IST 597: Foundations of Deep Learning
Homework 1: Regression & Gradient Descent
Submitted by: Pakhi Agarwal

Problem 1: Univariate Linear Regression

For this part of the assignment, you will **build a simple univariate linear regression model to predict profits for a food truck. Fit your linear regression model on the food truck data and save a plot showing your linear model against the data samples. You should record the settings you tried and what you found that worked also in the answer sheet. 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 α? How many epochs did you need to converge to a reasonable solution (for any given step size?**

Solution.

Figure 1 shows the scatter plot of Dataset. For doing all our experiments with the model we will use this dataset.

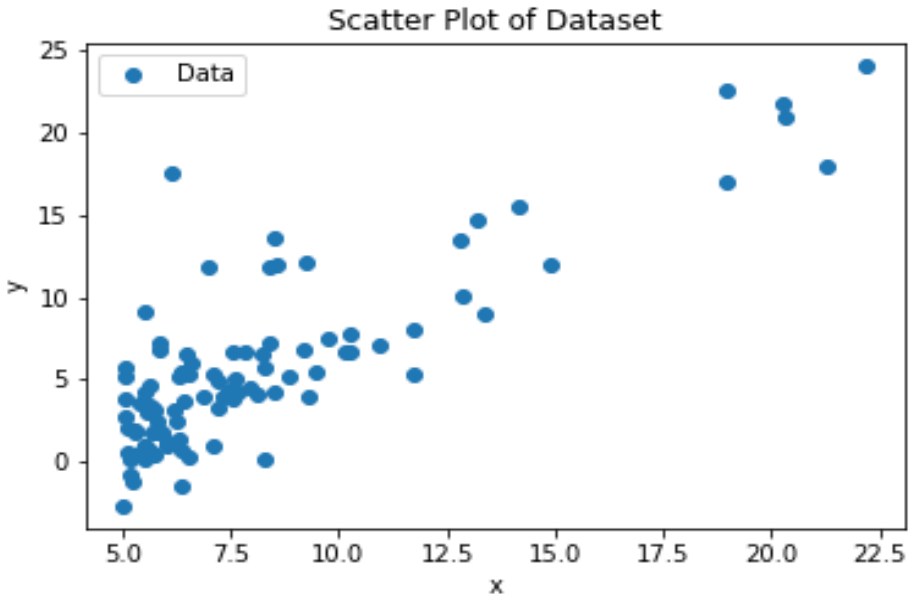
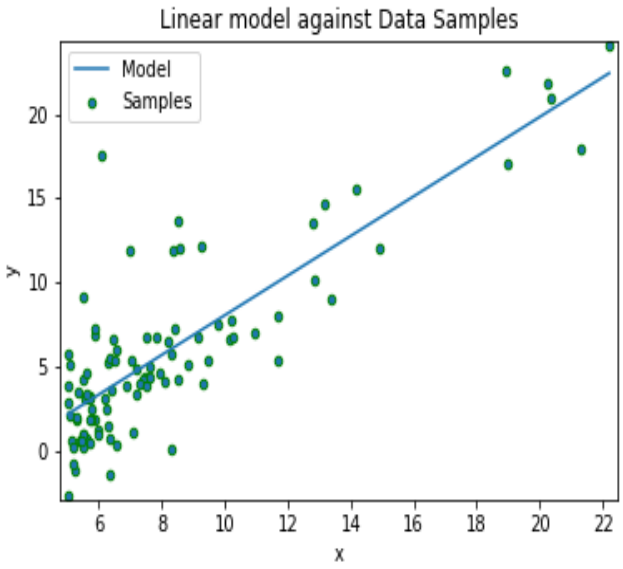


Figure 1. Dataset

Hyper-parameters used to generate the plot in Figure 2 are:

Learning rate = 0.02
Convergence = 0.00001
No. of Epoch = 950

Initially, when I started with setting up of hyer-parameters, I started with a learning rate of 0.01 and a fixed number of epochs around 1700. Then I tried the experiments by slightly differing the hyper-parameters and incrementing the learning rate roughly by a factor of 10 and decrementing the number of epochs in multiples of 20 until the point when the learning rate becomes too high causing the cost function to diverge during gradient descent.



