IST 597: Homework #1

Due on October 2, 2017

HW 1 :Regression and Gradient Descent

Sourabh Dalvi

October 1, 2017

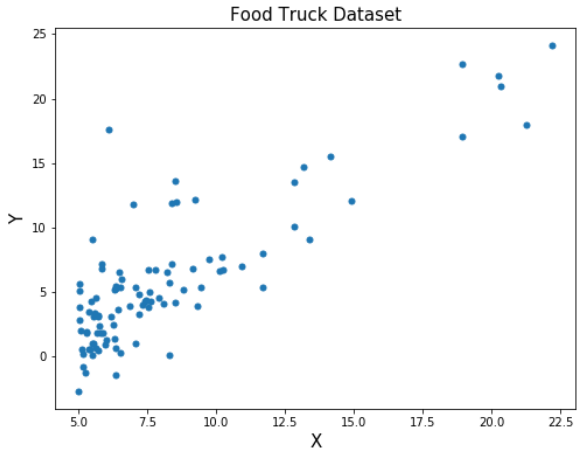
Contents

Problem #1 Univariate Linear Regression

Write a few sentences describing what you learned from the training/model fitting process. Things to discuss: What happens when you change the step-size ? How many epochs did you need to converge to a reasonable solution (for any given step size?)

The Linear Regression models that fitted to the data are shown in the Fig.1 below along with plot of the data. As we try various learning rate's α to get a reasonable solution, you see in the table below that some of the smaller α converge but takes more no. of epochs, while some diverge due to having very large α. For determining the no. of epochs, I have used a while loop with a convergence criterion that the decrease in Loss function value between consecutive gradient descent steps be less than 10^5 which is good enough to reach the best solution for various values of α for a problem of this size. As we wanted a model with out regularization I let the convergence criterion be very small to allow for a little over fitting. In each case the algorithm mostly converges after 1000 epochs

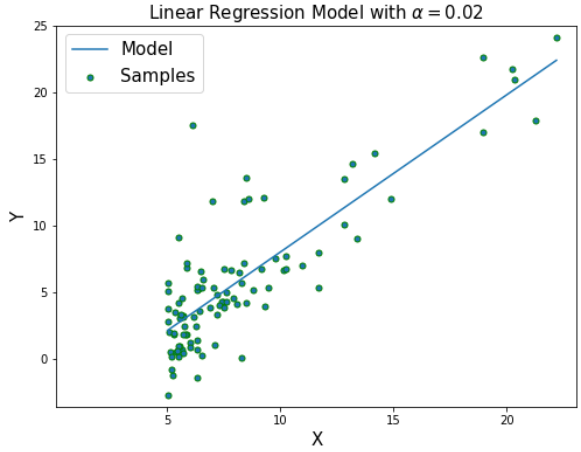
α	# of Epochs	Final Cost($J(\theta)$)	Bias(θ_0)	Weights (θ_1)	Comment
10	1	17641010	58	653	Diverges
1	1	172570	5	65	Diverges
0.1	1	1370	0.5	6.5	Diverges
0.02	963	4.478	-3.77	1.18	Converges (Best)
0.01	1734	4.479	-3.27	1.1	Converges
0.001	10953	4.51	-3.34	1.13	Converges



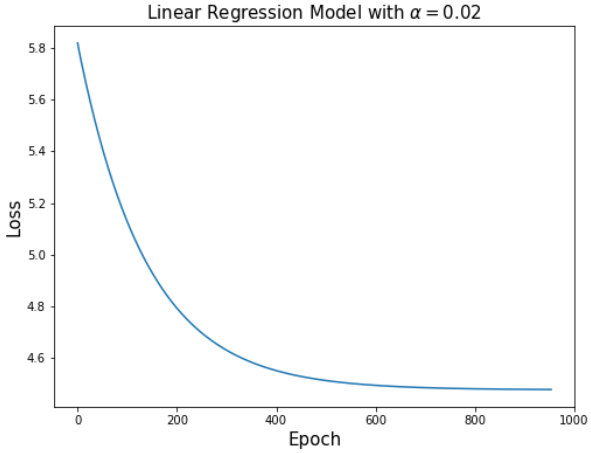
(a)

Figure 1:

