****

**INDEPENDENT UNIVERSITY,**

**BANGLADESH**

**Artificial Intelligence (CSE425)**

**Report**

**Final Assignment**

**Topic: Multiple Regression Models**

**Faculty: Amin Ahsan Ali, Associate Professor, Dept of CSE, IUB**

**Email:** [aminali@iub.edu.bd](mailto:aminali@iub.edu.bd)

**Prepared by  
Syed Niaz Mohtasim (2021607)**

**Date of Submission: 5th September 2023**

**Semester: Summer 2023**

# What is Regression?

Regression is a statistical technique used in finance, investing, and other fields that aims to ascertain the nature and strength of the relationship between a single dependent variable (often represented by Y) and a number of additional factors (sometimes referred to as independent variables). [1]

## Linear Regression

Linear regression is a statistical method that can be used to model the relationship between one or more explanatory variables (also known as independent variables) and a dependent variable. The goal of linear regression is to find a line that best fits the data points, or to estimate the value of the dependent variable based on the values of the explanatory variables.

If we have more than one explanatory variable, we can use multiple linear regression, which extends the simple linear regression model by adding more terms to the equation. The equation for multiple linear regression is y = b0 + b1x1 + b2x2 + … + bkxk, where y is the dependent variable, x1, x2, …, xk are the explanatory variables, b0, b1, b2, …, bk are the coefficients of the terms. [2]

# Gradient Descent Functions

Gradient descent functions are functions that are used to optimize the parameters of a machine learning model by moving in the direction of the steepest decrease of the error or cost function. [3]

## Batch Gradient Descent

This type of gradient descent uses the entire training dataset to calculate the error and update the parameters in each iteration.

wi+1 = wi – a . ∆wi J(wi)

where w denotes the weights that need to be updated, alpha is the Learning Rate of the algorithm, a hyperparameter that controls how much to change the model in response to the estimated error, J is the cost function, while i refers to the iteration index.

## Stochastic Gradient Descent

This type of gradient descent uses a single randomly selected training sample to calculate the error and update the parameters in each iteration.

wi+1 = wi – a . ∆wi J(xi, yi,wi)

The notations are the same with Gradient Descent while y is the target and x denotes a single observation in this case.

## Mini Batch Gradient Descent

This type of gradient descent uses a small subset (or batch) of training samples to calculate the error and update the parameters in each iteration. This is a compromise between batch and stochastic gradient descent, as it balances the speed and accuracy of the parameter updates. In this project, we will be using 10 per batch.

wi+1 = wi – a . ∆wi J(xi:i+b, yi:i+b,wi)

The notations are the same with Stochastic Gradient Descent where b is a hyperparameter that denotes the size of a single batch. [4]

# Radial Basis Function

A basis function is a mathematical function that can be used to construct or approximate other functions. Sigmoid function is a mathematical function which has a characteristic S-shaped curve. There are a number of common sigmoid functions, such as the logistic function, the hyperbolic tangent, and the arctangent. [5]

# Objectives

In this assignment we were tasked to use (multiple) regression for the prediction of Chance of Graduate Admissions of students using a number of relevant features. The dataset (Admission\_Predict.csv) contains seven features arranged into columns in a CSV file. The last column is the value we were asked to predict – “Chance of Admit”.

We were also asked to choose a basis function and recreate the steps. I chose Sigmoid Basis Function and I will talk about why I chose it over polynomial and radial basis functions in the results part with numbers.

# Experimental Setup

I used Visual Studio Code as my code editor and wrote the code for the models in Python, relying on Python libraries such as numpy, pandas and matplotlib to produce charts. I created custom functions for training the linear regression model, for the gradient descent functions and MSE (mean squared error) function. For nomalising the dataset between 0 and 1, I used Min-Max normalisation.

