Practical No. 1 Aim: Write a program to implement logical gates AND, OR and NOT with McCullochPitts.

The McCulloch-Pitts (M-P) neuron model is one of the earliest formalizations of how individual neurons might work. Proposed by Warren McCulloch and Walter Pitts in 1943, it laid the groundwork for the development of artificial neural networks. The M-P neuron is a simplified mathematical model inspired by the behavior of biological neurons in the brain. Unlike modern neural network models, the M-P neuron model does not include a learning mechanism. The weights and thresholds are typically set manually based on the desired behavior of the neuron. In practical applications, more sophisticated learning algorithms, such as the perceptron learning rule or backpropagation, are used to adjust the parameters of artificial neural networks based on training data.

Practical No. 2 Aim: Write a program to implement Hebb‟s rule. Theory: Hebb’s Rule, also known as Hebb’s Law or the Theory of Hebbian Learning, is singlehandedly Hebb’s most well known contribution to neuropsychology. First introduced in Hebb’s book, “The Organization of Behaviour”, this theory has transformed our understanding of cognitive processes and continues to shape our knowledge of human behaviour today. Hebb’s Rule describes how when a cell persistently activates another nearby cell, the connection between the two cells becomes stronger. Specifically, when Neuron A axon repeatedly activates neuron B’s axon, a growth process occurs that increases how effective neuron A is in activating neuron B. As a result, the connection between those two neurons is strengthened over time. 4 This idea is frequently summarized as “neurons that fire together, wire together”, and can be seen in many aspects of our daily lives. For example, when you repeat something over and over again, such as when learning how to play an instrument, the connections between the activated brain cells are strengthened. Much like the common saying that “practice makes perfect”, these neural connections grow to become faster, stronger, and more efficient over time. This new understanding provided the basis for numerous further scientific discoveries, such as the concept of synaptic plasticity, which describes how synapses (the space between cells where messages are passed along) can change in strength over time, as a result of increases or decreases in activity.

Hebbian Learning Rule, also known as Hebb Learning Rule, was proposed by Donald O Hebb. It is one of the first and also easiest learning rules in the neural network. It is used for pattern classification. It is a single layer neural network, i.e. it has one input layer and one output layer. The input layer can have many units, say n. The output layer only has one unit. Hebbian rule works by updating the weights between neurons in the neural network for each training sample.

Practical No. 4 Aim: Solve the Hamming network given the exemplar vectors. Theory: Hamming Network: The Hamming network, also known as the binary associative memory, is a type of neural network used for pattern recognition and retrieval. It is based on the principles of the Hopfield network but uses binary activation states and a different update rule. Exemplar Vectors: Exemplar vectors, also called prototype vectors or reference patterns, are the key input to the Hamming network. These vectors represent the patterns or memories that the network should learn and later recall. Each exemplar vector consists of binary values (0 or 1) corresponding to the presence or absence of features in the pattern. Solving the Hamming Network:

Initialization:

o Initialize the Hamming network with the exemplar vectors.

o Each exemplar vector becomes a neuron in the network, and its binary values are

the states of the neuron.

2. Training:

o There is no explicit training phase in the Hamming network. Instead, the

exemplar vectors are directly used to set the initial states of the network.

3. Activation:

o The Hamming network operates in parallel. At each iteration, the state of each

neuron is updated based on the majority rule.

o For each neuron, if the majority of its input connections are active (1), it remains

active (1). Otherwise, it becomes inactive (0).

o This process continues until the network stabilizes, meaning that the states of the

neurons no longer change.

4. Recall:

o Once the network stabilizes, the final state of each neuron represents the recalled

pattern.

o To recall a pattern, we can input a noisy or partial version of an exemplar vector

into the network and let it converge to the closest stored pattern.

o The network's ability to retrieve the correct pattern depends on its capacity and

the similarity between the input and stored patterns.

5. Output:

o The output of the Hamming network is the set of stable states of its neurons after

convergence.

o These stable states represent the recalled patterns or memories stored in the

network.

Practical No. 6

Aim: Implement a program to find the winning neuron using MaxNet.

The MaxNet is a type of artificial neural network that is used for competitive learning. In the

MaxNet, the neuron with the highest input value (i.e., the winner neuron) becomes more

dominant over time.

It is also known as Winner-Takes-All network is a type of artificial network where only the

neuron with the maximum input become active, suppressing the activity of all other neurons.

MaxNet is a neural network model used for competitive learning, where neurons compete to

become active based on their input values. It's often used for clustering and winner-take-all

tasks. In MaxNet, the neuron with the maximum input value wins and becomes the active

neuron.

Here's how to implement a program to find the winning neuron using MaxNet:

1. Neuron Activation:

• Initialize the input values for each neuron.

• Each neuron computes its net input, which is the sum of its input values.

• Neurons compete to become active based on their net inputs.

2. Winner Selection:

• Identify the neuron with the maximum net input.

• This neuron becomes the winning neuron.

3. Update Weights:

• Update the input values for all neurons based on the winner's input values.

• Increase the input values of the winning neuron to strengthen its influence.

• Decrease the input values of other neurons to reduce their influence.

4. Iteration:

• Repeat the process for multiple iterations or until convergence.

• Convergence occurs when the winning neuron remains stable, indicating that it has

learned the optimal input pattern.

Implementation Steps:

1. Initialize input values for each neuron.

2. Compute net inputs for all neurons.

3. Identify the neuron with the maximum net input (the winning neuron).

4. Update input values based on the winning neuron's input values.

5. Repeat steps 2-4 for multiple iterations or until convergence.

Practical No. 7

Aim: Implement Union, Intersection, Complement and Difference operations on fuzzy

sets.

Theory:

Implementing set operations like Union, Intersection, Complement, and Difference on fuzzy

sets involves extending these operations from crisp sets to fuzzy sets. Fuzzy sets allow

elements to have degrees of membership rather than being strictly in or out of the set.

Union of Fuzzy Sets:

• The union of two fuzzy sets A and B is another fuzzy set C, where

μC(x)=maxμA(x),μB(x))\mu\_C(x) for all x in the universe of discourse X.

• This operation results in a fuzzy set that includes elements that belong to either A or

B, or both.

Intersection of Fuzzy Sets:

• The intersection of two fuzzy sets A and B is another fuzzy set C, where

μC(x)=min(μA(x),μB(x))\mu\_C(x) for all x in X.

• This operation results in a fuzzy set that includes elements that belong to both A and

BBB.

Complement of a Fuzzy Set:

• The complement of a fuzzy set A is another fuzzy set C, where μC(x)=1−μA(x) for all

x in X.

• This operation results in a fuzzy set that includes elements that do not belong to A.

Difference of Fuzzy Sets:

• The difference between two fuzzy sets A and B is another fuzzy set C, where μC

(x)=max(μA(x)−μB(x),0) for all x in X.

• This operation results in a fuzzy set that includes elements that belong to A but not to

B.