CS 146 PS2

1.

a. OR- θ=(1,1,1) , θ=(1,0.7,0.8)

b. XOR - No valid perceptron, not linearly separable

2.

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

$= -\sum_{n=1}^{N} y_n \frac{1}{h_\theta(x_n)} h_\theta(x_n)(1 - h_\theta(x_n))x_j + (1 - y_n)\frac{1}{1-h_\theta(x_n)}(1 - h_\theta(x_n))(-h_\theta(x_n))x_j$

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

$= -\sum_{n=1}^{N} (y_n - h_\theta(x_n))x_{nj}$

b. $\frac{\partial^2 J}{\partial \theta_j \partial \theta_k} = -\sum_{n=1}^{N} \frac{\partial (y_n - h_\theta(x_n))x_{nj}}{\partial \theta_k}$

$= -\sum_{n=1}^{N} - h_\theta(x_n)(1 - h_\theta(x_n))x_k x_{nj}$

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

H=$\frac{\partial^2 J}{\partial \theta_j \partial \theta_k} = \sum_{n=1}^{N} h_\theta(x_n)(1 - h_\theta(x_n))x_n x_n^T$

c. $z^T H z$>0 determines if PSD

In H=$\frac{\partial^2 J}{\partial \theta_j \partial \theta_k} = \sum_{n=1}^{N} h_\theta(x_n)(1 - h_\theta(x_n))x_n x_n^T$, $x_n x_n^T$ means squaring, meaning it's always positive. Thus H is always positive.

=> $z^T H z$ can be written in only positive way

Therefore J is convex.

3.

a. $L(\theta) = P(X1, \ldots, Xn; \theta)$

$$= \prod_{i=0}^{n} P(x_i) \;=\; \prod_{i=0}^{n} \theta^{x_i}(1-\theta)^{1-X_i}$$

b. $l(L(\theta))=\log(\prod_{i=0}^{n} \theta^{x_i}(1-\theta)^{1-X_i})$

$$\frac{d\,log(L(\theta))}{d(\theta)} = \frac{d}{d(\theta)} \prod_{i=0}^{n} \log(\theta^{x_i}) + \log(1-\theta)^{1-X_i}$$

$$\frac{d\,log(L(\theta))}{d(\theta)} = \prod_{i=0}^{n} \frac{x_i}{\theta} - \frac{1-x_i}{1-\theta}$$

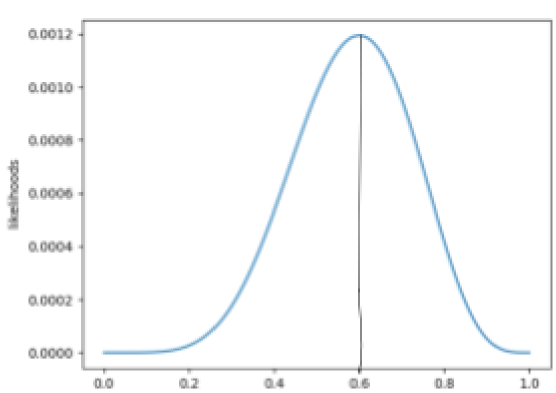
$$\frac{d^2 log(L(\theta))}{d(\theta)^2} = \prod_{i=0}^{n} \left(-\frac{x_i}{\theta^2} + \frac{1-x_i}{(1-\theta)^2}\right)$$

$$\prod_{i=0}^{n} \left(\frac{x_i}{\theta} - \frac{1-x_i}{1-\theta}\right) = 0$$

$$\Rightarrow \frac{x_i}{\theta} = \frac{1-x_i}{1-\theta}$$

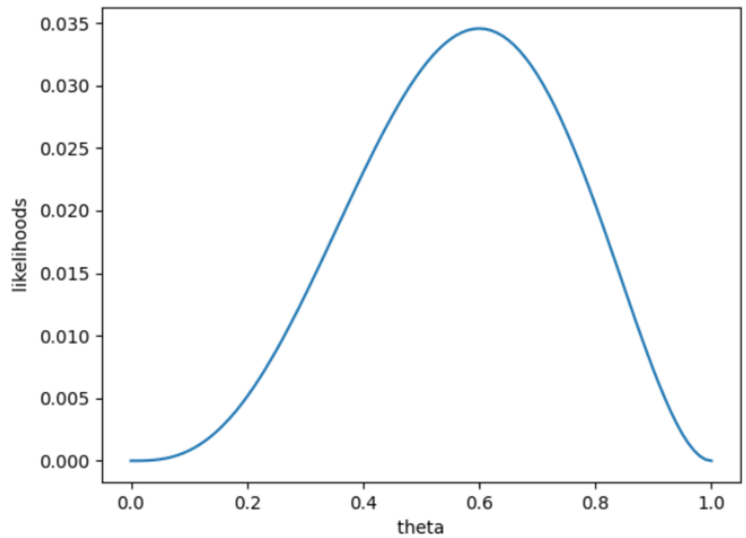
$\theta^{MLE} = x_i \frac{1}{x_i + (1-x_i)}$ (the # of times theta happens over all n)

c.

