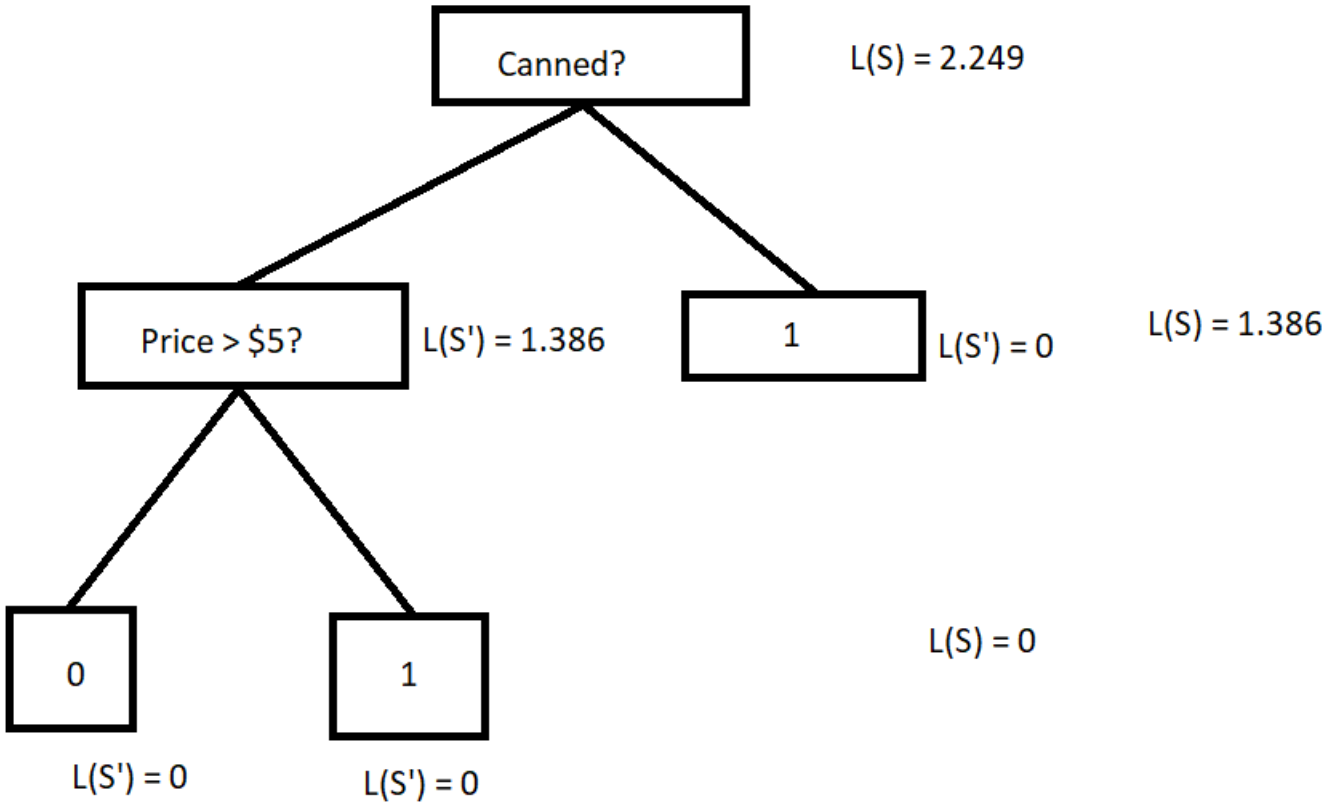
CS155 Homework 3

Ty Limpasuvan

7 late hours used; 31 late hours remaining

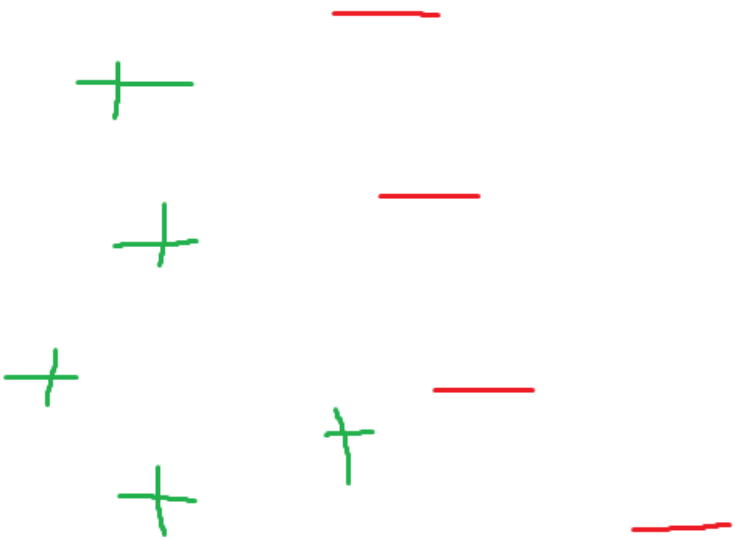
1 Comparing Different Loss Functions

1.1 A



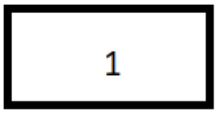
1.2 B

Decision trees are not always preferred over linear classifiers for classification problems. Below is an example of a 2D data set that can easily be classified by a linear classifier but requires a complex decision tree.



