IST 597: Foundation of Deep Learning
HW 1: Regression and Gradient Descent

Nikhil Sunil Nandoskar, nxn59@psu.edu

Part 1:

What happens when you change the step-size α? How many epochs did you need to converge to a reasonable solution?

Ans:

Different values for the learning rate were chosen. The details about the various step-size α, corresponding epochs, and the loss is given in the following table.

Step-Size α	**Number of epochs**	**Final Loss**	**Weight at the point of convergence**	**Bias at the point of convergence**
0.0001	4999	5.6784	0.8280	-0.2629
0.001	4565	4.7543	1.0177	-2.1506
0.01	1732	4.4797	1.1755	-3.7217
0.1	NAN	NAN	NAN	NAN
0.02	961	4.4783	1.1807	- 3.7730
0.04	NAN	NAN	NAN	NAN

Table 1: The parameters kept constant for this table are number of epochs: 5000 and epsilon: 0.0001

Points:

1) I started with the minimum learning-rate/ step-size (α) of 0.0001. Thereafter, I kept on increasing α by a factor of 10 (multiplying in orders of 10)

   NOTE: We can choose the number of epochs to be more than 5000. Can take 20000 or more I'm choosing 5000 to be constant throughout as it takes a lot of time to converge even on google colab.

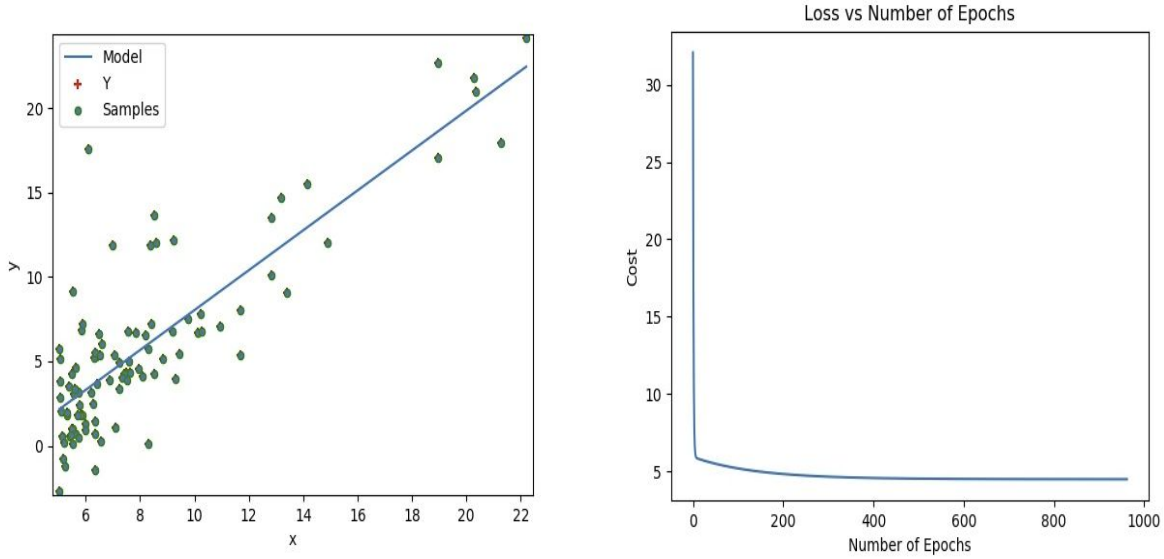
2) As we increase the step-size the model takes a smaller number of steps to find local minima. But we can not increase it after a particular step size as it overshoots and diverges (for $\alpha>0.02$) The reason behind the decrement in the number of epochs taken to converge is that we are taking a larger step towards obtaining local minima.

3) Best parameters observed:

   Learning rate/Step-Size α: 0.02

   Number of epochs: 961

   NOTE: These hyperparameters work perfectly for the provided dataset



(a) (b)

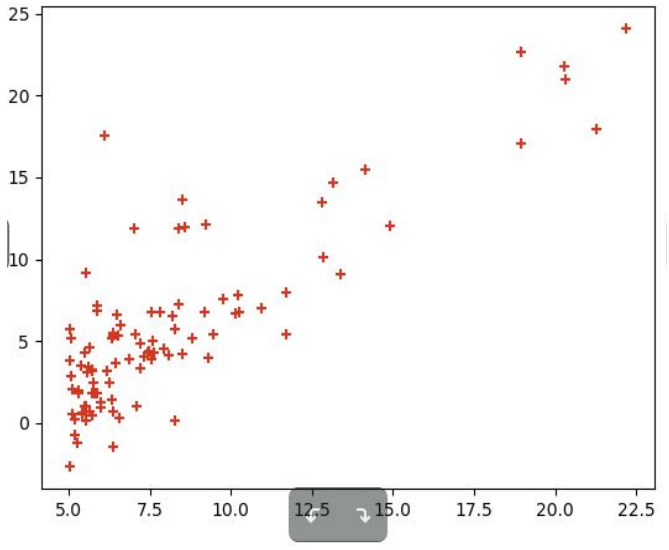
Figure 1: (a) is plot of our model fitting the given test-data. (b) is plot of observed loss vs the number of epochs

These were the best hyperparameters observed in my case because the loss occurred to be the minimum as well as the number of steps taken to converge.

