Mixmod plugin

An OpenTURNS module

N. Balin
1 What are MIXMOD and modules in OT?

2 Steps to create the new module ot-mixmod (for non C++ specialists)

3 Difficulties encountered

4 How to use ot-mixmod OT module?
1 What are MIXMOD and modules in OT?

2 Steps to create the new module ot-mixmod (for non C++ specialists)

3 Difficulties encountered

4 How to use ot-mixmod OT module?
MIXMOD software

- High Performance Model-Based Cluster and Discriminant Analysis software
- n-D data classification
- n-D data multimodal parametric reconstruction
- Many algorithms (with parameters) available
- http://www-math.univ-fcomte.fr/mixmod/

Components

- Free software (license GPL)
- Written in C++
- API available
- Already interfaced with Scilab and Matlab
Mixtures definition

- Probability density function

\[ f(x) = \sum_i \alpha_i p_i(x) \quad \sum_i \alpha_i = 1 \quad \text{with } 0 \leq \alpha_i \leq 1 \]

Construction of a Mixture from a Numerical Sample in OT ?

- **KernelMixture**
  - distribution built from a Numerical Sample,
  - linear combination of a kernel specified by the user

\[ f(x) = \sum_i \frac{1}{nh} K \left( \frac{X_i - x}{h} \right) \]

- a non-parametric reconstruction of the distribution
MIXMOD is a powerful tool for research in statistical pattern recognition.

Useful to be able to work in OT with a mixture distribution composed of Normal distributions (parametric reconstruction)

Example of use: to get a mixture distribution of modelling data coming from heterogeneous population (for instance).
OpenTURNS modules

- Simplified access to the development of OpenTURNS
- Faster development process (compilation, testing, ..)
- For new functionalities
- Callable as a library or from python interpreter

- MIXMOD will be interfaced through a module with OT
- Only a restricted interface between OT and MIXMOD
Content of the MIXMOD-OT interface

**What is already interfaced in version 0.1**

- n-D data multimodal parametric reconstruction with gaussian distributions
- default reconstruction algorithm

**What is in the development version**

- choice of the covariance model (among the models available in MIXMOD)
  - same covariance,
  - different marginal variances but same correlation,
  - ...

**What is expected in the longer term**

- Choice of the reconstruction algorithm
Outline

1. What are MIXMOD and modules in OT?

2. Steps to create the new module ot-mixmod (for non C++ specialists)

3. Difficulties encountered

4. How to use ot-mixmod OT module?
Access point for MIXMOD in OpenTURNS

- MIXMOD takes as input a sample and gives a multinomial mixture model
- tool to construct a distribution.
- must be a child of DistributionImplementationFactory
  - not in KernelMixture
  - the result will be a Distribution built from a Mixture, with a collection of Distribution.
Dependency graph (for modules)

- libOT.so
- otmixmod.so
- otmixmod.py
- opentums.py
Implementation 1

What do we have to implement?

- the constructors and accessors to the private parameters,
- as it is a child of DistributionImplementationFactory, methods that must be redefined are buildImplementation methods with the following parameters:
  - a NumericalSample
  - a NumericalPointCollection
  - no parameter (the clone method)

```cpp
DistributionImplementation * buildImplementation(const NumericalSample & sample) const
   /* throw(InvalidArgumentException, InternalException) */;

DistributionImplementation * buildImplementation(const NumericalPointCollection & parameters) const
   /* throw(InvalidArgumentException, InternalException) */;

DistributionImplementation * buildImplementation() const
   /* throw(InvalidArgumentException, InternalException) */;
```
**MIXMOD parameters**

- number of atoms `atomsNumber_`, required, implemented
- algorithm parameters, optional, as a default algorithm is set
  - not yet implemented
  - a new class will be developed for it

---

**Constructors and accessors to manage MIXMOD parameters**

- **constructors:**
  ```
  MixtureFactory();
  MixtureFactory(const UnsignedLong atomsNumber);
  ```

- **accessors:**
  ```
  void setAtomsNumber(const UnsignedLong & number);
  UnsignedLong getAtomsNumber() const;
  ```
buildImplementation ... from a NumericalSample

1. Transform NumericalSample into a format compatible with MIXMOD

   1. temporary file ← sample (with sample.storeToTemporaryFile())

