Package 'snap'

May 9, 2026

Type Package

Title Simple Neural Application

Version 1.1.0

Author Giancarlo Vercellino

Maintainer Giancarlo Vercellino <giancarlo.vercellino@gmail.com>

Description
A simple wrapper to easily design vanilla deep neural networks using 'Tensorflow'/'Keras' backend for regression, classification and multi-label tasks, with some tweaks and tricks (skip shortcuts, embedding, feature selection and anomaly detection).

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 3.6)

Imports keras (>= 2.3.0.0), tensorflow (>= 2.2.0), dplyr (>= 1.0.2), purrr (>= 0.3.4), forcats (>= 0.5.0), tictoc (>= 1.0.1), readr (>= 1.4.0), ggplot2 (>= 3.3.3), CORElearn (>= 1.54.2), dbscan (>= 1.1-5), stringr (>= 1.4.0), reticulate (>= 1.18)

URL https://rpubs.com/giancarlo_vercellino/snap

NeedsCompilation no

Repository CRAN

Date/Publication 2021-06-30 08:30:02 UTC

Contents

friedman3 *friedman3 data set*

Description

Data set to demonstrate regression task.

Usage

```
friedman3
```

Format

A dummy data frame with 5 columns and 150 rows created using Benchmark Problem Friedman 3 by mlbench. The target feature is "y".

Source

mlbench.friedman3(n = 150, sd = 3)

snap *snap*

Description

A simple wrapper to easily design vanilla deep neural networks using 'Tensorflow'/'Keras' backend for regression, classification and multi-label tasks, with some tweaks and tricks (skip shortcuts, embedding, feature selection and anomaly detection).

Usage

```
snap(
  data,
  target,
  task = NULL,
  positive = NULL,
  skip_shortcut = FALSE,
  embedding = "none",
  embedding_size = 10,
  folds = 3,
  reps = 1,
  holdout = 0.3,
  layers = 1,
  activations = "relu",
  regularization_L1 = 0,
  regularization_L2 = 0,
```

```
  nodes = 32,
  dropout = 0,
  span = 0.2,
  min_delta = 0,
  batch_size = 32,
  epochs = 50,
  imp_thresh = 0,
  anom_thresh = 1,
  output_activation = NULL,
  optimizer = "Adam",
  loss = NULL,
  metrics = NULL,
  winsor = FALSE,
  q_min = 0.01,
  q_max = 0.99,
  normalization = TRUE,
  seed = 42,
  verbose = 0
)
```

Arguments

data	A data frame including all the features and targets.
target	String. Single label for target feature when task is "regr" or "classif". String vector with multiple labels for target features when task is "multilabel".
task	String. Inferred by data type of target feature(s). Available options are: "regr", "classif", "multilabel". Default: NULL.
positive	String. Positive class label (only for classification task). Default: NULL.
skip_shortcut	Logical. Option to add a skip shortcut to improve network performance in case of many layers. Default: FALSE.
embedding	String. Available options are: "none", "global" (when identical values for different features hold different meanings), "sequence" (when identical values for different features hold the same meaning). Default: NULL.
embedding_size	Integer. Output dimension for the embedding layer. Default: 10.
folds	Positive integer. Number of folds for repeated cross-validation. Default: 3.
reps	Positive integer. Number of repetitions for repeated cross-validation. Default: 1.
holdout	Positive numeric. Percentage of cases for holdout validation. Default: 0.3.
layers	Positive integer. Number of layers for the neural net. Default: 1.
activations	String. String vector with the activation functions for each layer (for example, a neural net with 3 layers may have activations = c("relu", "gelu", "tanh")). Besides standard Tensorflow/Keras activations, you can also choose: "swish", "mish", "gelu", "bent". Default: "relu".
regularization_L1	Positive numeric. Value for L1 regularization of the loss function. Default: 0.

`regularization_L2`
: Positive numeric. Value for L2 regularization of the loss function. Default: 0.

`nodes`
: Positive integer. Integer vector with the nodes for each layer (for example, a neural net with 3 layers may have nodes = c(32, 64, 16)). Default: 32.