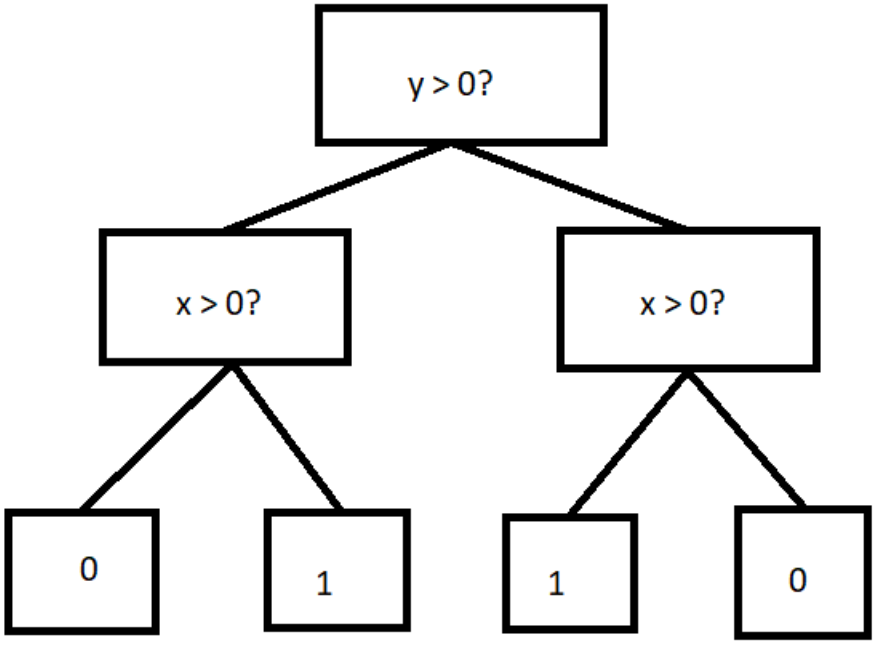
1.3 C

1.3.1 i



The Gini index of the starting (entire) set is 2. When I attempt to split the node based on X or Y value, the Gini index of both new nodes is 1. 1+1=2, so there would be no reduction of impurity from a split. The error of classification is 0.5.

1.3.2 ii



Taking the entropy measure but using the number of points squared at the front of the equation instead of just the number of points would be an example of an impurity measure that would produce the tree. The pro is that the training data would be correctly classified, but the con is that it would be heavily overfitted.

1.3.3 iii

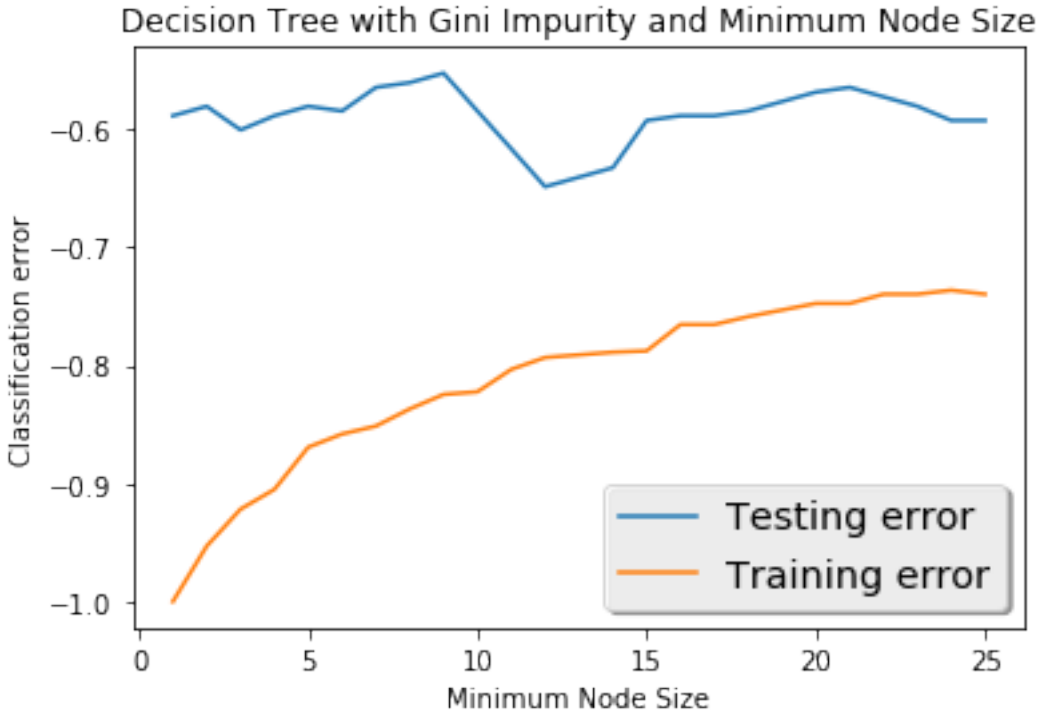
In the worst possible case, 99 unique leaf nodes could be needed. Having 100 nodes implies that at least 1 node is useless because of the fact that there 100 points.