**Iterations:** 1000

**Training Dataset:** 60%

**Learning Rate:** 0.1

**Batch Size:** For Mini-Batch Gradient, batch size = 10.

# Results

**Mean Square Error Values of Gradient Descents for Linear**

MSE for Batch Gradient Descent: 0.003915

MSE for Mini-Batch Gradient Descent: 0.003554

MSE for Stochastic Gradient Descent: 0.005155

**Prediction Accuracy**

The model got it right 189 times out of 200.

**The equation from the model:**

Chance of Admit = 0.40 + -0.05 \* GRE Score + 0.17 \* TOEFL Score + 0.02 \* University Rating + -0.02 \* SOP + 0.06 \* LOR + 0.39 \* CGPA + 0.04 \* Research  
  
**Feature Importance Ranking**

**Coefficient** **Feature**

1 0.387054 CGPA

2 0.171745 TOEFL Score

3 0.063418 LOR

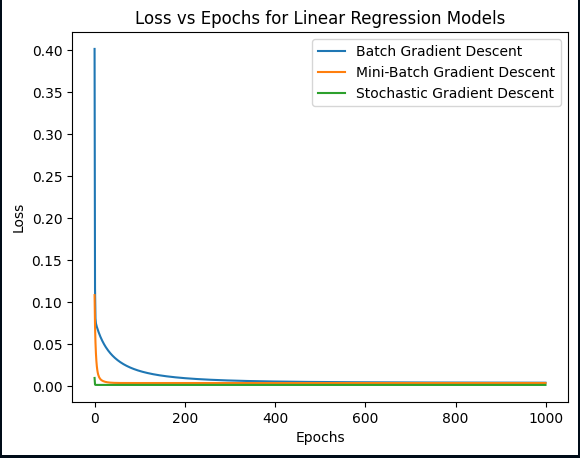
4 -0.048885 GRE Score

5 0.039227 Research

6 -0.024446 SOP

7 0.023851 University Rating

**Graph**

****

**For Basis Function:**

For basis functions, I tried three different basis functions. Polynomial (x2), radial basis and sigmoid basis functions.  
Here are the MSE values and charts.

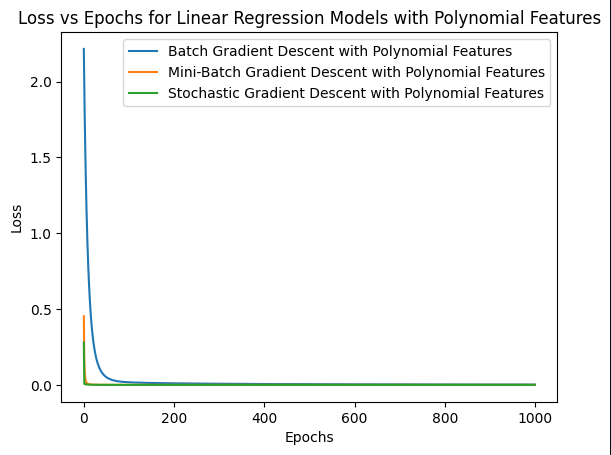
**Polynomial (x2)**

MSE for Batch Gradient Descent with Polynomial Features: 0.005648

MSE for Mini-Batch Gradient Descent with Polynomial Features: 0.003631

MSE for Stochastic Gradient Descent with Polynomial Features: 0.012330

**Graph**



**Radial Basis Function**

**The function: exp(-w \* (x - c)\*\*2)**

I tried a number of different combinations to try to tweak it for the best results. Here is a table.