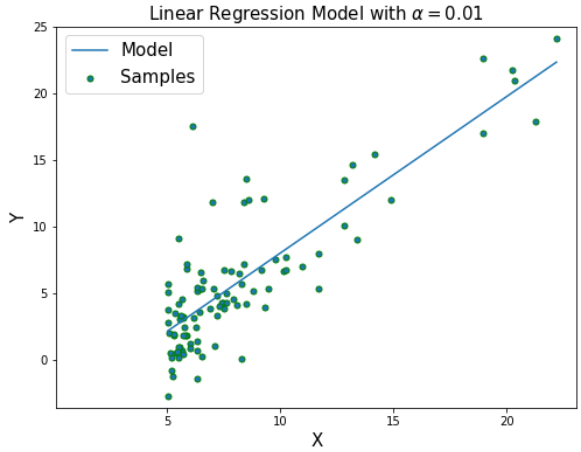
Note : I have used python 3 for this assignment.



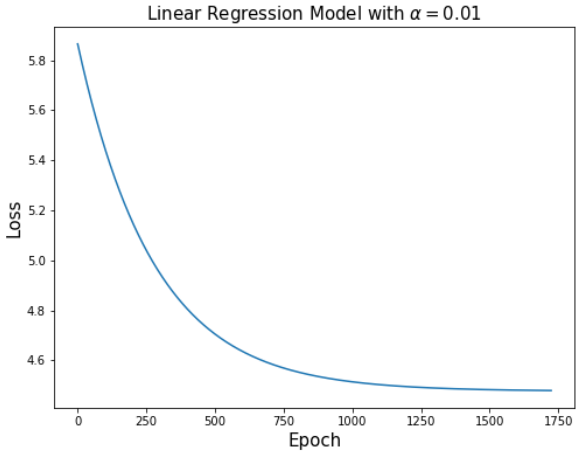
(b)



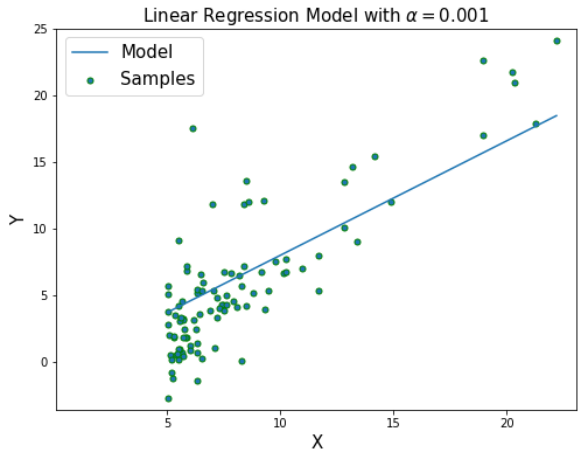
(c)



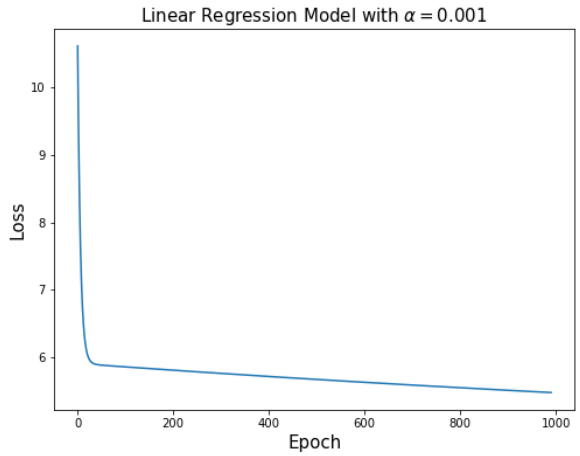
(d)



(e)



(f)



(g)

Figure 1:

Problem #2 Polynomial Regression & Regularization

With respect to the feature mapping, what is a potential problem that the scheme we have designed above might create? What is one solution to fixing any potential problems created by using this scheme(and what other problems might that solution induce)? Write your responses to these questions in your external answer document.

When using a feature map to increase the no. of features by taking the nth order terms of a single feature, the problem would be that some values may become very small or very large and this will result in underflow and overflow.
Another problem could be of determining the value of n, which gives us a model that is significant or if there are more than 2 variables we also need to take into account the interaction terms that would increase the capacity of the model even more and lead it to overfit, so again the question what should the right n be ?

