Assignment 1 - INFO7375
Aman khan
002050777

1. How does natural neuron work?

    Structure of a Neuron:
    Three major components make up a neuron:
    Dendrites: Take in information from neighboring neurons.
    The soma, or cell body, processes incoming impulses and decides whether to fire a neuron. Axon: Sends the message to muscles or other neurons.
    Operational Mechanism:
    The dendrites receive signals from nearby neurons, which are then integrated in the cell body. The neuron produces an action potential, or electrical impulse, if the sum of the signals is greater than a predetermined threshold.
    This impulse makes its way to the synaptic terminals via the axon, where it forms connections with other neurons. Important attributes:
    According to the all-or-none concept, neurons can either fire (produce an action potential) or stay dormant.
    The myelin sheath of the neuron and the stimulus's intensity determine the signal's speed and strength.

2. How does natural neuron transmit signal to other neurons?

    Communication Between Synapses:
    Chemicals known as neurotransmitters are released into the synaptic cleft (the space between two neurons) when the action potential reaches the axon terminal. Initiating a fresh signal in the subsequent cell, these neurotransmitters attach to receptors on the postsynaptic neuron.

    Signals that are excitatory vs inhibitory:
    Increase the probability that the subsequent cell will fire by sending out excitatory signals.
    Inhibitory Signals: Make firing less likely.
    The cumulative effect of all excitatory and inhibitory inputs determines a neuron's response.

    Learning and Adaptation:
    Learning and memory development are made possible by synaptic connections that either weaken if not used or strengthen with repeated usage (a process known as long-term potentiation). The brain can adapt and efficiently store information because to this dynamic change of synapse strength.
    The Refractory Period:
    A neuron goes through a refractory phase after firing, during which it resets and is unable to fire again right away.

3. Describe the McCulloch and Pitts model of artificial neuron?

Summary:

In 1943, Walter Pitts and Warren McCulloch presented the MCP model.
It is a mathematically simplified representation of a biological neuron that uses binary logic to carry out calculations.
How It Operates:

The inputs $X = \{x1, x2, ..., xN\}$ X={x 1,x 2,...,x N} are binary (0 or 1), signifying the presence or absence of a signal.
Each input's relevance is determined by its corresponding
weight, w k. The weighted sum of the inputs is calculated by the
neuron:
$g(X) = \sum k = 1\ Nwkx$
g(X) = k=1 ∑ N w k x k

The neuron fires and produces an output of $y = 1$ y=1 if $g(X)$ g(X) over a predetermined threshold θ θ; else, $y = 0$ y=0.
Qualities:
Carries out basic logical operations such as AND, OR, and NOT.
lacks time-based dynamics, meaning that signals don't build up or deteriorate over time. excludes the refractory period present in biological neurons as well as inhibitory inputs. Restrictions:

Only linearly separable data may be handled by a single MCP neuron (many neurons are needed for tasks like XOR). does not replicate complex biological processes like memory or learning. Importance:

The foundation for contemporary artificial neural networks was established by the MCP model. More complex learning algorithms (like gradient descent) and activation functions (like sigmoid and ReLU) are incorporated into artificial neurons nowadays.

Programming Assignment:

```
class MCPNeuron:

        """

        Simulates a McCulloch and Pitts artificial neuron.

        The neuron processes binary inputs and generates binary outputs using a
      threshold-based activation function.

        """     def __init__(self,

      weights, threshold):

            """

            Initialize the MCP neuron.


            Parameters:

       - weights (list of int/float): List of weights for the inputs.

       - threshold (int/float): The threshold for activation.

            """

      self.weights = weights

      self.threshold =

      threshold


        def activate(self, inputs):

            """

            Calculate the neuron's output based on inputs and the activation function.


            Parameters:

       - inputs (list of int): List of binary inputs (0 or 1).
```

```
        Returns:
      - int: 1 if the neuron activates, 0 otherwise.

        """         if len(inputs) !=

len(self.weights):

            raise ValueError("Number of inputs must match the number of weights.")


        # Calculate the weighted sum of inputs

weighted_sum = sum(w * x for w, x in zip(self.weights,

inputs))


        # Apply the activation function (step

function)         return 1 if weighted_sum >=

self.threshold else 0


    def __str__(self):

        """

        String representation of the MCP neuron.


        Returns:

        - str: Description of the neuron.

        """

        return f"MCP Neuron(weights={self.weights}, threshold={self.threshold})"


# Example usage if __name__

== "__main__":     # Define

weights and threshold     weights

= [1, 1]  # Weights for inputs

threshold = 2     # Activation

threshold     # Create an MCP
```