1.4 D

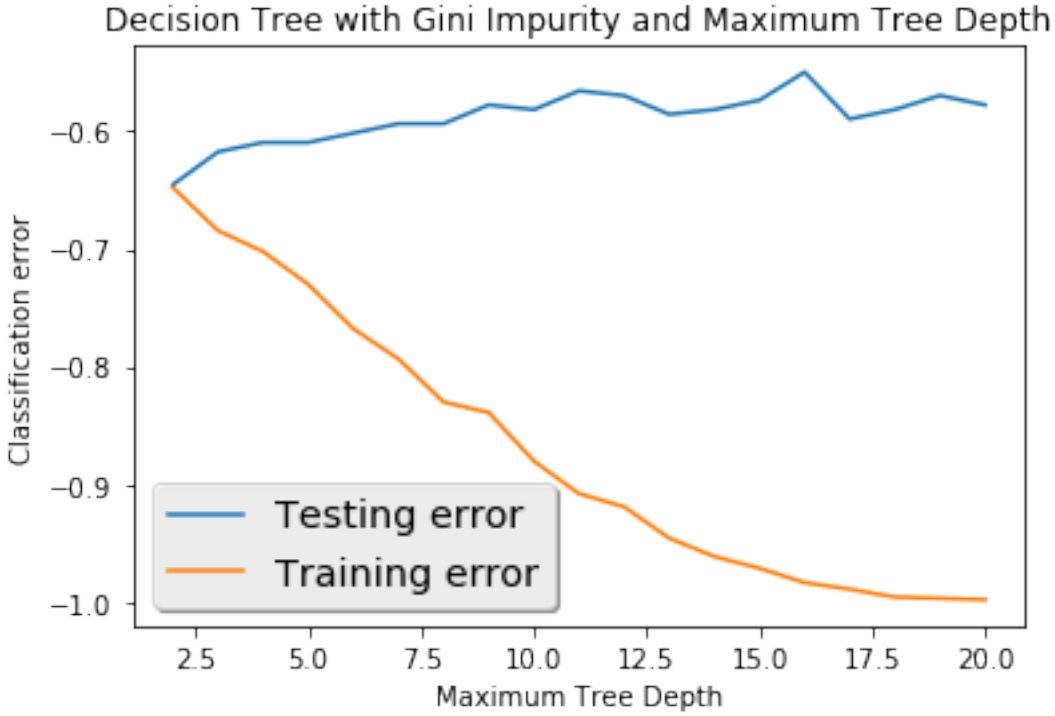
The worst case time complexity is $O(N * D)$. This counts for the looking at all possible splits (for each feature), and this needs to be done for each point (N).

