VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY (AUTONOMOUS)
Accredited by NBA (B.Tech program), Approved by AICTE, Permanently Affiliated to JNTUK,NAAC Accredited with 'A' Grade, ISO 9001:2015 Certified
Nambur (V), Pedakakani (M), Guntur (Dt.), Andhra Pradesh – 522 508,
www.vvitguntur.com

ARTIFICIAL NEURAL NETWORKS (CSM)

UNIT-1: Foundations

1. Write a python program for the addition, subtraction, division, dot product and multiplication of two vectors with example.
2. Write a python program for the addition, subtraction, division, dot product and multiplication of two matrices with example.
3. Discuss about data pre-processing steps in detail.
4. What is scikit-learn? Explain Sklearn SimpleImputer with simple Example.
5. Write a python program to read two matrices and to perform linear transformations?

UNIT-2: Introduction to Artificial Neural Network

6. What are the benefits of neural networks? Explain the functioning of Biological Neuron?
7. Why activation function is used in artificial neuron? Explain any diffrent activation functions.
8. Discuss the working of perceptron model with the relevant algorithm.
9. Explain with a neat sketch the McCulloch-Pitts model of artificial neural network.
10. Write a python program to implement logic gates with perceptron?

UNIT-3: Single Layer Perceptron

11. Write a short note on the artificial neural network architecture model.
12. Differentiate between supervised and unsupervised learning with suitable examples?
13. Explain learning methods in details.
14. "Perceptron cannot implement XOR gate"-Justify your answer with an example?
Differentiate between single layer feed forward networks and multilayer feed forward neural networks?

Certainly! Let's go through each question step by step.

Question 1: Write a python program for the addition, subtraction, division, dot product, and multiplication of two vectors with an example.

Here's a Python program to perform these operations on two vectors:

```python
import numpy as np

# Define two vectors as NumPy arrays
vector1 = np.array([1, 2, 3])
vector2 = np.array([4, 5, 6])

# Addition of vectors
addition_result = vector1 + vector2

# Subtraction of vectors
subtraction_result = vector1 - vector2

# Element-wise multiplication of vectors
elementwise_multiplication = vector1 * vector2

# Element-wise division of vectors
elementwise_division = vector1 / vector2

# Dot product of vectors
dot_product = np.dot(vector1, vector2)

# Display results
print("Vector 1:", vector1)
print("Vector 2:", vector2)
print("Addition:", addition_result)
print("Subtraction:", subtraction_result)
print("Element-wise Multiplication:", elementwise_multiplication)
print("Element-wise Division:", elementwise_division)
print("Dot Product:", dot_product)
```

Question 2: Write a python program for the addition, subtraction, division, dot product, and multiplication of two matrices with an example.

Here's a Python program for matrix operations:

```python
import numpy as np

# Define two matrices as NumPy arrays
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])

# Addition of matrices
addition_result = matrix1 + matrix2

# Subtraction of matrices
subtraction_result = matrix1 - matrix2

# Element-wise multiplication of matrices
elementwise_multiplication = matrix1 * matrix2

# Element-wise division of matrices
elementwise_division = matrix1 / matrix2

# Matrix multiplication (dot product)
matrix_multiplication = np.dot(matrix1, matrix2)

# Display results
print("Matrix 1:\n", matrix1)
print("Matrix 2:\n", matrix2)
print("Addition:\n", addition_result)
print("Subtraction:\n", subtraction_result)
print("Element-wise Multiplication:\n", elementwise_multiplication)
print("Element-wise Division:\n", elementwise_division)
print("Matrix Multiplication (Dot Product):\n", matrix_multiplication)
```

Question 3: Discuss data pre-processing steps in detail.

Data pre-processing steps typically include:

- Data Cleaning: Handling missing values, noisy data, and outliers.
- Data Transformation: Scaling, normalization, and encoding categorical data.
- Data Reduction: Feature selection and dimensionality reduction.
- Data Discretization: Binning numerical data.
- Data Sampling: Random sampling for large datasets.
- Data Splitting: Splitting data into training and testing sets.
- Data Integration: Combining data from multiple sources.

