MeGaMix Documentation

Release 0.2

Elina Thibeau-Sutre

Mar 29, 2021

Contents

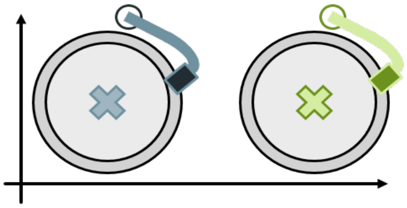

Table of contents

CHAPTER 1

Getting started

1.1 Installation

The package is registered on PyPI. It can be installed with the following command:

```
$ pip install megamix
```

If you want to install it manually, you can find the source code at https://github.com/14thibea/megamix.

MeGaMix relies on external dependencies. The setup script should install them automatically, but you may want to install them manually. The required packages are:

- NumPy 1.11.3 or newer
- scipy 0.18.1 or newer
- h5py 2.6.0 or newer
- joblib 0.11 or newer
- cython

Note: If you cannot compile the package, please dowload it and manually comment the line **ext_modules=cythonize(ext_modules),** in setup.py Doing so you will not compile the cython modules and only use pure python versions.

1.2 Description

The MeGaMix package (Methods for Gaussian Mixtures) allows Python developpers to fit different kind of models on their data. The different models are clustering methods of unsupervised machine learning. Four models have been implemented, from the most simple to the most complex:

- K-means

- GMM (Gaussian Mixture Model)
- VBGMM (Variational Bayesian Gaussian Mixture Model)
- DP-VBGMM (Dirichlet Process on Variational Bayesian Gaussian Mixture Model)
- PYP-VBGMM (Pitman-Yor Process on Variational Bayesian Gaussian Mixture Model)

These models are *batch* algorithms: they use the whole set of data during the computation. Some of these algorithms (K-means and GMM) have been implemented in an *online* way: in this way the model is adapted as the user bring new data to the program.

1.2.1 What will you be able to do ?

The main idea of clustering algorithms is to create groups by gathering points that are close to each other.

A cluster has three main parameters:

- A mean : the mean of all the points that belong to the cluster
- A weight : the number of points that belong to the cluster
- A covariance (except for K-means) : a matrix which specifies the form of the cluster

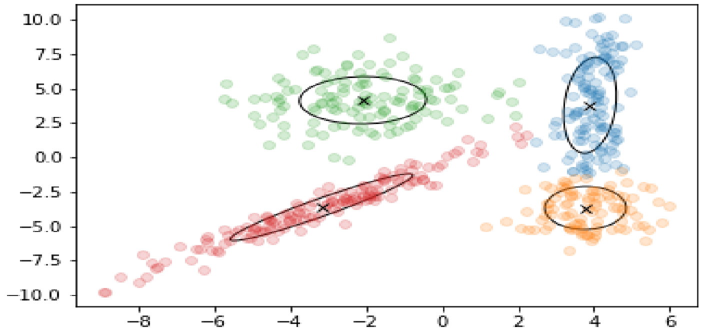

Fig. 1: A graphical example of a gaussian mixture model fit on a set of points

1.2.2 How do the algorithms work ?

After the initialisation, the algorithms alternate between two steps, the E step (Expectation) and the M step (Maximisation).

During the **E step**, the algorithm computes the probability for each point to belong to each cluster. It produces an array of *responsibilities*. At the ith row and the jth column of this array corresponds the probability of the ith point to belong to the jth cluster.

Here is an example of responsibilities that could be obtained with 6 points and 2 clusters :

	Cluster 1	Cluster 2
point 1	0.54	0.46
point 2	0.89	0.11
point 3	0.27	0.73
point 4	0.01	0.99
point 5	0.42	0.58
point 6	0.84	0.16

In this example, the first point has a 54% chance to belong to the first cluster and a 46% chance to belong to the second cluster.

Note: This is not the case with K-means which is not working with probabilities but with labels. A point belongs completely to a cluster or doesn't belong to it (this is called hard assignement).

Then during the **M step**, the algorithm re-estimates the parameters of the model in order to maximize a convergence criterion.

Finally the algorithm will stop if the difference between the value of the convergence criterion of the current and the previous is less than a threshold fixed by the user.

This is summarized in the following pseudo-code:

```
initialize(points)
while(cc-cc_previous > tol):
    cc_previous = cc
    responsabilities = E_step(points,parameters)
    parameters = M_step(responsabilities,points)
    cc = convergence_criterion(points,responsabilities,parameters)
```

1.2.3 What is it used for ?

MeGaMix has been implemented in order to process natural speech MFCC. Unlike the vision field where deep learning has overtaken such clustering models, they are still efficient in speech processing.

