TPGMM

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

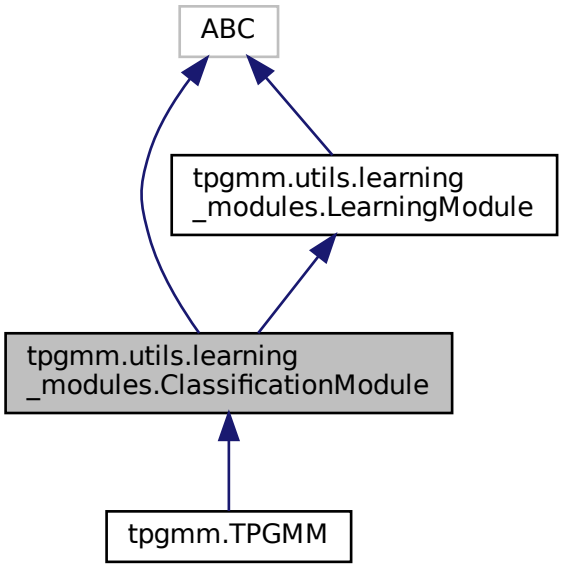
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

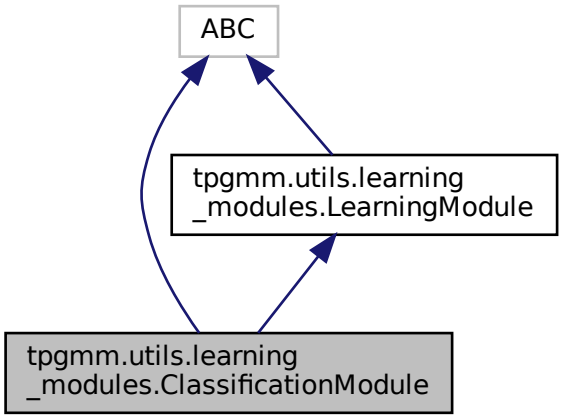
Class Documentation

3.1 tpgmm.utils.learning_modules.ClassificationModule Class Reference

Inheritance diagram for tpgmm.utils.learning_modules.ClassificationModule:



Collaboration diagram for tpgmm.utils.learning_modules.ClassificationModule:



Public Member Functions

- None **__init__** (self, int n_components)
- def predict_proba (self, ∗args, ∗∗kwargs)

  Predict class probabilities for the input data.
- float score (self, ndarray X)

  Calculate the score function from the descendant.
- float silhouette_score (self, ndarray X)

  Calculate the silhouette score for the given data.
- float inertia (self, ndarray X)

  Calculate the sum of squared distances of samples to their closest cluster center.
- float davies_bouldin_score (self, ndarray X)

  calculates davies_bouldin_score on given data
- float bic (self, ndarray X)

  calculates the bayesian information criterion as in
- def aic (self, ndarray X)

  calculates the Akaike information criterion as in https://scikit-learn.org/stable/modules/linear↩ _model.html#aic-bic
- Dict[str, Any] config (self)

  Get the configuration parameters.

3.1.1 Member Function Documentation

3.1.1.1 aic()

```
def tpgmm.utils.learning_modules.ClassificationModule.aic (
              self,
             ndarray X )
```

calculates the Akaike information criterion as in https://scikit-learn.org/stable/modules/linear_model.html#aic-bic

Parameters

X	data tensor with expected shape same form self.score(X)

Returns

float aic score

3.1.1.2 bic()

```
float tpgmm.utils.learning_modules.ClassificationModule.bic (
              self,
             ndarray X )
```

calculates the bayesian information criterion as in

```
https://scikit-learn.org/stable/modules/linear_model.html#aic-bic
```

Parameters

X	data tensor with expected shape (num_points, num_features)

Returns

float bic score

Reimplemented in tpgmm.TPGMM.

