MeGaMix Documentation

Release 0.2

Elina Thibeau-Sutre

Aug 28, 2017

Contents

Table of contents

CHAPTER 1

Getting started

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

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)

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

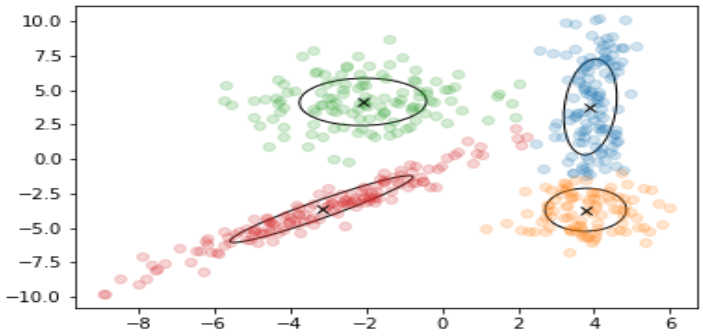

Fig. 1.1: A graphical example of a gaussian mixture model fit on a set of points

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)
```

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.

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)
"""

from megamix 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)

############################
# 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

Batch versions of the algorithm

Kmeans

class megamix.batch.kmeans.**Kmeans**(*n_components=1*, *init='plus'*, *n_jobs=1*)

Kmeans model.

Parameters

- **n_components** (*int, defaults to 1.*) – Number of clusters used.
- **init** (*str, defaults to 'kmeans'.*) – Method used in order to perform the initialization, must be in ['random', 'plus', 'AF_KMC'].

name

str – The name of the method : 'Kmeans'

means

array of floats (n_components,dim) – Contains the computed means of the model.

log_weights

array of floats (n_components,) – Contains the logarithm of the mixing coefficient of each cluster.

iter

int – The number of iterations computed with the method fit()

_is_initialized

bool – Ensures that the model has been initialized before using other methods such as distortion() or predict_assignements().

Raises ValueError : if the parameters are inconsistent, for example if the cluster number is negative, init_type is not in ['resp','mcw']...

References

'Fast and Provably Good Seedings for k-Means', O. Bachem, M. Lucic, S. Hassani, A.Krause 'Lloyd's algorithm <https://en.wikipedia.org/wiki/Lloyd's_algorithm>'_ 'The remarkable k-means++ <https://normaldeviate.wordpress.com/2012/09/30/the-remarkable-k-means/>'_

fit(*points_data*, *points_test=None*, *n_iter_max=100*, *n_iter_fix=None*, *tol=0*, *saving=None*, *file_name='model'*, *saving_iter=2*)
The k-means algorithm

Parameters

- **points_data** (*array (n_points,dim)*) – A 2D array of points on which the model will be trained
- **tol** (*float, defaults to 0*) – The EM algorithm will stop when the difference between two steps regarding the distortion is less or equal to tol.
- **n_iter_max** (*int, defaults to 100*) – number of iterations maximum that can be done
- **saving_iter** (*int | defaults 2*) – An int to know how often the model is saved (see saving below).
- **file_name** (*str | defaults model*) – The name of the file (including the path).

Other Parameters

- **points_test** (*array (n_points_bis,dim)* | *Optional*) – A 2D array of points on which the model will be tested.
- **n_iter_fix** (*int* | *Optional*) – If not None, the algorithm will exactly do the number of iterations of n_iter_fix and stop.
- **saving** (*str* | *Optional*) – A string in ['log','linear']. In the following equations x is the parameter saving_iter (see above).
  - **If 'log', the model will be saved for all iterations which verify :** log(iter)/log(x) is an int
  - **If 'linear' the model will be saved for all iterations which verify :** iter/x is an int

Returns

Return type None

predict_assignements(*points*)
This function return the hard assignements of points once the model is fitted.

score(*points*, *assignements=None*)
This method returns the distortion measurement at the end of the k_means.

Parameters

- **points**(*an array (n_points,dim)*) –
- **assignements** (*an array (n_components,dim)*) – an array containing the responsibilities of the clusters

Returns distortion

Return type (float)

megamix.batch.kmeans.**dist_matrix**(*points*, *means*)

Gaussian Mixture Model (GMM)

class megamix.batch.**GaussianMixture**(*n_components=1*, *covariance_type='full'*, *init='kmeans'*, *reg_covar=1e-06*, *type_init='resp'*, *n_jobs=1*)

Gaussian Mixture Model

Representation of a Gaussian mixture model probability distribution. This class allows to estimate the parameters of a Gaussian mixture distribution.

Parameters

- **n_components** (*int, defaults to 1.*) – Number of clusters used.
- **init** (*str, defaults to 'kmeans'.*) – Method used in order to perform the initialization, must be in ['random', 'plus', 'AF_KMC', 'kmeans'].
