Paper Problems

1.

a. i.

We should prefer H_2 since it is smaller and therefore more simple.

ii.

Not exactly sure what this question is asking but I'm guessing that it has to do with the fact that $\log(|H_1|) > \log(|H_2|)$ and therefore ϵ, and our max error, would have to be higher in order to offer the same guarantee.

b.

$$\begin{aligned} .95 = 1 - \delta &\implies \delta = 0.05 \\ .9 = 1 - \epsilon &\implies \epsilon = 0.1 \\ m &> \frac{1}{\epsilon}\left\{\ln 3^n + \ln\frac{1}{\delta}\right\} \\ &= \frac{1}{0.1}\left\{10\ln 3 + \ln\frac{1}{0.05}\right\} \\ &= 139.82 \\ \therefore m &= 140 \end{aligned}$$

2.

Recall that $y_i \neq h_t(x_i)$ means that $y_i h_t(x_i) = -1$ for any i that is incorrectly predicted, conversely $y_i h_t(x_i) = 1$ for any i that is correctly predicted.

This means that

$$\begin{aligned} \epsilon_t &= \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{m} D_t(i) y_i h_t(x_i)\right) \\ &= \frac{1}{2} - \frac{1}{2}\left(\sum_{y_i \neq h_t(x_i)} D_t(i) * -1 + \sum_{y_i = h_t(x_i)} D_t(i) * 1\right) \end{aligned}$$

I have no idea where to go from there... I thought I had a plan but it did not work out

3.

i. You don't say anything about a non-zero margin so I didn't bother worrying about it.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Classifier: $y = 1$ if $x_1 - x_2 - x_3 - 1 >= 0$
Weights: [1, -1, -1]
Bias: -1
Hyperplane: $1 = x_1 - x_2 - x_3$

ii.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Classifier: $y = 1$ if $-x_1 - x_2 - x_3 + 2 >= 0$
Weights: [-1, -1, -1]
Bias: 2
Hyperplane: $2 = x_1 + x_2 + x_3$

iii.

x_1	x_2	x_3	x_4	y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Function is not linearly separable with given features.
Fix by mapping features to x_1x_3, x_1x_4, x_2x_3, and x_2x_4.
Classifier: $y = 1$ if $x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 - 1 \geq 0$
Weights: [1, 1, 1, 1]
Bias: -1
Hyperplane: $1 = x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4$

iv.

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

Function is not linearly separable in 2d.
Fix by introducing cross term x_1x_2.
Classifier: $y = 1$ if $x_1 + x_2 - 2x_1x_2 + 1 >= 0$
Weights: [1, 1, -2]
Bias: 1
Hyperplane: $-1 = x_1 + x_2 - 2x_1x_2$

4.

i. $\left(x^\top y\right)^2 = \phi(x)^\top \phi(y)$

$$K(x,y) = \left(\sum_{i=1}^{2} x_i y_i\right)^2 = \sum_{i=1}^{2} x_i^2 y_i^2 + \sum_{i=2}^{2}\sum_{j=1}^{i-1}(\sqrt{2}x_i x_j)(\sqrt{2}y_i y_j) = \langle \phi(x), \phi(y)\rangle$$
$$\phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]$$
$$\phi(y) = [y_1^2, \sqrt{2}y_1y_2, y_2^2]$$

ii. $\left(x^\top y\right)^3 = \phi(x)^\top \phi(y)$

$$K(x,y) = \left(\sum_{i=1}^{2} x_i y_i\right)^3 = \sum_{i=1}^{2} x_i^3 y_i^3 + \sum_{i=2}^{2}\sum_{j=1}^{i-1}(\sqrt{3}x_i^2 x_j)(\sqrt{3}y_i^2 y_j) + \sum_{i=2}^{2}\sum_{j=1}^{i-1}(\sqrt{3}x_i x_j^2)(\sqrt{3}y_i y_j^2)$$
$$\phi(x) = [x_1^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, x_2^3]$$
$$\phi(y) = [y_1^3, \sqrt{3}y_1^2y_2, \sqrt{3}y_1y_2^2, y_2^3]$$

iii. $\left(x^\top y\right)^k$

According to my research, since as far as I know we haven't covered this, polynomial kernel of this form only has order the k interaction terms.
I'm not entirely sure how to write this so hopefully this is clear enough.
The exponents are just the square root of the binomial coefficient of that term.

