frovedis::matrix_factorization_model<T>

NAME

matrix_factorization_model<T> - A data structure used in modeling the outputs of the frovedis matrix factorization using ALS algorithm

SYNOPSIS

#include <frovedis/ml/recommendation/matrix_factorization_model.hpp>

Constructors

matrix_factorization_model ()
matrix_factorization_model (size_t num_row, size_t num_col,
size_t factor, size_t seed=0)
matrix_factorization_model (const matrix_factorization_model<T>& model)
matrix_factorization_model (matrix_factorization_model<T>&& model)

Overloaded Operators

matrix_factorization_model<T>& operator= (const matrix_factorization_model<T>& model)
matrix_factorization_model<T>& operator= (matrix_factorization_model<T>&& model)

Public Member Functions

T predict (size_t uid, size_t pid)
std::vector<T> predict_all (const std::vector<std::pair<size_t,size_t>> IDs)
rowmajor_matrix_local<T> predict_all()

std::vector<std::pair<size_t,T>> recommend_products(size_t uid, int num)
std::vector<std::pair<size_t,T>> recommend_users(size_t pid, int num)

void save (const std::string& path)
void savebinary (const std::string& path)
void load (const std::string& path)
void loadbinary (const std::string& path)

size_t get_rank ()
void debug_print ()
node_local<matrix_factorization_model<T>> broadcast()

DESCRIPTION

`matrix_factorization_model<T>` models the output of the frovedis matrix factorization using ALS (alternating least square) algorithm, the trainer interface of which aims to optimize an initial model and outputs the same after optimization. This model has the below structure:

```
template <class T>
struct matrix_factorization_model {
  std::vector<T> X;  // user-feature vector of the size numRows*factor
  std::vector<T> Y;  // product-feature vector of the size numCols*factor
  size_t numRows;
  size_t numCols;
  size_t factor;
  SERIALIZE (X, Y, numRows, numCols, factor)
};
```

This is a template based data structure, where "T" is supposed to be "float" (single-precision) or "double" (double-precision). Note this is a serialized data structure. The detailed description can be found in subsequent sections.

Constructor Documentation

matrix_factorization_model ()

Default constructor. It creates an empty matrix factorization model with numRows = numCols = factor = 0.

matrix_factorization_model (size_t num_row, size_t num_col,

size_t factor, size_t seed)

Parameterized constructor. It accepts number of rows(M), number of columns(N), latent factor(F) and seed value in order to create a model with "X" matrix of the dimension MxF and "Y" matrix of the dimension NxF initialized with random numbers according to the given seed.

matrix_factorization_model (const `matrix_factorization_model<T>`& model)

Copy constructor. It accepts an lvalue object of the same type and deep-copies the same in the newly constructed object.

matrix_factorization_model (`matrix_factorization_model<T>`&& model)

Move constructor. It accepts an rvalue object of the same type and instead of copying, it moves the contents in the newly constructed object.

Overloaded Operator Documentation

`matrix_factorization_model<T>`& operator= (const `matrix_factorization_model<T>`& model)

It deep-copies the contents of the input lvalue model into the left-hand side model of the assignment operator "=".

matrix_factorization_model<T>& operator= (matrix_factorization_model<T>&& model)

Instead of copying, it moves the contents of the input rvalue model into the left-hand side model of the assignment operator "=".

Pubic Member Function Documentation

T predict (size_t uid, size_t pid)

This method can be used on a trained model in order to predict the rating confidence value for the given product id, by the given user id.

"uid" should be in between 0 to numRows-1.
And "pid" should be in between 0 to numCols-1. Otherwise exception will be thrown.

std::vector<T> predict_all (const std::vector<std::pair<size_t,size_t>> IDs)

This method can be used to predict the rating confidence values for a given list of pair of some user ids and product ids.

In the list of pairs, "uid" should be in between 0 to numRows-1.
And "pid" should be in between 0 to numCols-1. Otherwise exception will be thrown.

On successful prediction, it returns the predicted scores in the form of `std::vector<T>`.

rowmajor_matrix_local<T> predict_all ()

This method can be used in order to predict the rating confidence values for all the users and for all the products. Thus internally it performs a product of X and Y component of the model (X * Yt) and returns the resultant scores in the form of a `rowmajor_matrix_local<T>` with MxN dimension, where M is the number of rows in X component and N is the number of rows in Y component. This method is useful in case of debugging the model.

std::vector<std::pair<size_t,T>> recommend_products(size_t uid, int num)

This method can be used to recommend given "num" number of products for the user with given user id in sorted order (highest scored products to lowest scored products).

"uid" should be in between 0 to numRows-1.
If num > numCols, then "numCols" number of products would be recommended. On success, it returns a vector of pairs containing recommended product ids and their corresponding rating confidence scores by the given user.

std::vector<std::pair<size_t,T>> recommend_users(size_t pid, int num)

This method can be used to recommend given "num" number of users for the product with given product id in sorted order (user with highest scores to user with lowest scores).

"pid" should be in between 0 to numCols-1.
If num > numRows, then "numRows" number of users would be recommended. On success, it returns a vector of pairs containing recommended user ids and their corresponding rating confidence scores for the given product.

size_t get_rank ()

It returns the latent factor of the target model.

void save (const std::string& path)

It saves the target model in the specified path in simple text format. It will throw an exception, if any error occurs during the save operation.

void savebinary (const std::string& path)

It saves the target model in the specified path in (little-endian) binary data format. It will throw an exception, if any error occurs during the save operation.

void load (const std::string& path)

It loads the target matrix factorization model from the data in specified text file. It will throw an exception, if any error occurs during the load operation.

void loadbinary (const std::string& path)

It loads the target matrix factorization model from the data in specified (little-endian) binary file. It will throw an exception, if any error occurs during the load operation.

void debug_print()

It prints the contents of the X and Y components of the model on the user terminal. It is mainly useful for debugging purpose.

node_local<matrix_factorization_model<T>> broadcast ()

It broadcasts the target model to all the participating MPI processes (worker nodes) in the system. This is an efficient implementation (as it does not involve serialization overhead of the X and Y components of the model) than simple "frovedis:broadcast(model)" call.

Public Data Member Documentation

X

An T type vector used to model the user-feature matrix of the model.

Y

An T type vector used to model the product-feature matrix of the model.

numRows

A size_t attribute containing the number of rows in X component of the model.

numCols

A size_t attribute containing the number of rows in Y component of the model.

factor

A size_t attribute containing the latent factor of the model.