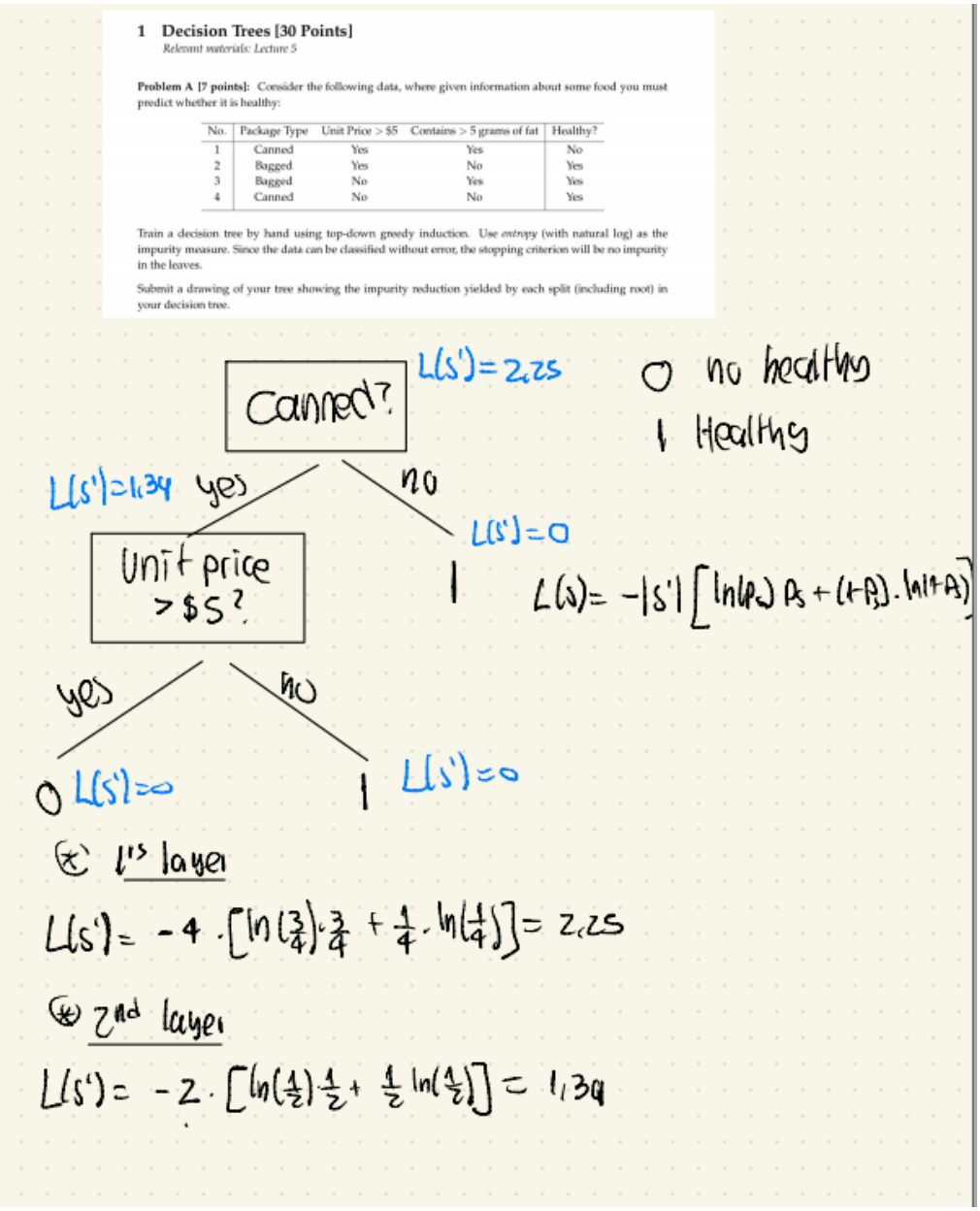
1 Decision Trees [30 Points]