Some additional observations:

(97, 2)

	X	Y
count	97.000000	97.000000
mean	8.159800	5.839135
std	3.869884	5.510262
min	5.026900	-2.680700
25%	5.707700	1.986900
50%	6.589400	4.562300
75%	8.578100	7.046700
max	22.203000	24.147000



(a) (b)

Figure 2: (a) shows the shape of dataset (rows: 97, columns: 2) ans statistic about it. (b) shows the Scattered plot of our dataset

Part 2:

i) 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)?

ii) What do you observe as the capacity of the model is increased? Why does this happen?

Ans:

The approach I took for this problem was I first changed the step-size and kept the beta, epsilon and degree constant. Doing this helped me figure out the best alpha I can choose. After this, I tried different degrees keeping the rest of the hyperparameters constant including the best alpha.

Following is the table of different step sizes.

Step-Size α	**Number of epochs**	**Final Loss**
0.1	2512	0.0788
0.5	2195	0.0515
1	1291	0.0601
1.5	1129	0.0568
2	1	0.2544

Table 2: The parameters kept constant for this table are number of epochs: 5000, epsilon: 0.0001, beta: 0.01 and degree: 15

As we can see from table 2, for step-size 1.5 we have the minimum loss and the minimum number of epochs required for convergence (for step-size 0.5 the loss is the lowest but the number are epochs are considerably greater than step-size 1.5).

As we are increasing the step-size we are taking a larger step towards the local minima. After studying Table 2 and Figure 3 (scroll down) I decided to choose step-size to be 1.5.

The best parameters observed for step-size 1.5 are as follows:

Beta = 0.01

eps = 0.00001

Number of epochs: 1129

Points:

i) With respect to feature mapping, the potential problem we might get is overfitting. The polynomial function of higher degree is prone to more oscillations within exact fit values. Although they provide a good fit within a certain range of data, they perform poorly outside the data range.

Polynomial models exhibit a poor trade-off between shape and degree. Higher degree is required in order to model data with a complicated structure. Thus, the parameters to be Estimated are also high. This leads to very unstable models.

The possible solution here I can think about is "Regularisation". It is a technique of avoiding our model from getting overfitted. Overfitting is a scenario wherein our model will perform good on training data (attempting to go through every point of dataset) but performs poorly on new data/test data. On the other hand, underfitting is a scenario where our model performs poorly on the training data. Regularisation penalizes model whenever it tries to overfit for the higher degree of polynomials.

In our case, we have chosen L2 regularisation for our dataset. The only disadvantage(not really) of using regularisation is that now we have one additional hyperparameter to tune.

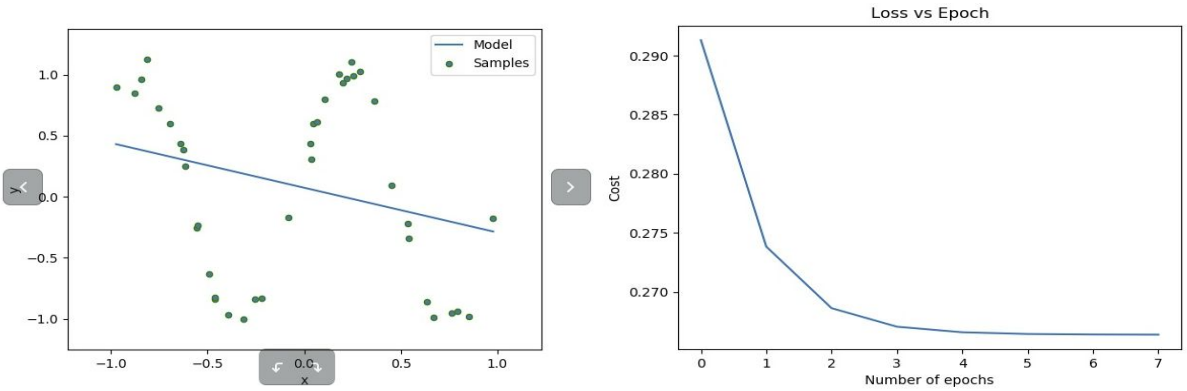
(reference: https://en.wikipedia.org/wiki/Polynomial_and_rational_function_modeling)

ii)

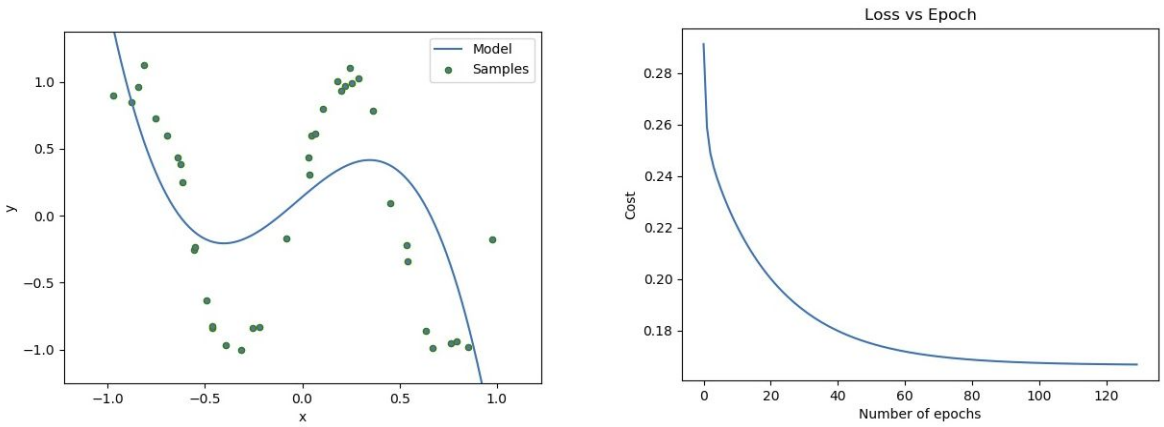
Step-Size α	**Degree**	**Number of epochs**	**Final Loss**
1.5	1	6	0.2663
1.5	3	128	0.1667
1.5	7	804	0.0568
1.5	11	716	0.0607
1.5	15	1129	0.0568