2 Overfitting Decision Trees

2.1 A



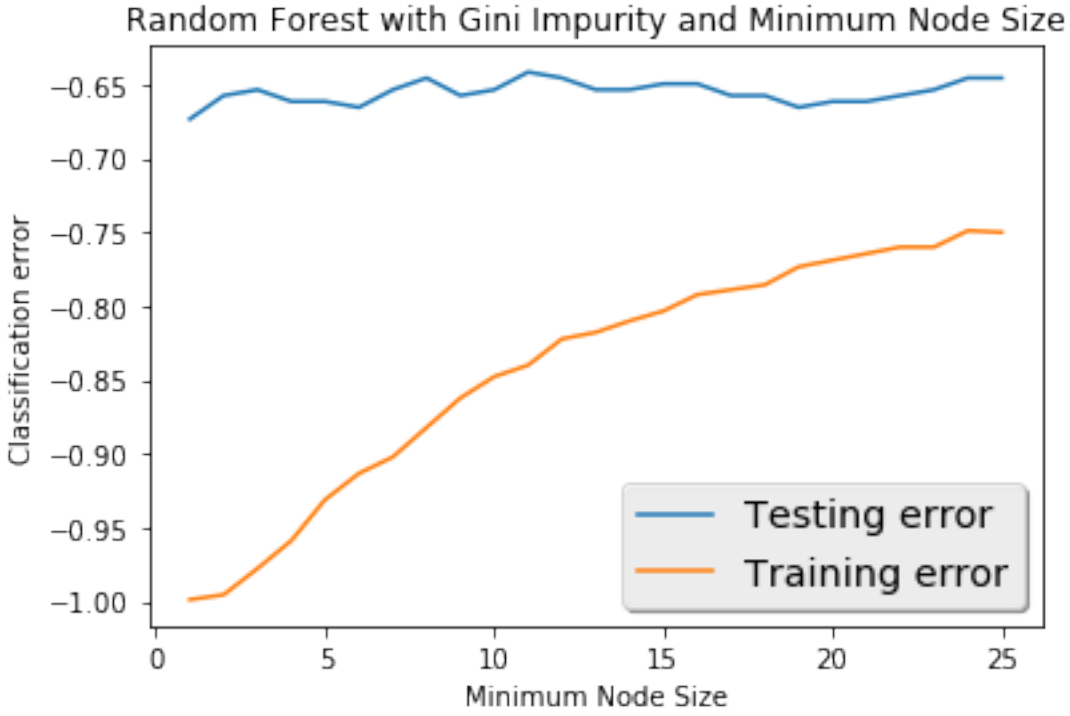
2.2 B



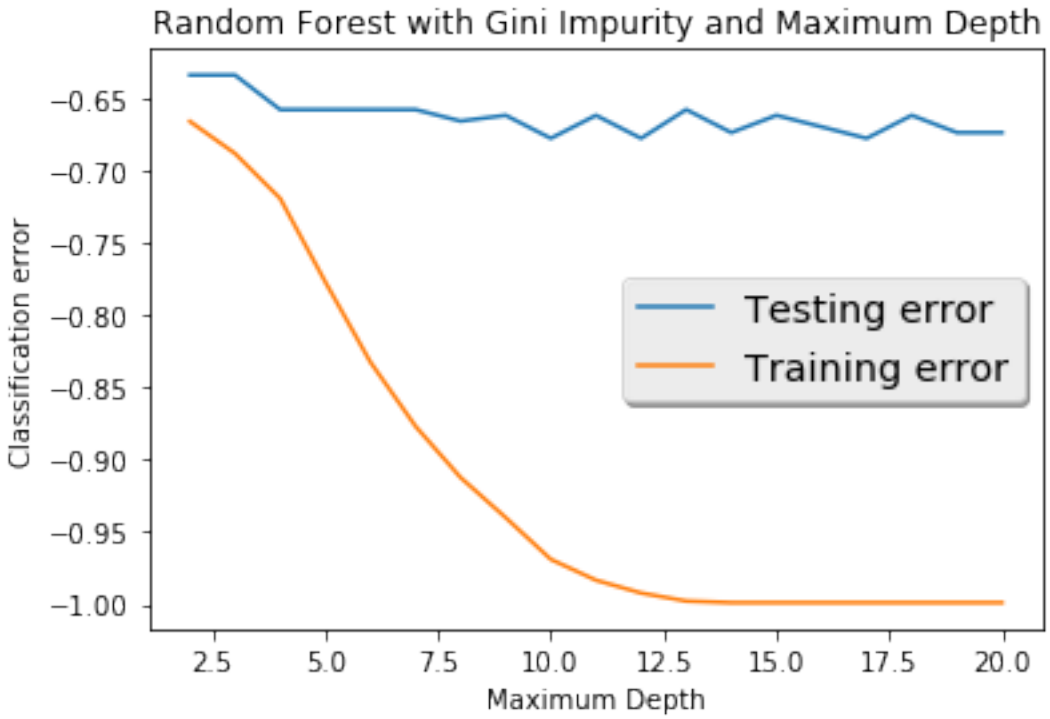
2.3 C

For min samples leaf, the value 12 minimized the test error. For max depth, the value 2 minimized the test error. It seems that stopping the runs early would help prevent overfitting; test error increases after these values are passed.

2.4 D



2.5 E



2.6 F

Test error was minimized for min samples leaf with a value of 1. It was minimized for max depth at a value of 10. Stopping the run early wouldn't have a large effect on the testing error, but the training error decreased with greater value of min samples leaf and increased with values of max depth.

2.7 G

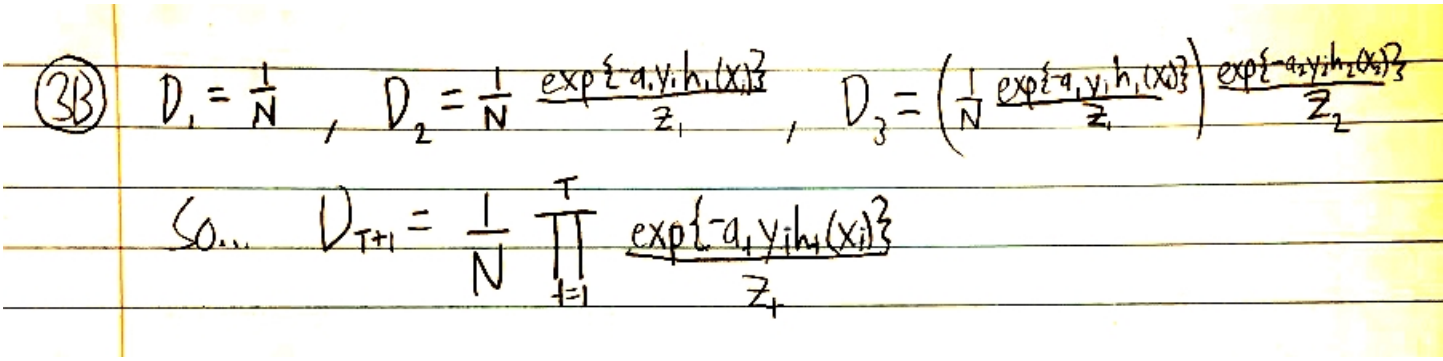
The rate at which the training error values increased or decreased were greater in the Random Forest runs, and the testing error values were more consistent in the Random Forest runs as well. I think that the training error aspect comes from the number of points used, and Random Forests keep the training error more consistent because they deal with the mode of the classifications and/or the mean predictions of individual trees.

