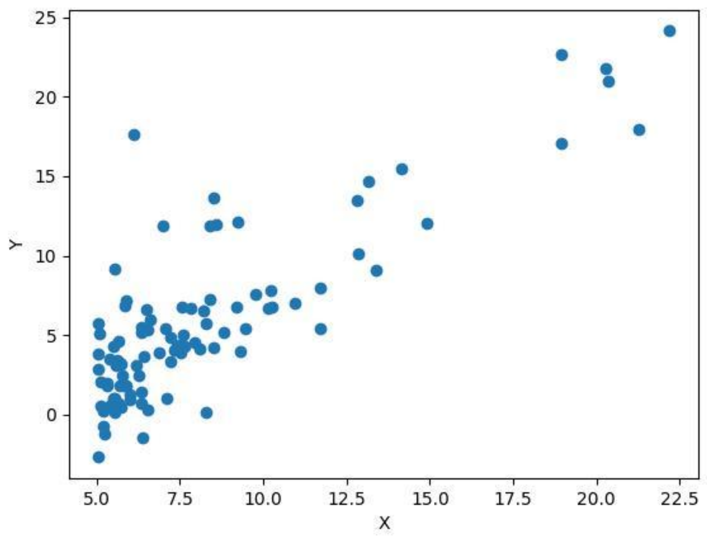
CSCI 635

HW2

Prakhar Gupta(pg9349)

Prob 1:



Scatter plot of the data for the food truck.

Descriptive statistics of the data is as follows-

```
Summary of the Data

================================================

            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
```

Q1:

You should record the settings you tried and write/discuss what you found that worked also in the answer sheet?

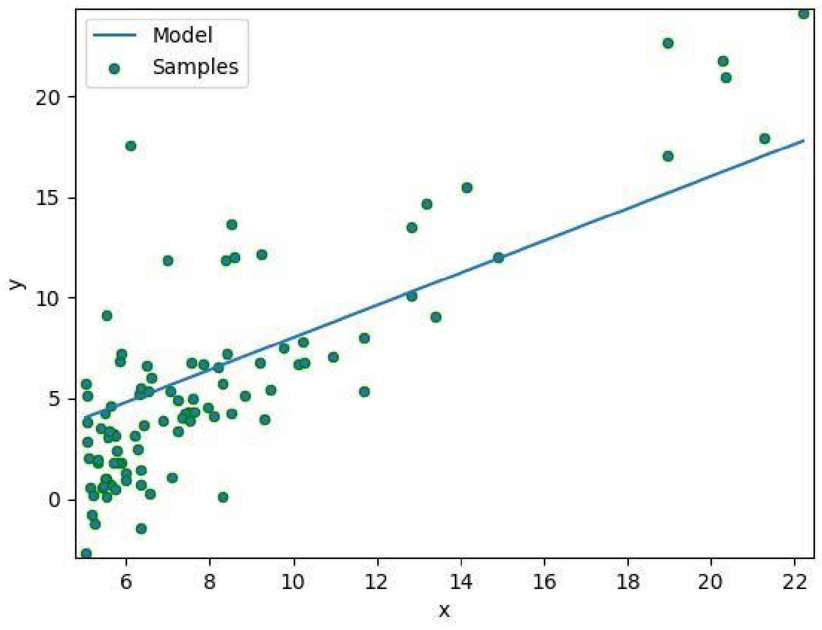
In total 5 Experiments were done results of 3 are given below

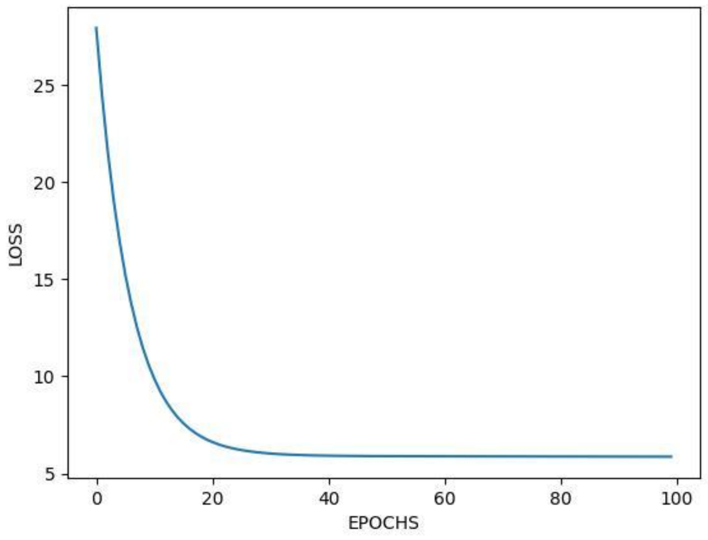
Exp 1

First run with lower learning rate and low epochs

alpha = 0.001 # step size coefficient

n_epoch = 100 # number of epochs (full passes through the dataset)





w = [[0.80063674]]

b = [0.00868909]

Loss at nth epoch-5.864868428785698

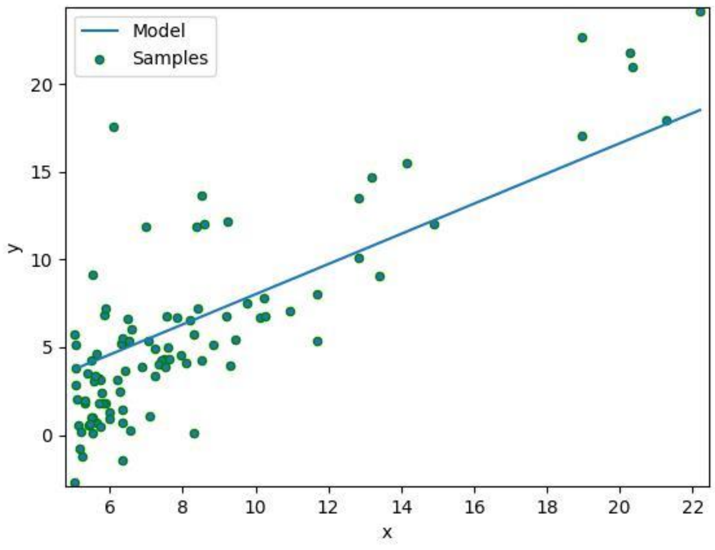
Model seemed to underfit the data.