MIXMOD call

- C++ API for MIXMOD,
- license incompatibilities and restricted API
- ⇒ MIXMOD executable and communication through temporary files
buildImplementation ... from a NumericalSample

1. Transform NumericalSample into a format compatible with MIXMOD
2. Call MIXMOD algorithm
   1: create temporary directory
      (Path::CreateTemporaryDirectory("mixmod"))
   2: write MIXMOD parameters file in the temporary directory from the factory parameters (atoms number and algorithm)
   3: call MIXMOD executable
   4: get MIXMOD return value
Implementation 3

buildImplementation ... from a NumericalSample

1. Transform NumericalSample into a format compatible with MIXMOD
2. Call MIXMOD algorithm
3. Transform MIXMOD results into an OpenTURNS Distribution

get output file name
initialize the new distribution collection \textit{coll}

\begin{verbatim}
for \( i = 0 \) to \text{atomsNumber}_- \ do
    read \( w, \text{mean}(\text{dim}) \) and \( \text{cov}(\text{dim}, \text{dim}) \)
    \( \text{atom} \leftarrow \text{Normal} (\text{mean}, \text{cov}) \)
    \( \text{atom}.\text{setWeight} (\text{w}) \)
    \( \text{coll}.\text{add}(\text{atom}) \)
\end{verbatim}

end \textit{for}
close the result file and remove the temporary directory

\textbf{return} \textit{coll}
Autotools

- m4 macro to detect MIXMOD executable with the autotools

Wrapping python

- it’s necessary to adapt the existing files from existing modules ...
Validation: brief

- tested the C++ library and python module
- OpenTURNS-MIXMOD results are equivalent to standalone MIXMOD results
- if we create a NumericalSample from a composed distribution, we find the input parameters
- computed PDF is correct
- Necessary to use getParametersCollection to get the mixture’s atoms.

Documentation

- brief: README to describe the installation method and to give an example of use
1. What are MIXMOD and modules in OT?

2. Steps to create the new module ot-mixmod (for non C++ specialists)

3. Difficulties encountered

4. How to use ot-mixmod OT module?
Some remarks

- Configuration method to improve if several installation openturns
- Useful to get an intelligent IDE (with completion)

Missing elements

- Documentation: no template for a LaTeX documentation ...
- Autotests: No example of autotests as for the OpenTURNS library ..
- Mechanism to print the version number of the module? (as the OT welcome message)
Outline

1. What are MIXMOD and modules in OT?
2. Steps to create the new module ot-mixmod (for non C++ specialists)
3. Difficulties encountered
4. How to use ot-mixmod OT module?
Module installation

Installation

- to install the module in $OPENTURNS_MODULE_PATH
  
  openturns-module --install=otmixmod-0.1.0.tar.bz2 \\ 
  --prefix=$OPENTURNS_MODULE_PATH \\ 
  MIXMOD_BIN=${MIXMOD_INSTALL_DIR}/BIN/mixmod

- the openturns-module OT can be used to remove modules, ...

Environment variables

- Only one version of OpenTURNS on the system:
  OPENTURNS_MODULE_PATH (and PYTHONPATH if needed)

- If more than one version of OpenTURNS is present on the system, it could be necessary to complete the PYTHONPATH variable:
  
  export PYTHONPATH=$OT_INSTALL_DIR/lib/python2.5/site-packages: 
  $OT_INSTALL_DIR/lib/python2.5/site-packages/openturns
from openturns import *
from otmixmod import *

...

factory = MixtureFactory(3)
estimatedDistribution = factory.buildImplementation(sample)
...
Conclusion

How to get this great module?

- on http://share.openturns.org
- on the OpenTURNS module repository: http://trac.openturns.org/browser/openturns-modules
  - you can find here some other interesting modules ...
How to get this great module?

- on http://share.openturns.org
- on the OpenTURNS module repository: http://trac.openturns.org/browser/openturns-modules
  - you can find here some other interesting modules...

Relatively easy to develop a new module for OpenTURNS

- even for non C++ specialists
Conclusion

How to get this great module?

- on http://share.openturns.org
- on the OpenTURNS module repository: http://trac.openturns.org/browser/openturns-modules
  - you can find here some other interesting modules ...

- Relatively easy to develop a new module for OpenTURNS
- even for non C++ specialists

Questions?