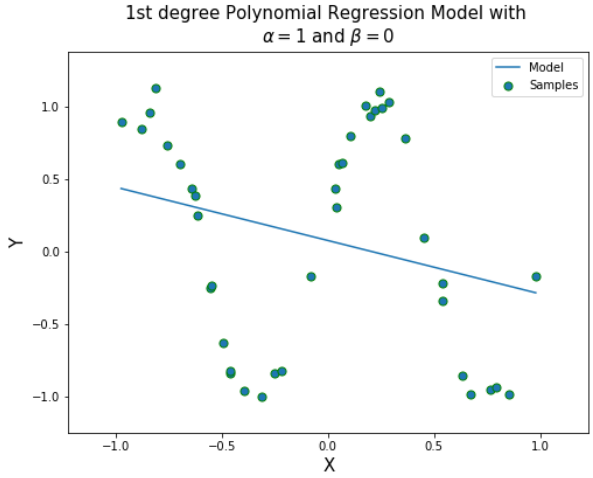
So, for Polynomial Regression model one potential problem with this feature mapping scheme would be of overfitting, our model would also capure the noise and will not generalise too well so when we are mapping our univariate feature to a higher dimension polynomial we should select a polynomial which has low test set error and moderate train set error.

One solution to this approach could be to regularize the weights of the polynomial features, this will penalize the weights of the features which will result in some useless feature to be dropped or have a lower value and prevent overfitting. Another method to tackle this problem can be to use an early stopping policy to regularize time or no. of epochs the model can perform. This can also prevent the model from taking very small steps so it does not overfit the data. One potential problem again with regularization is to know how much to regularize? What value of the regularizer will give us model that does not have much variance & bias. As too much of regularization could add bias to our model.
Fit a 1st order (i.e., linear), 3rd order, 7th order, 11th order, and 15th order polynomial regressor to the data in prob2_data.txt. What do you observe as the capacity of the model is increased? Why does this happen? Record your responses to these questions in your answer sheet.

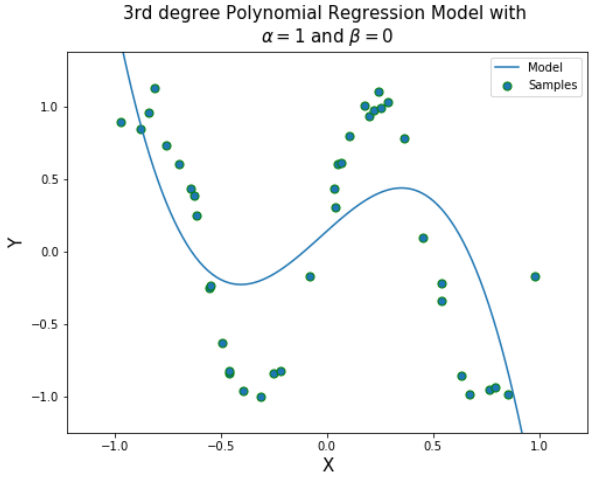
As we discuss above, when we increase the capacity of the model the regression line spreads out and the model captures the gaussian noise thinking its part of the cosine function, this results in our training error(Loss function value) to decrease. The reason behind this is simply that as the capacity increases the complexity of the model increases but the data that we are providing is not that complex.

α	# of Epochs	Training error($J(\theta)$)	Degree	β	Comment
1	14	0.26	1	0	Underfit
1	260	0.165	3	0	Underfit
1	7000	0.025	7	0	Overfit
1	1100	0.009	11	0	Overfit
1	7524	0.009	15	0	Overfit

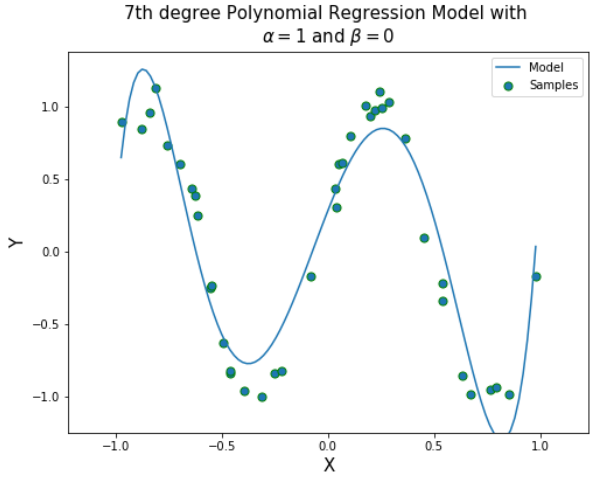
The Fitted model for 1st order, 3rd order, 7th order, 11th order and 15th order Polynomial Regression to the data are in Fig. 2 below, for training these models I have used an $\epsilon = 10^{-6}$ which is pretty small which does let the model converge to the minimum where these models will Overfit.



