ML Bootcamp

IIT ISM Dhanbad

By Anany Garg
(22JE0111)

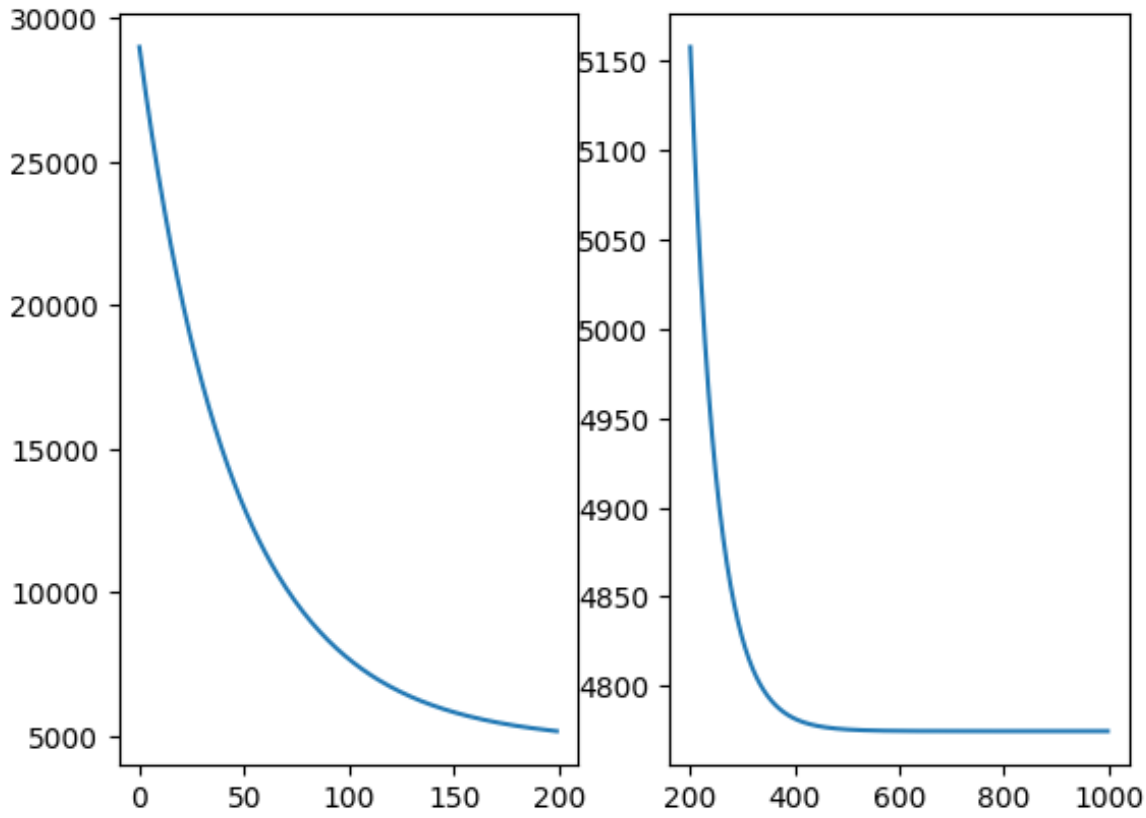
Report

1. Linear regression:

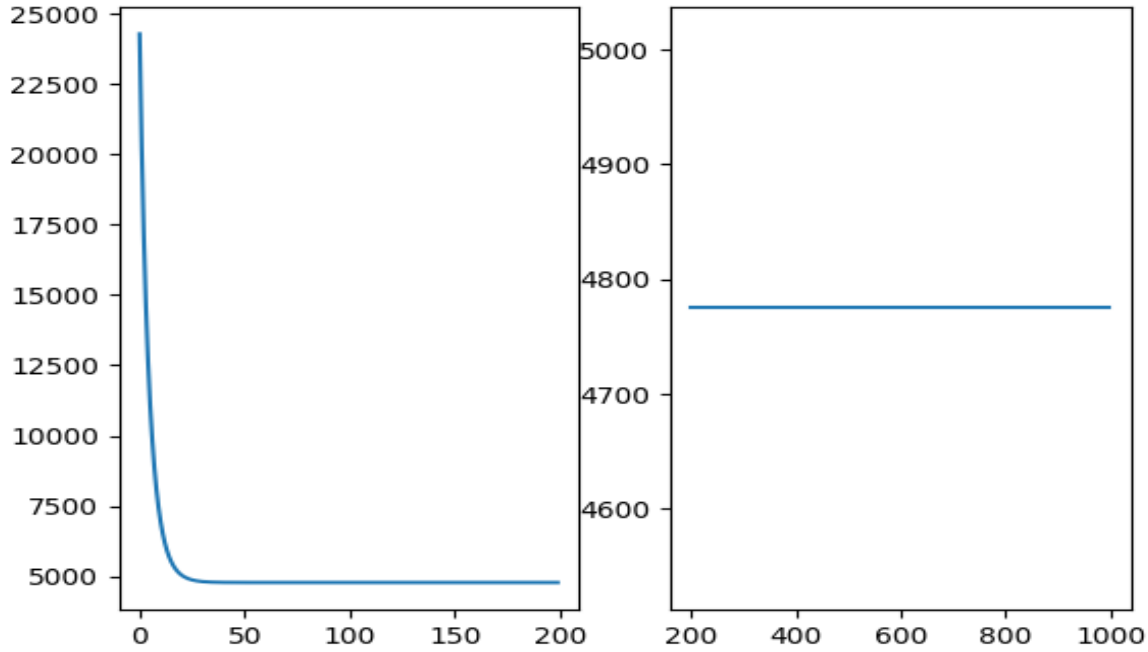
The hyperparameters used are:
Learning rate (alpha)

- The graph between cost and iterations at learning rate = 0.01

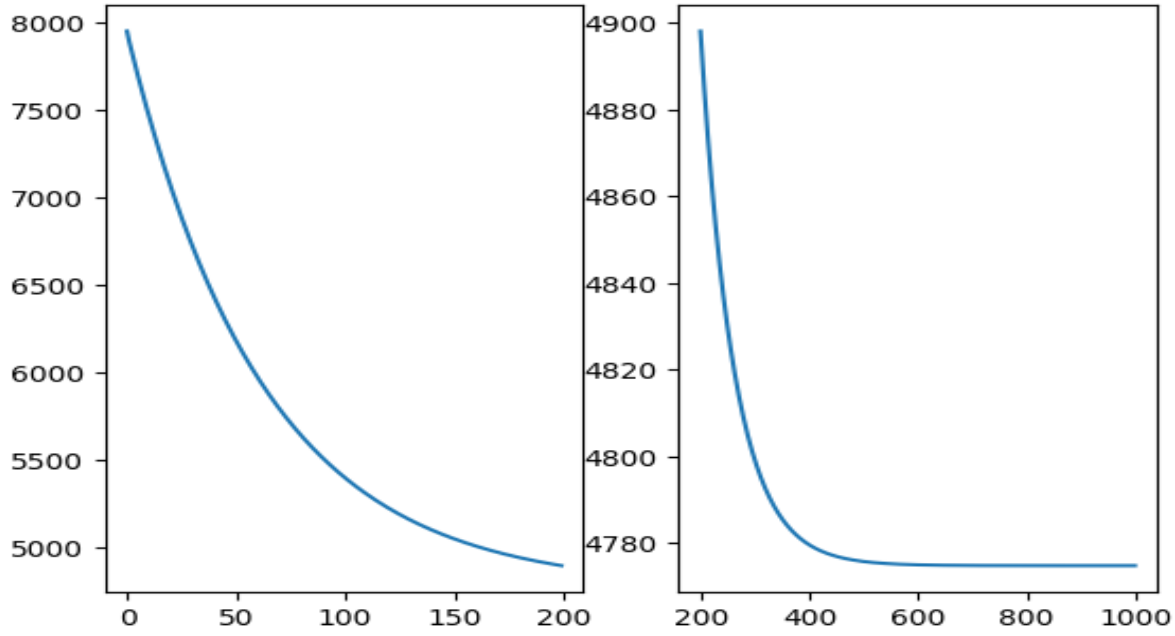
Final Cost = 4774.7492453619525



- At learning rate = 0.1
  Final Cost= 4774.749201291516



- At learning rate = 1
  Final cost =4774.749490013455



- So at learning rate = 0.1 the cost function is getting saturated the fastest
- And the R2 score at that in 1000 iterations is :
    - For train set: 0.8431150528729028
    - For cv set: 0.8417828448142812
- Time taken for 1000 iterations is:
  less than 5 seconds

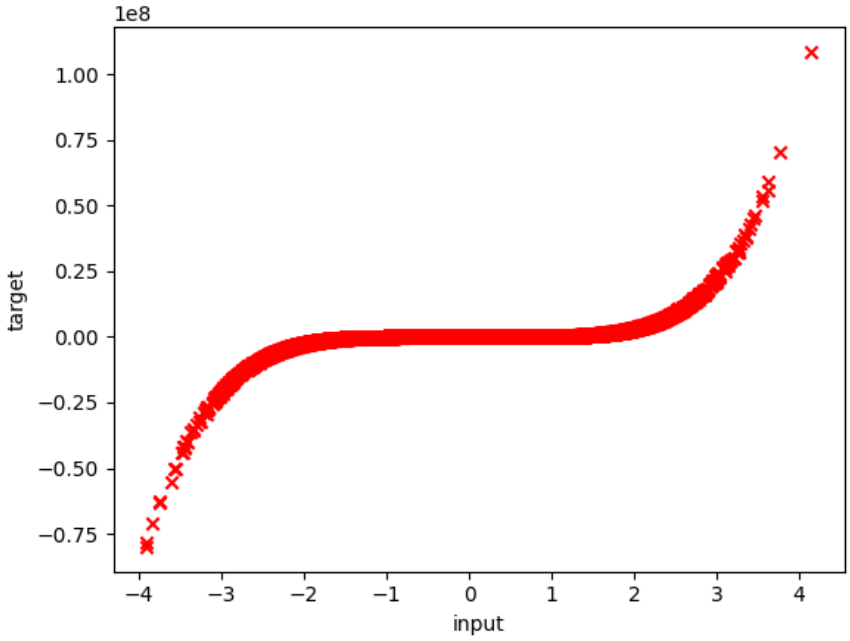
#cv set stands for cross validation set

- Normalised the input data so that I can train the model at a faster rate
  (I did this for all)

2. Polynomial regression:

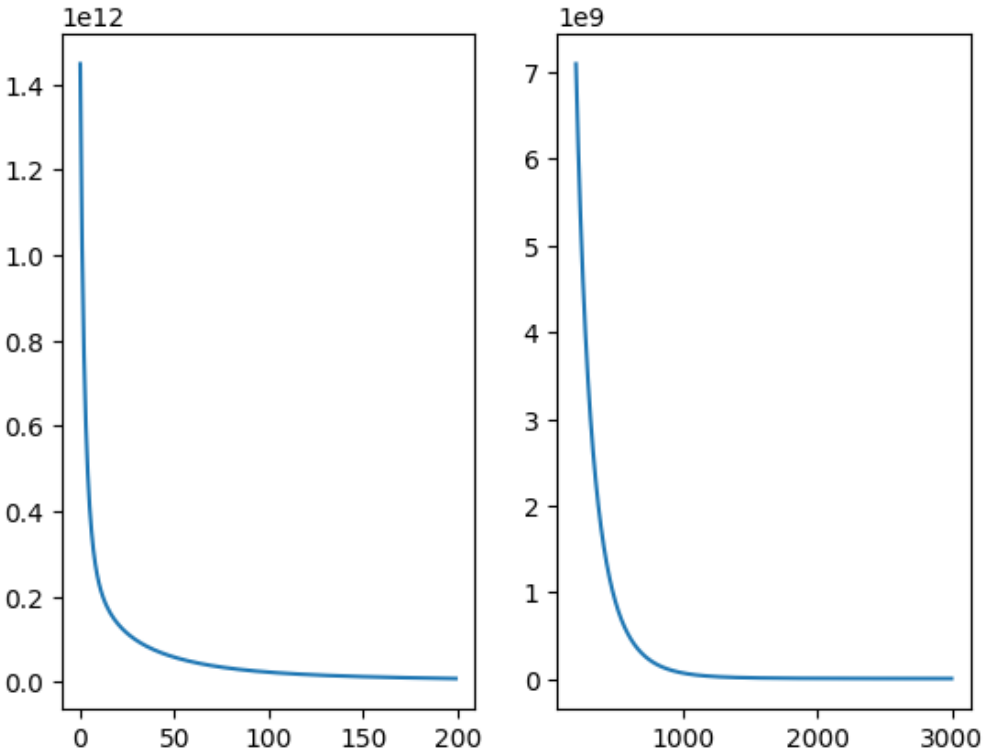
The hyperparameters are:
Maximum Degree of polynomial terms(d), learning rate(alpha), Regularization parameter(lambda)

- The graph between a input feature and target is:

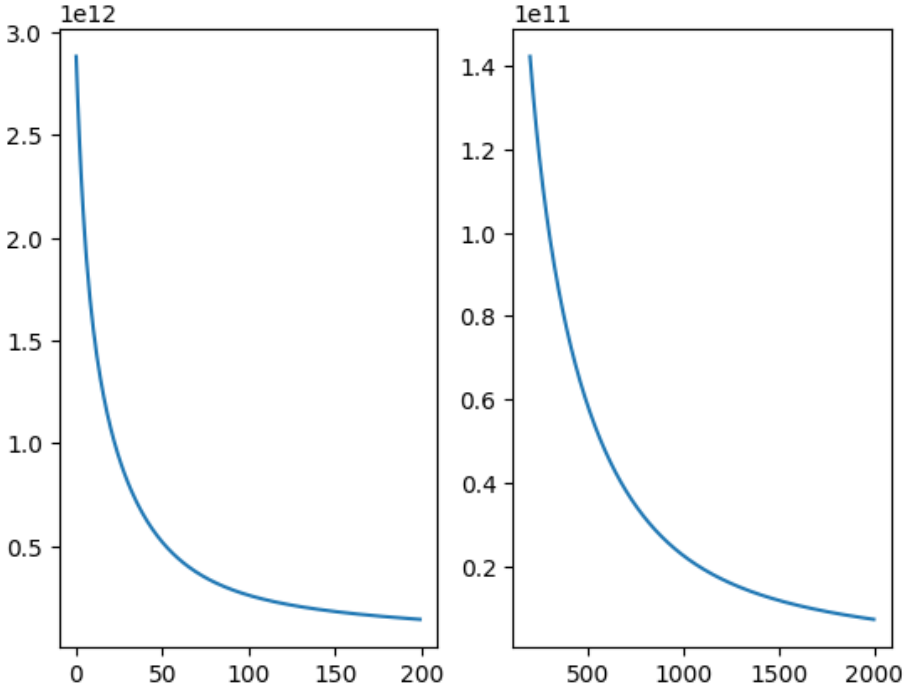


- So this hints that we should use polynomial terms
- So using 3 for loops , I generated all the polynomial, even the mixed terms of all degrees till max degree(d)

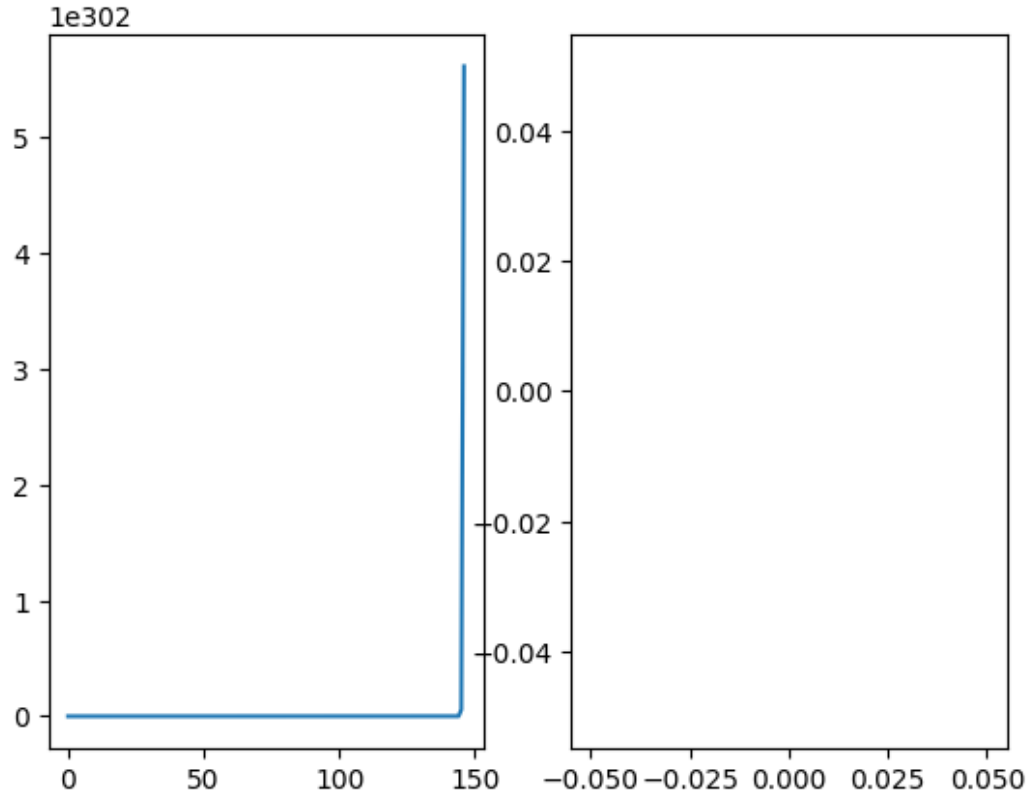
- At d=5 , alpha=0.1 and lamda=0.1
  Final cost = 8628185.619525356



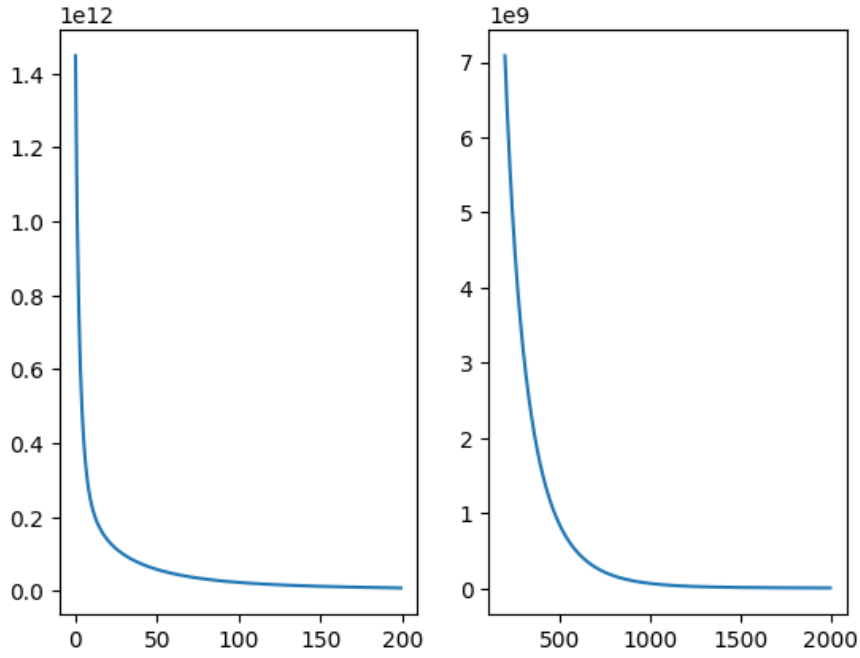
- At d=5 , alpha=0.01 and lamda=0.1
  Final cost = 7190375966.914106



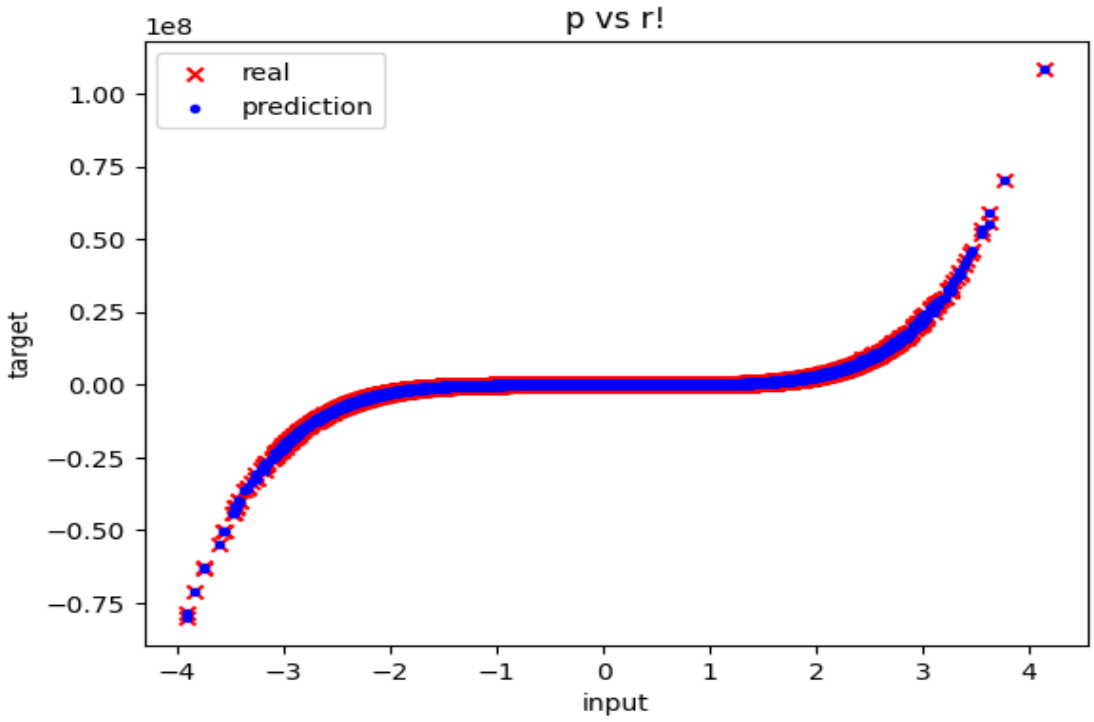
- At d=5 , alpha=1 and lamda=0.1



- At d=5 , alpha=0.1 and lamda=0
  Final cost = 2528050.9880997594



- At alpha =0.1 and lamda =0 , The R2 score on cv set for different d are
  - d=1: 0.251113695693011
  - d=2: 0.250222690330005
  - d=3: 0.905739184928297
  - d=4: 0.905576640988707 6
  - d=5: 0.999999292904175 7
  - d=6: 0.999818050343112 9
- So at alpha=0.1, lambda=0 and d=5 the model trains best
- And the R2 score at that in 2000 iterations is :
  - For train set: 0.999999198332898 3
  - For cv set: 0.999999292904175 7
- Time taken for 2000 iterations is:
  About 1 minute

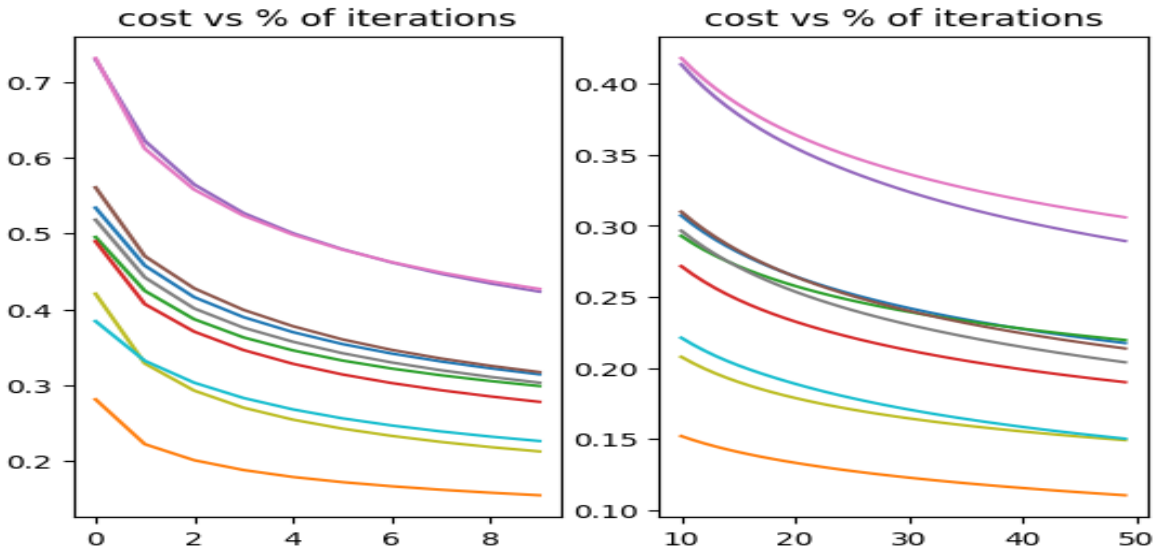


3. Multiclass Classification

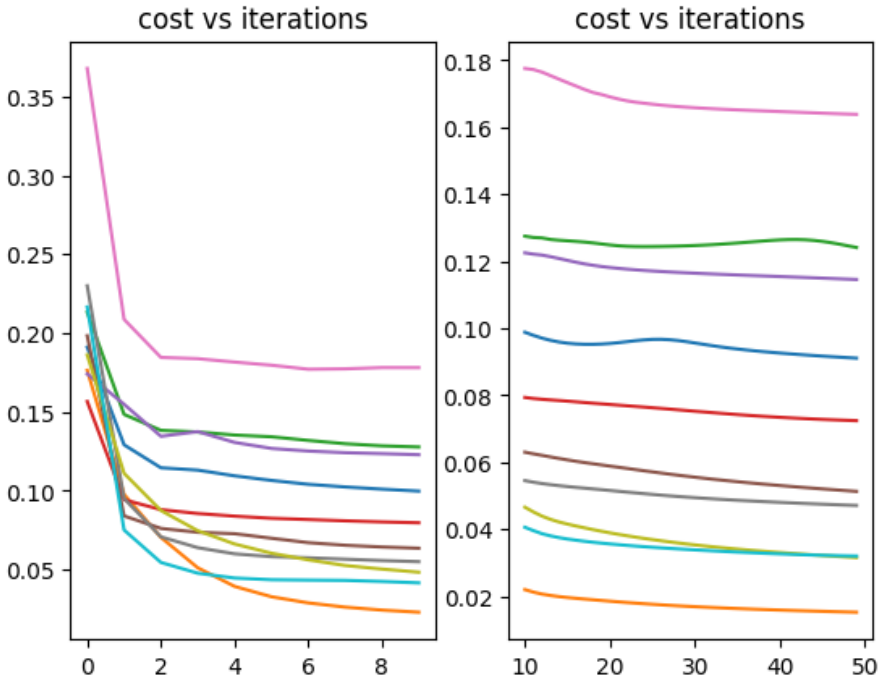
The hyperparameters are:
learning rate(alpha), Regularization parameter(lambda)

First I read about one vs all approach and created a basic model but Initially my model was taking too much time to train, so to reduce that I read about and used Adam optimization algorithm and mini batch gradient decent and learning rate decay(common method)

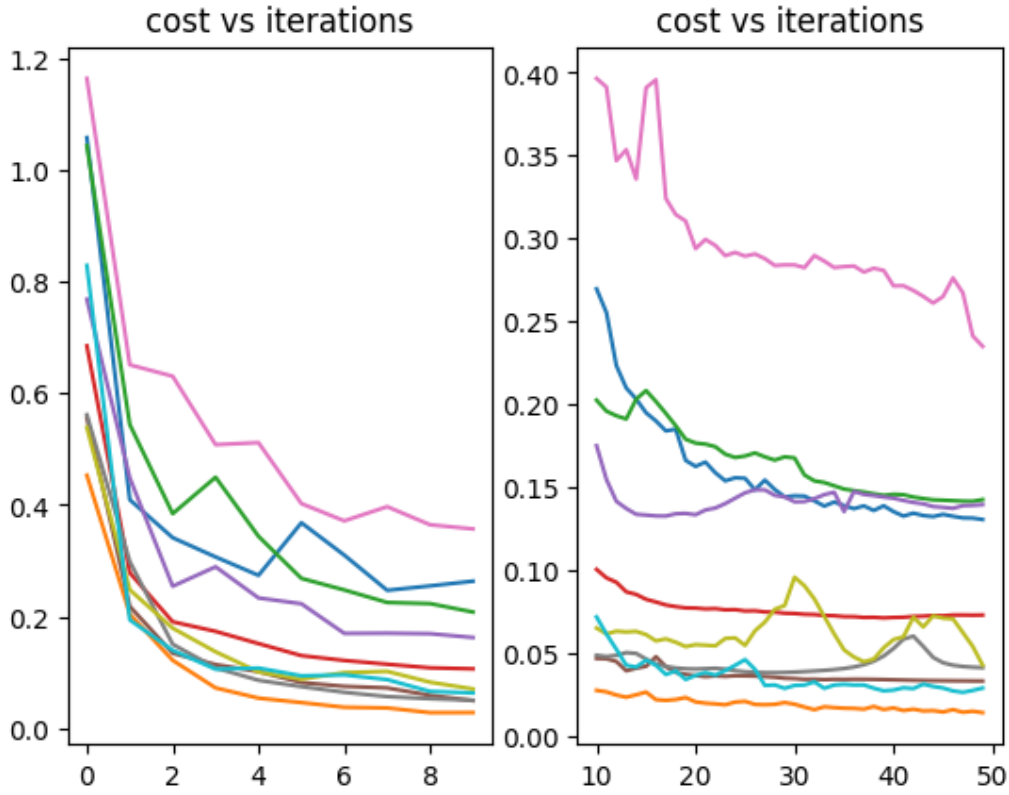
- I used the default value of hyperparameters of adam algorithm i.e beta1=0.9 and beta2=0.999
- And set the size of mini batch to 512
- And set the decay rate=1
- And lamda = 0.1(To reduce overfitting)
- At alpha=0.01
  Accuracy at cv set =80.867%



- At alpha=0.1
  Accuracy at cv set= 83.85 %



- At alpha=1
  Accuracy at cv set= 83.2 %



- So at alpha=0.1 model trains the best
- And the Accuracy at that in 50 iterations is :
    - For train set: 87.366666666666667 %
    - For cv set: 83.85 %
- Time taken for 50 iterations is:
  About 40 sec
- I also write the code to turn the labels into integer values (in case if they are non integers)
  (did this for other classification too i.e knn neural network)

4. KNN (for classification)

Initially my algorithm contained a for loop which was causing my model to take a huge amount of time predict the output for cv set, more than 10 minutes but then I removed the loop and made my code fully vectorised and loop free which increased its speed by more than 20 times , and now it takes less than 30 seconds

- The accuracies obtained on cv set for different values of k (the number of nearest neighbour to take) are:
  - 1:82.82442748091603 %
  - 2:82.89080650514438 %
  - 3:84.10222369731166 %
  - 4:84.38433455028212 %
  - 5:84.21838698971125 %
  - 6:84.6332558911384 %
  - 7:84.20179223365416 %
  - 8:84.35114503816794 %
  - 9:84.13541320942582 %
  - 10:84.35114503816794 %
- So k = 4,5,6 works the best

- (addition feature)KNN for linear regression

  I also implement KNN for linear regression model by taking the mean of 1st 'k' number of nearest neighbours
  This gave me better results than my simple linear regression model

- The R2 score obtained on cv set for different values of k (the number of nearest neighbour to take) are:
  - 15: 0.9207563116389279
  - 16: 0.9207616175867692
  - 17: 0.920957919720242
  - 18: 0.9210065318843899
  - 19: 0.9211988023507633
  - 20: 0.9213288061678868
  - 21: 0.9212083568569037
  - 22: 0.9212323592750407
  - 23: 0.9211719250465553
  - 24: 0.9210974146259545
- So k =20 works the best
- And the time taken is around 1 minute

5. N layer Neural Network(for classification)

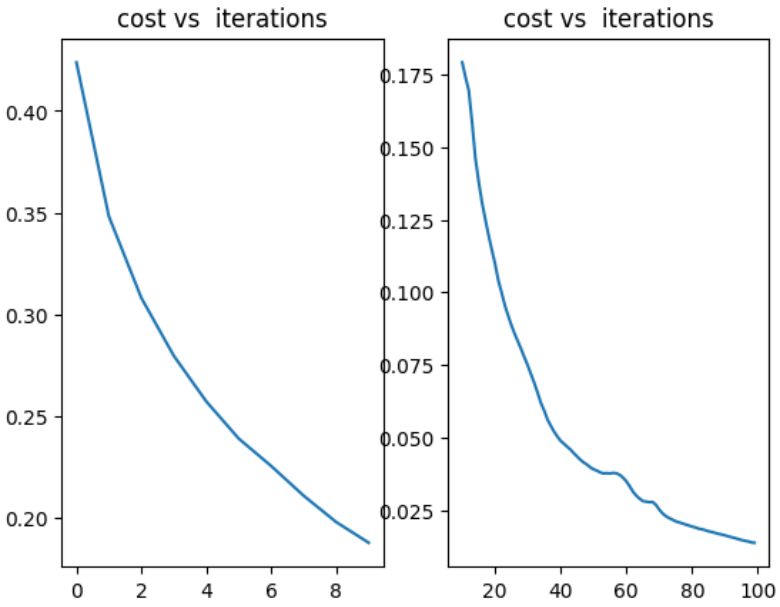
So at first I made a simple network with only one layer(output layers) containing 10 neurones(the number of unique output labels) of softmax activation function
Then I extend my network to 2 hidden layer of ReLu activation function and one output layer and then I made a N layer neural network for classification
Initially my model why showing nan as output so I read about and used He initialization to initialize the weights which solved this problem
Then my model was taking too much to train, so to reduce that I read about and used Adam optimization algorithm and mini batch gradient decent and learning rate decay (common method)
Now my model was showing high variance(the accuracy on train set was good was the accuracy on cv set was not that good), so to reduce that I read about and used L2 regularization and inverted dropout

Hyperparameters:

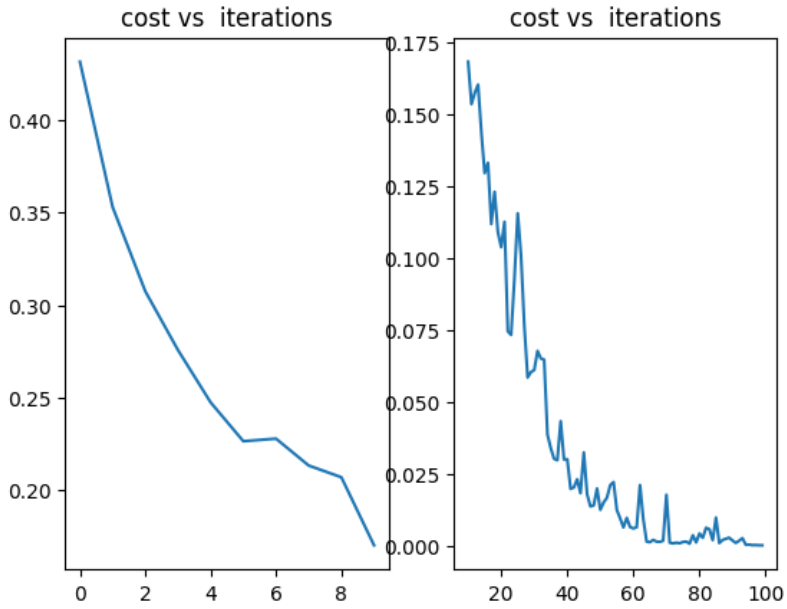
- I used the default value of hyperparameters of adam algorithm i.e beta1=0.9 and beta2=0.999
- The size of mini batch(smb)
- The decay rate(dr)
- Regularization parameter(lamda)
- And set the keeping probability of inverted dropout in hidden layers(kp)
- Structure of neural network used =200,100,10

- At Alpha =0.001, smb=512, kp=1, lambda=0 and dr=0.1 in 100 epochs in 4 minutes

Accuracy at train set= 99.0 %

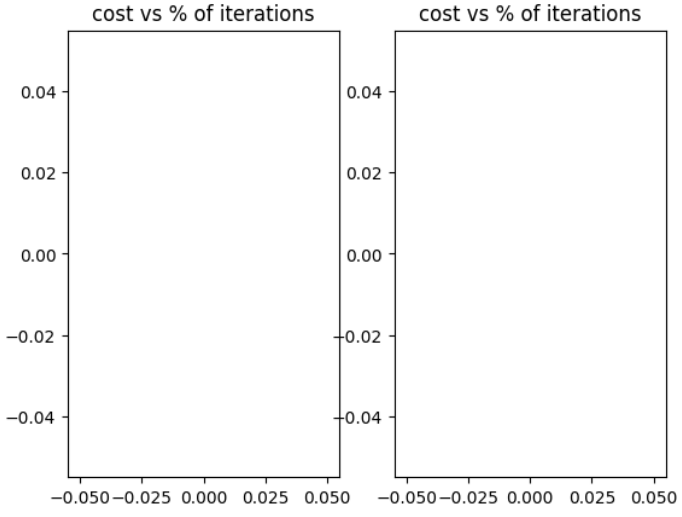


- At Alpha =0.01, smb=512, kp=1, lambda=0 and dr=0.1 in 100 epochs in 4 minutes
  Accuracy at train set= 99.99 %



- At Alpha =0.1, smb=512, kp=1, lambda=0 and dr=0.1 in 100 epochs

RuntimeWarning: overflow encountered in exp
ez=np.exp(Z(X,W,B))
Cost=nan



- At Alpha =0.01, smb=256, kp=1, lambda=0 and dr=0.1 in 100 epochs in 4 minutes
    - Accuracy For train set: 100%
    - Accuracy For cv set: 87 %

So the model was overfitting, i.e. showing high variance

- At Alpha =0.01, smb=2048, kp=0.8, lambda=0.1 and dr=0.1 in 64 epochs in 2 min
    - Accuracy For train set: 97%
    - Accuracy For cv set: 88.9 %

So using regularization techniques, model is showing a bit lesser variance
So this is a better choice of hyperparameters

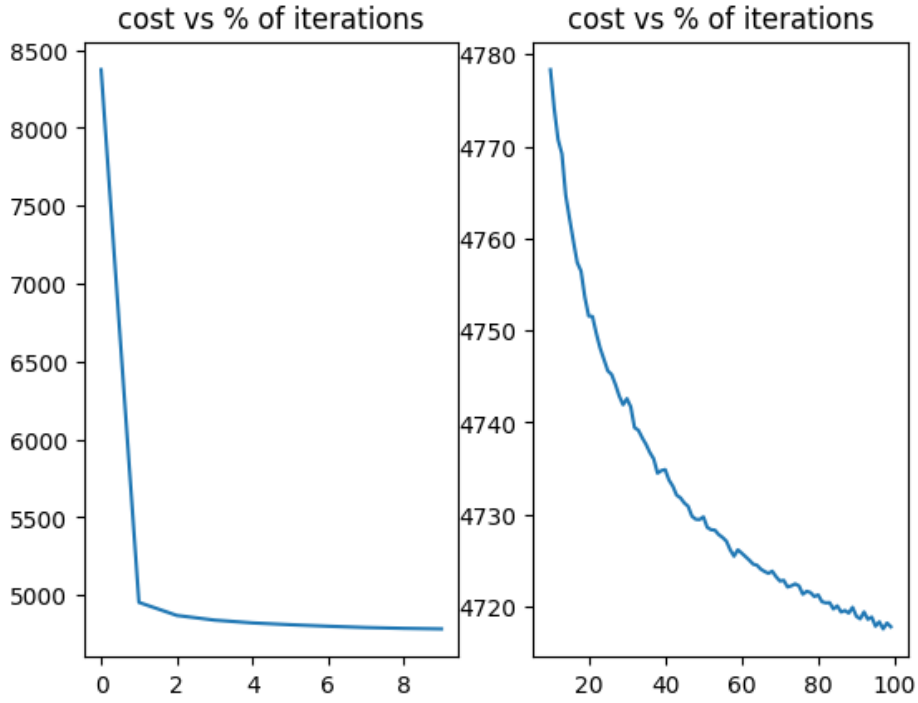
- <u>N layer neural network(for linear regression)</u>

  By changing the size of output layer to one and with no activation function, changing the cost function and prediction function, I made the neural network for regression

- Structure used=[100,50,1]

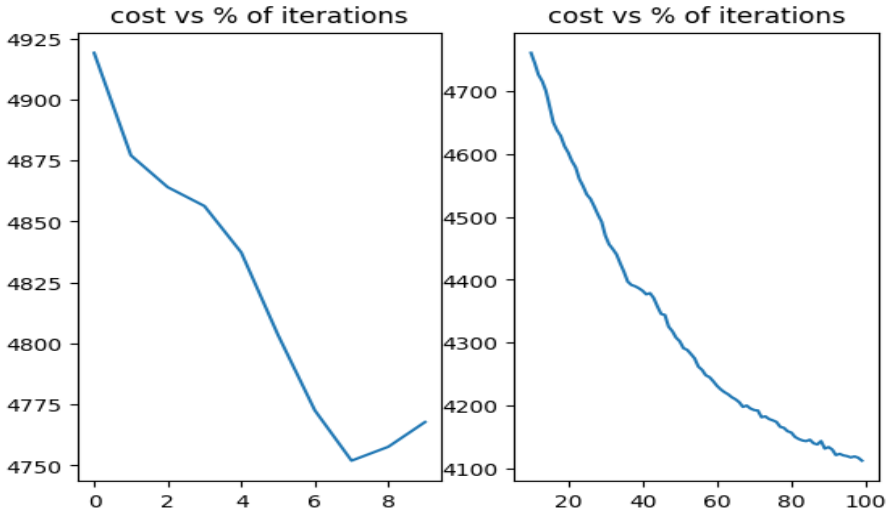
- At Alpha =0.001, smb=256, kp=0.9, lambda=0.1 and dr=0.1 in 100 epochs in 2 minutes

R2 score on cv set= 0.8431817962577718



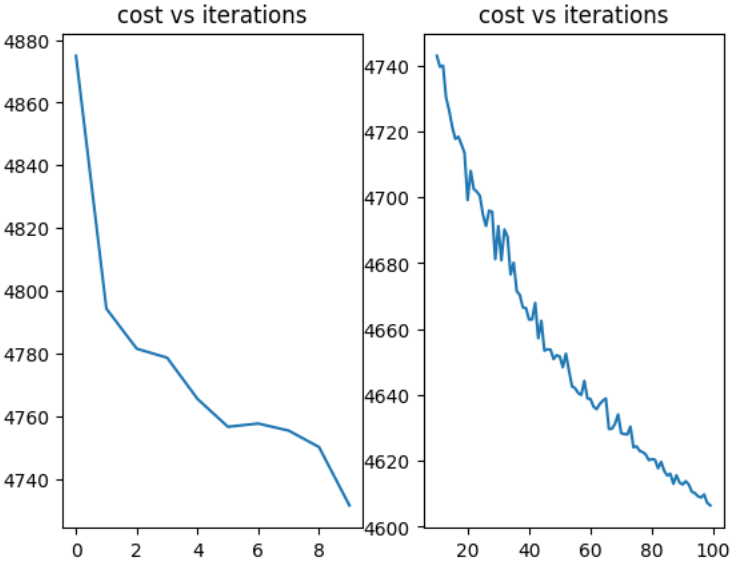
- At Alpha =0.1, smb=256, kp=0.9, lambda=0.1 and dr=0.1 in 100 epochs in 2 minutes

R2 score on cv set= 0.8149148717947012



- At Alpha =0.01, smb=256, kp=0.9, lambda=0.1 and dr=0.1 in 100 epochs in 2 minutes

R2 score on cv set=0.84023873783



- So alpha = 0.001 works best