Package 'cmenet'

October 12, 2022

Type Package

Title Bi-Level Selection of Conditional Main Effects

Version 0.1.2

Author Simon Mak

Maintainer Simon Mak <sm769@duke.edu>

Description Provides functions for implementing cmenet - a bi-level variable selection method for conditional main effects (see Mak and Wu (2018) <doi:10.1080/01621459.2018.1448828>). CMEs are reparametrized interaction effects which capture the conditional impact of a factor at a fixed level of another factor. Compared to traditional two-factor interactions, CMEs can quantify more interpretable interaction effects in many problems. The current implementation performs variable selection on only binary CMEs; we are working on an extension for the continuous setting.

License GPL (>= 2)

LazyData FALSE

Imports Rcpp (>= 0.12.4), MASS, glmnet, hierNet, sparsenet

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-05-27 07:10:02 UTC

R topics documented:

`cmenet` *Bi-level selection of conditional main effects (fixed parameters)*

Description

`cmenet` performs variable selection of conditional main effects (CMEs) via a bi-level penalization framework, given fixed penalty parameters.

Usage

```
cmenet(xme, xcme, y,
       lambda.sib=exp(seq(from=log(max.lambda),to=log(max.lambda*1e-6),length=20)),
       lambda.cou=exp(seq(from=log(max.lambda),to=log(max.lambda*1e-6),length=20)),
          max.lambda=lambda0.cme(cbind(xme,xcme),y),
          gamma=1/(0.5-tau)+0.001, tau=0.01,
          act.vec=rep(1,ncol(xme)+ncol(xcme)),
          beta0=rep(0,ncol(xme)+ncol(xcme)),
          it.max=250, lambda.flg=T)
```

Arguments

`xme`	An n x p binary model matrix for MEs.
`xcme`	An n x (4*choose(p,2)) model matrix for CMEs.
`y`	An n-length response vector.
`lambda.sib`	Penalty vector for sibling CMEs.
`lambda.cou`	Penalty vector for cousin CMEs.
`max.lambda`	Maximum penalty value.
`gamma`	Bridge parameter in MC+ penalty.
`tau`	Coupling parameter for CMEs.
`act.vec`	A (p+4*choose(p,2))-length binary vector for setting which variables are always active in optimization.
`beta0`	Initial regression coefficients.
`it.max`	Number of optimization iterations.
`lambda.flg`	Use the default option TRUE (unless within `cv.cmenet`).

Value

`coefficients`	Array of regression coefficients (over different `lambda.sib` and `lambda.cou`).
`residuals`	Array of regression residuals (over different `lambda.sib` and `lambda.cou`).
`inter`	Matrix of intercept estimates (over different `lambda.sib` and `lambda.cou`).

References

Mak and Wu (2018). cmenet: a new method for bi-level variable selection of conditional main effects. *Journal of the American Statistical Association*, to appear.

Examples

```
## Not run:
    library(MASS)
n <- 50 #number of observations
p <- 50 #number of main effects

## Simulate model matrix for MEs and CMEs
set.seed(1)
rho <- 0 #correlation
ones <- matrix(1,p,p)
covmtx <- rho*ones+(1-rho)*diag(p)
latmtx <- mvrnorm(n,p,mu=rep(0,p),Sigma=covmtx) #equicorrelated cov. matrix
memtx <- (latmtx>=0)-(latmtx<0) #simulate model matrix for MEs
model.mtx <- full.model.mtx(memtx)$model.mtx #generate model matrix for MEs and CMEs

## Set true model and generate response
num.act <- 2 # two siblings active
num.grp <- 4 # ... within four active groups
ind <- c()
for (ii in 1:num.grp){
  eff <- sample(seq(2*(p-1)),num.act)
  ind <- c(ind, p + eff + (ii-1)*(2*(p-1)))
}
colnames(model.mtx)[ind] # active CMEs

des.mtx <- model.mtx[,ind]
inter <- 12 #intercept
xbtrue <- inter + rowSums(des.mtx)
y <- xbtrue + rnorm(n,sd=1) #response
xme <- model.mtx[,1:p]
xcme <- model.mtx[,(p+1):ncol(model.mtx)]

## Run cmenet
cv.cme <- cv.cmenet(xme, xcme, y, var.names=colnames(model.mtx))

## End(Not run)
```

`cv.cmenet` *Bi-level selection of conditional main effects*

Description

The main function in this package. `cv.cmenet` performs variable selection of conditional main effects (CMEs) via a bi-level penalization framework, with penalty parameters tuned via cross-validation.