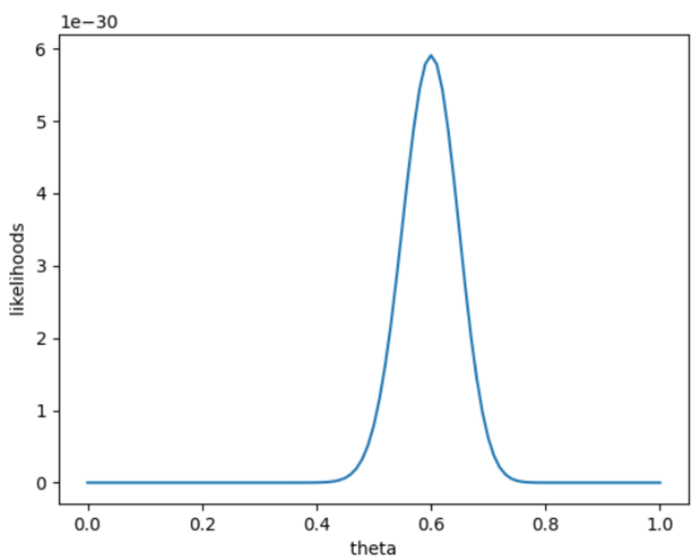


It does agree with the closed form , 6/10= 0.6

d. 3/2



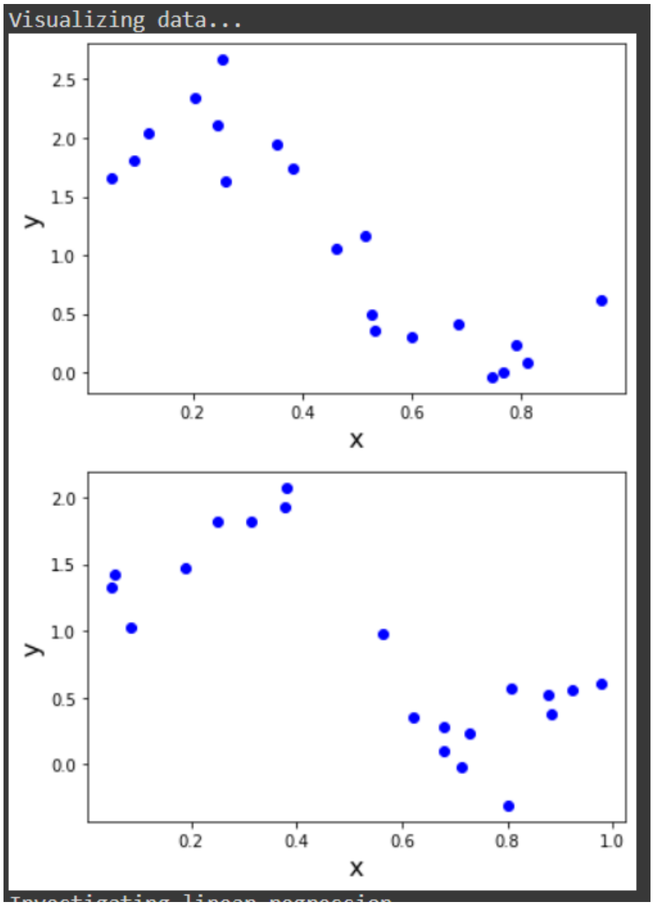
60/40



The MLEs are the same as they are the same ratio of ⅗ (0.6). As n increases, the standard deviation decreases, creating a narrower graph at 0.6.

4.

a.



There seems to be a possible negative linear relationship- linear regression could work, but polynomial seems superior.

b.

```
# part b: modify to create matrix for simple linear model

X = np.append(np.ones([n,1]), X, 1)
```

c.

```
y = None
y = np.dot(X, self.coef_)
```

d.

Eta	Coeff	# iter	Final
10^{-4}	[2.27044798 -2.46064834]	10000	4.0864
10^{-3}	[2.4464068 -2.816353]	7020	3.9126
10^{-2}	[2.44640703 -2.81635346]	764	3.9126

With larger steps, the number of iterations decreases- however, too large of a step size (0.1) causes an error as the minimum is passed and no convergence. The coefficients are fairly close.

e.

```
coefficient:[ 2.44640709 -2.81635359]   run time:0.018434762954711914
```

Closed form solution= [2.446, -2.816]

It is much faster compared to GD as shown by runtime- this makes sense since no iteration to convergence.

f.

```
number of iterations: 764
Coefficient: [ 2.44640703 -2.81635346]
```

It takes 764 iterations to converge

g.

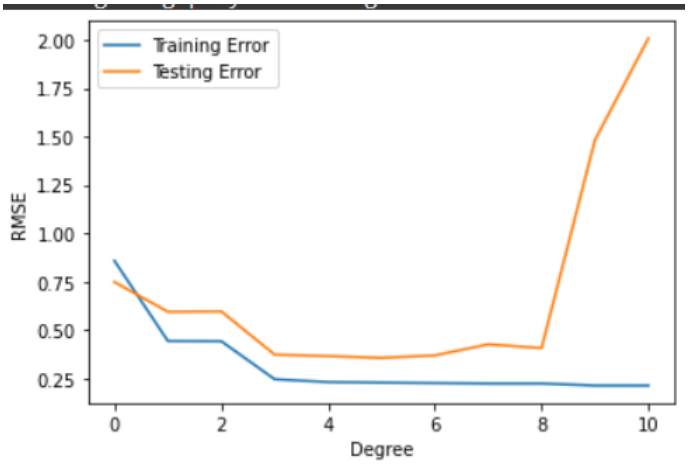
```
# part g: modify to create matrix

Phi = np.ones([n,1])
m = self.m_
for i in range(1, m + 1):
  Phi = np.append(Phi, X ** i, 1)
```

h.

We prefer RMSE as a metric over J because we remove the squaring that we did previously for J, which makes the result more comparable to the data.

```
# part h: compute RMSE
n, d = X.shape
error = np.sqrt(self.cost(X,y)/n)
```

i.



The degree polynomial that best fits the data is 4. There seems to be underfitting when m<3, and clearly overfitting as m>8.