Deep Learning Mini-Project 2 Report
Implementing from Scratch a Mini Deep-Learning Framework.

Group 78*
Caterina Bigoni and Nicolò Ripamonti

18 May 2018

1 Introduction

2 Code

The library handles the network structure by means of dictionaries, the ideal type to represent graphs in Python language . The generic dictionary that we have considered can be written as

$$\begin{aligned}\text{Dictionary} = \{\ &\text{key}_1\ : [\ \text{value}_{1,1},\ \text{value}_{1,2}, \dots],\\ &\text{key}_2\ : [\ \text{value}_{2,1},\ \text{value}_{2,2}, \dots], \}\\ &\dots\\ &\text{key}_N\ : [\ \text{value}_{N,1},\ \text{value}_{N,2}, \dots], \},\end{aligned}$$

where for each key we may have an arbitrary (and possibly different) number of values. We use the list constructor $[\dots]$ to gather all the values related to a certain key because lists are containers easily iterable and we need this feature as explained later on.
The first dictionary that has to be defined by the user, called `operators` in `test.py`, is the one that records the modules used in the network. Let us consider a simple network, as the one represented in Figure 1. The related dictionary is

$$\begin{aligned}\text{operators} = \{&1\ :\ \text{Linear1}\ ,\ 2\ :\ \text{Nonlinear1}\ ,\\ &3\ :\ \text{Linear2}\ ,\ 4\ :\ \text{Nonlinear2}\ ,\}\\ &5\ :\ \text{Sum}\ ,1\ :\ \text{Linear3}\ \}.\end{aligned}$$

The next step is the definition of a `connectivity` dictionary: from this variable, an object of the class `Network`, will generate two additional dictionaries, used to introduce and ordering in the forward pass.

2.1 General Network

2.2 Modules

With the term *module* we mean each possible element of the network from linear/nonlinear layers to the loss function. Moreover, other operators such as the sum of two different inputs are treated as modules, and hence we obtain an easier representation for more complex networks 3 . The general structure of a module object is given in `ModuleBase.py` and in each derived object we have to define the following methods:

*As agreed with Dr. F. Fleuret, L. Pegolotti has collaborated with group 78 for Project 1 and with group 79, with M. Martin, for Project 2. C. Bigoni and N. Ripamonti have worked together on both projects.

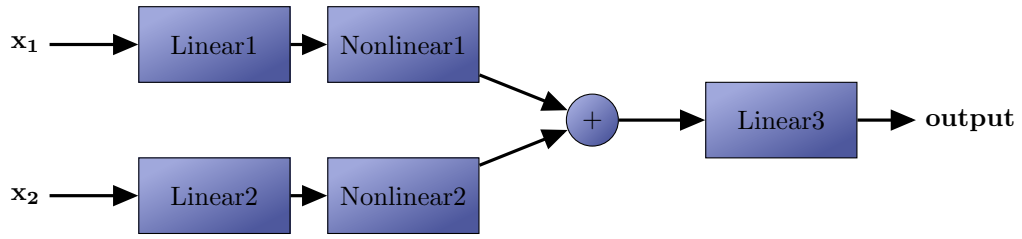

Figure 1: Example of network that can be represented using our library.

- `forward`: Implement the forward pass given the input of the module and store both the input and the output as attributes of the class.
- `backward`: Compute the gradient of the loss with respect to the input given the gradient of the loss with respect to the output. If the module contains parameters, it also computes the gradient of the loss with respect of them.
- `update_param`: Given a learning rate as input, updates the parameters of the module using a gradient descent based approach.

2.2.1 Linear Layer

3 Numerical Experiments

4 Conclusion