Evan Batzer

STA 141C

HW3

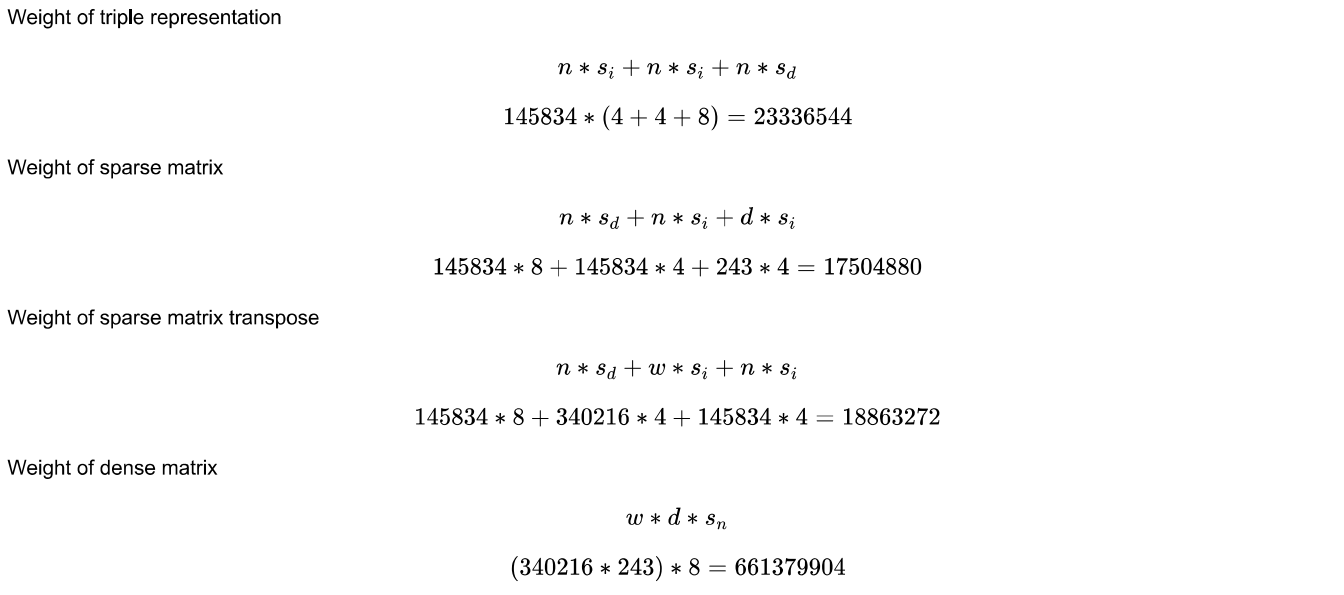
2/3/2019

Object sizes

**1. What are the advantages and disadvantages of using lookup tables to represent the agencies and words, compared to storing everything in one table?**

Using lookup tables to represent agencies and words is likely to speed computation and produces a more size-efficient object that can be manipulated. Calculations occurring on tables only consisting of floats or integers tend to be more efficient than data tables containing character vectors. Attributes of the data table are also shared and can be used across multiple datasets. On the other hand, the set of lookup tables will tend to use more totally memory in storage, as data must contain references to tables, then tables with reference attributes themselves.

**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.**



**Triple Rep Est:** 23336544 bytes (23.3 Mb)

**Triple Rep Obs:** 23337544 bytes (23.3 Mb)

**Sparse Est:** 17503380 bytes (17.5 Mb)

**Sparse Obs:** 17504880 bytes (17.5 Mb)

**Sparse Trans Est:** 18863272 bytes (18.9 Mb)

**Sparse Trans Obs:** 18864776 bytes (18.9 Mb)

**Dense Est:** 661379904 bytes (66.13 Mb)

**Dense Obs:** 661381080 bytes (66.13 Mb)

**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?*

The sparse representation on disk (roughly 17.5 – 18.9 Mb) is considerably smaller than the representation of weights.csv file on disk in ASCII text, which is roughly 44.3 Mb. Surprisingly, the used memory in R of the weights.csv file is nearly half of that on disk (23.3 Mb). I’m not entirely certain of why this would be the case – the file size on disk should be smaller than the memory usage in R, as R should add overhead to each object. However, I think the key is likely the bytes used in data storage of ASCII text, which is 8 bytes per character, rather than the more efficient integer or double-precision formats used by R. The dense matrix, with all its extra zero entries, was the least efficient of all formats.

Source: <http://www2.sluh.org/bioweb/cf/outlines/bitsandbytes.htm>

* *What is the sparsity of the matrix? (Sparsity is the number of zero-valued elements divided by the total number of elements)*

I’ve calculated the sparsity of this matrix to be about 0.982, which is a considerable number of zero-valued entries. This likely indicates that many terms in our data set are used very infrequently and any two agencies are unlikely to have substantial overlap.

* *What’s the most efficient memory representation for this particular data?*

The most efficient memory representation for this particular dataset appears to be the sparse matrix where rows correspond to terms and columns correspond to agencies. While the difference is slight (1.2 Mb) keeping row indices as terms is more efficient.

