GMM – Gaussian Mixture Model

MACHINE INTELLIGENCE LABORATORY

Assignment Overview

In this assignment, you will implement a Gaussian Mixture Model (GMM) using the Expectation Maximization (EM) algorithm. The GMM is a probabilistic model that represents data as a mixture of several Gaussian distributions. Your task is to complete the implementation of the GMM in the provided **GMM.py** file. You should complete the implementation of the functions and use the EM algorithm to estimate the parameters of the Gaussian components. The **Test.py** file contains test cases that you can use to validate your implementation.

Task Details

- Complete the code for the class **GMMModel** and its functions in the **GMM.py** file.

Files Provided

1. **GMM.py**: Contains the structure of the **GMMModel** class and partially implemented functions.
2. **Test.py**: Contains sample test cases that you can use for reference and validation.

Important Points

1. Do not make changes to the function definitions provided in the skeleton code.
2. You are free to write any helper functions in the file named GMM**.py**.
3. Your code will be auto-evaluated, and the dataset and test cases will not be revealed.
4. Avoid plagiarism; your code will be checked for plagiarism, and both the provider and receiver of plagiarized code will receive zero marks.
5. Do not change variable names or use techniques to evade plagiarism checks.
6. Hidden test cases will not be revealed post-evaluation.

Tasks to Complete Gaussian Mixture Model (GMM) Implementation
Tasks to Complete:

1. Constructor `__init__(self, n_components)`
   - Initialize model parameters:
   - `n_components`: Number of Gaussian components.
   - Set equal initial weights.
   - Initialize means with random values.
   - Initialize covariances as zeros.

2. Function `fit(self, X, max_iters=100, tol=1e-4)`
   - Implement EM algorithm for model fitting.
   - Complete E-step (compute responsibilities).
   - Complete M-step (update weights, means, and covariances).
   - Implement convergence check using `_is_converged`.

3. Function `_e_step(self, X)`
   - Implement E-step to compute responsibilities.

4. Function `_m_step(self, X, responsibilities)`
   - Implement M-step to update model parameters.

5. Function `_log_likelihood(self, X, responsibilities)`
   - Compute log-likelihood of data given model.

6. Function `_is_converged(self, X, responsibilities, tol)`
   - Implement convergence check based on log-likelihood change.

7. Function `predict(self, X)`
   - Assign cluster labels to data based on responsibilities.

8. Function `_inverse(self, matrix)`
   - Compute matrix inverse with regularization.

Additional Information

- You may write your own helper functions if needed.
- You can import libraries that come built-in with Python 3.7.
- Do not change the skeleton of the code.

Testing Your Code

1. Use **Test.py** to test your code.
2. Passing the sample cases does not guarantee full marks; consider edge cases.
3. Name your code file as **CAMPUS_SECTION_SRN_Lab6.py**

4. To run your code: in the terminal run: python Test.py –ID CAMPUS_SECTION_SRN.py