Module Interface Specification for GMM-EM

Kim Ying, WONG

April 18, 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

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 (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

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 Loading Input Data and Parameters Module Model Hyper-parameters Initiation Module Model Updates Module Model Predict Module Output Module
Software Decision	Sequence Data Structure GMM Model

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
csv_to_data	string s	$\mathbf{X} \in \mathbb{R}^{M \times N}$	FileError TypeError EmptyFile

6.4 Semantics

6.4.1 State Variables

none

6.4.2 Environment Variables

CSV File

6.4.3 Assumptions

none

6.4.4 Access Routine Semantics

csv_to_data(s):

- transition: none
- output: out := an record $\mathbf{X}$ that store dataset in csv with dimension $\mathbf{X} \in \mathbf{R}^{M \times N}$, where N is number of datapoints, and M is number of predictor variables. The record is defined by C++ Eigen Library and standard vector container library.
- exception: exc := a file name s cannot be found OR the format of inputFile is incorrect $\Rightarrow$ FileError
- exception: exc := encounter any missing data or illegal values contains in the csv file when function reads the csv (for example: string , boolean) $\Rightarrow$ TypeError
- exception: exc := encounter a empty csv file (no datapoints in the file) $\Rightarrow$ EmptyFile

6.4.5 Local Functions

none

7 MIS of Input 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

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
GMM()	numComponents $K \in \mathbb{Z}^+$, max_iter $\in \mathbb{Z}^+$, tol $\in \mathbb{R}^+$, init_method $\mathbb{Z}^+$	M(numComponents , tol , init_method , max_iter)	–

7.4 Semantics

7.4.1 State Variables

numComponents: number of cluster needed ,
tol : error tolerance for convergence
init_method : method to initiate the hyper-parameters in model (0 = random , 1 = K-means),
max_iter = maximum itertaion for optimization process.
M: An abstract object or class represented the GMM-GM model created with constructor.

7.4.2 Environment Variables

none

7.4.3 Access Routine Semantics

GMM(numComponents, tol ,init_method max_iter):

- transition: The parameters will be stored in a abstract object GMM, the model M. The input variables will become the class member for the abstract object (or defined as class in C++) for the Model M.

- output: out := The module will generate a raw model (without hyper-parameters initiated and trained).
- exception: none (error will be handled by compiler)

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

Input params (Section 7), Input (Section 6))

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
initiate	**X**	M($\mu, \mathbf{\Sigma}, \pi, \gamma$)	EmptyDataset NumDatapointsError

8.4 Semantics

8.4.1 State Variables

An implemented class M via GMM Model Module and parameters defined from input parameter module

8.4.2 Environment Variables

none

8.4.3 Access Routine Semantics

initiate(**X**):

- 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:
  exc := if $\mathbf{X}$ is empty $\Rightarrow$EmptyDataset
  exc := if number of datapoints in $\mathbf{X}$ is greater or equals numComponents K defined in Input Parameters Module $\Rightarrow$ NumDatapointsError

8.4.4 Local Functions

Function to realize the K-mean algorithm
kmeans.initiate(data): $f : \mathbf{X} \Rightarrow \mu$
kmeans.fit($\mathbf{X}$) $f : \mathbf{X} \Rightarrow \mu, \pi$
kmeans.get_mean_vector() f: $\Rightarrow \mu$
kmeans.get_mixing_coefficients() $f : \Rightarrow \pi$

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 **??**)

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
model_fit(**X**)	**X**	M(μ_k^{new} , $\mathbf{\Sigma}_k^{new}$, π_k^{new} , γ_{new})	CONVERGE_FAILED
.mean_vector	-	μ_k^{new}	-
.cov_matrices	-	$\mathbf{\Sigma}_k^{new}$	-
.mixing_coeffieients	-	γ_k^{new} π_k^{new}	- -

9.4 Semantics

9.4.1 State Variables

mean vector for cluster K : μ_k
covariance matrix for cluster K : $\mathbf{\Sigma}_k$,
mixing coefficient for cluster K : π_k ,
responsibility: γ
Fitted (boolean) : denote the model is fitted or not

9.4.2 Environment Variables

none

9.4.3 Assumptions

none

9.4.4 Access Routine Semantics

fit($\mathbf{X}$):

- transition: Re-estimate the parameters followed until reaching maximum iteration or the hyper-parameter converge ($||\mu_k^{new} - \mu|| \leq tol$, or μ could be replaced by $\mathbf{\Sigma}, \pi$) :
  $\gamma(z_{nk}) = \pi_k N(x_n|\mu_k, \mathbf{\Sigma_k}) / \sum_{j=1}^{k} \pi_j N(x_n|\mu_j, \mathbf{\Sigma_j})$
  $\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$
  $\mathbf{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k^{new})(\mathbf{x}_n - \mu_k^{new})^T$
  $\pi_k^{new} = \frac{N_k}{N}$
  The boolean value Fitted will turn True after training.

- output: none

- exception:
  CONVERGE_FAILED :=
  $||\mu_k^{new} - \mu|| > tol$ for iteration is larger than maximum iteration , OR
  any elements of these μ_k , $\mathbf{\Sigma}_k$, π_k , γ is out of range (Infinity).

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	-	$(l \in \mathbf{Z}^{+N})$	NoFittedError

10.4 Semantics

10.4.1 State Variables

predicted label array $(l \in \mathbf{Z}^{+N})$ for the datapoints $\mathbf{x_n} \in \mathbf{X}$

10.4.2 Environment Variables

none

10.4.3 Access Routine Semantics

model_pred() :

- transition := f : $\mathbb{R}^{K \times N} \Rightarrow \mathbb{R}^{N}$ Convert the γ into the label l. For each row vector γ_N in $\gamma \in \mathbb{R}^{K \times N}$. Pick the largest γ_{NK} The l_n will equal K.
- exception: exc:= if the model is not fitted by Training Module $\Rightarrow$ NoFittedError.

10.4.4 Local Functions

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

11 MIS of Output Module

The secret of this module is to save the output to a csv file.

11.1 Module

The secret of this module is to save the labels from predict module into a csv file Predict

11.2 Uses

Predict (Section 10)

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
writeLabels-ToCSV	($l \in \mathbf{Z}^{+N}$, string s)	string s'	FileError FormatError

11.4 Semantics

11.4.1 State Variables

none

11.4.2 Environment Variables

string s that denote the location of the CSV file.

11.4.3 Access Routine Semantics

writeLabelstoCSV() :

- transition : Write to environment variable the following: the predicted label, it will be an series of natural number seperated by comma
- exception: exc := if a file name cannot be found ⇒ FileError.
  if file name doesn't contain ".csv" at the end of string ⇒ FormatError
- output: string "labels have saved to csv." to denote a successful function call.

11.4.4 Local Functions

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

12 MIS of GMM Model Module

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

12.1 Module

plot

12.2 Uses

12.3 Syntax

12.3.1 Exported Constants

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
GMM_Model	-	-	-

12.4 Semantics

12.4.1 State Variables

none

12.4.2 Environment Variables

none

12.4.3 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

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.