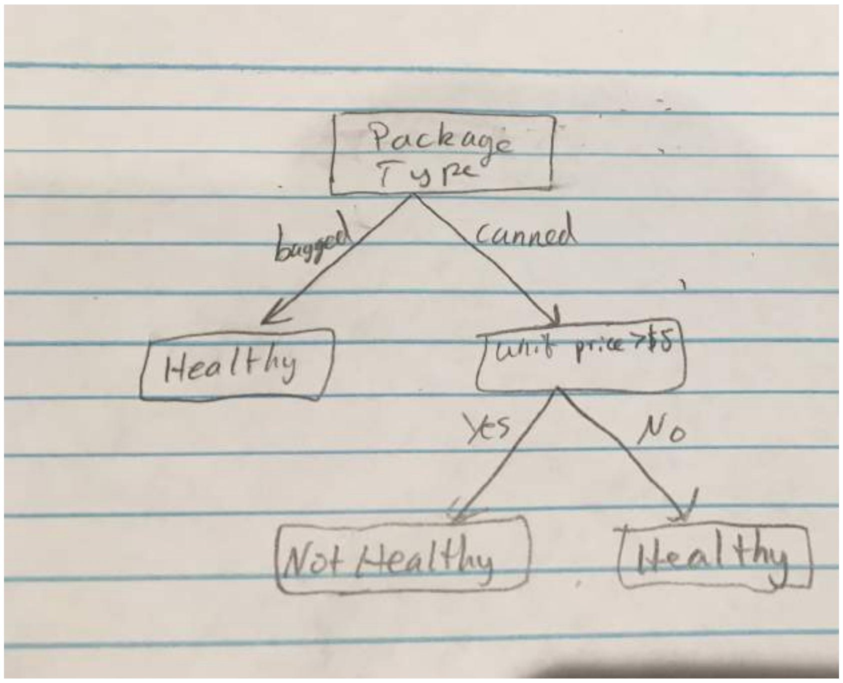
USING THREE LATE HOURS

1 Decision Trees [30 Points]

Relevant materials: Lecture 5

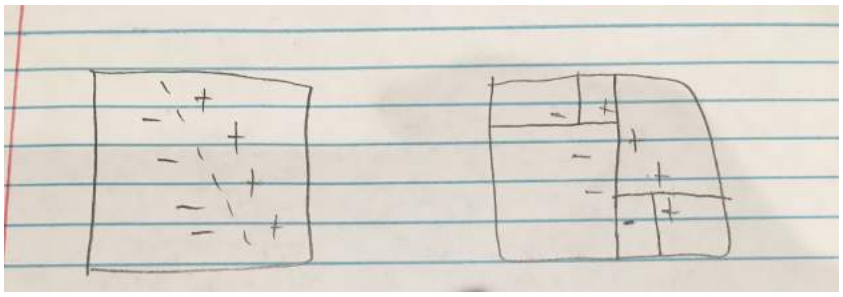
Problem A [7 points]:

Solution A: *We can calculate the entropies out to see which attribute to decide on. The original entropy is:*
$L(S^{'}) = -4(.75log(.75) + .25log(.25)) = 2.25$
Deciding with package type we get: $L(S^{'}) = -2(1log(1) + 0log(0)) - 2(.5log(.5) + .5log(.5)) = 1.39$
For unit price we get: $L(S^{'}) = -201log(0) + 1log(1)) - 2(.5log(.5) + .5log(.5)) = 1.39$
And for Contians $>$ *5g of fat we get the same thing as unit price. Thus, we can decide on any attribute and will choose package type.*
Next we see that Entropy goes to 0 as we reach leaf nodes regardless of attribute we choose. Thus, this is the tree we get:



Problem B [4 points]:

Solution B: *A decision tree is not always preferred to a linear classifier. Decision trees are better with non linear data but linear classfiers are good with linearly seperable data as we can see with this simple data set here.*

