Package 'catch'

July 22, 2025

Title Covariate-Adjusted Tensor Classification in High-Dimensions

Version 1.0.1

Description Performs classification and variable selection on high-dimensional tensors (multi-dimensional arrays) after adjusting for additional covariates (scalar or vectors) as CATCH model in Pan, Mai and Zhang (2018) <doi:10.48550/arXiv.1805.04421>. The low-dimensional covariates and the high-dimensional tensors are jointly modeled to predict a categorical outcome in a multi-class discriminant analysis setting. The Covariate-Adjusted Tensor Classification in High-dimensions (CATCH) model is fitted in two steps: (1) adjust for the covariates within each class; and (2) penalized estimation with the adjusted tensor using a cyclic block coordinate descent algorithm. The package can provide a solution path for tuning parameter in the penalized estimation step. Special case of the CATCH model includes linear discriminant analysis model and matrix (or tensor) discriminant analysis without covariates.

Depends R (>= 3.1.1)

License GPL-2

Encoding UTF-8

LazyData true

Imports tensr, Matrix, MASS, methods

NeedsCompilation yes

Author Yuqing Pan <yuqing.pan@stat.fsu.edu>,
Qing Mai <mai@stat.fsu.edu>,
Xin Zhang <henry@stat.fsu.edu>

Maintainer Yuqing Pan <yuqing.pan@stat.fsu.edu>

Repository CRAN

Date/Publication 2021-01-04 17:10:02 UTC

Contents

`adjten` *Adjust tensor for covariates.*

Description

Adjusts tensor with respect to covariates to achieve a more accurate performance. Tensor depends on the covariates through a linear regression model. The function returns the coefficients of covariates in regression and adjusted tensor list for further classifier modeling. It estimates coefficients based on training data, and then adjusts training tensor. When testing data is provided, the function will automatically adjust testing data by learned coefficients as well.

Usage

```
adjten(x, z, y, testx = NULL, testz = NULL, is.centered = FALSE)
```

Arguments

`x`	Input tensor or matrix list of length N, where N is the number of observations. Each element of the list is a tensor or matrix. The order of tensor can be any integer not less than 2.
`z`	Input covariate matrix of dimension $N \times q$, where $q < N$. Each row of `z` is an observation.
`y`	Class label vector of dimention $N \times 1$. For `K` class problems, `y` takes values in $\{1, \cdots, \texttt{K}\}$.
`testx`	Input testing tensor or matrix list. Each element of the list is a test case. When `testx` is not provided, the function will only adjust training data.
`testz`	Input testing covariate matrix with each row being an observation.
`is.centered`	Indicates whether the input tensor and covariates have already been centered by their within class mean or not. If `is.centered` is FALSE, the function `adjten` will center data by class. If `is.centered` is TRUE, the function will skip the centering step.

Details

The model CATCH assumes the linear relationship bewteen covariates and tensor as

$$\mathbf{X} = \boldsymbol{\mu}_k + \boldsymbol{\alpha} \overline{\times}_{M+1} \mathbf{Z} + \mathbf{E},$$

where $\boldsymbol{\alpha}$ is the matrix of estimated coefficient of covariates. The function removes the effects of covariates on response variable through tensor and obtain $\mathbf{X} - \boldsymbol{\alpha} \overline{\times}_{M+1} \mathbf{Z}$ as adjusted tensor to fit tensor discriminant analysis model.

In estimating $\boldsymbol{\alpha}$, which is the `alpha` in the package, `adjten` first centers both tensor and covariates within their individual classes, then performs tensor response regression which regresses $\mathbf{X}$ on $\mathbf{Z}$.

Value

gamma	The estimated coefficients of covariates to plug in classifier. gamma is the γ_k defined function catch of dimension $q\times(K-1)$, where q is the size of covariates and K is the number of classes.
xres	Adjusted training tensor list $\mathbf{X}-\boldsymbol{\alpha}\overline{\times}_{M+1}\mathbf{Z}$ after adjusting for covariates. The effect of the covariate is removed.
testxres	Adjusted testing tensor list $\mathbf{X}-\boldsymbol{\alpha}\overline{\times}_{M+1}\mathbf{Z}$ after adjusting for covariates. The effect of the covariate is removed.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

Pan, Y., Mai, Q., and Zhang, X. (2018) *Covariate-Adjusted Tensor Classification in High-Dimensions*, arXiv:1805.04421.