Usage

```
cv.cmenet(xme, xcme, y,
```

```
                nfolds = 10, var.names = NULL,
                nlambda.sib=20, nlambda.cou=20, lambda.min.ratio=1e-6,
                ngamma=10, max.gamma=150, ntau=20,
                max.tau=0.01, tau.min.ratio=0.01,
                it.max=250, it.max.cv=25, warm.str="lasso")
```

Arguments

xme	An n x p binary model matrix for MEs.
xcme	An n x (4*choose(p,2)) model matrix for CMEs.
y	An n-length response vector.
nfolds	Number of folds for cross-validation.
var.names	A (p+4*choose(p,2))-length string vector for variable names.
nlambda.sib	Number of values for lambda.sib.
nlambda.cou	Number of values for lambda.cou.
lambda.min.ratio	Smallest value for lambda.sib and lambda.cou, as a fraction of lambda.max (the smallest penalty for which all coefficients are zero).
ngamma	Number of values for gamma.
max.gamma	Maximum value for gamma.
ntau	Number of values for tau.
max.tau	Maximum value for tau.
tau.min.ratio	Smallest value for tau, as a fraction of max.tau.
it.max	Number of optimization iterations.
it.max.cv	Number of optimization iterations in cross-validation.
warm.str	A string indicating which method should be used for warm-starting active variables. Current options include "lasso" (default) and "hierNet".

Value

cme.fit	Fitted cmenet object.
params	Fitted penalty parameters (lambda.sib, lambda.cou, gamma and tau).
select.names	Selected ME and CME variables.
select.idx	Indices for select.names.

References

Mak and Wu (2018). cmenet: a new method for bi-level variable selection of conditional main effects. *Journal of the American Statistical Association*, to appear.

