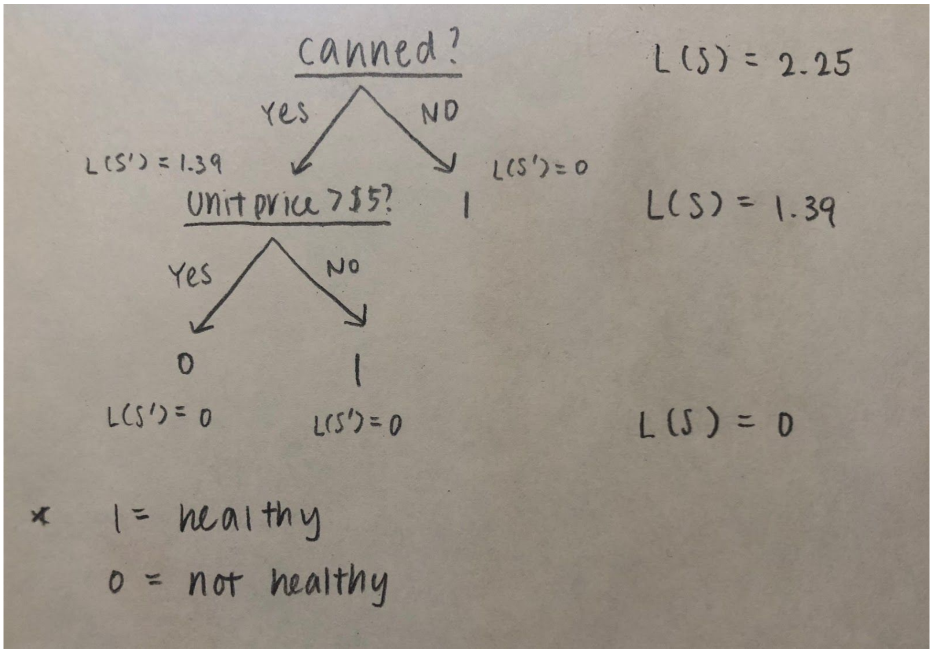
1 late hour used

CS 155 PS 3

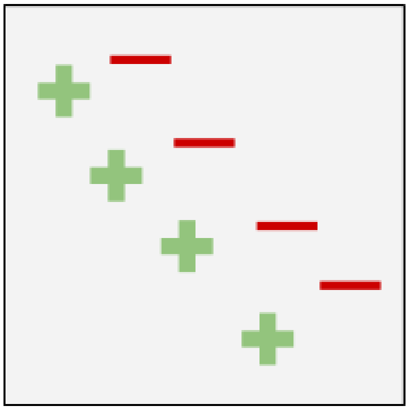
1 Decision Trees

Problem A



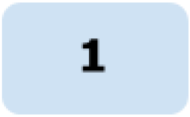
Problem B

A decision tree is not always preferred over linear classifiers for classification problems, because they are axis-aligned and cannot easily model diagonal boundaries. For example, this simple 2-D dataset in the drawing can be perfectly classified by a simple linear classifier but requires an overly complex decision tree to perfectly classify.



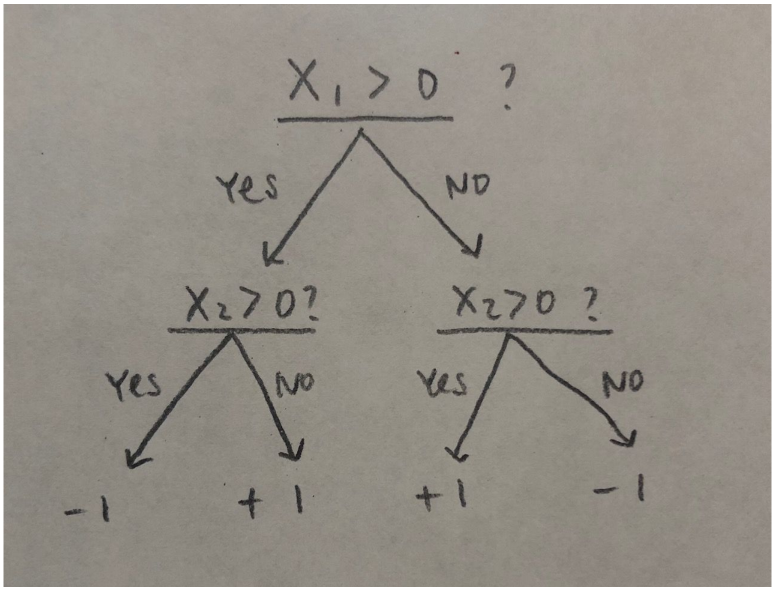
Problem C

i. The resulting tree is just the root node. Since there is no majority class, the root node's prediction could be either 1 or -1 (chosen with probability 0.5).



Its classification error is 0.5 whether the prediction is 1 or -1.

ii.



A modified version of Gini Index ($L(S') = |S'|^2 (1 - p_{S'}^2 - (1 - p_{S'})^2)$) with the |S'| term squared would have led top-down greedy induction with the same stopping condition to produce this tree, because the first split results in reduction in impurity from the initial dataset. The original Gini Index doesn't reduce impurity with a partition that may be beneficial in later splits (the linearity causes the the total impurity to stay constant if the fraction of positive values stays constant), causing the decision tree training to terminate too early. This modified Gini Index addresses this issue that arises with the stopping condition of reduction in impurity. The cons would be that this impurity function prefers leaves of smaller size, potentially causing the tree to overfit or continue splitting without improving the model and produce an overly complex model that may not generalize very well.

