Pattern Recognition Laboratories

Project 1

EE 516 - 01

Pattern Recognition

Spring 2023

Group 6

Students: Ronit Singh, Nathan Jaggers

Project Due: 04/12/23

Instructor: Dr. Zhang

Part A - Feature Analysis and Selection for Handwritten Digit Recognition

Assignment

The MNIST (Modified National Institute of Standards and Technology) dataset is a collection of handwritten digits used in OCR (optical character recognition) and is considered as one of the benchmark datasets for pattern recognition and machine learning. An image containing all training samples of digit 0 is shown below. Note that each sample has a standard size of 28x28 pixels arranged in a regular grid. Electronic copies of all ten images can be found on Canvas.

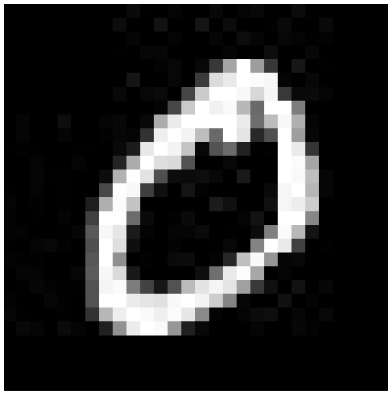
1. In this project, you will examine various features extracted from the sample digits and select two best features for digits classification of 0 and 1.
2. For each image, do the following:
    a. Load the image and convert it to a binary image.
    b. Find samples of the digit by extracting the regular grids in the binary image.
    c. Extract shape measurements (features) from each sample digit. (You can use regionprops() in MATLAB. You can also consider other derived features from these measurements.)
3. Analyze the extracted features by building and comparing the histograms. For this step you don't need to show all shape features. Choose a few.
4. Based on the results in Step 3, select two best features. Plot the training data in 2- D scatter plot. Do you think a good classifier can be built based on these two features?
5. Now find the best feature(s) to classify 0 and 2. Repeat steps 3 and 4.

Procedure

The first step in analyzing the features of the MNIST numbers dataset is importing the supplied photos into the Matlab workspace and using imread() to read them into variables. The photos given, although they look black and white are actually grayscale photos. We need our photos to be binary images so that we can later use regionprops on the number samples and extract features which differentiate the numbers and help with classification. There are a few ways to accomplish this. Ones attempted include creating a histogram and manually choosing a threshold number to then create a binary image or using built in Matlab functions to determine thresholding and binarize the image for us. The latter method proved to be less work because the imbinarize() function will take in a grayscale image, perform Otsu's method to determine the threshold and then create the binary image.

```
%read in image
im0 = imread("mnist_train0.jpg");
...
%create binary image
binary_im0 = imbinarize(im0);
```

Figure 1: Converting Images from Grayscale to Binary



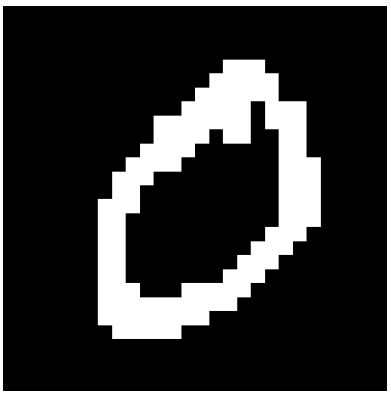


Figure 2 and 3: Grayscale and Binary 0 samples

Now, with binarized images, we can move on to the next step in extracting the features from the numbers. In each image there are multiple samples of the same number, each sample being 28 x 28 pixels. All the samples need to be isolated from each other so that way they can be fed into the regionprops function one by one. This is accomplished by using two for loops to index to the start position of the samples and using the knowledge of their size to extract them out one by one from the larger image.

```
for i = 1:28:(rows-28)
   for j = 1:28:(cols-28)
       digit = bim0(i:i+27, j:j+27);
       ...
   end
end
```

Figure 4: Indexing Digit Samples

The samples can now be analyzed by regionprops and different significant features can be explored.

