Table of Contents

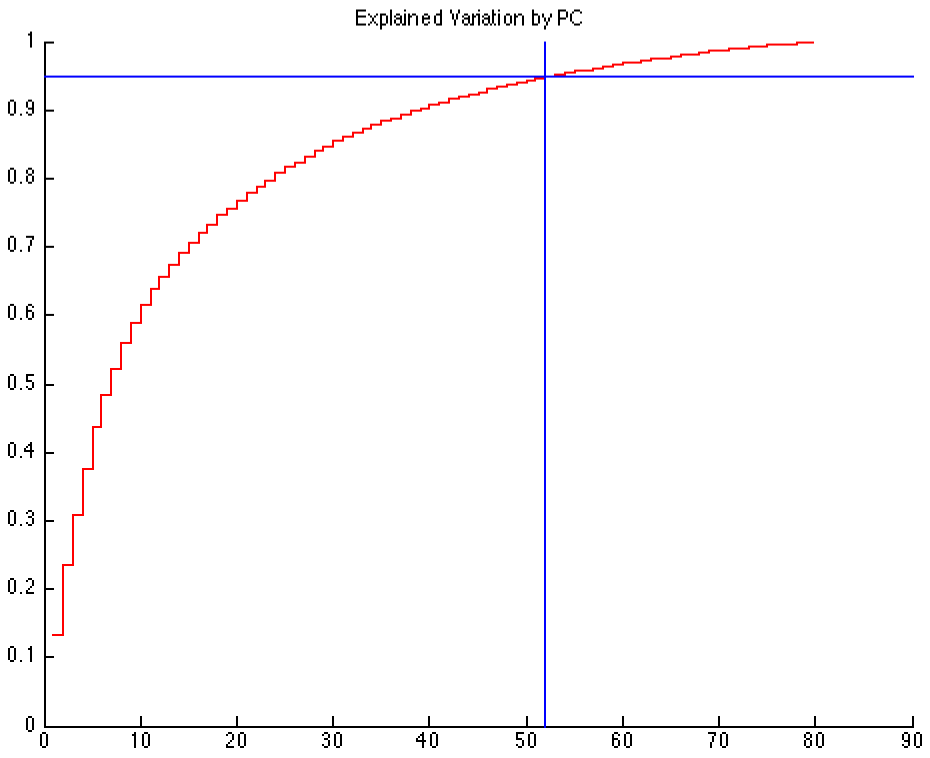
```
% Sean Bennett
% HW6
```

Question 1

```
load('CancerMicroarray.mat')

numTumors = length(G)
numGenes = length(X)

% There are 2308 different genes sequenced from 83 different tumors.

[coeffPC, scorePC, latentPC] = pca(X);

cumulativeLatent = cumsum(latentPC) / sum(latentPC);

numPC = max(find(cumulativeLatent < 0.95));

% You need 52 or 53 PCs to explain 95% of the data

figure;
hold on;
stairs(cumulativeLatent, 'Color', 'r');
title('Explained Variation by PC');
refline(0, 0.95);
line([52 52],[0 1]);
hold off;

linCValAcc = zeros(400,1);
quadCValAcc = zeros(400,1);

for i = 1:400

trainingindicies = randsample(1:83, 60);
testindicies = 1:83;
testindicies(ismember(testindicies,trainingindicies)) = [];

trainingdata = X(trainingindicies, :);
testdata = X(testindicies, :);

trainingclass = G(trainingindicies);
testclass = G(testindicies);
```

```
[trainingcoeffPC, trainingscorePC, traininglatentPC] = pca(trainingdata);

linClass = fitcdiscr(trainingscorePC(:, 1:10), trainingclass);
quadClass = fitcdiscr(trainingscorePC(:, 1:10), trainingclass, ...
    'DiscrimType', 'pseudoQuadratic');

predictedLinClass = predict(linClass, testdata(:,1:10));
predictedQuadClass = predict(quadClass, testdata(:,1:10));

linCValAcc(i) = mean(predictedLinClass == testclass');
quadCValAcc(i) = mean(predictedQuadClass == testclass');

end;

avgLinearCrossVal = mean(linCValAcc)
avgQuadraticCrossVal = mean(quadCValAcc)

% My mean linear and quadratic cross validated accuracies are 22.9% and
% 26.0%. This seems low, but I can't identify anything incorrect in my
% implementation of the PCA/fitcdiscr functionality so I assume there's
% some sort of logic error in my test/training data set creation in the
% loop, or the test data actually passed to predict. I was a little fuzzy
% on that.
% Not sure, but it at least compiles and produces a number!
```

```
numTumors =

    83


numGenes =

        2308


avgLinearCrossVal =

    0.2349


avgQuadraticCrossVal =

    0.2667
```

