Package ‘slimrec’

March 25, 2017

Title Sparse Linear Method to Predict Ratings and Top-N Recommendations

Version 0.1.0

Description Sparse Linear Method(SLIM) predicts ratings and top-n recommendations suited for sparse implicit positive feedback systems. SLIM is decomposed into multiple elasticnet optimization problems which are solved in parallel over multiple cores. The package is based on ``SLIM: Sparse Linear Methods for Top-
N Recommender Systems'' by Xia Ning and George Karypis <doi:10.1109/ICDM.2011.134>.

Depends R (>= 3.3.3), stats (>= 3.3.3)

Imports assertthat (>= 0.1), parallel (>= 3.3.3), Matrix (>= 1.2.8), glmnet (>= 2.0.5), bigmemory (>= 4.5.19), pbapply (>= 1.3.2)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Author Srikanth KS [aut, cre]

Maintainer Srikanth KS <`sri.teach@gmail.com`>

Repository CRAN

Date/Publication 2017-03-25 09:27:15 UTC

R topics documented:

`slimrec-package` *slimrec*

Description

Sparse Linear Method to Predict Ratings and Top-N Reccomendations

Details

Sparse linear method (DOI: 10.1109/ICDM.2011.134) predicts ratings and top-n recommendations suited for sparse implicit positive feedback systems. SLIM is decomposed into multiple elasticnet optimization problems which are solved in parallel over multiple cores. The package is based on "SLIM: Sparse Linear Methods for Top-N Recommender Systems" by Xia Ning and George Karypis.

The method predicts ratings of a user for a given item as a linear combination ratings of all other items provided by the user. The coefficients for an item are determined elastic-net regression (both L1 and L2 regularization) over ratings matrix.

The optimization problem solves:

$$\min_{c_{j,.}} \frac{1}{2}\|a_{j,.} - Ac_{j,.}\|_2^2 + \frac{\beta}{2}\|c_{j,.}\|_2^2 + \gamma\|c_{j,.}\|_1$$

subject to $c_{j,j} = 0$ and optional non-negative constraint $c_{j,.} >= 0$ where $a_{j,.}$ is the j th column of the input ratings matrix and $c_{j,.}$ is the j th column of the coefficient matrix(to be determined).

The method assumes that unknown rating values to be zero. Hence, it is primarily designed for implicit feeback mechanisms, but not restricted them. The main use of the ratings is to generate top-n lists of users and items.

The package provides three functions:

- `slim`: Function to compute ratings and coefficient matrix for the sparse ratings matrix using SLIM method.
- `tune_slim`: Function to arrive at an optimal value of `alpha` for `slim`.
- `top_rows/cols`: Functions to find row/column numbers corresponding the largest values in a particular column/row of a matrix. This is helpful to generate top-n users or items as recommendations.

The package is primarily based on the wonderful package **glmnet** by Jerome Friedman, Trevor Hastie, Noah Simon, Rob Tibshirani.

If you intend to use SLIM method for large matrices(around >= 1e7 ratings), this package might be slow enough to be practically useful even in parallel mode. You might want to look at **biglasso** and other implementations like librec.

`ft_explicit` *filmtrust explicit dataset*

Description

filmtrust explicit dataset

Usage

```
ft_explicit
```

Format

A sparse matrix of class `dgCMatrix` with 1508 users and 2071 items. A (i,j) th element is rating provided by user i for item j. The possible ratings are `(0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0)`. A (i,j) th element is zero if there is no rating value. There are 35494 ratings (sparsity = 0.0114). Source: librec datasets

`ft_implicit` *filmtrust implicit dataset*

Description

filmtrust implicit dataset

Usage

```
ft_implicit
```

Format

A sparse matrix of class `dgCMatrix` with 1508 users and 2071 items. A (i,j) th element is rating provided by user i for item j. A (i,j) th element is zero if there is no rating value. There are 35494 ratings (sparsity = 0.0114). Source: librec datasets

ft_small *filmtrust small dataset*

Description

filmtrust small dataset

Usage

```
ft_small
```

Format

A sparse matrix of class `dgCMatrix` with 100 users and 1000 items. A (i,j) th element is 1 only if user i has rated item j. A (i,j) th element is zero if there is no rating value. There are 7780 ratings (sparsity = 0.778). This is a simply `ft_implicit[1:100,1:1000]`. Source: librec datasets

slim *slim*

Description

Compute ratings and coefficient matrix for the sparse ratings matrix using SLIM

Usage

```
slim(mat, alpha = 0.5, lambda, nlambda, nonNegCoeff = TRUE, directory,
  coeffMat = FALSE, returnMat = FALSE, computeRMSE = FALSE, nproc = 1L,
  progress = TRUE, check = TRUE, cleanup = FALSE)
```