Examples

```
  ## Not run:
library(MASS)
n <- 50 #number of observations
p <- 50 #number of main effects

## Simulate model matrix for MEs and CMEs
set.seed(1)
rho <- 0 #correlation
ones <- matrix(1,p,p)
covmtx <- rho*ones+(1-rho)*diag(p)
latmtx <- mvrnorm(n,p,mu=rep(0,p),Sigma=covmtx) #equicorrelated cov. matrix
memtx <- (latmtx>=0)-(latmtx<0) #simulate model matrix for MEs
model.mtx <- full.model.mtx(memtx)$model.mtx #generate model matrix for MEs and CMEs

## Set true model and generate response
num.act <- 2 # two siblings active
num.grp <- 4 # ... within four active groups
ind <- c()
for (ii in 1:num.grp){
  eff <- sample(seq(2*(p-1)),num.act)
  ind <- c(ind, p + eff + (ii-1)*(2*(p-1)))
}
colnames(model.mtx)[ind] # active CMEs

des.mtx <- model.mtx[,ind]
inter <- 12 #intercept
xbtrue <- inter + rowSums(des.mtx)
y <- xbtrue + rnorm(n,sd=1) #response
xme <- model.mtx[,1:p]
xcme <- model.mtx[,(p+1):ncol(model.mtx)]

#-----------------------------------------------------------------
# Selection of MEs and CMEs:
#-----------------------------------------------------------------

## cmenet (parameters tuned via cross-validation)
cv.cme <- cv.cmenet(xme, xcme, y, var.names=colnames(model.mtx))
fit.cme <- cv.cme$cme.fit
sel.cme <- cv.cme$select.idx
colnames(model.mtx)[ind] #true model
colnames(model.mtx)[sel.cme] #selected effects from cmenet
colnames(model.mtx)[setdiff(sel.cme,ind)] #selected effects not in true model
colnames(model.mtx)[setdiff(ind,sel.cme)] #true effects not in selected model

## lasso
library(glmnet)
cv.las <- cv.glmnet(cbind(xme,xcme),y)
fit.las <- glmnet(cbind(xme,xcme),y)
sel.las <- which(fit.las$beta[,which(cv.las$lambda==cv.las$lambda.min)]!=0)
colnames(model.mtx)[ind] #true model
colnames(model.mtx)[sel.las] #selected effects from lasso
```

```
colnames(model.mtx)[setdiff(sel.las,ind)] #selected effects not in true model
colnames(model.mtx)[setdiff(ind,sel.las)] #true effects not in selected model

## sparsenet
library(sparsenet)
cv.sn <- cv.sparsenet(cbind(xme,xcme),y)
fit.sn <- sparsenet(cbind(xme,xcme),y)
sel.sn <- which(fit.sn$coefficients[[cv.sn$which.min[2]]]$beta[,cv.sn$which.min[1]]!=0)
colnames(model.mtx)[ind] #true model
colnames(model.mtx)[sel.sn] #selected effects from sparsenet
colnames(model.mtx)[setdiff(sel.sn,ind)] #selected effects not in true model
colnames(model.mtx)[setdiff(ind,sel.sn)] #true effects not in selected model

#-------------------------------------------------------------
## Comparison:
#-------------------------------------------------------------

## (a) Misspecifications
length(setdiff(sel.cme,ind)) + length(setdiff(ind,sel.cme)) # cmenet:    25
length(setdiff(sel.las,ind)) + length(setdiff(ind,sel.las)) # lasso:     29
length(setdiff(sel.sn,ind)) + length(setdiff(ind,sel.sn))   # sparsenet: 60

## (b) MSPE
set.seed(1000)
ntst <- 20
latmtx <- mvrnorm(ntst,p,mu=rep(0,p),Sigma=covmtx)
memtx <- (latmtx>=0)-(latmtx<0)
tst.mtx <- full.model.mtx(memtx)$model.mtx
xbtst <- inter + rowSums(tst.mtx[,ind])
ytst <- xbtst + rnorm(ntst,sd=1)

pred.cme <- predictcme(fit.cme,newx=tst.mtx)[,which(cv.cme$lambda.sib==cv.cme$params[1]),
            which(cv.cme$lambda.cou==cv.cme$params[2])]
pred.las <- predict(fit.las,newx=tst.mtx)[,which(cv.las$lambda==cv.las$lambda.min)]
pred.sn <- predict(fit.sn,newx=tst.mtx)[[which(fit.sn$gamma==cv.sn$parms.min[1])]][,
           which(fit.sn$lambda==cv.sn$parms.min[2])]
mean( (ytst-pred.cme)^2 ) # cmenet:    3.61
mean( (ytst-pred.las)^2 ) # lasso:     4.22
mean( (ytst-pred.sn)^2 )  # sparsenet: 4.00



## End(Not run)
```

`full.model.mtx` *Generate full model matrix for MEs and CMEs*

Description

`full.model.mtx` returns the full model matrix for main effects (MEs) and conditional main effects (CMEs).

Usage

```
full.model.mtx(xme)
```

Arguments

xme	An n x p binary model matrix (n observations, p binary MEs).

Value

model.mtx	An n x (p+4*choose(p,2)) full model matrix for MEs and CMEs.
cme.mtx	An n x (4*choose(p,2)) model matrix for only CMEs.

Examples

```
library(MASS)
n <- 50 #number of observations
p <- 50 #number of main effects

## Simulate model matrix for MEs and CMEs
set.seed(1)
rho <- 0 #correlation
ones <- matrix(1,p,p)
covmtx <- rho*ones+(1-rho)*diag(p)
latmtx <- mvrnorm(n,p,mu=rep(0,p),Sigma=covmtx) #equicorrelated cov. matrix
memtx <- (latmtx>=0)-(latmtx<0) #simulate model matrix for MEs
model.mtx <- full.model.mtx(memtx)$model.mtx #generate model matrix for MEs and CMEs
```

---

maize *Maize dataset*

Description

A subset of the maize dataset from Buckler et al. (2009), with n = 150 observations (days to male flowering time) and p = 40 main effects (binary SNP markers).

Usage

```
data(maize)
```

References

Buckler et al. (2009). The genetic architecture of maize flowering time. *Science* 325, 714-718.

