Artificial Intelligence — Lab Sessions 1-2

ESIR – Université Rennes 1

2021–2022

For any questions outside of the lab sessions, feel free to e-mail me at : **Tom.Bachard@inria.fr**.

The goal for this first set of exercises is to familiarize yourself with the programming environment that we will use. Particularly with *Keras*, the deep learning python API that enables to program neural networks in a concise way.

Evaluation : At the end of the AI practical session, you are required to submit a report. The report should consist of all the materials that you learned, experiments along with the insights. While we expect you to submit the final report after the last practical session, you should start editing the document from the first session on. Your final evaluation will be based on your gitlab (see Exercise 0) repository and on your report.

Exercise 0 : Setting up the working repository

We propose to use the gitlab of ESIR/ISTIC - https://gitlab.istic.univ-rennes1.fr/. Each student should have a git account. For each group, two persons maximum, you should create a single git repository for your lab sessions. The repository should be associated (add as a member) to the binom git accounts. Commit your code to the git frequently, but as an absolute minimum, at the end of each TP session.

Naming convention for the git repository : **AI_Name1_Name2**. Replace Name1 and Name2 with the names of the people in your group. *Example :* AI_Galassi_Miklos

During the creation of the repository, do not forget to give **Developer** role to :

— Zoltan Miklos (**@zmiklos**) ;
— Francesca Galassi (**@fgalassi**) ;
— Tom Bachard (**@tbachard**).

You can find below some guidelines and tutorials to use gitlab :

— https://docs.github.com/en/get-started/using-git/about-git
— https://www.tutorialspoint.com/git/index.htm

Exercise 0.5 : Tools and libraries

This exercise propose to get familiar with the different tools and library you will use for the different lab sessions.

Anaconda Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data analytics, data Mining, machine Learning, etc.) that aims to simplify package management and deployment. It helps to easily set up the whole environment for data analytics. We will use anaconda as a platform in the whole course. Some useful links : https://docs.anaconda.com/anaconda/navigator/getting-started/ and https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

We propose to create an environment with python version 3.5 in anaconda and install/update useful packages : Jupyter notebook, Numpy, Pandas, Matplotlib, seaborn, Scipy, scikit-learn, Tensorflow.

Example :
conda install -c anaconda jupyter
conda install -c conda-forge tensorflow

Jupyter Notebook We propose to write your code in the form of jupyter notebooks (instead of editing you files in an integrated programming environment, *IDE*, like Eclipe). In fact, notebooks can be used very efficiently for data analysis, you can realize simple analysis for data exploration tasks in notebook cells, and you can explain your code in textual comments. You can also visualize your data. Jupyter Notebook for Beginners : https://www.dataquest.io/blog/jupyter-notebook-tutorial/

Naming convention for the Jupyter notebooks : **TP1_2**, **TP3_4** and **TP5_6**.

Note : Use markdown cell to write your observations in Jupyter notebook, after the codes.

Tensorflow 2.0 TensorFlow, supported by google is an end-to-end open source platform for machine learning and in particularly deep learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that let researchers push the state-of-the-art in deep learning and developers easily build and deploy machine learning powered applications : https://www.tensorflow.org/tutorials

Check the Tensorflow version ! We want to use Tensorflow 2.0.

```
import tensorflow as tf
print(tf.version.VERSION)
```

Exercise 1 : First hand on a neural network

Objective : Understand how the basic blocks of a simple neural network work.

First, read the following code to grasp the main idea of the learning of the XOR gate : understanding-XOR-with-keras-and-tensorlow. Then, run and understand the provided python code (the *XOR_TP12.ipynb* file in moodle) in a Jupyter notebook in order to compute the XOR gate. Finally, based on the following lines, explain how the neural network is trained.

```
sgd = tf.keras.optimizers.SGD(learning_rate=0.1)
model.compile(loss='binary_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])
```

After quickly reminding their truth tables, extend the code for the OR gate, the AND gate, the NOT gate and the NAND gate. To help you with your report, you can try to answer the following questions :

- What is the influence of the parameter *epochs* in the function *fit* ? Is there an optimal value ?
- Does changing the values for the number of epochs, the number of neurons or the layers affect the training ? Experiment with varying epochs, number of neurons and layers. Write your observation in the jupyter notebook.
- What is the *batch_size* parameter in the function *fit* ?
- Is it possible to chose a smaller-size hidden layer to realize the gates ? Why ?
- Explore other activation functions, loss functions and optimizers. How they affect training the network.

Exercise 2 : Building a neural network to classify images

Objective : Build a multi-layer perceptron to classify images.

The *Fashion MNIST* dataset is a collection of 70,000 gray-scale images in 10 categories. Here we propose to construct and train a multi-layer perceptron to classify these images in the 10 categories. The dataset is widely used for image classification (https://keras.io/datasets/) and even used as a baseline in nowdays baseline for some classification models.

Here, to help you with your report, we advise you to follow this three-stepped protocol :

— First, explore your dataset. Observe some images, calculate some basic statistics on the data, look at the labels, *etc.*

— Then, run the code (https://www.tensorflow.org/tutorials/keras/classification) to train the classifier and label the images. Be sure to understand what each line is doing.

— Finally, toy with the different parameters (including, but *not limited to* topology, layer size, activation functions, *etc*). Observe the effects on the classification accuracy and run time, and explain why changing the parameters the way you did produced the changes you observed.

Exercise 3 : Building a neural network to classify texts

Objective : Understand the notion of over-fitting and under-fitting.

In this exercise, we propose you to witness the phenomena of over-fitting and under-fitting. To do so, you will train a MLP (see Exercise 2) on the IMDB dataset and classify its reviews. This dataset is made of 50,000 movies reviews from IMDB, labeled by sentiment : positive or negative. You can find more information here : https://keras.io/datasets/.

Like the previous exercise, we propose you to follow an classical approach to the problem :

— Explore the dataset and calculate some basic statistics. Try to recover at least one review with the help of the dictionary mapping (based on this Tensorflow tutorial).

— Experiment with different network sizes : try to reduce or increase the network size, and observe the effect on the performance (accuracy of the prediction) and the running time.

— Change the network parameters to demonstrate the effects of over-fitting. Example.

As an additional resource, you can find here some strategies that help preventing over-fitting.