3 The AdaBoost Algorithm

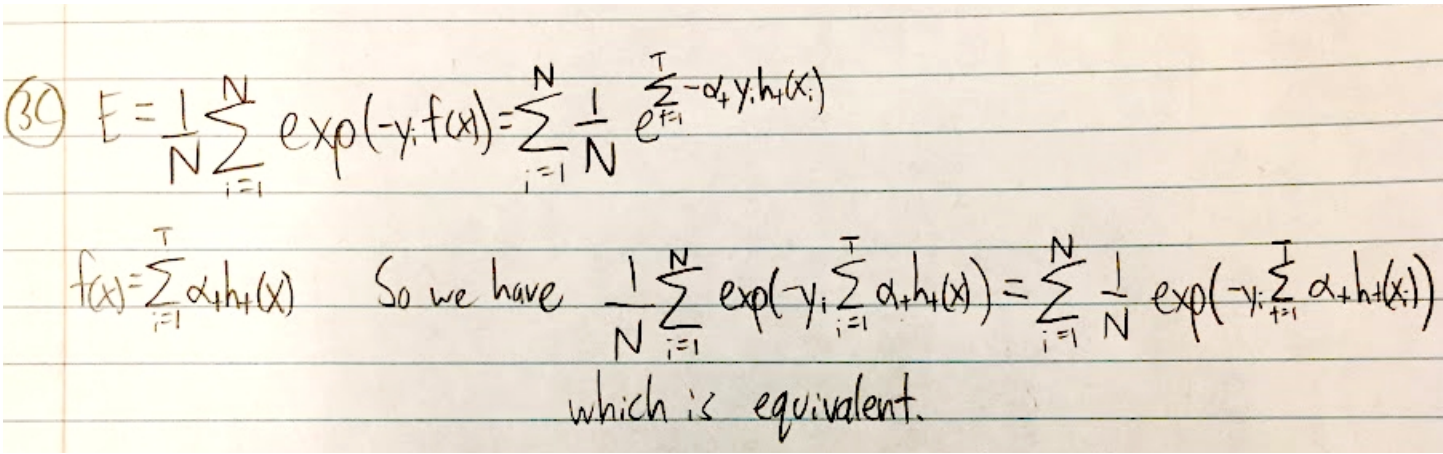
3.1 A

For the right side of the expression (the 1/0 loss) each value of i will produce either a 1 or a 0. In the case that a 1 is given, it means that a point was misclassified and a 1 is added to the current sum. However, on the left side of the expression, each value of i will produce the exp of either a positive or negative number. If a point is misclassified, the right side produces 1 and the left side gives exp or a positive number, which is greater than 1. If a point is classified correctly, the right side produces a loss of 0 and the left side produces a small positive number less than one (exp of a negative number). So in any case, the exponential loss upper bounds the training set error of the final classifier.

3.2 B


(3B) $D_1 = \frac{1}{N}$, $D_2 = \frac{1}{N}\frac{\exp\{-\alpha_1 y_i h_1(x_i)\}}{Z_1}$, $D_3 = \left(\frac{1}{N}\frac{\exp\{-\alpha_1 y_i h_1(x_i)\}}{Z_1}\right)\frac{\exp\{-\alpha_2 y_i h_2(x_i)\}}{Z_2}$
So... $D_{T+1} = \frac{1}{N}\prod_{t=1}^{T}\frac{\exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$

3.3 C


(3C) $E = \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i f(x_i)) = \sum_{i=1}^{N}\frac{1}{N}e^{\sum_{t=1}^{T}-\alpha_t y_i h_t(x_i)}$
$f(x) = \sum_{t=1}^{T}\alpha_t h_t(x)$ So we have $\frac{1}{N}\sum_{i=1}^{N}\exp(-y_i\sum_{t=1}^{T}\alpha_t h_t(x)) = \sum_{i=1}^{N}\frac{1}{N}\exp(-y_i\sum_{t=1}^{T}\alpha_t h_t(x_i))$
which is equivalent.

3.4 D

(3D) We have $E = \frac{1}{N}\sum_{i=1}^{N}\prod_{t=1}^{T}\exp(-y_i a_t h_t(x))$

$$= \sum_{i=1}^{N}\frac{1}{N}\exp(-y_i a_1 h_1(x))\prod_{t=2}^{T}\exp(-y_i a_t h_t(x))$$

$$= \sum_{i=1}^{N} D_1 Z_1 \prod_{t=2}^{T}\exp(-y_i a_t h_t(x))$$

$$= Z_1\sum_{i=1}^{N} D_2 \exp(-y_i a_2 h_2(x))\prod_{t=3}^{T}\exp(-y_i a_t h_t(x))$$

$$= Z_1 Z_2\sum_{i=1}^{N} D_3 \prod_{t=3}^{T}\exp(-y_i a_t h_t(x))$$

$$\cdots = \prod_{t=1}^{T} Z_T \sum_{i=1}^{N} D_t = \prod_{t=1}^{T} Z_T$$

3.5 E

(3E) $$Z_t = \sum_{i=1}^{N} D_t(i)\exp(-a_t y_i h_t(x_i)) = \sum_{i=1}^{N} D_t(i)\exp(-a_t) + \sum_{i=1}^{N} D_t(i)\exp(a_t)$$

$y_i h_t(x_i) = 1$ $\qquad$ $y_i h_t(x_i) = -1$

We can use the indicator function instead of checking if $y_i h_t(x_i) = 1$ or -1.