3.1.1.3 davies_bouldin_score()

```
float tpgmm.utils.learning_modules.ClassificationModule.davies_bouldin_score (
              self,
             ndarray X )
```

calculates davies_bouldin_score on given data

$$DB = \frac{1}{n} \sum_{i,j=1}^{N} \max_{j!=i} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

Where

N ... number of datapoints d() ... distance function c_i ... cluster center c_i sigma_i ... the average distance of all points in cluster i from the cluster centre ci

Parameters

X	data in the shape for self.predict()

Returns

float score value

Reimplemented in tpgmm.TPGMM.

3.1.1.4 inertia()

```
float tpgmm.utils.learning_modules.ClassificationModule.inertia (
              self,
             ndarray X )
```

Calculate the sum of squared distances of samples to their closest cluster center.

Parameters

X	data in local reference frames. Shape (num_frames, num_points, num_features)

Returns

float intertia

Reimplemented in tpgmm.TPGMM.

3.1.1.5 score()

```
float tpgmm.utils.learning_modules.ClassificationModule.score (
              self,
             ndarray X )
```

Calculate the score function from the descendant.

```
   Often the score is calculated based on the optimization objective.
```

Parameters

X	Data to calculate the score on.

Returns

float The calculated score.

Reimplemented in tpgmm.TPGMM.

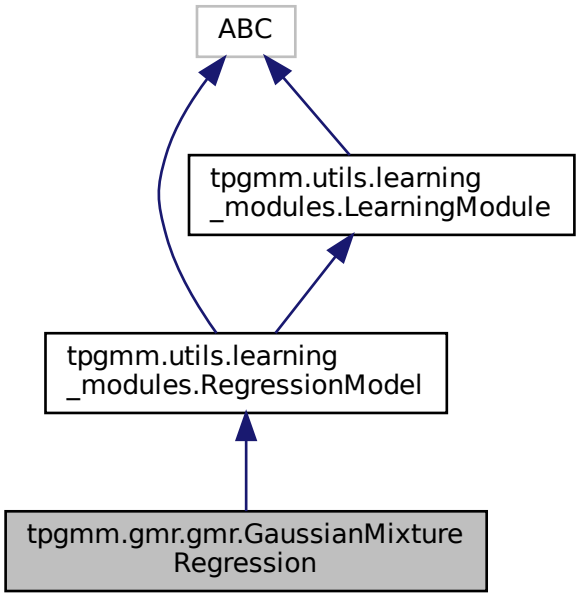
The documentation for this class was generated from the following file:

- documentation/tmp/tpgmm/utils/learning_modules.py

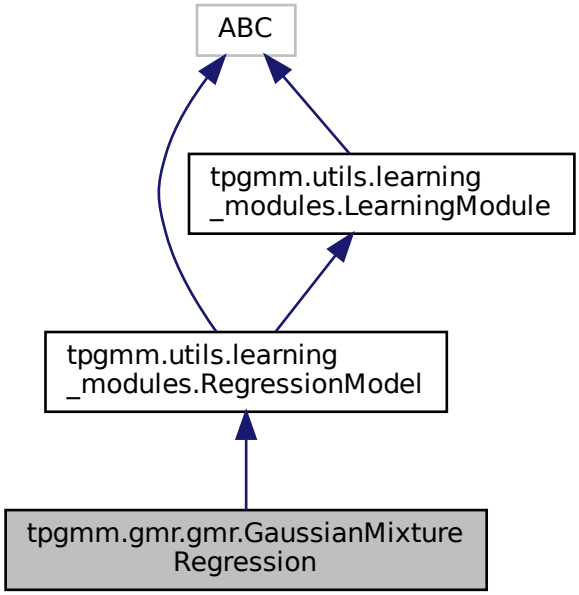
3.2 tpgmm.gmr.gmr.GaussianMixtureRegression Class Reference

This class implements a gaussian mixture regression model.

