Package ‘personalized2part’

September 10, 2020

Type Package
Title Two-Part Estimation of Treatment Rules for Semi-Continuous Data
Version 0.0.1
Description Implements the methodology of Huling, Smith, and Chen (2020) <doi:10.1080/01621459.2020.1801449>, which allows for subgroup identification for semi-continuous outcomes by estimating individualized treatment rules. It uses a two-part modeling framework to handle semi-continuous data by separately modeling the positive part of the outcome and an indicator of whether each outcome is positive, but still results in a single treatment rule. High dimensional data is handled with a cooperative lasso penalty, which encourages the coefficients in the two models to have the same sign.

URL https://github.com/jaredhuling/personalized2part
BugReports https://github.com/jaredhuling/personalized2part/issues
License GPL (>= 2)
Encoding UTF-8
LazyData true
Depends personalized, HDtweedie
LinkingTo Rcpp, RcppEigen
Imports Rcpp, foreach, methods
RoxygenNote 7.1.1
NeedsCompilation yes
Author Jared Huling [aut, cre] (<https://orcid.org/0000-0003-0670-4845>)
Maintainer Jared Huling <jaredhuling@gmail.com>
Repository CRAN
Date/Publication 2020-09-10 10:00:03 UTC

R topics documented:
cv.hd2part .................................................. 2
fit_subgroup_2part ......................................... 3
cv.hd2part

Cross validation for hd2part models

Description

Cross validation for hd2part models

Usage

cv.hd2part(
  x,
  z,
  x_s,
  s,
  weights = rep(1, NROW(x)),
  weights_s = rep(1, NROW(x_s)),
  offset = NULL,
  offset_s = NULL,
  lambda = NULL,
  type.measure = c("mae", "mse", "sep-auc-mse", "sep-auc-mae"),
  nfolds = 10,
  foldid = NULL,
  grouped = TRUE,
  keep = FALSE,
  parallel = FALSE,
  ...
)

Arguments

x an n x p matrix of covariates for the zero part data, where each row is an observation and each column is a predictor. MUST be ordered such that the first n_s rows align with the observations in x_s and s

z a length n vector of responses taking values 1 and 0, where 1 indicates the response is positive and zero indicates the response has value 0. MUST be ordered such that the first n_s values align with the observations in x_s and s

x_s an n_s x p matrix of covariates (which is a submatrix of x) for the positive part data, where each row is an observation and each column is a predictor
fit_subgroup_2part

$s$ a length $n_s$ vector of responses taking strictly positive values

`weights` a length $n$ vector of observation weights for the zero part data

`weights_s` a length $n_s$ vector of observation weights for the positive part data

`offset` a length $n$ vector of offset terms for the zero part data

`offset_s` a length $n_s$ vector of offset terms for the positive part data

`lambda` A user supplied lambda sequence. By default, the program computes its own lambda sequence based on `nlambda` and `lambda.min.ratio`. Supplying a value of `lambda` overrides this.

`type.measure` measure to evaluate for cross-validation. Will add more description later

`nfolds` number of folds for cross-validation. default is 10. 3 is smallest value allowed.

`foldid` an optional vector of values between 1 and `nfolds` specifying which fold each observation belongs to.

`grouped` Like in `glmnet`, this is an experimental argument, with default TRUE, and can be ignored by most users. For all models, this refers to computing `nfolds` separate statistics, and then using their mean and estimated standard error to describe the CV curve. If `grouped = FALSE`, an error matrix is built up at the observation level from the predictions from the `nfolds` fits, and then summarized (does not apply to `type.measure = "auc"`).

`keep` If `keep = TRUE`, a prevalidated list of array is returned containing fitted values for each observation and each value of `lambda` for each model. This means these fits are computed with this observation and the rest of its fold omitted. The `foldid` vector is also returned. Default is `keep = FALSE`

`parallel` If TRUE, use parallel foreach to fit each fold. Must register parallel beforehand, such as `doMC`.

... other parameters to be passed to `hd2part` function

Examples

```r
set.seed(1)
```

---

**Description**

Fits subgroup identification models for semicontinuous positive outcomes
Usage

```r
fit_subgroup_2part(
  x,
  y,
  trt,
  propensity.func = NULL,
  propensity.func.positive = NULL,
  match.id = NULL,
  augment.func.zero = NULL,
  augment.func.positive = NULL,
  cutpoint = 1,
  larger.outcome.better = TRUE,
  penalize.ate = TRUE,
  y_eps = 1e-06,
  ...
)
```