Table 3: The parameters kept constant for this table are number of epochs: 5000, epsilon: 0.0001, beta: 0.01

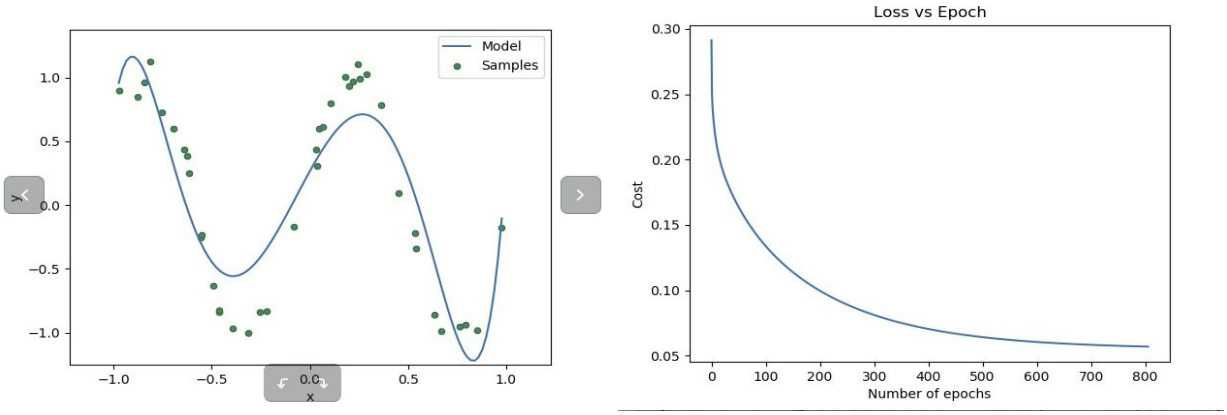
The following section contains graphs plotted for various degree and keeping rest of the hyperparameters same.



