Package 'kerasformula'

August 23, 2018

Type Package

Title A High-Level R Interface for Neural Nets

Version 1.5.1

Author Pete Mohanty [aut, cre]

Maintainer Pete Mohanty <pete.mohanty@gmail.com>

Description Adds a high-level interface for 'keras' neural nets. kms() fits neural net and accepts R formulas to aid data munging and hyperparameter selection. kms() can optionally accept a compiled keras_sequential_model() from 'keras'.
kms() accepts a number of parameters (like loss and optimizer) and splits the data into (optionally sparse) test and training matrices. kms() facilitates setting advanced hyperparameters (e.g., regularization). kms() returns a single object with predictions, a confusion matrix, and function call details.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

URL https://github.com/rdrr1990/kerasformula

BugReports https://github.com/rdrr1990/kerasformula/issues

Depends keras, dplyr, Matrix

Imports ggplot2

Suggests tensorflow, knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-08-23 17:52:14 UTC

R topics documented:

`confusion` *confusion*

Description

Confusion matrix or (for larger number of levels) confusion table.

Usage

```
confusion(object = NULL, y_test = NULL, predictions = NULL,
  return_xtab = NULL, digits = 3)
```

Arguments

`object`	Optional fit object. confusion() assumes object contains holdout/vaidation data as 'y_test' and the forecasts/classifications as 'predictions' but alternative variable names can be specified with the input arguments by those names.
`y_test`	A vector of holdout/validation data or the name in object (if fit object provided but alternative variable name required).
`predictions`	A vector predictions or the name in object (if fit object provided but alternative variable name required).
`return_xtab`	Logical. If TRUE, returns confusion matrix, which is a crosstable with correct predictions on the diagonal (if all levels are predicted at least once). If FALSE, returns data.frame with columns for percent correct, most common misclassification, second most common misclassification, and other predictions. Only defaults to crosstable-style if y_test has fewer than six levels.
`digits`	Number of digits for proportions when return_xtab=FALSE; if NULL, no rounding is performed.

Value

confusion matrix or table as specified by return_xtab.

Examples

```
mtcars$make <- unlist(lapply(strsplit(rownames(mtcars), " "), function(tokens) tokens[1]))
company <- if(is_keras_available()){
              kms(make ~ ., mtcars, Nepochs=1, verbose=0)
           }else{
                 list(y_test = mtcars$make[1:5],
                 predictions = sample(mtcars$make, 5))
                 }
confusion(company)     # same as above confusion$company if is_keras_available() == TRUE
confusion(company, return_xtab = FALSE) # focus on pCorrect, most common errors
```

---

kms *kms*

Description

A regression-style function call for keras_model_sequential() which uses formulas and, optionally, sparse matrices. A sequential model is a linear stack of layers.

Usage

```
kms(input_formula, data, keras_model_seq = NULL, N_layers = 3,
  units = c(256, 128), activation = c("relu", "relu", "softmax"),
  dropout = 0.4, use_bias = TRUE, kernel_initializer = NULL,
  kernel_regularizer = "regularizer_l1",
  bias_regularizer = "regularizer_l1",
  activity_regularizer = "regularizer_l1", embedding = FALSE,
  pTraining = 0.8, validation_split = 0.2, Nepochs = 15,
  batch_size = NULL, loss = NULL, metrics = NULL,
  optimizer = "optimizer_adam", scale_continuous = "zero_one",
  drop_intercept = TRUE, sparse_data = FALSE, seed = list(seed = NULL,
  disable_gpu = FALSE, disable_parallel_cpu = FALSE), verbose = 1, ...)
```

Arguments

input_formula	an object of class "formula" (or one coerceable to a formula): a symbolic description of the keras inputs. "mpg ~ cylinders". kms treats numeric data with more than two distinct values a continuous outcome for which a regression-style model is fit. Factors and character variables are classified; to force classification, "as.factor(cyl) ~ .".
data	a data.frame.
keras_model_seq	A compiled Keras sequential model. If non-NULL (NULL is the default), then bypasses the following ‘kms‘ parameters: N_layers, units, activation, dropout, use_bias, kernel_initializer, kernel_regularizer, bias_regularizer, activity_regularizer, loss, metrics, and optimizer.

