1 Decision Trees [30 Points]

Relevant materials: Lecture 5

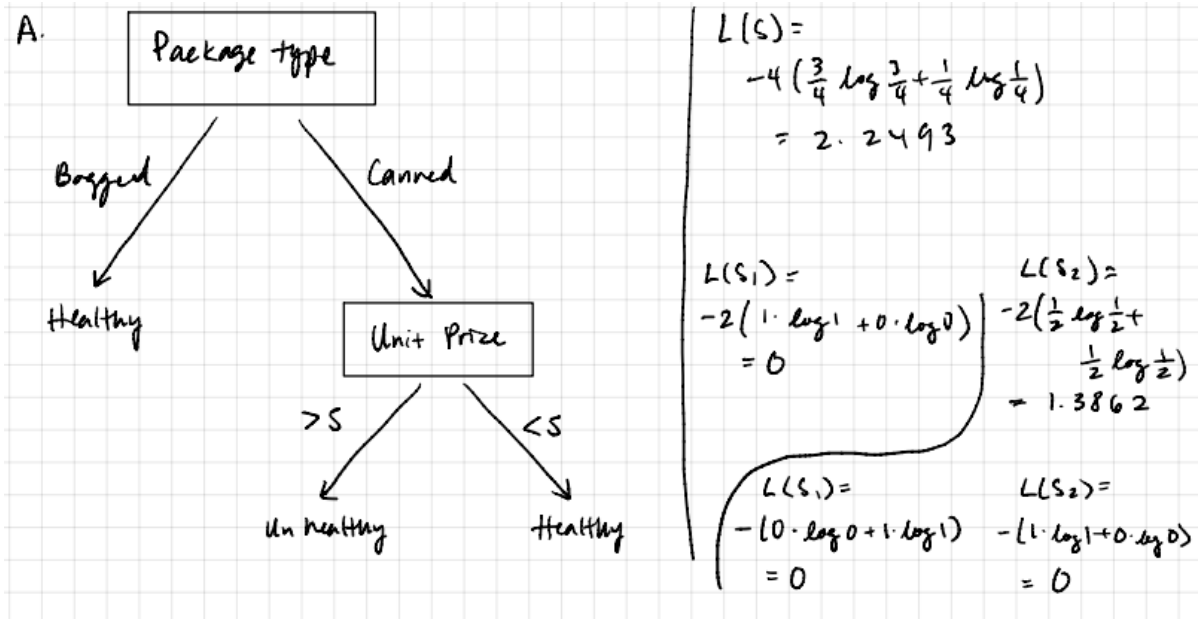
Problem A [7 points]: Consider the following data, where given information about some food you must predict whether it is healthy:

No.	Package Type	Unit Price $>$ \$5	Contains $>$ 5 grams of fat	Healthy?
1	Canned	Yes	Yes	No
2	Bagged	Yes	No	Yes
3	Bagged	No	Yes	Yes
4	Canned	No	No	Yes

Train a decision tree by hand using top-down greedy induction. Use *entropy* (with natural log) as the impurity measure. Since the data can be classified without error, the stopping criterion will be no impurity in the leaves.

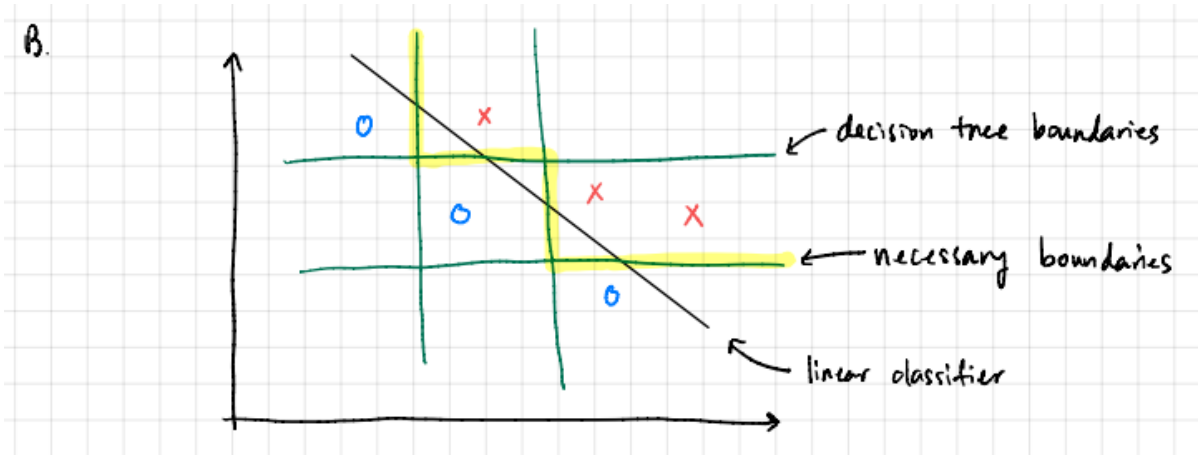
Submit a drawing of your tree showing the impurity reduction yielded by each split (including root) in your decision tree.

