OS4118: PC Exercise Fall AY 2020

**Introduction:** In this exercise we will look for structure in a set of documents, using principal components.

**Sample data set:** For this exercise we will use the “daily kos bag of words” data retrieved from the UC Irvine Machine Learning Repository. I have placed this on Sakai under Exercises/DocWord. The investigators acquired 3,430 documents from the political website dailykos. From that they extracted 6,906 words (after removing common “stop” words and other processing) that appeared at least ten times in total. The vocab.kos.txt file contains the list of those words, one per line. The docword.kos.txt file contains a line with 3430, a line with 6906, a line with 353160, and then 353,160 lines, each containing three numbers. Within each line we have doc, word, and count; so (1, 61, 2) tells us that document 1 contained word 61 (“action”) twice. I have also include the readme.txt that came with this data (as you will see, the original data set has five set of text document files, but I’ve only given you the “KOS blog entries” one).

If you want to read the data files in Windows, use Wordpad, not Notepad, since the line endings are Linux-style.

We would like to construct a matrix (call it kos) with 3,430 rows and 6,906 columns, such that the *ij*th entry gives the number of times terms *j* appears in row *i*. I discuss how you might do that below.

**Object:** First notice that the kos matrix is more than 98% empty, because most words are missing from most documents. So how can we characterize documents in a compact way? As you have guessed, I will recommend using principal components.

1.) Perform a principal component decomposition on the kos matrix. Use prcomp() rather than princomp(). (**Note:** This took most of eight minutes on my machine. Expect to wait.) Draw the screeplot and/or picture of (cumulative proportion of variance) against (component number). How many components would you need to keep in order to keep 90% of the variance?

2.) Draw a picture of the scores from the first PC against the scores from the second. Is there any evidence of structure?

3.) Which are the “important” words in the first PC? That is, which are the largest, say, 10 words in terms of their absolute loading? What about in the second PC?

**Getting the data into R:** There are techniques for handling this sort of sparse matrix in R directly, without converting it into a full-sized, mostly empty rectangle, but I don’t want that to distract us. This matrix is small enough that we can operate on it in the usual way.

The “vocab” data can be read in instantly with scan(). In order to get fill the kos matrix efficiently, it will be helpful to learn the following fact: in R, you can use a two-column matrix as an index into another matrix or a numeric data.frame. (To repeat, this index will be a matrix, not a data frame.) Here’s how that might work. Consider this matrix:

> (mat <- matrix (1:12, 4, 3)) # outer parens force print

[,1] [,2] [,3]

[1,] 1 5 9

[2,] 2 6 10

[3,] 3 7 11

[4,] 4 8 12

Now let’s create a two-column matrix to be used as an index. The first column holds the rows, the second, the columns of interest.

> (index <- cbind (c(3, 4, 1), c(3, 2, 2)))

[,1] [,2]

[1,] 3 3

[2,] 4 2

[3,] 1 2

This next command says “show me the (3,3), the (4,2) and the (1,2) entries.”

> mat[index]

[1] 11 8 5

We can use this syntax for replacing, too.

> mat[index] <- c(97, 98, 99)

> mat

[,1] [,2] [,3]

[1,] 1 99 9

[2,] 2 6 10

[3,] 3 7 97

[4,] 4 98 12

This approach provides the mechanism by which to fill our matrix in this example. Read in the docword.kos.txt file as a matrix (use.read.table(), and specify skip=3). Create an empty kos matrix of the proper size. Use the first and second columns of docwords as the index into kos, and the third as the value to be entered.

**Due Date:** Tuesday Oct 8th.