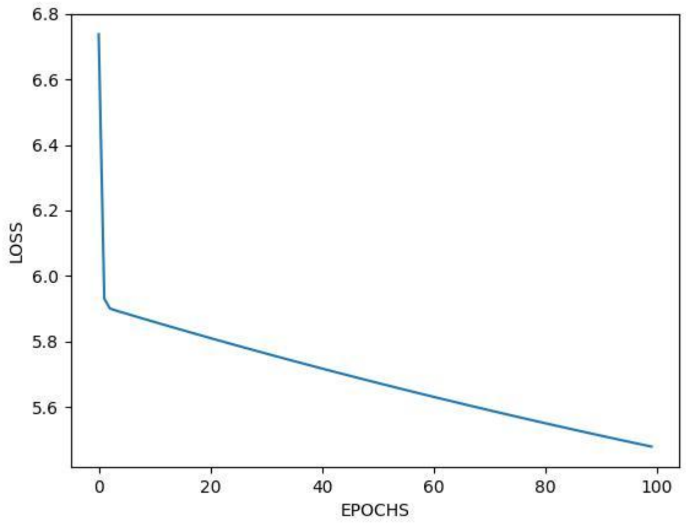
Exp 2

I then tried with increasing the learning rate and keeping epochs constant

alpha = 0.01 # step size coefficient

n_epoch = 100 # number of epochs (full passes through the dataset)





w = [[0.85958153]]

b = [-0.57655623]

Loss at nth epoch-5.479

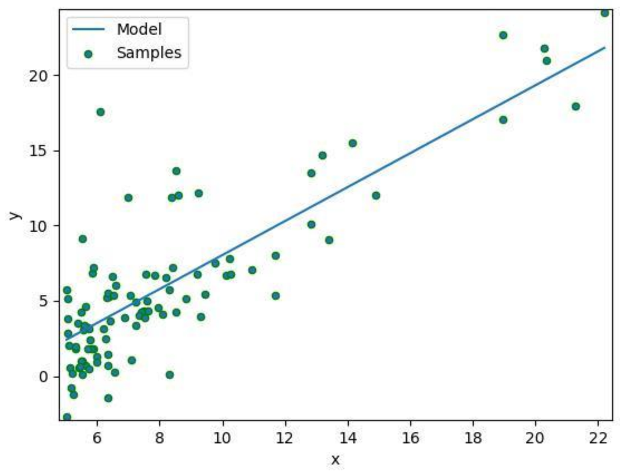
Model seemed to underfit the data. I can see that the loss hasn't flatlined so perhaps we can try with increasing epochs

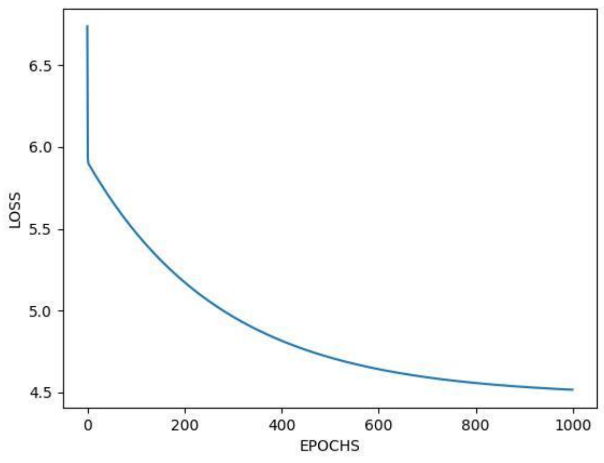
Exp 3(Final)

Model with increased epochs to 1000

alpha = 0.01 # step size coefficient

n_epoch = 1000 # number of epochs (full passes through the dataset)





w = [[1.1272942]]

b = [-3.24140214]

Loss at nth epoch-4.515955503078914

This models seems to fits better and training loss also appears to flatline as we reach 1000 epochs

Q2:

Write a few sentences describing what you learned from the training/model fitting process.

One thing is that hyper parameters are the key when it comes to model tuning or fitting. Also no one can tell in advance which hyper parameters will work better and it all depends on the problem and also the data.

One thing that appeared to work on this data was moderate learning rate of 0.01 and epoch of 1000 as the model appears to fit well based on the plots shown above.

Q3:

Things to discuss: What happens when you change the step-size α? How many epochs did

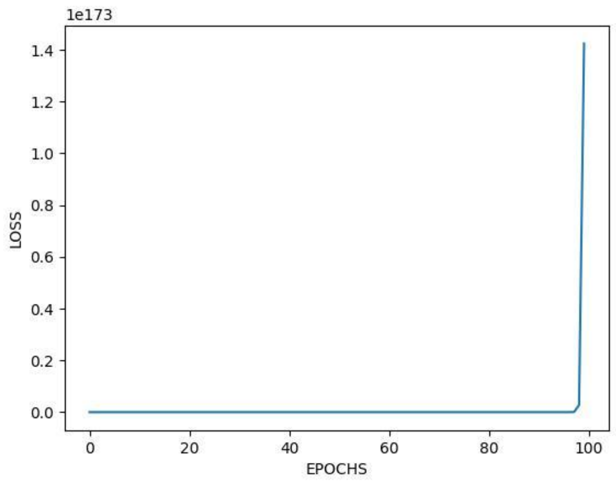
you need to converge to a reasonable solution (for any given step size?

I ran the experiment by increasing the alpha to 0.1 and lowering it to 0.0001 to compare results.

Exp 1

alpha = 0.1 # step size coefficient

n_epoch = 100



Loss at nth epochL = 1.4240099121234543e+173

w = [[-5.85588437e+85]]

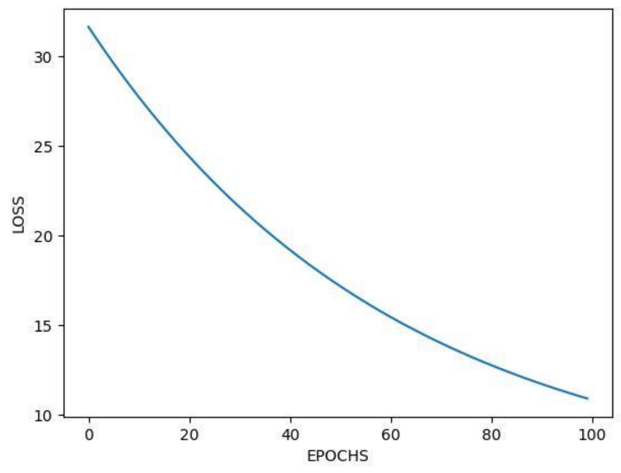
b = [-5.88287103e+84]

Model doesn’t even converge, instead we skip the local minima and the loss increases drastically. Loss is blown up