Solution A:

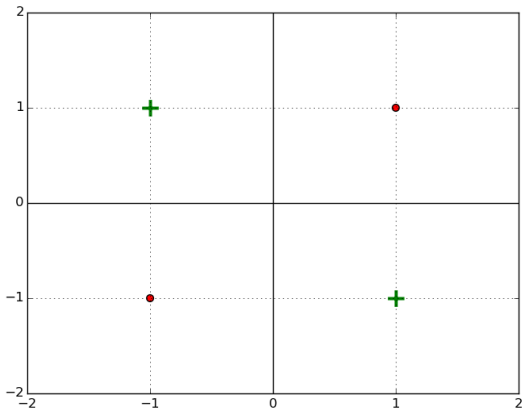


Problem B [4 points]: Compared to a linear classifier, is a decision tree always preferred for classification problems? If not, draw a simple 2-D dataset that can be perfectly classified by a simple linear classifier but which requires an overly complex decision tree to perfectly classify.

Solution B: *No. As demonstrated in lecture, 2D datasets with a diagonal separation can be easily classified by a linear model, but since decision trees can only split parallel to axis, it requires a more complex model to classify the same dataset.*



Problem C [15 points]: Consider the following 2D data set:



i. [5 points]: Suppose we train a decision tree on this dataset using top-down greedy induction, with the Gini index as the impurity measure. We define our stopping condition to be if no split of a node results in any reduction in impurity. Submit a drawing of the resulting tree. What is its classification error ((number of misclassified points) / (number of total points))?

ii. [5 points]: Submit a drawing of a two-level decision tree that classifies the above dataset with zero classification error. (You don't need to use any particular training algorithm to produce the tree.)

Is there any impurity measure (i.e. any function that maps the data points under a particular node in a tree to a real number) that would have led top-down greedy induction with the same stopping condition to produce the tree you drew? If so, give an example of one, and briefly describe its pros and cons as an impurity measure for training decision trees in general (on arbitrary datasets).

iii. [5 points]: Suppose there are 100 data points in some 2-D dataset. What is the largest number of unique thresholds (i.e., internal nodes) you might need in order to achieve zero classification training error (on the training set)? Please justify your answer.

Solution C: *Part i.*

C. i. $L(S) = 4\left(1 - \frac{1}{4} - \frac{1}{4}\right) = 2$

$L(S_1) = 2\left(1 - \frac{1}{4} - \frac{1}{4}\right) = 1$

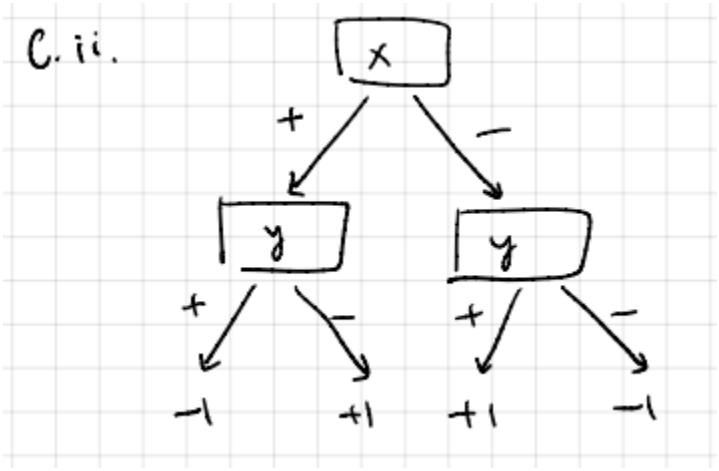
$L(S_2) = 2\left(1 - \frac{1}{4} - \frac{1}{4}\right) = 1$

} any split

1

The classification error is 0.5, since half to the points are misclassified.

Part ii.



One alternative impurity measure would be to simply take the Gini index and multiply it again by the size of each split, $|S|$. The Gini index then becomes $|S|^2(1 - p_s^2 - (1 - p_s)^2)$. This impurity measure is better when splitting may note decrease the impurity directly, but may have later benefits. However, one downside is that it encourages splitting, which could cause more overfitting.