We can use features of these samples to determine what class aka number they belong to. The features we want are distinct ones that only occur for specific digits. We can get an idea of what features are good representations of a number and which are distinct from each other by creating histograms of them and comparing between numbers. Once a few good candidate features are found they can be used to create scatter plots and these plots can be analyzed and used as training for a pattern recognition system that determines if a number belongs to one class or another.

Results

There are many features that regionprops detect in images. The features that we first explored to separate at least one digit from the others were eccentricity, circularity, and area. In

Figure 5, eccentricity feature measurements provided a distinct separation between the digit 1 to the digits 0 and 2. This feature is good for separating 0 and 1 or 2 and 1, but not 0 and 2, as they share similar eccentricity values.

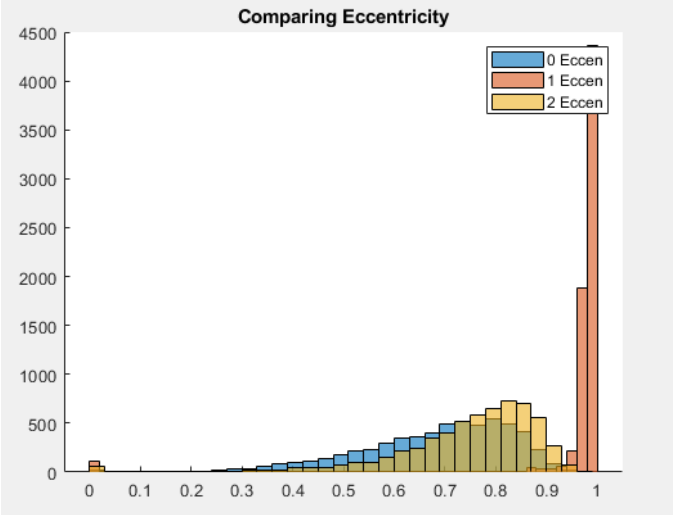


Figure 5: Eccentricity Comparison

In Figure 6, circularity is compared between all three digits. Although the three digits share similar values ranging from 0 to 1, with an outlier at 6.8, there is still a distinct difference between values for 1 and 2. The circularity of 0 lies in the middle with a lot of overlap with the other digits. It is also noted that 0 does not have the most circularity, which was interesting. Circularity is a good feature to use when detecting either a 1 or a 2.

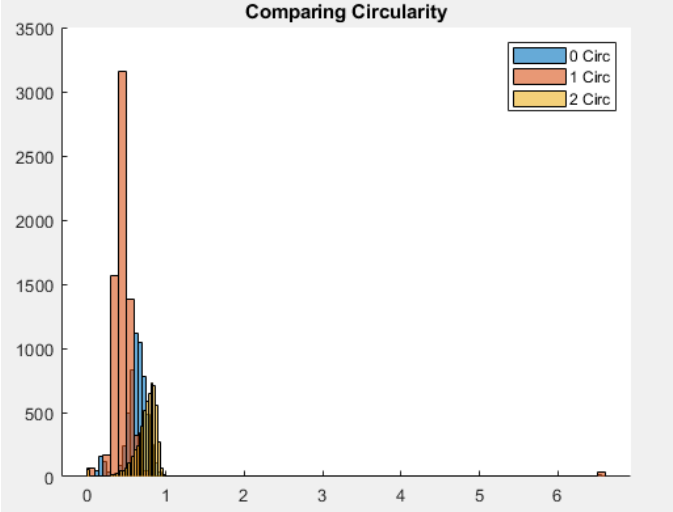


Figure 6: Circularity Comparison

Another feature used was area. As can be seen in Figure 7, 0 and 2 share similar area values with a lot of overlap, ranging from around 100 to 200. The areas of the 1 digit images were much lower, ranging from around 50 to 100. This feature is also good for detecting the difference between 1 and 0 or 1 and 2.