$$\phi(x) = [x_1^n, \sqrt{\binom{k}{n-1}}x_1^{n-1}x_2, \sqrt{\binom{k}{n-2}}x_1^{n-2}x_2^2, ..., \sqrt{\binom{k}{n-1}}x_1x_2^{n-1}, x_2^n]$$
$$\phi(y) = [y_1^n, \sqrt{\binom{k}{n-1}}y_1^{n-1}y_2, \sqrt{\binom{k}{n-2}}y_1^{n-2}y_2^2, ..., \sqrt{\binom{k}{n-1}}y_1y_2^{n-1}, y_2^n]$$

5.

i.

$$J(\mathbf{w}, b) = \frac{1}{2} \sum_{i=1}^{5} \Big(y_i - (b + \mathbf{w}^\top \mathbf{x}_i) \Big)^2$$

ii. Too much work for 3 points...

$$\frac{\nabla J}{\nabla w_j} = -\sum_{i=1}^{5} (y_i + x_{1i} - x_{2i} + x_{3i} + 1)\, x_{ij}$$

$$\frac{\nabla J}{\nabla b} = \sum_{i=1}^{5} (y_i + x_{1i} - x_{2i} + x_{3i} + 1)$$

iii.

iv.

Yeah, this one is too much work given how hard I crammed to finish this before finding out about the extension. Given the 20 points extra credit and the 8 points from problem 4 I think I'll be fine.

Practice

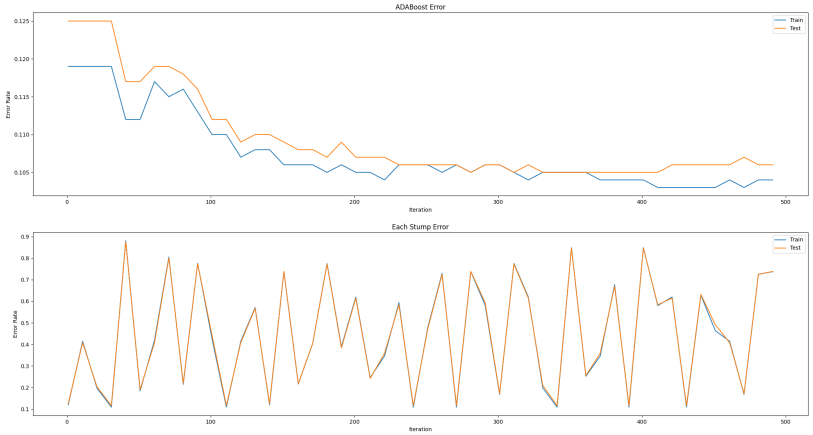
1.

Nothing needed for this.

2.

a.

Gonna start by saying that this takes forever to run. This mostly came from having to classify all of the samples and store it each iteration, taking that out brings the run time down to a much more manageable time (3 iterations per second instead of 15 seconds per iteration). In order to not take all of the grader's time running this I opted to run it for every 10 values of T rather than every value. Doing this brings the run time down to a few minutes instead of over an hour while effectively conveying the error rates for the algorithm for increasing T values. I'm hoping I don't lose too many points (if any) for this.

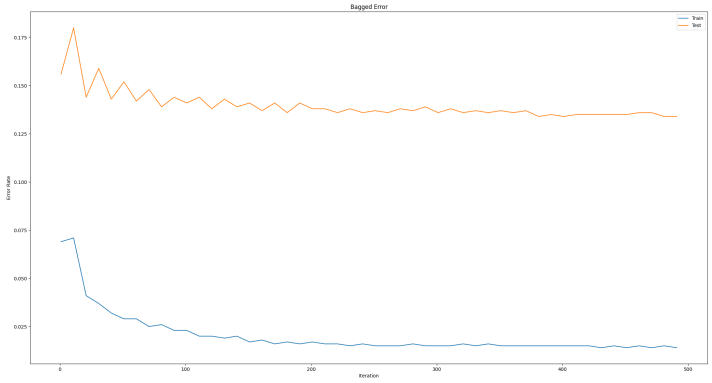


The plots for the ADABoost error rate as T increases behaves as one would expect. That is to say that the error rates for the training and test data start to converge before they start diverging as the model over fits the training data. The test error rate starts at 12.5% before eventually settling in at 10.6%. Obviously we would benefit from early stopping since we don't really see any improvement from T=250 onward. As far as the second plot, it wasn't terribly clear what you were asking for so I chose to interpret it as the error rate for the tree generated at the t^{th} iteration. This plot is kind of interesting to me because it oscillates wildly between around 20% to 80%. This kind of makes sense since every other iteration it's correcting the ones that the previous iteration missed meaning that it's going to miss more. I did not expect the test data to follow the training data so closely on a per tree basis but I suppose it makes sense if our training data is truly representative of our test data.

