Package 'samTEMsel'

February 1, 2020

Type Package

Title Sparse Additive Models for Treatment Effect-Modifier Selection

Version 0.1.0

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

Maintainer Hyung Park <parkh15@nyu.edu>

Description An implementation of a constrained sparse additive regression for modeling interaction effects between a categorical treatment variable and a set of pretreatment covariates on a scalar-valued outcome; the regression simultaneously conducts treatment effect-modifier variable selection. The method can effectively identify treatment effect-modifiers exhibiting possibly nonlinear interactions with the treatment. The selected pretreatment characteristics and the associated nonzero component functions can be used as a new set of data-driven features for making individualized treatment recommendations in further analysis. We refer to Park, Petkova, Tarpey, and Ogden (2020) <doi:10.1016/j.jspi.2019.05.008> and Park, Petkova, Tarpey, and Ogden (2020) ``A constrained sparse additive model for treatment effect-modifier selection'' (preprint) for detail of the method. The wrapper function of this package is cv.samTEMsel().

License GPL-3

Imports SAM, stats, splines, graphics

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

RemoteType github

RemoteHost api.github.com

RemoteRepo samTEMsel

RemoteUsername syhyunpark

RemoteRef master

RemoteSha b68ea89acd46312d6d624e83472479c0e2893ddf

GithubRepo samTEMsel

GithubUsername syhyunpark

GithubRef master

GithubSHA1 b68ea89acd46312d6d624e83472479c0e2893ddf

NeedsCompilation no

R topics documented:

`cv.samTEMsel` *Sparse Additive Models for Treatment Effect-Modifier Selection (cross-validation function)*

Description

Does k-fold cross-validation for `samTEMsel`, produces a plot, and returns the sequence of the fitted constrained additive models implied by the sequence of regularization parameters `lambda` and the index, `lambda.opt.index`, corresponding to the estimated optimal regularization parameter.

Usage

```
cv.samTEMsel(y, A, X, mu.hat = NULL, d = 3, n.folds = 10,
  nlambda = 50, lambda.min.ratio = 0.01, thol = 1e-05,
  max.ite = 1e+05, regfunc = "L1", row.ordering = NULL,
  cv1sd = FALSE, terms.fit = FALSE, plots = TRUE)
```

Arguments

`y`	a n-by-1 vector of responses
`A`	a n-by-1 vector of treatment variable; each element represents one of the L(>1) treatment conditions; e.g., c(1,2,1,1,3...); can be a factor-valued
`X`	a n-by-p matrix of pretreatment features
`mu.hat`	a n-by-1 vector of the fitted X main effect term of the model provided by the user; defult is `NULL`, in which case `mu.hat` is taken to be a vector of zeros; the optimal choice for this vector is E(y|X)
`d`	number of basis spline functions to be used for each component function; the default value is 3; d=1 corresponds to the linear model
`n.folds`	number of folds for cross-validation; the default is 10.
`nlambda`	total number of lambda values; the default value is 50.
`lambda.min.ratio`	the smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero); the default is 0.01.
`thol`	stopping precision for the coordinate-descent algorithm; the default value is 1e-5.
`max.ite`	number of maximum iterations; the default value is 1e5.
`regfunc`	type of the regularizer; the default is "L1"; can also be "MCP" or "SCAD".

row.ordering	a particular ordering (n-by-1 vector) of the rows (provided by the user) to be used for cross-validation; the default is NULL in which case the row order is randomly shuffled.
cv1sd	if TRUE, an optimal regularization parameter is chosen based on: the mean cross-validated error + 1 SD of the mean cross-validated error, which typically results in an increase in regularization; the defualt is FALSE.
terms.fit	if TRUE, in samTEMsel, compute and store the component-wise fitted vectors (y.hat.list) and partial residuals (resid.list), which are evaluated over a grid of lambda.
plots	if TRUE, produce a cross-validation plot of the estimated mean squared error versus the regulariation parameter index.
lambda	a user-supplied regularization parameter sequence; typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio.

