Package 'TensorComplete'

May 9, 2021

Type Package

Title Tensor Noise Reduction and Completion Methods

Version 1.1.0

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 the cumulative link model for ordinal tensor observation and nonparametric tensor method via sign series. The algorithms employ the alternating optimization approach. The detailed algorithm description can be found in Lee and Wang, Proceedings of International Conference on Machine Learning, 119:5778-5788, 2020 and <arXiv:2102.00384>.

URL http://proceedings.mlr.press/v119/lee20i.html,
http://arxiv.org/abs/2102.00384

License GPL(>=2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

R topics documented:

`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.

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`	Tensor rank to be fitted.
`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

C. Lee and M. Wang. Beyond the Signs: Nonparametric Tensor Completion via Sign Series. *arXiv preprint arXiv:2102.00384*, 2020.

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.

Usage

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

Arguments

`ttnsr`	an observed tensor.
`theta`	a continuous-valued tensor (latent parameters).
`omega`	the cut-off points.
`d`	dimension of the tensor.
`r`	rank of the tensor.

Value

BIC value at given inputs.

`estimation` *Estimation of tensor entries from the cumulative model.*

Description

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

Usage

```
estimation(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

an estimated ordinal tensor given latent parameters and a type of estimations.

References

C. Lee and M. Wang. Tensor denoising and completion based on ordinal observations. *International Conference on Machine Learning (ICML)*, 2020.

Examples

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

`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

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

Arguments

`data`	A given (possibly noisy and incomplete) data tensor.
`r`	Rank of the signal tensor.

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

```
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)
```

fit_continuous_tucker *Signal tensor estimation from a noisy and incomplete data tensor based on the Tucker model.*

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`	an observed 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.

`A` - estimated factor matrices.

`iteration` - the number of iterations.

`cost` - log-likelihood value at each iteration.

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
continuous_est = fit_continuous_tucker(ttnsr,c(2,2,2),alpha = 10)
```

`fit_nonparaT` *Signal tensor estimation from a noisy and incomplete data tensor based on nonparametric tensor method via sign series.*

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.
`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 pi.

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

References

C. Lee and M. Wang. Beyond the Signs: Nonparametric Tensor Completion via Sign Series. *arXiv preprint arXiv:2102.00384*, 2020.

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

# Estimate Theta from nonparametic completion method via sign series
hatTheta = fit_nonparaT(Y,truer = 3,H = 3,Lmin = -3,Lmax = 3, option =2)
print(hatTheta$est)
```

`fit_ordinal` *Signal tensor estimation from a noisy and incomplete ordinal-valued tensor based on the cumulative logistic model.*

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`	an observed tensor.
`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

C. Lee and M. Wang. Tensor denoising and completion based on ordinal observations. *International Conference on Machine Learning (ICML)*, 2020.

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.

`realization` *An ordinal tensor randomly simulated from the cumulative model.*

Description

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

Usage

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

Arguments

theta	a continuous-valued tensor (latent parameters).
omega	the cut-off points.

Value

an ordinal tensor randomly simulated from the cumulative logistic model.

References

C. Lee and M. Wang. Tensor denoising and completion based on ordinal observations. *International Conference on Machine Learning (ICML)*, 2020.

Examples

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

---

theta_to_p *The probability matrix given latent parameters.*

Description

Compute the probability matrix given latent parameters from the cumulative model.

Usage

```
theta_to_p(theta,omega)
```

Arguments

theta	a continuous-valued tensor (latent parameters).
omega	the cut-off points.

Value

a probability matrix in which the number of columns is possible outcomes and each row vector is corresponding probabilities at an entry of the tensor.

Examples

```
indices <- c(10,20,30)
arr <- array(runif(prod(indices),-2,2),dim = indices)
b <- c(-1.5,0,1.5)
probability <- theta_to_p(arr,b);probability
```

Index