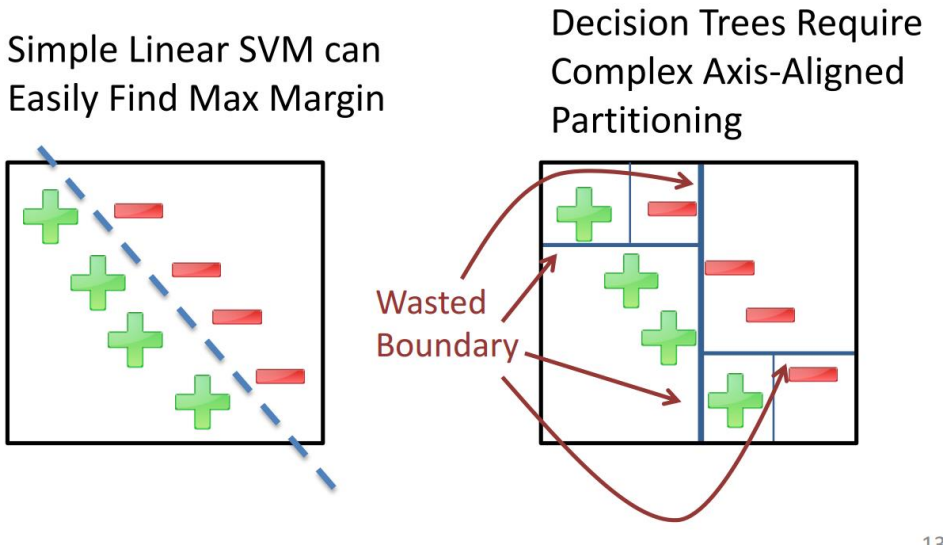
Relevant materials: Lecture 5

Problem A [7 points]:

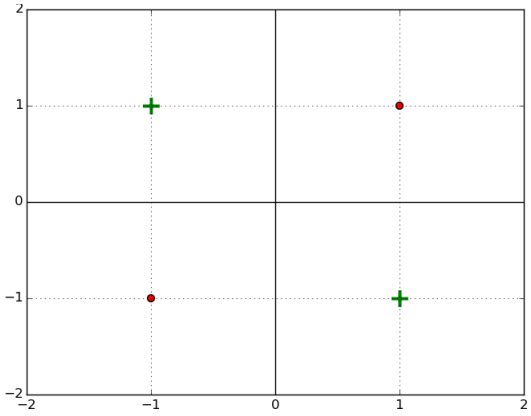


Problem B [4 points]: Compared to a linear classifier, is a decision tree always preferred for classification problems? Briefly explain why or why not. If not, draw a simple 2-D dataset that can be perfectly classified by a simple linear classifier but which requires an overly complex decision tree to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

Not always. Some problems are easier to classify using a linear classifier and using the decision tree just add more complexity to a problem that is easily solved by the linear classifier. We can see that in the k,example below (showed in class), where a linear classifier separates the data very easily, while the decision tree is more complex.



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



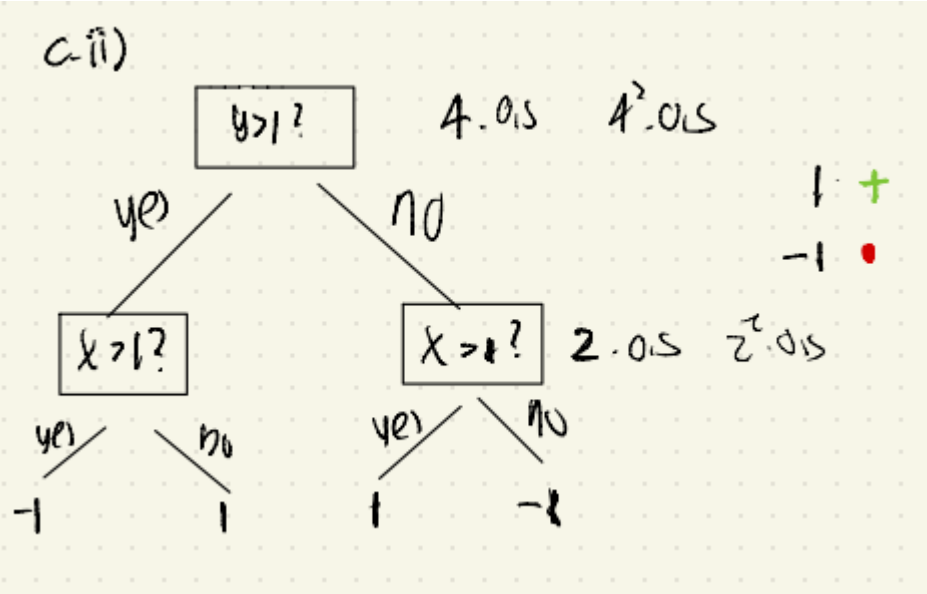
i. [5 points]: Suppose we train a decision tree on this dataset using top-down greedy induction, with the Gini index as the impurity measure. We define our stopping condition to be if no split of a node results inany reduction in impurity. Submit a drawing of the resulting tree. What is its classification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

