# Module Interface Specification for GMM-EM

Kim Ying, WONG

March 19, 2024

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| 12 Mar 2024 | 1.0 | First Draft |
| 18 Mar 2024 | 1.1 | Second Draft |

# 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at Here

# Contents


**1 Revision History** **i**

**2 Symbols, Abbreviations and Acronyms** **ii**

**3 Introduction** **1**

**4 Notation** **1**

**5 Module Decomposition** **1**

**6 MIS of Input Module** **3**
- 6.1 Module . . . 3
- 6.2 Uses . . . 3
- 6.3 Syntax . . . 3
  - 6.3.1 Exported Constants . . . 3
  - 6.3.2 Exported Access Programs . . . 3
- 6.4 Semantics . . . 3
  - 6.4.1 State Variables . . . 3
  - 6.4.2 Environment Variables . . . 3
  - 6.4.3 Assumptions . . . 3
  - 6.4.4 Access Routine Semantics . . . 3
  - 6.4.5 Local Functions . . . 4

**7 MIS of Loading Input Data and Parameters Module** **4**
- 7.1 Module . . . 4
- 7.2 Uses . . . 4
- 7.3 Syntax . . . 4
  - 7.3.1 Exported Constants . . . 4
  - 7.3.2 Exported Access Programs . . . 4
- 7.4 Semantics . . . 4
  - 7.4.1 State Variables . . . 4
  - 7.4.2 Environment Variables . . . 4
  - 7.4.3 Access Routine Semantics . . . 5
  - 7.4.4 Local Functions . . . 5

**8 MIS of Model Hyper-parameters Initiation Module** **5**
- 8.1 Module . . . 5
- 8.2 Uses . . . 5
- 8.3 Syntax . . . 5
  - 8.3.1 Exported Constants . . . 5
  - 8.3.2 Exported Access Programs . . . 5
- 8.4 Semantics . . . 6

8.4.1 State Variables . . . 6
8.4.2 Environment Variables . . . 6
8.4.3 Access Routine Semantics . . . 6
8.4.4 Local Functions . . . 6

**9 MIS of Model Training Module** **6**
9.1 Module . . . 6
9.2 Uses . . . 6
9.3 Syntax . . . 7
9.3.1 Exported Constants . . . 7
9.3.2 Exported Access Programs . . . 7
9.4 Semantics . . . 7
9.4.1 State Variables . . . 7
9.4.2 Environment Variables . . . 7
9.4.3 Assumptions . . . 7
9.4.4 Access Routine Semantics . . . 7
9.4.5 Local Functions . . . 7

**10 MIS of Model Predict Module** **8**
10.1 Module . . . 8
10.2 Uses . . . 8
10.3 Syntax . . . 8
10.3.1 Exported Constants . . . 8
10.3.2 Exported Access Programs . . . 8
10.4 Semantics . . . 8
10.4.1 State Variables . . . 8
10.4.2 Environment Variables . . . 8
10.4.3 Access Routine Semantics . . . 8
10.4.4 Local Functions . . . 8

**11 MIS of Control Module** **9**
11.1 Module . . . 9
11.2 Uses . . . 9
11.3 Syntax . . . 9
11.3.1 Exported Constants . . . 9
11.3.2 Exported Access Programs . . . 9
11.4 Semantics . . . 9
11.4.1 State Variables . . . 9
11.4.2 Environment Variables . . . 9
11.4.3 Access Routine Semantics . . . 9

**12 MIS of Parameters Specification Module** **10**
12.1 Module . . . 10
12.2 Uses . . . 10
12.3 Syntax . . . 10
12.3.1 Exported Constants . . . 10
12.3.2 Exported Access Programs . . . 10
12.4 Semantics . . . 10
12.4.1 State Variables . . . 10
12.4.2 Environment Variables . . . 10
12.4.3 Access Routine Semantics . . . 11

