MATH 156S HOMEWORK 7

Due May. 21

Exercise 1. Reproduce figures similar to Figure 38 in LN using (1) multiclass logistic regression; and (2) naive Bayes classifier. Conclude that whether these algorithms are able to learn shift-invariant features. (Ref: Jupyter Notebook – 'Classifying non-aligned MNIST images')

Exercise 2. Consider the CNN architecture shown in Figure 40 in LN. Suppose we use the following general choices of convolutional and pooling layers:

Conv1: n_1 filters of shape $1 \times r \times r$ with stride s_f
Conv2: n_2 filters of shape $n_1 \times r \times r$ with stride s_f
maxpool (both): $d \times d$ maxpool with stride s_p

Also assume that input image has dimension $1 \times a \times b$. Then compute the dimension of the feature vector that goes into the first dense layer as a function of $n_1, n_2, r, d, s_f, s_p, a, b$.

For the following three exercise, use the following CNN architecture:

$$\text{Input} \to \underbrace{\text{Conv1}}_{\text{filter1}\in\mathbb{R}^{n_1\times 1\times 5\times 5}} + \text{ReLU} + \text{maxpool} \to \underbrace{\text{Conv2}}_{\text{filter1}\in\mathbb{R}^{n_2\times n_1\times 5\times 5}} + \text{ReLU} + \text{maxpool} \tag{1}$$

$$\to \underbrace{\text{Dense1}}_{W1} + \text{ReLU} \to \underbrace{\text{Dense2}}_{W2} + \text{Softmax} \tag{2}$$

$$\to \text{Output} \tag{3}$$

The paramters we need to optimize are n_1 filters of shape 1×5 in the first convolutional layer, n_2 filters of shape $n_1 \times 5 \times 5$ in the second convolutional layer, and the two weight matrices for the dense layers. For all pooling layer, we use 2×2 maxpool with stride 2. All convolutions use stride 1.

Exercise 3. Reproduce Figures 39 and 41 in LN with larger number of training examples: 300, 500, 1000. Compare the performance of FFNN and CNN using your plots. (Ref: Jupyter notebook on FFNN and on CNN)

Remark 1. You may work with only two digits of 0 and 1 instead of five digits to keep the computation lighter. If you have already done this exercise with five classes, no need to redo it for two classes.
Remark 2. For testing, you may only choose 100 random test images to compute the classification accuracy to save computation time.

Exercise 4. Produce a plot similar to Figure 34 in LN for the CNN model in (3), where the horizontal axis correspond to the shared number $n_1 = n_2 \in \{2, 5, 10, 20\}$ of filters in each convolutional layer. Use padding thickness 10 and keep all the other settings the same as in Figure 41. What do you think the optimal number of filters to use is? (Ref: Jupyter notebook on CNN)

Remark 1. You may work with only two digits of 0 and 1 instead of five digits to keep the computation lighter. If you have already done this exercise with five classes, no need to redo it for two classes.
Remark 2. For testing, you may only choose 100 random test images to compute the classification accuracy to save computation time.

Exercise 5. Train the CNN in (3) for MNIST classification with digits 0 and 1. Pick two images of each digit. Using the trained model, plot the image after the first and the second convolutional layers for each of the four chosen images. (Ref: Jupyter notebook on CNN) (Hint: Modify the 'predict' function in src/CNN.py so that it also returns outputs of each convolutional layer.)