CM-146 Problem Set 1

Reinaldo Daniswara
604840665

1. Perceptron
   a. And

x	y	AND
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

Based on the table, we know that
$-w0 - w1 + b < 0$
$-w0 + w1 + b < 0$
$w0 - w1 + b < 0$
$w0 + w1 + b > 0$
Since $w0 + w1 + b > 0$ is a positive value, than one of the possible solution is $w0 = 1$, $w1 = 1$ and $b = 1$. This solution is not unique as well, another possible solution is $w0 = 2$, $w1 = 3$, $w2 = 4$

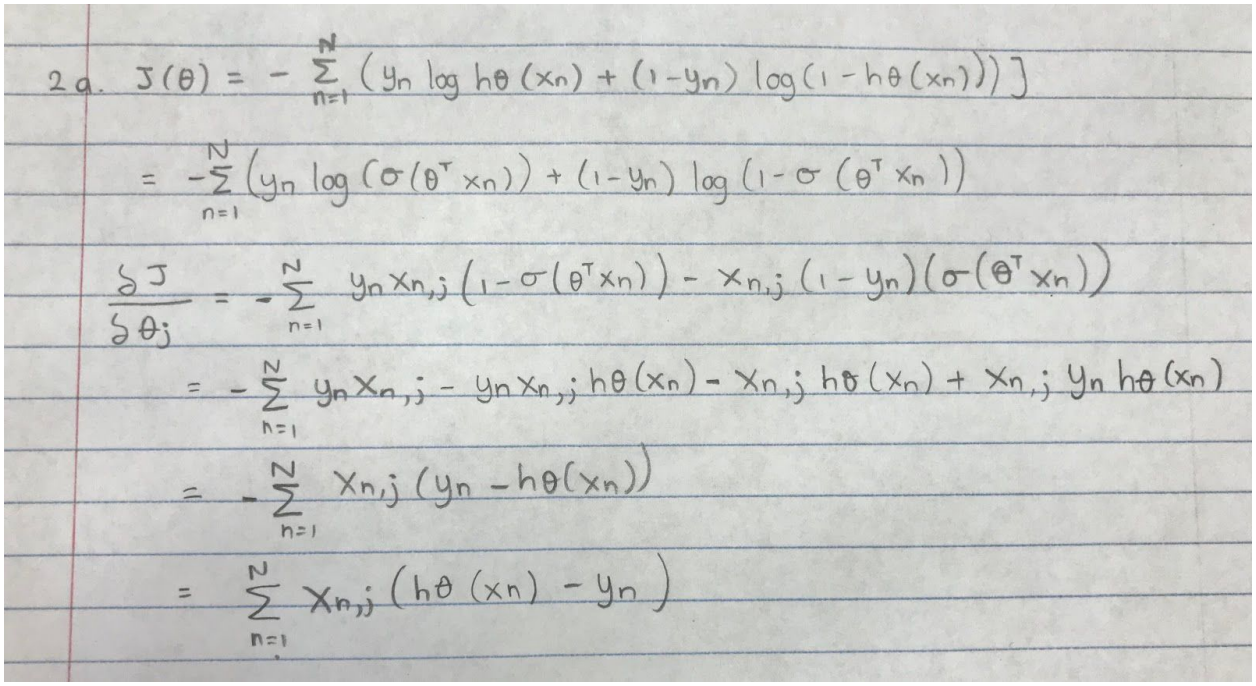
   b. Xor

x	y	AND
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

$-w0 - w1 + b < 0$
$-w0 + w1 + b > 0$
$w0 - w1 + b > 0$
$w0 + w1 + b < 0$

Since there is no pattern to make this data linearly separable, there is no perceptron exist.