Arguments

`mat`	(sparse matrix of class 'dgCMatrix') Rating matrix with items along columns and users along rows.
`alpha`	(0 <= alpha <= 1) Parameter to decide the relative weightage between the L1 and L2 penalities. See glmnet for more details. This is set by default at `0.5`.
`lambda`	(positive real number) Parameter to control shrinkage of coefficients. See glmnet for more details. Its advisable not to provide the lambda value, as the function figures out the optimal value by itself.
`nlambda`	(positive integer) Maximum length of the lambda sequence. See glmnet for more details. If `nlambda` argument is missing, it will be set to 100L. This is overridden if `lambda` is specified.

`nonNegCoeff` (flag) Whether the regression coefficients should be non-negative. There are instances where setting to FALSE decreases the RMSE, but sometimes this could lead to overfitting. Setting `nonNegCoeff` is FALSE, helps interpreting coefficients in the case of implicit feedback. This is set to TRUE by default.

`directory` (string) A writable directory where a sub-directory is created at the run time and `bigmatrix` objects will be written to. Predicted ratings data is stored in `ratingMat` file and the description is written to `ratingMat.desc` file. If `coeffMat` is TRUE, the coefficents matrix is stored in the file `coeffMat` and the description is written to `coeffMat.desc` file. When directory argument is missing, directory is set via `tempdir()`.

`coeffMat` (flag) Whether coeffMat is to be computed. This can be later used to predict recommendations for users not present in the `mat` (although `slimrec` package does not provide a `predict function`). Setting it TRUE increases the computation time. This is set to FALSE by default.

`returnMat` (flag) Whether the predicted ratings matrix and coefficient matrix (only if `coeffMat` is TRUE) to be read into memory as matrices and delete on disk `bigmatrix` objects. When output matrices are large, setting `returnMat` to TRUE is not advisable. This is set to FALSE by default.

`computeRMSE` (flag) Whether RMSE values have to be computed corresponding to non-zero values of the `mat`, both overall and columnwise.

`nproc` (positive integer) Number of parallel processes to be used to compute coefficients for items. If the machine has `k` (>1) cores, the function does not employ more than `k - 1` cores. This is set to 1L by default.

`progress` (flag) If TRUE(default), shows a progress bar and expected time. This is set to TRUE by default.

`check` (flag) If TRUE(default), ckecks like whether the matrix is sparse, matrix does not contains NAs, alpha lies between 0 and 1, directory if specified is writable and so on. This is set to TRUE by default.

`cleanup` (flag) Whether to delete the sub-directory. Note that `returnMat` cannot be set to FALSE when `cleanup` is TRUE. This is set to FALSE by default.

Details

Sparse linear method (DOI: 10.1109/ICDM.2011.134): The method predicts ratings of a user for a given item as a linear combination ratings of all other items provided by the user. The coefficients for an item are determined elastic-net regression (both L1 and L2 regularization) over ratings matrix.

The optimization problem solves:

$$\min_{c_{j,.}} \frac{1}{2}\|a_{j,.} - Ac_{j,.}\|_2^2 + \frac{\beta}{2}\|c_{j,.}\|_2^2 + \gamma\|c_{j,.}\|_1$$

subject to $c_{j,j} = 0$ and optional non-negative constraint $c_{j,.} >= 0$ where $a_{j,.}$ is the j th column of the input ratings matrix and $c_{j,.}$ is the j th column of the coefficient matrix(to be determined).

The method assumes that unknown rating values to be zero. Hence, it is primarily designed for implicit feeback mechanisms, but not restricted them. The main use of the ratings is to generate top-n lists of users and items.

Implementation: The non-negative ratings data is input as a sparse matrix of class dgCMatrix without any NA. The items should constitute columns and users should constitute rows. The elastic-net regression problem is solved using glmnet package. The coefficients for each item (a column of the ratings matrix) is computed, in parallel. To avoid memory overload, the output(s) is written to a disk based bigmatrix (using bigmemory package). The predicted rating matrix is the primary output. It is possible to obtain the matrix of coefficients, which will be helpful later to 'predict' the ratings for users not present in the ratings matrix. The RMSE may be computed itemwise and for the entire non-zero values of the ratings matrix. Since, lambda is auto-adjusted, change in alpha might not have significant impact on the RMSE. When it is necessary to get the best accuracy, there is a 'tune' function to arrive at the optimal alpha value by cross-validation. There are options to read the disk based matrix(s) into memory (as matrices) and remove the disk based ones.