Value

a list of information of the fitted constrained sparse additive model including

samTEMsel.obj	an object of class samTEMsel, which contains the sequence of the fitted constrained additive models implied by the sequence of the regularization parameters lambda; see samTEMsel for detail.
lambda.opt.index	an index number, indicating the index of the estimated optimal regularization parameter in lambda.
nonzero.index	a set of numbers, indicating the indices of estimated nonzero component functions, evalated at the regularization parameter index lambda.opt.index.
func_norm.opt	a p-by-1 vector, indicating the norms of the estimated component functions evaluatd at the regularization parameter index lambda.opt.index, with each element corresponding to the norm of each estimated component function.
cv.storage	a n.folds-by-nlambda matrix of the estimated mean squared errors, with each column corresponding to each of the regularization parameters in lambda and each row corresponding to each of the n.folds folds.
mean.cv	a nlambda-by-1 vector of the estimated mean squared errors, with each element corresponding to each of the regularization parameters in lambda.
sd.cv	a nlambda-by-1 vector of the standard deviation of the estimated mean squared errors, with each element corresponding to each of the regularization parameters in lambda.

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

samTEMsel, predict_samTEMsel, plot_samTEMsel

Examples

```
set.seed(112)
n.train = 300
n.test  = 1000
```

```
n = n.train + n.test
p = 50
A = rbinom(n, 1, 0.5) + 1  # treatment variable taking a value in {1,2} with equal prob.
X = matrix(runif(n*p, -pi/2,pi/2), n, p)  # pretreatment covariates
noise = rnorm(n, 0, 0.5)
# X main effect on y; a highly nonlinear (cosine) function; depends on the first 10 covariates
main.effect = rep(0, n); for(j in 1:10){
  main.effect = main.effect + cos(X[,j])
}
# A-by-X ineraction effect on y; depends only on X1 and X2.
interaction.effect = (A-1.5)*X[,1]  +  2 * (A-1.5)*(cos(X[,2]) - 0.5)
# generate outcome y
y = main.effect  + interaction.effect + noise


# train/test set splitting
train.index = 1:n.train
y.train = y[train.index]
X.train = X[train.index,]
A.train = A[train.index]
y.test  = y[-train.index]
X.test  = X[-train.index,]
A.test  = A[-train.index]


# obtain an optimal regularization parameter by running cv.samTEMsel().
cv.obj = cv.samTEMsel(y.train, A.train, X.train, nlambda = 100)

samTEMsel.obj = cv.obj$samTEMsel.obj
# samTEMsel.obj contains the sequence of fitted models over the grid of lambda
# see also, samTEMsel().

# lambda.opt.index corresponds to the optimal regularization parameter chosen from cv.samTEMsel().
lambda.opt.index = cv.obj$lambda.opt.index
lambda.opt.index

# plot the estimated component function of variable (j=)2, say.
plot_samTEMsel(samTEMsel.obj,  which.index = 2, lambda.index = lambda.opt.index)

# make ITRs for subjects with pretreatment characteristics, X.test
trt.rule = make_ITR(samTEMsel.obj, newX = X.test, lambda.index = lambda.opt.index)$trt.rule
head(trt.rule)

# an (IPWE) estimate of the "value" of this particualr treatment rule, trt.rule:
mean(y.test[A.test==trt.rule])

# compare the above value to the following estimated "values" of "naive" treatment rules:
mean(y.test[A.test==1])   # a rule that assigns everyone to A=1
mean(y.test[A.test==2])   # a rule that assigns everyone to A=2
```

`make_ITR` *make individualized treatment recommendations (ITRs) based on a* `samTEMsel` *object*

Description

The function make_ITR returns individualized treatment decision recommendations for subjects with pretreatment characteristics newX, given a samTEMsel object, samTEMsel.obj, and an (optimal) regularization parameter index, lambda.index.

