perceptron

① a. OR

given weights w_1 & w_2, bias b & x_1 & x_2

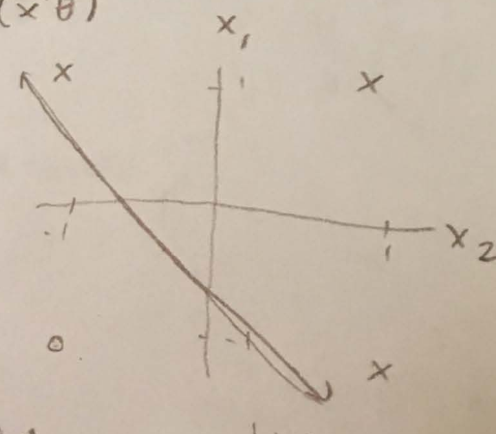
two vectors $\vec{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$ $\vec{\theta} = \begin{pmatrix} b \\ \theta_1 \\ \theta_2 \end{pmatrix}$ where x_0 is always 1

when $\vec{\theta}^T = (1, 1, 1)$

then

$y = \mathrm{Sign}(\vec{x}^T \vec{\theta})$

x_0	x_1	x_2	$\underbrace{x_0 b + \theta_1 x_1 + \theta_2 x_2}_{\vec{x}^T \cdot \vec{\theta}}$	out
1	-1	-1	-1	-1
1	-1	1	1	1
1	1	-1	1	1
1	1	1	3	1

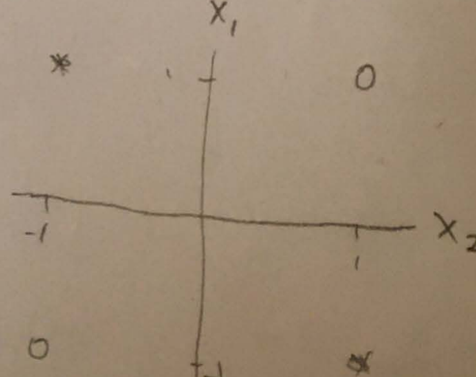


$b + w_1 x_1 + w_2 x_2$

A hyperplane which classifies the OR function of this data does exist. The perceptron is however not unique.
When $\vec{\theta}^T = \{2, 2, 3\}$, then

x_0	x_1	x_2	$x_0 b + x_1\theta_1 + x_2\theta_2$	y
1	-1	-1	-3	-1
1	1	-1	1	1
1	-1	1	1	1
1	1	1	7	1

→ this hyperplane classifies the data as well, the perceptron is not unique



① b. XOR

x_0	x_1	x_2	x	y out
1	-1	-1		1
1	1	-1		-1
1	-1	1		-1
1	1	1		1

↑ no perceptron exists for this data

$$b - \theta_1 x_1 - \theta_2 x_2 < 0$$
$$b + \theta_1 x_1 - \theta_2 x_2 > 0$$
$$b + \theta_1 x_1 + \theta_2 x_2 > 0$$
$$b - \theta_1 x_1 - \theta_2 x_2 < 0$$

mathematical contradiction

no perceptron exists for XOR

② $J(\theta) = -\sum_{n=1}^{N} \left[y_n \log h_\theta(x_n) + (1-y_n)\log(1-h_\theta(x_n)) \right]$

(2a) $\frac{\partial J}{\partial \theta_j} = -\sum_{n=1}^{N} y_n \cdot \frac{1}{h_\theta(x_n)} \cdot \frac{\partial h_\theta(x_n)}{\partial \theta_j} + (1-y_n) \cdot \frac{1}{1-h_\theta(x_n)} \cdot \frac{-\partial h_\theta(x_n)}{\partial \theta_j}$

where $\frac{\partial h_\theta(x)}{\partial \theta} = \frac{\partial}{\partial \theta} \cdot \frac{1}{(1+e^{-\theta^T x_n})} = -1 \cdot -1 \cdot \frac{e^{-\theta^T x_n} \cdot \vec{x}_{j,n}}{(1+e^{-\theta^T x_n})^2} = (1-h_\theta(x_n))(h_\theta(x_n))\vec{x}_{j,n}$

