Package 'matrans'

January 12, 2024

Title Model Averaging-Assisted Optimal Transfer Learning

Version 0.1.0

Description Transfer learning, as a prevailing technique in computer sciences, aims to improve the performance of a target model by leveraging auxiliary information from heterogeneous source data. We provide novel tools for multi-source transfer learning under statistical models based on model averaging strategies, including linear regression models, partially linear models. Unlike existing transfer learning approaches, this method integrates the auxiliary information through data-driven weight assignments to avoid negative transfer. This is the first package for transfer learning based on the optimal model averaging frameworks, providing efficient implementations for practitioners in multi-source data modeling. The details are described in Hu and Zhang (2023) <https://jmlr.org/papers/v24/23-0030.html>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Imports caret, formatR, MASS, quadprog, splines, stats

Suggests knitr, rmarkdown

Depends R (>= 2.10)

VignetteBuilder knitr

NeedsCompilation no

Author Xiaonan Hu [aut, cre] (<https://orcid.org/0000-0001-7614-2249>), Xinyu Zhang [aut]

Maintainer Xiaonan Hu <xiaonanhu@cnu.edu.cn>

Repository CRAN

Date/Publication 2024-01-12 16:30:06 UTC

R topics documented:

`pred.transsmap` *Prediction for new data based on Trans-SMAP.*

Description

Obtain predictions from a "trans.smap" object based on new samples.

Usage

```
pred.transsmap(object, newdata, bs.para, if.lm = FALSE)
```

Arguments

`object`	the output of function `trans.smap`.
`newdata`	a list containing the new observations of predictors for prediction, the components of which is named as "data.x" for parametric variables and "data.z" for nonparametric variables. Should be in accordance with the data for training `object`.
`bs.para`	a list containing the parameters for B-spline construction in function `bs`. Should be a vector with names "bs.df" and "bs.degree", each component of which is a vector with the same length as the number of nonparametric variables. For example, bs.para = list(bs.df=c(3,3,3), bs.degree=c(3,3,3)). • "bs.df": degrees of freedom for each nonparametric component; The details can be referred to the arguments in function `bs`. • "bs.degree": degree of the piecewise polynomial for each nonparametric component; The default is 3 for cubic splines.
`if.lm`	the logical variable, whether to set the target model as ordinary linear model. Default is False.

Value

a result list containing the predicted values on new data and the estimated coefficient vector.

References

Hu, X., & Zhang, X. (2023). Optimal Parameter-Transfer Learning by Semiparametric Model Averaging. Journal of Machine Learning Research, 24(358), 1-53.

See Also

`trans.smap`.

Examples

```
## correct target model setting

# generate simulation dataset
coeff0 <- cbind(
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3)),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3) + 0.02),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3) + 0.3),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3))
)
whole.data <- simdata.gen(
  px = 6, num.source = 4, size = c(150, 200, 200, 150), coeff0 = coeff0,
  coeff.mis = as.matrix(c(coeff0[, 2], 1.8)), err.sigma = 0.5, rho = 0.5, size.test = 500,
  sim.set = "homo", tar.spec = "cor", if.heter = FALSE
)
data.train <- whole.data$data.train
data.test <- whole.data$data.test

# running Trans-SMAP and obtain the optimal weight vector
data.train$data.x[[2]] <- data.train$data.x[[2]][, -7]
fit.transsmap <- trans.smap(
  train.data = data.train, nfold = 5,
  bs.para = list(bs.df = rep(3, 3), bs.degree = rep(3, 3))
)
ma.weights <- fit.transsmap$weight.est

# predict for new data
pred.res <- pred.transsmap(
  object = fit.transsmap, newdata = data.test,
  bs.para = list(bs.df = rep(3, 3), bs.degree = rep(3, 3))
)
pred.val <- pred.res$predict.val
predict.risk <- sum((pred.val - data.test$data.x %*% data.test$beta.true - data.test$gz.te)^2) / 500


## misspecified target model setting

# generate simulation dataset
coeff.mis <- matrix(c(c(coeff0[, 1], 0.1), c(coeff0[, 2], 1.8)), ncol = 2)
whole.data <- simdata.gen(
  px = 6, num.source = 4, size = c(150, 200, 200, 150), coeff0 = coeff0,
  coeff.mis = coeff.mis, err.sigma = 0.5, rho = 0.5, size.test = 500,
  sim.set = "homo", tar.spec = "mis", if.heter = FALSE
)
data.train <- whole.data$data.train
data.test <- whole.data$data.test

# running Trans-SMAP and obtain the optimal weight vector
data.train$data.x[[1]] <- data.train$data.x[[1]][, -7]
data.train$data.x[[2]] <- data.train$data.x[[2]][, -7]
fit.transsmap <- trans.smap(
  train.data = data.train, nfold = 5,
```

```
  bs.para = list(bs.df = rep(3, 3), bs.degree = rep(3, 3))
)
ma.weights <- fit.transsmap$weight.est

# predict for new data
data.test.mis <- data.test
data.test.mis$data.x <- data.test.mis$data.x[, -7]
pred.res <- pred.transsmap(
  object = fit.transsmap, newdata = data.test.mis,
  bs.para = list(bs.df = rep(3, 3), bs.degree = rep(3, 3))
)
pred.val <- pred.res$predict.val
predict.risk <- sum((pred.val - data.test$data.x %*% data.test$beta.true - data.test$gz.te)^2) / 500
```

