# Matrix Factorization using ALS

## NAME

Matrix Factorization using ALS - A matrix factorization algorithm commonly used for recommender systems.

## SYNOPSIS

class frovedis.mllib.recommendation.ALS (max_iter=100, alpha=0.01, regParam=0.01,
seed=0, verbose=0)

### Public Member Functions

fit(X, rank)
predict(id)
recommend_users (pid, k)
recommend_products (uid, k)
save(filename)
load(filename)
debug_print()
release()

## DESCRIPTION

Collaborative filtering is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix. Frovedis currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries.

Frovedis uses the alternating least squares (ALS) algorithm to learn these latent factors. The algorithm is based on a paper "Collaborative Filtering for Implicit Feedback Datasets" by Hu, et al.

This module provides a client-server implementation, where the client application is a normal python scikit-learn program. Scikit-learn does not have any collaborative filtering recommender algorithms like ALS. In this implementation, scikit-learn side recommender interfaces are provided, where 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 ALS call is linked with the frovedis ALS call to get the job done at frovedis server.

Scikit-learn side call for ALS 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 recommendation 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

### ALS ()

**Parameters**
*max_iter*: An integer parameter specifying maximum iteration count. (Default: 100)
*alpha*: A double parameter containing the learning rate (Default: 0.01)
*regParam*: A double parameter containing the regularization parameter (Default: 0.01)
*seed*: A long parameter containing the seed value to initialize the model structures with random values. (Default: 0)
*verbose*: An integer parameter specifying the log level to use. (Default: 0)

**Purpose**

It initialized an ALS object with the given parameters.

"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, rank)

**Parameters**
*X*: A scipy sparse matrix or any python array-like object or an instance of FrovedisCRSMatrix.
*rank*: An integer parameter containing the user given rank for the input matrix.

**Purpose**
It accepts the training matrix (X) and trains a matrix factorization model on that at frovedis server.

It starts with initializing the model structures of the size MxF and NxF (where M is the number of users and N is the products in the given rating matrix and F is the given rank) with random values and keeps updating them until maximum iteration count is reached.

For example,

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

# fitting input matrix on ALS object
als = ALS().fit(mat,rank=4)
```

**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(ids)**

**Parameters**
*ids*: A python tuple or list object containing the pairs of user id and product id to predict.

**Purpose**
It accepts a list of pair of user ids and product ids (0-based ID) in order to make prediction for their ratings from the trained model at frovedis server.

For example,

```
# this will print the predicted ratings for the given list of id pairs
print als.predict([(1,1), (0,1), (2,3), (3,1)])
```

**Return Value**
It returns a numpy array of double (float64) type containing the predicted ratings.

**recommend_users(pid, k)**

**Parameters**
*pid*: An integer parameter specifying the product ID (0-based) for which to recommend users.
*k*: An integer parameter specifying the number of users to be recommended.

**Purpose**
It recommends the best "k" users with highest rating confidence in sorted order for the given product.

If k > number of rows (number of users in the given matrix when training the model), then it resets the k as "number of rows in the given matrix" in order to recommend all the users with rating confidence values in sorted order.

**Return Value**
It returns a python list containing the pairs of recommended users and their corresponding rating confidence values in sorted order.

**recommend_products(uid, k)**

**Parameters**
*uid*: An integer parameter specifying the user ID (0-based) for which to recommend products.
*k*: An integer parameter specifying the number of products to be recommended.

**Purpose**
It recommends the best "k" products with highest rating confidence in sorted order for the given user.

If k > number of columns (number of products in the given matrix when training the model), then it resets the k as "number of columns in the given matrix" in order to recommend all the products with rating confidence values in sorted order.

**Return Value**
It returns a python list containing the pairs of recommended products and their corresponding rating confidence values in sorted order.

**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 (user-product features etc.) 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.