2. Logistic Regression



3. Locally Weighted Linear Regression

3a. $\frac{\partial J}{\partial \theta_0} = 2 \sum_{n=1}^{N} w_n (\theta_0 + \theta_1 x_{n,1} - y_n)$

$\frac{\partial J}{\partial \theta_1} = 2 \sum_{n=1}^{N} w_n (\theta_0 + \theta_1 x_{n,1} - y_n) \cdot (x_{n,1})$

b $2 \sum_{n=1}^{N} w_n (\theta_0 + \theta_1 x_{n,1} - y_n) = 0$

$\sum_{n=1}^{N} w_n \theta_0 + \sum_{n=1}^{N} w_n \theta_1 x_{n,1} - \sum_{n=1}^{N} w_n y_n = 0$

$\theta_0 \sum_{n=1}^{N} w_n = \sum_{n=1}^{N} w_n y_n - \sum_{n=1}^{N} w_n \theta_1 x_{n,1}$

$$\theta_0 = \frac{\sum_{n=1}^{N} w_n y_n - \sum_{n=1}^{N} w_n \theta_1 x_{n,1}}{\sum w_n} \quad \text{---} \ (1)$$

* $\frac{\partial J}{\partial \theta_1} = 0$

$2 \sum_{n=1}^{N} w_n x_{n,1} (\theta_0 + \theta_1 x_{n,1} - y_n) = 0$

$\sum_{n=1}^{N} w_n x_{n,1} \theta_0 + \sum_{n=1}^{N} w_n (x_{n,1})^2 \theta_1 - \sum_{n=1}^{N} w_n x_{n,1} y_n = 0$

$\theta_1 \sum_{n=1}^{N} w_n (x_{n,1})^2 = \sum_{n=1}^{N} w_n x_{n,1} y_n - \sum_{n=1}^{N} w_n x_{n,1} \theta_0$

$\theta_1 \sum_{n=1}^{N} w_n (x_{n,1})^2 = \sum_{n=1}^{N} w_n x_{n,1} y_n - \sum_{n=1}^{N} w_n x_{n,1} \left(\frac{\sum_{n=1}^{N} w_n y_n - \sum_{n=1}^{N} w_n \theta_1 x_{n,1}}{\sum_{n=1}^{N} w_n} \right)$

$\times (\sum w_n)$

$\theta_1 \left(\sum_{n=1}^{N} w_n \sum_{n=1}^{N} w_n (x_{n,1})^2 \right) = \left(\sum_{n=1}^{N} w_n x_{n,1} y_n \right) - \left(\sum_{n=1}^{N} w_n x_{n,1} \right) \left(\sum_{n=1}^{N} w_n y_n \right) + \left(\sum_{n=1}^{N} w_n x_{n,1} \right) \theta_1 \left(\sum_{n=1}^{N} w_n x_{n,1} \right)$

$$\theta_1 = \frac{\left(\sum_{n=1}^{N} w_n x_{n,1} y_n \sum_{n=1}^{N} w_n \right) - \left(\sum_{n=1}^{N} w_n x_{n,1} \right) \left(\sum_{n=1}^{N} w_n y_n \right)}{\left(\sum_{n=1}^{N} w_n \sum_{n=1}^{N} w_n (x_{n,1})^2 \right) - \left(\sum_{n=1}^{N} w_n x_{n,1} \right)^2}$$

4. Understanding Linear Separability

4a. If D is linearly separable, there is an optimal solution to the linear program with $\delta = 0$

→ If D is linearly separable, then there exists a hyperplane $\vec{v}^T\vec{x} + p$ s.t.

$$\min_{\substack{(\vec{x},y)\in D \\ y=1}} (\vec{v}^T\vec{x} + p) \geq 0 > \max_{\substack{(\vec{x},y)\in D \\ y=-1}} (\vec{v}^T\vec{x} + p)$$

→ Now, let assume $\vec{x}_i$ to be the positive sample that's closest to hyperplane $\vec{v}^T\vec{x} + p$, s.t.

$$p_i = \min_{\substack{(\vec{x},y)\in D \\ y=1}} (\vec{v}^T\vec{x} + p) = (\vec{v}^T\vec{x}_i + p)$$

And, $\vec{x}_j$ be the negative sample.

$$p_j = \max_{\substack{(\vec{x},y)\in D \\ y=1}} (\vec{v}^T\vec{x} + p) = (\vec{v}^T\vec{x}_j + p)$$

→ Since $p_i \geq 0 > p_j$, we could shift the hyperplane $p_i - \alpha \geq 0 > p_j - \alpha$ where $\alpha \geq 0$

→ For some α, now we have a hyperplane $\vec{v}^T\vec{x} + p - \alpha = 0$ where it separates D

$$\frac{|\vec{v}^T\vec{x}_i + p - \alpha|}{\|\vec{v}\|} = \frac{|\vec{v}^T\vec{x}_j + p - \alpha|}{\|\vec{v}\|}$$