Exp 2:

alpha = 0.0001 # step size coefficient

n_epoch = 100



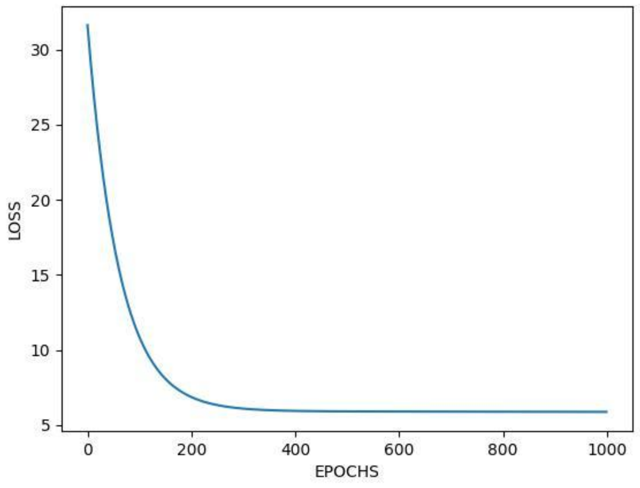
Loss at nth epoch = 10.927691822782235

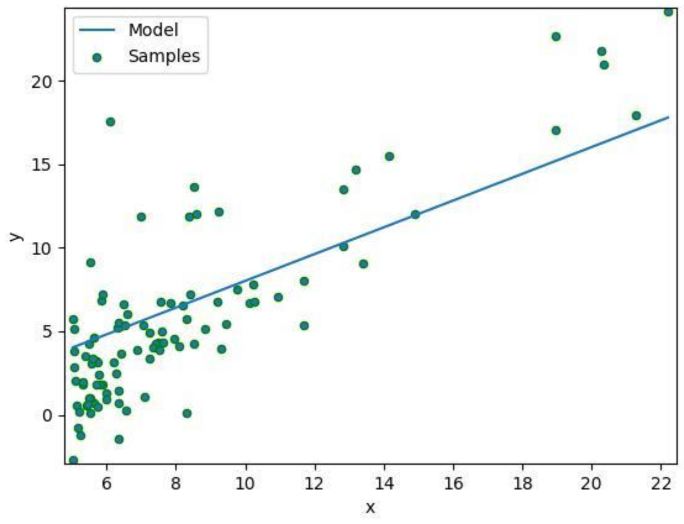
w = [[0.44678537]]

b = [0.03765233]

The loss very slowly decreases which is expected as the learning rate is low

Ran a follow up experiment with 1000 epochs to see the difference





Loss at nth epoch = 5.8648733284436645

w = [[0.80057917]]

b = [0.00868908]

Models still under fits.However when we used 0.01 as our learning rate the model was able to fit well.

Prob 2:

Q1:

With respect to the feature mapping, what is a potential problem that the scheme we have

designed above might create?

Since we created features which will be correlated with each other we will have collinearity with our feature. Also we have like 40 data points which is comparatively very less to perform and a good ML approach on it. Hence we might run into underfitting if we regularize or overfit to 40 data points if we run it for high epochs,

Also we can't keep on taking more and more features as soon we will face the **curse of dimensionality** which will slow down our training.

Q2:

What is one solution to fixing any potential problems created by using this scheme (and

what other problems might that solution induce)?

One of the solutions is always to get more data. Also, we can have more useful features rather than fitting a polynomial regression on just one feature and taking powers of the same we can take more features and mix and match.

If we have only one feature like in the problem 2 above we can perform a linear dimensionality reduction technique like PCA or LDA. However for our use case(since we are taking powers) we can also try nonlinear techniques like kernelPCA and Laplacian EigenMaps to reduce or capture those non linear variations across dimensions.

Problem with approach one is that getting more data is not always viable. Problem with option 2 is that maybe one of the features which we are using is important and others are not that relevant as all dimensionality techniques mentioned above tries to capture the axis of maximum variation, picking that axis and ignoring other dims may lead us to an under-fitted model.

Fitting multi degree polynomials regressors

Degree 1

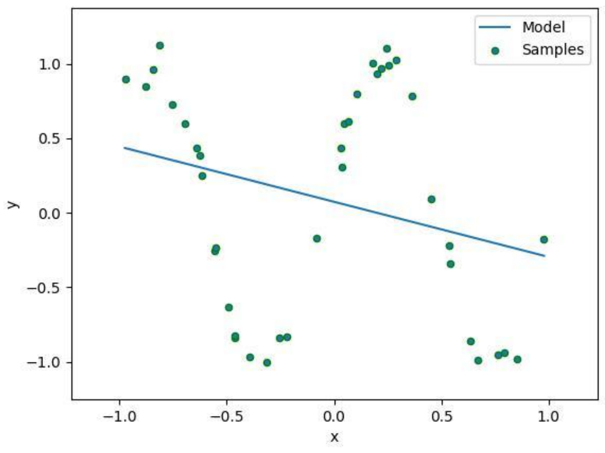
degree = 1 # p, order of model

beta = 0.0 # regularization coefficient

alpha = 0.05 # step size coefficient

eps = 0 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



epoch=1035 loss = 0.26635756457403986

w = [[-0.37043756]]

b = [0.0744876]

Model initial epochs set at 10000

Convergence happened at epoch 1035

Degree 3

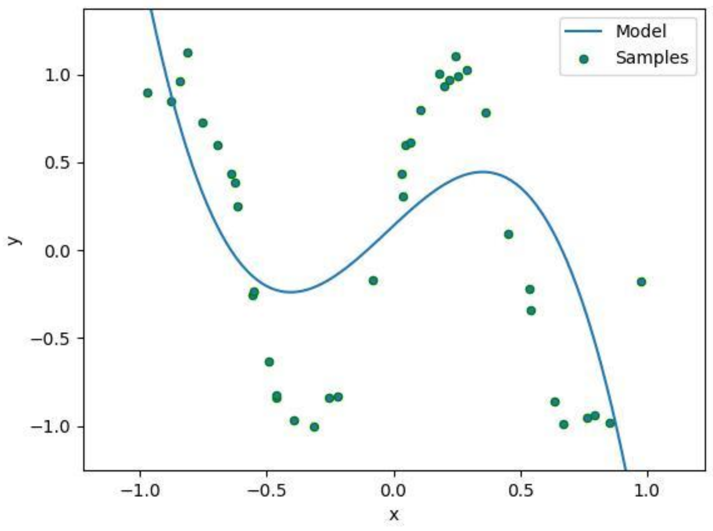
degree = 3 # p, order of model

beta = 0.0 # regularization coefficient

alpha = 0.05 # step size coefficient

eps = 0 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



9999 L = 0.165043840978074 73

w = [[1.34738144 -0.24957383 -3.14244643]]

b = [0.13937988]

