Package 'mvc'

August 29, 2016

Title Multi-View Clustering

Description

An implementation of Multi-View Clustering (Bickel and Scheffer, 2004). Documents are generated by drawing word values from a categorical distribution for each word, given the cluster. This means words are not counted (multinomial, as in the paper), but words take on different values from a finite set of values (categorical). Thus, it implements Mixture of Categoricals EM (as opposed to Mixture of Multinomials developed in the paper), and Spherical k-Means. The latter represents documents as vectors in the categorical space.

URL http://cs.maunz.de

Version 1.3

Maintainer Andreas Maunz <andreas@maunz.de>

Author Andreas Maunz <andreas@maunz.de>

Depends R (>= 2.14.1), rattle (>= 2.6.18)

Suggests

Imports

License BSD_3_clause + file LICENSE

NeedsCompilation no

Repository CRAN

Date/Publication 2014-02-24 08:05:39

R topics documented:

agreementRateBinM 2
assignFinIdxPerClSkm 3
assignIdxPerClMBinEM 3
checkViews 4
conceptIndicesSkm 4
conceptVectorsSkm 5
consensusMeansPerClVSkm 6
dbern 7
dcat 7

estLogPxBernGthetaJ 8
estLogPxCatGthetaJ 8
logsum 9
mApplyBern 10
mApplyCat 10
mvcmb 11
mvcsph 12
oFMixBinEM 13
oFSkm 14
rowWUL 14
toyView1 15
toyView2 15
toyViews 15
UL 16
vectorLength 16
viewsClasses 17

Index **18**

`agreementRateBinM` *Agreement rate by maximum posterior values.*

Description

Agreement rate by maximum posterior values.

Usage

```
agreementRateBinM(PjV1, PjV2)
```

Arguments

`PjV1`	Posterior matrix view 1 (by document).
`PjV2`	Posterior matrix view 2 (by document).

Value

agreement rate.

assignFinIdxPerClSkm *Assign final indices to means that have the smallest angle.*

Description

Assign final indices to means that have the smallest angle.

Usage

```
assignFinIdxPerClSkm(view1, view2, mPerClV)
```

Arguments

view1	data matrices (row-wise in unit length).
view2	data matrices (row-wise in unit length).
mPerClV	list of means per Cluster and View.

Value

vector of indices for each data point.

Examples

```
## Not run:
view1 = structure(c(1, 1, -1, 0, 1, 0, -1, -1), .Dim = c(4L, 2L))
view2 = structure(c(1, 1, -1, 0, 1, 0, -1, 0), .Dim = c(4L, 2L))
finIdx = assignFinIdxPerClSkm(view1,view2,mPerClV)
dput(finIdx)
# c(2, 2, 1, 1)

## End(Not run)
```

assignIdxPerClMBinEM *Assign final indices to data by maximum posterior value.*

Description

Assign final indices to data by maximum posterior value.

Usage

```
assignIdxPerClMBinEM(PjV1, PjV2)
```

Arguments

PjV1	Posterior matrix view 1 (by document).
PjV2	Posterior matrix view 2 (by document).

Value

vector of indices for each data point.

`checkViews` *Check views for consistency...*

Description

Check views for consistency Views must have exactly the same row names

Usage

```
checkViews(view1, view2)
```

Arguments

`view1`	View 1
`view2`	View 2

Value

Message and stop as appropriate

`conceptIndicesSkm` *Calculate partitions (concept indices) by assigning each vector to the closest concept vector.*

Description

Calculate partitions (concept indices) by assigning each vector to the closest concept vector.

Usage

```
conceptIndicesSkm(X, C, doOutput=F)
```

Arguments

`X`	data matrix (row-wise in unit length).
`C`	matrix with k rows, indicating concept vectors (row-wise in unit length).
`doOutput`	whether progress bar indicators should be output

Value

concept indices as vector.

