Machine Learning & Data Mining
Set 3

Caltech CS/CNS/EE 155
January 24th, 2024

Policies

- Due 11:59 PM, January 31th, via Gradescope.
- You are free to collaborate on all of the problems, subject to the collaboration policy stated in the syllabus.
- In this course, we will be using Google Colab for code submissions. You will need a Google account.
- You are allowed to use up to 48 late hours across the entire term. Late hours must be used in units of whole hours. Specify the total number of hours you have used when submitting the assignment.
- Students are expected to complete homework assignments based on their understanding of the course material. Student can use LLMs as a resource (e.g., helping with debugging, or grammar checking), but the assignments (including code) should be principally authored by the student.

Submission Instructions

- Submit your report as a single .pdf file to Gradescope (entry code DKB4KW), under "Problem Set 3".
- In the report, **include any images generated by your code** along with your answers to the questions.
- Submit your code by **sharing a link in your report** to your Google Colab notebook for each problem (see naming instructions below). Make sure to set sharing permissions to at least "Anyone with the link can view". **Links that can not be run by TAs will not be counted as turned in.** Check your links in an incognito window before submitting to be sure.
- For instructions specifically pertaining to the Gradescope submission process, see https://www.gradescope.com/get_started#student-submission.

Google Colab Instructions

For each notebook, you need to save a copy to your drive.

1. Upload all the files found in the 'code' folder from the set3.zip to your Google drive (put them all in one folder).
2. Edit the .ipynb file names to "lastname_firstname_originaltitle", e.g."yue_yisong_3_notebook_part1.ipynb"

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:

$$H(S) = -\frac{3}{4}\ln\frac{3}{4} - \frac{1}{4}\ln\frac{1}{4} \tag{1}$$

$$H(S) = -(0.75 \times (-0.2877) + 0.25 \times (-1.386)) = 0.5623 \tag{2}$$

*Splitting on **Package Type** gives:*

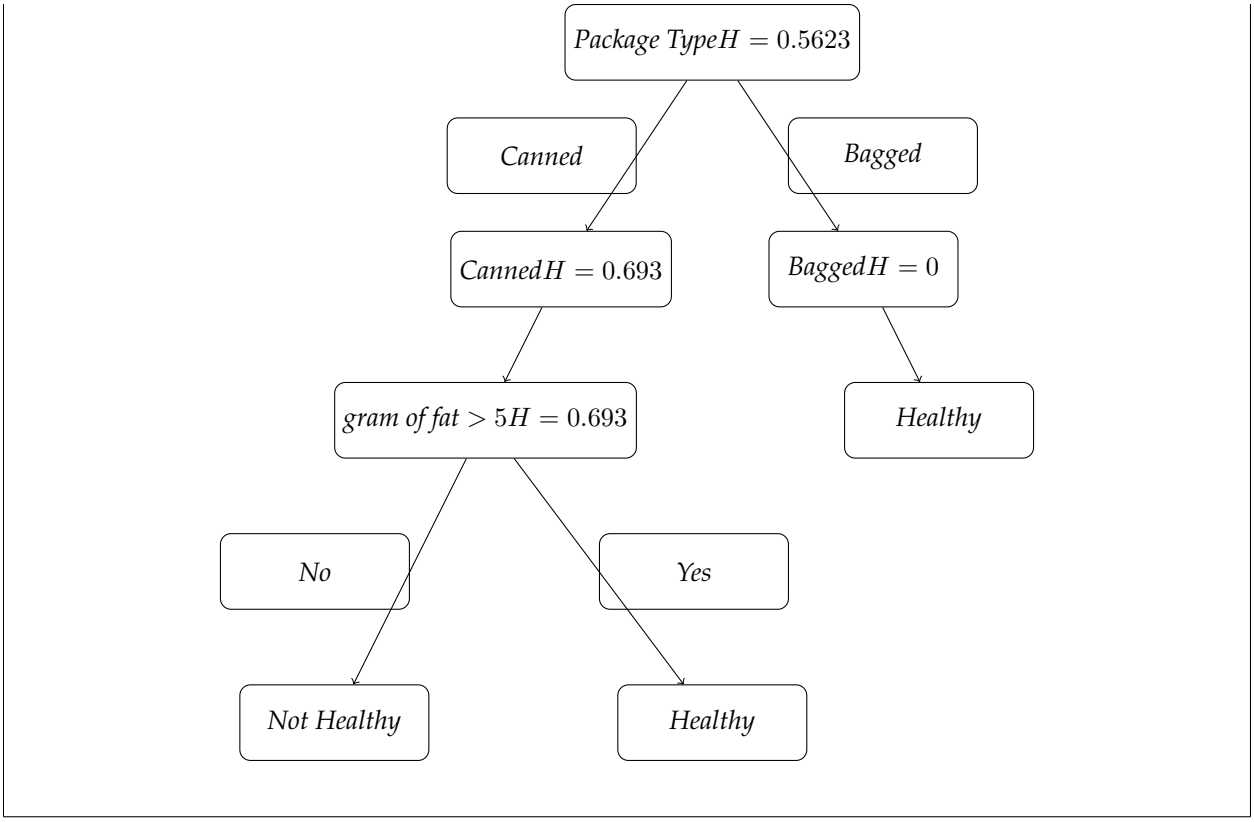
- *Canned (Entropy:* 0.693*)*
- *Bagged (Entropy:* 0*)*