Each step is performed to prepare data for machine learning or analysis.

Question 4: What is scikit-learn? Explain Sklearn SimpleImputer with a simple example.

Scikit-learn (sklearn) is a popular Python library for machine learning and data analysis. It provides tools for various machine learning tasks, including classification, regression, clustering, and more.

`SimpleImputer` is a class in scikit-learn used for handling missing data by imputing (replacing) missing values with specified strategies like mean, median, or constant values. Here's a simple example:

```python
from sklearn.impute import SimpleImputer
import numpy as np

# Create a dataset with missing values represented as NaN
data = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])

# Create a SimpleImputer instance with strategy 'mean'
imputer = SimpleImputer(strategy='mean')

# Fit the imputer to the data and transform the data
imputed_data = imputer.fit_transform(data)

# Display the imputed data
print("Original Data:\n", data)
print("Imputed Data:\n", imputed_data)
```

In this example, missing values in the dataset are replaced with the mean of the available values in each column.

Question 5: Write a Python program to read two matrices and perform linear transformations?

To perform linear transformations, you'll need to specify transformation matrices. Here's an example:

```python
import numpy as np

# Define a 2x2 matrix
matrix = np.array([[1, 2], [3, 4]])

# Define a transformation matrix for scaling by 2
scaling_matrix = np.array([[2, 0], [0, 2]])

# Define a transformation matrix for rotation by 90 degrees
rotation_matrix = np.array([[0, -1], [1, 0]])

# Apply scaling and rotation transformations


scaled_matrix = np.dot(scaling_matrix, matrix)
rotated_matrix = np.dot(rotation_matrix, matrix)

# Display original, scaled, and rotated matrices
print("Original Matrix:\n", matrix)
print("Scaled Matrix:\n", scaled_matrix)
print("Rotated Matrix:\n", rotated_matrix)
```

In this example, we define an original matrix and two transformation matrices for scaling and rotating. We then apply these transformations using matrix multiplication to obtain the scaled and rotated matrices.

Certainly! Let's go through each question step by step.

Question 6: What are the benefits of neural networks? Explain the functioning of a Biological Neuron.

Benefits of Neural Networks:
- **Non-Linearity**: Neural networks can model complex, non-linear relationships in data.
- **Adaptability**: They can adapt and learn from new data, making them suitable for various tasks.
- **Parallel Processing**: Neural networks can perform parallel processing, speeding up computations.
- **Robustness**: They are robust to noisy data and can generalize well.

Functioning of a Biological Neuron:
A biological neuron consists of:
- **Dendrites**: Receive signals from other neurons.
- **Cell Body (Soma)**: Integrates received signals.
- **Axon**: Transmits signals to other neurons.

When a signal (action potential) reaches the axon's end, it triggers the release of neurotransmitters, which affect neighboring neurons' dendrites, continuing the signal transmission.

Question 7: Why is an activation function used in an artificial neuron? Explain different activation functions.

An activation function is used to introduce non-linearity into the output of an artificial neuron, allowing neural networks to model complex relationships.

Different Activation Functions:
- **Sigmoid**: Smooth S-shaped curve, suitable for binary classification.
- **ReLU (Rectified Linear Unit)**: f(x) = max(0, x), efficient and commonly used.
- **Tanh (Hyperbolic Tangent)**: S-shaped curve like sigmoid but centered at 0.
- **Leaky ReLU**: f(x) = x if x > 0, else f(x) = ax (small slope for x < 0).
- **Softmax**: Used in the output layer for multi-class classification.

Question 8: Discuss the working of the perceptron model with the relevant algorithm.

The Perceptron Model:
- Inputs are weighted and summed.
- A bias is added.
- The result is passed through an activation function (typically a step function).
- Output is the result of the activation function (usually 0 or 1).

Perceptron Algorithm:
1. Initialize weights and bias.
2. For each training example:
   - Calculate the weighted sum of inputs.
   - Apply the activation function.
   - Update weights and bias based on the error.
3. Repeat until convergence.

Question 9: Explain with a neat sketch the McCulloch-Pitts model of an artificial neural network.