Inheritance diagram for tpgmm.gmr.gmr.GaussianMixtureRegression:



Collaboration diagram for tpgmm.gmr.gmr.GaussianMixtureRegression:



Public Member Functions

- None **__init__** (self, ndarray weights, ndarray means, ndarray covariances, Iterable[int] input_idx)
- "GaussianMixtureRegression" **from_tpgmm** ("GaussianMixtureRegression" cls, TPGMM tpgmm, Iterable[int] input_idx)
- None fit (self, ndarray translation, ndarray rotation_matrix)

  Turns the task_parameterized gaussian mixture model into a single gaussian mixture model.
- Tuple[ndarray, ndarray] predict (self, ndarray input_data)

  this function is inspired by formula 13 in Calinon paper it creates for each given data point its own parameterized gaussian distribution.
- int num_input_features (self)

  get number of input features.
- int num_output_features (self)

  get number of output features.
- List[int] **output_idx** (self)
- Dict[str, Any] config (self)

  Get all model config parameters.

Public Attributes

- **tpgmm_means_**
- **tpgmm_covariances_**
- **gmm_weights**
- **input_idx**
- **xi_**
- **sigma_**

3.2.1 Detailed Description

This class implements a gaussian mixture regression model.

This model was described in 'A Tutorial on Task-Parameterized Movement Learning and Retrival' from S.Calinon at: https://calinon.ch/papers/Calinon-JIST2015.pdf

The class fits a gaussian mixture regression on a given Gaussian Mixture model or a Task-Parameterized gaussian mixture model. Used equations are:

$$\mathcal{P}(\phi_t^{\mathcal{O}}|\phi_t^{\mathcal{I}}) \sim \sum_{i=1}^{K} h_i(\phi_t^{\mathcal{I}}) \mathcal{N}\left(\hat{\mu}_t^{\mathcal{O}}(\phi_t^{\mathcal{I}}), \hat{\Sigma}_t^{\mathcal{O}}\right)$$

$$\hat{\mu}_i^{\mathcal{O}}(\phi_t^{\mathcal{I}}) = \mu_i^{\mathcal{O}} + \Sigma_i^{\mathcal{O}\mathcal{I}} \Sigma_i^{\mathcal{I},-1} (\phi_t^{\mathcal{I}} - \mu_i^{\mathcal{I}})$$

$$\hat{\Sigma}_t^{\mathcal{O}} = \Sigma_i^{\mathcal{O}} - \Sigma_i^{\mathcal{O}\mathcal{I}} \Sigma_i^{\mathcal{I},-1} \Sigma_i^{\mathcal{O}\mathcal{I}}$$

$$h_i(\phi_t^{\mathcal{I}}) = \frac{\pi_i \mathcal{N}(\phi_t^{\mathcal{I}} \mid \mu_i^{\mathcal{I}}, \Sigma_i^{\mathcal{I}})}{\sum_k^K \pi_k \mathcal{N}(\phi_t^{\mathcal{I}} \mid \mu_k^{\mathcal{I}}, \Sigma_k^{\mathcal{I}})}$$

Examples

```
>> # trajectory data in local reference frames with shape: (num_reference_frames, num_points, 4)
>> # last dimension could be for example: x, y, z, time
>> # in case you got multiple trajectories: concatenate all of them in axis 1
>> trajectories = load_trajectories()
>> tpgmm = TPGMM(n_components=5)
>> tpgmm.fit(trajectories)
>> # gmr for the first reference frame
>> gmr = GaussianMixtureRegression(weights=tpgmm.weights_, means=tpgmm.means_[0],
       covariances=tpgmm.covariances_[0], input_idx=[3])
>> gmr.fit(trajectory[0])
```

3.2.2 Member Function Documentation

3.2.2.1 fit()

```
None tpgmm.gmr.gmr.GaussianMixtureRegression.fit (
              self,
             ndarray translation,
             ndarray rotation_matrix )
```

Turns the task_parameterized gaussian mixture model into a single gaussian mixture model.

```
   function is performing equation (5) and (6) from calinon paper
```

TODO write formulas

Parameters

translation	translation matrix for translating into desired frames. Shape (num_frames, num_output_features)
rotation_matrix	rotation matrix for rotating into desired frames. Shape (num_frames, num_output_features, num_output_features)

3.2.2.2 num_input_features()

```
int tpgmm.gmr.gmr.GaussianMixtureRegression.num_input_features (
              self )
```

get number of input features.