N_layers How many layers in the model? Default == 3. Subsequent parameters (units, activation, dropout, use_bias, kernel_initializer, kernel_regularizer, bias_regularizer, and activity_regularizer) may be inputted as vectors that are of length N_layers (or N_layers - 1 for units and dropout). The length of those vectors may also be length 1 or a multiple of N_layers (or N_layers - 1 for units and dropout).

units How many units in each layer? The final number of units will be added based on whether regression or classification is being done. Should be length 1, length N_layers - 1, or something that can be repeated to form a length N_layers - 1 vector. Default is c(256, 128).

activation Activation function for each layer, starting with the input. Default: c("relu", "relu", "softmax"). Should be length 1, length N_layers, or something that can be repeated to form a length N_layers vector.

dropout Dropout rate for each layer, starting with the input. Not applicable to final layer. Default: c(0.4, 0.3). Should be length 1, length N_layers - 1, or something that can be repeated to form a length N_layers - 1 vector.

use_bias See ?keras::use_bias. Default: TRUE. Should be length 1, length N_layers, or something that can be repeated to form a length N_layers vector.

kernel_initializer
Defaults to "glorot_uniform" for classification and "glorot_normal" for regression (but either can be inputted). Should be length 1, length N_layers, or something that can be repeated to form a length N_layers vector.

kernel_regularizer
Must be precisely either "regularizer_l1", "regularizer_l2", or "regulizer_l1_l2". Default: "regularizer_l1". Should be length 1, length N_layers, or something that can be repeated to form a length N_layers vector.

bias_regularizer
Must be precisely either "regularizer_l1", "regularizer_l2", or "regulizer_l1_l2". Default: "regularizer_l1". Should be length 1, length N_layers, or something that can be repeated to form a length N_layers vector.

activity_regularizer
Must be precisely either "regularizer_l1", "regularizer_l2", or "regulizer_l1_l2". Default: "regularizer_l1". Should be length 1, length N_layers, or something that can be repeated to form a length N_layers vector.

embedding If TRUE, the first layer will be an embedding with the number of output dimensions determined by ‘units‘ (so to speak, that means there will really be N_layers + 1). Note input ‘kernel_regularizer‘ is passed on as the ‘embedding_regularizer‘. Note pad_sequences() may be used as part of the input_formula and you may wish to set scale_continuous to NULL. See ?layer_embedding.

pTraining Proportion of the data to be used for training the model; 0 =< pTraining < 1. By default, pTraining == 0.8. Other observations used only postestimation (e.g., confusion matrix).

validation_split
Portion of data to be used for validating each epoch (i.e., portion of pTraining). To be passed to keras::fit. Default == 0.2.

Nepochs Number of epochs; default == 15. To be passed to keras::fit.

`batch_size` Default batch size is 32 unless emedding == TRUE in which case batch size is 1. (Smaller eases memory issues but may affect ability of optimizer to find global minimum). To be passed to several functions library(keras) functions like fit(), predict_classes(), and layer_embedding(). If embedding==TRUE, number of training obs must be a multiple of batch size.

`loss` To be passed to keras::compile. Defaults to "binary_crossentropy", "categorical_crossentropy", or "mean_squared_error" based on input_formula and data.

`metrics` Additional metric(s) beyond the loss function to be passed to keras::compile. Defaults to "mean_absolute_error" and "mean_absolute_percentage_error" for continuous and c("accuracy") for binary/categorical (as well whether whether examples are correctly classified in one of the top five most popular categories or not if the number of categories K > 20).

`optimizer` To be passed to keras::compile. Defaults to "optimizer_adam", an algorithm for first-order gradient-based optimization of stochastic objective functions introduced by Kingma and Ba (2015) here: https://arxiv.org/pdf/1412.6980v8.pdf.

`scale_continuous` Function to scale each non-binary column of the training data (and, if y is continuous, the outcome). The default 'scale_continuous = zero_one' places each non-binary column of the training model matrix on [0, 1]; 'scale_continuous = z' standardizes; 'scale_continuous = NULL' leaves the data on its original scale.

`drop_intercept` TRUE by default.

`sparse_data` Default == FALSE. If TRUE, X is constructed by sparse.model.matrix() instead of model.matrix(). Recommended to improve memory usage if there are a large number of categorical variables or a few categorical variables with a large number of levels. May compromise speed, particularly if X is mostly numeric.

`seed` Integer or list containing seed to be passed to the sources of variation: R, Python's Numpy, and Tensorflow. If seed is NULL, automatically generated. Note setting seed ensures data will be partitioned in the same way but to ensure identical results, set disable_gpu = TRUE and disable_parallel_cpu = TRUE. Wrapper for use_session_with_seed(), which is to be called before compiling by the user if a compiled Keras model is passed into kms. See also see https://stackoverflow.com/questions/42022950/.

