Package 'tabnet'

January 14, 2021

Title Fit 'TabNet' Models for Classification and Regression

Version 0.1.0

Description Implements the 'TabNet' model by Sercan O. Arik et al (2019) <arXiv:1908.07442> and provides a consistent interface for fitting and creating predictions. It's also fully compatible with the 'tidymodels' ecosystem.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports torch, hardhat, magrittr, glue, progress, rlang, methods, tibble, vctrs

Suggests testthat, modeldata, recipes, parsnip, dials, withr, knitr, rmarkdown, vip, tidyverse

VignetteBuilder knitr

NeedsCompilation no

Author Daniel Falbel [aut, cre], RStudio [cph]

Maintainer Daniel Falbel <daniel@rstudio.com>

Repository CRAN

Date/Publication 2021-01-14 09:30:02 UTC

R topics documented:

decision_width *Parameters for the tabnet model*

Description

Parameters for the tabnet model

Usage

```
decision_width(range = c(8L, 64L), trans = NULL)

attention_width(range = c(8L, 64L), trans = NULL)

num_steps(range = c(3L, 10L), trans = NULL)

feature_reusage(range = c(1, 2), trans = NULL)

num_independent(range = c(1L, 5L), trans = NULL)

num_shared(range = c(1L, 5L), trans = NULL)
```

Arguments

range	the default range for the parameter value
trans	wether to apply a transformation to the parameter

These functions are used with tune grid functions to generate candidates.

Value

A dials parameter to be used when tuning TabNet models.

resolve_data *Transforms input data into tensors*

Description

Transforms input data into tensors

Usage

```
resolve_data(x, y)
```

Arguments

x	a data frame
y	a response vector

tabnet *Parsnip compatible tabnet model*

Description

Parsnip compatible tabnet model

Usage

```
tabnet(
  mode = "unknown",
  epochs = NULL,
  penalty = NULL,
  batch_size = NULL,
  learn_rate = NULL,
  decision_width = NULL,
  attention_width = NULL,
  num_steps = NULL,
  feature_reusage = NULL,
  virtual_batch_size = NULL,
  num_independent = NULL,
  num_shared = NULL,
  momentum = NULL
)
```

Arguments

mode	A single character string for the type of model. Possible values for this model are "unknown", "regression", or "classification".
epochs	(int) Number of training epochs.
penalty	This is the extra sparsity loss coefficient as proposed in the original paper. The bigger this coefficient is, the sparser your model will be in terms of feature selection. Depending on the difficulty of your problem, reducing this value could help.
batch_size	(int) Number of examples per batch, large batch sizes are recommended. (default: 1024)
learn_rate	initial learning rate for the optimizer.
decision_width	(int) Width of the decision prediction layer. Bigger values gives more capacity to the model with the risk of overfitting. Values typically range from 8 to 64.
attention_width	(int) Width of the attention embedding for each mask. According to the paper n_d=n_a is usually a good choice. (default=8)
num_steps	(int) Number of steps in the architecture (usually between 3 and 10)

`feature_reusage`
: (float) This is the coefficient for feature reusage in the masks. A value close to 1 will make mask selection least correlated between layers. Values range from 1.0 to 2.0.

`virtual_batch_size`
: (int) Size of the mini batches used for "Ghost Batch Normalization" (default=128)

`num_independent`
: Number of independent Gated Linear Units layers at each step. Usual values range from 1 to 5.

`num_shared`
: Number of shared Gated Linear Units at each step Usual values range from 1 to 5

`momentum`
: Momentum for batch normalization, typically ranges from 0.01 to 0.4 (default=0.02)

Value

A TabNet `parsnip` instance. It can be used to fit tabnet models using `parsnip` machinery.

Threading

TabNet uses `torch` as it's backend for computation and `torch` uses all available threads by default.

You can control the number of threads used by `torch` with:

```
torch::torch_set_num_threads(1)
torch::torch_set_num_interop_threads(1)
```

See Also

tabnet_fit

Examples

```
if (torch::torch_is_installed()) {
library(parsnip)
data("ames", package = "modeldata")
model <- tabnet() %>%
  set_mode("regression") %>%
  set_engine("torch")
model %>%
  fit(Sale_Price ~ ., data = ames)
}
```

tabnet_config *Configuration for TabNet models*

Description

Configuration for TabNet models

Usage

```
tabnet_config(
  batch_size = 256,
  penalty = 0.001,
  clip_value = NULL,
  loss = "auto",
  epochs = 5,
  drop_last = FALSE,
  decision_width = NULL,
  attention_width = NULL,
  num_steps = 3,
  feature_reusage = 1.3,
  virtual_batch_size = 128,
  valid_split = 0,
  learn_rate = 0.02,
  optimizer = "adam",
  lr_scheduler = NULL,
  lr_decay = 0.1,
  step_size = 30,
  checkpoint_epochs = 10,
  cat_emb_dim = 1,
  num_independent = 2,
  num_shared = 2,
  momentum = 0.02,
  verbose = FALSE,
  device = "auto"
)
```

