Data Mining: 36-462/36-662
Homework 2

Due Tuesday February 19 2013
(at the beginning of lecture)

Append your R code to the end of your homework. In your solutions, you should just present your R output (e.g., numbers, table, figures) or snippets of R code as you deem it appropriate. Make sure to present your results (i.e., your R ouput) in a clear and readable fashion. Careless or confusing presentations will be penalized.

Problem 1

(a) You're going to write an R function to compute the CH index of K-means clustering assignments, over a range of the number of clusters K. Recall that for a given number of clusters K, we run K-means to get the cluster assignments $C(K)$, and then we compute the CH index as

$$\mathrm{CH}(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)},$$

where $W(K), B(K)$ are the within- and between-cluster variations for $C(K)$.

You're going to use the `kmeans` function to perform K-means clustering. This function returns a list, and two of the items are the within- and between-cluster variations (i.e., it computes these for you). Type `help(kmeans)` to read about the `kmeans` function and what is returns.

Your CH index function should look like:

```
ch.index = function(x, kmax, iter.max=100, nstart=10, algorithm="Lloyd") {
  ch = numeric(length=kmax-1)

  # Your code goes here to build the ch vector

  return(list(k=2:kmax, ch=ch))
}
```

The function takes arguments:

- `x`: the data matrix, which has observations along the rows, and features along the columns.
- `kmax`: this is the maximum number of clusters to consider. You must compute the CH index for the number of clusters $K = 2, 3, \ldots K_{\max}$. (Remember that the CH index is not defined for $K = 1$.)

- `iter.max`: to be passed to the `kmeans` function. This is the maximum number of iterations allowed before the `kmeans` algorithm terminates itself. It defaults to 100 (i.e., this is the default value if the user doesn't specify it when caling the function.)
- `nstart`: to be passed to the `kmeans` function. This is the number of times to run the `kmeans` algorithm with random starts. It defaults to 10.
- `algorithm`: to be passed to the `kmeans` function. This is the type of algorithm used by `kmeans`. It defaults to "Lloyd" (the one we learned in class).

The function returns a list with elements:

- `k`: these are the values of K that were tried, namely, $2, \ldots K_{\max}$.
- `ch.index`: the corresponding CH index scores, as computed by your function.

Because writing this function is the point of this part of the problem, you should give your R code for the `ch.index` function as your solution (i.e., don't just append it at the end of your homework).

Download the file "hw2prob1.Rdata" from the course website and load it into your R session using `load("hw2prob1.Rdata")`. Now you should have three data sets `x1`, `x2`, and `x3`.

(b) Run your `ch.index` function on `x1` with `kmax=10`, and the default options for the rest of the arguments. Plot the results, with the x-axis showing the number of clusters K and the y-axis giving the CH index score $\mathrm{CH}(K)$.

(c) What estimated number of clusters $\hat{K}$ would you choose based on the CH index scores? Rerun `kmeans` on `x1` with $\hat{K}$ as the number of clusters, and with the same options used by your function (`iter.max=100, nstart=10, algorithm="Lloyd"`) to get the cluster centers and cluster assignments. Plot `x1`, plot the cluster centers on top, and either color-code the points (using `col=`) or pch-code the points (using `pch=`) according to cluster membership. Does $\hat{K}$ look like a reasonable choice?

(d) Repeat parts (b) and (c) but on each of `x2` and `x3`. Do the CH index curves give just as obvious answers for the estimated number of clusters as they did for `x1`? What makes these data sets different or harder?

Problem 2

You're going to verify some claims from lectures 7 and 8.

(a) Supppose that the columns of $X \in \mathbb{R}^{n \times p}$ have sample mean zero. Prove that $Xv \in \mathbb{R}^n$ has sample mean zero for any vector $v \in \mathbb{R}^p$.

(b) Suppose that the columns of $X \in \mathbb{R}^{n \times p}$ have been centered (i.e., they have sample mean zero). Recall that:

- The singular value decomposition of X is $X = UDV^T$, where $U \in \mathbb{R}^{n\times p}$, and $D, V \in \mathbb{R}^{p\times p}$. The columns of U are the normalized principal component scores. The columns of V are the principal component directions. The matrix D is diagonal, $D = \text{diag}(d_1, \ldots d_p)$, where $d_1 \geq d_2 \geq \ldots d_p \geq 0$, and $d_j = \|Xv_j\|_2$. You know that V has orthonormal columns, by definition of the principal component directions. It is also true, and you may use this fact, that U has orthonormal columns.

- The total sample variance of X is defined as $\text{trace}(\frac{1}{n}X^TX)$, where the trace of a matrix is simply the sum of its diagonal entries.

