Package 'nntrf'

February 26, 2021

Title Supervised Data Transformation by Means of Neural Network Hidden Layer

Version 0.1.4

Description A supervised transformation of datasets is performed.
The aim is similar to that of Principal Component Analysis (PCA), that is, to carry out data transformation and dimensionality reduction, but in a supervised way.
This is achieved by first training a 3-layer Multi-Layer Perceptron and then using the activations of the hidden layer as a transformation of the input features.
In fact, it takes advantage of the change of representation provided by the hidden layer of a neural network.
This can be useful as data pre-processing for Machine Learning methods in general, specially for those that do not work well with many irrelevant or redundant features.
It uses the nnet package under the hood.
Valls, J.M., Aler, R., Galvan, I.M., and Camacho, D. (2021). ``Supervised data transformation and dimensionality reduction with a 3-layer multi-layer perceptron for classification problems". <doi:10.1007/s12652-020-02841-y>
Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) ``Learning representations by back-propagating errors" <doi:10.1038/323533a0>.

Depends R (>= 3.2.4),

Imports nnet, NeuralNetTools, FNN, pracma

Suggests dplyr, knitr, rmarkdown, ggplot2, ggridges, tidyr, forcats, mlr, mlrCPO

License GPL (>= 2)

LazyData true

Encoding UTF-8

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Ricardo Aler [aut, cre],
Jose Valls [aut],
Ines Galvan [aut],
David Camacho [aut]

Maintainer Ricardo Aler <ricardo.aler@uc3m.es>

Repository CRAN

Date/Publication 2021-02-26 09:50:05 UTC

R topics documented:

---

doughnut *Doughnut shaped dataset*

Description

Doughnut-shaped two-class 2-dimensinoal classification problem V1 and V2 are the input features, normalized to 0-1 V3 is the output (TRUE / FALSE)

Usage

```
data(doughnut)
```

Format

An object of class `data.frame` with 10000 rows and 3 columns.

Examples

```
data(doughnut)
plot(doughnut$V1, doughnut$V2, col=doughnut$V3)
```

doughnutRand *Doughnut shaped dataset with 8 extra random attributes*

Description

V1 to V8 are the extra random features (in the range 0-1) V9 and V10 are the original input features, normalized to 0-1 V11 is the output (TRUE / FALSE)

Usage

```
data(doughnutRand)
```

Format

An object of class `data.frame` with 10000 rows and 11 columns.

Examples

```
data(doughnutRand)
plot(doughnutRand$V9, doughnutRand$V10, col=doughnutRand$V11)
```

doughnutRandRotated *Doughnut shaped dataset with 8 extra random attributes and rotated*

Description

doughnutRandRotated randomly rotated

Usage

```
data(doughnutRandRotated)
```

Format

An object of class `data.frame` with 10000 rows and 11 columns.

`nntrf` *nntrf: a supervised transformation function based on neural networks (Neural Net based Transformations)*

Description

This function transforms a dataset into the activations of the neurons of the hidden layer of a neural network. This is done by training a neural network and then computing the activations of the neural network for each input pattern. It uses the **nnet** package under the hood.

Usage

```
nntrf(keep_nn = TRUE, repetitions = 1, random_seed = NULL, ...)
```

Arguments

`keep_nn`	(default TRUE) Keep NN model. In most cases, the actual NN and associated results obtained by nnet is not required
`repetitions`	(default 1) Repeat nnet several times with different random seeds and select the best run using nnet's minimum *value*. This is useful because some random initialization of weights may lead to local minima.
`random_seed`	(default NULL)
`...`	See `nnet` params. Most important: • formula • data :dataframe with data. The last column **must** be the dependent variable. The dependent variable **must** be a factor for classification problems and numeric for regression problems. The input/independent variables **must** contain numeric values only. • size :number of units in the hidden layer. • maxit : the number of iterations of the net.

Value

list of:

- trf: a function that transforms the input dataset using the weights of the hidden layer. This function has three arguments:
  - x: the input numeric **matrix** or **data.frame** to be transformed. Only numeric values.
  - use_sigmoid (default TRUE): Whether the sigmoid function should be used for the transformation. nnet uses the sigmoid in the hidden layer, but in some cases better results could be obtained with use_sigmoid=FALSE.
  - norm (default FALSE): If TRUE, this function's output is normalized (scaled) to range 0-1.
- mod: values returned by nnet (this includes the neural network trained by nnet). If keep_nn=FALSE, then NULL is returned here.
- matrix1: weights of hidden layer
- matrix2: weights of output layer

See Also

nnet

Examples

```
data("doughnutRandRotated")
rd <- doughnutRandRotated

# Make sure it is a classification problem
rd[,ncol(rd)] <- as.factor(rd[,ncol(rd)])
n <- nrow(rd)

# Split data into training and test
set.seed(0)
training_index <- sample(1:n, round(0.6*n))
train <- rd[training_index,]
test <- rd[-training_index,]
x_train <- train[,-ncol(train)]
y_train <- train[,ncol(train)]
x_test <- test[,-ncol(test)]
y_test <- test[,ncol(test)]

# Train nntrf transformation
set.seed(0)
nnpo <- nntrf(formula=V11~.,
              data=train,
              size=4, maxit=50, trace=TRUE)

# Apply nntrf transformation to the train and test splits
trf_x_train <- nnpo$trf(x=x_train, use_sigmoid=FALSE)
trf_x_test <- nnpo$trf(x=x_test, use_sigmoid=FALSE)

# Compute the success rate of KNN on the transformed feature space
outputs <- FNN::knn(trf_x_train, trf_x_test, y_train)
success <- mean(outputs == y_test)
print(success)
```

---

nntrf_doughnut *nntrf_doughnut: a benchmark of three doughnut datasets*

Description

This function compares KNN with data untransformed vs. data transformed with nntrf. In order to see what it does, check the code: nntrf::nntrf_doughnut

Usage

```
nntrf_doughnut(verbose = TRUE)
```

Arguments

verbose	(default TRUE) Print results to the console.

Value

list of:

- Accuracies of KNN on the Doughnut Dataset: with no nntrf, with nntrf and no sigmoid, and with nntrf and no sigmoid and 5 repetitions
- Accuracies of KNN on the DoughnutRand Dataset: with no nntrf, with nntrf and no sigmoid, and with nntrf and no sigmoid and 5 repetitions
- Accuracies of KNN on the DoughnutRandRotated Dataset: with no nntrf, with nntrf and no sigmoid, and with nntrf and no sigmoid and 5 repetitions

---

nntrf_iris *nntrf_iris: a benchmark for the iris dataset*

Description

This function compares KNN with data untransformed vs. data transformed with nntrf. In order to see what it does, check the code: nntrf::nntrf_iris

Usage

```
nntrf_iris(verbose = TRUE)
```

Arguments

verbose	(default TRUE) Print results to the console.

Value

Accuracies of KNN on the Iris Dataset: with no nntrf, and with nntrf and no sigmoid

Index