Reflection Report on GMM-EM

Kim Ying WONG

11 April 2024

Reflection is an important component of getting the full benefits from a learning experience. Besides the intrinsic benefits of reflection, this document will be used to help the TAs grade how well your team responded to feedback. In addition, several CEAB (Canadian Engineering Accreditation Board) Learning Outcomes (LOs) will be assessed based on your reflections.

1 Changes in Response to Feedback

This section summarize the changes made over the course of the project in response to feedback from TAs, the instructor, teammates, other teams, the project supervisor (if present), and from user testers.

The changes will be highlighted with respect to supervisor's response to the Rev 0 demonstration.

1.1 SRS and Hazard Analysis

A revised version of SRS has been updated. In regard to our reviewer team, we fix the typo or any grammatical issues in our documents. We listed out our changes by section. For the section of reference material, we modify the table of symbols by adding some more notations (on defining vector, matrix or statistics). We hope this could avoid any ambiguity and confusion. We also clarify the meaning of notations in section of mathematical notation and table of symbol to enhance the readiblity of the document. In section 2, we are giving more details on the characteristics of intended reader, and more information about the software name: GMM-EM. In section 4, we aims at a clearer problem description in the updated version. We first combine two goal statement into one : Predict which cluster should each data point belongs to, given that the number of cluster is specified. We also rewrite the terminology part with clearer explanation and some new terminology added. We also modify of description on the mathematical formulation for our software , including assumptions, theoretical model, general definition, instance model, in order to reduce the ambiguity.

1.2 Design and Design Documentation

In response to our reviewer teams, the final implementation for GMM-EM has been revised. We first aviod the confusing notation by adding more description on the symbol we used. We also reorganize our modules for a greater simplicity. We deleted several modules: main module and plotting module, specifcation parameters module, renaming and adding some modules. The main and plotting function now are kept as a demonstration for our GMM-EM but not part of our software, since our software is a library. The detailed design change could refer to the later part.

1.3 VnV Plan and Report

First, we added all the details for the unit tests. In response to our reviewer team, the input and output of tests are specified in the revised document. The confusing language usage has been modified for a better readability. In our VnV report, we listed out all our system test and unit test result and also in our GitHub repository, these result can be built by CMake for any further reference.

2 Design Iteration (LO11)

The overall design and implementation tries to follows python scikit-learn Gaussian mixture models (sklearn.mixture) [1], with modification in several implementation and functions. This is the basic idea for the GMM-EM. However, from the first design of our module, we receive feedback from our reviewer teams, including professor, domain expert and secondary reviewer. In our original plan, the software is divided as follows:

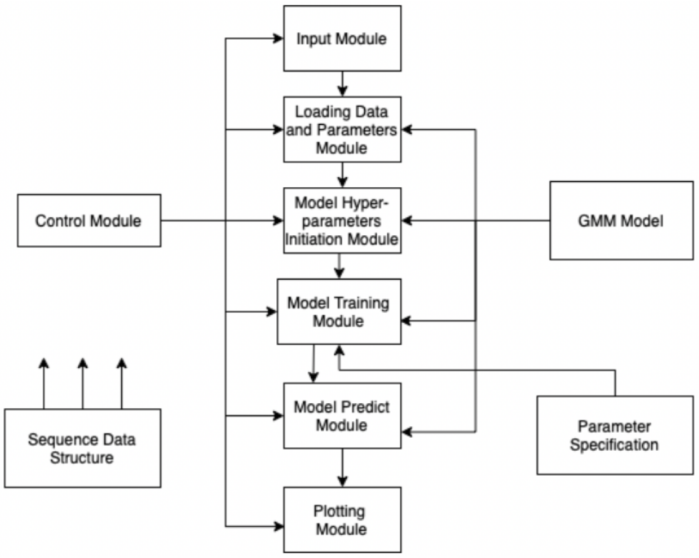

Figure 1: original design

However, we re-organize the module and change it as follows. The reasons behind is that the current module design could better implement the model. Unnecessary part is deleted. Repeated part are regroup into one module. For time limit, some extra module are deleted or simplified, but the function is kept.

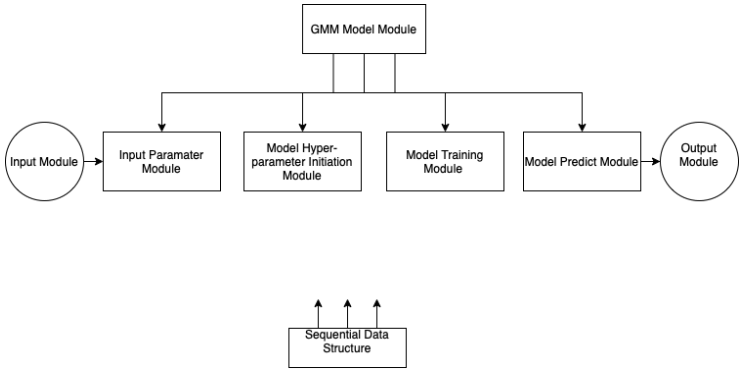

Figure 2: Updated design

- Input Module: The original input module is solely implemented by C++ library on input/output and file handling (iostream and fstream). However, this implementation could not adopt the data structure we used for the software. Therefore, we manually design a function to store the input with the iostream and fstream in C++. This function convert the input

into the data structure we used.

- Loading data and parameters module: Data is no longer stored in our model for better memory management. Therefore, this module only handle parameters for the module and rename into input parameters module. Also the function of parameter specification module is integrated into this new module and therefore the parameter specification module is deleted
- Output module: Created for saving the prediction into an external environment variables (csv file in our case)

3 Design Decisions (LO12)

Due to time limit, the design at the end aims for simplicity. In the design stage, we first think of to boost the performance, but this is hard to achieve at this stage. Therefore, we Reflect and justify your design decisions. How did limitations, assumptions, and constraints influence your decisions?

In our design,

4 Economic Considerations (LO23)

It will be a open source project. For current stage, this product is oversimplified and only limited functions are provided. However, the project is worth to further develop to fulfill the users. C++ machine learning library is not very common and there are only few projects on GMM is done by C++. Therefore, it is possible for us to attract user to develop or use the package.

5 Reflection on Project Management (LO24)

In our project, several tools has been used to facilitate the project building. The major tool we rely on is Git. Git provides a version control for our project and it work well on keeping our code and document traceable. The built-in tools in Visual Studio Code IDE also helps the project management. It keeps the folder in an organized and readable state. It also provides tools for CMake to auto build the project once update. The reviewer teams also greatly benefit our project

5.1 How Does Your Project Management Compare to Your Development Plan

Your Project Management is similar to our development plan with some modification due to time limit

5.2 What Went Well?

In overall, our project resolves most of the items suggested as the SRS plan or problem statement. In our VnV plan, we suggested for our test case. It shows same accuracy compared to the python package scikit-learn. It fulfills our requirement and expectation stated by SRS plan. The software also passed the test cases to ensure the basic function is usable.

5.3 What Went Wrong?

The automated test or intensive profiling are not done at this current stage for several reasons. The fundamental reason is time limit. Functions are basically implemented, but the performance is not good as expected (Refer to VnV report). Time is limited to develop a well-optimized code.

5.4 What Would you Do Differently Next Time?

Some future development could be considered.

- Performance: The code optimization should be considered to boost the performance for our library. Better memory management and code optimization skills should be used in the next development stage.
- Portability: The test for portability should be carried out as soon as possible to ensure the library could run in multi-platform.
- Functions: Further function should be developed. Compared to the python scikit-learn mixture model, many functions have not been implemented. We should develop these functions in our next stage.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.