Dhruv Chakraborty
CS M146 W2020

HW #2

1)

(a) A set of weights for a perceptron modelling the OR function are $\theta = (\theta_1, \theta_2, b) = (1, 1, 1)$

~~Proceed to the table below~~

We can see this from the truth table below:

x_1	x_2	$\theta_1 x_1 + \theta_2 x_2 + b$	Output
1	1	3	T
1	-1	1	T
-1	1	1	T
-1	-1	-1	F

Another set of weights that model an OR gate on a diff. hyperplane are $(\theta_1, \theta_2, b) = (1, 1, 2)$ showing such a design is not unique.

(b) ~~For an XOR function~~ If we were to model an XOR gate with a perceptron $y = \theta^T x + b$, we would need inputs $(-1,-1)$ and $(1,1)$ to evaluate to false, but this means $-\theta_1 - \theta_2 < -b$ and

$\theta_1 + \theta_2 < -b$ which are impossible.

Since the XOR problem is not linearly separable, we cannot design a perceptron to model it.

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

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

$$= -\sum_{n=1}^{N} \cancel{\frac{y_n}{\sigma(\theta^T x_n)}} \left(\cancel{\sigma(\theta^T x_n)}(1-\sigma(\theta^T x_n)) x_{nj}\right.$$

$$- \frac{(1-y_n)}{\cancel{1-\sigma(\theta^T x_n)}} \sigma(\theta^T x_n)\cancel{(1-\sigma(\theta^T x_n))} x_{nj}$$

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

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

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

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

$$= \sum_{n=1}^{N} \frac{\partial}{\partial \theta_j}\left[x_{nk}(h_\theta(x_n) - y_n)\right]$$

$$= \sum_{n=1}^{N} \sigma(\theta^T x_n)(1-\sigma(\theta^T x_n)) x_{nj} x_{nk}$$

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

$$H = \begin{pmatrix} \frac{\partial^2 J}{\partial \theta_1^2} & \cdots & \frac{\partial^2 J}{\partial \theta_1 \partial \theta_d} \\ \vdots & & \vdots \\ \partial^2 J/\partial \theta_d \partial \theta_1 & \cdots & \partial^2 J/\partial \theta_d^2 \end{pmatrix}$$

$$= \begin{pmatrix} \sum x_{n1}^2 h_\theta(x_n)(1-h_\theta(x_n)) & \cdots & \sum x_{n1} x_{nd} h_\theta(x_n)(1-h_\theta(x_n)) \\ \vdots & & \\ \sum x_{n1} x_{nd} h_\theta(x_n)(1-h_\theta(x_n)) & \cdots & \sum x_{nd}^2 h_\theta(x_n)(1-h_\theta(x_n)) \end{pmatrix}$$

$$= \sum_{n=1}^{N} h_\theta(x_n)(1-h_\theta(x_n)) \begin{pmatrix} x_{n1}^2 & \cdots & x_{n1}x_{nd} \\ \vdots & & \vdots \\ x_{n1}x_{nd} & \cdots & x_{nd}^2 \end{pmatrix}$$

$$H = \sum_{n=1}^{N} h_\theta(x_n)(1-h_\theta(x_n)) x_n x_n^T \quad \square$$

②

2)

(c) To show J is convex, we show H is +ve semidefinite as follows

$$z^T H z = z^T \left(\sum_{n=1}^{N} h_\theta(x_n)(1-h_\theta(x_n)) x_n x_n^T \right) z$$

$$= \sum_{1}^{N} h_\theta(x_n)(1-h_\theta(x_n)) z^T x_n x_n^T z$$

But since $z^T x_n = x_n^T z$, we get

$$z^T H z = \sum_{1}^{N} h_\theta(x_n)(1-h_\theta(x_n))(z^T x_n)^2$$

$$= \sum_{1}^{N} \sigma(\theta^T x_n)(1-\sigma(\theta^T x_n))(z^T x_n)^2$$

$\because \sigma(\theta^T x_n)$, $(1-\sigma(\theta^T x_n)) \in (0,1)$ and $(z^T x_n)^2 \geq 0$,

$z^T H z \geq 0$ and thus H is +ve semi definite i.e.

J is convex.

3)

(a) $L(\theta) = P(X_1, \dots, X_n; \theta) = P(X_1;\theta)\, P(X_2;\theta) \dots P(X_n;\theta)$

$\because X_i$ are independent

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

As we see in the expression obtained, the order of the RVs given is irrelevant due to their independence.