Arguments

- **x**: The design matrix (not including intercept term)
- **y**: The nonnegative response vector
- **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., $\Pr(trt = 1 \mid X = x)$. 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`.
- **propensity.func.positive**: function that inputs the design matrix x and the treatment vector trt and outputs the propensity score for units with positive outcome values, i.e., $\Pr(trt = 1 \mid X = x, Z = 1)$. 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`.
- **match.id**: a (character, factor, or integer) vector with length equal to the number of observations 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 example: 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))` as the covariates `x`, and `trt` and outputs predicted values (on the probability scale) for the response using a model constructed with `x`. `augment.func.zero()` can also be simply a function of `x` and `y`. This function is used for efficiency augmentation. When the form of the augmentation function is correct, it can provide efficient estimation of the subgroups. Some examples of possible augmentation functions are:

Example 1: `augment.func <- function(x, y) {lmod <- glm(y ~ x, family = binomial()); return(fitted(lmod))}`

Example 2:
```r
augment.func <- function(x, y, trt) {
  data <- data.frame(x, y, trt)
  lmod <- glm(y ~ x * trt, family = binomial())
  ## get predictions when trt = 1
  data$trt <- 1
  preds_1 <- predict(lmod, data, type = "response")

  ## get predictions when trt = -1
  data$trt <- -1
  preds_n1 <- predict(lmod, data, type = "response")

  ## return predictions averaged over trt
  return(0.5 * (preds_1 + preds_n1))
}
augment.func.zero
```

(similar to `augment.func.positive`) function which inputs the indicators of whether each response is positive (`1*(y > 0)`), the covariates `x`, and `trt` for all samples and outputs predicted values (on the link scale) for the response using a model constructed with `x`. `augment.func.positive()` can also be simply a function of `x` and `y`. This function is used for efficiency augmentation.

`augment.func.positive`

(similar to `augment.func.zero`) function which inputs the positive part response (ie all observations in `y` which are strictly positive), the covariates `x`, and `trt` and outputs predicted values (on the link scale) for the response using a model constructed with `x`. `augment.func.positive()` can also be simply a function of `x` and `y`. This function is used for efficiency augmentation.

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. Defaults to 1, since the benefit score is a risk ratio

`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`.

`penalize.ate` should the treatment main effect (ATE) be penalized too?

`y_eps` positive value above which observations in `y` will be considered positive

... options to be passed to `cv.hd2part`
Examples

```r
set.seed(42)

dat <- sim_semicontinuous_data(250, n.vars = 15)
x <- dat$x
y <- dat$y
trt <- dat$trt

prop_func <- function(x, trt)
{
  propensmod <- glm(trt ~ x, family = binomial())
  propens <- unname(fitted(propensmod))
  propens
}

fitted_model <- fit_subgroup_2part(x, y, trt, prop_func, prop_func)

fitted_model

## correlation of estimated covariate-conditional risk ratio and truth
cor(fitted_model$benefit.scores, dat$treatment_risk_ratio, method = "spearman")
```

---

hd2part  

Main fitting function for group lasso and cooperative lasso penalized two part models

**Description**

This function fits penalized two part models with a logistic regression model for the zero part and a gamma regression model for the positive part. Each covariate’s effect has either a group lasso or cooperative lasso penalty for its effects for the two constituent models

**Usage**

```r
hd2part(
  x,
  z,
  x_s,
  s,
  weights = rep(1, NROW(x)),
  weights_s = rep(1, NROW(x_s)),
  offset = NULL,
  offset_s = NULL,
  penalty = c("grp.lasso", "coop.lasso"),
```
penalty_factor = NULL,
nlambda = 100L,
lambda_min_ratio = ifelse(n_s < p, 0.05, 0.005),
lambda = NULL,
tau = 0,
opposite_signs = FALSE,
flip_beta_zero = FALSE,
intercept_z = FALSE,
intercept_s = FALSE,
strongrule = TRUE,
maxit_irls = 50,
tol_irls = 1e-05,
maxit_mm = 500,
tol_mm = 1e-05,
balance_likelihoods = TRUE
)

Arguments

x: an n x p matrix of covariates for the zero part data, where each row is an observation and each column is a predictor.
z: a length n vector of responses taking values 1 and 0, where 1 indicates the response is positive and zero indicates the response has value 0.
x_s: an n_s x p matrix of covariates (which is a submatrix of x) for the positive part data, where each row is an observation and each column is a predictor.
s: a length n_s vector of responses taking strictly positive values.
weights: a length n vector of observation weights for the zero part data.
weights_s: a length n_s vector of observation weights for the positive part data.
offset: a length n vector of offset terms for the zero part data.
offset_s: a length n_s vector of offset terms for the positive part data.
penalty: either "grp.lasso" for the group lasso penalty or "coop.lasso" for the cooperative lasso penalty.
penalty_factor: a length p vector of penalty adjustment factors corresponding to each covariate. A value of 0 in the jth location indicates no penalization on the jth variable, and any positive value will indicate a multiplicative factor on top of the common penalization amount. The default value is 1 for all variables.
nlambda: the number of lambda values. The default is 100.
lambda_min_ratio: Smallest value for lambda, as a fraction of lambda.max, the data-derived largest lambda value. The default depends on the sample size relative to the number of variables.
lambda: a user supplied sequence of penalization tuning parameters. By default, the program automatically chooses a sequence of lambda values based on nlambda and lambda_min_ratio.
tau: value between 0 and 1 for sparse group mixing penalty. 0 implies either group lasso or coop lasso and 1 implies lasso.
opposite_signs is a boolean variable indicating whether the signs of coefficients across models should be encouraged to have opposite signs instead of the same signs. Default is FALSE. This variable has no effect for group lasso.

