t-Student-Mixture-Models Documentation

Release 0.1.4

Luis Carlos Garcia Peraza Herrera

Jun 08, 2021

Contents

- This module allows you to model data by a mixture of t-Student distributions, estimating the parameters with Expectation-Maximisation.
- Implementation of the paper: 'Robust mixture modelling using the t distribution', D. Peel and G. J. McLachlan.
- This code has been produced during my PhD at University College London under the supervision of Prof. Sebastien Ourselin and Dr. Tom Vercauteren.
- Compatible with Python 2.7 and Python 3.

Contents:

CHAPTER 1

t-Student-Mixture-Models

Implementation of the paper: 'Robust mixture modelling using the t-distribution', D. Peel and G. J. McLachlan. Compatible with Python 2.7 and Python 3.

CHAPTER 2

Dependencies

- scikit-learn v0.18.1
- numpy v1.11.0
- scipy v0.19.0
- setuptools v36.0.1

CHAPTER 3

Install

Using pip (no need to clone this repo):

```
pip install smm --user
```

Manually:

```
git clone https://github.com/luiscarlosgph/t-Student-Mixture-Models.git
cd t-Student-Mixture-Models
python setup.py build
python setup.py install --user
```

CHAPTER 4

Usage

See example in src/smm/example.py.

```
python src/smm/example.py
```

CHAPTER 5

Tests

To run the tests execute:

```
python setup.py test
```

CHAPTER 6

Coverage

Current coverage: 79%. To re-run the coverage test (Ubuntu Ubuntu 16.04.2 LTS):

```
python-coverage run ./setup.py test
python-coverage html
```

Then open 'htmlcov/index.html' and check the line 'src/smm/smm'.

CHAPTER 7

Author

Luis Carlos Garcia-Peraza Herrera (luis.herrera.14@ucl.ac.uk).

CHAPTER 8

License

BSD 3-Clause License, see LICENSE file for more information.

CHAPTER 9

src

9.1 smm package

9.1.1 Submodules

9.1.2 smm.example module

9.1.3 smm.smm module

9.1.4 Module contents

t-Student Mixture Models Module (smm).

- This module allows you to model data by a mixture of t-Student distributions, estimating the parameters with Expectation-Maximisation. It is an implementation of the paper: 'Robust mixture modelling using the t distribution', D. Peel and G. J. McLachlan. Published at: Statistics and Computing (2000) 10, 339-348.
- This module has reused code and comments from sklearn.mixture.gmm.

class smm.**SMM**(*n_components=1*, *covariance_type='full'*, *random_state=None*, *tol=1e-06*, *min_covar=1e-06*, *n_iter=1000*, *n_init=1*, *params='wmcd'*, *init_params='wmcd'*)

Bases: sklearn.base.BaseEstimator

t-Student Mixture Model SMM class.

Representation of a t-Student mixture model probability distribution. This class allows for easy evaluation of, sampling from, and maximum-likelihood estimation of the parameters of an SMM distribution.

Initializes parameters such that every mixture component has zero mean and identity covariance.

Parameters

- **n_components** (*int, optional.*) – Number of mixture components. Defaults to 1.
- **covariance_type** (*string, optional.*) – String describing the type of covariance parameters to use. Must be one of 'spherical', 'tied', 'diag', 'full'. Defaults to 'full'.

- **random_state** (*RandomState or an int seed.*) – A random number generator instance. None by default.
- **tol** (*float, optional.*) – Convergence threshold. EM iterations will stop when average gain in log-likelihood is below this threshold. Defaults to 1e-6.
- **min_covar** (*float, optional.*) – Floor on the diagonal of the covariance matrix to prevent overfitting. Defaults to 1e-6.
- **n_iter** (*int, optional.*) – Number of EM iterations to perform. Defaults to 1000.
- **n_init** (*int, optional.*) –

  Number of initializations to perform. The best result is kept.

  Defaults to 1.
- **params** (*string, optional.*) – Controls which parameters are updated in the training process. Can contain any combination of 'w' for weights, 'm' for means, 'c' for covars, and 'd' for the degrees of freedom. Defaults to 'wmcd'.
- **init_params** (*string, optional.*) – Controls which parameters are updated in the initialization process. Can contain any combination of 'w' for weights, 'm' for means, 'c' for covars, and 'd' for the degrees of freedom. Defaults to 'wmcd'.

