|  |  |
| --- | --- |
| Subject: Machine Learning Lab | Course ID: CSL-604 |
| Semester: VI | Course: AI & DS |
| Laboratory: 406-B | Name of teacher: Prof. Seema Pawar |
| Name of Student: Ayush Gupta | Roll No: VU2S2324001 |

**EXPERIMENT NO. 6**

**Aim:**

To implement the Hebbian Learning Algorithm.

**Theory:**

Hebbian Learning Rule, proposed by Donald O. Hebb, is one of the earliest and simplest learning rules in neural networks. It is mainly used for pattern classification and is based on the principle that "neurons that fire together, wire together." This rule updates the weights between neurons for each training sample in a single-layer neural network, which consists of an input layer and an output layer.

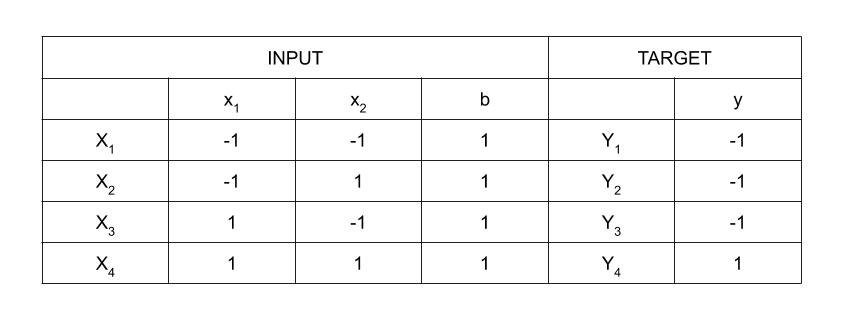
**Hebbian Learning Rule Algorithm:**

1. Initialize all weights to zero and set the bias to zero.
2. For each input vector and its corresponding target output, perform the following steps:
   * Set activations for input neurons based on the given input vector.
   * Assign the target output value to the output neuron.
   * Update the weights using the Hebbian rule:

wnew = wold + x . y

1. Repeat this process for all training samples.
2. The final weight matrix is obtained after processing all input vectors.

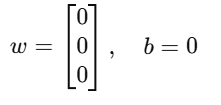
**Implementation of AND Gate using Hebbian Learning Rule:**



The AND gate follows a truth table with four training samples. The activation function used here is a bipolar sigmoidal function with values in the range [−1, 1]. The weight updating process follows the Hebbian rule iteratively for each sample.

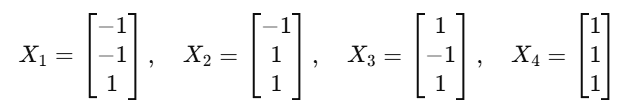
**Step 1: Initialize Weights and Bias**

* Set initial weights and bias to zero:

****

**Step 2: Define Input Vectors**

* Each input vector Xi​ is paired with its corresponding target output.



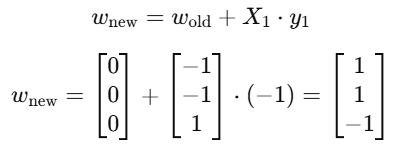
**Step 3: Set Output Values**

* For each training input Xi​, assign the corresponding target output y=t.

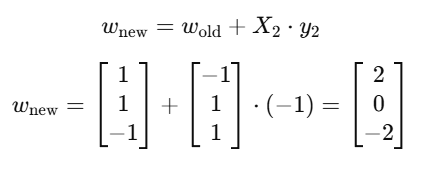
**Step 4: Update Weights using Hebbian Rule**

Weights are updated iteratively based on the Hebbian Learning Rule:

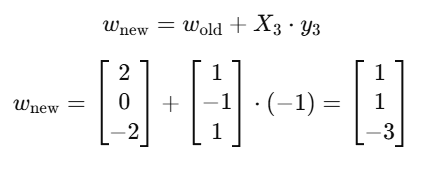
1. **First Iteration:**



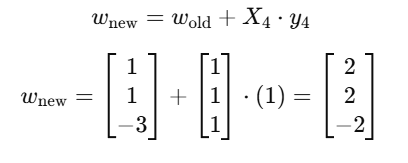
1. **Second Iteration:**



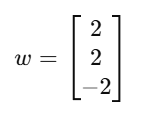
1. **Third Iteration:**

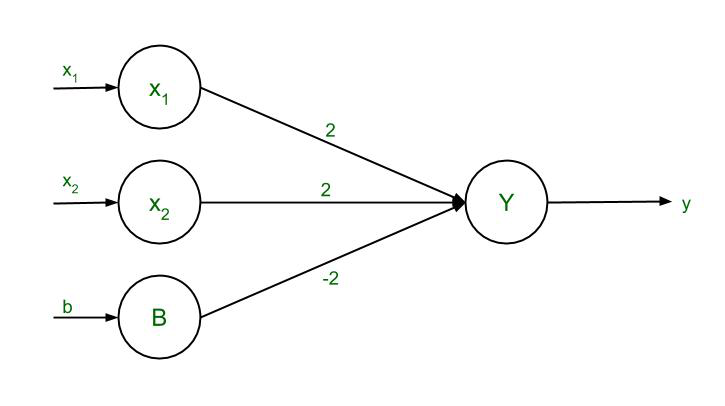


1. **Fourth Iteration:**



After training, the final weight matrix is obtained and used to test the network with new inputs. The decision boundary for the AND function is determined mathematically and graphically represented as:

****



**Learning Outcomes:**

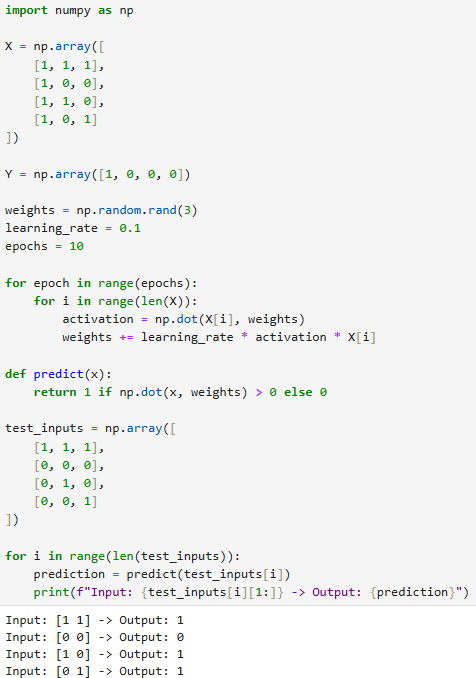
* Understanding the working principle of SVM.
* Implementing SVM for classification tasks.
* Learning how hyperplanes and kernel functions impact classification performance.
* Exploring different SVM kernel types (linear, polynomial, RBF) and their applications.

**Conclusion:**

Support Vector Machines are powerful tools for classification and regression problems. They provide optimal solutions for various data representations, including linear and non-linear separations using kernel functions. SVM is widely used due to its effectiveness in handling high-dimensional data and ensuring a robust decision boundary. Its ability to work well with small to medium-sized datasets while maintaining high accuracy makes it a preferred choice in machine learning applications.



**Program and Output:**

****