My code from hw1 has an error for a depth 16 tree of 15.3%. It also only took about 15 seconds to run vs about 2.5 minutes. So a depth 16 decision tree is faster but less accurate. This makes sense since ADABoost is less prone to over fitting as I understand it. That being said I prefer the full tree since its easier to implement and quite a bit faster.

b.

Similar to the previous one I only ran every 10 to save on time. Ended up with a run time of 16 minutes instead of about an hour.



My final results were a training error of 1.4% and a test error of 13.6%. So, as expected, the full decision trees over fit the training but, also as expected, this is offset by the bagging averaging them resulting in good test results. If we compare this to the max depth 16 tree ran for hw1 that had an error of 15.3% we can see that there is a sizable improvement over a single tree of comparable depth, again at the cost of some significant time.

Compared to AdaBoost the bagged trees preformed worse and it took longer to run.

c.

I'm assuming I did something wrong for this. I followed the instructions exactly and my results for the 100 trees isn't too far from the results from the bagged trees yet the question seems to imply that there should be a difference. I think it's because of how its worded that's causing confusion. How I read it is that for each test sample we're getting the predictions of the trees, averaging them into a single value then using that value to calculate the ME. What I think should be happening is that we calculate the ME for each tree then average those, but again I'm not entirely sure. Them being not far off from each other makes sense given that what I ended up doing is using the 100 samples as its own bag, which is why I think I did it wrong. I'm hoping I'm not going to be docked too many points for following the instructions exactly rather than intuiting what makes sense to do.

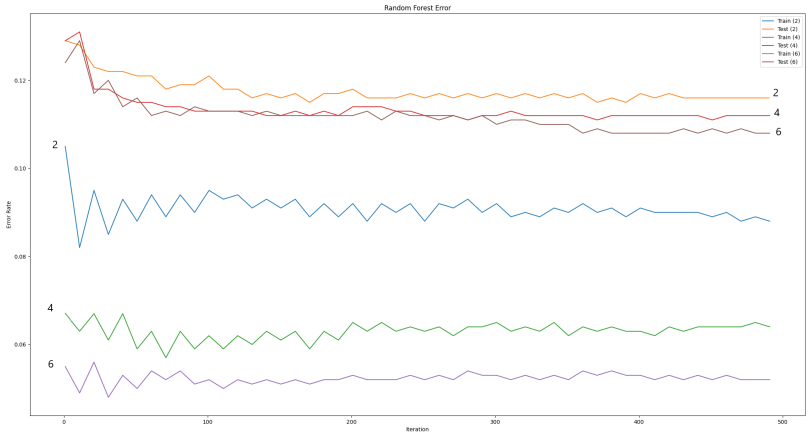
Anyway, if I HAD done it the way I outlined in the previous paragraph, which I can't because I don't have another 30 hours to run it, I would expect that the "single tree" bias would be similar to the bagged trees bias while the variance of the single tree is significantly higher. That would match what was gone over in class as well as what should intuitively happen given that we expect the variance to decrease as we increase the number of sources. Regardless, using the method outlined in the assignment my results can be found below.

Type	Bias	Variance	Squared Error
Single Tree	0.092376	0.028706	0.121082
Bagged Trees	0.091872	0.026476	0.118348

If I have time to go back and rerun it with what I believe to be the correct method I will include it under here. If there's nothing there assume I didn't have time.

d.

So the random forest method ended up with test error rates of 11.6%, 11.2%, and 10.8% for 2, 4, and 6 features respectively. This is better than AdaBoost and bagged trees. Additionally this didn't even come at the cost of an increased run time vs bagged trees. The total run time for all three feature size of about 36 minutes with each taking about 12 minutes to run. I added feature numbers next to the lines of the plot to make referencing it easier and, as expected, we kind of start to over fit the training data when selecting from more possible features. However the higher amount of features, which bear in mind is still a lot less than the actual feature list, gives decent improvement of results.



e.

