Aim: With a suitable example demonstrate the perceptron learning law with its decision region using python. Give the output in Graphical form.

Software for Python: Jupyter Notebook

Theory:

Perceptron Learning rule

It was introduced by Rosenblatt. It is an error-correcting rule of a single-layer feedforward network. it is supervised in nature and calculates the error between the desired and actual output and if the output is present then only adjustments of weight are done.

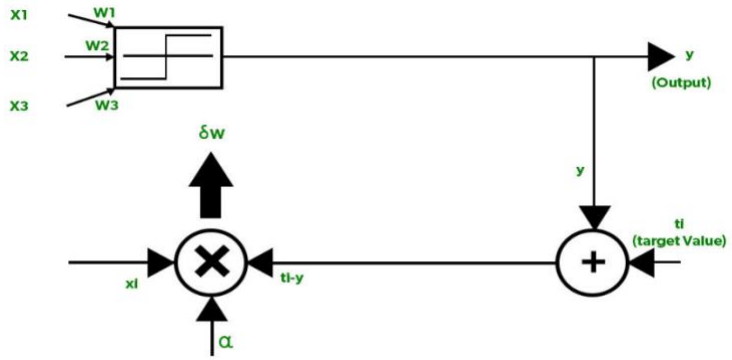


Fig.1.Perceptron Learning Rule

Computed as follows:

Assume (x1, x2, x3……. xn) –>set of input vectors and (w1, w2, w3……. wn) –>set of weights (Fig.1).

y=actual output

wo=initial weight

wnew=new weight

δw=change in weight

α=learning rate

actual output (y)=wi*xi

learning signal (ej)=ti-y difference between desired and actual output)

δw=α*xi*ej

wnew=wo+δw

Now, the output can be calculated on the basis of the input and the activation function applied over the net input and can be expressed as:

y=1, if net input>=θ

y=0, if net input<θ

Procedure:

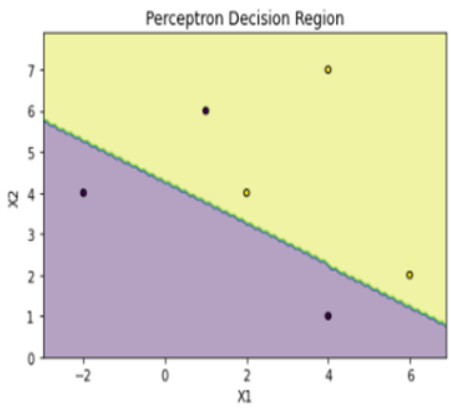
1.In this code, we first define our training data, which consists of six data points with two features each.
2.We label these points as either -1 or 1, depending on which class they belong to.
3.Next, we initialize the weights and bias of the perceptron to zero.
4.We then define the perceptron function, which takes the training data, weights, and bias as inputs, and trains the perceptron using the perceptron learning algorithm.
5.This algorithm updates the weights and bias for each data point, based on whether the perceptron's prediction is correct or not.
6.Finally, we call the perceptron function to train the perceptron on our training data.
7.We then plot the training data, along with the decision boundary learned by the perceptron.
8.The decision boundary is represented as a contour plot, which separates the two classes of data points.

Conclusion: Thus, we have studied to demonstrate the perceptron learning law with its decision region as the output in Graphical form using python.

Program code and Output

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

# Define the training data
X = np.array([    [-2, 4],
    [4, 1],
    [1, 6],
    [2, 4],
    [6, 2],
    [4, 7]
])
y = np.array([-1, -1, -1, 1, 1, 1])

# Initialize the weights and bias
w = np.zeros(X.shape[1])
b = 0

# Define the perceptron function
def perceptron(X, y, w, b, learning_rate=0.1, num_epochs=100):
    for epoch in range(num_epochs):
        for i in range(X.shape[0]):
            # Compute the output of the perceptron
            z = np.dot(X[i], w) + b
            if z > 0:
                y_pred = 1
            else:
                y_pred = -1
            # Update the weights and bias if the prediction is wrong
            if y_pred != y[i]:
                w = w + learning_rate * y[i] * X[i]
                b = b + learning_rate * y[i]
    return w, b

# Train the perceptron
w, b = perceptron(X, y, w, b)

# Plot the training data and the decision boundary
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max, 0.1))
Z = np.sign(np.dot(np.c_[xx.ravel(), yy.ravel()], w) + b)
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolor='k')
plt.title('Perceptron Decision Region')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```


Perceptron Decision Region
X1
X2
−2
0
2
4
6
0
1
2
3
4
5
6
7