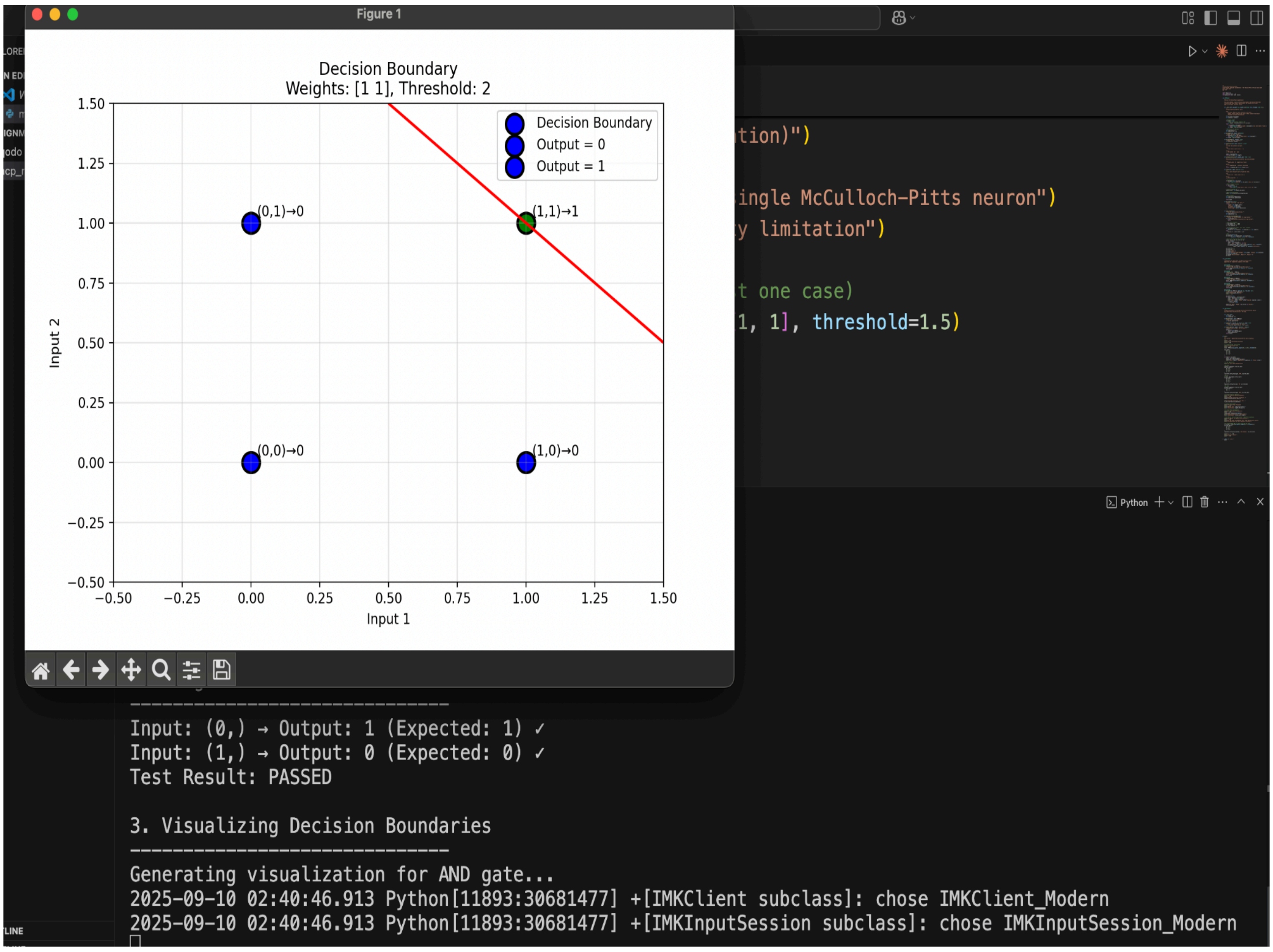
```
neuron     neuron =
MCPNeuron(weights, threshold)

    # Test inputs
test_inputs = [
        [0, 0],  # Expected output: 0
        [0, 1],  # Expected output: 0
        [1, 0],  # Expected
output: 0        [1, 1],  #
Expected output: 1
    ]

    # Display results
print(neuron)
print("\nTesting inputs:")
for inputs in test_inputs:
        output = neuron.activate(inputs)
print(f"Inputs: {inputs}, Output: {output}")
OUTPUT:
```

```
MCP Neuron(weights=[1, 1], threshold=2)

Testing inputs:
Inputs: [0, 0], Output: 0
Inputs: [0, 1], Output: 0
Inputs: [1, 0], Output: 0
Inputs: [1, 1], Output: 1
```



```
tion)")
ingle McCulloch-Pitts neuron")
y limitation")

t one case)
1, 1], threshold=1.5)
```

```
------------------------------
Input: (0,) → Output: 1 (Expected: 1) ✓
Input: (1,) → Output: 0 (Expected: 0) ✓
Test Result: PASSED

3. Visualizing Decision Boundaries
------------------------------
Generating visualization for AND gate...
2025-09-10 02:40:46.913 Python[11893:30681477] +[IMKClient subclass]: chose IMKClient_Modern
2025-09-10 02:40:46.913 Python[11893:30681477] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```