Package 'poismf'

May 27, 2020

Type Package

Title Factorization of Sparse Counts Matrices Through Poisson
Likelihood

Version 0.2.0

Date 2020-05-23

Author David Cortes

Maintainer David Cortes <david.cortes.rivera@gmail.com>

URL https://github.com/david-cortes/poismf

BugReports https://github.com/david-cortes/poismf/issues

Description Creates a low-rank factorization of a sparse counts matrix by maximizing Poisson likeli-
hood with l1/l2 regularization
with all non-negative latent factors (e.g. for recommender systems or topic model-
ing) (Cortes, (2018) <arXiv:1811.01908>).
Similar to hierarchical Poisson factorization, but follows an optimization-
based approach with regularization instead of a
hierarchical structure, and is fit through either proximal gradient or conjugate gradient instead of
variational inference.

License BSD_2_clause + file LICENSE

Imports Matrix, methods

Enhances SparseM

RoxygenNote 7.1.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-05-26 22:10:03 UTC

R topics documented:

---

calc.all.counts   *Predict whole input matrix*

Description

Outputs the predictions for the whole input matrix to which the model was fit. Note that this will be a dense matrix, and in typical recommender systems scenarios will likely not fit in memory.

Usage

```
calc.all.counts(model)
```

Arguments

model   A Poisson factorization model as output by function 'poismf'.

Value

A matrix 'dimA' x 'dimB' with the full predictions for all rows and column. If the inputs did not have numbers as IDs, the equivalences to their IDs in the outputs are in the 'model' object under fields 'levels_A' and 'levels_B', and can also be obtained through function get.model.mappings.

---

evaluate.poisson.llk   *Evaluate Poisson log-likelihood for counts matrix*

Description

Calculates Poisson log-likehood plus constant for new combinations of rows and columns of 'X'. Intended to use as a test metric or for monitoring a validation set.

By default, this Poisson log-likelihood is calculated only for the combinations of users (rows) and items (columns) provided in 'X_test' here, ignoring the missing entries. This is the usual use-case for evaluating a validation or test set, but can also be used for evaluating it on the training data with all missing entries included as zeros (see parameters for details).

Note that this calculates a **sum** rather than an average.

Usage

```
evaluate.poisson.llk(model, X_test, full_llk = FALSE, include_missing = FALSE)
```

Arguments

model	A Poisson factorization model object as returned by 'poismf'.
X_test	Input data on which to calculate log-likelihood, consisting of triplets. Can be passed as a 'data.frame' or as a sparse COO matrix (see documentation of [po-ismf](#) for details on the accepted data types). If the 'X' data passed to 'poismf' was a 'data.frame', should pass a 'data.frame' with entries corresponding to the same IDs, otherwise might pass either a 'data.frame' with the row and column indices (starting at 1), or a sparse COO matrix.
full_llk	Whether to add to the number a constant given by the data which doesn't depend on the fitted parameters. If passing 'False' (the default), there is some chance that the resulting log-likelihood will end up being positive - this is not an error, but is simply due to ommission of this constant. Passing 'TRUE' might result in numeric overflow and low numerical precision.
include_missing	
	If 'TRUE', will calculate the Poisson log-likelihood for all entries (i.e. all combinations of users/items, whole matrix 'X'), taking the missing ones as zero-valued. If passing 'FALSE', will calculate the Poisson log-likelihood only for the non-missing entries passed in 'X_test' - this is usually the desired behavior when evaluating a test dataset.

Details

If using more than 1 thread, the results might vary slightly between runs.

Value

Obtained Poisson log-likelihood (higher is better).

See Also

[poismf](#)

---

factors	*Determine latent factors for new rows/users*

Description

Determines the latent factors for new users (rows) given their counts for existing items (columns).

This function will use the same method and hyperparameters with which the model was fit. If using this for recommender systems, it's recommended to use instead the function [factors.single](#), even though the new factors obtained with that function might not end up being in the same scale as the original factors in 'model$A'.

