Name: Bhavya Narang

Roll No: 2019462

ML Assignment 1

Question 1)

Preprocessing:

Appended a constant feature 1 so that we do not have to keep the equation as w.T*x+b but we can directly go for w.T*x

I will be using one hot encoding for sex as we need to have numeric values and not characters, that is, will add 3 columns

if sex=M, then columns will contain 1 0 0

if sex=F, then columns will contain 0 1 0

if sex=I, then columns will contain 0 0 1

also I will be adding another feature which will always be 1 (to get the weight w0, known as bias) hence in this case I will have 10 (3+7) input features

Then random shuffling is done.

Results obtained:

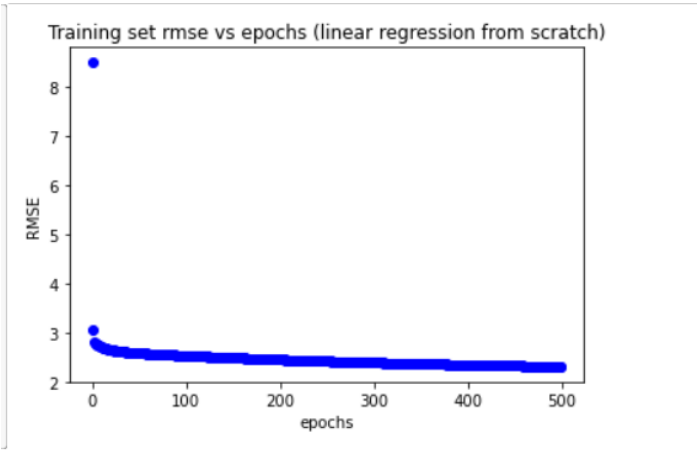
Weights for gradient descent, normal equation respectively:

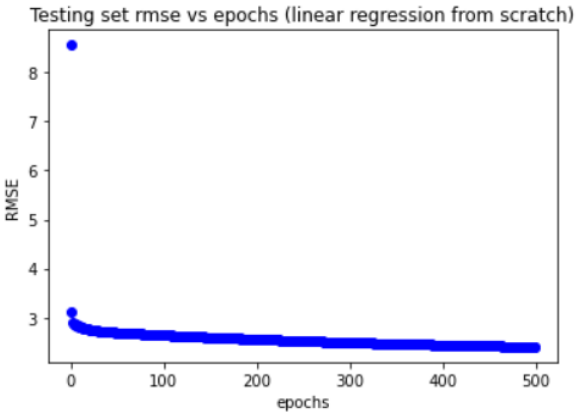
```
[[ 3.61638395]
 [ 1.66646466]
 [ 1.80694068]
 [ 0.60320573]
 [ 3.78423796]
 [ 3.46148114]
 [ 3.10294972]
 [ 2.88255042]
 [-8.29736872]
 [-0.37173961]
 [ 7.49250995]]
```

```
[[  2.82819714]
 [  1.23107879]
 [  1.24441768]
 [  0.35270067]
 [  0.46071015]
 [  9.65849988]
 [  9.83133022]
 [  8.66370115]
 [-19.28777235]
 [-10.08164727]
 [  9.12402236]]
Results using normal equation
RMSE obtained on training set: 2.1815450172812008
RMSE obtained on testing set: 2.233732312296196
```

As we can see the weights and RMSE are very close to each other hence the gradient solution is working nicely.

RMSE obtained after gradient descent:





Training set rmse: 2.25

Testing set rmse: 2.43

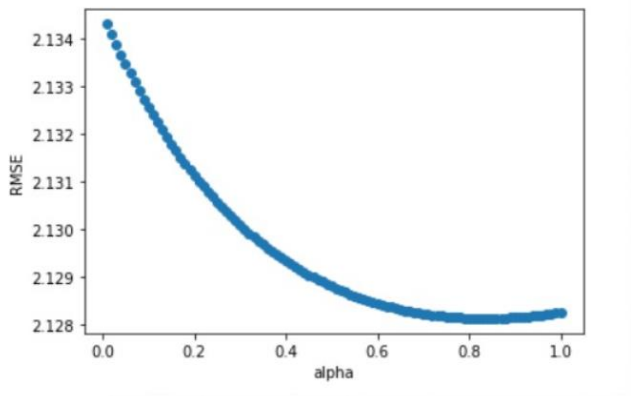
RMSE we have is similar to the RMSE obtained from the normal equation solution hence model is doing well.

Learning rate used for the above plots is: 0.4

Training is done for 500 epochs and error is decreasing gradually.

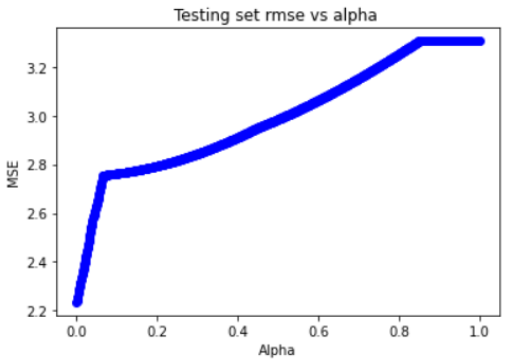
Ridge and Lasso vs Alpha, Gridsearch parameters:

Ridge:



Lasso:

```
lasso_regression(x_train,x_test,y_train,y_test)
```



```
find_best_params(x_train,x_test,y_train,y_test)
```

```
Best alpha estimate for ridge: 0.4995495495495495
Best alpha estimate for lasso: 0.0021018018018018015
Best coef of ridge [[  0.           0.29311446   0.31633844  -0.6094529    2.26100397
    7.16360971   8.15681689   7.36493183 -17.79441338  -7.79120804
   10.25734113]]
Best coef of lasso [ 0.00000000e+00 -0.00000000e+00  1.62016025e-02 -8.98358961e-01
  3.45082086e-01  8.81025497e+00  6.99382433e+00  6.99859674e+00
 -1.76085868e+01 -5.96291949e+00  1.06122607e+01]
```

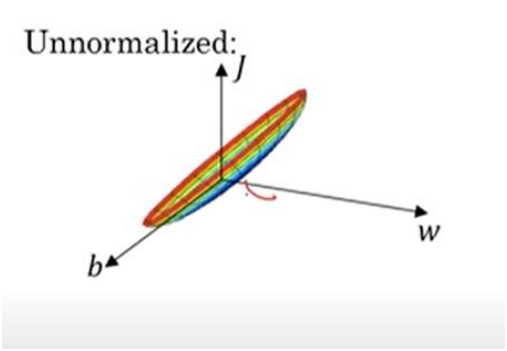
The weights obtained by gridsearch are similar to the weights obtained by normally running the models and plotting loss vs alpha as the minimum value of alpha is occurring around the best weights hence the model is performing well.

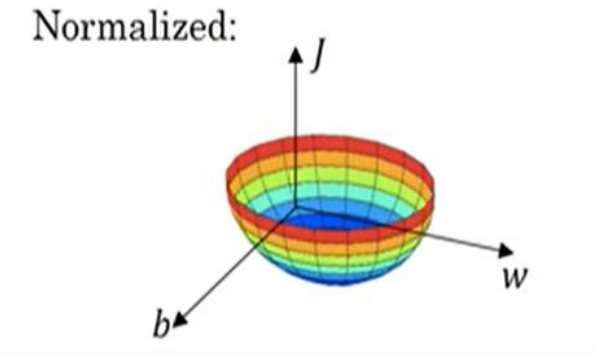
QUESTION 2)

Preprocessing:

Added one extra feature again in order to keep the w0 term in weights otherwise we have to consider another parameter b with w.

Normalized the data, that is subtracted the mean and divided by variance, so as to get similar set of weights and also it improves the time to train as the weights have a similar magnitude. Graphs for the intuition for the same.





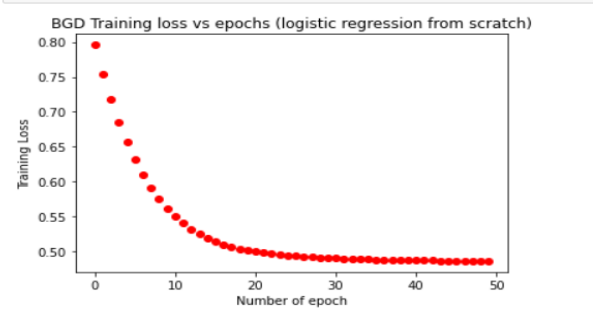
Source: https://www.youtube.com/watch?v=FDCfw-YqWTE

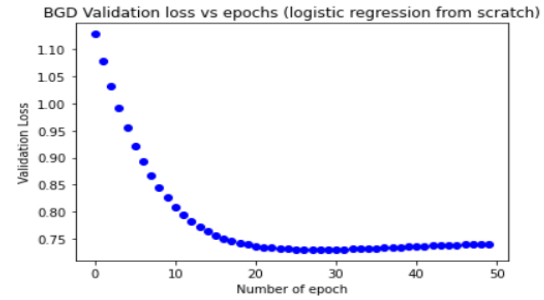
Data after normalizing:

```
df.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Extra
0	0.639530	0.847771	0.149543	0.906679	-0.692439	0.203880	0.468187	1.425067	1	1.0
1	-0.844335	-1.122665	-0.160441	0.530556	-0.692439	-0.683976	-0.364823	-0.190548	0	1.0
2	1.233077	1.942458	-0.263769	-1.287373	-0.692439	-1.102537	0.604004	-0.105515	1	1.0
3	-0.844335	-0.997558	-0.160441	0.154433	0.123221	-0.493721	-0.920163	-1.040871	0	1.0
4	-1.141108	0.503727	-1.503707	0.906679	0.765337	1.408828	5.481337	-0.020483	1	1.0
5	0.342757	-0.153085	0.252871	-1.287373	-0.692439	-0.810813	-0.817546	-0.275580	0	1.0
6	-0.250789	-1.341602	-0.987066	0.718617	0.071158	-0.125895	-0.675693	-0.615709	1	1.0
7	1.826623	-0.184362	-3.570271	-1.287373	-0.692439	0.419502	-1.019762	-0.360612	0	1.0
8	-0.547562	2.380333	0.046215	1.533551	4.019303	-0.189314	-0.947326	1.680164	1	1.0
9	1.233077	0.128406	1.389481	-1.287373	-0.692439	-4.057829	-0.723983	1.765196	1	1.0

BGD PLOTS:

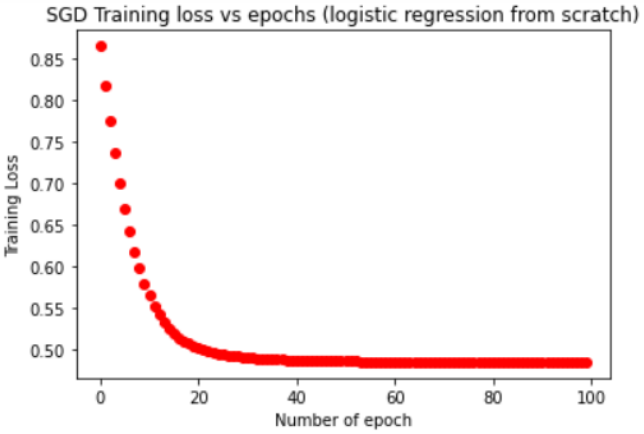


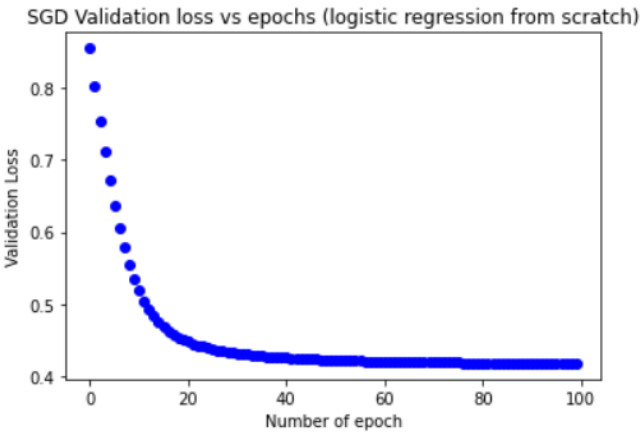


Analyis: Training loss is decreasing as it should but validation loss is increasing after the 30th epoch hence overfitting the model slightly

So number of epochs required to converge and not overfit will be around 30, also loss is quite low hence the model is tending towards low variance and low bias (if we stop training around 30 epochs, else after that variance will be high)

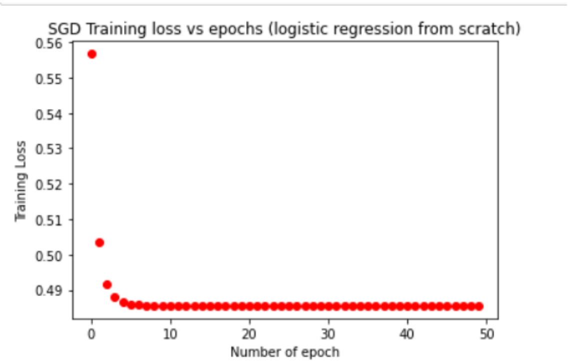
SGD PLOTS:

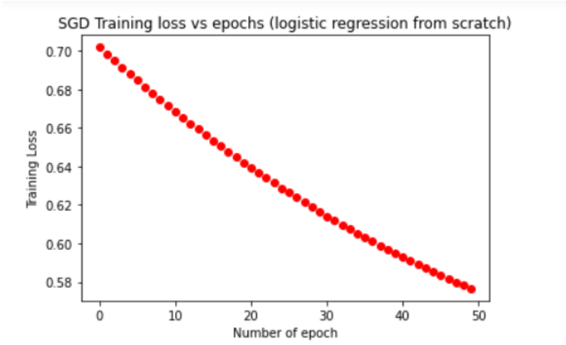


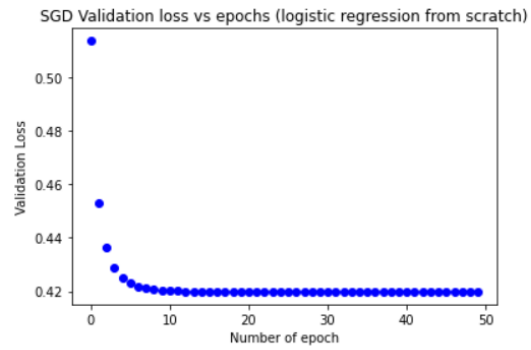


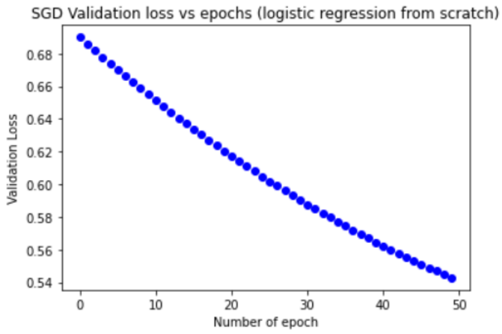
Analysis: Both training and validation loss are decreasing, hence model is tending towards low bias and low variance

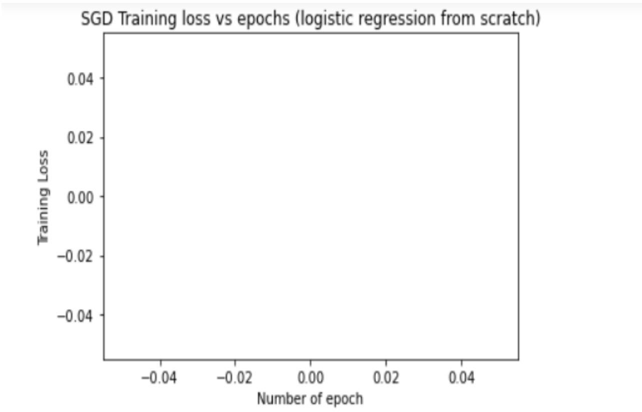
Trying different learning rates: (lr=0.01, 0.0001, 10 respectively)

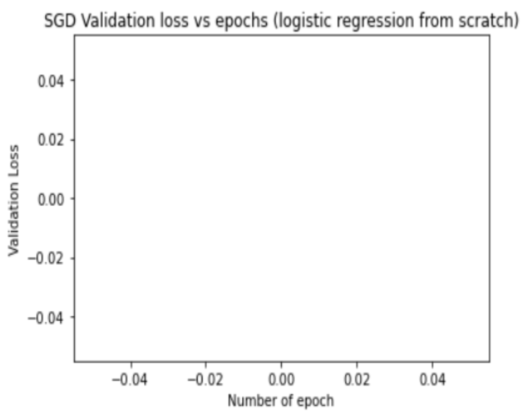












(Emty plot for lr=10 appears)

Analysis:

for 0.01 : the learning rate is good as Logistic Regression is able to converge and time is also less

for 0.0001 : the learning rate is too slow as Logistic Regression will converge very slowly (is not converging in 100 epochs)

for 10 : the learning rate is too high as Logistic Regression is unable to converge as loss is shooting up and going to infinity, so are the weights (warnings are ignored)

Hence we can see that lr=0.01 is the most optimal.

Predictions of the model on test data for model written from scratch:

Evalutation Metrics

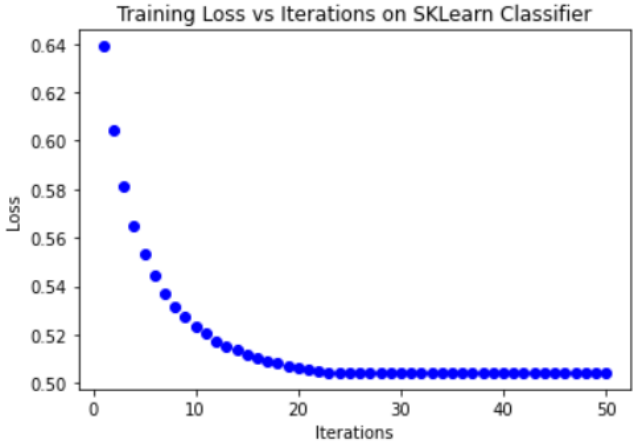
```
: final_prediction(test_data,test_labels,stochaistic_weights,batch_weights)
```

```
SGD Confusion Matrix:
[[18, 3], [13, 43]]
SGD Accuracy is: 79.22077922077922 %
SGD Precision is: 0.8571428571428571
SGD Recall is: 0.5806451612903226
SGD F1 score is: 0.6923076923076923

SGD Confusion Matrix:
[[18, 3], [13, 43]]
BGD Accuracy is: 79.22077922077922 %
BGD Precision is: 0.8571428571428571
BGD Recall is: 0.5806451612903226
BGD F1 score is: 0.6923076923076923
```

Using the sklearn Model:

```
Total training time: 0.03 seconds.
Convergence after 23 epochs took 0.03 seconds
```




```
Accuracy using sklearn model 0.7662337662337663
Precision using sklearn model 0.5161290322580645
F1 score using sklearn model 0.6399999999999999
Recall using sklearn model 0.8421052631578947
```

Analysis of the solution: It took 23 epochs to train on lr=0.001 and hence my own written model which takes 30 epochs on lr=0.0001 is doing well. Also the metrics are similar hence the scratch model is good.

QUESTION 3)

Preprocessing:

I tried binarizing but the result I got without binarizing were slightly better hence I commented out the binarizing part.

Also I divided value of each pixel by 255 so as to keep the values between 0 to 1 with the same logic as above of normalization.

Implementation:

I have used sum of log of probabilities instead of multiplication because after multiplication the number was getting very small.

Also I have converted the data into Gaussian so as to make it continuous.

Choice of K for cross validation: I have chosen k=5 for the number of folds, this is because choosing a higher k such as 10 and above was taking too much time on my system to run as we have to iterate over the dataset that many number of times. Hence I had to restrict myself for a smaller k.

Also choices of smaller k were producing varying accuracy over some cross val test sets, some were low and some were very high hence showing overfitting on the higher ones. This choice of k gives a similar 95-97 % range of accuracy. (as shown below)

```
Taking 1 fold as test set
Accuracy is: 95.45833333333333 %
```

```
Taking 2 fold as test set
Accuracy is: 96.875 %
```

```
Taking 3 fold as test set
Accuracy is: 96.375 %
```

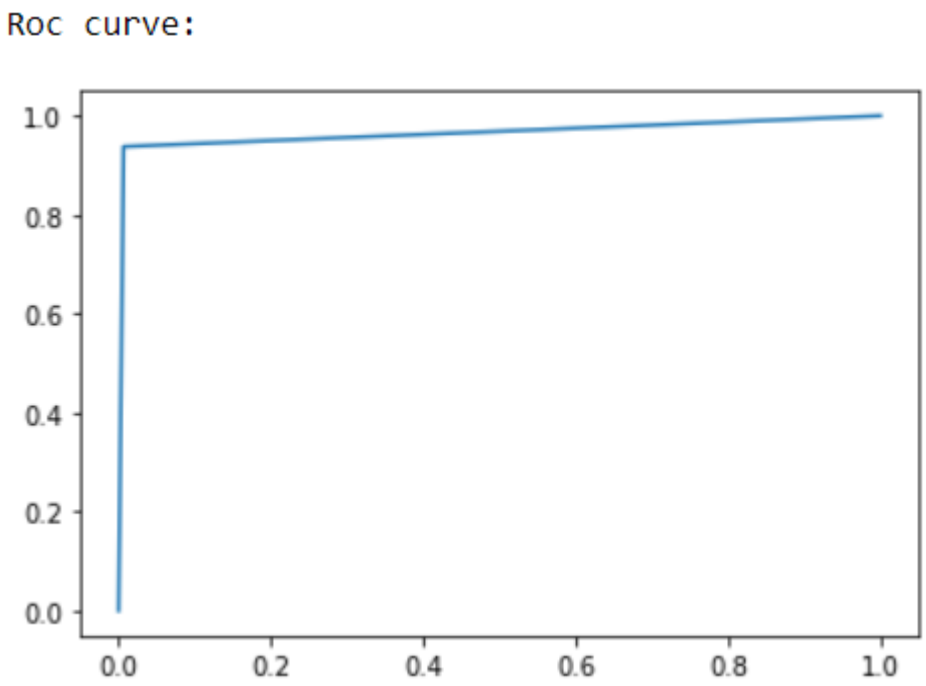
```
Taking 4 fold as test set
Accuracy is: 97.625 %
```

```
Taking 5 fold as test set
Accuracy is: 97.33333333333333 %
```

Evaluation metrics for final prediction:

```
Accuracy is: 96.4 %
[[934  66]
 [  6 994]]
Precision, recall, f1score is:
(array([0.934, 0.994]), array([0.99361702, 0.93773585]), array([0.9628866 , 0.96504854]), array([ 940, 1060], dtype=int64))
```

ROC Curve:



Question 4)

Q4] $W_i = \beta_0 + \beta_1 X_i + u_i$

$W_i \rightarrow$ wage
$X_i \rightarrow$ experience
$u_i \rightarrow$ scalar (as not mentioned)
} of ith worker.

a) $W_i = \beta_1 X_i + \beta_2 X_i' + \beta_3 X_i''$

$X_i' = \begin{cases} 1 & i = \text{male} \\ 0 & i = \text{female} \end{cases}$ | $X_i'' = \begin{cases} 0 & i = \text{male} \\ 1 & i = \text{female} \end{cases}$

I will train this model and we will have the respective weights β_1, β_2 and β_3.
β_2 acts as intercept in case of males
β_3 acts as intercept in case of female

if $\beta_2 = \beta_3 \Rightarrow$ suspicion false
$\beta_2 \neq \beta_3 \Rightarrow$ suspicion true

b) $W_i = \beta_1 X_i X_i' + \beta_2 X_i X_i'' + \beta_0$
Assuming β_0 is same for men and women.
$X_i' = \begin{cases} 1 & i = \text{male} \\ 0 & i = \text{female} \end{cases}$ | $X_i'' = \begin{cases} 0 & i = \text{male} \\ 1 & i = \text{female} \end{cases}$
Here now β_1 acts as slope for males and β_2 acts as slope for females.
$\beta_2 = \beta_1 \Rightarrow$ suspicion false
$\beta_2 \neq \beta_1 \Rightarrow$ suspicion true.

c) Train the model, $W_i = \beta_0 + \beta_1 X_i + u_i$
(the model given itself)
$\beta_1 > 0 \rightarrow$ suspicion true
$\beta_1 \leq 0 \rightarrow$ false

Date : / /
Page No.

Q4.2]

Q2] Loss $J(\theta) = \frac{1}{M} \sum_{i=1}^{M} (\hat{y} - y)^2 + \lambda w^2$

$\hat{y} = w^T x + b$ (assuming ~~logistic~~ linear regression)

$$J'(\theta) = \frac{2}{M} \sum_{i=1}^{M} (\hat{y} - y) \frac{d\hat{y}}{dw} + 2\lambda w$$

$$J'(\theta) = \frac{2}{M} \sum_{i=1}^{M} (w^T x + b - y) \times x + \lambda w$$

λw increases the cost derivative.

$w = w - \alpha \times J'(\theta)$

as cost increase w decreases.