Package: CMF (via r-universe)

November 3, 2024

Type Package

Title Collective Matrix Factorization

Version 1.0.3.99

Description Collective matrix factorization (CMF) finds joint low-rank representations for a collection of matrices with shared row or column entities. This code learns a variational Bayesian approximation for CMF, supporting multiple likelihood potentials and missing data, while identifying both factors shared by multiple matrices and factors private for each matrix. For further details on the method see Klami et al. (2014) <arXiv:1312.5921>. The package can also be used to learn Bayesian canonical correlation analysis (CCA) and group factor analysis (GFA) models, both of which are special cases of CMF. This is likely to be useful for people looking for CCA and GFA solutions supporting missing data and non-Gaussian likelihoods. See Klami et al. (2013) <https://research.cs.aalto.fi/pml/online-papers/klami13a.pdf> and Virtanen et al. (2012) <http://proceedings.mlr.press/v22/virtanen12.html> for details on Bayesian CCA and GFA, respectively.

License GPL (>= 2)

Imports stats

LinkingTo cpp11

Encoding UTF-8

Language en-US

NeedsCompilation yes

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

URL https://github.com/cyianor/CMF

BugReports https://github.com/cyianor/CMF/issues

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

RemoteUrl https://github.com/cyianor/cmf

RemoteRef HEAD

RemoteSha 93d52dbd4a6d7edc5b7c43be823d82546894c49f

Contents

CMF-package 2
CMF 4
getCMFopts 6
matrix_to_triplets 8
predictCMF 9
p_check_sparsity 10
triplets_to_matrix 10

Index **12**

`CMF-package` *Collective Matrix Factorization (CMF)*

Description

Collective matrix factorization (CMF) finds joint low-rank representations for a collection of matrices with shared row or column entities. This package learns a variational Bayesian approximation for CMF, supporting multiple likelihood potentials and missing data, while identifying both factors shared by multiple matrices and factors private for each matrix.

Details

This package implements a variational Bayesian approximation for CMF, following the presentation in "Group-sparse embeddings in collective matrix factorization" (see references below).

The main functionality is provided by the function `CMF()` that is used for learning the model, and by the function `predictCMF()` that estimates missing entries based on the learned model. These functions take as input lists of matrices in a specific sparse format that stores only the observed entries but that explicitly stores zeroes (unlike most sparse matrix representations). For converting between regular matrices and this sparse format see `matrix_to_triplets()` and `triplets_to_matrix()`.

The package can also be used to learn Bayesian canonical correlation analysis (CCA) and group factor analysis (GFA) models, both of which are special cases of CMF. This is likely to be useful for people looking for CCA and GFA solutions supporting missing data and non-Gaussian likelihoods.

Author(s)

Arto Klami <arto.klami@cs.helsinki.fi> and Lauri Väre

Maintainer: Felix Held <felix.held@gmail.se>

References

Arto Klami, Guillaume Bouchard, and Abhishek Tripathi. Group-sparse embeddings in collective matrix factorization. arXiv:1312.5921, 2013.

Arto Klami, Seppo Virtanen, and Samuel Kaski. Bayesian canonical correlation analysis. Journal of Machine Learning Research, 14(1):965–1003, 2013.

Seppo Virtanen, Arto Klami, Suleiman A. Khan, and Samuel Kaski. Bayesian group factor analysis. In Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, volume 22 of JMLR:W&CP, pages 1269-1277, 2012.

Examples

```
require("CMF")

# Create data for a circular setup with three matrices and three
# object sets of varying sizes.
X <- list()
D <- c(10, 20, 30)
inds <- matrix(0, nrow = 3, ncol = 2)

# Matrix 1 is between sets 1 and 2 and has continuous data
inds[1, ] <- c(1, 2)
X[[1]] <- matrix(
  rnorm(D[inds[1, 1]] * D[inds[1, 2]], 0, 1),
  nrow = D[inds[1, 1]]
)

# Matrix 2 is between sets 1 and 3 and has binary data
inds[2, ] <- c(1, 3)
X[[2]] <- matrix(
  round(runif(D[inds[2, 1]] * D[inds[2, 2]], 0, 1)),
  nrow = D[inds[2, 1]]
)

# Matrix 3 is between sets 2 and 3 and has count data
inds[3, ] <- c(2, 3)
X[[3]] <- matrix(
  round(runif(D[inds[3, 1]] * D[inds[3, 2]], 0, 6)),
  nrow = D[inds[3, 1]]
)

# Convert the data into the right format
triplets <- lapply(X, matrix_to_triplets)

# Missing entries correspond to missing rows in the triple representation
# so they can be removed from training data by simply taking a subset
# of the rows.
train <- list()
test <- list()
keep_for_training <- c(100, 200, 300)
for (m in 1:3) {
  subset <- sample(nrow(triplets[[m]]), keep_for_training[m])
```

```
  train[[m]] <- triplets[[m]][subset, ]
  test[[m]] <- triplets[[m]][-subset, ]
}

# Learn the model with the correct likelihoods
K <- 4
likelihood <- c("gaussian", "bernoulli", "poisson")
opts <- getCMFopts()
opts$iter.max <- 500 # Less iterations for faster computation
model <- CMF(train, inds, K, likelihood, D, test = test, opts = opts)

# Check the predictions
# Note that the data created here has no low-rank structure,
# so we should not expect good accuracy.
print(test[[1]][1:3, ])
print(model$out[[1]][1:3, ])

# predictions for the test set using the previously learned model
out <- predictCMF(test, model)
print(out$out[[1]][1:3, ])
print(out$error[[1]])
# ...this should be the same as the output provided by CMF()
print(model$out[[1]][1:3, ])
```

CMF *Collective Matrix Factorization*

Description

Learns the CMF model for a given collection of M matrices. The code learns the parameters of a variational approximation for CMF, and also computes predictions for indices specified in `test`.

