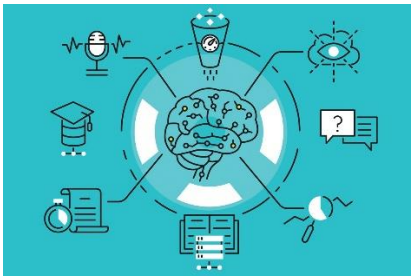
CYBER LABS

WINTER OF CODE

6.0

REPORT OF ML BOOTCAMP PROJECT



NAME-Shashwat Mandal

IIT(ISM) Dhanbad

Personal Details

Name : Shashwat Mandal

Place : Ranchi, Jharkhand

Branch : Integrated Mater of Technology in Mathematics And Computing

Year : First Year

Date of Birth : 31/01/2006

Admission No. : 23JE0905

Contact No. : 9262817505

Institute Email Id : 23je0905@iitism.ac.in

Personal Email Id : shashwatmandal18@gmail.com

LinkedIn profile : https://www.linkedin.com/in/shashwat-mandal-0b4191287?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app

Github Profile : https://github.com/Shashwatmandal

WEEK 1

- Work on Linear Regression

- Vectors used in place of loops makes it many times faster and short codes with much more efficiency.

- Trained model without normalization which takes much time due to the low value of learning rate and high no. of iterations.

- Model Function used →

$$f(x^i) = wx^i + b$$

- Normalisation used = Z-score Normalisation

$$Z = \frac{x - \mu}{\sigma}$$

where x is score

μ is mean

σ is Standard deviation.

. Cost function used = mean squared error (MSE)

$$\frac{1}{2m}\sum_{1}^{m}(h(x^{(i)})-y^{(i)})^2$$

. Gradients –

$$dj\_dw = (1/m)\sum(f(w,b) - y_i)x_i$$

$$dj\_db = (1/m)\sum(f(w,b) - y_i)$$

. Gradient descent:

w = w - alpha * dj_dw

b = b - alpha * dj_db

•

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Training logs

- **Time taken by model for training data = negligible**

- **MSE =** `0.1270781924099689 8`

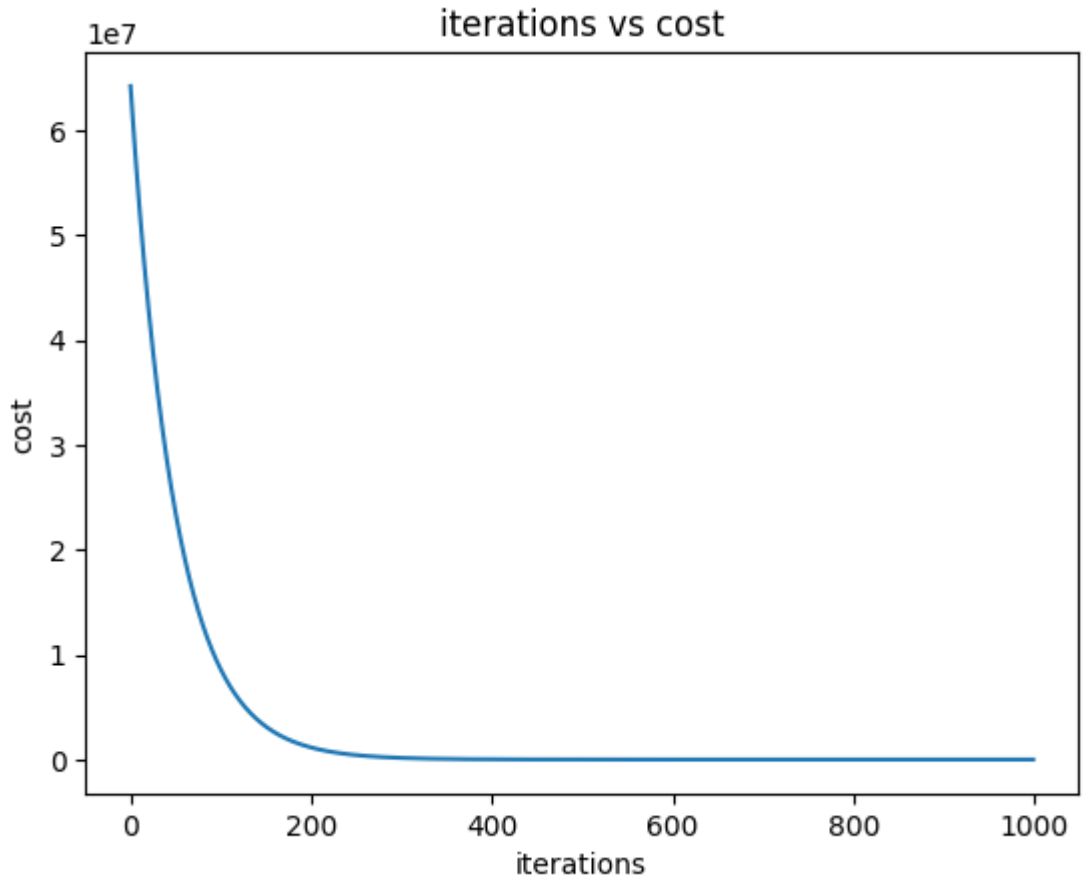
- `Iteration    0: Cost 64219095.90`
- `Iteration  100: Cost 8569112.53`
- `Iteration  200: Cost 1144470.29`
- `Iteration  300: Cost 152991.04`
- `Iteration  400: Cost 20469.92`
- `Iteration  500: Cost  2741.27`
- `Iteration  600: Cost   367.43`
- `Iteration  700: Cost    49.29`
- `Iteration  800: Cost     6.62`
- `Iteration  900: Cost     0.89`