Considering there is no majority class, the stop condition will make that the tree would be just a simple node, where the root note is also the leaf node with a prediction of either 1 or -1 with probability 0.5.

The classification error will be 0.5

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).



To classify this data we can use the Bernoulli variance as an impurity measure.

A pro of this impurity measure is that it will split the data even if the percentage of correct classification before splitting is equal to the percentage of correct classification after the split, for example in the example above.

A con is that because this impurity measure will continue splitting when other measures not, it could tend to overfit the data. There could be time when we don't want the algorithm to continue splitting.

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.

The largest number of thresholds to separate 100 data points in a 2-D dataset and having zero training error is 99. This is because if we have n points in a 2D plane, we can draw n-1 boundaries that separate all of them (in the case any of them share the same boundary)

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").

1.D

We know that for every point x_i in our training set, we have D features. Since decision trees make axis aligned partitions, it scans through each axis of x_i over all data (N).

For the first partition, we consider splitting at one of the axis of $\mathbb{R}^D$, N times. Thus, the worst case complexity is upper bounded by

$O(N \times D)$

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:

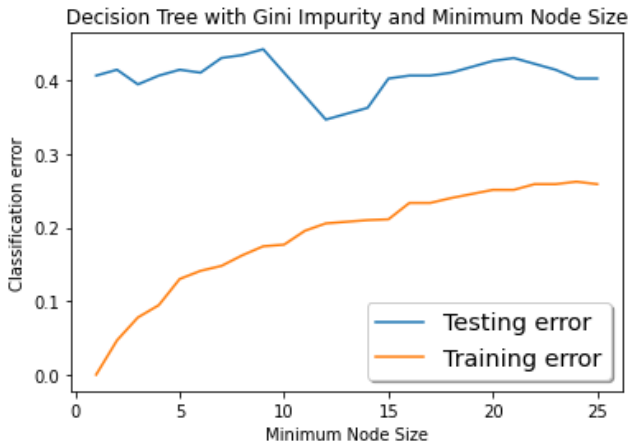
https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set

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:

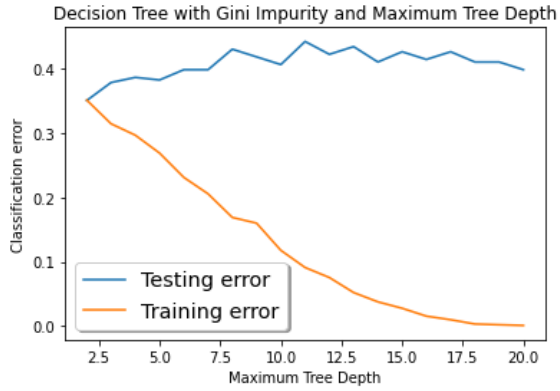
https://colab.research.google.com/drive/1OPjXSUrE7zzeKC1QHgxIlKpwUEk0nVHM?usp=sharing

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.



* `Test error minimized at min_samples_leaf = 12`

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.




```
Test error minimized at max_depth = 2
```

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.