- **reg_covar** (*float, defaults to 1e-6*) – In order to avoid null covariances this float is added to the diagonal of covariance matrices.
- **type_init** (*str, defaults to 'resp'.*) – The algorithm is initialized using this data (responsibilities if 'resp' or means, covariances and weights if 'mcw').

name
str – The name of the method : 'GMM'

cov
array of floats (n_components,dim,dim) – Contains the computed covariance matrices of the mixture.

means
array of floats (n_components,dim) – Contains the computed means of the mixture.

log_weights
array of floats (n_components,) – Contains the logarithm of the mixing coefficient of each cluster.

iter
int – The number of iterations computed with the method fit()

convergence_criterion_data
array of floats (iter,) – Stores the value of the convergence criterion computed with data on which the model is fitted.

convergence_criterion_test
array of floats (iter,) | *if _early_stopping only* – Stores the value of the convergence criterion computed with test data if it exists.

_is_initialized
bool – Ensures that the method _initialize() has been used before using other methods such as score() or predict_log_assignements().

Raises ValueError : if the parameters are inconsistent, for example if the cluster number is negative, init_type is not in ['resp','mcw']...

References

'Pattern Recognition and Machine Learning', Bishop

fit(*points_data*, *points_test=None*, *tol=0.001*, *patience=None*, *n_iter_max=100*, *n_iter_fix=None*, *saving=None*, *file_name='model'*, *saving_iter=2*)
The EM algorithm

Parameters

- **points_data** (*array (n_points,dim)*) – A 2D array of points on which the model will be trained
- **tol** (*float, defaults to 1e-3*) – The EM algorithm will stop when the difference between two steps regarding the convergence criterion is less than tol.
- **n_iter_max** (*int, defaults to 100*) – number of iterations maximum that can be done
- **saving_iter** (*int | defaults 2*) – An int to know how often the model is saved (see saving below).
- **file_name** (*str | defaults model*) – The name of the file (including the path).

Other Parameters

- **points_test** (*array (n_points_bis,dim) | Optional*) – A 2D array of points on which the model will be tested.
- **patience** (*int | Optional*) – The number of iterations performed after having satisfied the convergence criterion
- **n_iter_fix** (*int | Optional*) – If not None, the algorithm will exactly do the number of iterations of n_iter_fix and stop.
- **saving** (*str | Optional*) – A string in ['log','linear']. In the following equations x is the parameter saving_iter (see above). * If 'log', the model will be saved for all iterations which verify :

  log(iter)/log(x) is an int

  – **If 'linear' the model will be saved for all iterations which verify :** iter/x is an int

Returns

Return type None

predict_log_resp(*points*)

This function returns the logarithm of each point's responsibilities

Parameters **points** (*array (n_points_bis,dim)*) – a 1D or 2D array of points with the same dimension as the problem

Returns **log_resp** – the logarithm of the responsibilities

Return type array (n_points_bis,n_components)

read_and_init(*group*, *points*)

A method reading a group of an hdf5 file to initialize DPGMM

Parameters **group** (*HDF5 group*) – A group of a hdf5 file in reading mode

score(*points*)

This function return the score of the function, which is the logarithm of the likelihood for GMM and the logarithm of the lower bound of the likelihood for VBGMM and DPGMM

Parameters **points** (*array (n_points_bis,dim)*) – a 1D or 2D array of points with the same dimension as the problem

Returns **score**

Return type float

simplified_model(*points*)
A method creating a new model with simplified parameters: clusters unused are removed

Parameters **points**(*an array (n_points,dim)*) –

Returns GM

Return type an instance of the same type of self: GMM,*VBGMM* or *DPGMM*

write(*group*)
A method creating datasets in a group of an hdf5 file in order to save the model

Parameters **group**(*HDF5 group*) – A group of a hdf5 file in reading mode

Variational Gaussian Mixture Model (VBGMM)

class megamix.batch.VBGMM.**VariationalGaussianMixture**(*n_components=1*, *init='kmeans'*, *alpha_0=None*, *beta_0=None*, *nu_0=None*, *means_prior=None*, *cov_wishart_prior=None*, *reg_covar=1e-06*, *type_init='resp'*, *n_jobs=1*)
Variational Bayesian Estimation of a Gaussian Mixture