(b) $$\ell(\theta) = \log L(\theta) = \log\left(\prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i}\right)$$

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

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

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

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

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

To get critical pts, $\ell'(\theta) = 0 \Rightarrow \frac{1}{\theta}\sum_{i=1}^{n} x_i = \frac{1}{1-\theta}\sum_{i=1}^{n}(1-x_i)$

$$\cancel{\Rightarrow \sum x_i - \theta \sum_i x_i = \theta\left(\sum_{i=1}^{n} 1 - \sum_i x_i\right)}$$

$$\cancel{\Rightarrow \sum_i x_i - \theta\sum_i x_i = n\theta - \theta\sum_i x_i}$$

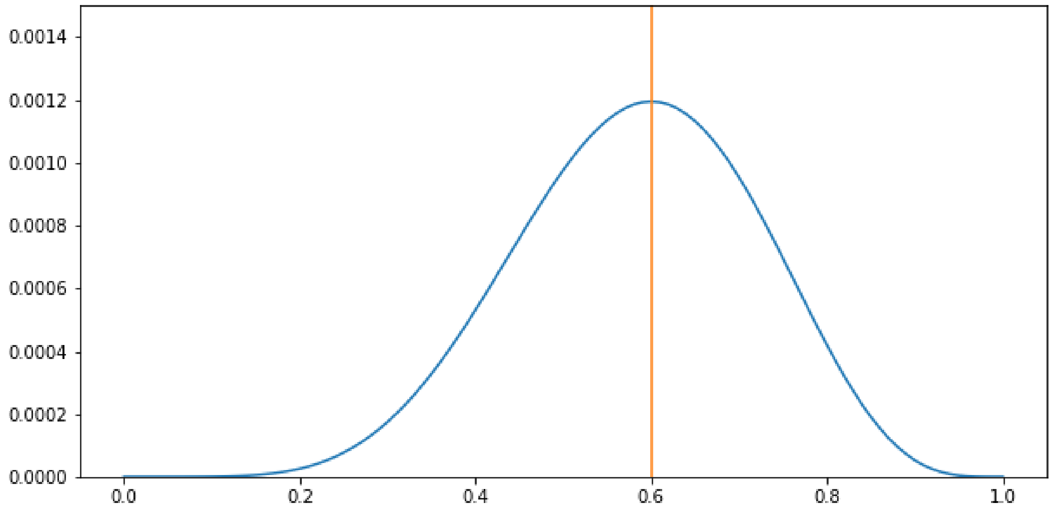
$$\Rightarrow \frac{1-\theta}{\theta} = \frac{\sum_i 1 - x_i}{\sum_i x_i} \Rightarrow \frac{1}{\theta} = \frac{\sum_i 1 - x_i}{\sum_i x_i} + 1.$$

So $\theta = \dfrac{\sum_i x_i}{\sum_i 1 - x_i + \sum_i x_i} = \dfrac{\sum_i x_i}{n} = \bar{X}$

So $\hat{\theta}_{MLE} = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

3. Maximum Likelihood Estimation

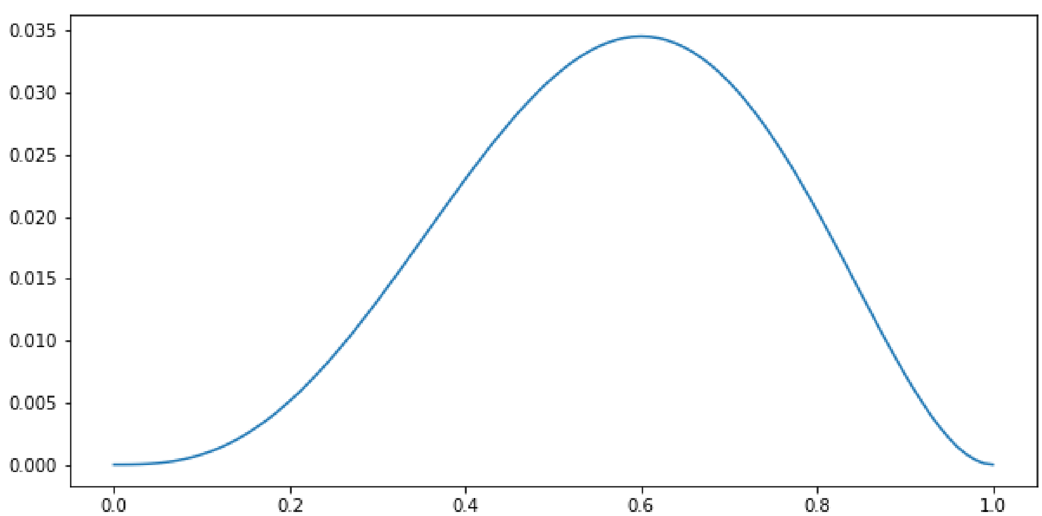
(c)