Examples

```
{
X=structure(c(1, 1, -1, 0, 1, 0, -1, -1), .Dim = c(4L, 2L))
C=structure(c(0.894427190999916, -0.447213595499958,
0.447213595499958, -0.894427190999916), .Dim = c(2L, 2L))
CIdx=conceptIndicesSkm(X,C)
dput(CIdx)
# c(1, 1, 2, 2)
}
```

`conceptVectorsSkm` *Calculate concept vectors for Spherical k-Means as unit length sum of vectors of the k clusters.*

Description

Calculate concept vectors for Spherical k-Means as unit length sum of vectors of the k clusters.

Usage

```
conceptVectorsSkm(X, CIdx, doOutput=F)
```

Arguments

`X`	data matrix (row-wise in unit length).
`CIdx`	vector of length NROW(X) with natural numbers 1..k, indicating cluster for each data vector.
`doOutput`	whether progress bar indicators should be output

Value

concept vectors as matrix (row-wise in unit length).

Examples

```
{
X=structure(c(1, 1, -1, 0, 1, 0, -1, -1), .Dim = c(4L, 2L))
CIdx=c(1, 1, 2, 2)
C=conceptVectorsSkm(X,CIdx)
dput(C)
# structure(c(0.894427190999916, -0.447213595499958,
# 0.447213595499958, -0.894427190999916), .Dim = c(2L, 2L))
}
```

consensusMeansPerClVSkm *Calculate means per Cluster and view for Spherical k-Means by using a consensus approach.*

Description

Calculate means per Cluster and view for Spherical k-Means by using a consensus approach.

Usage

```
consensusMeansPerClVSkm(view1, view2, view1Idx, view2Idx)
```

Arguments

view1	data matrix (row-wise in unit length).
view2	data matrix (row-wise in unit length).
view1Idx	vector of length NROW(view1) with natural numbers 1..k, indicating cluster for each data vector of view1.
view2Idx	vector of length NROW(view1) with natural numbers 1..k, indicating cluster for each data vector of view2.

Value

cluster means as matrices per view (row-wise in unit length).

Examples

```
{
view1 = structure(c(1, 1, -1, 0, 1, 0, -1, -1), .Dim = c(4L, 2L))
view2 = structure(c(1, 1, -1, 0, 1, 0, -1, 0), .Dim = c(4L, 2L))
view1Idx = c(2, 2, 1, 1)
view2Idx = c(2, 1, 1, 1)
mPerClV=consensusMeansPerClVSkm(view1,view2,view1Idx,view2Idx)
dput(mPerClV)
}
```

dbern *Calculate Bernoulli likelihood...*

Description

Calculate Bernoulli likelihood

Usage

```
dbern(x, prob)
```

Arguments

x	a binary event (vector)
prob	the Bernoulli probability (vector)

Value

Bernoulli likelihood

dcat *Calculate categorical likelihood...*

Description

Calculate categorical likelihood

Usage

```
dcat(x, prob)
```

Arguments

x	a categorical event vector
prob	the categorical probability matrix (rows along events, cols along event values)

Value

categorical likelihood

Examples

```
{
dcat(c(1,2,1),matrix(c(.9,.8,.9,.1,.2,.1),3,2))
}
```

`estLogPxBernGthetaJ` *Estimate log document probabilites given specific Bernoulli parameters...*

Description

Estimate log document probabilites given specific Bernoulli parameters

Usage

```
estLogPxBernGthetaJ(X, logprob)
```

Arguments

`X`	a matrix of binary events (row-wise)
`logprob`	the Bernoulli probability

Examples

```
{
X=matrix(c(0,1,0,0,0,0,1,0),2,4,byrow=TRUE) # two documents of length 4
prob=c(.1,.2,.1,.1) # prob per index
dput(mApplyBern(X,prob)) # likelihood for each index
#structure(c(0.9, 0.9, 0.2, 0.8, 0.9, 0.1, 0.9, 0.9), .Dim = c(2L, 4L))
dput(estLogPxBernGthetaJ(X,log(prob)))
# c(-1.92551945940758, -2.73644967562391)
}
```

`estLogPxCatGthetaJ` *Estimate log document probabilites given specific Categorical parameters...*

