UCSB, Physics 129AL, Computational Physics: Section Worksheet, Week 9B

Zihang Wang (UCSB), zihangwang@ucsb.edu

March 6, 2025

Section Participation and Submission Guidelines

Section attendance is required, but you do not need to complete all the work during the section. At each section, the TA will answer any questions that you might have, and you are encouraged to work with others and look for online resources during the section and outside of sections. Unless otherwise stated, the work will be due one week from the time of assignment. The TA will give you 1 point for each task completed. You can see your grades on Canvas.

We will use GitHub for section worksheet submissions. By the due date, you should have a single public repository on GitHub containing all the work you have done for the section work. Finally, upload a screenshot or a .txt file to Canvas with your GitHub username and repository name so the TA knows who you are and which repository you are using for the section.

Remember: talk to your fellow students, work together, and use GPTs. You will find it much easier than working alone. Good luck! All work should be done in the Docker container, and don't forget to commit it to Git!

Task 1: Multilayer Perceptron (MLP)

In the class, we give an simple example of the fully connected MLP. You are ask to construct a fully connected 4-layer MLP that solves classification problems,

$$\hat{y}_i = T(X_i) = \theta_0 + \theta_1 X_i.$$

By the end of this work, you should have a functional forward and backward structure, and you are asked to train this network in Task 3 with the MNIST dataset and benchmark with MLP in PyTorch.

A) Warm up

Draw the forward and backward computational **directed graphs** for the following fully connected neural network, In each MLP graph, write down the

Input Layer	1	2	3	4
Input Dimension	A	B	C	D
Number of Neurons	6	4	3	2

Table 1: 4 layer neural network

dimension for the weights and biases near each graph node.

B)

Construct a Python class for MLP. It can preform activation with both **ReLU** and **Sigmoid**. Do not use PyTorch. You should code the forward and backward propagation by hand with numpy.

One numpy function you might find very useful: np.einsum("bi,bj→bij", a,b). It preforms the outer product between a, b. This will be very important for backward propagation.

Task 2: Convolutional neural network (CNN)

A)

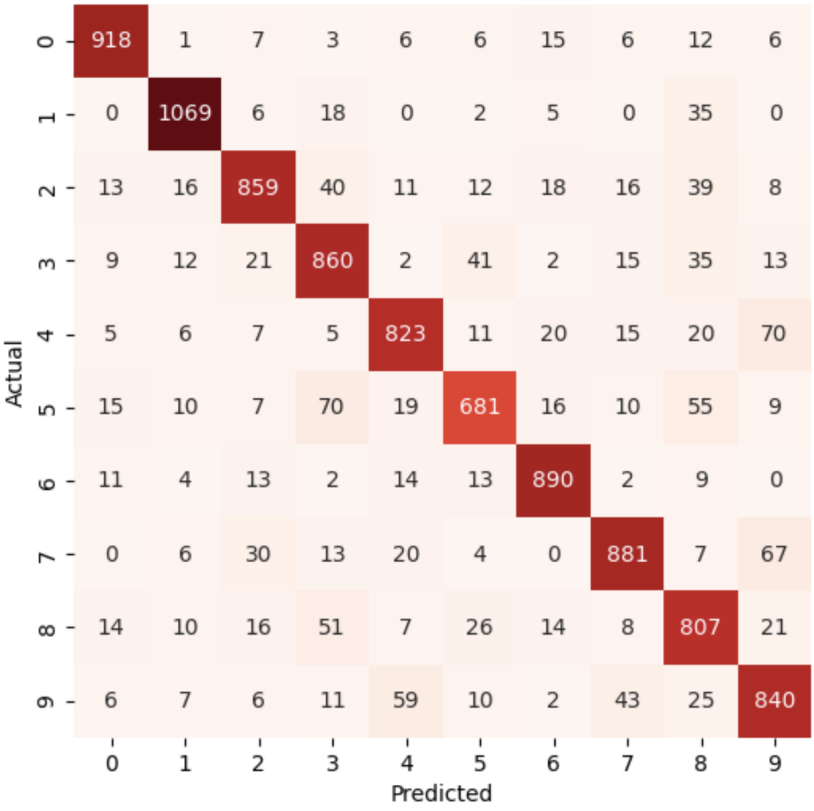
Draw the forward and backward computational **directed graphs** for a three layer CNN. Anotate those layers as what we did in the class.

B)

Construct a Python class for CNN. It can preform activation with both **ReLU** and **Sigmoid**. You can reuse the MLP code for the fully connected layer.

Task 3: MNIST dataset: Traning and validation of MLP and CNN

Train both your MLP and CNN in the previous tasks, and plot the confusion matrix, and convergence as a function of epoch. Do the training with 80-20 train-test split. The MNIST dataset can be downloaded in the course site. You should at least do better than that I have (confusion matrix),


Actual
Predicted
0 1 2 3 4 5 6 7 8 9
0 918 1 7 3 6 6 15 6 12 6
1 0 1069 6 18 0 2 5 0 35 0
2 13 16 859 40 11 12 18 16 39 8
3 9 12 21 860 2 41 2 15 35 13
4 5 6 7 5 823 11 20 15 20 70
5 15 10 7 70 19 681 16 10 55 9
6 11 4 13 2 14 13 890 2 9 0
7 0 6 30 13 20 4 0 881 7 67
8 14 10 16 51 7 26 14 8 807 21
9 6 7 6 11 59 10 2 43 25 840