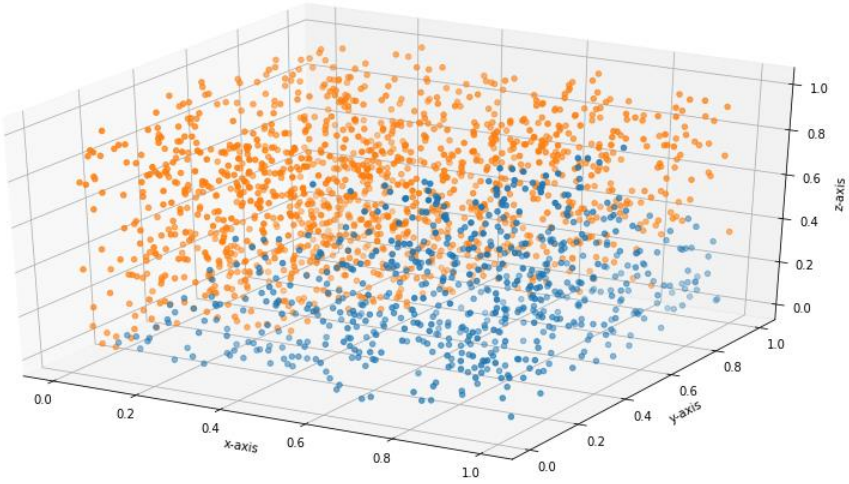
Assignment 4 Report

Perceptron Learning Algorithm

I first visualize the data as 3d point clouds using column 4 class label values to assign point colors and notice that this data is indeed linearly separable.

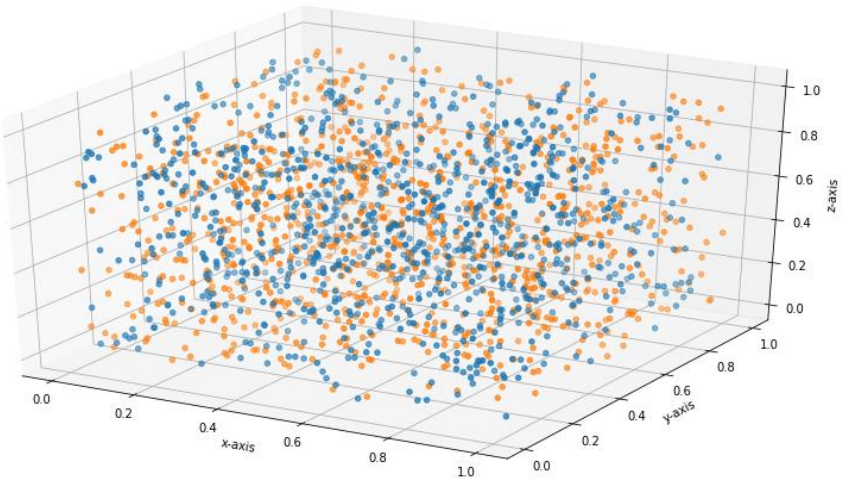


Then running 1000 iterations with a learning rate of 0.01 as my parameters I obtain the following results for accuracy and final weight values (w0,w1,w2,w3 respectively):

```
Perceptron Final Weights
 [-0.0111865   1.79844173 -1.43872591 -1.06058332]
================================
Perceptron Final Accuracy: 100.0%
```