See Also

catch

Examples

```
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars),nrow=n,ncol=nvars)
vec_x[1:10,] <- vec_x[1:10,]+2
z <- matrix(rnorm(n*2),nrow=n,ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,],dim=c(p,p,p))
}
obj <- adjten(x, z, y)
```

---

catch *Fit a CATCH model and predict categorical response.*

Description

The catch function solves classification problems and selects variables by fitting a covariate-adjusted tensor classification in high-dimensions (CATCH) model. The input training predictors include two parts: tensor data and low dimensional covariates. The tensor data could be matrix as a special case of tensor. In catch, tensor data should be stored in a list form. If the dataset contains no covariate, catch can also fit a classifier only based on the tensor predictors. If covariates are provided, the method will adjust tensor for covariates and then fit a classifier based on the adjusted tensor along with the covariates. If users specify testing data at the same time, predicted response will be obtained as well.

Usage

```
catch(x, z = NULL, y, testx = NULL, testz = NULL, nlambda = 100,
lambda.factor = ifelse((nobs - nclass) <= nvars, 0.2, 1E-03),
lambda = NULL,dfmax = nobs, pmax = min(dfmax * 2 + 20, nvars),
pf = rep(1, nvars), eps = 1e-04, maxit = 1e+05, sml = 1e-06,
verbose = FALSE, perturb = NULL)
```

Arguments

x	Input tensor (or matrix) list of length N, where N is the number of observations. Each element of the list is a tensor or matrix. The order of tensor can be any positive integer not less than 2.
z	Input covariate matrix of dimension $N \times q$, where $q < N$. z can be omitted if covariate is absent.
y	Class label. For K class problems, y takes values in $\{1, \cdots, K\}$.
testx	Input testing tensor or matrix list. Each element of the list is a test case. When testx is not provided, the function will only fit the model and return the classifier. When testx is provided, the function will predict response on testx as well.
testz	Input testing covariate matrix. Can be omitted if covariate is absent. However, training covariates z and testing covariates testz must be provided or not at the same time.
nlambda	The number of tuning values in sequence lambda. If users do not specify lambda values, the package will generate a solution path containing nlambda many tuning values of lambda. Default is 100.
lambda.factor	When lambda is not supplied, catch first finds the largest value in lambda which yields $\boldsymbol{\beta} = 0$. Then the minimum value in lambda is obtained by (largest value*lambda.factor). The sequence of lambda is generated by evenly sampling nlambda numbers within the range. Default value of lambda.factor is 0.2 if $N < p$ and 0.0001 if $N > p$.
lambda	A sequence of user-specified lambda values. lambda is the weight of L1 penalty and a smaller lambda allows more variables to be nonzero. If NULL, then the algorithm will generate a sequence of nlambda many potential lambdas according to lambda.factor.

dfmax	The maximum number of selected variables in the model. Default is the number of observations N.
pmax	The maximum number of potential selected variables during iteration. In middle step, the algorithm can select at most pmax variables and then shrink part of them such that the nubmer of final selected variables is less than dfmax. Default is $\min(dfmax \times 2 + 20, N)$.
pf	Weight of lasso penalty. Default is a vector of value 1 and length p, representing L1 penalty of length p. Can be mofidied to use adaptive lasso penalty.
eps	Convergence threshold for coordinate descent difference between iterations. Default value is 1e-04.
maxit	Maximum iteration times for all lambda. Default value is 1e+05.
sml	Threshold for ratio of loss function change after each iteration to old loss function value. Default value is 1e-06.
verbose	Indicates whether print out lambda during iteration or not. Default value is FALSE.
perturb	Perturbation scaler. If it is specified, the value will be added to diagonal of estimated covariance matrix. A small value can be used to accelerate iteration. Default value is NULL.

Details

The catch function fits a linear discriminant analysis model as follows:

$$\mathbf{Z}|(Y=k) \sim N(\boldsymbol{\phi_k}, \boldsymbol{\psi}),$$

$$\mathbf{X}|(\mathbf{Z}=\mathbf{z}, Y=k) \sim TN(\boldsymbol{\mu}_k + \boldsymbol{\alpha}\bar{\times}_{M+1}\mathbf{z}, \boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\Sigma}_M).$$

The categorical response is predicted from the estimated Bayes rule:

$$\widehat{Y} = \arg\max_{k=1,\cdots,K} a_k + \boldsymbol{\gamma}_k^T\mathbf{Z} + < \boldsymbol{\beta}_k, \mathbf{X} - \boldsymbol{\alpha}\overline{\times}_{M+1}\mathbf{Z} >,$$

where $\mathbf{X}$ is the tensor, $\mathbf{Z}$ is the covariates, a_k, $\boldsymbol{\gamma}_k$ and $\boldsymbol{\alpha}$ are parameters estimated by CATCH. A detailed explanation can be found in reference. When Z is not NULL, the function will first adjust tensor on covariates by modeling

$$\mathbf{X} = \boldsymbol{\mu}_k + \boldsymbol{\alpha}\overline{\times}_{M+1}\mathbf{Z} + \mathbf{E},$$

where $\mathbf{E}$ is an unobservable tensor normal error independent of $\mathbf{Z}$. Then catch fits model on the adjusted training tensor $\mathbf{X} - \boldsymbol{\alpha}\overline{\times}_{M+1}\mathbf{Z}$ and makes predictions on testing data by using the adjusted tensor list. If Z is NULL, it reduces to a simple tensor discriminant analysis model.

The coefficient of tensor $\boldsymbol{\beta}$, represented by beta in package, is estimated by

$$\min_{\boldsymbol{\beta}_2,\ldots,\boldsymbol{\beta}_K} \left[\sum_{k=2}^{K} \left(\langle \boldsymbol{\beta}_k, [\![\boldsymbol{\beta}_k; \widehat{\boldsymbol{\Sigma}}_1, \ldots, \widehat{\boldsymbol{\Sigma}}_M]\!] \rangle - 2\langle \boldsymbol{\beta}_k, \widehat{\boldsymbol{\mu}}_k - \widehat{\boldsymbol{\mu}}_1 \rangle \right) + \lambda \sum_{j_1 \ldots j_M} \sqrt{\sum_{k=2}^{K} \beta_{k,j_1\cdots j_M}^2} \right].$$

When response is multi-class, the group lasso penalty over categories is added to objective function through parameter lambda, and it reduces to a lasso penalty in binary problems.

The function catch will predict categorical response when testing data is provided. If testing data is not provided or if one wishes to perform prediction separately, catch can be used to only fit model with a catch object outcome. The object outcome can be combined with the adjusted tensor list from adjten to perform prediction by predict.catch.

Value

beta	Output variable coefficients for each lambda, which is the estimation of $\boldsymbol{\beta}$ in the Bayes rule. beta is a list of length being the number of lambdas. Each element of beta is a matrix of dimension $nvars \times (nclass - 1)$.
df	The number of nonzero variables for each value in sequence lambda.
dim	Dimension of coefficient array.
lambda	The actual lambda sequence used. The user specified sequence or automatically generated sequence could be truncated by constraints on dfmax and pmax.
obj	Objective function value for each value in sequence lambda.
x	The tensor list after adjustment in training data. If covariate is absent, this is the original input tensor list.
y	Class label in training data.
npasses	Total number of iterations.
jerr	Error flag.
sigma	Estimated covariance matrix on each mode. sigma is a list with the ith element being covariance matrix on ith mode.
delta	Estimated delta matrix $(vec(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1), \cdots, vec(\widehat{\boldsymbol{\mu}}_K - \widehat{\boldsymbol{\mu}}_1))$.
mu	Estimated mean array of $\mathbf{X}$.
prior	Proportion of samples in each class.
call	The call that produces this object.
pred	Predicted categorical response for each value in sequence lambda when testx is provided.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

Pan, Y., Mai, Q., and Zhang, X. (2018) *Covariate-Adjusted Tensor Classification in High-Dimensions*, arXiv:1805.04421.

See Also

cv.catch, predict.catch, adjten

Examples

```
#without prediction
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars), nrow=n, ncol=nvars)
```

```
vec_x[1:10,] <- vec_x[1:10,]+2
z <- matrix(rnorm(n*2), nrow=n, ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,],dim=c(p,p,p))
}
obj <- catch(x,z,y=y)
```

`catch_matrix` *Fit a CATCH model for matrix and predict categorical response.*

Description

Fits a classifier for matrix data. `catch_matrix` is a special case of `catch` when each observation $\mathbf{X}_i$ is a matrix. Different from `catch` takes list as input, data need to be formed in an array to call the function (see arguments). The function will perform prediction as well.

Usage

```
catch_matrix(x, z = NULL, y, testx = NULL, testz = NULL, ...)
```

Arguments

`x`	Input matrix array. The array should be organized with dimension $p_1 \times p_2 \times N$.
`z`	Input covariate matrix of dimension $N \times q$, where $q < N$. `z` can be omitted if covariate is absent.
`y`	Class label. For `K` class problems, `y` takes values in $\{1, \cdots, \mathtt{K}\}$.
`testx`	Input testing matrix array. When `testx` is not provided, the function will only fit model. When `testx` is provided, the function will predict response on `testx` as well.
`testz`	Input testing covariate matrix. Can be omitted if there is no covariate.
`...`	Other arguments that can be passed to `catch`.

Details

The function fits a matrix classifier as a special case of `catch`. The fitted model and predictions should be identical to `catch` when matrix data is provided. Input matrix should be organized as three-way array where sample size is the last dimension. If the matrix is organized in a list, users can either reorganize it or use `catch` directly to fit model, which takes a matrix or tensor list as input and has the same output as `catch_matrix`.

Value

beta	Output variable coefficients for each lambda. beta is a list of length being the number of lambdas. Each element of beta is a matrix of dimension $(p_1 \times p_2) \times (nclass - 1)$.
df	The number of nonzero variables for each value in sequence lambda.
dim	Dimension of coefficient array.
lambda	The actual lambda sequence used. The user specified sequence or automatically generated sequence could be truncated by constraints on dfmax and pmax.
obj	Objective function value for each value in sequence lambda.
x	The matrix list after adjustment in training data. If covariate is absent, this is the original input matrix.
y	Class label in training data.
npasses	Total number of iterations.
jerr	Error flag.
sigma	Estimated covariance matrix on each mode. sigma is a list with the ith element being covariance matrix on ith mode.
delta	Estimated delta matrix $(vec(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1), \cdots, vec(\widehat{\boldsymbol{\mu}}_K - \widehat{\boldsymbol{\mu}}_1))$.
mu	Estimated mean array.
prior	Prior proportion of observations in each class.
call	The call that produces this object.
pred	Predicted categorical response for each value in sequence lambda when testx is provided.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

Pan, Y., Mai, Q., and Zhang, X. (2018) *Covariate-Adjusted Tensor Classification in High-Dimensions*, arXiv:1805.04421.

See Also

catch

Examples

```
#without prediction
n <- 20
p <- 4
k <- 2
nvars <- p*p
x=array(rnorm(n*nvars),dim=c(p,p,n))
x[,,11:20]=x[,,11:20]+0.3
```

```
z <- matrix(rnorm(n*2), nrow=n, ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
obj <- catch_matrix(x,z,y=y)
```

---

csa *Colorimetric sensor array (CSA) data*

Description

A dataset collected from a series of CSA experiments to identify volatile chemical toxicants (VCT). Chemical dyes were exposed to VCT under different concentration conditions and colors of dyes were recorded to identify the class of VCT. There are two concentration conditions PEL (permissible exposure level) and IDLH (immediately dangerous to life of health).

Usage

```
data(csa)
```

Format

Two lists, *PEL* and *IDLH*, and a numeric vector y. Each list contains 147 matrices of dimension 36×3.

`PEL` A list of matrices containing the observations after exposure at PEL.

`IDLH` A list of matrices containing the observations after exposure at IDLH level.

`y` Class label ranging from 1 to 21.

Details

This dataset is provided in the Supplementary matrial of Zhong (2015). In each concentration case, there are 147 observations and 21 classes. We reorganize the data into a list to be directly called by `catch`. For matrices in the list, each row represents a dye and the three columns correspond to red, green and blue.

Source

Wenxuan Zhong and Kenneth S. Suslick (2015). "Matrix discriminant analysis with application to colorimetric sensor array data" *Technometrics* **57**(4), 524–534.

`cv.catch` *Cross-validation for CATCH*

Description

Performs k-fold cross validation for CATCH and returns the best tuning parameter λ in the user-specified or automatically generated choices.

Usage

```
cv.catch(x, z = NULL, y, nfolds = 5, lambda = NULL,
lambda.opt = "min",...)
```

Arguments

`x`	Input tensor or matrix list of length N, where N is the number of observations. Each element of the list is a tensor or matrix. The order of tensor can be any number and not limited to three.
`z`	Input covariate matrix of dimension $N \times q$, where $q < N$. `z` can be omitted if covariate is absent.
`y`	Class label. For `K` class problems, `y` takes values in $\{1, \cdots, K\}$.
`nfolds`	Number of folds. Default value is `5`.
`lambda`	User-specified `lambda` sequence for cross validation. If not specified, the algorithm will generate a sequence of `lambda`s based on all data and cross validate on the sequence.
`lambda.opt`	The optimal criteria when multiple elements in `lambda` return the same minimum classification error. `"min"` will return the smallest `lambda` with minimum cross validation error. `"max"` will return the largest `lambda` with the minimum cross validation error.
`...`	Other arguments that can be passed to `catch`.

Details

The function `cv.catch` runs function `catch` `nfolds+1` times. The first one fits model on all data. If `lambda` is specified, it will check if all `lambda` satisfies the constraints of `dfmax` and `pmax` in `catch`. If not, a `lambda` sequence will be generated according to `lambda.factor` in `catch`. Then the rest `nfolds` many replicates will fit model on `nfolds-1` many folds data and predict on the omitted fold, repectively. Return the `lambda` with minimum average cross validation error and the largest `lambda` within one standard error of the minimum.

Value

`lambda`	The actual `lambda` sequence used. The user specified sequence or automatically generated sequence could be truncated by constraints on `dfmax` and `pmax`.
`cvm`	The mean of cross validation errors for each `lambda`.

cvsd	The standard error of cross validaiton errors for each lambda.
lambda.min	The lambda with minimum cross validation error. If lambda.opt is min, then returns the smallest lambda with minimum cross validation error. If lambda.opt is max, then returns the largestlambda with minimum cross validation error.
lambda.1se	The largest lambda with cross validation error within one standard error of the minimum.
catch.fit	The fitted catchobj object.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

Pan, Y., Mai, Q., and Zhang, X. (2018) *Covariate-Adjusted Tensor Classification in High-Dimensions*, arXiv:1805.04421.

See Also

catch

Examples

```
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars), nrow=n, ncol=nvars)
vec_x[1:10,] <- vec_x[1:10,]+2
z <- matrix(rnorm(n*2),nrow=n,ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,], dim=c(p,p,p))
}
objcv <- cv.catch(x, z, y=y)
```

---

predict.catch *Predict categorical responses.*

Description

Predict categorical responses on new data given the fitted model input.

Usage

```
## S3 method for class 'catch'
predict(object, newx, z = NULL, ztest = NULL, gamma = NULL,...)
```

Arguments

object	Input catchobj class object as fitted model.
newx	Input adjusted testing tensor or matrix list. Each element of the list is a tensor. The tensor should of the same dimension as training data.
z	Input training covariates matrix. z can be omitted if there is no covariate.
ztest	Input testing covariates matrix. ztest can be omitted if there is no covariate.
gamma	Coefficients of covariates obtained from adjten. gamma is NULL if there is no covariate.
...	Other arguments that can be passed to predict.

Details

The function fits LDA model on selected discriminant vectors. Call predict or predict.catch to perform predictions.

There are two ways to make predictions. One way is to directly predict at the same time as fitting model by catch since predict.catch has already been embedded in catch and it will predicts response when testing data is provided. The other way is to first use adjten to adjuste tensor and catch to fit model. predict.catch will take the input adjusted tensor list newx, covariate coefficient gamma from adjten and the fitted model from catch to perform prediction. The prediction is identical to providing catch testing data.

Value

pred	Predicted response of newx for each lambda in model object.

Author(s)

Yuqing Pan, Qing Mai, Xin Zhang

References

Pan, Y., Mai, Q., and Zhang, X. (2018) *Covariate-Adjusted Tensor Classification in High-Dimensions*, arXiv:1805.04421.

See Also

catch, adjten

Examples

```
#generate training data
n <- 20
p <- 4
k <- 2
nvars <- p*p*p
x <- array(list(),n)
vec_x <- matrix(rnorm(n*nvars),nrow=n,ncol=nvars)
vec_x[1:10,] <- vec_x[1:10,]+2
```

```
z <- matrix(rnorm(n*2),nrow=n,ncol=2)
z[1:10,] <- z[1:10,]+0.5
y <- c(rep(1,10),rep(2,10))
for (i in 1:n){
  x[[i]] <- array(vec_x[i,],dim=c(p,p,p))
}

#generate testing data
newx <- array(list(),n)
vec_newx <- matrix(rnorm(n*nvars),nrow=n,ncol=nvars)
vec_newx[1:10,] <- vec_newx[1:10,]+2
newz <- matrix(rnorm(n*2),nrow=n,ncol=2)
newz[1:10,] <- newz[1:10,]+0.5
for (i in 1:n){
  newx[[i]] <- array(vec_newx[i,],dim=c(p,p,p))
}

#Make adjustment and fit model
obj <- adjten(x, z, y, newx, newz)
fit <- catch(x, z, y)
#Predict
pred <- predict.catch(fit, obj$testxres, z, newz, obj$gamma)

#The adjusting, fitting model and predicting step can also be completed
#by one command.
pred <- catch(x, z, y, newx, newz)$pred
```

Index