This class allows to infer an approximate posterior distribution over the parameters of a Gaussian mixture distribution.

The weights distribution is a Dirichlet distribution with parameter alpha (see Bishop's book p474-486)

Parameters

- **n_components**(*int, defaults to 1.*) – Number of clusters used.
- **init**(*str, defaults to 'kmeans'.*) – Method used in order to perform the initialization, must be in ['random', 'plus', 'AF_KMC', 'kmeans', 'GMM'].
- **reg_covar**(*float, defaults to 1e-6*) – In order to avoid null covariances this float is added to the diagonal of covariance matrices.
- **type_init**(*str, defaults to 'resp'.*) – The algorithm is initialized using this data (responsibilities if 'resp' or means, covariances and weights if 'mcw').

Other Parameters

- **alpha_0** (*float, Optional | defaults to None.*) – The prior parameter on the weight distribution (Dirichlet). A high value of alpha_0 will lead to equal weights, while a low value will allow some clusters to shrink and disappear. Must be greater than 0.

  If None, the value is set to 1/n_components
- **beta_0** (*float, Optional | defaults to None.*) – The precision prior on the mean distribution (Gaussian). Must be greater than 0.

  If None, the value is set to 1.0
- **nu_0** (*float, Optional | defaults to None.*) – The prior of the number of degrees of freedom on the covariance distributions (Wishart). Must be greater or equal to dim.

  If None, the value is set to dim
- **means_prior** (*array (dim,), Optional | defaults to None*) – The prior value to compute the value of the means.

  If None, the value is set to the mean of points_data

- **cov_wishart_prior** (*type depends on covariance_type, Optional | defaults to None*) – If covariance_type is 'full' type must be array (dim,dim) If covariance_type is 'spherical' type must be float The prior value to compute the value of the precisions.

  If None, the value is set to the covariance of points_data

name
str – The name of the method : 'VBGMM'

alpha
array of floats (n_components,) – Contains the parameters of the weight distribution (Dirichlet)

beta
array of floats (n_components,) – Contains coefficients which are multipied with the precision matrices to form the precision matrix on the Gaussian distribution of the means.

nu
array of floats (n_components,) – Contains the number of degrees of freedom on the distribution of covariance matrices.

_inv_prec
array of floats (n_components,dim,dim) – Contains the equivalent of the matrix W described in Bishop's book. It is proportional to cov.

_log_det_inv_prec
array of floats (n_components,) – Contains the logarithm of the determinant of W matrices.

cov
array of floats (n_components,dim,dim) – Contains the computed covariance matrices of the mixture.

means
array of floats (n_components,dim) – Contains the computed means of the mixture.

log_weights
array of floats (n_components,) – Contains the logarithm of weights of each cluster.

iter
int – The number of iterations computed with the method fit()

convergence_criterion_data
array of floats (iter,) – Stores the value of the convergence criterion computed with data on which the model is fitted.

convergence_criterion_test
array of floats (iter,) | if _early_stopping only – Stores the value of the convergence criterion computed with test data if it exists.

_is_initialized
bool – Ensures that the method _initialize() has been used before using other methods such as score() or predict_log_assignements().

Raises ValueError : if the parameters are inconsistent, for example if the cluster number is negative, init_type is not in ['resp','mcw']...

References

'Pattern Recognition and Machine Learning', Bishop

fit(*points_data*, *points_test=None*, *tol=0.001*, *patience=None*, *n_iter_max=100*, *n_iter_fix=None*, *saving=None*, *file_name='model'*, *saving_iter=2*)
The EM algorithm

Parameters

- **points_data** (*array (n_points,dim)*) – A 2D array of points on which the model will be trained
- **tol** (*float, defaults to 1e-3*) – The EM algorithm will stop when the difference between two steps regarding the convergence criterion is less than tol.
- **n_iter_max** (*int, defaults to 100*) – number of iterations maximum that can be done
- **saving_iter** (*int | defaults 2*) – An int to know how often the model is saved (see saving below).
- **file_name** (*str | defaults model*) – The name of the file (including the path).

Other Parameters