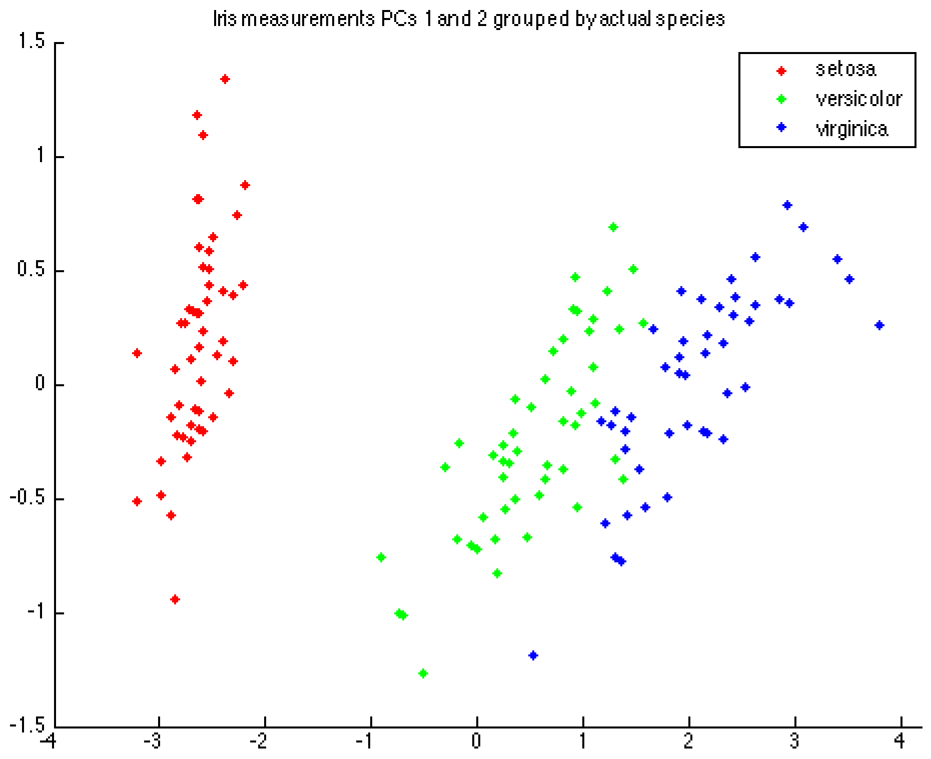


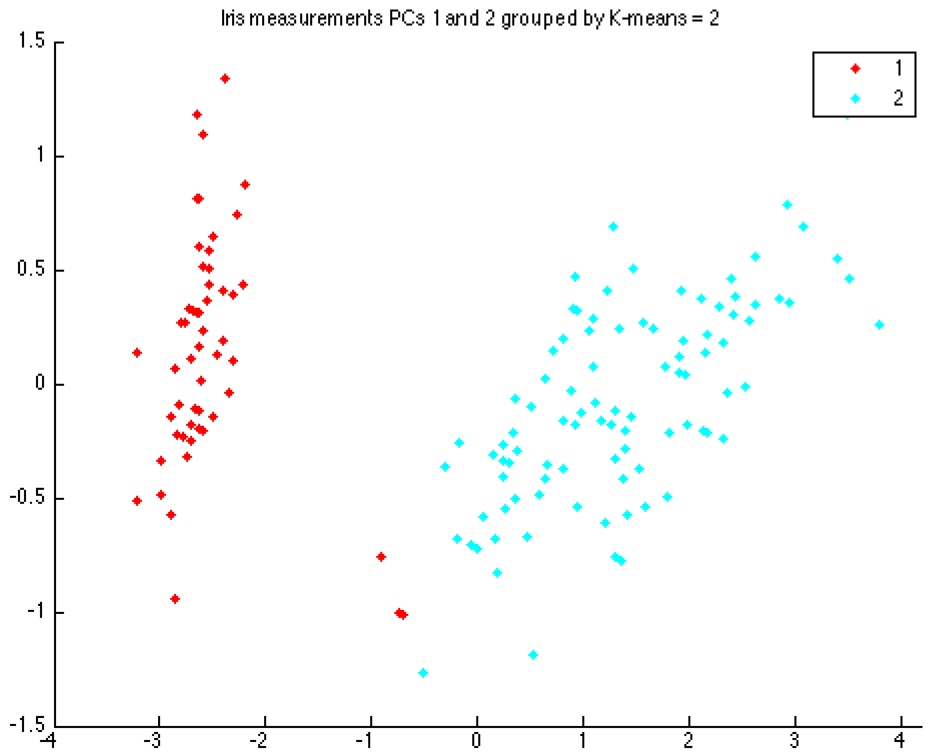
```
% Decision trees are super cool. They seek to partition data space into a
% dichotomous outcome, is a, or is not a for example that minimizes
% "impurity". It does this recursively, producing some sort of tree that
% will have multiple branches looking like is a / is not a with is not a
% subdivided further. Classification trees deal with classifications alone,
% and then there are regression trees that can deal with continuous data.
% There are then CART trees, which can deal with both. (I have played with
% these a little bit in analyzing large ecological data sets where we were
% trying to determine disease load based upon multiple terrain and
% oceanographic types)

treeCVal = zeros(400,1);

for i = 1:400

trainingindicies = randsample(1:83, 60);
testindicies = 1:83;
testindicies(ismember(testindicies,trainingindicies)) = [];

trainingdata = X(trainingindicies, :);
testdata = X(testindicies, :);

trainingclass = G(trainingindicies);
testclass = G(testindicies);

[trainingcoeffPC, trainingscorePC, traininglatentPC] = pca(trainingdata);
```