$\therefore \alpha = \frac{p_i + p_j}{2}$ → this indicates $\vec{v}^T\vec{x} + p - \alpha = 0$ separates

→ Now, normalize the hyperplane,

$$\min_{\substack{(\vec{x},y)\in D \\ y=1}} (\vec{v}^T\vec{x} + p - \alpha) = \frac{p_i - p_j}{2}$$

$$\max_{\substack{(\vec{x},y)\in D \\ y=1}} (\vec{v}^T\vec{x} + p - \alpha) = \frac{p_j - p_i}{2}$$

$$\therefore y(\vec{v}^T\vec{x} + p - \alpha) \geq \frac{p_i - p_j}{2} \quad \forall (\vec{x},y) \in D$$

→ Now, let set $\vec{w} = \frac{\vec{v}}{\alpha}$ $\theta = \frac{p - \alpha}{\alpha}$ $\delta = 0$ and $p_i > p_j$,

we know that $y(\vec{w}^T\vec{x} + \theta) \geq 1 - \delta$, $\forall (\vec{x},y) \in D$

4b If there exist a hyperplane $\vec{w}^T\vec{x} + \theta$ such that

$$y(\vec{w}^T\vec{x} + \theta) \geq 1 \quad , \quad \forall (\vec{x}, y) \in D$$

It's obvious that

$$(\vec{w}^T\vec{x} + \theta) \geq 1 \geq 0 \qquad \forall (\vec{x}, y) \in D \ , \ y = 1$$

$$(\vec{w}^T\vec{x} + \theta) \leq -1 \leq 0 \qquad \forall (\vec{x}, y) \in D \ , \ y = -1$$

Thus, we can conclude that $\delta = 0$ is an optimal solution, then D is linearly separable.

4c We will see when $\delta > 0$,

If $1 - \delta > 0 \rightarrow$ it's going to be similar, where we can apply the same argument with $\delta = 0$, and we can see the data is linearly separable.

If $\delta \geq 1$, we won't be able to determine clearly whether or not the data is separable. But,

If minimal $\delta \geq 1$, then it's not separable.

4d. The optimal solution is we set $\vec{w}$, θ, and δ to be 0.
The reason is because 0 won't give us a hyperplane, and we have already proven before that $\delta = 0$ gave us the optimal solution.

4e. Since there are only 2 data set, it could be easy to make it linearly separable by making $\delta = 0$. Thus, to find $\vec{w}$ & θ,

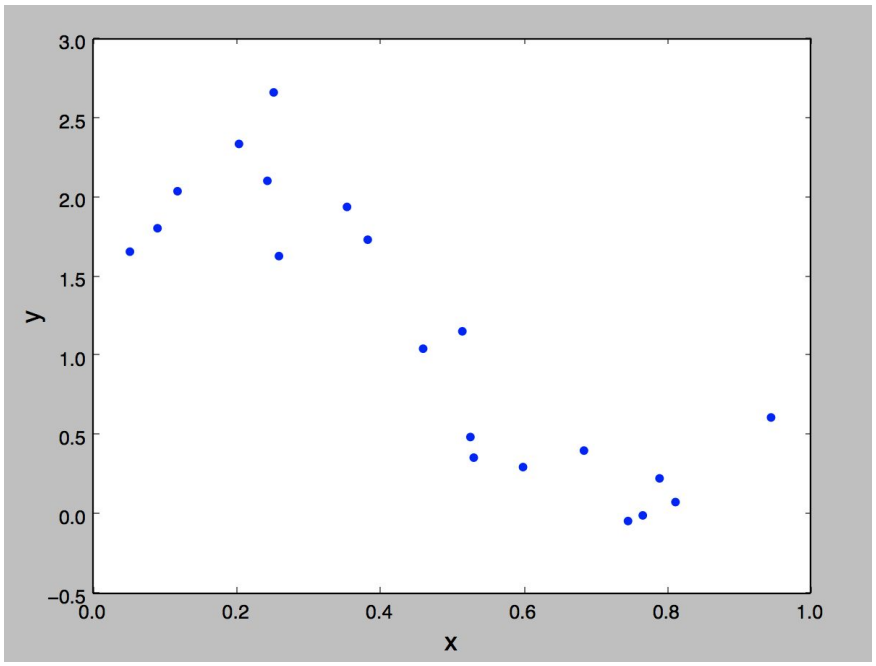
$$w_1 + w_2 \ldots + w_n + \theta \geq 1$$

