ECE469 Homework 3

Chase A. Lotito

December 4, 2024

Question 1

Train SVM classifiers using a Gaussian Kernel based on MATLAB.

Solution.

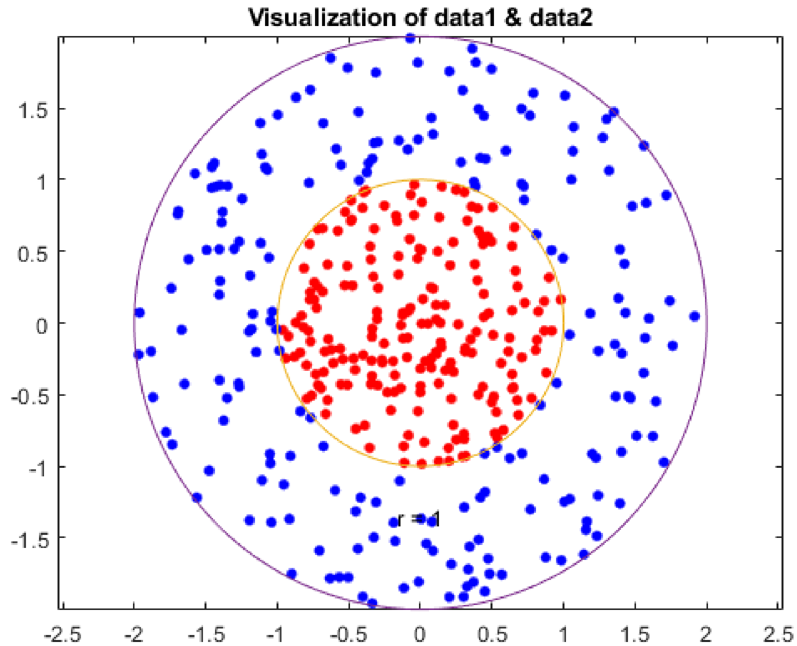
```
%{
==========================================
Homework 3 Question 1 Code
Name        : Chase Lotito
University  : Southern Illinois University
Course      : ECE469
==========================================
Description:
Training a SVM classifiers using a
Gaussian Kernel based on MATLAB
==========================================
%}

% (A) GENERATE 200 POINTS IN UNIT DISK
rng(1);
r = sqrt(rand(200,1));
theta = 2*pi*rand(200,1);
data1 = [r.*cos(theta), r.*sin(theta)];

% (B) GENERATE 200 POINTS IN ANNULUS (washer)
r2 = sqrt(3*rand(200,1)+1);
theta2 = 2*pi*rand(200,1);
data2 = [r2.*cos(theta2), r2.*sin(theta2)];

% (C) PLOT data1, data2, AND CIRCLES OF RADIUS 1 AND 2
figure;
plot(data1(:,1), data1(:,2), 'r.', 'MarkerSize', 15)
hold on
plot(data2(:,1), data2(:,2), 'b.', 'MarkerSize', 15)
ezpolar(@(x)1);
ezpolar(@(x)2);
axis equal
hold off

% (D) COMBINE data1 AND data2 INTO ONE MATRIX TO GENERATE
%     A VECTOR FOR CLASSIFICATIONS
data3 = [data1; data2];
```

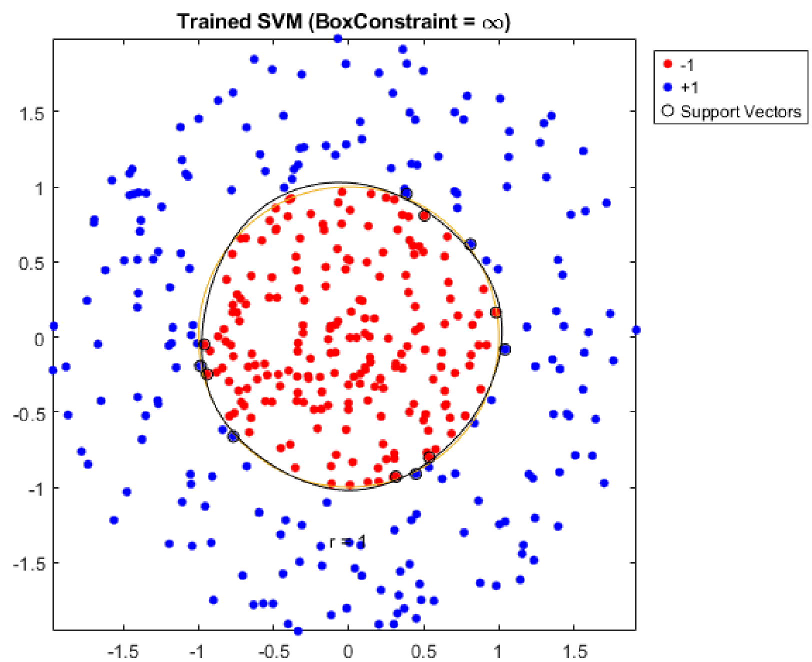
```
theclass = ones(400,1); % 400x1 vector full of 1s
theclass(1:200) = -1;   % set first 200 1s to -1s

% (E) TRAIN A SVM CLASSIFIER WITH "KernelFunction" SET TO
%      "rbf" AND BoxConstraint SET TO Inf. PLOT THE DECISION
%       BOUNDARY AND FLAG THE SUPPORT VECTORS

% train svm
cl = fitcsvm(data3, theclass, 'KernelFunction', 'rbf', 'BoxConstraint', 1,
    'ClassNames', [-1,1]);

% predict scores over grid
d = 0.02;
[x1Grid, x2Grid] = meshgrid(min(data3(:,1)):d:max(data3(:,1)),min(data3
   (:,2)):d:max(data3(:,2)));
xGrid = [x1Grid(:), x2Grid(:)];
[~, scores] = predict(cl, xGrid);

% plot data and decision boundary
figure;
h(1:2) = gscatter(data3(:,1), data3(:,2),theclass,'rb','.');
hold on
ezpolar(@(x)1);
h(3) = plot(data3(cl.IsSupportVector,1), data3(cl.IsSupportVector,2), 'ko'
   );
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k');
legend(h, {'-1', '+1', 'Support Vectors'});
axis equal
hold off
```

