Package: recosystem (via r-universe)

November 6, 2024

Type Package

Title Recommender System using Matrix Factorization

Version 0.5.1

Date 2023-05-05

Author Yixuan Qiu, David Cortes, Chih-Jen Lin, Yu-Chin Juan, Wei-Sheng Chin, Yong Zhuang, Bo-Wen Yuan, Meng-Yuan Yang, and other contributors. See file AUTHORS for details.

Maintainer Yixuan Qiu <yixuan.qiu@cos.name>

Description R wrapper of the 'libmf' library <https://www.csie.ntu.edu.tw/~cjlin/libmf/> for recommender system using matrix factorization. It is typically used to approximate an incomplete matrix using the product of two matrices in a latent space. Other common names for this task include ``collaborative filtering", ``matrix completion", ``matrix recovery", etc. High performance multi-core parallel computing is supported in this package.

License BSD_3_clause + file LICENSE

Copyright see file COPYRIGHTS

URL https://github.com/yixuan/recosystem

BugReports https://github.com/yixuan/recosystem/issues

Depends R (>= 3.3.0), methods

Imports Rcpp (>= 0.11.0), float

Suggests knitr, rmarkdown, prettydoc, Matrix

LinkingTo Rcpp, RcppProgress

VignetteBuilder knitr

RoxygenNote 7.2.3

Repository https://yixuan.r-universe.dev

RemoteUrl https://github.com/yixuan/recosystem

RemoteRef HEAD

RemoteSha f1207182ed700ec957cc36f25137e0d882566111

Contents

`data_source` *Specifying Data Source*

Description

Functions in this page are used to specify the source of data in the recommender system. They are intended to provide the input argument of functions such as `$tune()`, `$train()`, and `$predict()`. Currently three data formats are supported: data file (via function `data_file()`), data in memory as R objects (via function `data_memory()`), and data stored as a sparse matrix (via function `data_matrix()`).

Usage

```
data_file(path, index1 = FALSE, ...)

data_memory(user_index, item_index, rating = NULL, index1 = FALSE, ...)

data_matrix(mat, ...)
```

Arguments

`path`	Path to the data file.
`index1`	Whether the user indices and item indices start with 1 (`index1 = TRUE`) or 0 (`index1 = FALSE`).
`...`	Currently unused.
`user_index`	An integer vector giving the user indices of rating scores.
`item_index`	An integer vector giving the item indices of rating scores.
`rating`	A numeric vector of the observed entries in the rating matrix. Can be specified as `NULL` for testing data, in which case it is ignored.
`mat`	A `dgTMatrix` (if it has ratings/values) or `ngTMatrix` (if it is binary) sparse matrix, with users corresponding to rows and items corresponding to columns.

Details

In `$tune()` and `$train()`, functions in this page are used to specify the source of training data.

`data_file()` expects a text file that describes a sparse matrix in triplet form, i.e., each line in the file contains three numbers

```
row col value
```

representing a number in the rating matrix with its location. In real applications, it typically looks like

```
user_index item_index rating
```

The '`smalltrain.txt`' file in the '`dat`' directory of this package shows an example of training data file.

If the sparse matrix is given as a `dgTMatrix` or `ngTMatrix` object (triplets/COO format defined in the **Matrix** package), then the function `data_matrix()` can be used to specify the data source.

If user index, item index, and ratings are stored as R vectors in memory, they can be passed to `data_memory()` to form the training data source.

By default the user index and item index start with zeros, and the option `index1 = TRUE` can be set if they start with ones.

From version 0.4 **recosystem** supports two special types of matrix factorization: the binary matrix factorization (BMF), and the one-class matrix factorization (OCMF). BMF requires ratings to take value from $-1, 1$, and OCMF requires all the ratings to be positive.

In `$predict()`, functions in this page provide the source of testing data. The testing data have the same format as training data, except that the value (rating) column is not required, and will be ignored if it is provided. The '`smalltest.txt`' file in the '`dat`' directory of this package shows an example of testing data file.

Value

An object of class "DataSource" as required by `$tune()`, `$train()`, and `$predict()`.

Author(s)

Yixuan Qiu <https://statr.me>

See Also

`$tune()`, `$train()`, `$predict()`

`output` *Exporting Factorization Matrices*

Description

This method is a member function of class "RecoSys" that exports the user score matrix P and the item score matrix Q.