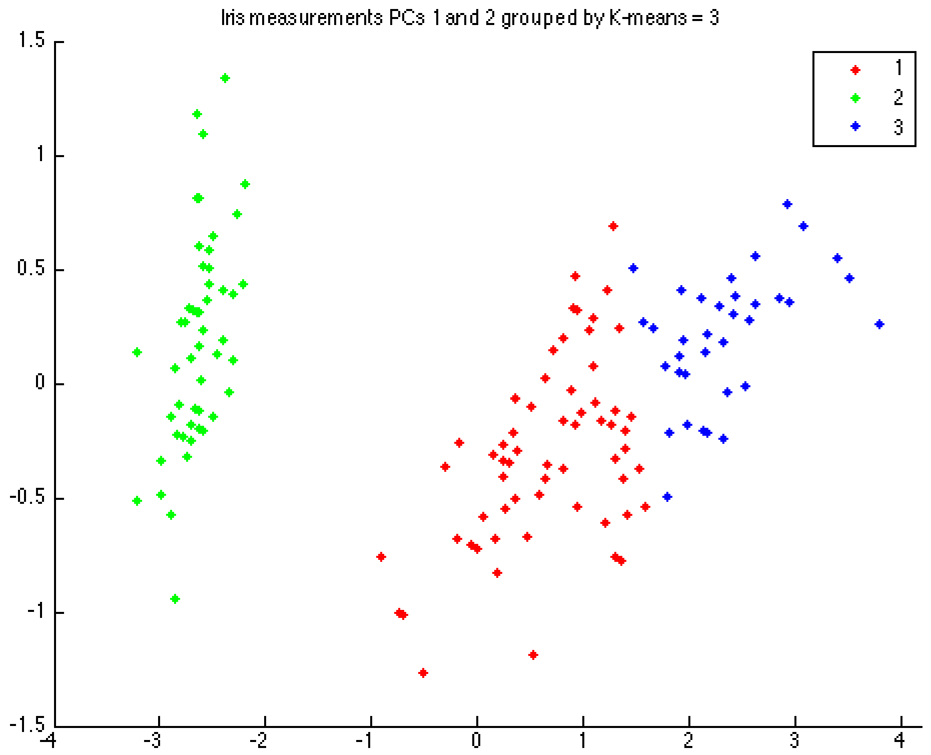
```
treeDiscr = fitctree(trainingscorePC(:, 1:10), trainingclass);

treePredict = predict(treeDiscr, testdata(:, 1:10));

treeCVal(i) = mean(treePredict == testclass');

end
% the Decision tree has ~ 25% cross validated accuracy. Still somewhat
% sucky. I'm pretty sure it's a problem with the predict function, but I
% don't know how to fix it. I'm still unsure of what exactly the first 10
% PCs represents in my data, like how do I access the test data that
% corresponds to those PCs, or is there any way to represent that data?
```