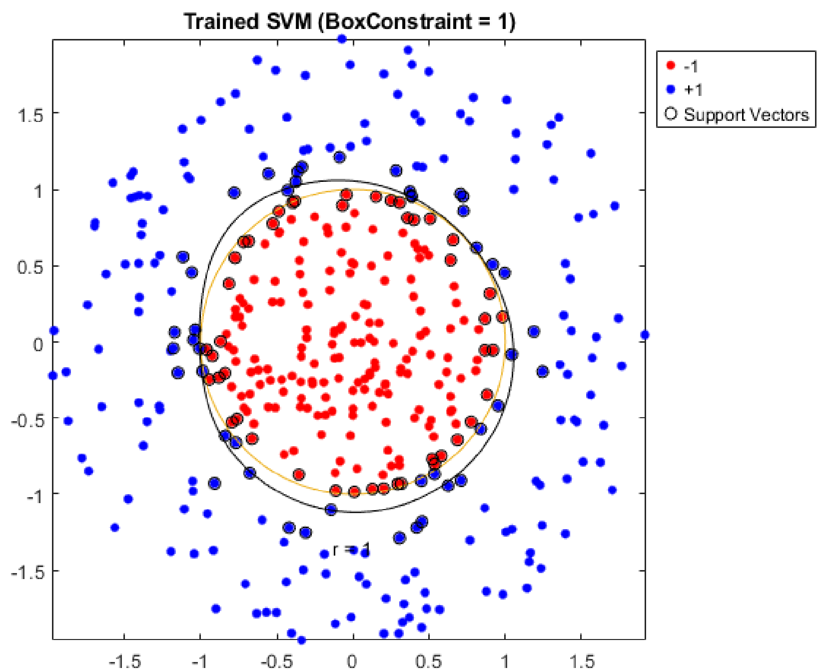
Code 1: q1.m

Code 1 utilizes a user-defined function to implement the Gaussian Kernel which can be referenced in Code 8.

For `BoxConstraint` $= 1$ the decision boundary is warped and we have more misclassifications. For `BoxConstraint` $= \infty$, the decision boundary is nearly the unit circle, and there are barely any misclassifications.


Visualization of data1 & data2
1.5
1
0.5
0
-0.5
-1
-1.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
r = 1


Trained SVM (BoxConstraint = ∞)
-1
+1
Support Vectors
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5
r = 1


Trained SVM (BoxConstraint = 1)
-1
+1
Support Vectors
r = 1
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

Question 2

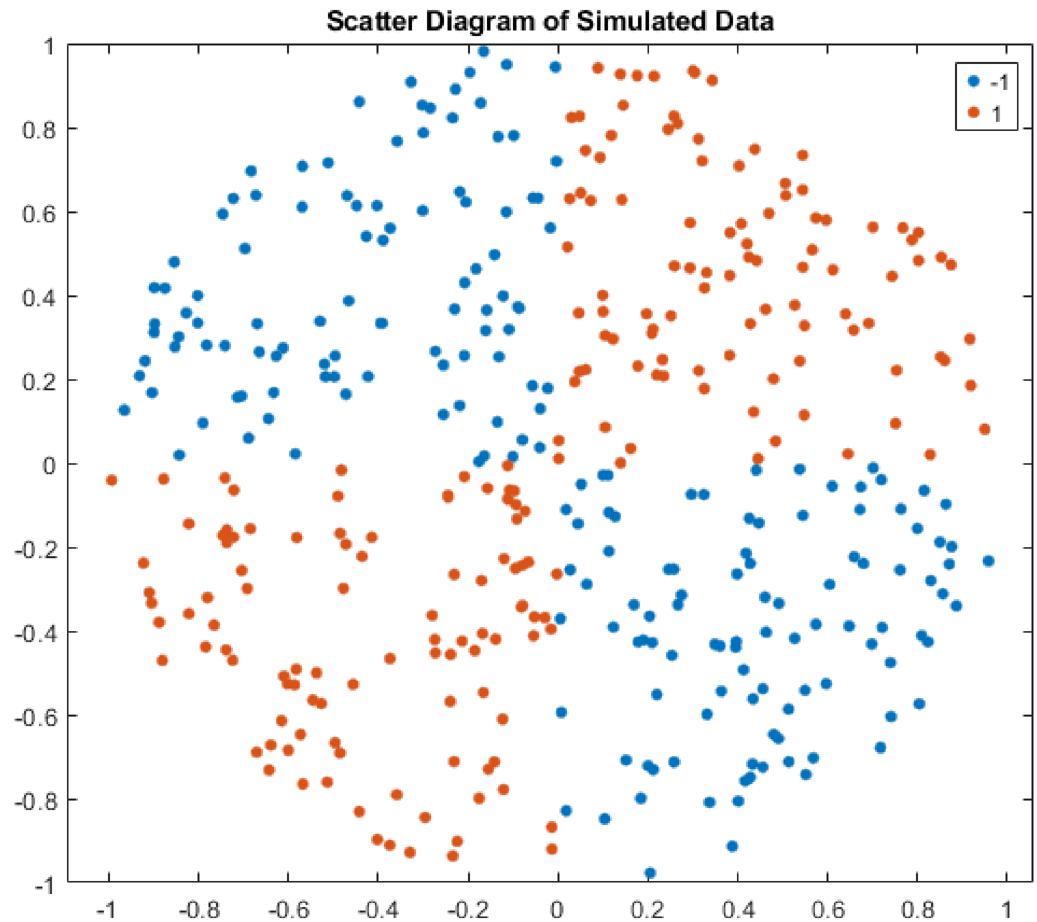
Solution.

