Uppsala University

Machine Learning

Assignment 3

General information

- The recommended tool in this course is R (with the IDE R-Studio). You can download R **here** and R-Studio **here**. You can use Python and Jupyter Notebooks, although the assignments may use data available only through the R package, a problem you would need to solve yourself.
- Report all results in a single, `*.pdf`-file. *Other formats, such as Word, Rmd, Jupyter Notebook, or similar, will automatically be failed.* Although, you are allowed first to knit your document to Word or HTML and then print the assignment as a PDF from Word or HTML if you find it difficult to get TeX to work.
- The report should be written in English.
- If a question is unclear in the assignment. Write down how you interpret the question and formulate your answer.
- You should submit the report to **Studium**. Deadlines for all assignments are **Sunday 23.59**. See **Studium** for dates. Assignments will be graded within 10 working days from the assignment deadline.
- To pass the assignments, *you should answer all questions not marked with **, and get at least 75% correct.
- To get VG on the assignment, *also the questions marked with a ** should be answered. Sometimes you can choose between different VG assignments, then this is explicitly stated. To get VG you need 75% both on questions to pass and the VG part of the assignment. VG will only be awarded on the first deadline of the assignment.
- A report that does not contain the general information (see the **template**), will be automatically rejected.
- When working with R, we recommend writing the reports using R markdown and the provided **R markdown template**. The template includes the formatting instructions and how to include code and figures.
- Instead of R markdown, you can use other software to make the pdf report, but you should use the same instructions for formatting. These instructions are also available in **the PDF produced from the R markdown template**.
- The course has its own R package `uuml` with data and functionality to simplify coding. To install the package just run the following:

  1. `install.packages("remotes")`
  2. `remotes::install_github("MansMeg/IntroML", subdir = "rpackage")`

- We collect common questions regarding installation and technical problems in a course Frequently Asked Questions (FAQ). This can be found **here**.
- You are not allowed to show your assignments (text or code) to anyone. Only discuss the assignments with your fellow students. The student that show their assignment to anyone else could also be considered to cheat. Similarly, on zoom labs, only screen share when you are in a separate zoom room with teaching assistants.

- The computer labs are for asking all types of questions. Do not hesitate to ask! The purpose of the computer labs are to improve your learning. We will hence focus on more computer labs and less on assignment feedback. *Warning!* There might be bugs in the assignments! Hence, it is important to ask questions early on so you dont waste time of unintentional bugs.
- If you have any suggestions or improvements to the course material, please post in the course chat feedback channel, create an issue **here**, or submit a pull request to the public repository.

Contents

Keras, Tensorflow, and Google Colab

This assignment can be computationally heavy. If your own an old computer, you might want to run this task in the computer room in the Department or on Google Colab. Google Colab also enable GPU support if you want to try that.

We have setup a Jupyter Notebook with R and Tensorflow at MansMeg/IntroML/additional_material/colab_nb. We suggest *not* to run the code in `markdown`. Instead, run the code in R/Colab and copy the results as output (see the assignment template for an example).

As a first step, you must install `tensorflow` and `keras` on your local computer. You can find detailed information on how to install `tensorflow` and `keras` **[here]**. You can find details on different architecture components in the `keras` reference found **[here]**.

1 Feed-Forward Neural Network using Keras and TensorFlow

If you have installed `keras`, you can load the MNIST dataset. This dataset contains data on handwritten digits that we want to classify. To access the data, we use `keras` as follows.

```
library(keras)
mnist <- dataset_mnist()
mnist$train$x <- mnist$train$x/255
mnist$test$x <- mnist$test$x/255
```

1. As a first step, we visualize a couple of digits. Include the first eight you can find in the training dataset in the report as an image.