Weighted entropy:

$$H_{split} = \frac{2}{4}(0.693) + \frac{2}{4}(0) = 0.3465 \tag{3}$$

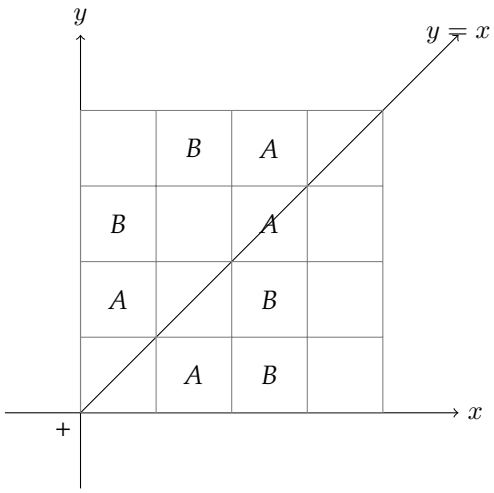
Entropy reduction:

$$0.5623 - 0.3465 = 0.2158 \tag{4}$$



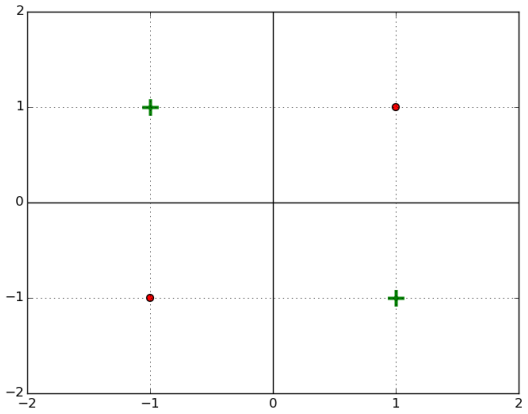
Problem B [4 points]: Compared to a linear classifier, is a decision tree always preferred for classification problems? If not, draw a simple 2-D dataset that can be perfectly classified by a simple linear classifier but which requires an overly complex decision tree to perfectly classify.

Solution B: *No, a decision tree is not always preferred over a linear classifier for classification problems. While decision trees excel at capturing non-linear relationships, they can struggle with simple, non-axis-aligned linear separations.*



The diagonal line $y = x$ cleanly separates classes A and B, but a decision tree would require multiple splits to mimic this boundary.