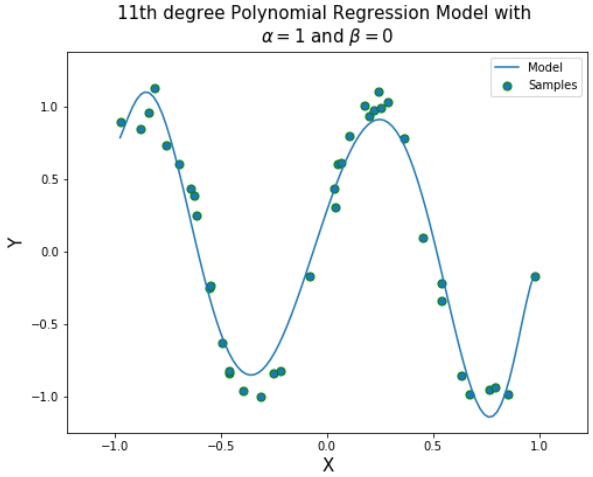
(a)



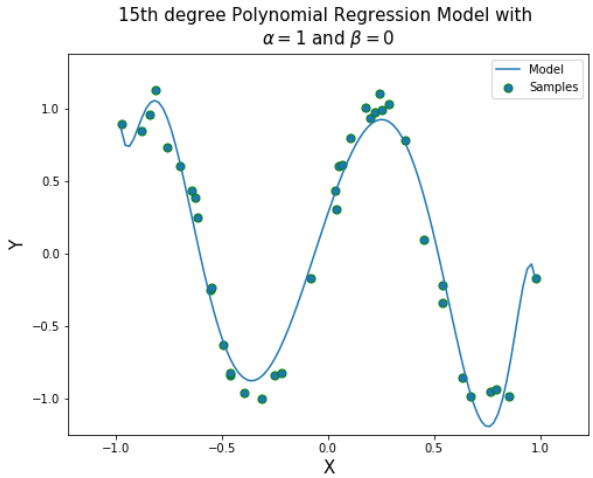
(b)



(c)



(d)



(e)

Figure 2:

Refit your 15th order polynomial regressor to the same data but this time vary the β meta-parameter using the specific values {0.001, 0.01, 0.1, 1.0} and produce a corresponding plot for each trial run of your model. What do you observe as you increase the value of β? How does this interact with the general model fitting process(such as the step size α and number of epochs needed to reach convergence)?

As we start to increase our meta-parameter β we see that our Loss function value at convergence slowly increases as we are adding the 2nd term to our loss function. Also our model starts to move away from the peaks of the wave and at $\beta = 1$ there is too much bias and model mostly underfits the data.

α	# of Epochs	Training error($J(\theta)$)	Degree	β	Comment
1	7717	0.0103	15	0.001	Overfit
1	5985	0.028	15	0.01	Overfit
1	1281	0.080	15	0.1	Overfit
1	231	0.1733	15	1	Underfit