Examples

```
  ## Not run:

library(cmenet)
library(hierNet)

## Load data
data(maize) #load in main effects (MEs) and response
xme <- as.matrix(maize[,1:(ncol(maize)-1)])
yy <- as.vector(maize[,ncol(maize)])
nn <- nrow(xme)
pp <- ncol(xme)
model.mtx <- full.model.mtx(xme)$model.mtx #full model matrix
xcme <- model.mtx[,(pp+1):ncol(model.mtx)] #model matrix for conditional main effects (CMEs)

#-----------------------------------------------------------------
## Selection:
#-----------------------------------------------------------------

## cmenet (new analysis: MEs and CMEs)
set.seed(1000)
cv.cme <- cv.cmenet(xme,xcme,yy,var.names=colnames(model.mtx)) #CV fit
cme.dat <- data.frame(y=yy,x=model.mtx[,cv.cme$select.idx])
cme.glm <- lm(y~.,data=cme.dat) #linear model on selected effects
cv.cme$select.names #selected effects
summary(cme.glm)$coefficients[,4] #p-values

## hierNet (traditional analysis: MEs and two-factor interactions)
set.seed(1000)
hnp <- hierNet.path(xme,yy) #hierNet path
cv.hn <- hierNet.cv(hnp,xme,yy) #CV fit
l.opt <- which(hnp$lamlist==cv.hn$lamhat)
me.sel <- (hnp$bp-hnp$bn)[,l.opt]
me.idx <- which(me.sel!=0) #selected main effects
int.sel <- hnp$th[,,l.opt]
int.idx <- which(int.sel!=0,arr.ind=T)
int.idx <- t(apply(int.idx,1,function(xx){sort(xx)}))
int.idx <- unique(int.idx) #selected interactions
model.mtx.hier <- xme[,me.idx] #model matrix on selected effects
for (ll in 1:nrow(int.idx)){
  model.mtx.hier <- cbind(model.mtx.hier, xme[,int.idx[ll,1]]*xme[,int.idx[ll,2]] )
}
int.nm <- sapply(1:nrow(int.idx),function(xx){
  paste0(colnames(xme)[int.idx[xx,1]],colnames(xme)[int.idx[xx,2]])
})
colnames(model.mtx.hier) <- c(colnames(xme)[me.idx],int.nm)
hn.dat <- data.frame(y=yy,x=model.mtx.hier)
hn.glm <- lm(y~.,data=hn.dat) #linear model on selected effects
colnames(model.mtx.hier) #selected effects
summary(hn.glm)$coefficients[,4] #p-values

#-----------------------------------------------------------------
```

```
## Analysis of selected effects:
# (a) cmenet: more parsimonious gene-gene interaction model
#     - hierNet: 66 variables
#     - cmenet:  17 variables
# (b) cmenet: greater insight on the conditional structure of
#     selected MEs from traditional analysis (w/ lower p-values)
#     - hierNet: g38
#     - cmenet:  g11|g38+, g12|g38-, g14|g38+
#     Interpretation:
#     - hierNet: gene 38 is active
#     - cmenet: gene 38 activates genes 11 and 14, and inhibits gene 12
# (c) cmenet: selected CMEs are more interpretable than selected
#     interactions from traditional analysis (w/ lower p-values)
#     - hierNet: g1*g39, g27*g39
#     - cmenet:  g1|g39-, g27|g39-
#     Interpretation:
#     - hierNet: interactions exist b/w g1 & g39, and g27 & g39
#     - cmenet:  gene 39 inhibits gene 1 and gene 27
#-----------------------------------------------------------------

#-----------------------------------------------------------------
## Prediction:
#-----------------------------------------------------------------

## cmenet (new analysis)
set.seed(1111)
test.prop <- 0.5 #
ntrials <- 10 # no. of replications
mspe1 <- rep(NA,ntrials)
for (i in 1:ntrials){

  # sample testing and training data
  foldid = sample(rep(seq(1/test.prop), length=length(yy)))
  yy.tr <- yy[which(foldid!=1)] #training
  xme.tr <- xme[which(foldid!=1),]
  xcme.tr <- xcme[which(foldid!=1),]
  yy.ts <- yy[which(foldid==1)] #testing
  xme.ts <- xme[which(foldid==1),]
  xcme.ts <- xcme[which(foldid==1),]

  # fit cmenet
  cv.cme <- cv.cmenet(xme.tr,xcme.tr,yy.tr,var.names=colnames(model.mtx))
  obj <- cv.cme$cme.fit
  pred <- predictcme(obj,newx=cbind(xme.ts,xcme.ts))
  mspe1[i] <- mean( (yy.ts-pred[,which(cv.cme$lambda.sib==cv.cme$params[1]),
                                which(cv.cme$lambda.cou==cv.cme$params[2])])^2 )
}
mean(mspe1) #avg. mspe = 10.80

## hierNet (traditional analysis)
set.seed(1111)
test.prop <- 0.5 #
ntrials <- 10 # no. of replications
```

```
mspe2 <- rep(NA,ntrials)
for (i in 1:ntrials){

  # sample testing and training data
  foldid = sample(rep(seq(1/test.prop), length=length(yy)))
  yy.tr <- yy[which(foldid!=1)]
  xme.tr <- xme[which(foldid!=1),]
  xcme.tr <- xcme[which(foldid!=1),]
  yy.ts <- yy[which(foldid==1)]
  xme.ts <- xme[which(foldid==1),]
  xcme.ts <- xcme[which(foldid==1),]

  # fit hierNet
  hnfit <- hierNet.path(xme.tr,yy.tr)
  cv.hn <- hierNet.cv(hnfit,xme.tr,yy.tr)
  l.opt <- which(hnfit$lamlist==cv.hn$lamhat)
  mspe2[i] <- mean( (yy.ts-predict(hnfit,newx=xme.ts)[,l.opt])^2 )
}

mean(mspe2) #avg. mspe = 11.31

#-----------------------------------------------------------------
## Analysis of MSPE:
# - cmenet gives lower prediction error, which suggests
#   underlying gene-gene interactions may indeed be conditional
#-----------------------------------------------------------------


## End(Not run)
```

