Report on

Linear Perceptron

Team

1. Sai Satvik Vuppala (2017B4A71449H)
2. Shreya Kotagiri (2017B3AA0296H)
3. Dhanush Karupakula(2017B3A71011H)

Introduction

We implement a linear perceptron using two different datasets with a 70:30 train-test splits on both the datasets. To handle the case where the model is not converging for a dataset, we perform 10^6 iterations and then calculate the accuracy of the model.

Libraries Used

1. Numpy
2. Pandas

Implementation

This algorithm tries to solve the binary classification problem. The two datasets have 5 and 4 dimensions (D) respectively, where we are starting with an assumption that the points are linearly separable, for the algorithm to converge after a finite number of iterations.

We proceed through the whole process in the following steps:

1. Splitting of the dataset

The dataset has been split in a 70-30 ratio for training and testing purposes.

2. Defining the perceptron loss function

Since the algorithm works on the loss minimization of misclassified points, we start by initializing the best fit vector (w) that separates the data classes to all ones in the D-dimensional space, i.e, a D x 1 vector.

The principle is that with each subsequent iteration, the total loss function

$$\min \sum_{n \in M} -t_n * w^T \Phi(x_n)$$

of the misclassified points will reduce as the w updates itself.

3. Minimizing loss using Stochastic Gradient Descent

Such minimisation where the loss reduces every iteration can be achieved using the stochastic gradient descent algorithm where it is proved that for each iteration there is a decrease in the loss that occurred.

For the next iteration, w would be updated as,

$$w^{\tau+1} = w^{\tau} + \eta(\nabla(t_n * w^T \Phi(x_n)))$$

$$-t_n (w^{\tau+1})^T * \Phi(x_n) < -t_n (w^{\tau})^T * \Phi(x_n)$$

Implementing the gradient descent algorithm with $\eta = 1$, we calculate the loss k (1 x 1 matrix). w2, the boundary is arrived at when either the number of misclassified points is zero or when the loss function is minimized.

4. Testing the data for accuracy

The method for accuracy implemented in our algorithm is discussed below.

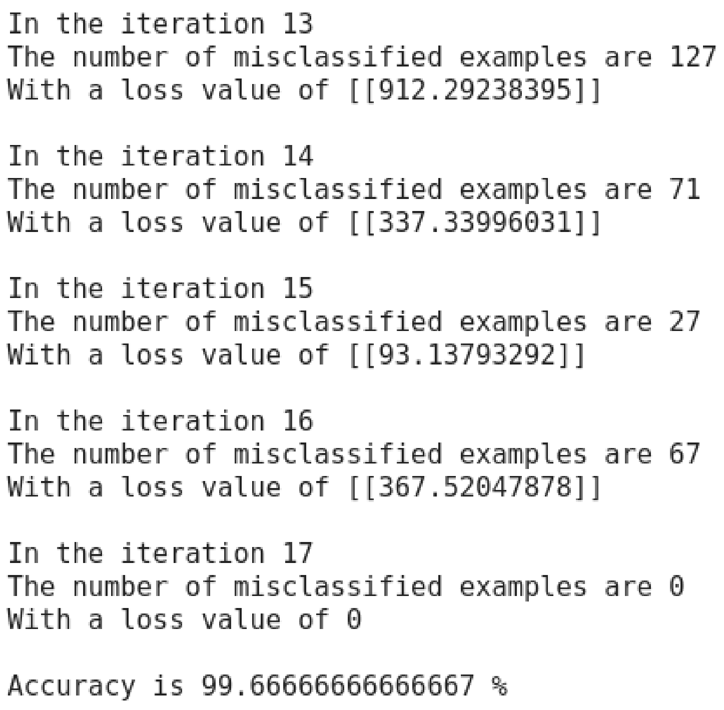
Accuracy

The w vector obtained by maturing the model with the training data is used by the testing data to check the accuracy of the model. The accuracy of the model is calculated as

$$1- \left({}^{\#\text{misclassified points}} / {}_{\#\text{size of test data}} \right)$$

