**Introduction to Machine Learning (Spring 2020)**

**Homework #1 (100 Pts, Due Date: Apr 29, 2020)**

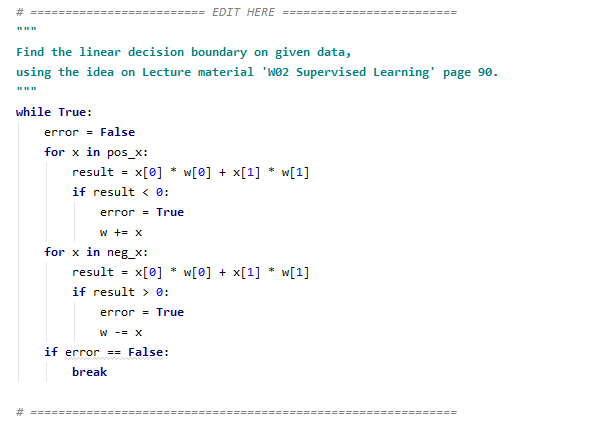
**Student ID 2015313754**

**Name 길태형**

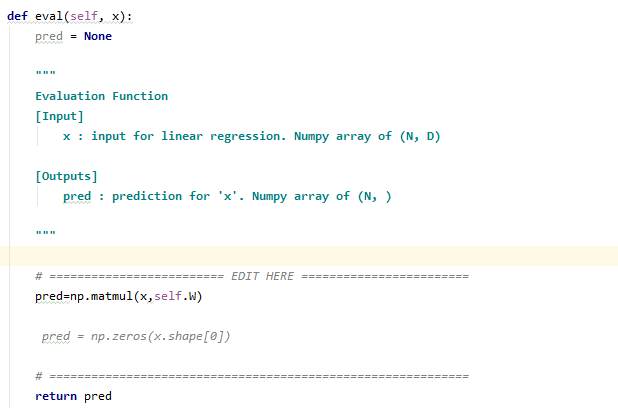
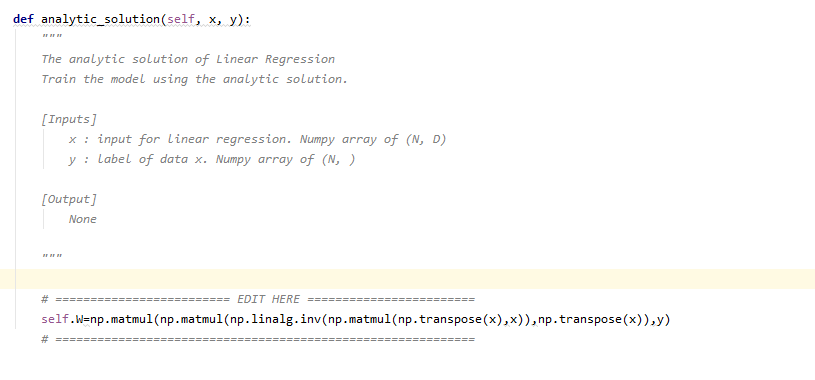
**Instruction:** We provide all codes and datasets in Python. Please write your codes to complete linear/logistic regression and SGD optimizer. You need to install numpy and matplotlib.

1. **[60 pts]** Implementation
2. **[Simple Classification]** Implement the procedure of finding the linear decision boundary vector ‘w’ on the ‘Simple classifier.py’ by adjusting angles between data and ‘w.’ After the procedure, all data points should be linearly separated.

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



1. **[Linear regression with the analytic solution]** Implement training and evaluation functions in ‘models/LinearRegression.py’ (‘analytic\_solution’ ’ and ‘eval’ respectively) using an analytic solution

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

1. **[Linear regression using Gradient Descent]** Implement training and evaluation functions in ‘models/LinearRegression.py’ (‘numerical\_solution’) with the **numerical solution**. The training model is on the minibatch gradient, not the whole data. Given a mini-batch data the error function for a mini-batch is defined as follows:

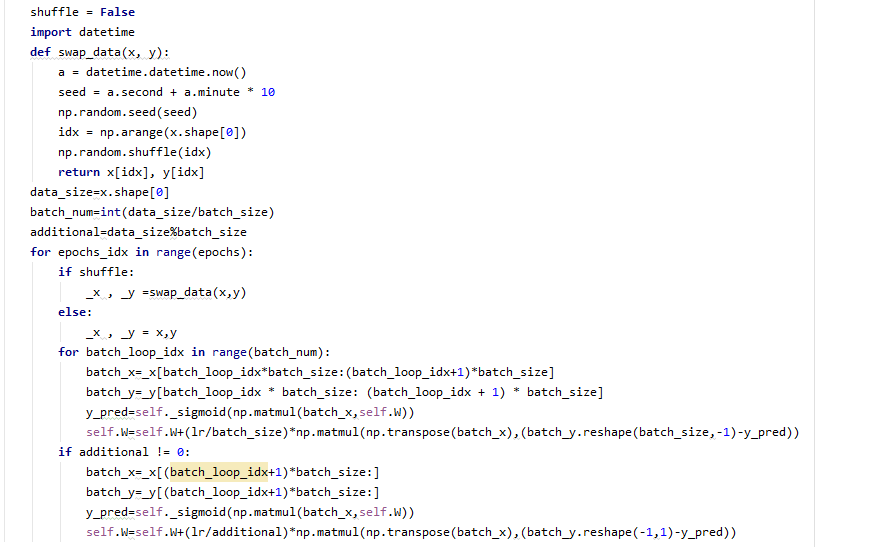
***Answer: Fill your code here. You also have to submit your code to i-campus.***

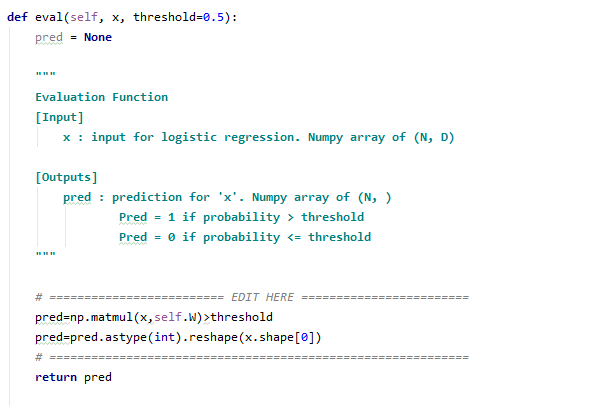
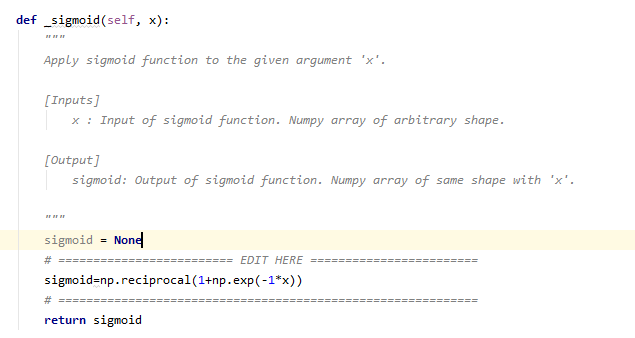


1. **[Logistic regression]** Implement the following codes: training, evaluation, and sigmoid functions in ‘models/LogisticRegression.py.’(‘fit’, ‘\_sigmoid’, and ‘eval’) The training model is based on the minibatch gradient, not the whole data. Given a mini-batch data the error function for a mini-batch is defined as follows:

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





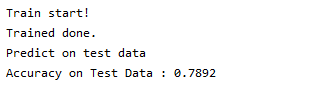
**(e) [Linear Regression in practice]** Given data, fill out the blank after training the model. Briefly write your opinion about the gradient descent solution in comparison with the analytic solution.  
(Hyperparameter for gradient: Epoch = 10000, Batch size = 32, learning rate = 0.01, optimizer=’SGD’)

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | **W1** | **W2** | **W3** |
| Initial value | 0.0 | 0.0 | 0.0 |
| Gradient Descent | 60.95784877 | 34.04094768 | 16.12081795 |
| Analytic solution | 63.493042 | 36.03978 | 17.446474 |
|  | **W4** | **W5** | **W6** |
| Initial value | 0.0 | 0.0 | 0.0 |
| Gradient Descent | -39.77357508 | 10.41451873 | 10.18109998 |
| Analytic solution | -34.943363 | 10.714367 | 16.288313 |
|  | **W7** | **W8** | **W9** | **RMSE** |
| Initial value | 0.0 | 0.0 | 0.0 | 39.6787 |
| Gradient Descent | 12.45202458 | 40.1281489 | 2.97741148 | 10.1168 |
| Analytic solution | 18.420761 | 40.215065 | -1.8608074 | 10.1148 |

