**Visualization and Learning Structure: Problem Session 7/18/16**

Let's play with generating curves for use with spectral clustering and determining the best mixture models for our data. We'll also re-visit our Italian olive oils; as a reminder, we have eight different chemical composition measurements on 572 Italian olive oils. There are two sets of labels, the three Regions and the nine Areas.

1. Download the `curvelib.R` and `generatesplinedata.R` files from our website and source into R: `source("curvelib.R"); source("generatesplinedata.R")`
2. Once we source those functions into R, we can generate data along any spline that we create using point-and-click with the mouse. Try creating four different groups:

   ```
   data<-gen.data(c(100,100,100,100),0.02)
   labels<-c(rep(1,100),rep(2,100),rep(3,100),rep(4,100))
   ```

   When you run that code, a graph window will open up and wait for you. You can take your mouse and click a sequence of points in the window, then right-click to hit stop. Then R will interpolate a spline through those points and add random scatter (the 0.02 argument). A new graph window will open up; point-and-click to start the next group. You'll cycle through that process four times. At the end, `data` has all four groups. You'll need to try to keep the group locations straight while doing it so you don't accidentally put groups too much on top of each other.

   Plot your data to see what your final groups look like:

   ```
   plot(data,type="n",xlab="x1",ylab="x2"); text(data,labels=labels,cex=1.2)
   ```

   You can re-do them if they weren't what you wanted.

3. Let's use spectral clustering to find your different cluster shapes.
   - First, create your affinity matrix. Note that we can change the `sigma` parameter to alter the size of our neighborhoods. We can visualize the affinity matrix to double check its structure. Since the groups were created in order, you should be able to see blocks in the matrix. Moreover, groups that are closer to each other should show some off-diagonal block structure.

     ```
     sigma<-0.4; dist.m<-as.matrix(dist(data))
     A<-as.matrix(exp(-1*(dist.m^2)/(2*sigma^2)))
     image(A)
     ```

   - Now normalize your affinity matrix (Ng, Jordan, Weiss). Again we can visualize:

     ```
     D<-matrix(0,nrow(A),ncol(A)); diag(D)<-rowSums(A)
     D.negsqrt<-solve(D^0.5)
     P<-D.negsqrt%*%A%*%D.negsqrt; image(P)
     ```

- Now find the eigendecomposition of the matrix. You should be able to see some cluster structure by plotting the first four eigenvectors:

  ```
  ev<-eigen(P)
  par(mfrow=c(2,2)); for(i in 1:4) plot(ev$vectors[,i])
  ```

- Choose the first four eigenvectors (i.e. four groups) and cluster with k-means:

  ```
  km<-kmeans(ev$vectors[,1:4],4)
  table(labels,km$cl)
  par(mfrow=c(1,1)); plot(data,pch=16,col=km$cl)
  ```

  How did we do? Do the mistakes (if any) make sense given the original data?

- Experiment with changing the `sigma` value to see the effect that changing the neighborhood has on the projection and the clustering results.

4. Let's see how model-based clustering performs on your created data. If you built anything that's spherical, it should be identified; otherwise, we'll watch how more non-standard shapes are modeled as a combination of Gaussians.

   - First, download/install/load the `mclust` library: `library(mclust)`; `help(Mclust)`
   - Now we'll search for the best fit for the known number of groups:

     ```
     library(mclust)
     mbc.spline<-Mclust(data,G=4)
     mbc.spline; plot(mbc.spline)
     table(labels,mbc.spline$cl)
     classError(labels,mbc.spline$cl)
     ```

     How did we do?

   - What happens when we let the procedure pick the number of clusters:

     ```
     mbc.spline<-Mclust(data,G=2:15)
     ```

     How many did it choose? Were any of your groups overfit?
     Do your results make sense?

5. Finally, explore using model-based clustering to find clusters for your olive oils. You might look for 1) three clusters and compare to Region, 2) nine clusters and compare to Area, or 3) search for a solution over a wide range of values.
   For example,

   ```
   olive<-read.table("olive.txt")
   Region<-olive[,1]; Area<-olive[,2]; chem.meas<-olive[,3:10]

   mbc.olive<-Mclust(chem.meas,G=3)
   mbc.olive; plot(mbc.olive)  ##check out what happens when you have higher-dim
   table(Region,mbc.olive$class)
   classError(Region, mbc.olive$class)
   ```