BITS F464 : Machine Learning
Assignment - 1
Linear Perceptron Algorithm

Omkar Pitale
2019A7PS0083H

Aditya Chopra
2019A7PS0178H

Anuradha Pandey
2019A7PS0265H

March, 2021

1 Introduction

The Linear Perceptron is a supervised learning algorithm, used for Binary Classification. It tries to emulate the synapses of a modern mammalian brain. It is a linear classification algorithm and assumes that the data is Linearly Separable. The Perceptron Convergence Theorem states that given a Linearly Separable Dataset, a Perceptron Model will converge in a finite limited amount of steps. We pass the matured featured vector to the model, and expect it to return either 1, if it is a positive point and -1 if it is negative.

2 Model Description

2.1 Dataset and Pre-Processing

Two datasets were provided as described in Table 1. The datasets were sampled as a PyData Dataframe, and converted to two NumPy Arrays: Feature Vectors, and Targets. The Feature vectors were padded by 1s (this enables us to ignore a bias term, by incorporating it into the weights). The The dataset has targets as 0 and 1, but the algorithm requires them to be -1 and 1. Hence, the y array is modified, and each 0 set to -1. The X and y are then split into training and testing sets, with a 70:30 split ratio, as asked in the problem.

Name	dataset_LP_1.txt
Type	csv file
Number of Features	4
Positive Label	1
Negative Label	0
Positive Sample Size	610
Negative Sample Size	762
Total Sample Size	1372

Name	dataset_LP_2.csv
Type	csv file
Number of Features	3
Positive Label	1
Negative Label	0
Positive Sample Size	500
Negative Sample Size	500
Total Sample Size	1000

Table 1: Description of Datasets

2.2 Model Design

The Model is defined as a class, with two instance variables, w, the weights and iterations, the number of iterations that the model has trained for. The parameter w can be initialized with a Gaussian distribution, zeros or ones. iterations is initialized to 0. The model has two important functions, fit and test. The method fit, takes in a feature set and optimizes the model over it. It makes use of two other methods, classify and get_misclassified which classifies a points as positive or negative, and return all misclassified points respectively. Loops have been avoided as much as possible in favour of vectorization of mathematical operations. The only loop is over the number of epochs that the training runs over.

2.3 Working Equations

1. Variables

$$x_i = (1, x_{i1}, x_{i2}, \ldots, x_{iD}), \quad x_i \in \mathbb{R}^{D+1}$$

$$w^T = \begin{bmatrix} w_0 & w_1 & \ldots & w_D \end{bmatrix}$$

2. Classification Criteria

$$t_i = \begin{cases} 1 & w^T x_i > 0 \\ -1 & w^T x_i < 0 \end{cases}$$

3. Perceptron Criteria

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

where M is the set of all misclassified points.

4. Optimization Equation

$$w^{\tau+1} \leftarrow w^{\tau} + \eta t_n x_n$$

5. Measures of Performance

$$\text{Accuracy} = \text{ACC} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}$$

$$\text{Precision} = \text{PPV} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

$$\text{Recall} = \text{TPR} = \frac{\text{tp}}{\text{tp} + \text{fn}}$$

$$\text{F}_1 = 2 \cdot \frac{\text{PPV} \cdot \text{TPR}}{\text{PPV} + \text{TPR}} = 2 \cdot \frac{2 \cdot \text{tp}}{2 \cdot \text{tp} + \text{fp} + \text{fn}}$$

3 Results

1. Four Measures of Performance: Accuracy, Precision, Recall and F1-Score were considered
2. No conclusive trend or relation between performance of the model and η (the "learning rate")
3. Linear Perceptron Model did not converge for Dataset 1 (dataset_LP_1.txt) with any η, over 10^6 training iterations. Hence, Dataset 1 is not Linearly Separable for all intents and purposes.
4. Perceptron Model converged for Dataset 2 (dataset_LP_2.csv) in under 1500 iterations for all values of η tested. Hence, Data 2 is linearly separable.
5. Dataset 2 (dataset_LP_2.csv) is more Linearly Separable than Dataset 1(dataset_LP_1.txt).

η	Model Convergence		Performance on Test Set			
	Did Converge	Iterations	Accuracy	Precision	Recall	F1-Score
1	No	-	0.9782	0.9669	0.9831	0.9749
0.99	No	-	0.983	0.9724	0.9888	0.9805
0.9	No	-	0.983	0.9724	0.9888	0.9805
0.85	No	-	0.983	0.9724	0.9888	0.9805
0.75	No	-	0.9757	0.9667	0.9775	0.9721
0.5	No	-	0.9927	0.9944	0.9888	0.9915
0.45	No	-	0.9854	0.9725	0.9944	0.9833
0.25	No	-	0.9879	0.9943	0.9775	0.9858
0.125	No	-	0.9927	0.9944	0.9888	0.9915

Table 2: Results on Dataset 1

η	Model Convergence		Performance on Test Set			
	Did Converge	Iterations	Accuracy	Precision	Recall	F1-Score
1	Yes	50	1	1	1	1
0.99	Yes	28	1	1	1	1
0.9	Yes	13	1	1	1	1
0.85	Yes	44	1	1	1	1
0.75	Yes	16	1	1	1	1
0.5	Yes	40	1	1	1	1
0.45	Yes	51	1	1	1	1
0.25	Yes	34	1	1	1	1
0.125	Yes	36	1	1	1	1

Table 3: Results on Dataset 2

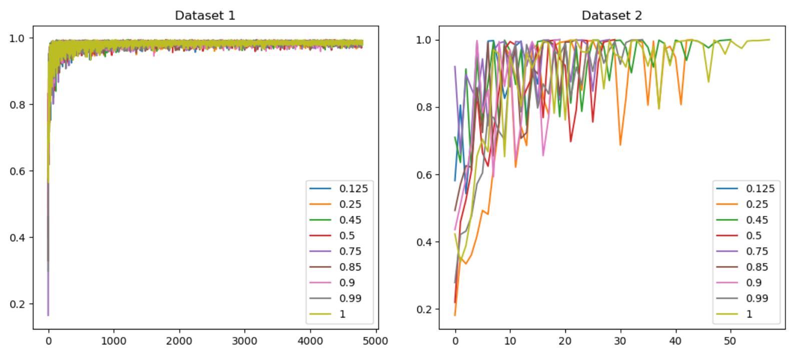

Figure 1: Accuracy vs Iterations Plot for the Datasets

4 Conclusion

The Linear Perceptron Model makes use of a Heaviside activation function, and are hence not Universal Function Approximators. The Linear Perceptron Model, although it performs very well for linearly separable data (i.e. There exists a $D - 1$ dimensional Hyperplane that separates the Positive and Negative Samples), it does not work perfectly with linearly non-separable datasets. If no $D - 1$ dimensional Hyperplane exists that separates the classes, a single Linear Perceptron can not perfectly classify the data. This is the major limitation with the Perceptron Model.