Package 'DNNSIM'

January 20, 2025

Type Package

Title Single-Index Neural Network for Skewed Heavy-Tailed Data

Version 0.1.1

Maintainer Qingyang Liu <rh8liuqy@gmail.com>

Description Provides a deep neural network model with a monotonic increasing single index function tailored for periodontal disease studies. The residuals are assumed to follow a skewed T distribution, a skewed normal distribution, or a normal distribution. More details can be found at Liu, Huang, and Bai (2024) <doi:10.1016/j.csda.2024.108012>.

License GPL (>= 3)

Encoding UTF-8

RdMacros Rdpack

SystemRequirements Python (>= 3.8.0); PyTorch (https://pytorch.org/); NumPy (https://numpy.org/); SciPy (https://scipy.org/); sklearn (https://scikit-learn.org/stable/);

RoxygenNote 7.3.2

Imports reticulate (>= 1.37.0), stats (>= 4.3.0), Rdpack (>= 2.6)

NeedsCompilation no

Author Qingyang Liu [aut, cre] (<https://orcid.org/0000-0003-3265-6330>),
Shijie Wang [aut],
Ray Bai [aut] (<https://orcid.org/0000-0002-7190-7844>),
Dipankar Bandyopadhyay [aut]

Repository CRAN

Date/Publication 2025-01-07 16:50:13 UTC

Contents

data_simulation *Simulate data for the DNN-SIM model*

Description

Simulate data for the DNN-SIM model

Usage

```
data_simulation(n, beta, w, sigma, delta, seed)
```

Arguments

n	an integer. The sample size.
beta	a vector. The covariate coefficients.
w	a number between 0 and 1. The skewness parameter.
sigma	a number larger than 0. The standard deviation parameter.
delta	a number larger than 0. The degree of freedom parameter.
seed	an integer. The random seed.

Details

This is a simple data generation function for a simulation study. All elements of the design matrix X follow a uniform distribution from -3.0 and 3.0 independently and identically. The true g function is the standard logistic function.

Value

a dataframe of the simulated response variable y and the design matrix X.

References

Liu Q, Huang X, Bai R (2024). "Bayesian Modal Regression Based on Mixture Distributions." *Computational Statistics & Data Analysis*, 108012. doi:10.1016/j.csda.2024.108012.

Examples

```
# check python module dependencies
if (reticulate::py_module_available("torch") &
    reticulate::py_module_available("numpy") &
    reticulate::py_module_available("sklearn") &
    reticulate::py_module_available("scipy")) {
  df1 <- data_simulation(n=50,beta=c(1,1,1),w=0.3,
                         sigma=0.1,delta=4.0,seed=100)
  print(head(df1))
}
```

DNNSIM *The 'DNNSIM' package.*

Description

Provides a deep neural network model with a monotonic increasing single index function tailored for periodontal disease studies. The residuals are assumed to follow a skewed T distribution, a skewed normal distribution, or a normal distribution. More details can be found at Liu, Huang, and Bai (2024) doi:10.1016/j.csda.2024.108012.

Value

This is the summary page. No return value.

Author(s)

Maintainer: Qingyang Liu <rh8liuqy@gmail.com> (ORCID)

Authors:

- Shijie Wang <shijiew.usc@gmail.com>
- Ray Bai <rbai@mailbox.sc.edu> (ORCID)
- Dipankar Bandyopadhyay <dbandyop@vcu.edu>

DNN_model *Define and train the DNN-SIM model*

Description

Define and train the DNN-SIM model

Usage

```
DNN_model(
  formula,
  data,
  model,
  num_epochs,
  verbatim = TRUE,
  CV = FALSE,
  CV_K = 10,
  bootstrap = FALSE,
  bootstrap_B = 1000,
```

```
  bootstrap_num_epochs = 100,
  U_new = FALSE,
  U_min = -4,
  U_max = 4,
  random_state = 100
)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
data	a data frame.
model	the model type. It must be be one of "N-GX-D","SN-GX-D","ST-GX-D","N-GX-B","SN-GX-B","ST-GX-B","N-FX","SN-FX","ST-FX".
num_epochs	an integer. The number of complete passes through the training dataset.
verbatim	TRUE/FALSE.If `verbatim` is `TRUE`, then log information from training the DNN-SIM model will be printed.
CV	TRUE/FALSE. Whether use the cross-validation to measure the prediction accuracy.
CV_K	an integer. The number of folders K-folder cross-validation.
bootstrap	TRUE/FALSE. Whether use the bootstrap method to quantify the uncertainty. The bootstrap option ONLY works for the "ST-GX-D" model.
bootstrap_B	an integer. The number of bootstrap iteration.
bootstrap_num_epochs	an integer. The number of complete passes through the training dataset in the bootstrap procedure.
U_new	TRUE/FALSE. Whether use self defined U for the estimation of single index function, g(U).
U_min	a numeric value. The minimum of the self defined U.
U_max	a numeric value. The maximum of the self defined U.
random_state	an integer. The random seed for initiating the neural network.

Details

The DNNSIM model is defined as:

$$Y = g(\mathbf{X}\boldsymbol{\beta}) + e.$$

The residuals e follow a skewed T distribution, skewed normal distribution, or normal distribution. The single index function g is assumed to be a monotonic increasing function.

Value

A list consisting of the point estimation, g function estimation (optional), cross-validation results (optional) and bootstrap results(optional).

References

Liu Q, Huang X, Bai R (2024). "Bayesian Modal Regression Based on Mixture Distributions." *Computational Statistics & Data Analysis*, 108012. doi:10.1016/j.csda.2024.108012.

Examples

```
# check python module dependencies
if (reticulate::py_module_available("torch") &
    reticulate::py_module_available("numpy") &
    reticulate::py_module_available("sklearn") &
    reticulate::py_module_available("scipy")) {

  # set the random seed
  set.seed(100)

  # simulate some data
  df1 <- data_simulation(n=100,beta=c(1,1,1),w=0.3,
                         sigma=0.1,delta=10.0,seed=100)

  # the cross-validation and bootstrap takes a long time
  DNN_model_output <- DNN_model(y ~ X1 + X2 + X3 - 1,
                                data = df1,
                                model = "ST-GX-D",
                                num_epochs = 5,
                                verbatim = FALSE,
                                CV = TRUE,
                                CV_K = 2,
                                bootstrap = TRUE,
                                bootstrap_B = 2,
                                bootstrap_num_epochs = 5,
                                U_new = TRUE,
                                U_min = -4.0,
                                U_max = 4.0)
  print(DNN_model_output)
}
```

Index