Package 'DistributionIV'

June 27, 2025

Type Package

Title Distributional Instrumental Variable (DIV) Model

Version 0.1.2

Description Distributional instrumental variable (DIV) model
for estimation of the interventional distribution of the outcome Y under a
do-intervention on the treatment X. Instruments, predictors and targets can
be univariate or multivariate. Functionality includes estimation of the (conditional)
interventional mean and quantiles, as well as sampling from the fitted (conditional)
interventional distribution.

License MIT + file LICENSE

Encoding UTF-8

Imports torch, checkmate

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Maintainer Anastasiia Holovchak <anastasiia.holovchak@stat.math.ethz.ch>

NeedsCompilation no

Author Anastasiia Holovchak [aut, cre, ctb],
Sorawit Saengkyongam [aut, ctb],
Nicolai Meinshausen [aut, ctb],
Xinwei Shen [aut, ctb]

Repository CRAN

Date/Publication 2025-06-27 13:40:09 UTC

Contents

div *Distributional IV Model Function*

Description

This function fits a distributional IV model to the data. It allows for the tuning of several parameters related to model complexity and model training. Variables are per default internally standardized (predictions are on the original scale).

Usage

```
div(
  Z,
  X,
  Y,
  W = NULL,
  epsx_dim = 50,
  epsy_dim = 50,
  epsh_dim = 50,
  hidden_dim = 100,
  num_layer = 3,
  num_epochs = 1000,
  lr = 10^(-3),
  beta = 1,
  silent = FALSE,
  standardize = TRUE
)
```

Arguments

Z	A data frame, matrix, vector, or factor variable representing the instrumental variable.
X	A data frame, matrix, vector, or factor variable representing the predictor.
Y	A data frame, matrix, vector, or factor variable representing the target variable.
W	(Optional) A data frame, matrix, vector, or factor variable representing the exogenous variable(s).
epsx_dim	The dimension of the noise corresponding to the predictor introduced in the model (default: 50).
epsy_dim	The dimension of the noise corresponding to the outcome introduced in the model (default: 50).
epsh_dim	The dimension of the noise corresponding to the hidden confounder introduced in the model (default: 50).
hidden_dim	The size of the hidden layer in the model (default: 100).
num_layer	The number of layers in the model (default: 3).

num_epochs	The number of epochs to be used in training (default: 1000).
lr	The learning rate to be used in training (default: 10^-3).
beta	The beta scaling factor for energy loss, numeric value from (0,2) (default: 1).
silent	A boolean indicating whether to suppress output during model training (default: FALSE).
standardize	A boolean indicating whether to standardize the input data (default: TRUE).

Value

A distributional IV model object with class 'DIV'.

Examples

```
## Not run:
# NOTE: This example requires torch runtime.
# Please install it with: torch::install_torch()

# Simulate data ------------------------------------------------------------
p_Z <- 4
p_X <- 2

set.seed(2209)
n_train <- 1000
Z <- matrix(rnorm(n_train * p_Z, mean = 2), ncol = p_Z)
H <- rnorm(n_train, mean = 0, sd = 1.5)
X1 <- 0.1 * (Z[, 1] + rnorm(n_train, sd = 0.1)) ^ 2 +
 (Z[, 2] + rnorm(n_train, sd = 1)) ^ 2 + H + rnorm(n_train, sd = 0.1)
X2 <- 0.5 * (Z[, 3] + Z[, 4]) ^ 2 + 0.1 * H ^ 2 + rnorm(n_train, sd = 0.1)
X <- matrix(cbind(X1, X2), ncol = p_X)
Y <- 0.5 * X[, 1] + 0.2 * (X[, 2] + rnorm(n_train, sd = 0.2) + H) ^ 2 +
 rnorm(n_train, sd = 0.1)

n_test <- n_train
Ztest <- matrix(rnorm(n_test * p_Z, mean = 2), ncol = p_Z)
Htest <- rnorm(n_test, mean = 0, sd = 1.5)
X1test <- 0.1 * (Ztest[, 1] + rnorm(n_test, sd = 0.1)) ^ 2 +
 (Ztest[, 2] + rnorm(n_test, sd = 1)) ^ 2 + Htest + rnorm(n_test, sd = 0.1)
X2test <- 0.5 * (Ztest[, 3] + Ztest[, 4]) ^ 2 + 0.1 * Htest ^ 2 + rnorm(n_test, sd = 0.1)
Xtest <- matrix(cbind(X1test, X2test), ncol = p_X)
Ytest <- 0.5 * Xtest[, 1] + 0.2 * (Xtest[, 2] + rnorm(n_test, sd = 0.2) + Htest) ^ 2 +
 rnorm(n_test, sd = 0.1)

# Fit generativeIV model ---------------------------------------------------
# Consider increasing number of epochs. Here: num_epochs = 100 for fast computation only.
DIV_model <- div(Z = Z, X = X, Y = Y, num_epochs = 100)
print(DIV_model)

# Prediction on test data --------------------------------------------------
Yhat <- predict(object = DIV_model, Xtest = Xtest, type = "mean")
cat("\n Correlation between predicted and realized values: ", signif(cor(Yhat, Ytest), 3))
plot(Yhat, Ytest, xlab = "model prediction", ylab = "observation")
```

```
# Quantile prediction -----------------------------------------------------
Yhat_quant <- predict(object = DIV_model, Xtest = Xtest, type = "quantile")
ord <- order(Yhat)
matplot(Yhat[ord], Yhat_quant[ord,], type = "l", col = 2, lty = 1,
xlab = "model prediction", ylab = "observation")
points(Yhat[ord], Ytest[ord], pch = 20, cex = 0.5)

#' # Sampling from estimated model ------------------------------------------
Ysample <- predict(object = DIV_model, Xtest = Xtest, type = "sample", nsample = 1)

#' # Plots of realized & sampled values against first variable ----------------
oldpar <- par(no.readonly = TRUE)
par(mfrow = c(1, 2))
plot(Xtest[, 1], Ytest, xlab = "Predictor Variable 1", ylab = "Observation")
plot(Xtest[, 1], Ysample, xlab = "Predictor Variable 1", ylab = "Model sample")
par(oldpar)

## End(Not run)
```