Example

if you have 4 features: x, y, z, time. You can define with input_idx=[3] time as the input feature. This function will return then 1

Returns

int number of input features

3.2.2.3 num_output_features()

```
int tpgmm.gmr.gmr.GaussianMixtureRegression.num_output_features (
              self )
```

get number of output features.

Example

if you have 4 features: x, y, z, time. You can define with input_idx=[3] time as the input feature. This function will return then 3 for x,y and z

Returns

int number of output features

3.2.2.4 predict()

```
Tuple[ndarray, ndarray] tpgmm.gmr.gmr.GaussianMixtureRegression.predict (
               self,
              ndarray input_data )
```

this function is inspired by formula 13 in Calinon paper it creates for each given data point its own parameterized gaussian distribution.

The mechanics are described by:

$$\mathcal{P}(\phi_t^{\mathcal{O}}|\phi_t^{\mathcal{I}}) \sim \sum_{i=1}^{K} h_i(\phi_t^{\mathcal{I}}) \mathcal{N}\left(\hat{\mu}_t^{\mathcal{O}}(\phi_t^{\mathcal{I}}), \hat{\Sigma}_t^{\mathcal{O}}\right)$$

in:
`self.h()`

$$h_i(\phi_t^{\mathcal{I}}) = \frac{\pi_i \mathcal{N}(\phi_t^{\mathcal{I}} \mid \mu_i^{\mathcal{I}}, \Sigma_i^{\mathcal{I}})}{\sum_k^K \pi_k \mathcal{N}(\phi_t^{\mathcal{I}} \mid \mu_k^{\mathcal{I}}, \Sigma_k^{\mathcal{I}})}$$

in:
`self.mu_hat()`

$$\hat{\mu}_i^{\mathcal{O}}(\phi_t^{\mathcal{I}}) = \mu_i^{\mathcal{O}} + \Sigma_i^{\mathcal{O}\mathcal{I}} \Sigma_i^{\mathcal{I},-1} (\phi_t^{\mathcal{I}} - \mu_i^{\mathcal{I}})$$

in:
`self.sigma_hat()`

$$\hat{\Sigma}_t^{\mathcal{O}} = \Sigma_i^{\mathcal{O}} - \Sigma_i^{\mathcal{O}\mathcal{I}} \Sigma_i^{\mathcal{I},-1} \Sigma_i^{\mathcal{O}\mathcal{I}}$$

Parameters

data	Shape: (num_points, num_input_features)

Returns

Tuple[ndarray, ndarray] : mu: shape -> (num_points, num_output_features), cov: shape (num_points, num↩ _output_features, num_output_features)

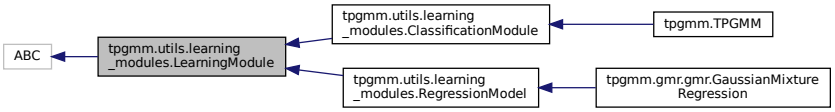
The documentation for this class was generated from the following file:

- documentation/tmp/tpgmm/gmr/gmr.py

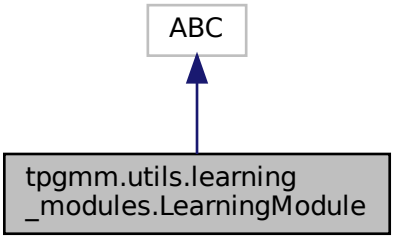
3.3 tpgmm.utils.learning_modules.LearningModule Class Reference

Basic abstract class for a generic learning module.

Inheritance diagram for tpgmm.utils.learning_modules.LearningModule:



Collaboration diagram for tpgmm.utils.learning_modules.LearningModule:



Public Member Functions

- None **__init__** (self)
- None fit (self, ∗args, ∗∗kwargs)

  Basic abstract class for a generic learning module.
- def predict (self, ∗args, ∗∗kwargs)

  Predict output using the fitted model.
- ndarray fit_predict (self, ndarray X)

  Convenience method; equivalent to calling fit(data) followed by predict(data).
- Dict[str, Any] config (self)

  Get all model config parameters.