RMSE =

**(f) [Logistic Regression in practice]** Given data and hyperparameter settings, report accuracy on test data after training the model. (Hyperparameter: Epoch = 1000, Batch size = 32, learning rate = 0.001, optimizer=’SGD’)

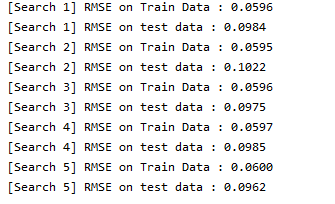


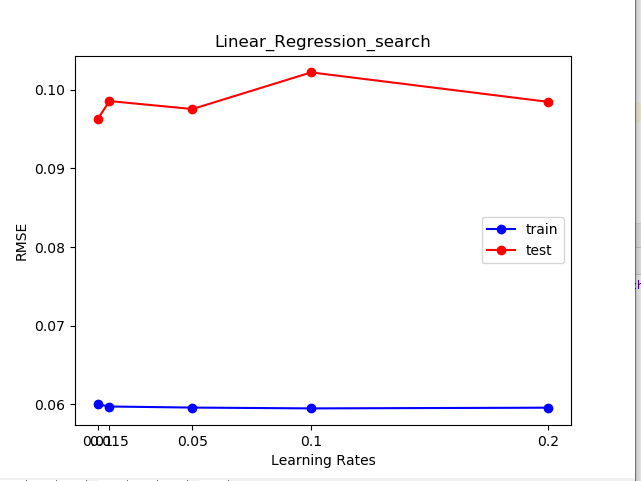
***NOTE: Write your codes in ‘EDIT HERE’ signs. It is not recommended to edit other parts.   
Once you complete your implementation, you can run ‘0\_1\_LinearRegression\_Checker.py’ and ‘0\_2\_LogisticRegression\_Checker.py’ to check whether your implementation is correct.***

***If you get the correct output, it means that you probably have implemented the code correctly.***

1. **[20 pts]** Parameter search on Linear Regression with Gradient Descent (‘2\_LinearRegression\_search.py’)
2. **[Plot Graph]** Given the chance of graduation regression data, draw a scatter plot on RMSE using different learning rates on fixed batch size and epoch (Try at least five different values) on the same plot. Use ‘matplotlib’ package to draw the plot. (X-axis: learning rates, Y-axis: RMSE)

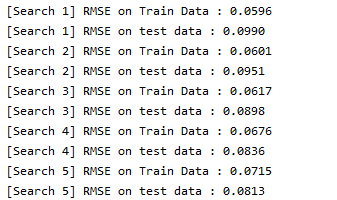
|  |  |
| --- | --- |
| **Parameter Settings** | |
| Batch size | 32 |
| Learning rate | 0.2,0.1,0.05,0.015,0.01 |
| # of Epochs | 10000 |
| Numpy seed | 2020 |
| Optimizer | SGD |

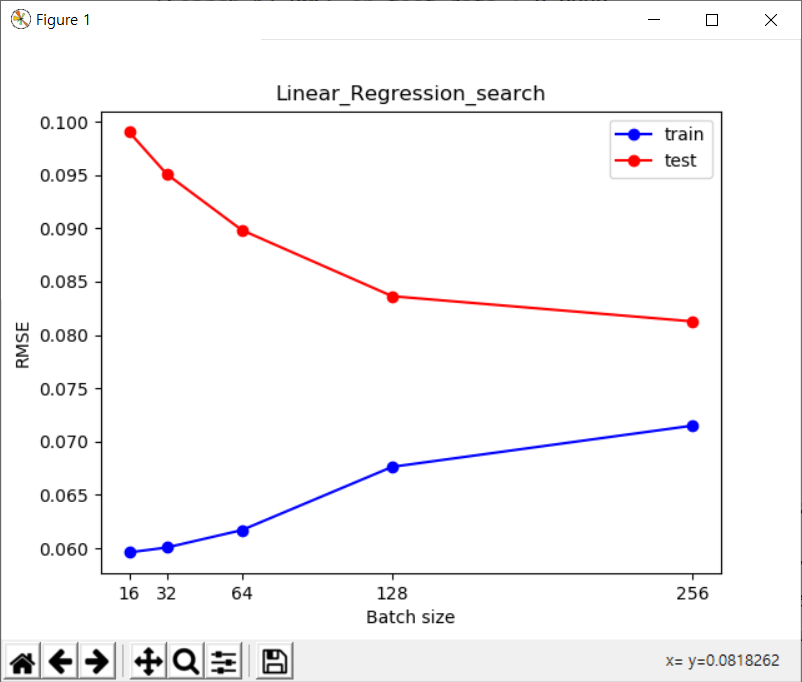




1. **[Plot Graph]** Given the chance of graduation regression data, plot graph on RMSE using different batch sizes on fixed learning rate and epoch (Try at least fix different values) on the same plot. Use ‘matplotlib’ package to draw the plot. (X-axis: batch sizes, Y-axis: RMSE)