Note that, when using proximal gradient method (the default), results from this function and from 'get.factor.matrices' on the same data might differ a lot. If this is a problem, it's recommended to use conjugate gradient instead.

Usage

```
factors(model, X, add_names = TRUE)
```

Arguments

model   A Poisson factorization model as returned by 'poismf'.

X   New data for whose rows to determine latent factors. Can be passed as a 'data.frame'
               or as a sparse or dense matrix (see documentation of poismf for details on the
               data type). While other functions only accept sparse matrices in COO (triplets)
               format, this function will also take CSR matrices from the 'SparseM' package
               (recommended) and CSC matrices from the 'Matrix' package ('Matrix::dgCMatrix',
               but it's not recommended). Inputs will be converted to CSR regardless of their
               original format.

               If passing a 'data.frame', the first column should contain row indices or IDs,
               and these will be internally remapped - the mapping will be available as the row
               names for the matrix if passing 'add_names=TRUE', or as part of the outputs
               if passing 'add_names=FALSE'. The IDs passed in the first column will not be
               matched to the existing IDs of 'X' passed to 'poismf'.

               If 'X' passed to 'poismf' was a 'data.frame', 'X' here must also be passed as
               'data.frame'. If 'X' passed to 'poismf' was a matrix and 'X' is a 'data.frame',
               the second column of 'X' here should contain column numbers (with numeration
               starting at 1).

add_names   Whether to add row names to the output matrix if the indices were internally
               remapped - they will only be so if the 'X' here is a 'data.frame'. Note that if the
               indices in passed in 'X' here (first and second columns) are integers, once row
               names are added, subsetting 'X' by an integer will give the row at that position
               - that is, if you want to obtain the corresponding row for ID=2 from 'X' in
               'A_out', you need to use 'A_out["2",]', not 'A_out[2,]'.

Details

The factors are initialized to the mean of each column in the fitted model.

Value

- If 'X' was passed as a matrix, will output a matrix of dimensions (n, k) with the obtained
  factors. If passing 'add_names=TRUE' and 'X' passed to 'poismf' was a 'data.frame', this
  matrix will have row names. **Careful with subsetting with integers** (see documentation for
  'add_names').

- If 'X' was passed as a 'data.frame' and passing 'add_names=FALSE' here, will output a list
  with an entry 'factors' containing the latent factors as described above, and an entry 'mapping'
  indicating to which row ID does each row of the output correspond.

See Also

[factors.single topN.new](#)

---

factors.single   *Get latent factors for a new user given her item counts*

Description

This is similar to obtaining topics for a document in LDA. See also function [factors](#) for getting factors for multiple users/rows at a time.

This function works with one user at a time, and will use a conjugate gradient approach regardless of how the model was fit. As well, it will use the regularization parameter passed here instead of the original one from the model, which means the obtained factors might not end up being in the same scale as the original ones which are stored under 'model$A'. See function [factors](#) for a different approach using the same method with which the model was fit.

Be aware that the proximal gradient method, which is the default for fitting the model and which doesn't try to reach global optima, requires larger regularization, whereas the conjugate gradient method, which tries to find the global optimum, will fail to fit with too larger regularization (i.e. the optimal will be all-zeros).

Usage

```
factors.single(
  model,
  X,
  l2_reg = 1e+05,
  l1_reg = 0,
  nupd = 100,
  weight_mult = 1
)
```

Arguments

model	Poisson factorization model as returned by 'poismf'.
X	Data with the non-zero item indices and counts for this new user. Can be passed as a sparse vector from package 'Matrix' ('Matrix::dsparseVector', which can be created from indices and values through function 'Matrix::sparseVector'), or as a 'data.frame', in which case will take the first column as the item/column indices (numeration starting at 1) and the second column as the counts. If 'X' passed to 'poismf' was a 'data.frame', 'X' here must also be a 'data.frame'.
l2_reg	Strength of L2 regularization to use for optimizing the new factors. Note that these are obtained through a conjugate-gradient method instead of proximal gradient, which works better with smaller regularization values.
l1_reg	Strength of the L1 regularization (see description of argument above). Not recommended.

nupd	Maximum number of conjugate gradient updates.
weight_mult	Weight multiplier for the positive entries over the missing entries. Not recommended.

