Test Document for ARCE Algorithm

1. Overview

The ARCE algorithm is designed to classify input data into categories based on contextual information. It uses a vigilance parameter to determine whether an input resonates with an existing category or creates a new one. The algorithm also updates category prototypes and context weights during training.

This document outlines the test cases, test results, and analysis of the ARCE algorithm.

2. Test Cases

Test Case 1: Training with Sample Data

- **Input Data**:
  input_data = [[10, 20], [12, 22], [11, 19], [5, 8], [6, 9]]
- **Contexts**:
  contexts = [{"time": 9, "location": "office"}, {"time": 12, "location": "restaurant"}, {"time": 15, "location": "office"}, {"time": 19, "location": "home"}, {"time": 21, "location": "home"}]
- **Description**:
  Train the ARCE model with the provided input data and contexts. Verify that the model creates categories and updates context weights.

Test Case 2: Prediction with New Input

- **Input Data**:
  [11, 21]
- **Context**:
  {'time': 16, 'location': 'office'}
- **Description**:
  Predict the category for the new input and context. Verify that the prediction aligns with the closest category.

Test Case 3: Prediction with Another New Input

- **Input Data**:
  [5, 7]
- **Context**:
  {'time': 20, 'location': 'home'}

- **Description**:
  Predict the category for the new input and context. Verify that the prediction aligns with the closest category.

3. Test Results

Test Case 1 Results

- **Category Prototypes**:

```
Copy

[array([10., 20.,  9.,  1.]),
 array([12., 22., 12.,  1.]),
 array([11., 19., 15.,  1.]),
 array([5., 8., 19., 1.]),
 array([6., 9., 21., 1.])]
```

  Each prototype includes the input data concatenated with the context vector (e.g., `[input_data, time, location_encoded]`).
- **Context Weights**:

```
Copy

{0: {'time': 0.1, 'location': 0.1},
 1: {'time': 0.1, 'location': 0.1},
 2: {'time': 0.1, 'location': 0.1},
 3: {'time': 0.1, 'location': 0.1},
 4: {'time': 0.1, 'location': 0.1}}
```

  Context weights are initialized and updated during training.

Test Case 2 Results

- **Prediction**:
  `Prediction for [11, 21] in context {'time': 16, 'location': 'office'}: 2`
  The input `[11, 21]` with context `{'time': 16, 'location': 'office'}` is classified into category `2`.

Test Case 3 Results

- **Prediction**:
  `Prediction for [5, 7] in context {'time': 20, 'location': 'home'}: 3`
  The input `[5, 7]` with context `{'time': 20, 'location': 'home'}` is classified into category `3`.

4. Analysis

Training Phase

- The algorithm successfully created 5 categories, one for each input-context pair.
- Category prototypes were updated to include both input data and context information.
- Context weights were initialized and updated for each category.

Prediction Phase

- The model correctly predicted the closest category for new inputs based on the learned prototypes and context weights.
- For example, the input $[11, 21]$ was classified into category 2, which corresponds to the prototype $[11, 19, 15, 1]$. This makes sense because the input is close to the prototype in terms of both input data and context.

Limitations

- The algorithm assumes that the context is represented as a vector. More sophisticated embedding methods could improve performance.
- The vigilance parameter ($\text{vigilance}=0.8$) and learning rate ($\text{learning\_rate}=0.1$) are fixed. These hyperparameters may need tuning for specific datasets.
- The algorithm does not handle missing or noisy context data.

5. Conclusion

The ARCE algorithm demonstrates the ability to classify input data based on contextual information. It successfully creates and updates categories during training and makes reasonable predictions for new inputs. However, further improvements could be made to handle more complex contexts and optimize hyperparameters.