Programming Assignment # 1

Due: Thursday, February 23, 2023, 11:59 p.m.
Total Points: 100

Grader: Junyi Fan
Office Hours: TW 3:30pm - 4:30pm (Zoom)
Contact: Slack

This assignment is designed to give you practical programming experience with neural networks. Many aspects of this class can be fully understood only with computer simulations. Programming assignments help you gain first-hand experience with the algorithms introduced in the class. Please use Python for the implementation, but NOT a neural network toolbox (message the TA/instructor in Slack if you are unsure). If you are unfamiliar with Python, please reach out to the instructor and grader as soon as possible.

Please carefully read all instructions below and also periodically check Slack for updates. Any questions that you post to the public Slack channels should NOT contain any portion of your solution. If you need help with your solution, you can send private messages to the instructor or grader. Post general questions to the programming-1 Slack channel.

This is an individual assignment, so you must submit your own work. You also cannot use other resources (e.g., web sites, books, blogs, research papers, other people, etc.) to solve the problems. **Your assignment must be submitted by committing your code to your Github repository before the deadline. Please create a folder called 'pa1' (without quotes) and commit all files for this assignment to this folder.** You are strongly encouraged to submit early and often to avoid any last minute issues and to confirm that you are committing your work correctly. The submission should be in the form of a Python 3 Jupyter Notebook file. **Be sure to run the file before committing, so that we can directly see your results**. Programs that were not run before submitting will result in a zero.

What you need to turn in: (1) a typed 2-page summary report, where you include a discussion of how performance varied with η, describe what is happening in 2-3 distinctive learning curves, discuss how the momentum term impacted convergence, and discuss any other conclusions and findings. Submit the report to Canvas; and (2) your source program with test results (submitted to GitHub).

This assignment must be submitted on time to receive credit. No late work will be accepted, unless you have a prior arrangement with the instructor. **Your grade will partially depend on completeness, correctness, code efficiency (e.g., using linear algebra whenever possible), and code readability (e.g., comments).**

Question 1. [100 POINTS]

Implement a two-layer perceptron with the backpropagation algorithm to solve the parity problem. **You must implement the forward and backpropagation paths entirely on your own, including an implementation of a perceptron and the activation function.** The desired output for the parity problem is 1 if an input pattern (which contains 4-binary bits) contains an odd number of 1's, and 0 otherwise. Follow the algorithm introduced in class and consult the textbook. Use a network with 4 binary input elements, 4 hidden units for the first layer, and one output unit for the second layer. The learning procedure is stopped when an absolute error (difference) of 0.05 is reached for **every** input pattern. Other implementation details are:

- Initialize all weights and biases to random numbers between -1 and 1.
- Use a logistic sigmoid with $a = 1$ as the activation function for all units.

After your basic programming is done, do the following:

1. Vary the value of η from 0.05 to 0.5 using an increment of 0.05, and report the number of epochs needed for each choice of η. Discuss how the value of η influences convergence. Generate a few learning curve plots to support your discussion.

2. Include a momentum term in the weight update with $\alpha = 0.9$ and report its effect on the speed of training for each value of η.