Robert Weischedel
CS 6350
9/24/17

Homework 2: Online Learning and Perceptron

Problem 1 - Warm Up: Linear Classifiers and Boolean Functions

It should be noted that I solved each of these by first building a truth table and then trying different combinations till I found one that worked.

1

$x1 \vee \neg x2 \vee x3$

0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Solution: $2x1 - x2 + 2x3 \geq 0$
This is linearly separable.

2

$(x1 \vee x2) \wedge (x2 \vee x3)$

0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Solution: $x1 + 2x2 + x3 \geq 2$
This is linearly separable.

3
$(x1 \wedge \neg x2) \vee x3$

0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Solution: $3x1 - x2 + 4x3 \geq 3$
This is linearly separable.

4
$x1 \oplus x2 \oplus x3$

0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Solution: This equation is not linearly separable, this is due to it having XORs in the equation.

5
$\neg x1 \wedge x2 \wedge \neg x3$

0	0	0	0
0	0	1	0

0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Solution: $-x1 + 2x2 - x3 \geq 2$
This is linearly separable.

Problem 2 - Mistake Bound Model of Learning

1a)
Because we know that l is an integer and that $1 \leq l \leq 80$, the size of the concept class has to be 80.

1b)
Since we have two outcomes in the given equation we must consider both and right a new equation for both cases. If we know that $y^t = 1$, and $|x_1| > l \;\&\&\; |x_2| > l$ or in the that case that $y^t = -1$ and $|x_1| \leq l \;\&\&\; |x_2| \leq l$ then we know that a mistake has occurred. Or when $y_t((|x_1| \leq l) \;\&\&\; (|x_2| \leq l)) \leq 0$

1c)
Since we don't know what our l value is going to be (it can be any integer from 1 to 80) we have to take into account for errors and updates in both directions. So in the case that we are expecting a $y_t = 1$ and we got -1, then we will have to shrink the l size by one (l--). If we are expecting $y_t = -1$ and we got 1 we will need to increase the size of l by one (l++).

1d)
LearningAlgorithm(x[], y_t, l){

```
        for each x_i in x
                if( (|x_1| <= l && |x_2| <= l) )
                        if( y_t == -1 )
                                l --
                else
                        if(y_t == 1 )
                                l++
        return l
}
```

If we were to set l to 1 and since the 80 is the maximum number of mistakes, we take $|C| - l$ which is equal to 80 - 1 = 79. So 79 would be out total number of mistakes.

2

For this problem we will look at the algorithm that was give on the online learn slides. Where we ask the question, How many mistakes will the Halving Algorithm make? Except like mentioned above we only consider the case of what if there is only one expert, when we need to consider the case that that are m experts. Or in other words that are m functions left over in $|C|$.

So our equation will look like this:

$$\begin{aligned} m = |c_n| &< \tfrac{1}{2}\left|C_{n-1}\right| \\ &< (\tfrac{1}{2})^2\left|C_{n-2}\right| \\ &\dots \\ &< (\tfrac{1}{2})^n |C| \end{aligned}$$

Next we multiply each side by 2^n and we get:

$$|C| > 2^n(m)$$

Since we know that $|C| == N$ we can substitute it into the equation and reduce it:

$$(2log2)log(n/m) > n$$

This means that the mistake bound for the Halving Algorithm for when we have m perfect experts would be:

$$O(\log(\tfrac{n}{m}))$$

Problem 3 The Perceptron Algorithm and its Variants

Algorithms

Each of the 5 variants are created in the attached python file, named very similiar to what variant they are. Example: Simple Perceptron -> simplePerceptron(), etc.

Experiment

1 - Write Up

For my project I decided to use Python for this assignment because I had heard of the different matrix libraries that were available for it, that would make it easy to perform all the required matrix operations. As you can see from the very ugly Python code, this is my first time working in Python and struggled to learn is various complexities. I definitely should've started earlier on this assignment, especially when I was attempting to do this on a new programming language. For all my input files, I turned the matrix or data and labels into their own separate matrices.