Usage

```
make_ITR(samTEMsel.obj, newX = NULL, lambda.index = NULL,
  maximize = TRUE)
```

Arguments

samTEMsel.obj	a samTEMsel object, containing the fitted models.
newX	a (n-by-p) matrix of new values for the covariates X at which predictions are to be made; if NULL, X from the training set is used.
lambda.index	an index of the regularization parameter lambda at which predictions are to be made; one can supply lambda.opt.index obtained from the function cv.samTEMsel(); the default is NULL, in which case the predictions are made based on the most non-sparse model.
maximize	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, with each column representing one of the L treatment options.
trt.rule	a (n-by-1) vector of the individualized treatment recommendations

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

samTEMsel,cv.samTEMsel, predict_samTEMsel

---

plot_samTEMsel *plot component functions from a* samTEMsel *object*

Description

Produces plots of the component functions from a samTEMsel object.

Usage

```
plot_samTEMsel(samTEMsel.obj, newX = NULL, newA = NULL,
  scatter.plot = TRUE, lambda.index, which.index, ylims,
  solution.path = FALSE)
```

Arguments

samTEMsel.obj	a samTEMsel object
newX	a (n-by-p) matrix of new values for the covariates X at which plots are to be made; the default is NULL, in which case X is taken from the training set.
newA	a (n-by-1) vector of new values for the treatment A at which plots are to be made; the default is NULL, in which case A is taken from the training set.
scatter.plot	if TRUE, draw scatter plots of partial residuals versus the covariates; these scatter plots are made based on the training observations; the default is TRUE.
lambda.index	an index of the tuning parameter lambda at which plots are to be made; one can supply lambda.opt.index obtained from the function cv.samTEMsel; the default is NULL, in which case plot_samTEMsel utilizes the most non-sparse model.
which.index	this specifies which component functions are to be plotted; the default is all p component functions, i.e., 1:p.
ylims	this specifies the vertical range of the plots, e.g., c(-10, 10).
solution.path	if TRUE, draw the functional norms of the fitted component functions (based on the training set) versus the regularization parameter; the default is FALSE.

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

samTEMsel,predict_samTEMsel, cv.samTEMsel

---

predict_samTEMsel samTEMsel *prediction function*

Description

predict_samTEMsel makes predictions given a (new) set of covariates newX and a (new) vector of treatment indicators newA based on a constrained sparse additive model samTEMsel.obj. Specifically, predict_samTEMsel predicts the responses y based on the X-by-A interaction effect (and the A main effect) portion of the model.

Usage

```
predict_samTEMsel(samTEMsel.obj, newX = NULL, newA = NULL,
  type = "response", lambda.index = NULL, basis = NULL)
```

Arguments

samTEMsel.obj	a samTEMsel object
newX	a (n by p) matrix of new values for the covariates X at which predictions are to be made; if NULL, X from the training set is used.
newA	a (n by 1) vector of new values for the treatment A at which predictions are to be made; if NULL, A from the training set is used.

type	the type of prediction required; the default "response" gives the predicted responses y based on the whole model; the alternative "terms" gives the component-wise predicted responses from each of the p components (and plus the treatment-specific intercepts) of the model.
lambda.index	an index of the tuning parameter lambda at which predictions are to be made; one can supply lambda.opt.index obtained from the function cv.samTEMsel; the default is NULL, in which case the predictions based on the most non-sparse model is returned.
basis	a basis (design) matrix associated with the testing set (newX and newA) provided by the user; the default is NULL; this is only to efficiently implement the cross-validation in cv.samTEMsel.

Value

value	a (n-by-length(lambda.index)) matrix of predicted values; a (n-by-length(lambda.index)*(p+1) matrix of predicted values if type = "terms".

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

samTEMsel,cv.samTEMsel, plot_samTEMsel

---

samTEMsel — *Sparse Additive Models for Treatment Effect-Modifier Selection (main function)*

Description

The function samTEMsel implements estimation of a constrained sparse additve model.