- **R2 Score = [1.]**

Hyperparameters

- **Weight Initializations = zeros**
- **Bias Initializations = zeros**
- **Learning rate (alpha) = 0.01**
- **Iterations = 1000**

Training Visualization



Inferring the cost function and R2 score, we might infer the danger of overfitting, thus we will take a cross validation set and try to predict the outputs of the cross validation sets.

- Cost of the cross validation set - 0.12888896652052217
- R2 score of cross validation set → 0.9998757489175094

. As we can see the R2 score of the cross validation set is pretty close to 1, so we can infer that the model is predicting fairly fine.

Polynomial regression

The data with a total length of 50,000 is first split into two sets. The train set with 30,000 training examples and the cross validation(CV) set with 20,000 training examples. Degree 5 polynomial is chosen for training the data because it gives the satisfactory R2 score for both train and CV. And also the cost vs degree curve also gives minima for CV set.

- Model function used

$$\mathbf{F(x^i)=\Sigma w^i x^i + b}$$

Where x^i are the elements upto 5 degree polynomial which are total 55 in number.

- Normalization, Cost function, gradient and gradient descent are similar to the linear regression.

Training logs

- **TIME taken by model for training data = 1 minute.**
- For train data set:

MSE = 3354113143391.25

(I tried many combinations of power and learning rate but this was the only cost that was coming pretty low and with a good R2 score)

R2 Score = [0.94632657]

• For cross validation set:

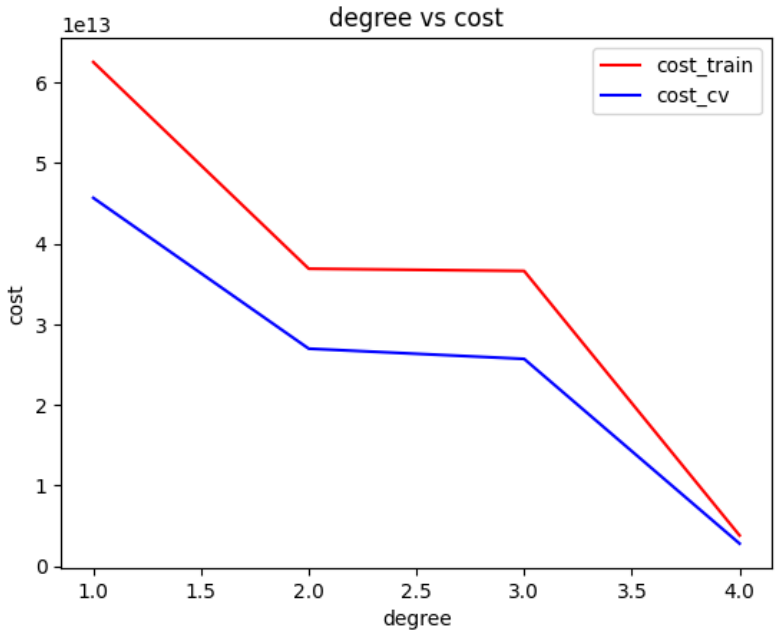
R2 Score = [0.93873675]

