Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Implementing the Perceptron Algorithm for Finding the Weights of Linear Discriminant Function

Course Title: Pattern Recognition Lab

Course No: CSE 4214

Assignment No: 02

Date of Submission: 16.8.2020

Submitted By:

Name: Md. Reasad Zaman Chowdhury

Section: A1

Student ID: 160104004

Implementing the Perceptron Algorithm for Finding the Weights of Linear Discriminant Function

Md. Reasad Zaman Chowdhury

16.01.04.004

Abstract:

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. The goal of the perceptron algorithm is to minimize the criterion function by updating the weight vector. There are two ways weights can be updated: batch perceptron and Single sample perceptron. Both of these algorithm yields an optimal weight vector for a linearly separable problem.

Keyword:

Perceptron,

Introduction:

Perceptron algorithm finds the optimal weights of a supervised learning problem by a continuous process of updating the weights. The updating process can be done by two well known perceptron algorithm:

1. Batch Perceptron or Many at a Time
2. Single-sample Perceptron or One at a Time

The batch Perceptron algorithm for finding a solution vector can be stated very simply: the next weight vector is obtained by adding some multiple of the sum batch of the misclassified samples to the present weight vector. We use the term "batch" training to refer to the fact that (in general) a large group of samples is used when computing each weight update.

Algorithm 3 (Batch Perceptron)

1 begin initialize $\mathbf{a}, \eta(\cdot)$, criterion $\theta, k = 0$
2 do $k \leftarrow k + 1$
3 $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y}$
4 until $\eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y} < \theta$
5 return a
6 end

One the other hand, rather than testing **a**(k) on all of the samples and basing our correction of the set of misclassified training samples, single-sample perceptron algorithm shall consider the samples in a sequence and shall modify the weight vector whenever it misclassifies a *single* sample.

Algorithm 4 (Fixed-increment single-sample Perceptron)

```
1 begin initialize  a, k = 0
2         do  k ← (k + 1) mod n
3            if  y_k is misclassified by a  then  a ← a − y_k
4         until all patterns properly classified
5    return a
6 end
```

Experimental Design:

a. **Plot All Sample Points:**
At first, we visualized our training data by using scatter plot. Since our training data has 2 classes, we plotted the data of different classes with different colors. Here blue points are class 1 samples and red points are class 2 samples.

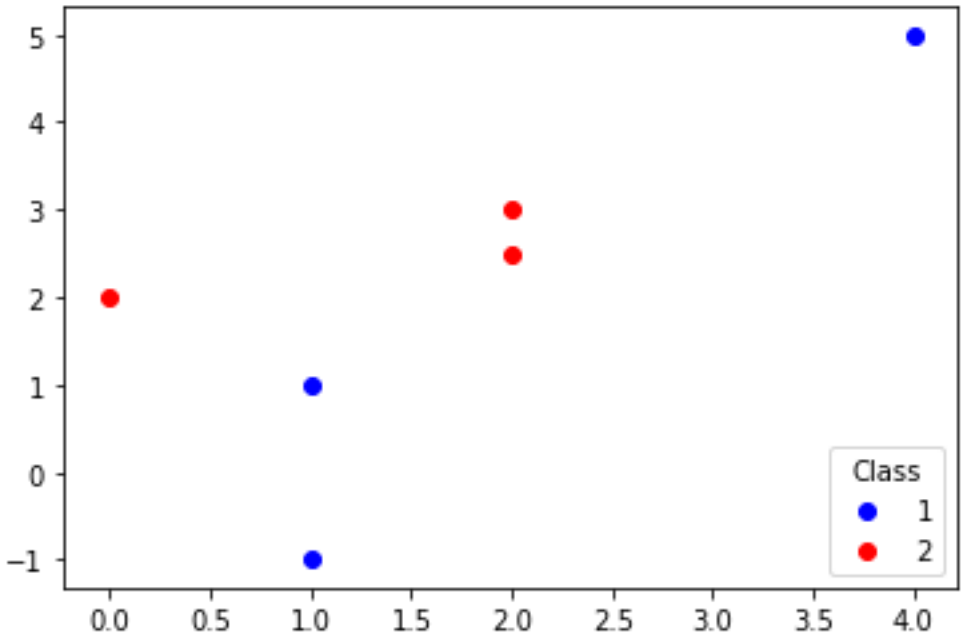

Fig 1: Scatter plot of training data points

b. **Generate High Dimensional Sample Points and Normalize Any of the Class:**
High dimensional data points are generated from the sample points by applying phi function to our sample points which transform data from lower dimension to higher dimension using the following formula:
y = [x1*x1, x2*x2, x1*x2, x1, x2, 1]