- **points_test** (*array (n_points_bis,dim) | Optional*) – A 2D array of points on which the model will be tested.
- **patience** (*int | Optional*) – The number of iterations performed after having satisfied the convergence criterion
- **n_iter_fix** (*int | Optional*) – If not None, the algorithm will exactly do the number of iterations of n_iter_fix and stop.
- **saving** (*str | Optional*) – A string in ['log','linear']. In the following equations x is the parameter saving_iter (see above). * If 'log', the model will be saved for all iterations which verify :

  log(iter)/log(x) is an int

  – **If 'linear' the model will be saved for all iterations which verify :** iter/x is an int

Returns

Return type None

predict_log_resp(*points*)

This function returns the logarithm of each point's responsibilities

Parameters **points** (*array (n_points_bis,dim)*) – a 1D or 2D array of points with the same dimension as the problem

Returns **log_resp** – the logarithm of the responsibilities

Return type array (n_points_bis,n_components)

read_and_init(*group*, *points*)

A method reading a group of an hdf5 file to initialize DPGMM

Parameters **group** (*HDF5 group*) – A group of a hdf5 file in reading mode

score(*points*)

This function return the score of the function, which is the logarithm of the likelihood for GMM and the logarithm of the lower bound of the likelihood for VBGMM and DPGMM

Parameters **points** (*array (n_points_bis,dim)*) – a 1D or 2D array of points with the same dimension as the problem

Returns **score**

Return type float

simplified_model(*points*)
A method creating a new model with simplified parameters: clusters unused are removed

Parameters **points** (*an array (n_points,dim)*) –

Returns GM

Return type an instance of the same type of self: GMM,*VBGMM* or *DPGMM*

write(*group*)
A method creating datasets in a group of an hdf5 file in order to save the model

Parameters **group** (*HDF5 group*) – A group of a hdf5 file in reading mode

Dirichlet Process Gaussian Mixture Model (DPGMM)

class megamix.batch.DPGMM.**DPVariationalGaussianMixture**(*n_components=1*, *init='kmeans'*, *alpha_0=None*, *beta_0=None*, *nu_0=None*, *means_prior=None*, *cov_wishart_prior=None*, *reg_covar=1e-06*, *type_init='resp'*, *n_jobs=1*, *pypcoeff=0*)
Variational Bayesian Estimation of a Gaussian Mixture with Dirichlet Process

This class allows to infer an approximate posterior distribution over the parameters of a Gaussian mixture distribution.

The weights distribution follows a Dirichlet Process with attribute alpha.

Parameters

- **n_components** (*int, defaults to 1.*) – Number of clusters used.
- **init** (*str, defaults to 'kmeans'.*) – Method used in order to perform the initialization, must be in ['random', 'plus', 'AF_KMC', 'kmeans', 'GMM', 'VBGMM'].
- **reg_covar** (*float, defaults to 1e-6*) – In order to avoid null covariances this float is added to the diagonal of covariance matrices.
- **type_init** (*str, defaults to 'resp'.*) – The algorithm is initialized using this data (responsibilities if 'resp' or means, covariances and weights if 'mcw').

Other Parameters

- **alpha_0** (*float, Optional | defaults to None.*) – The prior parameter on the weight distribution (Beta). A high value of alpha_0 will lead to equal weights, while a low value will allow some clusters to shrink and disappear. Must be greater than 0.

  If None, the value is set to 1/n_components
- **beta_0** (*float, Optional | defaults to None.*) – The precision prior on the mean distribution (Gaussian). Must be greater than 0.

  If None, the value is set to 1.0
- **nu_0** (*float, Optional | defaults to None.*) – The prior of the number of degrees of freedom on the covariance distributions (Wishart). Must be greater or equal to dim.

  If None, the value is set to dim

- **means_prior** (*array (dim,), Optional | defaults to None*) – The prior value to compute the value of the means.

  If None, the value is set to the mean of points_data

- **cov_wishart_prior** (*type depends on covariance_type, Optional | defaults to None*) – If covariance_type is 'full' type must be array (dim,dim) If covariance_type is 'spherical' type must be float The prior value to compute the value of the precisions.

- **pypcoeff** (*float | defaults to 0*) – If 0 the weights are generated according to a Dirichlet Process If >0 and <=1 the weights are generated according to a Pitman-Yor Process.