Description

Estimate log document probabilites given specific Categorical parameters

Usage

```
estLogPxCatGthetaJ(X, logprob)
```

Arguments

`X`	a matrix of categorical events (row-wise)
`logprob`	the Categorical probability

Examples

```
{
X=matrix(c(1,2,1,1,1,1,2,1),2,4,byrow=TRUE) # two documents of length 4
prob=matrix(c(.9,.8,.9,.9,.1,.2,.1,.1),4,2) # prob per index
dput(mApplyCat(X,prob)) # likelihood for each index
#structure(c(0.9, 0.9, 0.2, 0.8, 0.9, 0.1, 0.9, 0.9), .Dim = c(2L, 4L))
dput(estLogPxCatGthetaJ(X,log(prob)))
# c(-1.92551945940758, -2.73644967562391)
}
```

`logsum` *Computes the cumulative sum in terms of logarithmic in- and output...*

Description

Computes the cumulative sum in terms of logarithmic in- and output Useful to avoid numerical underflow when summing products of probabilities When using this function, one can sum sums of log probabilities See also: http://goo.gl/aJopi

Usage

```
logsum(logx)
```

Arguments

`logx`	a vector of log numbers (need not be probabilities)

Value

the log of the sum of the exponentiated input

Examples

```
{
x=c(1,2,3)
exp(logsum(log(x)))
# 6
}
```

mApplyBern *Calculate Bernoulli likelihood row-wise for binary events...*

Description

Calculate Bernoulli likelihood row-wise for binary events

Usage

```
mApplyBern(X, prob)
```

Arguments

X	a matrix of binary events (row-wise)
prob	the Bernoulli probability vector (along events)

Value

a matrix of Bernoulli likelihoods

mApplyCat *Calculate categorical likelihood row-wise for categorical events...*

Description

Calculate categorical likelihood row-wise for categorical events

Usage

```
mApplyCat(X, prob)
```

Arguments

X	a matrix of categorical events (row-wise)
prob	the categorical probability matrix (rows along events, cols along event values)

Value

a matrix of categorical likelihoods

mvcmb *Multi-View Clustering using mixture of categoricals EM.*

Description

Multi-View Clustering using mixture of categoricals EM. See: S. Bickel, T. Scheffer: Multi-View Clustering, ICDM 04.

Usage

```
mvcmb(view1, view2, k=Inf, startView="view1", nthresh=20, doOutput=F,
    doDebug=F)
```

Arguments

view1	View number one, a data frame with the same row names as view2. All columns numeric. Entries are natural numbers, starting from 1.
view2	View number two, a data frame with the same row names as view1. All columns numeric. Entries are natural numbers, starting from 1.
k	The maximum number of clusters to create
startView	String designating the view on which to perform the initial E step, one of "view1", "view2"
nthresh	The number of iterations to run without improvement of the objective function
doOutput	Whether output to the console should be done
doDebug	Whether debug output to the console should be done (implies normal output)

Value

A list reporting the final clustering, with names finalIndices, agreementRate, indicesSv, indicesOv. They designate final cluster indices as a vector, as well as agreement rate of the two views, and the individual indices given by the two views over the course of iterations as data frames.

Examples

```
{
# Demo program, showing how to run Multi-
# View Clustering using Mixture of Binomials EM.
# AM, 2011

# load toy data 'toyView1' and 'toyView2'
data(toyViews)

mvcmb(
view1=toyView1,
view2=toyView2,
nthresh=20,
```

```
k=4,
startView="view1"
)

}
```

---

mvcsph *Multi-View Clustering using Spherical k-Means for categorical data.*

Description

Multi-View Clustering using Spherical k-Means for categorical data. See: S. Bickel, T. Scheffer: Multi-View Clustering, ICDM 04. Hierachical clustering used to determine the initial centers for k-Means

Usage

```
mvcsph(view1, view2, k=Inf, startView="view1", nthresh=20, doOutput=F,
    doDebug=F, plotFile="Rplots.pdf")
```

