Teachers: J. Hennebert, M. Melchior
Assistants: Y.-I. Beffa, S. Hackstein, O. Zayene

Practical Work 03 – 10/03/2022
Back-Propagation

Objectives

Main objective is to explore the capabilities of MLPs for classifying MNIST digits and to implement the back propagation algorithm for an arbitrary MLP. Again, the solutions can be elaborated by starting from Jupyter notebooks that will give you guidance.

Submission

— **Deadline** : Wednesday 23 March, noon

— **Format** :

Completed and well documented Jupyter notebooks with the blanks filled in.

Plots with learning curves showing the results for different model complexities, final performance achieved (in numeric format) **and comments** also in the same notebook.

Indicate the name of the group in the name of the notebook.

— **One submission per group !**

Exercise 1 Optional : Object-Oriented Python

Get up to speed with object-oriented python (if not up-to-speed already). See a suitable online tutorial (such as e.g. https ://realpython.com/python3-object-oriented-programming/).

Exercise 2 Exploring Performance of MLPs in MNIST Classification

Experiment with different variants of MLPs to explore the performance in classifying MNIST images.
Use as a basis the Jupyter notebook `pw03_mlp_mnist_classification_stud.ipynb`.

Proceed as follows :

a) Split the training dataset (as loaded by using pytorch functionality) into a training and a validation partition.
b) Train various different models (at least 5 variants) with different numbers or layers and units per layer and by using the cross entropy cost function. For each variant, properly tune the learning rate and make sure that you run the training sufficiently long. Choose a batch size of 64 and use ReLu or Sigmoid as activation function for the hidden layers. Create suitable plots with the cost and accuracy vs number of epochs to demonstrate that the model learns something (incl. training and validation cost and accuracy).
c) Compare the validation accuracy for the different models and identify your favourite model. Create a summary table with the training and validation accuracy. Finally also compute the test accuracy for your favourite model (number of layers, units per layer, activation function, learning rate, number of epochs) and include it in the summary table.

Exercise 3 Implementation of Back-Propagation without Autograd

Here you are asked to implement the back-propagation algorithm for an arbitrary MLP **without using pytorch's autograd functionality**.
Use as a basis the Jupyter notebook `pw03_backprop_stud.ipynb`. The notebook should guide you through all the steps needed.

Finally, you will be able to

a) validate whether you compute correct gradients by comparing with the results provided by pytorch's autograd ;
b) train an MLP with an arbitrary number of layers and units and compare its performance with what you have obtained in exercise 2.

Exercise 4 Optional : Autograd

Watch the two videos sessions 4.1 and 4.2 from Francois Fleuret's deep learning lectures (https ://fleuret.org/dlc/). Play through at least some of the examples.

Exercise 5 Optional : Review Questions

a) Determine the number of model parameters of an MLP for original MNIST with 3 hidden layers of 100, 200, 50 units.
b) Could it be beneficial to define a model with 3 hidden, linear layers of 100, 200 and 50 units without activation function in between ? Why or why not ?
c) What are the benefits of using the backprop algorithm for computing gradients ?
d) What are the variables in MLP that need to be remembered per layer during the forward pass to have them available in the backward pass ?