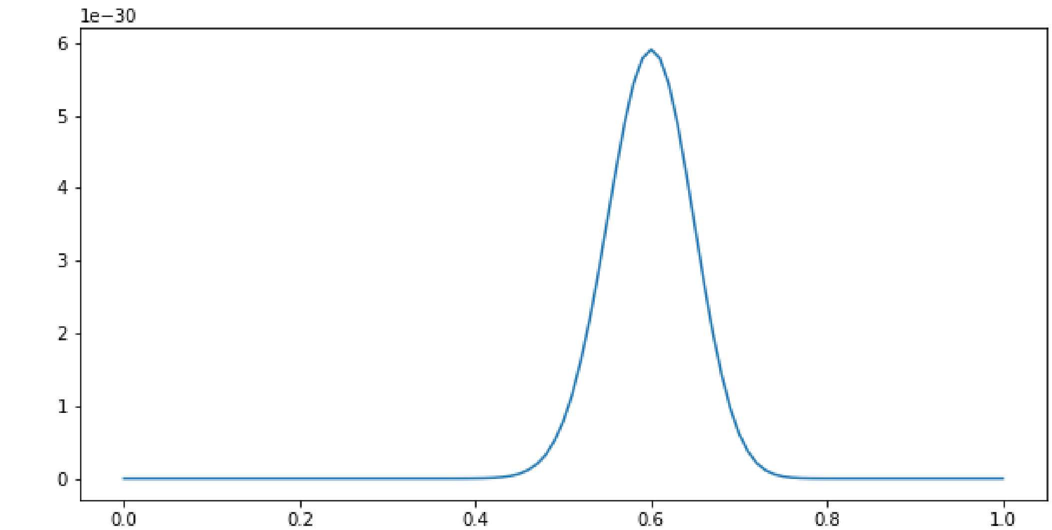
As seen marked by the orange line in the figure above, it appears that the max point on the likelihood function is 0.6, which agrees with the closed form solution.

(d) The maximum likelihood estimates for the datasets with their likelihood functions graphed below all agree with the closed form solutions, and are 0.6, 0.6 and 0.5 respectively. We also clearly see how the estimates are simply the number of hits (i.e. 1s) over the size of the dataset.

