Homework # 3: Deep Neural Networks

Due: Friday, April 9, 2021, 11:59 p.m.
Total Points: 100

This assignment is designed to give you practical programming experience with deep neural networks. **Please carefully read all instructions below and also periodically check Piazza for updates.**

You have been assigned a partner to work with on this homework assignment (see end of document). A spreadsheet with student contact information is on the Syllabus page of Canvas. Use this information to contact your partner and to establish your working relationship. Submit a post to Piazza (to Instructors) if you have any issues connecting or working with your partner.

Each individual within the pairing should submit a solution to the assignment, to avoid miscommunication issues with your partner. Be sure to specify in the file, whose submission should be graded, as only one will be graded for the pairing. The submitted homework must include your names, as well as all resources that were used to solve the problem (e.g. web sites, books, research papers, other people, etc.). Academic honesty is taken seriously; for detailed information see Indiana University Code of Student Rights, Responsibilities, and Conduct. **Please write a post to Piazza if you are considering using Python libraries that have not been discussed in class.**

Your assignment must be submitted by committing your code to your IU Github repository before the deadline. Please create a folder called 'hw3' (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, along with any special instructions for running the program. **Be sure to run the file before committing, so that we can directly see your results**. Programs that fail to run will not be graded and will result in a zero.

Late submission policy: We expect all work to be finished and submitted on time. However, we do understand that there might be times when something unexpected comes up which delays you. Hence, as a late submission policy, we will allow late submissions up to 3 days late, each day carrying a 5 point penalty. If you submit 70 hours after the original deadline, the highest possible score for that assignment will be 85 points, out of 100. If the assignment is submitted 1 minute after the deadline, then a 5 point penalty will be assessed. Late assignments submitted at 72:01 hours after the original deadline will not be accepted. Please, make the arrangements to start and finish your assignments early. This means you should push to Github well before the time stated on the homework. This means you must start the homework as soon as you can and continually commit.

Question 1. [50 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. Use a network where the input has 4 binary 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 as the activation function for all units.
- Randomize the order of samples for each epoch, using stochastic gradient descent to update the weights.

After 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 for each choice of η. Also generate learning curve plots for each case. Discuss how the value of η influences convergence.

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.

Question 2. [50 POINTS]

You must submit 4 separate files for this problem (see details below). Using PyTorch, implement a version of the CNN network architecture that was discussed in class, to train and test a handwritten digit recognition system. You can read the paper "Backpropagation Applied to Handwritten Zip Code Recognition" by LeCun et al. 1989 for more details, but your architecture will not follow exactly what was mentioned in the paper. Use the MNIST dataset to train and test the system. Be sure to divide the data into a single training, validation and testing set. Note that you do not need to resize the inputs to size 16×16.

The baseline system should use the following: Glorot initialization, ReLU activations, mini-batch stochastic gradient descent with momentum ($\beta = 0.9$), model selection and a cross-entropy loss function. Use a learning rate scheduler to adjust the learning rate by 10% every 10 epochs, starting with a learning rate of 0.05. **You are encouraged to use UITS Carbonate Deep Learning cluster to develop your system, since they have GPUs which will improve the training process.** Generate learning curves for the validation and training set. Discuss whether this baseline system overfits, underfits or reasonably fits the validation data. Test this baseline system with the testing data and report the accuracy and show a confusion matrix. **Submit your solution to this part of the problem in a Jupyter notebook document named: baseline.ipynb**.

Separately incorporate the following changes to the baseline model (e.g. do not do dropout and RMSprop at the same time), and generate learning curves along with a confusion matrix for the testing set. **Create and submit 3 separate Jupyter notebook files, one for each update to test (e.g., baseline-dropout.ipynb, baseline-batch.ipynb, baseline-optim.ipynb)**. Be sure to summarize what is happening in the curves due to each change and to discuss how the changes impact testing performance

1. Add dropout using drop rates of 0.25, 0.5 and 0.75, respectively
2. Incorporate batch normalization before each convolutional hidden layer.
3. Separately train using RMSProp, ADAM, and Nesterov optimizers