Package ‘TensorComplete’

May 11, 2021

Type Package
Title Tensor Noise Reduction and Completion Methods
Version 0.1.0
Author Chanwoo Lee <chanwoo.lee@wisc.edu>, Miaoyan Wang <miaoyan.wang@wisc.edu>
Maintainer Chanwoo Lee <chanwoo.lee@wisc.edu>
Imports pracma, methods, utils, tensorregress, MASS
Description Efficient algorithms for tensor noise reduction and completion. This package includes a suite of parametric and nonparametric tools for estimating tensor signals from noisy, possibly incomplete observations. The methods allow a broad range of data types, including continuous, binary, and ordinal-valued tensor entries. The algorithms employ the alternating optimization. The detailed algorithm description can be found in the following three references.
License GPL (>= 2)
Encoding UTF-8
RoxygenNote 7.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2021-05-11 14:52:15 UTC

R topics documented:

Altopt ........................................ ........................................ 2
bic ........................................ ........................................ 3
fit_continuous_cp .......................... ........................................ 4
Altopt

Alternating optimization of the weighted classification loss

Description
Optimize the weighted classification loss given a weight tensor, an observed data tensor, and a large margin loss. This function is used as a subroutine in the main function fit_nonparaT.

Usage
Altopt(Ybar,W,r,type = c("logistic","hinge"),start = "linear")

Arguments
Ybar A given (possibly noisy and incomplete) data tensor.
W A weight tensor used in the weighted classification loss.
r A rank to be fitted (CP rank).
type A large margin loss to be used. Logistic or hinge loss is available.
start Choice of initialization method. Use random initialization if start = "random"; Use the initialization based on low rank approximation if start = "linear". Linear initialization is default.

Value
The returned object is a list of components.
  binary_obj - Trajectory of binary loss values over iterations.
  obj - Trajectory of weighted classification loss values over iterations.
  iter - The number of iterations.
  error - Trajectory of errors over iterations.
  fitted - A tensor that optimizes the weighted classification loss.

References
Examples

library(tensorregress)
indices = c(2,3,4)
noise = rand_tensor(indices)@data
Theta = array(runif(prod(indices),min=-3,max = 3),indices)

# The signal plus noise model
Y = Theta + noise

# Optimize the weighted classification for given a sign tensor sign(Y) and a weight tensor abs(Y)
result = Altopt(sign(Y),abs(Y),r = 3,type = "hinge",start = "linear")
signTheta = sign(result$fitted)

bic

Bayesian Information Criterion (BIC) value.

Description

Compute Bayesian Information Criterion (BIC) given a parameter tensor, an observed tensor, the dimension, and the rank based on cumulative logistic model. This BIC function is designed for selecting rank in the fit_ordinal function.

Usage

bic(ttnsr,theta,omega,d,r)

Arguments

<table>
<thead>
<tr>
<th>ttnsr</th>
<th>An observed tensor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>theta</td>
<td>A continuous-valued tensor (latent parameters).</td>
</tr>
<tr>
<td>omega</td>
<td>The cut-off points.</td>
</tr>
<tr>
<td>d</td>
<td>Dimension of the tensor.</td>
</tr>
<tr>
<td>r</td>
<td>Rank of the tensor.</td>
</tr>
</tbody>
</table>

Value

BIC value at given inputs based on cumulative logistic model.
**fit_continuous_cp**  
*Signal tensor estimation from a noisy and incomplete data tensor based on CP low rank tensor method.*

### Description
Estimate a signal tensor from a noisy and incomplete data tensor using CP low rank tensor method.

### Usage

```r
fit_continuous_cp(data, r)
```

### Arguments
- **data**: A given (possibly noisy and incomplete) data tensor.
- **r**: A rank to be fitted (CP rank).

### Value
The returned object is a list of components.
- **est**: An estimated signal tensor based on CP low rank tensor method.
- **U**: A list of factor matrices.
- **lambda**: A vector of tensor singular values.

### Examples
```r
library(tensorregress)
indices = c(2,3,4)
noise = rand_tensor(indices)@data
Theta = array(runif(prod(indices), min=-3, max = 3), indices)

# The signal plus noise model
Y = Theta + noise

# Estimate Theta from CP low rank tensor method
hatTheta = fit_continuous_cp(Y,3)
print(hatTheta$est)
```
**Description**

Estimate a signal tensor from a noisy and incomplete data tensor using the Tucker model.

**Usage**

`fit_continuous_tucker(ttnsr,r,alpha = TRUE)`

**Arguments**

- `ttnsr` A given (possibly noisy and incomplete) data tensor.
- `r` A rank to be fitted (Tucker rank).
- `alpha` A signal level
  - `alpha = TRUE` If the signal level is unknown.

**Value**

A list containing the following:
- `C` - An estimated core tensor.
- `iteration` - The number of iterations.
- `cost` - Log-likelihood value at each iteration.

**Examples**

```r
# Latent parameters
library(tensorregress)
alpha = 10
A_1 = matrix(runif(15*2,min=-1,max=1),nrow = 15)
A_2 = matrix(runif(15*2,min=-1,max=1),nrow = 15)
A_3 = matrix(runif(15*2,min=-1,max=1),nrow = 15)
C = as.tensor(array(runif(2^3,min=-1,max=1),dim = c(2,2,2)))
theta = ttm(ttm(ttm(C,A_1,1),A_2,2),A_3,3)@data
theta = alpha*theta/max(abs(theta))
adj = mean(theta)
theta = theta-adj
omega = c(-0.2,0.2)+adj

# Observed tensor
ttnsr <- realization(theta,omega)@data

# Estimation of parameters
continuous_est = fit_continuous_tucker(ttnsr,c(2,2,2),alpha = 10)
```
fit_nonparaT  

Main function for nonparametric tensor estimation and completion based on low sign rank model.

Description

Estimate a signal tensor from a noisy and incomplete data tensor using nonparametric tensor method via sign series.

Usage

fit_nonparaT(Y, truer, H, Lmin, Lmax, option = 2)

Arguments

Y  A given (possibly noisy and incomplete) data tensor. The function allows both continuous- and binary-valued tensors. Missing value should be encoded as NA.

truer  Sign rank of the signal tensor.

H  Resolution parameter.

Lmin  Minimum value of the signal tensor (or minimum value of the tensor Y).

Lmax  Maximum value of the signal tensor (or maximum value of the tensor Y).

option  A large margin loss to be used. Use logistic loss if option = 1, hinge loss if option = 2. Hinge loss is default.

Value

The returned object is a list of components.

fitted - A series of optimizers that minimize the weighted classification loss at each level.

est - An estimated signal tensor based on nonparametric tensor method via sign series.

References


Examples

library(tensorregress)
indices = c(2,2,2)
noise = rand_tensor(indices)@data
Theta = array(runif(prod(indices), min=-2, max = 2), indices)

# The signal plus noise model
Y = Theta + noise

# Estimate Theta from nonparametric completion method via sign series
hatTheta = fit_nonparaT(Y,truer = 2,H = 2,Lmin = -2,Lmax = 2, option =2)
print(hatTheta$est)

fit_ordinal

Main function for parametric tensor estimation and completion based 
on ordina l observations.

Description

Estimate a signal tensor from a noisy and incomplete ordinal-valued tensor using the cumulative logistic model.

Usage

fit_ordinal(ttnsr,r,omega=TRUE,alpha = TRUE)

Arguments

ttnsr  A given (possibly noisy and incomplete) data tensor. The function allows binary- and ordinal-valued tensors. Missing value should be encoded as NA.

r  A rank to be fitted (Tucker rank).

omega  The cut-off points if known,
omega = TRUE if unknown.

alpha  A signal level
alpha = TRUE if the signal level is unknown.

Value

A list containing the following:
C - An estimated core tensor.
A - Estimated factor matrices.
theta - An estimated latent parameter tensor.
iteration - The number of iterations.
cost - Log-likelihood value at each iteration.
omega - Estimated cut-off points.

References

Examples

# Latent parameters
library(tensorregress)
alpha = 10
A_1 = matrix(runif(15*2,min=-1,max=1),nrow = 15)
A_2 = matrix(runif(15*2,min=-1,max=1),nrow = 15)
A_3 = matrix(runif(15*2,min=-1,max=1),nrow = 15)
C = as.tensor(array(runif(2^3,min=-1,max=1),dim = c(2,2,2)))
theta = ttm(ttm(ttm(C,A_1,1),A_2,2),A_3,3)@data
theta = alpha*theta/max(abs(theta))
adj = mean(theta)
theta = theta-adj
omega = c(-0.2,0.2)+adj

# Observed tensor
ttnsr <- realization(theta,omega)@data

# Estimation of parameters
ordinal_est = fit_ordinal(ttnsr,c(2,2,2),omega = TRUE,alpha = 10)

likelihood

Log-likelihood function (cost function).

Description

Return log-likelihood function (cost function) value evaluated at a given parameter tensor, an observed tensor, and cut-off points.

Usage

likelihood(ttnsr,theta,omega,type = c("ordinal","Gaussian"))

Arguments

ttnsr An observed tensor data.
theta A continuous-valued tensor (latent parameters).
omega The cut-off points.
type Types of log-likelihood function.
"ordinal" specifies log-likelihood function based on the cumulative logistic model.
"Gaussian" specifies log-likelihood function based on the Gaussian model.

Value

Log-likelihood value at given inputs.
predict_ordinal

Description

Predict ordinal-valued tensor entries given latent parameters and a type of estimations.

Usage

predict_ordinal(theta, omega, type = c("mode","mean","median"))

Arguments

theta
A continuous-valued tensor (latent parameters).

omega
The cut-off points.

type
Type of estimations:
"mode" specifies argmax based label estimation.
"mean" specifies mean based label estimation.
"median" specifies median based label estimation.

Value

A predicted ordinal-valued tensor given latent parameters and a type of estimations.

References


Examples

indices <- c(10,20,30)
arr <- array(runif(prod(indices),-2,2),dim = indices)
b <- c(-1.5,0,1.5)
r_predict <- predict_ordinal(arr,b,type = "mode");r_predict
**realization**

An ordinal-valued tensor randomly simulated from the cumulative model.

**Description**

Simulate an ordinal-valued tensor from the cumulative logistic model with the parameter tensor and the cut-off points.

**Usage**

```r
realization(theta, omega)
```

**Arguments**

- `theta`: A continuous-valued tensor (latent parameters).
- `omega`: The cut-off points.

**Value**

An ordinal-valued tensor randomly simulated from the cumulative logistic model.

**References**


**Examples**

```r
indices <- c(10, 20, 30)
arr <- array(runif(prod(indices)), dim = indices)
b <- qnorm((1:3)/4)
r_sample <- realization(arr, b);
r_sample
```
Index

 Altopt, 2
 bic, 3
 fit_continuous_cp, 4
 fit_continuous_tucker, 5
 fit_nonparaT, 6
 fit_ordinal, 7
 likelihood, 8
 predict_ordinal, 9
 realization, 10