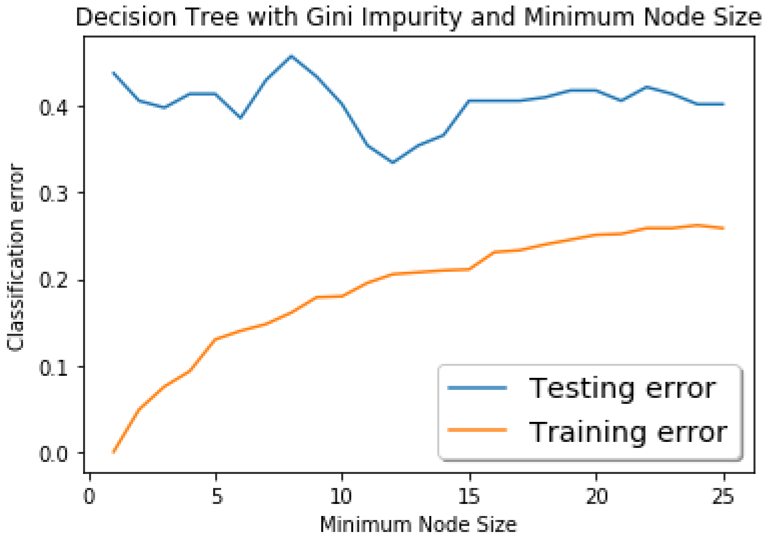
iii. The largest number of unique thresholds needed in order to achieve zero classification training error on a 2-D dataset of 100 data points is 99. Suppose no data point in the dataset shares the same region with another point (i.e. + - + - + …). Then there are 99 boundaries needed to separate all 100 points correctly.

Problem D

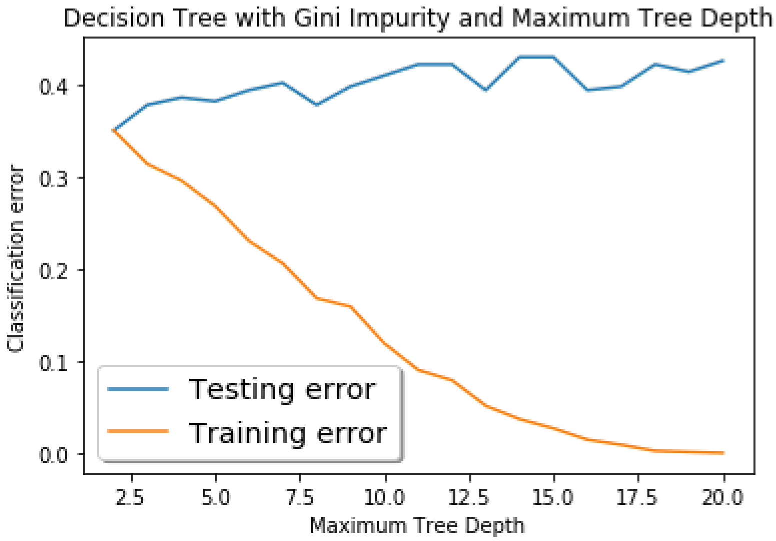
The worst-case complexity of the number of possible splits is O(D * N). We have at most D(N - 1) splits to consider, since there are N - 1 maximum possible splits among N data points for each feature, if all the data points are aligned without overlapping along each feature axis (no two data points share the same value for the same feature).

2 Overfitting Decision Trees

Problem A



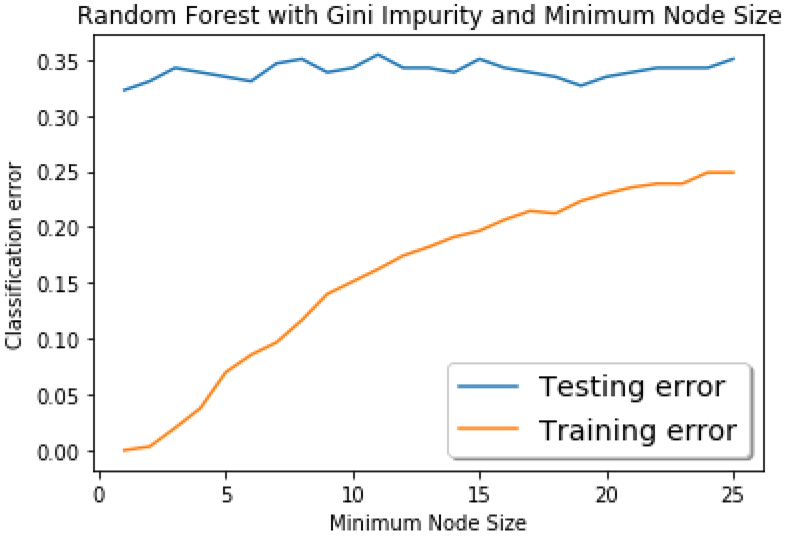
Problem B


Decision Tree with Gini Impurity and Maximum Tree Depth
Classification error
0.0
0.1
0.2
0.3
0.4
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Maximum Tree Depth
Testing error
Training error