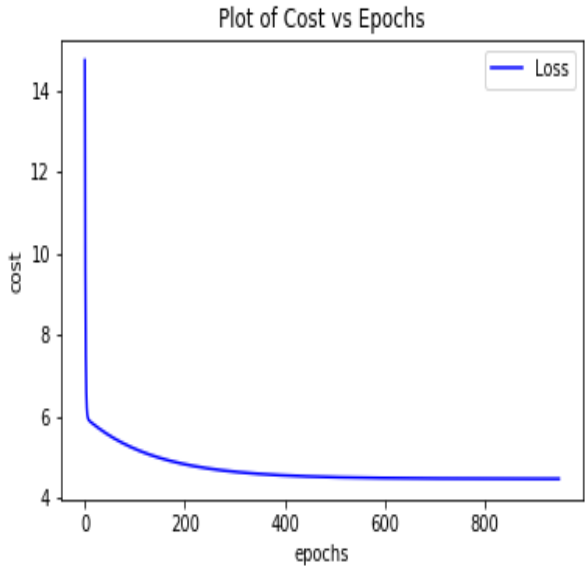


Figure 2. (a) Linear model against Data Samples *(b) Cost vs. Epoch*

Number of epochs required to converge to a reasonable solution was found to be around 10000 for any step size.
The table below(Figure 3) shows the varying step sizes and corresponding number of epochs required for convergence and final loss values.

Step Size (alpha)	Convergence (eps)	No. of Epochs	Final Loss Value
0.0001	0.00001	9980	5.48098618238
0.001	0.00001	9000	4.53702711358
0.01	0.00001	1700	4.48052366318
0.015	0.00001	1200	4.47952660187
0.02	0.00001	950	4.47868080064
0.03	0.00001	1	51.3560224441

Figure 3. Changing step size (alpha)

When I increased the step-size, the number of epochs needed for convergence decreases. One of the reasons for this could be that now we are taking larger steps in the direction of gradient towards the local minima. However, the learning is unstable beyond a certain value for step size and the cost keeps on diverging with each epoch.

Problem 2: Polynomial Regression & Regularization

2.a. 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.

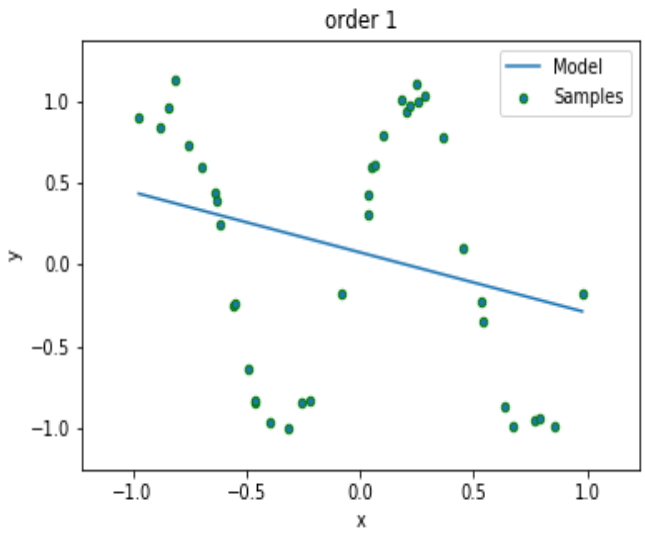
Sol. The feature mapping employed converted 1-D variable ‘x’ inti N-D vector x. This is polynomial mapping scheme. Possible issues with this feature mapping could be:

a. Poor asymptotic properties of Polynomial models.

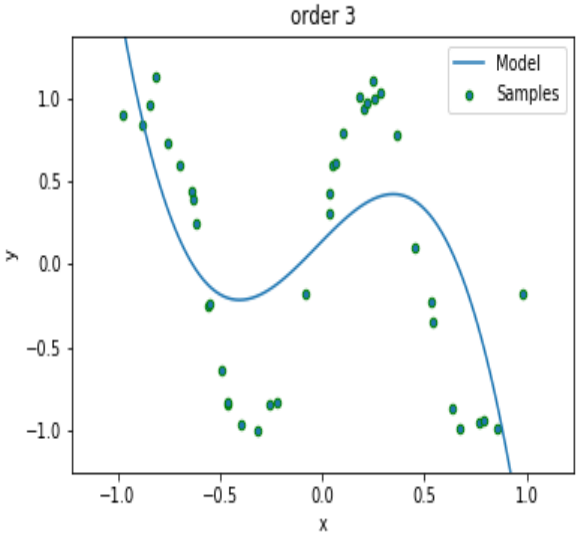
b. Poor Extrapolatory properties of polynomial models.
c. Poor Interpolatory properties of polynomial functions.

