ECE 203: Introduction to MATLAB Programming

Final Project: Digit Recognition

Author:
Gabriel Hoban

Instructor:
Dr. Barath Narayanan

December 8, 2019

Abstract

Through using distance-based classification on a given test data set, digit recognition can be achieved. In this project, a set of 5000 handwritten digits is loaded into MATLAB and is put through a loop comparing the Euclidean distance from a testing and training data set. Overall, the program ran with a 95% accuracy rate and only had 5 incorrect image matches.

1. Introduction

The aim of this project is to implement digit recognition of handwritten numbers in MATLAB. Initially given a data set of 5000 images containing handwritten values ranging from 0-9, MATLAB is able to classify each image based on its calculated Euclidean distance. In order to be able to train the program, the initial data set has to be split into a training set and a testing set. For this project, every 50_{th} image is used for testing and the rest of the 4900 images are used for training. After the data set is split, a nested loop for determining the Euclidean distance is created. Once the Euclidean distance has been calculated, the minimum distance for each value is determined and is connected to the corresponding label.

2. Theoretical Calculations

While there are many ways to classify an image for digit recognition, the Euclidean distance is chosen due to its simplicity. The Euclidean distance can be defined as

$$d = \sqrt{(x_2 - x)^2 + (y_2 - y_1)^2}. \quad (1)$$

In terms of the digit recognition program, the Euclidean distance equation can be rewritten to

$$d = \sqrt{(testImages(:,:,idx_1) - trainImages(:,:,idx_2))^2}, \quad (2)$$

where the length of the testing labels is idx_1 and the length of the training labels is idx_2.

2 Experimental Results

Overall, the program is a success, with the percent accuracy at 95%. As seen in Appendix A, the code uses the formula found in Equation 2 in order to classify the images. The program uses the minimum distance in the training data set in order to differentiate between different digits.

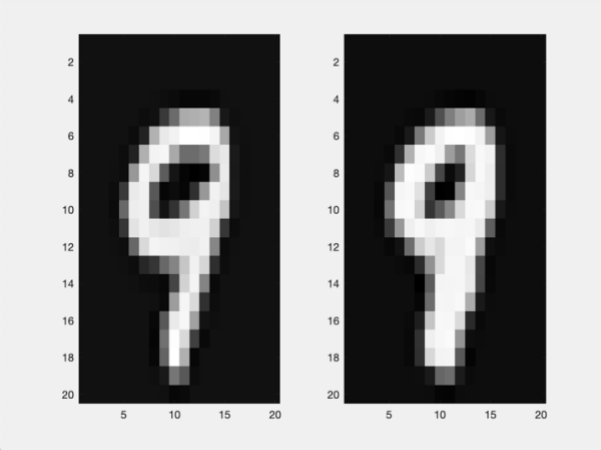

Figure 1: MATLAB plot of correct digit classification

Figure 1 shows how the training image on the left correctly classified the testing image on the right as the number 9. While using the Euclidean distance is simple, it is not perfect. As seen in Figure 2, the program failed to differentiate the number 8 on the left to the number one on the right. By comparing the testLabel and the trainLabel, it can be determined if the digit recognition is accurate.

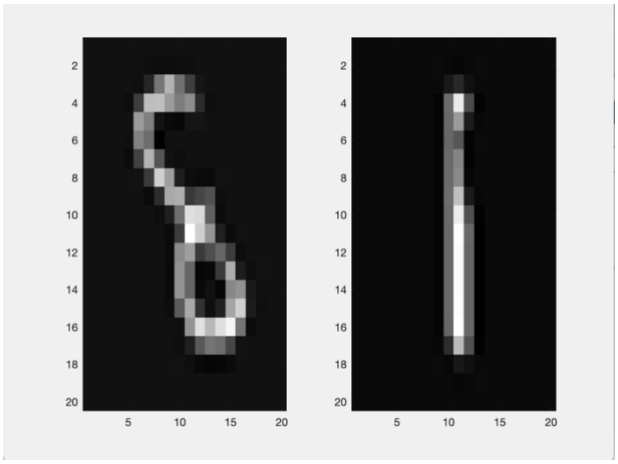

Figure 2: MATLAB plot of failed digit classification

3 Conclusion

In this project, a set of data is split into two separate groups, one with 100 images and another with the other 4900 images. It is important to point out that from finding the minimum Euclidean distance for each number, it can be matched to a separate image in the testing images. In order to increase the accuracy of the program, it would need a much larger data set of images.

APPENDIX A: MATLAB Code for Digit Recognition

```
%% Clear Workspace

close all;
clc;
clear;

%% Step 1

load Data;

%% Step 2

for x=1:500:5000
    imagesc(images(:,:,x));
    colormap(gray(256))
    title(sprintf('Index: %0.0f',x))
    pause(.3);
end

%% Step3

testImages = images(:,:,1:50:5000);
trainImages = images(:,:,rem(1:5000,50) ~= 1);

%% Step 4

testLabels = labels(1:50:5000);
trainLabels = labels(rem(1:5000,50) ~= 1);


%% Steps 5-8

correct = 0;
for idx_1 = 1:length(testLabels)
    for idx_2 = 1:length(trainLabels)
        % Euclidean distance for each image
        D = (testImages(:,:,idx_1) -
trainImages(:,:,idx_2)).^ 2;
        Distance(idx_2, 1) = sqrt(sum(D(:)));
```

```
    end
    % Determine minDistance for each image
    [minDistNum,min_idx]=min(Distance(:, 1));
    % Display train and test images
    subplot(1,2,1);
    imagesc(testImages(:,:,idx_1));
    subplot(1,2,2);
    imagesc(trainImages(:,:, min_idx));
    title(sprintf('Digit Number:
%0.0f',testLabels(idx_1)));
    % Training
    if(testLabels(idx_1)==trainLabels(min_idx))
        correct = correct+1;
    else
        fprintf('Testing image %d index %d against
training image %d index %d failed\n',testLabels(idx_1),
idx_1, trainLabels(min_idx), min_idx);
    end
    pause(0.3);
end

%% Step 9

accuracy = (correct / length(testImages)) * 100;
fprintf('\nThe overall accuracy is %d%%', accuracy);
```