flip_beta_zero should we flip the signs of the parameters for the zero part model? Defaults to FALSE. Should only be used for good reason.

intercept_z whether or not to include an intercept in the zero part model. Default is TRUE.

intercept_s whether or not to include an intercept in the positive part model. Default is TRUE.

strongrule should a strong rule be used? Defaults to TRUE.

maxit_irls maximum number of IRLS iterations

tol_irls convergence tolerance for IRLS iterations

maxit_mm maximum number of MM iterations. Note that for algorithm = "irls", MM is used within each IRLS iteration, so maxit_mm applies to the convergence of the inner iterations in this case.

tol_mm convergence tolerance for MM iterations. Note that for algorithm = "irls", MM is used within each IRLS iteration, so tol_mm applies to the convergence of the inner iterations in this case.

balance_likelihoods should the likelihoods be balanced so variables would enter both models at the same value of lambda if the penalty were a lasso penalty? Recommended to keep at the default, TRUE.

Examples

library(personalized2part)

hdgamma

Fitting function for lasso penalized GLMs

Description

This function fits penalized gamma GLMs.

Usage

hdgamma(
  x,
  y,
  weights = rep(1, NROW(x)),
  offset = NULL,
  penalty_factor = NULL,
  nlambda = 100L,
  lambda_min_ratio = ifelse(n < p, 0.05, 0.005),
  lambda = NULL,
  ...)
tau = 0,
intercept = TRUE,
strongrule = TRUE,
maxit_irls = 50,
tol_irls = 1e-05,
maxit_mm = 500,
tol_mm = 1e-05
)

Arguments

x an n x p matrix of covariates for the zero part data, where each row is an observation and each column is a predictor

y a length n vector of responses taking strictly positive values.

weights a length n vector of observation weights

offset a length n vector of offset terms

penalty_factor a length p vector of penalty adjustment factors corresponding to each covariate. A value of 0 in the jth location indicates no penalization on the jth variable, and any positive value will indicate a multiplicative factor on top of the common penalization amount. The default value is 1 for all variables

nlambda the number of lambda values. The default is 100.

lambda_min_ratio Smallest value for lambda, as a fraction of lambda.max, the data-derived largest lambda value. The default depends on the sample size relative to the number of variables.

lambda a user supplied sequence of penalization tuning parameters. By default, the program automatically chooses a sequence of lambda values based on nlambda and lambda_min_ratio

tau a scalar numeric value between 0 and 1 (included) which is a mixing parameter for sparse group lasso penalty. 0 indicates group lasso and 1 indicates lasso, values in between reflect different emphasis on group and lasso penalties

intercept whether or not to include an intercept. Default is TRUE.

strongrule should a strong rule be used?

maxit_irls maximum number of IRLS iterations

tol_irls convergence tolerance for IRLS iterations

maxit_mm maximum number of MM iterations. Note that for algorithm = "irls", MM is used within each IRLS iteration, so maxit_mm applies to the convergence of the inner iterations in this case.

tol_mm convergence tolerance for MM iterations. Note that for algorithm = "irls", MM is used within each IRLS iteration, so tol_mm applies to the convergence of the inner iterations in this case.

Examples

library(personalized2part)
HDtweedie_kfold_aug  Fit a penalized gamma augmentation model via cross fitting

Description

Fits a penalized gamma augmentation model via cross fitting and returns vector of length n of out of sample predictions on the link scale from cross fitting

Usage

HDtweedie_kfold_aug(
  x,
  y,
  trt,
  wts = NULL,
  K = 10,
  p = 1.5,
  interactions = FALSE
)

Arguments

x  an n x p matrix of covariates for the zero part data, where each row is an observation and each column is a predictor. MUST be ordered such that the first n_s rows align with the observations in x_s and s

y  a length n vector of responses taking positive values

trt  a length n vector of treatment variables with 1 indicating treatment and -1 indicating control

wts  a length n vector of sample weights

K  number of folds for cross fitting

p  tweedie mixing parameter. See HDtweedie for details

interactions  boolean variable of whether or not to fit model with interactions. For predictions, interactions will be integrated out

plot.hd2part  Plot method for hd2part fitted objects

Description

Plot method for hd2part fitted objects
Usage

## S3 method for class 'hd2part'
plot(x,  
    model = c("zero", "positive"),  
    xvar = c("loglambda", "norm", "lambda"),  
    labsize = 0.6,  
    xlab = iname,  
    ylab = NULL,  
    main = paste(model, " model"),  
    ...  
)