then $\frac{\partial J}{\partial \theta_j} = -\sum_{n=1}^{N} y_n \cdot \frac{1}{h_\theta(x_n)} \cdot h_\theta(x_n)(1-h_\theta(x_n))x_{j,n} + (1-y_n) \cdot \frac{1}{1-h_\theta(x_n)} \cdot h_\theta(x_n)(1-h_\theta(x_n))x$

$= -\sum_{n=1}^{N} \left[y_n(1-h_\theta(x_n)) + (1-y_n)h_\theta(x_n) \right] \vec{x}_{j,n}$

$= -\sum_{n=1}^{N} \left[y_n - h_\theta(x_n) \right] x_{j,n}$

(2b) $\frac{\partial^2 J}{\partial \theta_j \partial \theta_k} = -\sum_{n=1}^{N} \left[0 - \frac{\partial h_\theta(x_n)}{\partial \theta_k} \right] \vec{x}_{j,n}$

$= -\sum_{n=1}^{N} \left[(1-h_\theta(x_n))(h_\theta(x_n)) \cdot \vec{x}_{k,n} \cdot \vec{x}_{j,n} \right.$

therefore

$H_{jk} = \sum_{n=1}^{N} \left[h_\theta(x_n)(1-h_\theta(x_n)) \, x_n x_n^T \right]$

(2c) J is convex if H is positive semi def. We know $H_{jk} = -\sum_{n=1}^{N} h_\theta(x_n)(1-h_\theta(x_n)) \vec{x}_{j,n} \vec{x}_{k,n}$

Let $S = -\sum_{n=1}^{N} h_\theta(x_n)(1-h_\theta(x_n))$

then

$$H = S \begin{bmatrix} \vec{x}_{n,1}^2 & \vec{x}_{n,\cdot} & \cdots & x_{n,1}x_{n,d} \\ \vdots & \ddots & & \\ x_{n,2}x_{n,1} & & \ddots & \\ & & & x_{n,d}^2 \end{bmatrix}_{d \times d}$$

because $x_n^T x_n = x_n^2$

→ all the diagonal values of H are positive because $\vec{x}_n^T \vec{x}_n \to x^2$ is always positive as is $z^T z$ because $z^T z \to z^2$

Finally S must lie between 0 and 1 so is also positive

thus H is positive semi-definite and J is convex!

3a $P(x) = \theta^{x} \cdot (1-\theta)^{1-x}$

$L(\theta) = P(x_1 \ldots x_n; \theta) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} = \sum_{i=1}^{n} x_i(\log\theta) + (1-x_i)\log(1-\theta)$

The likelihood does not depend on the order of observation, $x_1 \ldots x_n$ are assumed to be i.i.d.

3b

$$L'(\theta) = \frac{\sum_{i=1}^{n} x_i}{\theta} + \frac{-\sum_{i=1}^{n}(1-x_i)}{(1-\theta)} =$$

$$L''(\theta) = \frac{-\sum_{i=1}^{n} x_i}{\theta^2} + \frac{\sum_{i=1}^{n}(1-x_i)}{(1-\theta)^2}$$

Closed form MLE estimate

$$0 = \frac{\sum_{i=1}^{n} x_i}{\theta} - \frac{\sum_{i=1}^{n}(1-x_i)}{1-\theta} \Rightarrow \frac{\sum_{i=1}^{n}(1-x_i)}{1-\theta} = \frac{\sum_{i=1}^{n}(x_i)}{\theta}$$

$$\Rightarrow \sum_{i=1}^{n}(1-x_i)\theta = \sum_{i=1}^{n}(x_i) - \sum_{i=1}^{n}(x_i)\theta$$