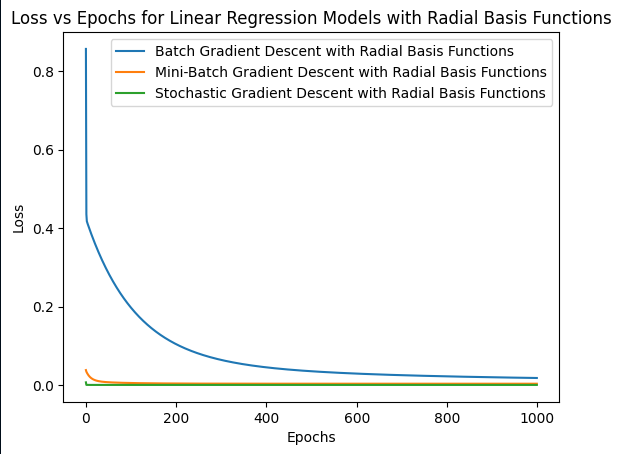
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Center | Width | Batch (MSE) | Mini Batch (MSE) | Stochastic (MSE) |
| 0.5 | 0.15 | 0.024051950792940396 | 0.038233868615417596 | 0.03599904443583522 |
| 0.5 | 0.1 | 0.021175135193772458 | 0.03843297941740976 | 0.033029444298602544 |
| 0.1 | 0.15 | 0.026007631026274475 | 0.011023800464791678 | 0.02796002147715823 |
| 0.1 | 0.1 | 0.02851207975045833 | 0.014173702325983856 | 0.02913616238122576 |
| 0.1 | 0.2 | 0.02015286325402194 | 0.02015286325402194 | 0.009252981050228527 |
| 0 | 0.2 | 0.03020008922642284 | 0.007420978166007857 | 0.023186063437956463 |
| 0.05 | 0.2 | 0.023328941232826597 | 0.007862192845341015 | 0.024825093070832756 |
| 0.05 | 0.1 | 0.02627198135223214 | 0.011616497208426087 | 0.028244629081623528 |
| 0 | 0.1 | 0.020105855473564596 | 0.009660869316760705 | 0.027102554397027755 |
| -0.9 | 0.1 | 0.014829113787817894 | 0.00415906735443143 | 0.010336006110312792 |
| -0.8 | 0.1 | 0.02801364832717309 | 0.004326348078160385 | 0.011647042042986307 |
| -0.8 | 0.2 | 0.009324353632267007 | 0.003632482441864627 | 0.005387051276612237 |
| -0.9 | 0.2 | 0.024230402468028177 | 0.003633098126201184 | 0.005387051276612246 |
| -0.9 | 0.4 | 0.008966475922230916 | 0.003534738875797605 | 0.0049247059739145485 |

The best results came at center -0.9 and width 0.2  
MSE for Batch Gradient Descent with Radial Basis Functions: 0.024230

MSE for Mini-Batch Gradient Descent with Radial Basis Functions: 0.003633

MSE for Stochastic Gradient Descent with Radial Basis Functions: 0.005387

**Graph**



**Sigmoid Basis Function**

**The function : 1 / (1 + exp(-s \* x + b))**

I tried a number of different combinations to try to tweak it for the best results. Here is a table.

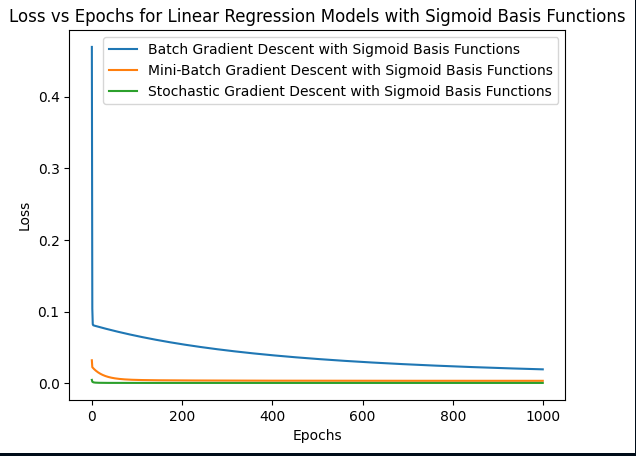
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Slopes | Intercepts | Batch (MSE) | Mini Batch (MSE) | Stochastic (MSE) |
| 10 | -5 | 0.020281485675583095 | 0.03563603689279608 | 0.0165322714135156 |
| 10 | -10 | 0.02033934297008947 | 0.03934351320056703 | 0.025985888226467652 |
| 10 | 0 | 0.013664889024466527 | 0.010244527152856609 | 0.018216316380877187 |
| 20 | 0 | 0.019544761928592908 | 0.012579609700154567 | 0.03041574744751553 |
| 15 | 0 | 0.01467312802210758 | 0.011836155825072393 | 0.027479136885554426 |
| 8 | 0 | 0.01678752683664935 | 0.00895923269519183 | 0.011792565731750582 |
| 5 | 0 | 0.012522202905247503 | 0.005953107196784291 | 0.006245753001477956 |
| 3 | 0 | 0.006063421155062785 | 0.004108647179697087 | 0.00648209494513807 |
| 1 | 0 | 0.021805645845153904 | 0.0036243964206128534 | 0.004924705973888114 |
| 0.5 | 0 | 0.011674075080149534 | 0.0037619372004554772 | 0.004414697050395433 |
| 0 | 0 | 0.020341211111111118 | 0.02000080712389166 | 0.023140131728292417 |