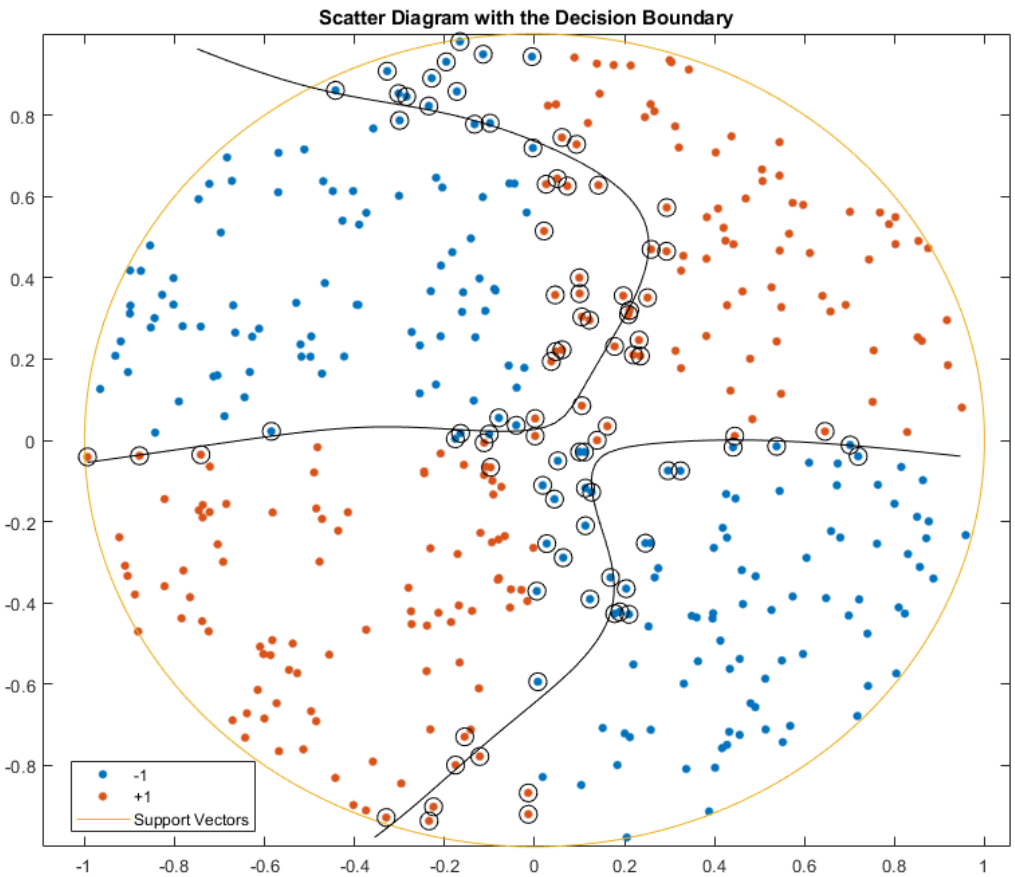
```
%{
==========================================
Homework 3 Question 2 Code
Name       : Chase Lotito
University : Southern Illinois University
Course     : ECE469
==========================================
Description:
Training a SVM classifiers using a custom
Kernel (Sigmoid kernel) based on MATLAB
==========================================
%}

% (A) GENERATE RANDOM SET OF POINTS IN UNIT CIRCLE.
%     Q1 AND Q3 POINTS POSITIVE CLASS.
%     Q2 AND Q4 POINTS NEGATIVE CLASS.

rng(1);
n = 100;

r1 = sqrt(rand(2*n,1));
t1 = [pi/2*rand(n,1) ; (pi/2*rand(n,1)+pi)];
X1 = [r1.*cos(t1) r1.*sin(t1)];

r2 = sqrt(rand(2*n,1));
t2 = [pi/2*rand(n,1) + pi/2 ; (pi/2*rand(n,1) - pi/2)];
X2 = [r2.*cos(t2) r2.*sin(t2)];

X = [X1 ; X2];
Y = ones(4*n,1);
Y(2*n + 1:end) = -1;

% (B) PLOT DATAPOINTS
figure;
gscatter(X(:,1),X(:,2),Y);
title('Scatter Diagram of Simulated Data');

% (C) USING THE SIGMOID KERNEL. WRITE A TRANSFORMATION
%     FUNCTION WHICH GENERATES THE GRAM MATRIX GIVEN
%     TWO MATRICIES AND THE SIGMOID KERNEL

% DEFINED IN SEPARATE FUNCTION FILEs

% (D) TRAIN SVM CLASSIFIER USING SIGMOID KERNEL
Mdl1 = fitcsvm(X, Y, 'KernelFunction', 'mysigmoid', 'Standardize',true);

% (E) PLOT DATA. IDENTIFY SUPPORT VECTORS AND BOUNDARY
% predict scores over grid
d = 0.02;
[x1Grid, x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),min(X(:,2)):d:max(X
```

```
(:,2)));
xGrid = [x1Grid(:), x2Grid(:)];
[~, scores] = predict(Mdl1, xGrid);

% plot data and decision boundary
figure;
h(1:2) = gscatter(X(:,1), X(:,2),Y);
hold on
ezpolar(@(x)1);
h(3) = plot(X(Mdl1.IsSupportVector,1), X(Mdl1.IsSupportVector,2), 'ko', 'MarkerSize',10);
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k');
title('Scatter Diagram with the Decision Boundary');
legend({'-1', '+1', 'Support Vectors'}, 'Location', 'Best');
hold off

% (F) DETERMINE OUT-OF-SAMPLE MISCLASSIFICATION RATE
%     USING 10-FOLD CROSS VALIDATION
CVMdl = crossval(Mdl1);
misclass1 = kfoldLoss(CVMdl1);
misclass1

% (G) WRITE NEW SIGMOID KERNEL W/ gamma=0.5 AND RETRAIN

% (D) TRAIN SVM CLASSIFIER USING SIGMOID KERNEL
Mdl2 = fitcsvm(X, Y, 'KernelFunction', 'mysigmoid_2', 'Standardize',true);

% (E) PLOT DATA. IDENTIFY SUPPORT VECTORS AND BOUNDARY
% predict scores over grid
d = 0.02;
[x1Grid, x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:), x2Grid(:)];
[~, scores] = predict(Mdl2, xGrid);

% plot data and decision boundary
figure;
h(1:2) = gscatter(X(:,1), X(:,2),Y);
hold on
ezpolar(@(x)1);
h(3) = plot(X(Mdl2.IsSupportVector,1), X(Mdl2.IsSupportVector,2), 'ko', 'MarkerSize',10);
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k');
title('Scatter Diagram (Model 2) with the Decision Boundary');
legend({'-1', '+1', 'Support Vectors'}, 'Location', 'Best');

CVMdl2 = crossval(Mdl2);
misclass2 = kfoldLoss(CVMdl2);
misclass2
```