$$-(-w_1 - w_2 \ldots - w_n + \theta) \geq 1$$

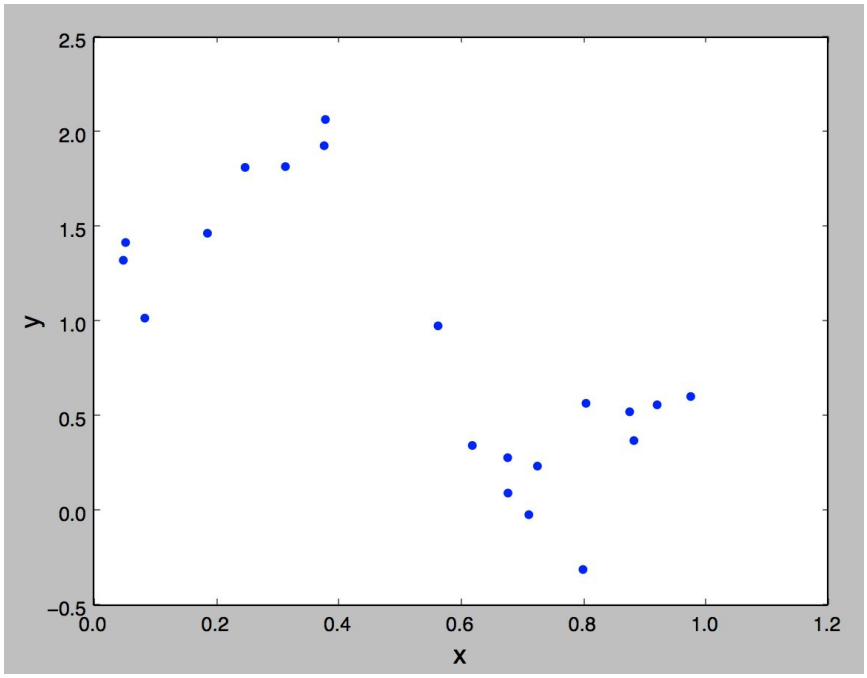
$$w_1 + w_2 + \ldots + w_n \geq 1 + |\theta|$$

Thus, the optimal solution will be $\delta = 0$ and $w_1 + w_2 + \ldots + w_n \geq 1 + |\theta|$

5. Implementation: Polynomial Regression

a. 

Training Data



Test Data

Based on the plot above, for training data, we know that y is decreasing as x is increasing. However, there is some exceptions when x is around 0 to 0.3 as y also increases when x is increases, but the rest of them is quite linearly separable, with a bit of noise. Thus, this problem can be tackled using a linear regression method. However, for

the test data, there is no actual pattern to separate data linearly while we can see it's quite random. Thus, it is not a good idea to use linear regression method.

b. Yes, it is showing 40.234, and I change the code
c. Change the code
d.

Step Size	Coefficients	Iterations	Cost	ETA
10^{-4}	[1.91573585 -1.74358989]	10000	5.49356558874	0.322247
10^{-3}	[2.4463815 -2.81630184]	10000	3.91257640947	0.296573
10^{-2}	[2.44640699 -2.81635338]	1490	3.91257640579	0.039341
0.0407	[2.44640706 -2.81635351]	382	3.91257640579	0.01004

