Homework 4
CS584: Deep Learning
Spring 2020

This assignment is due on **Wednesday, February 19, 2020 at 4:00pm ET** on our course website. Submit your written responses as a PDF file and your Jupyter notebook as a separate file.

The MNIST database of handwritten digits is available online at `http://yann.lecun.com/exdb/mnist/`. The training set contains 60,000 labeled examples, and the test set contains 10,000 labeled examples. Each example is given by a 28×28 array with integer entries between 0 and 255, inclusively, for the intensity of each pixel in an image.

The following code may be helpful for loading the data and labels, which are not stored in a standard image format.

```
import numpy as np, matplotlib.pyplot as plt

def load_data(filename):
    contents = np.fromfile(filename, dtype=np.ubyte)
    return np.frombuffer(contents, np.uint8, offset=4*4).reshape(-1, 28, 28)

def load_labels(filename):
    contents = np.fromfile(filename, dtype=np.ubyte)
    return np.frombuffer(contents, np.uint8, offset=4*2).reshape(-1)

training_data = load_data('train-images-idx3-ubyte')
training_labels = load_labels('train-labels-idx1-ubyte')

plt.imshow(training_data[12345], cmap='binary')
training_labels[12345]
```

Problem 1. *(10 points)* For our last homework, we used backpropagation to train a classifier for the MNIST database. We used a linear layer, a softmax layer, and a cross-entropy loss function layer, which had the the update rule

$$\Delta w_{i,j} = \eta(y_j - p_j)x_i, \tag{1}$$

where $\Delta w_{i,j}$ is the update to the weights $w_{i,j}$ for the ith feature and jth output layer, η is the learning rate, $y_j = (\mathbf{e}_j)_j$ is the one-hot encoding of the label, p_j is the jth output of the softmax layer, and x_i is the ith feature.

Use TensorFlow or PyTorch to implement the "same" classifier: a linear layer, a softmax layer, and a cross-entropy loss function layer. Use mini-batch stochastic gradient descent and

whatever other options that you'd like. How does the performance of this classifier compare to your classifier from our last homework?

Problem 2. *(10 points)* For our last homework, we also implemented a classifier that used the mean pixel intensity of each image instead of the pixel intensity of each pixel in the image. This classifier did not do well. For this problem, use the mean pixel intensity as an additional feature for your classifier from Problem 1. Your new classifier should have 785 features corresponding to 784 pixel intensities and 1 mean pixel intensity. How does the performance of this classifier compare to your classifier from Problem 1?

Problem 3. *(10 points)* Use TensorFlow or PyTorch to implement a classifier for the MNIST database with multiple convolutional layers. There are *many* tutorials online that use CNNs and MNIST that you can use as examples, but use the same training options as Problem 1 so that you can compare the two networks, and only use parts that we know from class or the book. How does the performance of this classifier compare to your classifiers in Problems 1 and 2? Describe (or sketch) the architecture of your network.