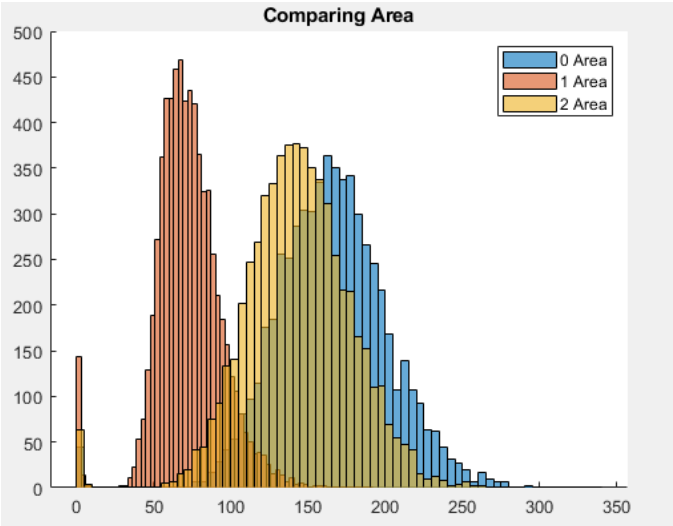


Figure 7: Area Comparison

After finding the best features for determining what digit an image contains between two digits, scatter plots were created to classify 0 and 1, and 0 and 2. Figure 10 displays the scatter plot for detecting 0 and 1. The features chosen to compare 0 and 1 were Filled Area and Minor Axis Length. Filled area was chosen because the number 0 would have much more area than the number 1. The minor axis length also provided a good distinction, with 0 having larger values for that feature. There is strong clustering of these values for each digit, with a clear separation at around 150 area and 10 axis length. This method is an adequate method to determine whether something is either a 0 or 1, although there are a few overlapping values with 10-150 area and around 15 axis length.