Part iii.

The number of internal nodes necessary is maximized when each individual data point is classified in its own leaf node. If we imagine a colinear set of datapoints with alternating classification, we would need $100 - 1 = 99$ different splits to achieve zero classification training error on the training set. Binary methods of splitting the dataset are more efficient at separating data, since they are logarithmic in scale. Thus, 99 is the maximjm possible number of unique thresholds.

Problem D [4 points]: Suppose in top-down greedy induction we want to split a leaf node that contains N data points composed of D continuous features. What is the worst-case complexity (big-O in terms of N and D) of the number of possible splits we must consider in order to find the one that most reduces impurity? Please justify your answer.

Note: Recall that at each node-splitting step in training a DT, you must consider all possible splits that you can make. While there are an infinite number of possible decision boundaries since we are using continuous features, there are not an infinite number of boundaries that result in unique child sets (which is what we mean by "split").

Solution D: *From the above problem, we know that the number of splits for each feature is maximized with a split for each dataset in the collection, which is $O(N)$. Since this can be done among D different features, the total number of possible splits is $O(D \cdot N)$.*

2 Overfitting Decision Trees [30 Points, EC 7 Points]

Relevant materials: Lecture 5

In this problem, you will use the Diabetic Retinopathy Debrecen Data Set, which contains features extracted from images to determine whether or not the images contain signs of diabetic retinopathy. Additional information about this dataset can be found at the link below:

https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set

In the following question, your goal is to predict the diagnosis of diabetic retinopathy, which is the final column in the data matrix. Use the first 900 rows as training data, and the last 251 rows as validation data. Please feel free to use additional packages such as Scikit-Learn. Include your code in your submission.

Problem A [10 points]: Choose one of the following from i or ii:

i. Train a decision tree classifier using Gini as the impurity measure and minimal leaf node size as early stopping criterion. Try different minimal leaf node sizes from 1 to 25 in increments of 1. Then, on a single plot, plot both training and test classification error versus leaf node size. To do this, fill in the `classification_err` and `eval_tree_based_model_min_samples` functions in the code template for this problem.

ii. Train a decision tree classifier using Gini as the impurity measure and maximal tree depth as early stopping criterion. Try different tree depths from 2 to 20 in increments of 1. Then, on a single plot, plot both training and test classification error versus tree depth. To do this, fill in the `eval_tree_based_model_max_depth` function in the code template for this problem.

Solution A: *Code link*

https://colab.research.google.com/drive/13og2U2eVRLJxlJ7sp6yC1IQ_QUfd4Ktb?usp=sharing

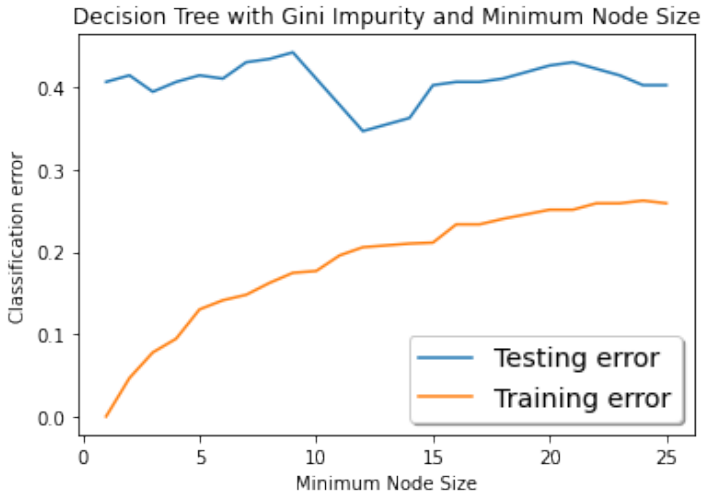


Figure 1: Decision tree classification error with Gini impurity and minimum leaf node size early stopping

Problem B [6 points]: For either the minimal leaf node size or maximum depth parameters in the previous problem, which parameter value minimizes the test error? What effects does early stopping have on the performance of a decision tree model? Please justify your answer based on the plot you derived.

