Fast Prototyping 3
Linear Discriminant Analysis

Name: Rohan Patel
SFSU ID#: 917583698

Introduction:

Linear Discriminant Analysis (LDA) is a popular method that is commonly used in pattern recognition and machine learning. It is generally used as a dimensionality reduction technique in the preprocessing step of pattern classification. The goal is to project a dataset onto a lower dimensional space with good class separability in order to avoid overfitting, ‘curse of dimensionality’ and also reduce computation.
The main objective of this fast prototyping exercise is to perform LDA on a given set of 32X32 male and female facial images and train a statistical model that can classify any given test image as male or female.

Methodology:

1) Load the initial 32x32 image dataset containing male and female facial images.
2) Perform PCA on this dataset.
3) Select top K Principal Components that cover majority of the variance.
4) Project all data points to the new eigen feature space.
5) Calculate the between class scatter Sb and within class scatter Sb for both male and female.
6) Solve a generalized eigenvalue problem with Sw and Sb.
7) This results in a single Principal Component.
8) Use this PC to compute the slope and intercept of the linear discriminant function.
8) Load the test set to test the model and determine model accuracy.

Matlab Code:

```
% Loading the data
img_files1 = dir('FaceClassification_Data/Male/*.TIF');
img_files2 = dir('FaceClassification_Data/Female/*.TIF');
M1 = length(img_files1);
M2 = length(img_files2);
M = M1 + M2;

for i = 1:M1
    img = imread(strcat('FaceClassification_Data/Male/', img_files1(i).name));
    img = img(:);
    X(:,i) = img;
end
for i = 1:M2
   img = imread(strcat('FaceClassification_Data/Female/', img_files2(i).name));
   img = img(:);
   X(:,(i+M1)) = img;
end

% Calculating avg matrix
avg = mean(X,2);

% removing avg from training matrix X
X = double(X);
norm_matrix = [];
for i = 1:M
    norm_matrix(:, i) = X(:, i) - avg;
end

% covariance
cov_matrix = norm_matrix*norm_matrix';
%disp(cov_matrix)

% eigen values
[V, D] = eig(cov_matrix);
evalues = diag(D);

[~,indices] = sort(evalues,'descend');

% Selecting top K eigenvalues
K = 16;
F = zeros(32*32, K);
```

```
for i = 1:K
    F(:, i) = V(:, indices(i));
end

F = F';

% Project all data points to FX
FX = zeros(K, M);
for i = 1:M
            FX(:, i) = F*(X(:,i) - avg);
end

% Compute Sw and Sb from FX
mean_m = mean(FX(:,1:M1),2);
mean_fm = mean(FX(:,(M1+1):M),2);
mean_FX = ((M1*mean_m) + (M2*mean_fm))/M;

