Homework 9

36-350: Data Mining

Due at 10 pm on Wednesday, 30 October 2019

AGENDA: Clustering.

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, 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. *Online questions* (10 are online and are due at 10 pm on Sunday, 27 October.

[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.

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. (3) 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. (3) 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. (3) How many lumping errors does k-means make on each of your three runs? How many splitting errors?

   (b) (4) Are there any classes which seem particularly hard for k-means to pick up?

   (c) (3) Are there any pairs of cells which are always clustered together, and if so, are they of the same class?

3. *Variation-of-information metric*

   (a) (3) Calculate 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 $(5, 8, 11)$.

   (b) (4) 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 or previous homeworks here, but make it clear that's what you are doing.)

   (c) (5) Calculate the distances between the true classes and each of your three k-means clusterings for the data, and between the three k-means clusterings.

4. *Hierarchical clustering* The basic 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) (2) 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) (5) Which cell classes seem are best captured by each clustering method? Explain.

(c) (4) Which method best recovers the cell classes? Explain.

(d) (4) 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) (5) 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?

5. *Visualization with PCA* Use `prcomp` to do a principal components analysis of the data set.

   (a) (1) Why are there only 64 principal components, rather than 6830?

   (b) (1) What fraction of the variance is retained by the first two principal components?

   (c) (4) Plot the projection of each cell on to the first two principal components. Label each cell by its type.

   (d) (4) One tumor class (at least) forms a cluster in the projection. Say which, and explain your answer.

   (e) (4) Identify a tumor class which does *not* form a compact cluster in this plot.

   (f) (2) 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* Run k-means with $k = 14$ on the projections of the cells on to the first two principal components; do this three times. Include the code you used to do this.

   (a) (4) Calculate the number of splitting and lumping errors, as you did with k-means clustering based on all the genes. Is this better or worse than using all the genes for clustering?

   (b) (4) Are there any pairs of cells which are always clustered together? If so, do they have the same cell type?

   (c) (4) 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.)

   (d) (4) Does k-means find a cluster corresponding to the cell type you thought be be especially hard to identify? (Again, explain.)

Rubric (10): The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow. Questions which ask for a plot or table are answered with both the figure itself and the command (or commands) use to make the plot. Plots are carefully labeled, with informative and legible titles, axis labels, and (if called for) sub-titles and legends; they are placed near the text of the corresponding problem. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. Code is properly integrated with a tool like R Markdown or knitr, and both the knitted file and the source file are submitted. The code is indented, commented, and uses meaningful names. All code is relevant; there are no dangling or useless commands. All parts of all problems are answered with actual coherent sentences, and raw computer code or output are only shown when explicitly asked for.