Installing and Using TensorFlow in R/RStudio to conduct classification

Marvine Hamner

It is very important to read through all these instructions carefully before attempting to install TensorFlow and completing this optional, additional assignment – VERY IMPORTANT!

Go through the instructions to install TensorFlow but keep in mind the tips I've written below. Unfortunately, installing and using TensorFlow in R/RStudio is not as straight forward as is written online!

Once you've got TensorFlow installed in the R/RStudio environment then follow the directions for classification of the "MNIST" dataset at https://tensorflow.rstudio.com/guide/keras/ to check and make sure everything is working properly. If it is you should be able to use the commands written on this webpage exactly as they are written. If you try a command and start getting errors then something did not install completely or correctly.

If you have followed the commands for the "MNIST" classification and gotten close to the answers recorded online, then the additional exercise for you to follow is to complete the "Fashion MNIST" classification at https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/tutorial_basic_classification/. As you complete this exercise if you use RMarkdown or a Jupyter notebook it will be much easier for you to submit your work when you are done training and running your code!

Good Luck!!!!!!!

1. Install TensorFlow. Use the instructions at: https://tensorflow.rstudio.com/installation/ to help you do this. Unfortunately, unless this is the first installation of any of these programs you will have to be very, very lucky for this to work. What follows are some tips and suggestions to help you get the installation completed.
   - First, install Anaconda. Anaconda is a "platform" that encompasses many programs that are used in data science. Overall, using Anaconda to get and install these programs is much easier than doing each one individually. These programs have been written in any of a variety of programming languages, e.g. TensorFlow written in Python.
     - You will need to make sure that Anaconda's installation is complete, likely including Miniconda, in order to get access to a version of Python through Anaconda. I'll say more about that a bit later.
   - Second, install a compatible version of R from https://www.r-project.org/. Today, August 29, 2021, the best, compatible and stable version is R-4.1.1. That means everything else must also be compatible with this version of R, R-4.1.1. Since I've used R for a long time I've got many versions of R.

R is open source meaning that people all over the world are contributing packages to R, all the time. This means that a package coming online today probably won't be backward compatible with R-3.4.1, if you are still running that version. Unfortunately, there are often very good, useful packages that are deprecated so that they are no longer compatible with versions of R past some specific point. Just make sure that everything is operating in a compatible version to get started. In the future you can update whichever packages you want to switch things around.

- Third, install a compatible version of RStudio. You can also do this through Anaconda. Today, that version is RStudio 1.1.456. Typically, RStudio is backward compatible with various versions of R. To change the version of R you are using in RStudio go to "Global Options" in the "Tools" menu.
- Fourth, go into RStudio and install the R package "devtools". You can do that through the Tools drop down menu and "Install Packages".
- Fifth, you need to install the compatible version of Rtools which, at the time of this writing, is Rtools4 from https://cran.r-project.org/bin/windows/Rtools/. To install this compatible version of Rtools you will have to download the executable file from this webpage and execute it. Unfortunately, you cannot install this from inside RStudio.

You also have to ensure that the correct path to this is in your environment variables, otherwise R/RStudio cannot find Rtools. To make sure that Rtools has properly installed open R, not RStudio just R, copy and paste the "writeLines" command from this webpage:

```
writeLines('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"', con = "~/.Renviron")
```

You may have to look for the executable for R from when you originally installed it to run R alone. Next, from inside R-4.1.1 (assuming you are using that version) at the prompt type the command:

```
>  Sys.which("make")
```

This should return the path to where rtools40 is installed. If it does not you need to reinstall Rtools4 until this works properly insuring that Rtools will work when required.

To make sure that the Rtools4 installation has all compiled correctly, and R can see all the paths it should, you can use a command from the R package Rcpp (R to C++) by entering the following commands:

```
> library("Rcpp")
> evalCpp("1+1")
```

It may take a bit of time for everything to finish compiling but eventually it should return the correct answer "2" as:

```
[1] 2
```

If you got the correct path from the Sys.which("make") command, and then the correct answer "2" from Rcpp, then you are good to go to the next step. When you exit R you do not need to save a workspace image. You will be working through RStudio from now on.

Believe it or not, this is where things may get a bit messy. Keep in mind that RStudio is essentially a graphical user interface (GUI) to use the R programming language. Installing TensorFlow in RStudio simply extends that interface to encompass the backend of TensorFlow which is written in Python. This is the point where you may find that when Anaconda was installed it did not completely install a version of Python with accompanying path. I spent countless hours trying to get a path setup for RStudio 1.1.456 to see any version of Python I had already installed on my computer (I have three versions of Python installed). In the end what I wound up doing was installing Anaconda again and again followed by installing keras until I got the message:

```
> install_keras()
No non-system installation of Python could be found.
Would you like to download and install Miniconda?
Miniconda is an open source environment management system for Python.
See https://docs.conda.io/en/latest/miniconda.html for more details.

Would you like to install Miniconda? [Y/n]: y
```

IMPORTANT: the correct answer here is "y" for yes! I missed that the first time not knowing that I would need Miniconda to get everything setup properly for Python. After answering "y" to installing Miniconda, then Anaconda installed a compatible version of Python and the path to it inside Anaconda so that you can actually use keras and TensorFlow in R/RStudio. <u>This seems to be a key part of getting all this to work.</u>

a. Make sure versions of R, RStudio, Rtools, Python, and TensorFlow are all compatible
b. Make sure everything is installed "inside" Anaconda so that all the paths to everything can be seen by all programs.
c. Make sure that all installations are complete, particularly the installation of Python as part of installing Anaconda.

2. Install keras, kerasR, and kerastuneR in RStudio in the working directory you will be using to conduct your analysis. If everything else is installed correctly, you can install all these all at once. If you are not consistently in the same working directory you will get path errors when you try to conduct your analysis.
3. If everything installed completely and correctly you are good to go at this point. The required "MNIST datasets will be automatically available. If you have done something by hand then (down)load the "MNIST" train and test datasets into your working directory. If you have downloaded zip files don't forget to unzip them for use.
    a. Make sure your path in RStudio is set to your working directory.

b. Import the train and test datasets into RStudio. Note that the train dataset is particularly large and will take some time to import.

c. Look at your data! Very large datasets can be tricky to use. If you use the str() command in RStudio to look at your mnist_train dataset you'll get the maximum number of variables (columns) back but then the output is truncated. If you scroll back up you'll see that you have a dataframe with 60001 observations and 785 variables. That is, there are really 60,000 observations plus the column headings row. There are 784 variables, one each for the values of each of the pixels in the 28x28 pixel images, plus one variable for the image label for a total of 785 variables.

   Remember that pre-processing the data can take up to 80%, and sometimes more of the total time to conduct an analysis!

d. Split out the "labels" from the image data in the train and test datasets.

```
> x_train <- mnist_train[2:60001,2:785]
> y_train <- mnist_train[2:60001,1]
> x_test <- mnist_train[2:10001,2:785]
> y_test <- mnist_train[2:10001,1]
```

e. Put the data in the proper array shape for Keras, i.e. img_rows = 28 and img_cols = 28. Follow the remaining directions from the https://tensorflow.rstudio.com/guide/keras/ webpage as required. It will be much easier if you've gotten everything installed completely and correctly and can just use the prescribed commands.

Again, good luck!