Submission format

Submit a single PDF-version report for all problems, including explanations, results, codes (if needed), and discussions. Other formats are not acceptable.

Problem 1. Neural Network Optimizer

Review lecture 9 and answer the following questions.

In this problem, you are asked to train and test a multilayer perceptron (MLP), i.e., a fully connected neural network, for **entire** MNIST handwritten digit dataset. The network structure is **784-200-50-10**, where 784 means the input layer has 784 input neurons. This is because each image in MNIST dataset is 28x28 and you need to stretch them to a length-784 vector. 200 and 50 are the number of neurons in hidden layers. 10 is the number of neurons in output layer since there are 10 types of digits. Use the **softmax** as the output layer. For other hyperparameters, like activation functions, dropout, batchnorm, etc., you can adjust or decide if adding them by yourself.

- a) Summarize briefly the differences among different network optimizers, including Conventional SGD, SGD with Momentum, AdaGrad, and RMSprop, as shown in Lecture 9. Also, explain briefly why the optimizer "Adam" is so popular.
- b) Implement the above MLP and train with different optimizers (Conventional SGD, SGD with Momentum, AdaGrad, RMSprop, and Adam) and step size schedulers, and report your test accuracies and compare them.

  Note: you can implement them manually or via built-in libraries.

Problem 2. Neural Network Approximation

In modern deep learning, It is very important to learn how to read and understand codes. In this problem, you are given a piece of Python code written in a Jupyter notebook (see "Intro2DL_HW4.ipynb"), which includes two neural network models, i.e., MLP and Fourier model. Both networks have the same number of parameters and learning rate.

Please understand and run the provided code to answer the following questions.

This problem is to examine your code understanding skills; you do not need to modify the code. Also, you can still answer the questions even if you're not familiar with the Fourier model.

- What is the input and output of the model? What does this code do?
- What a role does the LFF layer shown in the code play? Explain briefly.
- Run the code and describe what you observe. How do the results relate to the **Universal Function Approximation Theorem** of neural networks?