Normalization are done by negating resulting vector of class 2. In this way we are left with the single criterion function. If the criterion function is not satisfied it is considered as misclassified.

c. Use Perceptron Algorithm (both one at a time and many at a time) for finding the weight coefficients:

Perceptron algorithm for both one at a time and many at a time are implemented to find the weight coefficients. Two functions one_at_a_time(y, w, alpha) and many_at_a_time(y, w, alpha) are created which takes feature vector y, weight vector w and learning rate alpha and returns the number of iteration to converge to an optimal solution.

d. Create Table and Visualize the Performance of Different Perceptron algorithm for Different Combination of Weight Initialization, Learning Rate:

Three initial weights have been used (all one, all zero, randomly initialized with seed fixed). For all of these three cases learning rate is varied between 0.1 and 1 with step size Finally, number of iterations for both one at a time and many at a time is obtained and visualized through tabular representation and bar plot.

Tabular Representation:

1. Initial Weight Vector: All One

Alpha (Learning Rate)	One at a Time	Many at a Time
0.1	6	102
0.2	147	104
0.3	149	91
0.4	149	116
0.5	141	105
0.6	157	114
0.7	136	91
0.8	136	91
0.9	140	105
1.0	141	93

2. Initial Weight Vector: All Zero

Alpha (Learning Rate)	One at a Time	Many at a Time
0.1	141	105
0.2	141	105
0.3	141	92
0.4	141	105
0.5	141	92
0.6	141	105
0.7	141	105
0.8	141	105

0.9	141	105
1.0	141	92

3. Initial Weight Vector: Random (Seed=2)

Alpha (Learning Rate)	One at a Time	Many at a Time
0.1	144	95
0.2	154	87
0.3	141	92
0.4	153	90
0.5	157	88
0.6	157	139
0.7	135	138
0.8	136	116
0.9	136	118
1.0	136	97

Bar Chart:

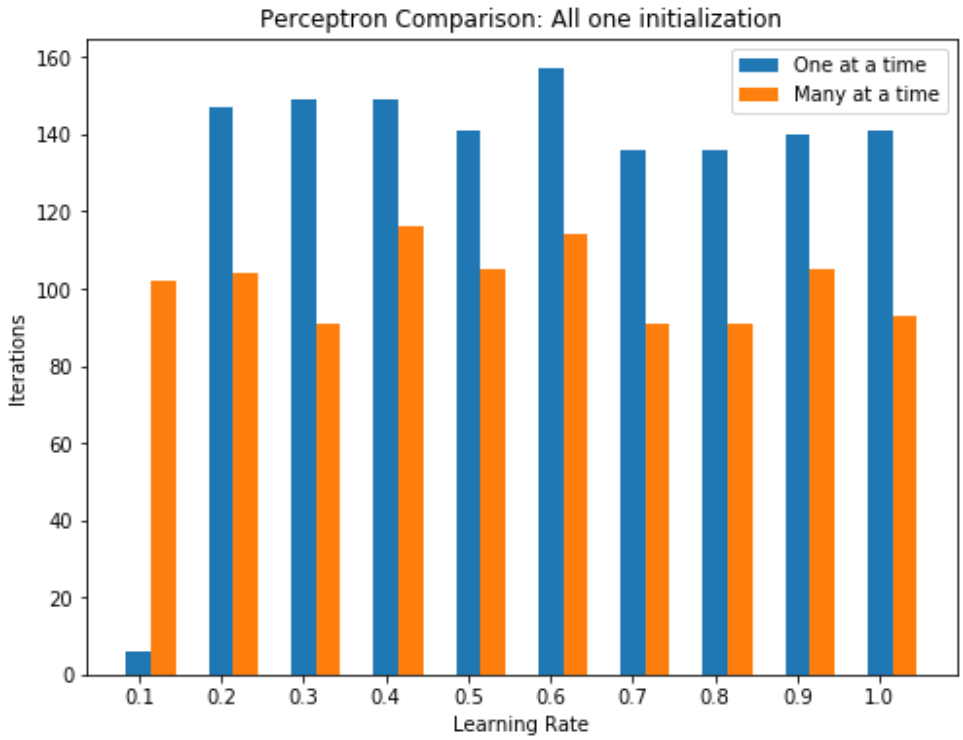

Fig: Perceptron Comparison of All One Initialization

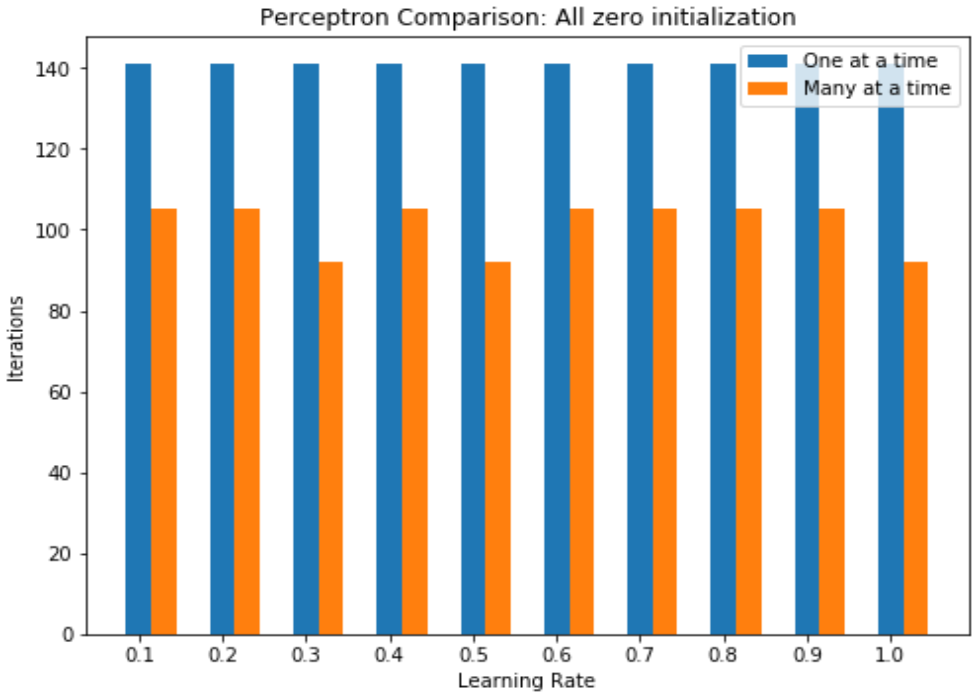

Fig: Perceptron Comparison of All Zero Initialization

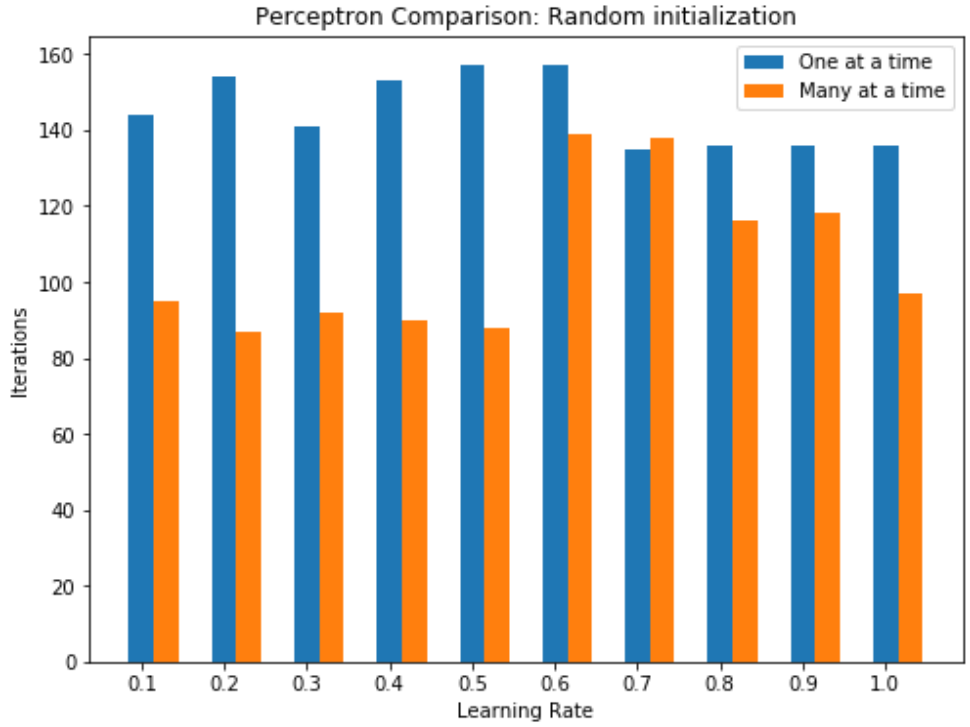

Fig: Perceptron Comparison of Random Initialization