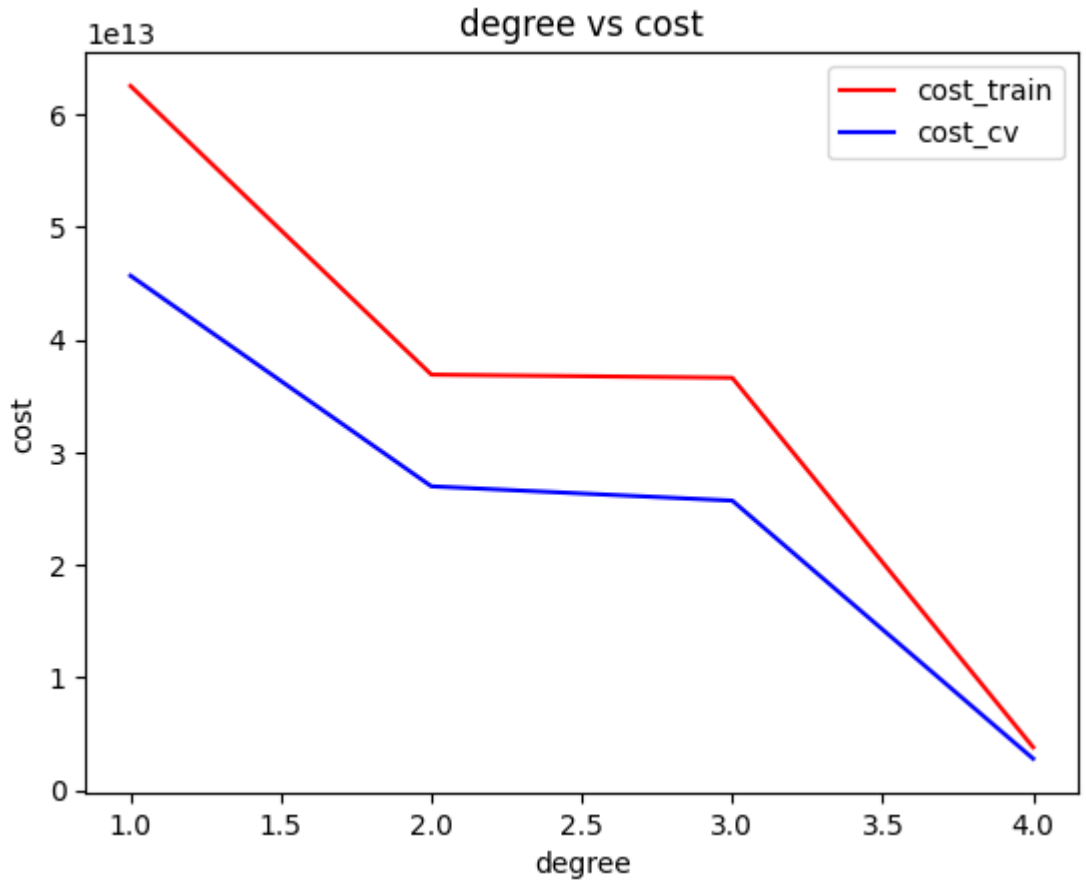
```
Iteration    0: Cost 54627042343426.41
Iteration 2000: Cost 4671493626027.91
Iteration 4000: Cost 3884794946921.65
Iteration 6000: Cost 3604953643791.07
Iteration 8000: Cost 3481770466525.66
Iteration 10000: Cost 3420896889017.98
Iteration 12000: Cost 3389036680431.28
Iteration 14000: Cost 3371841054519.17
Iteration 16000: Cost 3362378369615.43
Iteration 18000: Cost 3357095886231.46
```