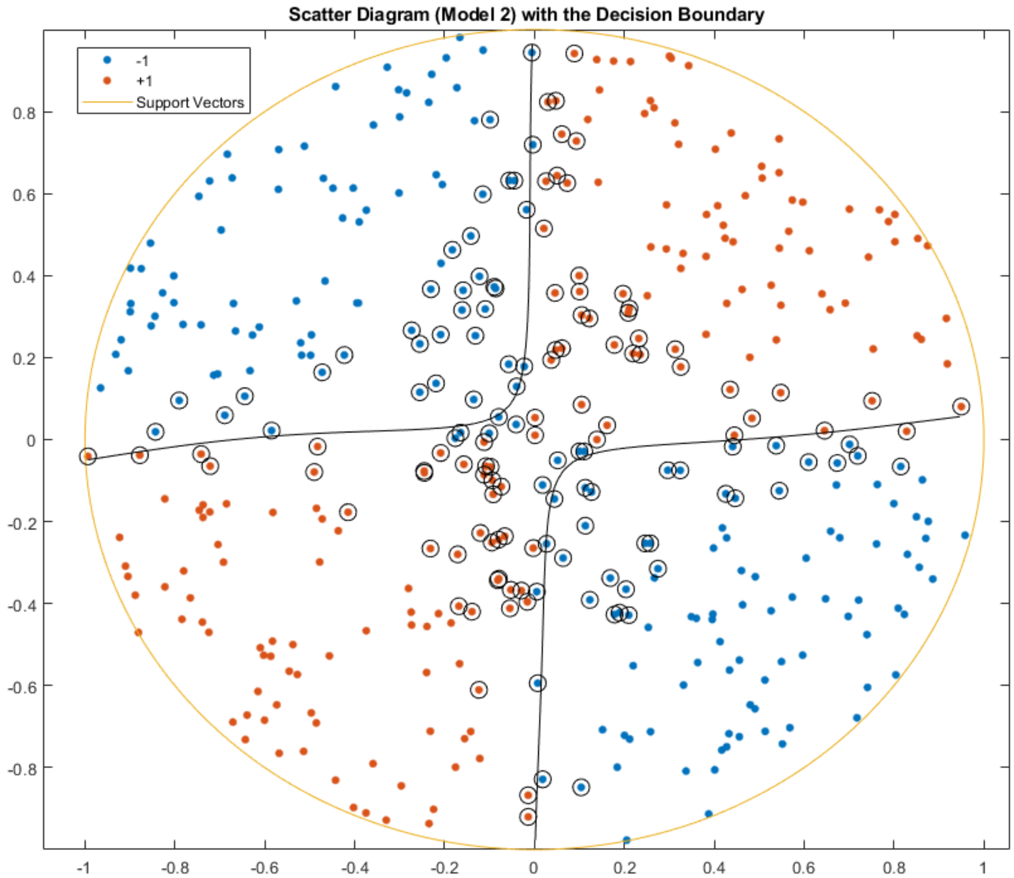
Code 2: q2.m

For Gaussian Kernel, `gamma` $= 1$ (Model 1) misclassification rate is 13.5%, `gamma` $= 0.5$ (Model 2) misclassification rate is 4.5%.