Usage

```
CMF(X, inds, K, likelihood, D, test = NULL, opts = NULL)
```

Arguments

`X`	List of input matrices.
`inds`	A `length(X)` times 2 matrix that links dimensions of the matrices in `X` to object sets. `inds[m, 1]` tells which object set corresponds to the rows in matrix `X[[m]]`, and `inds[m, 2]` tells the same for the columns.
`K`	The number of factors.
`likelihood`	A list of likelihood choices, one for each matrix in X. Each entry should be a string with possible values of: "gaussian", "bernoulli" or "poisson".
`D`	A vector containing sizes of each object set.

test	A list of test matrices. If not NULL, the code will compute predictions for these elements of the matrices. This duplicates the functionality of predictCMF().
opts	A list of options as given by getCMFopts(). If set to NULL, the default values will be used.

Details

The variational approximation is fully factorized over all of the model parameters, including individual elements of the projection matrices. The parameters for the projection matrices are updated jointly by Newton-Raphson method, whereas the rest use closed-form updates.

Note that the input data needs to be given in a specific sparse format. See matrix_to_triplets() for details.

The behavior of the algorithm can be modified via the opts parameter. See getCMFopts() for details. Of particular interest are the elements useBias and method.

For full description of the output parameters, see the referred publication. The notation in the code follows roughly the notation used in the paper.

Value

A list of

U	A list of the mean parameters for the rank-K projection matrices, one for each object set.
covU	A list of the variance parameters for the rank-K projection matrices, one for each object set.
tau	A vector of the precision parameter means.
alpha	A vector of the ARD parameter means.
cost	A vector of variational lower bound values.
inds	The input parameter inds stored for further use.
errors	A vector containing root-mean-square errors for each iteration, computed over the elements indicated by the test parameter.
bias	A list (of lists) storing the parameters of the row and column bias terms.
D	The sizes of the object sets as given in the parameters.
K	The number of components as given in the parameters.
Uall	Matrices of U joined into one sum(D) by K matrix, for easier plotting of the results.
items	A list containing the running number for each item among all object sets. This corresponds to rows of the Uall matrix. Each part of the list contains a vector that has the numbers for each particular object set.
out	If test matrices were provided, returns the reconstructed data sets. Otherwise returns NULL.
M	The number of input matrices.
likelihood	The likelihoods of the matrices.
opts	The options used for running the code.

Author(s)

Arto Klami and Lauri Väre

References

Arto Klami, Guillaume Bouchard, and Abhishek Tripathi. Group-sparse embeddings in collective matrix factorization. arXiv:1312.5921, 2014.

Examples

```
# See CMF-package for an example.
```

`getCMFopts` *Default options for CMF*

Description

A helper function that creates a list of options to be passed to `CMF`. To run the code with other option values, first run this function and then directly modify the entries before passing the list to `CMF`.

Usage

```
getCMFopts()
```

Details

Most of the parameters are for controlling the optimization, but some will alter the model itself. In particular, `useBias` is used for turning the bias terms on and off, and `method` will change the prior for `U`.

The default choice for `method` is `"gCMF"`, providing the group-wise sparse CMF that identifies both shared and private factors (see Klami et al. (2013) for details). The value `"CMF"` turns off the group-wise sparsity, providing a CMF solution that attempts to learn only factors shared by all matrices. Finally, `method="GFA"` implements the group factor analysis (GFA) method, by fixing the variance of `U[[1]]` to one and forcing `useBias = FALSE`. Then `U[[1]]` can be interpreted as latent variables with unit variance and zero mean, as assumed by GFA and CCA (special case of GFA with `M = 2`). Note that as a multi-view learning method `"GFA"` requires all matrices to share the same rows, the very first entity set.

