Package: mvTargetOpt (via r-universe)

May 18, 2026

Type Package

Title Multivariate multi-target optimization

Version 0.2.0

Author Andreas Maendle

Maintainer Andreas Maendler <maendle@uni-bremen.de>

Description Multi-objective optimization algorithm. A new approach, using ideas from PCA dimension reduction and PLS regression.

License GPL-3

Encoding UTF-8

LazyData true

Imports plsdepot

RoxygenNote 7.0.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository https://amaendle.r-universe.dev

Date/Publication 2020-06-08 15:34:39 UTC

RemoteUrl https://github.com/amaendle/mvTargetOpt

RemoteRef HEAD

RemoteSha 501c5ab317afa38ec84d27210d372b6d7e80bc51

Contents

`autosolve` *autosolve*

Description

Applies the iterative optimization algorithm suggested in this package. To perform the simulations automatically, the true model has to be specified. When in practice the true model is unknown, use `opt.onestep` function instead to get a candidate for the unknown optimum in a single iteration.

Usage

```
autosolve(
  startx,
  tgmean,
  tgerr,
  reps = 25,
  maxit = 10,
  reality = foo,
  xeps = 0.01,
  pplot = FALSE,
  pcnr = c(1, 2),
  maxarea = NULL,
  useweights = TRUE,
  mknormweights = F,
  gr2retlimit = T,
  mindeg = 0,
  sequential = F,
  tgpcawg = 1,
```

```
  yweights = F,
  datlim = NULL,
  knearest = NULL,
  tgdim = 1,
  ylast = NULL,
  sto = T,
  mod.sd = NULL,
  ...
)
```

Arguments

startx	numeric matrix, start values
tgmean	numeric v4ector, target value vector
tgerr	numeric vector, defines acceptable error range
reps	integer, number of repeated measurements
maxit	integer, maximum number of iterations
reality	function, real model
xeps	nunmeric, smallest (reasonably) distinguishable epsilon
pplot	boolean, diagnostic plots
pcnr	integer vector, defines which principal directions will be considered
maxarea	numeric matrix, area range, which will be explored
useweights	boolean
mknormweights	boolean
gr2retlimit	boolean
mindeg	integer, minimal degree of order of polynomial model
sequential	boolean
tgpcawg	numeric
yweights	boolean
datlim	NULL or integer
knearest	integer
tgdim	integer
ylast	integer
sto	boolean
mod.sd	numeric
...	

Value

data.frame

Examples

```
# example 1: 2x2 MModell
tfoo <- function(x) {
  x1 <- x[,1]
  x2 <- x[,2]
  return( data.frame( y1=0.8*x1 - 1.2*x2,
                      y2=0.8*x1 - 1.2*abs(x2)^0.25
 ) ) }
dstgmean <- tfoo(cbind(1.35,1.4))
tgerr <- c(0.025,0.025)
xeps <- 0.01
startx <- expand.grid(x1=c(-1,3),x2=c(-1,3))
set.seed(123)
autosolve(startx,dstgmean,tgerr,reps=7,maxit=10,tfoo, xeps, F, pcnr=c(1,2), mod.sd=0.2)

# example 2, 3x3 model
set.seed(123)
startx <- data.frame(x1=runif(4,-4,4),x2=runif(4,-4,4),x3=runif(4,-4,4))
tfoo <- function(x) {
  x1 <- x[,1]
  x2 <- x[,2]
  x3 <- x[,3]
  return( data.frame( y1=0.8*x1 - 1.2*x2,
                      y2=0.8*x1 - 1.2*abs(x2)^0.25,
                      y3=0.8*x1 - 0.6*abs(x2)^0.25 + x3
 ) ) }
tgmean <- tfoo(cbind(1.35,1.4,1.5))#tfoo(cbind(0.35,0.4,0.5))
tgerr <- c(0.5,0.5,0.5)
tmp<-autosolve(startx,tgmean,tgerr*0.125,reps=4,maxit=6,tfoo, xeps=0.01, F, pcnr=c(1,2), mod.sd=0.2)
```

`degByBIC` *degByBIC*

Description

internal function, determines the polynomial model order <=`maxorder` for the data `dat` which minimizes the BIC information criterion

Usage

```
degByBIC(dat, maxorder = 5, weights = NULL, mindeg = 0)
```

Arguments

`dat`	list containing the data for predictors `dat$x` and descriptors `dat$y`
`maxorder`	integer, maximal order of polynomial model
`weights`	vector of weights for the data in `dat`
`mindeg`	integer, minimal order of polynomial model

Value

integer, recommended degree for polynomial model

`euclw` *euclw*

Description

Takes a matrix/data.frame `x` with each coloumn having the scores corresponding to a principal component. Computes weights as needed in 1d regression models, i.e. returns for each of the 1d-projections on the PC the Euclidean distances from the n-dimensional point in the n-space.

Usage

```
euclw(x, normalize = T, sto = T)
```

Arguments

`x`	matrix, data.frame
`normalize`	boolean, `TRUE` for normalized weights.
`sto`	boolean, if `TRUE` ignore higher principal component coordinates.

Value

matrix

`ftweights` *ftweights*

Description

Internal function (for `pred.patch`, `ored.solution`). Replaces infinite numbers (weights) by large finite values. (clipping)

Usage

```
ftweights(w1)
```

Arguments

`w1`	numeric vector

Value

numeric vector

getroots2 *getroots2*

Description

getroots2

Usage

```
getroots2(dat, degree, target, limit = 40, weights = NULL, retlimit = FALSE)
```

Arguments

dat	matrix
degree	integer
target	numeric vector
limit	integer
weights	numeric vector
retlimit	boolean

mimascores *mimascores*

Description

Internal function

Usage

```
mimascores(maxarea, mwgs)
```

Arguments

maxarea	matrix
mwgs	numeric

Value

matrix

mimascores2 *mimascores2*

Description

Internal function

Usage

```
mimascores2(maxarea, mwgs)
```

Arguments

maxarea	matrix
mwgs	numeric

Value

matrix

mknorm *mknorm*

Description

internal function, standardizes a vector x by vector of means means and vector of standard deviations sds.

Usage

```
mknorm(x, means = NULL, sds = NULL)
```

Arguments

x	input vector
means	vector of means
sds	vector of standard deviations

Value

standardized vector

mkreg *mkreg*

Description

internal function, inverse of mknorm. Back-transformation of a standardized vector.

Usage

```
mkreg(x, means, sds)
```

Arguments

x	input vector (standardized)
means	vector of means
sds	vector of standard deviations

Value

back-transformed vector

mvdistance *mvdistance*

Description

internal function, computes the Euclidean (euclid=T) or Manhattan distances (euclid=F) between vector y and the rows of matrix xs.

Usage

```
mvdistance(xs, y, euclid = F)
```

Arguments

xs	matrix, each row representing a vector
y	vector
euclid	TRUE for Euclidean distance, otherwise MAnhattan distance

Value

vector of the distances

nclose2mean *nclose2mean*

Description

Internal function. Returns a subset of size `n` of the points in `datx` which are closest to `center`.

Usage

```
nclose2mean(datx, center, n = 1)
```

Arguments

`datx`	matrix
`center`	numeric vector
`n`	integer

Value

matrix

newexp *newexp*

Description

Creates new samples from model function. Used for simulations in the function autosolve as a helper function.

Usage

```
newexp(n = 25, xpos, foo, sd = 0.001)
```

Arguments

`n`	integer, number of repeated measurements
`xpos`	numeric matrix, coordinates at which the measurement takes place
`foo`	function, model for the real relationship
`sd`	, standard deviation

Value

a matrix containing the samples

Examples

```
#not to be used
```

newexp2 *newexp2*

Description

Creates new samples from model function. Used for simulations in the function autosolve as a helper function.

Usage

```
newexp2(n = 25, xpos, foo, sd = 0.001)
```

Arguments

n	integer, number of repeated measurements
xpos	numeric matrix, coordinates at which the measurement takes place
foo	function, model for the real relationship
sd	numeric, standard deviation

Value

a matrix containing the samples

Examples

```
#not to be used
```

opt.onestep *opt.onestep*

Description

Performs one iteration of the implemented optimization procedure. Suggest points for future measurements in order to find a solution that reaches the desired target value. One or more suggested points are returned.

Usage

```
opt.onestep(
  dat,
  tgmean,
  tgerr = NULL,
  xeps = 0.001,
  pcnr,
  maxarea = NULL,
```

```
  ptchoice = 1,
  useweights = TRUE,
  mknormweights = F,
  allpts = F,
  gr2retlimit = TRUE,
  bpcenter = F,
  mindeg = 0,
  wfun = function(x) {     (1/x)^2 },
  sequential = F,
  ptchng = F,
  nptc = 0,
  tgpcawg = 1,
  betterweights = F,
  yweights = F,
  datlim = NULL,
  knearest = NULL,
  tgdim = 1,
  ylast = NULL,
  sto = T,
  ...
)
```

Arguments

dat	data.frame, data set
tgmean	numerig vector, target mean
tgerr	NULL or numeric vector, maximally accepted deviation from target value
xeps	numeric, smallest delta
pcnr	integer vector, numbers the principal components that shall be considered
maxarea	matrix or NULL, maximal area that can/should be explored
ptchoice	integer
useweights	boolean
mknormweights	boolean
allpts	boolean
gr2retlimit	boolean
bpcenter	boolean
mindeg	integer, minimal degree for polynomial mdel
wfun	function, weight function
sequential	boolean
ptchng	boolean
nptc	integer
tgpcawg	numeric
betterweights	boolean

yweights	boolean
datlim	integer or NULL
knearest	integer, if specified only the knearest nearest observations to the target value are considered
tgdim	integer
ylast	integer, if a positive integer is defined, observations from the last ylast iterations are used only
sto	boolean
...	

Value

matrix with recommended points (process parameters for future measurements) in each line

Examples

```
library(mvTargetOpt)
# Let there be the following true model tfoo:
tfoo <- function(x) {
  x1 <- x[,1]
  x2 <- x[,2]
  return( data.frame(y1=0.8*x1 - 1.2*x2,
                     y2=0.8*x1 - 1.2*abs(x2)^0.25) )
}
# assume we have measurements dat taken at startx:
startx <- expand.grid(x1=c(0,6,7,8,9,10,11,12),x2=c(-1,3,4,5,6,7))
dat <- cbind(startx,tfoo(startx))
# assume we want to find process parameters close to tgmean
tgmean <- tfoo(cbind(0.35,0.4))
# make a guess for a solution based on the specified parameters:
opt.onestep(dat, tgmean=tgmean,tgerr=c(0.2,0.2), pcnr=1:2)
```

PIhcheck *PIhcheck*

Description

internal function, determinas a prediction interval for given linear model model with confidence level alpha. currently unused.

Usage

```
PIhcheck(model, alph = 0.05)
```

Arguments

model	linear model
alph	numeric, confidence level

Value

vector, upper and lower prediction intervall

`plotexp` *plotexp*

Description

Function which plots...

Usage

```
plotexp(
  dat,
  target,
  tgerr,
  prediction = NULL,
  model = NULL,
  limit0 = FALSE,
  reality = NULL,
  xlab = "Process parameter (p)",
  ylab = "Descriptor (d)"
)
```

Arguments

`dat`	data.frame, with dat$x matrix of proces parameter data
`target`	numeric vector
`tgerr`	numeric vector
`prediction`	numeric
`model`	function
`limit0`	boolean
`reality`	function

Value

plot

polymodel *polymodel*

Description

internal function, returns polynomial regression model of degree `degree` for the predictor and descriptor data given in `dat`. If `weights` are provided, the weighted regression model is returned.

Usage

```
polymodel(dat, degree = 1, weights = NULL)
```

Arguments

`dat`	list containing the data for predictors `dat$x` and descriptors `dat$y`
`degree`	integer, degree of polynomial regression
`weights`	vector of weights for the data provided in `dat`

Value

linear model

polymodel2 *polymodel*

Description

internal function, returns polynomial regression model of degree `degree` for the predictor and descriptor data given in `dat`. If `weights` are provided, the weighted regression model is returned.

Usage

```
polymodel2(dat, degree, weights = NULL)
```

Arguments

`dat`	list containing the data for predictors `dat$x` and descriptors `dat$y`
`degree`	integer, degree of polynomial regression
`weights`	vector of weights for the data provided in `dat`

Details

like `polymodel`, but here `degree` has no default

Value

linear model

pred.ptch *pred.ptch*

Description

Internal function. Extends the function `opt.onestep`.

Usage

```
pred.ptch(
  shift,
  pcnr,
  pls1,
  mknormweights,
  dat1d.PC12,
  mindeg,
  tgmean_norm,
  gr2retlimit,
  wfun,
  sto
)
```

Arguments

`shift`	numeric vector
`pcnr`	integer vector
`pls1`	result of PLS1-function
`mknormweights`	boolean
`dat1d.PC12`	matrix
`mindeg`	integer
`tgmean_norm`	numeric vector
`gr2retlimit`	boolean
`wfun`	function
`sto`	boolean

ptchoose *ptchoose*

Description

Internal function. Continues search in unexplored space, when no solution can be found.

Usage

```
ptchoose(
  ptchoice = ptchoice,
  dat1d,
  tgmean_norm,
  maxarea,
  xmeans,
  xsds,
  pls1,
  jjj
)
```

Arguments

ptchoice	numeric, 1 to continue search outside the explored parameters
dat1d	data.frame, predictor data must be in dat1d$x
tgmean_norm	numeric vector
maxarea	NULL or matrix
xmeans	numeric vector
xsds	numeric vector
pls1	result of the PLS1 function
jjj	integer, counts how often this function has been called

Value

matrix

repna *repna*

Description

Internal function. Replaces NAs in a vector x by 0.

Usage

```
repna(x)
```

Arguments

x	numeric vector

Value

numeric vector

sumthelowers *sumthelowers*

Description

internal helper function which is used by euclw .

Usage

```
sumthelowers(x)
```

Arguments

x	numeric vector

Value

numeric vector

sumtheothers *sumtheothers*

Description

internal helper function which is used by `euclw` .

Usage

```
sumtheothers(x)
```

Arguments

x	vector

Value

vector

tgpca *tgpca*

Description

Function which performs a principal component analysis (PCA) on the descriptor variable data (in the target space) given by `dat`, In order to choose a certain direction through the target point for the projections, `wg` has to be set to 1 – then the target point is chosen as center for the PCA. If `wg` lies between 0 and 1, pseudo observations at the target point are created such that a ratio of `wg` of the observations are pseudo observations. Then `prcomp` is applied to the standardized data and pseudo data.

Usage

```
tgpca(
  dat,
  tgmean = NULL,
  tgerr = NULL,
  wg = 1,
  wfun,
  mknormweights,
  yweights = F,
  ylast = NULL
)
```

Arguments

dat	matrix, data.frame
tgmean	numeric vector, optional
tgerr	numeric vector, optional
wg	numeric, weight for the target value. If wg equals 1 or 2 then the pca is performed with the target value as center
wfun	function, weight function
mknormweights	unused
yweights	boolean, use weights?
ylast	integer or NULL, if integer, ignore observations older than the last ylast evaluation points.

Value

returns the results of the pca and some extra stuff

Examples

```
tgpca(matrix(rnorm(20),ncol=2), tgmean=c(0,0))
```

---

uniqP *uniqP*

Description

Internal function. Returns a subset of the unique points of newx whgich are not (up to an epsilon xeps) identical to the points in datx.

Usage

```
uniqP(newx, xeps, datx)
```

Arguments

newx	matrix
xeps	numeric
datx	matrix

Value

matrix

Index