Computational Physics III
Machine Learning
WS-18/19

Distribution: 30/11/18, Due: 13/12/18

Overview

The aim of this tutorial is to design a Neural Network to further test the understanding of *Back-propagation*. The final product should be a neural network capable of simulating the classical *Logic Gates*, i.e. AND, OR, XOR, and NAND. To construct such a network it is important to have 2 inputs (i_1, i_2), a single output (o_1) and (for this exercise) one hidden layer network with 3 neurons (h_i, $i = \{1, 2, 3\}$).
This exercise is a natural extension of the previous one and represents one of the most common applications of NN.

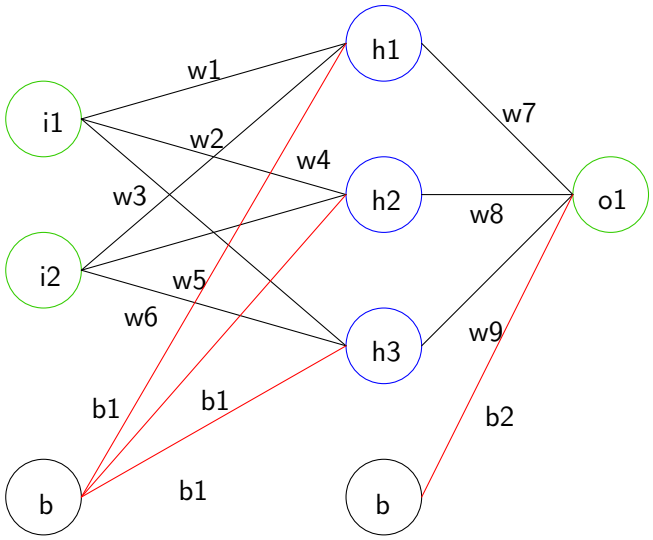

Figure 1: Neural network for logic gate

The initial weights and biases are listed in the table below:

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	b	b_1	b_2
0.15	0.20	0.25	0.30	0.40	0.45	0.50	0.55	0.35	1.00	0.60	0.05

Backpropagation is used to optimize the weights (ignore optimizing the bias weights for this exercise and keep them as they are given in the table) so that the neural network can learn how to correctly map arbitrary inputs to outputs. We will work with the usual training set for logic gates (see below) while the outputs need to match the gate chosen.

i_1	0	0	1	1
i_2	0	1	0	1

1. Choose whether to use the *sigmoid* activation function:

$$f(x) = \frac{1}{1+e^{-x}}. \tag{1}$$

or the following:

$$f(x) = \frac{1+tanh(x)}{2}. \tag{2}$$

Plot both functions (choose a proper x interval to show the difference between the two). What do you expect will change by using one instead of the other?

The error function used here is the same as in Exercise 4. Refer to it (or your notes) for all the formulas for backpropagation and feed forward pass.

Programming

Implement now your choices into a python code (it is easier if you all use the same programming language).

1. Define separate functions for the the activation and the error function. Set different arrays for inputs, outputs and weights+biases and initialize it to the values.
Hint: pay attention to the shapes and try to make use of scalar products (numpy.dot(x,y)) to optimize the code.

2. Implement the Feed Forward pass (FF) and the backpropagation step (BP) into two separate functions.

3. Set the learning rate $\eta = 0.5$ and execute a trial run with just one FF+BP step and show that the error is decreasing. (1 point)

4. Set the total number of steps/epochs, $N_{tot} = 10^4$ and evolve your system using a for/while loop. Show your final results and how the error decreases from the initial trial run. Show also the results obtained in a truth table for the chosen logic gate.
Note : Use the original weights and not the updated weights while continuing the backpropagation algorithm until the next iteration of forward feed phase.

Further Analysis

After getting our results we want to derive some considerations from them. To do so you need to change some parameters and run again your code:

1. Pick a different logic gate with respect to your initial choice and try running again the program. Try using as initial weights for the new run the one you got at the end of the previous one (for a different logic gate). How do they evolve? Try now using the initial set of weights. What differences can you see?

2. Change now your activation function and run again your program. Can you see a difference?

3. Play around with η and try to optimize it for the problem at hand. Show your results and considerations.
Hint: If your total step number is too high try reducing it in order not to spend too much time into the runs.

4. Make a brief summary of your results with some comments about your choices and what they have

Final Remarks

This exercise is meant to help you design from a scratch a neural network and address some of the main problems when working with such systems.
The exercise is also intended to work with both the coding and the planning/understanding aspect, giving you the possibility of walking out of the path and to make some practical experience before starting your work for the seminar. Since results may vary a lot given your preferences and decisions *feel free to contact us for any help or doubts*.
Regarding the evaluation we're interested in seeing your way of reasoning and how you approach to the problem. Less importance will be given to the coding part (it is, in fact, easy to incur into bugs that may impede you to complete the tasks: if that is the case contact us).

Points: we assign 3 point to each of the step of the exercise unless stated differently. Please provide a pdf file with your comments and separate files for the code and the plots. Any problem with the grading or the solution can be discussed with us.