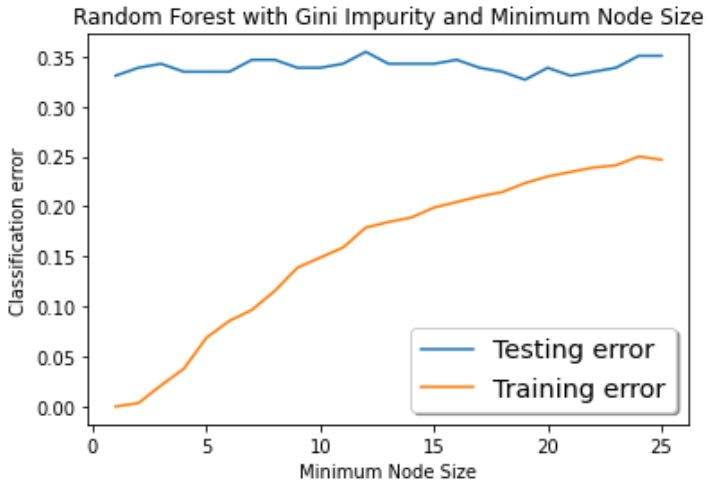
For minimal leaf node, the test error is minimized when the minimum size of a leaf node is 12. For leaf nodes size below this the model seems to be overfitting, because the training error is much lower than the testing error, but when the size approximate to 12, the test error decrease and is closer to the training error. For sizes greater than 12, the testing error keep growing steadily meanwhile the test error increase until size_leaf_node=15 and then stays almost the same. So in conclusion for sizes above 12, the model tends to underfit the data and thus, it doesn't generalize very well from the training. Early stopping can have a an important impact on the model, because if we choose it correctly, the model will stop in an optimal point where there is a good trade off between overfitting and underfitting.

For maximum tree depth criteria, both the training and test error starts with the same error value (for depth=2), which gives the smallest testing error. As the depth increases, the training error decreases being almost neglectable for depth=20, meanwhile the testing error increases. Thus, for this model, as the depth of the trees increase, the model tends to overfit (the model is not good for generalizing for data outside of the training set).

Minimum size of leads seems to be a better early stopping criterion, considering that it allows to childrens to continue splitting until they reach the minimum, while other nodes have stopped splitting when met this criterion. However, the maximum depth force every child to split until a specific layer, which can leads to overfit, which happened in this case.

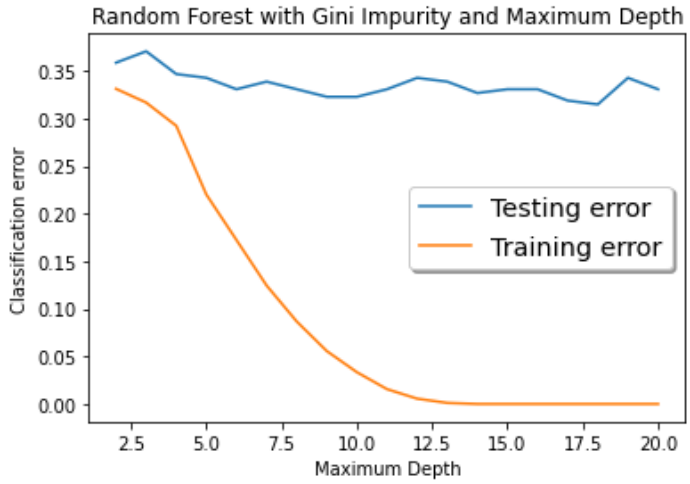
Problem C [4 points]: Choose one of the following from i or ii:

i. Train a random forest classifier using Gini as the impurity measure, minimal leaf node size as earlystopping 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.




```
Test error minimized at min_samples_leaf = 19
```

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.




```
Test error minimized at max_depth = 18
```

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.

For minimal node size, we have the minimum testing error for leaf size=19. But, if we look at the plot, we can see that the testing error remains pretty similar when the leaf size vary, but the training error increases as the leaf size increase, passing from overfitting for low leaf sizes to overfitting for high leaf sizes values. For this case, a correct selection of early stopping is important, because if we stop too early (considering that the test errors remains very similar) the model will overfit. Thus, in this case, probably the early stopping wot improve the performance of the random forest model.

For max depth, the parameter that minimizes the testing error is 18 layers of depth in the trees. For this case, the model is clearly overfitted. For this model and according to the plots, if the stopping criterion was set between 4 and 6, the testing and training error would have had similar values than the minimum size criterion (in size=19), but if the maximum layer is set below or above these values, the model tends to quickly underfit and overfit, respectively. So this is signal the model is extremely sensitive to a good election of the depth.

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.

Yes, for the case of minimum leaf stopping, random forest has a testing error which remains almost steady as the leaf size increase, meanwhile decision tree model has a clearly minimum around size=12. So for the case of decision tree, the early stopping can lead to improving the model significantly (. Another aspect is that the curves in the random forest model seem to look smoother than the decision tree, and this is probably because the random forest model decrease the variance. For the best-case scenario in the decision tree, the testing error is higher than the average of the random forest testing error, so the random forest seems to be a more stable method.

