Package ‘personalized’

November 28, 2017

Type  Package
Title  Estimation and Validation Methods for Subgroup Identification and Personalized Medicine
Version  0.1.2
Description  Provides functions for fitting and validation of subgroup identification and personalized medicine models under the general subgroup identification framework of Chen et al. (2017) <doi:10.1111/biom.12676>. This package is intended for use for both randomized controlled trials and observational studies.

URL  https://jaredhuling.github.io/personalized/

BugReports  https://github.com/jaredhuling/personalized/issues
License  GPL-2
Encoding  UTF-8
LazyData  true
Suggests  knitr, rmarkdown, testthat
Imports  survival, kernlab, methods, foreach
Depends  glmnet (>= 2.0-13), mgcv, gbm, ggplot2, plotly
RoxygenNote  6.0.1
VignetteBuilder  knitr
NeedsCompilation  no
Author  Jared Huling [aut, cre], Aaron Potvien [ctb]
Maintainer  Jared Huling <jaredhuling@gmail.com>
Repository  CRAN
Date/Publication  2017-11-28 09:38:47 UTC
R topics documented:

check.overlap .......................................................... 2
fit.subgroup .............................................................. 4
plot.subgroup_fitted ..................................................... 9
plotCompare ............................................................. 11
predict.subgroup_fitted .................................................. 13
print.subgroup_fitted .................................................. 15
subgroup.effects ....................................................... 16
summarize.subgroups .................................................. 16
summary.subgroup_fitted ............................................. 18
validate.subgroup ..................................................... 19
weighted.ksvm .......................................................... 21

Index 23

check.overlap  Check propensity score overlap

Description

Results in a plot to check whether the propensity score has adequate overlap between treatment groups.

Usage

check.overlap(x, trt, propensity.func, type = c("histogram", "density", "both"), bins = 50L)

Arguments

x  The design matrix (not including intercept term)

trt  treatment vector with each element equal to a 0 or a 1, with 1 indicating treatment status is active.

propensity.func  function that inputs the design matrix x and the treatment vector trt and outputs the propensity score, i.e. Pr(trt = 1 | X = x). Function should take two arguments 1) x and 2) trt. See example below. For a randomized controlled trial this can simply be a function that returns a constant equal to the proportion of patients assigned to the treatment group, i.e.: propensity.func = function(x, trt) 0.5.

type  Type of plot to create. Options are either a histogram (type = "histogram") for each treatment group, a density (type = "density") for each treatment group, or to plot both a density and histogram (type = "code")

bins  integer number of bins for histograms when type = "histogram"
Examples

```r
library(personalized)

set.seed(123)
n.obs <- 250
n-vars <- 15
x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)

# simulate non-randomized treatment
xbetat <- 0.25 + 0.5 * x[,11] - 0.5 * x[,12]
trt.prob <- exp(xbetat) / (1 + exp(xbetat))
trt01 <- rbinom(n.obs, 1, prob = trt.prob)

# create function for fitting propensity score model
propensity.func <- function(x, trt)
{
  # fit propensity score model
  propens.model <- cv.glmnet(y = trt,
    x = x, family = "binomial")
  pi.x <- predict(propens.model, s = "lambda.min",
    newx = x, type = "response")[,1]

  pi.x
}

check.overlap(x = x,
  trt = trt01,
  propensity.func = propensity.func)

# now add density plot with histogram
check.overlap(x = x,
  trt = trt01,
  type = "both",
  propensity.func = propensity.func)

# simulated non-randomized treatment with multiple levels
xbetat_1 <- 0.15 + 0.5 * x[,9] - 0.25 * x[,12]
xbetat_2 <- 0.15 - 0.5 * x[,11] + 0.25 * x[,15]
trt.1.prob <- exp(xbetat_1) / (1 + exp(xbetat_1) + exp(xbetat_2))
trt.2.prob <- exp(xbetat_2) / (1 + exp(xbetat_1) + exp(xbetat_2))
trt.3.prob <- 1 - (trt.1.prob + trt.2.prob)
prob.mat <- cbind(trt.1.prob, trt.2.prob, trt.3.prob)
trt <- apply(prob.mat, 1, function(rr) rmultinom(1, 1, prob = rr))
trt <- apply(trt, 2, function(rr) which(rr == 1))

# use multinomial logistic regression model with lasso penalty for propensity
propensity.multinom.lasso <- function(x, trt)
{
  if (!is.factor(trt)) trt <- as.factor(trt)
  gfit <- cv.glmnet(y = trt, x = x, family = "multinomial")
```

fit.subgroup

Fitting subgroup identification models

Description


Usage


Arguments

x The design matrix (not including intercept term)
y The response vector
trt treatment vector with each element equal to a 0 or a 1, with 1 indicating treatment status is active.
function that inputs the design matrix $x$ and the treatment vector $trt$ and outputs the propensity score, i.e. $Pr(trt = 1 | X = x)$. Function should take two arguments 1) $x$ and 2) $trt$. See example below. For a randomized controlled trial this can simply be a function that returns a constant equal to the proportion of patients assigned to the treatment group, i.e.: $propensity.func = function(x, trt) \ 0.5$.

