Package 'twinning'

January 28, 2022

Type Package

Title Data Twinning

Version 1.0

Date 2022-01-26

Description An efficient algorithm for data twinning. This work is supported by U.S. National Science Foundation grants DMREF-1921873 and CMMI-1921646.

License GPL (>= 2)

Imports Rcpp (>= 1.0.4)

LinkingTo Rcpp

RoxygenNote 7.1.2

Encoding UTF-8

NeedsCompilation yes

Author Akhil Vakayil [aut, cre],
Roshan Joseph [aut, ths]

Maintainer Akhil Vakayil <akhilv@gatech.edu>

Repository CRAN

Date/Publication 2022-01-28 08:20:02 UTC

R topics documented:

twinning-package *Data Twinning*

Description

An efficient implementation of the twinning algorithm proposed in Vakayil and Joseph (2022) for partitioning a dataset into statistically similar twin sets. It is orders of magnitude faster than the SPlit algorithm proposed in Joseph and Vakayil (2021) for splitting a dataset into training and testing sets, and the support points algorithm of Mak and Joseph (2018) for subsampling from Big Data.

Details

The package provides functions `twin()`, `multiplet()`, and `energy()`. `twin()` partitions datasets into statistically similar disjoint sets, termed as *twins*. The twins themselves are statistically similar to the original dataset (Vakayil and Joseph, 2022). Such a partition can be employed for optimal training and testing of statistical and machine learning models (Joseph and Vakayil, 2021). The twins can be of unequal size; for tractable model building on large datasets, the smaller twin can serve as a compressed (lossy) version of the original dataset. `multiplet()` is an extension of `twin()` to generate multiple disjoint partitions that can be used for k-fold cross validation, or with divide-and-conquer procedures. `energy()` computes the energy distance (Székely and Rizzo, 2013) between a given dataset and a set of points, which is the metric minimized by twinning.

Author(s)

Akhil Vakayil, V. Roshan Joseph

Maintainer: Akhil Vakayil <akhilv@gatech.edu>

References

Vakayil, A., & Joseph, V. R. (2022). Data Twinning. Statistical Analysis and Data Mining: The ASA Data Science Journal, to appear. arXiv preprint arXiv:2110.02927.

Joseph, V. R., & Vakayil, A. (2021). SPlit: An Optimal Method for Data Splitting. Technometrics, 1-11. doi:10.1080/00401706.2021.1921037.

Mak, S. & Joseph, V. R. (2018). Support Points. Annals of Statistics, 46, 2562-2592.

Székely, G. J., & Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances. Journal of statistical planning and inference, 143(8), 1249-1272.

energy *Energy distance computation*

Description

`energy()` computes the energy distance (Székely and Rizzo, 2013) between a given dataset and a set of points in same dimensions.

Usage

```
energy(data, points)
```

Arguments

`data`	The dataset including both the predictors and response(s). A numeric matrix is expected. If the dataset has factor columns, the user is expected to convert them to numeric using a coding method.
`points`	The set of points for which the energy distance with respect to `data` is to be computed. A numeric matrix is expected.

Details

Smaller the energy distance, the more statistically similar the set of points is to the given dataset. The minimizer of energy distance is known as support points (Mak and Joseph, 2018), which is the basis of the twinning method. Computing energy distance between `data` and `points` involves Euclidean distance calculations among the rows of `data`, among the rows of `points`, and between the rows of `data` and `points`. Since, `data` serves as the reference, the distance calculations among the rows of `data` are ignored for efficiency. Before computing the energy distance, the columns of `data` are scaled to zero mean and unit standard deviation. The mean and standard deviation of the columns of `data` are used to scale the respective columns in `points`.

Value

Energy distance.

References

Vakayil, A., & Joseph, V. R. (2022). Data Twinning. Statistical Analysis and Data Mining: The ASA Data Science Journal, to appear. arXiv preprint arXiv:2110.02927.

Székely, G. J., & Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances. Journal of statistical planning and inference, 143(8), 1249-1272.

Mak, S. & Joseph, V. R. (2018). Support Points. Annals of Statistics, 46, 2562-2592.

Examples

```
## Energy distance between a dataset and a random sample
X = rnorm(n=100, mean=0, sd=1)
Y = rnorm(n=100, mean=X^2, sd=1)
data = cbind(X, Y)
energy(data, data[sample(100, 20), ])
```

`multiplet` *Partition datasets into multiple statistcally similar disjoint sets*

Description

`multiplet()` extends `twin()` to partition datasets into multiple statistically similar disjoint sets, termed as *multiplets*, under the three different strategies described in Vakayil and Joseph (2022).