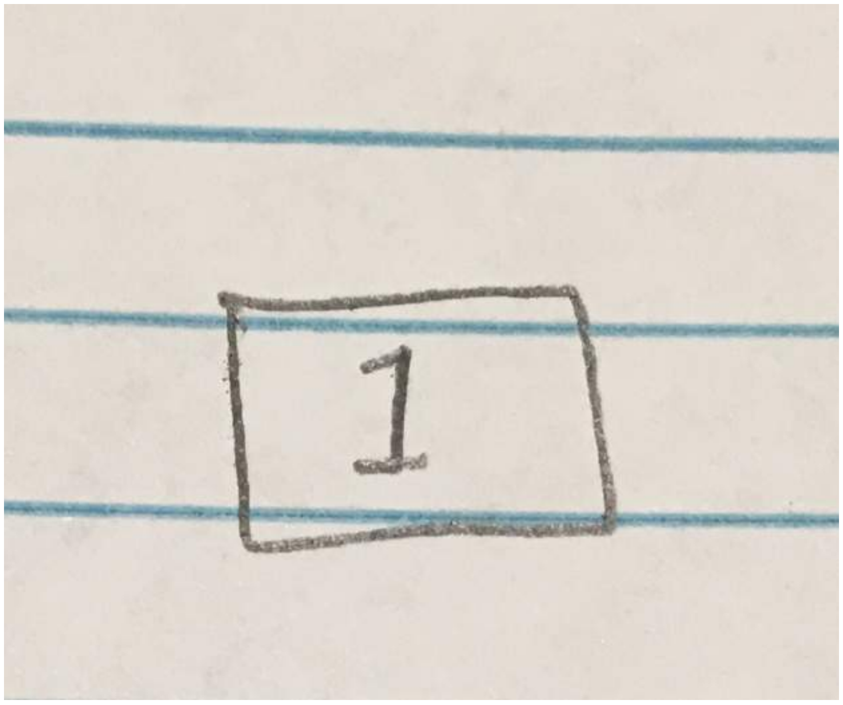


We can see a linear classifier would be fairly simple but a decision tree is very complicated.

Problem C [15 points]:

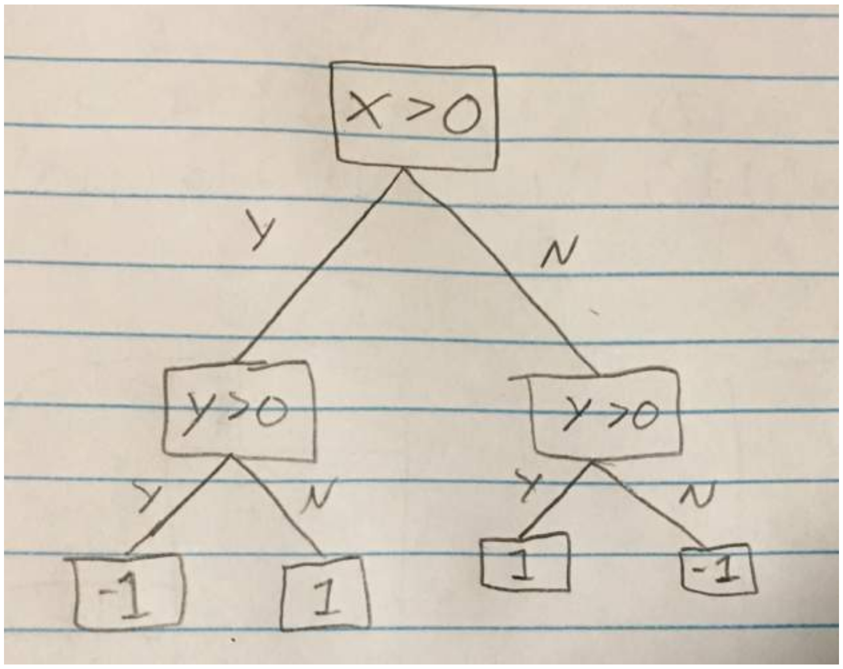
i. [5 points]:

Solution C.i: *We have impurity as* $L(S^{'}) = 4(1 - .5^2 - (1 - .5)^2) = 0.5$ *to start. We have to splits to consider: a vertical one and a horizontal one.*
For the vertical one we get $L(S^{'}) = 2(1 - .5^2 - (1 - .5)^2) + 2(1 - .5^2 - (1 - .5)^2) = 0.5$
For the horizontal split we get $L(S^{'}) = 2(1 - .5^2 - (1 - .5)^2) + 2(1 - .5^2 - (1 - .5)^2) = 0.5$
Thus, we see no split reduces impurity so we stop and are left with a root node that randomly selects a classification because there are equal amounts of positive and negative points. So, the classification error = 0.5 and the tree looks like ths:



ii. [5 points]:

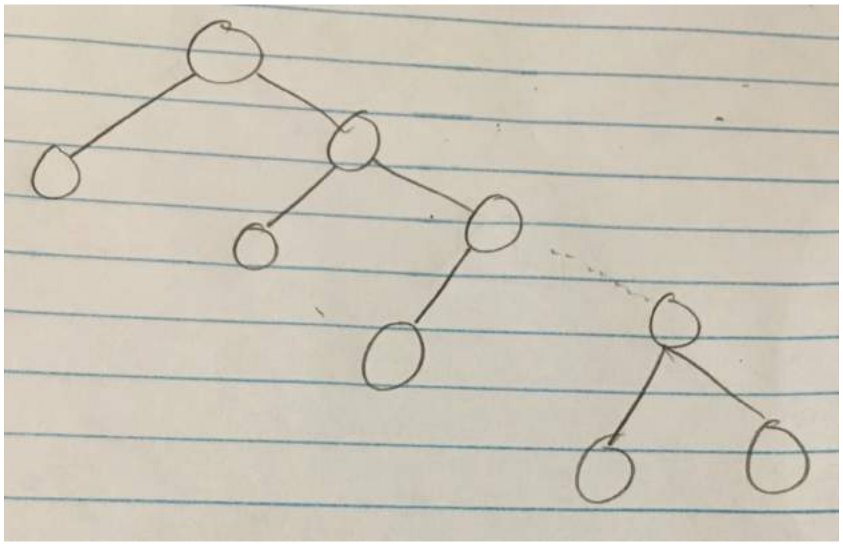
Solution C.ii:



This is a tree with 0 classification error. If you were to square Gini index you would get a tree that looked somewhat like mine. This is bacuase squaring the Gini index encourages splitting the dataset into smaller groupings even if classification error isn't improved. The pros of this are that it encourages smaller groupings which could help in some cases such as those where there is linear symmetry. Cons are that it it could be prone to overfitting as it would result in every node having it's own grouping. Basically giving the stopping condition, our modified error would result in the branching not stopping until every node was in it's own group.

iii. [5 points]:

Solution C.iii: *In the worst case the largest number of unique thresholds needed would be 99 (or N-1 in the general case). This would be if data were structured in such a way that you could only classify one data point at a time. The resulting tree would look something like this:*



Problem D [4 points]:

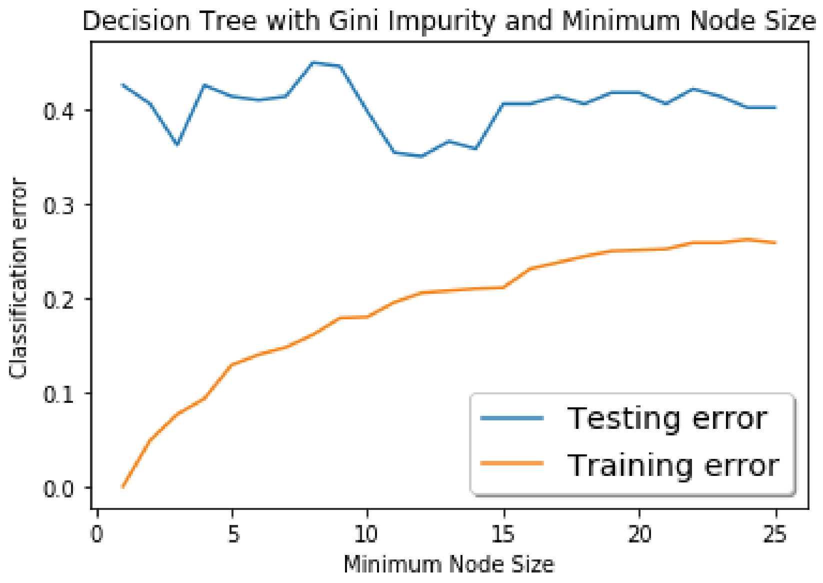
Solution D: *In the worst case you would have the N data points in a line and for every feature you would have to check between every point as well as at the ends to check where to split the data. This means every feature has N + 1 choices for its split. Thus, the worst case complexity of this problem is* $D * (N + 1)$.

2 Overfitting Decision Trees [30 Points]

Relevant materials: Lecture 5

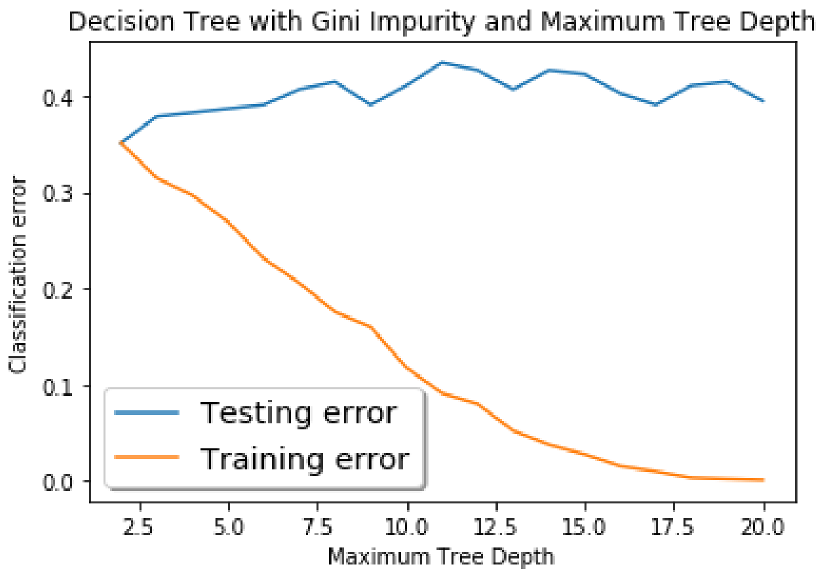
Problem A [7 points]:

Solution A:



Problem B [7 points]:

Solution B:

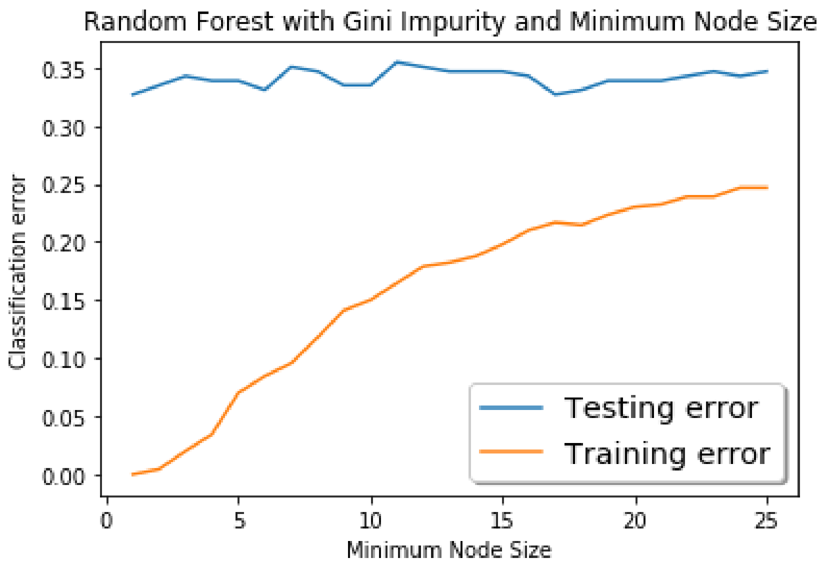


Problem C [4 points]:

Solution C: *A minimum lead node size of 12 achieves the minimum test error. A max depth of 2 minimizes the test error. Early stopping with minimum leaf size mean stoping when the size is large. Thus, we can see the earlier we stop the more likely we are to prevent overfitting becuase the earlier we stop, the less complex the model and we can see in the plot that with very complex models the test error isn't better than with mildly complex models. We can also see, however, that stopping too early could cause underfitting as the lowest test error occurs in the middle of the plot. With max depth early stopping can prevent overfitting as the best test error happens with a shallow depth.*

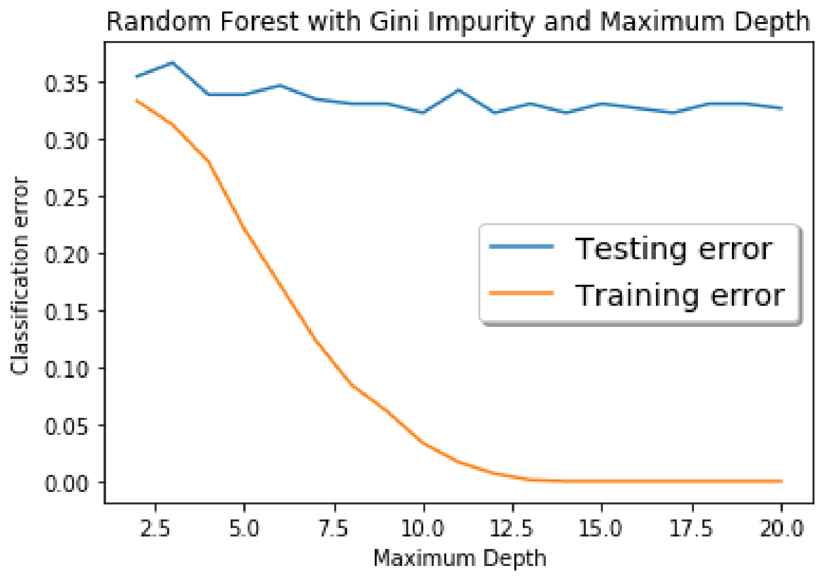
Problem D [2 points]:

Solution D:



Problem E [2 points]:

Solution E:



Problem F [4 points]:

Solution F: *A minimum lead node size of 1 achieves the minimum test error. A max depth of 10 minimizes the test error. We can see from the plots that test error doesn't change a lot with random forrests so early stopping will have less of an effect. In fact with minimum leaf nodes we can see that the minimum is when the model is most complex so early stopping could be a slight hinderance. However, with max depth we still see it will stop overfitting because testing error goes up after a depth of 10 but training error goes down. The effect is just not as large as with decision trees.*

Problem G [4 points]:

Solution G: *In general the plots of the two models are generally the same for the training error. As complexity goes up, training error goes down. The test error is more stable for random forrest than it is for the decision tree. Regarding test error decision trees are clearly more prone to overfitting than the random forrest algorithm. This is because decision trees are high variance and low bias while random forrest is low variance and higher bias. This explains the behavior of the plots where complex decision tree models have higher testing error than less complex ones and complex random forrest models don't overfit as much.*