choice of both the M function from Chen, et al (2017) and potentially the penalty used for variable selection. All loss options starting with sq_loss use $M(y, v) = (v - y)^2$, all options starting with logistic_loss use the logistic loss: $M(y, v) = y \log(1 + \exp{-v})$, and all options starting with cox_loss use the negative partial likelihood loss for the Cox PH model. All options ending with lasso have a lasso penalty added to the loss for variable selection. sq_loss_lasso_gam and logistic_loss_lasso_gam first use the lasso to select variables and then fit a generalized additive model with nonparametric additive terms for each selected variable. sq_loss_gam involves a squared error loss with a generalized additive model and no variable selection. sq_loss_gbm involves a squared error loss with a gradient-boosted decision trees model for the benefit score; this allows for flexible estimation using machine learning and can be useful when the underlying treatment-covariate interaction is complex.

- Continuous Outcomes
  - "sq_loss_lasso" - $M(y, v) = (v - y)^2$ with linear model and lasso penalty
  - "owl_logistic_loss_lasso" - $M(y, v) = y \log(1 + \exp{-v})$ (method of Regularized Outcome Weighted Subgroup Identification)
  - "owl_logistic_flip_loss_lasso" - $M(y, v) = y \log(1 + \exp{-\text{sign}(y)v})$
  - "owl_hinge_loss" - $M(y, v) = y \max(0, 1 - v)$ (method of Estimating individualized treatment rules using outcome weighted learning)
  - "owl_hinge_flip_loss" - $M(y, v) = y \max(0, 1 - \text{sign}(y)v)$
  - "sq_loss_lasso_gam" - $M(y, v) = (v - y)^2$ with variables selected by lasso penalty and generalized additive model fit on the selected variables
  - "sq_loss_gam" - $M(y, v) = (v - y)^2$ with generalized additive model fit on all variables
  - "owl_logistic_loss_gam" - $M(y, v) = y \log(1 + \exp{-v})$ with generalized additive model fit on all variables
  - "owl_logistic_flip_loss_gam" - $M(y, v) = y \log(1 + \exp{-\text{sign}(y)v})$ with generalized additive model fit on all variables
  - "owl_logistic_loss_lasso_gam" - $M(y, v) = y \log(1 + \exp{-v})$ with variables selected by lasso penalty and generalized additive model fit on the selected variables
  - "owl_logistic_flip_loss_lasso_gam" - $M(y, v) = y \log(1 + \exp{-\text{sign}(y)v})$ with variables selected by lasso penalty and generalized additive model fit on the selected variables
  - "sq_loss_gbm" - $M(y, v) = (v - y)^2$ with gradient-boosted decision trees model
  - "abs_loss_gbm" - $M(y, v) = |v - y|$ with gradient-boosted decision trees model
• Binary Outcomes
  – All losses for continuous outcomes can be used plus the following:
  – "logistic_loss_lasso" - M(y, v) = -[yv - log(1 + exp{-v})] with
    linear model and lasso penalty
  – "logistic_loss_lasso_gam" - M(y, v) = y * log(1 + exp{-v}) with
    variables selected by lasso penalty and generalized additive model
    fit on the selected variables
  – "logistic_loss_gam" - M(y, v) = y * log(1 + exp{-v}) with general-
    ized additive model fit on all variables
  – "logistic_loss_gbm" - M(y, v) = -[yv - log(1 + exp{-v})] with gradient-
    boosted decision trees model

