PERCEPTRON AND PROBLEM (FROM SCRATCH)

**AIM:-**

**To demonstrate the training of a perceptron to simulate the behavior of an AND gate**

**PROBLEM DESCRIPTION:-**

**The AND operation is a fundamental logic gate that takes two binary inputs and produces an output based on the following rule: the output is true (1) only when both inputs are true (1) otherwise the output is false(0).**

| **INPUT A** | **INPUT B** | **OUTPUT** |
| --- | --- | --- |
| **0** | **0** | **0** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

**The problem is to train the perceptron model to learn the behavior of AND gate based on the provided INPUT -OUTPUT pair. Then validate its performance by checking whether it correctly predicts the OUTPUT.**

**ALGORITHM:-**

1. **Define Activation Function:**
   * **activation(out,threshold):**
     + **Takes out (output) and threshold as inputs.**
     + **If out is greater than or equal to threshold, returns 1; otherwise, returns 0.**

**2) Define Perceptron Function (perceptron):**

* + **Initialize input (a and b), actual output (y), weights (w), threshold, learning rate, and index i.**
  + **Print "PERCEPTRON TRAINING".**
* **Perceptron Training Loop:**
  + **While i < 4 (number of training samples):**
    - **Calculate summation as a[i]\*w[0] + b[i]\*w[1].**
    - **Compute perceptron output (o) using the activation function with summation and threshold.**
    - **Print input values, current weights, summation, and actual vs. predicted output.**
    - **If predicted output (o) doesn't match the actual output (y[i]):**
      * **Print "Updating weights".**
      * **Update weights (w[0] and w[1]) using the perceptron learning rule.**
      * **Print updated weights.**
      * **Reset i to -1 (to retrain with the updated weights).**
      * **Print "Weights Updated, Training Again".**
    - **Increment i by 1.**
* **Prediction Part:**
  + **Calculate summation using the provided and\_input and trained weights.**
  + **Return the output of the activation function using summation and threshold.**
* **Execute the Code:**
  + **Set and\_input as [0, 1].**
  + **Print "AND Gate Output for [0, 1]:" and the result of the perceptron function with and\_input.**

**Overall, this algorithm trains a perceptron using the provided AND gate input-output pairs (a, b, y) to mimic the AND gate behavior. It updates weights based on the error between predicted and actual outputs and tests the trained perceptron's accuracy by predicting the output for a new input ([0, 1]).**

**PROGRAM CODE / PSEUDOCODE:-**

**def activation(out,threshold):**

**if out>=threshold:**

**return 1**

**else:**

**return 0**

**def perceptron(and\_input):**

**a=[0,0,1,1]**

**b=[0,1,0,1]**

**y=[0,0,0,1] #actual output**

**w=[0.1,0.8]**

**threshold=1**

**learning\_rate=0.5**

**i=0**

**print("PERCEPTRON TRAINING :")**

**while i<4:**

**summation = a[i]\*w[0] +b[i]\*w[1]**

**o=activation(summation,threshold)**

**print("Input :"+str(a[i])+","+str(b[i]))**

**print("Weights : "+str(w[0])+","+str(w[1]))**

**print("Summation : "+str(summation)+"threshold : "+str(threshold))**

**print("Actual output :"+str(y[i])+"Predicted Output :"+str(o))**

**if(o!=y[i]):**

**print("\_\_\_\_\nUpdating weights")**

**w[0] =w[0]+learning\_rate\*(y[i]-o)\*a[i]**

**w[1] =w[1]+learning\_rate\*(y[i]-o)\*b[i]**

**print("Updated Weights : "+str(w[0])+","+str(w[1]))**

**i=-1**

**print("\nWeights Updated Training Again : ")**

**i=i+1**

**#prediction part**

**summation= and\_input[0] \* w[0] + and\_input[1]\*w[1]**

**return activation(summation,threshold)**

**and\_input =[0,1]**

**print("AND Gate Output for "+str(and\_input) +" : " +str(perceptron(and\_input)))**

**RESULT:-**

**The trained perceptron model correctly predicts the AND gate OUTPUT of various combinations of INPUT.**