`simdata.gen` *Generate multi-source data from partially linear models.*

Description

Generate simulation datasets containing training data and testing data from partially linear models under various settings.

Usage

```
simdata.gen(
  px,
  num.source = 4,
  size,
  coeff0,
  coeff.mis,
  err.sigma,
  rho,
  size.test,
  sim.set = c("heter", "homo"),
  tar.spec = c("cor", "mis"),
  if.heter = FALSE
)
```

Arguments

`px`	the dimension of the shared parametric component for all models. Should be an integer smaller than sample size.
`num.source`	the number of datasets. Should be the value 4 or 7.
`size`	the sample size of different datasets. Should be a vector of `num.source`.
`coeff0`	a px * num.source matrix of the shared coefficient vector for all models.

coeff.mis	the shared coefficient vector for the misspecified model. If tar.spec = 'cor', it should be a parameter vector of length px + 1 for the second misspecified source model. If tar.spec = 'mis', it should be a (px+1) * 2 matrix, in which the first column is the parameter vector for the misspecified target model and the second column is for the second misspecified source model. The last component of predictors for the misspecified model will be omitted in the estimation.
err.sigma	the standard deviations of the normal random errors in regression models.
rho	the correlation coefficient in the multivariate normal distribution of the parametric variables.
size.test	the sample size of the testing target data.
sim.set	the type of the nonparametric settings. Can be "heter" or "homo", which represents the heterogeneous and homogeneous dimension settings, respectively.
tar.spec	the type of the target model specification. Can be "cor" or "mis", which represents the corrected and misspecified target model, respectively.
if.heter	the logical variable, whether to allow a heteroscedastic setup. Default is False.

Value

a list of the training data and testing data, including the response, parametric predictors, nonparametric predictors, nonparametric values, coefficient vector.

References

Hu, X., & Zhang, X. (2023). Optimal Parameter-Transfer Learning by Semiparametric Model Averaging. Journal of Machine Learning Research, 24(358), 1-53.

Examples

```
coeff0 <- cbind(
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3)),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3) + 0.02),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3) + 0.3),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3))
)
# correct target model setting
whole.data <- simdata.gen(
  px = 6, num.source = 4, size = c(150, 200, 200, 150), coeff0 = coeff0,
  coeff.mis = as.matrix(c(coeff0[, 2], 1.8)), err.sigma = 0.5, rho = 0.5, size.test = 500,
  sim.set = "homo", tar.spec = "cor", if.heter = FALSE
)

# misspecified target model setting
coeff.mis <- matrix(c(c(coeff0[, 1], 0.1), c(coeff0[, 2], 1.8)), ncol = 2)
whole.data <- simdata.gen(
  px = 6, num.source = 4, size = c(150, 200, 200, 150), coeff0 = coeff0,
  coeff.mis = coeff.mis, err.sigma = 0.5, rho = 0.5, size.test = 500,
  sim.set = "homo", tar.spec = "mis", if.heter = FALSE
)
```

`trans.smap` *Parameter-transfer learning for partially linear models based on semi-parametric model averaging.*

Description

Obtain optimal weights and estimated coefficients based on Trans-SMAP.