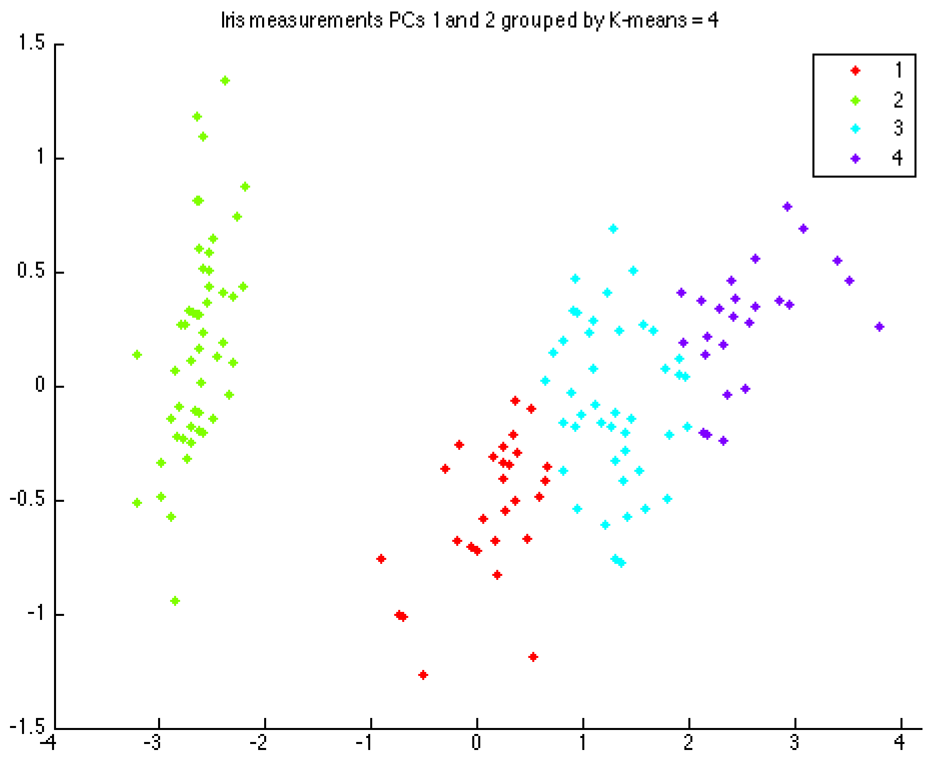
Question 2

```
load('fisheriris');
numSpecies = length(unique(species));
numIris = length(species);
numMeas = size(meas,2);

% The measruements Ronald Fisher took were Petal Width, Petal Length, Sepal
% Width, and Sepal Length, measured in mm There were three different
% species, setsoa, versicolor, and virginica. There were 150 different
% flowers sampled.

[coeffPC2, scorePC2, latentPC2] = pca(meas);

figure;
gscatter(scorePC2(:, 1), scorePC2(:, 2), species);
title('Iris measurements PCs 1 and 2 grouped by actual species');

kmeans2 = kmeans(meas, 2);
figure;
gscatter(scorePC2(:, 1), scorePC2(:, 2), kmeans2);
title('Iris measurements PCs 1 and 2 grouped by K-means = 2');

kmeans3 = kmeans(meas, 3);
figure;
gscatter(scorePC2(:, 1), scorePC2(:, 2), kmeans3);
title('Iris measurements PCs 1 and 2 grouped by K-means = 3');

kmeans4 = kmeans(meas, 4);
figure;
gscatter(scorePC2(:, 1), scorePC2(:, 2), kmeans4);
title('Iris measurements PCs 1 and 2 grouped by K-means = 4');

% Determining the number of groups, k, is hard algorithmically. Some guys
% out of Stanford apparently have developed a Gap statistic that determines
% the k such that the Gap(k) >= Gap(k + 1) - sim error(k+1), It seems to
% work well. Some other guys ahve figured out an algorithm based on a
% statistical test that determines a K in which the data about each k
% centroid closest approximates a Gaussian distribution. There's also a way
% to do it graphically via silhouette graphs or information theory
% critirions, but i'm well out of sentence now so lets just say those
% exist.
```