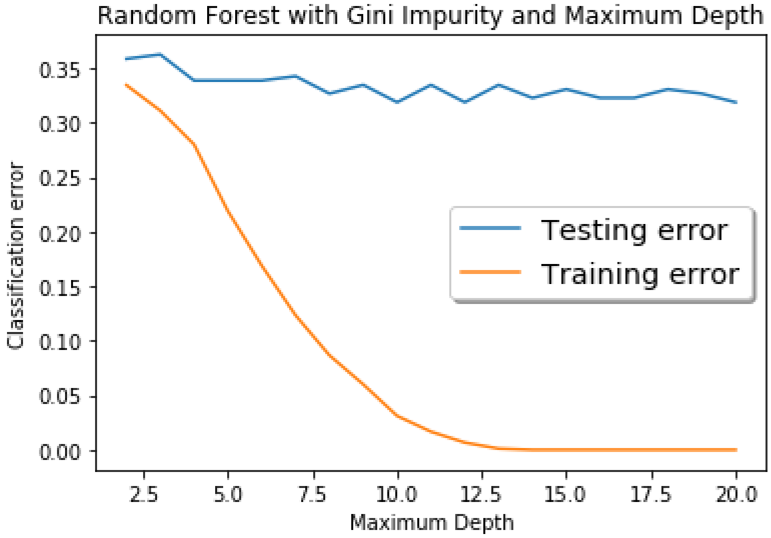
Problem C

The minimum leaf node size of 12 and the maximum depth value of 2 minimize the test error. Choosing the right early stopping condition and parameter improves the performance of a decision tree model by preventing overfitting, which is very common for decision tree models. The minimum node size plot shows that the test error is minimized at the node size of 12, and node sizes smaller than 12 cause overfitting, as test error goes up as node size approaches 1, and node sizes greater than 12 cause underfitting, as test error goes up as node size approaches 25. The maximum tree depth plot shows that test error goes up as the depth goes up, meaning each layer in the tree causes more overfitting. Minimum node size could be a better early stopping method in this case, because it allows some nodes to continue splitting while others remain constant, rather than forcing every child in a layer to split until a certain depth is reached.

Problem D


Random Forest with Gini Impurity and Minimum Node Size
Classification error
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
5
10
15
20
25
Minimum Node Size
Testing error
Training error

Problem E


Random Forest with Gini Impurity and Maximum Depth
Classification error
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Maximum Depth
Testing error
Training error

Problem F

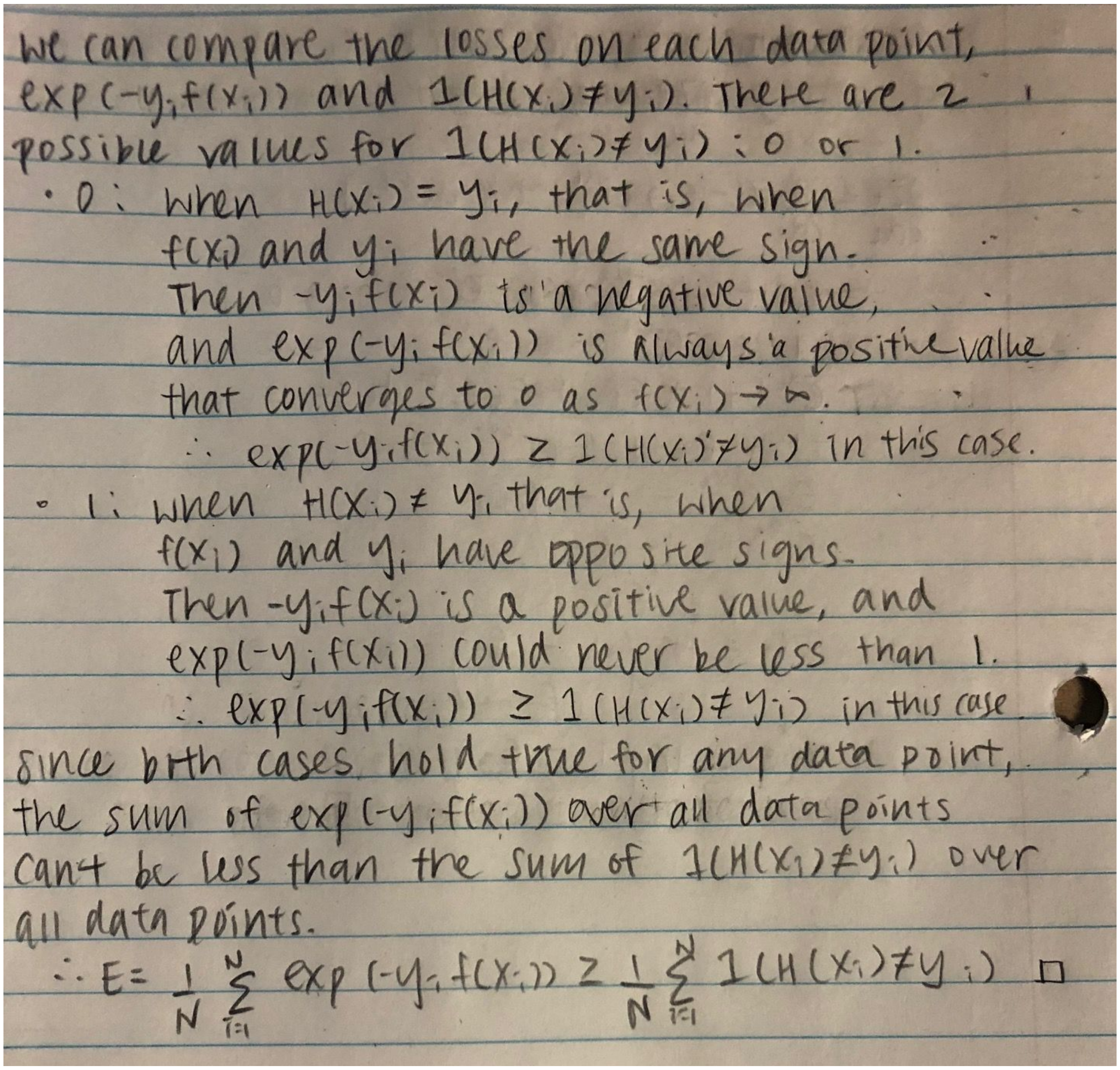
The minimum leaf node size of 1 and the maximum depth value of 10 minimize the test error. Early stopping using minimum node size doesn't help improve the random forest model, as the plot shows that the test error is the smallest when minimum leaf node size is the smallest, and slightly increases as the leaf node size increases. Early stopping using maximum depth also seems to have a very minimal effect on the performance of the model; although the test error is minimized at the depth value of 10, the plot shows that test error rather stays relatively constant as the maximum depth varies.