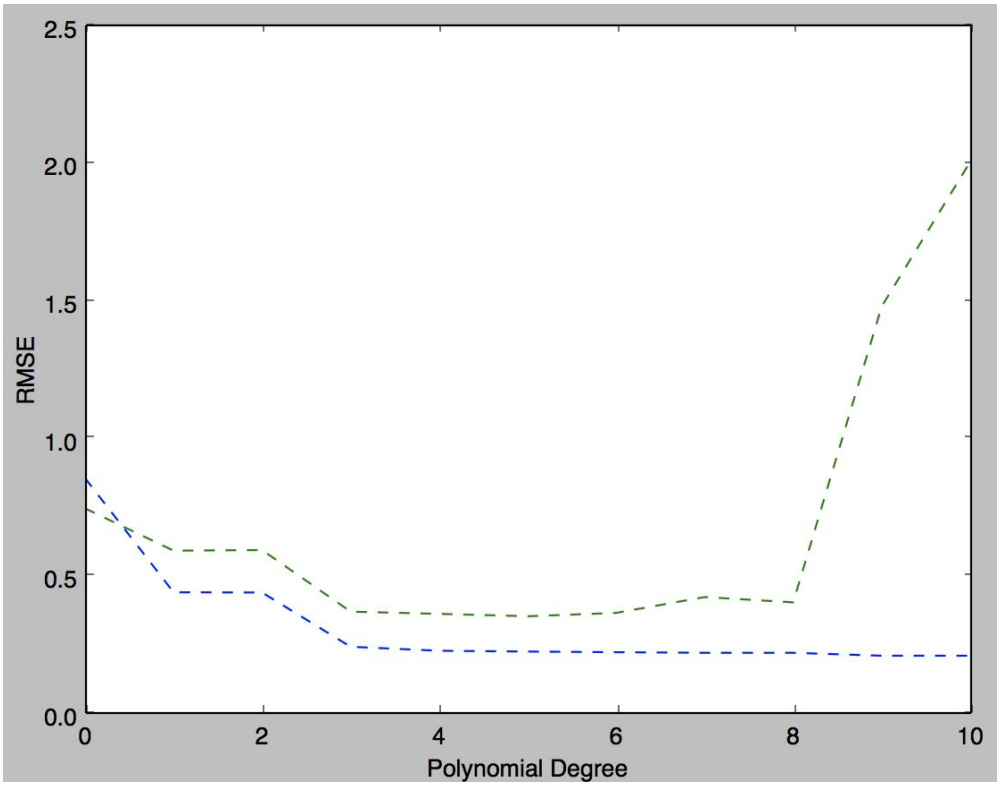
From the data above, we know that for a step size that able to converge, it's going to be have a similar number of coefficients, and cost (in this case is 0.0407, 10^{-2}, 10^{-3}). Based on the data above again, we will be able to determine that the smaller the steps, the time to converge will be slower. For example, a larger steps like 0.0407 will have 382 iterations to converge, then linearly increases as the number of steps getting smaller and smaller. This number of iterations also reflected correctly by the time to converge that implement to improve the accuracy of timing comparison.

e. "In mathematics, an expression is said to be a closed-form expression if it can be expressed analytically in terms of a finite number of certain "well-known" functions"[1]. The closed form solution cost is 3.91257640579 with the coefficients of [2.44640709 -2.81635359], which is fairly similar with the GD for step size 0.0407, 10^{-2}, 10^{-3}. However, timewise, the PolynomialRegression.fit is faster than GD. Since there is no iterations in closed-form solution, I count the time using time() method and compare it with the time that I calculate for GD. As a result, for closed-form solution, it ran for 0.0056, which is faster than GD with step size 0.0407 (which was faster in GD method, eta 0.01004). However, closed form solution will not always faster, for example, when the data is getting larger and larger, closed form solution will be slower that GD method.

f. In here, I limited the number of iterations to be 10000, and the gd algorithm will converge on 0.382107 with the cost of 3.91257642432. Moreover, the coefficient is [2.44634965 -2.81623746]

g. Change the code

h. In the actual implementation, RMS error means taking the square root of the average value of the actual value minus the predicted value. While on the other hand, $J(\theta)$ only measures a magnitude of a data that does not match. Thus, RMSE is a better fit to measure the overfitting problem because it will not only rely of the data that does not match only, but comparing the actual value with the predicted value as well. Thus, it is clear that RMS is a better approach to use.

i. 

Blue line : Training Error
Green Line : Test Error
The best model to fit the data is when the training error and test error is stabile, not indicating a lot of increasing or decreasing, which in polynomial degree around 4 until 5. There is also overfitting in this polynomial degree as we see that at degree 8, training error is still decreasing, but all the sudden, test error is increasing.

Reference

[1]
https://stats.stackexchange.com/questions/70848/what-does-a-closed-form-solution-mean