**13 MIS of GMM Model Module** **11**
13.1 Module . . . 11
13.2 Uses . . . 11
13.3 Syntax . . . 11
13.3.1 Exported Constants . . . 11
13.3.2 Exported Access Programs . . . 11
13.4 Semantics . . . 11
13.4.1 State Variables . . . 11
13.4.2 Environment Variables . . . 11
13.4.3 Access Routine Semantics . . . 12

**14 MIS of Plotting Module** **12**
14.1 Module . . . 12
14.2 Uses . . . 12
14.3 Syntax . . . 12
14.3.1 Exported Constants . . . 12
14.3.2 Exported Access Programs . . . 12
14.4 Semantics . . . 12
14.4.1 State Variables . . . 12
14.4.2 Environment Variables . . . 12
14.4.3 Assumptions . . . 13
14.4.4 Access Routine Semantics . . . 13

# 3 Introduction

The following document details the Module Interface Specifications for Fill in your project name and description

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at . . . . provide the url for your repo

# 4 Notation

You should describe your notation. You can use what is below as a starting point.

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by .

| Data Type | Notation | Description |
|---|---|---|
| character | char | a single symbol or digit |
| integer | $\mathbb{Z}$ | a number without a fractional component in (-$\infty$, $\infty$) |
| natural number | $\mathbb{N}$ | a number without a fractional component in [1, $\infty$) |
| real | $\mathbb{R}$ | any number in (-$\infty$, $\infty$) |

The specification of uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

# 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

| **Level 1** | **Level 2** |
|---|---|
| Hardware-Hiding | |
| Behaviour-Hiding | Input Module<br>Loading Input Data and Parameters Module<br>Model Hyper-parameters Initiation Module<br>Model Updates Module<br>Model Predict Module<br>Control Module<br>Specification Parameters Module |
| Software Decision | Sequence Data Structure<br>GMM Model<br>Plotting |

Table 1: Module Hierarchy

# 6 MIS of Input Module

The secret of this module is how the csv file is read into the C++ program and store in the program. It is implemented by any C++ CSV library or a customized function. It serves as a bridge of the data stored in the Abstract class model and the C++ program.

## 6.1 Module

Input

## 6.2 Uses

none

## 6.3 Syntax

### 6.3.1 Exported Constants

none

### 6.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| read_csv | string | $\mathbf{X} \in \mathbf{R}^{M \times N}$ | FileError |

## 6.4 Semantics

### 6.4.1 State Variables

inputCSVFile: a sequence of string end with ".csv"

### 6.4.2 Environment Variables

CSV File

### 6.4.3 Assumptions

The file contains the all numeric values for each predictors in order, each on a new line.

### 6.4.4 Access Routine Semantics

read_csv(s):

- transition: none
- output: out := an ADT $\mathbf{X}$ that store dataset in csv with dimension $\mathbf{X} \in \mathbf{R}^{M \times N}$

- exception: exc := a file name s cannot be found OR the format of inputFile is incorrect ⇒ FileError

#### 6.4.5 Local Functions

none

# 7 MIS of Loading Input Data and Parameters Module

The secret of this module is the data structure to store the input parameters. How our software GMM-EM will store and verify the dataset and any other input paramters needed

## 7.1 Module

Loading params

## 7.2 Uses

Input (Section 6)

## 7.3 Syntax

### 7.3.1 Exported Constants

### 7.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| load_params | $\mathbf{X}_{old} \in \mathbf{R}^{M \times N}$<br>K $\in \mathbf{R}$ | $M(\mathbf{X}, K)$ | FormatError<br>ArgError<br>ValueError |

## 7.4 Semantics

### 7.4.1 State Variables

$M(\mathbf{X}, K)$

### 7.4.2 Environment Variables

none

#### 7.4.3 Access Routine Semantics

load_params(s):

- transition: The $\mathbf{X}$ will be converted and stored into the data structure used for our software (possibly Matrix class provided by C++ Eigen Library). Then the transformed $\mathbf{X}$ and $K$ will be store as a class member of our software GMM-EM
- output: out := The module will store the dataset and parameter(s) into the model. A raw model (without hyper-parameters initiated and trained) will be generated.
- exception:
  ArgError := Argument provided less than we needed
  FormatError:= The $\mathbf{X}$ could not converted to the data structure that we used for our software.
  ValueError:= Some values are out of range (Infinity or undefined number).