Solution B: *The value of minimum leaf node size that minimized the test error was 12. Early stopping with large minimum node size underfit the data, since the training error was very high. One the opposite end, small minimum leaf size overfit the data, since test error was much higher than the training error. In between these two regimes is where the testing error was minimized.*

Problem C [4 points]: Choose one of the following from i or ii:

i. Train a random forest classifier using Gini as the impurity measure, minimal leaf node size as early stopping criterion, and 1,000 trees in the forest. Try different node sizes from 1 to 25 in increments of 1. Then, on a single plot, plot both training and test classification error versus leaf node size.

ii. Train a random forest classifier using Gini as the impurity measure, maximal tree depth as early stopping criterion, and 1,000 trees in the forest. Try different tree depths from 2 to 20 in increments of 1. Then, on a single plot, plot both training and test classification error versus tree depth.

Solution C:

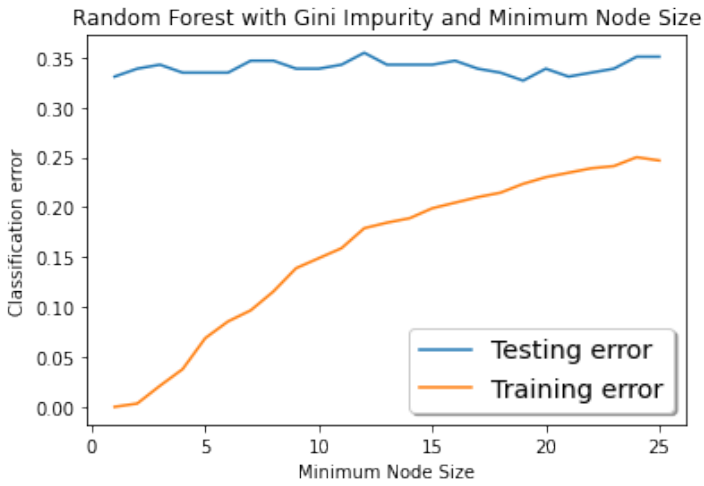


Figure 2: Random forest classification error with Gini impurity and minimum leaf node size early stopping

Problem D [6 points]: For either the minimal leaf node size or maximum depth parameters tested, which parameter value minimizes the random forest test error? What effects does early stopping have on the performance of a random forest model? Please justify your answer based on the plot you derived.

Solution D: *For minimal leaf node size, the test error was minimized at 19 minimum leaf node size. The test error did not noticeably change when the minimum leaf size was changed. However, the training error indicated that a smaller minimum leaf size results in overfitting, while a larger minimum leaf size underfit the data.*

Problem E [4 points]: Do you observe any differences between the curves for the random forest and decision tree plots? If so, explain what could account for these differences.

Solution E: *The optimal value for the parameter was more noticeable for the decision tree compared to the random forest. This may be because the inherent randomness from the sampling of random forest, it is more smooth and there is less specific difference between instances of the model.*

Extra Credit [7 points total] :

Problem F: [5 points, Extra Credit] Complete the other option for **Problem A** and **Problem C**.

Solution F:

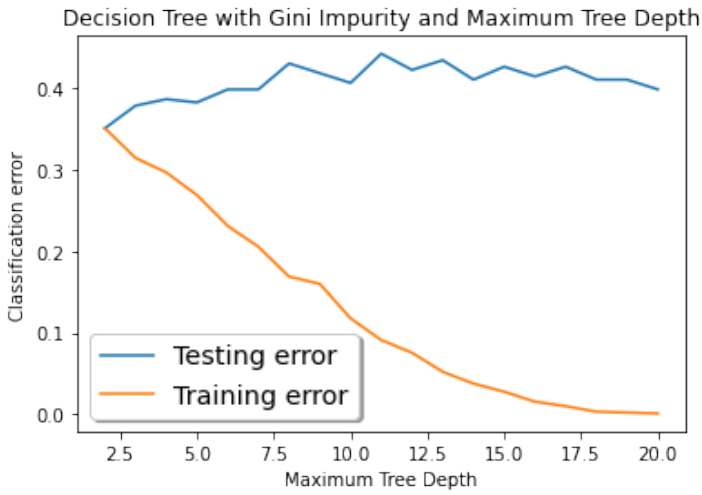


Figure 3: Decision tree classification error with Gini impurity and maximal tree depth early stopping

