**EXPERIMENT 7**

**AIM:** To design a pattern classifier using perceptron training algorithm for the given pattern of alphabets L and M.

\*   .   .   .   .                                                                              \*   .   .   .   \*     ​

\*   .   .   .   .                                  and                                     \*   \*  .   \*  \*     ​

\*   .   .   .   .                                                                              \*   .   \*  .   \*     ​

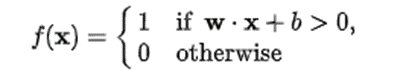
\*   \*  \*  \*  \*                                                                             \*   .   .   .   \*     ​

Pattern (a): L                                                                        Pattern (b): M ​

**THEORY:**

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, i.e. a classification algorithm that makes it’s predictions based on a linear predictor function combining a set of weights with the feature vector.

Perceptron has a threshold function: a function that maps its input X(a real-valued vector) to an output value f(X)(a single binary value):



where **w** is a vector of real-valued weights, **w.X** is the dot product, **∑ wi.Xi** where m is the number of inputs to the perceptron, and **b** is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value.

The value of f(X) => (0 or 1) is used to classify X as either a positive or a negative instance, in the case of a binary classification problem. If b is negative, then the weighted combination of inputs must produce a positive value greater than |b| in order to push the classifier neuron over the 0 **threshold**. Spatially, the bias alters the position (though not the orientation) of the decision boundary. The perceptron learning algorithm does not terminate if the learning set is not linearly separable. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly.

Diagram

Description automatically generated

A perceptron is an artificial neuron using the **Heaviside step function** as the activation function.

**CODE:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn import datasets

class Perceptron:

def \_\_init\_\_(self, learning\_rate=0.01, n\_iters=1000):

self.lr = learning\_rate

self.n\_iters = n\_iters

self.activation\_func = self.\_unit\_step\_func

self.weights = None

self.bias = None

def fit(self, X, y):

n\_samples, n\_features = X.shape

# initialise parameters

self.weights = np.zeros(n\_features)

self.bias = 0

y\_ = np.array([1 if i > 0 else 0 for i in y])

for \_ in range(self.n\_iters):

for idx, x\_i in enumerate(X):

linear\_output = np.dot(x\_i, self.weights) + self.bias

y\_predicted = self.activation\_func(linear\_output)

# Perceptron update rule

update = self.lr \* (y\_[idx] - y\_predicted)

self.weights += update \* x\_i

self.bias += update

def predict(self, X):

linear\_output = np.dot(X, self.weights) + self.bias

y\_predicted = self.activation\_func(linear\_output)

return y\_predicted

def \_unit\_step\_func(self, x):

return np.where(x >= 0, 1, 0)

if \_\_name\_\_ == "\_\_main\_\_":

def accuracy(y\_true, y\_pred):

accuracy = np.sum(y\_true == y\_pred) / len(y\_true)

return accuracy

def displayIP(arr):

for i in range(len(arr)):

if i%5==0 and i !=0:

print("")

print(arr[i], end=" ")

# TRAINING DATA

X = np.array([(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),

(1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),

(1, 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, 0, 0, 0, 1, 1, 1, 1, 1),

(1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),

(1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),

(1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),

(1, 0, 0, 0, 0, 1, 0, 1, 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, 1, 1, 1),

(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1),

(1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),

(1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),

(1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),

(1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),

(1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),

(1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1),

(1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1),

(1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1),

(1, 0, 0, 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, 1, 1)])

y = np.array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1, random\_state=123 )

p = Perceptron(learning\_rate=0.01, n\_iters=50)

p.fit(X\_train, y\_train)

predictions = p.predict(X\_test)

# ACCURACY

print("Perceptron Classification Accuracy- TEST SET : ", accuracy(y\_test, predictions))

# TEST 1

test\_1 = np.array([1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1])

pred\_1 = p.predict(test\_1)

if pred\_1 == 1:

output = "L"

else:

output = "M"

print("\nTEST INPUT:")

displayIP(test\_1)

print("\nPREDICTED OUTPUT : ", output)

# TEST 2

test\_2 = np.array([1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1])

pred\_2 = p.predict(test\_2)

if pred\_2 == 1:

output = "L"

else:

output = "M"

print("\nTEST INPUT 2:")

displayIP(test\_2)

print("\nPREDICTED OUTPUT : ", output)

**OUTPUT:**

Graphical user interface, text, application

Description automatically generated

**CONCLUSION:** In this experiment, I designed a pattern classifier using perceptron training algorithm for the alphabets L and M. I made a class Perceptron with basic parameters like learning\_rate(default=0.01), n\_iters(default=1000), activation function as unit step function and the initial weights and bias to none. The accuracy of the test input for the perceptron classifier was 1.0, overfitting as the dataset is quite small. The current perceptron model creates a decision boundary for binary classification i.e. output is labelled ‘L’ if the model output is 1 else ‘M’. Finally, when I fed the model with distorted L and M inputs it successfully classified between the alphabets.