Prove that the total sample variance of X is $\frac{1}{n}\sum_{j=1}^{p} d_j^2$.

(Hint 1: use the singular value decomposition of X, $X = UDV^T$, and use the fact that U has orthnormal columns, i.e., $U^TU = I$.)

(Hint 2: you can commute the product of two matrices under the trace operation, i.e., $\text{trace}(AB) = \text{trace}(BA)$.

(c) If again $X = UDV^T$, with centered columns, and $V_k \in \mathbb{R}^{p\times k}$ denotes the first k columns of V, prove that $XV_kV_k^T$ has total sample variance $\frac{1}{n}\sum_{j=1}^{k} d_j^2$.

(Hint: notice that $V_k^TV = [\,I\ \ 0\,]$ where I is the $k \times k$ identity matrix.)

Problem 3

Download the file "hw2prob3.Rdata" from the course website and load it into your R session with `load("hw2prob3.Rdata")`. Now you should have a matrix `threes` that has dimension 658×256. (This data set was taken from the data page on `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. 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`. 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, and 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 may 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?

Problem 4

(a) Suppose that X, Y are independent, discrete random variables. Prove that X, Y are uncorrelated. (Note: you can't just state that independence implies zero covariance—I'm asking you to prove this first step!)

(b) Give an example (either mathematical or empirical) of two random variables X, Y that are uncorrelated but are not independent.

(c) Describe an application (scientific, financial, business, etc.) in which zero correlation and independence really do mean different things, in terms of the application. This could be a real or hypothetical example.

Bonus problem

For a centered data matrix $X \in \mathbb{R}^{n\times p}$ (column-centered), consider the pairwise distances between points $\Delta_{ij} = \|x_i - x_j\|_2$, where $x_1, \dots x_n$ are the rows of X. Recall that in lecture 9, we claimed that the following procedure recovers the matrix of inner-products $B = XX^T \in \mathbb{R}^{n\times n}$ from the distance matrix $\Delta \in \mathbb{R}^{n\times n}$:

1. Define the matrix $A \in \mathbb{R}^{n\times n}$ by $A_{ij} = -\frac{1}{2}\Delta_{ij}^2$.
2. Double center A (center both its columns and rows) to recover B, i.e., take $B = (I - M)A(I - M)$, where $M = \frac{1}{n}\mathbb{1}\mathbb{1}^T$ (this is $1/n$ times the matrix of all 1s).

Here you're going to prove that this procedure indeed gives us $B = XX^T$.

(a) Show that $\Delta_{ij}^2 = B_{ii} + B_{jj} - 2B_{ij}$.

(b) Argue that X being column centered implies that

$$\sum_{\ell=1}^{n} B_{i\ell} = 0 \;\text{ and }\; \sum_{\ell=1}^{n} B_{\ell j} = 0 \;\text{ for any } i, j = 1, \dots n.$$

(c) Starting with the identity in (a), summing over the appropriate indices, and using the results in (b), show that

$$\sum_{i=1}^{n} \Delta_{ij}^2 = \text{trace}(B) + nB_{jj}$$
$$\sum_{j=1}^{n} \Delta_{ij}^2 = nB_{ii} + \text{trace}(B)$$
$$\sum_{i,j=1}^{n} \Delta_{ij}^2 = 2n \cdot \text{trace}(B).$$

(d) Using the three identities in (c), solve for trace(B) and substitute appropriately to conclude that

$$B_{ii} = \frac{1}{n}\sum_{j=1}^{n} \Delta_{ij}^2 - \frac{1}{2n^2}\sum_{i,j=1}^{n} \Delta_{ij}^2$$
$$B_{jj} = \frac{1}{n}\sum_{i=1}^{n} \Delta_{ij}^2 - \frac{1}{2n^2}\sum_{i,j=1}^{n} \Delta_{ij}^2.$$

(e) Plug the identities from (d) into that from (a) to give:

$$B_{ij} = -\frac{1}{2}\Delta_{ij}^2 + \frac{1}{2n}\sum_{i=1}^{n} \Delta_{ij}^2 + \frac{1}{2n}\sum_{j=1}^{n} \Delta_{ij}^2 - \frac{1}{2n^2}\sum_{i,j=1}^{n} \Delta_{ij}^2.$$

(f) Defining $A_{ij} = -\frac{1}{2}\Delta_{ij}^2$ and $M = \frac{1}{n}\mathbb{1}\mathbb{1}^T$, prove that the identity in (e) is the same as

$$B = (I - M)A(I - M).$$