3.3.1 Detailed Description

Basic abstract class for a generic learning module.

3.3.2 Member Function Documentation

3.3.2.1 fit_predict()

```
ndarray tpgmm.utils.learning_modules.LearningModule.fit_predict (
             self,
            ndarray X )
```

Convenience method; equivalent to calling fit(data) followed by predict(data).

Parameters

X	Data in local reference frames. Shape (num_frames, num_points, num_features).

Returns

ndarray The label for each data-point. Shape (num_points).

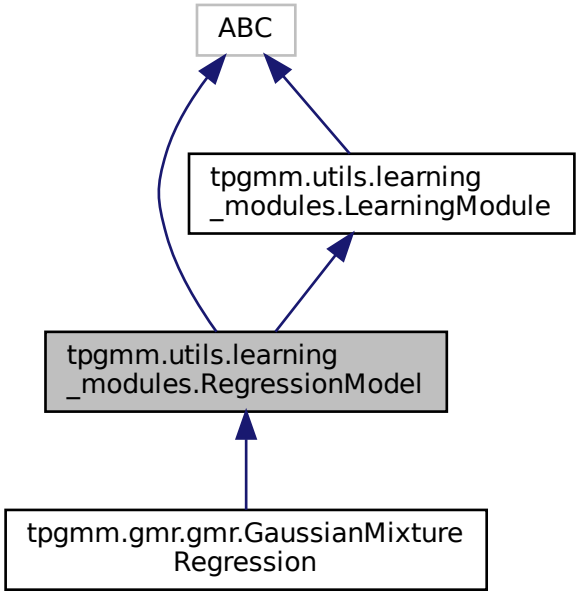
The documentation for this class was generated from the following file:

- documentation/tmp/tpgmm/utils/learning_modules.py

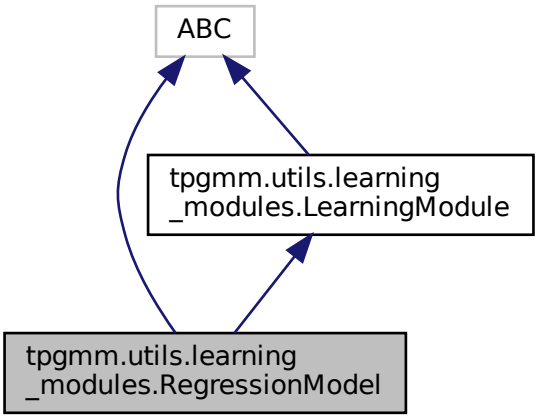
3.4 tpgmm.utils.learning_modules.RegressionModel Class Reference

Basic Regression Model.

Inheritance diagram for tpgmm.utils.learning_modules.RegressionModel:



Collaboration diagram for tpgmm.utils.learning_modules.RegressionModel:



Additional Inherited Members

3.4.1 Detailed Description

Basic Regression Model.

Implements all interfaces and common methods for regression models.

Basic Classification Model. Implements all interfaces and common methods for classification models.

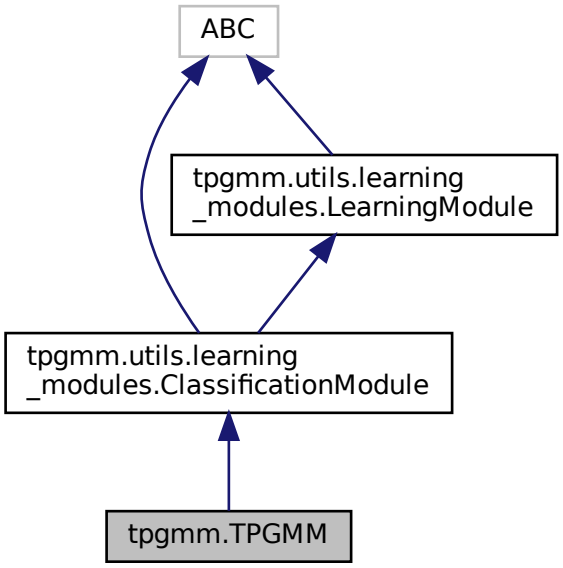
The documentation for this class was generated from the following file:

- documentation/tmp/tpgmm/utils/learning_modules.py

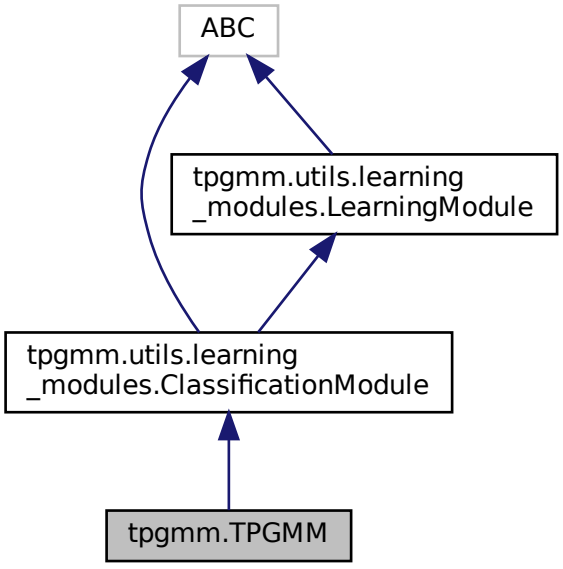
3.5 tpgmm.TPGMM Class Reference

This class in an implementation of the task parameterized gaussian mixture model according to Calinon Paper @https://calinon.ch/papers/Calinon-JIST2015.pdf.

Inheritance diagram for tpgmm.TPGMM:



Collaboration diagram for tpgmm.TPGMM:



Public Member Functions

- None __init__ (self, int n_components, float threshold=1e-7, int max_iter=100, int min_iter=5, ndarray weights_init=None, ndarray means_init=None, float reg_factor=1e-5, bool verbose=False)

  Initialize the TPGMM class.
- None fit (self, ndarray X)

  fits X on the task parameterized gaussian mixture model using K-Means clustering as default initialization method and executes the expectation maximization algorithm:
  E-Step:
- ndarray predict (self, ndarray X)

  predict cluster labels for each data point in X
- ndarray predict_proba (self, ndarray X)

  predict cluster labels for each data point
- float silhouette_score (self, ndarray X)

  calculated the silhouette score of the model over the given metric and given data x TODO(Marco Todescato): please review this function if the merge for the silhouette score is correct
- float inertia (self, ndarray X)

  Sum of squared distances of samples to their closest cluster center.
- float score (self, ndarray X)

  calculate log likelihood score given data
- float bic (self, ndarray X)

  calculates the bayesian information criterion as in
- float davies_bouldin_score (self, ndarray X)

  calculates the davies bouldin score for each frame and averages them
- ndarray gauss_cdf (self, ndarray X)

  calculate the gaussian probability for a given data set.
- Dict[str, Any] config (self)

  Returns a dictionary containing the configuration parameters of the TPGMM model.

Public Attributes

- weights_

  ndarray of shape (n_components,) Weights between gaussian components.
- means_

  ndarray of shape (num_frames, n_components, num_features) Mean matrix for each frame and component.
- covariances_

  ndarray of shape (num_frames, n_components, num_features, num_features).

3.5.1 Detailed Description

This class in an implementation of the task parameterized gaussian mixture model according to Calinon Paper @https://calinon.ch/papers/Calinon-JIST2015.pdf.