Moreover, using Fixed feature mapping, constrains us to a particular class of hypothesis functions, as it needs us to understand and interpret the properties of data and evaluate the fitness of a class of hypothesis functions before using such fixed feature mapping strategy.
One of the solutions to fix this is to normalize the input data to have a zero mean and unit variance which inturn will keep the range of the data in a small interval hence preventing higher order polynomials from blowing up.
Another solution to prevent coefficients from becoming extremely large is by adding some regularization term to the cost function. Potential problem here could be that now we have an extra meta-parameter to tune.
Another way could be to learn the features themselves. Here, we could generate a hierarchical set of features and train the entire system using regular feedforward neural network.

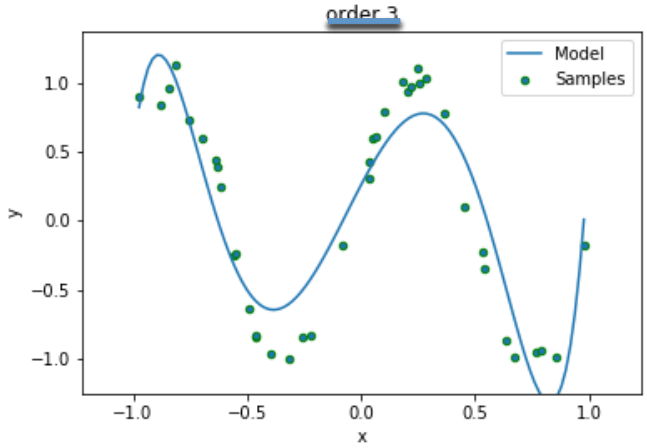
2.b. Fit a 1st order (i.e., linear), 3rd order, 7th order, 11th order, and 15th order polynomial regressor to the data in prob2_data.txt. What do you observe as the capacity of the model is increased? Why does this happen? Record your responses to these questions in your answer sheet.
Sol.