Extra Credit [7 points total] :

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

See above.

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?

See above.

3 The AdaBoost Algorithm [40 points]

Relevant materials: Lecture 6

Problem A [3 points]:

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.

In order to show this, we need to show that the inequality

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

holds for the different cases we can have.

1) y_i and $F(x_i)$ have the same sign, then

$$\begin{cases} \exp(-y_i F(x_i)) \geq 0 \\ \mathbb{1}(H(x_i) \neq y_i) = 0 \end{cases}$$

$\therefore \exp(-y_i \cdot F(x_i)) \geq \mathbb{1}(H(x_i) \neq y_i)$

2) y_i and $F(x_i)$ have opposite sign, then

$$\begin{cases} \exp(-y_i \cdot F(x_i)) \geq 1 \\ \mathbb{1}(H(x_i) \neq y_i) = 1 \end{cases}$$

$\therefore \exp(-y_i \cdot F(x_i)) \geq \mathbb{1}(H(x_i) \neq y_i)$

$\therefore \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i \cdot F(x_i)) \geq \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(H(x_i) \neq y_i)$ □

Problem B [3 points]:

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}:

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

From lecture, we have that:

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

So, $D_2 = \frac{D_1 \cdot \exp(-\alpha_1 \cdot y_i \cdot h_t(x_i))}{Z_1}$

From lecture, we know that $D_1 = \frac{1}{N}$ ← initialized as uniformly

$\therefore D_2 = \frac{1}{N} \frac{\exp(-\alpha_1 \cdot y_i \cdot h_t(x_i))}{Z_1}$

Then, $D_3 = \frac{D_2 \cdot \exp(-\alpha_2 \cdot y_i \cdot h_t(x_i))}{Z_2} =$

$D_3 = \frac{1}{N} \cdot \frac{\exp(-\alpha_1 \cdot y_i \, h_t(x_i)) \cdot \exp(-\alpha_2 \cdot y_i \cdot h_t(x_i))}{Z_1 \cdot Z_2}$

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

Problem C [2 points]:

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

Considering that

- $E = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i \cdot F(x_i))$ -- (1)
- $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ -- (2)

Using (1) and (2)

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

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

Problem D [5 points]:

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.*

1) From part C, we have that:

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

2) From part B, we have that:

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

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

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

Thus,

$$E = \sum_{i=1}^{N} D_{T+1} \prod_{t=1}^{T} Z_t = \left(\prod_{t=1}^{T} Z_t \right) \underbrace{\sum_{i=1}^{N} D_{T+1}}_{1 \text{ because } D \text{ is a distribution}}$$

$$\boxed{E = \prod_{t=1}^{T} Z_t} \quad \square$$

Problem E [5 points]:

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:

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

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

Considering the values of y_i and $h_t(x_i)$, we have that

1) case $h_t(x_i) \neq y_i$

$$Z_t = \sum_{i=1}^{T}\left(D_t(i)\exp(-\alpha_t y_i h_t(x_i))\right)$$

$$Z_t = \sum_{i=1}^{T}\left(D_t(i)\exp(\alpha_t)\right)$$

$$Z_t = \exp(\alpha_t)$$

We know that $\epsilon_t = \sum_{i=1}^{N}(D_t(i)\,\mathbb{1}(h_t(x_i) \neq y_i))$, and that for the case $(h_t(x_i) \neq y_i)$, $\epsilon_t = 1$

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

($1-\epsilon_t \to 0$, $\epsilon_t \to 1$)

2) Case $h_t(x_i) = y_i$

$$Z_t = \sum_{i=1}^{T}\left(D_t(i)\exp(-\alpha_t y_i h_t(x_i))\right)$$

$$Z_t = \sum_{i=1}^{T}\left(D_t(i)\exp(-\alpha_t)\right)$$

$$Z_t = \exp(-\alpha t)$$

Thus, using again the formula, and knowing that for this case $\epsilon = 0$, we have:

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

$$z_t = \exp(-\alpha_t)$$

Thus, the normalizer can be written as shown in the statement of the exercise.