Degree 7

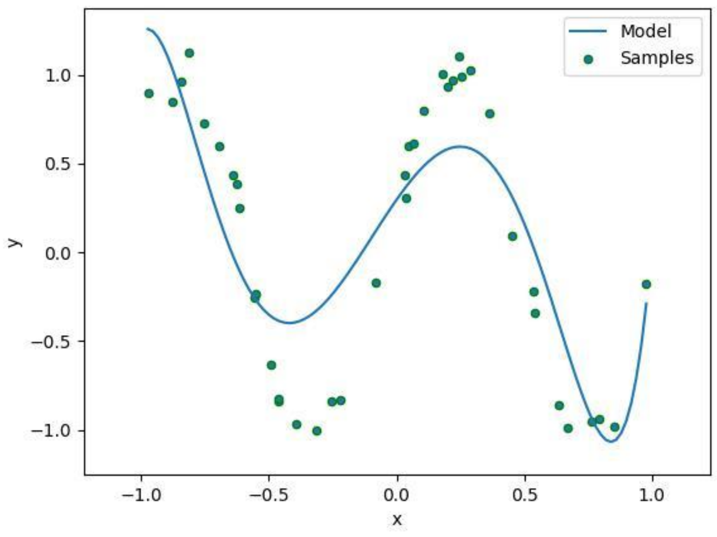
degree = 7 # p, order of model

beta = 0.0 # regularization coefficient

alpha = 0.05 # step size coefficient

eps = 0 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



```
w =  [[ 2.05560636 -1.91302133 -6.38280902  1.11691111 -0.11818298  1.13988775
   3.84970886]]
b =  [0.29649948]
```

Degree 11

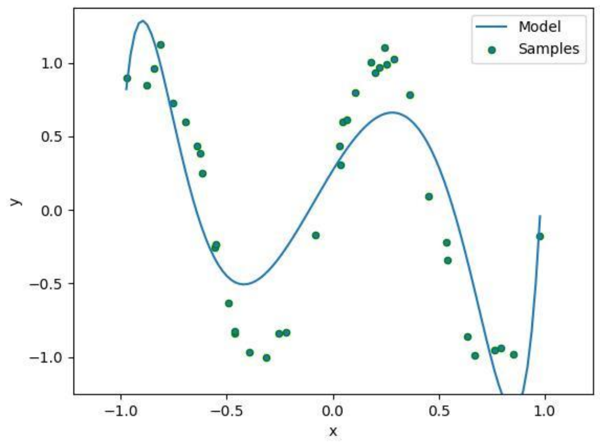
degree = 11 # p, order of model

beta = 0.0 # regularization coefficient

alpha = 0.05 # step size coefficient

eps = 0 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



```
 Epoch 9999 L = 0.058472792636457493

w =  [[ 2.33764739 -1.67375809 -6.0707797   0.96216309 -1.64766384  0.78925576

   0.92632274  0.26509459  2.07804061 -0.12783161  2.52716707]]

b =  [0.26846]
```

Degree 15

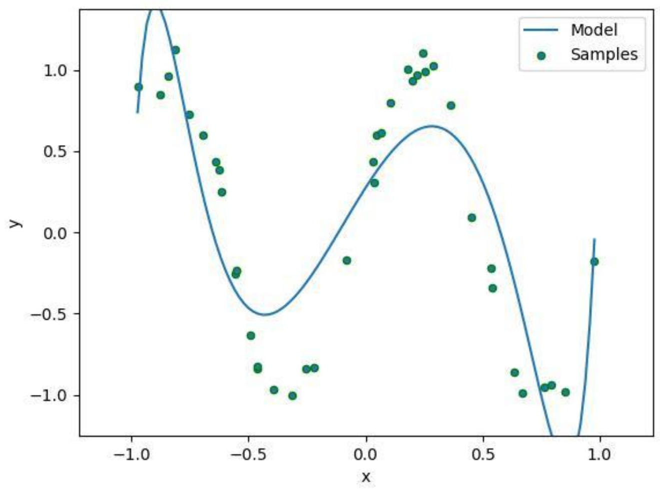
degree = 15 # p, order of model

beta = 0.0 # regularization coefficient

alpha = 0.05 # step size coefficient

eps = 0 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



```
 Epoch 9999 L = 0.06545166182463676

w =  [[ 2.25332649 -1.7007314  -5.55463987  0.94284267 -1.63485557  0.83947338

   0.40457276  0.38557801  1.15269078  0.04749785  1.32861541 -0.15279647

   1.28066782 -0.25814668  1.15856052]]

b =  [0.27437645]
```

Q1:

What do you observe as the capacity of the model is increased?

As the capacity of the model increases from linear we are now able to capture the non linear patterns in the data.However as we reach beyond 7 degree there is almost no change in the regression line. Also for lower degree loss quickly flattens however for higher degree we are able to run our model for higher epochs.

Q2:

Why does this happen?

Since our data has a polynomial nature a polynomial regression model is able to capture the essence better. But once we reach a good enough degree for example 7 we don't get any benefits of increasing the degree.

Regularisation

Beta =0.001

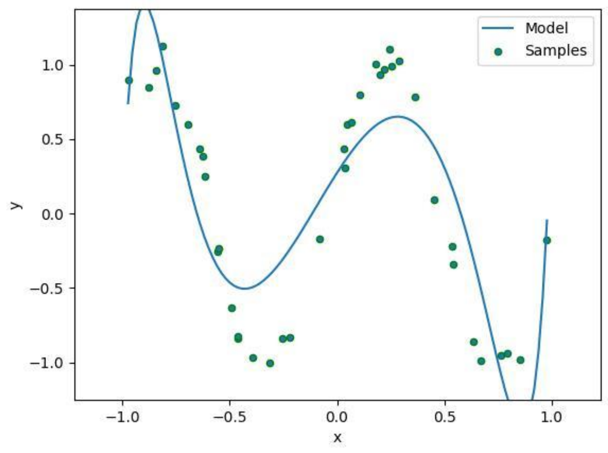
degree = 15 # p, order of model

beta = 0.001 # regularization coefficient

alpha = 0.05 # step size coefficient

eps = 0 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



```
Epoch 9999 L = 0.06558635635060857

w =  [[ 2.24481439 -1.69580738 -5.53263442  0.9371693  -1.63291049  0.83600835

   0.39903779  0.38509289  1.14637756  0.04888487  1.32382028 -0.15046362

   1.27785358 -0.25543206  1.15754451]]

b =  [0.27407711]
```

Beta =0.01

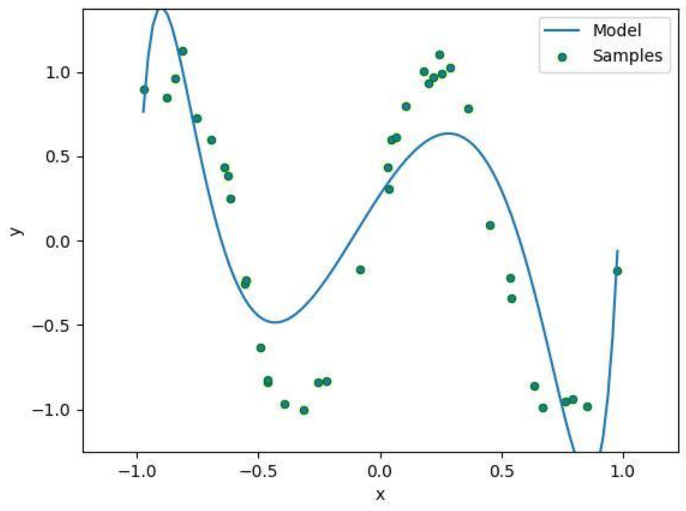
```
degree = 15 # p, order of model

beta = 0.01 # regularization coefficient

alpha = 0.05 # step size coefficient

eps = 0 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)
```

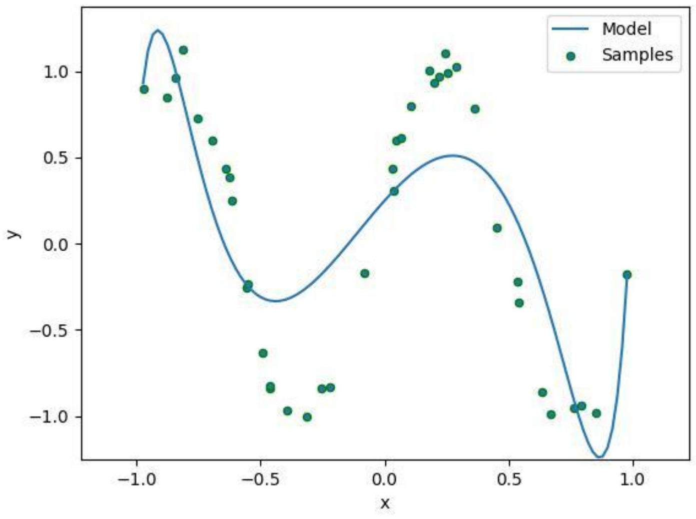


```
Epoch 9999 L = 0.0668767821136696
```

```
w =  [[ 2.17053766 -1.65289758 -5.34117387  0.88803936 -1.61524422  0.80590195
   0.35146463  0.38078831  1.09161425  0.06082604  1.28185596 -0.13029016
   1.25279942 -0.2319144   1.14790132]]
b =  [0.27144119]
```

Beta =0.1

```
degree = 15 # p, order of model
beta = 0.1 # regularization coefficient
alpha = 0.05 # step size coefficient
eps = 0 # controls convergence criterion
n_epoch = 10000 # number of epochs (full passes through the dataset)
```

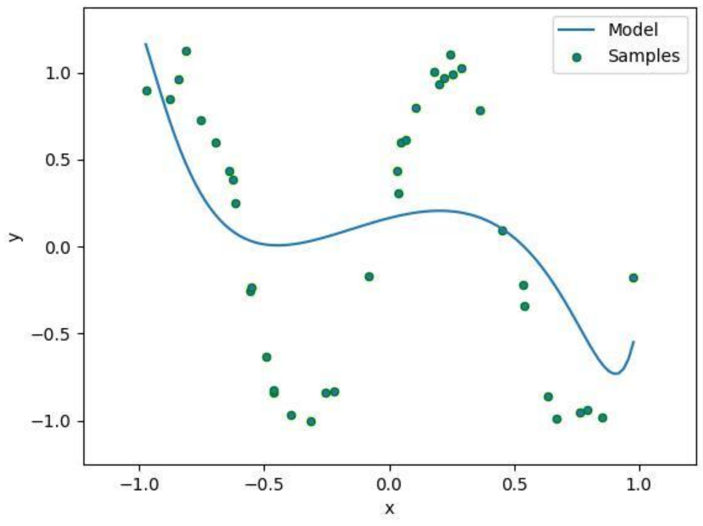


```
Epoch 9999 L = 0.08355936706500683
w =  [[ 1.60886162 -1.33113584 -3.93061876  0.54214917 -1.43544019  0.58686826
```

```
  0.04008435  0.34309352  0.69931092  0.13970905  0.95652143  0.00929819
  1.03035654 -0.06626911  1.02332439]]
b =  [0.24965779]
```

Beta =1

```
degree = 15 # p, order of model
beta = 1 # regularization coefficient
alpha = 0.05 # step size coefficient
eps = 0 # controls convergence criterion
n_epoch = 10000 # number of epochs (full passes through the dataset)
```



```
Epoch 9999 L = 0.17697835300508225
w =  [[ 0.36670931 -0.55144412 -1.14379882  0.04249541 -0.62791031  0.17429548
```

```
 -0.2134317   0.18241105  0.03670677  0.16474935  0.18133884  0.14514609
  0.26350969  0.12863441  0.30861744]]
b =  [0.16406763]
```

Q1:

What do you observe as you increase the value of β?

As the value of B increases we see that overfitting is reduced. With low beta value we have a line which fits the data kind of perfectly but as we increase beta the line tends to regularize more.

Q2:

How does this interact with the general model fitting process (such as the step size α and number of epochs needed to reach convergence)?

4 Experiments below are when we change the value of beta between 0.1 and 0.001 we change alpha between 0.01 and 0.1

Exp1

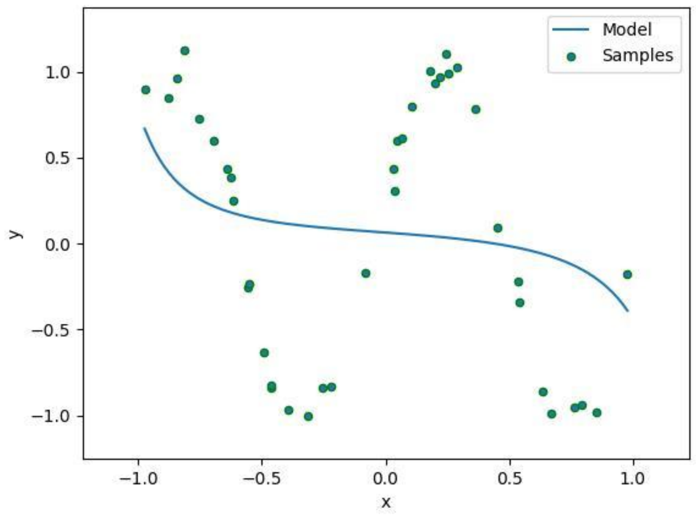
degree = 15 # p, order of model

beta = 0.1 # regularization coefficient

alpha = 0.01 # step size coefficient

eps = 0.0001 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



Epochs 167 L = 0.2479465766788433

w = [[-0.1058768 -0.01516522 -0.15436342 0.01439467 -0.11154481 0.02212275

-0.07843149 0.02281615 -0.05666116 0.02161231 -0.04238454 0.02000056

-0.03280953 0.01841394 -0.02621531]]

b = [0.06489656]

Model initial epochs set at 10000

Convergence happened at epoch 167

Exp 2

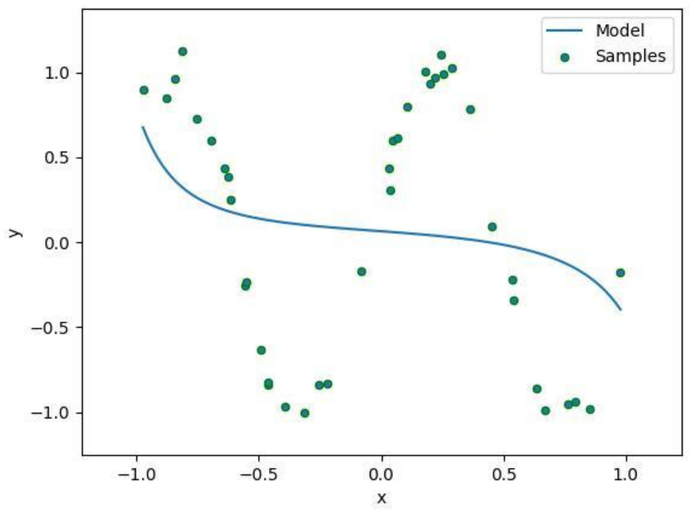
degree = 15 # p, order of model

beta = 0.001 # regularization coefficient

alpha = 0.01 # step size coefficient

eps = 0.0001 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



Epoch 170 L = 0.24728564620560767

w = [[-0.10682099 -0.01581246 -0.15666025 0.01446451 -0.11317841 0.02240733

  -0.07951626 0.02315014 -0.05738462 0.02194585 -0.04287458 0.02031874

  -0.03314677 0.01871305 -0.02645067]]

b = [0.06517939]

Model initial epochs set at 10000

Convergence happened at epoch 170

Exp3

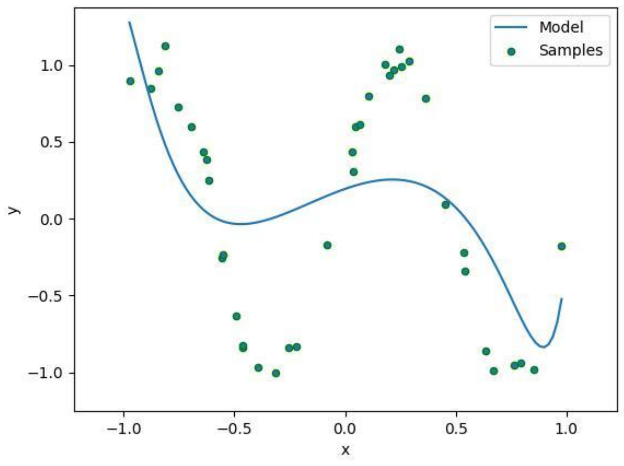
degree = 15 # p, order of model

beta = 0.1 # regularization coefficient

alpha = 0.1 # step size coefficient

eps = 0.0001 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



```
 Epoch 491 L = 0.15889130872371726

w =  [[ 0.50496085 -0.72220389 -1.39134454  0.02998023 -0.76845105  0.21477961

  -0.25420312  0.23852413  0.06143281  0.22431762  0.24646631  0.2040247

   0.35283434  0.18541074  0.41185356]]

b =  [0.19407596]

Model initial epochs set at  10000

Convergence happened at epoch  491
```

Exp 4

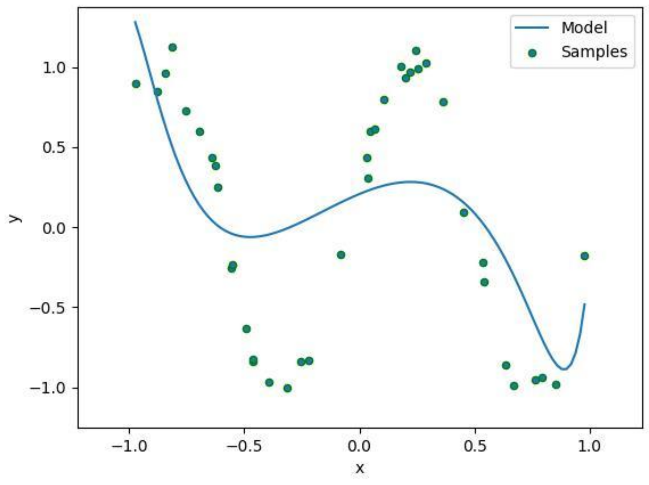
degree = 15 # p, order of model

beta = 0.001 # regularization coefficient

alpha = 0.1 # step size coefficient

eps = 0.0001 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)



```
Epoch  544 L = 0.14946646346883188

w =  [[ 0.59815728 -0.8052592  -1.57988762  0.04245694 -0.85560201  0.24232003

  -0.26334151  0.26233781  0.0976751   0.24213221  0.30781405  0.2169227

   0.4275838   0.19479376  0.49322088]]

b =  [0.20758531]

Model initial epochs set at  10000

Convergence happened at epoch  544
```

Conclusion

So with a higher learning rate and with lower beta value the model is able to run for longer epochs before halting occurs due to delta loss falling below the eps value. Also the loss is much lower for higher alpha and lower beta value which is expected

Bonus Part-

1.How many steps did it then took with this early halting scheme to reach convergence?

Tried with the following hyperparameters-

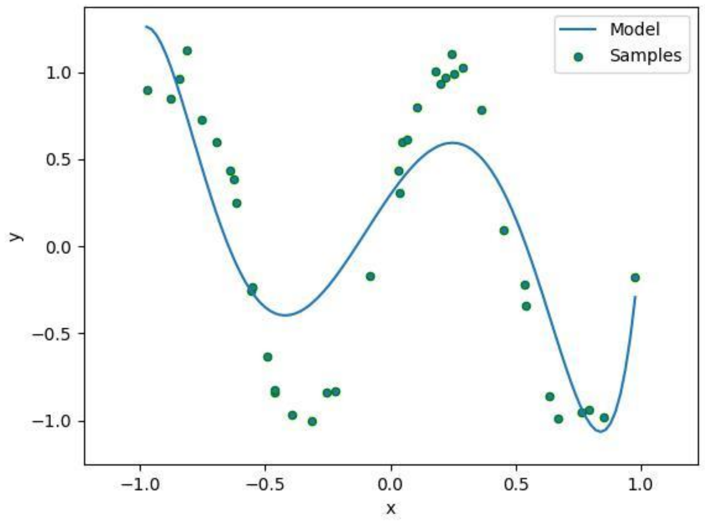
degree = 7 # p, order of model

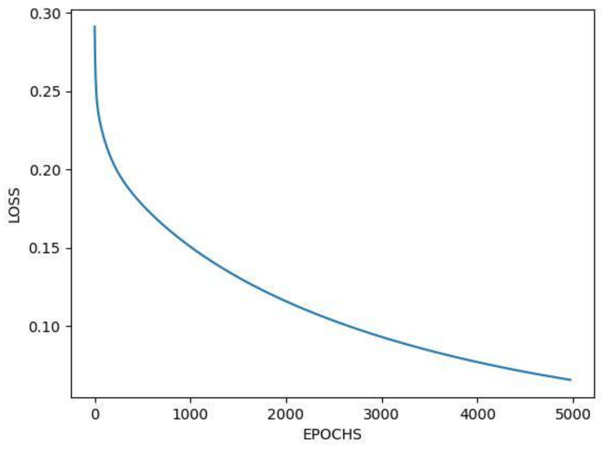
beta = 0.0001 # regularization coefficient

alpha = 0.1 # step size coefficient

eps = 0.00001 # controls convergence criterion

n_epoch = 10000 # number of epochs (full passes through the dataset)





 4970 Loss = 0.06561894658660723

w = [[2.04804942 -1.91177368 -6.35749146 1.11364996 -0.12142707 1.14203296

   3.83090794]]

b = [0.29659204]

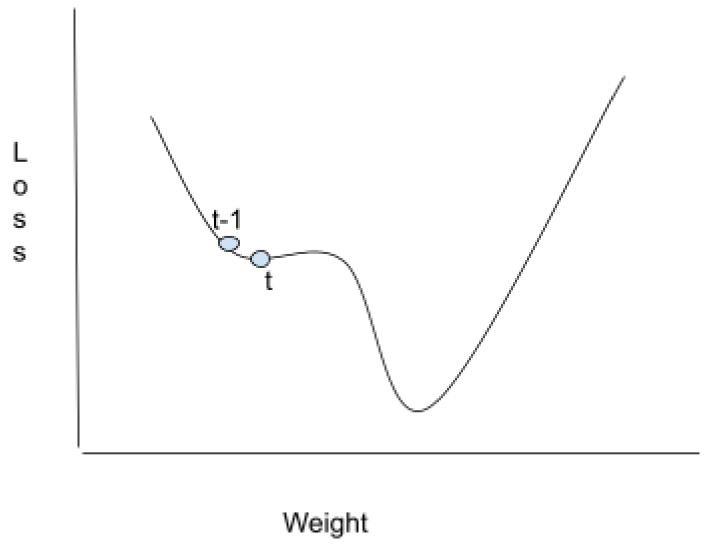
Model initial epochs set at 10000

Convergence happened at epoch 4970

So instead of running for full 10000 epochs convergence was achieved at 4970 epochs. With a eps of 0.00001

2. 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?

The logic of taking delta between cost at t-1 and t has a problem stated below-



So if there is a saddle point the model will terminate as the delta cost between t-1 and t would be very low however if we kept on training the model it might have reached the minima which we miss because we terminated the model earlier.

Prob 3:

Beta 0

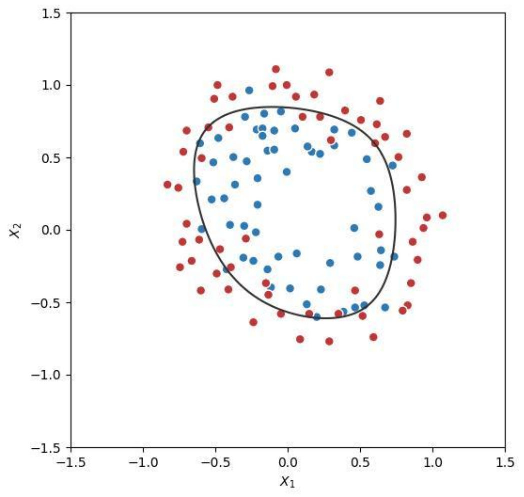
degree = 6 # p, degree of model (PLEASE LEAVE THIS FIXED TO p = 6 FOR THIS PROBLEM)

beta = 0 # regularization coefficient

alpha = 0.05 # step size coefficient

n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0.00001# controls convergence criterion



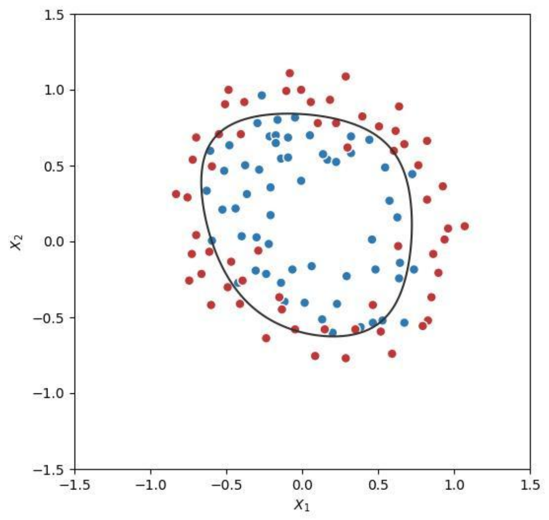
Epoch 7773 L = 0.4650343714477497

w = [[6.66450754e-01 1.22807498e+00 -1.93068897e+00 -8.67210518e-01

```
 -1.23548355e+00  1.41990333e-01 -3.58101106e-01 -3.40869552e-01
 -1.92405383e-01 -1.45987833e+00 -8.42018430e-02 -6.01168125e-01
 -2.43229514e-01 -1.21264686e+00 -2.30717322e-01 -2.15587657e-01
 -6.63109230e-02 -2.70038885e-01 -2.66829109e-01 -5.60589448e-01
 -1.05887947e+00  3.09789573e-03 -2.92982190e-01 -9.62387245e-05
 -3.25131826e-01 -1.15043147e-01 -1.04122139e+00]]
b =  [1.18989273]
Model initial epochs set at  10000
Convergence happened at epoch  7774
Accuracy = 82.20338983050848%
Error = 17.796610169491522%
```

Beta 0.1

```
degree = 6 # p, degree of model (PLEASE LEAVE THIS FIXED TO p = 6 FOR THIS PROBLEM)
beta = 0.1 # regularization coefficient
alpha = 0.05 # step size coefficient
n_epoch = 10000 # number of epochs (full passes through the dataset)
eps = 0.00001# controls convergence criterion
```



```
 Epoch 4594 L = 0.5401313754113924

w =  [[ 3.54296624e-01  7.91390012e-01 -1.32015231e+00 -4.77403319e-01

  -7.79318438e-01  2.36084580e-02 -2.08866661e-01 -2.05352052e-01

  -1.67055648e-01 -1.00125609e+00 -4.57258453e-02 -3.85875439e-01

  -1.24449504e-01 -8.14586844e-01 -2.00179126e-01 -1.31460065e-01

  -5.09022046e-02 -1.67199181e-01 -1.51761154e-01 -4.32339614e-01

  -7.31399130e-01  6.06905746e-04 -1.91163943e-01  8.96500327e-04

  -2.09253137e-01 -5.40500031e-02 -7.36624593e-01]]

b =  [0.8255859]

Model initial epochs set at  10000

Convergence happened at epoch  4595

Accuracy = 82.20338983050848%
```

Error = 17.796610169491522%

Beta 1

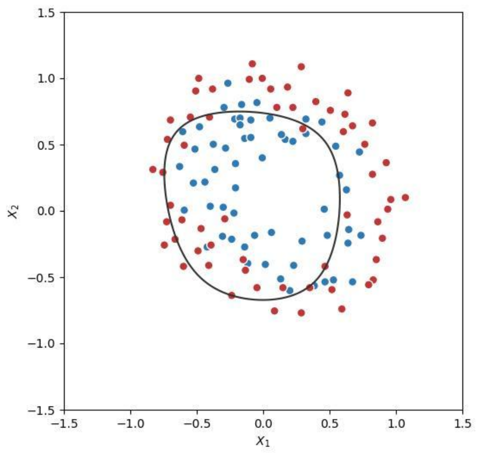
degree = 6 # p, degree of model (PLEASE LEAVE THIS FIXED TO p = 6 FOR THIS PROBLEM)

beta = 1 # regularization coefficient

alpha = 0.05 # step size coefficient

n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0.00001# controls convergence criterion



Epoch 864 L = 0.6541047820080598

w = [[-0.04093176 0.12838537 -0.33904311 -0.0918811 -0.19902363 -0.07594673

  -0.04507004 -0.05279419 -0.09467081 -0.2704595 -0.01251046 -0.09025232

  -0.02292722 -0.23211301 -0.10693991 -0.02942927 -0.01937368 -0.03907953

  -0.03231434 -0.16494331 -0.20924741 -0.00390844 -0.04574892 -0.00106329

  -0.05065503 -0.00942623 -0.23108509]]

b = [0.19360789]

Model initial epochs set at 10000

Convergence happened at epoch 865

Accuracy = 72.88135593220339%

Error = 27.118644067796616%

Beta 10

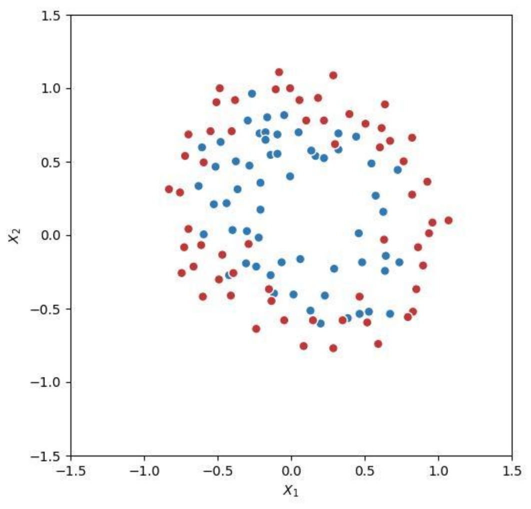
degree = 6 # p, degree of model (PLEASE LEAVE THIS FIXED TO p = 6 FOR THIS PROBLEM)

beta = 10 # regularization coefficient

alpha = 0.05 # step size coefficient

n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0.00001# controls convergence criterion

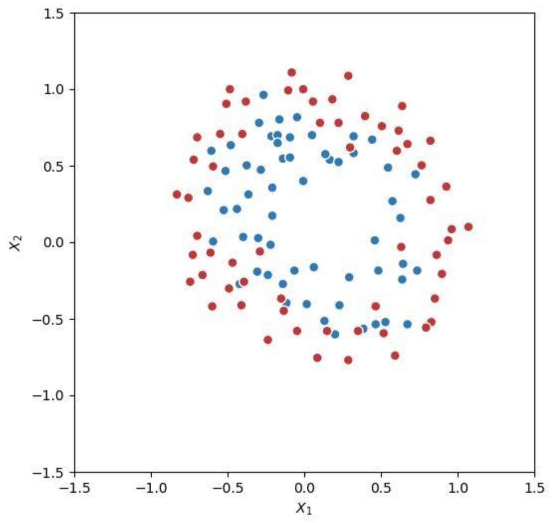


66 L = 0.6883684444001467

```
w =  [[-0.01253007  0.00171277 -0.03534741 -0.00833859 -0.02556029 -0.01255338
  -0.0050695  -0.00578225 -0.01569087 -0.02782799 -0.00157393 -0.00906184
  -0.00223941 -0.02720222 -0.01389557 -0.00304577 -0.00236208 -0.00409293
  -0.00320073 -0.0213141  -0.02195529 -0.00074921 -0.0044792  -0.00027719
  -0.00512464 -0.0009957  -0.02696936]]
b =  [-0.00252628]
Model initial epochs set at  10000
Convergence happened at epoch  67
Accuracy = 50.847457627118644%
Error = 49.152542372881356%
```

Beta 100

```
degree = 6 # p, degree of model (PLEASE LEAVE THIS FIXED TO p = 6 FOR THIS PROBLEM)
beta = 100 # regularization coefficient
alpha = 0.05 # step size coefficient
n_epoch = 10000 # number of epochs (full passes through the dataset)
eps = 0.00001# controls convergence criterion
```



```
 Epoch 7 L = 0.6926254719760093

w =  [[-1.60699784e-03  1.73172737e-05 -4.32856104e-03 -9.92605511e-04

  -3.22622243e-03 -1.57407566e-03 -6.28719128e-04 -7.04891333e-04

  -2.00788914e-03 -3.38605878e-03 -1.92617125e-04 -1.10612039e-03

  -2.67164352e-04 -3.37404099e-03 -1.71568240e-03 -3.72368947e-04

  -2.90991825e-04 -5.01889870e-04 -3.85716947e-04 -2.66011961e-03

  -2.67055174e-03 -9.41079649e-05 -5.43631397e-04 -3.50158587e-05

  -6.24973136e-04 -1.18808891e-04 -3.33210851e-03]]

b =  [-0.00078444]

Model initial epochs set at  10000

Convergence happened at epoch  8

Accuracy = 50.847457627118644%

Error = 49.152542372881356%
```

How does the regularization change your model's accuracy as well as its learned decision boundary?

For low regularization I was getting a crisp or a well defined boundary and high accuracy of 82.20 and low error of 17.79 this drastically changed accuracy kept on getting low as we increased the beta value. Also the decision boundary contained values of both classes and wasn't able to separate the two classes of data.

One interesting thing was that as the value of beta became 10 and 100 I was not able to get a decision boundary as all the values were predicted to be zeros hence no decision boundary was created in the plots.

Why might regularizing our model be a good idea if it changes what appears to be such a well-fit decision boundary?

Regularization is a good idea because the model might overfit on the training data and regularization tries to reduce this overfitting of the model. This will have positive impact because the unseen data might not be same a train and there is always variation in it so regularization accounts for the same.