`predict.DIV` *Prediction Function for a DIV Model Object*

Description

This function computes predictions from a trained DIV model. It allows for estimation of the interventional mean and quantiles, as well as sampling from the fitted interventional distribution. If the model includes exogenous predictors, it allows for estimation of the conditional interventional mean and quantiles, as well as sampling from the fitted conditional interventional distribution.

Usage

```
## S3 method for class 'DIV'
predict(
  object,
  Xtest,
  Wtest = NULL,
  type = c("mean", "sample", "quantile")[1],
  trim = 0.05,
  quantiles = 0.1 * (1:9),
  nsample = 200,
  drop = TRUE,
  ...
)
```

Arguments

`object`	A trained DIV model returned from div or divfit functions.

Xtest	A matrix or data frame representing predictors in the test set.
Wtest	A matrix or data frame representing exogenous predictors in the test set. If the model includes exogenous predictors, 'Wtest' has to be specified for computation of conditional treatment estimates or to draw samples from the conditional interventional distribution.
type	The type of prediction to make: * 'mean': for point estimates (the default). * 'sample': for samples from the estimated distribution. * 'quantile'/'quantiles': for quantiles of the estimated distribution.
trim	The proportion of extreme values to trim when calculating the mean (default: 0.05).
quantiles	The quantiles to estimate if type is 'quantile' (default: 0.1*(1:9)).
nsample	The number of samples to draw if type is 'sample' (default: 200).
drop	A boolean indicating whether to drop dimensions of length 1 from the output (default: TRUE).
...	additional arguments (currently ignored).

Value

A vector or matrix/array of predictions.

