Lab 3: Image Classification

A Report Submitted in Partial Fulfillment
of the Requirements for SYDE 372

David Kadish, 20176757
Zhao Peng, 20326604
Matt Stewart, 20205320

Faculty of Engineering
Department of Systems Design Engineering

April 3, 2009.

Course Instructor: Professor P. Fieguth

1

Introduction

2

Implementation and Results

2.1 Labelled Classification

The MCID classifier was developed for each of the feature matrices for $n = 2$, $n = 8$, and $n = 32$. The provided feature matrices, **f2**, **f8**, and **f32** were used as learning data sets to teach the MCID classifiers the location and shape of the clusters. This trained MCID classifier was then applied to the test data sets **f2t**, **f8t**, and **f32t**. The performance of the classifier was assessed for each quantity of features using a confusion matrix. The total probability of error is also shown for each case, shown in Figure 2.1.

As we can see from the table on page 3, the probability of a misclassification drastically increases as we decrease our block size. With many features included, the resulting classifier performance is perfectly acceptable and quite comparable with results obtained in previous labs. However, as we reduce the block size down to only 2x2, the resulting classifier performance is abysmal. First, consider the data set based on 32x32 feature blocks. The classifier performs quite well on this data set. The confusion matrix shows relatively few misclassifications, The large majority of the classifications appearing on the diagonal of the confusion matrix, signifying a majority of correct classifications. This observation is perhaps best summarized by considering the associated probability of error. The overall probability of having a classification error is about $14 Wewillnextjumptothecaseinwhichonly 2x2 featureblocksareusedtodeveloptheMCIDclassifier.$

Table 2.1: Summary of Error analysis for $n = 2, 8, 32$ with confusion matrices $M_{confusion}$, the probability of error for each image $P(\varepsilon|i)$, and the total probabiliy of error $P(\varepsilon)$ for each feature matrix

	$M_{confusion}$	$P(\varepsilon\|i)$	$P(\varepsilon)$
$2x2$ Feature Blocks	$\begin{bmatrix} 1 & 0 & 0 & 2 & 3 & 0 & 1 & 4 & 5 & 0 \\ 0 & 7 & 4 & 0 & 2 & 1 & 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 3 & 0 & 0 & 1 & 1 & 7 & 1 \\ 0 & 3 & 0 & 1 & 4 & 0 & 1 & 1 & 6 & 0 \\ 1 & 0 & 0 & 2 & 4 & 0 & 0 & 5 & 4 & 0 \\ 0 & 2 & 3 & 2 & 0 & 2 & 1 & 2 & 3 & 1 \\ 0 & 5 & 4 & 0 & 2 & 0 & 1 & 0 & 4 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 9 & 3 & 2 \\ 2 & 1 & 2 & 4 & 2 & 0 & 2 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 14 \end{bmatrix}$	$\begin{bmatrix} 0.9375 \\ 0.5625 \\ 1.0000 \\ 0.9375 \\ 0.7500 \\ 0.8750 \\ 0.9375 \\ 0.4375 \\ 0.8750 \\ 0.1250 \end{bmatrix}$	0.7438
$8x8$ Feature Blocks	$\begin{bmatrix} 9 & 0 & 0 & 3 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 10 & 2 & 0 & 0 & 1 & 3 & 0 & 0 & 0 \\ 0 & 1 & 4 & 2 & 0 & 4 & 3 & 0 & 2 & 0 \\ 1 & 0 & 0 & 12 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 5 & 0 & 0 & 1 & 9 & 0 \\ 0 & 0 & 4 & 3 & 0 & 2 & 4 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 & 0 & 8 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 10 & 0 & 4 \\ 0 & 0 & 1 & 1 & 3 & 0 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 13 \end{bmatrix}$	$\begin{bmatrix} 0.4375 \\ 0.3750 \\ 0.7500 \\ 0.2500 \\ 0.6875 \\ 0.8750 \\ 0.6250 \\ 0.3750 \\ 0.3125 \\ 0.1875 \end{bmatrix}$	0.4875
$32x32$ Feature Blocks	$\begin{bmatrix} 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 6 & 1 & 0 & 7 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 5 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 15 \end{bmatrix}$	$\begin{bmatrix} 0.2500 \\ 0.0000 \\ 0.0625 \\ 0.0000 \\ 0.0625 \\ 0.5625 \\ 0.0625 \\ 0.3125 \\ 0.0625 \\ 0.0625 \end{bmatrix}$	0.1438