Usage

```
samTEMsel(y, A, X, mu.hat = NULL, d = 3, lambda = NULL,
  nlambda = 50, lambda.min.ratio = 0.01, thol = 1e-05,
  max.ite = 1e+05, regfunc = "L1", terms.fit = FALSE, basis = NULL,
  basisc = NULL)
```

Arguments

y	a n-by-1 vector of responses
A	a n-by-1 vector of treatment variable; each element represents one of the L(>1) treatment conditions; e.g., c(1,2,1,1,3...); can be a factor-valued
X	a n-by-p matrix of pretreatment features
mu.hat	a n-by-1 vector of the fitted X main effect term of the model provided by the user; defult is NULL, in which case mu.hat is taken to be a vector of zeros; the optimal choice for this vector is E(y|X)

d	number of basis spline functions to be used for each component function; the default value is 3; d=1 corresponds to the linear model
lambda	a user-supplied lambda sequence; typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio.
nlambda	total number of lambda values; the default value is 50.
lambda.min.ratio	the smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero); the default is 0.01.
thol	stopping precision; the default value is 1e-5.
max.ite	number of maximum iterations; the default value is 1e5.
regfunc	type of the regularizer; the default is "L1"; can also be "MCP" or "SCAD".
terms.fit	if TRUE, return the component-wise fitted vectors (y.hat.list) and the partial residuals (resid.list), evaluated over a grid of lambda.
basis	default is NULL; one can provide a n-by-d*p*L basis matrix associted with the spline representation of the model; this is to efficiently implement cross-validation in cv.samTEMsel.
basisc	default is NULL; one can provide a n-by-d*p*(L-1) basis matrix associted with the spline representation that incorporates the "orthogonality" constraint; this is only to efficiently implement cross-validation in cv.samTEMsel.

Details

A constrained additive regression model represents the joint effects of treatment and p pretreatment covariates on an outcome via treatment-specific additive component functions defined over the p covariates, subject to the constraint that the expected value of the outcome given the covariates equals zero, while leaving the main effects of the covariates unspecified. Under this flexible representation, the treatment-by-covariates interaction effects are determined by distinct shapes (across treatment levels) of the unspecified component functions. Optimized under a penalized least square criterion with a L1 (or SCAD/MCP) penalty, the constrained additive model can effectively identify/select treatment effect-modifiers (from the p pretreatment covariates) that exhibit possibly nonlinear interactions with the treatment variable; this is achieved by producing a sparse set of estimated component functions. The estimated nonzero component functions (available from the returned samTEMsel object) can be used to make individualized treatment recommendations (ITRs) for future subjects; see also make_ITR for such ITRs.

The regularization path is computed at a grid of values for the regularization parameter lambda.

Value

a list of information of the fitted constrained additive models including

w	the solution path matrix; the estimated d*p*L-by-nlambda coefficient matrix associted with B-splines, with each column corresponding to a regularization parameter in lambda
wc	the solution path matrix; the estimated d*p*(L-1)-by-nlambda coefficient matrix associted with B-splines that incorports the "orthogonality" constraint, with each column corresponding to a regularization parameter in lambda
lambda	a sequence of regularization parameter
d	the number of basis spline functions used for each component function

