Package ‘BMN’

February 19, 2015

Type Package

Title The pseudo-likelihood method for pairwise binary markov networks

Version 1.02

Date 2010-04-25

Author Holger Hoefling

Maintainer Holger Hoefling <hhoeflin@gmail.com>

Description This package implements approximate and exact methods for pairwise binary markov models. The exact method uses an implementation of the junction tree algorithm for binary graphical models. For more details see the help files

License GPL-2

Depends R (>= 2.6.0)

Repository CRAN

Date/Publication 2010-04-25 16:00:14

NeedsCompilation yes

R topics documented:

BMNExact ................................................................. 2
BMNExamples .......................................................... 3
BMNJT ................................................................. 4
BMNPseudo ............................................................ 5

Index 8
BMNEexact

Exact inference in L1-penalized Binary Markov Model

Description

Functions to calculate a parameter estimates for L1-penalized Binary Markov Models.

Usage

BMNEexact(X, rhoVec, thrCheck=1e-3, thrPseudo=1e-5, ThetaStart=NULL, verbose = FALSE, maxIter=100, timeout=60, penalize.diag=FALSE)
BMNEexact.single(X, rho, thrCheck=1e-3, thrPseudo=1e-5, ThetaStart=NULL, verbose = FALSE, maxIter=100, timeout=60, penalize.diag=FALSE)

Arguments

x
Input data matrix consisting of 0-1 entries. Has n rows and p columns.
rho
Value of the penalty parameter; If a non-negative p-by-p matrix is given, it is used as the penalty structure.
rhoVec
Gives all values of rho for which the solution should be calculated.
thrCheck
Error threshold at which convergence is declared.
thrPseudo
Error threshold for the internal pseudolikelihood algorithm.
ThetaStart
Starting value for Theta, has to be a p-by-p matrix.
verbose
Should status messages be printed.
maxIter
Maximum number of iterations to run.
timeout
Number of seconds after which the procedure is stopped; for the path algorithm, this is reset for every value of rho.
penalize.diag
Should the diagonal be penalized?

Details

The function BMNEexact fits a penalized pairwise binary Markov model to the data provided as matrix X for each of the elements in the penalty parameter vector rhoVec (note that rhoVec will be sorted in increasing order). Internally, the function BMNEexact.single is called for each entry in rhoVec and the results are collected as described below.

Value

rho
Vector of penalty parameters sorted in increasing order.
ThetaList
A list of Theta pxp matrices, corresponding to the penalty parameters in rho.
success
A logical vector of the same length as rho. True, if the function succeeded for the corresponding value in rho.
penalize.diag
Logical. Indicates if the diagonal was penalized (same as input value penalize.diag.)
Author(s)
Holger Hoefling

See Also
BMNPseudo, BMNJ

Examples

library(BMN)
theta = matrix(numeric(25), ncol=5);
theta[1,1]=0.5; theta[2,2]=0.5; theta[3,3]=0; theta[4,4]= -0.5; theta[5,5]= 0.5;
umSamples=1000; burnIn=100; skip=1;
simData = BMNSamples(theta, numSamples, burnIn, skip)
rhoVec = c(0.01, 0.02, 0.03)
exactPath = BMNExact(simData, rhoVec)
exactSingle = BMNExact.single(simData, 0.02)

Description
Sampling data using Gibbs sampling for use in the examples

Usage
BMNSamples(Theta, numSamples, burnIn, skip)

Arguments

Theta Parameter matrix for the model from which the data is being generated.
numSamples Number of samples to return.
burnIn Number of samples to discard as burn in.
skip Number of samples to discard in-between returned samples.

Details
BMNSamples generates numSamples by using Gibbs sampling. When using Gibbs sampling, it is necessary to discard the initial samples, which is controlled by the parameter burnIn. In order for the drawn samples to be independent, samples in-between also have to be discarded, which is controlled by skip.
Value

Returns a matrix of 0 and 1 of size `numSamples` times `p` where `p` is the number of rows of `Theta`.

Author(s)

Holger Hoefling

See Also

`BMNPseudo`, `BMNExact`

---

**BMNJT**

*Junction tree algorithm for pairwise binary Markov networks*

Description

Calculates the expections and second moments for a pairwise binary Markov network using the junction tree algorithm.

Usage

`BMNJT(thetaMat, adjMat = NULL, var = NULL, onlyActive = FALSE, timeout = 60)`

Arguments

- `thetaMat` 
  Theta matrix of the model (assumed to be symmetric).
- `adjMat` 
  Adjacency matrix of the underlying graph. If not given, inferred from `thetaMat`. If given, values in `thetaMat` that do not correspond to an edge in the graph are being ignored.
- `var` 
  if given, the expectation and second moments of variable `var` only will be calculated. Otherwise, for all variables.
- `onlyActive` 
  If TRUE, only values in the second moment matrix corresponding to edges in the graph will be calculated (much faster). Otherwise, all entries will be calculated.
- `timeout` 
  Number of seconds until the functions terminates automatically.

Details

Uses the junction tree algorithm to calculate the matrix of second moments of the underlying pairwise binary Markov model. This is useful for inference on this type of models as the matrix of second moments is the derivative of the partition function in this class of models. The function provides the option to calculate the second moments only for one variable with all the others (variables are numbered from 1 to `p`; see `var`). Also, the junction tree algorithm is by a factor `p` more efficient if only second moments corresponding to edges in the underlying graph (non-zero values in the `thetaMat` or `adjMat`) are being calculated. This can be done using the switch `onlyActive`. 
Value

The return value is a list with elements:

- **Expectation**: Expectation of the variables.
- **SecondMomentMatrix**: Second moments of the variables (only present if `Var=NULL`).
- **SecondMomentVector**: Vector of second moments of variable `Var` with the others if `Var != NULL`.

Author(s)

Holger Hoefling

See Also

`BMNPseudo`, `BMNExact`

Examples

```r
library(bmn)
Theta = matrix(numeric(25), ncol=5)
Theta[1,1]=0.5; Theta[2,2]=0.5; Theta[3,3]=0; Theta[4,4]=-0.5; Theta[5,5]= 0.5;
BMNJ(Theta)
```

Description

Functions to calculate approximate parameter estimates for L1-penalized Binary Markov Models.

Usage

```r
BMNPseudo(X, rhoVec, Delta=NULL, ThetaStart=NULL, maxError=1e-5, verbose=FALSE, maxIter=100, penalize.diag=FALSE, stepSize=1, performLineSearch=FALSE)
BMNPseudo.single(X, rho, Delta=NULL, ThetaStart=NULL, maxError=1e-5, maxIter=100, penalize.diag=FALSE, stepSize=1, performLineSearch=FALSE)
```
Arguments

- **X**: Input data matrix consisting of 0-1 entries. Has n rows and p columns.
- **rho**: Value of the penalty parameter; If a non-negative p-by-p matrix is given, it is used as the penalty structure.
- **rhoVec**: Gives all values of rho for which the solution should be calculated.
- **Delta**: Adjustment to the gradient.
- **ThetaStart**: Starting value for Theta, has to be a p-by-p matrix.
- **maxError**: Convergence threshold for the algorithm.
- **verbose**: Print status messages.
- **maxIter**: Maximum number of iterations to run.
- **penalize.diag**: Should the diagonal be penalized?
- **stepSize**: Stepsize of the algorithm; should be 1 or less.
- **performLineSearch**: If TRUE, a line search is performed; takes longer but is guaranteed to converge.

Details

The function `BMNPseudo` fits an approximate penalized pairwise binary Markov model to the data provided as matrix `X` for each of the elements in the penalty parameter vector `rhoVec` (note that `rhoVec` will be sorted in increasing order). Internally, the function `BMNExact` is called for each entry in `rhoVec` and the results are collected as described below.

Value

- **rho**: Vector of non-negative penalty parameters sorted in decreasing order.
- **ThetaList**: A list of Theta pxp matrices, corresponding to the penalty parameters in rho.
- **success**: A logical vector of the same length as rho. True, if the function succeeded for the corresponding value in rho.
- **penalize.diag**: Logical. Indicates if the diagonal was penalized (same as input value `penalize.diag`).

Author(s)

Holger Hoefling

See Also

`BMNExact`, `BMNJT`

Examples

```r
library(BMN)
Theta = matrix(numeric(25), ncol=5);
Theta[1,1]=0.5; Theta[2,2]=0.5; Theta[3,3]=0; Theta[4,4]=-0.5; Theta[5,5]= 0.5;
umSamples=1000; burnIn=100; skip=1;
```
```r
simData = BMNSamples(Theta, numSamples, burnIn, skip)
rhoVec = c(0.01, 0.02, 0.03)

pseudoPath = BMNPseudo(simData, rhoVec)
pseudoSingle = BMNPseudo.single(simData, 0.02)
```
Index

*Topic multivariate
  BMNExact, 2
  BMNExamples, 3
  BMNJt, 4
  BMNPseudo, 5
  BMNExact, 2, 4–6
  BMNExamples, 3
  BMNJt, 3, 4, 6
  BMNPseudo, 3–5, 5
  BMNSamples (BMNExamples), 3