Sb = (M1*(mean_m - mean_FX)*(mean_m - mean_FX)') + (M2*(mean_fm -
mean_FX)*(mean_fm - mean_FX)');

sw_m = zeros(K, M1);
sw_fm = zeros(K, M2);
for i = 1:M1
    sw_m(:,i) = FX(:,i) - mean_m;
end
for i = 1:M2
    sw_fm(:,i) = FX(:,(i+M1)) - mean_fm;
end
sw_m = sw_m*sw_m';
sw_fm = sw_fm*sw_fm';

Sw = sw_m + sw_fm;

%  Perform eigendecomposition & compute PC for new feature space
[V1, D1] = eig(Sb, Sw);

[c, ind] = sort(diag(D1), 'descend');
v = V1(:, ind(1));

w = v'*F; %discriminant slope
b = v'*(mean_m + mean_fm)/2; %discriminant intercept

% Testing
```

```
img_m = dir('FaceClassification_Data/Male/*.TIF');
img_fm = dir('FaceClassification_Data/Female/*.TIF');
len_m = length(img_files1);
len_fm = length(img_files2);
tp = 0;
fp = 0;
tn = 0;
fn = 0;

fprintf('Male:\n');
for i = 1:len_m
   test_m = double(imread(strcat('FaceClassification_Data/Male/', img_m(i).name)));
   test_m = test_m(:);
   result = w*(test_m - avg) - b;
   if result < 0
      tp = tp + 1;
   else
      fp = fp +1;
   end
   fprintf('Filename: %s ---> Result: %d \n', img_m(i).name, result)
end
fprintf('Female:\n');
for i = 1:len_fm
  test_fm = double(imread(strcat('FaceClassification_Data/Female/', img_fm(i).name)));
  test_fm = test_fm(:);
  result = w*(test_fm - avg) - b;
  if result > 0
      tn = tn + 1;
   else
      fn = fn +1;
  end
  fprintf('Filename: %s ---> Result: %d \n', img_fm(i).name, result)
end

fprintf('\n*********Classification Report*******')
fprintf('\nTrue Positive: %d', tp)
fprintf('\tFalse Positive: %d', fp)
fprintf('\nTrue Negative: %d', tn)
fprintf('\tFalse Negative: %d \n', fn)

precision = tp/(tp+fp);
recall = tp/(tp+fn);
f_1 = 2*((precision*recall)/(precision+recall));
```

```
accuracy = (tp+tn)/(tp+tn+fp+fn);

fprintf('\nPrecision\tRecall\tF-1 Score\tAccuracy\n');
fprintf('%s\t\t%s\t%s\t\t%s\n', num2str(precision,'%.2f'), num2str(recall,'%.2f'),
num2str(f_1,'%.2f'), num2str(accuracy,'%.2f'));

% few plots
h1 = figure;
subplot(1,2,1);
plot(evalues);
title('PCA Eigenvalues');

subplot(1,2,2);
plot(diag(D1));
title('LDA Eigenvalues');

h2 = figure;
colormap('gray');
subplot(1, 3, 1);
imagesc(reshape(avg, 32, 32));
title('Average Face');

subplot(1,3,2);
imagesc(reshape(mean(X(:,1:M1),2), 32, 32));
title('Mean Male');

subplot(1,3,3);
imagesc(reshape(mean(X(:,(M1+1):M),2), 32, 32));
title('Mean Female');

h3 = figure;
plot(w);
title('Discriminant Slope');

h4 = figure;
colormap('gray');
subplot(1, 2, 1);
hist(FX(:,1:M1));
title('Distribution of Male samples');

subplot(1, 2, 2);
hist(FX(:,(M1+1):M));
title('Distribution of Female samples');
```