• Time-to-Event Outcomes
  – "cox_loss_lasso" - M corresponds to the negative partial likelihood
    of the cox model with linear model and additionally a lasso penalty
  – "cox_loss_gbm" - M corresponds to the negative partial likelihood of
    the cox model with gradient-boosted decision trees model

method
  subgroup ID model type. Either the weighting or A-learning method of Chen et
  al, (2017)

match.id
  a (character, factor, or integer) vector with length equal to the number of obser-
  vations in x indicating using integers or levels of a factor vector which patients
  are in which matched groups. Defaults to NULL and assumes the samples are
  not from a matched cohort. Matched case-control groups can be created using
  any method (propensity score matching, optimal matching, etc). If each case
  is matched with a control or multiple controls, this would indicate which case-
  control pairs or groups go together. If match.id is supplied, then it is unnecessary
  to specify a function via the propensity.func argument. A quick usage exam-
  ple: if the first patient is a case and the second and third are controls matched
  to it, and the fourth patient is a case and the fifth through seventh patients are
  matched with it, then the user should specify match.id = c(1,1,1,2,2,2,2)
  or match.id = c(rep("Grp1", 3),rep("Grp2", 4))

augment.func
  function which inputs the response y, the covariates x, and trt and outputs pre-
  dicted values (on the link scale) for the response using a model constructed with
  x. augment.func() can also be simply a function of x and y. This function
  is used for efficiency augmentation. When the form of the augmentation func-
  tion is correct, it can provide efficient estimation of the subgroups Example 1:
  augment.func <- function(x, y) {lmmod <- lm(y ~ x); return(fitted(lmmod))}
  Example2: augment.func <- function(x, y, trt) {lmmod <- lm(y ~ x + trt); return(fitted(lmmod))}
  For binary and time-to-event outcomes, make sure that predictions are returned
  on the scale of the predictors
  Example 3:
  augment.func <- function(x, y) { bmod <- glm(y ~ x, family = binomial());
      return(predict(bmod, type = "link"))}

cutpoint
  numeric value for patients with benefit scores above which (or below which if
  larger.outcome.better = FALSE) will be recommended to be in the treat-
  ment group
**fit.subgroup**

- **larger.outcome.better**
  - boolean value of whether a larger outcome is better/preferable. Set to `TRUE` if a larger outcome is better/preferable and set to `FALSE` if a smaller outcome is better/preferable. Defaults to `TRUE`.

- **reference.trt**
  - which treatment should be treated as the reference treatment. Defaults to the first level of `trt` if `trt` is a factor or the first alphabetical or numerically first treatment level. Not used for multiple treatment fitting with OWL-type losses.

- **recall**
  - boolean value. if `TRUE` then the passed arguments will be saved. Do not set to `FALSE` if the `validate.subgroup()` function will later be used for your fitted subgroup model. Only set to `FALSE` if memory is limited as setting to `TRUE` saves the design matrix to the fitted object.

- **...**
  - options to be passed to underlying fitting function. For all loss options with 'lasso', this will be passed to `cv.glmnet`. For all loss options with 'gam', this will be passed to `gam` from the `mgcv` package. Note that for all loss options that use gam() from the mgcv package, the user cannot supply the gam argument method because it is also an argument of `fit.subgroup`, so instead, to change the gam method argument, supply `methodNgam`, ie `methodNgam = "REML".`
  - For all loss options with 'hinge', this will be passed to both `weighted.ksvm` and `ipop` from the `kernlab` package.

**References**


**See Also**

- `validate.subgroup` for function which creates validation results for subgroup identification models.
- `predict.subgroup_fitted` for a prediction function for fitted models from `fit.subgroup`.
- `plot.subgroup_fitted` for a function which plots results from fitted models, and `print.subgroup_fitted` for arguments for printing options for `fit.subgroup()` from `fit.subgroup`.

**Examples**

```r
library(personalized)

set.seed(123)
n.obs <- 500
n.vars <- 15
x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)
```
# simulate non-randomized treatment
xbetat <- 0.5 + 0.5 * x[,7] - 0.5 * x[,9]
trt.prob <- exp(xbetat) / (1 + exp(xbetat))
trt01 <- rbinom(n.obs, 1, prob = trt.prob)
trt <- 2 * trt01 - 1

# simulate response
delta <- 2 * (0.5 + x[,2] - x[,3] - x[,11] + x[,1] * x[,12])
xbeta <- x[,1] + x[,11] - 2 * x[,12]^2 + x[,13] + 0.5 * x[,15] ^ 2
xbeta <- xbeta + delta * trt

# continuous outcomes
y <- drop(xbeta) + rnorm(n.obs, sd = 2)

# binary outcomes
y.binary <- 1 * (xbeta + rnorm(n.obs, sd = 2) > 0)

# time-to-event outcomes
surv.time <- exp(-20 - xbeta + rnorm(n.obs, sd = 1))
cens.time <- exp(rnorm(n.obs, sd = 3))
y.time.to.event <- pmin(surv.time, cens.time)
status <- 1 * (surv.time <= cens.time)