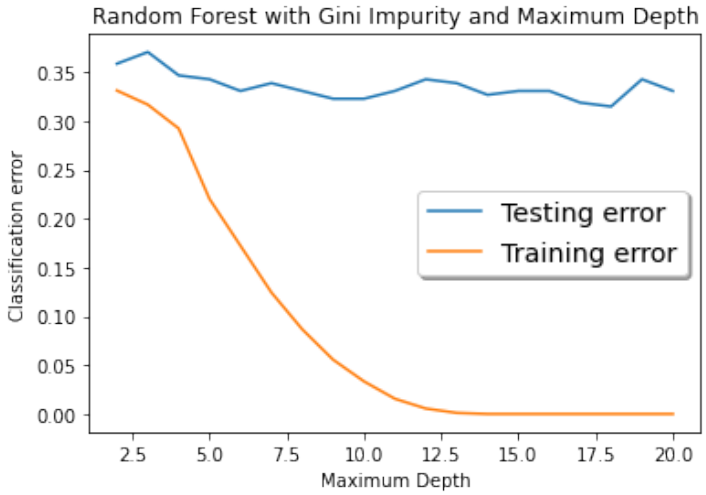


Figure 4: Random forest classification error with Gini impurity and maximal tree depth early stopping

Problem G: [2 points, Extra Credit] For the stopping criterion tested in **Problem F**, which parameter value minimizes the decision tree and random forest test error respectively?

Solution G: *The maximum depth values that minimized the test error were 2 for the decision tree and 18 for the random forest.*

3 The AdaBoost Algorithm [40 points]

Relevant materials: Lecture 6

In this problem, you will show that the choice of the α_t parameter in the AdaBoost algorithm corresponds to greedily minimizing an exponential upper bound on the loss term at each iteration.

Problem A [3 points]: Let $h_t : \mathbb{R}^m \to \{-1, 1\}$ be the weak classifier obtained at step t, and let α_t be its weight. Recall that the final classifier is

$$H(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{i=1}^{T} \alpha_t h_t(x)\right).$$

Suppose $\{(x_1, y_1), ..., (x_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ is our training dataset. Show that the training set error of the final classifier can be bounded from above if an an exponential loss function is used:

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

where $\mathbb{1}$ is the indicator function.

Solution A: *We can show that the sum is bounded by comparing each term individually. In other words, we can show that*

$$\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i) \neq y_i)$$

In the case that $H(x_i) = y_i$, $\exp(n)$ for any real n is greater than 0, so $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i) \neq y_i) = 0$.

In the case that $H(x_i) \neq y_i$, we can prove this in 2 separate cases. In the case that $H(x_i) = 1, y_i = -1$, then $f(x_i) > 0$. Then, $-y_i f(x_i) > 0$. We know that $\exp(n) > 1$ for $n > 0$, so $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i)$. In the case that $H(x_i) = -1, y_i 1$, then $f(x_i) < 0$. Then, $-y_i f(x_i) > 0$. We know that $\exp(n) > 1$ for $n > 0$, so $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i)$.

Problem B [3 points]: Find $D_{T+1}(i)$ in terms of Z_t, α_t, x_i, y_i, and the classifier h_t, where T is the last timestep and $t \in \{1, \ldots, T\}$. Recall that Z_t is the normalization factor for distribution D_{t+1}:

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

Solution B:

$$\begin{aligned}
D_{T+1}(i) &= \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T} D_T \\
D_T &= \frac{\exp(-\alpha_{T-1} y_i h_{T-1}(x_i))}{Z_{T-1}} D_{T-1} \\
D_{T+1}(i) &= \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T} \cdot \frac{\exp(-\alpha_{T-1} y_i h_{T-1}(x_i))}{Z_{T-1}} \ldots \cdot D_1 \\
&= \frac{1}{N} \cdot \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T} \cdot \frac{\exp(-\alpha_{T-1} y_i h_{T-1}(x_i))}{Z_{T-1}} \cdot \ldots \cdot \frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1} \\
&= \frac{1}{N} \prod_{t=1}^{T} \frac{\exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\end{aligned}$$

Problem C [2 points]: Show that $E = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)}$.

Solution C: *We know that*

$$E = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(x_i))$$

and

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Plugging these values in, we get

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

Problem D [5 points]: Show that

$$E = \prod_{t=1}^{T} Z_t.$$

Hint: *Recall that* $\sum_{i=1}^{N} D_t(i) = 1$ *because* D *is a distribution.*

Solution D: *We know that*

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

so

$$\begin{aligned} E &= \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)} \\ &= \sum_{i+1}^{N} \frac{1}{N} \left(\prod_{t=1}^{T} \exp(-\alpha_t y_i h_t(x_i)) \right) \\ &= \sum_{i=1}^{N} D_{T+1}(i) \prod_{t=1}^{T} Z_t \\ &= \prod_{t=1}^{T} Z_t \end{aligned}$$