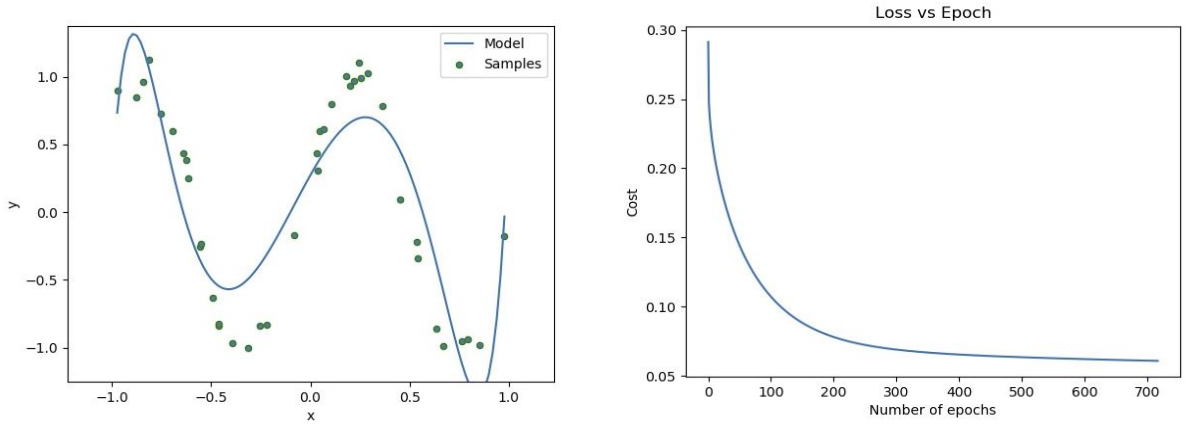
(a) For Degree: 1



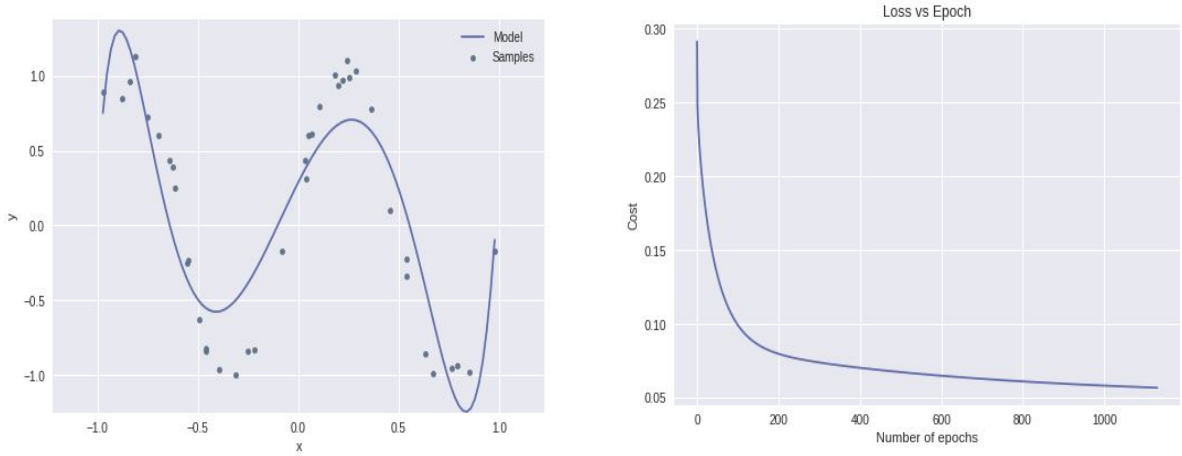
(b) For Degree: 3



(c) For Degree: 7



(d) For Degree: 11



(e) For Degree: 15

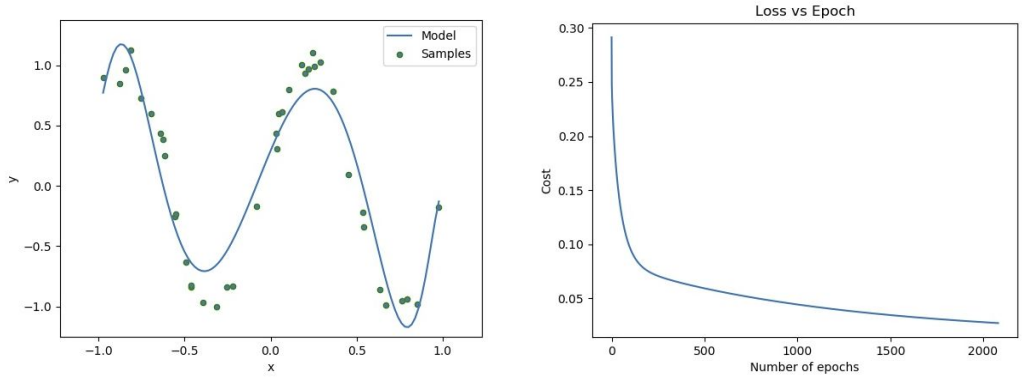
Figure 3: The above plots are plotted for various degrees keeping the hyperparameters constant (number of epochs: 5000, α: 1.5, beta: 0.01, epsilon: 0.0001)

The plots for various degrees are shown in Figure 3 above. Referring to (a) we can clearly see that it is a case of underfitting. This occurs because we are having the lowest model capacity. It cannot follow the trend of the dataset. Thus, it performs poorly and the loss is also high(refer Table 3). As we increase the model capacity our model tries to fit the dataset properly. This is occurring because we are providing more free parameters so it is trying to learn the complexity with more efforts. As we increase the degree we observe more sharp curves in the plot. For degree 15 it is clearly visible that our model is getting towards overfitting. Without proper L2 penalization, our model will overfit. Increasing the value of beta in this case, we'll comparatively get flatter curves.

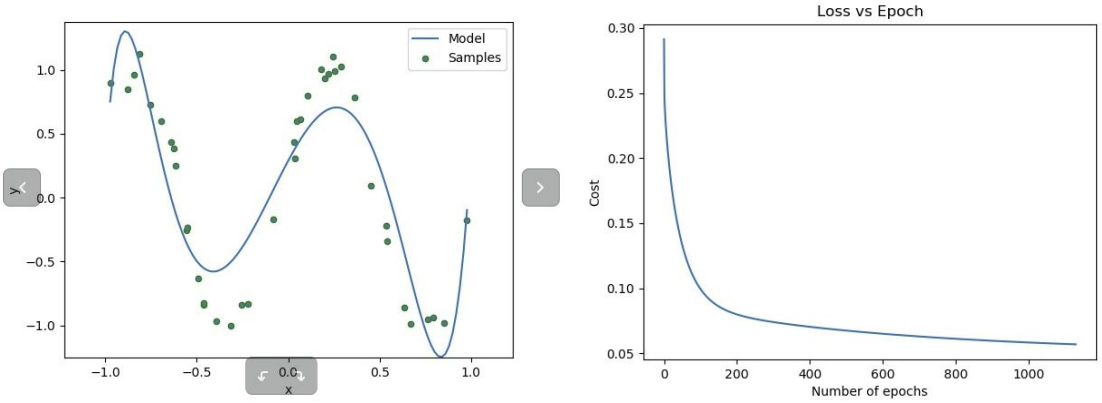
iii) Refit your 15th order polynomial regressor to the same data but this time vary the β meta-parameter using the specific values {0.001, 0.01, 0.1, 1.0}and produce a corresponding plot for each trial run of your model. What do you observe as you increase the value of β? How does this interact with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

Points:

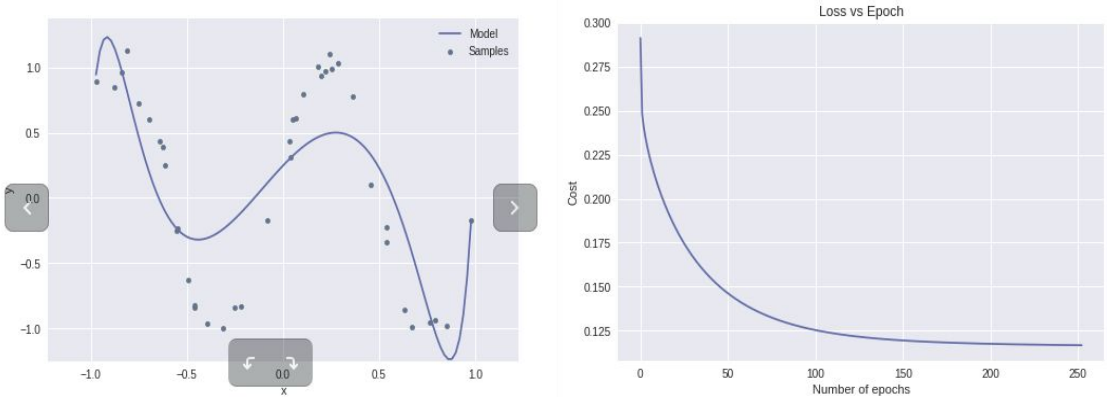
The following section contains graphs plotted for various beta and degree 15