# create function for fitting propensity score model
prop.func <- function(x, trt)
{
  # fit propensity score model
  propens.model <- cv.glmnet(y = trt,
                           x = x, family = "binomial")
  pi.x <- predict(propens.model, s = "lambda.min",
                  newx = x, type = "response")[,1]
  pi.x
}

subgrp.model <- fit.subgroup(x = x, y = y,
                           trt = trt01,
                           propensity.func = prop.func,
                           loss = "sq_loss_lasso",
                           nfolds = 10)  # option for cv.glmnet

summary(subgrp.model)

# fit lasso + gam model with REML option for gam
subgrp.modelg <- fit.subgroup(x = x, y = y,
                           trt = trt01,
                           propensity.func = prop.func,
                           loss = "sq_loss_lasso_gam",
                           method.gam = "REML",  # option for gam
                           nfolds = 5)  # option for cv.glmnet
plot.subgroup_fitted

subgrp.model <- fit.subgroup(x = x, y = y.binary,
   trt = trt01,
   propensity.func = prop.func,
   loss = "logistic_loss_lasso",
   type.measure = "auc",       # option for cv.glmnet
   nfolds = 5)                   # option for cv.glmnet

subgrp.model.bin

library(survival)
subgrp.model.cox <- fit.subgroup(x = x, y = Surv(y.time.to.event, status),
   trt = trt01,
   propensity.func = prop.func,
   loss = "cox_loss_lasso",
   nfolds = 5)                   # option for cv.glmnet

subgrp.model.cox

---

plot.subgroup_fitted  
*Plotting results for fitted subgroup identification models*

**Description**

Plots results for estimated subgroup treatment effects

Plots validation results for estimated subgroup treatment effects

**Usage**

```r
## S3 method for class 'subgroup_fitted'
plot(x, type = c("boxplot", "density", "interaction"), avg.line = TRUE, ...)

## S3 method for class 'subgroup_validated'
plot(x, type = c("boxplot", "density", "interaction", "stability"), avg.line = TRUE, ...)
```

**Arguments**

- `x`  
  fitted object returned by `validate.subgroup()` or `fit.subgroup()` function

- `type`  
  type of plot. "density" results in a density plot for the results across all observations (if `x` is from `fit.subgroup()`) or if `x` is from `validate.subgroup()` across iterations of either the bootstrap or training/test re-fitting. For the latter case the test results will be plotted. "boxplot" results in boxplots across all observations/iterations of either the bootstrap or training/test re-fitting. For the
latter case the test results will be plotted. "interaction" creates an interaction plot for the different subgroups (crossing lines here means a meaningful subgroup).

avg.line boolean value of whether or not to plot a line for the average value in addition to the density (only valid for type = "density")

See Also

- `fit.subgroup` for function which fits subgroup identification models.
- `validate.subgroup` for function which creates validation results and `fit.subgroup` for function which fits subgroup identification models.

Examples

```r
library(personalized)

set.seed(123)
n.obs <- 500
n.vars <- 15
x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)

# simulate non-randomized treatment
xbetat <- 0.5 + 0.5 * x[,11] - 0.5 * x[,13]
trt.prob <- exp(xbetat) / (1 + exp(xbetat))
trt01 <- rbinom(n.obs, 1, prob = trt.prob)

trt <- 2 * trt01 - 1

delta <- 2 * (0.5 + x[,2] - x[,3] - x[,11] + x[,1] * x[,12])
xbeta <- x[,1] + x[,11] - 2 * x[,12]*2 + x[,13]
xbeta <- xbeta + delta * trt

delta <- 2 * (0.5 + x[,2] - x[,3] - x[,11] + x[,1] * x[,12])
xbeta <- x[,1] + x[,11] - 2 * x[,12]*2 + x[,13]
xbeta <- xbeta + delta * trt

# continuous outcomes
y <- drop(xbeta) + rnorm(n.obs, sd = 2)

# create function for fitting propensity score model
prop.func <- function(x, trt)
{
    # fit propensity score model
    propens.model <- cv.glmnet(y = trt,
                               x = x, family = "binomial")
    pi.x <- predict(propens.model, s = "lambda.min",
                     newx = x, type = "response")[,1]
    pi.x
}

subgrp.model <- fit.subgroup(x = x, y = y,
                              trt = trt01,
                              ... not used
}`
propensity.func = prop.func,
loss = "sq_loss_lasso",
nfolds = 5) # option for cv.glmnet

subgrp.model$subgroup.trt.effects

plot(subgrp.model)
plot(subgrp.model, type = "boxplot")
plot(subgrp.model, type = "interaction")

valmod <- validate.subgroup(subgrp.model, B = 3,
method = "training_test",
train.fraction = 0.75)

valmod$avg.results
plot(valmod)
plot(valmod, type = "interaction")

# visualize the frequency of particular variables
# of being selected across the resampling iterations with
# 'type = "stability"
# not run:
# plot(valmod, type = "stability")

---

**plotCompare**

*Plot a comparison results for fitted or validated subgroup identification models*

**Description**

Plots comparison of results for estimated subgroup treatment effects

**Usage**

