Package ‘rBMF’

October 14, 2022

Type Package

Title Boolean Matrix Factorization

Version 1.1

Date 2021-1-13

Author Abdelmoneim Amer Desouki

Maintainer Abdelmoneim Amer Desouki <desouki@hhu.de>

Depends R (>= 3.2.0), Matrix, methods, Rcpp

Description Provides four boolean matrix factorization (BMF) methods.
BMF has many applications like data mining and categorical data
analysis. BMF is also known as boolean matrix decomposition (BMD)
and was found to be an NP-hard (non-deterministic polynomial-time)
problem. Currently implemented methods are
'Asso' Miettinen, Pauli and others (2008)
<doi:10.1109/TKDE.2008.53>,
'GreConD' R. Belohlavek, V. Vychodil (2010)
<doi:10.1016/j.jcss.2009.05.002> ,
'GreConDPlus' R. Belohlavek, V. Vychodil (2010)
<doi:10.1016/j.jcss.2009.05.002> ,
'topFiberM' A. Desouki, M. Roeder, A. Ngonga (2019)
<arXiv:1903.10326>.

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-13 16:40:02 UTC

R topics documented:

`rBMF-package` *Boolean Matrix Factorization*

Description

Provides four boolean matrix factorization (BMF) methods. BMF has many applications like data mining and categorical data analysis. BMF is also known as boolean matrix decomposition (BMD) and was found to be an NP-hard (non-deterministic polynomial-time) problem. Currently implemented methods are 'Asso' Miettinen, Pauli and others (2008) <doi:10.1109/TKDE.2008.53>, 'GreConD' R. Belohlavek, V. Vychodil (2010) <doi:10.1016/j.jcss.2009.05.002> , 'GreConDPlus' R. Belohlavek, V. Vychodil (2010) <doi:10.1016/j.jcss.2009.05.002> , 'topFiberM' A. Desouki, M. Roeder, A. Ngonga (2019) <arXiv:1903.10326>.

Details

The DESCRIPTION file:

Package:	rBMF
Type:	Package
Title:	Boolean Matrix Factorization
Version:	1.1
Date:	2021-1-13
Author:	Abdelmoneim Amer Desouki
Maintainer:	Abdelmoneim Amer Desouki <desouki@hhu.de>
Depends:	R (>= 3.2.0), Matrix, methods, Rcpp
Description:	Provides four boolean matrix factorization (BMF) methods. BMF has many applications like data mining and c
License:	GPL-3

Index of help topics:

```
Asso_approximate        Asso: Boolean Matrix Factorization
Chess                   Chess dataset
DBLP                    DBLP dataset
GreConD                 GreConD Boolean matrix factorization
GreConDPlus             GreConDPlus Boolean Matrix Factorization
rBMF-package            Boolean Matrix Factorization
topFiberM               topFiberM
```

Author(s)

Abdelmoneim Amer Desouki

References

topFiberM -Desouki, A. A., Röder, M., & Ngomo, A. C. N. (2019). topFiberM: Scalable and Efficient Boolean Matrix Factorization. arXiv preprint arXiv:1903.10326.

Asso -Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., & Mannila, H. (2008). The discrete basis problem. IEEE transactions on knowledge and data engineering, 20(10), 1348-1362.

GreConD, GreConDPlus -Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. Journal of Computer and System Sciences 76(1)(2010), 3-20