Details

The factors are initialized to the mean of each column in the fitted model.

Value

Vector of dimensionality 'model$k' with the latent factors for the user, given the input data.

See Also

[factors topN.new](#)

---

get.factor.matrices   *Extract Latent Factor Matrices*

Description

Extract the latent factor matrices for users (rows) and columns (items) from a Poisson factorization model object, as returned by function 'poismf'.

Usage

```
get.factor.matrices(model, add_names = TRUE)
```

Arguments

model	A Poisson factorization model, as produced by 'poismf'.
add_names	Whether to add row names to the matrices if the indices were internally remapped - they will only be so if the 'X' passed to 'poismf' was a 'data.frame'. Note that if passing 'X' as 'data.frame' with integer indices to 'poismf', once row names are added, subsetting such matrix by an integer will give the row at that position - that is, if you want to obtain the corresponding row for ID=2 from 'X' in 'factors$A', you need to use 'factors$A["2",]', not 'factors$A[2,]'.

Details

If 'X' passed to 'poismf' was a 'data.frame', the mapping between IDs from 'X' to row numbers in 'A' and column numbers in 'B' are avaiable under 'model$levels_A' and 'model$levels_B', respectively. They can also be obtained through 'get.model.mappings', and will be added as row names if using 'add_names=TRUE'. **Be careful about subsetting with integers** (see documentation for 'add_names' for details).

Value

List with entries 'A' (the user factors) and 'B' (the item factors).

See Also

[get.model.mappings](#)

---

get.model.mappings   *Extract user/row and item/column mappings from Poisson model.*

Description

Will extract the mapping between IDs passed as 'X' to function 'poismf' and row/column positions in the latent factor matrices and prediction functions.

Such a mapping will only be generated if the 'X' passed to 'poismf' was a 'data.frame', otherwise they will not be re-mapped.

Usage

```
get.model.mappings(model)
```

Arguments

model   A Poisson factorization model as returned by 'poismf'.

Value

A list with row entries:

- 'rows': a vector in which each user/row ID is placed at its ordinal position in the internal data structures. If there is no mapping (e.g. if 'X' passed to 'poismf' was a sparse matrix), will be 'NULL'.

- 'columns': a vector in which each item/column ID is placed at its ordinal position in the internal data structures. If there is no mapping (e.g. if 'X' passed to 'poismf' was a sparse matrix), will be 'NULL'.

See Also

[get.factor.matrices](#)

---

poismf   *Factorization of Sparse Counts Matrices through Poisson Likelihood*

Description

Creates a low-rank non-negative factorization of a sparse counts matrix by maximizing Poisson likelihood minus L1/L2 regularization, using optimization routines based on either proximal gradient or conjugate gradient.

Ideal for usage in recommender systems, in which the 'X' matrix would consist of interactions (e.g. clicks, views, plays), with users representing the rows and items representing the columns.

Be aware that this model is prone to numerical instability when using proximal gradient, and can turn out to spit all NaNs or zeros in the fitted parameters. As an alternative, can use a conjugate gradient approach instead, which is slower but not susceptible to such failures.

The default hyperparameters are geared towards very fast fitting times, and might not be very competitive against other implicit-feedback methods when used as-is. It's also possible to obtain better quality models that compare very favorably against e.g. implicit-ALS/HPF/BPR/etc. by using a much larger number of iterations and updates, lower regularization, and the conjugate gradient option - this will take many times longer than with the default hyperparameters, but usually it's still faster than other factorization models.