However the use of this package is more general and may serve another purpose.

1.3 Basic usage

```
########################
# Prelude to the example
########################
"""
This example is realized with a DP-VBGMM model
The other mixtures and the K-means are working in the same way
The available classes are:
   - Kmeans (kmeans)
   - GaussianMixture (GMM)
   - VariationalGaussianMixture (VBGMM)
   - DPVariationalGaussianMixture (DP-VBGMM)
"""

```

(continues on next page)

(continued from previous page)

```
from megamix.batch import DPVariationalGaussianMixture
import numpy as np

########################
# Features used
########################

"""
Features must be numpy arrays of two dimensions:
the first dimension is the number of points
the second dimension is the dimension of the space
"""

# Here we use a radom set of points for the example
n_points = 10000
dim = 39

points = np.random.randn(n_points,dim)

########################
# Fitting the model
########################

# We choose the number of clusters that we want
n_components = 100

# The model is instantiated
GM = DPVariationalGaussianMixture(n_components)

# The model is fitting
GM.fit(points)

# It is also possible to do early stopping in order to avoid overfitting
points_data = points[:n_points//2:]
points_test = points[n_points//2::]

# In this case the model will fit only on points_data but will use points_test
# to evaluate the convergence criterion.
GM.fit(points_data,points_test)

# Some clusters may disappear with the DP-VBGMM model. You may want to
# simplify the model by removing the useless information
GM_simple = GM.simplified_model(points)

##########################
# Analysis of the model
##########################

other_points = np.random.randn(n_points,dim)

# We can obtain the log of the reponsibilities of any set of points when the
# model is fitted (or at least initialized)
log_resp = GM.predict_log_resp(other_points)
# log_resp.shape = (n_points,n_components)

# We can obtain the value of the convergence criterion for any set of points
score = GM.score(other_points)
```

(continues on next page)

(continued from previous page)

```
#############################
# Writing or reading a model
#############################

# It is possible to write your model in a group of a h5py file
import h5py

file = h5py.File('DP_VBGMM.h5','w')
grp = file.create_group('model_fitted')

GM.write(grp)
file.close()

# You also can read data from such h5py file to initialize new models
GM_new = DPVariationalGaussianMixture()

file = h5py.File('DP_VBGMM.h5','r')
grp = file['model_fitted']

GM_new.read_and_init(grp,points)
file.close()

# You can also save regurlarly your code while fitting the model by using
# the saving parameter

GM.fit(points,saving='log',directory='mypath',legend='wonderful_model')
```

CHAPTER 2

API Reference

Two versions of the EM algorithms exist [*batch* and *online* :]

- The batch version takes all the point at the same time, using more CPU and memory but may lead to more accurate results.
- The online version takes the points w by w. In this way the program uses less CPU and memory, but there may be a loss of accuracy.

2.1 Batch versions of the algorithm

Four different algorithms have been developped in batch: K-means, GMM, VBGMM and DPGMM.

2.1.1 Kmeans

2.1.2 Gaussian Mixture Model (GMM)

2.1.3 Variational Gaussian Mixture Model (VBGMM)

2.1.4 Dirichlet Process Gaussian Mixture Model (DPGMM)

2.2 Online versions of the algorithm

Only two algorithms have been developped in batch: K-means and GMM.

2.2.1 Kmeans

2.2.2 Gaussian Mixture Model (GMM)

CHAPTER 3

Theory of Gaussian Mixture models

In this part are detailed the equations used in each algorithm. We use the same notations as Bishop's *Pattern Recognition and Machine Learning*.

Features:

- $\{x_1, x_2, ..., x_N\}$ is the set of points

Parameters:

- μ_k is the center of the k^{th} cluster
- π_k is the weight of the k^{th} cluster
- Σ_k is the covariance matrix of the k^{th} cluster
- K is the number of clusters
- N is the number of points
- d is the dimension of the problem

Other notations specific to the methods will be introduced later.

3.1 K-means

An iteration of K-means includes:

- The *E step* : a label is assigned to each point (hard assignement) arcording to the means.
- The *M step* : means are computed according to the parameters.
- The computation of the *convergence criterion* : the algorithm uses the distortion as described below.

3.1.1 E step

The algorithm produces a matrix of responsibilities according to the following equation:

$$r_{nk} = \begin{cases} 1 \text{ if } k = \arg \min\limits_{1 \leq j \leq k} \|x_n - \mu_j\|^2 \\ 0 \text{ otherwise} \end{cases}$$

The value of the case at the i^{th} row and j^{th} column is 1 if the i^{th} point belongs to the j^{th} cluster and 0 otherwise.

