Homework 4

36-350: Data Mining

Due at the start of class Friday, 25 September 2009

This homework set applies methods we have developed so far to a medical problem, gene expression in cancer. In some questions there are calculations which you may find faster to do with R than by hand.

Biological background: A gene is a stretch of DNA inside the cell that tells the cell how to make a specific protein. All cells in the body contain the same genes[1], but they do not always make the same proteins in the same quantities; the genes have different **expression levels** in different cell types, and cells can **regulate** gene expression levels in response to their environment. Different types of cells thus have different expression profiles. Many diseases, including cancer, fundamentally involve breakdowns in the regulation of gene expression. The expression profile of cancer cells becomes abnormal, and different kinds of cancers have different expression profiles.[2]

Our data are gene expression measurements from cells drawn from 64 different tumors (from 64 different patients). In each case, a device called a **microarray** (or **gene chip**) measured the expression of each of 6830 distinct genes[3], essentially the logarithm of the chemical concentration of the gene's product. Thus, each record in the data set is a vector of length 6830. (The website for the data is `http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/nci.info`.)

The cells mostly come from known cancer types, so there are classes, in addition to the measurements of the expression levels. The classes are BREAST, CNS (central nervous system), COLON, LEUKEMIA, MELANOMA, NSCLC (non-small-cell lung cancer), OVARIAN, PROSTATE, RENAL, K562A, K562B, MCF7A, MCF7D (those three are laboratory tumor cultures) and UNKNOWN.

You will need to install the `ElemStatLearn` package from CRAN, and load the data set with the command `data(nci)`. This gives genes as the rows and cells as the columns; transpose it.

[1] Except, oddly, red blood cells, which do not contain any DNA.

[2] A very good place to begin learning more is, in all seriousness, *The Cartoon Guide to Genetics*, by Larry Gonick and Mark Wheelis.

[3] Strictly speaking, genes are first **transcribed** into RNA sequences, which are then **translated** into proteins, and gene chips really measure the RNA level, not the protein level. The difference can be important, but we will not get into that here.

1. *Similarity searching for cells*

   (a) How many features are there in the raw data? Are they discrete or continuous? (If there are some of each, which is which?) If some are continuous, are they necessarily positive?

   (b) Describe a method for finding the k cells whose expression profiles are most similar to that of a given cell. Carefully explain what you mean by "similar". How could you evaluate the success of the search algorithm? (You do not need to code this up.)

   (c) Would it make sense to weight genes by "inverse cell frequency"? Explain.

2. *k-means clustering* Use the `kmeans` function in R to cluster the cells, with $k = 14$ (to match the number of true classes). Repeat this three times to get three different clusterings.

   (a) Say that k-means makes a **lumping error** whenever it assigns two cells of different classes to the same cluster, and a **splitting error** when it puts two cells of the same class in different clusters. Each *pair* of cells can give an error. (With n objects, there are $n(n-1)/2$ distinct pairs.)

      i. Write a function which takes as inputs a vector of classes and a vector of clusters, and gives as output the number of lumping errors. Test your function by verifying that when the classes are $(1, 2, 2)$, the three clusterings $(1, 2, 2)$, $(2, 1, 1)$ and $(4, 5, 6)$ all give zero lumping errors, but the clustering $(1, 1, 1)$ gives two lumping errors.

      ii. Write a function, with the same inputs as the previous one, that calculates the number of splitting errors. Test it on the same inputs. (What should the outputs be?)

      iii. How many lumping errors does k-means make on each of your three runs? How many splitting errors?

   (b) Are there any classes which seem particularly hard for k-means to pick up?

   (c) Are there any pairs of cells which are always clustered together, and if so, are they of the same class?

   (d) *Variation-of-information metric*

      i. Calculate, by hand, the variation-of-information distance between the partition $(1, 2, 2)$ and $(2, 1, 1)$; between $(1, 2, 2)$ and $(2, 2, 1)$; and between $(1, 2, 2)$ and $(5, 8, 11)$.

      ii. Write a function which takes as inputs two vectors of class or cluster assignments, and returns as output the variation-of-information distance for the two partitions. Test the function by checking that it matches your answers in the previous part. (Feel free to

re-use code from lectures 5 and 6 here, but make it clear that's what you are doing.)

iii. Calculate the distances between the k-means clusterings and the true classes, and between the k-means clusterings and each other.

3. *Hierarchical clustering* The command for hierarchical clustering in R is `hclust`. It takes as its argument not the data frame, but rather a matrix of dissimilarities, produced by `dist`. See `help(hclust)` and `help(dist)`.

   (a) Produce dendrograms using Ward's method, single-link clustering and complete-link clustering. Include both the commands you used and the resulting figures in your write-up. Make sure the figures are legible. (Try the `cex=0.5` option to `plot`.)

   (b) Which cell classes seem are best captured by each clustering method? Explain.

   (c) Which method best recovers the cell classes? Explain.

   (d) The `hclust` command returns an object whose `height` attribute is the sum of the within-cluster sums of squares. How many clusters does this suggest we should use, according to Ward's method? Explain. (You may find `diff` helpful.)

   (e) Suppose you did not know the cell classes. Can you think of any reason to prefer one clustering method over another here, based on their outputs and the rest of what you know about the problem?

4. *Retained variance* The function `prcomp` does principal components analysis of its argument, using a linear-algebra technique called "singular value decomposition". For large data sets, this is better than directly taking the eigenvalues of the covariance matrix. One of the attributes of the object it returns is `sdev`, a vector of the standard deviations associated with the principal components, i.e., the square roots of the eigenvalues.

   (a) Use `prcomp` to find the variances (not the standard deviations) associated with the principal components. Include a print-out of these variances, to exactly two significant digits, in your write-up.

   (b) Why are there only 64 principal components, rather than 6830? (*Hint:* Read the lecture notes.)

   (c) Plot the fraction of the total variance retained by the first q components against q. Include both a print-out of the plot and the commands you used.

   (d) Roughly how much variance is retained by the first two principal components?

   (e) Roughly how many components must be used to keep half of the variance? To keep nine-tenths?

   (f) Is there any point to using more than 50 components? (Explain.)

(g) How much confidence should you have in results from visualizing the first two principal components? Why?

5. *Visualization with PCA* If run with the `retx=TRUE` option, one attribute of the object `prcomp` returns is a matrix, `x`, containing the projections of the original data on to the principal components.

   (a) Plot the projection of each cell on to the first two principal components. Label each cell by its type. *Hint:* See the solutions to the first homework. *Note:* ordinarily this could be done through the `biplot` command, but we don't want to see 6380 vectors for the original features.

   (b) One tumor class (at least) forms a cluster in the projection. Say which, and explain your answer.

   (c) Identify a tumor class which does *not* form a compact cluster.

   (d) Of the two classes of tumors you have just named, which will be more easily classified with the prototype method? With the nearest neighbor method?

6. *Combining dimensionality reduction and clustering*

   (a) Run k-means with $k = 14$ on the projections of the cells on to the first two principal components. Give the commands you used to do this, and get three clusterings from three different runs of k-means.

   (b) Calculate the number of errors, as with k-means clustering based on all genes.

   (c) Are there any pairs of cells which are always clustered together? If so, do they have the same cell type?

   (d) Does k-means find a cluster corresponding to the cell type you thought would be especially easy to identify in the previous problem? (Explain your answer.)

   (e) Does k-means find a cluster corresponding to the cell type you thought be be especially hard to identify? (Again, explain.)

7. *Combining dimensionality reduction and classification*

   (a) Calculate the error rate of the prototype method using *all* the gene features, using leave-one-out cross-validation. You may use the solution code from previous assignments, or your own code. Remember that in this data set, the class labels are the row names, not a separate column, so you will need to modify either your code or the data frame. (Note: this may take several minutes to run. [Why so slow?])

   (b) Calculate the error rate of the prototype method using the first two, ten and twenty principal components. Include the R commands you used to do this.

(c) (Extra credit.) Plot the error rate, as in the previous part, against the number q of principal components used, for q from 2 to 64. Include your code and comment on the graph. (*Hint*: Write a function to calculate the error rate for arbitrary q, and use `sapply`.)