INF552 Homework 4

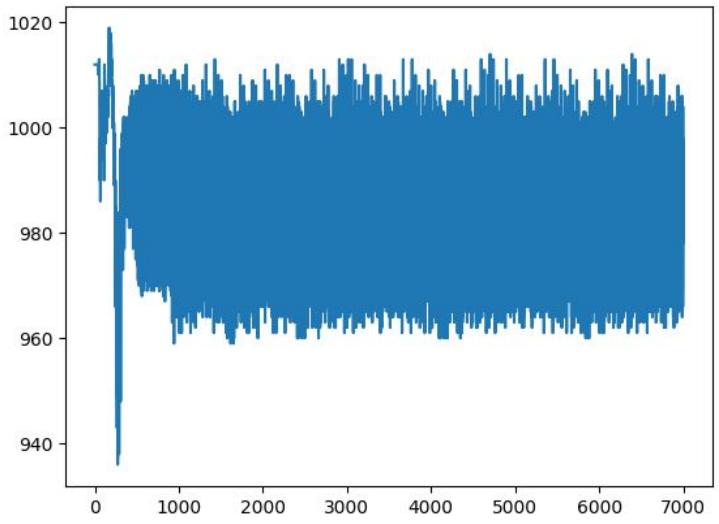
Group members and contribution :
Krishna Akhil Maddali (Code development and report)
Yash Shahapurkar (Code development, research, report)
Myungjin Lee (Code development and research)

Part 1: Implementation

- Output after running the program

  - **Perceptron (our code)**
    Weights :
    [-0.00118649607267559
    1.18630982391036266
    -0.95009322881483149
    -0.71007814116135193]
    Accuracy : 100 %

  - **Pocket algorithm (our code)**
    Weights :
    [0.00703656498231543
    -0.04943933777663556
     0.07635737741829697
     -0.02294096341852973]
    Accuracy : 50.1%



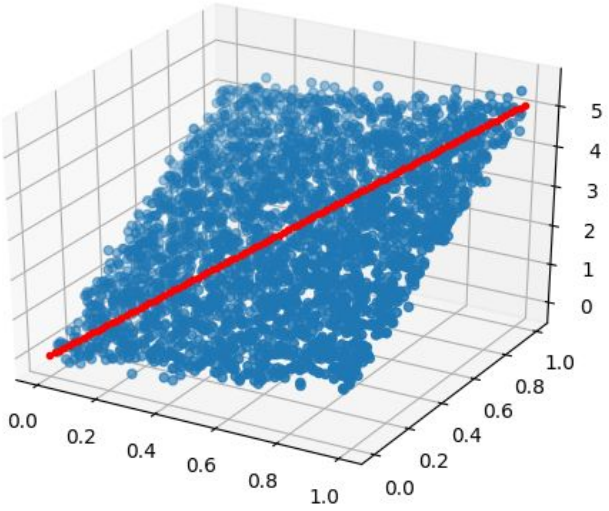
< pocket algorithm >

- **Linear Classification (Scikitlearn)**
  Weights :
  [[0.
  0.03553271799886835
  -0.02833146132860808
  -0.02131155465035351]]
  Accuracy : 99.9%

- **Logistic Regression (our code)**
  Weights :
  [[-0.01148439334562285]
  [0.01264640189206689]
  [-0.0166455196892677]
  [-0.01867201619523541]]
  Accuracy : 50.6%

- **Logistic Regression (Scikitlearn)**
  Weights :
  [[-0.17358115016451661
  0.1117687626267248
  0.07492529785961738]]
  Bias (W0) is [-0.0313139734009043]
  Accuracy : 52.95%

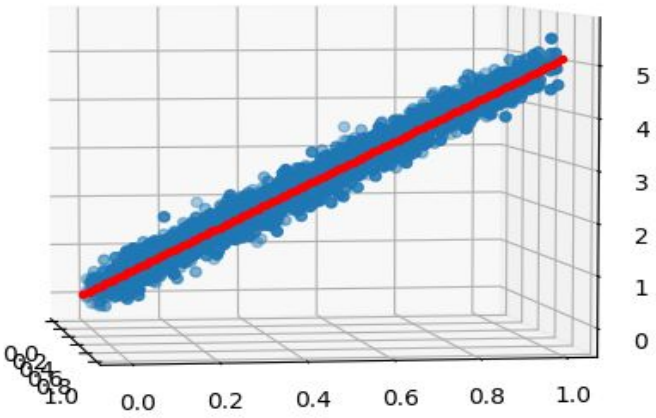
- **Linear Regression (our code)**
  Optimum weights :
  [0.01523534828888806
  1.08546356797937227
  3.99068854861238798]



