

Practical work 07 – 04/04/2023

Deep Learning Frameworks

Objectives

The main objective of this PW is to understand the overall functioning of TensorFlow 2.0, a computational graph framework used for machine learning and deep learning. More specifically, we want to experiment with the definition of a graph composing a multi-layer perceptron (MLP).

Submission

- **Deadline** : Tuesday 11th April, 15pm
- **Format** :
 - Completed Jupyter notebook with the blanks filled in (the sections to be completed marked as usual).
 - Comments and results (plot with weight histograms showing the results for the different parameter settings) either in the notebook or in a pdf-report.

Submission of all files in a single zip-file using the naming convention (for team of two students #1, #2) :

family name_given name #1- family name_given name #2.zip

Exercise 1 Computational graph exercise - pen & paper

The logistic regression (LR) is a well known 2-class classification algorithm. LR is actually similar to the generalised perceptron - cf week 2. A LR outputs values between 0 and 1.0 with the following equation :

$$\hat{y} = h_{\theta}(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x} + b)} \quad (1)$$

where the parameters θ of the model are defined with a weight vector \mathbf{w} and a scalar value b . Assuming an input space with 2 dimensions, the LR function becomes :

$$h_{\theta}(x_1, x_2) = \frac{1}{1 + \exp(-(w_1 x_1 + w_2 x_2 + b))} \quad (2)$$

- a) Revise slides 17–28 from the slides (SW06) and propose a computational graph with an input space of 2 dimensions (Eq. 2). Use operators as granular as you can. You should have about 7 nodes in your graph.
- b) On the graph, illustrate the forward pass using numerical values.
- c) Illustrate the back-propagation of a gradient value of 1.0 at the output of the graph using the numerical values of b).

Exercise 2 TensorFlow 2.0 installation

Install TensorFlow 2.0 on your machine. Follow the instructions given in slide 34 and under <https://www.tensorflow.org/install/pip>. We recommend that you use a virtual environment to isolate your Python packages in a directory.

As alternative you can also :

- Use our class JupyterLab server at <https://icolab-gpu-2.tic.heia-fr.ch>. See instructions to login in document
- Use notebooks in Google Colaboratory.
- Use Docker with an image including Python, numpy, Jupyter, TensorFlow 2.0, etc.

Exercise 3 MLP with TensorFlow 2.0

The Figure 1 below illustrates a Multi-Layer Perceptron. The network takes inputs of dimension D , has H hidden neurons and K output neurons. In this case we assume that all activation functions of the neurons are ReLU for the hidden layer and sigmoids for the output layer. The objective of the exercise is to implement computational graphs in TensorFlow 2.0 to train and use such an architecture. The constraints we put ourselves is to use “low-level” functions of

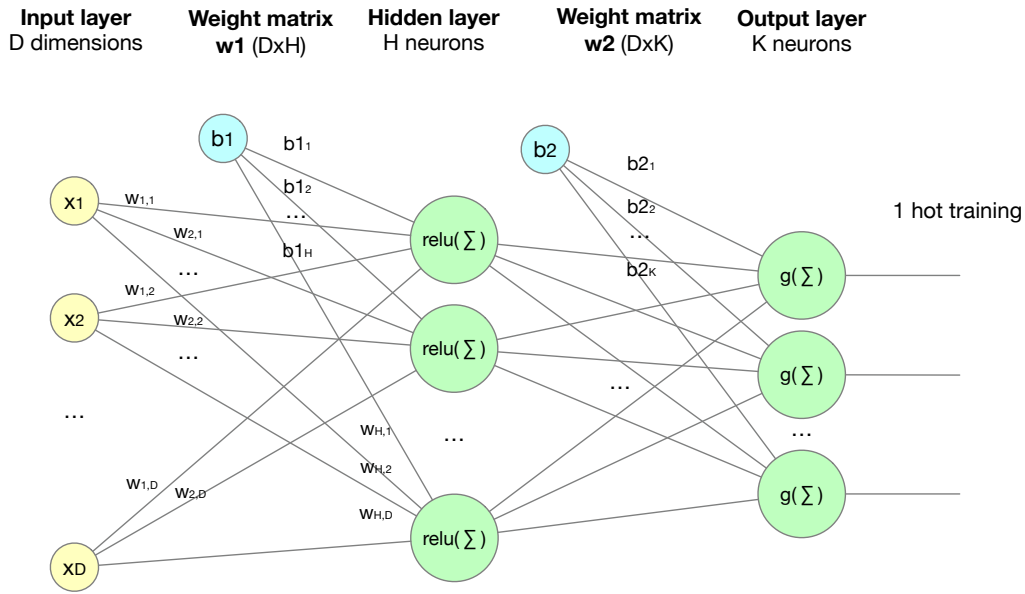


FIGURE 1 – Multi-layer perceptron with one hidden layer.

TensorFlow, i.e. we will not use high-level Keras functions to compose layers and to train the parameters.

- Revise slides 42 –49 from the class on TensorFlow and make sure you understand the principles and the different steps that were described.
- Use the iPython notebook provided on Moodle and complete the missing parts of the code in order to perform a training and testing phase. We assume that the loss function used will be the Mean Squared Error (MSE) loss function.
- Make sure that the network is converging and compute the error rate on the test set. You should observe an error rate at about 8-12%.

Optional. Implement a simple learning rate schedule strategy, i.e. reducing the α along the training (piecewise).

Exercise 4 MLP with TensorFlow 2.0 and Keras

Analyse the code provided in slides 58-60 and implement the MLP of Exercise 3 using Keras. Note : let Keras handle the batch size in the `model.fit()` method.

- Provide the code in new cells at the end of your notebook of Exercise 3.
- Compare the performances with your implementation of Exercise 3 (cpu, accuracy) and comment.

Optional. Redo this exercise using a softmax activation at the output layer and with a cross-entropy loss function. Compare the performances with the other implementations.