3 The AdaBoost Algorithm [40 points]

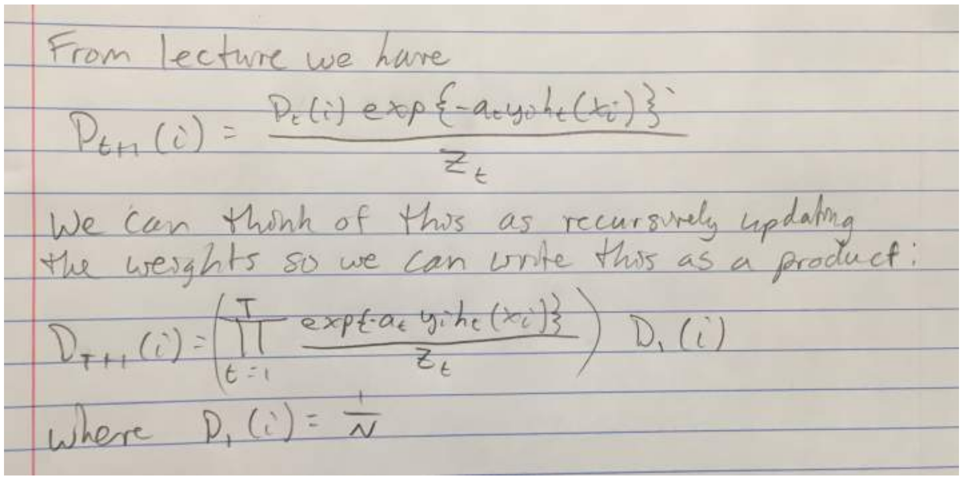
Relevant materials: Lecture 6

Problem A [3 points]:

Solution A: *We will consider 2 cases: one where* $\mathbb{1}(H(x_i) \neq y_i) = 0$ *and* $\mathbb{1}(H(x_i) \neq y_i) = 1$.
When $\mathbb{1}(H(x_i) \neq y_i) = 0$ *we trivially have that* $exp(-y_i * f(x_i)) > \mathbb{1}(H(x_i) \neq y_i)$ *because* $exp(-y_i * f(x_i)) > 0$ *for all* x_i *and* y_i *by definition. For* $\mathbb{1}(H(x_i) \neq y_i) = 1$ *we have that* $exp(-y_i * f(x_i)) > 1$ *because* $y_i * f(x_i)$ *will be less than or equal to 0 which means* $exp(-y_i * f(x_i)) > 1$ *as noted. Because* $exp(-y_i * f(x_i)) > \mathbb{1}(H(x_i) \neq y_i)$ *for all* x_i *and* y_i*, we can safely conclude* $E = (1/N)\sum_{i=1}^{N} exp(-y_i * f(x_i)) \geq (1/N)\sum_{i=1}^{N} \mathbb{1}(H(x_i) \neq y_i)$

Problem B [3 points]:

Solution B:



From lecture we have

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

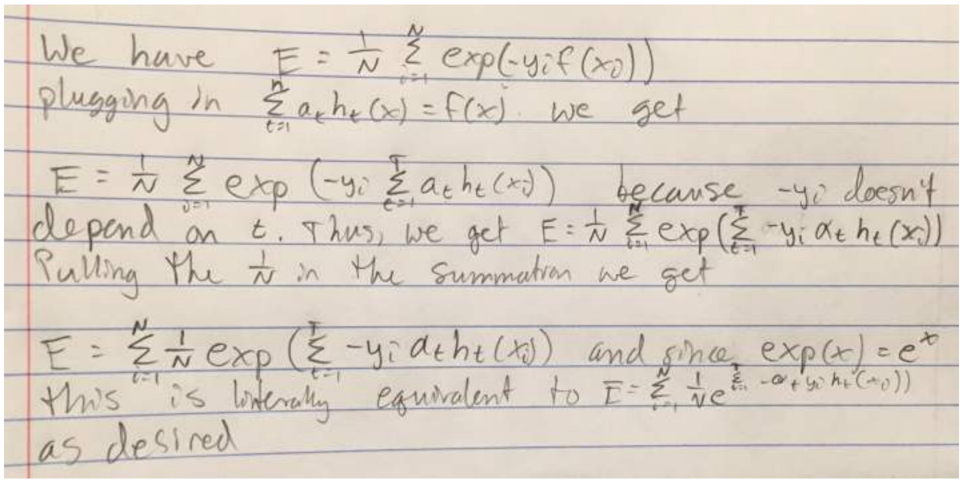
We can think of this as recursively updating the weights so we can write this as a product:

$$D_{T+1}(i) = \left(\prod_{t=1}^{T} \frac{\exp\{-a_t y_i h_t(x_i)\}}{Z_t} \right) D_1(i)$$

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

Problem C [2 points]:

Solution C:



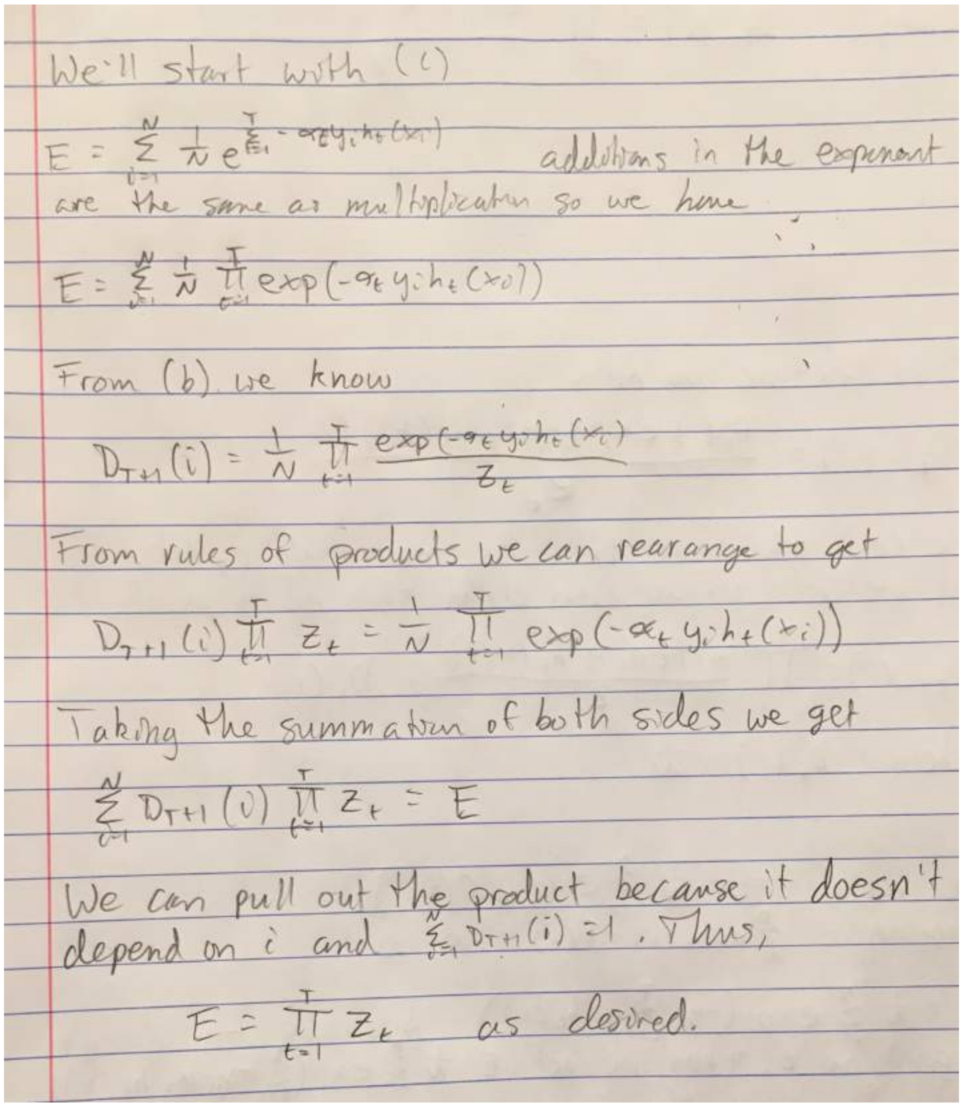
We have $E = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(x_i))$
plugging in $\sum_{t=1}^{n} a_t h_t(x) = f(x)$. we get

$E = \frac{1}{N} \sum_{i=1}^{N} \exp\left(-y_i \sum_{t=1}^{T} a_t h_t(x_i)\right)$ because $-y_i$ doesn't depend on t. Thus, we get $E = \frac{1}{N} \sum_{i=1}^{N} \exp\left(\sum_{t=1}^{T} -y_i \alpha_t h_t(x_i)\right)$
Pulling the $\frac{1}{N}$ in the summation we get

$E = \sum_{i=1}^{N} \frac{1}{N} \exp\left(\sum_{t=1}^{T} -y_i \alpha_t h_t(x_i)\right)$ and since $\exp(x) = e^x$ this is literally equivalent to $E = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i))}$
as desired

Problem D [5 points]:

Solution D:



We'll start with (c)

$E = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)}$ additions in the exponent are the same as multiplication so we have

