Homework 3 Code

Jared Yu

February 4, 2019

```
library(Matrix); library(cluster); library(microbenchmark) # Load libraries
### 1 Object sizes
# Load data
weights_file <- data.frame(read.csv(unz('award_words.zip', 'weights.csv')))
words_file <- read.table(unz('award_words.zip', 'words.csv'))
agencies_file <- read.csv(unz('award_words.zip', 'agencies.csv'))

# Notation for algebra
d <- length(unique(weights_file$agency_index))
n <- nrow(weights_file)
w <- length(unique(weights_file$word_index))
s_i <- 4
s_d <- 8

# 2 / 3
# create sparse matrix, transpose matrix, and dense matrix
# also calculate difference in estimated vs. actual sizes
weights_file_size <- object.size(weights_file) # csv dataframe
expected_size <- (n * 2 * s_i) + (n * 1 * s_d)
expected_size - weights_file_size


sparse_mat <- sparseMatrix(i = weights_file$word_index, # sparse matrix
                           j = weights_file$agency_index,
                           x = weights_file$weight)
expected_size_sparse <- (n * s_d) + ((d + 1) * s_i) + (n * s_i)
actual_sparse_size <- object.size(sparse_mat)
expected_size_sparse - actual_sparse_size

sparse_mat_t <- t(sparse_mat) # transpose matrix
expected_size_sparse_t <- (n * s_d) + ((w + 1) * s_i) + (n * s_i)
actual_sparse_t_size <- object.size(sparse_mat_t)
expected_size_sparse_t - actual_sparse_t_size
weights_file_size - actual_sparse_t_size

dense_mat <- as(sparse_mat, "dgeMatrix") # dense matrix
expected_dense_size <- w * d * s_d
actual_dense_size <- object.size(dense_mat)
expected_dense_size - actual_dense_size
weights_file_size - actual_dense_size

# 3
# compare each object with the disk size of weights.csv
weights_disk_size <- as.numeric(file.info('weights.csv')[1])
weights_disk_size - weights_file_size
weights_disk_size - actual_sparse_size
weights_disk_size - actual_sparse_t_size
```

```r
weights_disk_size - actual_dense_size

# sparsity of matrix
1 - (n / (sparse_mat@Dim[1] * sparse_mat@Dim[2]))

### 2 Clustering
# calculate run times
manual_time <- microbenchmark(t(sparse_mat)%*%sparse_mat)
function_time <- microbenchmark(crossprod(x = sparse_mat))
function_time_dense <- system.time(crossprod(x = dense_mat))
manual_time_dense <- system.time(t(dense_mat)%*%dense_mat)

# Create D
D <- 1 - crossprod(sparse_mat)

# find closest/most similar elements
diag(D) <- -1 # make diagonals negative
which(D == min(D[D > 0]), # exclude diagonals from search
      arr.ind = TRUE)

# range of distances
range(D[D > 0])

# create hierarchical clusters
cluster1 <- agnes(as.dist(D), method = 'simple')
cluster2 <- agnes(as.dist(D), method = 'complete')

# analyze the first group of 3 and 4 agencies
head(cluster1$merge, 20)
head(cluster2$merge, 20)

# analyze the agencies which are clustered together
agencies_file[148,]; agencies_file[226,]; agencies_file[180,]
agencies_file[218,]; agencies_file[211,]

# separate the clusters into 2 groups
cutree1 <- cutree(cluster1 , k = 2)
cutree2 <- cutree(cluster2 , k = 2)
table(cutree2)

# create pam cluster
pam1 <- pam(D, k = 2)
table(pam1$clustering)

# Analyze overlap between clusters
sum(which(cutree2 == 2)%in%which(pam1$clustering == 2)) # 28 not in pam
```