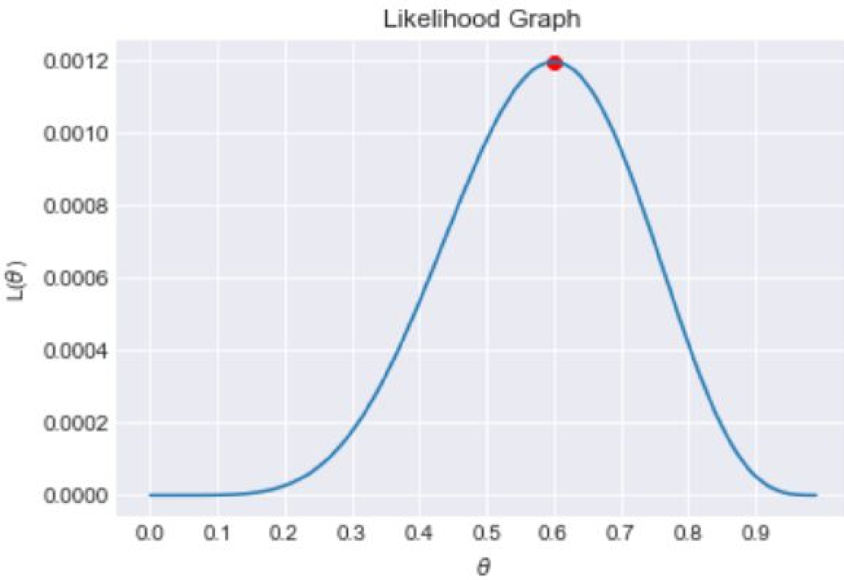
$$\sum_{i=1}^{n}(1-x_i)\theta + \sum_{i=1}^{n}(x_i)\theta = \sum_{i=1}^{n}(x_i)$$

$$\theta\left(\sum_{i=1}^{n}(1-x_i) + \sum_{i=1}^{n} x_i\right) = \sum_{i=1}^{n}(x_i)$$

then

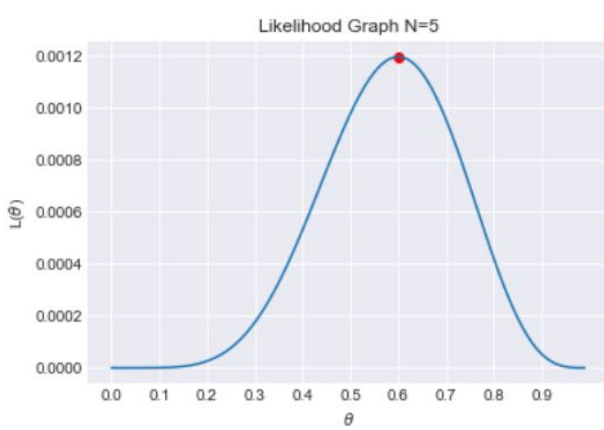
$$\boxed{\hat{\theta}_{MLE} = \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n}(1-x_i) + \sum_{i=1}^{n} x_i}}$$

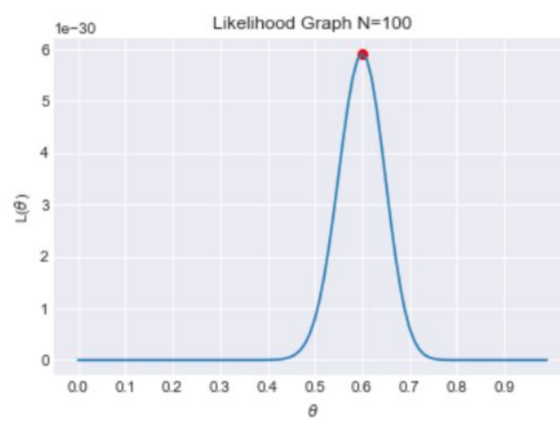
3c.