## S3 method for class 'cv.hd2part'
plot(x, sign.lambda = 1, ...)

Arguments

x fitted "hd2part" model object
model either "zero" for the zero part model or "positive" for the positive part model
xvar What is on the X-axis. "norm" plots against the L1-norm of the coefficients, "lambda" against the log-lambda sequence, and "dev" against the percent deviance explained.
labsize size of labels for variable names. If labsize = 0, then no variable names will be plotted
xlab label for x-axis
ylab label for y-axis
main main title for plot
... other graphical parameters for the plot
sign.lambda Either plot against log(lambda) (default) or its negative if sign.lambda = -1.

Examples

set.seed(123)
set.seed(123)

predict.cv.hd2part Prediction function for fitted cross validation hd2part objects

Description

Prediction function for fitted cross validation hd2part objects
Usage

## S3 method for class 'cv.hd2part'
predict(
  object,
  newx,
  model = c("zero", "positive"),
  s = c("lambda.min", "lambda.1se"),
  type = c("link", "model_response", "response", "coefficients", "nonzero"),
  ...
)

Arguments

object      fitted "cv.hd2part" model object
newx        Matrix of new values for x at which predictions are to be made. Must be a
            matrix; can be sparse as in the CsparseMatrix objects of the Matrix package
            This argument is not used for type = c("coefficients", "nonzero")
model       either "zero" for the zero part model or "positive" for the positive part model
s           Value(s) of the penalty parameter lambda at which predictions are required. De-
            fault is the entire sequence used to create the model. For predict.cv.hd2part(),
            can also specify "lambda.1se" or "lambda.min" for best lambdas estimated by
            cross validation.
type        Type of prediction required. type = "link" gives the linear predictors; type
            = "model_response" gives the fitted probabilities for the zero part and fitted
            expected values for the positive part. type = "response" gives the combined
            response prediction across the two models using the full unconditional expected
            value of the response. When type = "response", argument "model" is unused.
            type = "coefficients" computes the coefficients at the requested values for
            s.
...          arguments to be passed to predict.hd2part

Examples

set.seed(123)

---

predict.hd2part  Prediction method for two part fitted objects

Description

Prediction method for two part fitted objects
## S3 method for class 'hd2part'
predict(
  object,
  newx,
  s = NULL,
  model = c("zero", "positive"),
  type = c("link", "model_response", "response", "coefficients", "nonzero"),
  newoffset = NULL,
  ...
)

Arguments

- **object**: fitted "hd2part" model object
- **newx**: Matrix of new values for x at which predictions are to be made. Must be a matrix
- **s**: Value(s) of the penalty parameter lambda for the zero part at which predictions are required. Default is the entire sequence used to create the model.
- **model**: either "zero" for the zero part model or "positive" for the positive part model
- **type**: Type of prediction required. type = "link" gives the linear predictors; type = "model_response" gives the fitted probabilities for the zero part and fitted expected values for the positive part. type = "response" gives the combined response prediction across the two models using the full unconditional expected value of the response. When type = "response", argument "model" is unused. type = "coefficients" computes the coefficients at the requested values for s.
- **newoffset**: f an offset is used in the fit, then one must be supplied for making predictions
- **...**: not used

Value

An object depending on the type argument

Examples

```r
set.seed(1)
```

**sim_semicontinuous_data**

Generates data from a two part distribution with a point mass at zero and heterogeneous treatment effects

**Description**

Generates semicontinuous data with heterogeneity of treatment effect
sim_semicontinuous_data

Usage

sim_semicontinuous_data(n.obs = 1000, n.vars = 25)

Arguments

n.obs number of observations
n.vars number of variables. Must be at least 10

Value

returns list with values y for outcome, x for design matrix, trt for treatment assignments, betanonzero for true coefficients for treatment-covariate interactions for model for whether or not a response is nonzero, betapos for true coefficients for treatment-covariate interactions for positive model, treatment_risk_ratio for the true covariate-conditional treatment effect risk ratio for each observation, pi.x for the true underlying propensity score
Index

cv.hd2part, 2, 5
fit_subgroup_2part, 3
hd2part, 3, 6
hdgamma, 8
HDtweedie, 10
HDtweedie_kfold_aug, 10

cv.hd2part (plot.hd2part), 10
plot.hd2part, 10
predict.cv.hd2part, 11
predict.hd2part, 12, 12

sim_semicontinuous_data, 13