Package 'TensorTest2D'

July 21, 2025

Type Package

Title Fitting Second-Order Tensor Data

Version 1.1.2

Depends R (>= 3.5.0)

Imports stats, grDevices, graphics

Suggests abind, glmnet, testthat, knitr, rmarkdown

Description An implementation of fitting generalized linear models on second-order tensor type data. The functions within this package mainly focus on parameter estimation, including parameter coefficients and standard deviation.

License GPL

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

RoxygenNote 7.2.3

NeedsCompilation no

Author Mark Chen [aut, cre],
Sheng-Mao Chang [aut],
Wenbin Lu [aut],
Jung-Ying Tzeng [aut],
Ping-Yang Chen [aut]

URL https://github.com/yuting1214/TensorTest2D

BugReports https://github.com/yuting1214/TensorTest2D/issues

Maintainer Mark Chen <l50l1501l@gmail.com>

Repository CRAN

Date/Publication 2024-07-25 19:20:02 UTC

Contents

`ALS` *The function performing the Alternating Least Square (ALS) Algorithm.*

Description

The function performing the Alternating Least Square (ALS) Algorithm.

Usage

```
ALS(DATA, n_R, family, opt = opt, max_ite = max_ite, tol = tol)
```

Arguments

`DATA`	A list. The input data. `DATA$y` is the dependent variable. `DATA$X` is the 3-D tensor independent variables. `DATA$W` is other independent variables.
`n_R`	A numerical constant. A predefined value determines the rank of the approximate matrix
`family`	Family of `generalized linear model`. Provide three options for model.(see more details in **Details**)
`opt`	Optimization options. Provide two options for optimization stopping criterion. **opt = 1 or 2**. (see more details in **Details**)
`max_ite`	Maximum iteration. The value of maximum iterations for the algorithm.
`tol`	Tolerance. The value of tolerance with respect to optimization.

Value

A list. See `tensorReg2D`.

`Calculate_IC_Dev` *The function computing information criterion values and deviances.*

Description

The function computing information criterion values and deviances.

Usage

```
Calculate_IC_Dev(y, w_seq, df, family)
```

Arguments

`y`	A numerical vector. Dependent variable.
`w_seq`	A numerical vector. Fitted values of the tensor regression model.
`df`	integer. Degree of freedom.
`family`	Family of `generalized linear model`. Provide three options for model.(see more details in **Details**)

Value

A list.

`IC`: The values of Akaike information criterion (AIC) and Bayesian information criterion (BIC).

`DoF`: The values of the residual degrees of freedom for the null model and the residual degrees of freedom.

`Dev_res`: The deviance values of residuals.

`Dev`: The deviance values for the null model and the working model.

`Check_tidy_input` *The function confirming the input variables fit the requirements of the tensorReg2D function.*

Description

The function confirming the input variables fit the requirements of the tensorReg2D function.

Usage

```
Check_tidy_input(y, X, W, n_R, family)
```

Arguments

y	A numerical vector. Dependent variable.
X	A numerical 3-D array. Independent variable(3-D tensor).
W	A numerical matrix. Independent variable.
n_R	A numerical constant. A predefined value determines the rank of the approximate matrix
family	Family of `generalized linear model`. Provide three options for model.(see more details in **Details**)

Value

A list. `DATA`: list of input data.`DATA$y` is the dependent variable. `DATA$X` is the 3-D tensor independent variables. `DATA$W` is other independent variables.

`fm`: The model expression shown in `summary.tsglm`.

---

draw.coef *Marking Specific Pixels on the Given Image Plot*

Description

Marking Specific Pixels on the Given Image Plot

Usage

```
draw.coef(
  img,
  marks,
  markstyle = c("black", "bi-dir"),
  showlabels = TRUE,
  plot.legend = TRUE,
  grids = FALSE,
  ...
)
```

Arguments

img	a matrix of image data.
marks	a matrix of the same size as `img`. On the image plot, the pixels are marked if the corresponding cells in `marks` are non-zero. The user can specify the style of the marks through `markstyle`.
markstyle	string. The style of pixels' marks. If `markstyle = 'black'`, the rectangles are marked by black edges for non-zero cells in `marks`. If `markstyle = 'bi-dir'`, "red" rectangles are marked on the pixels in which the cells in `marks` are positive, and, "blue" rectangles are marked on the pixels in which the cells in `marks` are negative.

showlabels	boolean. For showlabels = TRUE, if dimnames(img) exists, the row and column names are shown on the sides of the image plot; otherwise, the row and column indices are shown.
plot.legend	boolean. Set plot.legend = TRUE if the colorbar legend is needed. The default is TRUE.
grids	boolean. If grids = TRUE, grid lines are added for the image plot.
...	further arguments passed to the image function.

Author(s)

Ping-Yang Chen

See Also

plot.tsglm

Examples

```
#
```

---

getGLMCoef *getGLMCoef: Computing the regression coefficients of generalized linear model.*

Description

getGLMCoef: Computing the regression coefficients of generalized linear model.

Usage

```
getGLMCoef(X, y, family, offset_vec)
```

Arguments

X	A numerical matrix. Independent variable.
y	A numerical vector. Dependent variable.
family	Family of generalized linear model. Provide three options for model.(see more details in **Details**)
offset_vec	A numerical vector. Prior knowledge to be included in the predictors.

Value

list.

`mnist_mp2c2` *Read the pre-processed MNIST dataset*

Description

A pre-processed MNIST dataset with each image of 10*10 = 100 pixels.

Usage

```
data(mnist_mp2c2)
```

Format

A list of two sublists, `mnist_mp2c2$train` and `mnist_mp2c2$test`. In each sublist, the data is stored as a list of length two, image and label. The image is a 3-dimensional array of size (10, 10, n), where n represents the data size. For i = 1, ..., n the i-th slice of image is an integer matrix with elements in [0, 255] representing the image of 10*10 = 100 pixels in grey scale. The label is a vector of length n. The i-th value is the digit of the i-th slice of image.

Details

The original MNIST handwritten digit is the image of 28*28 = 784 pixels. The pre-processing procedure is as follows. First, the original 28 by 28 image is separated into 14 by 14 clusters, each one is a 2 by 2 block. For each cluster, the maximal value in its cells is taken, and therefore, the original MNIST image is reduced into 14 by 14 image. Because the surrounding cells of the reduced image are usually zero value, we only take the center 10 by 10 sub-image by cutting the edge cells. Thus, the pre-processed MNIST dataset has images of 10*10 = 100 pixels.

Source

https://CRAN.R-project.org/package=dslabs

References

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. (URL)

Rafael A. Irizarry and Amy Gill (2019). dslabs: Data Science Labs. R package version 0.7.3. (URL)

LeCun, Y. http://yann.lecun.com/exdb/mnist/

Examples

```
data(mnist_mp2c2)
dim(mnist_mp2c2$train$image)
# 10    10 60000
image(mnist_mp2c2$train$image[,,1])
mnist_mp2c2$train$label[1]
```

`omics` *Lung-cancer cell lines data in cancer cell line encyclopedia (CCLE) dataset*

Description

The omics data is a subset of the dataset provided by cancer cell line encyclopedia (CCLE) project (Barretina et al., 2012; https://sites.broadinstitute.org/ccle/).

Usage

```
data(omics)
```

Format

A list contains two objects:

omics a 3-dimensional array with size (3, 10, 68)

Y a 68-length vector representing the response variable

Details

This data consists of one response variable and ten genes evaluated under three different platforms.

The response variable measures the log-transformed activity area of taking Vandertanib, a drug targeting on EGFR gene for lung cancer.

The three platforms are DNA copy number variation (CNV), methylation and mRNA expression.

Among the 10 genes, 7 of them (EGFR, EREG, HRAS, KRAS, PTPN11, STAT3, and TGFA) are involved in the protein-protein interaction network of EGFR (https://string-db.org) and the rest (ACTB, GAPDH, and PPIA) are arbitrarily chosen housekeeping genes and play the role of negative control.

Detailed pre-processing procedure is available in Chang et al. (2021).

Source

https://string-db.org

References

Barretina, J., Caponigro, G., Stransky, N. et al. The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature 483, 603–607 (2012). (Link)

Sheng-Mao Chang, Meng Yang, Wenbin Lu, Yu-Jyun Huang, Yueyang Huang, Hung Hung, Jeffrey C Miecznikowski, Tzu-Pin Lu, Jung-Ying Tzeng, Gene-set integrative analysis of multi-omics data using tensor-based association test, Bioinformatics, 2021;, btab125, (Link))

Examples

```
data(omics)
names(omics)
dim(omics$omics)
# 3 10 68
```

`plot.tsglm` *Plot Effective Image Pixels for A* `"tsglm"` *Object*

Description

`plot` method for self-defined class `"tsglm"`.

Usage

```
## S3 method for class 'tsglm'
plot(
  x,
  method = p.adjust.methods,
  alpha = NULL,
  type = c("coef", "tval"),
  background = NULL,
  showlabels = TRUE,
  plot.legend = TRUE,
  ...
)
```

Arguments

`x`	an object of class `"tsglm"`.
`method`	p-value correction method. See `p.adjust`. Its default value is `"none"`.
`alpha`	double. The value of significance level. Its default value is `NULL`. If specify real-valued `alpha`, for example 0.05, the image plot marks the pixels with p-values smaller than `alpha`.
`type`	string. The type of values shown on the image pixels when `background = NULL`. Set `type = 'coef'` for showing the values of estimated coefficients of the \(B\) matrix. Set `type = 'tval'` for showing the t-statistics of the coefficients of the \(B\) matrix. If `background` is not `NULL`, the plot will neglect the choice for `type` and show the background image as per user's interest.
`background`	an image data that used as the background of the effectiveness markers. If `background = NULL`, the background color shows the effect size of the each pixel according to the setting in `type`.
`showlabels`	boolean. For `showlabels = TRUE`, if the row and column names of the image exist, the row and column names are shown on the sides of the image plot; otherwise, the row and column indices are shown.

plot.legend	boolean. Set plot.legend = TRUE if the colorbar legend is needed. The default is TRUE.
...	further arguments passed to the image function.

Author(s)

Ping-Yang Chen

See Also

tensorReg2D, draw.coef

Examples

```
# Simulation data
n <- 500 # number of observations
n_P <- 3; n_G <- 16 # dimension of 3-D tensor variables.
n_d <- 1 # number of numerical variable, if n_d == 1, numerical variable equals to intercept.
beta_True <- rep(1, n_d)
B_True <- c(1, 1, 1)%*%t(rnorm(n_G)) + c(0, .5, .5)%*%t(rnorm(n_G))
B_True <- B_True / 10
W <- matrix(rnorm(n*n_d), n, n_d); W[,1] <- 1
X <- array(rnorm(n*n_P*n_G), dim=c(n_P, n_G, n))

# Binomial Responses
p_B <- exp(W%*%beta_True + X%hp%B_True); p_B <- p_B/(1+p_B)
y_B <- rbinom(n, 1, p_B)
DATA_B <- list(y = y_B, W = W, X = X)

# Binomial Model
result_B <- tensorReg2D(y = DATA_B$y, X = DATA_B$X, W=NULL, n_R = 1,
family = "binomial", opt = 1, max_ite = 100, tol = 10^(-7) )

# Plot the effect size of the pixels
plot(result_B, method = "fdr", alpha = 0.05, type = "coef")
# Plot the t-statistics of the coefficients of the pixels
plot(result_B, method = "fdr", alpha = 0.05, type = "tval")

# Plot the effective pixels with data image as the background
x0 <- DATA_B$X[,,which(DATA_B$y == 0)]
x1 <- DATA_B$X[,,which(DATA_B$y == 1)]
m0 <- m1 <- matrix(0, dim(DATA_B$X)[1], dim(DATA_B$X)[2])
for (i in 1:dim(x0)[3]) m0 <- m0 + x0[,,i]/dim(x0)[3]
for (i in 1:dim(x1)[3]) m1 <- m1 + x1[,,i]/dim(x1)[3]
par(mfrow = c(1, 2), mar = c(2, 2, 2, 2))
plot(result_B, method = "fdr", alpha = 0.05,
background = m0, col = gray(seq(0, 1, 0.05)))
title("Category 0")
plot(result_B, method = "fdr", alpha = 0.05,
background = m1, col = gray(seq(0, 1, 0.05)))
title("Category 1")
```

predict.tsglm *Predict by Second-order Tensor Generalized Regression*

Description

predict method for self-defined class "tsglm".

Usage

```
## S3 method for class 'tsglm'
predict(object, newx, neww = NULL, type = c("link", "response"), ...)
```

Arguments

object	Fitted "tsglm" object.
newx	a 3-dimensional array for x for which the prediction are interested.
neww	a numerical matrix for W for which the prediction are interested.
type	the type of prediction required. The default is type = "link" that returns prediction values on the scale of the linear predictors (eta). Alternatively, set type = "response" for returning predictions on the scale of the response variable.
...	further arguments passed to or from other methods.

Value

There are two types of the output of predict.tsglm function. By setting type = "link", it returns the values of the linear predictors; and by setting type = "response", it returns the the expected values of response variable. For example, for a binomial model, the predictions are log-odds (probabilities on logit scale) if type = "link", and type = "response" gives the predicted probabilities of Y=1.

Author(s)

Ping-Yang Chen

See Also

tensorReg2D, summary.tsglm

Examples

```
# Predefined function: sum of hadamard product in each array
`%hp%` <- function(X, B) sapply(1:dim(X)[3], function(i) sum(X[,,i]*B))

# Simulation data
n <- 500 # number of observations
n_P <- 3; n_G <- 64 # dimension of 3-D tensor variables.
n_d <- 1 # number of numerical variable, if n_d == 1, numerical variable equals to intercept.
```

```
beta_True <- rep(1, n_d)
B_True <- c(1,1,1)%*%t(rnorm(n_G)) + c(0, .5, .5)%*%t(rnorm(n_G))
B_True <- B_True / 10
W <- matrix(rnorm(n*n_d), n, n_d); W[,1] <- 1
X <- array(rnorm(n*n_P*n_G), dim=c(n_P, n_G, n))
## Regression
y_R<- as.vector(W%*%beta_True + X%hp%B_True + rnorm(n))
DATA_R <- list(y = y_R, X = X, W = W)
## Binomial
p_B <- exp(W%*%beta_True + X%hp%B_True); p_B <- p_B/(1+p_B)
y_B <- rbinom(n, 1, p_B)
DATA_B <- list(y = y_B, W = W, X = X)
## Poisson
p_P <- exp(W%*%beta_True + X%hp%B_True)
y_P <- rpois(n, p_P)
y_P[which(y_P > 170)] <- 170 # If y_P > 170, factorial(y_P) == inf.
DATA_P <- list(y = y_P, W = W, X = X)

# Execution
## Regression
result_R <- tensorReg2D(y = DATA_R$y, X = DATA_R$X, W=NULL, n_R = 1, family = "gaussian",
opt = 1, max_ite = 100, tol = 10^(-7) )
## Prediction
head(predict(result_R, DATA_R$X))

## Binomial
result_B <- tensorReg2D(y = DATA_B$y, X = DATA_B$X, W=NULL, n_R = 1, family = "binomial",
opt = 1, max_ite = 100, tol = 10^(-7) )
## Prediction
head(predict(result_B, DATA_B$X))

## Poisson
result_P <- tensorReg2D(y = DATA_P$y, X = DATA_P$X, W=NULL, n_R = 1, family = "poisson",
opt = 1, max_ite = 100, tol = 10^(-7) )
## Prediction
head(predict(result_P, DATA_P$X))
```

`summary.tsglm` *Summarizing Second-order Tensor Generalized Regression Fits*

Description

`summary` method for self-defined class "`tsglm`".

Usage

```
## S3 method for class 'tsglm'
summary(object, ...)
```

Arguments

object	an object of class "tsglm".
...	further arguments passed to or from other methods.

Details

summary.tsglm is combined with print to provide formatting the coefficients, standard errors, etc. and additionally gives 'significance stars'

Value

summary.tsglm returns a printout similar to summary.glm.

The printout contains the following components:

Call: The formula for fitted model.

Deviance Residuals: The summary statistics of deviance residuals. Provide for model except family = "gaussian".

Residuals: The summary statistics of residuals. Provide for family = "gaussian".

Coefficients: The coefficient table includes estimation, standard deviation, test statistics, and p-value of parameters and significance stars.

Deviance: The deviance of a fitted model. Provide for model except family = "gaussian".

AIC: Akaike information criterion.

Author(s)

Mark Chen

See Also

summary, predict.tsglm

Examples

```
# Simulation data
n <- 500 # number of observations
n_P <- 3; n_G <- 64 # dimension of 3-D tensor variables.
n_d <- 2 # number of numerical variable, if n_d == 1, numerical variable equals to intercept.
beta_True <- rep(1, n_d)
B_True <- c(1,1,1)%*%t(rnorm(n_G)) + c(0, .5, .5)%*%t(rnorm(n_G))
B_True <- B_True / 10
W <- matrix(rnorm(n*n_d), n, n_d); W[,1] <- 1
W_named <- W
colnames(W_named) <- paste0("w", 1:ncol(W_named))
X <- array(rnorm(n*n_P*n_G), dim=c(n_P, n_G, n))
X_named <- X
dimnames(X_named) <- list(paste0("R", 1:nrow(X_named)),paste0("C", 1:ncol(X_named)))
## Regression
y_R <- as.vector(W%*%beta_True + X%hp%B_True + rnorm(n))
DATA_R <- list(y = y_R, X = X, W = W)
```

```
y_R_named <- as.vector(W_named%*%beta_True + X_named%hp%B_True + rnorm(n))
DATA_R_named <- list(y = y_R_named, X = X_named, W = W_named)
## Binomial
p_B <- exp(W%*%beta_True + X%hp%B_True); p_B <- p_B/(1+p_B)
y_B <- rbinom(n, 1, p_B)
DATA_B <- list(y = y_B, W = W, X = X)
## Poisson
p_P <- exp(W%*%beta_True + X%hp%B_True)
y_P <- rpois(n, p_P)
y_P[which(y_P > 170)] <- 170 # If y_P > 170, factorial(y_P) == inf.
DATA_P <- list(y = y_P, W = W, X = X)

# Execution
## Regression
result_R <- tensorReg2D(y = DATA_R$y, X = DATA_R$X, W=NULL, n_R = 1, family = "gaussian",
opt = 1, max_ite = 100, tol = 10^(-7) )
summary(result_R)
head(predict(result_R, DATA_R$X))

## Regression with specified names
result_R_named <- tensorReg2D(y = DATA_R_named$y, X = DATA_R_named$X, W=DATA_R_named$W,
n_R = 1, family = "gaussian", opt = 1, max_ite = 100, tol = 10^(-7) )
summary(result_R_named)

## Binomial
result_B <- tensorReg2D(y = DATA_B$y, X = DATA_B$X, W=NULL, n_R = 1, family = "binomial",
opt = 1, max_ite = 100, tol = 10^(-7) )
summary(result_B)
head(predict(result_B, DATA_B$X))

## Poisson
result_P <- tensorReg2D(y = DATA_P$y, X = DATA_P$X, W=NULL, n_R = 1, family = "poisson",
opt = 1, max_ite = 100, tol = 10^(-7) )
summary(result_P)
head(predict(result_P, DATA_P$X))
```

`tensorReg2D` *Fitting Second-order Tensor Generalized Regression*

Description

`tensorReg2D` is used to fit second-order tensor generalized regression model. It mainly focus on parameter estimation, including parameter coefficients and standard deviation. The function is built upon **Alternating Least Square Algorithm**, so we provide two criterion to determine optimal result (see more details below in Arguments). Also, we offer model complexity measurement,including AIC and BIC.

Usage

```
tensorReg2D(y, X, W = NULL, n_R, family, opt = 1, max_ite = 100, tol = 10^(-7))
```

Arguments

y	A numerical vector. Dependent variable.
X	A numerical 3-D array Independent variable(3-D tensor).
W	A numerical matrix. Independent variable.
n_R	A numerical constant. A predefined value determines the rank of the approximate matrix
family	Family of generalized linear model. Provide three options for model.(see more details in **Details**)
opt	Optimization options. Provide two options for optimization stopping criterion. **opt = 1 or 2**. (see more details in **Details**)
max_ite	Maximum iteration. The value of maximum iterations for the algorithm.
tol	Tolerance. The value of tolerance with respect to optimization.

Details

tensorReg2D focuses on second-order tensor generalized regression problems. To be more specific, it provides statistical inference for input variables. Moreover, the function isn't restricted to second-order tensor input X; it could combine with other meaningful numerical variables W.

Since tensorReg2D is based on **Alternating Least Square Algorithm**, we need to pre-define following arguments to meet favorable optimization result.

n_R: In the case of regression with the order 2, P-by-G-by-n tensor, we can break a unknown parameter matrix **B**(P-by-G) into multiplication of two matrix **B_1**(P-by-R) and **t(B_2)** (R-by-G), which means that we can estimate the original matrix **B** by iteratively updating **B_1** and **B_2**. In this scenario, n_R equals to the rank of these two approximate matrix **B_1** and **B_2**. Conceivably, `1 <= n_R <= min(P,G)`, and by properly pre-appointing n_R, we can estimate a unknown parameter matrix. By default, `n_R = 1`.

opt: In optimization algorithm, we have to determine stopping criterion. In tensorReg2D, we offer two criteria. `If opt = 1`, the criterion is that we stop our execution when the maximum difference between the elements among an estimated parameter matrix **B** with an estimated parameter vector **b** and preceding ones is less than predefined tolerance (tol) . `If opt = 2`, the criterion is that we stop our execution when the maximum difference between the elements among an estimated approximate parameter matrix **B_1** , **B_2** with an estimated parameter vector **b** and preceding ones is less than predefined tolerance (tol).

family: In tensorReg2D, we provide three options for specific generalized regression problem. First, `family = "gaussian"` using identity link function corresponds to linear regression model, where dependent variable is real number. Next, `family = "binomial"` based on logit link function corresponds to logistic regression, where dependent variable is restricted to zero or one binary variable. Finally, `family = "poisson"` built upon log link function corresponds to poisson regression, where dependent variable is non-negative integer.

max_ite: In optimization algorithm, we have to beforehand determine maximum iteration beforehand. By default, `max_ite = 100`.

tol: In optimization algorithm, we have to beforehand determine maximum tolerance to cooperate with stopping criterion(opt).

Value

tensorReg2D returns an object of "tsglm".

The function, summary.tsglm a customized method from generic function summary, can be used to obtain and print a summary and analysis of variance table of the results.

An object of class tsglm is a list containing at least the following components:

ite: The number of executed times when stopping the function.

b_EST: The estimated coefficients for numerical variables.

b_SD: The estimated standard deviation for numerical variables.

b_PV: The p-value for numerical variables.

B_EST: The estimated coefficients for 3-D tensor variables.

B_SD: The estimated standard deviation for 3-D tensor variables.

B_PV: The p-value for 3-D tensor variables.

Residuals: The differences between true values and prediction values. Provide for family = "gaussian".

Dev_res: Deviance residuals for glm. Provide for model except family = "gaussian".

Dev: The value of Null deviances and Residual deviance. Provide for model except family = "gaussian".

IC: The value of AIC and BIC.

DoF: Degree of freedom.

call: The formula of fitted model.

family: The family for model.

Author(s)

Sheng-Mao Chang

References

Mengyun Wu, Jian Huang, and Shuangge Ma (2017). Identifying gene-gene interactions using penalized tensor regression.

Sheng-Mao Chang, Meng Yang, Wenbin Lu, Yu-Jyun Huang, Yueyang Huang, Hung Hung, Jeffrey C Miecznikowski, Tzu-Pin Lu, Jung-Ying Tzeng, Gene-set integrative analysis of multi-omics data using tensor-based association test, Bioinformatics, 2021;, btab125, (Link))

Examples

```
# Simulation data
n <- 500 # number of observations
n_P <- 3; n_G <- 64 # dimension of 3-D tensor variables.
n_d <- 1 # number of numerical variable, if n_d == 1, numerical variable equals to intercept.
beta_True <- rep(1, n_d)
B_True <- c(1,1,1)%*%t(rnorm(n_G)) + c(0, .5, .5)%*%t(rnorm(n_G))
B_True <- B_True / 10
W <- matrix(rnorm(n*n_d), n, n_d); W[,1] <- 1
```

```
X <- array(rnorm(n*n_P*n_G), dim=c(n_P, n_G, n))
## Regression
y_R<- as.vector(W%*%beta_True + X%hp%B_True + rnorm(n))
DATA_R <- list(y = y_R, X = X, W = W)
## Binomial
p_B <- exp(W%*%beta_True + X%hp%B_True); p_B <- p_B/(1+p_B)
y_B <- rbinom(n, 1, p_B)
DATA_B <- list(y = y_B, W = W, X = X)
## Poisson
p_P <- exp(W%*%beta_True + X%hp%B_True)
y_P <- rpois(n, p_P)
y_P[which(y_P > 170)] <- 170 # If y_P > 170, factorial(y_P) == inf.
DATA_P <- list(y = y_P, W = W, X = X)

# Execution
## Regression
result_R <- tensorReg2D(y = DATA_R$y, X = DATA_R$X, W=NULL, n_R = 1, family = "gaussian",
opt = 1, max_ite = 100, tol = 10^(-7) )
## Visualization
image(B_True);image(result_R$B_EST)
head(predict(result_R, DATA_R$X))

## Binomial
result_B <- tensorReg2D(y = DATA_B$y, X = DATA_B$X, W=NULL, n_R = 1, family = "binomial",
opt = 1, max_ite = 100, tol = 10^(-7) )
## Visualization
image(B_True);image(result_B$B_EST)
head(predict(result_B, DATA_B$X))

## Poisson
result_P <- tensorReg2D(y = DATA_P$y, X = DATA_P$X, W=NULL, n_R = 1, family = "poisson",
opt = 1, max_ite = 100, tol = 10^(-7) )
## Visualization
image(B_True);image(result_P$B_EST)
head(predict(result_P, DATA_P$X))
```

TensorTest2D *The TensorTest2D Package*

Description

TensorTest2D is a tool used to fit second-order tensor type data.

Details

This package contains functions for fitting generalized linear models on second-order tensor type data.

It mainly focus on parameter estimation, including parameter coefficients and standard deviation.

Currently, this package includes generalized linear models with `identity link` (Regression), `logit link` (Logistic regression), and `log link` (Poisson regression).

For a complete list of functions, use `help(package = TensorTest2D)`.

Author(s)

Sheng-Mao Chang <smchang110@gm.ntpu.edu.tw>

Wenbin Lu <wlu4@ncsu.edu>

Jung-Ying Tzeng <jytzeng@ncsu.edu>

Ping-Yang Chen <pychen.ping@gmail.com>

Maintainer: Mark Chen <l501l501l@gmail.com>

`VAR_ALS` *The function computing the covariance matrices of the tensor regression parameters.*

Description

The function computing the covariance matrices of the tensor regression parameters.

Usage

```
VAR_ALS(DATA, n_R, B1, B2, beta, family)
```

Arguments

`DATA`	A list. The input data. `DATA$y` is the dependent variable. `DATA$X` is the 3-D tensor independent variables. `DATA$W` is other independent variables.
`n_R`	A numerical constant. A predefined value determines the rank of the approximate matrix
`B1`	A numerical matrix. Parameter matrix B1 of the tensor regression.
`B2`	A numerical matrix. Parameter matrix B2 of the tensor regression.
`beta`	A numerical vector. Parameter vector of the covariates, W.
`family`	Family of `generalized linear model`. Provide three options for model.(see more details in **Details**)

Value

A list. `V_B`: A numerical matrix. Covariance matrix of the vectorized tensors' parameters.

`V_b`: A numerical matrix. Covariance matrix of the covariates' parameters.

`%b%` *Computation of two matrices: the column of the output matrix is the Kronecker product of two columns of each input matrix.*

Description

Computation of two matrices: the column of the output matrix is the Kronecker product of two columns of each input matrix.

Usage

```
A %b% B
```

Arguments

`A`	A numerical matrix.
`B`	A numerical matrix.

Value

numerical matrix.

`%hp%` *Computation of the vector of Hadamard product values of the matrices* `X[,,i]` *and* `B`.

Description

Computation of the vector of Hadamard product values of the matrices `X[,,i]` and `B`.

Usage

```
X %hp% B
```

Arguments

`X`	A numerical 3D array. Each slice is a matrix of size the same as `B`.
`B`	A numerical matrix.

Value

numerical vector.

Author(s)

Sheng-Mao Chang

`%w%` *Computation of the matrix with rows being the linearized matrix products of* `X[,,i]` *and* `B`.

Description

Computation of the matrix with rows being the linearized matrix products of `X[,,i]` and `B`.

Usage

```
X %w% B
```

Arguments

`X`	A numerical 3D array. Each slice is a matrix of column size the same as the row size of `B`.
`B`	A numerical matrix.

Value

numerical 3D array.

`%wt%` *Computation of the matrix with rows being the linearized matrix products of transposed* `X[,,i]` *and* `B`.

Description

Computation of the matrix with rows being the linearized matrix products of transposed `X[,,i]` and `B`.

Usage

```
X %wt% B
```

Arguments

`X`	A numerical 3D array. Each slice is a matrix of row size the same as the row size of `B`.
`B`	A numerical matrix.

Value

numerical vector.

Index