Results:

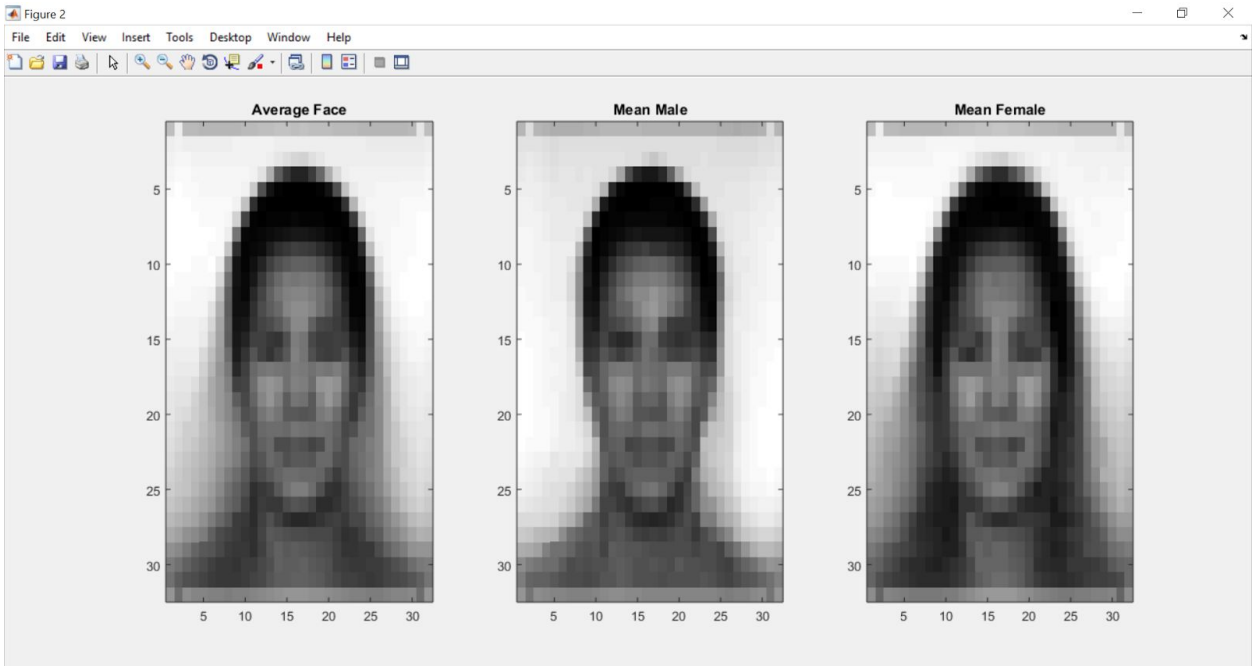

Fig 1: Mean Images

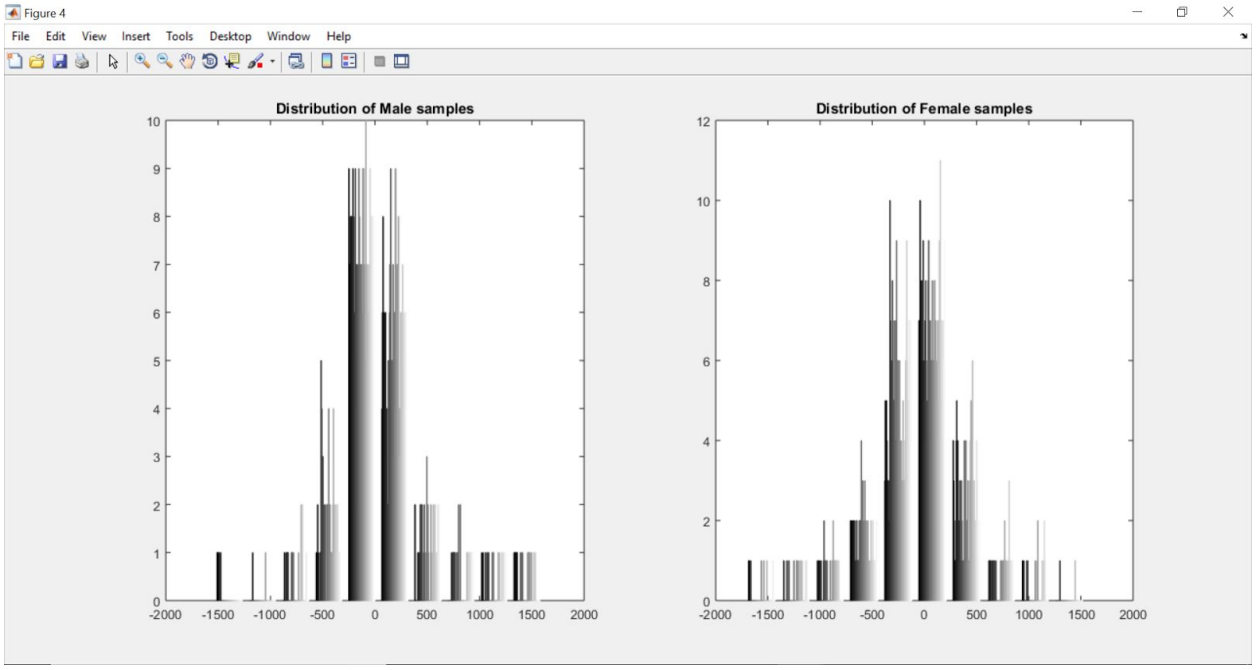

Fig 2: Distribution of male and female samples

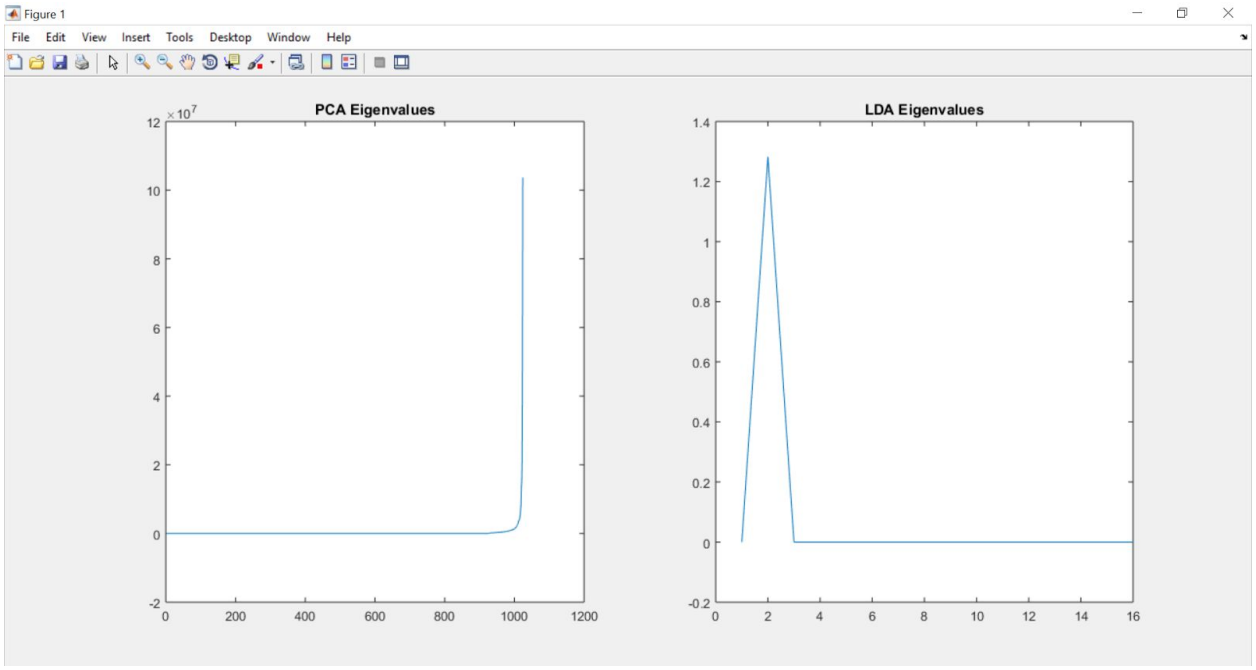

Fig 3: This figure shows the eigenvalues we get after performing PCA and LDA. Here we can see that solving the generalized eigenvalue problem with Sw and Sb results in a single PC.

