EECS 442 Computer Vision: HW6

Term: Fall 2017
Instructor: Jason J. Corso, EECS, University of Michigan
Due Date: 12/8 23:59 Eastern Time

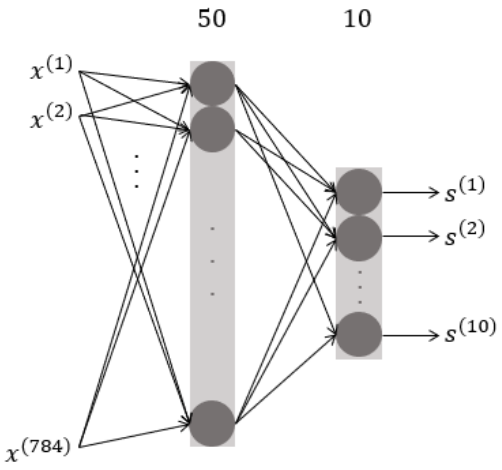
Constraints: This assignment may be discussed with other students in the course but must be written independently. Over-the-shoulder Matlab coding is strictly prohibited.

Goals: Test background material and ability to implement common tools in Matlab or Python. Deepen the understanding of neural networks

Data: You need to download the supplemental data and the helper code to complete this assignment. All paths in the assignment assume the data is on the local directory. The download has data and code; `hw6.zip`.

Problem 1 (15): Simple fully-connected NN (2-layer)

We discussed neural networks in the class. In this problem, we will implement very simple fully-connected neural network. Here is an example of 2-layer neural network.

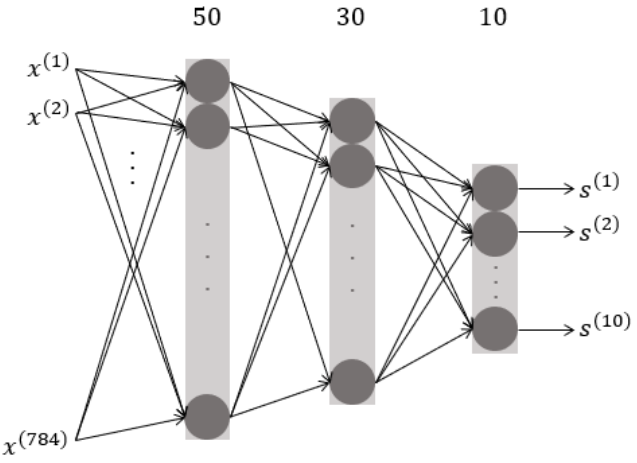


We will use this model to the MNIST dataset. As you all know, each image of MNIST is of 28 x 28, and it contains a single hand-written digit. If we vectorize it, it will be 784 vector. We will use the notation x to represent this input vector. The hidden layer consists of 50 neurons, and the output layer has 10 neurons. The final output is a score vector, of which element represents a probability of each digit. That is, $s^{(1)}$ represents the probability of x being digit 0, $s^{(2)}$ is for 1, ... $s^{(10)}$ is for 9. If we train the model with the input vectors and the corresponding labels, the model can be used to predict which digit the given image contains from 0 to 9. Now, go to `p1.m`.

1. (5) First of all, implement the sigmoid function by filling up sigmoid.m. Then, fill in the blanks of `p1.m`, with in the comment "FILL THIS UP WITH YOUR CODE". Remind that each output is just the weighted sum with the bias applied with some activation function. Include the code in the pdf report and submit your `p1.m` to the Canvas.
2. (8) Now, run the code. It will give you two plots. The first one is the loss graph, and the other one is accuracy graph. Attach your plots and report the last test accuracy.
3. (7) How many "learnable" parameters (all the weights and bias) are there in this network? Get the number and report it.

Problem 2 (15): 3-layer NN

Let's add one more hidden layer. Then, our network will be 3-layer neural network, and it will look like as below.

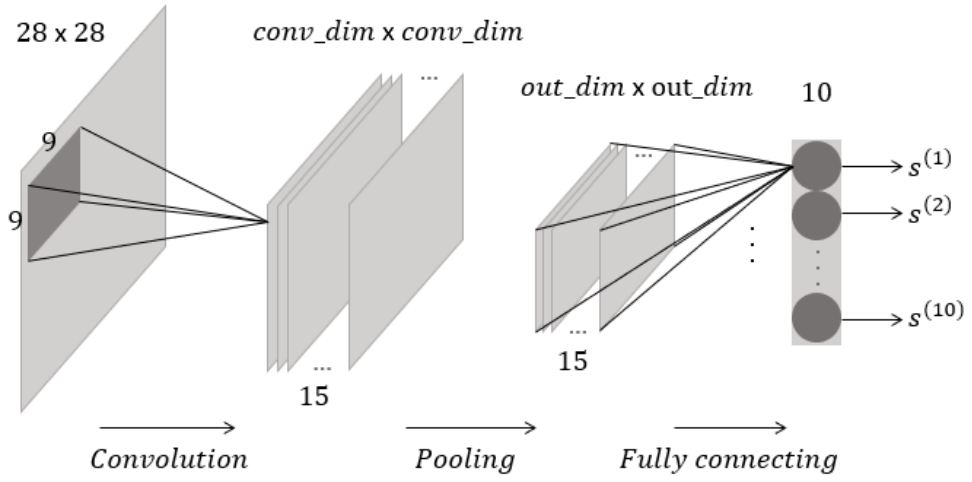


Now, let's repeat the tasks.

1. (5) Fill in the blanks of `p2.m`, with in the comment "FILL THIS UP WITH YOUR CODE". Note that we use Xaxier's scaling factor. Include the code in the pdf report and submit your `p2.m` to the Canvas.
2. (8) Run the code. It will also give you two plots, which are the loss graph and the accuracy graph. Attach your plots and report the last test accuracy.
3. (7) How many "learnable" parameters (all the weights and bias) are there in this network? Get the number and report it.

Problem 3 (25): Simple CNN

Pixels of natural images are highly spatially correlated. Also, they are usually spatially stationary, which means that they can have common basis. In this perspective, neurons are only locally connected to the previous input in the convolution layer, which is not the case of the fully connected layers. In this problem, we will consider CNN with only a single convolution layer and fully connected layer. The architecture is as below.



The detailed information of this network is in `p3.m`.

1. (5) Fill in the blanks of `p3.m`, with in the comment "FILL THIS UP WITH YOUR CODE". Include the code in the pdf report and submit your `p3.m` to the Canvas.
2. (10) Now, run the code. It will be really slow. I recommend to test your code with lower number of epochs (or lower number of training data). It will give you two plots. The first one is the loss graph, and the other one is accuracy graph. Attach your plots and report the last test accuracy.
3. (7) How many "learnable" parameters (all the weights and bias) are there in this network? Get the number and report it.

4. (3) Even if this network is really simple, it takes long time to be well trained. Describe simply why the GPU is necessary for deep neural networks.

Submission Process:

Submit a single pdf with your answers to these problems, include the code verbatim as text in the pdf. Include all plots and discussion in the pdf. Submit the pdf to Gradescope. The entry code of this course is **M5VRZV**.

Pack the original program files into one file and upload your code to Canvas. The problem description will clarify whether you should turn in the code for that problem.

Grading and Evaluation:

The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given for both paper and Matlab questions. For Matlab questions, if the code does not run, then limited or no credit will be given.