```r
plotCompare(..., type = c("boxplot", "density", "interaction"),
avg.line = TRUE)
```

**Arguments**

- **...** the fitted (model or validation) objects to be plotted. Must be either objects returned from `fit.subgroup()` or `validate.subgroup()`
- **type** type of plot. "density" results in a density plot for the results across all observations (if `x` is from `fit.subgroup()`) or if `x` is from `validate.subgroup()` across iterations of either the bootstrap or training/test re-fitting. For the latter
case the test results will be plotted. "boxplot" results in boxplots across all observations/iterations of either the bootstrap or training/test re-fitting. For the latter case the test results will be plotted. "interaction" creates an interaction plot for the different subgroups (crossing lines here means a meaningful subgroup)

`avg.line` boolean value of whether or not to plot a line for the average value in addition to the density (only valid for type = "density")

**See Also**

`fit.subgroup` for function which fits subgroup identification models and `validate.subgroup` for function which creates validation results.

**Examples**

```r
library(personalized)

set.seed(123)
n.obs <- 1000
n.vars <- 50
x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)

# simulate non-randomized treatment
xbetat <- 0.5 + 0.5 * x[,21] - 0.5 * x[,41]
trt.prob <- exp(xbetat) / (1 + exp(xbetat))
trt01 <- rbinom(n.obs, 1, prob = trt.prob)
trt <- 2 * trt01 - 1

# simulate response
delta <- 2 * (0.5 + x[,2] - x[,3] - x[,11] + x[,1] * x[,12])
xbeta <- xbeta + delta * trt

# continuous outcomes
y <- drop(xbeta) + rnorm(n.obs, sd = 2)

# create function for fitting propensity score model
prop.func <- function(x, trt)
{
    # fit propensity score model
    propens.model <- cv.glmnet(y = trt,
                                x = x, family = "binomial")
    pi.x <- predict(propens.model, s = "lambda.min",
                     newx = x, type = "response")[,1]
    pi.x
}

subgrp.model <- fit.subgroup(x = x, y = y,
                              trt = trt01,
                              propensity.func = prop.func,
                              ...)
```
predict.subgroup_fitted

## S3 method for class 'subgroup_fitted'
predict(object, newx, type = c("benefit.score", "trt.group"), cutpoint = 0, ...)

## S3 method for class 'wksvm'
predict(object, newx, type = c("class", "linear.predictor"), ...)

### Arguments

- **object**: fitted object returned by validate.subgrp() function. For predict.wksvm(), this should be a fitted wksvm object from the weighted.ksvm() function.
- **newx**: new design matrix for which predictions will be made.
- **type**: type of prediction. type = "benefit.score" results in predicted benefit scores and type = "trt.group" results in prediction of recommended treatment group. For predict.wksvm(), type = 'class' yields predicted class and type = 'linear.predictor' yields estimated function (the sign of which is the estimated class).
- **cutpoint**: numeric value for patients with benefit scores above which (or below which if larger.outcome.better = FALSE) will be recommended to be in the treatment group.
- **...**: not used.

### Description

Predicts benefit score based on a fitted subgroup identification model. Function to obtain predictions for weighted ksvm objects.

### Usage

```r
subgrp.modelg <- fit.subgroup(x = x, y = y,
trt = trt01,
propensity.func = prop.func,
loss = "sq_loss_lasso_lam")

plotCompare(subgrp.model, subgrp.modelg)
```
See Also

- `fit.subgroup` for function which fits subgroup identification models.
- `weighted.ksvm` for fitting weighted.ksvm objects

Examples

```r
library(personalized)
set.seed(123)
n.obs <- 1000
n.vars <- 50
x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)

# simulate non-randomized treatment
xbetat <- 0.5 + 0.5 * x[,21] - 0.5 * x[,41]
trt.prob <- exp(xbetat) / (1 + exp(xbetat))
trt01 <- rbinom(n.obs, 1, prob = trt.prob)

trt <- 2 * trt01 - 1

# simulate response
delta <- 2 * (0.5 + x[,2] - x[,3] - x[,11] + x[,1] * x[,12])
xbeta <- x[,1] + x[,11] - 2 * x[,12]^2 + x[,13]
xbeta <- xbeta + delta * trt