The McCulloch-Pitts (M-P) Model:
- Inputs are received from other neurons.
- Each input is multiplied by a weight.
- The weighted inputs are summed.
- If the sum exceeds a threshold, the neuron fires (output is 1); otherwise, it doesn't fire (output is 0).

[Sketch of the M-P Model]

Question 10: Write a Python program to implement logic gates with a perceptron?

Here's a Python program to implement the AND logic gate using a perceptron:

```python
import numpy as np

# Define the AND gate truth table (inputs and corresponding outputs)
data = np.array([[0, 0, 0],
                 [0, 1, 0],
                 [1, 0, 0],
                 [1, 1, 1]])

# Define initial weights and bias
weights = np.array([0.5, 0.5])
bias = -0.7  # Adjust the bias as needed

# Define the perceptron function
def perceptron(input_data, weights, bias):
    weighted_sum = np.dot(input_data[:2], weights) + bias
    if weighted_sum >= 0:
        return 1
    else:
        return 0

# Test the perceptron with the AND gate truth table
for row in data:
    input_data = row[:2]
    target_output = row[2]
    output = perceptron(input_data, weights, bias)
    print(f"Input: {input_data}, Target Output: {target_output}, Perceptron Output: {output}")
```

This program implements the AND gate logic using a perceptron with adjustable weights and bias. You can similarly implement other logic gates like OR and NOT using perceptrons.

Certainly! Let's continue with the answers to the next set of questions:

Question 11: Write a short note on the artificial neural network architecture model.

Artificial Neural Network (ANN) Architecture Model:
- ANNs are composed of layers of interconnected nodes (neurons).
- Common architectures include feedforward, recurrent, and convolutional neural networks.
- Input layer receives data, hidden layers process it, and the output layer produces results.
- Neurons in layers communicate via weighted connections.
- ANNs can have various topologies, such as deep networks with many hidden layers.

Question 12: Differentiate between supervised and unsupervised learning with suitable examples?

Supervised Learning:
- Requires labeled data (input-output pairs) for training.
- Aims to learn a mapping from inputs to outputs.
- Examples: Classification (e.g., spam detection), Regression (e.g., house price prediction).

Unsupervised Learning:
- Works with unlabeled data to find patterns or structures.
- Includes clustering and dimensionality reduction.
- Examples: K-Means Clustering, Principal Component Analysis (PCA).

Question 13: Explain learning methods in detail.

Learning Methods:
- **Supervised Learning**: Learns from labeled data, makes predictions.
- **Unsupervised Learning**: Extracts patterns or structures from unlabeled data.
- **Semi-Supervised Learning**: Uses a mix of labeled and unlabeled data.
- **Reinforcement Learning**: Learns by interacting with an environment, aiming to maximize rewards.
- **Self-Supervised Learning**: Uses data's inherent structure (e.g., predicting missing parts of data).
- **Transfer Learning**: Applies knowledge from one task to another (e.g., pre-trained models).

Question 14: “Perceptron cannot implement XOR gate” - Justify your answer with an example?

The Perceptron cannot implement XOR gate due to its linearity. XOR is a non-linearly separable function, which means a single straight line cannot separate its outputs. Here's an example:

XOR Truth Table:
```
| Input 1 | Input 2 | Output |
|---------|---------|--------|
|   0     |   0     |   0    |
|   0     |   1     |   1    |
|   1     |   0     |   1    |
|   1     |   1     |   0    |
```

As you can see, there's no single linear equation that can correctly classify the XOR function's outputs. The Perceptron's activation function (e.g., step function) is linear, so it cannot handle XOR.

Question 15: Differentiate between single-layer feedforward networks and multilayer feedforward neural networks?

Single-Layer Feedforward Networks:
- Consist of an input layer and an output layer.
- Typically used for linearly separable problems.
- Cannot represent complex relationships.
- Suitable for simple tasks like linear regression.

Multilayer Feedforward Neural Networks (MLP):
- Have multiple hidden layers between the input and output layers.
- Use non-linear activation functions (e.g., ReLU).
- Can approximate complex functions.
- Suitable for a wide range of tasks, including deep learning.

MLPs can capture intricate patterns and relationships in data, making them more versatile than single-layer networks.