Introduction to Deep Learning using PyTorch

Antoine Perquin

March 14, 2022

1 Introduction

The purpose of this short exercise is to model simple mathematical functions using Neural Networks implemented with the PyTorch framework. For example, given inputs x and y, we need to learn approximations of linear functions such as $f : x, y \rightarrow x + y$ and $f : x, y \rightarrow x * y$ but also non-linear ones such as $f : x, y \rightarrow x^2$ and $f : x, y \rightarrow log(x)$.

The code needed to load data but also to perform the training and evaluation of a model has already been written. The student only has to write the code to build the neural network. In this work, we only consider feed-forward fully-connected neural networks. As such, the models can simply be written using the Sequential[1] container using the following syntax :

```
model = Sequential(
    Linear(2, 3),
    Tanh(),
    Linear(3, 3),
    Tanh(),
    Linear(3, 1),
    Sigmoid()
)
```

This short code implements a neural network with 3 layers. The first 2 are each of dimension 3 with *tanh* activation functions, the last one is of dimension 1 with a *sigmoid* activation function.

2 Linear Functions

Run the `train.py` file as is, without modification. It will train the model given as example in the introduction to $x + y$ given x and y. After the training is finished, a graph summarizing the training progress is drawn in the file `training_plot.png`, and an example of inputs, ground truths and predictions are printed in the console.

[1]https://pytorch.org/docs/stable/generated/torch.nn.Sequential.html

Question 1 The example given in the introduction does not give good performances. How can we see it ? Why can it not offer good performances ?

Question 2 Build a neural network with a single layer and train it to predict $x + y$ from x and y.