See Also

`topFiberM` `Asso_approximate` `GreConD` `GreConDPlus`

Examples

```
data(DBLP)
 X=DBLP
    r=7
     Xb=X==1#Convert to boolean
    tempX=as(X,'TsparseMatrix')
    stats=NULL
    for(tP in c(0.2,0.3,0.4,0.5,0.6,0.7,0.8,1)){

      Res=topFiberM(Xb,r=r,tP=tP,SR=100,verbose=1)

    X_=Res$A %*% Res$B
    X_=as(X_,'TsparseMatrix')
    #Calculate metrics
    li=tempX@i[tempX@x==1]+1
    lj=tempX@j[tempX@x==1]+1
    tp=sum(X_[cbind(li,lj)]>0)
    fn=sum(X)-tp#sum(!X_[cbind(li,lj)])
    fp=sum(X_@x>0)-tp
    cv=1-(fp+fn)/(tp+fn)
    stats=rbind(stats,cbind(tP,tp,fn,fp,cv,P=tp*1.0/(tp+fp),R=tp*1.0/(tp+fn)))
    }


    plot(stats[,'tP'],stats[,'R'],type='b',col='red',lwd=2,
    main=sprintf('topFiberM, dataset: %s,
         #Known facts:%d','DBLP',sum(X)),ylab="",xlab='tP',
    xlim=c(0,1),ylim=c(0,1))
    HM=apply(stats,1,function(x){2/(1/x['P']+1/x['R'])})
    points(stats[,'tP'],stats[,'P'],col='blue',lwd=2,type='b')
    points(stats[,'tP'],HM,col='green',lwd=2,type='b')
    grid(nx=10, lty = "dotted", lwd = 2)
    legend(legend=c('Recall','Precision','Harmonic mean'),col=c('red','blue','green'),
    x=0.6,y=0.2,pch=1,cex=0.75,lwd=2)
```

Asso_approximate — *Asso: Boolean Matrix Factorization*

Description

Given binary matrix S (m x n), and a scalar k (number of factors), Asso finds two matrices A (m x r), B (r x n) such taht S~= A * B

Usage

```
Asso_approximate(S, k, opti)
```

Arguments

`S`	input matrix to be factorized
`k`	number of factors (rank)
`opti`	options : list containing components verbose:control of showing messages , threshold:precision limit,penalty_overcovered,bonus_covered.

Value

LIST of three components

`B`	k x n factor matrix
`O`	n x k factor matrix
`D`	n x n matrix, the result from calculate_association

Author(s)

Abdelmoneim Amer Desouki

References

Miettinen, P., Mielikaeinen, T., Gionis, A., Das, G., & Mannila, H. (2008). The discrete basis problem. IEEE transactions on knowledge and data engineering, 20(10), 1348-1362.

See Also

See Also `topFiberM`

Examples

```
data(DBLP)
X=DBLP
Xb=X==1
Res=Asso_approximate(Xb,7,list(threshold=0.5,penalty_overcovered=1,bonus_covered=1,verbose=0))

    X_=Res$O %*% Res$B
    X_=as(X_,'TsparseMatrix')
    X=as(X,'TsparseMatrix')
li=X@i[X@x==1]+1
lj=X@j[X@x==1]+1
tp=sum(X_[cbind(li,lj)]>0)
fn=sum(X)-tp#sum(!X_[cbind(li,lj)])
fp=sum(X_@x>0)-tp
cv=1-(fp+fn)/(tp+fn)
print(sprintf("tp:%d, fp:%d,fn:%d, Error:%d, covered=%.3f",tp,fp,fn,fn+fp,cv))
```

Chess *Chess dataset*

Description

The Chess data describes chess games of certain kind described by attributes representing the positions of pieces on the board. More info: http://fimi.ua.ac.be/data/ dimension: 3196 objects, 76 attributes, number of values in the scale of grades: 2 Downloaded from: http://datasets.inf.upol.cz/

Usage

```
data(Chess)
```

Format

Binary matrix, name Chess

References

Lichman, M: UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science, 2013

DBLP *DBLP dataset*

Description

The DBLP dataset contains records of in which of the 19 conferences the 6980 authors had published. The dataset is collected from the DBLP database . dimension: 6980 objects, 19 attributes, number of values in the scale of grades: 2 Downloaded from: http://datasets.inf.upol.cz/

Usage

```
data(DBLP)
```

Format

Binary matrix, name DBLP

References

Miettinen, P.: Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms, PhD thesis, University of Helsinki, 2009

GreConD *GreConD Boolean matrix factorization*

Description

implements GreConD algorithm for Boolean matrix factorization. returns A . B = X (if the no. of factors is not limited).

Usage

```
GreConD(X, no_of_factors = NULL)
```

Arguments

`X`	input binary matrix to be factorized.
`no_of_factors`	number of factors of the result (optional).