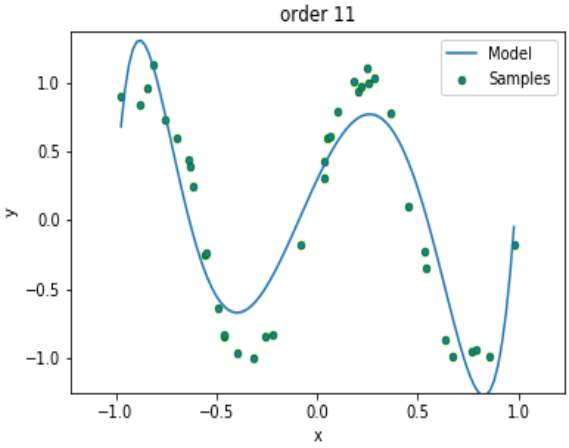
(a) 1st Order



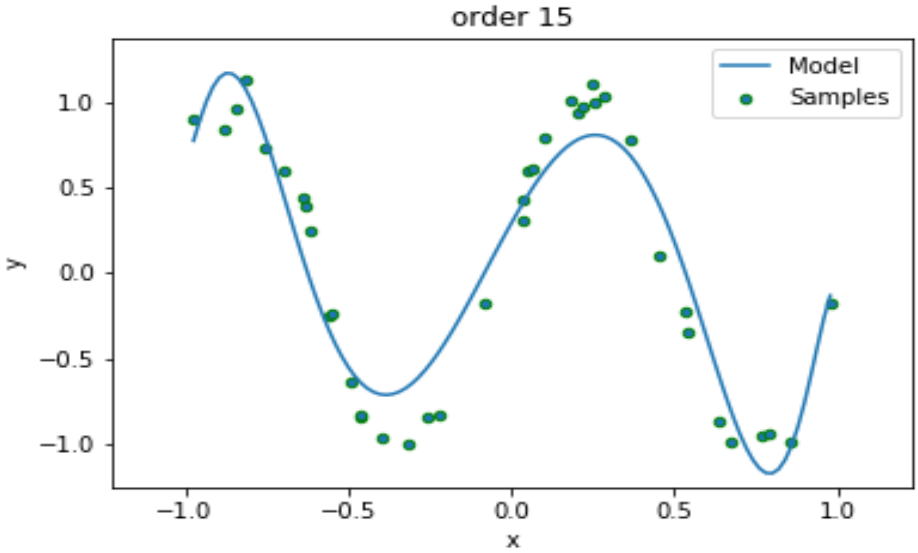
(b) 3rd Order



(c) 7th Order



(d) 11th Order



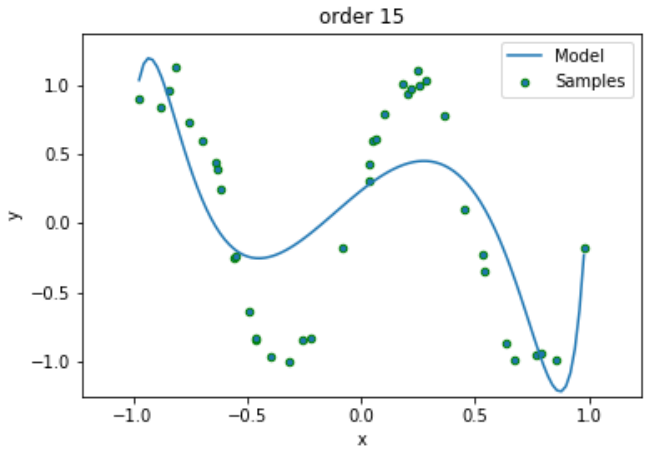
(e) Order 15th

Figure 4. Different order of Polynomial regression models

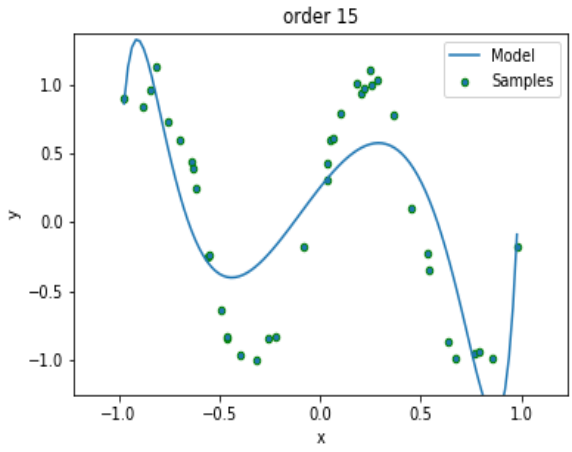
As the capacity of model is increased its ability to fit the data becomes much better. This is because we are increasing the number of parameters of the model that can be learnt during training. As the model has more number of parameters so it can handle the complexity of data distribution more efficiently.

2.c. 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)? Record this in your answer document.

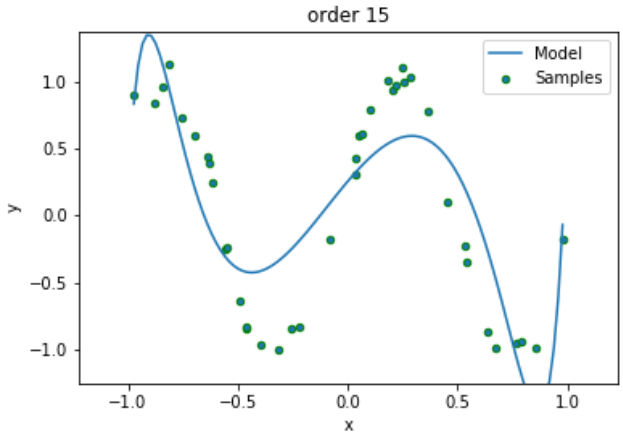
Sol.