# continuous outcomes
y <- drop(xbeta) + rnorm(n.obs, sd = 2)

# create function for fitting propensity score model
prop.func <- function(x, trt)
{
  # fit propensity score model
  propens.model <- cv.glmnet(y = trt,
                            x = x, family = "binomial")
  pi.x <- predict(propens.model, s = "lambda.min",
                   newx = x, type = "response")[,1]
  pi.x
}

subgrp.model <- fit.subgroup(x = x, y = y,
                              trt = trt01,
                              propensity.func = prop.func,
                              loss = "sq_loss_lasso",
                              nfolds = 5)  # option for cv.glmnet

subgrp.model$subgroup.trt.effects
benefit.scores <- predict(subgrp.model, newx = x, type = "benefit.score")

rec.trt.grp <- predict(subgrp.model, newx = x, type = "trt.group")
```
Description

Prints results for estimated subgroup treatment effects
Prints summary results for estimated subgroup treatment effects

Usage

```r
## S3 method for class 'subgroup_fitted'
print(x, digits = max(getOption("digits") - 3, 3),
     ...)

## S3 method for class 'subgroup_validated'
print(x, digits = max(getOption("digits") - 3, 3), sample.pct = FALSE, ...)

## S3 method for class 'subgroup_summary'
print(x, p.value = 1, 
     digits = max(getOption("digits") - 3, 3), ...)
```

Arguments

- `x` a fitted object from either `fit.subgroup`, `validate.subgroup`, or `summarize.subgroups`
- `digits` minimal number of significant digits to print.
- `...` further arguments passed to or from `print.default`.
- `sample.pct` boolean variable of whether to print the percent of the test sample within each subgroup. If false the sample size itself, not the percent is printed. This may not be informative if the test sample size is much different from the total sample size.
- `p.value` a p-value threshold for mean differences below which covariates will be displayed. For example, setting `p.value = 0.05` will display all covariates that have a significant difference between subgroups with p-value less than 0.05. Defaults to 1, which displays all covariates.

See Also

- `validate.subgroup` for function which creates validation results and `fit.subgroup` for function which fits subgroup identification models.
- `summarize.subgroups` for function which summarizes subgroup covariate values.
subgroup.effects | Computes treatment effects within various subgroups

Description

Computes treatment effects within various subgroups to estimate subgroup treatment effects

Usage

subgroup.effects(benefit.scores, y, trt, cutpoint = 0,
                   larger.outcome.better = TRUE, reference.trt = NULL)

Arguments

- **benefit.scores**: vector of estimated benefit scores
- **y**: The response vector
- **trt**: treatment vector with each element equal to a 0 or a 1, with 1 indicating treatment status is active.
- **cutpoint**: numeric value for patients with benefit scores above which (or below which if larger.outcome.better = FALSE) will be recommended to be in the treatment group
- **larger.outcome.better**: boolean value of whether a larger outcome is better. Set to TRUE if a larger outcome is better and set to FALSE if a smaller outcome is better. Defaults to TRUE.
- **reference.trt**: index of which treatment is the reference (in the case of multiple treatments). This should be known already, as for a trt with K-levels, there will be K-1 benefit scores (1 per column) of benefit.scores, where each column is a comparison of each K-1 treatments with the reference treatment. The default is the last level of trt if it is a factor.

See Also

- fit.subgroup for function which fits subgroup identification models which generate benefit scores.

summarize.subgroups | Summarizing covariates within estimated subgroups

Description

Summarizes covariate values within the estimated subgroups
Usage

summarize.subgroups(x, ...)

## Default S3 method:
summarize.subgroups(x, subgroup, ...)

## S3 method for class 'subgroup_fitted'
summarize.subgroups(x, ...)

Arguments

x
a fitted object from fit.subgroup() or a matrix of covariate values

... optional arguments to summarize.subgroups methods

subgroup vector of indicators of same length as the number of rows in x if x is a matrix. A
value of 1 in the ith position of subgroup indicates patient i is in the subgroup
of patients recommended the treatment and a value of 0 in the ith position of
subgroup indicates patient i is in the subgroup of patients recommended the
control. If x is a fitted object returned by fit.subgroup(), subgroup is not
needed.

See Also

fit.subgroup for function which fits subgroup identification models and print.subgroup_summary
for arguments for printing options for summarize.subgroups().

Examples

library(personalized)

set.seed(123)
n.obs <- 1000
n.vars <- 50
x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)

# simulate non-randomized treatment
xbetat <- 0.5 + 0.5 * x[,2] - 0.5 * x[,4]
trt.prob <- exp(xbetat) / (1 + exp(xbetat))
trt01 <- rbinom(n.obs, 1, prob = trt.prob)

trt <- 2 * trt01 - 1

# simulate response
delta <- 2 * (0.5 + x[,2] - x[,3] - x[,11] + x[,1] * x[,12])
xbeta <- xbeta + delta * trt

# continuous outcomes
y <- drop(xbeta) + rnorm(n.obs, sd = 2)
# create function for fitting propensity score model
prop.func <- function(x, trt)
{
  # fit propensity score model
  propens.model <- cv.glmnet(y = trt, 
    x = x, family = "binomial")
  pi.x <- predict(propens.model, s = "lambda.min", 
    newx = x, type = "response")[,1]
  pi.x
}

subgrp.model <- fit.subgroup(x = x, y = y, 
  trt = trt01, 
  propensity.func = prop.func, 
  loss = "sq_loss_lasso", 
  nfolds = 5)  # option for cv.glmnet

comp <- summarize.subgroups(subgrp.model)
print(comp, p.value = 0.01)

# or we can simply supply the matrix x and the subgroups
comp2 <- summarize.subgroups(x, subgroup = 1 * (subgrp.model$benefit.scores > 0))

print(comp2, p.value = 0.01)

summary.subgroup_fitted

*Summary of results for fitted subgroup identification models*

**Description**

Prints summary of results for estimated subgroup treatment effects
Prints summary of results for estimated weighted ksvm

**Usage**