Arguments

view1	View number one, a data frame with the same row names as view2. All columns numeric. Entries are natural numbers, starting from 1.
view2	View number two, a data frame with the same row names as view1. All columns numeric. Entries are natural numbers, starting from 1.
k	The maximum number of clusters to create
startView	The view on which to perform the initial E step, one of "view1", "view2"
nthresh	The number of iterations to run without improvement of the objective function
doOutput	Whether output to the console should be done
doDebug	Whether debug output to the console should be done (implies normal output)
plotFile	File name where the hierarchical clustering plot should be stored

Value

A list reporting the final clustering, with names finalIndices, agreementRate, indicesSv, indicesOv. They designate final cluster indices as a vector, as well as agreement rate of the two views, and the individual indices given by the two views over the course of iterations as data frames.

Examples

```
{
# Demo program, showing how to run Multi-
# View Clustering using Spherical k-Means
# AM, 2011

# load toy data 'toyView1' and 'toyView2'
data(toyViews)

mvcsph(
view1=toyView1,
view2=toyView2,
nthresh=20,
k=4,
startView="view1"
)

}
```

---

oFMixBinEM *objective function for mixture of binomials EM:...*

Description

objective function for mixture of binomials EM:

Usage

```
oFMixBinEM(DPS)
```

Arguments

DPS	documents weighted by cluster priors

Value

sum of log likelihood of documents

oFSkm *Objective Function (sum of cosines)...*

Description

Objective Function (sum of cosines)

Usage

```
oFSkm(X, C, CIdx)
```

Arguments

X	data matrix (row-wise vectors in unit length).
C	concept vectors as matrix (row-wise in unit length).
CIdx	vector of length NROW(X) with natural numbers 1..k, indicating cluster for each data vector.

Value

sum of cosine-similarities.

Examples

```
{
X=structure(c(0.707, 0.707, 0.707, 0.707), .Dim = c(2L, 2L))
C=structure(c(1, 0, 0, 1), .Dim = c(2L, 2L))
CIdx=c(2, 1)
oFSkm(X,C,CIdx) # 1.414
}
```

rowWUL *Unit length of all vectors row-wise...*

Description

Unit length of all vectors row-wise

Usage

```
rowWUL(X)
```

Arguments

X	matrix

Value

X row-wise converted to unit length

`toyView1` *Toy View 1...*

Description

Toy View 1

Author(s)

Andreas Maunz, 2012

`toyView2` *Toy View 2...*

Description

Toy View 2

Author(s)

Andreas Maunz, 2012

`toyViews` *Toy Views...*

Description

Toy Views

Author(s)

Andreas Maunz, 2012

UL *Unit length for vector...*

Description

Unit length for vector

Usage

```
UL(x)
```

Arguments

x	vector

Value

x converted to unit length

vectorLength *Euclidean length of vector...*

Description

Euclidean length of vector

Usage

```
vectorLength(x)
```

Arguments

x	vector

Value

length of x

`viewsClasses` *Counts unique values in both views...*

Description

Counts unique values in both views Stops on any non-numeric values

Usage

```
viewsClasses(view1, view2)
```

Arguments

`view1`	View 1
`view2`	View 2

Value

list containing unique values for each view

Index

∗Topic **data**
toyView1, 15
toyView2, 15
toyViews, 15

agreementRateBinM, 2
assignFinIdxPerClSkm, 3
assignIdxPerClMBinEM, 3

checkViews, 4
conceptIndicesSkm, 4
conceptVectorsSkm, 5
consensusMeansPerClVSkm, 6

dbern, 7
dcat, 7

estLogPxBernGthetaJ, 8
estLogPxCatGthetaJ, 8

logsum, 9

mApplyBern, 10
mApplyCat, 10
mvcmb, 11
mvcsph, 12

oFMixBinEM, 13
oFSkm, 14

rowWUL, 14

toyView1, 15
toyView2, 15
toyViews, 15

UL, 16

vectorLength, 16
viewsClasses, 17