This gives $(1-\varepsilon_t)\exp(-a_t) + \varepsilon_t \exp(a_t)$

if $\varepsilon_t = \sum_{i=1}^{N} D_t(i)\,\mathbb{1}(h_t(x_i) \neq y_i)$

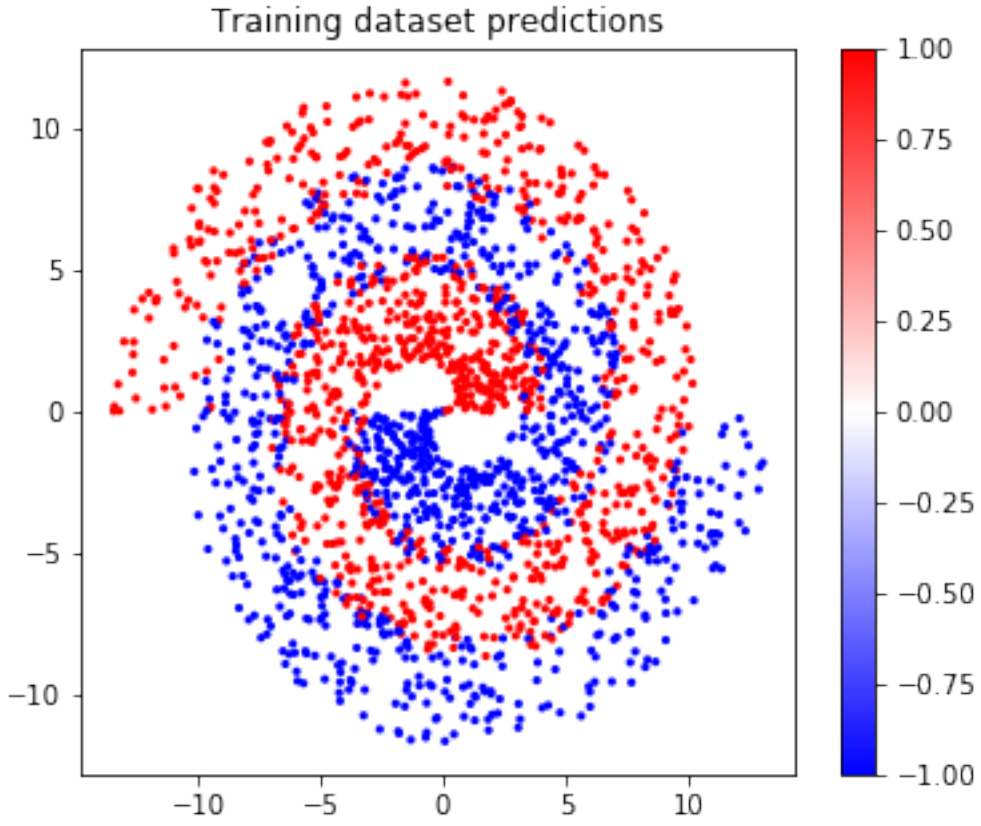
3.6 F

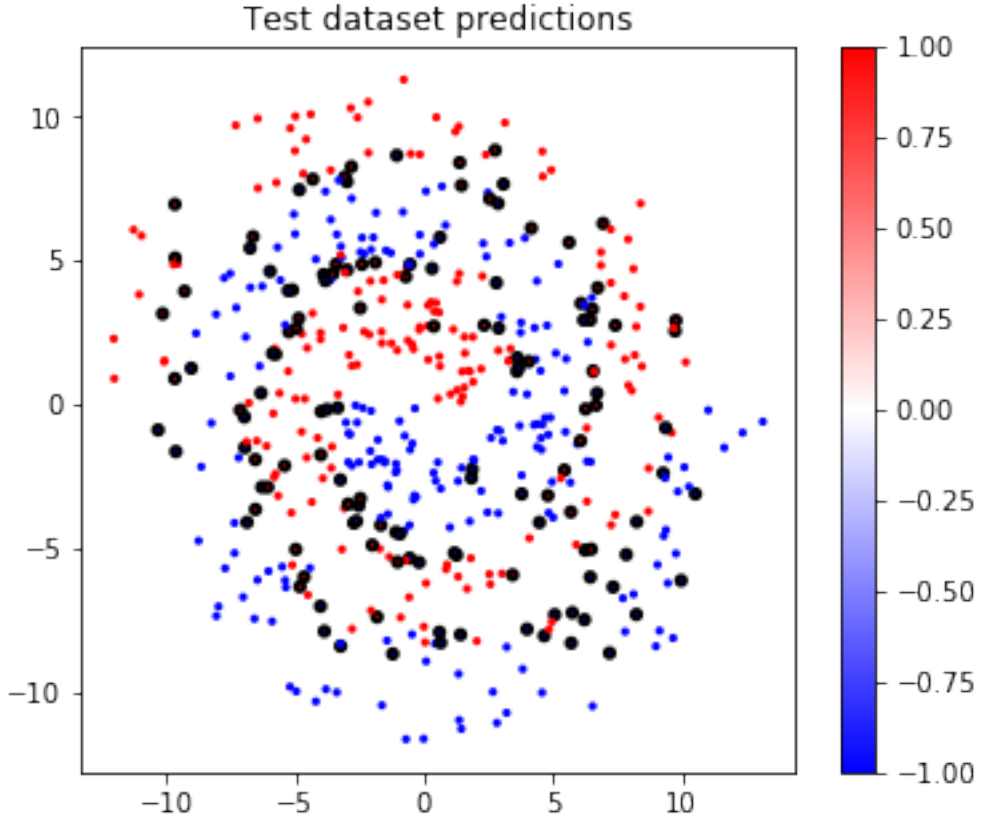
(3F) We minimize the training error by taking the derivative and setting it equal to zero.