Problem G

The curves for the random forest show that test error is not very susceptible to or benefits from the early stopping parameter, while the curves for the decision tree show that test error is clearly minimized at certain early stopping parameters. Random forest models are less likely to overfit the training data because of the extra randomness added at each node, and effectively using this randomness may require more nodes to be used in training rather than terminating early. The curves for the random forest also seem to be smoother, as the random forest reduces the variance is more effectively.

3 The AdaBoost Algorithm

Problem A



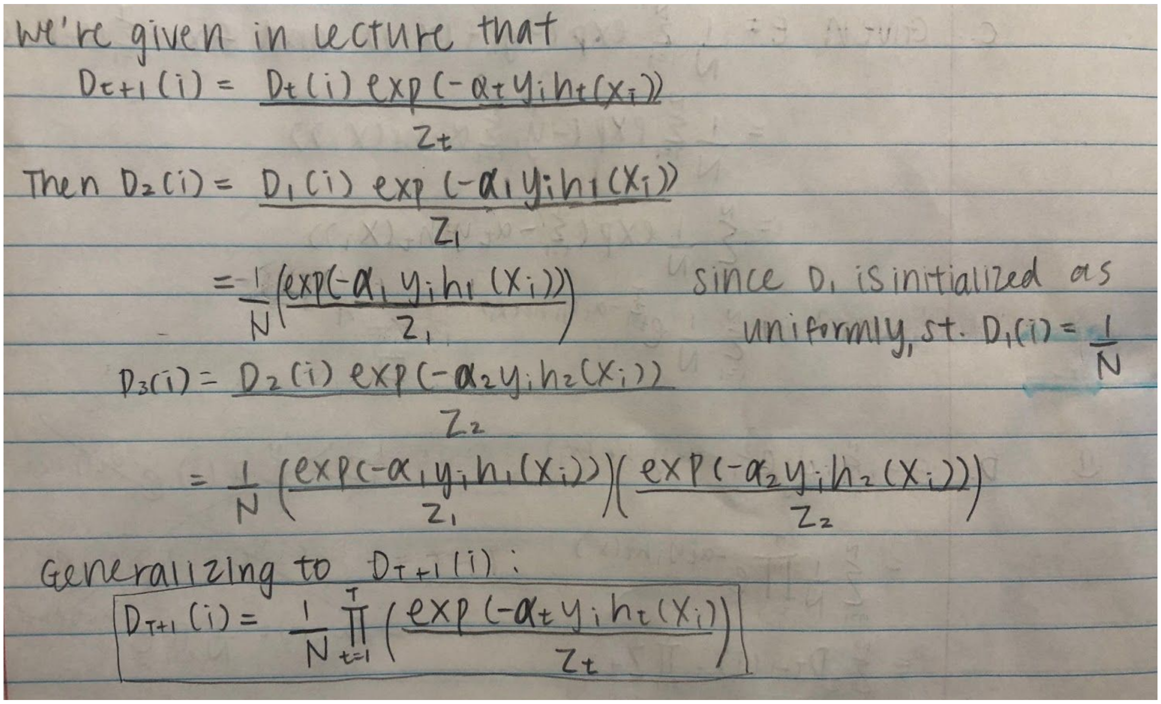
We can compare the losses on each data point, $\exp(-y_i f(x_i))$ and $1(H(x_i) \neq y_i)$. There are 2 possible values for $1(H(x_i) \neq y_i)$: 0 or 1.

- 0: when $H(x_i) = y_i$, that is, when $f(x_i)$ and y_i have the same sign. Then $-y_i f(x_i)$ is a negative value, and $\exp(-y_i f(x_i))$ is always a positive value that converges to 0 as $f(x_i) \to \infty$.
  $\therefore \exp(-y_i f(x_i)) \geq 1(H(x_i) \neq y_i)$ in this case.
- 1: when $H(x_i) \neq y_i$, that is, when $f(x_i)$ and y_i have opposite signs. Then $-y_i f(x_i)$ is a positive value, and $\exp(-y_i f(x_i))$ could never be less than 1.
  $\therefore \exp(-y_i f(x_i)) \geq 1(H(x_i) \neq y_i)$ in this case.

Since both cases hold true for any data point, the sum of $\exp(-y_i f(x_i))$ over all data points can't be less than the sum of $1(H(x_i) \neq y_i)$ over all data points.

$$\therefore E = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(x_i)) \geq \frac{1}{N} \sum_{i=1}^{N} 1(H(x_i) \neq y_i) \quad \square$$