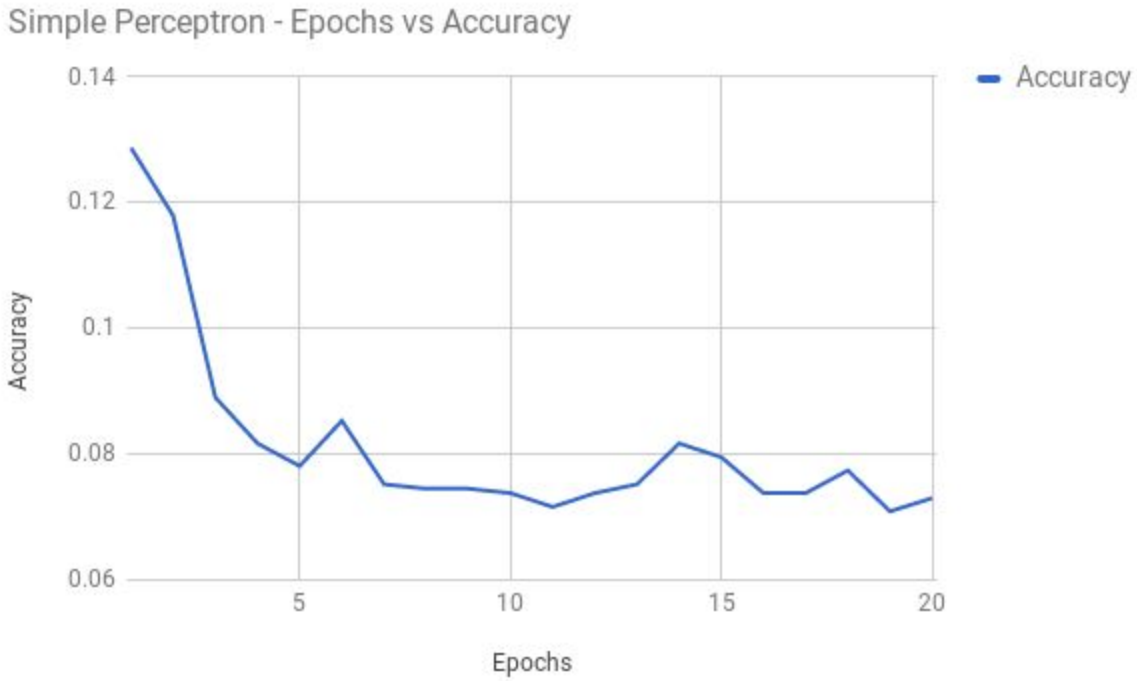
Then I created a separate function for each of the five variants. The functions at are called experiments complete what each of the three experiment requirements are asked to do. For example, the function called experiment1() performs the requirements in part 1 in 3.3. The script goes as follows first experiment1 is called where I perform the required cross validations and find the best hyper parameter for each variation. Then with those best hyper parameters I call experiement2 on phishing.dev and phishing.test where I use those values to find the best epoch for each variation, and then determine the best of the best. Then I call experiment3 with the best variation, epoch value, and hyper parameters on the phishing.train and phishing.test files.

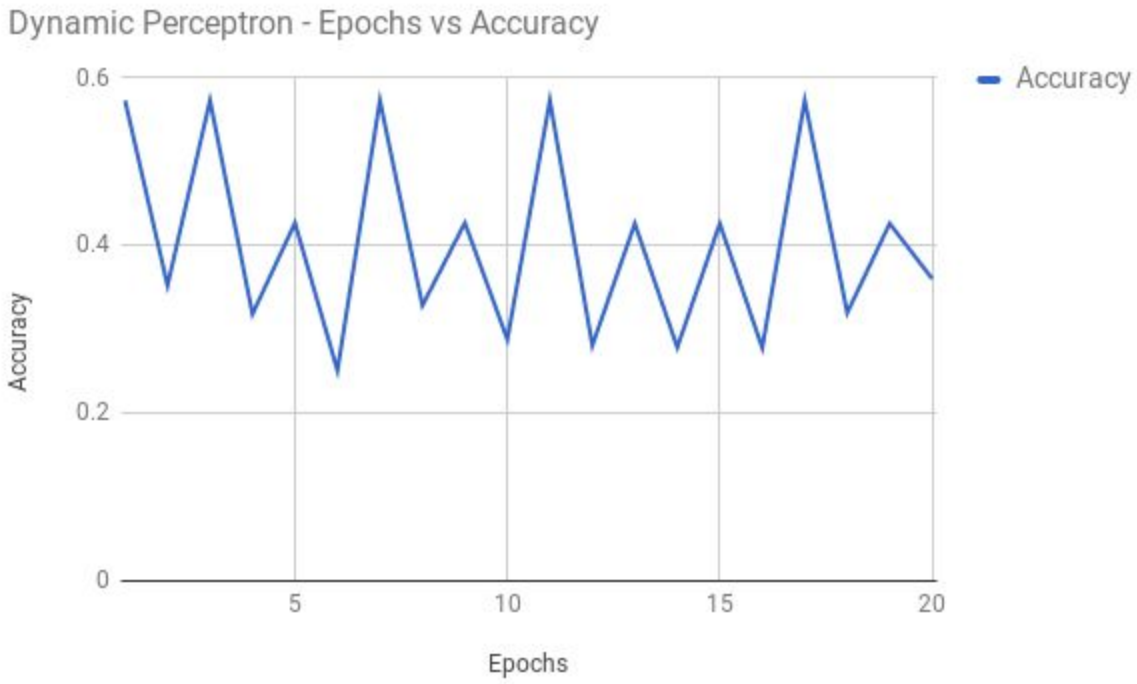
Although I have completed all the requirements for the assignment, unfortunately they are far from perfect. I struggled to get good accuracy values for the entire assignment, like from the first experiment where you are supposed to train for 10 epochs to find the best hyper parameter, I struggled to get above the mid 60% range. (Note all accuracies are in decimal form) And this unfortunately caused the rest of the experiments to suffer. Although my returns are terrible, I ask and beg you that you would still consider the amount of work and effort I put into the assignment, and if possible give as much partial credit as you can. I really struggled with this assignment. As a recap, although my values are terrible, I did do all that I was asked for it. I just have some bug that I can't figure out, and even after delaying turning in the assignment for another day I still couldn't find it. My best guess would have to be that I might be updated my weights for each variant incorrectly, but like I said I'm not sure. Thank you.