`verbose` Default == 1. Setting to 0 disables progress bar and epoch-by-epoch plots (disabling them is recommended for knitting RMarkdowns if X11 not installed).

`...` Additional parameters to be passsed to Matrix::sparse.model.matrix.

Value

kms_fit object. A list containing model, predictions, evaluations, as well as other details like how the data were split into testing and training. To extract or save weights, see https://tensorflow.rstudio.com/keras/reference/save_m

Author(s)

Pete Mohanty

Examples

```
if(is_keras_available()){

 mtcars$make <- unlist(lapply(strsplit(rownames(mtcars), " "), function(tokens) tokens[1]))
 company <- kms(make ~ ., mtcars, Nepochs = 1, verbose=0)
 # out of sample accuracy
 pCorrect <- mean(company$y_test == company$predictions)
 pCorrect
 company$confusion
 # plot(history$company) # helps pick Nepochs
 # below
 # find the default settings for layers
 company <- kms(make ~ ., mtcars,
                units = c(256, 128),
                activation = c("relu", "relu", "softmax"),
                dropout = 0.4,
                use_bias = TRUE,
                kernel_initializer = NULL,
                kernel_regularizer = "regularizer_l1",
                bias_regularizer = "regularizer_l1",
                activity_regularizer = "regularizer_l1",
                Nepochs = 1, verbose=0
                )
 # ?predict.kms_fit to see how to predict on newdata
}else{
  cat("Please run install_keras() before using kms(). ?install_keras for options like gpu.")
}
```

---

plot_confusion *plot_confusion*

Description

plot_confusion

Usage

```
plot_confusion(..., display = TRUE, return_ggplot = FALSE, title = "",
  subtitle = "")
```

Arguments

...	kms_fit objects. (For each, object$y_test must be binary or categorical.)
display	Logical: display ggplot comparing confusion matrices? (Default TRUE.)
return_ggplot	Default FALSE (if TRUE, returns the ggplot object for further customization, etc.).
title	ggplot title
subtitle	ggplot subtitle

Value

(optional) ggplot. set return_ggplot=TRUE

Examples

```
if(is_keras_available()){

   model_tanh <- kms(Species ~ ., iris,
                     activation = "tanh", Nepochs=5,
                     units=4, seed=1, verbose=0)
   model_softmax <- kms(Species ~ ., iris,
                        activation = "softmax", Nepochs=5,
                        units=4, seed=1, verbose=0)
   model_relu <- kms(Species ~ ., iris,
                     activation = "relu", Nepochs=5,
                     units=4, seed=1, verbose=0)

   plot_confusion(model_tanh, model_softmax, model_relu,
                  title="Species",
                  subtitle="Activation Function Comparison")

}
```

---

predict.kms_fit *predict.kms_fit*

Description

predict function for kms_fit object. Places test data on same scale that the training data were by kms(). Wrapper for keras::predict_classes(). Creates a sparse model matrix with the same columns as the training data, some of which may be 0.

Usage

```
## S3 method for class 'kms_fit'
predict(object, newdata, batch_size = 32, verbose = 0,
  ...)
```

Arguments

object	output from kms()
newdata	new data. Performs merge so that X_test has the same columns as the object created by kms_fit using the user-provided input formula. y_test is also generated from that formula.
batch_size	To be passed to keras::predict_classes. Default == 32.
verbose	0 ot 1, to be passed to keras::predict_classes. Default == 0.
...	additional parameters to build the sparse matrix X_test.

Value

list containing predictions, y_test, confusion matrix.

Author(s)

Pete Mohanty

Examples

```
if(is_keras_available()){

 mtcars$make <- unlist(lapply(strsplit(rownames(mtcars), " "), function(tokens) tokens[1]))
 company <- kms(make ~ ., mtcars[3:32, ], Nepochs = 2, verbose=0)
 forecast <- predict(company, mtcars[1:2, ])
 forecast$confusion

 # example where y_test is unavailable

 trained <- kms(log(mpg) ~ ., mtcars[4:32,], Nepochs=1, verbose=0)
 X_test <- subset(mtcars[1:3,], select = -mpg)
 predictions <- predict(trained, X_test)

}else{
  cat("Please run install_keras() before using kms(). ?install_keras for options like gpu.")
}
```

Index