Problem F [2 points]:

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:

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

In order to minimize Z_t w.r.t α_t, we need to take the derivative and make it equal to zero

$$\therefore \frac{\partial Z_t}{\partial \alpha_t} = (1 - \epsilon_t)(-\alpha_t) \exp(-\alpha_t) + \epsilon_t \alpha_t \exp(\alpha_t) = 0$$

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

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

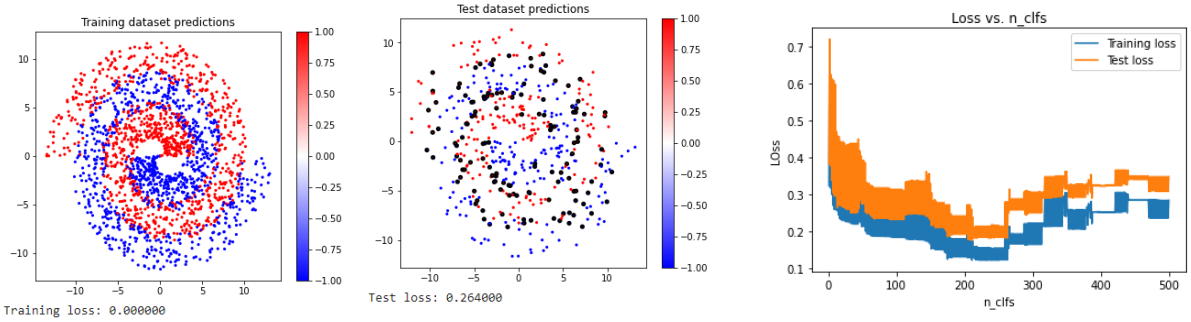
$$\ln(1 - \epsilon_t) - \ln(\epsilon_t) = 2\alpha_t$$

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

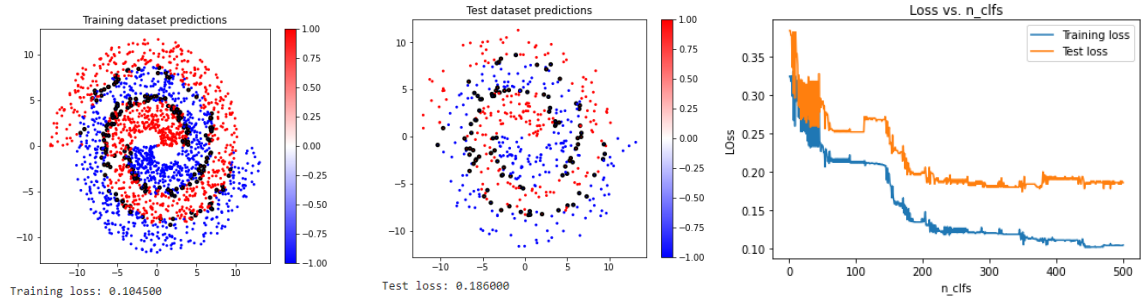
Problem G [14 points]:

https://colab.research.google.com/drive/1DuoLP_70s_N8OOTNXltR6RQNtu5UgRUl?usp=sharing

Gradient Boosting



AdaBoosting



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

Smoothness: For this behavior, the AdaBoost curves are smoother than the gradient boosting. We know that gradient boosting is designed for regression, meanwhile Adaboost is designed for classification. The fact that gradient boosting is designed for regression results in this pattern of being less smoother because weak classifiers regressors tends to have more impact and this leads to more variance in the model.

Final values: We can see that for gradient boosting, he error both of the training and test error decrease until some number of new weak regressor, but then both starts to increase again, meanwhile for Adaboost, both tends to decrease until some point and then stabilize in this value. So for gradient boosting, adding more new weak regressors improve the model until some point, but then it just makes the model to get worse. This is not the case for Adaboost, where each new weak regressor helps to improve the model and decrease the test error. This is because Adaboost trains weaks classifiers on reweighted training set.

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

Gradient boosting has less training error but higher test error than Adaboost. Adaboost performed better on the classification data set with a test loss of 0.186 vs 0.264 from gradient boosting.

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.*

We can see that the weights are largest for misclassified points and lower for points that were correctly classified

4 Convex Functions [7 points, EC 3 Points]

Problem A [3 points]:

4 Convex Functions [7 points, EC 3 Points]

This problem further develops the ideas of convex functions, and provides intuition for why convex optimization is so important for Machine Learning.