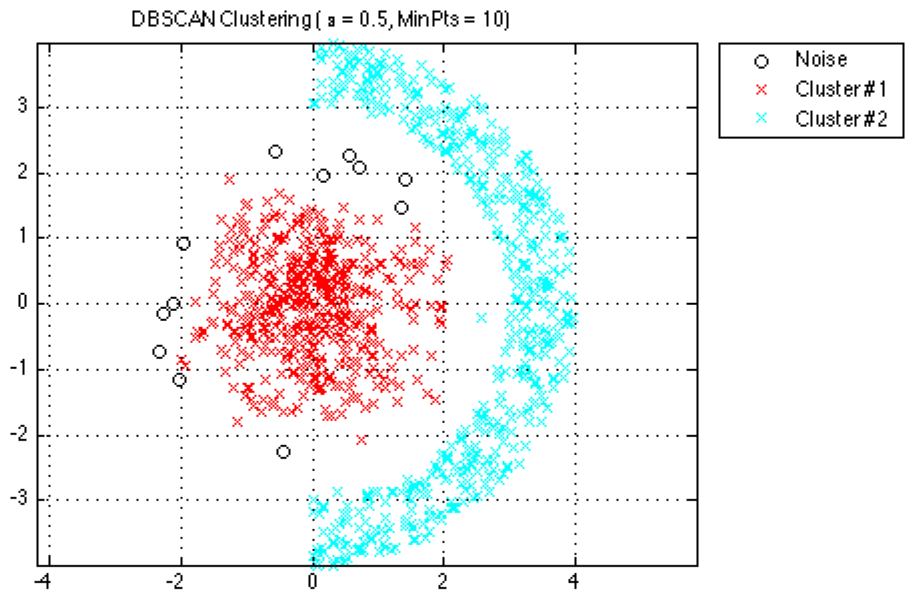

Iris measurements PCs 1 and 2 grouped by K-means = 2
1
2
1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4

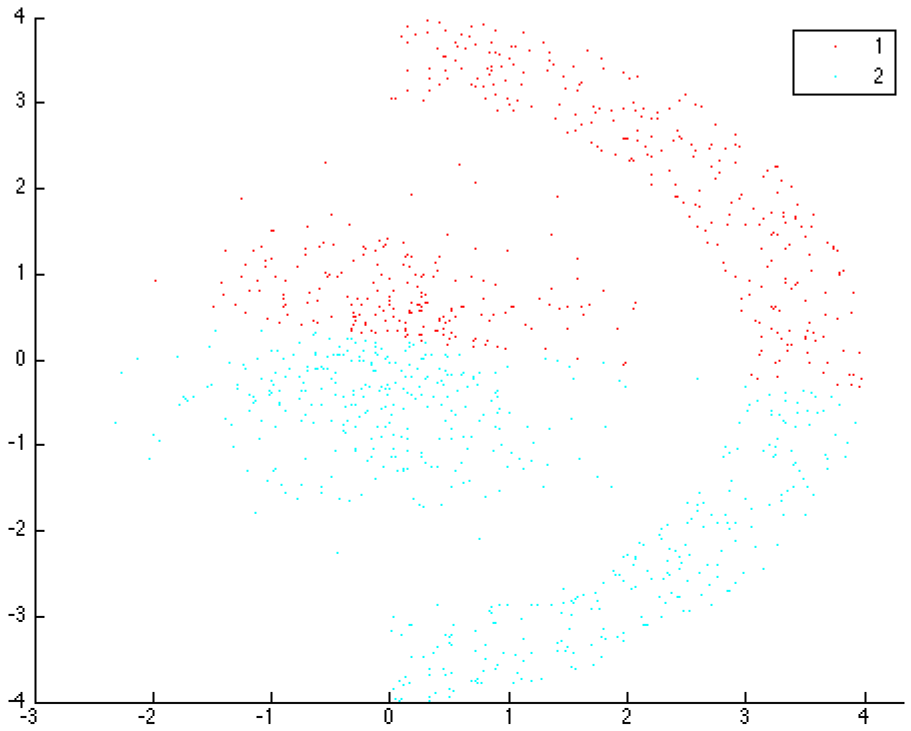

Iris measurements PCs 1 and 2 grouped by K-means = 3
1
2
3
1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4



Question 3

```
% Pt a.

% DBSCAN uses three different types of points, core points, border points,
% and noise points to determine clusters based upon the comparison of high
% density and low density regions. Core points have some minimum number of
% points within some radius epsilon, border points fall within a
% neighborhood of a core point, but do not have the minimum number of points
% to qualify as a core point, and noise points are any points which are not
% noise or border points. The general algoritmn is the classify points as
% one of the three, remove noise points, and then cluster based upon
% remaining points.

% Pt b.

load('mydata');

epsilon=0.5;
MinPts=10;
IDX=DBSCAN(X,epsilon,MinPts);

figure;
PlotClusterinResult(X, IDX);
title(['DBSCAN Clustering (\epsilon = ' num2str(epsilon) ...
    ', MinPts = ' num2str(MinPts) ')']);


Q3kmeans = kmeans(X, 2);
figure;
gscatter(X(:, 1), X(:, 2), Q3kmeans);

% pt c.

% The DBSCAN algorithm is density based and has the ability to deal with
% non-linear groupings. Visible in the comparison between the two graphs is
% k-means needs to draw a straight line between the two groups, while
% DBSCAN can draw a non-straight/curved line. I found a thing that
% describes it as k-means can't deal with "non-globular" shapes.
```





Published with MATLAB® R2014a