It implements also an Expectation Maximization Algorithm with:
E-Step:

$$h_{t,i} = \frac{\pi_i \prod_{j=1}^{P} \mathcal{N}\left(X_t^{(j)} \mid \mu_i^{(j)}, \Sigma_i^{(j)}\right)}{\sum_{k=1}^{K} \pi_k \prod_{j=1}^{P} \mathcal{N}\left(X_t^{(j)} \mid \mu_k^{(j)}, \Sigma_k^{(j)}\right)}$$

M-Step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}}{N}$$

$$\mu_i^{(j)} \leftarrow \frac{\sum_{t=1}^{N} h_{t,i} X_t^{(j)}}{\sum_{t=1}^{N} h_{t,i}}$$

$$\Sigma_i^{(j)} \leftarrow \frac{\sum_{t=1}^{N} h_{t,i} \left(X_t^{(j)} - \mu_i^{(j)}\right) \left(X_t^{(j)} - \mu_i^{(j)}\right)^T}{\sum_{t=1}^{N} h_{t,i}}$$

The optimization criterion is the log-likelihood implemented with:

$$LL = \frac{\sum_{t=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \prod_{j=1}^{J} \mathcal{N}\left(X_t^{(j)} \mid \mu_k^{(j)}, \Sigma_k^{(j)}\right)\right)}{N}$$

Variable explanation:
N ... number of components
π ... weights between components
i ... component index
j ... frame index (like pick or place frame)
μ ... mean
Σ ... variance / covariance matrix
LL ... log likelihood

Examples

```
>> trajectories = load_trajectories()  # trajectory data in local reference frames with shape:
       (num_reference_frames, num_points, 3)
>> tpgmm = TPGMM(n_components=5)
>> tpgmm.fit(trajectories)
```

Parameters

n_components	number of components
tol	threshold to break from EM algorithm. Defaults to 1e-3.
max_iter	maximum number of iterations to perform the expectation maximization algorithm. Defaults to 100.
min_iter	minimum number of iterations to perform the expectation maximization algorithm. Defaults to 5.
weights_init	initial weights between each component. If set replaces initialization from K-Means. Defaults to None.
means_init	initial means between each component. If set replaces initialization from K-Means. Defaults to None.
reg_factor	regularization factor for empirical covariance matrix. Defaults to 1e-5.
verbose	Triggers print of learning stats. Defaults to False.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 __init__()

```
None tpgmm.TPGMM.__init__ (
              self,
             int n_components,
             float  threshold = 1e-7,
```

```
            int  max_iter = 100,
            int  min_iter = 5,
            ndarray  weights_init = None,
            ndarray  means_init = None,
            float  reg_factor = 1e-5,
            bool  verbose = False )
```

Initialize the TPGMM class.

Parameters

n_components	Number of Gaussian multidimensional distributions to mix.
threshold	Threshold to break from EM algorithm. Defaults to 1e-3.
max_iter	Maximum number of iterations to perform the expectation maximization algorithm. Defaults to 100.
min_iter	Minimum number of iterations to perform the expectation maximization algorithm. Defaults to 5.
weights_init	Initial weights between each component. If set, it replaces initialization from K-Means. Defaults to None.
means_init	Initial means between each component. If set, it replaces initialization from K-Means. Defaults to None.
reg_factor	Regularization factor for the empirical covariance matrix. Defaults to 1e-5.
verbose	Triggers print of learning stats. Defaults to False.

3.5.3 Member Function Documentation

3.5.3.1 bic()

```
float tpgmm.TPGMM.bic (
             self,
            ndarray X )
```

calculates the bayesian information criterion as in

```
https://scikit-learn.org/stable/modules/linear_model.html#aic-bic
```

Parameters

X	data tensor with expected shape (num_frames, num_points, num_features)

Returns

float bic score

Reimplemented from tpgmm.utils.learning_modules.ClassificationModule.

3.5.3.2 config()

```
Dict[str, Any] tpgmm.TPGMM.config (
               self )
```

Returns a dictionary containing the configuration parameters of the TPGMM model.

Returns

Dict[str, Any]: Dictionary containing the configuration parameters.

Reimplemented from tpgmm.utils.learning_modules.ClassificationModule.