Usage

```
poismf(
  X,
  k = 30,
  use_cg = FALSE,
  l2_reg = ifelse(use_cg, 1e+05, 1e+09),
  l1_reg = 0,
  niter = 10,
  nupd = ifelse(use_cg, 5, 1),
  initial_step = 1e-07,
  weight_mult = 1,
  init_type = "gamma",
  seed = 1,
  nthreads = parallel::detectCores()
)
```

Arguments

X   The counts matrix to factorize. Can be:

- A 'data.frame' with 3 columns, containing in this order: row index or user ID, column index or item ID, count value. The first two columns will be converted to factors to enumerate them internally, and will return those same values from 'topN'. In order to avoid this internal re-enumeration, can pass 'X' as a sparse COO matrix instead.

- A sparse matrix from package 'Matrix' in triplets (COO) format (that is: 'Matrix::dgTMatrix') (recommended). Such a matrix can be created from row/column indices through function 'Matrix::sparseMatrix' (with 'giveCsparse=FALSE'). Will also accept them in CSC format ('Matrix::dgCMatrix'), but will be converted along the way (so it will be slightly slower).
- A sparse matrix in COO format from the 'SparseM' package. Such a matrix can be created from row/column indices through 'new("matrix.coo", ra=values, ja=col_ix, ia=row_ix, dim=as.integer(c(m,n)))'. Will also accept them in CSR and CSC format, but will be converted along the way, (so it will be slightly slower).
- A full matrix (of class 'base::matrix') - this is not recommended though.

Passing sparse matrices is faster as it will not need to re-enumerate the rows and columns, but if passing a sparse matrix, it cannot have any row or column with all-zero values, otherwise the optimization procedure might fail. Full matrices will be converted to sparse.

k	Number of latent factors to use (dimensionality of the low-rank factorization). Note that since this model deals with non-negative latent factors only and the optimal number for every entry is a very small number (depending on sparsity and regularization), the optimal 'k' is likely to be low, while passing large values (e.g. > 100) is likely to produce bad results. If passing large 'k', it's recommended to use 'use_cg=TRUE'.
use_cg	Whether to fit the model through conjugate gradient method. This is slower, but less prone to failure, usually reaches better local optima, and is able to fit models with lower regularization values.
l2_reg	Strength of L2 regularization. Proximal gradient method will likely fail to fit when the regularization is too small, and the recommended value is 10^9. Recommended to decrease it when using conjugate gradient method to e.g. 10^5 or even zero.
l1_reg	Strength of L1 regularization. Not recommended.
niter	Number of alternating iterations to perform. One iteration denotes an update over both matrices.
nupd	• When using proximal gradient, this is the number of proximal gradient updates to perform to each vector per iteration. Increasing the number of iterations instead of this has the same computational complexity and is likely to produce better results.
	• When using conjugate gradient, this is the maximum number of updates per iteration, and it's recommended to set it to a larger value such as 5 or 10, while perhaps decreasing 'niter' (for faster fitting), as it will perform a line search in this case. Alternatively, might instead set 'nupd' to 1 (in which case it becomes gradient descent) and 'niter' to a large number such as 100. If using large 'k', and/or l1 regularization, it's recommended to increase 'nupd' due to the way constraints are handled in the CG method (see reference [2] for details) - e.g. if using 'k=100', set 'nupd=25', and maybe also 'niter=50' too.
initial_step	Initial step size to use. Larger step sizes reach converge faster, but are more likely to result in failed optimization. Ignored for conjugate gradient as it uses a line search instead.

weight_mult	Extra multiplier for the weight of the positive entries over the missing entries in the matrix to factorize. Be aware that Poisson likelihood will implicitly put more weight on the non-missing entries already. Passing larger values will make the factors have larger values (which might be desirable), and can help with instability and failed optimization cases. If passing this, it's recommended to try very large values (e.g. 10^2), and might require adjusting the other hyperparameters. Not recommended.
init_type	How to initialize the model parameters. One of ''gamma'' (will initialize them '~ Gamma(1, 1))' or ''unif'' (will initialize them '~ Unif(0, 1))'..
seed	Random seed to use for starting the factorizing matrices.
nthreads	Number of parallel threads to use.

Value

An object of class 'poismf' with the following fields of interest:

Fields

A   The user/document/row-factor matrix (as a vector in row-major order, has to be reshaped to (k, nrows) and then transposed to obtain an R matrix).