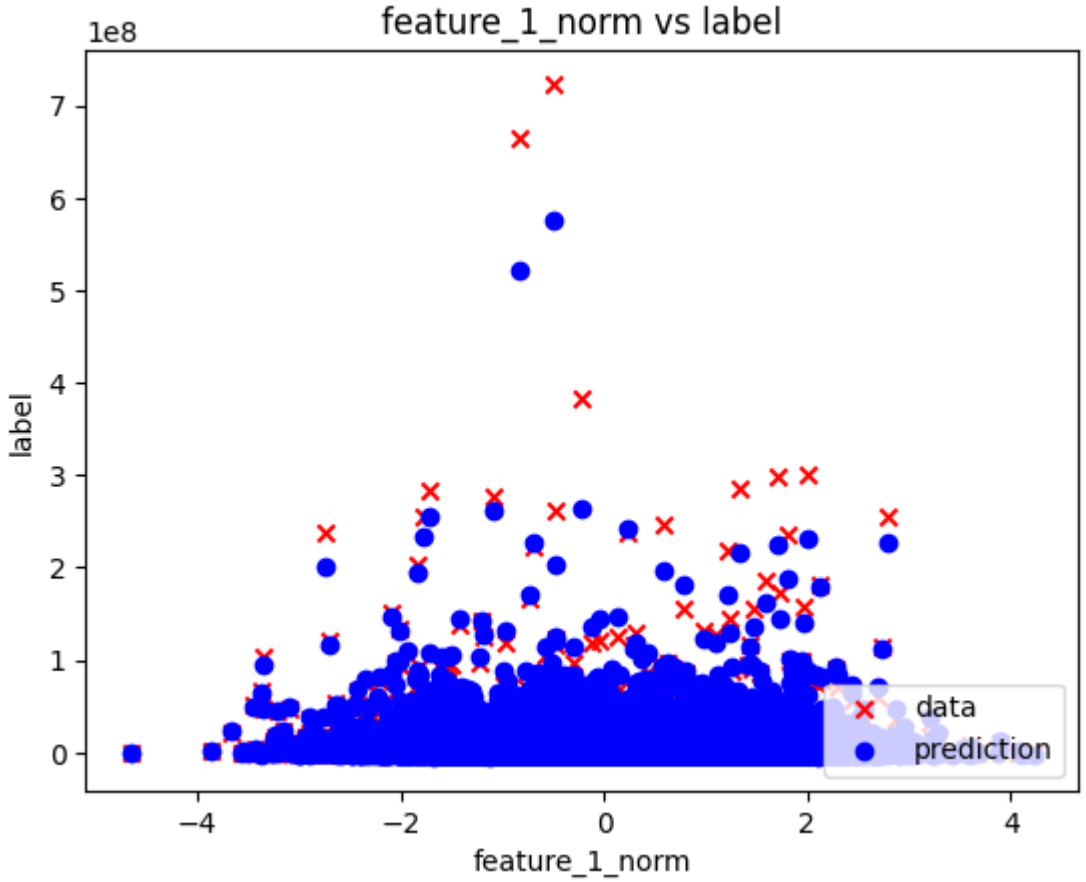
Hyperparameters

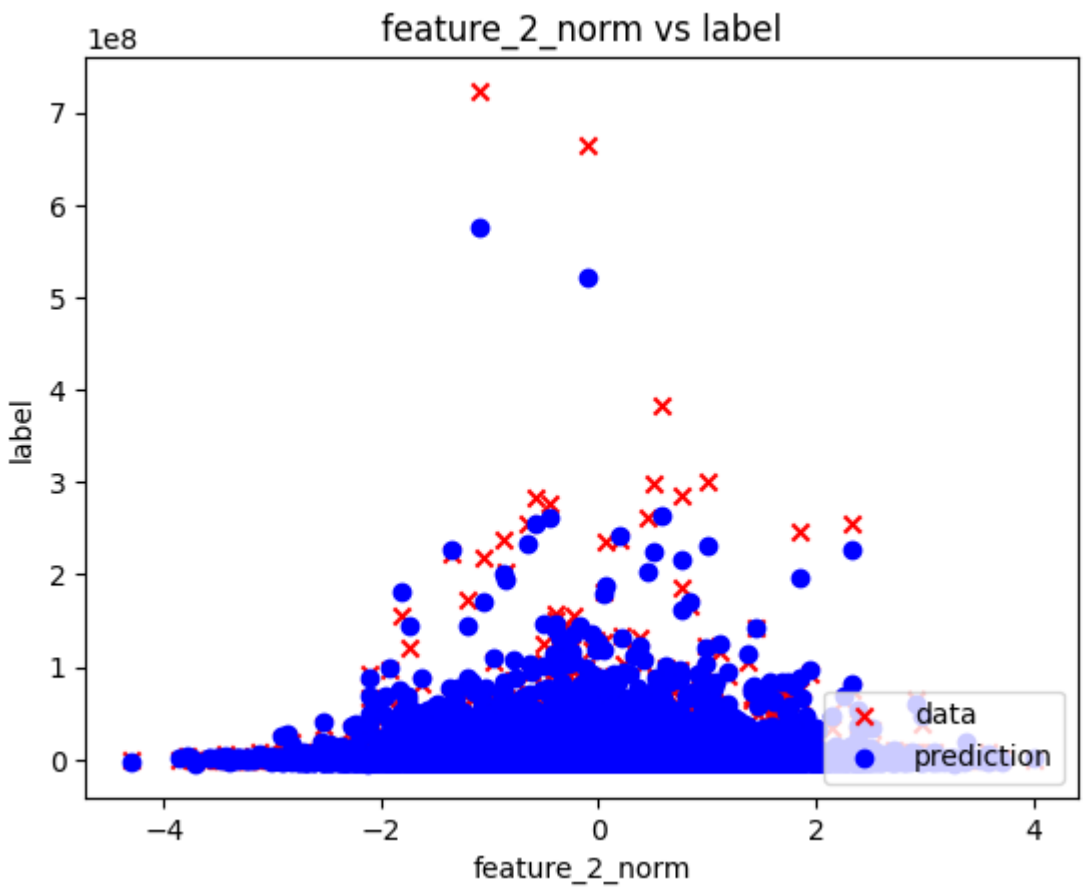
- Weights Initialization = zeros
- Bias Initializations = zero
- Learning rate(alpha) = 0.001
- Iterations = 20,000

