**Description of Algorithm:**

**Sample data generation - [X, Y] = generateData(N):**

**Step 1** – Created random sample of X (N x 2) matrix of point’s uniformly sampled from [−1, 1] × [−1, 1]

**Step 2** – Created two random points from [−1, 1] × [−1, 1] to draw line.

**Step 3** – Applied Logic and find out points from one side of line and another side of line. Marked points on one side of line as +1 otherwise -1. Stored into Y (N x 1)

**Perceptron Learning Algorithm - pla(X,Y,w0):**

**Step 1:** Initialize the weight vector[w] as either 0 or from pseudo inverse function.

**Step 2:** Add an artificial coordinate X [0] = 1 into X.

**Step 3:** Calculated *Yprediction* through Sign(WtX) function

**Step 4:** Compared *Yprediction* with actual Y

**Step 5:** If both are not same then take one randomly generated misclassified point and updated weight through w = w + YnXn (misclassified point).

**Step 6:** Repeat step 3, 4, 5 till it matches.

**Pseudo Inverse:**

**Step 1:** Add an artificial coordinate X [0] = 1 before each point.

**Step 2:** Computed the (X dagger) X+. Where X+= (X+\*X)-1 XT

**Step 3:** Calculated final weights through W = X+\*Y.

**Assumption:**

We have generated sample data set which is already linearly separable.

**Experiment Results**

|  |  |  |
| --- | --- | --- |
| Average # of Iteration after 100 execution for each N | | |
| **N** | **Iteration without Regression** | **Iteration with Pseudoinverse** |
| 10 | 7.39 | 3.61 |
| 50 | 91.15 | 90.07 |
| 100 | 88.68 | 75.17 |
| 200 | 228.88 | 252.31 |
| 500 | 816.63 | 810.67 |
| 1000 | 884.57 | 947.71 |

**Observation:**

1. # of iterations increased with sample size
2. Execution time increased with sample size
3. Providing learned weight through regression is reducing iteration in some cases but not all time.