PS 2

Shalin Shah

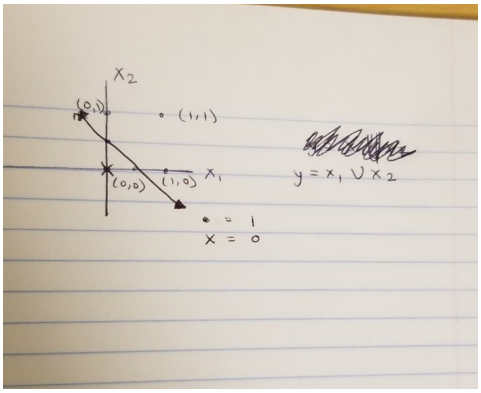
February 9, 2020

1 Perceptron

1.a Design two-input perceptron for following:

1.a.i OR

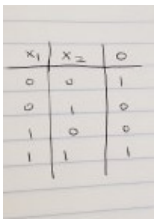
What we want to do here is find weights, bias, and threshold that make the OR gate possible.
Mathematically, if $\sum_{d=1}^{2} w_d x_d + b > \theta$, then our output, y, is 1. Else, y is -1. Expanding this out, we get that if: $w_1 x_1 + w_2 x_2 + b - \theta > 0$, then y is 1. This is the hyperplane that we need to solve for. If we set the bias equal to 0, $b = 0$, then our updated plane is: $w_1 x_1 + w_2 x_2 - \theta > 0$. Some work that I've done to understand the problem better:



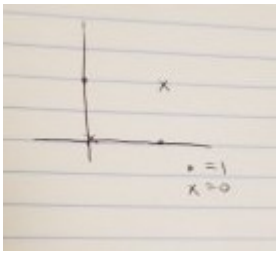
What I get from here is that for the OR gate, $(0, 0)$ should be categorized as a no but everything else a yes. So, an inequality that could do this could be: $1 * x_1 + 1 * x_2 > 0.5$. Through this means, we see that $\theta = 0.5$ and that $\overrightarrow{w} = [1, 1]$. To show that this isn't unique, we could also do $2 * x_1 + x_2 > 0.5$. This would be the same line but with a steeper slope, and would still categorize the data correctly. In this case, $\overrightarrow{w} = [2, 1]$.

1.a.ii XOR

We want to do the same thing that we did for the last part, but this time implement an XOR gate. An XOR gate works with the following truth table:



If we look at the following figure, we see that this problem is unsolvable with this perceptron model due to the fact that we can't separate the data linearly:

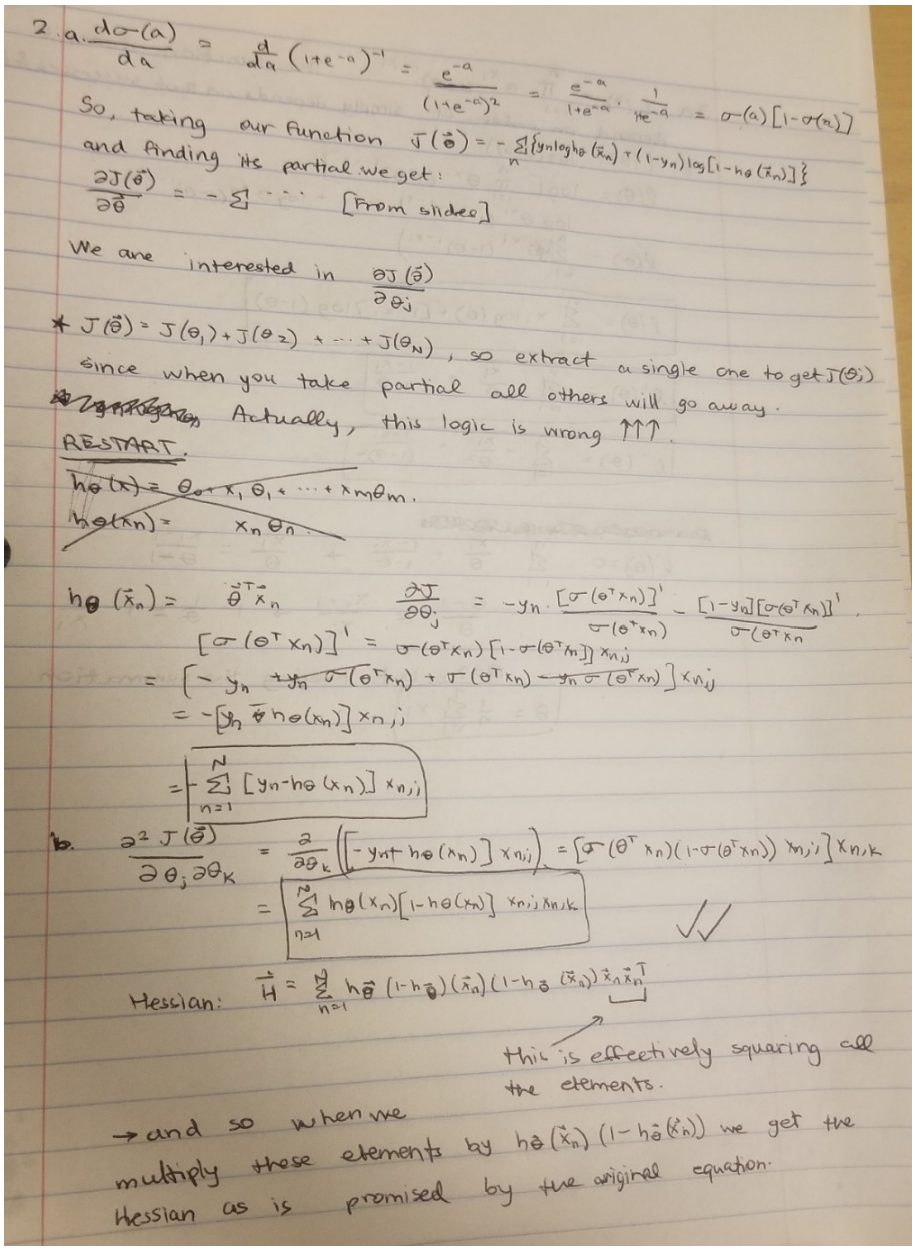


So, using the current model, no perceptron exists.

2 Logistic Regression

2.a Find partial derivative

The answers for question 2 are included in the picture below:



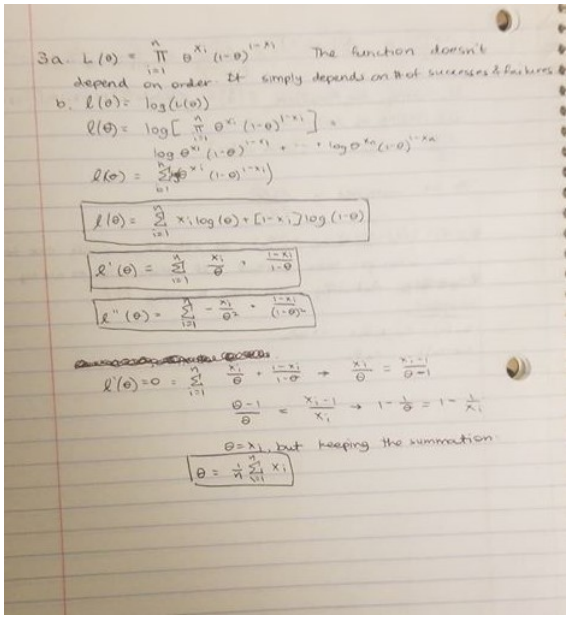
3 Maximum Likelihood Estimation

3.a Write a formula for likelihood function $L(\theta)$

See picture included in part (b).

3.b Find *log likelihood* function $l(\theta)$ and its first and second derivatives, and use these to find closed-form formula for MLE.

See picture below.



