Project Part 2 [15 points] MLP, Deep Learning with CNN
(Due Thursday, Dec. 10, 11:59pm)

This part of the project contains two tasks. Task 1 is to implement a three-layer MLP from scratch and then train and test it with the given data. Task 2 is to experiment with "deep" learning (compared with the "shallow" 3-layer MLP) on the MNIST dataset.

Task 1 specific requirements: (10 points)

1. Implement a 2-n_H-1 MLP from scratch, without using any built-in library. Write your code in such a way that you may try different number of hidden nodes easily. Pick your own activation function; use MSE error as the loss. Decide on other details such as batch/mini-batch/stochastic learning, learning rate, whether to use momentum term, etc.

2. Let n_H take the following values: 2, 4, 6, 8, 10. For each case, train the network with the first 1500 training samples from each class, given in Train1.txt (for class 1) and Train2.txt (for class 2), respectively. In the data files, each line is a 2-D sample, and the number of lines is the number samples in that class. We have 2000 training samples for each class. You will use only the first 1500 training samples from each class for training the network and reserve the remaining 500 (1000 total for 2 class) as the "validation set". Train the network until the learning loss/error (J or J/n as defined in the lecture slides) for the validation set no longer decreases. Then test the network on the testing data, given in Test1.txt (for class 1) and Text2.txt (for class 2), respectively. We have 1000 testing samples for each class.

   Note: you may want to do "feature normalization", as done in Project Part 1 (i.e., using the training data to estimate the mean and STD, and then use them for normalizing all the data before any further processing).

3. Plot the learning curves (for the training set, validation set, and the test set, respectively), for each value n_H.

4. Report at which value of n_H, your network gives the best classification accuracy for the testing set. (Note: if you start learning from different random initializations, you might have different results even for a given n_H. So a more meaningful answer to this question would require you to try different initializations and then take the average.)

Task 2 specific requirements: (5 points)

In this task, we will revisit the Handwritten Digits Recognition task, using a convolutional neural network. The dataset is the MNIST dataset. We will experiment with a convolutional neural network with the following parameter settings:

(1) The input size is the size of the image (28x28).

(2) The first hidden layer is a convolutional layer, with 6 feature maps. The convolution

kernels are of 3x3 in size. Use stride 1 for convolution.

(3) The convolutional layer is followed by a max pooling layer. The pooling is 2x2 with stride 1.

(4) After max pooling, the layer is connected to the next convolutional layer, with 16 feature maps. The convolution kernels are of 3x3 in size. Use stride 1 for convolution.

(5) The second convolutional layer is followed by a max pooling layer. The pooling is 2x2 with stride 1.

(6) After max pooling, the layer is fully connected to the next hidden layer with 120 nodes and ReLu as the activation function.

(7) The fully connected layer is followed by another fully connected layer with 84 nodes and ReLu as the activation function, then connected to a softmax layer with 10 output nodes (corresponding to the 10 classes).

We will train such a network with the training set and then test it on the testing set.

You will be given some baseline code, and you are only required to experiment with the code by changing some of the hyper-parameters (the kernel size, the number of feature maps, etc.), and then report the test accuracy for at least five different settings.

Algorithm: Convolutional Neural Network
Resources: MNIST dataset, Google CoLab
Workspace: Google CoLab (see file intro_to_colab.docx for more details)
Software: Google CoLab
Language(s): Python

Your specific job:

1. Read intro_to_colab.pdf to get familiar with the platform.
2. Run the baseline code (as provided) and report the test accuracy.
3. Change the kernel size, number of the feature maps in the first and second convolutional layers, number of neurons in the fully connected layers, learning rate, etc. Try at least five different combinations of these parameters and report the test accuracy from each setting.

   Note: Change the parameters in some way that is meaningful to assist your understanding of the behavior/performance of the network. For example, try to verify a hypothesis you might have about the kernel size (the bigger the better, or else?) Or, try to answer "Can we reduce the number of neurons in the fully connected layers to a very small value (like, 2 or 3) and still obtain high accuracy?" For this reason, for each of the combinations you choose to try, you must explain what you hope to explore by doing that (e.g., one or two sentences describing what you expect to see).