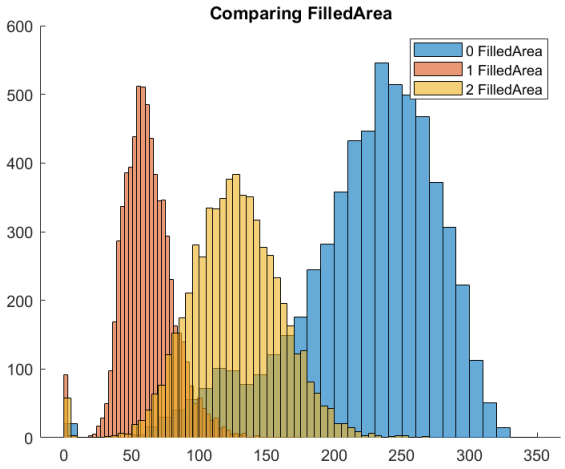


Figure 8: Filled Area Comparison

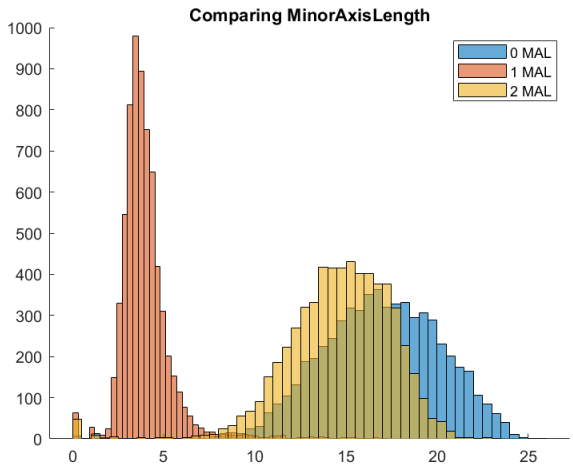


Figure 9: Minor Axis Length Comparison

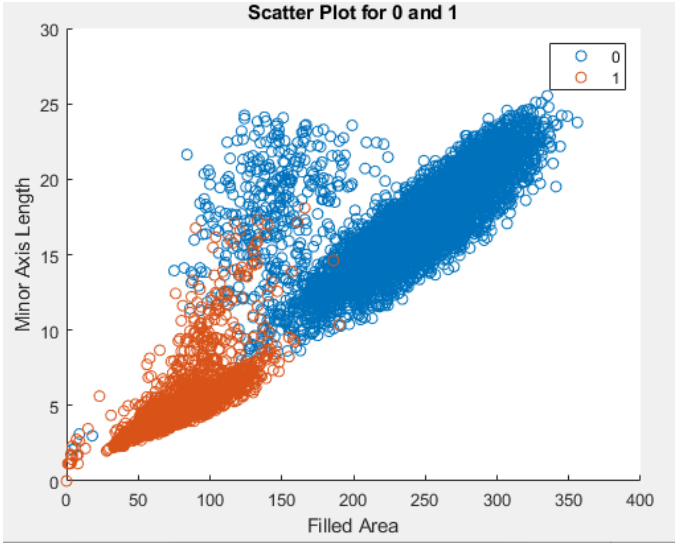


Figure 10: Scatter Plot for 0 and 1

To determine whether a digit was a 0 or a 2, the features chosen were Filled Area and Circularity. Circularity is expected to be larger for the 0 value, as can be seen in Figure 6. This data ranged from 0-1, with 2 having circularity values below 0.5 and 0 having values above 0.5 mainly. There is some overlap, but most of the data is separated towards 0 and 1 circularity. The

filled area for each digit was also somewhat different to each other. This feature had much more overlap, with values ranging from 50 - 250 for 2 and 125 to 350 for 0. Due to the difference in circularity, this pair of features worked well in finding a 0 or a 2.

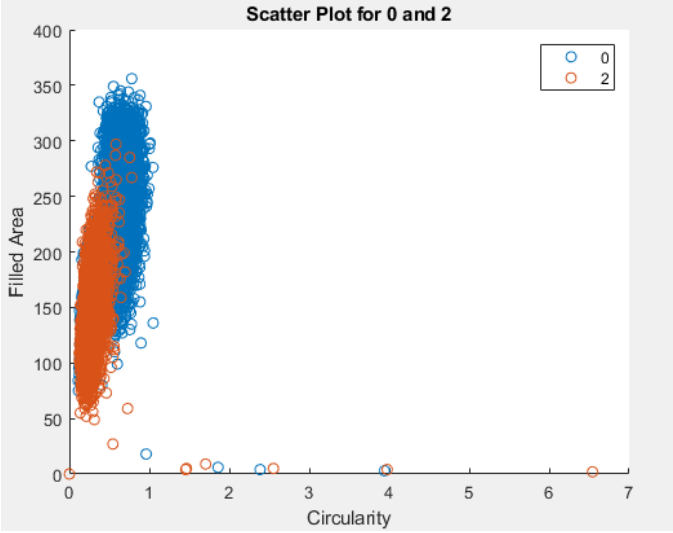


Figure 11: Scatter Plot for 0 and 2

Reflection

Nathan Jaggers

This lab is an excellent introduction to Matlab and a valuable exercise showing what is required for classification and some of the steps that must be completed prior to that. I feel I got a lot more comfortable with histograms during this lab and I see how they are crucial when deciding what will be used to differentiate a subject into which class it matches best. The process of finding features that are distinct can be very difficult. For many of the fields in region props, there were several times where good feature candidates to differentiate 0 and 1 were found. However finding ones to differentiate 0 and 2 were much more difficult. This is a good demonstration to show how what works for one set of classifications wont work for others.

Ronit Singh

I learned a lot about regionprops and how to read the data that this function provides. There are many features that this function can detect from an image, and finding the best features to detect each digit required testing and experimenting. I also learned how to use scatter plots to display different data together to get a better visualization of the data. Finding a clear distinction with features to have no overlap was hard, but there is modern software that accomplishes this with all digits. I wonder if they use template matching or have better features that would help determine what digit is what.

Teamwork

Nathan Jaggers

I worked on taking the initial code that Ronit developed, refactoring it into functions and testing out different features for significance. I also worked on the procedure part of the report.

Ronit Singh

I worked on finding each digit in each image and extracting the features from each digit, and the corresponding report sections.

