Artificial Intelligence — Lab Sessions 1-2

ESIR – Université Rennes

2023–2024

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**) ;

— Thibaut Le Marre

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 : Building a neural network to classify texts (multi-layer perceptron (MLP))

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 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.

Exercise 2 : Recurrent Neural Network and IMDB classification

Objective : In this exercise you will familiarize yourself with recurrent neural networks and in particular with LSTM.

Dataset : Download and load the IMDB dataset.

Analyze this LSTM code up to testing. It is the code given in example in the Keras library. It demonstrates how to construct a text classifier using LSTM for a text classification task. We would like to predict the sentiment of short textual comments about movies, recorded on the site IMDB. Based only on the text content, we would like to predict whether a comment is positive or negative.

Run the example code. Analyze the performance of the classifier on the test set and training set :
— If your training loss is smaller than test loss, then your network is likely to *overfit*. In that case, add (or increase) dropout and/or decrease the network size.
— If your training loss and test loss is almost the same, then your model is likely to *underfit*. In that case, increase the size of the network and/or the number of nodes/layer.
Do not forget to include all your remarks in your report !

Further tutorial and other usefull links that we (**strongly !**) advise you to read :
— Tensorflow text classification with RNN ;
— Sequence classification using LSTM. Unfortunately, the blog is not updated and the tutorial is implemented in tensorflow 1.0. You may tune it for running on tensorflow2.0 environment ;
— Understanding LSTM ;
— The notebook file *Explore Overfitting and Underfitting with IMDB dataset* in Moodle.

Exercise 3 : Text classification on the Ohsumed dataset

Objective : The goal of this exercise is to realize a text classifier using deep neural networks. Your task is to construct a classifier, using the available training set, and evaluate it using the test set. The classifier should predict the category for the articles.

Dataset : We will work with the Ohsumed dataset that contains abstracts of scientific articles from a large medical publication database. The articles are related to one of 23 categories of cardiovascular diseases. The dataset is uploaded on moodle.

Dataset description : The dataset has two versions :
— One that contains the first 20000 articles (split into training and test set), available on Moodle ;
— A more complete version that has all articles (50000 articles), available here.
You should work with the version that has 20000 articles, and only use the complete version **only once** you constructed and analyzed an efficient classifier for the smaller set. As usual, the dataset is split into two parts : a training set and a test set. Each article of the collection is labeled with one of the 23 categories.

Overall strategy : To realize a classifier you should use an **iterative** approach : try to realize a simple classifier first, then analyze its performances (e.g. using the accuracy metric), then chose techniques to improve the performance (in terms of accuracy) of your classifier. Try to describe and document the development process in your report : which experiments did you run ? What were the results ? What did you do to improve the performance and why ?

To implement your first text classifier from scratch, we advise you to :
— Look into the data, get familiar with its structure and the type of text you will process ;
— Explore the data and compute some basic statistics. For example, what is the size of the vocabulary ? What is the number of examples in the training/test set per category ? What is the frequency of the words ? What is the most common, the least common ? Feel free to add anything you find relevant on the dataset ;

— Preprocessing : adjust your data to be able to pass them into a neural network ;
— Create a neural network with a simple architecture and train it ;
— Evaluate the performance of your classifier on the training set and the test set. These numbers can give hints about the quality of your classifier and can also can guide you to decide what to do in order to improve the performance. Beware of under-fitting and over-fitting ;
— Interpret your results (in plain text) and set your next objective (for example, if you had an over-fitting and you decide to add regularization, explain it) ;
— Iterate (many many **many** times) and try to improve your model. Tunning a network is sometimes difficult due to all the possibilities for the hyper-parameters.

In order to improve your network during your iterations, you can try to :
— Invest time to do more preprocessing. For example, remove frequent words, remove stopwords, perform stemming. A quick introduction here ;
— Change the parameters of your algorithms and your optimizers, cost functions, *etc* ;
— Add dropout, early stopping, regularization ;
— Change the network topology/use different techniques (multilayer perceptron, LSTM, *etc*) ;
— Only when you managed to obtain good results, witch to the bigger dataset.