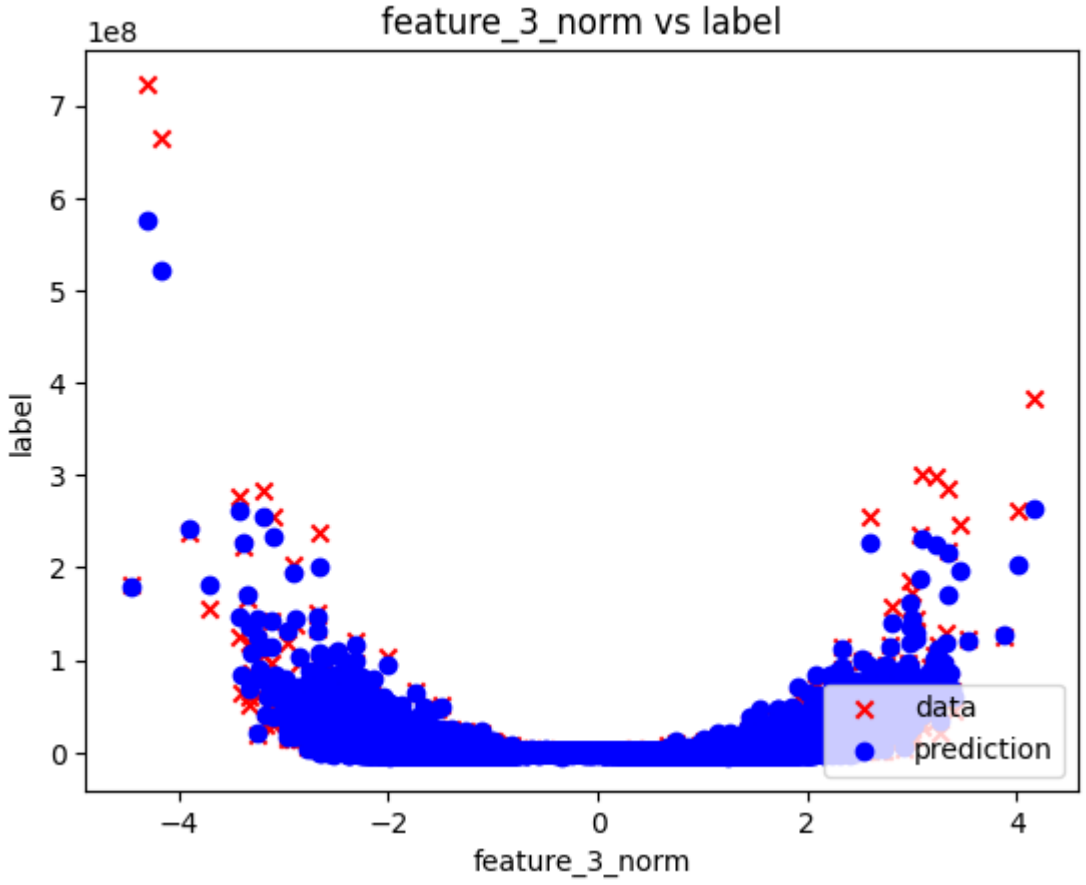
Training Visualization




degree vs cost
1e13
cost_train
cost_cv
cost
degree
0
1
2
3
4
5
6
1.0
1.5
2.0
2.5
3.0
3.5
4.0


