Practical No: 07

Title: To apply the artificial immune pattern recognition to perform a task of structure damage Classification

Problem Statement: Develop an artificial immune pattern recognition system for the task of structural damage classification.

Prerequisite:
Basics of Python

Software Requirements: Jupyter Notebook or any Python IDE
Required Python libraries (scikit-learn, numpy, matplotlib)

Hardware Requirements:
Core I3 Processor, 4GB RAM, 500 GB HDD

Learning Objectives:

1. Design a robust algorithm capable of identifying patterns indicative of structural damage within complex datasets.

2. Implement a scalable solution that can accurately classify various types and extents of structural damage with high efficiency.

Outcomes:
After completion of this assignment students are able to understand how to The developed system will enable automated and accurate identification of structural damage, aiding in timely maintenance and ensuring safety.

Theory:
Introduction to AIS:
- Briefly discuss the biological immune system and its key functions.
- Introduce the concept of AIS and its principles for pattern recognition.
- Explain the analogy between antigens (foreign invaders) and damage patterns, and antibodies (immune cells) and damage detectors.

2. Setting Up the AIS Model:

- Divide the students into groups and assign each group a specific damage type (e.g., crack, corrosion). Provide each group with a training dataset containing features of healthy and damaged structures related to their assigned damage type.
- Guide students through the process of encoding the data into a format suitable for the AIS algorithm (e.g., binary strings representing feature values).

- Introduce the chosen AIS algorithm and its key parameters (e.g., population size, mutation rate).

Assist students in implementing the algorithm using the provided library or framework.

3. Training and Testing the Model:

- Run the AIS algorithm on the training data for each group.
- Explain the selection, mutation, and cloning processes within the algorithm.
- Observe how the "antibodies" (damage detectors) evolve over generations to better recognize the assigned damage pattern.
- Once training is complete, test the model with unseen data containing both healthy and damaged structures.
- Evaluate the performance of the AIS model based on its accuracy in detecting the assigned damage type.

4. Analysis and Discussion:

- Each group presents their results and discusses the performance of their AIS model.
- Compare the effectiveness of different damage detection algorithms or parameter settings across groups.
- Discuss the advantages and limitations of using AIS for damage detection.
- Explore potential applications of AIS in real-world engineering scenarios.

5. Diagram:

A simplified diagram illustrating the key steps of the lab experiment:

```
| Structure Data | (Healthy & Damaged)
+--------------------+
|
v
+--------------------+
| Feature Extraction |
+--------------------+
|
v
+--------------------+
| Data Encoding | (Binary Strings)
+--------------------+

|
v
+--------------------+
| AIS Algorithm | (Clonal Selection)
+--------------------+
|
v
+--------------------+
| Trained Detectors | (Damage Patterns)
+--------------------+
|
v
+--------------------+
| Test Data | (Unseen Structures)
+--------------------+
|
v
+--------------------+
| Damage Detection | (Accuracy Evaluation)
+--------------------+
|
v
+--------------------+
| Results & Discussion| (Performance Analysis)
+--------------------+
```

Note: This is a general outline and can be adapted based on the specific AIS algorithm, chosen damage types, and available resources.

This code represents a process flow for a damage detection system using an AIS algorithm.

1. Structure Data: Represents the input data for the system, which includes information about healthy and damaged structures.
2. Feature Extraction: Extracts relevant features from the input data to be used in the detection process.
3. Data Encoding: Converts the extracted features into binary strings for processing by the AIS algorithm.
4. AIS Algorithm: Utilizes a Clonal Selection algorithm to detect damage patterns in the encoded data.
5. Trained Detectors: Represents the detectors that have been trained to recognize specific damage patterns.
6. Test Data: Contains unseen structures that will be used to test the performance of the detection system.
7. Damage Detection: Evaluates the accuracy of the detection system in identifying damage in the test data.
8. Results & Discussion: Analyzes the performance of the system and discusses the results obtained from the damage detection process.

Code:
To apply the artificial immune pattern recognition to perform a task of structure damage Classification

```
import numpy as np
# Generate dummy data for demonstration purposes (replace this with your actual data)
def generate_dummy_data(samples=100, features=10):
    data = np.random.rand(samples, features)
    labels = np.random.randint(0, 2, size=samples)
    return data, labels
# Define the AIRS algorithm
class AIRS:
    def __init__(self, num_detectors=10, hypermutation_rate=0.1):
        self.num_detectors = num_detectors
        self.hypermutation_rate = hypermutation_rate

    def train(self, X, y):
        self.detectors = X[np.random.choice(len(X), self.num_detectors, replace=False)]

    def predict(self, X):
        predictions = []
        for sample in X:
            distances = np.linalg.norm(self.detectors - sample, axis=1)
            prediction = int(np.argmin(distances))
            predictions.append(prediction)
        return predictions
    # Generate dummy data
data, labels = generate_dummy_data()
# Split data into training and testing sets
split_ratio = 0.8
split_index = int(split_ratio * len(data))
train_data, test_data = data[:split_index], data[split_index:]
train_labels, test_labels = labels[:split_index], labels[split_index:]

# Initialize and train AIRS
airs = AIRS(num_detectors=10, hypermutation_rate=0.1)
airs.train(train_data, train_labels)
# Test AIRS on the test set
predictions = airs.predict(test_data)
# Evaluate accuracy
accuracy = np.mean(predictions == test_labels)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.15
```

Conclusion: This way, Implemented the artificial immune pattern recognition to perform a task of structure damage Classification