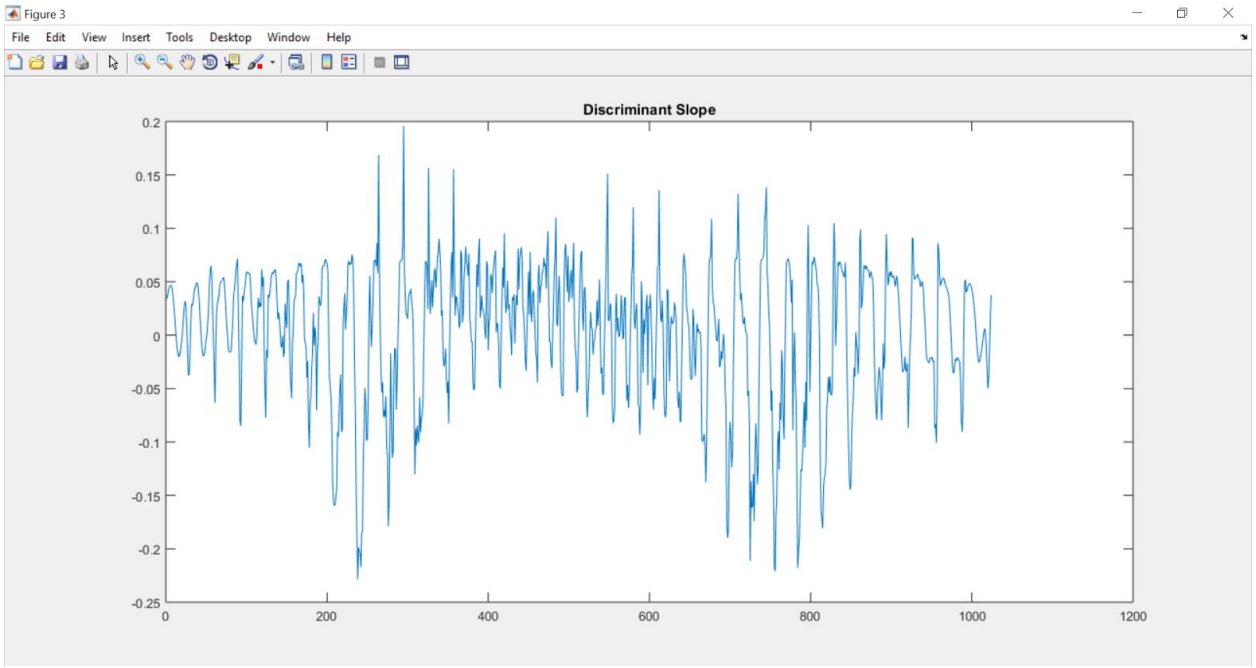

Fig 4: Slope of the Linear Discriminant function

Evaluation:

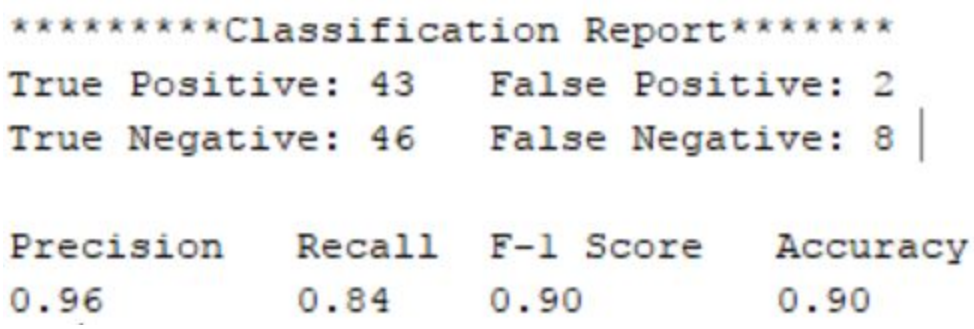

Fig 5: Classification Report for K = 16

The above figure 5 shows the confusion matrix and classification report for K = 16. The model is able to accurately classify 43 out of 45 male images and 46 out 54 female images. We're getting an average accuracy of 90% which is pretty good for starters.

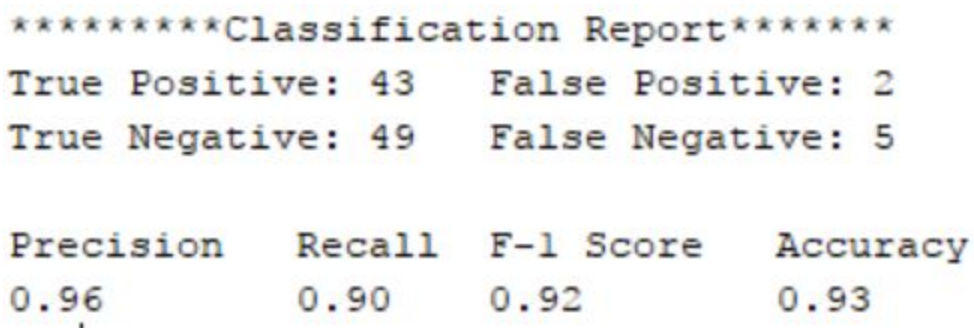

Fig 5: Classification Report for K = 30

For K = 30 that covers almost 90% of the variance, the model does slightly better and we get an average accuracy of 93%. The model is able to accurately classify 43 out of 45 male images and 49 out 54 female images in this case.