Package ‘dipw’

November 30, 2020

Type Package
Title Debiased Inverse Propensity Score Weighting
Description Estimation of the average treatment effect when controlling for high-dimensional confounders using debiased inverse propensity score weighting (DIPW). DIPW relies on the propensity score following a sparse logistic regression model, but the regression curves are not required to be estimable. Despite this, our package also allows the users to estimate the regression curves and take the estimated curves as input to our methods. Details of the methodology can be found in Yuhao Wang and Rajen D. Shah (2020) “Debiased Inverse Propensity Score Weighting for Estimation of Average Treatment Effects with High-Dimensional Confounders” <arXiv:2011.08661>. The package relies on the optimisation software ‘MOSEK’ <https://www.mosek.com/> which must be installed separately; see the documentation for ‘Rmosek’.

Version 0.1.0
Maintainer Yuhao Wang <yuhaow.thu@gmail.com>
Imports glmnet, Rmosek, Matrix, methods, stats
License GPL-3
Encoding UTF-8
LazyData false
RoxygenNote 7.1.1
NeedsCompilation no
Author Yuhao Wang [cre, aut], Rajen Shah [ctb]
Repository CRAN
Date/Publication 2020-11-30 09:00:05 UTC

R topics documented:

  dipw.ate ................................................................. 2
  dipw.mean ............................................................ 3

Index  5
**dipw.ate**

*Estimate the Average treatment effect \( E[Y(1) - Y(0)] \) from observational data*

---

**Description**

Estimate the Average treatment effect \( E[Y(1) - Y(0)] \) from observational data

**Usage**

```r
dipw.ate(
  X,
  Y,
  W,
  r1 = NULL,
  r0 = NULL,
  kappa = 0.5,
  splitting = c("1", "3", "random"),
  B = 1,
  ...
)
```

**Arguments**

- **X**: the \( n \) by \( p \) input covariance matrix
- **Y**: the \( n \) dimensional observed response
- **W**: the \( n \) dimensional binary vector indicating treatment assignment
- **r1**: optional \( n \) dimensional vector of an initial estimate of \( E[Y(1)|X_i] \) for \( i = 1, \ldots, n \). The default is NULL.
- **r0**: optional \( n \) dimensional vector of an initial estimate of \( E[Y(0)|X_i] \) for \( i = 1, \ldots, n \). The default is NULL.
- **kappa**: the weight parameter for quadratic programming. Default is 0.5.
- **splitting**: the options for splitting. "1" means \( B = 1 \) split, "3" means \( B = 3 \) splits, "random" means random splits.
- **B**: the number of iterations for random splits, the default is 1. Only used when splitting is set to "random".
- **...**: additional arguments that can be passed to `cv.glmnet`

**Value**

- **tau**: the estimated average treatment effect

**References**

Examples

```r
## Not run:
# Estimating average treatment effect with a toy data
# Notice that the external optimisation software \code{MOSEK}
# must be installed separately before running the example code.
# Without \code{MOSEK}, the example code is not executable.
# For how to install \code{MOSEK}, see documentation of \code{\link[Rmosek]{Rmosek}}.
set.seed(1)
n <- 100; p <- 200
X <- scale(matrix(rnorm(n*p), n, p))
W <- rbinom(n, 1, 1 / (1 + exp(-X[, 1])))
Y <- X[,1] + W * X[,2] + rnorm(n)
# Getting an estimate of average treatment effect
(est <- dipw.ate(X, Y, W))
## End(Not run)
```

```r
dipw.mean
```

### Not run:

```
# Estimating average treatment effect with a toy data
# Notice that the external optimisation software \code{MOSEK}
# must be installed separately before running the example code.
# Without \code{MOSEK}, the example code is not executable.
# For how to install \code{MOSEK}, see documentation of \code{\link[Rmosek]{Rmosek}}.
```

```
set.seed(1)
n <- 100; p <- 200
X <- scale(matrix(rnorm(n*p), n, p))
W <- rbinom(n, 1, 1 / (1 + exp(-X[, 1])))
Y <- X[,1] + W * X[,2] + rnorm(n)
# Getting an estimate of average treatment effect
(est <- dipw.ate(X, Y, W))
```

### Description

Estimation of \(E[Y(1)]\) or \(E[Y(0)]\) from observational data

### Usage

```
dipw.mean(
  X,
  Y,
  W,
  Treated = TRUE,
  r = NULL,
  kappa = 0.5,
  splitting = c("1", "3", "random"),
  B = 1,
  ...
)
```

### Arguments

- **X**: the \(n\) by \(p\) input covariance matrix
- **Y**: the \(n\) dimensional observed response
- **W**: the \(n\) dimensional binary vector indicating treatment assignment
- **Treated**: \(\text{TRUE}\) if we seek to estimate \(E[Y(1)]\). \(\text{FALSE}\) if we instead wish to estimate \(E[Y(0)]\). The default is \(\text{TRUE}\)
- **r**: optional \(n\) dimensional vector containing initial estimates of \(E[Y(\text{Treated}) \mid X_i] \) for \(i = 1, \ldots, n\). The default is \(\text{NULL}\)
kappa  the weight parameter for quadratic programming. Default is 0.5
splitting the options for splitting. "1" means B = 1 split, "3" means B = 3 splits, "random" means random splits.
B  the number of iterations for random splits, the default is 1. Only valid when splitting is set to "random".
...  additional arguments that can be passed to cv.glmnet

Value
the expectation E[Y(1)] or E[Y(0)]

References

Examples
## Not run:
# Estimating mean of the potential outcome with a toy data
# Notice that the external optimisation software \code{MOSEK}
# must be installed separately before running the example code.
# Without \code{MOSEK}, the example code is not executable.
# For how to install \code{MOSEK}, see documentation of \code{\link[Rmosek]{Rmosek}}.
set.seed(1)
n <- 100; p <- 200
X <- scale(matrix(rnorm(n*p), n, p))
W <- rbinom(n, 1, 1 / (1 + exp(-X[, 1])))
Y <- X[,1] + W * X[,2] + rnorm(n)
# Getting an estimate of potential outcome mean
(est <- dipw.mean(X, Y, W, Treated=TRUE))

## End(Not run)
Index

cv.glmnet, 2, 4

dipw.ate, 2
dipw.mean, 3