8 feature case provides a middle ground between the poor performance of the 2x2 feature blocks and the 32x32 feature blocks. As shown in Table 2.1, the 8x8 probability of misclassification is an improvement on the 2x2 case, but far from being as good as the 32x32 case. The 8x8 feature blocks might be used in an instance which there

Figure 2.1: K-means clusters, colour-coded with prototypes outlined in black

Figure 2.2: Prototypes from multiple runs of the k-means clustering algorithm (distinct prototype shape and colour for each run)

was some cost associated with feature block size, making the 32x32 blocks too expensive to use. In this type of situation, the 8x8 blocks might provide an acceptable compromise between cost and performance.

2.2 Image Classification and Segmentation

2.3 Unlabelled Clustering

The classification of unlabelled data is completed using the code presented in Section A.1. MATLAB actually provides a number of built in functions for classifying data by k-means and fuzzy k-means which are used extensively in this implementation.

MATLAB's `kmeans` and `fcm` functions are used to cluster the provided data. Much of the remainder of the code in Section A.1 is devoted to creating the plots in Figures 2.3, 2.3 and 2.3.

Figure 2.3: Prototypes from fuzzy k-means clusters. Data points are shaded based on probability of belonging to a cluster.

3

Conclusions

Appendix A

Code

A.1 q5.m

```
%% Loading and setup
load feat.mat; % Load the feature data
x = f32(1:2,:); % Extract x_ij from f32

%% Perform clustering
[labels,c] = kmeans(x',10,'onlinephase','off');
cluster = cell(10,1);

for i=1:10
    match = labels == i;
    x_var = x(1,:);
    y_var = x(2,:);
    cluster{i} = [x_var(match); y_var(match)];
end

%% Plotting
figure;
colours = {'blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black', [1 0.5 0], [0.5 0 0.5],
    [0 0.5 0.5]};
for i=1:10
    cluster{i};
    scatter(cluster{i}(1,:),cluster{i}(2,:),'.', 'MarkerEdgeColor', colours{i}); % Plot the
        original data
    hold on;
    scatter(c(i,1)',c(i,2)','filled', 'MarkerEdgeColor', 'black', 'MarkerFaceColor', colours{i});
    hold on;
end
hold off;

%% FUZZY K-MEANS
% Set-up
figure;
marks = {'+','o','*','x','s','d'};

%Plot
scatter(x(1,:),x(2,:),'.','MarkerEdgeColor',[0.5 0.5 0.5]);
```

```
hold on;
for i=1:6
    [centres, U] = fcm(x',10); % Run MATLAB's fuzzy c-means calculator
    scatter(centres(:,1)',centres(:,2)', marks{i}, 'filled','MarkerEdgeColor', 'black','
        MarkerFaceColor', colours{i},'SizeData',10^2);
    hold on;
end
hold off;

% Contour Plot
figure;
colours = [
    0 0 0;
    1 0 0;
    0 1 0;
    0 0 1;
    1 0 1;
    0 1 1;
    0.5 0.5 0.5;
    0.5 0 0;
    1 0.62 0.40;
    0.49 1 0.83
];

[centres, U] = fcm(x',10); % Run MATLAB's fuzzy c-means calculator

colourmap = U'*colours;
for i=1:160
    scatter(x(1,i),x(2,i),'s','filled', 'MarkerEdgeColor', colourmap(i,:), 'MarkerFaceColor',
        colourmap(i,:),'SizeData',3^2);
    hold on
end


for i=1:10
    scatter(centres(i,1)',centres(i,2)','filled','MarkerEdgeColor', 'black','MarkerFaceColor',
        colours(i,:),'SizeData',10^2);
    hold on;
end
```