Installation Guide: Python, Reticulate and Keras

2021-07-28

We provide these instructions to help users with the installation of `python` , and the `reticulate` and `keras` packages used in the labs for the *Deep Learning Chapter* of **An Introduction to Statistical Learning, with Applications in R, Second Edition**. We thank Balasubramanian Narasimhan for creating this vignette and the code that supports it, and for guiding us through this process.

These instructions were confirmed to work on Mac and Windows machines as of July 6, 2021. We will update them from time to time as is needed.

1. Introduction

The R package `reticulate` allows you to call python from R.

Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. It was primarily written by Francois Chollet (Google) in python.

The R package `keras` allows one to interface with the python Keras API and is authored by J.J. Allaire (RStudio). Key features of Keras are:

- Allows the same code to run on CPU or on GPU, seamlessly.
- User-friendly API which makes it easy to quickly prototype deep learning models.
- Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.
- Allows for use of multiple backends `Tensorflow` , `Theano` , `CNTK` , etc., although we will only focus on tensorflow.

Obviously, a python installation is a requirement. One can try using an existing python installation using our instructions below. However, we find that a degree of daredevilry is an asset: when something doesn't work, it is easiest to tear down the python environment and build it again.

1.1. Brittle Junctions

The following details are useful to know, but if you are impatient to get started, just skip to Section 2 (Installation Overview).

1. The glue that lets R and python work together is the `reticulate` package.
2. The R `tensorflow` package, which is necessary to use `keras` is essentially an interface to the python `tensorflow` implementation: calls in R are translated into appropriate calls to the python `tensorflow` library. Results and attributes of python objects are exposed appropriately on the R side taking care to avoid make unnecessary copies of objects.
3. The R `keras` package is also an interface to the python `keras` package. Lately, however, the `keras` package in python is bundled together with `tensorflow` .

4. New architectures such as Apple M1 macs require installation of a binary from the Apple channel.

For all these parties to work together, some harmony has to reign between the exposed and expected interfaces. While functions such as `tensorflow::install_tensorflow()` and `keras::install_keras()` are good attempts at ensuring this matrimonial harmony, they fail for many reasons.

- The changes to operating systems over which one has no control.
- Multiple python installations on a machine. The use of python *virtual environments*/*conda environments* surely help but still problems can arise if the R packages are unable to locate such environments or the environment itself gets corrupted.
- Asynchronous updates to python and R packages. While the R packages try to keep up with the python updates, incompatibilities often creep in over time.
- Platform-specific builds of `tensorflow`. For example, Apple provides specific builds for both M1 machines. So does Anaconda, but the repositories are sometimes not updated with the links to the latest versions of the packages.

2. Installation Overview

We assume:

- you have no shortage of disk space,
- none of the directory paths in what follows contain spaces. (The specific case of a **Windows home directory** containing spaces **must** be corrected, and is addressed in section 4.1)

The general installation procedure can be described as follows.

1. Dissociate R with any previous installations of `miniconda` by removing previously installed versions of `reticulate`, `tensorflow` and `keras`. (This enables us to provide a clean set of instructions without having to account for the rather large number of possibilities!)
2. Install the R package `keras` anew.
3. Install specific `python` implementations of `tensorflow` customized for Apple Macs (M1 or Intel), Linux or Windows, downloaded from appropriate websites so that R can hand off tasks to them.

The `ISLR2` package provides helper functions that automate the installation and get used below.

3. Installation Steps

0. Install the most recent version of the `ISLR2` package from `CRAN`.
1. Remove any pre-existing installations of the R packages `keras`, `reticulate` and `tensorflow`. This is just so that we begin with a clean slate.

```
tryCatch(
  remove.packages(c("keras", "tensorflow", "reticulate")),
  error = function(e) "Some or all packages not previously installed, that's ok!"
)
```

2. Install the latest `keras` package from CRAN. This will also install dependencies `tensorflow` and `reticulate`.

```
install.packages("keras", repos = 'https://cloud.r-project.org')
```

3. Decide where you would like to install the miniconda installation of python. You may already have an installation of `miniconda`, in which case choose a different directory path, with not too long a name, though. For example, a reasonable choice is `islr-miniconda` in your home directory. (Remember: we assume **no spaces** in directory names; see Section 4.1). Once you have decided on the path, we let R know this fact, in addition to preventing `reticulate` from making wrong guesses. Execute the following code, changing the `islr-miniconda` as appropriate. (i.e. if you follow our example and choose to use the name `islr-miniconda`, nothing to be changed; otherwise substitute your choice in place of `islr-miniconda` in the second `write` string below, and and the next step as well.)