Problem B



we're given in lecture that

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Then $D_2(i) = \dfrac{D_1(i)\exp(-\alpha_1 y_i h_1(X_i))}{Z_1}$

$$= \frac{1}{N}\left(\frac{\exp(-\alpha_1 y_i h_1(X_i))}{Z_1}\right)$$

since D_1 is initialized as uniformly, s.t. $D_1(i) = \frac{1}{N}$

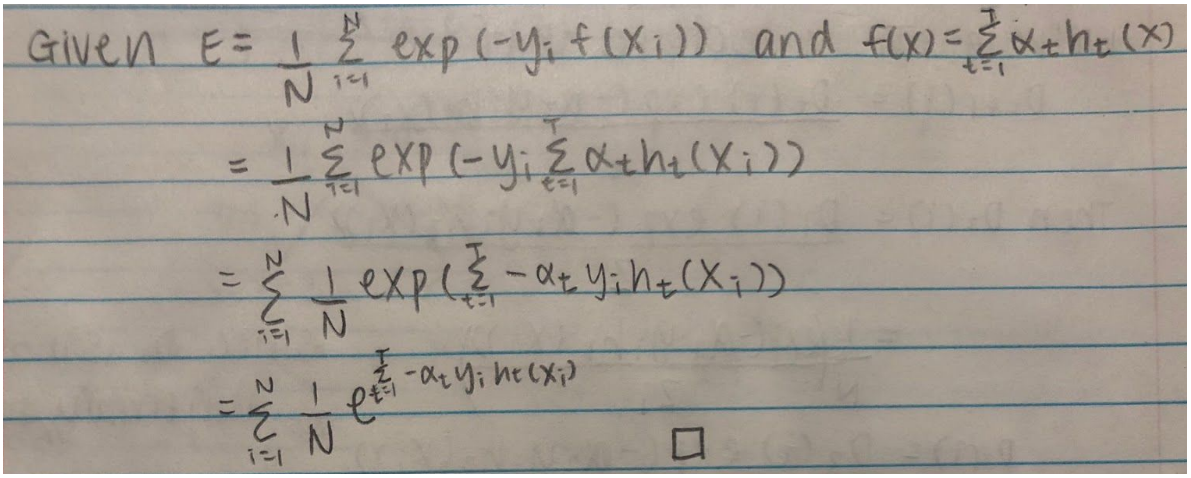
$$D_3(i) = \frac{D_2(i)\exp(-\alpha_2 y_i h_2(X_i))}{Z_2}$$

$$= \frac{1}{N}\left(\frac{\exp(-\alpha_1 y_i h_1(X_i))}{Z_1}\right)\left(\frac{\exp(-\alpha_2 y_i h_2(X_i))}{Z_2}\right)$$

Generalizing to $D_{T+1}(i)$:

$$\boxed{D_{T+1}(i) = \frac{1}{N}\prod_{t=1}^{T}\left(\frac{\exp(-\alpha_t y_i h_t(X_i))}{Z_t}\right)}$$

Problem C



Given $E = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(X_i))$ and $f(X) = \sum_{t=1}^{T} \alpha_t h_t(X)$

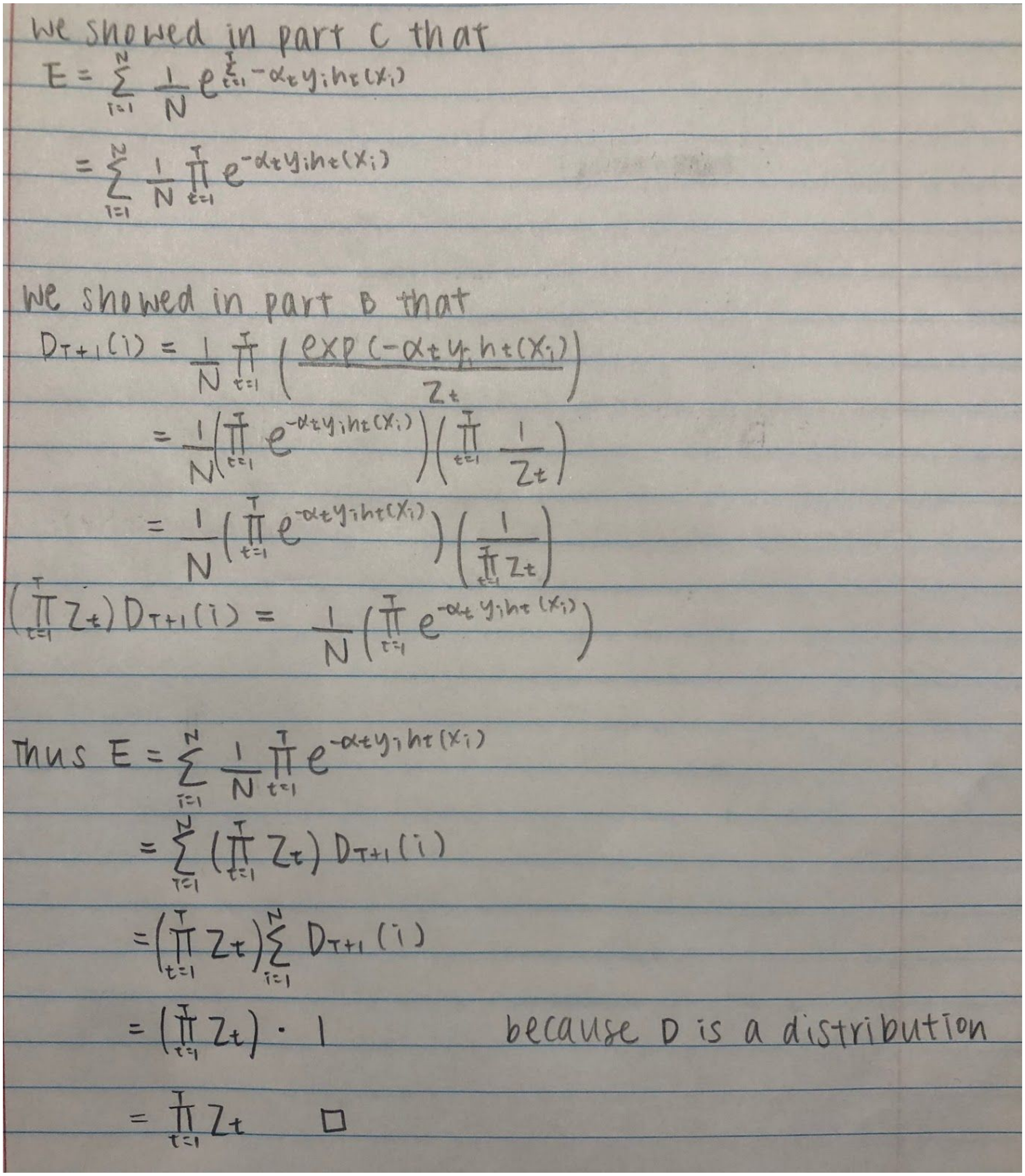
$$= \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(X_i))$$