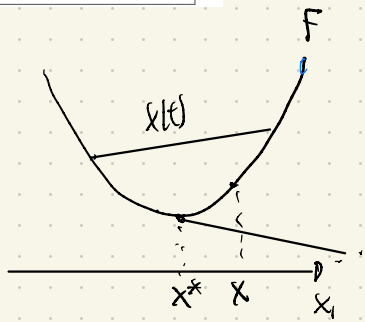
Given a convex set $\mathcal{X}$, a function $f : \mathcal{X} \to \mathbb{R}$ is **convex** if for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and all $t \in [0, 1]$:

$$f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$$

Problem A [3 points]: Let $\mathcal{X}$ be a convex set. If f is a convex function, show that any local minimum of f in $\mathcal{X}$ is also a global minimum.

Solution A:

First, suppose F is convex, and let x^* be a local minimum of F in X. So, for some neighborhood $N \subseteq X$ about x^*, we have that $F(x) \geq F(x^*)$ for all $x \in N$.



Using contradiction, we can suppose that there exists $\tilde{x} \in X$ such that $F(\tilde{x}) < F(x^*)$

Considering the line segment $x(t) = tx^* + (1-t)\tilde{x}$, $t \in [0,1]$ and noting that $x(t) \in X$ because of the convexity of X. Then, by the convexity of F

$$F(x(t)) \leq t \cdot F(x^*) + (1-t)F(\tilde{x}) < tF(x^*) + (1-t)F(x^*) = F(x^*)$$

For all $t \in [0,1]$

We can pick t to be sufficiently close to 1, then $F(x(t)) \geq F(x^*)$ by the definition of N, but $F(x(t)) < F(x^*)$ by the above inequality, which is a contradiction. Thus, it follows that $F(x^*) \leq F(x)$ for all $x \in X$, so x^* is a global minimum of F in X

Problem B [4 points]: *Using part A*, explain why convex loss functions are desirable when training learning models.

This is because in ML we are using optimization techniques to find the optimal value for errors (lower value) in order to have the best model, and as we see in the previous exercise, is the function is convex, we can guarantee that a local minimum of the function is also the global minimum. This is, if he function converges to a minimum, is guaranteed that this is the global minimum of the function, and the function is not converging in a local minimum. This makes optimization of convex functions faster, simpler and less computationally expensive, which is translated into a less time when running the models.

Problem C : [3 points, Extra Credit]

Problem C: [3 points, Extra Credit] The Kullback-Leibler (KL) divergence is a measure of statistical distance between two probability distributions (p, q), also called the relative entropy. KL divergence can be used to generate optimal parameters for visualization models (which we will also see in set 4).

$$\mathrm{KL}[P\|Q] = \sum_{x\in\mathcal{X}} p(x)\cdot\log\frac{p(x)}{q(x)}$$

Show that the KL divergence is a convex loss function.

Hint: *Use the log sum inequality*

The log sum inequality states the following:

$$\sum_{i=1}^{n} a_i \log\left(\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{n} a_i\right)\log\left(\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}\right)$$

So we can rewrite the expression of the problem as follow:

$$KL[|\theta P_1 + (1-\theta)P_2 || \theta q_1 + (1-\theta)q_2|] \quad \theta\in[0,1]$$

$$= \sum_{x\in X}\left[(\theta P_1(x) + (1-\theta)P_2)\cdot\log\left(\frac{\theta P_1 + (1-\theta)P_2}{\theta q_1 + (1-\theta)q_2}\right)\right.$$

$$\leq \sum_{x\in X}\left[\theta P_1(x)\cdot\log\left(\frac{\theta P_1(x)}{\theta q_1(x)}\right) + (1-\theta)P_2\log\left(\frac{(1-\theta)P_2(x)}{(1-\theta)q_2(x)}\right)\right]$$

$$= \theta\sum_{x\in X} P_1(x)\log\left(\frac{P_1(x)}{q_1(x)}\right) + (1-\theta)\sum_{x\in X} P_2\log\left(\frac{P_2(x)}{q_2(x)}\right)$$

$$= \theta\, KL|P_1||q_1| + (1-\theta)\, KL|P_2||q_2|$$

Thus, we have showed that $KL|P||q|$ is convex.