Scatter Diagram of Simulated Data
-1
1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1



Scatter Diagram with the Decision Boundary
-1
+1
Support Vectors
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1




```
misclass1 =

    0.1350


misclass2 =

    0.0450
```

Question 3

Solution.

```
%{
============================================
Homework 3 Question 3 Code
Name       : Chase Lotito
University : Southern Illinois University
Course     : ECE469
============================================
Description:
Design of a support vector machine for an
object recognition system
============================================
%}

% Load in dataset from ../svm/dataset2.mat
% Dataset is a struct of X (100x4)-matrix
% and Y (100x1)-vector
dataset2 = load('C:\Users\cloti\OneDrive\Desktop\SCHOOL\SIUC\f24\ece469\hw\hw3\svm\svm\dataset2.mat');

% Extract input features X and output target y
X = dataset2.X;
y = dataset2.Y;  % y = {1, 0}


% visualize dataset via heatmap
figure;
correlationMatrix = corr(X);
heatmap(correlationMatrix, 'Colormap', jet, 'ColorbarVisible', 'on');
title('Correlation Matrix of Features');

% Train SVM Classifier
% ((1) Linear Kernel)
mdl1 = fitcsvm(X, y, 'KernelFunction','linear');
% ((2) Polynomial Kernel)
mdl2 = fitcsvm(X, y, 'KernelFunction','polynomial', 'PolynomialOrder', 2);
% ((3) Guassian Kernel)
mdl3 = fitcsvm(X, y, 'KernelFunction','mysigmoid_2', 'Standardize', true);

% Cross-validate model with 4-fold cross-validation
cvMdl1 = crossval(mdl1, 'KFold', 4);     % Linear 4-fold cross-val
misclas1 = kfoldLoss(cvMdl1);            % calc losses
cvMdl2 = crossval(mdl2, 'KFold', 4);     % Poly  4-fold cross-val
misclas2 = kfoldLoss(cvMdl2);            % calc losses
cvMdl3 = crossvUal(mdl3, 'KFold', 4);     % Guassian  4-fold cross-val
misclas3 = kfoldLoss(cvMdl3);            % calc losses
```