   ```
   idx <- 3
   im <- mnist$train$x[idx,,]
   # Transpose the image
   im <- t(apply(im, 2, rev))
   image(1:28, 1:28, im, col=gray((0:255)/255), xlab = "", ylab = "",
           xaxt='n', yaxt='n', main=paste(mnist$train$y[idx]))
   ```

2. How large is the training and test set?

3. Implement a simple feed-forward neural network in R using TensorFlow and Keras. See the introduction **[here]** or **[here]**. Implement a neural network with one hidden layer with 16 units and the sigmoid as the activation function for the hidden layer. The output layer should be a softmax. Print your `keras` model description and include it in your report.

   (a) How many parameters do your model have in total?

   (b) How many parameters does the input layer have?

   (c) How many parameters does the output layer have?

(d) What do you get as classification accuracy? Include the validation accuracy and error per epoch figure/plot that is automatically generated.

Hint! The classification accuracy should be around 0.85.

4. Now, we will play around with the essential concepts in feed-forward networks. Do the steps below, but feel free to tinker around and test other values. Try to understand the effect of the different choices. Report the *validation* set accuracy/error and the validation accuracy/error per epoch for each step in a figure/plot together with the `keras` model description for the *final* neural network with all changes. Start from the simple neural network above, then make the following changes.

   (a) Increase the number of hidden units to 128.

   (b) Change the activation function to reLU.

   (c) Change the optimizer to RMSprop.

   (d) Add a second layer with 128 hidden units.

   (e) Add dropout with 0.2 dropout probability for both of your hidden layers.

   (f) Add batch normalization for both of your hidden layers.

5. If you would use early stopping regularization, how would you do that?

   Note! Only describe what you would do conceptually. You don't need to do it.

6. Now try to build the best possible network architecture for this data (but still only using a feed-forward network). How good can you get validation set accuracy (or how low is the error)? What is the architecture used for the best model you can get? Include the final `keras` code, and the *validation* set accuracy figure. Feel free to use early stopping!

   Hint! The general idea is to find the optimal capacity of the model. Try to start with a network that overfits the data and then regularize that model to improve the generalization error on the validation set.

7. Identify two digits that your best network has classified incorrectly. Visualize them and include the digit number and what your network classified it as. Why do you think the network made the classification mistake?

8. Use your model to compute the accuracy and loss on the MNIST hold-out test set included with the data. If you get a lower accuracy (or higher error) than on the validation set (or higher error), why is this the case?

2 * A simple neural network from scratch

This task is only necessary to get one point toward a *pass with distinction* (VG) grade. Hence, if you do not want to get a *pass with distinction* (VG) point, you can ignore this part of the assignment.

We want to implement one of the most straightforward neural networks possible.

1. Implement the neural network in Goodfellow et al. (2016, Ch. 6.1) as a function that takes as an input a 2×2 weight matrix `W`, a vector `c`, a vector `w` and the scalar b. It should work as in Ch. 6.1., here is an example.

```
W <- matrix(1, nrow = 2, ncol = 2)
c <- matrix(c(0, -1), ncol = 2, nrow = 4, byrow = TRUE)
X <- matrix(c(0,0,1,1,0,1,0,1), ncol = 2)
w <- matrix(c(1, -2), ncol = 1)
b <- 0

mini_net(X, W, c, w, b)
```

```
##      [,1]
## [1,]    0
## [2,]    1
## [3,]    1
## [4,]    0
```

```
mini_net(X, W*0.9, c, w, b)
```

```
##      [,1]
## [1,]  0.0
## [2,]  0.9
## [3,]  0.9
## [4,]  0.2
```

2. Now test changing the value $W_{1,1}$ to 0. What do you get for the results of the network?

3. What is the current output function? Is the network output function reasonable? Do you have any other output function that might be better?

4. Now implement a mean squared error loss function for the network. It should work as follows.

```
y <- c(0,1,1,0)
mini_net_loss(y, X, W, c, w, b)
```

```
## [1] 0
```

```
mini_net_loss(y, X, 0.9*W, c, w, b)

## [1] 0.015
```

5. Again, change the value $W_{1,1}$ to 0. What is the value of the loss function?

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.