B   The item/word/column-factor matrix (as a vector in row-major order, has to be reshaped to (k, ncols) and then transposed to obtain an R matrix).

levels_A   A vector indicating which user/row ID corresponds to each row position in the 'A' matrix. This will only be generated when passing 'X' as a 'data.frame', otherwise will not remap them.

levels_B   A vector indicating which item/column ID corresponds to each row position in the 'B' matrix. This will only be generated when passing 'X' as a 'data.frame', otherwise will not remap them.

References

- Cortes, David. "Fast Non-Bayesian Poisson Factorization for Implicit-Feedback Recommendations." arXiv preprint arXiv:1811.01908 (2018).
- Li, Can. "A conjugate gradient type method for the nonnegative constraints optimization problems." Journal of Applied Mathematics 2013 (2013).

See Also

predict.poismf topN factors get.factor.matrices get.model.mappings

Examples

```
library(poismf)

### create a random sparse data frame in COO format
nrow <- 10^2 ## <- users
ncol <- 10^3 ## <- items
nnz  <- 10^4 ## <- events (agg)
```

```
set.seed(1)
X <- data.frame(
        row_ix = sample(nrow, size=nnz, replace=TRUE),
        col_ix = sample(ncol, size=nnz, replace=TRUE),
        count  = rpois(nnz, 1) + 1
    )
X <- X[!duplicated(X[, c("row_ix", "col_ix")]), ]

### can also pass X as sparse matrix - see below
### X <- Matrix::sparseMatrix(
###         i=X$row_ix, j=X$col_ix, x=X$count,
###         giveCsparse=FALSE)
### the indices can also be characters or other types:
### X$row_ix <- paste0("user", X$row_ix)
### X$col_ix <- paste0("item", X$col_ix)

### factorize the randomly-generated sparse matrix
### good speed (proximal gradient)
model <- poismf(X, nthreads=1)

### good quality, but slower (conjugate gradient)
model <- poismf(X, use_cg=TRUE, nthreads=1)

### good quality, but much slower (gradient descent)
model <- poismf(X, use_cg=TRUE, niter=100, nupd=1, nthreads=1)

### predict functionality (chosen entries in X)
### predict entry [1, 10] (row 1, column 10)
predict(model, 1, 10)
### predict entries [1,4], [1,5], [1,6]
predict(model, c(1, 1, 1), c(4, 5, 6))

### ranking functionality (for recommender systems)
topN(model, user=2, n=5, exclude=X$col_ix[X$row_ix==2])
topN.new(model, X=X[X$row_ix==2, c("col_ix","count")],
    n=5, exclude=X$col_ix[X$row_ix==2])

### obtaining latent factors
a_vec  <- factors.single(model,
            X[X$row_ix==2, c("col_ix","count")])
A_full <- factors(model, X)
A_orig <- get.factor.matrices(model)$A

### (note that newly-obtained factors will differ slightly)
sqrt(mean((A_full["2",] - A_orig["2",])^2))
```

---

poismf_unsafe   *Poisson factorization with no input casting*

Description

This is a faster version of [poismf](#) which will not make any checks or castings on its inputs. It is intended as a fast alternative when a model is to be fit multiple times with different hyperparameters, and for allowing custom-initialized factor matrices. **Note that since it doesn't make any checks or conversions, passing the wrong kinds of inputs or passing inputs with mismatching dimensions will crash the R process**.

For most use cases, it's recommended to use the function 'poismf' instead.

