Homework 2: September 14, 2016

Machine Learning

Susan Cherry

Question 1

Part A) First, I calculate the value of g(x) for each data point. I find the following output:g(x)= [-1.70 , 0.70, 3.55 , 1.10, 0.65, -1.65, -1.55, 0.55, 3.65 ,-1.85], which correspond to the following y labels: y = [0, 1, 1, 1, 1, 0, 0, 0, 1, 0]. A natural choice of threshold would be 0. For any g(x)<=0, we classify as 0 and for any g(x)>0, we classify as 1. In this case, however, we would have one misclassification: We would classify the 8th element of the dataset as 1 (the g(x) output gives it a value of 0.55), while the y label is actually 0.

In order to minimize the classification error, we could use any value in the range of [0.55,0.65). This means any value greater than or equal to 0.55 but also less than 0.65. Let's call a number in this range k. For any g(x)<= k, we classify as 0 and for any g(x)> k, we classify as 1. Now, all of the points are correctly classified.

Below is the confusion matrix for a value in this range of thresholds. Because all points are correctly classified, there are no false positive or false negative points. Half of the points are true positives and the other half are true negatives (in this case, we can think of y=0 as a negative. If we wanted, we could relabel the y=0s as y=-1).

Part A Confusion Matrix

	y=1	y=0
$\hat{y}$=1	5	0
$\hat{y}$=0	0	5

Part B) Next, I calculate f(x) for each data point. I find the following output: f(x)=[0.1544653, 0.6681878, 0.9720774, 0.7502601, 0.6570105, 0.1611089, 0.1750863, 0.6341356, 0.9746673, 0.1358729], which again corresponds to the following labels: y= [0, 1, 1, 1, 1, 0, 0, 0, 1, 0]. In this case, I can chose the threshold in the range [0.6341356, 0.6570105) because the largest point with

y=0 has an f(x) value of 0.6341356 and the smallest point with y=1 has an f(x) value of 6570105, making any point in this range a good choice for correctly splitting all points into the training set into the correct labels. Again, this range means means any value greater than or equal to 0.6341356 but also less than 0.6570105. For example, let k be a value in the range (k is less than or equal to 0.6341356 and strictly less than 0.6570105). If f(x)<= k, it is classified as 0. If f(x)> k, it is classified as 1.

Below is the confusion matrix for this threshold. It looks identical to the confusion matrix found in Part A.

Part B Confusion Matrix

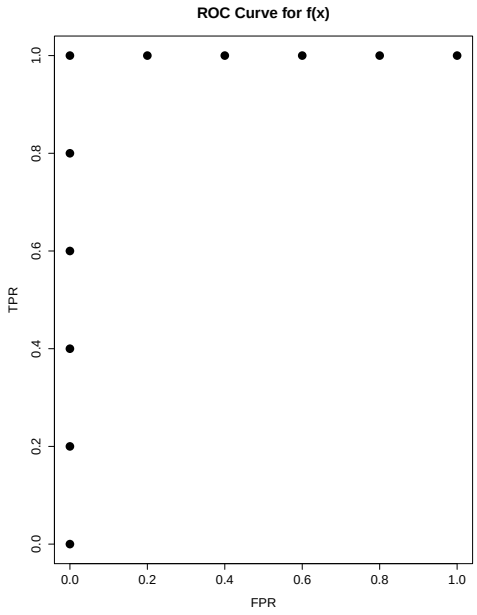
	y=1	y=0
$\hat{y}$=1	5	0
$\hat{y}$=0	0	5

We also find the following:
Precision: $= \frac{TP}{TotalPredictedPos.} = 1$
Recall: $= \frac{TP}{TotalLabeledPos.} = 1$
F1 Score: $= 2 * \frac{Precision*Recall}{Precision+Recall} = 1$

Part C) For this question, I'm asked to compute the ROC for the classifier f(x). The plots the FPR against the TPR. To do this, I change the threshold and calculate the False Positives and True Positives by threshold. Let k be the specific threshold. Then, I classify as follows: If f(x)<= k, it is classified as 0. If f(x)> k, it is classified as 1. Below is a chart of the FP and TP values I got by changing the threshold.