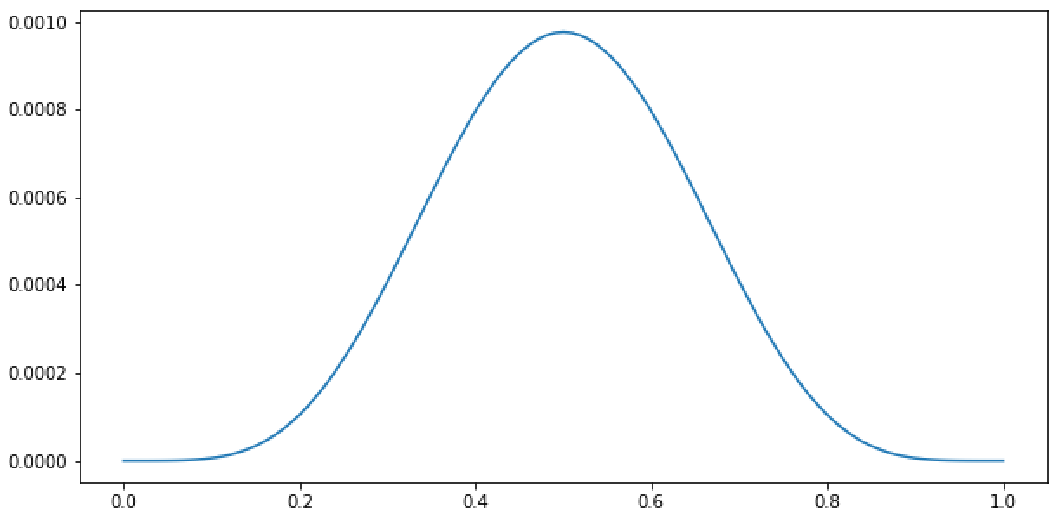
n = 5 with 3 hits



n = 100 with 60 hits

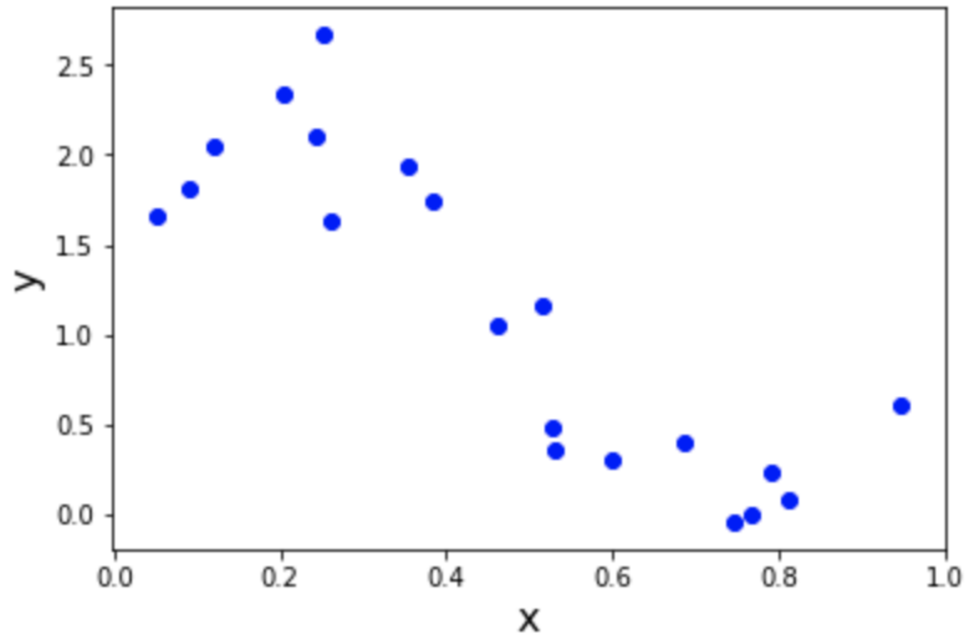


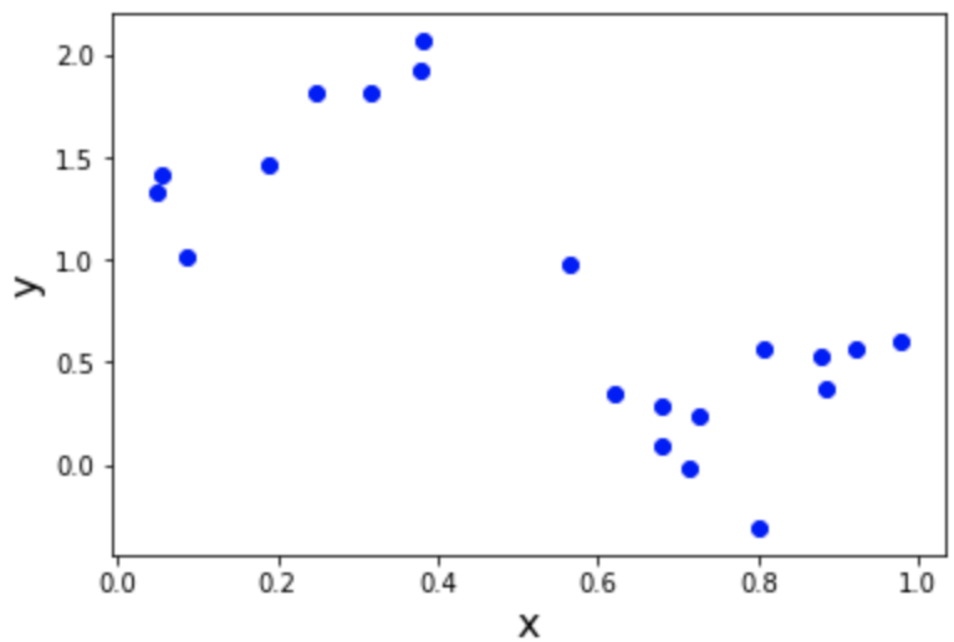
n = 10 with 5 hits



4. Implementation: Polynomial Regression

(a) Training data visualization followed by test data visualization.





