CS/CNS/EE 155 Philip Carr
Set 3
10 late hours were used for this set. January 30th, 2019

---

1 Decision Trees [30 Points]

Relevant materials: Lecture 5

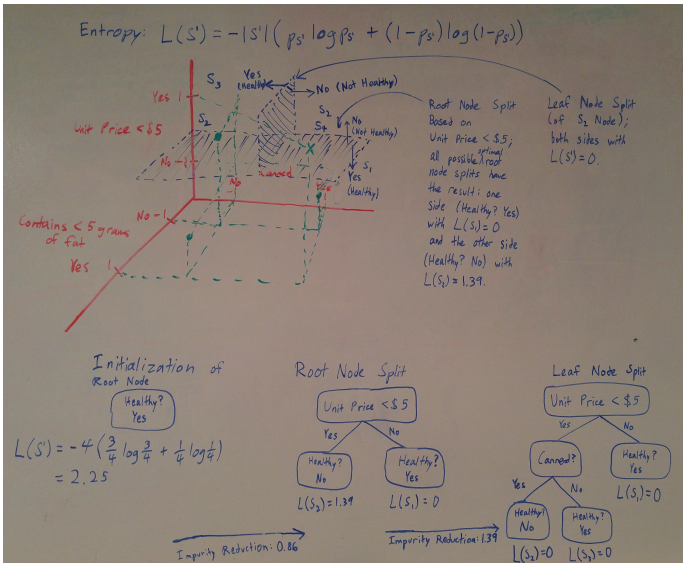
Problem A [7 points]: Consider the following data, where given information about some food you must predict whether it is healthy:

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:



Along the bottom of the image above is the decision tree as would make successive splits in the root and leaf nodes using top-down greedy induction, along with the impurity reduction yielded by each

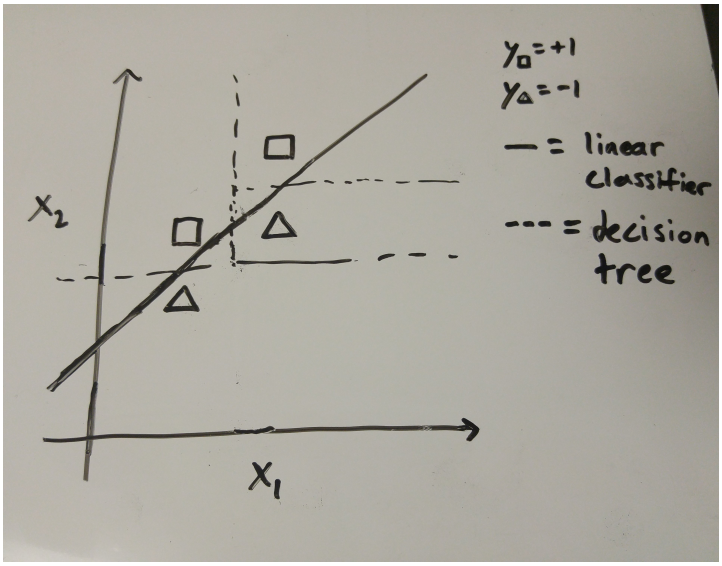
split (including the root) of the decision tree. The decision tree at the bottom right corner of the image is the completed decision tree with 0 impurity (all data classified without error). The question "Canned?" refers to the feature Package Type, with yes and no responses to "Canned" refering to Canned and Bagged respectively. All impurity measures shown in the image were calculated using entropy (with natural log).

In the top left side of the image is a plot of the data points in the 3D feature space, where the dot points represent healthy food items, and where the x point represents the unhealthy food item. The decision tree in this plot is represented by its separating planes indicating the node boundaries of the decision tree.

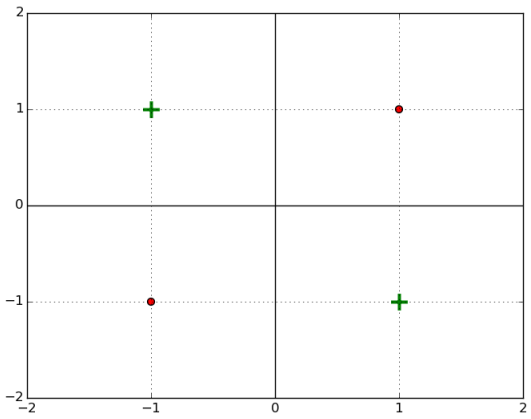
Problem B [4 points]: Compared to a linear classifier, is a decision tree always preferred for classification problems? If not, draw a simple 2-D dataset that can be perfectly classified by a simple linear classifier but which requires an overly complex decision tree to perfectly classify.