	Less than 0.1358729	0.1358729	0.1544653	0.1611089	0.1750863	0.6341356	0.6570105	0.6681878	0.7502601	0.9746673	0.1358729
FP	5	4	3	2	1	0	0	0	0	0	0
TP	5	5	5	5	5	5	4	3	2	1	0

Then, I convert the FP and TP values into False Positive Rates and True Positive Rates and plot them to get the ROC curve. See below.



It is clear from the ROC curve that the AUC is equal to 1.

Part D) Given a monotonic function $h(x)$, the ROC curve of $f(x)$ is exactly the same as the ROC curve $h(f(x))$. A monotonic transformation preserves the order of a set of numbers, so the FPR and TPR will remain the same, leading to an unchanged ROC curve.

Question 2

Part A) To determine the best feature to split on, I need to calculate the Gini Reduction for each feature. First, notice that there are 5 "0"s and 5 "1"s labeled in the Y column. So the original Gini Index is $GiniIndex(0.5, 0.5) = 2(0.5)(0.5) = 0.5$

X_1: In this feature, we have 4 "1" values and 6 "0" values. $GiniReduction = 0.5 - (\frac{4}{10} * 2 * 1 * 0) - (\frac{6}{10} * 2 * \frac{5}{6} * \frac{1}{6}) = 0.5 - \frac{1}{6} = \frac{1}{3}$
X_2: In this feature, we have 5 "1" values and 5 "0" values. $GiniReduction = 0.5 - (\frac{1}{2} * 2 * \frac{3}{5} * \frac{2}{5}) - (\frac{1}{2} * 2 * \frac{2}{5} * \frac{3}{5}) = 0.5 - \frac{24}{50} = \frac{1}{50}$
X_3: In this feature, we have 6 "1" values and 4 "0" values. $GiniReduction = 0.5 - (\frac{6}{10} * 2 * \frac{4}{6} * \frac{2}{6}) - (\frac{4}{10} * 2 * \frac{3}{4} * \frac{1}{4}) = \frac{1}{12}$.

X_1 has the largest Gini Reduction, so I would split on this feature.

Part B) Next, I do calculate the information gains for each feature. First,

notice that $H(0.5, 0.5) = 1$ is the beginning entropy.

X_1: $InformationGain = H(1/2, 1/2) - \frac{4}{10}H(0, 1) - \frac{6}{10}H(5/6, 1/6) = 1 - 0.6(-\frac{5}{6}log_2(\frac{5}{6}) - \frac{1}{6}log_2(\frac{1}{6})) = 0.60998654701$
X_2: $InformationGain = H(1/2, 1/2) - \frac{1}{2}H(3/5, 2/5) - \frac{1/2}{10}H(3/4, 1/4) = 1 - 0.5(-\frac{3}{5}log_2(\frac{3}{5}) - \frac{2}{5}log_2(\frac{2}{5})) - 0.5(-\frac{2}{5}log_2(\frac{2}{5}) - \frac{3}{5}log_2(\frac{3}{5})) = 0.02904940554$
X_3: $InformationGain = H(1/2, 1/2) - \frac{6}{10}H(4/6, 2/6) - \frac{4}{10}H(3/4, 1/4) = 1 - 0.6(-\frac{4}{6}log_2(\frac{4}{6}) - \frac{2}{6}log_2(\frac{2}{6})) - 0.4(-\frac{1}{4}log_2(\frac{1}{4}) - \frac{3}{4}log_2(\frac{3}{4})) = 0.12451124978$

X_1 has the largest information gain, so I would also split on this feature if I was using information gain instead of the gini reduction as the splitting criteria.

