DSCI 552 – Assignment 4: Perceptron Learning algorithm, Pocket algorithm, Logistic Regression and Linear Regression

Team Members	Sanjana Gopnal Swamy (gopnalsw@usc.edu)
	Amit Sankhla (asankhla@usc.edu)

Implementation:

For implementation of all the algorithms, we have used standard python libraries such as numpy and python list/dict. To plot the number of misclassified points against the number of iterations in the Pocket algorithm, we have used matplotlib.pyplot library.

Perceptron Learning Algorithm

The data structure we use to store the data is a 2D nested array using numpy.genfromtxt function which is easy for matrices calculation. Random assignment of weights is done using numpy.random.rand function. We calculate the dot product of the input and the weights and if the product is greater than 0 and the corresponding output is less than 0 or if the product is less than 0 and corresponding output is greater than 0, the weight gets updated. was max number of iterations is fixed for this input as 4000. This value was the most optimum one. Any value greater than 4000 gave no significant change in weights. Also, the learning coefficient alpha was fixed to 0.01

Output:

```
--------- Perceptron Learning Algorithm ---------
Max Iterations =  4000
Learning Rate =  0.01
Initial Weights =  [0.00990013 0.04875504 0.55831901 0.9046508 ]

----- Training Complete -----

----- Result after final iteration-----
No. of iteration =  1431
Weights =  [-9.98728740e-05  9.93959210e-01 -7.97845665e-01 -5.95135465e-01]
Accuracy =  1.0

Process finished with exit code 0
```

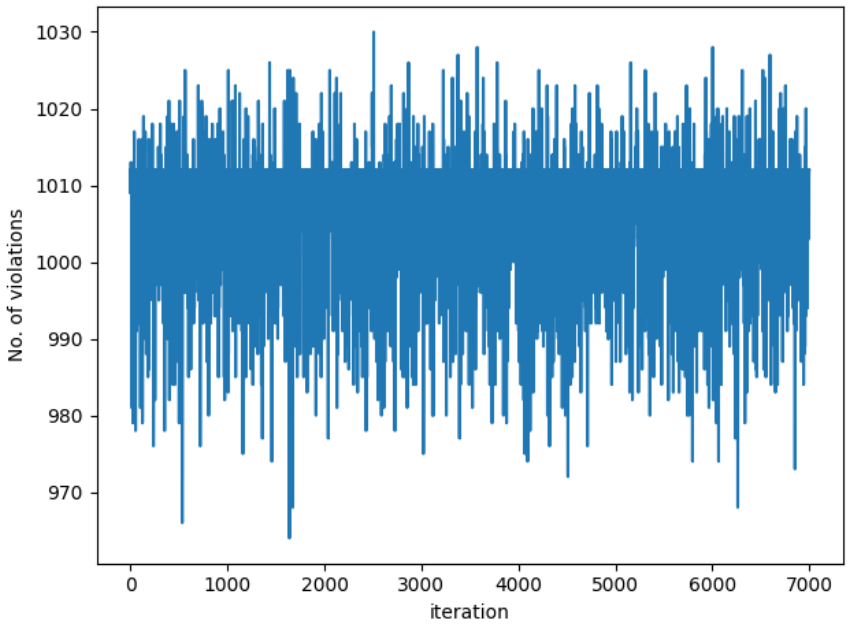
Pocket Algorithm

The implementation of this algorithm is very similar to Perceptron learning algorithm. We run perceptron algorithm while keeping extra set of weights to store pocket weights or the best weights which has the least number of mismatches. Whenever the perceptron has the least number of mismatches, these perceptron weights replace the pocket weights. The pocket weights are the outputs of the program. The learning coefficient alpha was fixed to 0.01 and the max number of iterations is fixed to 7000.

Output:

```
--------- Pocket Algorithm ---------
Max Iterations =  7000
Learning Rate =  0.01
Initial Weights =  [0.87342412 0.80745531 0.38792825 0.05987714]

----- Training Complete -----

----- Result after final iteration (Perceptron) -----
No. of iteration =  7000
Weights =  [ 0.01342412  0.00362709  0.01356282 -0.00570335]
Accuracy =  0.494

----- Best Result (Pocket) -----
Best iteration =  1639
Best weights =  [ 0.00342412 -0.00689288  0.00385474 -0.00348989]
Best Accuracy / Least mismatches = 0.518 / 964

Process finished with exit code 0
```

Plot of number of mismatches vs iterations



Logistic Regression

The data structure we use to store the data is again a 2D nested array using numpy.genfromtxt function. Then we call function train(). This function computes the gradient descent according to the sigmoid function and then updates weights according to the current value of weights, learning rate and the gradient. After training, we call function predict() to predict values for the given data. Within the predict function, we compute the dot product of weights with x (the features of the data) and then it computes the sigmoid on this computed value.

Output:

```
Logistic Regression
Max Iterations =  7000
Learning Rate =  0.01


1000 iterations completed
2000 iterations completed
3000 iterations completed
4000 iterations completed
5000 iterations completed
6000 iterations completed
7000 iterations completed
----- Training Complete -----

Result
No. of iteration =  7000
Weights =  [-0.15080072 -0.13360186  0.11474986  0.26202627]
Accuracy =  0.515

Process finished with exit code 0
```

Linear Regression

To implement Linear regression, we have used Least squares solution.

$$\text{Formula} = (X^T X)^{-1} X^T Y$$

We have used built in functions from the numpy library for computing inverse of matrices.

Output:

```
Linear Regression Result

Weights =  [0.01523535 1.08546357 3.99068855]

Process finished with exit code 0
```

Contributions:

1: Sanjana's contribution

- Implemented Perceptron Learning algorithm
- Implemented Pocket algorithm
- Wrote the Report

2: Amit's contribution

- Implemented Logistic Regression
- Implemented Linear Regression