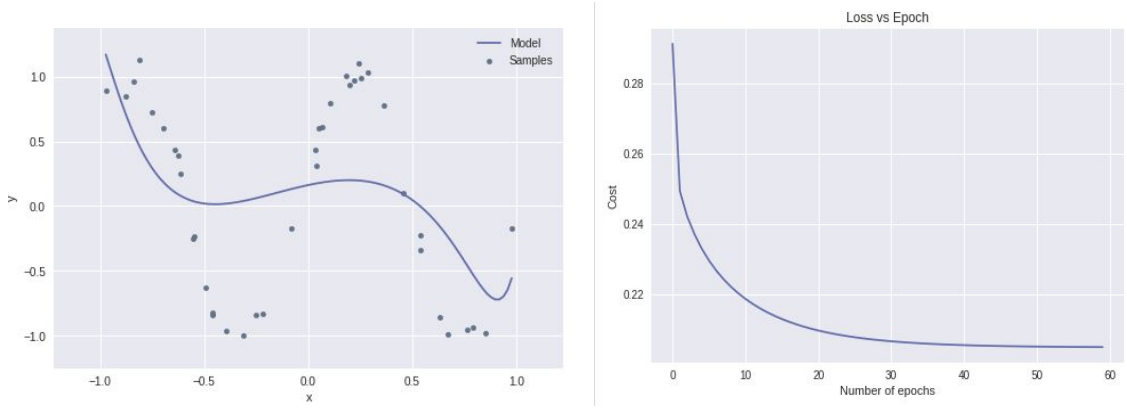
(a) Beta: 0.001



(b) Beta: 0.01



(c) Beta: 0.1



(d) Beta: 1

Figure 4: The above plots are plotted for various betas keeping the α: 1.5

Beta	Number of epochs	Final Loss
0.001	2080	0.0272
0.01	1129	0.0568
0.1	251	0.1167
1	58	0.2048

Table 4: The parameters kept constant for this table are number of epochs: 5000, epsilon: 0.0001, α: 1.5

As it is evident from Figure 4 and Table 3 as we increase the value of beta our model gets penalized more. In this case, our model capacity is decreasing i.e. it is getting fewer parameters. Also, the L2 regularisation is putting a heavy penalty on higher weights. Due to these effects, our model converges faster.
Table 2 shows the relationship between beta and step-size. Keeping beta constant and increasing step-size for degree 15 we take reach convergence at a faster pace. This is simply due to liger step-size we are proceeding rapidly to reach local minima.

iv) Comment (in your answer sheet) as to how many steps it then took with this early halting scheme to reach convergence. 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?

Points:

Table 2 shows the result of the number of final steps taken by our model before reaching a convergence point. In my case, it took 1129 steps to reach convergence for alpha: 1.5, degree: 15
The problem with such convergence check scheme I can think of are as follows:

1) Real-life datasets don't have a convex shape. In such cases, if we reach a convergence point then there are possibilities that we can be one of the local minima and not global minima.
2) Also if we are traversing on a relatively flatter surface there is a possibility that we can land in a global minima if we keep on moving in that direction.

Fixing:

We can increase the number of epochs and observe the output. Also, we can check for bias and variance problem.

Part 3:

i) Fit your (non-linear) logistic regression5 to the data found in data/prob3.dat, keeping beta Fixed at 0 (which means no regularization), tuning only the number of epochs and your α.

Ans:

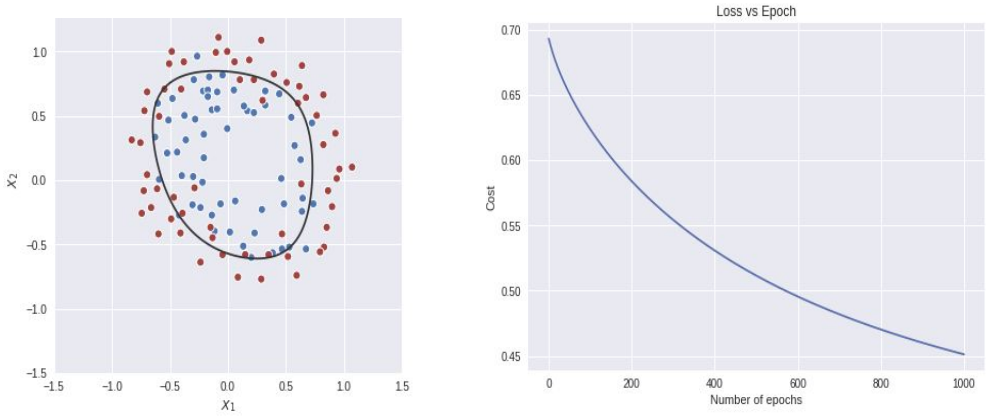
Step-Size α	**Number of epochs**	**Final Loss**
0.1	1000 (ran for 1000 epochs)	0.4511
0.5	1837	0.3419
1	1397	0.3352
1.5	1244	0.3314
2	1169	0.3287

