```python
import numpy as np

class ART1:
    def __init__(self, num_features, num_categories, vigilance=0.8):
        self.num_features = num_features      # Size of input vector
        self.num_categories = num_categories  # Max number of
categories
        self.vigilance = vigilance            # Vigilance threshold
        self.b = num_features + 1             # Choice parameter

        # Initialize weights
        self.bottom_up = np.ones((num_categories, num_features)) / (1
+ num_features)
        self.top_down = np.ones((num_categories, num_features))

    def train(self, patterns):
        for input_vector in patterns:
            input_vector = np.array(input_vector)
            # Calculate activations
            activations = np.sum(np.minimum(self.bottom_up,
input_vector), axis=1)
            activations /= (self.b + np.sum(self.bottom_up, axis=1))

            # Sort by highest activation
            sorted_indices = np.argsort(-activations)

            for j in sorted_indices:
                # Predictive match (top-down expectation)
                match = np.sum(np.minimum(input_vector,
self.top_down[j])) / np.sum(input_vector)

                # Check vigilance
                if match >= self.vigilance:
                    # Resonance — update weights
                    self.bottom_up[j] = (input_vector *
self.top_down[j]) / (0.5 + np.sum(input_vector * self.top_down[j]))
                    self.top_down[j] = input_vector * self.top_down[j]
                    print(f"Pattern {input_vector.tolist()} -->
Category {j}")
                    break
            else:
                print(f"Pattern {input_vector.tolist()} --> No
category found (vigilance too high)")

# Example input patterns (binary)
patterns = [
    [1, 0, 0, 1, 0],
    [1, 1, 0, 1, 0],
    [0, 0, 1, 0, 1],
    [0, 1, 1, 0, 1],
```

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

# Create ART1 network
art = ART1(num_features=5, num_categories=5, vigilance=0.6)


# Train the network
art.train(patterns)
```

```
Pattern [1, 0, 0, 1, 0] --> Category 0
Pattern [1, 1, 0, 1, 0] --> Category 0
Pattern [0, 0, 1, 0, 1] --> Category 1
Pattern [0, 1, 1, 0, 1] --> Category 1
Pattern [1, 0, 0, 1, 1] --> Category 0
```