**Fundamentals of Machine Learning (Fall 2022)**

**Homework #1 (80 Pts, Due date: Sep 21, 2022)**

**Student ID 2017314712**

**Name 김우진**

**(1)** Given training samples, , we want to find a constant that minimizes the following error function.

Assume that we have eight training samples such that .

**(a) [10 pts]** Find an analytic solution for optimal .

**Answer:**

|  |
| --- |
|  |

**(b) [10 pts]** Explain how the optimal solution is related to normal distribution .

**Answer:**

|  |
| --- |
| **만약 데이터가 엄청많고 그것의 대한 분포를 그려볼경우 그 분포가 normal distribution과 비슷하게 나오게된다. 현재 error function을 보게되면 w0가 데이터의 분포가 많은곳의 값일 수록 값이 작게 나오기 때문에 그 분포가 가장 많은 부분인 평균이 된다는 것을 알수 있다.** |

**(2)** We provide all template code and datasets in Python. Write your code to implement linear regression. You may need to install NumPy and Matplotlib libraries.

**(a) [5 pts]** Implement the util function “add\_bias” in ‘models/LinearRegression.py’. You should add a column of ones for bias after the last column of input matrix.

**Note: Fill in your code here. You also have to submit your code to i-campus.**

**Answer:**

|  |
| --- |
| row , col = x.shape  bias = np.ones(row)  x\_new = np.c\_[x,bias] |

**(b) [5 pts]** Implement the training function “numerical\_solution” in ‘models/LinearRegression.py’ using the **batch gradient descent method**. The error function is defined as follows:

**Note: Fill in your code here. You also have to submit your code to i-campus.**

**Answer:**

|  |
| --- |
| y\_hat = x\_new @ self.W  y\_new = (y - y\_hat) \*\* 2  loss\_vector = (1/(2\*num\_data)) \* y\_new.sum()  grad = -(1/num\_data)\* x\_new.T @ (y - y\_hat) |

**(c)** **[10 pts]** Implement training function “numeric\_solution” in ‘models/LinearRegression.py’ using the **mini-batch stochastic** **gradient descent method**. The error function is defined as follows:

**Note: Fill in your code here. You also have to submit your code to i-campus.**

**Answer:**

|  |
| --- |
| y\_hat = batch\_x @ self.W  y\_new = (batch\_y - y\_hat) \*\* 2  loss\_vector = 1/(2\*num\_samples\_in\_batch) \* ((batch\_y - y\_new)).sum()  grad = -(1/num\_samples\_in\_batch)\* batch\_x.T @ (batch\_y - y\_hat) |

**(d) [10 pts]** Implement the training function “analytic\_solution” in ‘models/LinearRegression.py’ using the **normal equation**.

**Note: Fill in your code here. You also have to submit your code to i-campus.**

**Answer:**

|  |
| --- |
|  |

**(3)** We evaluate our code using two datasets, “Wave” and “Diabetes.”

**(a) [10 pts]** After training the model, fill out the blank using the following metrics for the “Diabetes” dataset.

RMSE =

Write your opinion briefly for two solutions, analytic solution and gradient descent method. (Hyperparameter for Gradient: Epoch = 10,000, Batch size = 32, learning rate = 0.01, optimizer=’SGD’)

**Answer: Fill the blank in the table.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **W1** | **W2** | **W3** | **RMSE** |
| Initial value | 0.0 | 0.0 | 0.0 | 168.7240 |
| Gradient Descent | 11.78335332 | -46.9188066 | 230.93742695 | 52.7739 |
| Analytic solution | 8.32446097 | -46.17648745 | 226.87223856 | 52.6349 |

시간적으로는 gradient descent 방법이 오래 걸린다고 생각한다 반복을 많이 할수록 그 시간은 증가 될 것이다. 하지만 analytic은 공식이 있기에 사용 가능하지만 너무 복잡한 것일 경우 analytic solution이 없을 수 있다고 생각한다.

1. **[20 pts]** Given the “Wave” data, draw several plots by adjusting learning rates, where the other hyperparameters are same to (a). For each plot, the x-axis is the adjusted learning rates, and the y-axis is the error. Try at least 5 different values.

**Answer:**

|  |
| --- |
|  |

1. **[20 pts]** Given the “Wave” data, draw several plots by adjusting batch sizes, where the other hyperparameters are same to (a). For each plot, the x-axis is the adjusted batch sizes, and the y-axis is the error. Try at least 5 different values.

**Answer:**

|  |
| --- |
|  |