Package 'simsl'

October 4, 2019

Type Package

Title Single-Index Models with a Surface-Link

Version 0.1.0

Author Park, H., Petkova, E., Tarpey, T., Ogden, R.T.

Maintainer Hyung Park <parkh15@nyu.edu>

Description The package simsl implements a single-index regression for optimizing an individualized dose rule from an observational study. To model interaction effects between baseline covariates and a treatment variable defined on a continuum, we employ two-dimensional penalized spline regression on an index-treatment domain, where the index is defined as a linear combination of the covariates (a single-index). An unspecified main effect of the covariates on the outcomes is allowed. A unique contribution of this work is in the parsimonious single-index parametrization specifically defined for the interaction effect term. We refer to Park, Petkova, Tarpey, and Ogden (2020) <doi: 10.1016/j.jspi.2019.05.008> (for the case of a discrete treatment) and ``A single-index model with a surface-link for optimizing individualized dose rules" (preprint, 2019) for detail of the method. The main function of this package is simsl().

License GPL-3

Imports mgcv, stats

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

chicago *Air pollution dataset*

Description

Daily air pollution and death rate data for Chicago

Format

A data frame with 7 columns and 5114 rows; each row refers to one day; the columns correspond to:

death total deaths (per day).

pm10median median particles in 2.5-10 per cubic m

pm25median median particles < 2.5 mg per cubic m (more dangerous).

o3median Ozone in parts per billion

so2median Median Sulpher dioxide measurement

time time in days

tmpd temperature in fahrenheit

Details

The data are from Peng and Welty (2004) and are available from R (R Core Team, 2019) package gamair (Wood, 2019).

The daily death in the city of Chicago is recorded over a number of years (about 14 years). Each observation is a time series of daily mortality counts, indicating the number of deaths that occurred on each day.

Source

The chicago dataset is available from package gamair (Wood, 2019).

References

Peng, R.D. and Welty, L.J. (2004) The NMMAPSdata package. R News 4(2)

Wood, S.N. (2017) Generalized Additive Models: An Introduction with R

Wood, S.N. (2019) gamair: Data for 'GAMs: An introduction with R'. R package version 1.0.2

der.link *A subfunction used in estimation*

Description

This function computes the 1st derivative of the surface-link function with respect to the argument associated with the pure interaction effect term of the smooth, using finite difference.

Usage

```
der.link(g.fit, arg.number = 2, eps = 10^(-6))
```

Arguments

g.fit	a mgcv::gam object
arg.number	the argument of g.fit that is taken derivative with respect to. The default is arg.number=2 (i.e., take deriviative with respect to the single-index).
eps	a small finite difference used in numerical differentiation.

See Also

fit.simsl, simsl

fit.simsl *Single-index models with a surface-link (workhorse function)*

Description

fit.simml is the workhorse function for Single-index models with a surface-link (SIMSL).

Usage

```
fit.simsl(y, A, X, mu.hat = NULL, family = "gaussian", bs = c("ps",
  "ps"), k = c(8, 8), knots = NULL, sp = NULL, method = "GCV.Cp",
  beta.ini = NULL, beta.ini.gam = FALSE, ind.to.be.positive = 1,
  pen.order = 0, lambda = 0, max.iter = 30, eps.iter = 0.01,
  trace.iter = TRUE, scale.X = TRUE, center.X = TRUE,
  si.main.effect = TRUE)
```

Arguments

y	a n-by-1 vector of treatment outcomes; y is assumed to follow an exponential family distribution; any distribution supported by mgcv::gam.
A	a n-by-1 vector of treatment variable; each element represents one of the L(>1) treatment conditions; e.g., c(1,2,1,1,1...); can be a factor-valued.
X	a n-by-p matrix of pre-treatment covarates.
mu.hat	a n-by-1 vector for efficinecy augmentation provided by the user; the defult is NULL; the optimal choice for this vector is h(E(y|X)), where h is the canonical link function.

family	specifies the distribution of y; e.g., "gaussian", "binomial", "poisson"; the defult is "gaussian"; can be any family supported by mgcv::gam.
bs	type of basis for representing the treatment-specific smooths; the defult is "ps" (p-splines); any basis supported by mgcv::gam can be used, e.g., "cr" (cubic regression splines)
k	basis dimension; the same number (k) is used for all treatment groups, however, the smooths of different treatments have different roughness parameters.
knots	a list containing user specified knot values to be used for basis construction (for the treatment and the index variables, respectively).
sp	a vector of smoothing parameters associated with the 2-dimensional smooth
method	the smoothing parameter estimation method; "GCV.Cp" to use GCV for unknown scale parameter and Mallows' Cp/UBRE/AIC for known scale; any method supported by mgcv::gam can be used.
beta.ini	an initial solution of beta.coef; a p-by-1 vector; the defult is NULL.
beta.ini.gam	if TRUE, employ a mgcv::gam smooth function representation of the variable A effect when inializing beta.coef; otherwise use a linear model representation for the A effect at initialization.
ind.to.be.positive	for identifiability of the solution beta.coef, we restrict the jth component of beta.coef to be positive; by default j=1.
pen.order	0 indicates the ridge penalty; 1 indicates the 1st difference penalty; 2 indicates the 2nd difference penalty, used in a penalized least squares (LS) estimation of beta.coef.
lambda	a regularziation parameter associated with the penalized LS of beta.coef.
max.iter	an integer specifying the maximum number of iterations for beta.coef update.
eps.iter	a value specifying the convergence criterion of algorithm.
trace.iter	if TRUE, trace the estimation process and print the differences in beta.coef.
scale.X	if TRUE, scale X to have unit variance.
center.X	if TRUE, center X to have zero mean.
si.main.effect	if TRUE, once the convergece in the estimates of beta.coef is reached, include the main effect associated with the fitted single-index (beta.coef'X) to the final surface-link estimate.

Details

The function estimates a linear combination (a single-index) of covariates X, and captures a non-linear interactive structure between the single-index and the treatment defined on a continuum via a smooth surface-link on the index-treatment domain.

SIMSL captures the effect of covariates via a single-index and their interaction with the treatment via a 2-dimensional smooth link function. Interaction effects are determined by shapes of the link function. The model allows comparing different individual treatment levels and constructing individual treatment rules, as functions of a biomarker signature (single-index), efficiently utilizing information on patient's characteristics. The resulting simsl object can be used to estimate an optimal dose rule for a new patient with pretreatment clinical information.

Value

a list of information of the fitted SIMSL including

beta.coef	the estimated single-index coefficients.
g.fit	a mgcv:gam object containing information about the estimated 2-dimensional link function.
beta.ini	the initial value used in the estimation of beta.coef
beta.path	solution path of beta.coef over the iterations
d.beta	records the change in beta.coef over the solution path, beta.path
X.scale	sd of pretreatment covariates X
X.center	mean of pretreatment covariates X
A.range	range of the observed treatment variable A
p	number of baseline covariates X
n	number of subjects

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

pred.simsl, fit.simsl

---

pred.simsl *SIMSL prediction function*

Description

This function makes predictions from an estimated SIMSL, given a (new) set of covariates. The function returns a set of predicted outcomes given the treatment values in a dense grid of treatment levels for each individual, and a recommended treatment level (assuming a larger value of the outcome is better).

Usage

```
pred.simsl(simsl.obj, newx, newA = NULL, L = 30, type = "response",
  maximize = TRUE)
```

Arguments

simsl.obj	a simsl object
newx	a (n-by-p) matrix of new values for the covariates X at which predictions are to be made.
newA	a (n-by-L) matrix of new values for the treatment A at which predictions are to be made.
L	when newA=NULL, a value specifying the length of the grid of A at which predictions are to be made.
type	the type of prediction required; the default "response" is on the scale of the response variable; the alternative "link" is on the scale of the linear predictors.
maximize	the default is TRUE, assuming a larger value of the outcome is better; if FALSE, a smaller value is assumed to be prefered.

Value

pred.new	a (n-by-L) matrix of predicted values; each column represents a treatment option.
trt.rule	a (n-by-1) vector of suggested treatment assignments

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

simsl,fit.simsl

---

simsl *Single-index models with a surface-link (main function)*

Description

simsl is the wrapper function for fitting a single-index model with a surface-link (SIMSL). The function estimates a linear combination (a single-index) of baseline covariates X, and models a nonlinear interactive structure between the single-index and a treatment variable defined on a continuum, via estimating a smooth link function on the index-treatment domain.

Usage

```
simsl(y, A, X, mu.hat = NULL, family = "gaussian", bs = c("ps",
  "ps"), k = c(8, 8), knots = NULL, sp = NULL, method = "GCV.Cp",
  beta.ini = NULL, beta.ini.gam = FALSE, ind.to.be.positive = 1,
  pen.order = 0, lambda = 0, max.iter = 30, eps.iter = 10^{     -2
  }, trace.iter = TRUE, center.X = TRUE, scale.X = TRUE,
  si.main.effect = TRUE, bootstrap = FALSE, nboot = 200,
  boot.conf = 0.95, seed = 1357)
```

Arguments

y	a n-by-1 vector of treatment outcomes; y is assumed to follow an exponential family distribution; any distribution supported by mgcv::gam.
A	a n-by-1 vector of treatment variable; each element is assumed to take a value on a continuum.
X	a n-by-p matrix of baseline covarates.
mu.hat	a n-by-1 vector of the fitted main effect term of the model provided by the user; the defult is NULL and it is taken as a vector of zeros; the optimal choice for this vector is h(E(y|X)), where h is the canonical link function.
family	specifies the distribution of y; e.g., "gaussian", "binomial", "poisson"; the defult is "gaussian"; can be any family supported by mgcv::gam.
bs	type of basis for representing the treatment-specific smooths; the defult is "ps" (p-splines); any basis supported by mgcv::gam can be used, e.g., "cr" (cubic regression splines)

k	basis dimension; the same number (k) is used for all treatment groups, however, the smooths of different treatments have different roughness parameters.
knots	a list containing user specified knot values to be used for basis construction (for the treatment and the index variables, respectively).
sp	a vector of smoothing parameters associated with the 2-dimensional smooth
method	the smoothing parameter estimation method; "GCV.Cp" to use GCV for unknown scale parameter and Mallows' Cp/UBRE/AIC for known scale; any method supported by mgcv::gam can be used.
beta.ini	an initial solution of beta.coef; a p-by-1 vector; the defult is NULL.
beta.ini.gam	if TRUE, employ a mgcv::gam smooth function representation of the variable A effect when inializing beta.coef; otherwise use a linear model representation for the A effect at initialization.
ind.to.be.positive	for identifiability of the solution beta.coef, we restrict the jth component of beta.coef to be positive; by default j=1.
pen.order	0 indicates the ridge penalty; 1 indicates the 1st difference penalty; 2 indicates the 2nd difference penalty, used in a penalized least squares (LS) estimation of beta.coef.
lambda	a regularziation parameter associated with the penalized LS of beta.coef.
max.iter	an integer specifying the maximum number of iterations for beta.coef update.
eps.iter	a value specifying the convergence criterion of algorithm.
trace.iter	if TRUE, trace the estimation process and print the differences in beta.coef.
center.X	if TRUE, center X to have zero mean.
scale.X	if TRUE, scale X to have unit variance.
si.main.effect	if TRUE, once the convergece in the estimates of beta.coef is reached, include the main effect associated with the fitted single-index (beta.coef'X) to the final surface-link estimate.
bootstrap	if TRUE, compute bootstrap confidence intervals for the single-index coefficients, beta.coef; the default is FALSE.
nboot	when bootstrap=TRUE, a value specifying the number of bootstrap replications.
boot.conf	a value specifying the confidence level of the bootstrap confidence intervals; the defult is boot.conf = 0.95.
seed	when bootstrap=TRUE, randomization seed used in bootstrap resampling.

Details

SIMSL captures the effect of covariates via a single-index and their interaction with the treatment via a 2-dimensional smooth link function. Interaction effects are determined by shapes of the link surface. The SIMSL allows comparing different individual treatment levels and constructing individual treatment rules, as functions of a biomarker signature (single-index), efficiently utilizing information on patient's characteristics. The resulting simsl object can be used to estimate an optimal dose rule for a new patient with baseline clinical information.

Value

a list of information of the fitted SIMSL including

beta.coef	the estimated single-index coefficients.

g.fit	a mgcv:gam object containing information about the estimated 2-dimensional link function.
beta.ini	the initial value used in the estimation of beta.coef
beta.path	solution path of beta.coef over the iterations
d.beta	records the change in beta.coef over the solution path, beta.path
X.scale	sd of pretreatment covariates X
X.center	mean of pretreatment covariates X
A.range	range of the observed treatment variable A
p	number of baseline covariates X
n	number of subjects
boot.ci	boot.conf-level bootstrap CIs (LB, UB) associated with beta.coef
boot.mat	a (nboot x p) matrix of bootstrap estimates of beta.coef

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

pred.simsl, fit.simsl

Examples

```
set.seed(1234)
n <- 200
n.test <- 500

## simulation 1
# generate training data
p <- 30
X <- matrix(runif(n*p,-1,1),ncol=p)
A <- runif(n,0,2)
f_opt <- 1 + 0.5*X[,2] + 0.5*X[,1]
mu <- 8 + 4*X[,1] - 2*X[,2] - 2*X[,3] - 25*((f_opt-A)^2)
y <- rnorm(length(mu),mu,1)
# fit SIMSL
simsl.obj <- simsl(y=y, A=A, X=X)

# generate testing data
X.test <- matrix(runif(n.test*p,-1,1),ncol=p)
A.test <- runif(n.test,0,2)
f_opt.test <- 1 + 0.5*X.test[,2] + 0.5*X.test[,1]
pred <- pred.simsl(simsl.obj, newx= X.test)  # make prediction based on the estimated SIMSL
value <- mean(8 + 4*X.test[,1] - 2*X.test[,2] - 2*X.test[,3] - 25*((f_opt.test- pred$trt.rule)^2))
value  # the "value" of the estimated treatment rule; the "oracle" value is 8.

## simulation 2
p <- 10
# generate training data
X = matrix(runif(n*p,-1,1),ncol=p)
A = runif(n,0,2)
```

```
f_opt = I(X[,1] > -0.5)*I(X[,1] < 0.5)*0.6 + 1.2*I(X[,1] > 0.5) +
 1.2*I(X[,1] < -0.5) + X[,4]^2 + 0.5*log(abs(X[,7])+1) - 0.6
mu =   8 + 4*cos(2*pi*X[,2]) - 2*X[,4] - 8*X[,5]^3 - 15*abs(f_opt-A)
y = rnorm(length(mu),mu,1)
Xq <- cbind(X, X^2)  # include a quadratic term
# fit SIMSL
simsl.obj <- simsl(y=y, A=A, X=Xq)

# generate testing data
X.test = matrix(runif(n.test*p,-1,1),ncol=p)
A.test = runif(n.test,0,2)
f_opt.test = I(X.test[,1] > -0.5)*I(X.test[,1] < 0.5)*0.6 + 1.2*I(X.test[,1] > 0.5) +
 1.2*I(X.test[,1] < -0.5) + X.test[,4]^2 + 0.5*log(abs(X.test[,7])+1) - 0.6
Xq.test <- cbind(X.test, X.test^2)
pred <- pred.simsl(simsl.obj, newx= Xq.test)  # make prediction based on the estimated SIMSL
value <- mean(8 + 4*cos(2*pi*X.test[,2]) - 2*X.test[,4] - 8*X.test[,5]^3 -
              15*abs(f_opt.test-pred$trt.rule))
value  # the "value" of the estimated treatment rule; the "oracle" value is 8.




### air pollution data application
#data(chicago); head(chicago)
#chicago <- chicago[,-3][complete.cases(chicago[,-3]), ]
##plot(chicago$death)
##chicago$death[2856:2859]
#chicago <- chicago[-c(2856:2859), ]  # get rid of the gross outliers in y
##plot(chicago$pm10median)
#chicago <- chicago[-which.max(chicago$pm10median), ] # get rid of the gross outliers in x

## create lagged variables
#lagard <- function(x,n.lag=5) {
#  n <- length(x); X <- matrix(NA,n,n.lag)
#  for (i in 1:n.lag) X[i:n,i] <- x[i:n-i+1]
#  X
#}
#chicago$pm10 <- lagard(chicago$pm10median)
#chicago <- chicago[complete.cases(chicago), ]
## create season varaible
#chicago$time.day <- round(chicago$time %%  365)
#
## fit SIMSL for modeling the season-by-pm10 interactions on their effects on outcomes
#simsl.obj <- simsl(y = chicago$death, A = chicago$time.day, X=chicago[,7], bs= c("cc", "ps"),
#                   beta.ini.gam = TRUE, family=poisson(), method = "REML")
#simsl.obj$beta.coef  # the estimated single-index coefficients
#summary(simsl.obj$g.fit)
#simsl.obj.boot <- simsl(y = chicago$death, A = chicago$time.day, X=chicago[,7],
#                        bs= c("cc", "ps"), family=poisson(), beta.ini.gam = TRUE,
#                        method = "REML", bootstrap = TRUE, nboot=5)  # nboot =500
#simsl.obj.boot$boot.ci


#par(mfrow=c(1,3), mar=c(5.1,4.7,4.1,2.1))
#additive.fit  <- mgcv::gam(chicago$death ~
#                        s(simsl.obj$g.fit$model[,3], k=8, bs="ps") +
#                        s(chicago$time.day, k=8, bs="cc"),
```

```
#                        family = poisson(), method = "REML")
#plot(additive.fit, shift= additive.fit$coefficients[1], select=2,
#    ylab= "Linear predictor", xlab= "A", main = expression(paste("Individual A effect")))
#plot(additive.fit, shift= additive.fit$coefficients[1], select = 1,
#     xlab= expression(paste(beta*minute,"x")), ylab= " ",
#     main = expression(paste("Individual ", beta*minute,"x effect")))
#par(mar=c(2.1,2.5,4.1,2.1))
#mgcv::vis.gam(simsl.obj$g.fit, view=c("A","single.index"), theta=-135, phi = 30,color="heat", se=1,
#      ylab = "single-index", zlab = " ", main=expression(paste("Interaction surface ")))



### Warfarin data application
#data(warfarin)
#X <- warfarin$X
#A <- warfarin$A
#y <- -abs(warfarin$INR - 2.5)  # the target INR is 2.5
#X[,1:3] <- scale(X[,1:3]) # standardize continuous variables

## Estimate the main effect, using an additive model for continous variables and
## a linear model for the indicator variables
#mu.fit <- mgcv::gam(y-mean(y)  ~ X[, 4:13] +
#                    s(X[,1], k=5, bs="ps")+
#                    s(X[,2], k=5, bs="ps") +
#                    s(X[,3], k=5, bs="ps"), method="REML")
#summary(mu.fit)
#mu.hat <- predict(mu.fit)
## fit SIMSL (we do not scale/center X for the interpretabilty of the indicator variables in X).
#simsl.obj <- simsl(y, A, X, mu.hat=mu.hat, scale.X = FALSE, center.X=FALSE, method="REML")
#simsl.obj$beta.coef
## can also compute bootstrap CIs for the single-index coefficients (beta.coef)
#simsl.obj.boot <- simsl(y, A, X, mu.hat=mu.hat, scale.X=FALSE, center.X=FALSE,
#                        bootstrap = TRUE, nboot=5, method="REML")  # nboot = 500
#simsl.obj.boot$boot.ci


###
#par(mfrow=c(1,3), mar=c(5.1,4.7,4.1,2.1))
#additive.fit  <- mgcv::gam(y-mu.hat ~
#                         s(A, k=8, bs="ps") +
#                         s(simsl.obj$g.fit$model[,3], k=8, bs="ps"),
#                       method = "REML" )
#plot(additive.fit, shift= additive.fit$coefficients[1], select=1,
#     ylab= "Y", main = expression(paste("Individual A effect")))
#plot(additive.fit, shift= additive.fit$coefficients[1], select=2,
#     xlab= expression(paste(beta*minute,"x")), ylab= " ",
#     main = expression(paste("Individual ", beta*minute,"x effect")))
#par(mar=c(2.1,2.5,4.1,2.1))
#mgcv::vis.gam(simsl.obj$g.fit, view=c("A","single.index"), theta=55, phi = 30,color="heat", se=1,
#      ylab = "single-index", zlab = "Y", main=expression(paste("Interaction surface ")))
```

`warfarin` *Warfarin dataset*

Description

The dataset provided by International Warfarin Pharmacogenetics Consortium et al. (2009). Warfarin is an anticoagulant agent widely used as a medicine to treat blood clots and prevent forming new harmful blood clots.

Format

A list containing `INR`, `A`, `X`:

INR a vector of treatment outcomes of the study (INR; International Normalized Ratio)

A a vector of therapeutic warfarin dosages

X a data frame consist of 13 patient characteristics

Details

The dataset onsists of 1780 subjects (after removing patients with missing data and data cleaning), including information on patient covariates (X), final therapeutic warfarin dosages (A), and patient outcomes (INR, International Normalized Ratio).

There are 13 covariates in the dataset: height (X1), weight (X2), age (X3), use of the cytochrome P450 enzyme inducers (X4; the enzyme inducers considered in this analysis includes phenytoin, carbamazepine, and rifampin), use of amiodarone (X5), gender (X6; 1 for male, 0 for female), African or black race (X7), Asian race (X8), the VKORC1 A/G genotype (X9), the VKORC1 A/A genotype (X10), the CYP2C9 1/2 genotype (X11), the CYP2C9 1/3 genotype (X12), and the other CYP2C9 genotypes (except the CYP2C9 1/1 genotype which is taken as the baseline genotype) (X13).

The details of these covariate information are given in International Warfarin Pharmacogenetics Consortium et al. (2009).

Source

The data can be downloaded from https://www.pharmgkb.org/downloads/.

References

International Warfarin Pharmacogenetics Consortium, Klein, T., Altman, R., Eriksson, N., Gage, B., Kimmel, S., Lee, M., Limdi, N., Page, D., Roden, D., Wagner, M., Caldwell, M., and Johnson, J. (2009). Estimation of the warfarin dose with clinical and pharmacogenetic data. The New England Journal of Medicine 360:753–674

Chen, G., Zeng, D., and Kosorok, M. R. (2016). Personalized dose finding using outcome wieghted learning. Journal of the American Medical Association 111:1509–1547.

Index