|  |  |
| --- | --- |
| **Parameter Settings** | |
| Batch size | 16,32,64,128,256 |
| Learning rate | 0.01 |
| # of Epochs | 10000 |
| Numpy seed | 2020 |
| Optimizer | SGD |



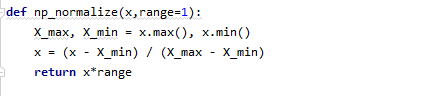


1. **[20 pts]** Parameter Search and Feature Engineering on Logistic Regression (‘3\_LogisticRegression\_search.py’)
2. **[Describe and Fill in the Blanks]** Given the titanic survival prediction data, tune hyperparameters and apply feature engineering technique (if you want) to get the best accuracy on test data. The description of each feature of data is written in ‘3\_LogisticRegression\_search.py’. Report your best accuracy and hyperparameters. (Fixed hyperparameter: batch size = 50, optimizer=’SGD’). Briefly explain the pre-processing and overall procedure for training.

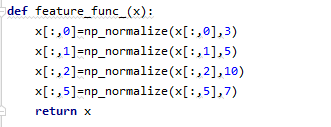
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Batch size** | **Epochs** | **Learning rate** | **Accuracy** |
| Initial value | 50 | 2000 | 0.02 | 0.8010 |

**<Feature engineering>**

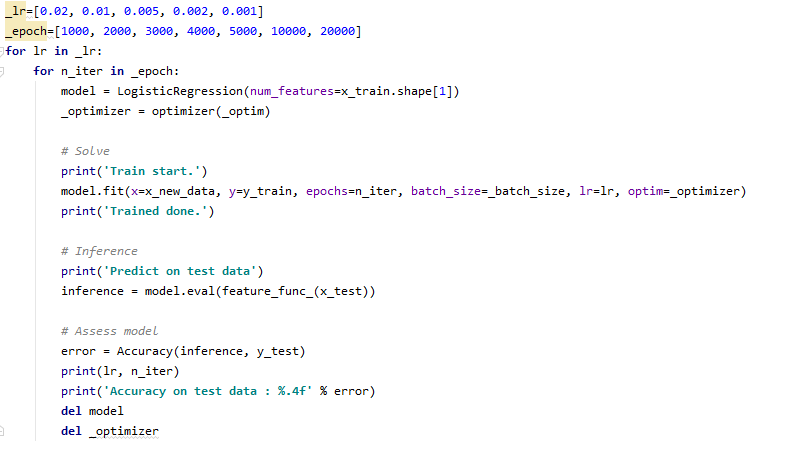
**I made the function which normalizes given feature to 0 ~ given value.**



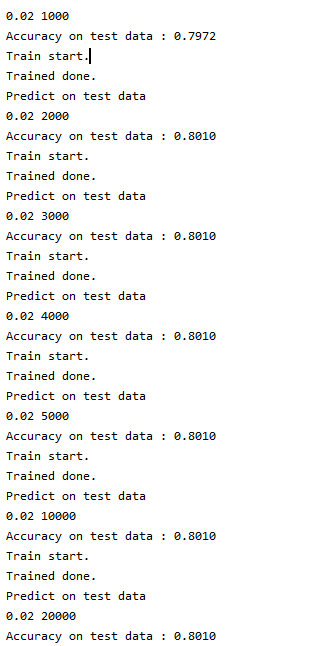
**And I did normalization to some features. And I set different range values to weight different importances. Especailly for the case of column 1(sex), I thought this column is important, so I set the range as 5. Initially the max value of column 1 is only 1, so I thought its influence on the result was very small. But after I set the max value as 5, its influence gets 5 times as large as before.**



And I did grid search to get the optimul epochs and learning rate using for loops as below.



The accuracy on test data never got over 0.8010. Many combinations of learning rates and epochs got 0.8010, but the case when the learning rate is 0.02 was very stable.



It recorded the accuracy score 0.8010 for many epochs. So I chose the learning rate as 0.02