Usage

```
poismf_unsafe(A, B, Xcsr, Xcsc, k, ...)
```

Arguments

A	Initial values for the user-factor matrix of dimensions [dimA, k], assuming row-major order. Can be passed as a vector of dimension [dimA*k], or as a matrix of dimension [k, dimA]. Note that R matrices use column-major order, so if you want to pass an R matrix as initial values, you'll need to transpose it, hence the shape [k, dimA]. Recommended to initialize '~ Gamma(1,1)'. **Will be modified in-place**.
B	Initial values for the item-factor matrix of dimensions [dimB, k]. See documentation about 'A' for more details.
Xcsr	The transpose of the 'X' matrix in CSC format (so that its structure would match a CSR matrix). Should be an object of class 'Matrix::dgCMatrix'.
Xcsc	The 'X' matrix in CSC format. Should be an object of class 'Matrix::dgCMatrix'.
k	The number of latent factors. **Must match with the dimension of 'A' and 'B'**.
...	Other hyperparameters that can be passed to 'poismf'. See the documentation for [poismf](#) for details about possible hyperparameters. Init type and seed are ignored as the 'A' and 'B' matrices are supposed to be passed already-initialized.

Value

A 'poismf' model object. See the documentation for [poismf](#) for details.

See Also

[poismf](#)

---

predict.poismf	*Predict expected count for new row(user) and column(item) combinations*

Description

Predict expected count for new row(user) and column(item) combinations

Usage

```
## S3 method for class 'poismf'
predict(object, a, b = NULL, ...)
```

Arguments

object	A Poisson factorization model as returned by 'poismf'.
a	Can be either:

- A vector of length N with the users/rows to predict - each entry will be matched to the corresponding entry at the same position in 'b' - e.g. to predict value for entries (3,4), (3,5), and (3,6), should pass 'a=c(3,3,3), b=c(3,5,6)'. If 'X' passed to 'poismf' was a 'data.frame', should match with the entries in its first column. If 'X' passed to 'poismf' was a matrix, should indicate the row numbers (numeration starting at 1).
- A sparse matrix, ideally in COO (triplets) format from package 'Matrix' ('Matrix::dgTMatrix') or from package 'SparseM' ('matrix.coo'), in which case it will make predictions for the non-zero entries in the matrix and will output another sparse matrix with the predicted entries as values. In this case, 'b' should not be passed. This option is not available if the 'X' passed to 'poismf' was a 'data.frame'.

b	A vector of length N with the items/columns to predict - each entry will be matched to the corresponding entry at the same position in 'a' - e.g. to predict value for entries (3,4), (3,5), and (3,6), should pass 'a=c(3,3,3), b=c(3,5,6)'. If 'X' passed to 'poismf' was a 'data.frame', should match with the entries in its second column. If 'X' passed to 'poismf' was a matrix, should indicate the column numbers (numeration starting at 1). If 'a' is a sparse matrix, should not pass 'b'.
...	Not used.

Value

- If 'a' and 'b' were passed, will return a vector of length N with the predictions for the requested row/column combinations.
- If 'b' was not passed, will return a sparse matrix with the same entries and shape as 'a', but with the values being the predictions from the model for the non-missing entries.

See Also

[poismf topN factors](#)

| `print.poismf` | *Get information about poismf object* |

Description

Print basic properties of a "poismf" object.

Usage

```
## S3 method for class 'poismf'
print(x, ...)
```

Arguments

x	An object of class "poismf" as returned by function "poismf".
...	Extra arguments (not used).

| `summary.poismf` | *Get information about poismf object* |

Description

Print basic properties of a "poismf" object (same as 'print.poismf' function).

Usage

```
## S3 method for class 'poismf'
summary(object, ...)
```

Arguments

object	An object of class "poismf" as returned by function "poismf".
...	Extra arguments (not used).

See Also

[print.poismf](#)

---

topN *Rank top-N highest-predicted items for an existing user*

Description

Rank top-N highest-predicted items for an existing user

Usage

```
topN(model, user, n = 10, include = NULL, exclude = NULL, output_score = FALSE)
```