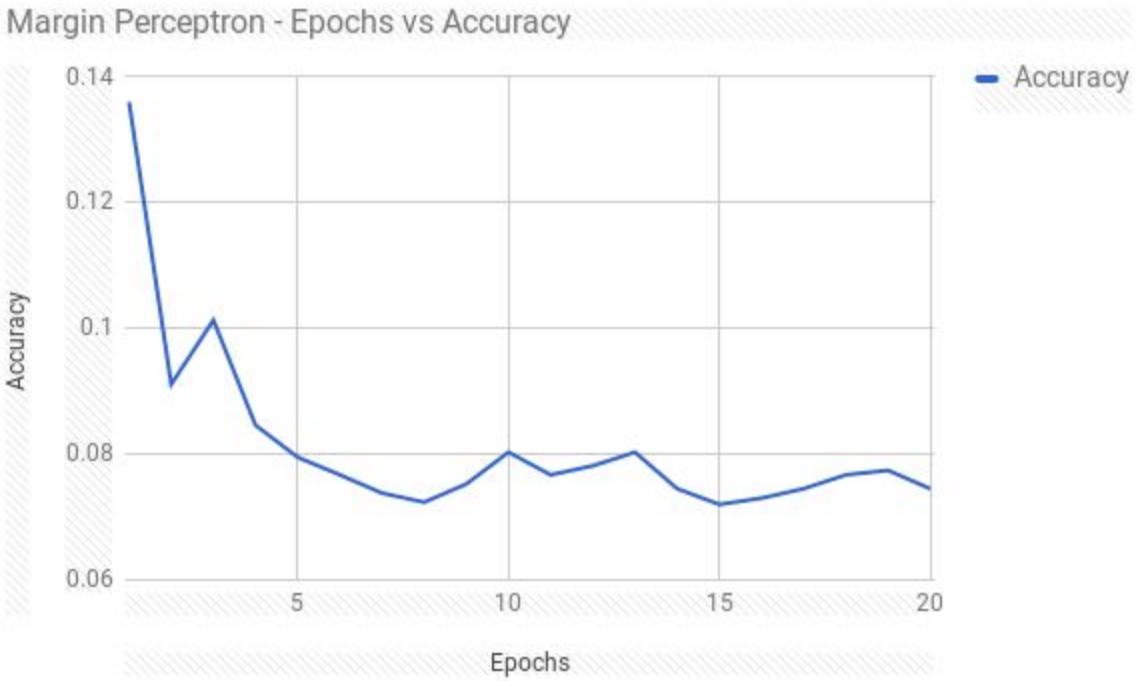
2 - Majority Base Line
See code

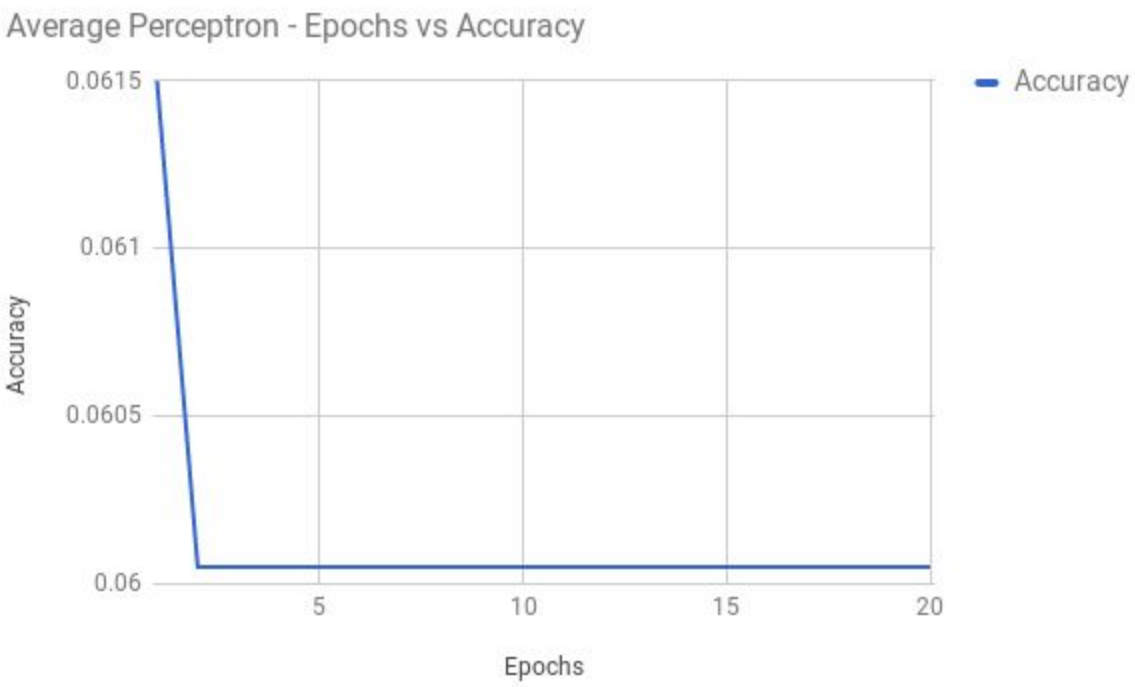
3 - Reports for Variants
I have printed all information that is required in the program. See the code printouts.

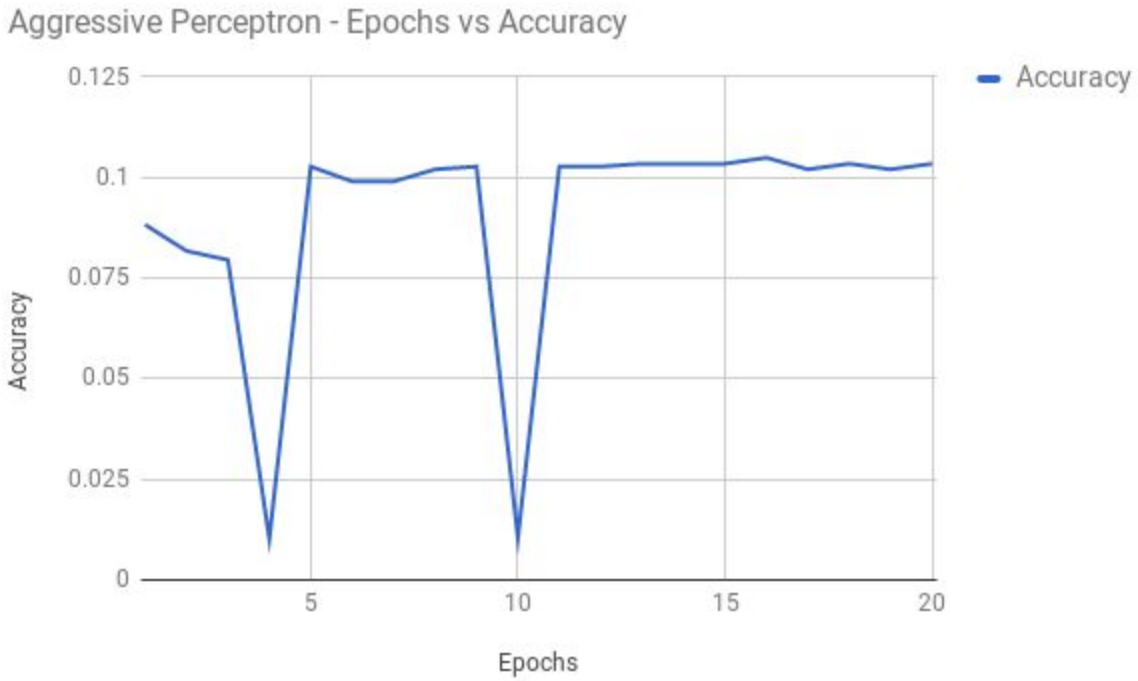
4 - Graphs


Simple Perceptron - Epochs vs Accuracy
Accuracy
0.14
0.12
0.1
0.08
0.06
Accuracy
5
10
15
20
Epochs


Dynamic Perceptron - Epochs vs Accuracy
Accuracy
0.6
0.4
0.2
0
Accuracy
5
10
15
20
Epochs


Margin Perceptron - Epochs vs Accuracy
Accuracy
0.14
0.12
0.1
0.08
0.06
Accuracy
5
10
15
20
Epochs


Average Perceptron - Epochs vs Accuracy
Accuracy
0.0615
0.061
0.0605
0.06
Accuracy
5
10
15
20
Epochs


Aggressive Perceptron - Epochs vs Accuracy
Accuracy
0.125
0.1
0.075
0.05
0.025
0
5
10
15
20
Epochs
Accuracy