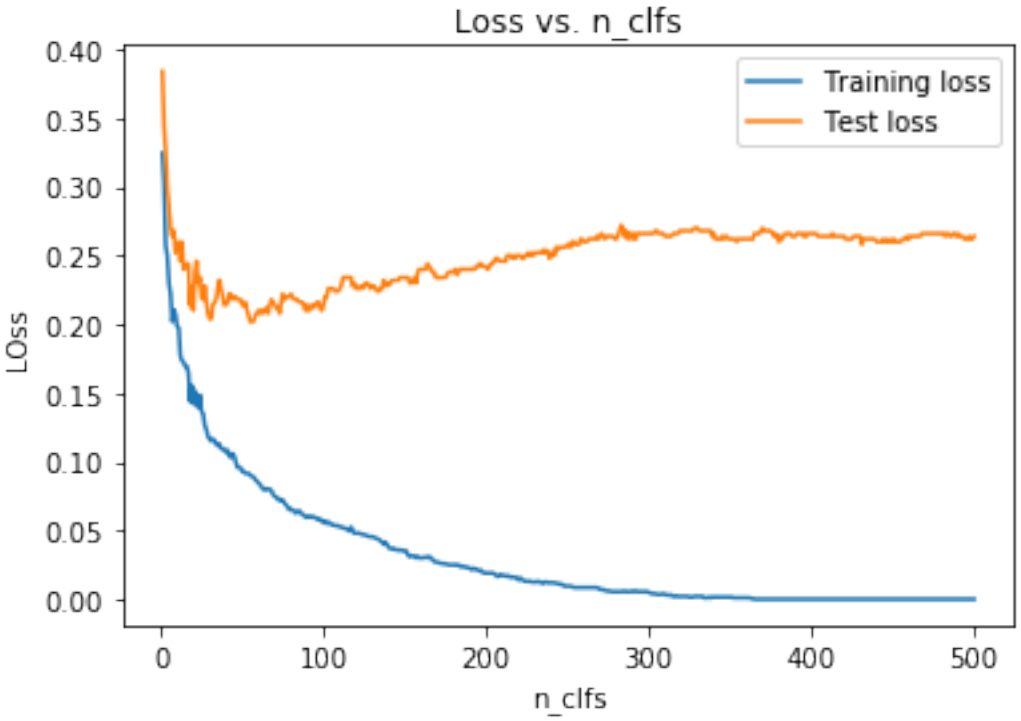
$$\frac{\partial}{\partial \alpha}\left((1-\varepsilon)e^{-\alpha} + \varepsilon e^{\alpha}\right) = 0$$

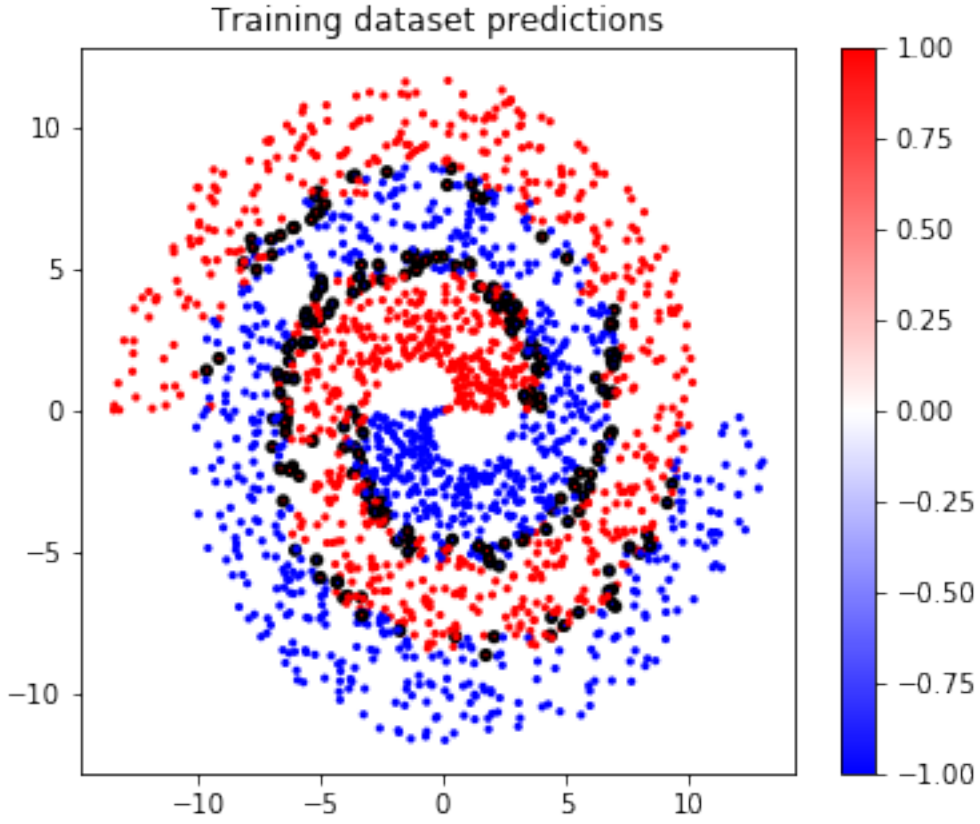
$$(\varepsilon - 1)e^{-\alpha} + \varepsilon e^{\alpha} = 0 \rightarrow \alpha = \frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$$