#### 7.4.4 Local Functions

none

# 8 MIS of Model Hyper-parameters Initiation Module

The secret of this module is the algorithm to initiate the hyper-parameters for the model and pass the hyper-parameters to update module.

## 8.1 Module

HyperInit

## 8.2 Uses

Loading params (Section 7)

## 8.3 Syntax

### 8.3.1 Exported Constants

### 8.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| Hyper_Init | $M(\mathbf{X}, K)$ | $M(\mathbf{X}, K, \mu, \Sigma, \pi)$ | – |

## 8.4 Semantics

### 8.4.1 State Variables

$M(\mathbf{X}, K)$

### 8.4.2 Environment Variables

none

### 8.4.3 Access Routine Semantics

Hyper_Int():

- transition: Hyper-parameters in the model will be generated. There are potentially 2 ways to generate it:
  1. Random initiation (generated from a random Gaussian distribution )
  2. K-mean algorithm
- output: out := A model with Hyper-parameters initiated.
- exception: none

### 8.4.4 Local Functions

none

# 9 MIS of Model Training Module

The secret of the module is the algorithm to train and update the model hyper-parameters.

## 9.1 Module

Train

## 9.2 Uses

HyperInit (Section 8) , SpecParams (Section 12)

## 9.3 Syntax

### 9.3.1 Exported Constants

### 9.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| model_fit | M($\mu_k$ , $\mathbf{\Sigma}_k$ , $\pi_k$ ,$\gamma$) | M($\mu_k^{new}$ , $\mathbf{\Sigma}_k^{new}$ , $\pi_k^{new}$ , $\gamma_{new}$) | OVERFLOW CONVERGE_FAILED |

## 9.4 Semantics

### 9.4.1 State Variables

$\mu_k$ , $\mathbf{\Sigma}_k$ , $\pi_k$ , $\gamma$

### 9.4.2 Environment Variables

none

### 9.4.3 Assumptions

none

### 9.4.4 Access Routine Semantics

fit():

- transition: Re-estimate the parameters followed until reaching maximum iteration or the hyper-parameter converge ($||\mu_k^{new} - \mu|| \leq tol$, or $\mu$ could be replaced by $\mathbf{\Sigma}, \pi$) :
  $\gamma(z_{nk}) = \pi_k N(x_n|\mu_k, \mathbf{\Sigma_k}) / \sum_{j=1}^{k} \pi_j N(x_n|\mu_j, \mathbf{\Sigma_j})$
  $\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$
  $\mathbf{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k^{new})(\mathbf{x}_n - \mu_k^{new})^T$
  $\pi_k^{new} = \frac{N_k}{N}$
- output: if appropriate
- exception:
  OVERFLOW := any elements of these $\mu_k$ , $\mathbf{\Sigma}_k$ , $\pi_k$ , $\gamma$ is out of range (Infinity).
  CONVERGE_FAILED := $||\mu_k^{new} - \mu|| > tol$ for iteration is larger than maximum iteration

### 9.4.5 Local Functions

none

# 10 MIS of Model Predict Module

The secret of this module is data structure and algorithm to output the model prediction.

