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

Worksheet 13
Neural Networks for image data

In this workshop you will build neural networks to classify the MNIST digit data as discussed in the lecture.

The MNIST data are available from many sources. The original source is the website of LeCun et al.[1], who first provided the data and have done plenty of research in this field, e.g. developing the feature detectors mentioned in the lecture. The data are also available directly from many other sources, including the R-torch package (specifically in the `torchvision` package) and the R-Keras package. Note that exact format of the supplied is slightly different for each data source.

Work through the R-source file supplied in Moodle. The source file is commented, but please read through this overview first as there are some important comments, especially at the end.

Exercise 1

In the first exercise you will follow Lab 10.9.2 on page 445 of James et al. As with the last two workshops the book uses Keras, but the website supports R-Torch, meaning that some of the specific code related aspects are different. I have found an online tutorial: https://rpubs.com/dfalbel/815416 from Daniel Falbel, who developed the R-Torch package. The tutorial runs through the R-Torch version of James et al. Lab 10. Read the second section *Multilayer Network on the MNIST Digit Data* of this webpage in parallel as you work through the R-code. If you have already installed Keras you can use the code in the book, it runs quicker anyway.

The NN implemented in this exercise is not the one developed the lecture. Each image is first “flattened out’ and fed into a fully connected NN, so spatial similarities are ignored.

At the end I have added code so that you can see the confusion matrix and plot some individual handwritten digits. From the confusion matrix you can see which number combinations are more likely to be misclassified. As an Example 4 and 9 are often misclassified. The code supplied plots the first digit and the first misclassified digit in the test data. You are encouraged to look at many more examples of correctly and wrongly classified digits by changing the value of `i`.

[1] http://yann.lecun.com/exdb/mnist/

This fully connected MLP doesn't take advantage of the spatial information in the image, but it still gets around a 94% correct classification rate.

Exercise 2 Optional
This covers the last half page of the Lab 10.9.2, where a NN is implemented without a hidden layer to mimic multiclass logistic regression, also called multinomial regression. The claim is that this is quick and easy, which is not my impression. It took a long time to run, but it does at least show that a simple MLP works better than multinomial regression on these data.

Exercise 3
The website from Wenbo Zhao fits a convolutional neural network (CNN) with max-pooling to the MNIST data: https://jtr13.github.io/cc21fall2/tutorial-on-r-torch-package.html
Note that the local receptive field has optimised weights, no feature detector is used.

Last year, I found that running the code on the full data was not feasible on my computer. One epoch took for ever to run and eventually RStudio run out of memory. With a new desktop PC the full data and 10 epchs runs in an acceptable time. The code provided code runs on a reduced number of images. The code trains the NN on 10 000 images and tests using 5 000 images. Feel free to increase the sample sizes if you think your computer can cope with these.

Compare the three methods by noting the time taken to train the NNs and the accuracy measures for all three exercises.
Overall comments

- The NN models fitted in this Workshop are pushing the computing boundaries of a standard PC. There are some tricks which can help. One is to regularly free up spare memory using the function gc(). Another tactic is to restart RStudiofor each exercise. In the code provided the number of epochs has been reduced to 5 epochs. The main reason is to reduce the running time, but also to help prevent running out of memory. You can investigate increasing the number of epochs if your set-up allows it.
- The function `mnist_dataset()` with argument `download=TRUE` will create a subdirectory called `mnist`, then download and save the data in this directory. There is no need download these data multiple times, so change the line `download<-TRUE` to `FALSE` once you have downloaded the data. So that you know where the data is saved, be sure to set the *working directory* each time you start Rstudio. To do this use the command `setwd()` with the appropriate path or use the menus *Session > Set working directory.*
- If you find the CNN in Exercise 3 runs quickly enough. You could increase the size of the dataset which should improve the accuracy.