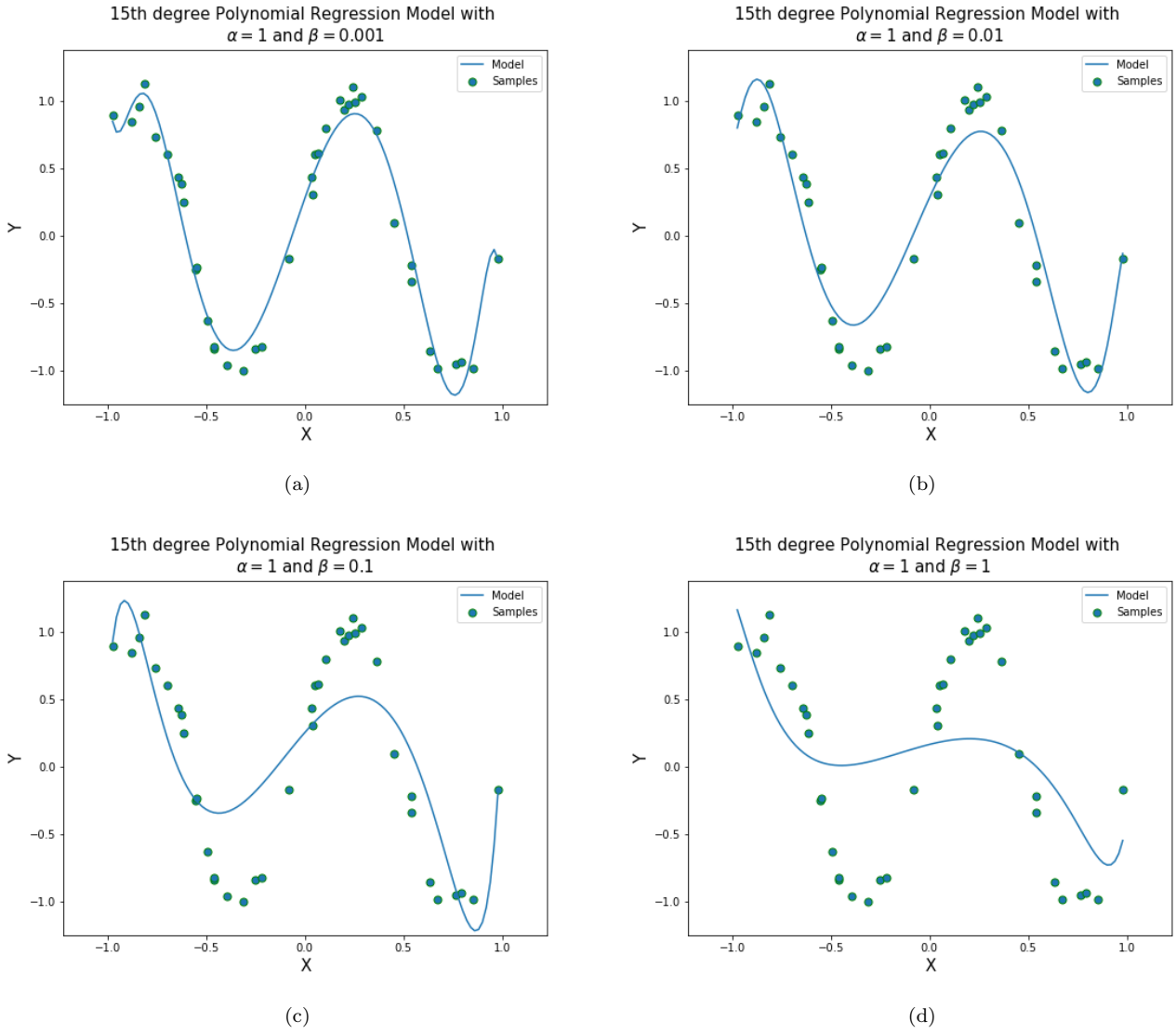


Figure 3:

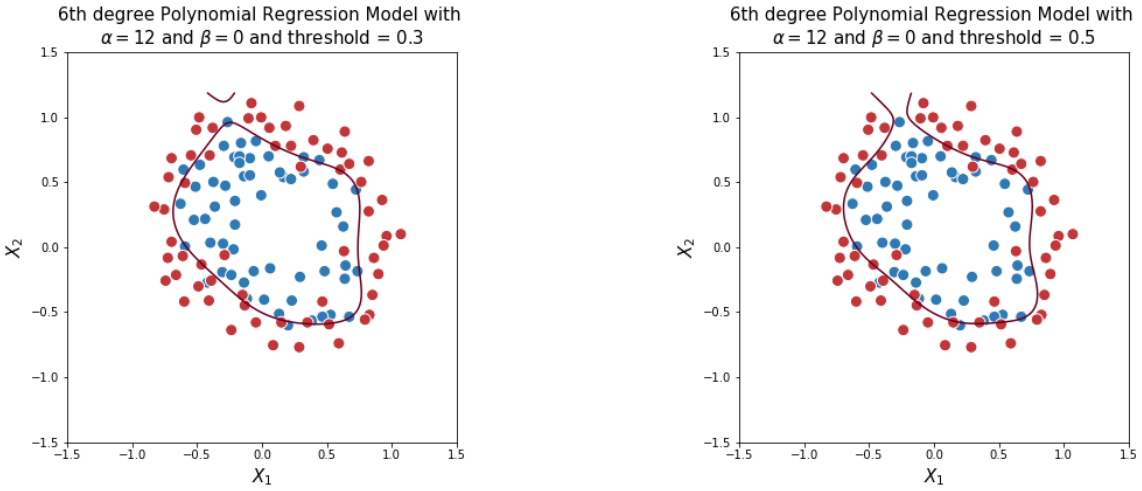
What might be a problem with a convergence check that compares the current cost with the previous cost (i.e., looks at the deltas between costs at time t and t - 1), especially for a more complicated model? How can we fix this ?