Solution B: Compared to a linear classifier, a decision tree is not always preferred for classification problems. One reason why this results is because decision trees only separate data points by boundaries parallel to some axis, but data points can end up being most easily separable along off-axis boundaries.

Below is a drawing of 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 (both the region with triangle points and the region with square points are split into two separate regions by the decision tree, while the linear classifier simply divides the points into their correct regions without an overly complicated model of the separated regions):

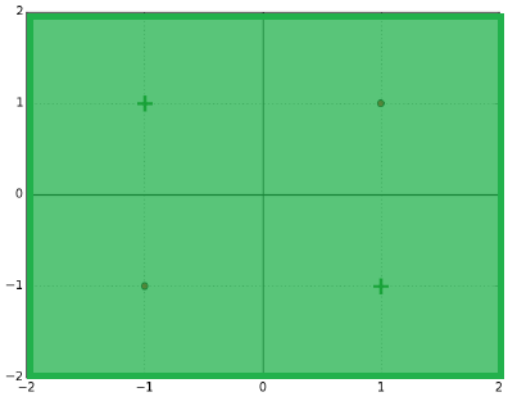


Problem C [15 points]: Consider the following 2D data set:



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))?

Solution C.i:



At the initialization of the root node, all points in the dataset are classified as the majority class. In this dataset, both the green and red points are evenly represented, so without loss of generality, the root

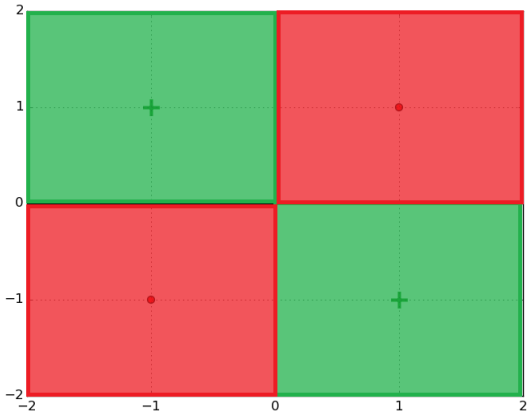
node was set to classify all points as green points (as indicated by the green shading over the entire space). This decision tree has a Gini index of $L(S') = |S'|(1 - p_{S'}^2 - (1 - p_{S'})^2) = (4)(1 - (\frac{1}{2})^2 - (1 - (\frac{1}{2})^2)) = 2$. Since every possible split of the root node results in the same Gini index of $L(S') = 2$, the root node is not split and the decision tree creation is stopped a the state shown above in the drawing.

---

ii. [5 points]: Submit a drawing of a two-level decision tree that classifies the above dataset with zero classification error. (You dont 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).

Solution C.ii:

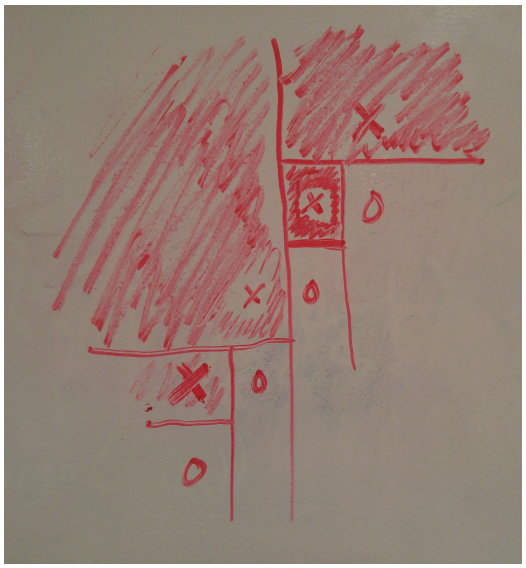


Bernoulli measure $L(S') = |S'| - 1$

---

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.iii:



As shown in the above example, data points arranged in this diagonal pattern will result in the decision tree having to make (a maximum of) as many unique thresholds as there are data points for the decision tree to achieve perfect classification training error (on the training set). Therefore, for 100 data points in some 2-D dataset, the largest number of unique thresholds needed to achieve zero classification training error (on the training set).