Question 2.1

See hand drawn diagrams appended to the end of this pdf.

Question 2.2

Part A) Let x be drawn from the uniform distribution from [0,1]. If $x_i \in [0, \delta], y_i = 1$ and if $x_i \in [\delta, 1], y_i = -1$. ($p(y_i = 1|x_i < \delta) = 1$ and $p(y_i = -1|x_i >= \delta) = 1$). Because our training set has n_δ points, it will contain nearly $n_\delta * \delta$ 1 labels and $1 - \delta - 1$ values. Our goal is to have our decision tree correctly predict the 1 and -1 y labels. From our training set, we can find the minimum value x_i of all of the x_i values with the -1 label. We will call this value k. Thus, if we split on$k$, all $x_i$values such that $x_i >= k$ will be labeled as -1 and all x_i values $< k$ will be labeled as 1.

We can prune with decision trees, however. Specifically, with the CART algorithm, we can chose the value of C to determine the cost of subtrees and decide where to prune. Assume that δ is very large, say 0.9999. Then nearly all of the training set consists of points labeled 1 and the value of k is very near 1. If we set the C value high enough, then the decision tree will decide to prune and the decision tree will collapse into only 1 node, classifying everything as 1. When n_δ is large, the probability of making a mistake is $1 - n_\delta$ because after our pruning, the tree classifies all points as 1, including those that should be negative.

Random forests, we have many decision trees and we do not do pruning. This is the key point to this problem. Decision trees prune, so we can over or underfit them depending on how we adjust the parameters. Above, I adjusted the C parameter so that the decision tree underfits. Random forests tend to overfit, however, because we don't do the pruning process. In this case, a random forest would perform better than the decision tree described above because it creates many different decision trees, doesn't prune them, and then takes the majority vote. The trees in the decision random forest are able to classify the negative

labels, avoiding the problem we had above with the pruned decision tree.

Part B) For this question, we try to find a distribution for which decision trees have a high probability of performing better than random forests. First consider the following training set consisting of 2 pairs of (x,y) inputs. Note that in this example, there is only one feature and that y is the label. Suppose we have (0,-1), and (1,1). x only takes on the values of 0 and 1. $p(x = 0) = 0.5$ and $p(x = 1) = 0.5$ and $p(y = 1|x = 1) = 1$ and $p(y = -1|x = 0) = 1$. The decision tree could split based on whether x is 0 or 1. If x=0, the decision tree classifies as -1 and if x=1, the decision tree classifies as 1. Thus, it is completely accurate.

Random forest, however uses bootstrapping, so n samples are drawn from replacement. Therefore, it is possible for the random forest to draw the following sample: (0,-1), (0,-1) if n=2. In this case, it would classify all points as -1, because none of the sample points are labeled 1. There is a $(1/2)^2$ probability of a tree in the random forest picking all 1 labels and a $(1/2)^2$ probability of it picking all -1 labels. If we have three trees in the forest for example, and two trees draw the (0,-1), (0,-1) sample, then these trees will always classify as -1. If the third tree uses the (0,-1), (1,1) sample, it will classify correctly, but because random forests use the majority vote rule, the random forest will classify all points as -1 anyway. (If the random forest is trying to classify $x = 1$, the first two trees will automatically classify it as -1. The third will classify correctly as 1, but it will be overruled.). In this case, the random forest would only classify half of the training set correctly, while the decision tree classifies all of the training set correctly.

As n increases, the likelihood of the random forest behaving this way decreases because the likelihood of the random forest drawing a bootstrap sample of points only labeled -1 or only labeled 1 will decrease. Assuming that our training set consists of half -1 and half 1 labels, then the probability of all of the labels in the bootstrap sample having the value 1 is $(1/2)^n$ and the probability of all having the value -1 is $(1/2)^n$, which goes to 0 and n goes to infinity. Therefore, as n increases, the advantage of the decision tree over the random forest will decrease.