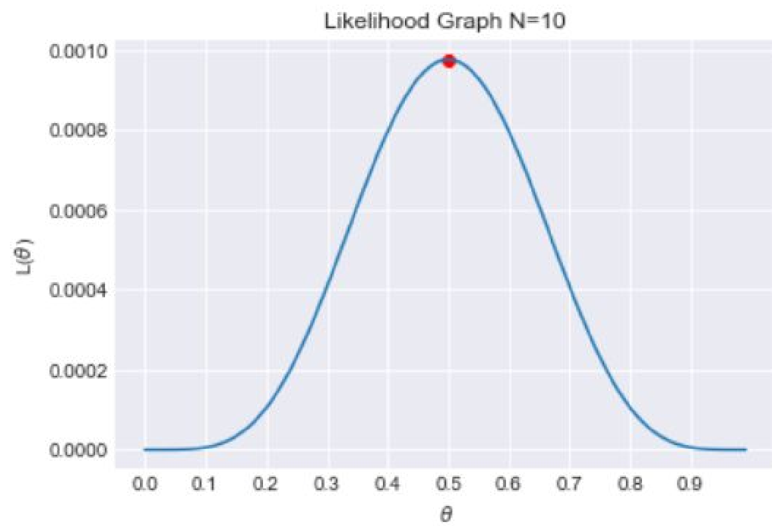
One the graph, the maximum lies around theta hat = 0.595 which agrees with the close form solution estimate of theta of 0.6.

3d.



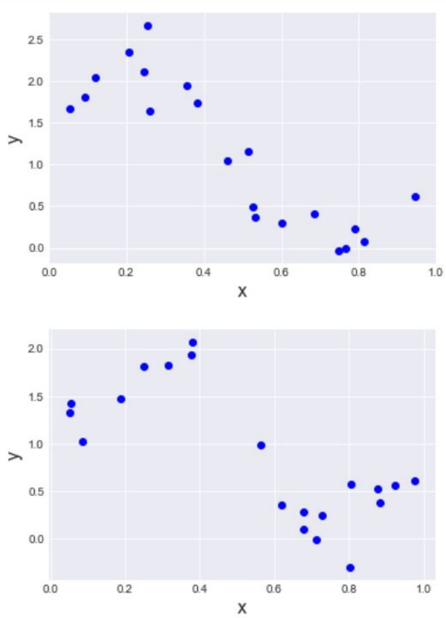


Both of these data sets have the same estimate for theta because they have the same ratio of 0's to 1's in each data set. In both cases, graphs' theta estimate is around 0.59 and the closed form estimate is 0.6.



This graph’s theta estimate agrees with our closed form solution.

4a)



In general, as the value input (x) increases in value, the output decreases--indicating it follows a linear trend. Therefore, the values Linear regression is effective for predicting this data because both the training and testing datasets follow a trend predictable by a function.

4b) In code
4c) In code

4d)

Trial	Learning Rate	Time (seconds)	Iterations	θ_1	θ_2	Objective Function (cost)
1	0.0001	0.3983573913574219	10000	2.27044798	-2.46064834	4.0863970367957645
2	0.001	0.28759121894836426	7021	2.4464068	-2.816353	3.9125764057919463
3	0.01	0.032036781311035156	765	2.44640703	-2.81635346	3.912576405791487
4	0.0407	0.4293661117553711	10000	-9.40470931e+18	-4.65229095e+18	2.710916520014386e+39

Trial 1's low learning rate causes the algorithm to not converge and terminate early. Trials 2 and 3 both converge. Trial 3 has a slightly lower cost compared to Trial 2 (a difference of 4.5918824e-13) however, Trial 3 was 0.25555443763 seconds faster than Trial 2.. Trial 4's step size was too high, it overstepped the minimum and as a result the parameters did not converge and the algorithm took longest, had the highest cost, and maxed out at 10000 iterations.

