STA 141C - HW 3 - Data Representations and Clustering

Version 1.1

- See Canvas for due dates and grading rubric.
- Turn in a neatly typed 3-5 page report that answers the questions in clear English, using complete sentences.
- Use a format that Canvas can preview: pdf, html, Word are all fine. Raw Jupyter Notebook files and zip files do not work.
- Attach your code in an appendix, or as another file with extension `.R.txt`, so the graders can preview it on Canvas.
- Code must run, and support your answers.
- Use R for this assignment. You're welcome to compare results with another programming language.

Overview

In this assignment we will build and analyze the term document matrix based on the work from the previous assignment. We'll learn about sparse data representations, and experiment with clustering algorithms.

Data

The data is based on the result of hw2, and is available at http://anson.ucdavis.edu/~clarkf/sta141c/

`award_words.zip` contains 3 files:

- `weights.csv`, the word weights in a triple store sparse representation
- `words.csv`, a lookup table of words
- `agencies.csv`, a lookup table of agencies

`weights.csv` represents a a term-document matrix, with agencies as documents and words as terms.

Lookup table means that the value of the `agency_index` column in the `weights` table maps to the row number identifying that agency. For example, an `agency_index` value of 3 means that this is for the third row of the `agency` table, which is the Department of State.

The idea is the same as factors in R. A factor `f` is stored internally as integers, and `levels(f)` returns the lookup table of values.

Questions

1. Object sizes

All of the following questions refer to the `weights` table.

Show your work, i.e. the formulas and expressions you are using. Use the following notation:

- d number of distinct documents / agencies in this data set
- w number of distinct words
- n number of observations in `weights.csv`
- s_i size of an integer (4 bytes in R)
- s_d size of a double precision floating point number (8 bytes in R).

1. What are the advantages and disadvantages of using lookup tables to represent the agencies and words, compared to storing everything in one table?
2. Compute the sizes of the following objects algebraically, and verify the sizes in R. Recall from lecture that there may be around 1 KB memory overhead per object, so your theoretical results will not match exactly.
   - the triple representation in memory (the data frame `weights = read.csv("weights.csv")`)
   - the sparse matrix `X` with columns for each agency and rows for each word (Use `Matrix::sparseMatrix`)
   - the transpose of `X`, with columns for each word and rows for each agency
   - the dense matrix version of `X`
3. Comment on the sizes of the objects you calculated and verified above. Here are some questions to get you thinking:
   - How do the sizes compare to the sparse representation on disk in ASCII text, the `weights.csv` file?
   - What is the sparsity of the matrix? (Sparsity is the number of zero-valued elements divided by the total number of elements)
   - What's the most efficient memory representation for this particular data?
   - Under what conditions would a different representation work better? Could dense ever be better than sparse?

Hints: The `Matrix` package uses a variant of compressed sparse row matrix representation, which you can read about. Inspect the matrices using `str`.

2. Clustering

Let X be the weights matrix with columns for each agency and rows for each word. I normalized X such that columns have L2 norm equal to 1. This lets us compute a measure of similarity or correlation between agencies by taking the dot product of the columns representing those agencies. We can compute all the pairwise similarities simultaneously with $X^T X$. Values range between 0 and 1; a value of 0 means the agencies share no words at all, and a value of 1 means the agencies give exactly the same weight to each word. Thus

$D = 1 - X^T X$ acts like a distance matrix between the agencies, where 1 is a matrix with every entry equal to scalar 1.

1. Is `crossprod` faster than explicitly computing $X^T X$? Why?
2. Is `crossprod` faster on the sparse version of X compared to the dense version of X? Why?
3. What is the range of similarity scores that appear between different agencies? What two agencies are most similar? What does this mean in terms of the words they are using?
4. Fit an agglomerative clustering model using `cluster::agnes(as.dist(D))`. Agglomerative clustering iteratively builds clusters by adding points to groups. What 2 agencies are grouped together first? Is this what you expected based on the previous question?
5. What is the first group of 3 agencies? The first group of 4 agencies? Does agglomerative clustering appear to be doing something reasonable?
6. Fit a partitioning around medoids clustering model using `cluster::pam` with `k = 2` clusters. To what extent do the cluster assignments agree with the agglomerative clustering model?
7. Think about all the steps we've taken in preparing the data and coming this far. Would you say that clustering is a subjective task?

Bonus question

No points, no extra credit This is just something fun for those mathematically inclined.

Consider the setup above. Let vectors x, y be normalized such that $x \cdot x = y \cdot y = 1$ and let

$$d(x, y) = 1 - x \cdot y$$

Is d a distance function a.k.a. metric? If so, prove it. If not, give a counter example.