Problem E [5 points]: Show that the normalizer Z_t can be written as

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

where ϵ_t is the training set error of weak classifier h_t for the weighted dataset:

$$\epsilon_t = \sum_{i=1}^{N} D_t(i)\mathbb{1}(h_t(x_i) \neq y_i).$$

Solution E: *We know that if* $y_i = h_t(x_i)$, $\exp(-\alpha_t y_i h_t(x_i)) = \exp(-\alpha)$, *since* $y_i, h_t(x_i)$ *are either* $(1, 1)$ *or* $(-1, -1)$. *Otherwise,* $\exp(-\alpha_t y_i h_t(x_i)) = \exp(\alpha)$. *Additionally,*

$$\epsilon_t = \sum_{i=1}^{N} D_t(i)\mathbb{1}(h_t(x_i) \neq y_i)$$
$$1 - \epsilon_t = \sum_{i=1}^{N} D_t(i)\mathbb{1}(h_t(x_i) = y_i)$$

since the total sum of weights is 1. Then,

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} D_t(i)\exp(-\alpha_t y_i h_t(x_i)) \\ &= \sum_{i=1}^{N} D_t(i)\exp(-\alpha_t y_i h_t(x_i))\mathbb{1}(h_t(x_i) \neq y) + \sum_{i=1}^{N} D_t(i)\exp(-\alpha_t y_i h_t(x_i))\mathbb{1}(h_t(x_i) = y) \\ &= \epsilon_t \exp(\alpha_t) + (1 - \epsilon_t)\exp(-\alpha_t) \end{aligned}$$

Problem F [2 points]: We derived all of this because it is hard to directly minimize the training set error, but we can greedily minimize the upper bound E on this error. Show that choosing α_t greedily to minimize Z_t at each iteration leads to the choices in AdaBoost:

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

Solution F: *To find the minimum, we take the derivative of Z_t with respect to α_t and set to 0.*

$$\begin{aligned}
\frac{d}{d\alpha_t} = 0 &= \epsilon_t \exp(\alpha_t) - (1 - \epsilon_t) \exp(-\alpha_t) \\
\frac{1 - \epsilon_t}{\exp(\alpha_t)} &= \epsilon_t \exp(\alpha_t) \\
\frac{1 - \epsilon_t}{\epsilon_t} &= \exp(2\alpha_t) \\
2\alpha_t &= \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \\
\alpha_t &= \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)
\end{aligned}$$

Problem G [14 points]: Implement the `GradientBoosting.fit()` and `AdaBoost.fit()` methods in the notebook provided for you. Some important notes and guidelines follow:

- For both methods, make sure to work with the class attributes provided to you. Namely, after `GradientBoosting.fit()` is called, `self.clfs` should be appropriately filled with the `self.n_clfs` trained weak hypotheses. Similarly, after `AdaBoost.fit()` is called, `self.clfs` and `self.coefs` should be appropriately filled with the `self.n_clfs` trained weak hypotheses and their coefficients, respectively.
- `AdaBoost.fit()` should additionally return an (N, T) shaped numpy array `D` such that `D[:, t]` contains D_{t+1} for each $t \in \{0, \ldots,$ `self.n_clfs`$\}$.
- For the `AdaBoost.fit()` method, **use the 0/1 loss** instead of the exponential loss.
- The only Sklearn classes that you may use in implementing your boosting fit functions are the DecisionTreeRegressor and DecisionTreeClassifier, not GradientBoostingRegressor.

Problem H [2 points]: Describe and explain the behaviour of the loss curves for gradient boosting and for AdaBoost. You should consider two kinds of behaviours: the smoothness of the curves and the final values that the curves approach.

Solution H: *Code link*

https://colab.research.google.com/drive/1muS8OdfVyDek-gQTPZgbkWI1pzXMtHs4?usp=sharing

Gradient boosting produced much smoother loss curves than AdaBoost. AdaBoost outperforms gradient boosting in test error, ending at a slightly lower final value. However, gradient boosting achieves a lower error on the training set compared to AdaBoost. Gradient boosting still had a large difference between training and test error, while for AdaBoost, the difference was much smaller.

Problem I [2 points]: Compare the final loss values of the two models. Which performed better on the classification dataset?

Solution I: *The AdaBoost model performed better on the classification dataset.*

Problem J [2 points]: For AdaBoost, where are the dataset weights the largest, and where are they the smallest?

Hint: *Watch how the dataset weights change across time in the animation with a lower final classification error on the test dataset.*

Solution J: *The points with the largest weight are near the borders between the two types of points, which are the misclassified points.*