$$= \sum_{i=1}^{N} \frac{1}{N} \exp(\sum_{t=1}^{T} -\alpha_t y_i h_t(X_i))$$

$$= \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(X_i)}$$

□

Problem D



we showed in part C that

$$E = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)}$$

$$= \sum_{i=1}^{N} \frac{1}{N} \prod_{t=1}^{T} e^{-\alpha_t y_i h_t(x_i)}$$

we showed in part B that

$$D_{T+1}(i) = \frac{1}{N} \prod_{t=1}^{T} \left(\frac{\exp(-\alpha_t y_i h_t(X_i))}{Z_t} \right)$$

$$= \frac{1}{N} \left(\prod_{t=1}^{T} e^{-\alpha_t y_i h_t(X_i)} \right) \left(\prod_{t=1}^{T} \frac{1}{Z_t} \right)$$

$$= \frac{1}{N} \left(\prod_{t=1}^{T} e^{-\alpha_t y_i h_t(X_i)} \right) \left(\frac{1}{\prod_{t=1}^{T} Z_t} \right)$$

$$\left(\prod_{t=1}^{T} Z_t \right) D_{T+1}(i) = \frac{1}{N} \left(\prod_{t=1}^{T} e^{-\alpha_t y_i h_t(X_i)} \right)$$

Thus $E = \sum_{i=1}^{N} \frac{1}{N} \prod_{t=1}^{T} e^{-\alpha_t y_i h_t(X_i)}$

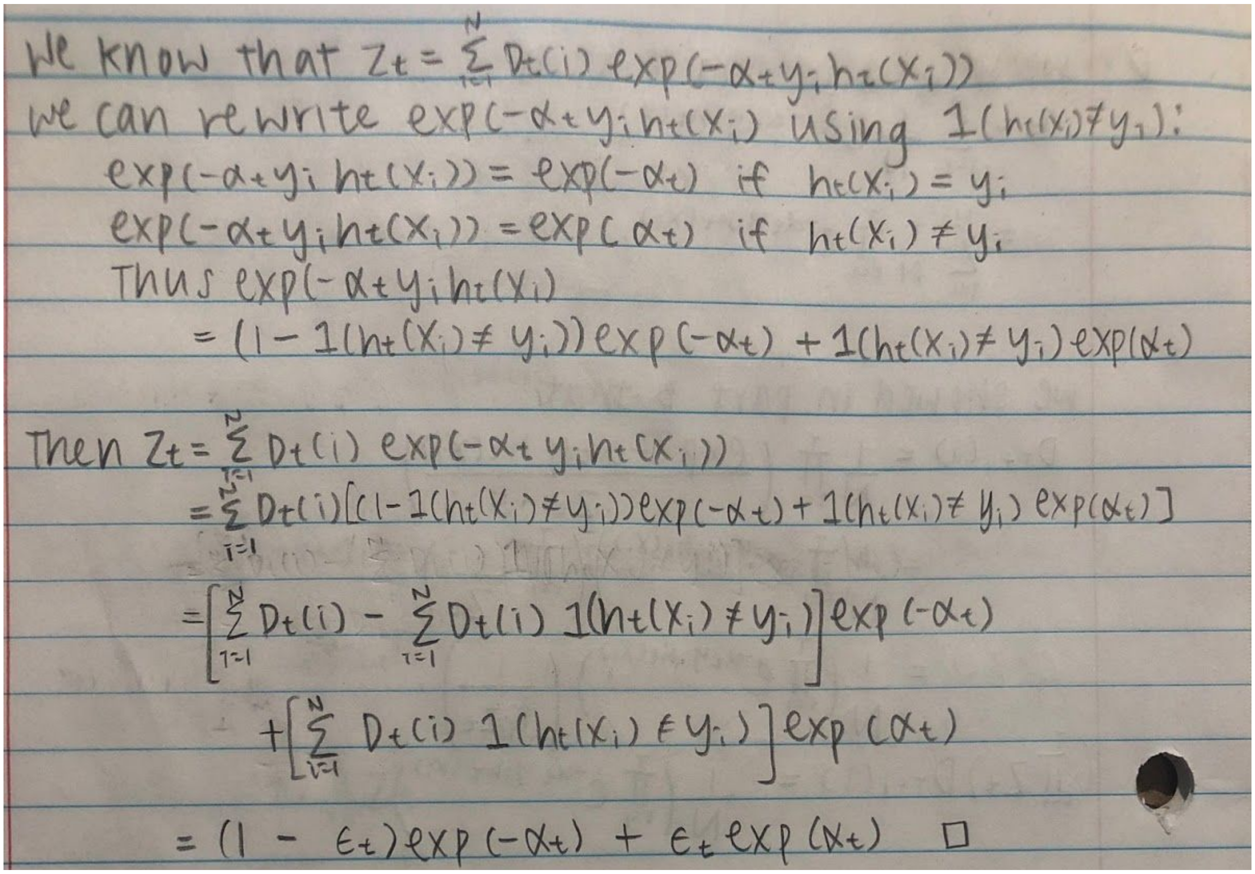
$$= \sum_{i=1}^{N} \left(\prod_{t=1}^{T} Z_t \right) D_{T+1}(i)$$

$$= \left(\prod_{t=1}^{T} Z_t \right) \sum_{i=1}^{N} D_{T+1}(i)$$

$= \left(\prod_{t=1}^{T} Z_t \right) \cdot 1$ because D is a distribution

$= \prod_{t=1}^{T} Z_t$ □

Problem E



We know that $Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$

we can rewrite $\exp(-\alpha_t y_i h_t(x_i)$ using $1(h_t(x_i) \neq y_i)$:

$\exp(-\alpha_t y_i h_t(x_i)) = \exp(-\alpha_t)$ if $h_t(x_i) = y_i$

$\exp(-\alpha_t y_i h_t(x_i)) = \exp(\alpha_t)$ if $h_t(x_i) \neq y_i$

Thus $\exp(-\alpha_t y_i h_t(x_i)$

$= (1 - 1(h_t(x_i) \neq y_i)) \exp(-\alpha_t) + 1(h_t(x_i) \neq y_i) \exp(\alpha_t)$

Then $Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$

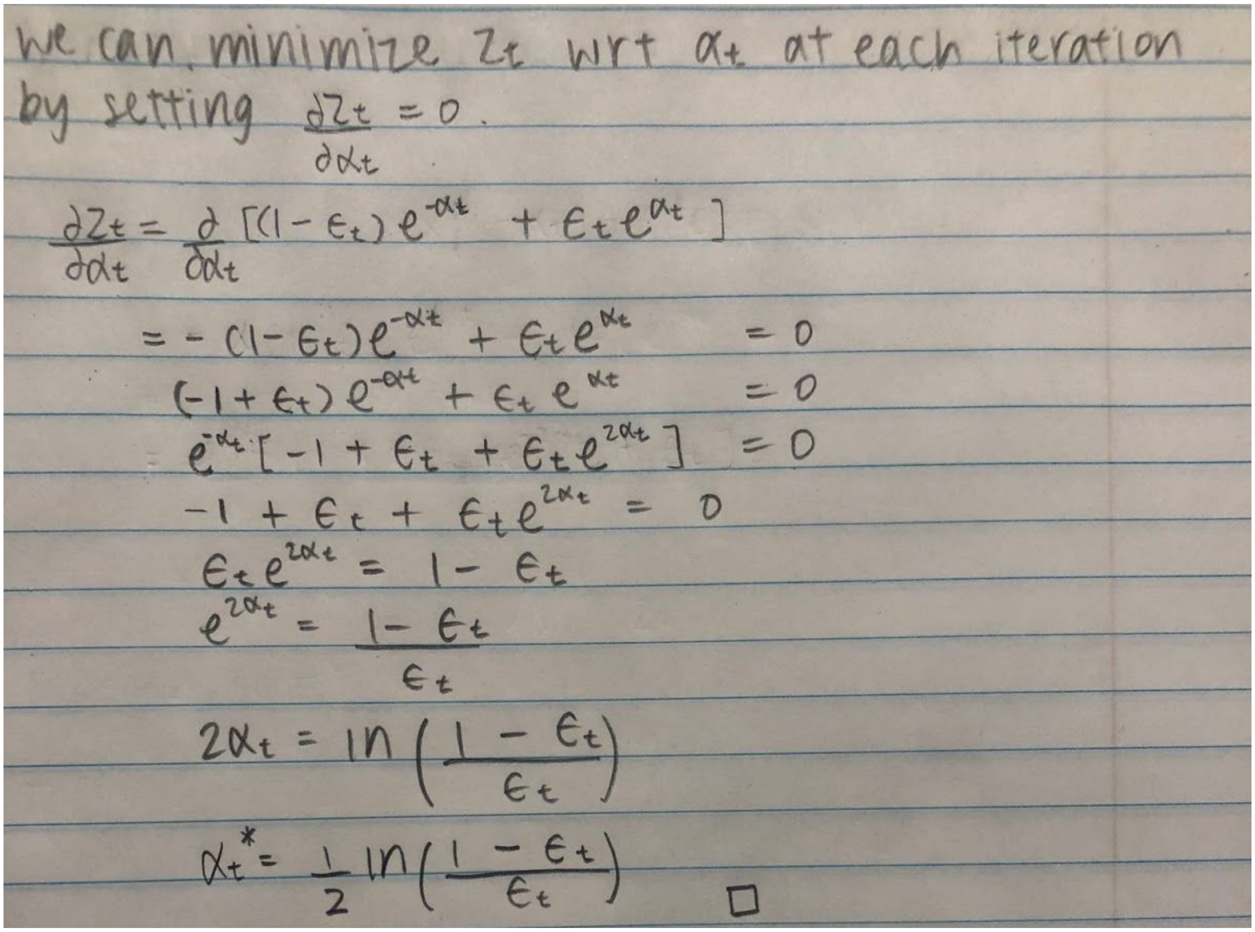
$$= \sum_{i=1}^{N} D_t(i) [(1 - 1(h_t(x_i) \neq y_i)) \exp(-\alpha_t) + 1(h_t(x_i) \neq y_i) \exp(\alpha_t)]$$

$$= \left[\sum_{i=1}^{N} D_t(i) - \sum_{i=1}^{N} D_t(i)\, 1(h_t(x_i) \neq y_i)\right] \exp(-\alpha_t)$$

$$+ \left[\sum_{i=1}^{N} D_t(i)\, 1(h_t(x_i) \neq y_i)\right] \exp(\alpha_t)$$

$= (1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t)$ □

Problem F



we can minimize Z_t wrt α_t at each iteration by setting $\frac{\partial Z_t}{\partial \alpha_t} = 0$.