Code 3: q3.m

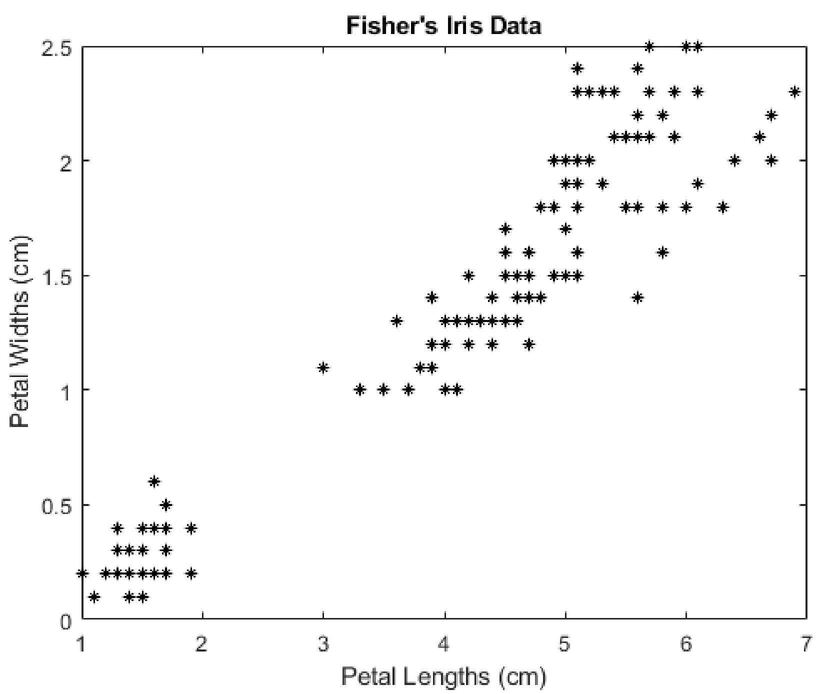
Question 4

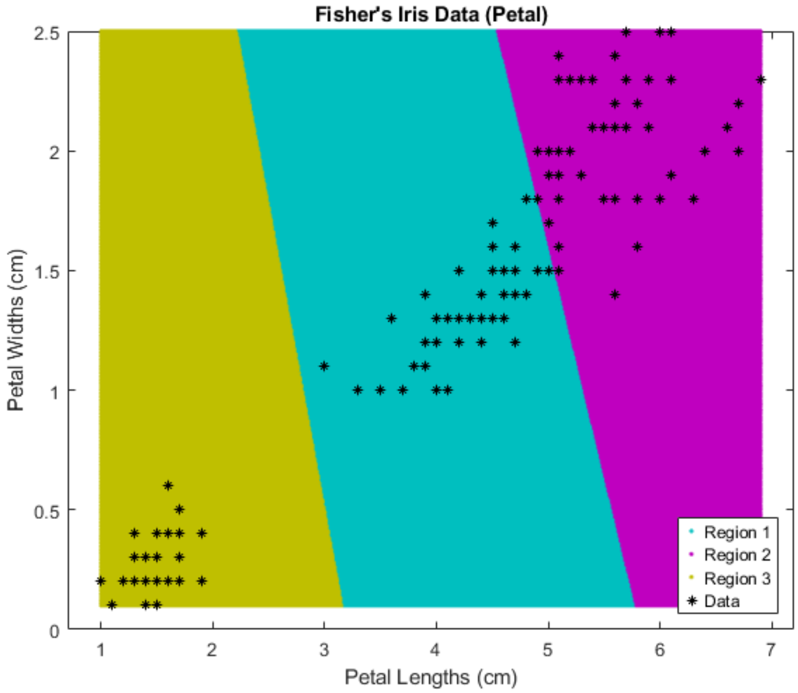
Solution.

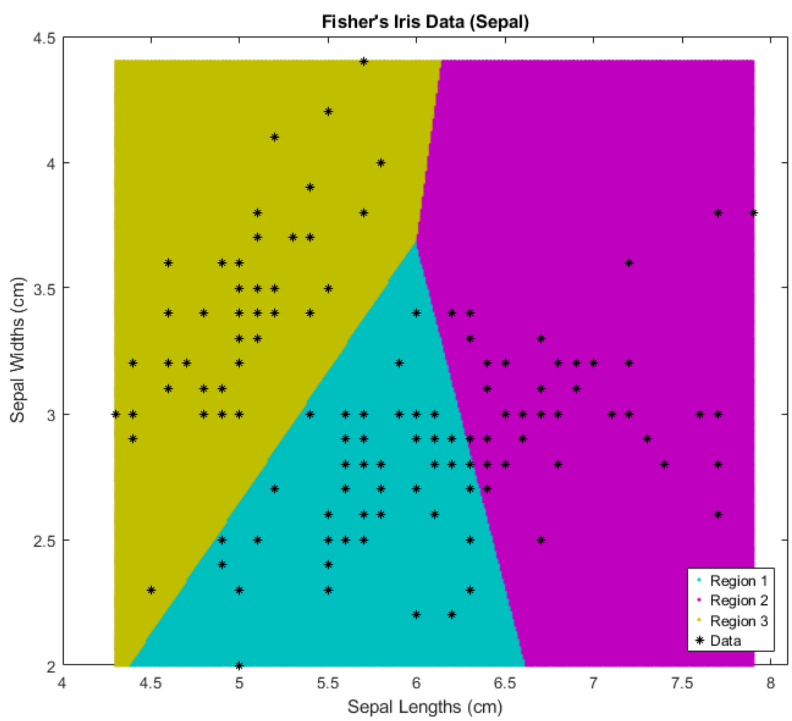
```
%{
=========================================
Homework 3 Question 4 Code
Name        : Chase Lotito
University : Southern Illinois University
Course      : ECE469
=========================================
Description:
K-Means Clustering
=========================================
%}

load fisheriris
X = meas(:,3:4);  % extract petal lengths and widths

% (A) X IS 2D, VISUALIZE IN 2D SPACE
figure;
plot(X(:,1), X(:,2), 'k*', 'MarkerSize', 5);
title('Fisher''s Iris Data (Petal)');
xlabel('Petal Lengths (cm)');
ylabel('Petal Widths (cm)');

% (B) RUN K=3 K-MEANS
rng(1);    % reproducibility

% idx: vector of predicted cluster id's
%   C: matrix of centroid locations
[idx, C] = kmeans(X,3);

% (C) COMPUTE CENTROID DISTANCE BY PASSING C INTO KMEANS
% create grid
x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:), x2G(:)];
idx2Region = kmeans(XGrid, 3, MaxIter=1,Start=C);

% (D) VISUALIZE CLUSTERS
figure;
gscatter(XGrid(:,1), XGrid(:,2), idx2Region,
    [0,0.75,0.75;0.75,0,0.75;0.75,0.75,0], '.'); % vector is colors
hold on;
plot(X(:,1), X(:,2), 'k*', 'MarkerSize',5);
title('Fisher''s Iris Data (Petal)');
xlabel('Petal Lengths (cm)');
ylabel('Petal Widths (cm)');
legend('Region 1', 'Region 2', 'Region 3', 'Data', Location='SouthEast');
hold off;

% (E) EXTRACT LENGTH AND WIDTH OF SEPALS AND REPEAT ABOVE
```

```
X = meas(:,1:2);   % extract sepal lengths and widths

% idx: vector of predicted cluster id's
%    C: matrix of centroid locations
[idx, C] = kmeans(X,3);

% (C) COMPUTE CENTROID DISTANCE BY PASSING C INTO KMEANS
% create grid
x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:), x2G(:)];
idx2Region = kmeans(XGrid, 3, MaxIter=1,Start=C);

% (D) VISUALIZE CLUSTERS
figure;
gscatter(XGrid(:,1), XGrid(:,2), idx2Region,
    [0,0.75,0.75;0.75,0,0.75;0.75,0.75,0], '.'); % vector is colors
hold on;
plot(X(:,1), X(:,2), 'k*', 'MarkerSize',5);
title('Fisher''s Iris Data (Sepal)');
xlabel('Sepal Lengths (cm)');
ylabel('Sepal Widths (cm)');
legend('Region 1', 'Region 2', 'Region 3', 'Data', Location='SouthEast');
hold off;
```

