**Adaptive Gradient Descent Optimizer - Equations**

We can define the following equation as the hypothesis function. Where θ is a matrix containing the relative weights(coefficients in linear regression) and x is matrix of relevant data.

A picture containing clock

Description automatically generated

The following function is the cost function that calculates total cost of the dataset with current valued parameters(mean squared error).

A picture containing object, clock

Description automatically generated

The following equations represent the gradient updates with respect to each feature in x. It computes the rate of change with respect to feature θi .

A picture containing knife, table

Description automatically generated

The Adaptive Gradient Descent Optimizer also known as AdaGrad when gradients increases the learning rate decreases and when the gradients decreases the learning rate increases. Essentially, the learning rate value changes. The value of the learning rate is constantly changing depending on the gradient updates.

The Adaptive Gradient Descent optimizer update equation is as follow:

A picture containing object, clock

Description automatically generated

|  |
| --- |
| **ISTANBUL STOCK EXCHANGE Data Set**  Data sets includes returns of Istanbul Stock Exchange with seven other international index; SP, DAX, FTSE, NIKKEI, BOVESPA, MSCE\_EU, MSCI\_EM from Jun 5, 2009 to Feb 22, 2011. |

About the dataset:

**Target** is to predict the price of the Istanbul stock exchange national 100 index (ISE in Lira currency and USD in Dollars).

**From** SP 500 return index, Stock market return index of Germany, Stock market return index of UK, Stock market return index of Japan, Stock market return index of Brazil, MSCI European index, MSCI emerging markets index

We performed a thorough and rigorous analysis of the dataset. We started by checking the linearity of the dataset.

A picture containing large, sitting, parked

Description automatically generated

We then moved on to apply a heat to choose attributes with correlation higher than 50%.

A screenshot of a video game

Description automatically generated

**Final Fitted Model Performance**

After running our final fitted model with the optimal parameters(thetas/weights). The model performance metrics from our model to the test data set is as follows:

﻿MSE 0.00021469897563200808

R2 0.4634065153987402

Adjusted R2 0.4371029132124039

These values can be found by on the final\_model.txt file that is created at the end of the part1.py. The mean squared error is low and R squared and Adjusted R squared are moderately appropriate and very close to the standard model implementation given by the Python Libraries.

Below we can see plots from running the model with different parameters:

For 100 iterations using .01, .001 and .0001 learning rate value.

A picture containing large

Description automatically generatedA close up of a person

Description automatically generatedA close up of a person

Description automatically generated

For 250 iterations using .01, .001 and .0001 learning rate value.

A picture containing person

Description automatically generatedA close up of a mans face

Description automatically generatedA picture containing person

Description automatically generated

**Final Fitted Model Equations**

**Using custom Adaptive Gradient Descent Optimizer:**

[0.001023]intercept+[-0.004590]x1 +[0.01299]x2+[0.0001234]x3+[0.009954]x4

**Using standard model implementation given by the Python Libraries:**

[0]intercept + [-.2280]x1 + [-.2809] x2 + [1.1369]x3 + [.9066]x4

**Are you satisfied that you have found the best solution?**

Yes, from the output file final\_model.txt we can see that the model performance metrics are with acceptable deviations.

Below is a table containing the model performance metrics for Linear Regression, Stochastic Gradient Descent.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Linear Regression | SGD | AdaGrad |
| MSE | .00019121 | .0001921 | ﻿0.0002146 |
| R2 | .52208 | .52847 | ﻿0.43710 |

From the table we can conclude that our model is around 10% away from the standard model implementation given by the Python Libraries.

**Are you satisfied that the package has found the best solution.**

While the libraries readily available give an acceptable solution, I would argue that it could always be better. Given that the Standard Linear Regression algorithm performs worse than the Stochastic Gradient Descent, there would most likely be an algorithm that performs better than both of these. However, given that many of the augmented gradient descent algorithms do not have a standard library implementation, I would say that the SGDRegression gives the best “ready-made” solution.

﻿