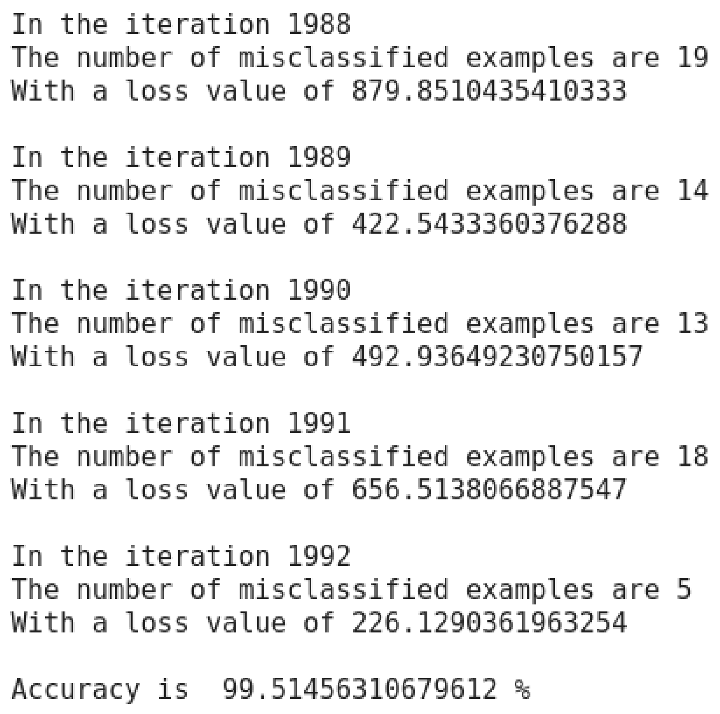
The accuracy was found to be **99.67%** for the second dataset and **99.51%** for the first one.

The dataset ‘dataset_LP_2.csv’ yielded the following accuracy:



```
In the iteration 13
The number of misclassified examples are 127
With a loss value of [[912.29238395]]

In the iteration 14
The number of misclassified examples are 71
With a loss value of [[337.33996031]]

In the iteration 15
The number of misclassified examples are 27
With a loss value of [[93.13793292]]

In the iteration 16
The number of misclassified examples are 67
With a loss value of [[367.52047878]]

In the iteration 17
The number of misclassified examples are 0
With a loss value of 0

Accuracy is 99.66666666666667 %
```

The dataset ‘dataset_LP_1.csv’ yielded the following accuracy:



```
In the iteration 1988
The number of misclassified examples are 19
With a loss value of 879.8510435410333

In the iteration 1989
The number of misclassified examples are 14
With a loss value of 422.5433360376288

In the iteration 1990
The number of misclassified examples are 13
With a loss value of 492.93649230750157

In the iteration 1991
The number of misclassified examples are 18
With a loss value of 656.5138066887547

In the iteration 1992
The number of misclassified examples are 5
With a loss value of 226.1290361963254

Accuracy is  99.51456310679612 %
```

Dataset which is more linearly separable

The second dataset, ‘dataset_LP_2.csv’ is more linearly separable as the loss function is converging to zero in a finite number of iterations. The first dataset ‘Dataset_LP_1.txt’ is not linearly converging (not able to find a suitable hyperplane to classify all points) and is found to be running for the total specified 10^6 iterations.

Limitations of Linear Perceptron

1. The output values of a perceptron can take on only one of two values (0 or 1), it cannot solve the n-classification problem.
2. Perceptrons can only classify linearly separable datasets, i.e, if a straight line or a plane can be drawn to separate the dataset into their correct categories, then the dataset is termed to be linearly separable.
3. We cannot guarantee on the time, i.e. by when the algorithm will converge. If a dataset of type ‘dataset_LP_1’ is provided it does not even converge when it reaches 10^6 iterations either,