feature_1_norm vs label
1e8
data
prediction
label
feature_1_norm
0
1
2
3
4
5
6
7
−4
−2
0
2
4


feature_2_norm vs label
1e8
label
feature_2_norm
data
prediction


feature_3_norm vs label
1e8
label
feature_3_norm
data
prediction

WEEK 2

Logistic regression

I had split the data of 30,000 training examples into train with 20,000 dataset and test with 10,000.
Then I convert my output vector y into one hot encoded Y. One hot encoding gives the particular item a value of 1 and others 0 for a particular training example. One-hot encoding is basically a probability but the difference is that it is an absolute probability because we already know the output of the training example.
I had also added a 1's row at 0 index to remove the need of extra separate bias 'b' in the z-normalized X(input data).

- **Model function :**

  F(X)= sigmoid(X.theta)

- **Cost function:**

  -(1/m) * (sum(Y*log(f(x)) + (1-Y)*log(1-f(x)))

- **Gradient descent:**

  (1/m) * np.dot(X.T, f(x) - Y)

- **Gradient:**

  theta= theta - eta * gradient(theta, X, Y)

Training logs

- Time taken by model for training data: 4 minute 25 seconds
- Accuracy of train data: 90.48%
- Accuracy of test data: 90.37%

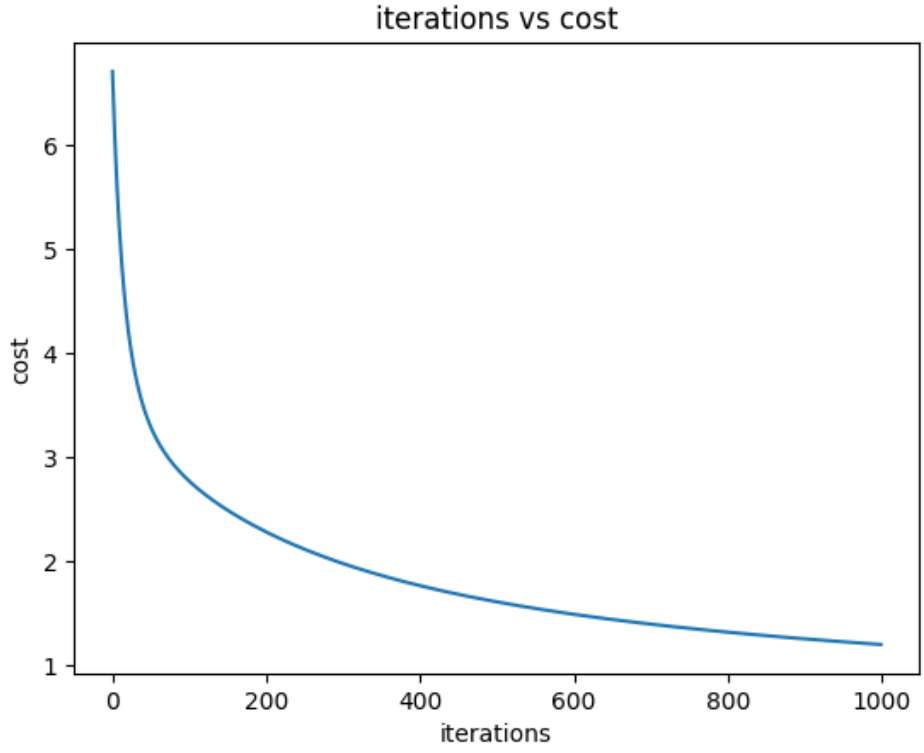
Cost after 0 iterations is : 6.714661801072694
Cost after 100 iterations is : 2.772866963087103
Cost after 200 iterations is : 2.2847318384604396
Cost after 300 iterations is : 1.9804771453345391
Cost after 400 iterations is : 1.7681673075337754
Cost after 500 iterations is : 1.612211805802549
Cost after 600 iterations is : 1.4927475086702044
Cost after 700 iterations is : 1.3980669957980383
Cost after 800 iterations is : 1.3209414143987117
Cost after 900 iterations is : 1.2567013976326347
(785, 10)

Hyperparameters

- Weight Initializations :

  theta = (np.random.randn(785, 10))*0.01.
- Learning rate(alpha) : 0.01
- Iterations : 1000

Training Visualization



WEEK 3

K-Nearest Neighbour

What is KNN?