* *Under what conditions would a different representation work better? Could dense  
  ever be Dense could be better than sparse as the number of non-zero cells in the matrix approaches zero.*

In the sparse representation of the matrices in this example, each matrix object contains a vector of non-zero entries, a vector of the row positions of these non-zero entries, and a vector of column indices. Here, we have matrices with very few non-zero entries, so a set of vectors specifying their positions does not take up too much memory. However, in a case where a majority of cells in the matrix are nonzero, the size of a matrix object with this data storage format will be far greater than a dense matrix, which only needs a vector of entries (i \* j in length) and a 2-d vector specifying its dimensions. This dense representation would likely be more efficient than sparse, as it does not need to include information on all non-zero entry positions.

**Clustering**

1. **Is crossprod faster than explicitly computing XT X? Why?**

***XT X Time (seconds):***

## Min Mean Max

## 0.44 0.46 0.52

***Crossprod Time (seconds):***

## Min Mean Max

## 0.42 0.44 0.47

In this case, the crossprod function is only marginally faster than manually computing XT X (if at all). From crossprod()’s manual: “The functions crossprod and tcrossprod are matrix products or “cross products”, ideally implemented efficiently without computing t(.)’s unnecessarily”. In this case, the time is roughly the same, likely due to the fact that the computation of XT is quick given its sparsity.

1. **Is crossprod faster on the sparse version of X compared to the dense version of X? Why?**

***Crossprod Time (seconds):***

## Min Mean Max

## 0.42 0.44 0.47

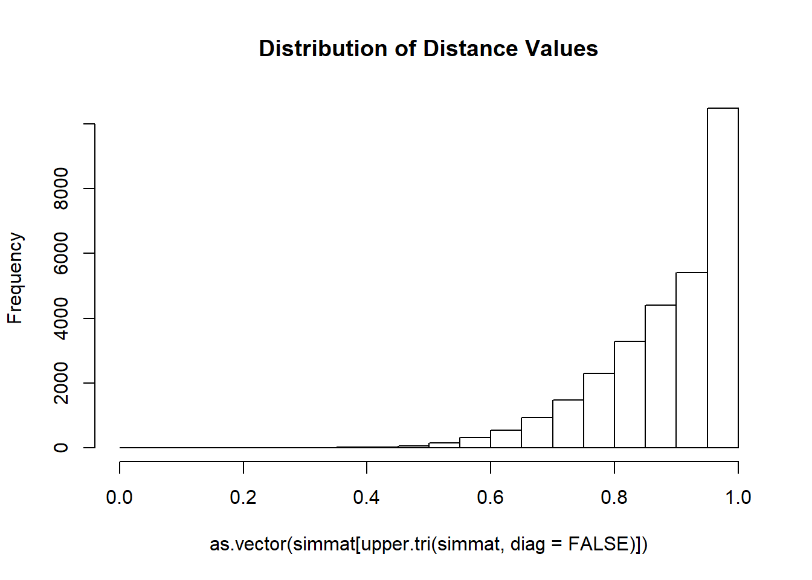
*Dense Version Time* ***(seconds)****:*

## Min Mean Max

## 38.81 39.84 41.63

In another trial, crossprod on the sparse matrix was several orders of magnitude faster than the dense version of X. Calculating this large number of terms from 0,0 value pairs likely slows down the system considerably.

1. **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?**

Dissimilarity scores exist on a distribution between 0 and 1, with a large proportion of the distribution falling towards the higher end of this scale. This indicates that very few organizations tend to share terms (a 1 value) or share very few terms (values close to 1).

The most similar agencies in terms of word usage are the “Office of Job Corps” and the “Employment and Training Administration”. This is perhaps unsurprising given that these two organizations are focused on job training and employment, and thus share a large proportion of their terms.

**Figure 1:** Histogram of similarity scores (1 – XTX) from terms in the “weights.csv” dataset. To reduce repetition, only the upper diagonal values are shown, with no diagonal values (which are all equal to 0).

1. **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?**

The first 2 terms fit were agencies 163 and 122, the “Office of Job Corps” and the “Employment and Training Administration”. This is consistent with the previous observation that these two agencies had the highest similarity scores and the methodology of a hierarchical, agglomerative clustering algorithm.

1. **What is the first group of 3 agencies? The first group of 4 agencies? Does agglomerative clustering appear to be doing something reasonable?**

The first group of 3 agencies was 138, 226, and 180, the “Foreign Agriculture Service”, the “Office of the Inspector General” and the “Office of the Inspector General” (duplicate name in two corpora). The first group of four agencies was the same set in addition to the “Assistant Secretary for Administration”. Agglomerative clustering does appear to be doing something reasonable, as all these administrations appears to have a similar focus on inspection and administration.

1. **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?**

There is only limited similarity in these two clusering assignments. By and large, PAM and Agnes both group a large number of organizations together, but disagree considerably on the membership of a second, smaller group. This is likely to arise due to the difficulty in determining groups in datasets where so few pairs of corpora share a large fraction of terms – it is likely very challenging to find a meaningful partition.

**Figure 2:** Barplot of group assignment frequencies as determined by PAM (x-axis) and Agnes (bar fill). High agreement in assignment will be indicated by bars filled with one consistent color.

1. **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?**

Clustering is definitely a subjective task. As mentioned before, this seems to be particularly difficult in situations where the data does not readily form easily distinguished groups. While it may be possible to find small clusters of similarity in our data, broad distinctions into groups “1” or “2” is not likely to provide meaningful clustering.