|  |  |
| --- | --- |
| **Task 10** | Create a perceptron with appropriate number of inputs and outputs. Train it using fixed incrementlearning algorithm until no change in weights is required. Output the final weights. |

**Tools: Google co-lab, Python, Scikitlearn, Anaconda navigator**

**Algorithm:**

1. Start
2. Initialize weights = 0 and bias = 0
3. Set learning rate (lr = 0.1)
4. Repeat for each epoch:  
   a. Set error\_count = 0  
   b. For each input sample (X[i], y[i]):  
   i. Compute output = activation( dot(X[i], weights) + bias )  
   ii. Compute error = y[i] – output  
   iii. If error ≠ 0:  
   - Update weights = weights + lr \* error \* X[i]  
   - Update bias = bias + lr \* error  
   - Increment error\_count  
   c. If error\_count = 0 → stop training
5. Display final weights and bias
6. Stop

**PROGRAM:**

# Import required libraries

import numpy as np

# Define input features (X) and target output (y)

# Example: OR Gate

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

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

# Initialize weights and learning rate

weights = np.zeros(X.shape[1])

bias = 0

lr = 0.1

# Fixed Increment Learning Algorithm

def activation(x):

return 1 if x >= 0 else 0

epochs = 10

or epoch in range(epochs):

error\_count = 0

for i in range(len(X)):

z = np.dot(X[i], weights) + bias

output = activation(z)

error = y[i] - output

if error != 0:

weights += lr \* error \* X[i]

bias += lr \* error

error\_count += 1

if error\_count == 0:

break

print(" Final Weights:", weights)

print("Final Bias:", bias)

**OUTPUT:**

Final Weights: [0.1 0.1]

Final Bias: -0.1

**RESULT:**

Thus the above program was executed successfully and output was verified