$$\frac{\partial Z_t}{\partial \alpha_t} = \frac{\partial}{\partial \alpha_t} [(1-\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t}]$$

$$= -(1-\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 0$$

$$(-1+\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 0$$

$$e^{-\alpha_t} [-1 + \epsilon_t + \epsilon_t e^{2\alpha_t}] = 0$$

$$-1 + \epsilon_t + \epsilon_t e^{2\alpha_t} = 0$$

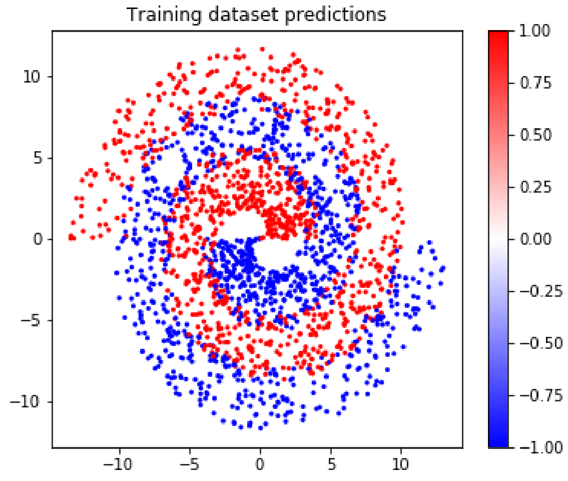
$$\epsilon_t e^{2\alpha_t} = 1 - \epsilon_t$$

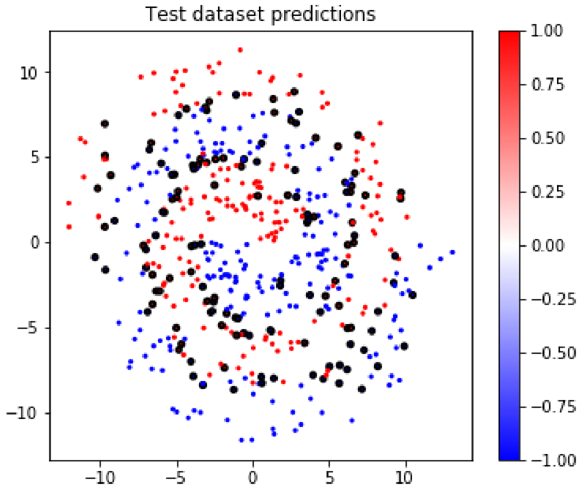
$$e^{2\alpha_t} = \frac{1-\epsilon_t}{\epsilon_t}$$

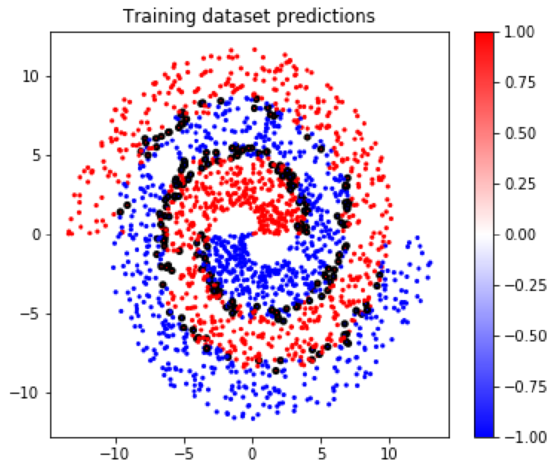
$$2\alpha_t = \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

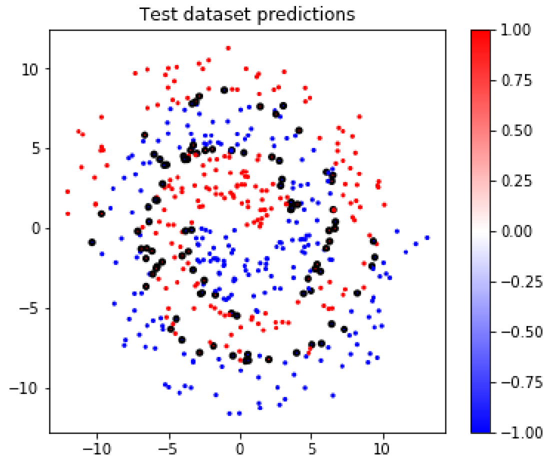
$$\alpha_t^* = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \quad \square$$

Problem G


Training dataset predictions


Test dataset predictions


Training dataset predictions


Test dataset predictions

Problem H

The loss curves for AdaBoost are smoother (less fluctuations) than the loss curves for Gradient Boosting. The loss curves for both models get smoother with each new weak regressor / classifier, but the curves for Adaboost become more significantly smoother. This is because using a regressor rather than classifier results in some weak regressors to have a larger impact on the result than others and thus introducing more variance. For Gradient Boosting, the loss decreases with each new weak regressor until a certain point, at which it starts to go back up. For AdaBoost, the loss continues to decrease with each new weak classifier. This shows that for Gradient Boosting, after a certain number of weak regressors, introducing more doesn't help improve the model and only causes the different regressors to compete with each other. On the other hand, AdaBoost allows each new classifier to focus on what needs to be improved to classify the points better, resulting in each new classifier decreasing test loss.

Problem I

The Gradient Boosting has a lower training loss but a higher test loss than the AdaBoost model. The AdaBoost performed better on the classification dataset; the test loss for Gradient Boosting was .258, while the test loss for AdaBoost was .186.

Problem J

The dataset weights are the largest for misclassified points and smallest for correctly classified points.