< Linear Regression(our code)

>

- **Linear Regression (Scikitlearn)**
  Weights :
  [1.08546356797937693
  3.99068854861238842]
  Bias (W0) is 0.0152353482889



< Linear Regression (Scikitlearn) >

- Data structures

PERCEPTRON -

We used a pandas dataframe object to tabularize the data.
After sorting out required columns, we create arrays for the features and target.
We initialize weights in a list which is converted to an array. We update this same array for new weights.

POCKET ALGORITHM -

We used a pandas dataframe object to tabularize the data.
After sorting out required columns, we create arrays for the features and target.
Initialize lists for iteration count and number of misclassifications. Append the value to the list in every iteration
We initialize weights in a list which is converted to an array. We update this same array for new weights.

LOGISTIC REGRESSION -

We used a pandas dataframe object to tabularize the data.
After sorting out required columns, we create arrays for the features and target.
We initialize weights and update them in a dictionary for efficient storage and retrieval.

Store predicted outputs in an array to compare with the targets

LINEAR REGRESSION -

We used a pandas dataframe object to tabularize the data.
After sorting out required columns, we create arrays for the features and target.
Optimum weights obtained are stored in an array

- ## Code-level optimization

PERCEPTRON -

Used pandas dataframes for efficient storage of structured data
Using dot product instead of explicitly multiplying different values
Eliminating need of excessive conditional statements using statements like v[v>0]=1 instead of if loops

POCKET ALGORITHM -

Used pandas dataframes for efficient storage of structured data
Using dot product instead of explicitly multiplying different values
Using in-built numpy functionality like np.count instead of using excessive conditional statements

LOGISTIC REGRESSION -

Significant optimization due to vectorization of input space. Complete elimination of for loops. Only one for loop required for iterations.
Separate functions for each task allows modularity.
Dictionaries allow to store and retrieve objects efficiently
Used pandas dataframes for efficient storage of structured data
Using dot product instead of explicitly multiplying different values
Eliminating need of excessive conditional statements using statements like ycl[ycl>0.5]=1 instead of if loops
Using in-built numpy functionality like np.count instead of using excessive conditional statements

LINEAR REGRESSION -

Used pandas dataframes for efficient storage of structured data
Using in-built numpy linear algebra functions like np.linalg.inv

- Challenges

- Managing loop indentation (tabs) in perceptron and pocket algorithm
- Managing dimensionality requirement to perform matrix operations
- Deciding on random initialization of weights and choice for learning rate
- Conditional checking for misclassifications and weight updation for perceptron and pocket algorithm
- Getting bigger formulae like grad_w (in logistic regression) correctly implemented in code

Part 2: Software Familiarization

[**COMPARISON BETWEEN OUR CODE AND LIBRARY IS REPORTED IN PART 1.]**

- How to improve our code

- Use regularization techniques to improve accuracy
- Try to vectorize implementation of perceptron algorithm
- Keep track of error so that algorithm terminates within some error threshold criterion
- Better initial weight guess and learning rate, using some kind of cross-validation
- Use linear solver instead of finding inverse of a matrix

Part 3: Applications

LINEAR CLASSIFICATION -

Linear classification is employed as a data mining, while in others more detailed statistical modeling is undertaken such as computer vision, drug discovery, medical imaging.

LOGISTIC REGRESSION -

Logistic regression may be used to predict the risk of developing a given disease based on observed characteristics of the patient (age, sex, blood test). Another example might be to predict whether an American voter will vote Democratic or Republican, based on age, income, sex, race, state of residence, etc.

LINEAR REGRESSION -

In finances, The uses linear regression as well as the concept of beta for analyzing and quantifying the systematic risk of an investment. This comes directly from the beta coefficient of the linear regression model that relates the return on the investment to the return on all risky assets. In addition, linear regression can be applied in economics as well. For example, it is used to predict consumption pattern, investment spending, and labor supply.