Machine learning II
Master Data Science
Winter Semester 2023/24 **Prof. Tim Downie**

Workshop 11
Fitting Regression Neural Networks in R using Torch.

There are many R packages which fit artificial neural networks. They all have different advantages, and disadvantages.

The best in terms of performance and being up to date is Keras, but this requires a python installation and needs to connect to python. This is fiddly at the setup stage and the code is very fussy with the data specification. This makes it a poor choice for ML2 Workshops. The package R-Torch is fairly new. It is much easier to use than Keras and it is robuster, but is much slower.

For background information here is a brief summary of the different R neutral network packages that I have tried:

`nnet`[1] by Venables and Ripley is well written code, but is quite limited, e.g. it only allows for one hidden layer and cannot not graphically display the network.

`neuralnet`[2] has more options, but has some annoying aspects such as factor variables need to be one hot encoded. The implementation is fairly basic.

`RSNNS`: Neural Networks in R using the Stuttgart Neural Network Simulator (SNNS)[3] connects to the pre-compiled code. This package has a lot more functionality than the first two, and is fairly easy to use but the SNNS code is "now very outdated", and is no longer being maintained.

`mxnet`[4] is also a front end R package which connect to a compiled neural network library and can be be accessed by different platforms. This is more comprehensive, but is less robust in terms of installation and version updates, and the code is less intuitive.

`Keras` and `tensorflow`[5]. Keras is the high level interface to the low level open source code Tensorflow. A disadvantage is that the data preparation can be very fiddly and errors can be difficult

[1]*Venables W.N. & Ripley B.D.*, Modern Applied Statistics with S, Springer. 2002 (MASS Library)
[2]*Fritsch S., Günther F., Suling M., & Müller S.M.* (2016)
[3]*Bergmei C. and Benítez, J.M.* (2012)
[4]*Tianqi Chen, Qiang Kou, Tong H.*(2017)
[5]Kalinowski et al., R Interface to Keras https://keras.rstudio.com/ and
R Interface to Tensorflow https://tensorflow.rstudio.com/

to trace. Sometimes memory can be an issue with larger NNs or data sets. This is a good choice if you are prepared to invest the time in the set up and learning curve. You also have an advantage, if you are already familiar with the concepts and code from Python the environment.

If you are interested in learning and using Keras in R the tutorial "keras: Deep Learning in R " https://www.datacamp.com/community/tutorials/keras-r-deep-learning is recommended.

`torch` is based on the python package `PyTorch` but unlike Keras calls its own C code and does not need to communicate with Python. Its big advantage is that it is fairly easy to use and is much more comprehensive than the `nnet` and `neuralnet` packages. Both Keras and Torch can run using a GPU if you have one available. A disadvantage with `torch` is that it is a new package, and there are not yet many resources available in the internet for this package. The few that are around tend to repeat the same examples. Another disadvantage is that it is slower than Keras and my initial impression is the predicted values are not as good.

One of the better introduction websites to `torch` is https://anderfernandez.com/en/blog/how-to-create-neural-networks-with-torch-in-r/. At the end of this website there is an interesting section titled *Comparison creating neural networks with Torch vs Keras / Tensorflow in R* The official R package manual can be found at https://cran.r-project.org/web/packages/torch/torch.pdf and the external website for the R Torch project is at https://torch.mlverse.org/docs/index.html

Exercise 1 Background reading
Read through James 10.6 pp. 432 and 433. This discusses the NN regression you will replicate in Exercise 2.

Exercise 2 MLP Regression for the Baseball Hitters data
Download the code in `ML2_Workshop11.R`. The first part replicates the regression in Lab 10.9.1. pp. 443. but implemented using Torch. This exercise is an adaptation from the code supplied by James et al. in their internet resources pages. Unfortunately the second edition of the book was written using Keras. As you work through the code read the comments provided by James, but keep in mind that many of the commands given are Keras specific. Notice that the results from the `torch` Neural Network are not as good as with Keras.

If you are using the lab computers the Torch package should be already installed. If you are using your own computer, then you will need to install the packages `torch` and `luz`. The complete list of packages needed for this workshop are at the top of the R script file..

Comment about preparing the data for MLPs: When calling the R function `lm()` the data included in the predictor variables are internally passed to the function `model.matrix`, the output of which is then used to calculate the model estimates and all the other model statistics. A particularly useful feature of the function `model.matrix` is that factor variables are converted to several numeric variables. This avoids having to "one hot encode" the factor variables, which is often done in for other packages and has to be coded for every factor variable separately. The `model.matrix` model formula uses includes the outcome variable (in this exercise `Salary`) only because the function requires it, it is not included in the function output, and the `-1` term means that there is no constant vector in the data, because neural networks don't require an input variable consisting of just ones.

Exercise 3 MLP tutorial using the Auto data
This replicates the Keras code in https://tensorflow.rstudio.com/tutorials/keras/regression which is applied to the Auto data. You have used this data before in Workshops 5 & 6 from ML1 investigating the fuel consumption in "miles per gallon" `mpg`. It is worth quickly reading the end of "Inspect the data" in the above website, as this gives a brief overview of the data.

You start off by fitting the same MLP as you used in Exercise 2. Then there is some example code to specify two hidden layers and no dropout. At the end of the exercise you should copy the code to fit an MLP with tanh activation function, 3 hidden layers with 50, 30 and 10 nodes, a dropout rate of 0.2 in each hidden layer and the "adam" optimisation routine.

Exercise 4 MLP with the College data
The key to learning from a tutorial is to adapt what was covered in the tutorial to a data set that you already know. In Worksheet 3 you developed a GAM model to the College data with `log(Accept)` as the outcome variable. In this exercise you will fit an MLP with the same set of predictor variables. Code is included to fit the "best" GAM model, using the same training and test data as for the MLP as a comparison.

Try out different configurations with more nodes and more hidden layers, different drop out parameters, different optimizer engines, learning rate etc. Each time write down the set-up and the test MAE. N.B. the way to alter the learning rate is not very clear, so there is an example given at the end of the source code.