3.1.2 M step

The mean of a cluster is simply the mean of all the points belonging to this latter:

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}}$$

The weight of the cluster k, which is the number of points belonging to this latter, can be expressed as:

$$N_k = \sum_{n=1}^{N} r_{nk}$$

The mixing coefficients, which represent the proportion of points in a cluster, can be expressed as:

$$\pi_k = \frac{N_k}{N}$$

3.1.3 Convergence criterion

The convergence criterion is the distortion defined as the sum of the norms of the difference between each point and the mean of the cluster it is belonging to:

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

The distortion should only decrease during the execution of the algorithm. The model stops when the difference between the value of the convergence criterion at the previous iteration and the current iteration is less or equal to a threshold tol :

$$D_{previous} - D_{current} \leq tol$$

3.2 Gaussian Mixture Model (GMM)

An iteration of GMM includes:

- The *E step* : K probabilities of belonging to each cluster are assigned to each point
- The *M step* : weights, means and covariances are computed according to the parameters.
- The computation of the *convergence criterion* : the algorithm uses the loglikelihood as described below.

3.2.1 E step

The algorithm produces a matrix of responsibilities according to the following equation:

$$r_{nk} = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)}$$

The value of the case at the i^{th} row and j^{th} column is the probability that the point i belongs to the cluster j.

3.2.2 M step

The weight of the cluster k, which is the number of points belonging to this latter, can be expressed as:

$$N_k = \sum_{n=1}^{N} r_{nk}$$

The mixing coefficients, which represent the proportion of points in a cluster, can be expressed as:

$$\pi_k = \frac{N_k}{N}$$

As in the Kmeans algorithm, the mean of a cluster is the mean of all the points belonging to this latter:

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{N_k}$$

The covariance of the cluster k can be expressed as:

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk}(x_n - \mu_k)(x_n - \mu_k)^T$$

These results have been obtained by derivating the maximum loglikelihood described in the following section.

3.2.3 Convergence criterion

The convergence criterion used in the Gaussian Mixture Model algorithm is the maximum log likelihood:

$$\sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)$$

Setting its derivatives to 0 gives the empirical terms described in the M step.

3.3 Variational Gaussian Mixture Model (VBGMM)

In this model, we introduce three new hyperparameters and two distributions which governs the three essential parameters of the model: the mixing coefficients, the means and the covariances.

The mixing coefficients are generated with a Dirichlet Distribution:

$$q(\pi_k) = \text{Dir}(\pi|\alpha_k) = \text{C}(\alpha_k)\pi_k^{\alpha_k - 1}$$

The computation of α_k is described in the M step.

Then we introduce an independant Gaussian-Wishart law governing the mean and precision of each gaussian component:

$$\begin{aligned} q(\mu_k, \Sigma_k) &= q(\mu_k|\Sigma_k)q(\Sigma_k) \\ &= \mathcal{N}(\mu_k|m_k, (\beta_k\Sigma_k)^{-1})\mathcal{W}(\Gamma_k|W_k, \nu_k) \end{aligned}$$

The computation of the terms involved in this equation are described in the M step.

The aim of VBGMM is to reduce the variability of the model by introducing a bias: prior values on the parameters of the model.

3.3.1 E step

It is not possible to compute directly r_{nk}, another quantity ρ_{nk} is calculated instead and r_{nk} is obtained after normalization.

$$\ln\rho_{nk} = \mathbb{E}[\ln\pi_k] + \frac{1}{2}\mathbb{E}[\ln\det\Lambda_k] - \frac{d}{2}\ln 2\pi - \frac{1}{2}\mathbb{E}_{\mu_k,\Lambda_k}[(x_n-\mu_k)^T\Lambda_k(x_n-\mu_k)]$$

$$\ln\rho_{nk} = \psi(\alpha_k) - \psi(\sum_{i=1}^{K}\alpha_i) + \frac{1}{2}\sum_{i=1}^{d}\psi(\frac{\nu_k+1-i}{2}) + \frac{1}{2}\ln\det W_k$$

$$-\frac{d}{2}\ln\pi - \frac{d}{2\beta_k} - \frac{\nu_k}{2}(x_n-m_k)^T W_k(x_n-m_k)$$

3.3.2 M step

3.3.3 Convergence criterion

3.4 Dirichlet Process Gaussian Mixture Model (DPGMM)

3.4.1 E step

3.4.2 M step

3.4.3 Convergence criterion

3.5 Pitman-Yor Process Gaussian Mixture Model (PYPGMM)