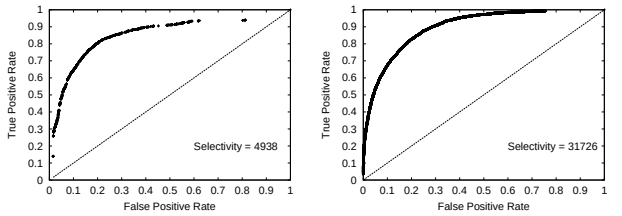
One important, but often ignored, property of a model is its "selectivity". The problem of selectivity comes from the fact that most models, especially decision trees, can only output a discrete set of "unique probability values" since there are only a limited number of terminal nodes and each node can just output one probability value. This fact has a serious practical consequence. With different threshold values, there will be only a limited number of true positive and false positive values. When we draw the paired true positive and false positive values on the ROC curve, we will actually never see a continuous line ranging from (0,0) to (1,1). What we will see is a discrete set of points that are scattered in this range. Any area *without a point* is undefined. Some suggest to draw a line connecting these defined points. However, the area connecting the points are undefined; we cannot choose any undefined point since there is actually no value there at all. Due to this reason, it is practically not very useful to connect points at all.

[1]The MSE is computed using binning, so it still has a non-zero value

We define "selectivity" to measure how good is an ROC curve. It takes two facts into account. The first fact is how uniform it is between consecutive false positive values. When all available false positives spread evenly between 0-1, the variance between consecutive values will be 0. In other words, the lower this variance is, the more evenly false positive values spreads. The second fact is how many unique false positive values there are. The more unique values there are, the more choices we can choose from. So we define selectivity as the following $d = \frac{u}{s}$, where s is the variance between consecutive false positive values and u is the number of unique false positive values.

We have computed ROC curves for adult and donation datasets exhaustively by choosing all unique probability values in posterior probability output as the decision threshold and record its corresponding false positive and true positive rates. Any duplicate entries are removed and the results are shown in Figures 6 and 7. To show the idea of "selectivity" clearly, we plot with "points" rather than "lines" on the two top plots. This way, any areas in ROC without false positive and true positive rates will be clearly exposed. In each ROC curve, we have also shown the "selectivity" measure as defined previously. The "selectivity" problem for single best decision tree is clearly shown, particularly in the donation dataset. For the donation dataset as shown in Figure 6, the single best tree only has 11 broken clusters spread out the range, while the random tree has visually continuous ROC curve. The selectivity measure is 238 for single unpruned tree and 23358 for random tree (100 times higher than 238). As shown in Figure 7, the ROC curves for adult data have similar phenomenon, but the single best tree's ROC is not as broken as that of the donation data. In the bottom part

Plot with “points” to show broken ROC curve



(a) Best Unpruned Tree (b) Random Tree Depth=0.5

Compare: plot with lines that “connect” breaking points

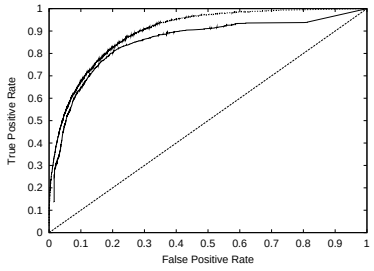


Figure 7: adult: exhaustive ROC plot of best and random tree

of Figure 6 and Figure 7, we plot the ROC curves of single best unpruned tree and random tree in the same figure. For visual clarity, we plot with “lines” to connect broken curves. Strictly speaking, any points not in the top exhaustive ROC curve is not defined. This is just to compare the difference in true positive value for the same false positive value. As we can see in the ROC curve, for the donation dataset, true positive of the random tree is mostly the same or slightly higher than single best unpruned tree for the same false positive value. For the adult data set, the random tree’s true positive rate is consistently higher than that of single best unpruned tree.

Related Work

Breiman has proposed the “random forests” method (Breiman 2001). In random forests, randomness is injected by randomly sampling a subset of remaining features (those not chosen yet by a decision path) and then choosing the best spliting criteria from this feature subset. The chosen size of the subset has to be provided by the user of random forests. However, in random decision tree, the spliting feature is randomly chosen from any remaining features not chosen yet in the current decision path. There is no information gain (or any other criteria) involved in choosing this feature and when it will be chosen. In other words, random decision tree doesn’t use any heuristics to choose feature. The data is used to update the class distribution in each node. However, in Breiman’s random forests, information gain or other criteria is still used to choose the best feature among randomly chosen feature subsets. This has great impact on the efficiency between training a random forest or a random tree since computing any heuristics is expensive. Another important distinction is that Breiman’s random forests performance simple voting on the final prediction. In other words, each tree votes 1 on one of the class labels. The class labels with the highest vote is the final prediction. However, random tree, each tree output raw probability and the probability outputs from multiple tree are averaged as the final probability. .

Conclusion

We have extensively studied the posterior probability estimation by previously proposed random decision method. We have found that random decision tree can faithfully estimate the empirical class membership probability. In the statistical reliability curve, the estimated probability by random decision tree closely matches the empirical class membership probability measured from the data. We also found that random decision tree’s ability to estimate posterior probability does not overfit at all. The probability reliability curve of the training set is very close to the reliability curve on a unseen validation or test dataset. We have also studied the “selectivity” of ROC curve and defined how to measure “selectivity”. We have found that the ROC curve of random decision tree is remarkably continuous throughout the range. Comparing with random decision tree, a single best decision doesn’t match the empirical class membership probability well, overfits on the training data easily, and has broken or discontinuous ROC curves.

References

Bradford, J. P.; Kunz, C.; Kohavi, R.; Brunk, C.; and Brodley, C. E. 1998. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, 131–136.

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

Fan, W.; Wang, H.; Yu, P. S.; and Ma, S. Nov 2003. Is random model better? on its accuracy and efficiency. In *Proceedings of Third IEEE International Conference on Data Mining (ICDM-2003)*.

Kearns, M., and Mansour, Y. 1996. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the Annual ACM Symposium on the Theory of Computing*, 459–468.

Mehta, M.; Agrawal, R.; and Rissanen, J. 1996. SLIQ: A fast scalable classifier for data mining. In *Extending Database Technology*, 18–32.

Mehta, M.; Rissanen, J.; and Agrawal, R. 1995. MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD’95)*, 216–221.

Quinlan, R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.