Title: - Adaptive Resonance Theory (ART) Neural Network

Aim/Objective: - To implement and demonstrate an Adaptive Resonance Theory (ART) neural network for pattern recognition and clustering.

Software Required:

- Python programming environment (Jupyter Notebook, Google Colab, or any Python IDE)
- NumPy library
- Matplotlib library

Hardware Required:

- 4GB RAM
- Intel i3 or higher / AMD equivalent
- 120 GB SSD
- GPU for accelerated computations (if using deep learning frameworks)

Theory:
Adaptive Resonance Theory (ART) is a family of neural networks developed for pattern recognition and clustering. It enables stable learning while allowing new patterns to be learned. ART networks use a **vigilance parameter** to determine whether an input pattern belongs to an existing category or if a new category should be created.

Key components of ART networks:

1. **Comparison Field:** The input pattern is compared with stored patterns.
2. **Recognition Field:** If a pattern matches within the vigilance threshold, it is classified into an existing category; otherwise, a new category is created.
3. **Vigilance Parameter (ρ):** Controls the degree of similarity required for pattern grouping.

ART networks are widely used in applications like character recognition, signal processing, and clustering.

Procedure:

1. Define input patterns for training.
2. Initialize weight matrices and vigilance parameter.
3. For each input pattern:
    - Compute similarity with stored categories.

- If similarity is above the vigilance threshold, assign the pattern to an existing category.
- Otherwise, create a new category.

4. Repeat the process for all input patterns.
5. Test the network with new inputs.

Code:

```
import numpy as np

class ART:
    def __init__(self, input_size, vigilance=0.7):
        self.input_size = input_size
        self.vigilance = vigilance
        self.weights = []  # List to store category weights

    def train(self, inputs):
        for x in inputs:
            matched = False
            for i, w in enumerate(self.weights):
                similarity = np.sum(np.minimum(x, w)) / np.sum(x)
                print(f"Checking similarity with Category {i+1}:
{similarity:.2f}")
                if similarity >= self.vigilance:
                    print(f"Pattern matched with Category {i+1}. Updating
weights.")
                    self.weights[i] = np.minimum(x, w)  # Update existing
category
                    matched = True
                    break
            if not matched:
                print("No match found. Creating a new category.")
                self.weights.append(x)  # Create new category

    def test(self, x):
        print("\nTesting new pattern:", x)
        for i, w in enumerate(self.weights):
            similarity = np.sum(np.minimum(x, w)) / np.sum(x)
            print(f"Similarity with Category {i+1}: {similarity:.2f}")
            if similarity >= self.vigilance:
                return f"Pattern belongs to Category {i+1}"
        return "New category required"

# Define input patterns
inputs = np.array([
    [1, 1, 0, 0],
    [1, 1, 1, 0],
```

```
    [0, 0, 1, 1],
    [0, 1, 1, 1]
])

# Create ART network with vigilance 0.7
art_network = ART(input_size=4, vigilance=0.7)
art_network.train(inputs)

# Test with a new pattern
new_pattern = np.array([1, 1, 0, 1])
print(art_network.test(new_pattern))
```

Output:

```
PS D:\College Work\ANN\ANN Practical 4> & C:/Python312/python.exe "d:/College Work/ANN/ANN Practical 4/ART.py"
No match found. Creating a new category.
Checking similarity with Category 1: 0.67
No match found. Creating a new category.
Checking similarity with Category 1: 0.67
Checking similarity with Category 1: 0.67
No match found. Creating a new category.
Checking similarity with Category 1: 0.00
Checking similarity with Category 2: 0.50
No match found. Creating a new category.
Checking similarity with Category 1: 0.33
Checking similarity with Category 2: 0.67
Checking similarity with Category 3: 0.67
No match found. Creating a new category.

Testing new pattern: [1 1 0 1]
Similarity with Category 1: 0.67
Similarity with Category 2: 0.67
Similarity with Category 3: 0.33
Similarity with Category 4: 0.67
New category required
```

Observations:

- The ART network successfully classifies input patterns into categories based on similarity.
- When a pattern does not match any existing category, a new category is created.
- The vigilance parameter determines the strictness of category formation.

Conclusion:

The Adaptive Resonance Theory (ART) neural network is an effective clustering model that allows stable learning while accommodating new patterns. By adjusting the vigilance parameter, the network can control how strictly it groups similar inputs, making it useful for real-world classification tasks.