Much like part c, I'm not convinced I did this correctly but regardless I will post what I did in hopes of getting at least some points on it. The following table shows the results of the bias variance decomposition for random forests. Unlike the bagged trees that saw a (very) slight reduction in variance, the random forest's variance is basically the same for the individual tree as it is for the entire forest. I recall hearing something about an individual tree from a forest being an accurate estimation for the forest itself but I cannot recall the specifics about that, maybe that's what I'm seeing here? In which case that might mean that I interpreted the instructions correctly... I GUESS a 10% decrease in variance for bagged trees might be considered significant, idk just seems smaller than I would have expected. Whatever, time is short, moving on.

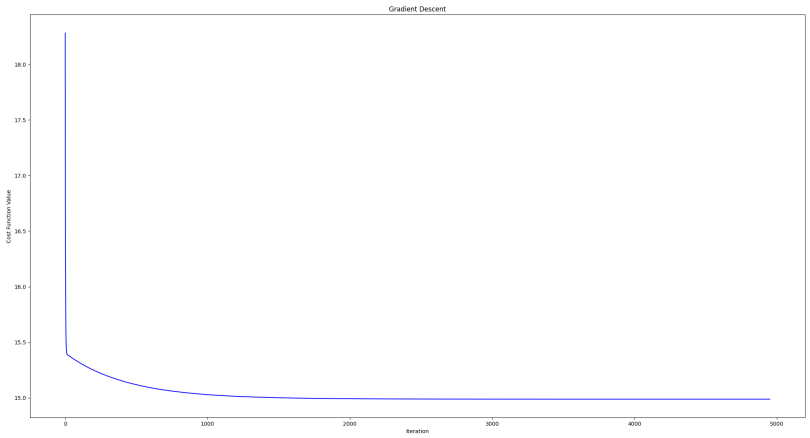
Type	Bias	Variance	Squared Error
Single Tree	0.093462	0.027906	0.121368
Random Forest	0.093056	0.028038	0.121094

3.

Skipping

4.

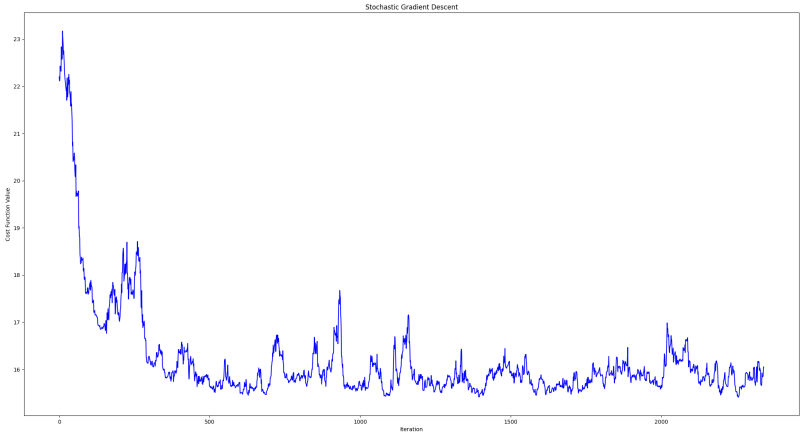
a. Playing with learning rates, I didn't find one that converged until I hit r = 0.01. The plot of the cost over iterations is shown below and it looks as expected, quickly gets "near" the solution then takes a while to finish getting there.



Weights:[0.8972, 0.7828, 0.8473, 1.2962, 0.1293, 1.5677, 0.9952, -0.0153]

Cost function value: 23.357

b. I didn't play with the learning rate here, the 0.01 from the previous problem worked for this one as well.



Weights:[0.0754, -0.0982 , -0.0910, 0.6028, -0.0469, 0.3547, 0.1394, -0.0218]

Cost function value: 21.245

c.

Weights: [0.9006, 0.7863, 0.8510, 1.2989, 0.1299, 1.5722, 0.9987, -0.0152]

Cost of Test: 23.362

So both batch gradient and stochastic gradient descent end up with a lower cost function value for the test data than this directly computed optimal weight vectors. This is because optimal for training obviously doesn't mean optimal for test data.