Examples

```
## Not run:
# NOTE: This example requires torch runtime.
# Please install it with: torch::install_torch()

# Simulate data ------------------------------------------------------------
p_Z <- 4
p_X <- 2

set.seed(2209)
n_train <- 1000
Z <- matrix(rnorm(n_train * p_Z, mean = 2), ncol = p_Z)
H <- rnorm(n_train, mean = 0, sd = 1.5)
X1 <- 0.1 * (Z[, 1] + rnorm(n_train, sd = 0.1)) ^ 2 +
 (Z[, 2] + rnorm(n_train, sd = 1)) ^ 2 + H + rnorm(n_train, sd = 0.1)
X2 <- 0.5 * (Z[, 3] + Z[, 4]) ^ 2 + 0.1 * H ^ 2 + rnorm(n_train, sd = 0.1)
X <- matrix(cbind(X1, X2), ncol = p_X)
Y <- 0.5 * X[, 1] + 0.2 * (X[, 2] + rnorm(n_train, sd = 0.2) + H) ^ 2 +
 rnorm(n_train, sd = 0.1)
n_test <- n_train
Ztest <- matrix(rnorm(n_test * p_Z, mean = 2), ncol = p_Z)
Htest <- rnorm(n_test, mean = 0, sd = 1.5)
X1test <- 0.1 * (Ztest[, 1] + rnorm(n_test, sd = 0.1)) ^ 2 +
 (Ztest[, 2] + rnorm(n_test, sd = 1)) ^ 2 + Htest + rnorm(n_test, sd = 0.1)
X2test <- 0.5 * (Ztest[, 3] + Ztest[, 4]) ^ 2 + 0.1 * Htest ^ 2 + rnorm(n_test, sd = 0.1)
Xtest <- matrix(cbind(X1test, X2test), ncol = p_X)
Ytest <- 0.5 * Xtest[, 1] + 0.2 * (Xtest[, 2] + rnorm(n_test, sd = 0.2) + Htest) ^ 2 +
 rnorm(n_test, sd = 0.1)
```

```
# Fit DIV model ------------------------------------------------------------
# Consider increasing number of epochs. Here: num_epochs = 100 for fast computation only.
DIV_model <- div(Z = Z, X = X, Y = Y, num_epochs = 100)
print(DIV_model)

# Prediction on test data --------------------------------------------------
Yhat <- predict(object = DIV_model, Xtest = Xtest, type = "mean")
cat("\n Correlation between predicted and realized values: ", signif(cor(Yhat, Ytest), 3))
plot(Yhat, Ytest, xlab = "model prediction", ylab = "observation")
# Quantile prediction ------------------------------------------------------
Yhat_quant <- predict(object = DIV_model, Xtest = Xtest, type = "quantile")
ord <- order(Yhat)
matplot(Yhat[ord], Yhat_quant[ord,], type = "l", col = 2, lty = 1,
xlab = "model prediction", ylab = "observation")
points(Yhat[ord], Ytest[ord], pch = 20, cex = 0.5)

#' # Sampling from estimated model ------------------------------------------
Ysample <- predict(object = DIV_model, Xtest = Xtest, type = "sample", nsample = 1)

#' # Plots of realized & sampled values against first variable ----------------
oldpar <- par(no.readonly = TRUE)
par(mfrow = c(1, 2))
plot(Xtest[, 1], Ytest, xlab = "Predictor Variable 1", ylab = "Observation")
plot(Xtest[, 1], Ysample, xlab = "Predictor Variable 1", ylab = "Model sample")
par(oldpar)

## End(Not run)
```

`print.DIV` *Print Function for a DIV Model Object*

Description

This function is a utility that displays a summary of a fitted DIV model object.

Usage

```
## S3 method for class 'DIV'
print(x, ...)
```

Arguments

`x`	A trained DIV model returned from the divfit() function.
`...`	additional arguments (currently ignored).

Value

This function does not return anything. It prints a summary of the model, including information about its architecture and training process, and the loss values achieved at several epochs during training.

Examples

```
## Not run:
# NOTE: This example requires torch runtime.
# Please install it with: torch::install_torch()
# Simulate data ------------------------------------------------------------
p_Z <- 4
p_X <- 2

set.seed(2209)
n_train <- 1000
Z <- matrix(rnorm(n_train * p_Z, mean = 2), ncol = p_Z)
H <- rnorm(n_train, mean = 0, sd = 1.5)
X1 <- 0.1 * (Z[, 1] + rnorm(n_train, sd = 0.1)) ^ 2 +
 (Z[, 2] + rnorm(n_train, sd = 1)) ^ 2 + H + rnorm(n_train, sd = 0.1)
X2 <- 0.5 * (Z[, 3] + Z[, 4]) ^ 2 + 0.1 * H ^ 2 + rnorm(n_train, sd = 0.1)
X <- matrix(cbind(X1, X2), ncol = p_X)
Y <- 0.5 * X[, 1] + 0.2 * (X[, 2] + rnorm(n_train, sd = 0.2) + H) ^ 2 +
 rnorm(n_train, sd = 0.1)
n_test <- n_train
Ztest <- matrix(rnorm(n_test * p_Z, mean = 2), ncol = p_Z)
Htest <- rnorm(n_test, mean = 0, sd = 1.5)
X1test <- 0.1 * (Ztest[, 1] + rnorm(n_test, sd = 0.1)) ^ 2 +
 (Ztest[, 2] + rnorm(n_test, sd = 1)) ^ 2 + Htest + rnorm(n_test, sd = 0.1)
X2test <- 0.5 * (Ztest[, 3] + Ztest[, 4]) ^ 2 + 0.1 * Htest ^ 2 + rnorm(n_test, sd = 0.1)
Xtest <- matrix(cbind(X1test, X2test), ncol = p_X)
Ytest <- 0.5 * Xtest[, 1] + 0.2 * (Xtest[, 2] + rnorm(n_test, sd = 0.2) + Htest) ^ 2 +
 rnorm(n_test, sd = 0.1)

# Fit DIV model ------------------------------------------------------
# Consider increasing number of epochs. Here: num_epochs = 100 for fast computation only.
DIV_model <- div(Z = Z, X = X, Y = Y, num_epochs = 100)
print(DIV_model)

## End(Not run)
```

Index