Variables

- ***weights*** (array, shape (*n_components*,).) – This attribute stores the mixing weights for each mixture component.
- ***means*** (array_like, shape (*n_components*, *n_features*).) – Mean parameters for each mixture component.
- **covars** (*array_like.*) – Covariance parameters for each mixture component. The shape depends on *covariance_type*:

  (n_components, n_features) if 'spherical', (n_features, n_features) if 'tied', (n_components, n_features) if 'diag', (n_components, n_features, n_features) if 'full'
- **converged** (*bool.*) – True when convergence was reached in fit(), False otherwise.

aic(*X*)

Akaike information criterion for the current model fit and the proposed data.

Parameters **X** (*array_like, shape (n_samples, n_features).*)

Returns

Return type A float (the lower the better)

bic(*X*)

Bayesian information criterion for the current model fit and the proposed data.

Parameters **X** (*array_like, shape (n_samples, n_features).*)

Returns

Return type A float (the lower the better)

covariances

Covariance parameters for each mixture component.

Returns

- *The covariance matrices for all the classes.*

- *The shape depends on the type of covariance matrix* – (n_classes, n_features) if 'diag', (n_classes, n_features, n_features) if 'full' (n_classes, n_features) if 'spherical', (n_features, n_features) if 'tied',

degrees
Returns the degrees of freedom of each component in the mixture.

static dist_covar_to_match_cov_type(*tied_cv*, *covariance_type*, *n_components*)
Create all the covariance matrices from a given template.

Parameters

- **tied_cv** (*array_like, shape (n_features, n_features).*) – Tied covariance that is going to be converted to other type.
- **covariance_type** (*string.*) – Type of the covariance parameters. Must be one of 'spherical', 'tied', 'diag', 'full'.
- **n_components** (*integer value.*) – Number of components in the mixture.

Returns **cv** – parameter). Tied covariance in the format specified by the user.

Return type array_like, shape (depends on the covariance_type

fit(*X*, *y=None*)
Estimate model parameters with the EM algorithm.

A initialization step is performed before entering the em algorithm. If you want to avoid this step, set the keyword argument init_params to the empty string '' when creating the SMM object. Likewise, if you would like just to do an initialization, set n_iter=0.

Parameters

- **X** (*array_like, shape (n_samples, n_features).*)
- **y** (*not used, just for compatibility with sklearn API.*)

means
Returns the means of each component in the mixture.

static multivariate_t_rvs(*m*, *S*, *df=inf*, *n=1*)
Generate multivariate random variable sample from a t-Student distribution.

Original code by Enzo Michelangeli. Modified by Luis C. Garcia-Peraza Herrera. This static method is exclusively used by 'tests/smm_test.py'.

Parameters

- **m** (*array_like, shape (n_features,).*) – Mean vector, its length determines the dimension of the random variable.
- **S** (*array_like, shape (n_features, n_features).*) – Covariance matrix.
- **df** (*int or float.*) – Degrees of freedom.
- **n** (*int.*) – Number of observations.

Returns **rvs** – Each row is an independent draw of a multivariate t distributed random variable.

Return type array_like, shape (n, len(m))

predict(*X*)
Predict label for data.

This function will tell you which component of the mixture most likely generated the sample.

Parameters **X** (*array-like, shape (n_samples, n_features).*)

Returns **r_argmax**

Return type array_like, shape (n_samples,)

predict_proba(*X*)
Predict label for data.

This function will tell the probability of each component generating each sample.

Parameters **X** (*array-like, shape (n_samples, n_features).*)

Returns **responsibilities**

Return type array_like, shape (n_samples, n_components)

score(*X*, *y=None*)
Compute the log probability under the model.

Parameters **X** (*array_like, shape (n_samples, n_features).*)

Returns **prob** – Probabilities of each data point in X.

Return type array_like, shape (n_samples,)

score_samples(*X*)
Per-sample likelihood of the data under the model.

Compute the probability of X under the model and return the posterior distribution (responsibilities) of each mixture component for each element of X.

Parameters **X** (*array_like, shape (n_samples, n_features).*)

Returns

- **prob** (*array_like, shape (n_samples,).*) – Unnormalised probability of each data point in X, i.e. likelihoods.
- **responsibilities** (*array_like, shape (n_samples,*) – n_components). Posterior probabilities of each mixture component for each observation.

weights
Returns the weights of each component in the mixture.

exception smm.**dofMaximizationError**(*message*)
Bases: exceptions.ValueError

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index