   ```
   write('RETICULATE_AUTOCONFIGURE=FALSE', file = "~/.Renviron", append = TRUE)
   write(sprintf('RETICULATE_MINICONDA_PATH=%s',
                 normalizePath("~/islr-miniconda", winslash = "/", mustWork = FALSE)),
      file = "~/.Renviron", append = TRUE)
   ```

   (Although the `~/` hides your full home directory path, this string expands to it, and if it has spaces, there can be problems. See Section 4.1)

4. The specification in step 3 will become effective only when R is started anew. So either restart R and jump to step 5, or make it effective in the current session via

```
Sys.setenv(RETICULATE_AUTOCONFIGURE='FALSE',
           RETICULATE_MINICONDA_PATH=normalizePath("~/islr-miniconda", winslash = "/", mustWork
 = FALSE))
```

(Note the `islr-miniconda` string again.)

5. In this step, we will use the helper functions in the `ISLR2` package, so we need to source it.

```
source(system.file("helpers", "install.R", package = "ISLR2"))
```

Two functions `install_miniconda()` and `install_tensorflow()` do the brunt of the work. So you can invoke them in the following order:

```
install_miniconda()
```

followed by

```
install_tensorflow()
```

For the impatient, another function `install_miniconda_and_tensorflow()` is provided that invokes those two in the above order.

You will see a lot of action on your screen as this process proceeds. This is standard, so be patient, because it can take several minutes.

6. If all goes well, you may wish to print out the full configuration:

```
print_py_config()
```

This function is useful when you want to explain your configuration to anyone else or provide debugging information in any correspondence with the package authors.

4. Notes

1. Warnings and informational messages from `tensorflow` and `keras` are common. Some lab exercises may emit warnings of the form:

```
"Method on_batch_end() is slow compared to the batch update (factor_of_slowness)"
```

where the `factor_of_slowness` is a number like `5.233` or something. This is a note that the validation step has taken 5.233 times the time of the training epoch. When this happens, it is important to see if the results are still reasonable before relying on them.

Other deprecation warnings may also occur, e.g.,

```
Warning: The `lr` argument is deprecated, use `learning_rate` instead.
```

or

```
Warning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01.
```

These can be ignored.

4.1. Spaces in directory paths

Avoid them. This is particularly annoying on Windows. If your home directory path itself contains spaces, follow the solution below *before* running through the installation steps in Section 3 above. (Although the `~/` hides your full home directory path in steps 3. and 4. of Section 3, this string expands to it, and if it has spaces, there can be problems.)

Suppose your home directory is `C:\Users\Joe Blow`. We will show you how to create a *link* (think alias) `jblow` that can be used instead of `Joe Blow`, without actually disturbing the original name. Of course `jblow` has no spaces. (Remember to substitute `Joe Blow` and `jblow` to accomodate your own *spacy* name in what follows.)

Create a *link* to your home directory *without* spaces. In a command window *with administrative privileges*, create a link as follows:

```
cd C:\Users
mklink /D jblow "Joe Blow"
```

This is essentially an alias `C:\Users\jblow` (without spaces) that points to the your home directory. (Alternative solutions are possible too, such as creating a directory at the top level, e.g. `C:\miniconda3` but we leave such experimentation to you.)

You can then use the new link, in this case when specifying a directory path for miniconda to use in steps 3. and 4. of Section 3.

In this case we would replace the string

```
normalizePath("~/islr-miniconda", winslash = "/", mustWork = FALSE)
```

used in items 3. and 4. of Section 3 with

```
normalizePath("C:/Users/jblow/islr-miniconda", winslash = "/", mustWork = FALSE)
```

4.2. Precise configuration details

Included in the helper functions is a function `print_py_config()` that will print complete details on your `reticulate`, `tensorflow` configuration. This is useful when you want to explain your configuration to anyone else or provide debugging information in any correspondence.