One possible problem with setting a convergence check scheme is that we have to also fine tune ϵ because it is also a kind of regularizer on the model which regularizes across time. So if ϵ is too high the model will underfit and if it is very small (like 10^{-10}) it will overfit the model by letting the gradient descent algorithm take very small steps. One way to tackle this is to plot the convergence of the model or to start from a higher value of ϵ then decrease it and check the decrease in training error.

Problem #3 Multivariate Regression & Decision Boundaries

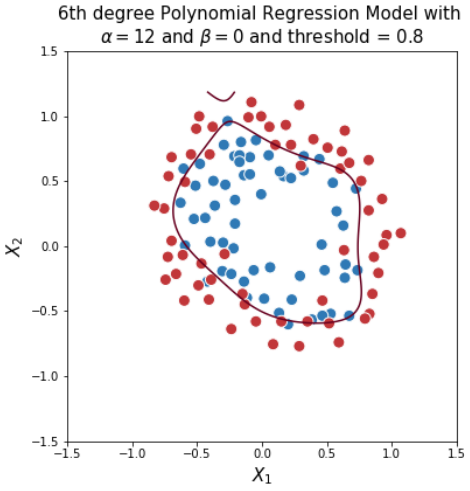
In this assignment, this threshold will be fixed to 0.5. (Feel free to play around with this value and write your observations in the answer document.)
If we change the threshold between 0.5, 0.3 and 0.8 we can see that the decision boundary changes very little but this affects the accuracy of the model. At threshold 0.5, the accuracy is the best as shown in the table below and with any change in the threshold the accuracy decreases.

α	Training error($J(\theta)$)	Accuracy	β	Threshold
12	0.2789	88.59 %	0	0.3
12	0.2789	87.28 %	0	0.5
12	0.2789	77.96 %	0	0.8
5	0.5290	68.64 %	1	0.3
5	0.5290	83.05 %	1	0.5
5	0.5290	51.69 %	1	0.8
1	0.6864	49.15 %	100	0.3
1	0.6864	62.71 %	100	0.5
1	0.6864	50.84 %	100	0.8

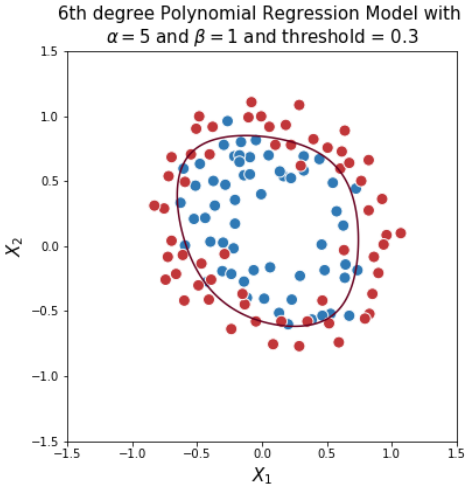


(a) (b)

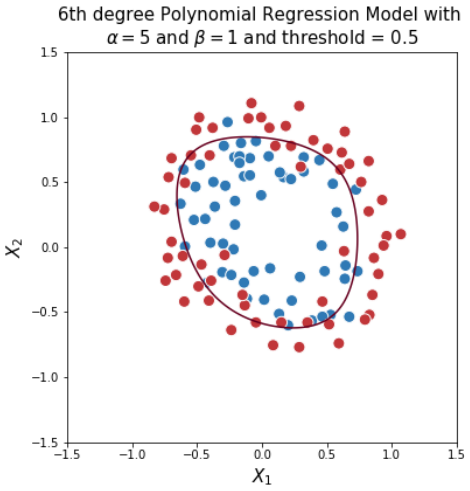
Figure 4:



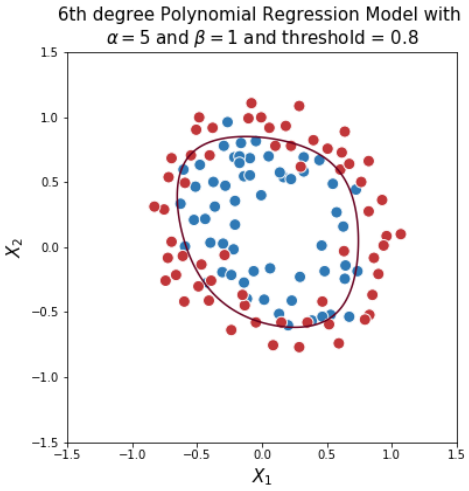
(c)



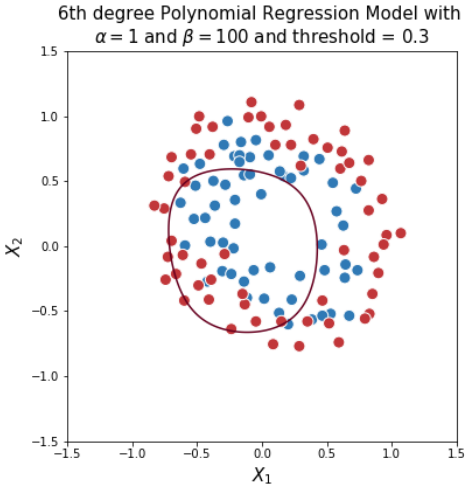
(d)



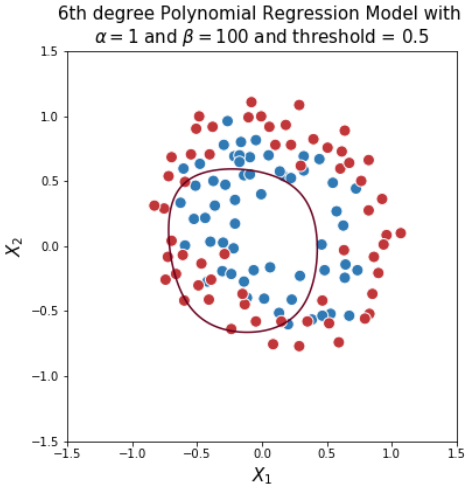
(e)



(f)



(g)



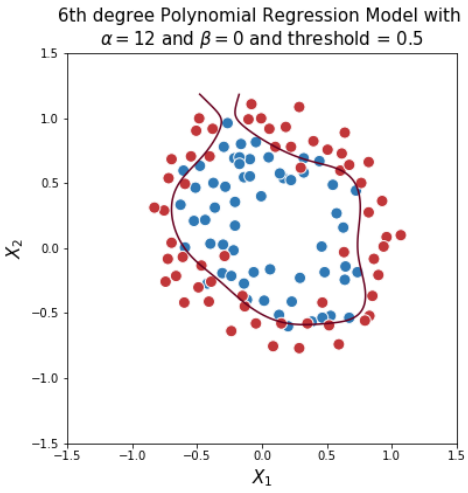
(h)

Figure 4:

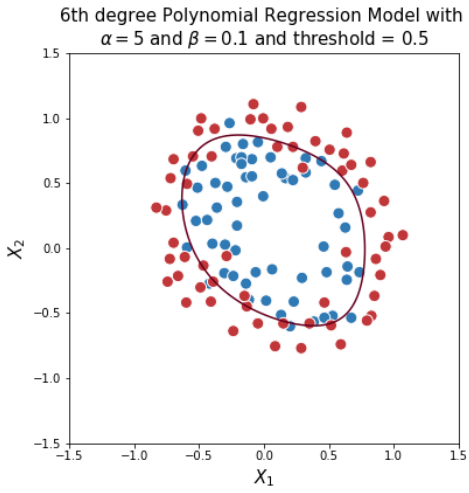
For the last part of this assignment, re-fit your logistic regression but this time with $= \{0.1,1,10,100\}$ again, tuning the number of epochs and the learning rate. Copy each of the resultant contour plots to your answer sheet and report your classification error for each scenario. Comment (in the answer sheet) how the regularization changed your model's accuracy as well as the learned decision boundary. Why might regularizing our model be a good idea if it changes what appears to be such a well-fit decision boundary?