```r
## S3 method for class 'subgroup_fitted'
summary(object, digits = max(getOption("digits") - 3, 3), ...)

## S3 method for class 'wksvm'
summary(object, digits = max(getOption("digits") - 3, 3), ...)
```

**Arguments**

- **object** a fitted object from either fit.subgroup or validate.subgroup
- **digits** minimal number of significant digits to print.
- **...** further arguments passed to or from `print.default`. 
validate.subgroup

See Also

validate.subgroup for function which creates validation results and fit.subgroup for function
which fits subgroup identification models.

validate.subgroup  Validating fitted subgroup identification models

Description

Validates subgroup treatment effects for fitted subgroup identification model class of Chen, et al
(2017)

Usage

validate.subgroup(model, B = 50L, method = c("training_test_replication", 
"boot_bias_correction"), train.fraction = 0.5, parallel = FALSE)

Arguments

model  fitted model object returned by fit.subgroup() function
B  integer. number of bootstrap replications or refitting replications.
method  validation method. "boot_bias_correction" for the bootstrap bias correction
training and test splitting of the data (train.fraction should be specified for
this option)
train.fraction  fraction (between 0 and 1) of samples to be used for training in training/test
replication. Only used for method = "training_test_replication"
parallel  Should the loop over replications be parallelized? If FALSE, then no, if TRUE,
then yes. If user sets parallel = TRUE and the fitted fit.subgroup() ob-
ject uses the parallel version of an internal model, say for cv.glmnet(), then
the internal parallelization will be overridden so as not to create a conflict of
parallelism.

References

Chen, S., Tian, L., Cai, T. and Yu, M. (2017), A general statistical framework for subgroup iden-
models: issues in developing models, evaluating assumptions and adequacy, and measuring and re-
ducing errors. Statistics in medicine, 15, 361-387. doi:10.1002/(SICI)1097-0258(19960229)15:4<361::AID-
SIM168>3.0.CO;2-4

See Also

fit.subgroup for function which fits subgroup identification models, plot.subgroup_validated
for plotting of validation results, and print.subgroup_validated for arguments for printing op-
tions for validate.subgroup().
Examples

library(personalized)

set.seed(123)
n.obs <- 500
n.vars <- 20
x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)

# simulate non-randomized treatment
xbetat <- 0.5 + 0.5 * x[,11] - 0.5 * x[,13]
trt.prob <- exp(xbetat) / (1 + exp(xbetat))
trt01 <- rbinom(n.obs, 1, prob = trt.prob)
trt <- 2 * trt01 - 1

# simulate response
delta <- 2 * (0.5 + x[,2] - x[,3] - x[,11] + x[,1] * x[,12])
xbeta <- x[,1] + x[,11] - 2 * x[,12]^2 + x[,13]
xbeta <- xbeta + delta * trt

# continuous outcomes
y <- drop(xbeta) + rnorm(n.obs, sd = 2)

# create function for fitting propensity score model
prop.func <- function(x, trt)
{
    # fit propensity score model
    propens.model <- cv.glmnet(y = trt,
                              x = x, family = "binomial")

    pi.x <- predict(propens.model, s = "lambda.min",
                    newx = x, type = "response")[,1]

    pi.x
}

subgrp.model <- fit.subgroup(x = x, y = y,
                              trt = trt01,
                              propensity.func = prop.func,
                              loss = "sq_loss_lasso",