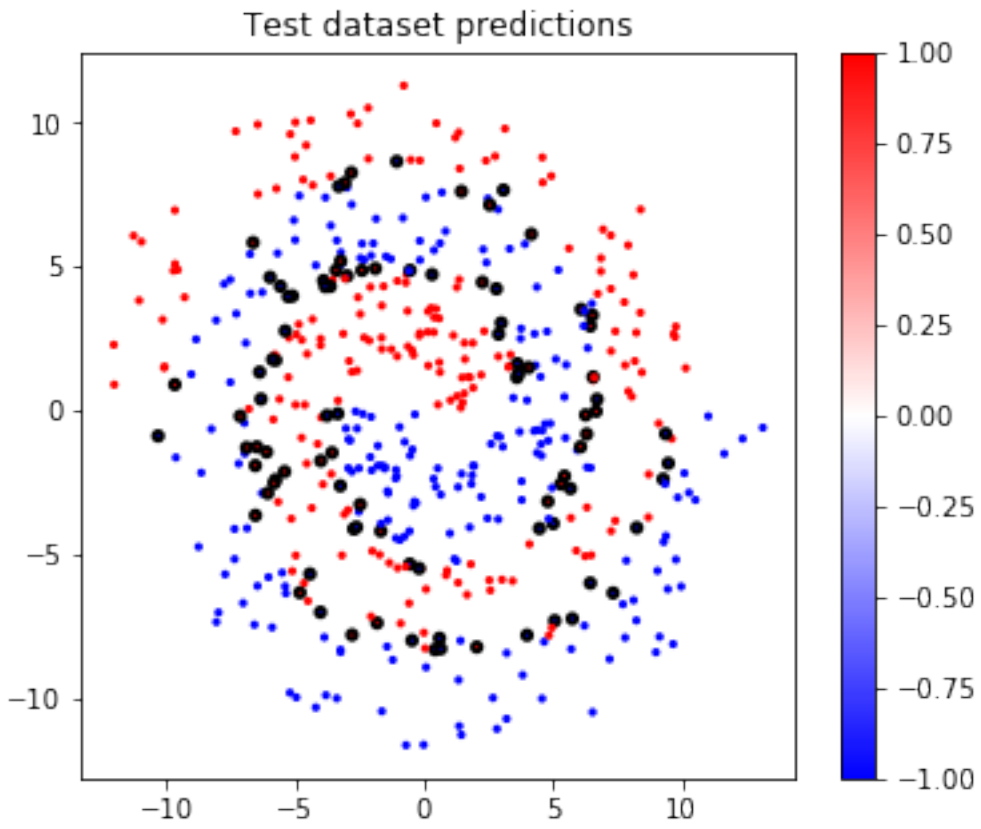
3.7 G

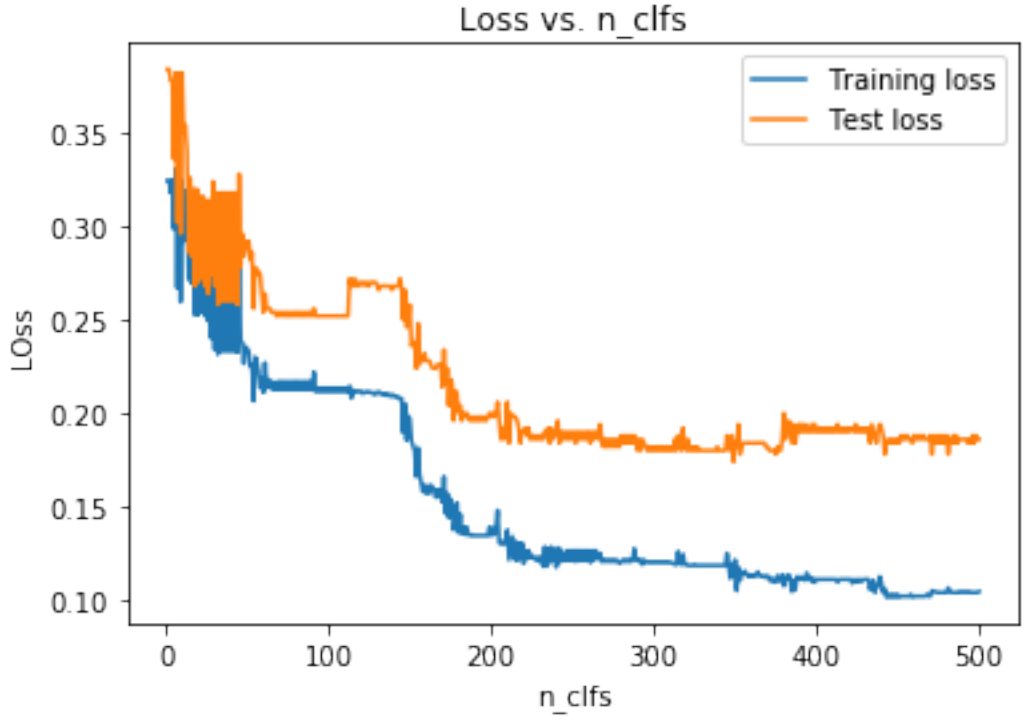

Training dataset predictions
10
5
0
−5
−10
−10
−5
0
5
10
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Test dataset predictions
10
5
0
−5
−10
−10
−5
0
5
10
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Loss vs. n_clfs
Training loss
Test loss
LOss
n_clfs
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
100
200
300
400
500


Training dataset predictions
10
5
0
−5
−10
−10
−5
0
5
10
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Test dataset predictions
10
5
0
−5
−10
−10
−5
0
5
10
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Loss vs. n_clfs
Training loss
Test loss
LOss
0.35
0.30
0.25
0.20
0.15
0.10
0
100
200
300
400
500
n_clfs

3.8 H

The curves of the losses start out extremely rigid and smooth out over time. The losses also decreased with the number of classifiers. This is because the boosting and high number of classifiers decrease the losses.

3.9 I

AdaBoost performed much better. The test loss ended at around 0.2, and the training loss ended at around 0.11.

3.10 J

The weights appear to be greatest toward the center of the data set.