README.md

Install-Keras-R-Ubuntu

1st step: Install devtools

Devtools is necesary because it allow us to install and packages from GitHub.

- Install system dependencies for devtools (in console):

  sudo apt-get install build-essential libcurl4-gnutls-dev libxml2-dev libssl-dev

- Install devtools package (in R):

  install.packages('devtools')

2nd step: Install Keras

- Install keras from github repository (in R):

  devtools::install_github("rstudio/keras")

- To make sure Keras is installed (in R):

  packageVersion("keras")

3rd step: Install TensorFlow

- Install TensorFlow (in R):

  install_tensorflow() #for cpu#

  install_tensorflow(gpu = T) #for nvdidia gpu#

- To make sure TensorFlow is installed (in R):

  packageVersion("tensorflow")

2nd-3rd in one step: Install Keras and TensorFlow simultaneously:

- Install keras from github repository (in R):

  devtools::install_github("rstudio/keras")

- Install system dependencies for TensorFlow (in console):

  sudo apt-get install python-pip python-virtualenv

- Install Keras and TensorFlow (in R):

  install_keras()

References

https://keras.rstudio.com/

https://medium.com/towards-data-science/how-to-implement-deep-learning-in-r-using-keras-and-tensorflow-82d135ae4889

https://www.digitalocean.com/community/tutorials/how-to-install-r-packages-using-devtools-on-ubuntu-16-04

https://tensorflow.rstudio.com/installation_gpu.html

Functions-Keras-R

Load the library

library(keras)

Check versions

packageVersion("keras")

packageVersion("tensorflow")

Create the model

- The main data structure in Keras is a model, a way to organize a linear stack of layers.

  model <- keras_model_sequential()

Add layers to the model (using the pipe (%>%) operator)

- Fully connected layers

  model %>% layer_dense(units, activation, input_shape)

  - units: numbers of neurons in the first hidden layer
  - activation: activation function ('tanh', 'relu', 'linear', 'softmax' ...)
  - input_shape: number of neurons in the input layer (the first layer in a sequential model (and only the first) needs to receive information about its input shape

- Long Short Term Memory

  model %>% layer_lstm(units, activation, input_shape or batch_input_shape, return_sequences, stateful)

  - units: numbers of lstm neurons in the first hidden layer
  - activation: activation function ('tanh', 'relu', 'linear', 'softmax' ...)
  - input_shape: dimensionality of the input -> c(timestep (number of time steps per inputs), features (number of columns))
  - batch_imput_shape: shape of the data -> c(batch_size (normally the number os samples), timestep (number of time steps per inputs), features (number of columns))
  - return_sequences: true or false.
  - stateful: true or false. The states computed for the samples in one batch will be reused as initial states for the samples in the next batch. Stateful to true needs a fixed batch size for your model (with batch_input_shape), and shuffle = False in fit().

Dropout

- The dropout rate for regularization, is an effort to limit overfitting and improve the model's ability to generalize.

  model %>% layer_dropout(rate)

  - rate: fraction of the input units to drop (between 0 and 1)

Print the details of the model

summary(model)

Compile the model

- Configure a Keras model for training

  model %>% compile(loss, optimizer, metrics)

  - loss: objective function ('mean_squared_error', 'binary_crossentropy', ...)
  - optimizer: optimizer for estimating and updating the model parameters ('sgd', 'rmsprop', ...)
  - metrics: the metric to assess the performance of the model ('accuracy', ...) (for classification problem)

Fit the model

- Function to train the model

  model %>% fit(X_train, Y_train, epochs, batch_size, shuffle)

  - X_train: explicative variable/variables for training
  - Y_train: explicated variable for training
  - epochs: the number of times the algorithm work with the entire training data
  - batch_size: the size of sample to be passed through the algorithm in each epoch (32 by default)
  - shuffle: true or false. Shuffle the training data before each epoch.

Plot the training phase

- The Keras fit() method returns an R object containing the training history, including the value of metrics at the end of each epoch

  plot(Fit_Return)

  - Fit_Return: the object returned by fit() function

Predict with the model

- Generate predictions on new data (or on train and test data)(for regression)

  model %>% predict(X_data)

  - X_data: explicative data for training to predict the data train, or explicative data for test to predict the test data

- Generate predictions on new data (or on train and test data)(for classification)

  model %>% predict_classes(X_data)

  - X_data: explicative data for training to predict the data train, or explicative data for test to predict the test data

Evaluate the model

- Evaluate the model's performance on the training and test data

  model %>% evaluate(X_data, Y_data)

  - X_train: explicative data for training to evaluate the training, or explicative data for test to predict the testing
  - Y_train: explicated data for training to evaluate the training, or explicated data for test to predict the testing

Help

help(package = keras)

References

https://keras.rstudio.com/

https://www.linkedin.com/pulse/finally-deep-learning-keras-tensorflow-r-richard-wanjohi-ph-d/

Releases

No releases published

Packages

No packages published