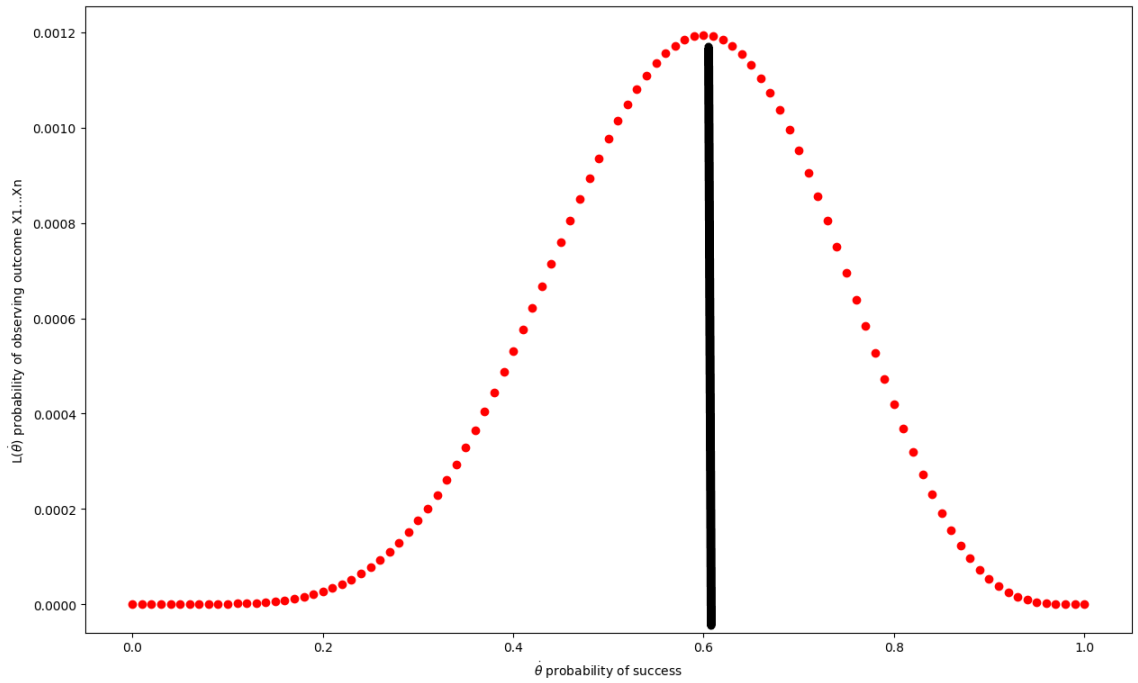
We see here that when we take the log in part (b) using the equation from part (a), that the pi notation turns into a sigma notation due to the properties of log, and the exponents drop to multiply by the log of θ and $1-\theta$. This allows for us to do calculations more easily. When we find a closed-form formula the MLE in part (b), we see that:

$$\theta = \frac{\sum_{i=1}^{n}}{n}$$

which is the value of theta that will maximize the probability that we see the current distribution. This is maximized (through intuition) at the θ value that corresponds to the true proportion of successes we see.

3.c $n = 10$ with 6 1s and 4 0s.

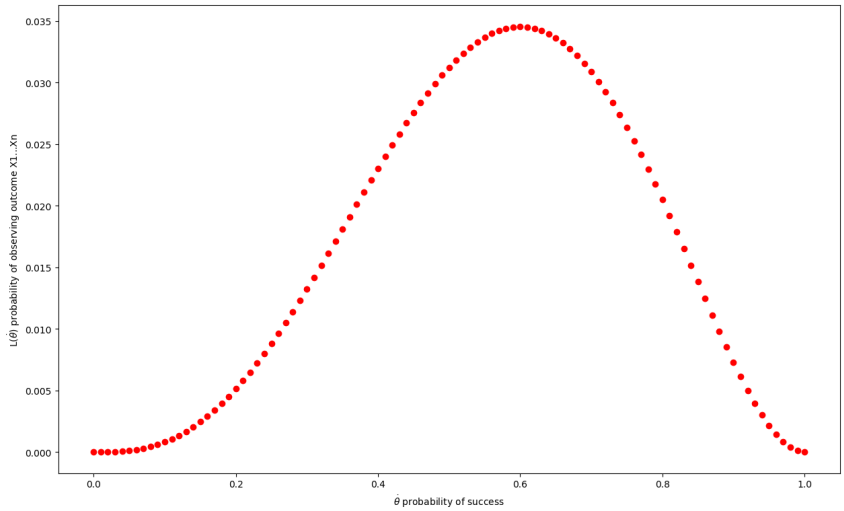
The plot is shown below:



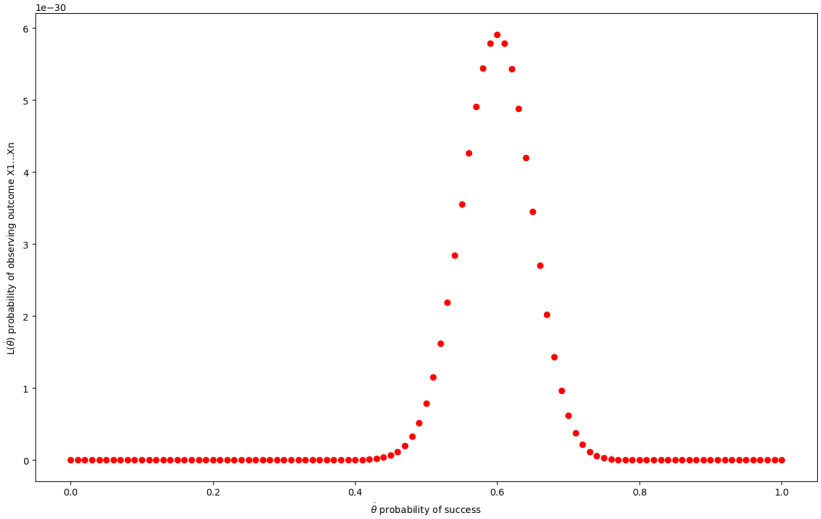
The line that's drawn in the figure corresponds to $\hat{\theta}_{MLE}$. We see that this isn't *exactly* at 0.6, which is the proportion of successes, but is *very* close to it. So, this does agree with the closed form answer from part (b).

3.d $n = 5$ with 3 1s and 2 0s, $n = 100$ with 60 1s and 40 0s, $n = 10$ with 5 1s and 5 0s.

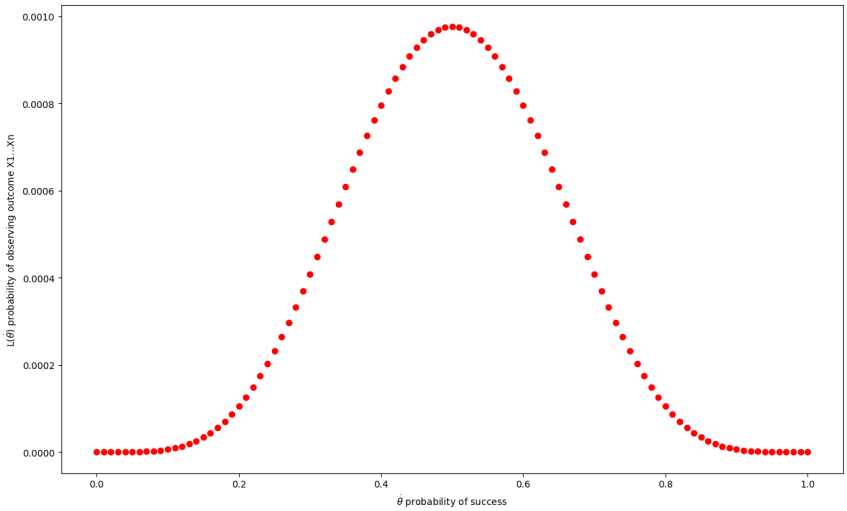
The plots are shown below:



Plot 1 – $n = 5$ with 3 1s and 2 0s



Plot 2 – $n = 100$ with 60 1s and 40 0s



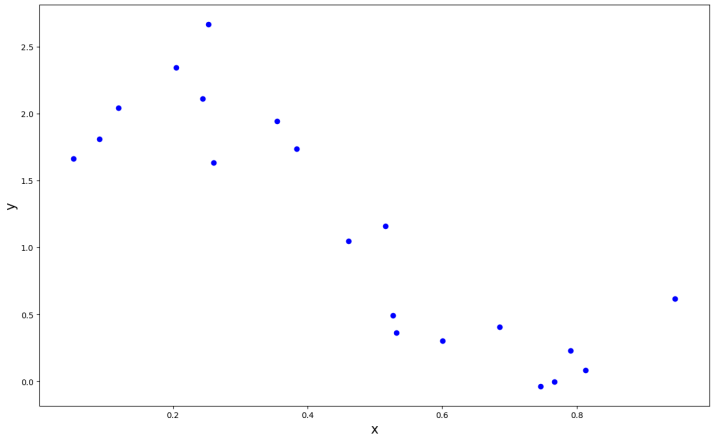
Plot 3 – $n = 10$ with 5 1s and 5 0s

What we see here is that the likelihood of a specific value is lesser and lesser for values further away from the true proportion for a higher number of samples. We also see that for the same true proportion, as we increase the samples that the distribution gets "tighter" in the sense that it bunches up around the true proportion and peaks there. For $n = 5$, the distribution follows the same pattern, but is very wide compared to $n = 10$ and definitely compared to $n = 100$. This shows that with a lower sample, there's more variability in our results.

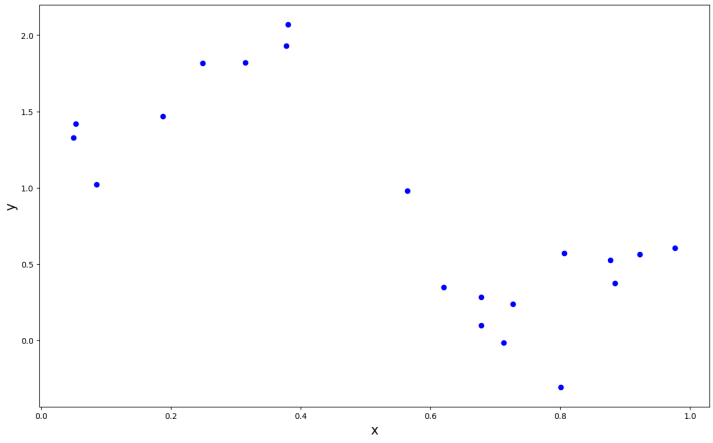
4 Implementation: Polynomial Regression

4.a Visualization

The plots for part (a) are shown below:



Plot 1 – training data



Plot 2 – testing data

4.b Linear Regression

4.c

4.d

When looking at the outputs that I got, for a learning rate of 0.01, the closed-form solution was only 0.00000003 off of the value we got for the learning rate, and the cost and time were the least out of all of the iterations. Thus, we realize that this one is the quickest.

4.e

The closed form solution I got was [2.44640709 -2.81635359]. The numbers match the gradient descent outputs as well., but it runs much quicker than the GD algorithm does.

4.f

It took 0.041s.

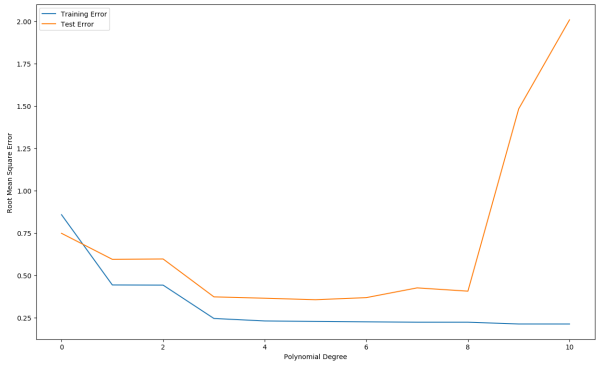
4.g

4.h

RMSE is preferable over J(θ) because it provides us with a quantity that's standardized and thus usable for us in our interpretations.

4.i

The graph is included below:



I would estimate that degree 5 would best fit our data because as the degrees increase, we see overfitting (since the test error goes up while training error stays down) and before that there are slight signs of underfitting because the test error behaves exactly the way that the training error does