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

From (b) we know

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

From rules of products we can rearange to get

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

Taking the summation of both sides we get

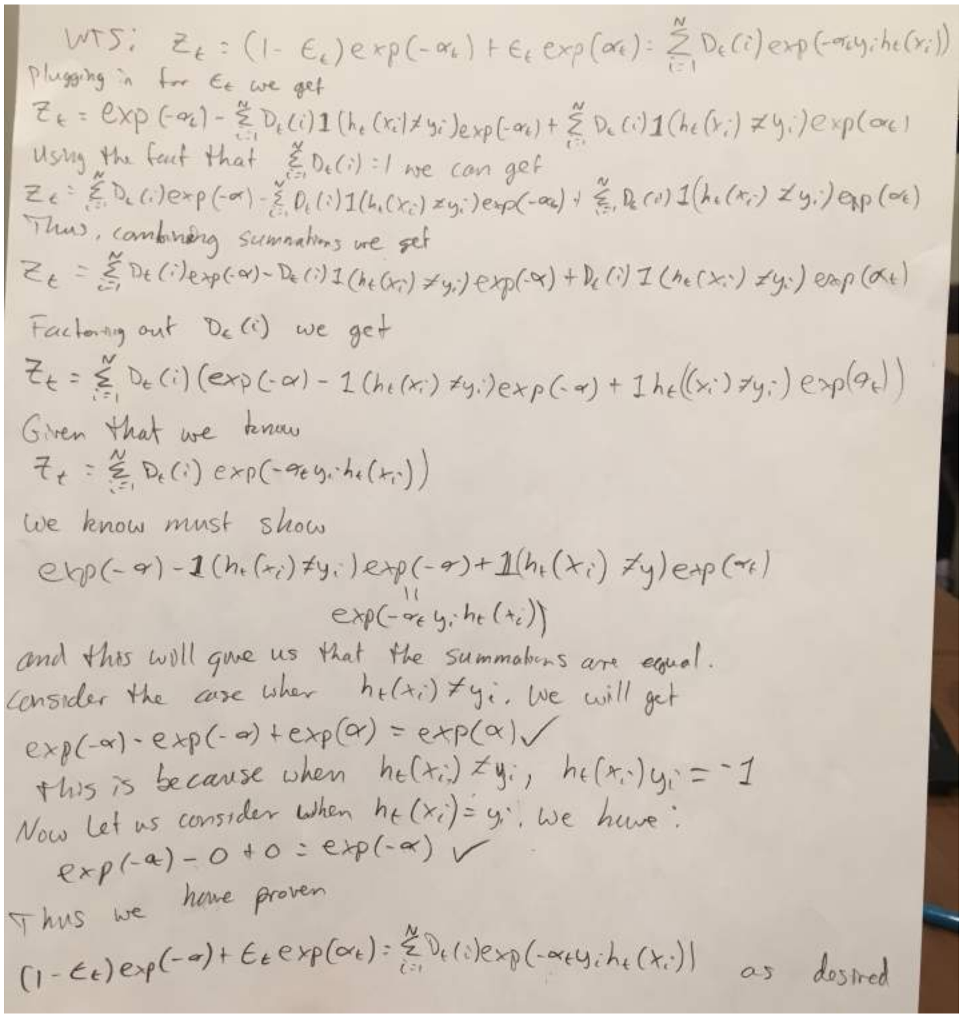
$$\sum_{i=1}^{N} D_{T+1}(i) \prod_{t=1}^{T} Z_t = E$$

We can pull out the product because it doesn't depend on i and $\sum_{i=1}^{N} D_{T+1}(i) = 1$. Thus,

$$E = \prod_{t=1}^{T} Z_t \quad \text{as desired.}$$

Problem E [5 points]:

Solution E:



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

Plugging in for ϵ_t we get

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

Using the fact that $\sum_{i=1}^{N} D_t(i) = 1$ we can get

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

Thus, combining summations we get

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

Factoring out $D_t(i)$ we get

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

Given that we know

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

We know must show

$\exp(-\alpha) - \mathbb{1}(h_t(x_i) \neq y_i)\exp(-\alpha) + \mathbb{1}(h_t(x_i) \neq y)\exp(\alpha_t)$
$\qquad \| $
$\exp(-\alpha_t y_i h_t(x_i)))$

and this will give us that the summations are equal.

Consider the case wher $h_t(x_i) \neq y_i$. We will get

$\exp(-\alpha) - \exp(-\alpha) + \exp(\alpha) = \exp(\alpha)$ ✓

this is because when $h_t(x_i) \neq y_i$, $h_t(x_i)y_i = -1$

Now let us consider when $h_t(x_i) = y_i$. We have:

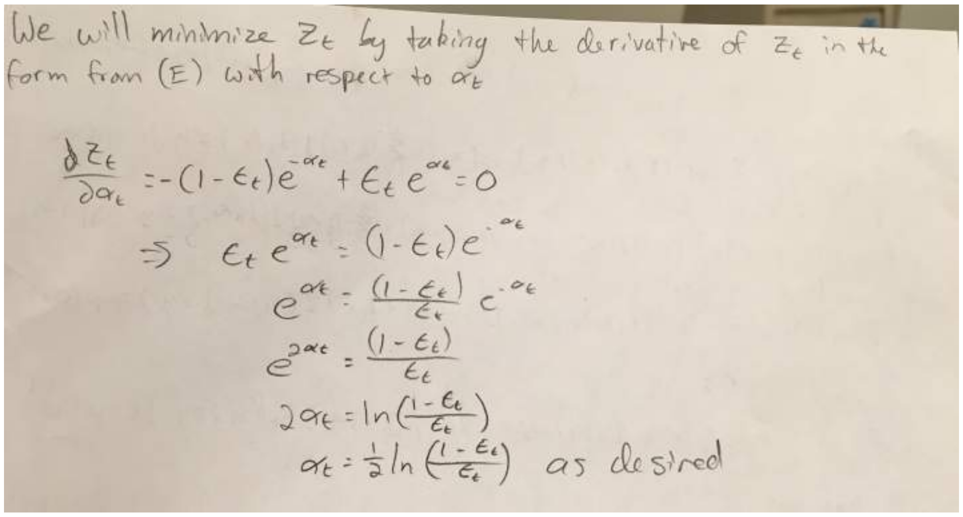
$\exp(-\alpha) - 0 + 0 = \exp(-\alpha)$ ✓

Thus we have proven

$(1-\epsilon_t)\exp(-\alpha) + \epsilon_t\exp(\alpha_t) = \sum_{i=1}^{N} D_t(i)\exp(-\alpha_t y_i h_t(x_i)))$ as desired

Problem F [2 points]:

Solution F:



We will minimize Z_t by taking the derivative of Z_t in the form from (E) with respect to α_t

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

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

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

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

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

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

Problem G [14 points]:

Solution G:

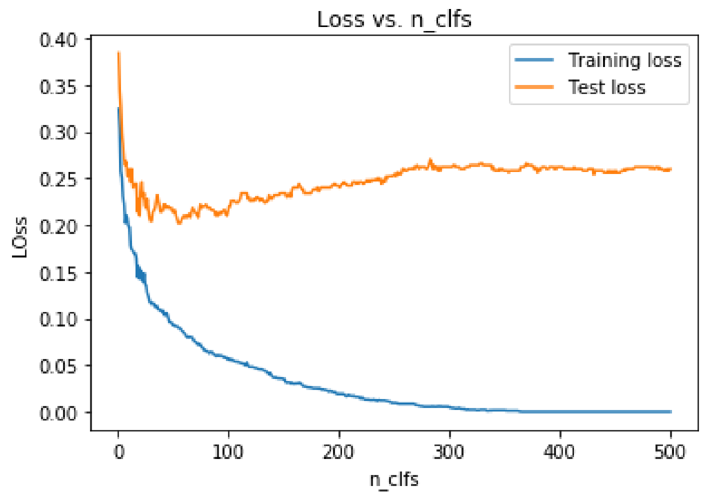


Figure 1: loss for gradient boosting

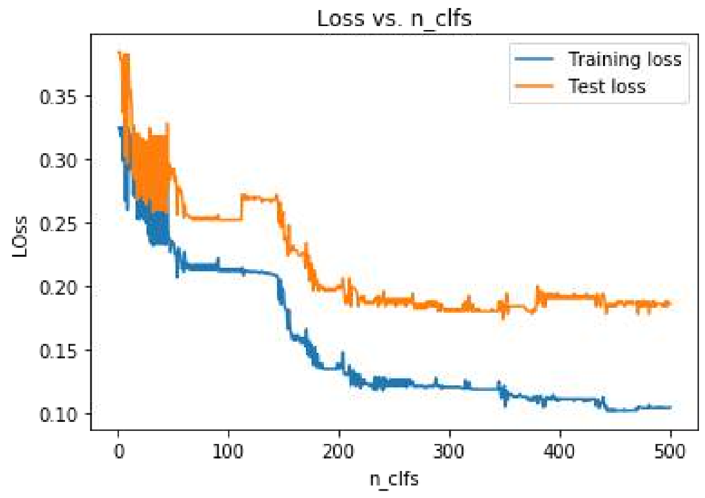


Figure 2: loss for adaboost

Problem H [2 points]:

Solution H: *Adaboost is much less smooth than gradient boosting. This is because adaboost changes the sample distribution based on how well it predicted last iteration. So especially when there are few learners outliers can sort of skew the learner as a whole which makes the loss curve jagged. Gradient boost on the other hand just works on the difficult to classify points as it adds more weak learners so the curve is more smooth. Adaboost's test error converges to a lower point but its training error converges to a higher one which is non zero. This is because as prevoiusly mentioned adaboost changes the sample distribution so getting no training error isn't likely to happen whereas with gradient boost it will eventually converge to 0 as all the points get classified correctly. However gradient boosting leads to overfitting while adaboost doesn't so adaboost gets better test error with more weak learners but gradient boosting overfits.*

Problem I [2 points]:

Solution I: *Adaboost converges to a lower test error because it doesn't overfit like gradient boosting.*

Problem J [2 points]:

Solution J: *The dataset weights are largest where the points are currently the most misclassified and they're the smallest where the points are already classified mostly correctly.*