# Cluster

## Contents



Clustering algorithm implementations using PyTorch.

***class*** **anomalib.models.components.cluster.GaussianMixture(*n_components*, *n_iter=100*, *tol=0.001*)**

Bases: `DynamicBufferMixin`

Gaussian Mixture Model.

**Parameters:**

- **n_components** (*int*) – Number of components.
- **n_iter** (*int*) – Maximum number of iterations to perform. Defaults to `100`.
- **tol** (*float*) – Convergence threshold. Defaults to `1e-3`.

### Example

The following examples shows how to fit a Gaussian Mixture Model to some data and get the cluster means and predicted labels and log-likelihood scores of the data.

```
>>> import torch
>>> from anomalib.models.components.cluster import GaussianMixture
>>> model = GaussianMixture(n_components=2)
>>> data = torch.tensor(
...     [
...             [2, 1], [2, 2], [2, 3],
...             [7, 5], [8, 5], [9, 5],
...     ]
... ).float()
>>> model.fit(data)
```

```
>>> model.means  # get the means of the gaussians
tensor([[8., 5.],
        [2., 2.]])
>>> model.predict(data)  # get the predicted cluster label of each sample
tensor([1, 1, 1, 0, 0, 0])
>>> model.score_samples(data)  # get the log-likelihood score of each sample
tensor([3.8295, 4.5795, 3.8295, 3.8295, 4.5795, 3.8295])
```

### `fit(data)`

Fit the model to the data.

**Parameters:**

**data** (*Tensor*) – Data to fit the model to. Tensor of shape (n_samples, n_features).

**Return type:**

`None`

### `predict(data)`

Predict the cluster labels of the data.

**Parameters:**

**data** (*Tensor*) – Samples to assign to clusters. Tensor of shape (n_samples, n_features).

**Returns:**

Tensor of shape (n_samples,) containing the predicted cluster label of each sample.

**Return type:**

Tensor

### `score_samples(data)`

Assign a likelihood score to each sample in the data.

**Parameters:**

**data** (*Tensor*) – Samples to assign scores to. Tensor of shape (n_samples, n_features).

**Returns:**

Tensor of shape (n_samples,) containing the log-likelihood score of each sample.

**Return type:**

Tensor

### *class* anomalib.models.components.cluster.KMeans(*n_clusters*, *max_iter=10*)

Bases: `object`

Initialize the KMeans object.

**Parameters:**

- **n_clusters** (*int*) – The number of clusters to create.
- **max_iter** (*int, optional*) – The maximum number of iterations to run the algorithm. Defaults to 10.

#### fit(*inputs*)

Fit the K-means algorithm to the input data.

**Parameters:**

**inputs** (*torch.Tensor*) – Input data of shape (batch_size, n_features).

**Returns:**

A tuple containing the labels of the input data with respect to the identified clusters and the cluster centers themselves. The labels have a shape of (batch_size,) and the cluster centers have a shape of (n_clusters, n_features).

**Return type:**

tuple

**Raises:**

**ValueError** – If the number of clusters is less than or equal to 0.

#### predict(*inputs*)

Predict the labels of input data based on the fitted model.

**Parameters:**

**inputs** (*torch.Tensor*) – Input data of shape (batch_size, n_features).

**Returns:**

The predicted labels of the input data with respect to the identified clusters.

**Return type:**

torch.Tensor

**Raises:**

**AttributeError** – If the KMeans object has not been fitted to input data.

Previous
Classification

Next
Stats Components