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: At each node-splitting step in training a DT, all possible splits that can be made all must be considered. Because of this, the number of possible unique splits (where a unique split is distinct from other splits in terms of which data points are located in one side of the decision tree split or the other) is linearly proportional to the number of data points N, so this operation's complexity is $O(N)$. Since there are D dimesions of the dataset, all possible splits in all possible dimensions must also be made. Thus, this operation's complexity is linearly proportional to D, so this operation's complexity is $O(D)$. Combining the two operations, since the decision tree model iterates through all possible splits of the N data points for each dimension D, the worst-case complexity of the number of possible splits that must be considered in order to find the one that most reduces impurity is $O(ND)$.

2 Overfitting Decision Trees [30 Points]

Relevant materials: Lecture 5

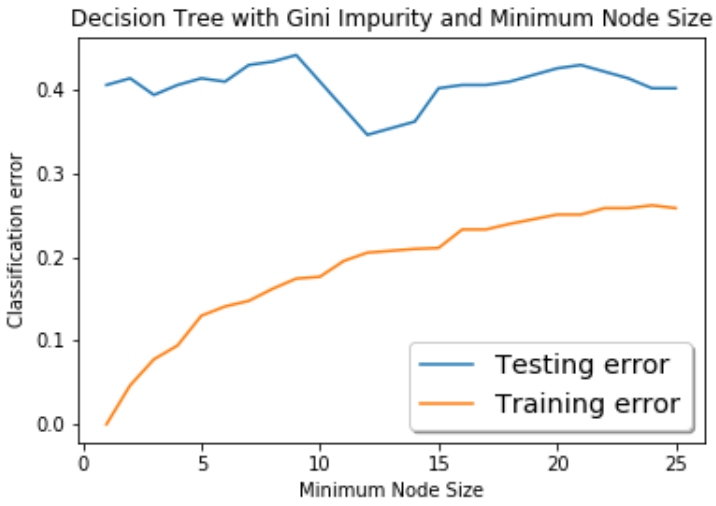
In this problem, you will use the Diabetic Retinopathy Debrecen Data Set, which contains features extracted from images to determine whether or not the images contain signs of diabetic retinopathy. Additional information about this dataset can be found at the link below:

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.

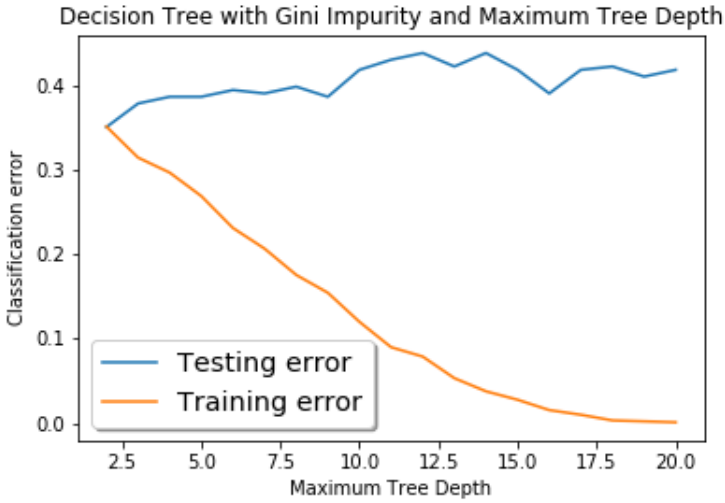
Problem A [7 points]: Train a decision tree classifier using Gini as the impurity measure and minimal leaf node size as early stopping criterion. Try different minimal leaf node sizes from 1 to 25 in increments of 1. Then, on a single plot, plot both training and test classification error versus leaf 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.

Solution A:



Problem B [7 points]: 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 B:

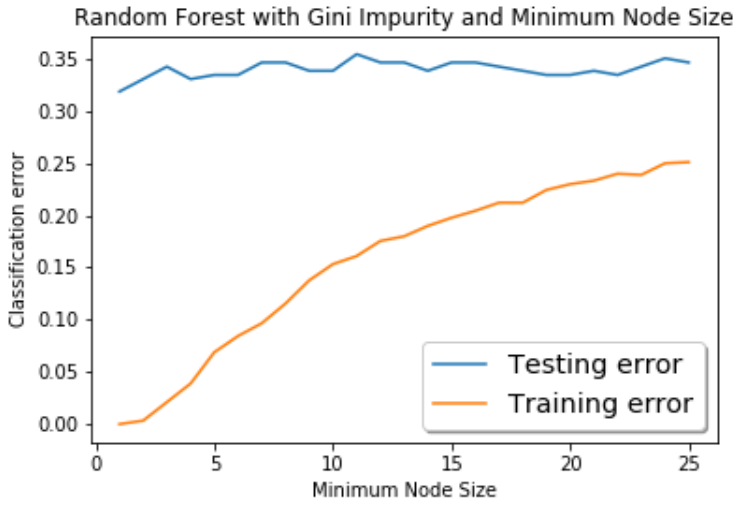


Problem C [4 points]: For both the minimal leaf node size and maximum depth parameters tested in the last two questions, which parameter value minimizes the test error? What effects does early stopping have on the performance of a decision tree model? Please justify your answer based on the two plots you derived.