`dropout`
: Positive numeric. Value for the dropout parameter for each layer (for example, a neural net with 3 layers may have dropout = c(0, 0.5, 0.3)). Default: 0.

`span`
: Positive numeric. Percentage of epoch for the patience parameter. Default: 0.2.

`min_delta`
: Positive numeric. Minimum improvement on metric to trigger the early stop. Default: 0.

`batch_size`
: Positive integer. Maximum batch size for training. Default: 32.

`epochs`
: Positive integer. Maximum number of forward and backward propagations. Default: 50.

`imp_thresh`
: Positive numeric. Importance threshold (in percentiles) above which the features are included in the model (using ReliefFbestK metric by COREl earn). Default: 0 (all features included).

`anom_thresh`
: Positive numeric. Anomaly threshold (in percentiles) above which the instances are excluded by the model (using lof by dbscan). Default: 1 (all instances included).

`output_activation`
: String. Default: NULL. If not specified otherwise, it will be "Linear" for regression task, "Softmax" for classification task, "Sigmoid" for multilabel task.

`optimizer`
: String. Standard Tensorflow/Keras Optimization methods are available. Default: "Adam".

`loss`
: Default: NULL. If not specified otherwise, it will be "mean_absolute_error" for regression task, "categorical_crossentropy" for classification task, "binary_crossentropy" for multilabel task.

`metrics`
: Default: NULL. If not specified otherwise, it will be "mean_absolute_error" for regression task, "categorical_crossentropy" for classification task, "binary_crossentropy" for multilabel task.

`winsor`
: Logical. Set to TRUE in case you want to perform Winsorization on regression tasks. Default: FALSE.

`q_min`
: Positive numeric. Minimum quantile threshold for Winsorization. Default: 0.01.

`q_max`
: Positive numeric. Maximum quantile threshold for Winsorization. Default: 0.99.

`normalization`
: Logical. After each layer it performs a batch normalization. Default: TRUE.

`seed`
: Positive integer. Seed value to control random processes. Default: 42.

`verbose`
: Positive integer. Set the level of information from Keras. Default: 0.

Value

This function returns a list including:

- task: kind of task solved

- configuration: main hyper-parameters describing the neural net (layers, activations, regularization_L1, regularization_L2, nodes, dropout)
- model: Keras standard model description
- pred_fun: function to use on the same data scheme to predict new values
- plot: Keras standard history plot
- testing_frame: testing set with the related predictions, including
- trials: statistics for each trial during the repeated cross-validation (train set and validation set):
  - task "classif": balanced accuracy (bac), precision (prc), sensitivity (sen), critical success index (csi), FALSE-score (fsc), Kappa (kpp), Kendall (kdl)
  - task "regr": root mean square error(rmse), mean absolute error (mae), median absolute error (mdae), relative root square error (rrse), relative absolute error (rae), Pearson (prsn)
  - task "multilabel": macro bac, macro prc, macro sensitivity, macro sen, macro csi, macro fsc, micro kpp, micro kdl
- metrics: summary statistics as above for training, validation (both averaged over trials) and testing
- selected_feat: labels of features included within the model
- selected_inst: index of instances included within the model
- time_log

Author(s)

Giancarlo Vercellino <giancarlo.vercellino@gmail.com>

See Also

Useful links:

- https://rpubs.com/giancarlo_vercellino/snap

Examples

```
## Not run:
snap(friedman3, target="y")

snap(threenorm, target="classes", imp_thresh = 0.3, anom_thresh = 0.95)

snap(threenorm, "classes", layers = 2, activations = c("gelu", "swish"), nodes = c(32, 64))

## End(Not run)
```

`threenorm` *threenorm data set*

Description

Data set to demonstrate classification task.

Usage

```
threenorm
```

Format

A dummy data frame with 5 columns and 150 rows created using Threenorm Benchmark Problem by mlbench. The target feature is "classes".

Source

mlbench.threenorm(150, d = 20)

Index