Value

List of the following four components:

`A`	Factor matrix A
`B`	Factor matrix B

Author(s)

Abdelmoneim Amer Desouki

References

Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. Journal of Computer and System Sciences 76(1)(2010), 3-20

See Also

topFiberM, Asso_approximate

Examples

```
data(DBLP)
X=DBLP
Xb=X==1
Res=GreConD(Xb,7)

   X_=Res$A %*% Res$B
    X_=as(X_,'TsparseMatrix')
    X=as(X,'TsparseMatrix')
li=X@i[X@x==1]+1
lj=X@j[X@x==1]+1
tp=sum(X_[cbind(li,lj)]>0)
fn=sum(X)-tp#sum(!X_[cbind(li,lj)])
fp=sum(X_@x>0)-tp
cv=1-(fp+fn)/(tp+fn)
print(sprintf("tp:%d, fp:%d,fn:%d, Error:%d, covered=%.3f",tp,fp,fn,fn+fp,cv))
```

GreConDPlus *GreConDPlus Boolean Matrix Factorization*

Description

implements GreConDPlus Boolean Matrix Factorization Algorithm.

Usage

```
GreConDPlus(X, no_of_factors = NULL, w, verbose = 2)
```

Arguments

X	input binary matrix to be factorized.
no_of_factors	number of factors of the result (optional).
w	wieght parameter for the algorithm : represents the weight of type1 and type2 errors
verbose	integer value to control the appearance of messages. 0 minimal messages will be showed. Default 2

Value

List of the following four components:

`A`	Factor matrix A
`B`	Factor matrix B

Author(s)

Abdelmoneim Amer Desouki

References

Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. Journal of Computer and System Sciences 76(1)(2010), 3-20

See Also

`topFiberM`, `GreConD`, `Asso_approximate`

Examples

```
data(DBLP)
X=DBLP
Xb=X==1
Res=GreConDPlus(Xb,7,w=4,verbose=1)

    X_=Res$A %*% Res$B
    X_=as(X_,'TsparseMatrix')
    X=as(X,'TsparseMatrix')
li=X@i[X@x==1]+1
lj=X@j[X@x==1]+1
tp=sum(X_[cbind(li,lj)]>0)
fn=sum(X)-tp#sum(!X_[cbind(li,lj)])
fp=sum(X_@x>0)-tp
cv=1-(fp+fn)/(tp+fn)
print(sprintf("tp:%d, fp:%d,fn:%d, Error:%d, covered=%.3f",tp,fp,fn,fn+fp,cv))
```

`topFiberM` *topFiberM*

Description

implements topFiberM Boolean matrix factorization algorithm. topFiberM chooses in a greedy way the fibers (rows or columns) to represent the entire matrix. Fibers are extended to rectangles according to a threshold on precision. The search for these" top fibers" can continue beyond the required rank and according to an optional parameter that defines the limit for this search.

Usage

```
topFiberM(X, r = 2, tP = 0.5, verbose = 2, SR = NULL)
```

Arguments

X	the input boolean sparse matrix
r	rank (number of factors) required.
tP	parameter to put threshold on precision
verbose	integer value to control the appearance of messages. 0 minimal messages will be showed. Default 2
SR	search limit which defines the number iterations, minimum value is rank and maximum value is minimum number of columns and number of rows

Value

List of the following four components:

A	Factor matrix A
B	Factor matrix B
X1	remaining uncovered ones, (False negatives)
tf	dataframe logging of steps giving description of each factor, contains index, based on column (2) / row (1), nnz, TP, FP

Author(s)

Abdelmoneim Amer Desouki

References

Desouki, A. A., Roeder, M., & Ngomo, A. C. N. (2019). topFiberM: Scalable and Efficient Boolean Matrix Factorization. arXiv preprint arXiv:1903.10326.

See Also

GreConD Asso_approximate

Examples

```
data(DBLP)
    r=7
    tP=0.6
    X=DBLP
    Xb=X==1#Convert to boolean

 Res=topFiberM(Xb,r=r,tP=tP,SR=100,verbose=1)
    X_=Res$A %*% Res$B
    X_=as(X_,'TsparseMatrix')
```

```
    #Calculate metrics
    tempX=as(X,'TsparseMatrix')
    li=tempX@i[tempX@x==1]+1
    lj=tempX@j[tempX@x==1]+1
    tp=sum(X_[cbind(li,lj)]>0)
    fn=sum(X)-tp#sum(!X_[cbind(li,lj)])
    fp=sum(X_@x>0)-tp
    cv=1-(fp+fn)/(tp+fn)

print(sprintf("tp:%d, fp:%d,fn:%d, Error:%d, covered=%.3f",tp,fp,fn,fn+fp,cv))
```

Index