Table 5: The parameters kept constant for this table are the number of epochs: 5000, degree: 6

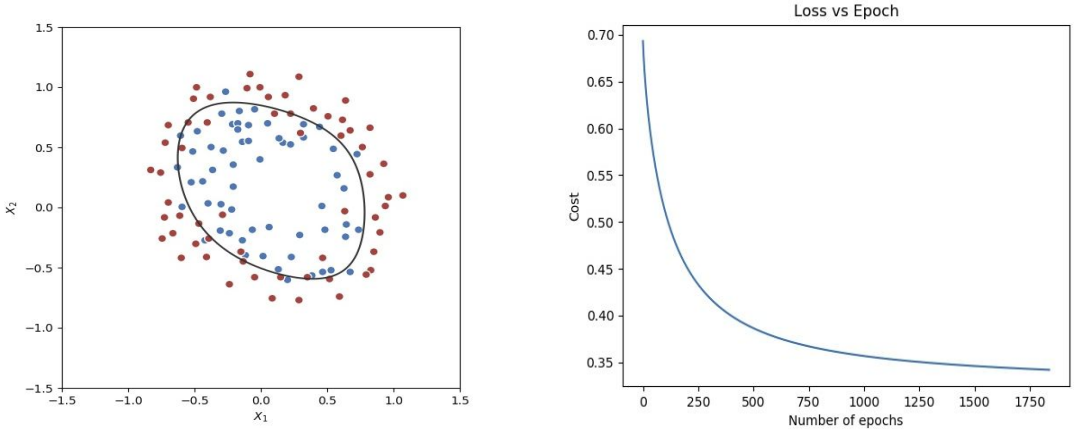
NOTE: For stepsize 0.1 it was taking a lot time to converge I ran it for 1000 epochs.

As shown in Table 5 I'm increasing the step-size by a factor of 0.5. Increasing the step-size we move quickly towards local minima. This simply means we take fewer epochs for convergence. In our case, the best step-size values could be 1, 1.5 and 2. Comparing the number of epochs taken and final loss obtained for both the step sizes one might consider going with the step-size 2, but here's is the catch. We are not regularising our model! For a larger step-size, our model will "overfit". Though it may perform very well on our training data, given a new data it will perform poorly. Thus, I chose to use step-size 1.5 for the rest of the computations.

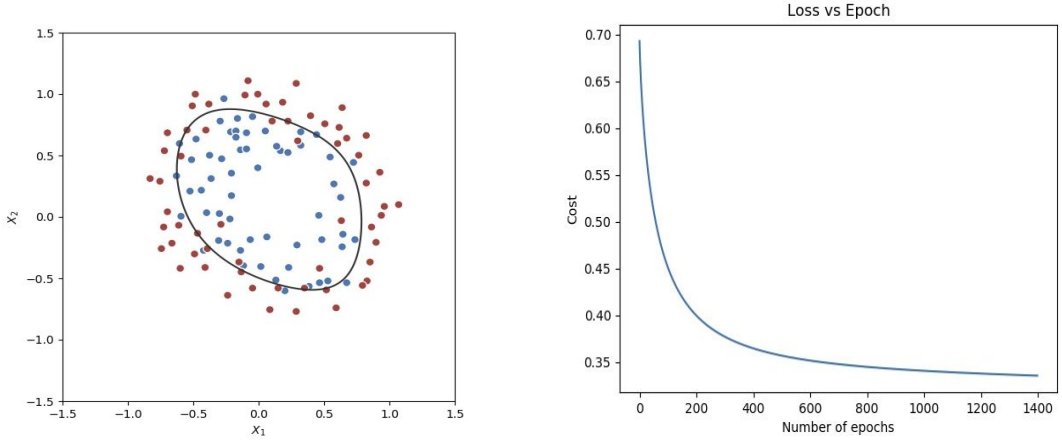
The following section contains graphs plotted for various step-size for degree 6



