NAME: Meith Navlakha
SAP: 60004190068
AI Practical

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.

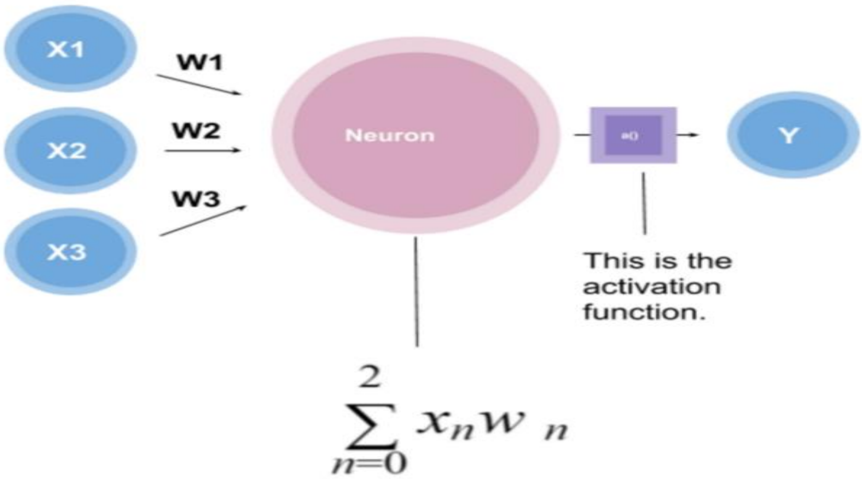
Perceptron has a threshold function: a function that maps its input X(a real-valued vector) to an output value f(X)(a single binary value):

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}\cdot\mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

where **w** is a vector of real-valued weights, **w.X** is the dot product, $\sum \mathbf{w_i.X_i}$ 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.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical



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)
```

NAME: Meith Navlakha
SAP: 60004190068

OUTPUT:

```
Perceptron Classification Accuracy- TEST SET :  1.0

TEST INPUT 1:
1 0 1 0 0
1 0 0 0 0
1 0 0 0 0
1 1 1 1 1
PREDICTED OUTPUT :  L

TEST INPUT 2:
1 0 0 0 1
1 1 0 1 1
1 0 1 0 1
1 0 1 0 1
PREDICTED OUTPUT :  M
```

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.