Homework 3
CS584: Deep Learning
Spring 2020

This assignment is due on **Wednesday, February 5, 2020 at 4:00pm ET** on our course website. Submit your written responses as a PDF 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. *(5 points)* For $\mathbf{x} = (x_i)_{i=1}^n \in \mathbb{R}^n$, let

$$p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \text{ and } q_i = \frac{e^{x_i - \alpha}}{\sum_{j=1}^n e^{x_j - \alpha}}, \text{ where } \alpha = \max_{1 \le k \le n} x_k. \tag{1}$$

a. Show that $p_i = q_i$ for each $\mathbf{x} = (x_i)_{i=1}^n \in \mathbb{R}^n$.

b. Let $n = 2$. Plot, as a function of x_1 and x_2, where p_i and q_i do not overflow[1]. You may want to use grid search and contour plots. Which expression is better numerically?

c. Can you use logarithms to improve the range of these expressions? Why or why not?

[1]Use double precision floating point arithmetic, which is usually the default for working with real numbers in many languages and scientific computing libraries – except when using GPUs.

Problem 2. *(20 points)* Perform the following experiments in a Jupyter notebook.

a. In class, we saw how to use backpropagation to train a classifier for the MNIST database with a linear layer, a softmax layer, and a cross-entropy loss function layer using the update rule

$$\Delta w_{i,j} = \eta(y_j - p_j)x_i, \tag{2}$$

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.

Implement backpropagation to train this classifier using the pixel intensity values as features. You should separate implement functions for the linear unit, the softmax function, and the cross-entropy function.

b. Using your code from Problem 2a, train this classifier on the training data and labels. Try $\eta = 0.25$ and 100 epochs. Plot the error rates (the fraction of the data for which the classifier is incorrect) and values of the loss function for the training and test sets as functions of the number of epochs, and plot the error rates and loss for the training and test sets as functions of the elapsed training time. Your classifier should achieve an error rate of $< 15\%$ on the test data.

c. Update your code from Problem 2a to use stochastic gradient descent.

d. Using your code from Problem 2c, train this classifier on the training data and labels. Try $\eta = 0.25$ and 100 epochs with mini-batch sizes of 100 images. Plot the error rates and loss for the training and test sets as functions of the number of epochs, and plot the error rates and loss for the training and test sets as functions of the elapsed training time. Does stochastic gradient descent improve training?

Problem 3. *(5 points)* Perform the following experiments in the same Jupyter notebook.

a. Update your code from Problem 2c to use the mean pixel intensity of an image instead of the pixel intensity of each pixel in the image to replace 784 features with 1 feature.

b. Using your code from Problem 3a, train this classifier on the training data and labels. Plot the error rates and loss for the training and test sets as functions of the number of epochs, and plot the error rates and lost for the training and test sets as functions of the elapsed training time. Does this classifier perform better or worse than expected?

c. For each distinct digit, what is the mean of the mean pixel intensity? Do any digits have significantly higher or lower pixel intensities than the other digits?

d. For each digit, plot a histogram of the mean pixel intensity. You may want to plot these histograms on the same plot. Do any digits have significantly different distributions than the other digits?

e. Even if this classifier does not perform well using this feature, could this feature help improve the classifier in Problem 2?