Solution C: For the minimal leaf node size, the test error is minimized at min_samples_leaf = 12. For maximum depth, the test error is minimized at max_depth = 2. The effect that early stopping has on the performance of a decision tree model is preventing overfitting, as when the minimal leaf node size becomes too large or the maximum depth becomes too large, the decision tree has a worse test error than with respective smaller value parameters. This effect is seen in both plots in Problems 2A and 2B, as increasing the magnitude of either the minimal leaf node size or the maximum depth causes test error to increase.

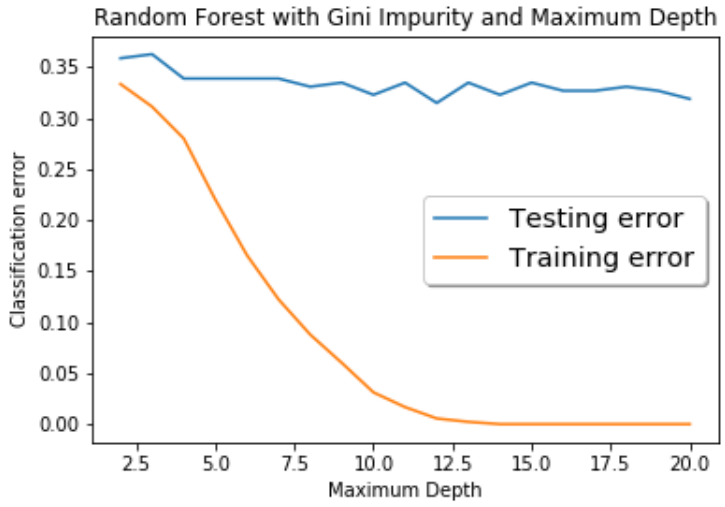
Problem D [2 points]: Train a random forest classifier using Gini as the impurity measure, minimal leaf node size as early stopping criterion, and 1,000 trees in the forest. Try different node sizes from 1 to 25 in increments of 1. Then, on a single plot, plot both training and test classification error versus leaf node size.

Solution D:



Problem E [2 points]: 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 E:



Problem F [4 points]: For both the minimal leaf node size and 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 two plots you derived.

Solution F: For the minimal leaf node size, the test error is minimized at min_samples_leaf = 1. For maximum depth, the test error is minimized at max_depth = 12. The effect that early stopping has on the performance of a random forest model is preventing slight overfitting, as the plot in Problem 2D shows that the minimum test error is for min_samples_leaf = 1, and the plot in Problem 2E shows that overfitting occurs past max_depth = 12. This effect is seen in both plots in Problems 2D and 2E, as increasing the magnitude of either the minimal leaf node size or the maximum depth causes test error to increase.

Problem G [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 G: The plots of varying max_depth for the decision tree and random forest differ, as the training error of random forest converges more quickly than the training error of decision tree does. Additionally, the test error for the max_depth plot of decision tree is minimized at max_depth = 1, while for random forest, max_depth = 12 is where the minimum test error is in the plot. The differences in these plots originate from the different natures of the models used, as random forest is an ensemble model, while decision tree is only one structure.

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

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

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: There are two cases to consider for this system for any given (x_i, y_i).

Case 1: $H(x_i) \neq y_i$.

Since $H(x_i) \neq y_i$, by definition, $\mathbb{1}(H(x_i) \neq y_i) = 1$. Since $H(x_i) \neq y_i$, $\text{sign}(-y_i f(x_i)) = 1$, so $\exp(-y_i f(x_i)) \geq 1$. Therefore, $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i) \neq y_i)$
$\implies \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i f(x_i)) \geq \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(H(x_i) \neq y_i)$ for this case.

Case 2: $H(x_i) = y_i$.

Since $H(x_i) \neq y_i$, by definition, $\mathbb{1}(H(x_i) \neq y_i) = 0$. For any possible $-y_i f(x_i)$, $\exp(-y_i f(x_i)) \geq 0$. Therefore, $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i) \neq y_i) \implies \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i f(x_i)) \geq \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(H(x_i) \neq y_i)$ for this case.

