Package ‘orthoDr’

September 5, 2019

Type Package

Title Semi-Parametric Dimension Reduction Models Using Orthogonality Constrained Optimization

Version 0.6.4

Author Ruilin Zhao, Ruoqing Zhu, Jiyang Zhang, Wenzhuo Zhou and Peng Xu

Maintainer Ruoqing Zhu <teazrq@gmail.com>

Description Utilize an orthogonality constrained optimization algorithm of Wen & Yin (2013) <DOI:10.1007/s10107-012-0584-1> to solve a variety of dimension reduction problems in the semiparametric framework, such as Ma & Zhu (2012) <DOI:10.1080/01621459.2011.646925>, Ma & Zhu (2013) <DOI:10.1214/12-AOS1072>, Sun, Zhu, Wang & Zeng (2017) <arXiv:1704.05046> and Zhou & Zhu (2018+) <arXiv:1802.06156>. It also serves as a general purpose optimization solver for problems with orthogonality constraints. Parallel computing for approximating the gradient is enabled through `OpenMP'.

License GPL (>= 2)

Encoding UTF-8

LazyData TRUE

RoxygenNote 6.1.1

NeedsCompilation yes

Repository CRAN

Imports Rcpp (>= 0.12.12), survival, dr, pracma, plot3D, rgl, MASS

LinkingTo Rcpp, RcppArmadillo

Date/Publication 2019-09-05 12:30:02 UTC

R topics documented:

CP_SIR 2
distance 3
dist_cross 4
hMave 5

kernel_weight 6
orthoDr_pdose 6
orthoDr_reg 8
orthoDr_surv 10
ortho_optim 11
predict.orthoDr 13
pSAVE 14
silverman 15
skcm.clinical 16
skcm.melgene 16
view_dr_surv 17

Index **18**

CP_SIR *Counting process based sliced inverse regression model*

Description

The CP-SIR model for right-censored survival outcome. This model is correct only under very strong assumptions, however, since it only requires an SVD, the solution is used as the initial value in the orthoDr optimization.

Usage

```
CP_SIR(x, y, censor, bw = silverman(1, length(y)))
```

Arguments

`x`	A matrix for features (continuous only).
`y`	A vector of observed time.
`censor`	A vector of censoring indicator.
`bw`	Kernel bandwidth for nonparametric estimations (one-dimensional), the default is using Silverman's formula.

Value

A list consisting of

`values`	The eigenvalues of the estimation matrix
`vectors`	The estimated directions, ordered by eigenvalues

References

Sun, Q., Zhu, R., Wang, T. and Zeng, D. "Counting Process Based Dimension Reduction Method for Censored Outcomes." (2017) `https://arxiv.org/abs/1704.05046` .

Examples

```
# This is setting 1 in Sun et. al. (2017) with reduced sample size
library(MASS)
set.seed(1)
N = 200; P = 6
V=0.5^abs(outer(1:P, 1:P, "-"))
dataX = as.matrix(mvrnorm(N, mu=rep(0,P), Sigma=V))
failEDR = as.matrix(c(1, 0.5, 0, 0, 0, rep(0, P-5)))
censorEDR = as.matrix(c(0, 0, 0, 1, 1, rep(0, P-5)))
T = rexp(N, exp(dataX %*% failEDR))
C = rexp(N, exp(dataX %*% censorEDR - 1))
ndr = 1
Y = pmin(T, C)
Censor = (T < C)

# fit the model
cpsir.fit = CP_SIR(dataX, Y, Censor)
distance(failEDR, cpsir.fit$vectors[, 1:ndr, drop = FALSE], "dist")
```

`distance` *distance correlation*

Description

Calculate the distance correlation between two linear spaces

Usage

```
distance(s1, s2, type = "dist", x = NULL)
```

Arguments

`s1`	first space
`s2`	second space
`type`	type of distance measures: "dist" (default), "trace", "canonical" or "sine"
`x`	the covariate values, for canonical correlation only

Value

The distance between `s1` and `s2`.

Examples

```
# two spaces
failEDR = as.matrix(cbind(c(1, 1, 0, 0, 0, 0),
                          c(0, 0, 1, -1, 0, 0)))
B = as.matrix(cbind(c(0.1, 1.1, 0, 0, 0, 0),
                    c(0, 0, 1.1, -0.9, 0, 0)))
```

```
distance(failEDR, B, "dist")
distance(failEDR, B, "trace")

N=300
P=6
dataX = matrix(rnorm(N*P), N, P)
distance(failEDR, B, "canonical", dataX)
```

---

dist_cross *dist_cross*

Description

Cross distance matrix. An extension to the dist() function. Calculate the Gaussian kernel distance between rows of X1 and rows of X2

Usage

```
dist_cross(x1, x2)
```

Arguments

x1	first data matrix
x2	second data matrix

Value

A distance matrix, with its (i, j)th element being the Gaussian kernel distance between ith row of `X1` jth row of `X2`.

Examples

```
# two matrices
set.seed(1)
x1 = matrix(rnorm(10), 5, 2)
x2 = matrix(rnorm(6), 3, 2)
dist_cross(x1, x2)
```

hMave *Hazard Mave for Censored Survival Data*

Description

This is an almost direct R translation of Xia, Zhang & Xu's (2010) hMave Matlab code. We implemented further options for setting a different initial value. The computational algorithm does not utilize the orthogonality constrained optimization.

Usage

```
hMave(x, y, censor, m0, B0 = NULL)
```

Arguments

`x`	A matrix for features.
`y`	A vector of observed time.
`censor`	A vector of censoring indicator.
`m0`	number of dimensions to use
`B0`	initial value of B. This is a feature we implemented.

Value

A list consisting of

`B`	The estimated B matrix
`cv`	Leave one out cross-validation error

References

Xia, Y., Zhang, D., & Xu, J. (2010). Dimension reduction and semiparametric estimation of survival models. Journal of the American Statistical Association, 105(489), 278-290. `http://dx.doi.org/10.1198/jasa.2009.tm09372`.

Examples

```
# generate some survival data
set.seed(1)
P = 7
N = 150
dataX = matrix(runif(N*P), N, P)
failEDR = as.matrix(cbind(c(1, 1.3, -1.3, 1, -0.5, 0.5, -0.5, rep(0, P-7))))
T = exp(dataX %*% failEDR + rnorm(N))
C = runif(N, 0, 15)
Y = pmin(T, C)
Censor = (T < C)

# fit the model
hMave.fit = hMave(dataX, Y, Censor, 1)
```

kernel_weight *kernel_weight*

Description

Calculate the Gaussian kernel weights between rows of X1 and rows of X2

Usage

```
kernel_weight(x1, x2, kernel = "gaussian", dist = "euclidean")
```

Arguments

x1	first data matrix
x2	second data matrix
kernel	the kernel function, currently only using Gaussian kernel
dist	the distance metric, currently only using the Euclidean distance

Value

A distance matrix, with its (i, j)th element being the kernel weights for the ith row of `X1` jth row of `X2`.

Examples

```
# two matrices
set.seed(1)
x1 = matrix(rnorm(10), 5, 2)
x2 = matrix(rnorm(6), 3, 2)
kernel_weight(x1, x2)
```

orthoDr_pdose *orthoDr_pdose model*

Description

The "Direct Learning & Pseudo-direct Learning" Method for personalized medicine.

Usage

```
orthoDr_pdose(x, a, r, ndr = ndr, B.initial = NULL, bw = NULL,
  lambda = 0.1, K = sqrt(length(r)), method = c("direct",
  "pseudo_direct"), keep.data = FALSE, control = list(),
  maxitr = 500, verbose = FALSE, ncore = 0)
```

Arguments

`x`	A matrix or data.frame for features (continuous only).
`a`	A vector of observed dose
`r`	A vector of observed reward
`ndr`	A dimension structure
`B.initial`	Initial `B` values. Will use the counting process based SIR model CP_SIR as the initial if leaving as `NULL`. If specified, must be a matrix with `ncol(x)` rows and `ndr` columns. Will be processed by Gram-Schmidt if not orthogonal
`bw`	A Kernel bandwidth, assuming each variables have unit variance
`lambda`	The penalty level for kernel ridge regression. If a range of values is specified, the GCV will be used to select the best tuning
`K`	A number of grids in the range of dose
`method`	A method the user will implement
`keep.data`	Should the original data be kept for prediction
`control`	A list of tuning variables for optimization. `epsilon` is the size for numerically appriximating the gradient. For others, see Wen and Yin (2013).
`maxitr`	Maximum number of iterations
`verbose`	Should information be displayed
`ncore`	the number of cores for parallel computing

Value

A `orthoDr` object; a list consisting of

`B`	The optimal `B` value
`fn`	The final functional value
`itr`	The number of iterations
`converge`	convergence code

References

Zhou, W., Zhu, R. "A Parsimonious Personalized Dose Model vis Dimension Reduction." (2018+) `https://arxiv.org/abs/1802.06156`.

Wen, Z. and Yin, W., "A feasible method for optimization with orthogonality constraints." Mathematical Programming 142.1-2 (2013): 397-434. DOI: `https://doi.org/10.1007/s10107-012-0584-1`

Examples

```
# generate some personalized dose scenario

exampleset <- function(size,ncov){

 X = matrix(runif(size*ncov,-1,1),ncol=ncov)
 A = runif(size,0,2)
 Edr =  as.matrix(c(0.5,-0.5))

 D_opt = X %*% Edr +1

 mu = 2 + 0.5*(X %*% Edr) - 7*abs(D_opt-A)

 R = rnorm(length(mu),mu,1)

 R = R - min(R)

 datainfo = list(X=X,A=A,R=R,D_opt=D_opt,mu=mu)
 return(datainfo)
}

# generate data

set.seed(123)
n = 150
p = 2
ndr =1
train = exampleset(n,p)
test = exampleset(500,p)

# the direct learning method
 orthofit = orthoDr_pdose(train$X,train$A,train$R,ndr = ndr,lambda = 0.1,
                       method = "direct", K = sqrt(n),keep.data = TRUE,
                       maxitr = 150, verbose = FALSE, ncore = 2)

dose = predict(orthofit,test$X)

dosedistance = mean((test$D_opt-dose$pred)^2)
print(dosedistance)

# the pseudo direct learning method
orthofit = orthoDr_pdose(train$X,train$A,train$R,ndr = ndr,lambda = seq(0.1,0.2,0.01),
                     method = "pseudo_direct", K = as.integer(sqrt(n)), keep.data = TRUE,
                       maxitr = 150, verbose = FALSE, ncore = 2)

dose = predict(orthofit,test$X)

# compare with the optimal dose

dosedistance = mean((test$D_opt-dose$pred)^2)
print(dosedistance)
```

orthoDr_reg *orthoDr_reg*

Description

The semiparametric dimension reduction method from Ma & Zhu (2012).

Usage

```
orthoDr_reg(x, y, method = "sir", ndr = 2, B.initial = NULL,
  bw = NULL, keep.data = FALSE, control = list(), maxitr = 500,
  verbose = FALSE, ncore = 0)
```

Arguments

`x`	A matrix or data.frame for features (continous only). The algorithm will not scale the columns to unit variance
`y`	A vector of continuous outcome
`method`	Dimension reduction methods (semi-): `sir`, `save`, `phd`, `local` or `seff`. Currently only `sir` and `phd` are available.
`ndr`	The number of directions
`B.initial`	Initial `B` values. If specified, must be a matrix with `ncol(x)` rows and `ndr` columns. Will be processed by Gram-Schmidt if not orthogonal. If the initial value is not given, three initial values (`sir`, `save` and `phd`) using the traditional method will be tested. The one with smallest l2 norm of the estimating equation will be used.
`bw`	A Kernel bandwidth, assuming each variables have unit variance
`keep.data`	Should the original data be kept for prediction. Default is `FALSE`
`control`	A list of tuning variables for optimization. `epsilon` is the size for numerically approximating the gradient. For others, see Wen and Yin (2013).
`maxitr`	Maximum number of iterations
`verbose`	Should information be displayed
`ncore`	Number of cores for parallel computing. The default is the maximum number of threads.

Value

A `orthoDr` object; a list consisting of

`B`	The optimal `B` value
`fn`	The final functional value
`itr`	The number of iterations
`converge`	convergence code

References

Ma, Y., & Zhu, L. (2012). A semiparametric approach to dimension reduction. Journal of the American Statistical Association, 107(497), 168-179. DOI: `https://doi.org/10.1080/01621459.2011.646925`.

Ma, Y., & Zhu, L. (2013). Efficient estimation in sufficient dimension reduction. Annals of statistics, 41(1), 250. DOI: 10.1214/12-AOS1072 `https://projecteuclid.org/euclid.aos/1364302742`

Wen, Z. and Yin, W., "A feasible method for optimization with orthogonality constraints." Mathematical Programming 142.1-2 (2013): 397-434. DOI: `https://doi.org/10.1007/s10107-012-0584-1`.

Examples

```
# generate some regression data
set.seed(1)
N = 100; P = 4; dataX = matrix(rnorm(N*P), N, P)
Y = -1 + dataX[,1] + rnorm(N)
# fit the semi-sir model
orthoDr_reg(dataX, Y, ndr = 1, method = "sir")
# fit the semi-phd model
Y = -1 + dataX[,1]^2 + rnorm(N)
orthoDr_reg(dataX, Y, ndr = 1, method = "phd")
```

`orthoDr_surv` *IR-CP model*

Description

The counting process based semiparametric dimension reduction (IR-CP) model for right censored survival outcome.

Usage

```
orthoDr_surv(x, y, censor, method = "dm", ndr = ifelse(method ==
  "forward", 1, 2), B.initial = NULL, bw = NULL, keep.data = FALSE,
  control = list(), maxitr = 500, verbose = FALSE, ncore = 0)
```

Arguments

`x`	A matrix or data.frame for features. The algorithm will not scale the columns to unit variance
`y`	A vector of observed time
`censor`	A vector of censoring indicator
`method`	Which estimating equation to use: should be `forward` (1-d model), `dn` (counting process) or `dm` (martingale)
`ndr`	The number of directions
`B.initial`	Initial `B` values. Will use the counting process based SIR model CP_SIR as the initial if leaving as `NULL`. If specified, must be a matrix with `ncol(x)` rows and `ndr` columns. Will be processed by Gram-Schmidt if not orthogonal
`bw`	A Kernel bandwidth, assuming each variables have unit variance
`keep.data`	Should the original data be kept for prediction. Default is `FALSE`
`control`	A list of tuning variables for optimization. `epsilon` is the size for numerically approximating the gradient. For others, see Wen and Yin (2013).
`maxitr`	Maximum number of iterations
`verbose`	Should information be displayed
`ncore`	Number of cores for parallel computing. The default is the maximum number of threads.

Value

A orthoDr object; a list consisting of

B	The optimal B value
fn	The final functional value
itr	The number of iterations
converge	convergence code

References

Sun, Q., Zhu, R., Wang, T. and Zeng, D. "Counting Process Based Dimension Reduction Method for Censored Outcomes." (2017) DOI: https://arxiv.org/abs/1704.05046.

Wen, Z. and Yin, W., "A feasible method for optimization with orthogonality constraints." Mathematical Programming 142.1-2 (2013): 397-434. DOI: https://doi.org/10.1007/s10107-012-0584-1

Examples

```
# This is setting 1 in Sun et. al. (2017) with reduced sample size
library(MASS)
set.seed(1)
N = 200; P = 6
V=0.5^abs(outer(1:P, 1:P, "-"))
dataX = as.matrix(mvrnorm(N, mu=rep(0,P), Sigma=V))
failEDR = as.matrix(c(1, 0.5, 0, 0, 0, rep(0, P-5)))
censorEDR = as.matrix(c(0, 0, 0, 1, 1, rep(0, P-5)))
T = rexp(N, exp(dataX %*% failEDR))
C = rexp(N, exp(dataX %*% censorEDR - 1))
ndr = 1
Y = pmin(T, C)
Censor = (T < C)

# fit the model
forward.fit = orthoDr_surv(dataX, Y, Censor, method = "forward")
distance(failEDR, forward.fit$B, "dist")

dn.fit = orthoDr_surv(dataX, Y, Censor, method = "dn", ndr = ndr)
distance(failEDR, dn.fit$B, "dist")

dm.fit = orthoDr_surv(dataX, Y, Censor, method = "dm", ndr = ndr)
distance(failEDR, dm.fit$B, "dist")
```

---

ortho_optim *Orthogonality constrained optimization*

Description

A general purpose optimization solver with orthogonality constraint. The orthogonality constrained optimization method is a nearly direct translation from Wen and Yin (2010)'s Matlab code.

Usage

```
ortho_optim(B, fn, grad = NULL, ..., maximize = FALSE,
  control = list(), maxitr = 500, verbose = FALSE)
```

Arguments

`B`	Initial `B` values. Must be a matrix, and the columns are subject to the orthogonality constrains. Will be processed by Gram-Schmidt if not orthogonal
`fn`	A function that calculate the objective function value. The first argument should be `B`. Returns a single value.
`grad`	A function that calculate the gradient. The first argument should be `B`. Returns a matrix with the same dimension as `B`. If not specified, then numerical approximation is used.
`...`	Arguments passed to `fn` and `grad`
`maximize`	By default, the solver will try to minimize the objective function unless `maximize = TRUE`
`control`	A list of tuning variables for optimization. `epsilon` is the size for numerically approximating the gradient. For others, see Wen and Yin (2013).
`maxitr`	Maximum number of iterations
`verbose`	Should information be displayed

Value

A `orthoDr` object; a list consisting of

`B`	The optimal `B` value
`fn`	The final functional value
`itr`	The number of iterations
`converge`	convergence code

References

Wen, Z. and Yin, W., "A feasible method for optimization with orthogonality constraints." Mathematical Programming 142.1-2 (2013): 397-434. DOI: https://doi.org/10.1007/s10107-012-0584-1

Examples

```
# an eigen value problem
library(pracma)
set.seed(1)
n = 100; k = 6
A = matrix(rnorm(n*n), n, n)
A = t(A) %*% A
B = gramSchmidt(matrix(rnorm(n*k), n, k))$Q
fx <- function(B, A) -0.5 * sum(diag(t(B) %*% A %*% B ))
gx <- function(B, A) -A %*% B
fit = ortho_optim(B, fx, gx, A = A)
```

```
fx(fit$B, A)

# compare with the solution from the eigen function
sol = eigen(A)$vectors[, 1:k]
fx(sol, A)
```

`predict.orthoDr` *predict.orthoDr*

Description

The prediction function for orthoDr fitted models

Usage

```
## S3 method for class 'orthoDr'
predict(object, testx, ...)
```

Arguments

`object`	A fitted orthoDr object
`testx`	Testing data
`...`	...

Value

The predicted object

Examples

```
# generate some survival data
N = 100; P = 4; dataX = matrix(rnorm(N*P), N, P)
Y = exp(-1 + dataX[,1] + rnorm(N))
Censor = rbinom(N, 1, 0.8)

# fit the model with keep.data = TRUE
orthoDr.fit = orthoDr_surv(dataX, Y, Censor, ndr = 1,
                           method = "dm", keep.data = TRUE)

#predict 10 new observations
predict(orthoDr.fit, matrix(rnorm(10*P), 10, P))

# generate some personalized dose scenario

exampleset <- function(size,ncov){

 X = matrix(runif(size*ncov,-1,1),ncol=ncov)
 A = runif(size,0,2)
 Edr =  as.matrix(c(0.5,-0.5))

 D_opt = X %*% Edr + 1

 mu = 2 + 0.5*(X %*% Edr) - 7*abs(D_opt-A)

 R = rnorm(length(mu),mu,1)

 R = R - min(R)

 datainfo = list(X=X,A=A,R=R,D_opt=D_opt,mu=mu)
 return(datainfo)
}

# generate data

set.seed(123)
n = 150
p = 2
ndr =1
train = exampleset(n,p)
test = exampleset(500,p)

# the direct learning method
orthofit = orthoDr_pdose(train$X, train$A, train$R, ndr = ndr, lambda = 0.1,
                      method = "direct", K = as.integer(sqrt(n)), keep.data = TRUE,
                      maxitr = 150, verbose = FALSE, ncore = 2)

predict(orthofit,test$X)

# the pseudo direct learning method
orthofit = orthoDr_pdose(train$X, train$A, train$R, ndr = ndr, lambda = seq(0.1,0.2,0.01),
                     method = "pseudo_direct", K = as.integer(sqrt(n)), keep.data = TRUE,
                      maxitr = 150, verbose = FALSE, ncore = 2)

predict(orthofit,test$X)
```

---

pSAVE *Partial Sliced Averaged Variance Estimation*

Description

The partial-SAVE model. This model is correct only under very strong assumptions, the solution is used as the initial value in the orthoDr optimization.

Usage

```
pSAVE(x, a, r, ndr = 2, nslices0 = 2)
```

Arguments

`x`	A matrix for features (continuous only).
`a`	A vector of observed dose levels (continuous only).
`r`	A vector of reward (outcome).
`ndr`	The dimension structure
`nslices0`	Number of slides used for save

Value

A list consisting of

`vectors`	The basis of central subspace, ordered by eigenvalues

References

Feng, Z., Wen, M.X, Yu, Z. and Zhu L. "On Partial Sufficient Dimension Reduction With Applications to Partially Linear Multi-Index Models" (2013) https://arxiv.org/abs/1704.05046 .

`silverman` *A simple Silverman bandwidth formula*

Description

Silverman bandwidth

Usage

```
silverman(d, n)
```

Arguments

`d`	Number of dimension
`n`	Number of observation

Value

A simple bandwidth choice

Examples

```
silverman(1, 300)
```

`skcm.clinical` *skcm.clinical*

Description

The clinical variables of the SKCM dataset. The original data was obtained from The Cancer Genome Atlas (TCGA).

Usage

```
skcm.clinical
```

Format

Contains 469 subjects with 156 failures. Each row contains one subject, subject ID is indicated by row name. Variables include `Time`, `Censor`, `Gender` and `Age`. `Age` has 8 missing values.

References

https://cancergenome.nih.gov/

`skcm.melgene` *skcm.melgene*

Description

The expression of top 20 genes of cutaneous melanoma literature based on the MelGene Database.

Usage

```
skcm.melgene
```

Format

Each row contains one subject, subject ID is indicated by row name. Gene names in the columns. The columns are scaled.

References

Chatzinasiou, Foteini, Christina M. Lill, Katerina Kypreou, Irene Stefanaki, Vasiliki Nicolaou, George Spyrou, Evangelos Evangelou et al. "Comprehensive field synopsis and systematic meta-analyses of genetic association studies in cutaneous melanoma." Journal of the National Cancer Institute 103, no. 16 (2011): 1227-1235.

http://bioinformatics.cing.ac.cy/MelGene/

https://cancergenome.nih.gov/

view_dr_surv *2D or 2D view of survival data on reduced dimension*

Description

Produce 2D or 3D plots of right censored survival data based on a given dimension reduction space

Usage

```
view_dr_surv(x, y, censor, B = NULL, bw = NULL, FUN = "log",
  type = "2D", legend.add = TRUE, xlab = "Reduced Direction",
  ylab = "Time", zlab = "Survival")
```

Arguments

x	A matrix or data.frame for features (continuous only). The algorithm will not scale the columns to unit variance
y	A vector of observed time
censor	A vector of censoring indicator
B	The dimension reduction subspace, can only be 1 dimensional
bw	A Kernel bandwidth (3D plot only) for approximating the survival function, default is the Silverman's formula
FUN	A scaling function applied to the time points y. Default is "log".
type	2D or 3D plot
legend.add	Should legend be added (2D plot only)
xlab	x axis label
ylab	y axis label
zlab	z axis label

References

Sun, Q., Zhu, R., Wang, T. and Zeng, D. "Counting Process Based Dimension Reduction Method for Censored Outcomes." (2017) https://arxiv.org/abs/1704.05046.

Examples

```
# generate some survival data
N = 100; P = 4; dataX = matrix(rnorm(N*P), N, P)
Y = exp(-1 + dataX[,1] + rnorm(N))
Censor = rbinom(N, 1, 0.8)

orthoDr.fit = orthoDr_surv(dataX, Y, Censor, ndr = 1, method = "dm")
view_dr_surv(dataX, Y, Censor, orthoDr.fit$B)
```

Index

∗Topic **skcm.clinical**
    skcm.clinical, 16
∗Topic **skcm.melgene**
    skcm.melgene, 16

CP_SIR, 2, *7*, *10*

dist_cross, 4
distance, 3

hMave, 5

kernel_weight, 6

ortho_optim, 11
orthoDr_pdose, 6
orthoDr_reg, 8
orthoDr_surv, 10

predict.orthoDr, 13
pSAVE, 14

silverman, 15
skcm.clinical, 16
skcm.melgene, 16

view_dr_surv, 17