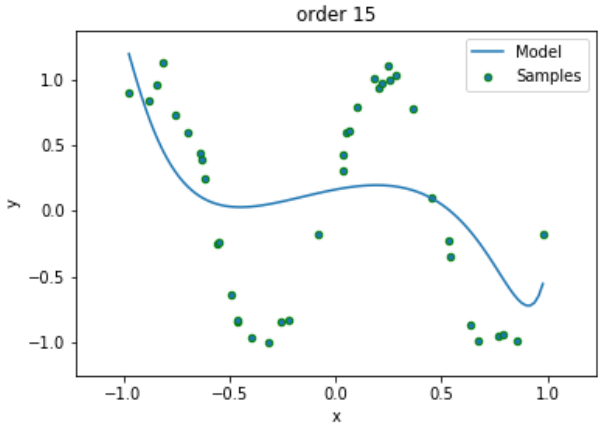
(a) beta 0.1



(b) beta 0.01



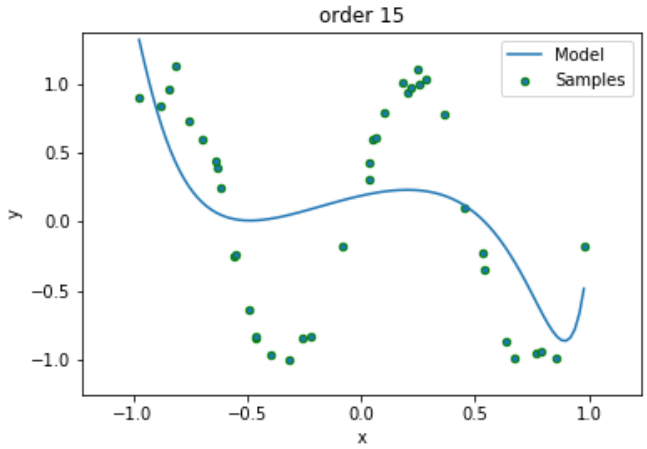
(c) beta 0.001



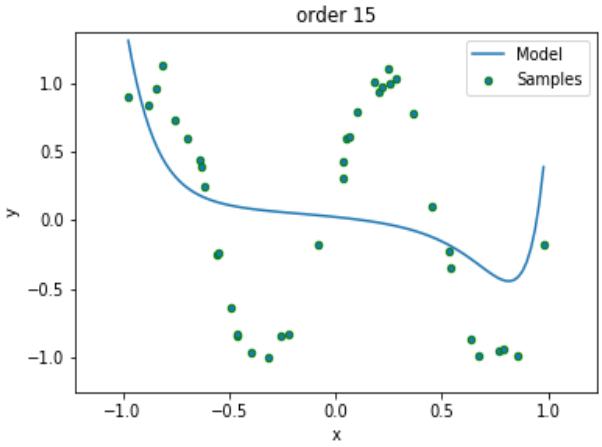
(d) beta 1.0

Figure 5. Different values of ***β*** *while keeping other parameters fixed*

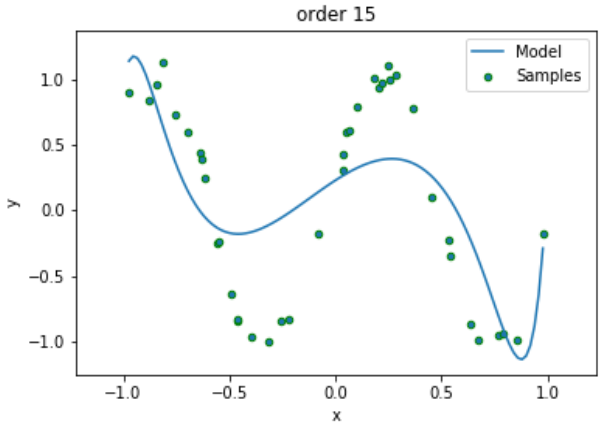
Increasing **β** reduces the epochs for convergence and subsequently the model starts underfitting. Increasing value of beta lowers the effective model capacity and hence the model has lesser free parameters to optimize, because of this the model converges faster.



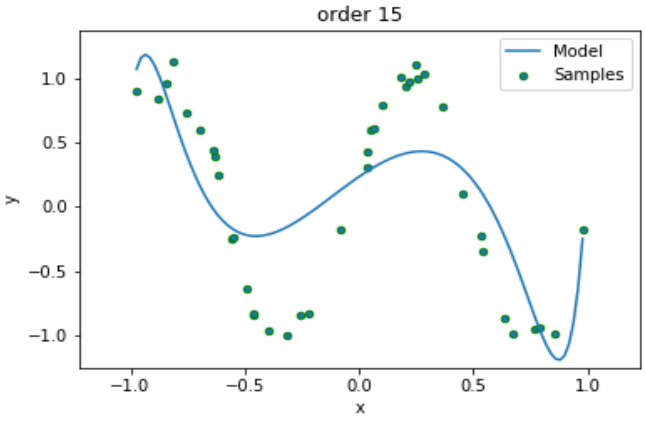
(a) alpha 0.1



(b) alpha 0.01



(c) alpha 0.5



(d) alpha 1.0

Figure 6. Different values of **α** *while keeping other parameters fixed*

The convergence is faster with greater value of step-size, as we are taking greater strides in the direction of the gradient, therefore reaching local minima is faster. As step size and number of epochs are inversely related so greater the step size, lower the number of epochs needed for convergence and vice versa.

Degree	Beta	Alpha	eps	n_epoch(converged at)	Loss
15	0.1	0.01	0.0001	3000 (584)	0.232454136
15	0.1	0.1	0.0001	3000 (330)	0.1748520135
15	0.1	1.0	0.0001	3000 (167)	0.1242050230
15	0.1	0.5	0.0001	3000 (248)	0.13051162660
15	0.1	1.56	0.0001	3000 (126)	0.1217425614

Figure 7. Relationship between alpha and beta and no. of epochs needed for convergence

2.d. 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?

Sol.