Code 4: q4.m




Fisher's Iris Data (Petal)
Petal Widths (cm)
Petal Lengths (cm)
Region 1
Region 2
Region 3
Data
0
0.5
1
1.5
2
2.5
1
2
3
4
5
6
7


Fisher's Iris Data (Sepal)
Sepal Widths (cm)
Sepal Lengths (cm)
Region 1
Region 2
Region 3
Data
2
2.5
3
3.5
4
4.5
4
4.5
5
5.5
6
6.5
7
7.5
8

Question 5

Solution.

```
%{
============================================
Homework 3 Question 5 Code
Name        : Chase Lotito
University : Southern Illinois University
Course      : ECE469
============================================
Description:
Principal Component Analysis (PCA)
============================================
%}

% load hald (cement heat) dataset
load('hald.mat');

% (i) PCA X, X ZERO MEAN (SAMPLE MEAN),
%      COVARIANCE MATRIX, EIGENVECTORS
%      OF COVARIANCE MATRIX

% let ingredients be X (13x4)-matrix
X = ingredients;

% get zero mean for X (sample mean)
meanX = mean(X(:));
X = X - meanX;
zero_meanX = mean(X(:));
disp(['meanX = ', num2str(meanX), ' | zero_meanX = ', num2str(zero_meanX)
    ]);

% (4x4) covariance matrix (4 b/c 4 features)
covX = cov(X);

% find the eigenvectors of covariance matrix
% V: eigenvector column matrix
% D: eigenvalue diagonal matrix
[V,D] = eig(covX);

% sort the eigenvalues largest to smallest (noting location)
[D_sort, idx] = sort(diag(D), 'descend');
V_sort = V(:, idx);    % rearrange eigenvector matrix accordingly


% (ii) use built-in MATLAB PCA function
Mat_pc = pca(ingredients);

disp('MANUAL:');
disp(V_sort);
disp('BUILT-IN:');
disp(Mat_pc);

```

```
% Run PCA on the input matrix X
X = [ 2 4 5 5 3 2 ; 2 3 4 5 4 3 ];
k = 1;
n = 6;
d = 2;

pcaX = pca(X);

disp('pca(X)');
disp(pcaX);
```

Code 5: q5.m

```
>> q5
meanX = 24.3462 | zero_meanX = -4.0993e-16
MANUAL:
   -0.0678    0.6460    0.5673    0.5062
   -0.6785    0.0200   -0.5440    0.4933
    0.0290   -0.7553    0.4036    0.5156
    0.7309    0.1085   -0.4684    0.4844

BUILT-IN:
   -0.0678   -0.6460    0.5673    0.5062
   -0.6785   -0.0200   -0.5440    0.4933
    0.0290    0.7553    0.4036    0.5156
    0.7309   -0.1085   -0.4684    0.4844
```

(ii)

```
pca(X)
              0
         0.5000
         0.5000
              0
        -0.5000
        -0.5000
```