So as we start to increase our regularization term β we can see its effects on the decision boundary. It slowly starts to become smaller due to the fact that the batch gradient descent algorithm can't take any further steps as the cost function will only increase as the value of our weights increase and if we increase β to 100 the algorithm does not take any step as setting weights θ_i would increase the cost to very high value and the value of weights is very small.
As the value of weights decrease due to regularization the model accuracy also decreases as our model does not get to train properly i.e. it underfits the data due to over regularization. This might not be the case always. If we use regularization properly, we can drop the variables that are non significant in our model, which is similar to using ANOVA model and dropping variables with over R^2 values. These non-significant variables are more likely to overfit our model and cause our model to not generalize well. So its a good idea to use regularization even though it seems to change our decision boundary slightly.

α	Training error($J(\theta)$)	Accuracy	β	Threshold
12	0.2789	87.28 %	0	0.5
5	0.3945	83.89 %	0.1	0.5
5	0.5290	83.05 %	1	0.5
2	0.6480	72.03 %	10	0.5
1	0.6864	62.71 %	100	0.5

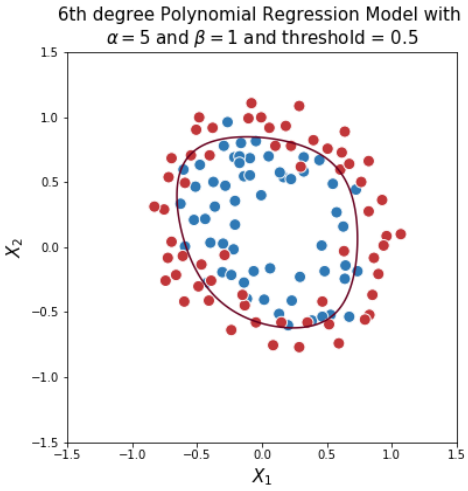


(a)

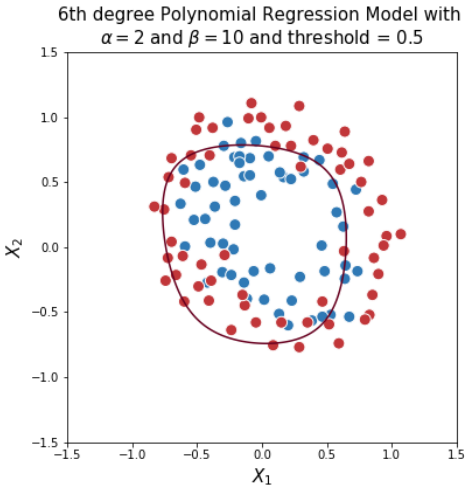


(b)

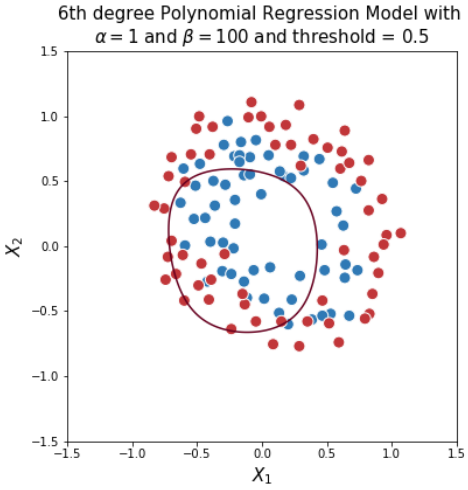
Figure 5:


6th degree Polynomial Regression Model with
$\alpha = 5$ and $\beta = 1$ and threshold = 0.5
X_1
X_2

(c)


6th degree Polynomial Regression Model with
$\alpha = 2$ and $\beta = 10$ and threshold = 0.5
X_1
X_2

(d)


6th degree Polynomial Regression Model with
$\alpha = 1$ and $\beta = 100$ and threshold = 0.5
X_1
X_2

(e)

Figure 5: