Homework 8

Advanced Methods for Data Analysis (36-402/36-608)

Due Tues April 22, 2014 at 11:59pm

Instructions: each homework will be submitted in two parts.

1. *Writeup.* An important part of the learning the data analysis trade is learning how to communicate. Prepare a writeup to the questions below and work hard to make your submission as readable as you possibly can—this means no raw R code as part of the writeup, unless you find it necessary for clarity; no raw R output with 12 significant digits, etc., unless again you deem it necessary; making sure figures are of proper (large) size with labeled, readable axes; and so forth. Your submitted writeup must be in PDF format. You can produce this PDF using LaTeX, Word, or whatever you want, but at the end of the day, it must be a PDF. Any other type of file will not be accepted.

2. *Code.* Also submit your R code for the homework as a plain text file. You should demarcate sections of the code that correspond to the different homework problems using clearly visible comments (e.g., as in `##### Problem 1 #####`).

1 Handwritten digits have never been so fun

Download the file "digits.Rdata" from the course website and load it into your R session with `load("digits.Rdata")`. Now you should have a matrix `threes` of dimension 658×256. (This data set was taken from the data page of `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.) Each row of the matrix corresponds to an image of a "3" that was written by a different person (grabbed from a postal code that they wrote on a letter). Hence each row vector is of length 256, corresponding to a 16×16 pixels image that has been unraveled into a vector, and each pixel takes grayscale values between -1 and 1.

Download the file "plot.digit.R" from the course website and load it into your session with `source("plot.digit.R")`. This gives you a function `plot.digit` that can plot any of the images, i.e., any row of the matrix `threes`. Try it out with `plot.digit(threes[1,])`.

(a) Compute the principal component directions and principal component scores of `threes`. You should use the function `prcomp`, and make sure it knows to center the columns of the data matrix threes, but not to scale them. Plot the first two principal component scores (the x-axis being the first score and the y-axis being the second score). Note that each point in this plot corresponds to an image of a "3".

(b) For each of the first two principal component scores, compute the following percentiles: 5%, 25%, 50%, 75%, 95%. Draw these values as vertical and horizontal lines on top of your plot (i.e., vertical for the percentiles of the first principal component score, and horizontal for those of the second.)

(Hint: use `quantile` for the percentiles, and `abline` to draw the lines.)

(c) Now you want to identify a point (i.e., an image of a "3") close to each of the vertices of the grid on your plot. This can be done by using the `identify` function with `n=25`, which allows you to click on the plot 25 times (since there are 25 vertices). Each time you click, it will print the index of the point that is closest to your click's location. Make sure you click left-to-right, then top-to-bottom, and record the indices in that order.

Note: although the `identify` function returns a vector of indices, and it claims that this vector is ordered by the order of your clicks, I have found that it actually gives them in sorted order. This isn't what you want—you want them in the order that you clicked, so you will have to build this vector manually.

(d) Plot all of the images of "3"s that you picked out in part (c), in an order that corresponds to the vertices of the grid. For example, if you saved the vector of indices that you built in (c) as `inds`, and you built them by clicking left-to-right and top-to-bottom as instructed, this can be done with:

```
par(mfrow=c(5,5))           # allow for 5 x 5 plots
par(mar=c(0.2,0.2,0.2,0.2)) # set small margins
for (i in inds) {
  plot.digit(threes[i,])
}
```

(e) Looking at these digits, what can be said about the nature of the first two principal component scores? (The first principal component score is increasing as you move from left-to-right in any of the rows. The second principal component score is decreasing as you move from top-to-bottom in any of the columns.) In other words, I'm asking you to explain what changes with respect to changes in each of the component scores.

(f) Plot the proportion of variance explained by the first k principal component directions, as a function of $k = 1, \ldots 256$. How many principal component directions would we need to explain 50% of the variance? How many to explain 90% of the variance?

(g) Reproduce the plot you made as in part (d), except, instead of plotting the original "3"s themselves, plot the reconstructed digits using only the first k principal components, where k is be the minimum number of components needed to explain 50% of the variance, as you calculated in part (f). How do they look? Do the same thing, but now for k equal to the number components needed to explain 90% of the variance. Again, how do they look?

Hint: a bit of caution must be taken here with the centering. Recall that the principal components were computed on the centered data matrix `threes`. Call this centered matrix X; then recall that the desired reconstruction is given by $Z = XV_kV_k^T$, where V_k is a matrix whose columns contain the first k principal component directions. Now, you *almost* want to plot select rows of Z, but not quite—first you must add the column means of the threes matrix *back into* Z (otherwise you would be looking at deviations of images from the pixel-wise means, not at images themselves). The `scale` function in R is helpful for adding/subtracting constants to each column of a matrix; e.g.,

```
mat2 = scale(mat, center=TRUE, scale=FALSE)
```

centers a matrix `mat`, and

```
mat3 = scale(mat, center=a, scale=FALSE)
```

subtracts `a[1]` from the first column of `mat`, `a[2]` from the second column of `mat`, and so on, for a vector `a`.

(h) Finally, run hierarchical clustering on the `threes` data set. Use Euclidean distance for the pairwise dissimilarity between images (easiest is just to call `dist`). Run hierarhical clustering with `single` and `complete` linkages. Plot the dendrograms, and describe your impressions of the resulting cluster structures. Do you see evidence of chaining with single linkage? Now, for each method, cut the dendrograms so that there are $K = 2$ clusters. What are the resulting numbers of points in the clusters, for the two methods? Are the clusters balanced for single linkage, complete linkage?