                              nfolds = 5)  # option for cv.glmnet

subgrp.model$ subgroup.trt.effects

x.test <- matrix(rnorm(10 * n.obs * n.vars, sd = 3), 10 * n.obs, n.vars)

# simulate non-randomized treatment
xbetat.test <- 0.5 + 0.5 * x.test[,11] - 0.5 * x.test[,13]
trt.prob.test <- exp(xbetat.test) / (1 + exp(xbetat.test))
trt01.test <- rbinom(10 * n.obs, 1, prob = trt.prob.test)
trt.test <- 2 * trt01.test - 1
# simulate response
delta.test <- 2 * (0.5 + x.test[,2] - x.test[,3] - x.test[,11] + x.test[,1] * x.test[,12])
xbeta.test <- xbeta.test + delta.test * trt.test

ty.test <- drop(xbeta.test) + rnorm(10 * n.obs, sd = 2)

valmod <- validate.subgroup(subgrp.model, B = 10,
method = "training_test",
train.fraction = 0.75)

valmod

bene.score.test <- subgrp.model$predict(x.test)

mean(y.test[bene.score.test > 0 & trt01.test == 1]) -
mean(y.test[bene.score.test > 0 & trt01.test == 0])
mean(y.test[bene.score.test <= 0 & trt01.test == 0]) -
mean(y.test[bene.score.test <= 0 & trt01.test == 1])

quantile(valmod$boot.results[[1]][,1], c(0.025, 0.975))
quantile(valmod$boot.results[[1]][,2], c(0.025, 0.975))

---

**weighted.ksvm**

*Fit weighted kernel SVM model.*

**Description**

Fits weighted kernel SVM. To be used for OWL with hinge loss (but can be used more generally)

**Usage**

`weighted.ksvm(y, x, weights, C = c(0.1, 0.5, 1, 5, 10), kernel = "rbfdot",
 kpar = "automatic", nfolds = 10, foldid = NULL, ...)`

**Arguments**

- `y` The response vector (either a character vector, factor vector, or numeric vector with values in -1, 1)
- `x` The design matrix (not including intercept term)
- `weights` vector of sample weights for weighted SVM
- `C` cost of constraints violation, see `ksvm`
- `kernel` kernel function used for training and prediction. See `ksvm` and `kernels`
- `kpar` list of hyperparameters for the kernel function. See `ksvm`
- `nfolds` number of cross validation folds for selecting value of C
- `foldid` optional vector of values between 1 and nfolds specifying which fold each observation is in. If specified, it will override the nfolds argument.
- `...` extra arguments to be passed to `ipop` from the kernlab package
See Also

`predict.wksvm` for predicting from fitted `weighted.ksvm` objects

Examples

```r
library(kernlab)

x <- matrix(rnorm(200 * 2), ncol = 2)

y <- 2 * (sin(x[,2]) ^ 2 * exp(-x[,2]) - 0.2 > rnorm(200, sd = 0.1)) - 1

weights <- runif(100, max = 1.5, min = 0.5)

wk <- weighted.ksvm(x = x[1:100,], y = y[1:100], C = c(0.1, 0.5, 1, 2, 10),
weights = weights[1:100])

pr <- predict(wk, newx = x[101:200],)

mean(pr == y[101:200])
```
Index

check.overlap, 2
cv.glmnet, 7

fit.subgroup, 4, 10, 12, 14–17, 19

gam, 7

ipop, 7, 21

kernels, 21
ksvm, 21

plot.subgroup_fitted, 7, 9
plot.subgroup_validated, 19
plot.subgroup_validated
  (plot.subgroup_fitted), 9
plotCompare, 11
predict.subgroup_fitted, 7, 13
predict.wksvm, 22
predict.wksvm
  (predict.subgroup_fitted), 13
print.default, 15, 18
print.subgroup_fitted, 7, 15
print.subgroup_summary, 17
print.subgroup_summary
  (print.subgroup_fitted), 15
print.subgroup_validated, 19
print.subgroup_validated
  (print.subgroup_fitted), 15

subgroup.effects, 16
summarize.subgroups, 15, 16
summary.subgroup_fitted, 18
summary.wksvm
  (summary.subgroup_fitted), 18

validate.subgroup, 7, 10, 12, 15, 19, 19

weighted.ksvm, 7, 14, 21