**JUNAID GIRKAR**

**60004190057**

**TE COMPS A4**

**AI**

**Experiment - 7**

**Aim: Perceptron training algorithm for L and M classification**

**Theory**:

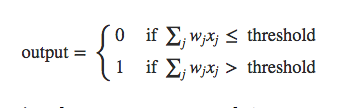
Perceptrons are a type of artificial neuron that predates the sigmoid neuron. It appears that they were invented in 1957 by Frank Rosenblatt at the Cornell Aeronautical Laboratory.

A perceptron can have any number of inputs, and produces a binary output, which is called its activation.

First, we assign each input a weight, loosely meaning the amount of influence the input has over the output.

To determine the perceptron’s activation, we take the weighted sum of each of the inputs and then determine if it is above or below a certain threshold, or \*bias, \*represented by b.

The formula for perceptron neurons can can be expressed like this:



**Algorithm**:

| def perceptron(inputs, bias)   weighted\_sum = sum {  for each input in inputs  input.value \* input.weight  }   if weighted\_sum <= bias  return 0  if weighted\_sum > bias  return 1  end |
| --- |

**Code**:

| def sgn(net\_input):  if net\_input <= 0 :  return -1  return 1  def pattern\_classifier(n\_iterations, input, weight, desired\_output, learning\_rate):  for iteration in range(n\_iterations):  print(f'Iteration {iteration+1}')  output = []  for i,X in enumerate(input):  net\_input = 0  for j in range(len(X)):  net\_input+=weight[j]\*X[j]  generated\_output = sgn(net\_input)  output.append(generated\_output)  if generated\_output != desired\_output[i]:  difference = desired\_output[i] - generated\_output  for position in range(len(weight)):  weight[position] = float("{:.2f}".format(weight[position] + learning\_rate\*difference\*X[position]))  print(f'Generated Output vector for Iteration {iteration+1} : {output}')  print(f'Weight vector after Iteration {iteration+1} : {weight}')  print("------"\*25)  if output == desired\_output:  break  return output,weight  def main():  input = [  [1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1], #L starts here  [1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,1,1,1],   [1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1],   [0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,1,1,1],   [1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,1,1,1,1],   [0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,1,1,1,1],   [0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1],   [1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,1,0,1,1],   [0,1,0,0,0,0,1,0,0,0,0,1,1,0,0,1,1,0,1,1],   [1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,0,1,1],     [0,1,0,1,0,1,1,0,1,1,1,0,1,0,1,1,0,0,0,1], #M starts here  [1,0,0,0,1,1,1,0,1,1,1,0,1,0,1,1,0,0,0,1],   [1,0,0,0,1,1,1,0,1,1,1,0,1,0,1,1,0,1,0,1],   [1,1,0,1,1,1,0,1,0,1,1,0,1,0,1,1,0,0,0,1],   [1,1,0,1,1,1,0,1,0,1,1,0,0,0,1,1,0,0,0,1],   [1,0,0,0,1,1,1,0,1,1,1,0,0,0,1,1,0,0,0,1],   [1,0,0,0,1,1,1,0,1,1,1,1,0,1,1,1,0,1,0,1],   [1,1,0,1,1,1,0,1,0,1,1,0,1,0,1,1,0,1,0,1],   [1,0,0,0,1,1,1,0,1,1,1,0,1,0,1,0,0,0,0,0],   [1,0,0,0,1,1,1,1,1,1,1,0,1,0,1,1,0,0,0,1],   ]  desired\_output = [1,1,1,1,1,1,1,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]  initial\_weight = [1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1]  learning\_rate = 0.05  n\_iterations = 3   classification\_output, weight\_vector = pattern\_classifier(n\_iterations, input, initial\_weight, desired\_output, learning\_rate)   count = 0  for i, output in enumerate(classification\_output):  if output == desired\_output[i]:  count+=1   accuracy = (count / len(input))\*100   print(f'Accuracy of Classifier : {accuracy} %')   print('Classifying an Unknown Sample of L (Output = 1)')  unknown\_sample = [1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,0]  print('Unknown Sample : ',unknown\_sample)  net\_input=0  for i in range(len(unknown\_sample)):  net\_input+=weight\_vector[i]\*unknown\_sample[i]  predicted\_output = sgn(net\_input)  print('Predicted Output : ', predicted\_output)  print("\n")  main() |
| --- |

**Output**:

| Iteration 1 Generated Output vector for Iteration 1 : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1] Weight vector after Iteration 1 : [0.2, 0.6, 0.0, 0.6, 0.2, -0.9, 0.4, 0.6, -0.6, 0.1, 0.1, -0.1, 0.4, 0.9, -0.9, 0.1, 1.0, -0.3, 1.0, 0.1] ------------------------------------------------------------------------------------------------------------------------------------------------ Iteration 2 Generated Output vector for Iteration 2 : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1] Weight vector after Iteration 2 : [0.1, 0.5, 0.0, 0.5, 0.1, -1.0, 0.4, 0.5, -0.6, 0.0, 0.0, -0.1, 0.3, 0.9, -1.0, 0.0, 1.0, -0.3, 1.0, 0.0] ------------------------------------------------------------------------------------------------------------------------------------------------ Iteration 3 Generated Output vector for Iteration 3 : [1, 1, 1, 1, -1, 1, 1, 1, 1, 1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1] Weight vector after Iteration 3 : [0.1, 0.4, 0.0, 0.4, 0.0, -1.0, 0.4, 0.4, -0.6, -0.1, 0.0, -0.1, 0.2, 0.9, -1.0, 0.0, 1.1, -0.2, 1.1, 0.0] ------------------------------------------------------------------------------------------------------------------------------------------------ Accuracy of Classifier : 90.0 % Classifying an Unknown Sample of L (Output = 1) Unknown Sample : [1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0] Predicted Output : 1 |
| --- |