

Introduction to deep learning

Mathieu Lefort

November 8 & 15, 2023

This project can be done in pair working (in this case you will precise in your report what each one of you did). You have to provide a report including your code and your answers to the questions at the end of each session and the final version before December 3. Do not forget to put your(s) name(s) in the uploaded file.

1 Aim

The aim of the project is to implement the MLP and CNN models seen during the courses, to understand how they work and to become familiar with the PyTorch framework.


2 Installation

You have to install python, pytorch and numpy (and matplotlib if you intend to do some graphs/plots). To install PyTorch, have a look here : <https://pytorch.org/get-started/locally/>. To check if installation was successful, open a python terminal and type the following command : `import torch`.

3 PyTorch

You will find the list of available functions here : <https://pytorch.org/docs/stable/index.html>. Have a look to this mini tutorial : http://pytorch.org/tutorials/beginner/pytorch_with_examples.html. Key concepts are presented here : <https://pytorch.org/tutorials/beginner/basics/intro.html> (you can skip chapters 3 and 7).

4 Data

You have at your disposal the MNIST dataset than contains images (with size of 28×28) of hand written digits (e.g. here is an image of a 5 in the dataset : ). The dataset has the following structure : ((train_images_array, train_labels_array), (test_images_array, test_labels_array)). The images are represented as a vector of dimension 28×28 , and the labels are in one-hot encoding (e.g. if the image corresponds to a 5, the label will be : [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]).

Thus, the input layer of each network has 784 dimensions (28×28 pixels) and the output layer has 10 dimensions (the i^{th} value represents how much the network recognizes the digit i in the image).

5 Part 1 : Perceptron

You have at your disposal two files implementing (the same) perceptron :

1. PERCEPTRON_PYTORCH.PY which uses only tensors
 2. PERCEPTRON_PYTORCH_DATA_AUTO_LAYER_OPTIM.PY which uses most of PyTorch tools (dataloaders, automatic gradient, layers and optimizers)
- Indicate the size of each tensor of the provided file PERCEPTRON_PYTORCH.PY. Explain.

6 Part 2 : Shallow network

In this part, you will implement a MLP with only one hidden layer and a linear output layer.

- By taking inspiration from PERCEPTRON_PYTORCH_DATA_AUTO_LAYER_OPTIM.PY, implement a shallow network using the tools provided by PyTorch.
- Find some hyperparameters (η and the number of neurons in the hidden layer) that provide a good performance. Explain precisely your methodology and the influence of each hyperparameter on the performance.

7 Part 3 : Deep network

As you are now more familiar with PyTorch, you can use it to test deeper models.

- Implement a deep network (i.e. with at least two hidden layers) using the tools provided by PyTorch.
- Find some hyperparameters (η , number of hidden layers and the number of neurons in the hidden layers) that provide a good performance. Explain precisely your methodology and the influence of each hyperparameter on the performance.

8 Part 4 : CNN

As you have now a deeper understanding of the MLP, it's time to implement a CNN architecture (more adapted to images).

- Implement a simple CNN using the tools provided by PyTorch (you can take inspiration from LeNet5).

9 Part 5 : To push forward (optional)

If you have time and if you want, you can look at the following questions (non exhaustive list) :

- Try another loss function (as cross entropy).
- Try other activation functions as tanh or ReLU e.g..
- Try other initialisation functions for the weights (e.g. gaussian or Xavier).
- Try a better gradient method than SGD such as Adam or Adagrad.
- Use a pretrained ResNet model.