## 10.1 Module

The secret of this module is to how to predict the clustering result and output to the user via file or terminal. Predict

## 10.2 Uses

Train (Section 9)

## 10.3 Syntax

### 10.3.1 Exported Constants

### 10.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| model_pred | | - | - |
| model_pred_csv | string | - | FileError |

## 10.4 Semantics

### 10.4.1 State Variables

predicted label array ($\in \mathbf{R}^M$) for the datapoints in $\mathbf{X}$

### 10.4.2 Environment Variables

file , terminal
none

### 10.4.3 Access Routine Semantics

model_pred(), model_pred_to_csv :

- transition : Write to environment variable the following: the predicted label, it will be an series of natural number seperated by comma if in model_pred_to_csv. It will be an output as [ , , , ] in an array format in terminal for model_pred.
- exception: model_pred_to_csv:= FileError, if a file name cannot be found.

### 10.4.4 Local Functions

pred($\gamma$) := f : $\mathrm{R}^{K\times M} \Rightarrow \mathrm{R}^M$

# 11 MIS of Control Module

The secret of the module is to illustrate the workflow for the software. main

## 11.1 Module

main

## 11.2 Uses

Input (Section 6), Loading Params (Section 7), HyperInit (Section 8), Train (Section 9), Predict (Section 10), Plot (Section 14)

## 11.3 Syntax

### 11.3.1 Exported Constants

### 11.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| main | - | - | - |

## 11.4 Semantics

### 11.4.1 State Variables

none

### 11.4.2 Environment Variables

none

### 11.4.3 Access Routine Semantics

main:

- transition: Modify the state of Input module and the environment variables for the Plot and Output modules by following these steps:

  Get (CSVfilenameIn: string) and (CSVfilenameOut: string) from user

  read_csv(CSVfilenameIn) to the program and input to the model by Loading_params. The model with dataset and input parameters will generate the hyper-parameter with HyperInit. The initiated model then will starting model training module until convergence. The result would be output by predict module and plotting module in the

environment variables.

# 12 MIS of Parameters Specification Module

The secret of the module is to specific some extra parameters defined in the model. main

## 12.1 Module

SpecParams

## 12.2 Uses

## 12.3 Syntax

### 12.3.1 Exported Constants

Max_Iter: Maximum iteration till the convergence happens = 100
tol : Convergence threshold = 1e-3

### 12.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| get_tol | - | tol $\in \mathbf{R}$ | - |
| get_iter | - | max_iter $\in \mathbf{N}$ | - |

## 12.4 Semantics

### 12.4.1 State Variables

none

### 12.4.2 Environment Variables

none

#### 12.4.3 Access Routine Semantics

get_tol:

- output: tol $\in \mathbf{R}$

get_iter:

- output: max_iter $\in \mathbf{N}$

# 13 MIS of GMM Model Module

The secret of the module is abstract class template for our GMM model.

## 13.1 Module

plot

## 13.2 Uses

## 13.3 Syntax

### 13.3.1 Exported Constants

### 13.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| GMM_Model | - | - | - |

## 13.4 Semantics

### 13.4.1 State Variables

none

### 13.4.2 Environment Variables

none

#### 13.4.3 Access Routine Semantics

GMM_Model():

- transition: Create a template class for GMM. Any model initiation, training and prediction could be cooperated and implemented with the template.
- output: none
- exception: none

# 14 MIS of Plotting Module

The secret of the module is how to visualize the clustering result in a graph.

## 14.1 Module

plot

## 14.2 Uses

## 14.3 Syntax

#### 14.3.1 Exported Constants

#### 14.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| plot | $\mathbf{X}(\mathbf{p})$, $z$ | - | - |

## 14.4 Semantics

#### 14.4.1 State Variables

none

#### 14.4.2 Environment Variables

win: 2D sequence of pixels displayed on the screen

#### 14.4.3 Assumptions

Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate.

#### 14.4.4 Access Routine Semantics

plot( $\mathbf{X}(\mathbf{p})$, $z$):

- transition: Modify win to display a plot where the vertical axis is first selected predictor and horizontal axis is second selected predictor with a color label from the predict module
- output: none
- exception: none

# References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.