The best results came at slope 1 and intercept 0  
MSE for Batch Gradient Descent with Sigmoid Basis Functions: 0.021805

MSE for Mini-Batch Gradient Descent with Sigmoid Basis Functions: 0.003624

MSE for Stochastic Gradient Descent with Sigmoid Basis Functions: 0.004414

**Graph**



So for the **basis functions,** the best results

MSE for Batch Gradient Descent with Polynomial Features: 0.005648

MSE for Mini-Batch Gradient Descent with Polynomial Features: 0.003631

MSE for Stochastic Gradient Descent with Polynomial Features: 0.012330

MSE for Batch Gradient Descent with Radial Basis Functions: 0.024230

MSE for Mini-Batch Gradient Descent with Radial Basis Functions: 0.003633

MSE for Stochastic Gradient Descent with Radial Basis Functions: 0.005387

MSE for Batch Gradient Descent with Sigmoid Basis Functions: 0.021805

MSE for Mini-Batch Gradient Descent with Sigmoid Basis Functions: 0.003624

MSE for Stochastic Gradient Descent with Sigmoid Basis Functions: 0.004414

Among the linear and basis functions, Mini-Batch was more often than not, giving the lowest values among Batch and Stochastic gradient descents.

For basis, since Sigmoid Basis function gave the smallest value of MSE (with mini-batch gradient), I chose to go with Sigmoid Basis function as my basis function. But the difference was very marginal, almost insignificant. The difference came at the 6th decimal place. Both polynomial and radial basis also gave very small values for MSE and both are good choices.

# Conclusion

Regression analysis was conducted to predict the chances of graduate admissions for students using various features. The dataset contained seven relevant features, with the target variable being "Chance of Admit." Three gradient descent methods—Batch, Mini-Batch, and Stochastic—were employed to optimize the linear regression model. Notably, Mini-Batch gradient descent consistently yielded the lowest Mean Squared Error (MSE), indicating its effectiveness in model training.

The linear regression model revealed that CGPA had the most significant positive impact on admission chances, followed by TOEFL Score and Letter of Recommendation (LOR). Conversely, GRE Score and Statement of Purpose (SOP) had a negative influence, while Research and University Rating had relatively smaller effects.

Furthermore, three basis functions - Polynomial, Radial, and Sigmoid were explored. Sigmoid Basis Function produced the lowest MSE, particularly with Mini-Batch gradient descent, making it the preferred choice. However, the performance differences between the basis functions were marginal, with both Polynomial and Radial Basis functions also demonstrating strong predictive power.

In summary, this project successfully employed regression analysis to predict graduate admission chances. Mini-Batch gradient descent was the preferred optimization technique, and the Sigmoid Basis Function emerged as the most effective choice among basis functions. The results emphasize the importance of key features like CGPA, TOEFL Score, and LOR in determining admission outcomes. Overall, this analysis provides valuable insights for improving the graduate admission process.