Prior to calling this method, model needs to be trained using member function `$train()`.

The common usage of this method is

```
r = Reco()
r$train(...)
r$output(out_P = out_file("mat_P.txt"), out_Q = out_file("mat_Q.txt"))
```

Arguments

`r`	Object returned by `Reco()`.
`out_P`	An object of class `Output` that specifies the output format of the user matrix, typically returned by function `out_file()`, `out_memory()` or `out_nothing()`. `out_file()` writes the matrix into a file, with each row representing a user and each column representing a latent factor. `out_memory()` exports the matrix into the return value of `$output()`. `out_nothing()` means the matrix will not be exported.
`out_Q`	Ditto, but for the item matrix.

Value

A list with components `P` and `Q`. They will be filled with user or item matrix if `out_memory()` is used in the function argument, otherwise `NULL` will be returned.

Author(s)

Yixuan Qiu <https://statr.me>

References

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems. ACM TIST, 2015.

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. PAKDD, 2015.

W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. Technical report, 2015.

See Also

`$train()`, `$predict()`

Examples

```
train_set = system.file("dat", "smalltrain.txt", package = "recosystem")
r = Reco()
set.seed(123) # This is a randomized algorithm
r$train(data_file(train_set), out_model = file.path(tempdir(), "model.txt"),
        opts = list(dim = 10, nmf = TRUE))

## Write P and Q matrices to files
P_file = out_file(tempfile())
Q_file = out_file(tempfile())
r$output(P_file, Q_file)
head(read.table(P_file@dest, header = FALSE, sep = " "))
head(read.table(Q_file@dest, header = FALSE, sep = " "))

## Skip P and only export Q
r$output(out_nothing(), Q_file)

## Return P and Q in memory
res = r$output(out_memory(), out_memory())
head(res$P)
head(res$Q)
```

`output_format` *Specifying Output Format*

Description

Functions in this page are used to specify the format of output results. They are intended to provide the argument of functions such as `$output()` and `$predict()`. Currently there are three types of output: `out_file()` indicates that the result should be written into a file, `out_memory()` makes the result to be returned as R objects, and `out_nothing()` means the result is not needed and will not be returned.

Usage

```
out_file(path, ...)

out_memory(...)

out_nothing(...)
```

Arguments

`path`	Path to the output file.
`...`	Currently unused.

Value

An object of class "Output" as required by `$output()` and `$predict()`.

Author(s)

Yixuan Qiu <https://statr.me>

See Also

`$output()`, `$predict()`

`predict` *Recommender Model Predictions*

Description

This method is a member function of class "RecoSys" that predicts unknown entries in the rating matrix.

Prior to calling this method, model needs to be trained using member function `$train()`.

The common usage of this method is

```
r = Reco()
r$train(...)
r$predict(test_data, out_pred = out_file("predict.txt"))
```

Arguments

`r`	Object returned by `Reco()`.
`test_data`	An object of class "DataSource" that describes the source of testing data, typically returned by function `data_file()`, `data_memory()`, or `data_matrix()`.
`out_pred`	An object of class `Output` that specifies the output format of prediction, typically returned by function `out_file()`, `out_memory()` or `out_nothing()`. `out_file()` writes the result into a file, `out_memory()` exports the vector of predicted values into the return value of `$predict()`, and `out_nothing()` means the result will be neither returned nor written into a file (but computation will still be conducted).

Author(s)

Yixuan Qiu <https://statr.me>

References

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems. ACM TIST, 2015.

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. PAKDD, 2015.

W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. Technical report, 2015.