Appendix

```
%% Project 1
%
% EE 516 - Pattern Recognition
% Spring 2023
%
% Group 7: Nathan Jaggers, Ronit Singh
%
%% Part A
close all;
clear;
clc;
%%
%read in image
im0 = imread("mnist_train0.jpg");
im1 = imread("mnist_train1.jpg");
im2 = imread("mnist_train2.jpg");
%%
%create binary image and show results
binary_im0 = imbinarize(im0);
%%imshow(binary_im0);
binary_im1 = imbinarize(im1);
%%imshow(binary_im1);
binary_im2 = imbinarize(im2);
%%imshow(binary_im2);
%%
features_0 = get_features(binary_im0);
features_1 = get_features(binary_im1);
features_2 = get_features(binary_im2);
%%
featarry_0 = make_feat_array(features_0);
featarry_1 = make_feat_array(features_1);
featarry_2 = make_feat_array(features_2);
%%
figure;
hold on
histogram(featarry_0(:,1));
histogram(featarry_1(:,1));
histogram(featarry_2(:,1));
title("Comparing Area");
legend("0 Area","1 Area","2 Area");
hold off
figure;
hold on
histogram(featarry_0(:,2));
histogram(featarry_1(:,2));
histogram(featarry_2(:,3));
title("Comparing Circularity");
legend("0 Circ","1 Circ","2 Circ");
hold off
```

```
figure;
hold on
histogram(featarry_0(:,3));
histogram(featarry_1(:,3));
histogram(featarry_2(:,3));
title("Comparing Eccentricity");
legend("0 Eccen","1 Eccen","2 Eccen");
hold off
%%
figure;
hold on
num = 5;
histogram(featarry_0(:,num));
histogram(featarry_1(:,num));
histogram(featarry_2(:,num));
title("Comparing Test");
legend("0 test","1 test","2 test");
hold off
%%
% Create scatter plot 0 and 1
figure;
hold on;
first = 6;
second = 7;
scatter(featarry_0(:,first),featarry_0(:,second));
scatter(featarry_1(:,first),featarry_1(:,second));
title("Scatter Plot for 0 and 1");
xlabel('Filled Area');
ylabel('Minor Axis Length');
legend("0","1");
hold off
%%
% Create scatter plot 0 and 2
figure;
hold on;
first = 2;
second = 6;
scatter(featarry_0(:,first),featarry_0(:,second));
scatter(featarry_2(:,first),featarry_2(:,second));
title("Scatter Plot for 0 and 2");
xlabel('Circularity');
ylabel('Filled Area');
legend("0","2");
hold off
```

FUNCTIONS

```
function features = get_features(bw_mnist_im)
%GET_FEATURES get all regionprop features from mnist training pictures
%   input photo must be a binary image
% go through 28x28 bit samples one by one
i = 1;
total_samples = (size(bw_mnist_im,1)/28)*(size(bw_mnist_im,2)/28);
```

```
features = cell(total_samples,1);
for m = 1:28:size(bw_mnist_im,1)
   for n = 1:28:size(bw_mnist_im,2)
       sample = bw_mnist_im(m:m+27, n:n+27);
       %row = m/28 + 1;
       %col = n/28 + 1;
       %imshow(sample);
       features{i} = regionprops(sample,"all");
       %features{i} = regionprops(sample,"Area", "Circularity",
"Eccentricity","Orientation");
       i = i + 1;
   end
end

function featureArray = make_feat_array(features)
%MAKE_FEAT_ARRAY take features, select certain ones and put into an array
%   Detailed explanation goes here
count = 1;
featureArray = zeros(length(features),length(fieldnames(features{1})));
for index = 1 : length(features);
   if (length(features{index}) > 0)
       featureArray(count,1) = features{index}.Area;
       featureArray(count,2) = features{index}.Circularity;
       featureArray(count,3) = features{index}.Eccentricity;
       featureArray(count,4) = features{index}.EquivDiameter;
       featureArray(count,5) = features{index}.EulerNumber;
       featureArray(count,6) = features{index}.FilledArea;
       featureArray(count,7) = features{index}.MinorAxisLength;
       count = count + 1;
   end
end
end
```