ML Assignment 1

Anshul Mendiratta
2018219

Question 1 Linear Regression

a) Choose an appropriate value of K and justify it in your report along with the preprocessing strategy.
Answer. Value of K chosen was 10. I chose K as a starting point because it was mentioned in class that the value of K is usually chosen to be 10. Further with K = 10, the RMS and MAE for each fold were as expected and some even performed better than training on the entire model. Hence, I kept K as 10.

b) Implement gradient descent using two losses - RMSE loss and MAE loss
Answer. Written in code (scratch.py)

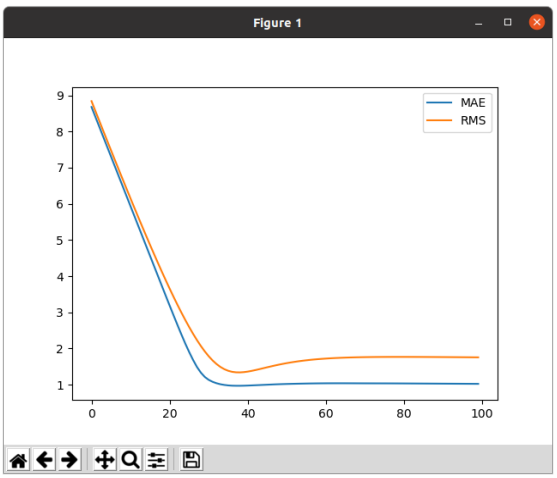
Analysis -

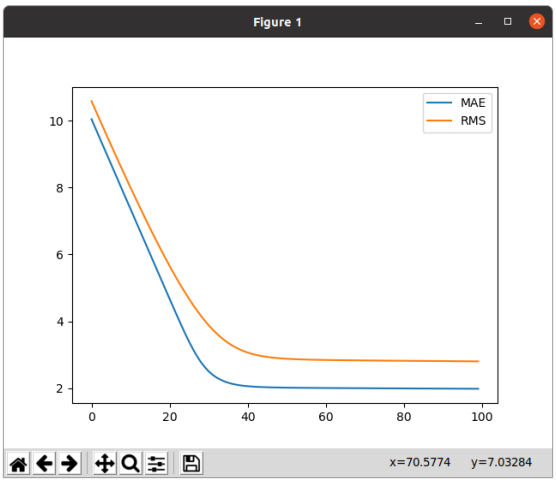
Dataset 1
Learning rate = 0.1
Iterations = 100

Dataset 2
Learning rate = 0.00001
Iterations = 200

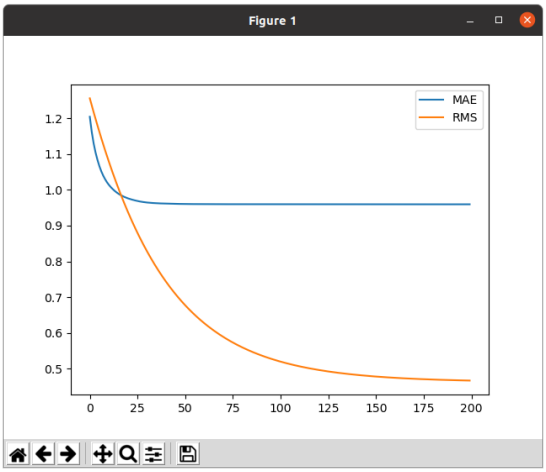
a) Include plots between training loss v/s iterations and validation loss v/s iterations.(total 4 plots - 2 plots for each dataset)

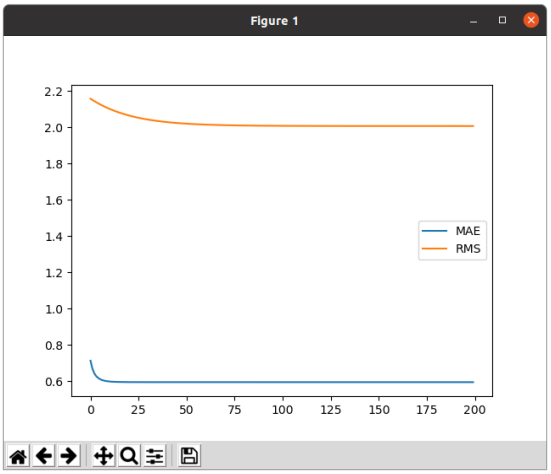
Dataset 1 Validation Loss and Training Loss vs Iterations Fold 4





Dataset 2 Validation Loss and Training Loss vs Iterations Fold 2





b) Include the best RMSE and MAE value achieved (as well as which fold achieves this) in your report.
Dataset 1 -
Best RMSE achieved was 1.757 and MAE achieved was 1.024 at fold 4
Dataset 2 -
Best RMSE achieved was 0.20 at fold 3
Best MAE achieved was 0.06 achieved at fold 5

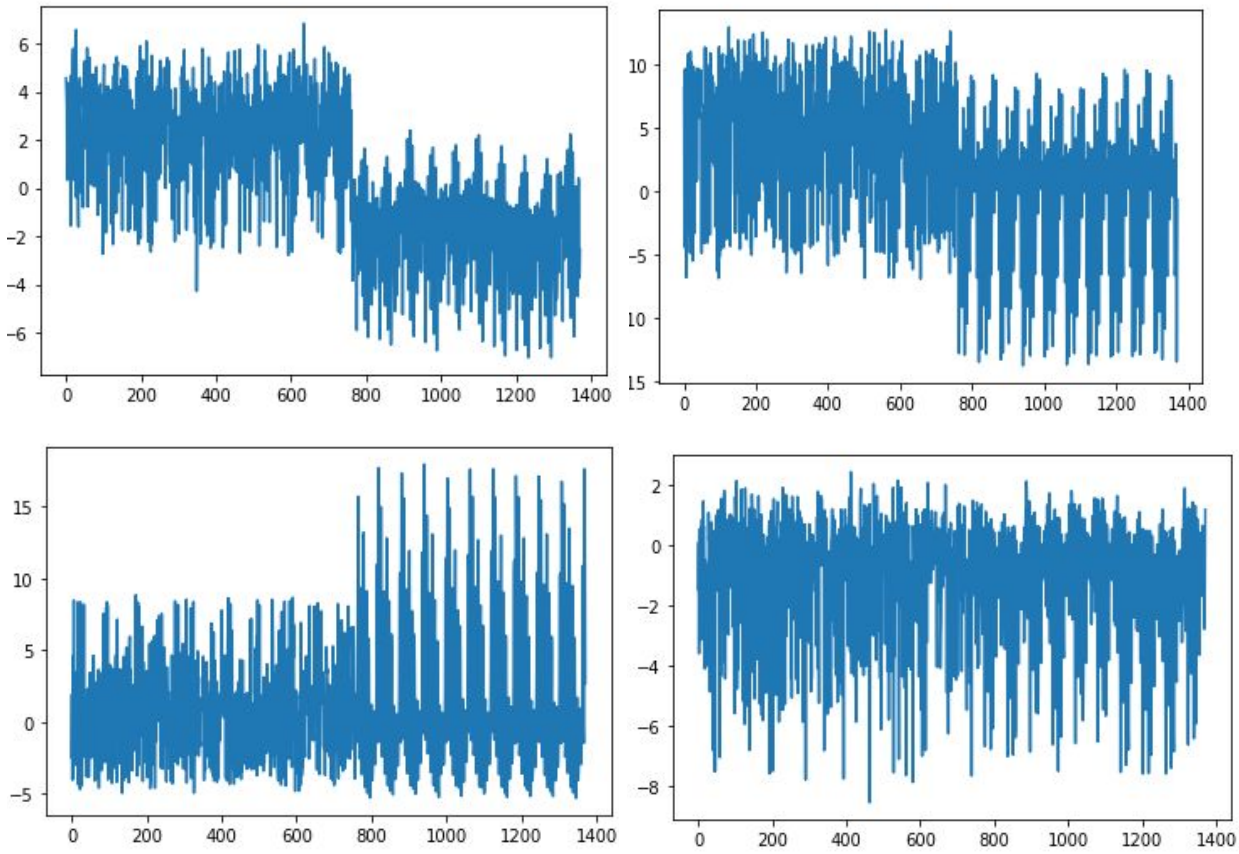
c) For each dataset, analyze and describe which of the loss leads to better performance.
In both cases MAE seems to perform better than RMSE. While there are a few folds for which RMSE performs better, in general MAE has a lower value than RMSE, both for training and testing data and in the testing data MAE per epoch remains more consistent than RMS per epoch

d) What is the relationship between MAE and RMSE? Under what conditions are RMSE and MAE expected to give similar values? Which loss will you prefer in such a case and why?
RMSE is always greater than or equal to MAE. (We can get the relation by squaring both and comparing. MAE will have n as the denominator which is a natural number). They are equal when all errors are the same. In general we can choose between MAE and RMS depending on how much we want to penalize outliers. In case we want to penalize outliers we can use RMS else we can use MAE. In case both are similar we may prefer using RMS as it is differentiable throughout and hence helpful for gradient descent in linear regression.

e) Implement the normal equation form
Answer. Applying normal equation on Dataset 1 for fold 4 using MAE we obtain the coefficients -
[[3.65510393] [2.84097268] [3.71660144] [0.28261723]
[11.06513331] [11.17120238] [8.77262893] [-20.49679083]
[-9.83073096] [9.33070933]]
Training Loss Obtained was - 1.62367053
Testing Loss Obtained was - 1.27424648

Question 2 Logistic Regression

a) Analyze the class distributions and comment on the feature values for each of the given features. (5 points)
We have been variance, skewness, curtosis and entropy as features and we need to predict the class using logistic regression.
Plots for variance, skewness, curtosis and entropy vs the datapoints are as follows (left to right) -



We also know that the first half of the dataset belongs to class 0 and the other half to class 0 from looking at the data. Hence we can see that variance and skewness will majorly affect our result whereas entropy will barely have any effect.

Perform a train:val:test split in the ratio 7:1:2 and implement Logistic Regression based on the given template functions.

a) Using Stochastic Gradient Descent (SGD) and Batch Gradient Descent, choose an appropriate learning rate and the number of epochs (iterations). Report the accuracy obtained on both the training and test set.

   Answer.

   1) Stochastic
      Learning rate = 0.1, epochs = 2000
      Accuracy obtained for testing data in Stochastic is 0.97
      Accuracy obtained for training data in Stochastic is 0.98

Learning rate = 0.0001, epochs = 2000
Accuracy obtained for testing data in Stochastic is 0.58
Accuracy obtained for training data in Stochastic is 0.87

Learning rate = 0.01, epochs = 2000
Accuracy obtained for testing data in Stochastic is 0.93
Accuracy obtained for training data in Stochastic is 0.98

Learning rate = 10, epochs = 2000
Accuracy obtained for testing data in Stochastic is 0.90
Accuracy obtained for training data in Stochastic is 0.94

2) Batch
Learning rate = 0.1, epochs = 2000
Accuracy obtained for testing data in Batch is 0.99
Accuracy obtained for training data in Batch is 0.989

Learning rate = 0.0001, epochs = 2000
Accuracy obtained for testing data in Batch is 0.59
Accuracy obtained for training data in Batch is 0.87

Learning rate = 0.01, epochs = 2000
Accuracy obtained for testing data in Batch is 0.94
Accuracy obtained for training data in Batch is 0.98

Learning rate = 10, epochs = 2000
Accuracy obtained for testing data in Batch is 0.99
Accuracy obtained for training data in Batch is 0.98

Graphs are below in order 0.1, 0.0001, 0.01, 10
(Left is validation and right is training)

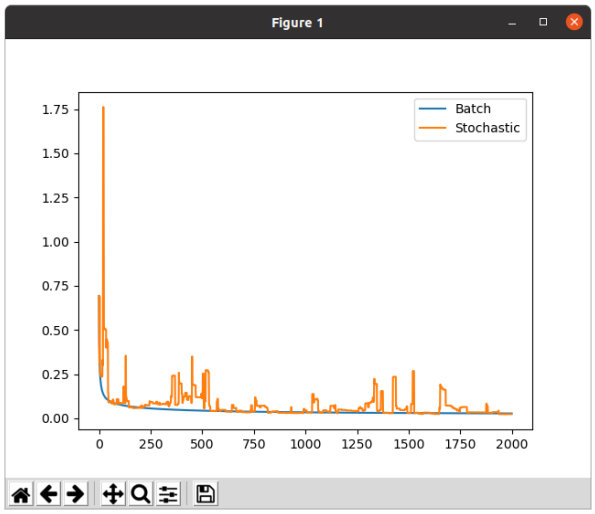


Figure 1
Batch
Stochastic
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
250
500
750
1000
1250
1500
1750
2000

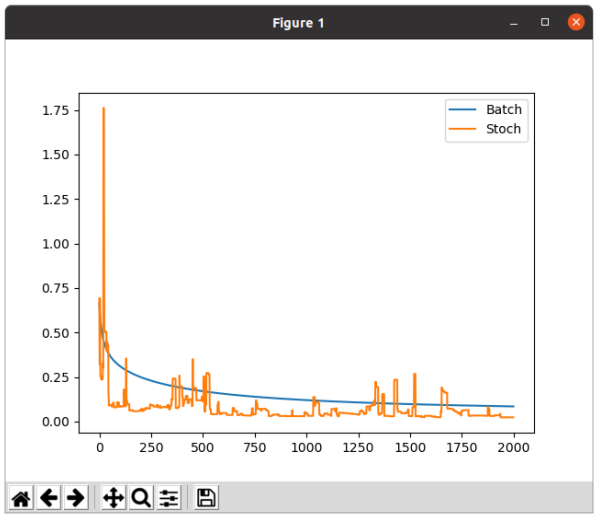


Figure 1
Batch
Stoch
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
250
500
750
1000
1250
1500
1750
2000

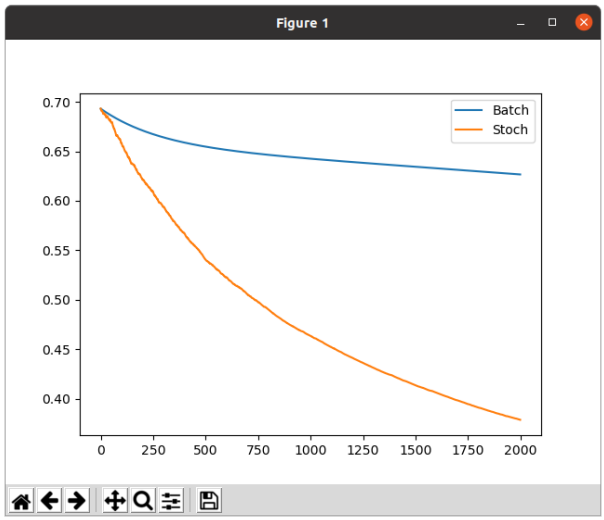


Figure 1
Batch
Stoch
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
250
500
750
1000
1250
1500
1750
2000

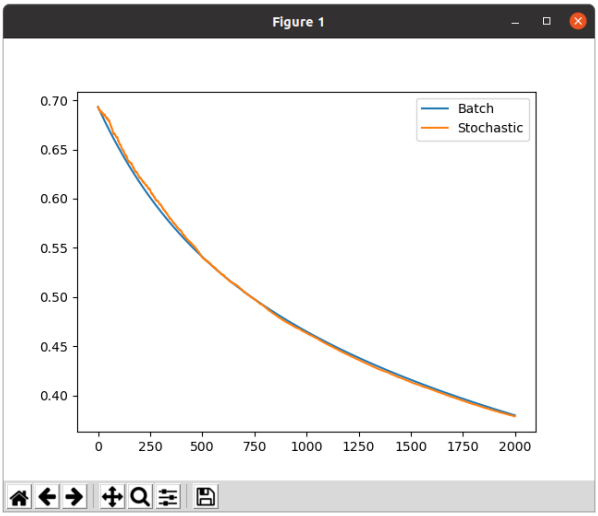


Figure 1
Batch
Stochastic
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
250
500
750
1000
1250
1500
1750
2000

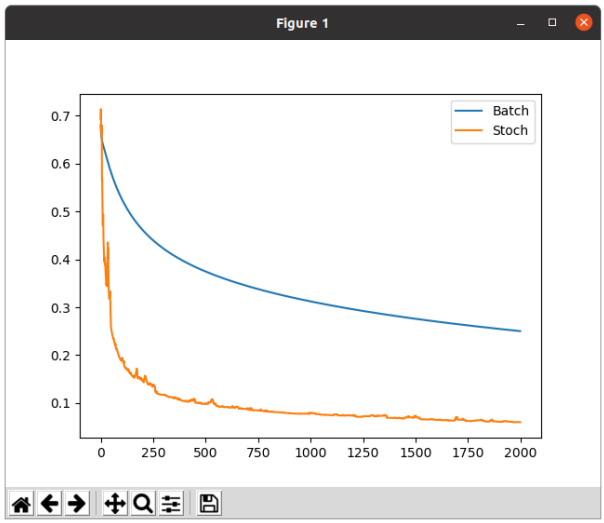


Figure 1
Batch
Stoch
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
250
500
750
1000
1250
1500
1750
2000

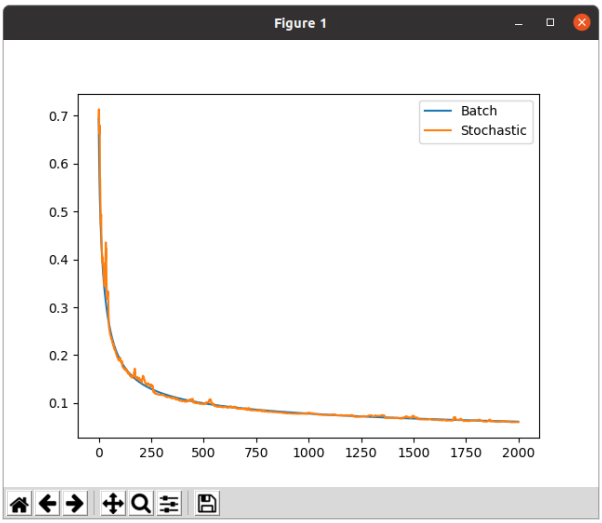
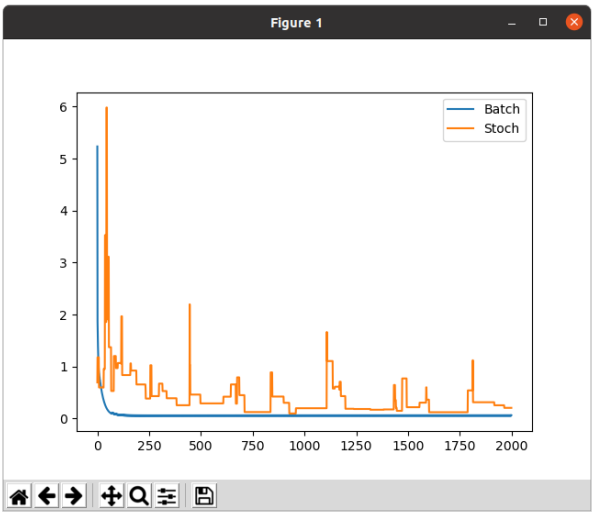
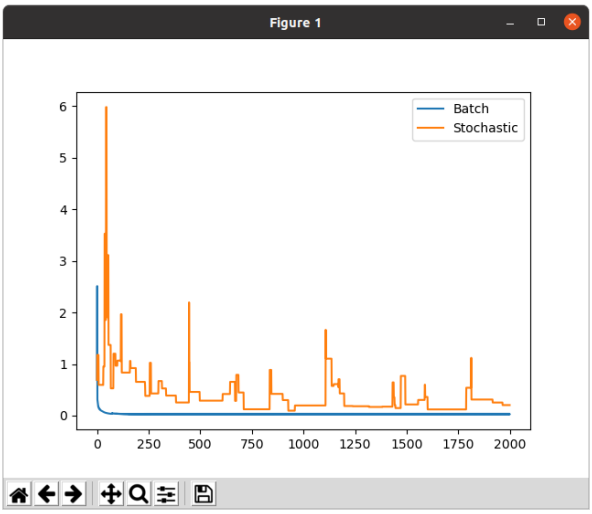


Figure 1
Batch
Stochastic
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
250
500
750
1000
1250
1500
1750
2000





a) Loss Plots
The graph of Stochastic Gradient Descent is much more noisier than that of Batch Gradient Descent. This is because instead considering the entire dataset we only choose at random to modify the parameters. However, even after the noise, after a certain number of epochs we can get good accuracy at faster speeds

b) Number of epochs taken to converge
They both take approximately the same time to converge. However, as Stochastic Gradient Descent is noisier, it may so happen that we unfortunately end up at a noisy part. An example of this would be when we took alpha as 10 and epochs as 2000, the accuracy was 90, had we taken a few more or few less epochs we would've gotten better accuracy. Hence to make sure that noise doesn't affect the data much we need to a lot more epochs than Batch Gradient Descent.

c) Implement SK Learn Logistic Regression
Implemented as a function in scratch.py in class Logistic Regression. However, we can't pass learning rate as a parameter, only max number of iterations

d) Accuracy for SK Learn Logistic Regression
Accuracy for both training and test data was 0.99 while that achieved by us have been mentioned above

Question 3 Attached at the end

Question 4 Cancer Research Question using MLE

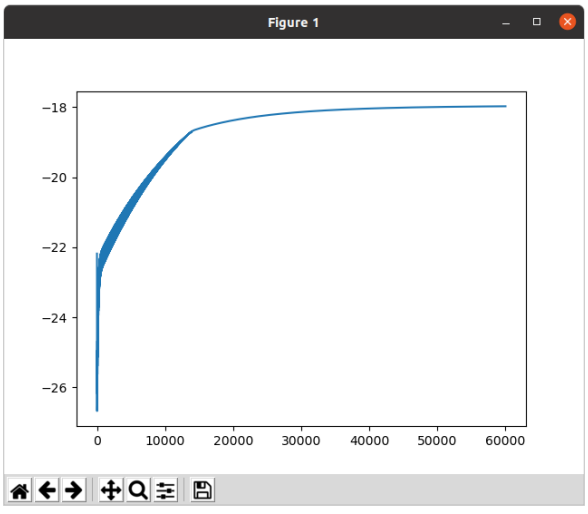
60,000 iterations were taken and 0.005 was the learning rate

a) Estimate B0, B1, B2 using MLE
[0.06145208] [0.51346168] [-4.16857388]

b) The fitted function,
F(x1, x2) = $1 / (1 + e^{(-0.061*x1 + 0.51*x2 + -4.16)})$

c) Obtain and interpret exp(B0) and exp(B1)
exp(B0) = 1.06
exp(B1) = 1.64
This means that with increase in one unit of spread, the odds of disease occurring increases by 1.06 and increase in one unit of age increases the odds by 1.64

d) What is the estimated probability that a patient with 75% of disease spread andan age of 2 years will have a recurrence of disease in the next 5 years?
We can plug x1 = 75 and x2 = 2 in our formula obtained above. We get a probability of 79%.
Attached graph of training log loss vs epochs is -



Question 5 Attached at the end

Page No.:
Date: / /

Q.3 Suppose, we choose MSE as loss fn

then cost, $J(\theta) = \sqrt{\frac{\Sigma (h_\theta(x) - y)^2}{m}}$

Now, $\frac{\partial J(\theta)}{\partial(\theta)} = \frac{1}{2\sqrt{\frac{(h_\theta(x)-y)^2}{m}}} \cdot \frac{\Sigma \cdot 2(h_\theta(x)-y) \cdot \partial(h_\theta x - y)}{m}$

Consider, $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$$\partial(h_\theta(x) - y) = \partial\left(\frac{1}{1+e^{-\theta^T x}}\right)$$

$$= \frac{-1}{(1+e^{-\theta^T x})^2} \cdot (-1) \cdot e^{-\theta^T x}$$

$$= \frac{e^{-\theta^T x}}{(1+e^{-\theta^T x})^2}$$

$$= \left(\frac{1}{1+e^{-\theta^T x}} - \frac{1}{(1+e^{-\theta^T x})^2}\right) \times (-1)$$

$$= h_\theta(x) - \left[h_\theta(x) - h_\theta(x)^2\right] \times (-1)$$

$$= (h_\theta x)^2 - h_\theta(x)$$

$$= h_\theta x (h_\theta x - 1)$$

∴ We have,

$$\frac{1}{\sqrt{\frac{\sum (h_\theta(x)-y)^2}{m}}} \cdot \sum \left(\frac{h_\theta(x)-y}{m}\right) \cdot h_\theta(x) \cdot (h_\theta(x)-1) \quad (1)$$

Now, we are given that when $y_{true} = 0$, $y_{pred} = 1$ and vice versa for some θ_j

hence, when $y_{true} = 0$, $h_\theta(x) = 1$

$$\therefore \frac{\partial J(\theta)}{\partial \theta_j} = 0 \text{ , from (1)}$$

and when $y_{true} = 1$, $h_\theta(x) = 0$

then, $\frac{\partial J(\theta)}{\partial \theta_j} = 0$, from (1)

∴ For such a datapoint gradient approaches 0

In such a scenario, if we ~~every~~ ever reach θ_j during gradient descent. then we won't be able to optimize any further. If we use Cross entropy loss instead then we get a convex fn instead where gradient becomes 0 only when convergence is achieved

Question 5

We have been $Y = X\beta + \varepsilon$, where I have considered ε as an additional vector and Y is the vector containing predicted values. Consid $\hat{Y}$, a vector containing actual values.

Now, cost $= \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$

on matrix form, this can be written as

$$C(\beta) = \frac{1}{n} \left(\hat{Y} - (X\beta + \varepsilon)\right)^T \cdot \left[\hat{Y} - (X\beta + \varepsilon)\right]$$

$$= \frac{1}{n} \left(\hat{Y}^T - \beta^T X^T - \varepsilon^T\right)\left(\hat{Y} - X\beta - \varepsilon\right)$$

$$= \frac{1}{n} \left(\hat{Y}^T\hat{Y} - \hat{Y}^T X\beta - \hat{Y}^T\varepsilon - \beta^T X^T\hat{Y} + \beta^T X^T X\beta + \beta^T X^T\varepsilon - \varepsilon^T\hat{Y} + \varepsilon^T X\beta + \varepsilon^T\varepsilon\right)$$

This is matrix form of cost fn.

Now to find β^*,

$$\frac{\partial C(\beta)}{\partial \beta} = \frac{1}{n} \left(0 - \hat{Y}^T X - 0 - X^T\hat{Y} + 2X^T X\beta + X^T\varepsilon - 0 + \varepsilon^T X + 0\right)$$

$$= \frac{1}{n} \left(2X^T X\beta - \hat{Y}^T X - X^T\hat{Y} + X^T\varepsilon + \varepsilon^T X\right) = 0$$

Also, $\hat{Y}^T X$, $X^T\hat{Y}$, $X^T\varepsilon$, $\varepsilon^T X$ are all 1×1

and $Y^T X = (X^T Y)^T$, $X^T\varepsilon = (\varepsilon^T X)^T$

$\therefore Y^T X = X^T Y$ and $X^T\varepsilon = \varepsilon^T X$

$$\therefore \frac{1}{n}\left(2x^T x\hat{\beta} - 2\hat{y}^T x + 2x^T \varepsilon\right) = 0$$

$$\therefore x^T x\beta = y^T x - x^T \varepsilon$$

$$\beta = (x^T y - x^T \varepsilon)(x^T x)^{-1}$$

$$\therefore \beta^* = (x^T y - x^T \varepsilon)(x^T x)^{-1}$$

This will exist when inverse of $x^T x$ exists.