MixtComp Programmer Guide

Vincent KUBICKI

17th August 2015

Contents

Abstract

The MixtComp code is documented inline, but it might be difficult at times to get the big picture. Some sequences, like the initialization of the computations, might be complex and vary from one model to another. Some models use straight forward analytical estimator, while some use Markov chains that need to be initialized. The architecture must accomodate all kind of situations while keeping complexity to a bearable level when implementing a new model. Hence there are key steps in the initializations, and a new model must dispatch its various steps using those provided steps.

1 Environment variables

The following environment variables are used:

- MC_VERBOSE: change the level of details output to the standard output.
- MC_DEBUG, MC_DEBUG_NEW: some debug flags.
- MC_DETERMINISTIC: change the seeding value, from pseudo-random to 0. Used to run the unit test in a deterministic way.

2 Global algorithm

3 Sequence of initialization

There are several objects that are key to the initialization of MixtComp. Some of those objects are template argument to templated classes. For example, a MixtureManager object needs a DataHandler type argument. But the DataHandler is an interface to read and transform data, so there must be different DataHandler to cover the various inputs. The first DataHandler that has been written was DataHandlerR, which transforms the data structure obtained from R in a data structure suitable for MixtComp.

The MixtureManager reads the list of models and instantiate the corresponding C++ objects (from classes that all inherit from IMixture). It then register it to the MixtureComposer which will take care of it from there to the end of the run.

MixtureComposer::setDataParam resize various arrays in the MixtureComposer, and then calls various IMixture setDataparam, which use the DataHandler they got at construction time to get the correct data, to get the parameters from the ParamSetter in prediction mode, and to resize all important arrays.

	setDataParam	initData	
SimpleMixtureBridge	Nothing special.	AugmentedData::removeMissing	
Functional	Parse data and parameters compute quantiles.	Function::removeMissingQuantile	
Ordinal	setPath set the head and tail of each path	BOSPath::initPath	sampleMu
Rank	Parse and check data.	RankIndividual::removeMissing	mu initialized

Table 1: Operations performed in the various steps of initialization, for each model

Note that setDataParam is not supposed to initialize computations. Because MixtComp might have to restart a computation (if a degeneracy is detected for example), and setDataParam is only called once at the beginning. The initialization at the start of each computation is instead managed by MixtureComposer::InitParam, which is called in the Strategy, and not during the global initialization.

initData and initParam are both called at the beginning of SemStrategy::initSEMCheck and SemStrategy::initSEMNoCheck.

The operations performed by all variables are:

- getting the data from the DataHandler
- checking the data validity (its range for example)
- initializing various storages, including the storages for the statistics

The specific operations needed by the various models are described in the table 1.

4 Questions

- Why is there a Gibbs in initParam for Ordinal, but not for Rank ? Because there are possible 0 probability completed individuals for Ordinal, but not for Rank ?