name
str – The name of the method : 'VBGMM'

alpha
array of floats (n_components,2) – Contains the parameters of the weight distribution (Beta)

beta
array of floats (n_components,) – Contains coefficients which are multipied with the precision matrices to form the precision matrix on the Gaussian distribution of the means.

nu
array of floats (n_components,) – Contains the number of degrees of freedom on the distribution of covariance matrices.

_inv_prec
array of floats (n_components,dim,dim) – Contains the equivalent of the matrix W described in Bishop's book. It is proportional to cov.

_log_det_inv_prec
array of floats (n_components,) – Contains the logarithm of the determinant of W matrices.

cov
array of floats (n_components,dim,dim) – Contains the computed covariance matrices of the mixture.

means
array of floats (n_components,dim) – Contains the computed means of the mixture.

log_weights
array of floats (n_components,) – Contains the logarithm of weights of each cluster.

iter
int – The number of iterations computed with the method fit()

convergence_criterion_data
array of floats (iter,) – Stores the value of the convergence criterion computed with data on which the model is fitted.

convergence_criterion_test
array of floats (iter,) | if _early_stopping only – Stores the value of the convergence criterion computed with test data if it exists.

_is_initialized
bool – Ensures that the method _initialize() has been used before using other methods such as score() or predict_log_assignements().

Raises ValueError : if the parameters are inconsistent, for example if the cluster number is negative, init_type is not in ['resp','mcw']...

References

'Variational Inference for Dirichlet Process Mixtures', D. Blei and M. Jordan

fit(*points_data*, *points_test=None*, *tol=0.001*, *patience=None*, *n_iter_max=100*, *n_iter_fix=None*, *saving=None*, *file_name='model'*, *saving_iter=2*)
The EM algorithm

Parameters

- **points_data** (*array (n_points,dim)*) – A 2D array of points on which the model will be trained
- **tol** (*float, defaults to 1e-3*) – The EM algorithm will stop when the difference between two steps regarding the convergence criterion is less than tol.
- **n_iter_max** (*int, defaults to 100*) – number of iterations maximum that can be done
- **saving_iter** (*int | defaults 2*) – An int to know how often the model is saved (see saving below).
- **file_name** (*str | defaults model*) – The name of the file (including the path).

Other Parameters

- **points_test** (*array (n_points_bis,dim)* | *Optional*) – A 2D array of points on which the model will be tested.
- **patience** (*int* | *Optional*) – The number of iterations performed after having satisfied the convergence criterion
- **n_iter_fix** (*int* | *Optional*) – If not None, the algorithm will exactly do the number of iterations of n_iter_fix and stop.
- **saving** (*str* | *Optional*) – A string in ['log','linear']. In the following equations x is the parameter saving_iter (see above). * If 'log', the model will be saved for all iterations which verify :

  log(iter)/log(x) is an int

  – **If 'linear' the model will be saved for all iterations which verify :** iter/x is an int

Returns

Return type None

predict_log_resp(*points*)
This function returns the logarithm of each point's responsibilities

Parameters **points** (*array (n_points_bis,dim)*) – a 1D or 2D array of points with the same dimension as the problem

Returns **log_resp** – the logarithm of the responsibilities

Return type array (n_points_bis,n_components)

read_and_init(*group*, *points*)
A method reading a group of an hdf5 file to initialize DPGMM

Parameters **group** (*HDF5 group*) – A group of a hdf5 file in reading mode

score (*points*)
This function return the score of the function, which is the logarithm of the likelihood for GMM and the logarithm of the lower bound of the likelihood for VBGMM and DPGMM

Parameters **points** (*array (n_points_bis,dim)*) – a 1D or 2D array of points with the same dimension as the problem

Returns score

Return type float

simplified_model (*points*)
A method creating a new model with simplified parameters: clusters unused are removed

Parameters **points** (*an array (n_points,dim)*) –

Returns GM

Return type an instance of the same type of self: GMM,*VBGMM* or *DPGMM*

write (*group*)
A method creating datasets in a group of an hdf5 file in order to save the model