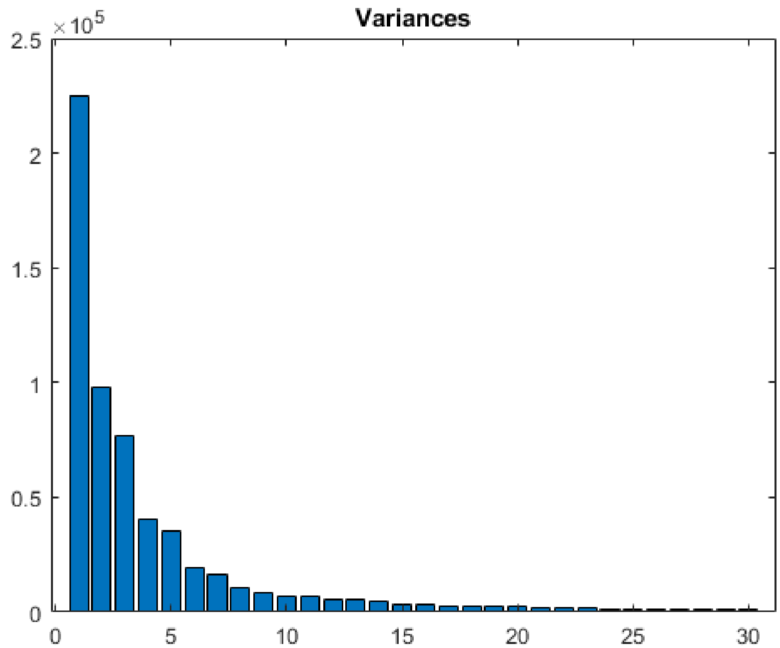
Question 6

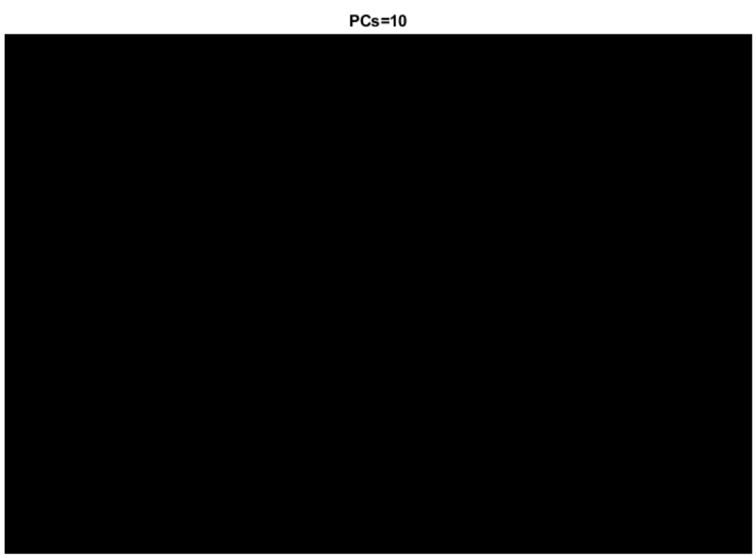
Solution.

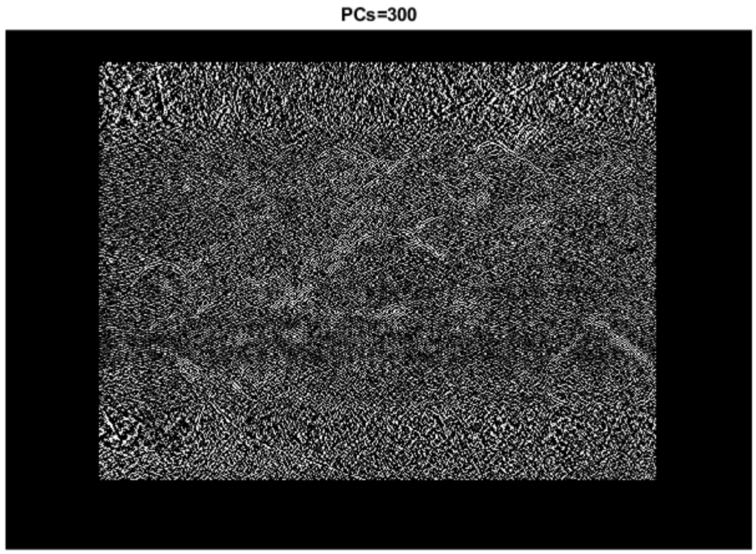
```
%{
===========================================
Homework 3 Question 6 Code
Name       : Chase Lotito
University : Southern Illinois University
Course     : ECE469
===========================================
Description:
Image Compression with PCA
===========================================
%}

[RGB map] = imread('peppers.png');
imshow(RGB)
peppers_gray = rgb2gray(RGB);
save peppers_gray;
figure;
imshow(peppers_gray)
peppers_gray = double(peppers_gray); % convert to double precision
axis off, axis equal
X = peppers_gray; %raw data matrix
[m, n] = size(X);
mean_X = mean(X,2); % compute row mean
reformat_mean = repmat(mean_X,1,n);
tilde_X = X - reformat_mean; % subtract row mean to obtain X
covX = tilde_X*tilde_X'/(n-1); %Sample covariance matrix of X
[U,S] = eig(covX);
[Ordered_eigValue,ind] = sort(diag(S),'descend');
U_od = U(:,ind);
variances = (Ordered_eigValue); % compute variances
figure;
bar(variances(1:30)) % plot of variances
%% Extract first 40 principal components
PCs = 477; % Number of principle components used for compression
U_red = U(:,1:PCs);
Z = U_red'*tilde_X; % project data onto PCs
X_hat = U_red*Z; % convert back to original basis
figure;
imshow(X_hat), axis off; % display results
```

Code 6: q6.m

The image quality for the compressed PCA images is very low, but for `PCs=477` you can see strong outlines of the peppers.


Variances
×10^5
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30


PCs=10


PCs=300


PCs=477

Question 7

Solution.

```
'''
==========================================
Homework 3 Question 7 Code
Name       : Chase Lotito
University : Southern Illinois University
Course     : ECE469
==========================================
Description:
Feed-forward neural network
==========================================
'''

import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report
from scipy.io import loadmat

# import dataset
data = loadmat('../dataset1.mat')
X = data['X']
y = data['Y']

plt.scatter(X[y[:,0] == 0, 0], X[y[:,0] == 0, 1], color='red', label='Class 0')
plt.scatter(X[y[:,0] == 1, 0], X[y[:,0] == 1, 1], color='blue', label='Class 1')
plt.title('Scatterplot of dataset1.mat')
plt.ylabel('$x_2$')
plt.xlabel('$x_1$')
plt.legend()
plt.show()
```

Code 7: q7.py

APPENDIX A: Auxillary Code

The following codes was used to implement the Gaussian Kernel for Support Vector Machines (SVM).

```
function G = mysigmoid(U,V)
gamma = 1;
c = -1;
G = tanh(gamma*U*V' + c);
end
```

Code 8: mysigmoid.m

```
function G = mysigmoid_2(U,V)
gamma = 0.5;
c = -1;
G = tanh(gamma*U*V' + c);
end
```

Code 9: mysigmoid_2.m