`predictcme` *Predict using a fitted* `cmenet` *object*

Description

`predictcme` performs prediction at new ME settings `newx`, given fitted `cmenet` object.

Usage

```
predictcme(fit.cme,newx)
```

Arguments

`fit.cme`	Fitted object from `cmenet`.
`newx`	An m x p binary matrix for prediction (m new ME settings, p binary MEs).

Examples

```
  ## Not run:

library(MASS)
library(cmenet)
n <- 50 #number of observations
p <- 50 #number of main effects

## Simulate model matrix for MEs and CMEs
set.seed(1)
rho <- 0 #correlation
ones <- matrix(1,p,p)
covmtx <- rho*ones+(1-rho)*diag(p)
latmtx <- mvrnorm(n,p,mu=rep(0,p),Sigma=covmtx) #equicorrelated cov. matrix
memtx <- (latmtx>=0)-(latmtx<0) #simulate model matrix for MEs
model.mtx <- full.model.mtx(memtx)$model.mtx #generate model matrix for MEs and CMEs

## Set true model and generate response
num.act <- 2 # two siblings active
num.grp <- 4 # ... within four active groups
ind <- c()
for (ii in 1:num.grp){
  eff <- sample(seq(2*(p-1)),num.act)
  ind <- c(ind, p + eff + (ii-1)*(2*(p-1)))
}
colnames(model.mtx)[ind] # active CMEs

des.mtx <- model.mtx[,ind]
inter <- 12 #intercept
xbtrue <- inter + rowSums(des.mtx)
y <- xbtrue + rnorm(n,sd=1) #response
xme <- model.mtx[,1:p]
xcme <- model.mtx[,(p+1):ncol(model.mtx)]

## Run cv.cmenet
cv.cme <- cv.cmenet(xme, xcme, y, var.names=colnames(model.mtx))
fit.cme <- cv.cme$cme.fit
sel.cme <- cv.cme$select.idx
colnames(model.mtx)[ind] #true model
colnames(model.mtx)[sel.cme] #selected effects from cmenet
colnames(model.mtx)[setdiff(sel.cme,ind)] #selected effects not in true model
colnames(model.mtx)[setdiff(ind,sel.cme)] #true effects not in selected model

## Prediction
set.seed(1000)
ntst <- 20
latmtx <- mvrnorm(ntst,p,mu=rep(0,p),Sigma=covmtx)
memtx <- (latmtx>=0)-(latmtx<0)
tst.mtx <- full.model.mtx(memtx)$model.mtx
xbtst <- inter + rowSums(tst.mtx[,ind])
ytst <- xbtst + rnorm(ntst,sd=1)
pred.cme <- predictcme(fit.cme,newx=tst.mtx)[,which(cv.cme$lambda.sib==cv.cme$params[1]),
```

```
            which(cv.cme$lambda.cou==cv.cme$params[2])]

## End(Not run)
```

Index