Parameters **group** (*HDF5 group*) – A group of a hdf5 file in reading mode

Online versions of the algorithm

Kmeans

class megamix.online.kmeans.**Kmeans** (*n_components=1*, *window=1*, *kappa=1.0*)
Kmeans model.

Parameters

- **n_components** (*int, defaults to 1.*) – Number of clusters used.
- **window** (*int, defaults to 1*) – The number of points used at the same time in order to update the parameters.
- **kappa** (*double, defaults to 1.0*) – A coefficient in]0.0,1.0] which give weight or not to the new points compared to the ones already used.
  - If kappa is nearly null, the new points have a big weight and the model may take a lot of time to stabilize.
  - If kappa = 1.0, the new points won't have a lot of weight and the model may not move enough from its initialization.

name
str – The name of the method : 'Kmeans'

log_weights
array of floats (n_components) – Contains the logarithm of the mixing coefficients of the model.

means
array of floats (n_components,dim) – Contains the computed means of the model.

N
array of floats (n_components,) – The sufficient statistic updated during each iteration used to compute log_weights (this corresponds to the mixing coefficients).

x
array of floats (n_components,dim) – The sufficient statistic updated during each iteration used to compute the means.

iter
int – The number of points which have been used to compute the model.

_is_initialized
bool – Ensures that the model has been initialized before using other methods such as fit(), distortion() or predict_assignements().

Raises ValueError : if the parameters are inconsistent, for example if the cluster number is negative, init_type is not in ['resp','mcw']...

References

Online but Accurate Inference for Latent Variable Models with Local Gibbs Sampling, C. Dupuy & F. Bach
'The remarkable k-means++ <https://normaldeviate.wordpress.com/2012/09/30/the-remarkable-k-means/>'_

fit(*points*, *saving=None*, *file_name='model'*, *saving_iter=2*)
The k-means algorithm

Parameters

- **points** (*array (n_points,dim)*) – A 2D array of points on which the model will be trained.
- **saving_iter** (*int | defaults 2*) – An int to know how often the model is saved (see saving below).
- **file_name** (*str | defaults model*) – The name of the file (including the path).

Other Parameters **saving** (*str | Optional*) – A string in ['log','linear']. In the following equations x is the parameter saving_iter (see above).

- **If 'log', the model will be saved for all iterations which verify :** log(iter)/log(x) is an int
- **If 'linear' the model will be saved for all iterations which verify :** iter/x is an int

Returns

Return type None

get(*name*)

initialize(*points*)
This method initializes the Gaussian Mixture by setting the values of the means, covariances and weights.

Parameters

- **points_data** (*an array (n_points,dim)*) – Data on which the model is fitted.
- **points_test** (*an array (n_points,dim) | Optional*) – Data used to do early stopping (avoid overfitting)

predict_assignements(*points*)
This function return the hard assignements of points once the model is fitted.

score(*points*, *assignements=None*)
This method returns the distortion measurement at the end of the k_means.

Parameters

- **points**(*an array (n_points,dim)*)–
- **assignements** (*an array (n_components,dim)*) – an array containing the responsibilities of the clusters

Returns **distortion**

Return type (float)

megamix.online.kmeans.**dist_matrix**(*points*, *means*)

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.

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.

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.

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 can be expressed as:

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

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 :

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

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.

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.

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.

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.

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.

E step

M step

Convergence criterion

Dirichlet Process Gaussian Mixture Model (DPGMM)

E step

M step

Convergence criterion

Pitman-Yor Process Gaussian Mixture Model (PYPGMM)

Python Module Index

m

Index

Symbols

A

B

C

D

F

G

I