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

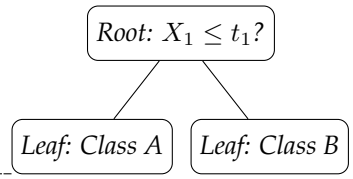


i. [5 points]: Suppose we train a decision tree on this dataset using top-down greedy induction, with the Gini index as the impurity measure. We define our stopping condition to be if no split of a node results in any reduction in impurity. Submit a drawing of the resulting tree. What is its classification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

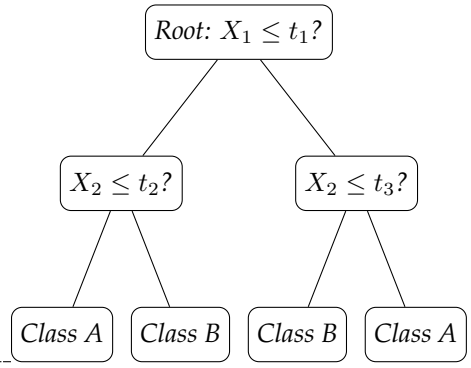
ii. [5 points]: Submit a drawing of a two-level decision tree that classifies the above dataset with zero classification error. (You don't need to use any particular training algorithm to produce the tree.)

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: *i)* 

If it gets 2 wrong out of 10, then it's $2/10$

ii) 

Pros: Captures fine-grained class distributions. Cons: Computationally intensive and prone to overfitting.

iii)

- *For* $n = 100$ *points in 2D, the worst-case scenario requires separating each point individually. Since each internal node splits the space into two regions, the number of unique thresholds equals the number of splits. A binary tree with* n *leaves requires* $n - 1$ *splits. If all splits are on distinct thresholds (axis-aligned), the maximum number is:*

$$\boxed{99}$$

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: *In the worst-case scenario, we must assess a split for every feature, where each feature can have up to* $N+1$ *distinct split points. As a result, the total number of possible splits to consider is:* $(N+1) \cdot D = O(ND)$

2 Overfitting Decision Trees [30 Points]

Relevant materials: Lecture 5

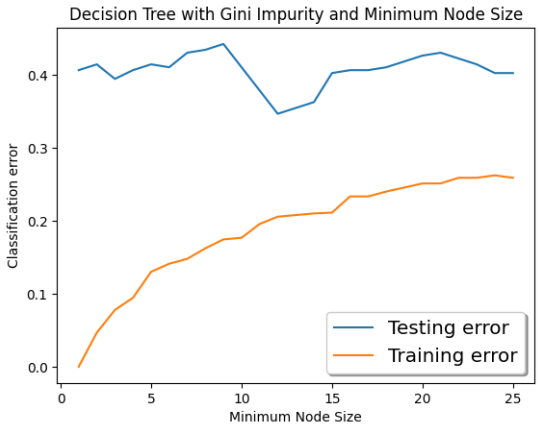
In this problem, you will use the Diabetic Retinopathy Debrecen Data Set, which contains features extracted from images to determine whether or not the images contain signs of diabetic retinopathy. Additional information about this dataset can be found at the link below:

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.

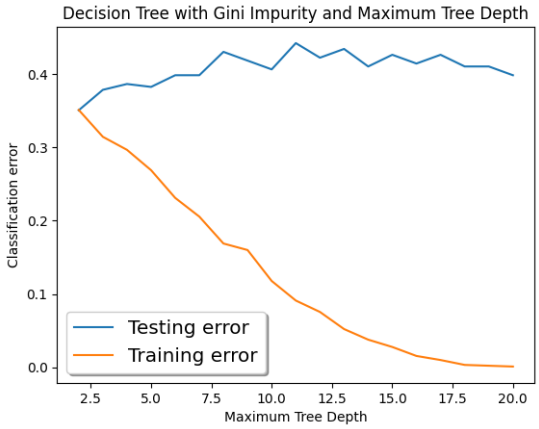
Solution A:



Code

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:

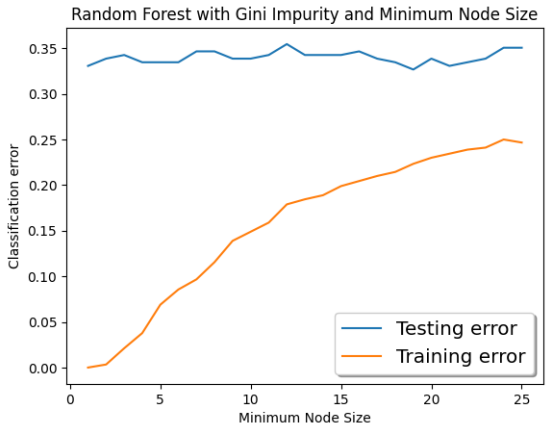


Problem C [4 points]: For both the minimal leaf node size and maximum depth parameters tested in the last two questions, which parameter value minimizes the test error? What effects does early stopping have on the performance of a decision tree model? Please justify your answer based on the two plots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