Value

Returns a list of:

`init.tau`	Initial value for the noise precisions. Only matters for Gaussian likelihood.
`init.alpha`	Initial value for the automatic relevance determination (ARD) prior precisions.

grad.reg	The regularization parameter for the under-relaxed Newton iterations. 0 = no regularization, larger values provide increasing regularization. The value must be below 1.
gradIter	How many gradient steps for updating the projections are performed during each iteration of the whole algorithm. Default is 1.
grad.max	Maximum absolute change for the elements of the projection matrices during one gradient step. Small values help to prevent over-shooting, wheres inf results to no constraints. Default is inf.
iter.max	Number of iterations for the whole algorithm.
computeCost	Should the cost function values be computed or not. Defaults to TRUE.
verbose	0 = supress all printing, 1 = print current iteration and test RMSE every now and then, 2 = in addition to level 1 print also the current gradient norm.
freq	How often progress updates are printed if verbose is TRUE. Default is 10.
useBias	Set this to FALSE to exclude the row and column bias terms. The default is TRUE.
method	Default value of "gCMF" computes the CMF with group-sparsity. The other possible values are "CMF" for turning off the group-sparsity prior, and "GFA" for implementing group factor analysis (and canonical correlation analysis when M = 2).
prior.alpha_0	Hyperprior values for the gamma prior for ARD.
prior.alpha_0t	Hyperprior values for the gamma prior for tau.

Author(s)

Arto Klami and Lauri Väre

References

Arto Klami, Guillaume Bouchard, and Abhishek Tripathi. Group-sparse embeddings in collective matrix factorization. arXiv:1312.5921, 2014.

Seppo Virtanen, Arto Klami, Suleiman A. Khan, and Samuel Kaski. Bayesian group factor analysis. In Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, volume 22 of JMLR:W&CP, pages 1269-1277, 2012.

See Also

'CMF'

Examples

```
CMF_options <- getCMFopts()
CMF_options$iter.max <- 500 # Change the number of iterations from default
                            # of 200 to 500.
CMF_options$useBias <- FALSE # Do not take row and column means into
                             # consideration.
# These options will be in effect when CMF_options is passed on to CMF.
```

`matrix_to_triplets` *Conversion from matrix to coordinate/triplet format*

Description

The CMF code requires inputs to be speficied in a specific sparse format. This function converts regular R matrices into that format.

Usage

```
matrix_to_triplets(orig)
```

Arguments

`orig`	A matrix of class `matrix`

Details

The element `X[i, j]` on the `i`-th row and `j`-th column is represented as a triple `(i, j, X[i,k])`. The input for CMF is then a matrix where each row specifies one element, and hence the representation is of size `N x 3`, where `N` is the total number of observed entries.

In the original input matrix the missing entries should be marked as `NA`. In the output they will be completely omitted.

Even though this format reminds the representation often used for representing sparse matrices, it is important to notice that observed zeroes are retained in the representation. The elements missing from this representation are considered unknown, not zero.

Value

The input matrix in triplet/coordinate format.

Author(s)

Arto Klami and Lauri Väre

See Also

`triplets_to_matrix()`

Examples

```
x <- matrix(c(1, 2, NA, NA, 5, 6), nrow = 3)
triplet <- matrix_to_triplets(x)
print(triplet)
```

predictCMF *Predict with CMF*

Description

Code for predicting missing elements with an existing CMF model. The predictions are made for all of the elements specified in the list of input matrices `X`. The function also returns the root mean square error (RMSE) between the predicted outputs and the values provided in `X`.

Usage

```
predictCMF(X, model)
```

Arguments

`X`	A list of sparse matrices specifying the indices for which to make the predictions. These matrices must correspond to the structure used for `X` when learning the model with `CMF`.
`model`	A list of model parameter values provided by `CMF`.

Details

Note that `X` needs to be provided as a set of triplets instead of as a regular matrix. See `matrix_to_triplets()`.

Value

A list of

`out`	A list of matrices corresponding to predictions for each matrix in `X`.
`error`	A vector containing the root-mean-square error for each matrix separately.

Author(s)

Arto Klami and Lauri Väre

Examples

```
# See CMF-package for an example.
```

p_check_sparsity *Internal function for checking whether the input is in the right format*

Description

Internal function for checking whether the input is in the right format

Usage

```
p_check_sparsity(mat, max_row, max_col)
```

Arguments

mat	An input matrix of class matrix
max_row	Maximum row index for mat
max_col	Maximum column index for mat

Value

TRUE if the input is in coordinate/triplet format. FALSE otherwise.

triplets_to_matrix *Conversion from triplet/coordinate format to matrix*

Description

This function is the inverse of matrix_to_triplets(). It converts a matrix represented as a set of triplets into an object of the class matrix. The missing entries (the ones not present in the triplet representation) are filled in as NA.

Usage

```
triplets_to_matrix(triplets)
```

Arguments

triplets	A matrix in triplet/coordinate format

Details

See matrix_to_triplets() for a description of the representation.

Value

The input matrix as a normal matrix of class matrix

Author(s)

Arto Klami and Lauri Väre

See Also

matrix_to_triplets()

Examples

```
x <- matrix(c(1, 2, NA, NA, 5, 6), nrow = 3)
triplet <- matrix_to_triplets(x)
print(triplet)
xnew <- triplets_to_matrix(triplet)
print(xnew)
```

Index

CMF, 4
CMF(), *2*
CMF-package, 2

getCMFopts, 6
getCMFopts(), *5*

matrix_to_triplets, 8
matrix_to_triplets(), *2*, *5*, *9–11*

p_check_sparsity, 10
predictCMF, 9
predictCMF(), *2*, *5*

triplets_to_matrix, 10
triplets_to_matrix(), *2*, *8*