KNN basically classifies the new data points based on the similarity with the earlier stored data.

This algorithm basically works in three basic steps:

1. Calculating Distances:

Distance functions

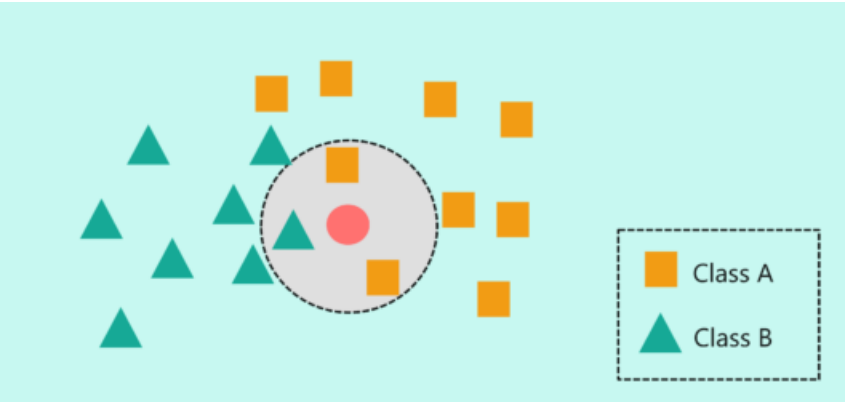
Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

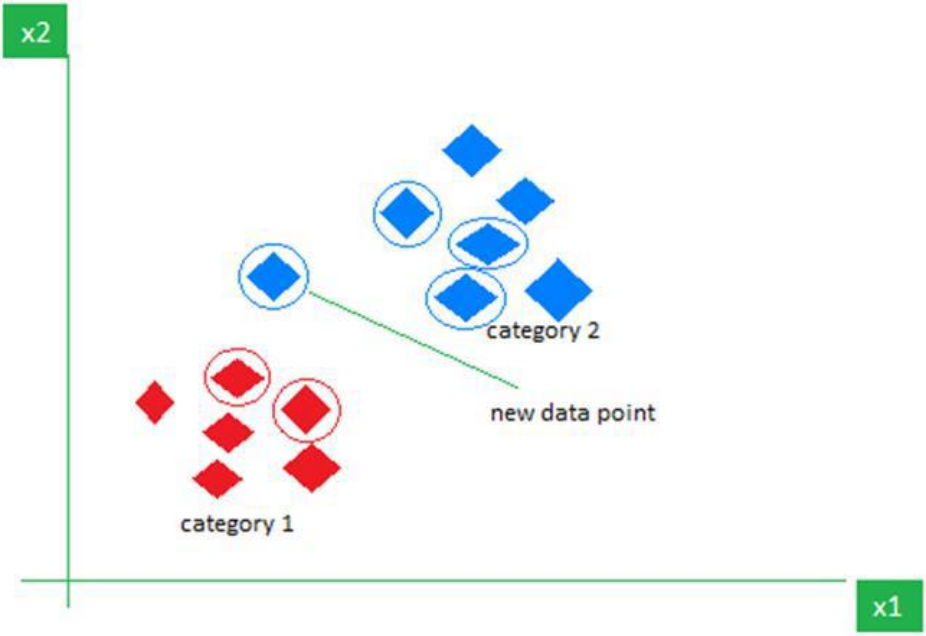
I had used Euclidean distance to calculate the distances of each test data from every given datapoints and store them in matrix.

2. Sorting first K distances:



Sorting the calculated distances of each test data in ascending order and taking out the first k distances.

3. Finding the most frequent category:



The most frequently occurred category will be the category of the particular training example.

I had split the data of 30,000 length into known datapoints of 20,000 and test data points of length 10,000.

Training logs

- Time taken = approx. 8 minutes
- Accuracy = 95.59%
- Distance Formula Used = Euclidean

Hyperparameter

- **K = 60**

WEEK 4

Neural Network

I had split the data of 30,000 training examples into train with 20,000 dataset and test with 10,000.

I had pre processed the data by normalization of input data(X) and then one hot encoding of output data(Y).

I had first made the hardcore 1 hidden layer neural network from scratch.

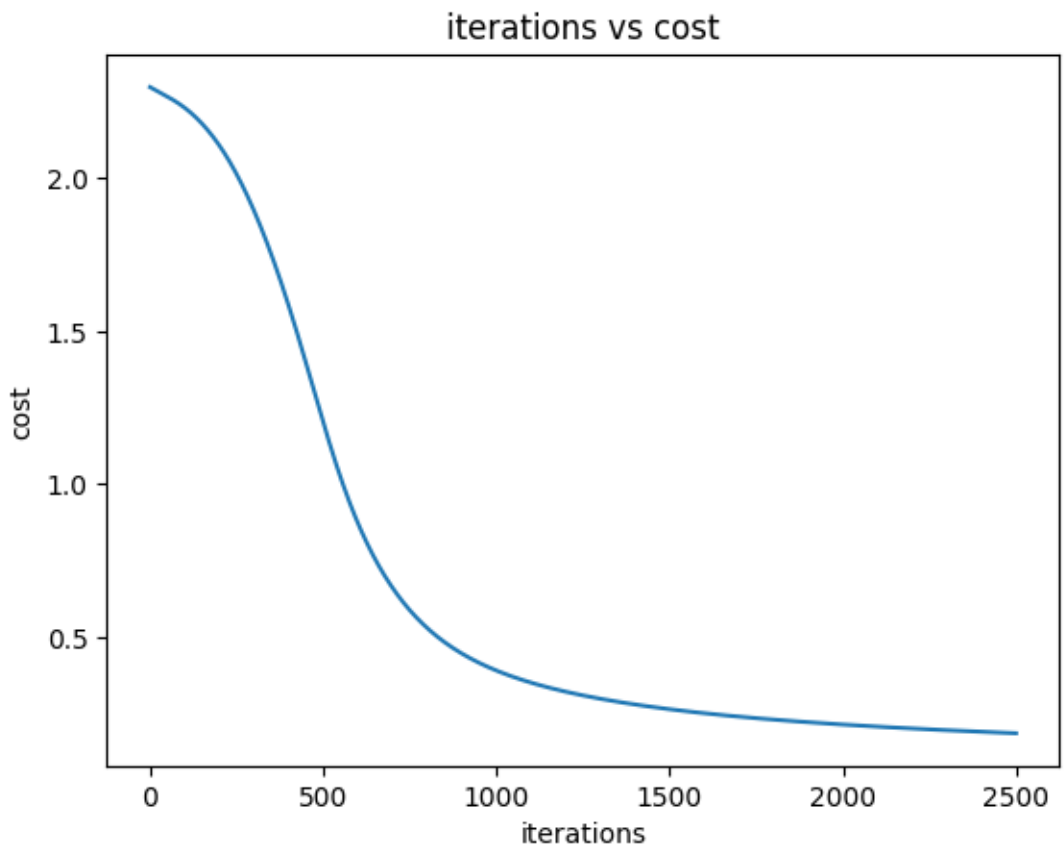
Training logs

- Time taken – approx. 10 minutes
- Accuracy on test data = 94.53%

iterations = 0 cost = 2.29866599448333
iterations = 250 cost = 2.013776721324811
iterations = 500 cost = 1.202210250803944
iterations = 750 cost = 0.588034223866607
iterations = 1000 cost = 0.390897083267094
iterations = 1250 cost = 0.309643652845983 9
iterations = 1500 cost = 0.264585198398486 3
iterations = 1750 cost = 0.2353152519712707 3
iterations = 2000 cost = 0.2145050397172148 8
iterations = 2250 cost = 0.1987114955961451 5

Hyperparameters

- Learning rate(alpha) – 0.009
- No. of layers – 4(3 hidden and 1 output layers)
- No. of neurons in each layer-
  Hidden layer 1 – 50
  Hidden layer 2 – 25
  Hidden layer 3 – 15
  Output layer – 10
- Activation function in each layer :
  Hidden layers – Relu
  Output Layer - Softmax


iterations vs cost
cost
2.0
1.5
1.0
0.5
0
500
1000
1500
2000
2500
iterations