Intro to ML and Big Data
Assignment 3

CS 4731 — Dr. Mihail
Department of Computer Science
Valdosta State University

October 3, 2019

Do not attempt a Mihail homework the night it's due.

1 Introduction

In this assignment, you will implement a deterministic classifier, 1-Nearest Neighbor, to create a written digit classifier, similar to what the United States Postal Service uses in their automated systems. You will use the MNIST (Modified National Institute of Standards and Technology) database of handwritten digits, which consists of 60,000 training images and 10,000 testing images. This is a widely used dataset throughout the machine learning community for benchmarking algorithms.

2 Background

The data points consist of 28x28 grayscale images, linearized as 1x768 row vectors. If you consider each digit as a point in a 768 dimensional Euclidean space, then a memory-based technique such as Nearest Neighbor consists simply of storing the original data points in this space. At test time, a new digit is classified by computing some distance (e.g., Euclidean) between the new datapoint and EVERY point in this space. The closest digit (in terms of *some* distance metric) is then looked up and the class of that digit is then used to classify the new digit.

3 Dataset

You will use a preprocessed dataset from: `https://github.com/daniel-e/mnist_octave/`. The code to load the dataset into Python is as follows:

```
!wget https://github.com/daniel-e/mnist_octave/raw/master/mnist.mat
import scipy.io
mat = scipy.io.loadmat('mnist.mat')
```

Figure 1: Accuracy as a function of number of training samples, with a fixed number of 200 test samples. Notice the accuracy plateaus after about 2000 training images.

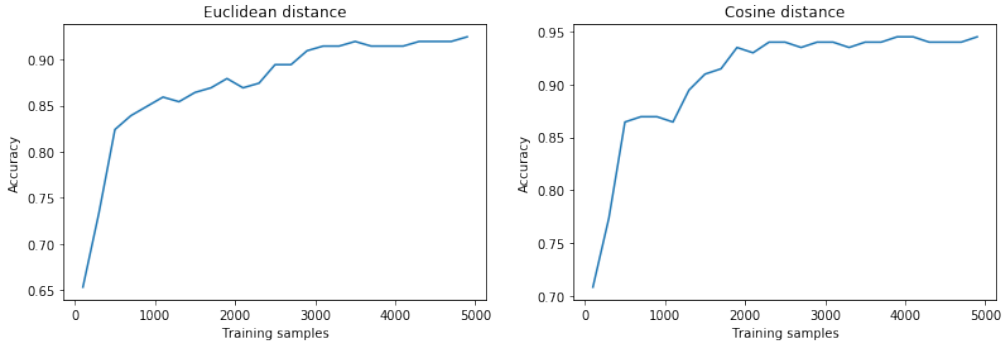
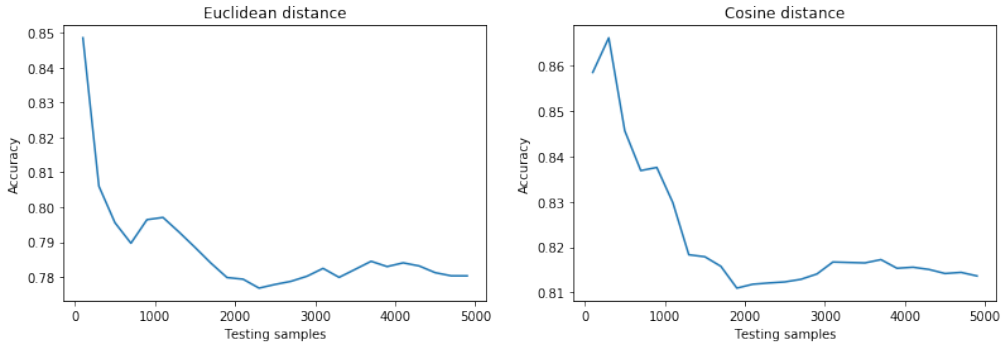


Figure 2: Accuracy as a function of number of testing samples, with a fixed number of 500 training samples. Notice the accuracy plateaus after about 2000 training images.



4 Project requirements

It is highly recommended that you modularize your code such that the following experiments can be run:

- Compute accuracy with respect to the number of training samples. See Figure 1
- Compute accuracy with respect to the number of testing samples. See Figure 2
- Compute the confusion matrix without the main diagonal and normalize such that it represents a probability distribution, as in Figure 3
- Compute likelihood of confusion for each digit as in Figure 4
- **Write-up** You will write-up your project in a LaTeX document with all the decision points along the way documented and discussed. This has to have a section titled "What I've learned" where you will discuss what you learned.

5 Due date

The project is due before the end of the day Sunday, October 13th.

Figure 3: Confusion matrix without main diagonal.

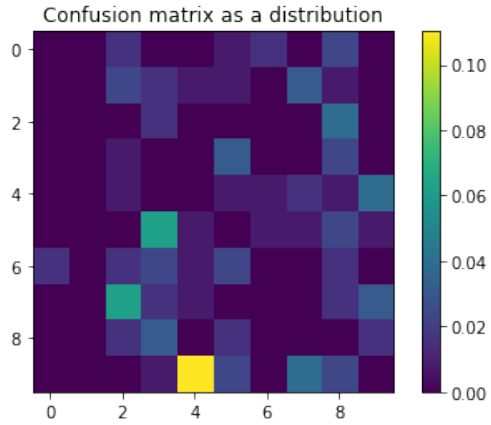


Figure 4: Per digit confusion.