Assignment-1 Report

1. Introduction:

In this assignment, we tried to have an understanding about the various optimization techniques for linear regression model training.

The following methods were implemented using python, numpy and matplotlib:

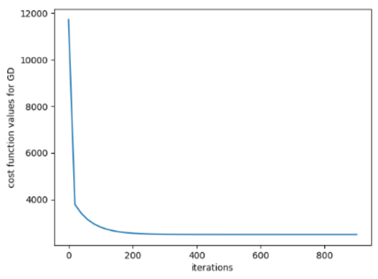
1. Gradient Descent
2. Stochastic Gradient Descent
3. GD with L1 regularization
4. GD with L2 regularization
5. Normal Equation solution

The given data consisted of 434874 rows and 2 columns (latitude and longitude). Before the model was trained, all data features were max-min normalized (column wise), shuffled and split into training (70%), validation (20%) and test data (10%).
Stopping Criteria: `E-E’ < 10`$^{-5}$

2. Optimization Algorithms:

2.1 Gradient Descent:

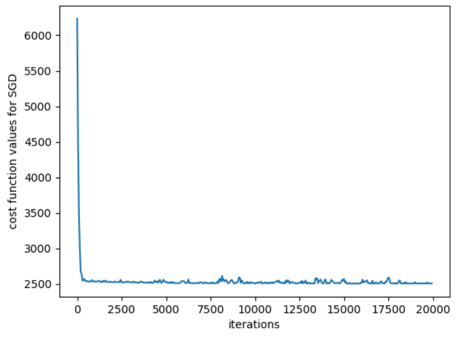
In this model, we attempted to replicate a rudimentary Gradient Descent algorithm without any regularization terms. The equation assumed to be for gradient descent fit was $y = w_0 + w_1x_1 + w_2x_2$. The error function (cost function) associated with this was $E = \frac{1}{2}\sum_{i=0}^{N}(y_i - Y_i)^2$, where y_i denotes the predicted value and Y_i denotes the actual target value. The estimated learning rate, in this case was around 0.0000005 and the model converged to a minimum squared error of 2502.667 after 900 iterations. The weights obtained in this case were [0.20843566, -0.09952162, 0.0956096] which were updated after every iteration (batch size = 304113)



`np.random.randn()` method was used to initialize the weights to random values initially. The R^2 and RMSE error in this case were 0.027390 and 19.09221 respectively.

2.2 Stochastic Gradient Descent:

In this model, a rudimentary SGD was applied on the given training set. The error/cost function in this as well as held to be $E = \frac{1}{2}\sum_{i=0}^{N}(y_i - Y_i)^2$. The weights were updated after every data point analysis (batch size = 1). The squared error converged to an average minimum of 2507.0480 after 150 iterations, but kept fluctuating between 2530 and 2500. The weights (initially initialized to random values) obtained after training were [0.20441129, -0.11359489, 0.10721501] and the R^2 and RMSE values were 0.02859 and 19.0249 respectively.



2.3 Gradient Descent with L1 regularization (Lasso):

In this model, the cost function was modified to $E = \frac{1}{2}\sum_{i=0}^{N}(y_i - Y_i)^2 + \frac{\beta}{2}(\sum |W_i|)$ where β is the L1 regularization coefficient. This technique helps us to curb the higher values of weights. A list of L1 values ([0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95]) were used for training and the L1 hyper parameter which gave the minimum VLE was considered. The average minimum error obtained was 2510.24742 with an average VLE of 717.667. The final weights and β value obtained were [0.2086843, -0.1008773, 0.09468503] and 0.05 respectively. We notice that there is not much effect of L1 regularization on this model as we are using a linear regression model of degree one. The R^2 and RMSE values obtained finally were 0.026627 and 18.9852 respectively.

2.4 Gradient Descent with L2 regularization (Ridge):

In this model, the cost function was modified to $E = \frac{1}{2}\sum_{i=0}^{N}(y_i - Y_i)^2 + \frac{\beta}{2}(\sum W_i^2)$ where β is the L2 regularization coefficient. This technique helps us to curb the higher values of weights. A list of L2 values ([0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95]) were used for training and the L1 hyper parameter which gave the minimum VLE was considered. An average minimum error of 2506.2064 with an average VLE of 718.261. The final weights and β value obtained were [0.20889902, -0.101212, 0.090636] and 0.05 respectively. We notice that there is not much effect of L2 regularization on this model as we are using a linear regression model of degree one. The R^2 and RMSE values obtained finally were 0.02822 and 18.9209 respectively.

2.5 Solution using Normal Equations:

In this method, a simple linear equation solution was followed. The partial derivate of the error with respect to each weight were set to 0, and the resulting three equations were solved for the three unknown weights. The linear equations obtained were as follows:

- $w_0 N + w_1 \sum_{i=0}^{N} x_{i2} + w_2 \sum_{i=0}^{N} x_{i3} = \sum_{i=0}^{N} y_i$
- $w_0 \sum_{i=0}^{N} x_{i2} + w_1 \sum_{i=0}^{N} x_{i2}^2 + w_2 \sum_{i=0}^{N} x_{i2}.x_{i3} = \sum_{i=0}^{N} x_{i2}.y_i$
- $w_0 \sum_{i=0}^{N} x_{i3} + w_1 \sum_{i=0}^{N} x_{i2}.x_{i3} + w_2 \sum_{i=0}^{N} x_{i3}^2 = \sum_{i=0}^{N} x_{i3}.y_i$

By taking w_0, w_1, w_2 as a column vector from all the above equations, we can represent the same in a matrix notation as **AW = B** and **W = BA^{-1}.** The final weights obtained were [0.20863356, -0.0988354, 0.09444759]. The R^2 and RMSE values obtained finally were 0.02791 and 18.81713 respectively.

3. Conclusions:

- The fastest and most accurate prediction of the parameters was obtained using normal equations, as they give a straightforward mathematical solution.
- Gradient Descent converged after 150 odd iterations and the thereafter the error almost remained constant with no fluctuations, and the weights obtained were quite close to the actual value obtained by normal equations.
- All the data was shuffled randomly prior to running the SGD algorithm to ensure that the model trained on random points without an order. The SGD model converged after 1000 odd iterations but the value kept fluctuating between 2530 and 2500. The weights generated in this case were almost close to the real weights (obtained from normal equations). In addition, the training time in case of SGD was comparatively lesser than the other methods.
- L1 and L2 regularization were not of much significance in this model as the weights were quite small and there was no over-fitting.