See Also

$train()

Examples

```
## Not run:
train_file = data_file(system.file("dat", "smalltrain.txt", package = "recosystem"))
test_file = data_file(system.file("dat", "smalltest.txt", package = "recosystem"))
r = Reco()
set.seed(123) # This is a randomized algorithm
opts_tune = r$tune(train_file)$min
r$train(train_file, out_model = NULL, opts = opts_tune)

## Write predicted values into file
out_pred = out_file(tempfile())
r$predict(test_file, out_pred)

## Return predicted values in memory
pred = r$predict(test_file, out_memory())

## If testing data are stored in memory
test_df = read.table(test_file@source, sep = " ", header = FALSE)
test_data = data_memory(test_df[, 1], test_df[, 2])
pred2 = r$predict(test_data, out_memory())

## Compare results
print(scan(out_pred@dest, n = 10))
head(pred, 10)
head(pred2, 10)

## If testing data are stored as a sparse matrix
if(require(Matrix))
{
    mat = Matrix::sparseMatrix(i = test_df[, 1], j = test_df[, 2], x = -1,
                               repr = "T", index1 = FALSE)
    test_data = data_matrix(mat)
    pred3 = r$predict(test_data, out_memory())
    print(head(pred3, 10))
}

## End(Not run)
```

Reco *Constructing a Recommender System Object*

Description

This function simply returns an object of class "RecoSys" that can be used to construct recommender model and conduct prediction.

Usage

```
Reco()
```

Value

`Reco()` returns an object of class "RecoSys" equipped with methods `$train()`, `$tune()`, `$output()` and `$predict()`, which describe the typical process of building and tuning model, exporting factorization matrices, and predicting results. See their help documents for details.

Author(s)

Yixuan Qiu <https://statr.me>

References

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems. ACM TIST, 2015.

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. PAKDD, 2015.

W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. Technical report, 2015.

See Also

`$tune()`, `$train()`, `$output()`, `$predict()`

train *Training a Recommender Model*

Description

This method is a member function of class "RecoSys" that trains a recommender model. It will read from a training data source and create a model file at the specified location. The model file contains necessary information for prediction.

The common usage of this method is

```
r = Reco()
r$train(train_data, out_model = file.path(tempdir(), "model.txt"),
        opts = list())
```

Arguments

r	Object returned by Reco().
train_data	An object of class "DataSource" that describes the source of training data, typically returned by function data_file(), data_memory(), or data_matrix().
out_model	Path to the model file that will be created. If passing NULL, the model will be stored in-memory, and model matrices can then be accessed under r$model$matrices.
opts	A number of parameters and options for the model training. See section **Parameters and Options** for details.

Parameters and Options

The opts argument is a list that can supply any of the following parameters:

loss Character string, the loss function. Default is "l2", see below for details.

dim Integer, the number of latent factors. Default is 10.

costp_l1 Numeric, L1 regularization parameter for user factors. Default is 0.

costp_l2 Numeric, L2 regularization parameter for user factors. Default is 0.1.

costq_l1 Numeric, L1 regularization parameter for item factors. Default is 0.

costq_l2 Numeric, L2 regularization parameter for item factors. Default is 0.1.

lrate Numeric, the learning rate, which can be thought of as the step size in gradient descent. Default is 0.1.

niter Integer, the number of iterations. Default is 20.

nthread Integer, the number of threads for parallel computing. Default is 1.

nbin Integer, the number of bins. Must be greater than nthread. Default is 20.

nmf Logical, whether to perform non-negative matrix factorization. Default is FALSE.

verbose Logical, whether to show detailed information. Default is TRUE.

The loss option may take the following values:

For real-valued matrix factorization,

"l2" Squared error (L2-norm)

"l1" Absolute error (L1-norm)

"kl" Generalized KL-divergence

For binary matrix factorization,

"log" Logarithmic error

"squared_hinge" Squared hinge loss

"hinge" Hinge loss

For one-class matrix factorization,

"row_log" Row-oriented pair-wise logarithmic loss

"col_log" Column-oriented pair-wise logarithmic loss

Author(s)

Yixuan Qiu <https://statr.me>

References

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems. ACM TIST, 2015.

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. PAKDD, 2015.

W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. Technical report, 2015.

See Also

$tune(), $output(), $predict()