Arguments

model	A Poisson factorization model as returned by 'poismf'.
user	User for which to rank the items. If 'X' passed to 'poismf' was a 'data.frame', must match with the entries in its first column, otherwise should match with the rows of 'X' (numeration starting at 1).
n	Number of top-N highest-predicted results to output.
include	List of items which will be ranked. If passing this, will only make a ranking among these items. If 'X' passed to 'poismf' was a 'data.frame', must match with the entries in its second column, otherwise should match with the columns of 'X' (numeration starting at 1). Can only pass one of 'include' or 'exclude.' Must not contain duplicated entries.
exclude	List of items to exclude from the ranking. If passing this, will rank all the items except for these. If 'X' passed to 'poismf' was a 'data.frame', must match with the entries in its second column, otherwise should match with the columns of 'X' (numeration starting at 1). Can only pass one of 'include' or 'exclude'. Must not contain duplicated entries.
output_score	Whether to output the scores in addition to the IDs. If passing 'FALSE', will return a single array with the item IDs, otherwise will return a tuple with the item IDs and the scores.

Value

- If passing 'output_score=FALSE' (the default), will return a vector of size 'n' with the top-N highest predicted items for this user.If the 'X' data passed to 'poismf' was a 'data.frame', will contain the item IDs from its second column, otherwise will be integers matching to the columns of 'X' (starting at 1). If 'X' was passed as 'data.frame', the entries in this vector might be coerced to character regardless of their original type.

- If passing 'output_score=TRUE', will return a list, with the first entry being the vector described above under name 'ix', and the second entry being the associated scores, as a numeric vector of size 'n'.

See Also

[topN.new](#) [predict.poismf](#) [factors.single](#)

| topN.new | *Rank top-N highest-predicted items for a new user* |

Description

Rank top-N highest-predicted items for a new user

Usage

```
topN.new(
  model,
  X,
  n = 10,
  include = NULL,
  exclude = NULL,
  output_score = FALSE,
  l2_reg = 1e+05,
  l1_reg = 0,
  nupd = 50
)
```

Arguments

model	A Poisson factorization model as returned by 'poismf'.
X	Data with the non-zero item indices and counts for this new user. Can be passed as a sparse vector from package 'Matrix' ('Matrix::dsparseVector', which can be created from indices and values through 'Matrix::sparseVector'), or as a 'data.frame', in which case will take the first column as the item/column indices (numeration starting at 1) and the second column as the counts. If 'X' passed to 'poismf' was a 'data.frame', 'X' here must also be a 'data.frame'.
n	Number of top-N highest-predicted results to output.
include	List of items which will be ranked. If passing this, will only make a ranking among these items. If 'X' passed to 'poismf' was a 'data.frame', must match with the entries in its second column, otherwise should match with the columns of 'X' (numeration starting at 1). Can only pass one of 'include' or 'exclude.' Must not contain duplicated entries.
exclude	List of items to exclude from the ranking. If passing this, will rank all the items except for these. If 'X' passed to 'poismf' was a 'data.frame', must match with the entries in its second column, otherwise should match with the columns of 'X' (numeration starting at 1). Can only pass one of 'include' or 'exclude'. Must not contain duplicated entries.
output_score	Whether to output the scores in addition to the IDs. If passing 'FALSE', will return a single array with the item IDs, otherwise will return a tuple with the item IDs and the scores.

l2_reg	Strength of L2 regularization to use for optimizing the new factors. Note that these are obtained through a conjugate-gradient method instead of proximal gradient, which works better with smaller regularization values.
l1_reg	Strength of the L1 regularization (see description of argument above). Not recommended.
nupd	Maximum number of conjugate gradient updates.

Details

This function calculates the latent factors in the same way as 'factors.single' - see the documentation of factors.single for details.

The factors are initialized to the mean of each column in the fitted model.

Value

- If passing 'output_score=FALSE' (the default), will return a vector of size 'n' with the top-N highest predicted items for this user.If the 'X' data passed to 'poismf' was a 'data.frame', will contain the item IDs from its second column, otherwise will be integers matching to the columns of 'X' (starting at 1). If 'X' was passed as 'data.frame', the entries in this vector might be coerced to character regardless of their original type.

- If passing 'output_score=TRUE', will return a list, with the first entry being the vector described above under name 'ix', and the second entry being the associated scores, as a numeric vector of size 'n'.

See Also

factors.single topN

Index