Solution C: *Using the Minimum Node Size stopping criterion, we see that the minimum testing error is reached at a minimum node size of 12. For the Maximum Depth stopping criterion, the minimum test error occurs at a depth of around 6-8. Early stopping prevents overfitting by limiting tree complexity*

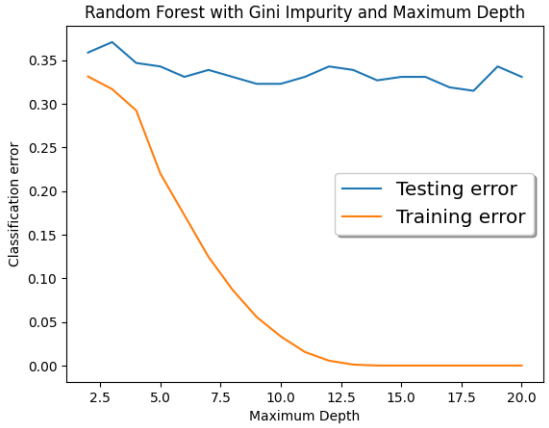
Problem D [2 points]: Train a random forest classifier using Gini as the impurity measure, minimal leaf node size as early stopping criterion, and 1,000 trees in the forest. Try different node sizes from 1 to 25 in increments of 1. Then, on a single plot, plot both training and test classification error 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: *Early stopping can sometimes worsen the test error. In the case of the random forest, the lowest test error occurs at a minimum node size of 19 and a maximum depth of 18. This suggests that deeper trees with larger leaf nodes are beneficial, as the averaging effect of the random forest helps control overfitting while maintaining model flexibility.*

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: *Random forest curves are smoother than decision trees because they aggregate multiple trees, reducing variance. Furthermore, random forests tend to achieve lower test errors, as they mitigate overfitting by averaging multiple models. The optimal depth for random forests is deeper than for individual decision trees, as decision trees overfit quickly while random forests maintain stability through ensemble learning.*

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: *The final classifier $H(x_i) \neq y_i$ if and only if $y_i f(x_i) \leq 0$, where*

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

The training error is:

$$\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(H(x_i) \neq y_i).$$

For each data point (x_i, y_i):

If $H(x_i) \neq y_i$ (misclassified):

$$y_i f(x_i) \leq 0 \implies -y_i f(x_i) \geq 0 \implies \exp(-y_i f(x_i)) \geq \exp(0) = 1.$$

Here, $\mathbf{1}(H(x_i) \neq y_i) = 1$, so

$$\exp(-y_i f(x_i)) \geq 1 = \mathbf{1}(H(x_i) \neq y_i).$$

If $H(x_i) = y_i$ (correctly classified):

$$y_i f(x_i) > 0 \implies -y_i f(x_i) < 0 \implies \exp(-y_i f(x_i)) < \exp(0) = 1.$$

Here, $\mathbf{1}(H(x_i) \neq y_i) = 0$, so

$$\exp(-y_i f(x_i)) \geq 0 = \mathbf{1}(H(x_i) \neq y_i).$$

Since $\exp(-y_i f(x_i)) \geq \mathbf{1}(H(x_i) \neq y_i)$ for all i, summing over all N points gives:

$$\sum_{i=1}^{N} \exp(-y_i f(x_i)) \geq \sum_{i=1}^{N} \mathbf{1}(H(x_i) \neq y_i).$$

Dividing both sides by N:

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

The exponential loss

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

upper bounds the training error

$$\frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(H(x_i) \neq y_i),$$

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:

$$D_1(i) = \frac{1}{N} \tag{5}$$

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}, \tag{6}$$

$$Z_t = \sum_{i=1}^{N} D_t(i)\exp(-\alpha_t y_i h_t(x_i)). \tag{7}$$

$$D_{T+1}(i) = \frac{1}{N}\prod_{t=1}^{T}\frac{\exp(-\alpha_t y_i h_t(x_i))}{Z_t}. \tag{8}$$

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:

$$E = \frac{1}{N}\sum_{i=1}^{N} e^{-y_i f(x_i)}. \tag{9}$$

$$f(x_i) = \sum_{t=1}^{T} \alpha_t h_t(x_i). \tag{10}$$

$$E = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)}. \tag{11}$$

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:

$$D_{T+1}(i) = \frac{1}{N}\prod_{t=1}^{T}\frac{e^{-\alpha_t y_i h_t(x_i)}}{Z_t}. \tag{12}$$

$$D_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t = \frac{1}{N}\prod_{t=1}^{T} e^{-\alpha_t y_i h_t(x_i)} \tag{13}$$

$$D_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t = \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)} \tag{14}$$

$$\sum_{i=1}^{N} D_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)} \tag{15}$$

$$\sum_{i=1}^{N} D_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)}. \tag{16}$$

$$\prod_{t=1}^{T} Z_t = E. \tag{17}$$

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:

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y_i h_t(x_i)). \tag{18}$$

$$D_t(i) \exp(-\alpha_t y_i h_t(x_i)) = D_t(i) \exp(-\alpha_t) = (1 - \mathbb{1}(h_t(x_i) \neq y_i)) D_t(i) \exp(-\alpha_t). \tag{19}$$

$$D_t(i) \exp(-\alpha_t y_i h_t(x_i)) = D_t(i) \exp(\alpha_t) = \mathbb{1}(h_t(x_i) \neq y_i) D_t(i) \exp(\alpha_t). \tag{20}$$

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \tag{21}$$

$$Z_t = \sum_{i=1}^{N} (1 - \mathbb{1}(h_t(x_i) \neq y_i)) D_t(i) \exp(-\alpha_t) + \mathbb{1}(h_t(x_i) \neq y_i) D_t(i) \exp(\alpha_t) \tag{22}$$

$$= \left(\sum_{i=1}^{N} D_t(i) - \sum_{i=1}^{N} \mathbb{1}(h_t(x_i) \neq y_i) D_t(i) \right) \exp(-\alpha_t) + \sum_{i=1}^{N} \mathbb{1}(h_t(x_i) \neq y_i) D_t(i) \exp(\alpha_t) \tag{23}$$

$$Z_t = (1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t). \tag{24}$$

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:

$$Z_t = (1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t). \tag{25}$$

$$\frac{dZ_t}{d\alpha_t} = -(1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) = 0. \tag{26}$$

$$-(1 - \epsilon_t) + \epsilon_t \exp(2\alpha_t) = 0. \tag{27}$$

$$\epsilon_t \exp(2\alpha_t) = 1 - \epsilon_t. \tag{28}$$

$$\exp(2\alpha_t) = \frac{1 - \epsilon_t}{\epsilon_t}. \tag{29}$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right). \tag{30}$$

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 $t \in \{0, \ldots, \texttt{self.n\_clfs}\}$.
- 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: *Code*

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: *In gradient boosting, the training loss gradually declines in a smooth manner, approaching zero. However, the test loss initially decreases rapidly before leveling off with a slight increase. In contrast, AdaBoost exhibits a steady decrease in both training and test losses, reaching relatively small values but displaying significant fluctuations rather than smooth curves. This irregularity arises because AdaBoost is based on classifiers rather than regressors.*

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

Solution I: *While gradient boosting showed superior performance on the training data, AdaBoost yielded better results on the test set. This suggests that AdaBoost was more effective overall, as it is inherently designed for classification tasks, utilizing classifiers rather than regressors.*

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: *The dataset weights in AdaBoost are largest where classification uncertainty is highest. In the given dataset, the weights are most significant at the boundaries of the spirals, where distinguishing between classes is difficult, and smallest in regions further away from these boundaries.*