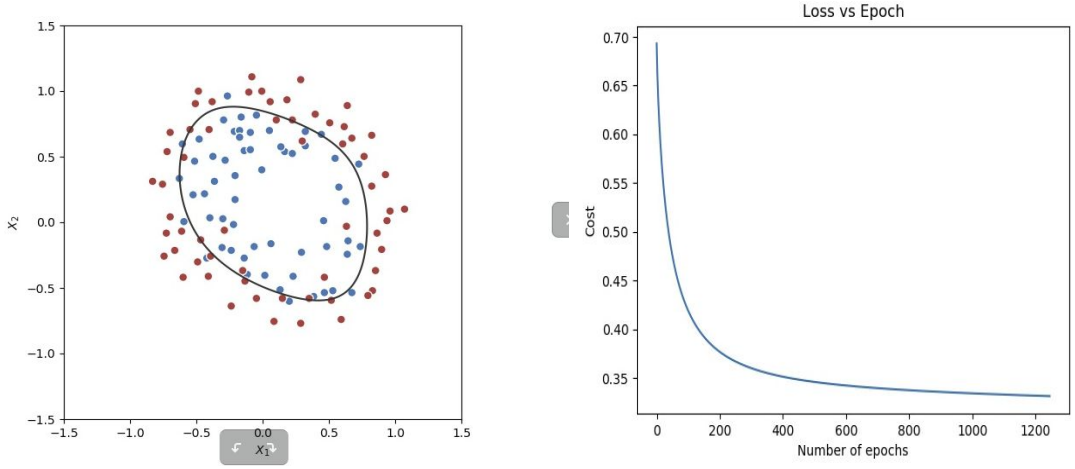
(a) α: 0.1



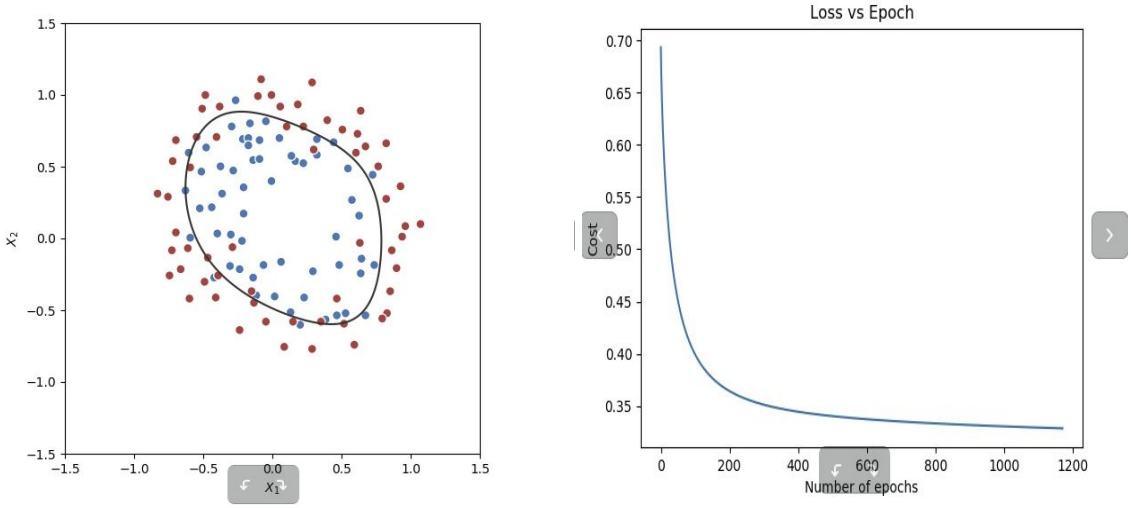
(b) α: 0.5



(c) α: 1



(d) α: 1.5



(e) α: 2

Figure 5: The above plots are plotted for various step-size, keeping degree: 6, beta: 0

ii) For the last part of this assignment, re-fit your logistic regression but this time with β = {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?

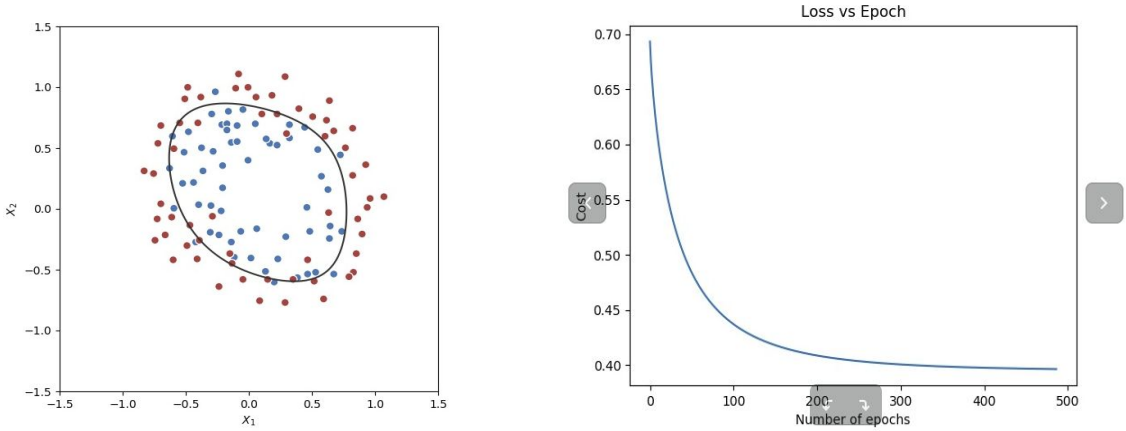
Ans:

For this part I have kept step-size fixed at 1.5, degree is 6 and number of epochs: 2000. The results that I obtained are shown in the following table.

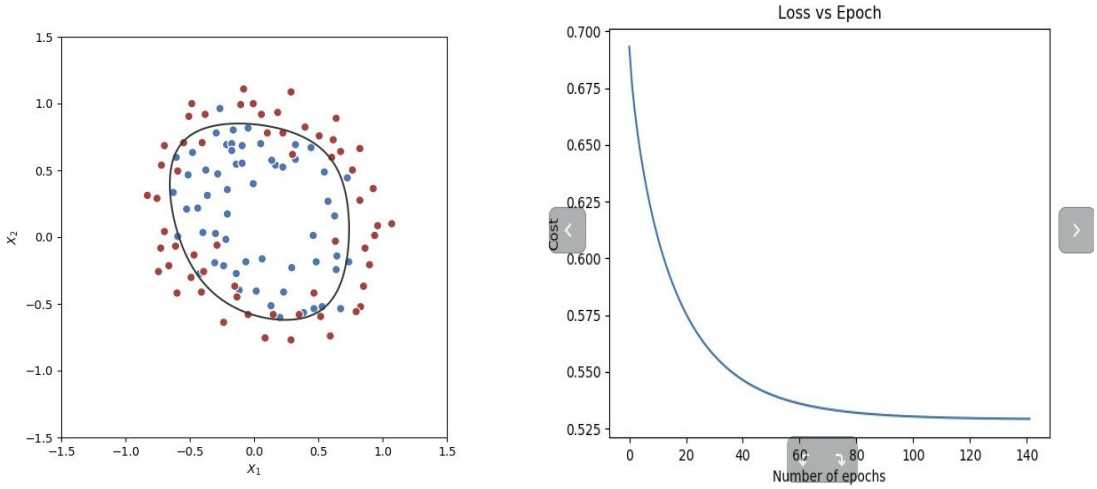
Beta	**Number of epochs**	**Final Loss**	**Classification Error (%)**
0.1	485	0.3964	16.9491
1	140	0.5292	16.9491
10	24	0.6482	25.4237
100	5	0.6864	36.4406