Upon observing the visualizations above, it looks like although linear regression would capture some of the general negative correlation trends (particularly in the training set), to capture the full trend in the data, we would need a polynomial regression function.

(d) Here are the coefficients, number of iterations, final value of objective function and time till convergence for the given values of eta:

Eta	Num iterations	Coefficients	Cost	Time
10^{-4}	10000	[2.27044798 -2.46064834]	4.0863970368	0.214260101318
10^{-3}	7020	[2.4464068 -2.816353]	3.91257640579	0.152067899704
10^{-2}	764	[2.44640703 -2.81635346]	3.91257640579	0.0213170051575
0.0407	10000	[-9.40470931e+18 -4.65229095e+18]	2.71091652001e+39	0.253114938736

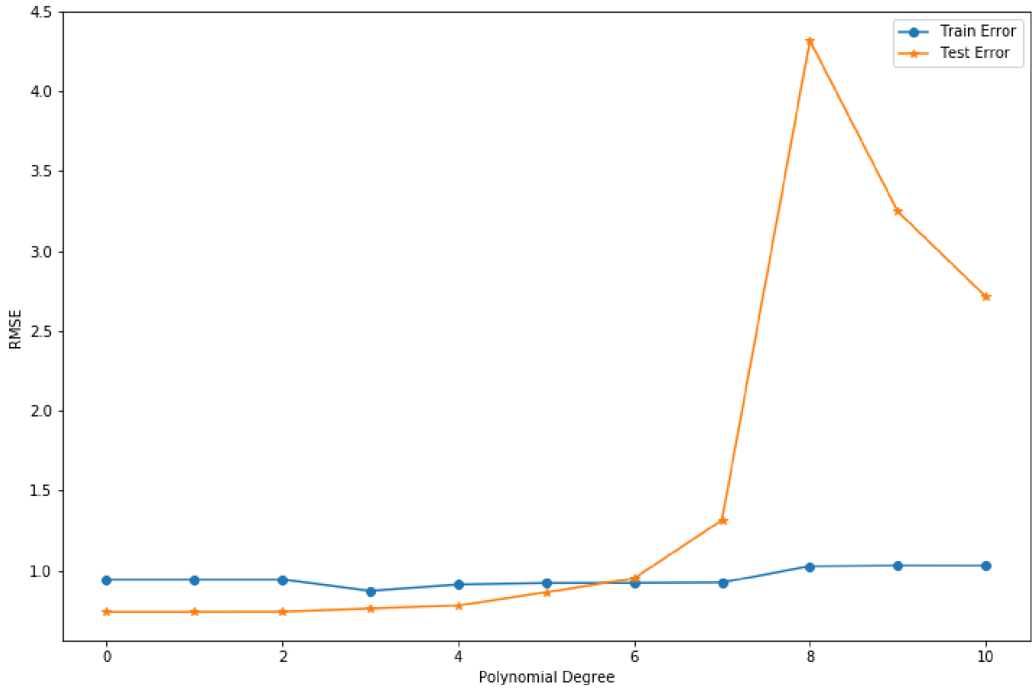
The coefficients converged with eta not too small or big, doing so particularly quickly with eta = 10^{-2}. The time to converge also increases with step size (as long as we don't use a value that's too big), as gradient descent is able to take sufficiently large steps. The algorithm performs worst with step size 0.0407 – not converging and probably bouncing around, given the extremely high cost.

(e) The model coefficients using the closed form solution are [2.44640709, -2.81635359], with cost 3.91257640579 and taking time 0.000427961349487. These coefficients and cost are the same as when the gradient descent algorithm converges with a good value of eta. However, the algorithm converges far quicker with the closed form solution.

(f) With the decreasing learning rate, the coefficients and cost are the same as the closed form solution (i.e. the algorithm converges) but it converges extremely fast, taking only 0.000357151 031494 seconds.

(h) We prefer RMSE to J(theta) since RMSE normalizes the error based on the number of samples we have. Thus, it provides a more balanced estimate of the errors, independent of the size of the dataset we're working with.

(i)



It looks like a polynomial of degree 6 best fits the data. The data is definitely underfit with lower degree polynomials and extremely overfit and erratic when using higher degree polynomials, as seen in the plot. The best options are between 3 and 6.