3.5.3.3 davies_bouldin_score()

```
float tpgmm.TPGMM.davies_bouldin_score (
               self,
              ndarray X )
```

calculates the davies bouldin score for each frame and averages them

3.5.4 TODO(Marco Todescator): is this score correct?

Parameters

X	data to calculate the score on. Expected shape: (num_frames, num_points, num_features)

Returns

float score value

Reimplemented from tpgmm.utils.learning_modules.ClassificationModule.

3.5.4.1 fit()

```
None tpgmm.TPGMM.fit (
               self,
              ndarray X )
```

fits X on the task parameterized gaussian mixture model using K-Means clustering as default initialization method and executes the expectation maximization algorithm:
E-Step:
`self._update_h()`

M-Step:
`self._update_weights_()`

```
self._update_means_()
self._update_covariances()
```

The optimization criterion is the log-likelihood implemented in:
`self._log_likelihood()`

The algorithm stops if $LL_{t-1} - LL_t < \textit{self.tol}$ with LL_t as the log-likelihood at time t.

Parameters

X	data tensor to fit the the task parameterized gaussian mixture model on. Expected shape: (num_frames, num_points, num_features)

3.5.4.2 gauss_cdf()

```
ndarray tpgmm.TPGMM.gauss_cdf (
              self,
             ndarray X )
```

calculate the gaussian probability for a given data set.

$$\mathcal{N}\left(X_t^{(j)} \mid \mu_i^{(j)}, \Sigma_i^{(j)}\right) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_i^{(j)}|}} \exp\left(\frac{1}{2}(X_t^{(j)} - \mu_i^{(j)})\Sigma_i^{(j),-1}(X_t^{(j)} - \mu_i^{(j)})^T\right)$$

Variable explanation: D ... number of features

Parameters

X	data with shape: (num_frames, num_points, num_features)

Returns

ndarray probability shape (num_frames, n_components, num_points)

3.5.4.3 inertia()

```
float tpgmm.TPGMM.inertia (
              self,
             ndarray X )
```

Sum of squared distances of samples to their closest cluster center.

In case of multiple frames we take the mean squared distance to their closest cluster center over all frames.

TODO(Marco Todescato): please review if this is correct

Parameters

X	data in local reference frames. Shape (num_frames, num_points, num_features)

Returns

float average inertia score over all frames

Reimplemented from tpgmm.utils.learning_modules.ClassificationModule.

3.5.4.4 predict()

```
ndarray tpgmm.TPGMM.predict (
              self,
             ndarray X )
```

predict cluster labels for each data point in X

Parameters

X	data in local reference frames. Shape (num_frames, num_points, num_features)

Returns

ndarray the label for each data-point. Shape (num_points)

3.5.4.5 predict_proba()

```
ndarray tpgmm.TPGMM.predict_proba (
              self,
             ndarray X )
```

predict cluster labels for each data point

Parameters

X	data in local reference frames. Shape (num_frames, num_points, num_features)

Returns

ndarray cluster probabilities for each data_point. Shape: (num_points, num_components)

3.5.4.6 score()

```
float tpgmm.TPGMM.score (
              self,
             ndarray X )
```

calculate log likelihood score given data

Parameters

X	data tensor with expected shape (num_frames, num_points, num_features)

Returns

float log likelihood of given data

Reimplemented from tpgmm.utils.learning_modules.ClassificationModule.

3.5.4.7 silhouette_score()

```
float tpgmm.TPGMM.silhouette_score (
              self,
             ndarray X )
```

calculated the silhouette score of the model over the given metric and given data x TODO(Marco Todescato): please review this function if the merge for the silhouette score is correct

Parameters

X	data in expected shape: (num_frames, num_points, num_features)
metric	*description*. Defaults to "euclidean".

Reimplemented from tpgmm.utils.learning_modules.ClassificationModule.

3.5.5 Member Data Documentation

3.5.5.1 covariances_

```
tpgmm.TPGMM.covariances_
```

ndarray of shape (num_frames, n_components, num_features, num_features).

Covariance matrix for each frame and component

The documentation for this class was generated from the following file:

- documentation/tmp/tpgmm/tpgmm/tpgmm.py

Index