kmeans

NAME

kmeans - A clustering algorithm commonly used in EDA (exploratory data analysis).

SYNOPSIS

class frovedis.mllib.cluster.KMeans (n_clusters=8, init='k-means++',
n_init=10, max_iter=300, tol=1e-4, precompute_distances='auto',
verbose=0, random_state=None, copy_x=True,
n_jobs=1, algorithm='auto')

Public Member Functions

fit(X, y=None)
predict(X)
save(filename)
load(filename)
debug_print()
release()

DESCRIPTION

Clustering is an unsupervised learning problem whereby we aim to group subsets of entities with one another based on some notion of similarity. K-means is one of the most commonly used clustering algorithms that clusters the data points into a predefined number of clusters (K).

This module provides a client-server implementation, where the client application is a normal python scikit-learn program. Scikit-learn has its own cluster module providing kmeans support. But that algorithm is non-distributed in nature. Hence it is slower when comparing with the equivalent Frovedis algorithm (see frovedis manual for ml/kmeans) with big dataset. Thus in this implementation, a scikit-learn client can interact with a frovedis server sending the required python data for training at frovedis side. Python data is converted into frovedis compatible data internally and the scikit-learn ML call is linked with the respective frovedis ML call to get the job done at frovedis server.

Scikit-learn side call for kmeans quickly returns, right after submitting the training request to the frovedis server with a unique model ID for the submitted training request.

When operations like prediction will be required on the trained model, scikit-learn client sends the same request to frovedis server on the same model (containing the unique ID) and the request is served at frovedis server and output is sent back to the scikit-learn client.

Detailed Description

KMeans()

Parameters *n_clusters*: An integer parameter specifying the number of clusters. (Default: 8)
init: A string object. (unused)
n_init: An integer parameter. (unused)
max_iter: An integer parameter specifying the maximum iteration count. (Default: 300)
tol: A double parameter specifying the convergence tolerance. (Default: 1e-4)
precompute_distances: A string object. (unused)
verbose: An integer object specifying the log level to use. (Default: 0)
random_state: An integer, None or RandomState instance. (unused)
copy_X: A boolean parameter. (unused)
n_jobs: An integer parameter. (unused)
algorithm: A string object. (unused)

Purpose
It initialized a KMeans object with the given parameters.

The parameters: "init", "n_init", "precompute_distances", "random_state", "copy_X", "n_jobs" and "algorithms" are not yet supported at frovedis side. Thus they don't have any significance in this call. They are simply provided for the compatibility with scikit-learn application.

"verbose" value is set at 0 by default. But it can be set to 1 (for DEBUG mode) or 2 (for TRACE mode) for getting training time logs from frovedis server.

Return Value
It simply returns "self" reference.

fit(X, y=None)

Parameters
X: A scipy sparse matrix or any python array-like object or an instance of FrovedisCRSMatrix.
y: None (simply ignored in scikit-learn as well).

Purpose

It clusters the given data points (X) into a predefined number (k) of clusters.

For example,

```
# loading sample CRS data file
mat = FrovedisCRSMatrix().load("./sample")

# fitting input matrix on kmeans object
kmeans = KMeans(n_clusters=2, verbose=2).fit(mat)
```

Return Value
It simply returns "self" reference.
Note that the call will return quickly, right after submitting the fit request at frovedis server side with a unique model ID for the fit request. It may be possible that the training is not completed at the frovedis server side even though the client scikit-learn side fit() returns.

predict(X)

Parameters
X: A scipy sparse matrix or any python array-like object or an instance of FrovedisCRSMatrix.

Purpose
It accepts the test data points (X) and returns the centroid information.

Return Value
It returns a numpy array of integer (int32) type containing the centroid values.

save(filename)

Parameters
filename: A string object containing the name of the file on which the target model is to be saved.

Purpose
On success, it writes the model information in the specified file as little-endian binary data. Otherwise, it throws an exception.

Return Value
It returns nothing.

load(filename)

Parameters
filename: A string object containing the name of the file having model information to be loaded.

Purpose
It loads the model from the specified file (having little-endian binary data).

Return Value
It simply returns "self" instance.

debug_print()

Purpose
It shows the target model information on the server side user terminal. It is mainly used for debugging purpose.

Return Value
It returns nothing.

release()

Purpose
It can be used to release the in-memory model at frovedis server.

Return Value
It returns nothing.