Value

A list with these elements:

- ratingMat: If returnMat is TRUE, the predicted ratings matrix. Else, NULL
- coeffMat: If returnMat is TRUE and coeffMat is TRUE, the coefficient matrix. Else, NULL
- lambdas: When lambda is not specified, a vector(length of number of columns of mat) of lambda values chosen. When lambda is specified, it is singleton lambda value.
- columnwiseNonZeroRMSE: If computeRMSE is TRUE, vector of RMSE for each column. The errors are computed over only non-zero values of the column of mat. If computeRMSE is FALSE, value is set to NULL.
- nonZeroRMSE: If computeRMSE is TRUE, RMSE value. The errors are computed over only non-zero values of the mat. If computeRMSE is FALSE, value is set to NULL.
- subdir: Path to the sub-directory where output are placed.
- call: function call

Examples

```
require("slimrec")
data(ft_small)
temp <- slim(ft_small)
str(temp)

## Not run:
temp <- slim(mat          = ft_implicit # input sparse ratings matrix
             , alpha       = 0.5         # 0 for ridge, 1 for lasso
             #, lambda                   # suggested not to set lambda
             #, nlambda                  # using default nlambda = 100
             , nonNegCoeff = TRUE        # better accuracy, lower interpretability
             , directory   = td          # dir where output matrices are stored
             , coeffMat    = TRUE        # helpful in 'predict'ing later
             , returnMat   = TRUE        # return matrices in memory
             , computeRMSE = TRUE        # RMSE over rated items
             , nproc       = 2L          # number of concurrent processes
             , progress    = TRUE        # show a progressbar
             , check       = TRUE        # do basic checks on input params
             , cleanup     = FALSE       # keep output matrices on disk
```

```
              )
str(temp)
# output ratings matrix would be comparatively denser
predMat <- temp[["ratingMat"]] != 0
sum(predMat)/((dim(predMat)[1])*(dim(predMat)[2]))
# recommend top 5 items for a user 10
top_cols(temp[["ratingMat"]]
         , row = 10
         , k   = 5
         )
# if you intend to avoid recommending 10, 215 and 3
top_cols(temp[["ratingMat"]]
         , row = 10
         , k   = 5
         , ignore = c(10, 215, 3)
         )

## End(Not run)
```

`top_cols` *top_cols*

Description

Find the column numbers corresponding the largest values in a particular row of a matrix

Usage

```
top_cols(mat, row, k = 5, ignore)
```

Arguments

`mat`	(Sparse matrix of class 'dgCmatrix' or a integer/numeric matrix or 'big.matrix') Rating matrix.
`row`	(positive integer) Row number in which top columns are to be selected.
`k`	(positive integer) Number of column numbers to be recommended. This might not be strictly adhered to, see Details.
`ignore`	(integer vector) Column numbers to be ignored.

Details

To find top-n recommendations of a ratings matrix given an item (or a user). Although k recommendations are expected to be returned, the function might sometimes return more or less than k recommendations.

- Less: This happens when it is not possible to recommend k elements. For example, k is larger than the number of elements.

- More: This happens when a few elements have same rating. The function returns the index corresponding to all the elements which have the same rating. If ratings were `3,2,2,2,3`:
  - k = 3: returns `1, 5, 2, 3, 4`
  - k = 2: returns `1, 5`
  - k = 1: returns `1, 5`

Examples

```
## Not run:
temp <- slim(mat           = ft_implicit # input sparse ratings matrix
             , alpha       = 0.5         # 0 for ridge, 1 for lasso
             #, lambda                   # suggested not to set lambda
             #, nlambda                  # using default nlambda = 100
             , nonNegCoeff = TRUE        # better accuracy, lower interpretability
             , directory   = td          # dir where output matrices are stored
             , coeffMat    = TRUE        # helpful in 'predict'ing later
             , returnMat   = TRUE        # return matrices in memory
             , computeRMSE = TRUE        # RMSE over rated items
             , nproc       = 2L          # number of concurrent processes
             , progress    = TRUE        # show a progressbar
             , check       = TRUE        # do basic checks on input params
             , cleanup     = FALSE       # keep output matrices on disk
             )
str(temp)
# output ratings matrix would be comparatively denser
predMat <- temp[["ratingMat"]] != 0
sum(predMat)/((dim(predMat)[1])*(dim(predMat)[2]))
# recommend top 5 items for a user 10
top_cols(temp[["ratingMat"]]
         , row = 10
         , k   = 5
         )
# if you intend to avoid recommending 10, 215 and 3
top_cols(temp[["ratingMat"]]
         , row = 10
         , k   = 5
         , ignore = c(10, 215, 3)
         )

## End(Not run)
```

`top_rows` *top_rows*

Description

Find the row numbers corresponding the largest values in a particular column of a matrix

Usage

```
top_rows(mat, col, k = 5, ignore)
```

Arguments

`mat`	(Sparse matrix of class 'dgCmatrix' or a integer/numeric matrix or 'big.matrix') Rating matrix.
`col`	(positive integer) Column number in which top rows are to be selected.
`k`	(positive integer) Number of row numbers to be recommended. This might not be strictly adhered to, see Details.
`ignore`	(integer vector) Row numbers to be ignored.