Degree	Beta	Alpha	eps	n_epoch	Loss
15	0.001	1.56	0.00001	2030	0.02732099033
15	0.01	1.56	0.00001	1100	0.0571036189
15	0.1	1.56	0.00001	300	0.116800397635
15	1.0	1.56	0.00001	100	0.204869151115

Figure 8. No. of epochs needed for convergence

It took around 2030 epochs to converge. The problem with this convergence check is that the error might increase after an iteration. Also if the loss gets stuck at a saddle point in local minima then maybe the loss might increase even more and then converge to a lower minima, as the loss tries to come out of the saddle point.

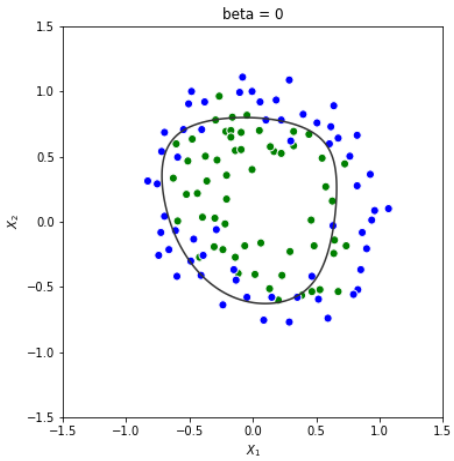
To fix this we can try running the model for as long as we can run keeping in mind the time and

computation costs. Also by plotting validation vs training loss gives us a rough idea about the model having high bias or high variance.

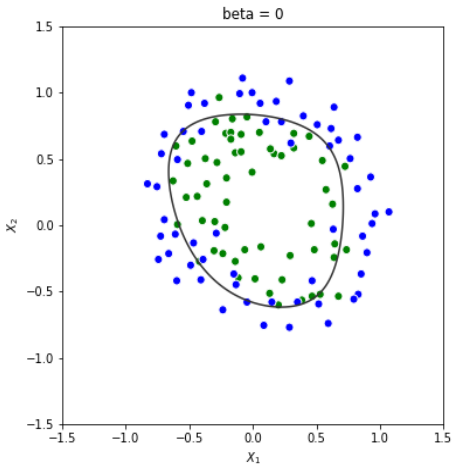
Problem 3: Multivariate Regression & Decision Boundaries

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?

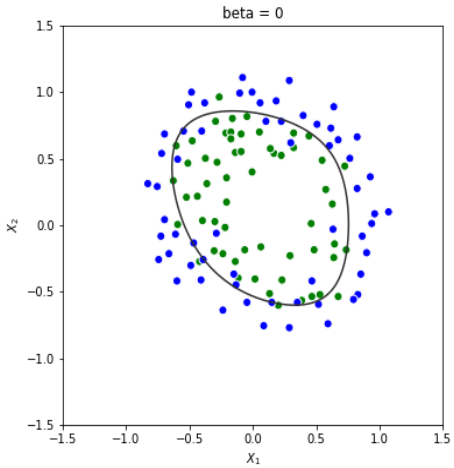
Sol. I tried this problem with different settings of alpha and beta. For the first part I kept changing alpha while keeping beta fixed at 0. For the other part I changed beta while keeping alpha fixed at 1.5.



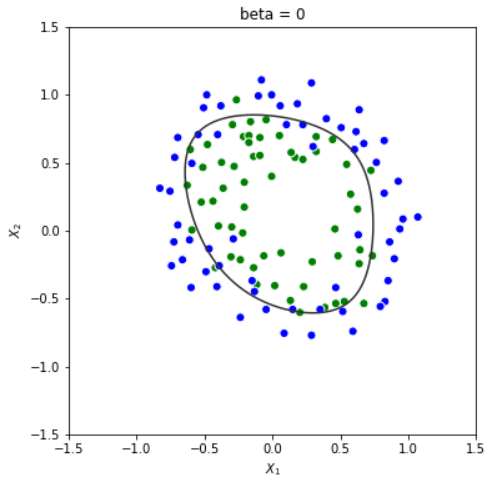
(a) alpha 0.1



(b) alpha 0.5



(c) alpha 1.5



(d) alpha 1

Figure 9. Changing alpha while keeping beta 0

Degree	Beta	Alpha	eps	n_epoch	Error	Loss
6	0	0.1	0.00001	500	22.0338	0.538364426
6	0	0.5	0.00001	500	17.7966	0.46280317355
6	0	1.0	0.00001	500	18.64406	0.4030636709
6	0	1.5	0.00001	500	16.10169	0.37813436411

Figure 10. Changing the value of alpha while keeping beta 0

Regularization helps to prevent the model from over-fitting itself to training data. It simplifies the model and hence reduces variance.
The regularization used in our case, creates a spherical ball of tension, here the parameters go farther away from the origin and the tension increases in proportion to the squared radius and creates a pulling force that keeps the parameters from growing too large. As it is clear from the figure as well, and as we see by setting beta to 100, if the regularization strength is too large it oversimplifies the model thereby resulting in poor data fitting. In our case, regularizing our model would be a good idea as it helps to reduce the model capacity and in turn helps the model to generalize better to unseen data and will help to balance out bias and variance.