Usage

```
trans.smap(train.data, nfold = NULL, bs.para, lm.set = NULL)
```

Arguments

`train.data`	a list containing the observations of predictors and response for fitting models. Should be a list with elements "data.y", "data.x" and "data.z", where "data.y" indicates a response list for all data sources, "data.x" indicates a parametric predictor list for all data sources, and "data.z" indicates a nonparametric predictor list for all data sources. Each element in "data.x" and "data.z" is a matrix with each row as an observation and each column as a variable. By default, the first element in "data.y", "data.x" and "data.z" is target data, and others are source data.
`nfold`	the number of folds for the cross-validation weight criterion. Default is NULL (leave-one-out).
`bs.para`	a list containing the parameters for B-spline construction in function `bs`. Should be a list with elements "bs.df" and "bs.degree", each component of which is a vector with the same length as the number of nonparametric variables. For example, bs.para = list(bs.df=c(3,3,3), bs.degree=c(3,3,3)). • "bs.df": degrees of freedom for each nonparametric component; The details can be referred to the arguments in function `bs`. • "bs.degree": degree of the piecewise polynomial for each nonparametric component; The default is 3 for cubic splines.
`lm.set`	the vector of indices for the linear regression models, which means the corresponding models are constructed by ordinary linear models instead of partially linear models. Default is NULL.

Value

a result list containing the estimated weight vector, the execution time of solving the optimal weights and the summarized results of fitting models.

References

Hu, X., & Zhang, X. (2023). Optimal Parameter-Transfer Learning by Semiparametric Model Averaging. Journal of Machine Learning Research, 24(358), 1-53.

Examples

```
## correct target model setting

# generate simulation dataset
coeff0 <- cbind(
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3)),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3) + 0.02),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3) + 0.3),
  as.matrix(c(1.4, -1.2, 1, -0.8, 0.65, 0.3))
)
whole.data <- simdata.gen(
  px = 6, num.source = 4, size = c(150, 200, 200, 150), coeff0 = coeff0,
  coeff.mis = as.matrix(c(coeff0[, 2], 1.8)), err.sigma = 0.5, rho = 0.5, size.test = 500,
  sim.set = "homo", tar.spec = "cor", if.heter = FALSE
)
data.train <- whole.data$data.train
data.test <- whole.data$data.test

# running Trans-SMAP and obtain the optimal weight vector
data.train$data.x[[2]] <- data.train$data.x[[2]][, -7]
fit.transsmap <- trans.smap(
  train.data = data.train, nfold = 5,
  bs.para = list(bs.df = rep(3, 3), bs.degree = rep(3, 3))
)
ma.weights <- fit.transsmap$weight.est


## misspecified target model setting

# generate simulation dataset
coeff.mis <- matrix(c(c(coeff0[, 1], 0.1), c(coeff0[, 2], 1.8)), ncol = 2)
whole.data <- simdata.gen(
  px = 6, num.source = 4, size = c(150, 200, 200, 150), coeff0 = coeff0,
  coeff.mis = coeff.mis, err.sigma = 0.5, rho = 0.5, size.test = 500,
  sim.set = "homo", tar.spec = "mis", if.heter = FALSE
)
data.train <- whole.data$data.train
data.test <- whole.data$data.test

# running Trans-SMAP and obtain the optimal weight vector
data.train$data.x[[1]] <- data.train$data.x[[1]][, -7]
data.train$data.x[[2]] <- data.train$data.x[[2]][, -7]
fit.transsmap <- trans.smap(
  train.data = data.train, nfold = 5,
  bs.para = list(bs.df = rep(3, 3), bs.degree = rep(3, 3))
)
ma.weights <- fit.transsmap$weight.est
```

Index