4e) With $J = 3.9125764057914636$ and Time: 0.003958225250244141 seconds with parameters: This calculation was faster than gradient descent. It also had a lower cost but this value is marginal in size. Gradient descent is preferable to the closed form solution because the computational cost of calculating the inverse matrix increases tremendously in large data sets. Our data set is small in this case so the closed form solution is faster and less expensive than gradient descent. The parameters are practically the same as the gradient descent calculation.

θ_1	θ_2
2.44640709	-2.81635359

4f) The algorithm converged in 0.07982325553894043 seconds with 1719 iterations. Here, J = 3.9125764057922705 and Coefficients = [2.44640672 -2.81635282]. In this process, the step size decreases with each iteration allowing the algorithm to converge faster but still ensure correct convergence values.

4h)
Polynomial Degree 0
Test Error: 0.5935949636028289
Train Error 0.44229946901344247

Polynomial Degree 1
Test Error: 0.5935949636028289
Train Error 0.44229946901344247

Polynomial Degree 2
Test Error: 0.5957110445316882
Train Error 0.441314965682919

Polynomial Degree 3
Test Error: 0.37194297617203903
Train Error 0.24426921898418788

Polynomial Degree 4
Test Error: 0.36393172002175844
Train Error 0.22968276125805662

Polynomial Degree 5

Test Error: 0.3551377428803351
Train Error 0.22681133051783178

Polynomial Degree 6
Test Error: 0.36745016928700935
Train Error 0.22445294828005166

Polynomial Degree 7
Test Error: 0.4250083097538674
Train Error 0.22228193952069397

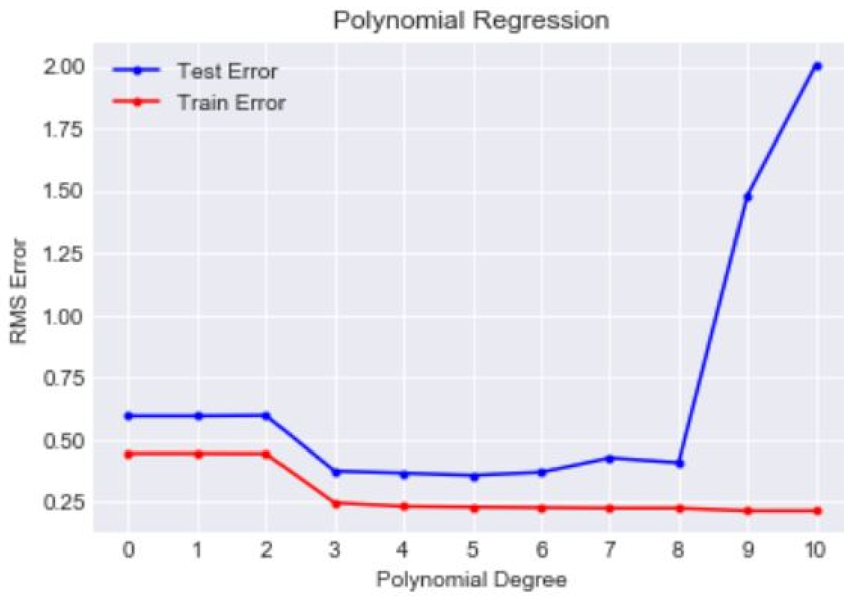
Polynomial Degree 8
Test Error: 0.40566871142272753
Train Error 0.222260574960435

Polynomial Degree 9
Test Error: 1.4824539726638684
Train Error 0.21177880107774616

Polynomial Degree 10
Test Error: 2.0078547606549306
Train Error 0.2116891695190847

We prefer RMSE in this scenario because RMSE provides a normalized measurement of error while *J* ‘s measurement increases with each iteration. RMSE is not susceptible to bias resulting from the size of the data set.

4i)



Degree 5 polynomial appears to best fit the data because it has the lowest testing and training error out of all the degrees. I do see over fitting which is most prominent with a polynomial degree greater than 8. At degree 9 and 10 specifically, the model is being fit almost perfectly to the training set but cannot generalize the pattern. As a result the training error falls while the testing error shoots up.