Since both possible cases show that $\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(H(x_i) \neq y_i)$, 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. ■

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}:

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

Solution B: Using the AdaBoost Weighting Update recursive forumula (lecture 6, slide 58),

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

where $D_1 = \frac{1}{N}$. Thus,

$$D_2(i) = \frac{D_1(i) \exp\left(-\alpha_1 y_i h_1(x_i)\right)}{Z_1},$$

and,

$$D_3(i) = D_2(i) \frac{\exp(-\alpha_2 y_i h_2(x_i))}{Z_2} = \frac{D_1(i) \exp(-\alpha_1 y_i h_1(x_i))}{Z_1} \frac{\exp(-\alpha_2 y_i h_2(x_i))}{Z_2}$$
$$= \frac{\frac{1}{N} \exp(-\alpha_1 y_i h_1(x_i) - \alpha_2 y_i h_2(x_i))}{Z_1 Z_2}.$$

Therefore, using the pattern that originates from the recursive formula, for $T+1$,

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

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: From Problem 3A, $E = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(x_i))$, where $f(x_i) = \sum_{t=1}^{T} \alpha_t h_t(x_i)$, so

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

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

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: From Problem 3B, $D_{T+1}(i) = \frac{1}{N} \exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)) \prod_{t=1}^{T} Z^{-1}$ Taking the sum from $\sum_{i=1}^{N}$ of both sides gives

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

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

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

$$= \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(x_i)) \implies E = \prod_{t=1}^{T} Z_t. \blacksquare$$

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: From Problem 3B, $Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y_t h_t(x_i))$. If $\text{sign}(y_t h_t(x_i)) = -1$, then $-\alpha_t y_t h_t(x_i) = \alpha_t$. If $\text{sign}(y_t h_t(x_i)) = 1$, then $-\alpha_t y_t h_t(x_i) = -\alpha_t$. Therefore,

$$Z_t = \sum_{i, h_t(x_i) = y_i} D_t(i) \exp(-\alpha_t) + \sum_{i, h_t(x_i) \neq y_i} D_t(i) \exp(\alpha_t)$$

$$Z_t = \exp(-\alpha_t) \sum_{i, h_t(x_i) = y_i} D_t(i) \mathbb{1}(h_t(x_i) \neq y_i) + \exp(\alpha_t) \sum_{i, h_t(x_i) \neq y_i} D_t(i) \mathbb{1}(h_t(x_i) \neq y_i).$$

Since $\mathbb{1}(h_t(x_i) \neq y_i) = 1 - \mathbb{1}(h_t(x_i) = y_i)$,

$$Z_t = \exp(-\alpha_t) \sum_{i, h_t(x_i) = y_i} D_t(i)(1 - \mathbb{1}(h_t(x_i) = y_i)) + \exp(\alpha_t) \sum_{i, h_t(x_i) \neq y_i} D_t(i) \mathbb{1}(h_t(x_i) \neq y_i)$$

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

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:

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

Solution F: Minimizing Z_t results in $\frac{\partial}{\partial\alpha}Z_t = \frac{\partial}{\partial\alpha}(\exp(-\alpha_t)(1-\epsilon_t) + \exp(\alpha_t)\epsilon_t) = 0$

$$\implies -\exp(-\alpha_t)(1-\epsilon_t) + \exp(\alpha_t)\epsilon_t = 0 \implies \exp(\alpha_t)\epsilon_t = \exp(-\alpha_t)(1-\epsilon_t)$$

$$\implies \ln(\exp(\alpha_t)\epsilon_t) = \ln(\exp(-\alpha_t)(1-\epsilon_t)) \implies \ln(\exp(\alpha_t)) + \ln(\epsilon_t) = \ln(\exp(-\alpha_t)) + \ln(1-\epsilon_t)$$

$$\alpha_t + \ln(\epsilon_t) = -\alpha_t + \ln(1-\epsilon_t) \implies 2\alpha_t = \ln(1-\epsilon_t) - \ln(\epsilon) \implies \alpha_t^* = \frac{1}{2}\ln\Big(\frac{1-\epsilon_t}{\epsilon_t}\Big).$$

Therefore, choosing α_t greedily to minimize Z_t at each iteration leads to the choices in AdaBoost:

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

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.

Solution G:

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: Gradient boost had a smoother loss curve, while AdaBoost had a lower final test loss.

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

Solution I: AdaBoost performed better than Gradient boost, with final test losses of about 0.20 and 0.25 respectively.

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.*

Solution J: