BITS PILANI HYDERABAD CAMPUS

ASSIGNMENT-2 CS F320

Foundations of Data Science

2018A7PS1215H - Himnish Kapoor

2018A7PS1217H – Rushabh Rakesh Parikh

2018A7PS1226H – Varun Narayanan

Contents

INTRODUCTION

In this assignment we try to predict the insurance of a person by fitting a linear regression model by taking into account the features age, number of children and the bmi of a person by

1. Solving normal equations
2. Gradient Descent
3. Stochastic Gradient Descent

PREPROCESSING

➢ First we accept the dataset and convert it into a dataframe. We then calculate the mean and variance for all the columns in the dataframe. These variables are then used to standardise each entry in each of these columns in the dataframe by using the formula

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

Where u is the mean and **σ** is the standard deviation to make sure all the features of the dataframe have a mean of 0 and a standard deviation of 1 to ensure good training of the model without any bias for any feature values which have a large value.

➢ After this, we randomly shuffle the dataset 20 times and in each shuffle of the model, we split the dataset into a training set which consists of 70 percent of the data and a testing test consisting of 30 percent of the data.

➢ We append a column called the ‘0th feature’ to the beginning of the dataset(having all values 1) to make computations easier while calculating the optimal weights.

FIRST ALGORITHM - By solving normal equations

➢ The weights of the function are calculated by calling a function train where we pass in the training data.
➢ The values of the weights are calculated by using the formula

$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

➢ X is the training data which has all the input features.
➢ y has all the target attributes of the training data.
➢ Theta cap represents the set of weights which we find by using the above formula.
➢ Minimum training and testing errors -
  - Achieved using Root Mean Squared Error(RMSE) are-
    - For the training set, mean RMSE = 0.9391708182726199 and variance RMSE = 0.0001880040508600869
      Minimum training error obtained among all 20 models = 0.9097938326398585 and weights associated with it are [-0.02616455 0.28014111 0.12667103 0.07141934]

    - For the test set, mean RMSE =0.9352961111353363 and variance RMSE = 0.0009966195478696414
      Minimum testing error obtained among all 20 models = 0.8974805409031043 and weights associated with it are [0.03095922 0.29496258 0.19909846 0.06845833]

- Achieved using sum of square of errors are -
  - For the training set -
    mean sum of square of errors = 412.8898074847315
    variance of sum of square of errors = 143.8310853643415
    Minimum training error obtained among all 20 models = 387.3666991581832 and weights associated with it are [-0.02616455 0.28014111 0.12667103 0.07141934]

  - For the test set -
    Mean of sum of square of errors = 176.21091324394159
    Variance of sum of square of errors = 144.961064614359
    Minimum testing error obtained among all 20 models = 161.89973558124547 and weights associated with it are [0.03095922 0.29496258 0.19909846 0.06845833]

SECOND ALGORITHM- Gradient Descent -

- It is an iterative algorithm to minimize the loss function by updating the weights at each iteration. It makes small steps in the negative direction of slope at each point of the loss function until it reaches the global minimum considering all the training examples at a time in one iteration. The loss function we optimize is the sum of squares of errors given by **Loss = ½*$\Sigma^{N}_{i=1}$(ypred - yreal)** where n is the number of training examples.

- We train the model using an optimal learning rate of 0.001.
- At first we initialize the weight vector with random values.
- In each iteration, we modify the weight vector such that the error value converges to the lowest possible error which is repeated 10000 times
- The weights are modified by using the formula -

$$w=w-1/n*\alpha dw$$

Where w=weight vector
α=learning rate

dw=gradient(partial derivative of Error function with respect to each of the weight parameters)

We vectorized gradient descent and found dw by using -

$$dw = AB$$

Given that-

$$A = X\_train^T$$

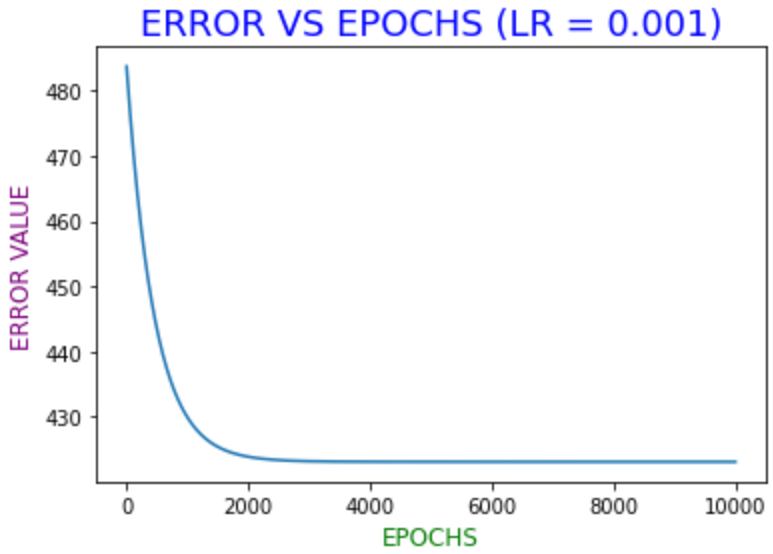
$$B = X\_train * w - y\_train$$

Where X_train contains values of the features of all the training data points and y_train contains the values for all the target attributes of the training data.

- ➢ After all the iterations are completed, we return the weight vector as well as the error value between the predicted values and the actual values from the gradient descent function.
- ➢ The error metric we used to evaluate the test data are the root mean squared error and the sum of squares of errors.
- ➢ Root means squared error is given by $\sqrt{\Sigma^{N}_{i=1}(\hat{y}_i - y_i)}$
- ➢ The sum of squared errors are given by $\Sigma^{N}_{i=1}(\hat{y}_i - y_i)$
- ➢ Here the $\hat{y}_i$ is the predicted output and y_i is the actual output.

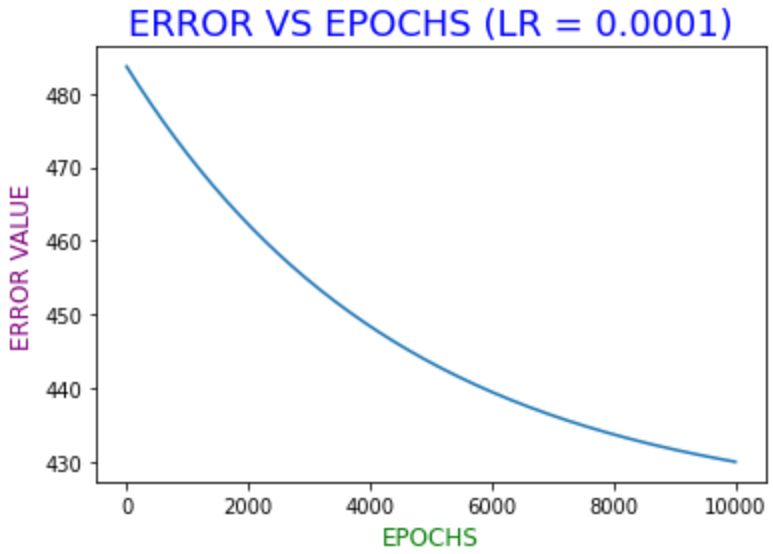
ERROR VS EPOCH WITH DIFFERENT LEARNING RATES FOR GRADIENT DESCENT-

- ➢ ERROR VS EPOCH FOR A GOOD LEARNING RATE(Learning rate = 0.001)



In this case the error function converges at a very good rate and is optimal for learning.

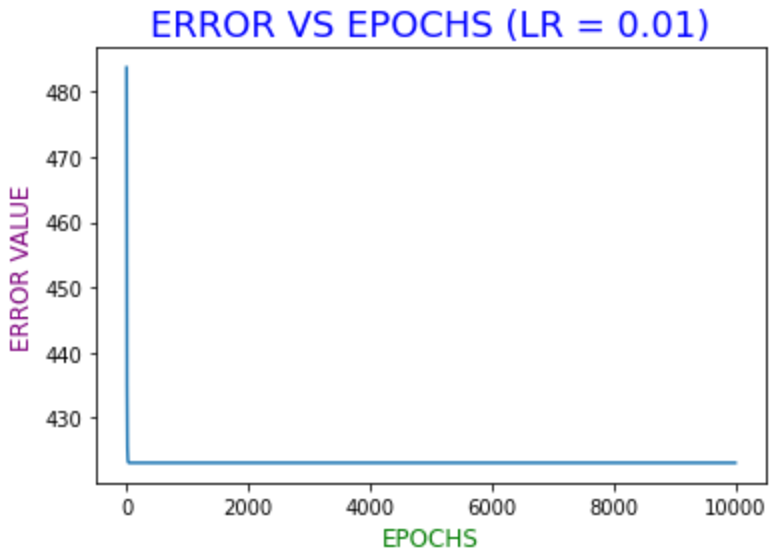
➢ ERROR VS EPOCH FOR A SLOW LEARNING RATE(Learning rate = 0.0001)



Here the learning rate is very slow and as a result the model converges very slowly. This means that it will take a lot of epochs to get close to the minimum error value.

➢ ERROR VS EPOCH FOR A FAST LEARNING RATE(Learning rate = 0.01)

➢



This learning rate is a little big and as a result the error value converges very fast. If we take an even higher learning rate, we could get a scenario where the error ends up diverging instead of converging and as a result the error value and the weights explode.

➢ Minimum training and testing errors for learning rate 0.001-

- Achieved using Root Mean Squared Error(RMSE) are-
  - For the training set, mean RMSE = 0.93917781827262 and variance RMSE = 0.00018820405086008652
    Minimum training error obtained among all 20 models = 0.9097838340915366 and weights associated with it are
    [[-0.02613632]
    [0.28011503]
    [0.12670654]
    [0.07143221]]

  - For the test set, mean RMSE = 0.9357961109726955 and variance RMSE = 0.0009966495297518557
    Minimum testing error obtained among all 20 models = 0.897485001077701 and weights associated with it are
    [[0.03097568]

[0.29493936]
[0.19912845]
[0.06846843]]

- Achieved using sum of square of errors are -
  - For the training set -
    mean sum of square of errors = 412.8898074847315
    variance of sum of square of errors = 143.83108536434108
    Minimum training error obtained among all 20 models = 387.3667003943708 and weights associated with it are
    [[-0.02613632]
    [0.28011503]
    [0.12670654]
    [0.07143221]]

  - For the test set -
    Mean of sum of square of errors = 176.21891317911604
    Variance of sum of square of errors = 144.962064614359
    Minimum testing error obtained among all 20 models = 161.90134475904762 and weights associated with it are
    [[0.03097568]
    [0.29493936]
    [0.19912845]
    [0.06846843]]

Third Algorithm- Stochastic Gradient Descent

➢ It is an iterative algorithm to minimize the loss function by updating the weights at each iteration. It makes small steps in the negative direction of slope at each point of the loss function until it reaches the global minimum considering only one training example in one

iteration. The loss function we optimize is the sum of squares of errors given by
Loss = ½*(ypred - yreal) in a single iteration.

- ➢ We train the model using a learning rate of 0.001.
- ➢ We take individual data points sequentially in a loop and update the weights with respect to that training example.
- ➢ We start by initializing the weight vector with random values.
- ➢ The weights are modified using -

$$w=w-\alpha dw$$

Where w=weight vector
α=learning rate
dw=gradient(partial derivative of Error function with respect to each of the weight parameters)

The Vectorized implementation of the gradient is given by:-

$$dw= AB$$

Given that-

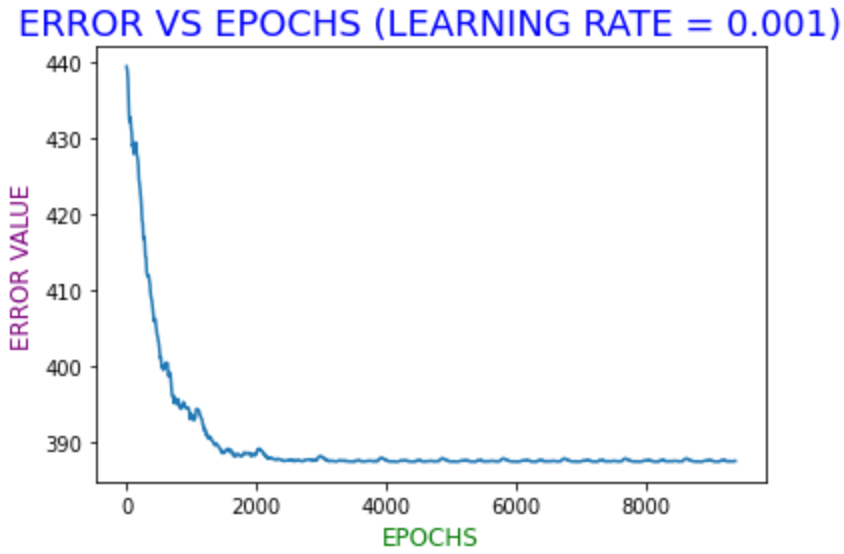
$$A= X\_train\_{}_j^T$$

$$B= X_jw-y\_train_j$$

Where X_train_j is the matrix consisting of the values of all the features of the data point picked in the jth iteration and y_train is the target variable of the data point picked in the jth iteration from the training set.

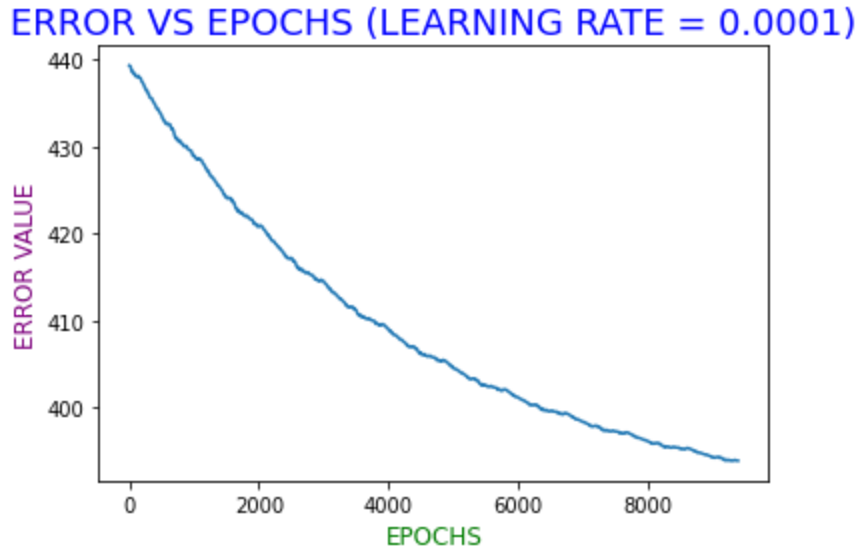
ERROR VS EPOCH WITH DIFFERENT LEARNING RATES FOR STOCHASTIC GRADIENT DESCENT-

- ➢ ERROR VS EPOCH FOR A GOOD LEARNING RATE(Learning rate = 0.001)



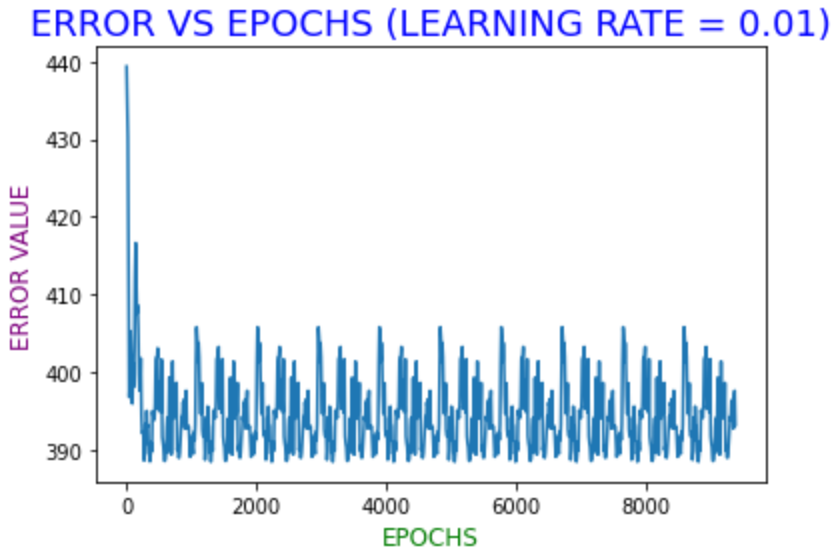
Although there is some noise(slight increases and decreases in the error value) as the weights are updated pertaining to one training example, the error value still ends up converging. Also as the size of the dataset increases, the gradient descent algorithm becomes very computationally intensive so SGD is a better algorithm to work with.

- ➢ ERROR VS EPOCH FOR A SLOW LEARNING RATE(Learning rate = 0.0001)
- ➢



The learning rate is very small and as a result of the model takes a very long time to converge at the minimum error value.

➢ ERROR VS EPOCH FOR A FAST LEARNING RATE(Learning rate = 0.01)



Here we can see that gradient descent's error just shoots up and this is because it finds the exact gradient and each iteration misses the minima and grows to an even greater value, which is astronomical in this context.

➢ Minimum training and testing errors for learning rate 0.001-

- Achieved using Root Mean Squared Error(RMSE) are-
  - For the training set, mean RMSE = 0.9393504581724382 and variance RMSE = 0.00018910583963801714
    Minimum training error obtained among all 20 models = 0.9099520578651441 and weights associated with it are [-0.02616455 0.28014111 0.12667103 0.07141934]
  - For the testing set, mean RMSE =0.936356194824287 and variance RMSE = 0.0010141155286432495
    Minimum testing error obtained among all 20 models = 0.8970754398474309 and weights associated with it are [0.03095922 0.29496258 0.19909846 0.06845833]

- Achieved using sum of square of errors are -
  - For the training set -
    mean sum of square of errors = 413.04200610273466
    variance of sum of square of errors = 147.67808174309795
    Minimum training error obtained among all 20 models = 387.5099658828889 and weights associated with it are
    [[-0.01705424]
    [0.28000808]
    [0.11214165]
    [0.07440171]]

  - For the testing set -
    Mean of sum of square of errors = 176.31253609016798
    Variance of sum of square of errors = 144.962064614359
    Minimum testing error obtained among all 20 models = 161.7536133002698 and weights associated with it are
    [[0.02504126]
    [0.29423704]
    [0.20254317]
    [0.06616653]]

Amongst Gradient Descent, Stochastic Gradient Descent and finding weights by normal equations, the minimum error is achieved by using Normal equations as we find the exact values of the optimal weights. Gradient Descent and Stochastic Gradient Descent however give error values very close to that of normal equations.

POINTS TO PONDER ON-

- ➢ Though all the three methods give similar results the most efficient method is the stochastic gradient descent algorithm in real world applications since the datasets we have to work in reality are quite large and the other two methods are very computationally intensive to work with.

- NORMALIZATION/STANDARDISATION helps to bring all the data down to the same scale. Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias while training the model and hence it is better to normalize/standardize our features. In the case of standardization, we standardize all data to a mean of 0 and a standard deviation of 1.
- Increasing the number of training iterations does decrease the loss. However, when the number of training iterations becomes very large, the reduction in error becomes very low as the error vs epochs graphs converge. When the number of training iterations become very very large, the error remains almost constant.
- One of the differences between gradient descent and stochastic gradient descent algorithms is that there was a little noise present in the error vs epochs graph of stochastic gradient descent as we update the weights of the model pertaining to only one training example at a time. So if a certain training example was very noisy, the effect on the error vs epochs graph was seen due to this training example. However the overall nature of the error vs epochs graph for Stochastic Gradient Descent converges. The error vs epochs graph for gradient descent gave us a very smooth and converging graph.
- If a very large value had been chosen as the learning rate, then we would not have been able to converge to the minima point of the error function . As a result of this, the error function would end up diverging And the weights could keep shifting between a positive gradient and a negative gradient and both the error and the weights could end up exploding.
- It is possible that the minimum error achieved could be different if we didn't use a bias term. Let us assume that a certain datapoint had all feature values as 0. If we did not use a bias term, the predicted target output would be 0. But in reality, the actual target value may be non 0 and as result we would have an error associated with that datapoint. If we didn't use a bias term, we would have a larger error associated with a data point that has all feature values 0.
- The optimal weight vectors after training signify how much of an influence a particular feature has on the target attribute. In all the

methods, the **AGE** feature had the maximum effect on the insurance prediction as the weight value associated with this feature had the largest value and the **NUMBER OF CHILDREN** feature had the lowest effect on the insurance prediction as the weight value associated with this feature had the lowest value.