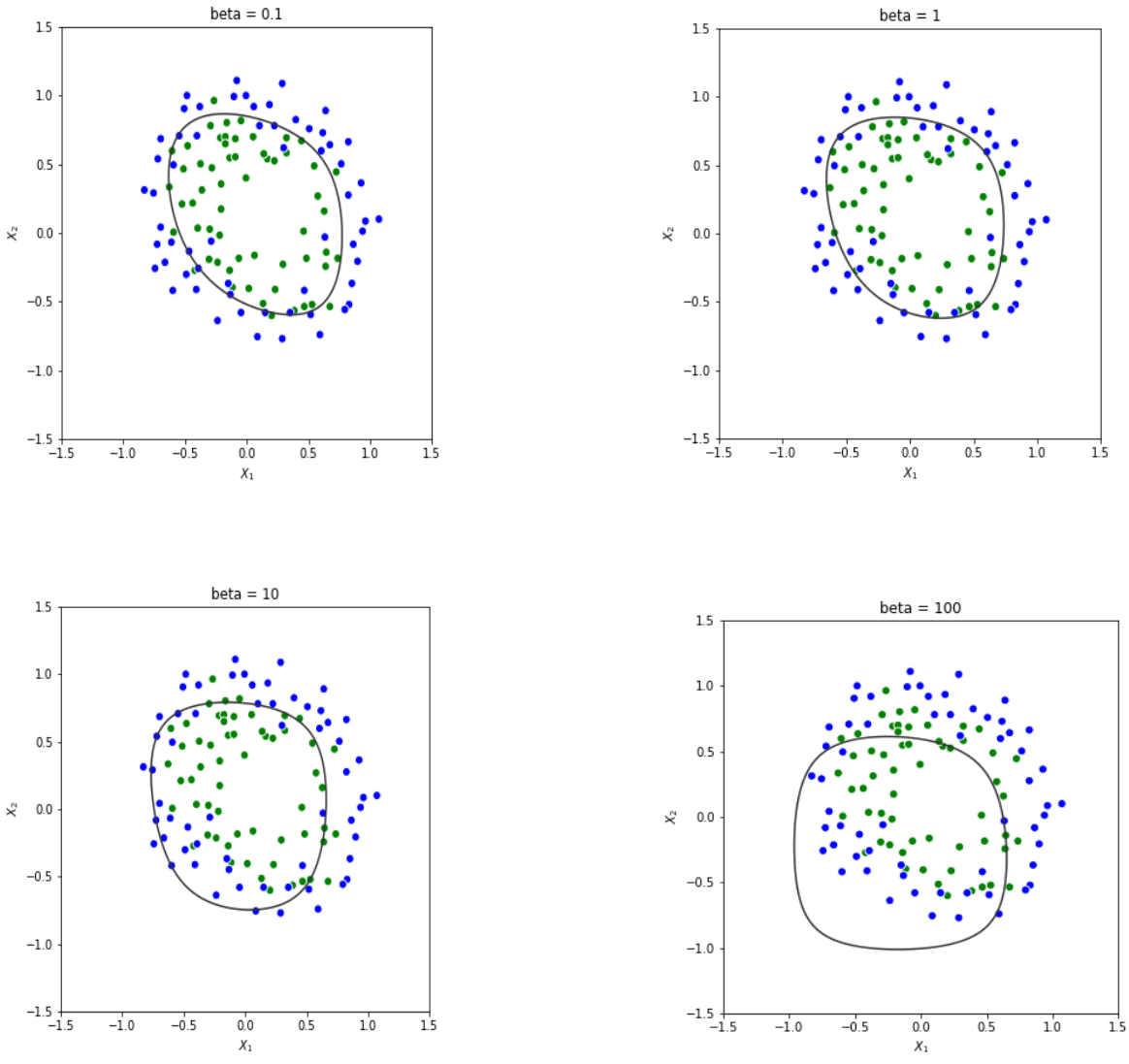
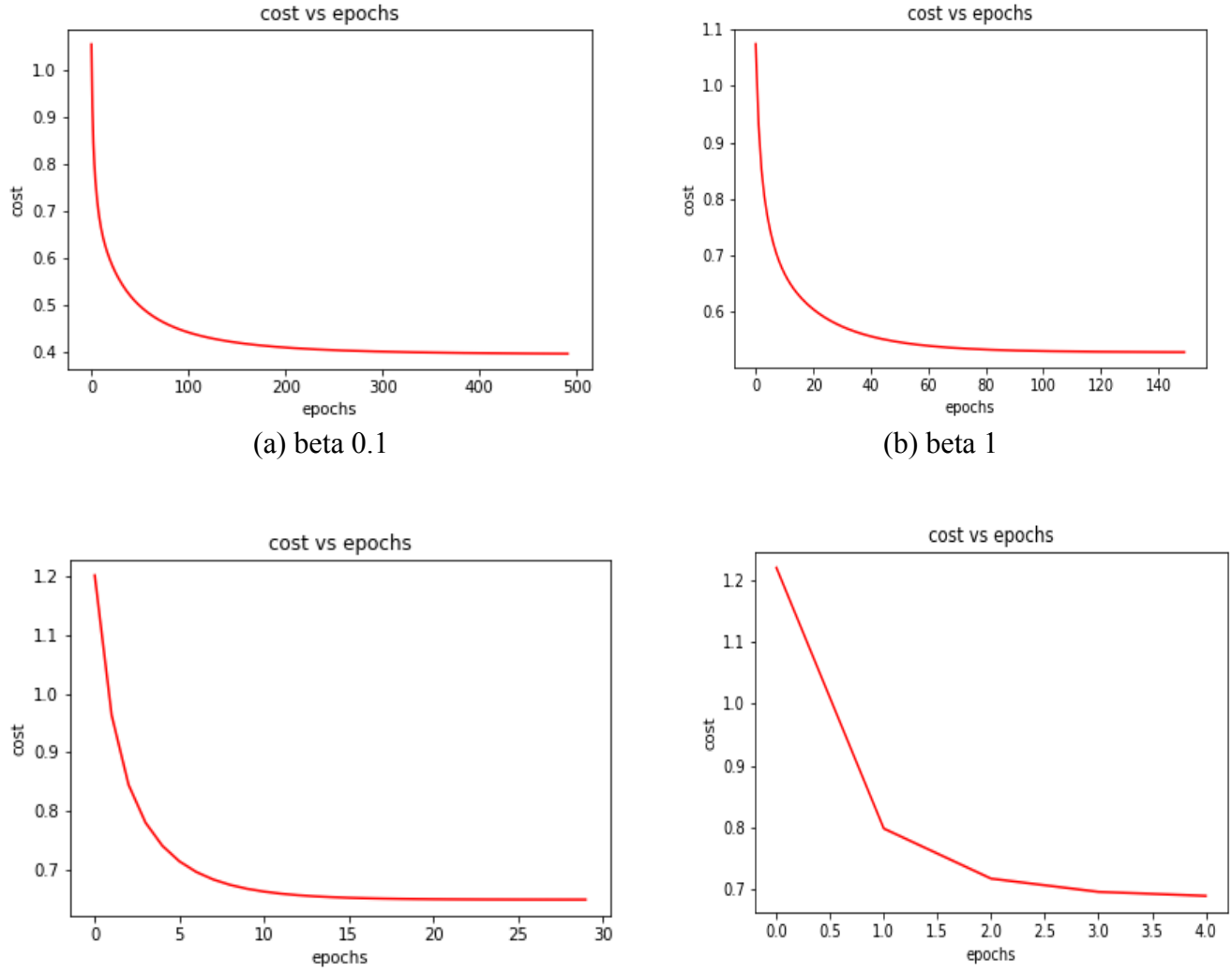


Figure 11. Varying Beta

Degree	Beta	Alpha	eps	n_epoch	Error	Loss
6	0.1	1.5	0.00001	500	16.1016	0.396431315538
6	1.0	1.5	0.00001	150	16.9491	0.529376478535
6	10	1.5	0.00001	30	26.2711	0.648274831612
6	100	1.5	0.00001	5	38.9830	0.68979204096

Figure 12. Varying Beta while keeping Alpha constant



(a) beta 0.1

(b) beta 1

(c) beta 10

(d) beta 100

Figure 13. Loss Vs. Epoch for different values of Beta