Table 6: The parameters kept constant for this table are the number of epochs: 2000, degree: 6, α: 1.5

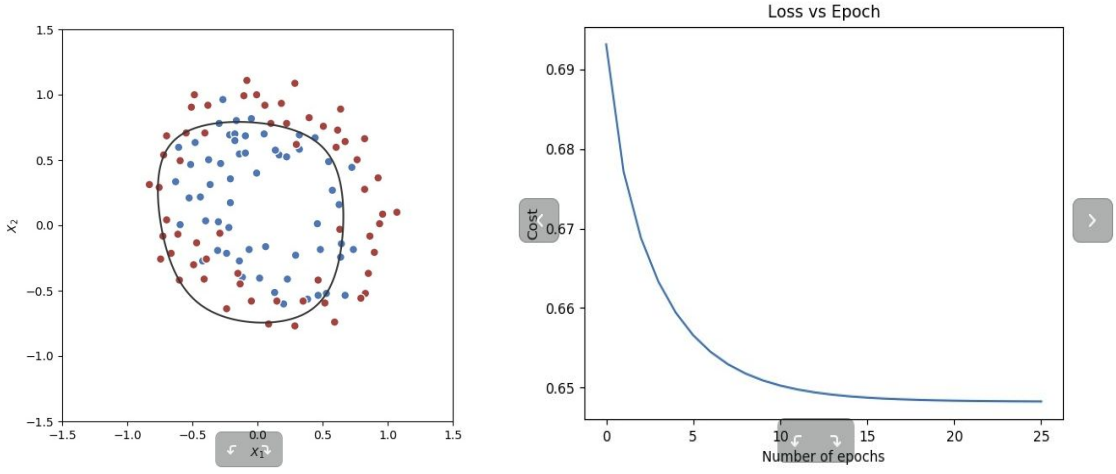
The following section contains graphs plotted for the various beta for degree 6



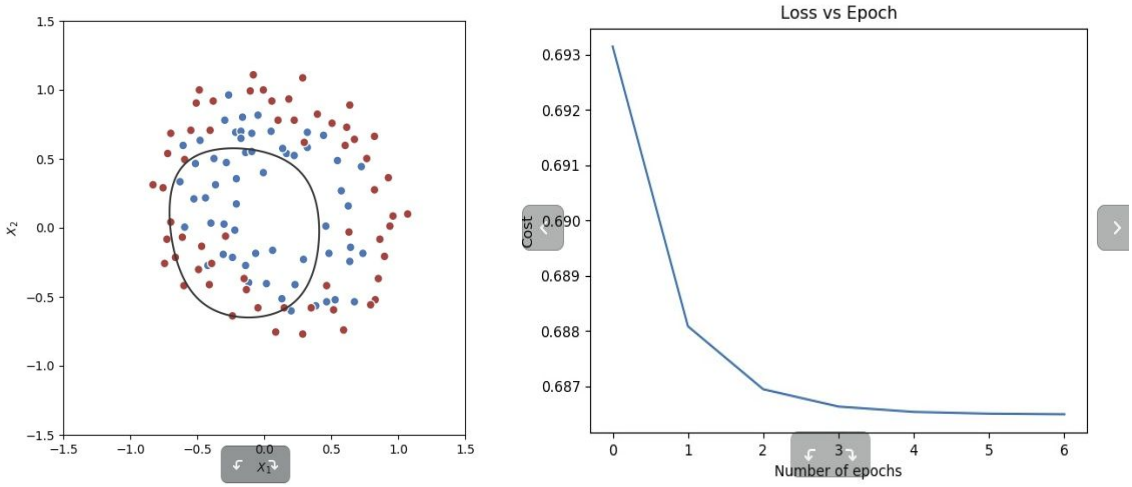
(a) Beta: 0.1



(b) Beta: 1



(c) Beta: 10



(c) Beta: 100

Figure 6: The above plots are plotted for the various beta, keeping degree: 6, α: 1.5

Points:

Table 6 shows the various beta values and the corresponding number of epochs, final loss and classification error. For bets values 0.1 and 1 though the classification error is same up to 4 decimals, the number of steps taken before convergence differs.

As we increase the value of beta the classification error is increasing.

Regularisation is required to avoid our model from getting overfitted. But, there's a upper limit to how much we can penalise our model. Before regularisation the weights have a huge range but after regularisation the variance decreseases and the range of weight decresases As evident from Table 6 and Figure 6 for beta 100, our classification error is the highest and the decision boundary doesn't cover all the blue dots.

If we don't add L2 regularisation to our model it will overfit and perform poorly whenever any testing data is added. Thus, it's always a good practice of regularising our model. For obtaining optimal parameters we need to follow the procedure we followed in this assignment.

The best parameters I obtained for this dataset are as follows:

Beta = 1

Alpha = 1.5

eps = 0.00001

Number of epochs: 140