Arguments

batch_size	(int) Number of examples per batch, large batch sizes are recommended. (default: 1024)
penalty	This is the extra sparsity loss coefficient as proposed in the original paper. The bigger this coefficient is, the sparser your model will be in terms of feature selection. Depending on the difficulty of your problem, reducing this value could help.
clip_value	If a float is given this will clip the gradient at clip_value. Pass NULL to not clip.

loss	(character or function) Loss function for training (default to mse for regression and cross entropy for classification)
epochs	(int) Number of training epochs.
drop_last	(bool) Whether to drop last batch if not complete during training
decision_width	(int) Width of the decision prediction layer. Bigger values gives more capacity to the model with the risk of overfitting. Values typically range from 8 to 64.
attention_width	(int) Width of the attention embedding for each mask. According to the paper n_d=n_a is usually a good choice. (default=8)
num_steps	(int) Number of steps in the architecture (usually between 3 and 10)
feature_reusage	(float) This is the coefficient for feature reusage in the masks. A value close to 1 will make mask selection least correlated between layers. Values range from 1.0 to 2.0.
virtual_batch_size	(int) Size of the mini batches used for "Ghost Batch Normalization" (default=128)
valid_split	(float) The fraction of the dataset used for validation.
learn_rate	initial learning rate for the optimizer.
optimizer	the optimization method. currently only 'adam' is supported, you can also pass any torch optimizer function.
lr_scheduler	if NULL, no learning rate decay is used. if "step" decays the learning rate by lr_decay every step_size epochs. It can also be a torch::lr_scheduler function that only takes the optimizer as parameter. The step method is called once per epoch.
lr_decay	multiplies the initial learning rate by lr_decay every step_size epochs. Unused if lr_scheduler is a torch::lr_scheduler or NULL.
step_size	the learning rate scheduler step size. Unused if lr_scheduler is a torch::lr_scheduler or NULL.
checkpoint_epochs	checkpoint model weights and architecture every checkpoint_epochs. (default is 10). This may cause large memory usage. Use 0 to disable checkpoints.
cat_emb_dim	Embedding size for categorial features (default=1)
num_independent	Number of independent Gated Linear Units layers at each step. Usual values range from 1 to 5.
num_shared	Number of shared Gated Linear Units at each step Usual values range from 1 to 5
momentum	Momentum for batch normalization, typically ranges from 0.01 to 0.4 (default=0.02)
verbose	(bool) wether to print progress and loss values during training.
device	the device to use for training. "cpu" or "cuda". The default ("auto") uses to "cuda" if it's available, otherwise uses "cpu".

Value

A named list with all hyperparameters of the TabNet implementation.

tabnet_explain *Interpretation metrics from a TabNet model*

Description

Interpretation metrics from a TabNet model

Usage

```
tabnet_explain(object, new_data)
```

Arguments

object	a TabNet fit object
new_data	a data.frame to obtain interpretation metrics.

Value

Returns a list with

- M_explain: the aggregated feature importance masks as detailed in TabNet's paper.
- masks a list containing the masks for each step.

Examples

```
if (torch::torch_is_installed()) {

set.seed(2021)

n <- 1000
x <- data.frame(
  x = runif(n),
  y = runif(n),
  z = runif(n)
)

y <- x$x

fit <- tabnet_fit(x, y, epochs = 20,
                  num_steps = 1,
                  batch_size = 512,
                  attention_width = 1,
                  num_shared = 1,
                  num_independent = 1)


 ex <- tabnet_explain(fit, x)
```

```
}
```

`tabnet_fit` *Tabnet model*

Description

Fits the TabNet: Attentive Interpretable Tabular Learning model

Usage

```
tabnet_fit(x, ...)

## Default S3 method:
tabnet_fit(x, ...)

## S3 method for class 'data.frame'
tabnet_fit(x, y, ...)

## S3 method for class 'formula'
tabnet_fit(formula, data, ...)

## S3 method for class 'recipe'
tabnet_fit(x, data, ...)
```

Arguments

`x`	Depending on the context: • A **data frame** of predictors. • A **matrix** of predictors. • A **recipe** specifying a set of preprocessing steps created from `recipes::recipe()`. The predictor data should be standardized (e.g. centered or scaled). The model treats categorical predictors internally thus, you don't need to make any treatment.
`...`	Model hyperparameters. See `tabnet_config()` for a list of all possible hyperparameters.
`y`	When `x` is a **data frame** or **matrix**, `y` is the outcome specified as: • A **data frame** with 1 numeric column. • A **matrix** with 1 numeric column. • A numeric **vector**.
`formula`	A formula specifying the outcome terms on the left-hand side, and the predictor terms on the right-hand side.
`data`	When a **recipe** or **formula** is used, `data` is specified as: • A **data frame** containing both the predictors and the outcome.

Value

A TabNet model object. It can be used for serialization and predictions.

Threading

TabNet uses `torch` as it's backend for computation and `torch` uses all available threads by default.

You can control the number of threads used by `torch` with:

```
torch::torch_set_num_threads(1)
torch::torch_set_num_interop_threads(1)
```

Examples

```
if (torch::torch_is_installed()) {
data("ames", package = "modeldata")
fit <- tabnet_fit(Sale_Price ~ ., data = ames, epochs = 1)
}
```

Index