func_norm	the functional norm matrix (p-by-nlambda) with each column corresponding to a regularization parameter in lambda; since we have p variables, the length of each column is p.
df	the degree of freedom of the solution path (the number of non-zero component functions).
knots	a (d-1)-by-p matrix; each column contains the knots applied to the corresponding variable.
Boundary.knots	a 2-by-p matrix; each column contains the boundary points applied to the corresponding variable.
sse	sums of square errors of the fitted models over the solution path.
intercept	the list of L treatment-specific intercepts; each element of the list is a 1-by-nlambda matrix, with each column corresponding to a regularization parameter in lambda.
X.min	a p-by-1 vector, with each entry corresponding to the minimum of each input variable; used for rescaling in testing.
X.ran	a p-by-1 vector, with each entry corresponding to the range of each input variable; used for rescaling in testing.
residuals	the n-by-nlambda matrix of residuals of the fitted models, with each column corresponding to each regularization parameter in lambda.
y.hat	the n-by-nlambda matrix of fitted values for y, with each column corresponding to each regularization parameter in lambda.
resid.list	the list of p component-wise partial residuals, in which each (the jth) element is a n-by-nlambda matrix of the jth partial residuals, with each column corresponding to each regularization parameter in lambda.
y.hat.list	the list of p component-wise fitted values, in which each (the jth) element is a n-by-nlambda matrix of the jth fitted values, with each column corresponding to each regularization parameter in lambda.
basis	the n-by-d*p*L basis matrix associted with the spline representation of the (unconstrained) additive model.
basisc	the n-by-d*p*(L-1) basis matrix associted with the spline representation of the model that incorporates the "orthogonality" constraint.

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

cv.samTEMsel, predict_samTEMsel, plot_samTEMsel, make_ITR

Examples

```
set.seed(112)
n.train = 400
n.test  = 50
n = n.train + n.test
p = 20
A = rbinom(n, 1, 0.5) + 1  # treatment variable taking a value in {1,2} with equal prob.
X = matrix(runif(n*p, -pi/2,pi/2), n, p)  # pretreatment covariates
noise = rnorm(n, 0, 0.5)
```

```
# X main effect on y; a highly nonlinear (cosine) function; depends on the first 10 covariates
main.effect = rep(0, n); for(j in 1:10){
  main.effect = main.effect + cos(X[,j])
}
# A-by-X ineraction effect on y; depends only on X1 and X2.
interaction.effect = (A-1.5)*X[,1]  +  2 * (A-1.5)*(cos(X[,2]) - 0.5)
# generate outcome y
y = main.effect  + interaction.effect + noise


# train/test set splitting
train.index = 1:n.train
y.train = y[train.index]
X.train = X[train.index,]
A.train = A[train.index]
y.test = y[-train.index]
X.test = X[-train.index,]
A.test = A[-train.index]

# fit samTEMsel() based on the training set
samTEMsel.obj = samTEMsel(y.train, A.train, X.train, nlambda = 50)

# a n.test-by-nlambda matrix of predicted values:
predict_samTEMsel(samTEMsel.obj, newX = X.test, newA = A.test)

# pick a particular lambda.index, say, 10, as the regularization parameter.
lambda.index = 10
# for an optimal selection of lambda.index, see cv.samTEMsel().

# a n.test-by-1 vector of predicted values (given lambda.index = 10)
predict_samTEMsel(samTEMsel.obj, newX = X.test, newA = A.test, lambda.index=lambda.index)

# the estimated L2 norm of the component functions (given lambda.index=10)
samTEMsel.obj$func_norm[,lambda.index]

# p component-wise fitted values (given lambda.index=10); the last column is the intercept
predict_samTEMsel(samTEMsel.obj, X.test, A.test, type="terms", lambda.index =lambda.index)

# can plot the estimated component functions (say, the first two functions, j=1,2)
plot_samTEMsel(samTEMsel.obj, which.index = c(1,2), lambda.index = lambda.index)

# can make inidividualized treatment recommendations (ITRs)
trt.rule = make_ITR(samTEMsel.obj, newX = X.test, lambda.index = lambda.index)$trt.rule
head(trt.rule)

# an (IPWE) estimate of the "value" of this particualr treatment rule, trt.rule:
mean(y.test[A.test==trt.rule])

# compare the above value to the following estimated "values" of "naive" treatment rules:
mean(y.test[A.test==1])   # just assign everyone to A=1
mean(y.test[A.test==2])   # just assign everyone to A=2
```

Index