Usage

```
multiplet(data, k, strategy = 1, format_data = TRUE, leaf_size = 8)
```

Arguments

`data`	The dataset including both the predictors and response(s); should not contain missing values, and only numeric and/or factor column(s) are allowed.
`k`	The desired number of multiplets.
`strategy`	An integer either 1, 2, or 3 referring to the three strategies for generating multiplets. Strategy 2 perfroms best, but requires `k` to be a power of 2. Strategy 3 is computatioanlly inexpensive, but performs worse than strategies 1 and 2.
`format_data`	If set to `TRUE`, constant columns in `data` are removed, factor columns are converted to numerical using Helmert coding, and then the columns are scaled to zero mean and unit standard deviation. If set to `FALSE`, the user is expected to perform data pre-processing.
`leaf_size`	Maximum number of elements in the leaf-nodes of the *kd*-tree.

Value

List with the multiplet id, ranging from 1 to `k`, for each row in `data`.

References

Vakayil, A., & Joseph, V. R. (2022). Data Twinning. Statistical Analysis and Data Mining: The ASA Data Science Journal, to appear. arXiv preprint arXiv:2110.02927.

Blanco, J. L. & Rai, P. K. (2014). nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. https://github.com/jlblancoc/nanoflann.

Examples

```
## 1. Generating 10 multiplets of a numeric dataset
X = rnorm(n=100, mean=0, sd=1)
Y = rnorm(n=100, mean=X^2, sd=1)
data = cbind(X, Y)
multiplet_idx = multiplet(data, k=10)
multiplet_1 = data[which(multiplet_idx == 1), ]
multiplet_10 = data[which(multiplet_idx == 10), ]

## 2. Generating 4 multiplets of the iris dataset using strategy 2
multiplet_idx = multiplet(iris, k=4, strategy=2)
multiplet_1 = iris[which(multiplet_idx == 1), ]
multiplet_4 = iris[which(multiplet_idx == 4), ]
```

`twin` *Partition datasets into statistcally similar twin sets*

Description

`twin()` implements the twinning algorithm presented in Vakayil and Joseph (2022). A partition of the dataset is returned, such that the resulting two disjoint sets, termed as *twins*, are distributed similar to each other, as well as the whole dataset. Such a partition is an optimal training-testing split (Joseph and Vakayil, 2021) for training and testing statistical and machine learning models, and is model-independent. The statistical similarity also allows one to treat either of the twins as a compression (lossy) of the dataset for tractable model building on Big Data.

Usage

```
twin(data, r, u1 = NULL, format_data = TRUE, leaf_size = 8)
```

Arguments

`data`	The dataset including both the predictors and response(s); should not contain missing values, and only numeric and/or factor column(s) are allowed.
`r`	An integer representing the inverse of the splitting ratio, e.g., for an 80-20 partition, `r = 1 / 0.2 = 5`.
`u1`	Index of the data point from where twinning starts; if not provided, twinning starts from a random point in the dataset. Fixing `u1` makes twinning deterministic, i.e., the same twins are returned.
`format_data`	If set to `TRUE`, constant columns in `data` are removed, factor columns are converted to numerical using Helmert coding, and then the columns are scaled to zero mean and unit standard deviation. If set to `FALSE`, the user is expected to perform data pre-processing.
`leaf_size`	Maximum number of elements in the leaf-nodes of the *kd*-tree.

Details

The twinning algorithm requires nearest neighbor queries that are performed using a *kd*-tree. The *kd*-tree implementation in the `nanoflann` (Blanco and Rai, 2014) C++ library is used.

Value

Indices of the smaller twin.

References

Vakayil, A., & Joseph, V. R. (2022). Data Twinning. Statistical Analysis and Data Mining: The ASA Data Science Journal, to appear. arXiv preprint arXiv:2110.02927.

Joseph, V. R., & Vakayil, A. (2021). SPlit: An Optimal Method for Data Splitting. Technometrics, 1-11. doi:10.1080/00401706.2021.1921037.

Blanco, J. L. & Rai, P. K. (2014). nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. https://github.com/jlblancoc/nanoflann.

Examples

```
## 1. An 80-20 partition of a numeric dataset
X = rnorm(n=100, mean=0, sd=1)
Y = rnorm(n=100, mean=X^2, sd=1)
data = cbind(X, Y)
twin1_indices = twin(data, r=5)
twin1 = data[twin1_indices, ]
twin2 = data[-twin1_indices, ]
plot(data, main="Smaller Twin")
points(twin1, col="green", cex=2)

## 2. An 80-20 split of the iris dataset
twin1_indices = twin(iris, r=5)
twin1 = iris[twin1_indices, ]
twin2 = iris[-twin1_indices, ]
```

Index