Details

To find top-n recommendations of a ratings matrix given an item (or a user). Although k recommendations are expected to be returned, the function might sometimes return more or less than k recommendations.

- Less: This happens when it is not possible to recommend k elements. For example, k is larger than the number of elements.
- More: This happens when a few elements have same rating. The function returns the index corresponding to all the elements which have the same rating. If ratings were `3,2,2,2,3`:
  - k = 3: returns `1, 5, 2, 3, 4`
  - k = 2: returns `1, 5`
  - k = 1: returns `1, 5`

Examples

```
## Not run:
temp <- slim(mat           = ft_implicit # input sparse ratings matrix
             , alpha       = 0.5         # 0 for ridge, 1 for lasso
             #, lambda                   # suggested not to set lambda
             #, nlambda                  # using default nlambda = 100
             , nonNegCoeff = TRUE        # better accuracy, lower interpretability
             , directory   = td          # dir where output matrices are stored
             , coeffMat    = TRUE        # helpful in 'predict'ing later
             , returnMat   = TRUE        # return matrices in memory
             , computeRMSE = TRUE        # RMSE over rated items
             , nproc       = 2L          # number of concurrent processes
             , progress    = TRUE        # show a progressbar
             , check       = TRUE        # do basic checks on input params
             , cleanup     = FALSE       # keep output matrices on disk
             )
str(temp)
# output ratings matrix would be comparatively denser
predMat <- temp[["ratingMat"]] != 0
sum(predMat)/((dim(predMat)[1])*(dim(predMat)[2]))
# recommend top 5 items for a user 10
top_cols(temp[["ratingMat"]]
         , row = 10
         , k   = 5
         )
# if you intend to avoid recommending 10, 215 and 3
top_cols(temp[["ratingMat"]]
         , row = 10
         , k   = 5
         , ignore = c(10, 215, 3)
         )

## End(Not run)
```

`tune_slim` *tune_slim*

Description

Arrive at an optimal value of `alpha` for `slim`

Usage

```
tune_slim(mat, alphaRange = seq(0, 1, 0.1), nonNegCoeff = TRUE,
  nfold = 5L, seed, directory, nproc = 1L, progress = TRUE)
```

Arguments

`mat` (sparse matrix of class 'dgCMatrix') Rating matrix with items along columns and users along rows.

`alphaRange` (numeric vector) A vector of alpha values with 0 <= alpha <= 1. Default is values 0 to 1, with a difference of 0.1.

`nonNegCoeff` (flag) Whether the regression coefficients should be non-negative. Default is TRUE.

`nfold` (positive integer) Number of folds for cross-validation. Only values between(inclusive) 2 and 10 are allowed.

`seed` (positive integer) Seed to be used to create folds for cross-validation. If missing, a random integer is chosen. Setting this is helpful for reproduciing the results. Default is 5.

`directory` (string) A writable directory where a sub-directory is created at the run time and `bigmatrix` objects will be written to. The sub-directories are deleted at the end. If missing, this is set using `tempdir()`

`nproc` (positive integer) Number of parallel processes to be used to compute coefficients for items. If the machine has `k` (>1) cores, the function does not employ more than `k - 1` cores. This is set to 1L by default.

`progress` (flag) If TRUE(default), shows a progress bar and expected time. This is set to TRUE by default.

Details

Runs nfold cross-validation to aid determining the optimal value of alpha(see slim for details). The coefficient matrix obtained from the training fold is used to predict ratings of the validation fold. The RMSE is evaluated for non-zero ratings and averaged over all the folds. Note that coefficient matrix is held in memory while computing RMSE. slim adjusts lambda while fitting an elastic-net, hence advantages in searching for optimal alpha might be limited.

Value

A dataframe with two columns: alpha and error.

Examples

```
require("slimrec")
data(ft_small)
## Not run:
temp <- tune_slim(ft_small)
temp
temp <- tune_slim(ft_small, alphaRange = c(0, 0.5, 1))
temp
temp <- tune_slim(ft_small, alphaRange = c(0, 0.5, 1), nproc = 2)
temp
temp <- tune_slim(ft_small, nonNegCoeff = FALSE)
temp

## End(Not run)
```

Index