Examples

```
## Training model from a data file
train_set = system.file("dat", "smalltrain.txt", package = "recosystem")
train_data = data_file(train_set)
r = Reco()
set.seed(123) # This is a randomized algorithm
# The model will be saved to a file
r$train(train_data, out_model = file.path(tempdir(), "model.txt"),
        opts = list(dim = 20, costp_l2 = 0.01, costq_l2 = 0.01, nthread = 1)
)

## Training model from data in memory
train_df = read.table(train_set, sep = " ", header = FALSE)
train_data = data_memory(train_df[, 1], train_df[, 2], rating = train_df[, 3])
set.seed(123)
# The model will be stored in memory
r$train(train_data, out_model = NULL,
        opts = list(dim = 20, costp_l2 = 0.01, costq_l2 = 0.01, nthread = 1)
)

## Training model from data in a sparse matrix
if(require(Matrix))
{
    mat = Matrix::sparseMatrix(i = train_df[, 1], j = train_df[, 2], x = train_df[, 3],
                               repr = "T", index1 = FALSE)
    train_data = data_matrix(mat)
    r$train(train_data, out_model = NULL,
            opts = list(dim = 20, costp_l2 = 0.01, costq_l2 = 0.01, nthread = 1))
}
```

tune *Tuning Model Parameters*

Description

This method is a member function of class "RecoSys" that uses cross validation to tune the model parameters.

The common usage of this method is

```
r = Reco()
r$tune(train_data, opts = list(dim      = c(10L, 20L),
                               costp_l1 = c(0, 0.1),
                               costp_l2 = c(0.01, 0.1),
                               costq_l1 = c(0, 0.1),
                               costq_l2 = c(0.01, 0.1),
                               lrate    = c(0.01, 0.1))
)
```

Arguments

r	Object returned by Reco().
train_data	An object of class "DataSource" that describes the source of training data, typically returned by function data_file(), data_memory(), or data_matrix().
opts	A number of candidate tuning parameter values and extra options in the model tuning procedure. See section **Parameters and Options** for details.

Value

A list with two components:

min Parameter values with minimum cross validated loss. This is a list that can be passed to the opts argument in $train().

res A data frame giving the supplied candidate values of tuning parameters, and one column showing the loss function value associated with each combination.

Parameters and Options

The opts argument should be a list that provides the candidate values of tuning parameters and some other options. For tuning parameters (dim, costp_l1, costp_l2, costq_l1, costq_l2, and lrate), users can provide a numeric vector for each one, so that the model will be evaluated on each combination of the candidate values. For other non-tuning options, users should give a single value. If a parameter or option is not set by the user, the program will use a default one.

See below for the list of available parameters and options:

dim Tuning parameter, the number of latent factors. Can be specified as an integer vector, with default value c(10L, 20L).

`costp_l1` Tuning parameter, the L1 regularization cost for user factors. Can be specified as a numeric vector, with default value `c(0, 0.1)`.

`costp_l2` Tuning parameter, the L2 regularization cost for user factors. Can be specified as a numeric vector, with default value `c(0.01, 0.1)`.

`costq_l1` Tuning parameter, the L1 regularization cost for item factors. Can be specified as a numeric vector, with default value `c(0, 0.1)`.

`costq_l2` Tuning parameter, the L2 regularization cost for item factors. Can be specified as a numeric vector, with default value `c(0.01, 0.1)`.

`lrate` Tuning parameter, the learning rate, which can be thought of as the step size in gradient descent. Can be specified as a numeric vector, with default value `c(0.01, 0.1)`.

`loss` Character string, the loss function. Default is "l2", see section **Parameters and Options** in `$train()` for details.

`nfold` Integer, the number of folds in cross validation. Default is 5.

`niter` Integer, the number of iterations. Default is 20.

`nthread` Integer, the number of threads for parallel computing. Default is 1.

`nbin` Integer, the number of bins. Must be greater than `nthread`. Default is 20.

`nmf` Logical, whether to perform non-negative matrix factorization. Default is `FALSE`.

`verbose` Logical, whether to show detailed information. Default is `FALSE`.

`progress` Logical, whether to show a progress bar. Default is `TRUE`.

Author(s)

Yixuan Qiu <https://statr.me>

References

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems. ACM TIST, 2015.

W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. PAKDD, 2015.

W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. Technical report, 2015.

See Also

`$train()`

Examples

```
## Not run:
train_set = system.file("dat", "smalltrain.txt", package = "recosystem")
train_src = data_file(train_set)
r = Reco()
set.seed(123) # This is a randomized algorithm
res = r$tune(
    train_src,
```

```
    opts = list(dim = c(10, 20, 30),
                costp_l1 = 0, costq_l1 = 0,
                lrate = c(0.05, 0.1, 0.2), nthread = 2)
)
r$train(train_src, opts = res$min)

## End(Not run)
```

Index