|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Hope Foundation’s**  **Finolex Academy of Management and Technology, Ratnagiri** | | | | | |
| **Department of Computer Science and Engineering (AIML)** | | | | | |
| Subject name: Machine Learning | | | | | | Subject Code: CSL604 | |
| Class | | TE CSE | | | Semester –VI (CBCGS) | Academic year: 2024-25 | |
| Name of Student | | GIRIPRASATH K | | | | **QUIZ Score :5** | |
| Roll No | | 29 | | Experiment No. | | 04 | |
| Title: **To perform Hebbian Learning** | | | | | | | |
|  | | | | | | | |
| **1. Lab objectives applicable:**  **LOB1:**To introduce platforms such as Anaconda, COLAB suitable to Machine Learning.  **LOB3:** To develop Neural Network based learning models. | | | | | | | |
| **2. Lab outcomes applicable:**  **LO1:**Implement various Machine learning models.  **LO3:** Implement Neural Network based models. | | | | | | | |
| **3. Learning Objectives:**   1. To perform Hebbian Learning. | | | | | | | |
| **4. Practical applications of the assignment/experiment:**  To update weights based on input and output. | | | | | | | |
| **5. Prerequisites**:   1. Python language | | | | | | | |
| **6. Minimum Hardware Requirements**:-  I series processor, RAM 4GB,  **7. Software Requirements:-**  Colab or Visual Studio or Jupyter notebook (Anaconda) | | | | | | | |
| **8. Quiz Questions :** [**https://docs.google.com/forms/d/e/1FAIpQLSeAw8lZ9N7j9QNJ3XliZL9Ry4pIzVMy7yU4mBszJ78atuCPbw/viewform?usp=dialog**](https://docs.google.com/forms/d/e/1FAIpQLSeAw8lZ9N7j9QNJ3XliZL9Ry4pIzVMy7yU4mBszJ78atuCPbw/viewform?usp=dialog) | | | | | | | |
| **9. Experiment/Assignment Evaluation:** | | | | | | | |
| **Sr. No.** | **Parameters** | | | | | **Marks obtained** | **Out of** |
| **1** | Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used - | | | | | 5 | 6 |
| **2** | Lab Performance | | | | |  | 2 |
| **3** | Punctuality | | | | |  | 2 |
| **Date of performance (DOP)** | | |  | | **Total marks obtained** |  | **10** |

**Signature of Faculty**

**10. Theory:**

|  |
| --- |
| We have learning rate=0.1  Input arrays=[[1, 0], [0, 1], [1, 1], [0, 0]]  Output arrays=[[1, 0], [0, 1], [1, 1], [0, 0]]  **1st iteration:**  Weight vector=  Updated Weights=learning rate \* input\*output + weight vector  =  =  =  **2nd Iteration:**  W =    =    =  **3rd Iteration:**  W =    =    =  **4th Iteration:**  W =    =  **Testing:**  Input = [ 1 0]  W =    = |
|  |

**11. Installation Steps / Performance Steps and Results –**

**Source Code:**

import numpy as np

class HebbianNetwork:

    def \_\_init\_\_(self, input\_size, output\_size, learning\_rate=0.01):

        self.weights = np.random.rand(input\_size, output\_size)

        self.learning\_rate = learning\_rate

    def train(self, inputs, outputs):

        for i in range(len(inputs)):

            input\_vector = inputs[i]

            output\_vector = outputs[i]

            self.weights += self.learning\_rate \* np.outer(input\_vector, output\_vector)

    def predict(self, input\_vector):

        return np.dot(input\_vector, self.weights)

# Example usage

# Define input and output patterns

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

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

# Initialize the Hebbian network

network = HebbianNetwork(input\_size=2, output\_size=2)

# Train the network

network.train(inputs, outputs)

# Test the network

test\_input = np.array([1, 0])

predicted\_output = network.predict(test\_input)

print(f"Predicted output for input {test\_input}: {predicted\_output}")

plt.plot(iterations, weight\_values[:, 0], label="Weight 1")

plt.plot(iterations, weight\_values[:, 1], label="Weight 2")

plt.plot(iterations, weight\_values[:, 2], label="Weight 3")

plt.xlabel('Iteration')

plt.ylabel('Weight Value')

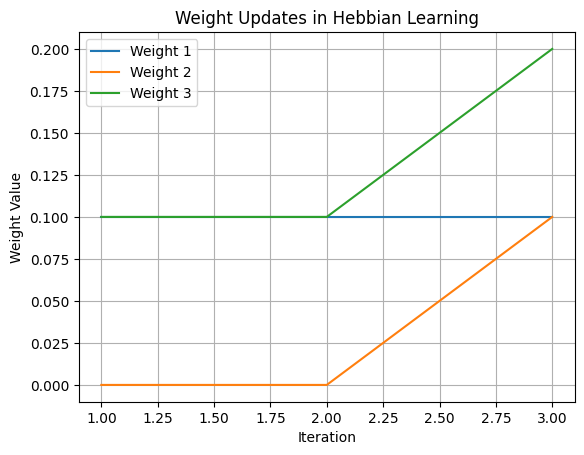
plt.title('Weight Updates in Hebbian Learning')

plt.legend()

plt.grid(True)

plt.show()

**Output:**



**12. Learning Outcomes Achieved**

1. Students are able to perform Hebbian Learning..

**13. Conclusion:**

**1. Applications of the Studied Technique in Industry**

Used in systems like Hopfield networks to store and retrieve patterns, such as handwriting or facial recognition. Also, used in modeling associations between user preferences and products in an unsupervised manner.

**2. Engineering Relevance**

Understanding Hebbian principles allows engineers to design and optimize unsupervised neural networks. It also enhances pattern recognition, such as identifying noise or recurring signals in engineering systems.

**3. Skills Developed**

The Hebbian learning experiment enhances several skills such as analytical skills, programming skill, critical thinking, mathematical understanding, which are relevant to engineering and computational problem-solving.

**14. References**:

1. Nathalie Japkowicz & Mohak Shah, ―Evaluating Learning Algorithms: A Classification Perspective‖, Cambridge.
2. Marc Peter Deisenroth, Aldo Faisal, Cheng Soon Ong, ―Mathematics for machine learning‖
3. Samir Roy and Chakraborty, ―Introduction to soft computing‖, Pearson Edition.
4. Ethem Alpaydın, ―Introduction to Machine Learning‖, MIT Press McGraw-Hill Higher

Education