Pocket Algorithm

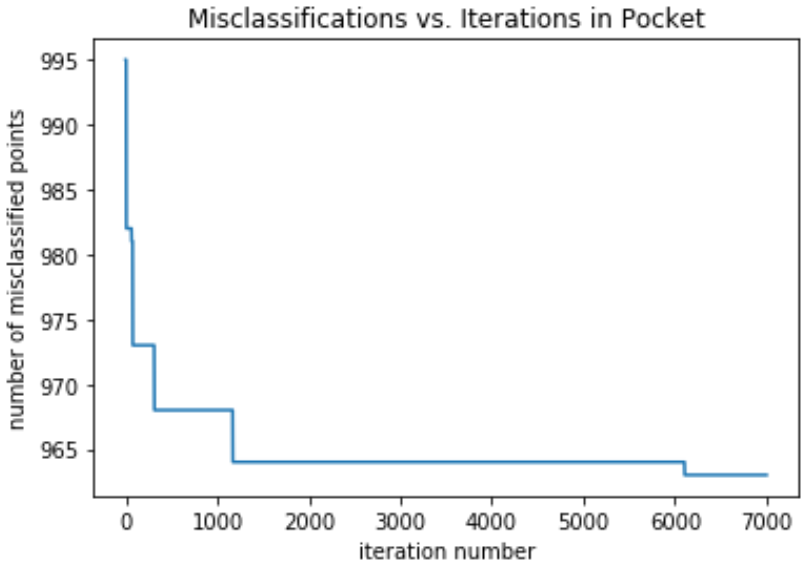
For the pocket algorithm, I again begin by an exploratory visualization the data as 3d point clouds but this time using column 5 as the label and notice that the data is very much not linearly separable.



Then 7000 iterations of the pocket algorithm are run with a learning rate of 0.01 and the following final weights (w0,w1,w2,w3 respectively) and final accuracy are obtained:

```
Pocket Algorithm Final Weights:
[ 0.0088135  -0.01753843  0.01015483 -0.00812551]
======================================
Pocket Algorithm Final Accuracy: 51.85%
```

In addition, the number of misclassified points in the pocket algorithm against the number of iterations is plotted below:



Logistic Regression

Using the same data as in the Pocket algorithm implementation (i.e. non-linearly separable with column 5 labels), I implement the requested logistic regression classifier. Since logistic regression is also a linear classifier, I expect (and indeed find it to be the case) that its performance is similarly poor on the non-linearly separable data. The logistic regression is run for 7000 iterations with a learning rate of 0.01, whereupon I obtain the following final weights (w0,w1,w2,w3 respectively) and accuracy:

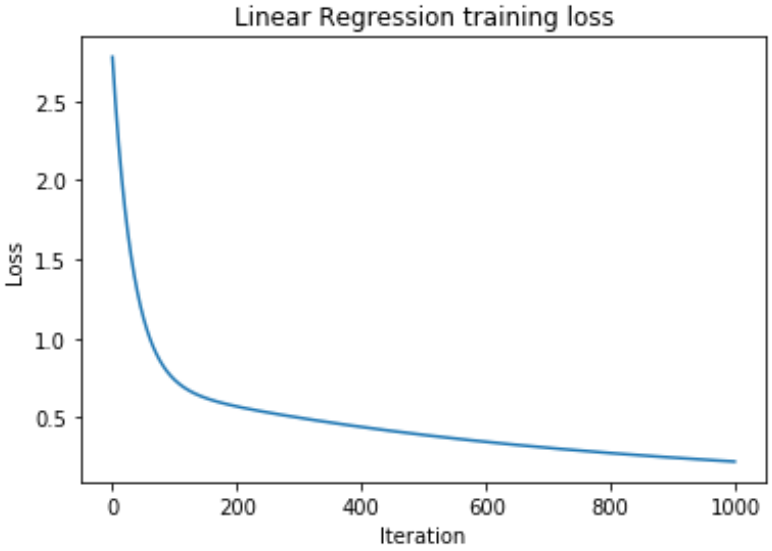
```
Logistic Regression Final Weights:
 [ 0.0005635  -0.01777426 -0.00054697 -0.00122984]
========================================
Logistic Regression Final Accuracy:  50.65%
```

Linear Regression

Lastly, I implement a linear regression class and fit the data in the linear-regression.txt file using a gradient descent optimization with learning rate 0.01 for 1000 iterations and obtain the following final weights (w0,w1,w2 respectively):

```
Linear Regression Final Weights:
 [0.88436499 0.91453353 2.51013005]
```

No accuracy metric was requested for this part since it is not a classification problem. Training loss was graphed over all 1000 iterations:



Additional Description:

The main data structures for the implementation of each algorithm in this assignment were python lists, numpy arrays, pandas dataframes. The requested outputs are described below as well as within the jupyter notebook. In addition the random state was set to seed = 0 for all runs.