Stat102C_HW_6

Junhyuk Jang

6/6/2018

The assignment is to code up the Gibbs sampler algorithm as described in the document "Gibbs Sampling for the Uninitiated"

The psuedo-code in section 2.5.4 will be very helpful.

The article describes using the algorithm for sentiment analysis: either classifying a document as having a positive or negative sentiment.

I have chosen to simply the task by using a set of 'documents' that contain a much smaller vocabulary. I took the text from two popular children's books: *Green Eggs and Ham* by Dr. Seuss, and *Go Dog, Go!* by P.D. Eastman. (They are also popular with my daughter.)

I took the raw text of the book, and chopped each book into about 10 chunks. I'm calling each of these text chunks a "document." The algorithm will then take the unlabeled chunks and sort them into two groups based on text similarity.

I have written the commands that create the corpus of "documents." The commands make use of the library `tm` (text mining) and `SnowballC` (truncating words to their roots).

If you code it correctly, the algorithm should be able to do a decent job of classifying the documents despite the data being completely unlabeled.

Keep track of the vector `L` (the document labels) after each iteration. We will want this to get a distribution of `L`. When you keep track of `L`, you will have a NxT matrix with a column for each document (N documents) and a row for each iteration (a total of T iterations).

At the end, print out the vector `L` which shows the current classifications after the final iteration. Also print out the vectors theta_0 and theta_1. Keep in mind, these are just random draws from the posterior distribution, and will not necessarily be the values that maximize the posterior probability. They should, however, be drawn from regions of fairly high probability and reflect values that are close to the true values. (In our simple example, however, the vector L should correctly classify the documents.)

After you have kept track of all the different `L` vectors that get sampled, discard the first handful of iterations until you see the vector `L` reaches stability.

Include any plots or other output if you feel they help show the success of the algorithm.

Please do not print out the results of each iteration or do something that produces many lines of unnecessary output.

```r
setwd("~/Desktop/FROM OLD MAC/FOLDERS FROM MAC/UCLA_Academic/Spring 2018/Stat_102C/HW/")
library(tm)
```

```
## Loading required package: NLP
```

```r
library(SnowballC)
```

```
## Warning: package 'SnowballC' was built under R version 3.5.2
```

```r
geah <- readLines("geah.txt")  # text of Green Eggs and Ham by Dr. Seuss
gdg <- readLines("gdg.txt")    # text of Go Dog Go by P.D. Eastman
corpus <- c(geah, gdg)
```

```r
doc.vec <- VectorSource(corpus)
doc.corpus <- Corpus(doc.vec)
# summary(doc.corpus)
doc.corpus <- tm_map(doc.corpus, tolower)
```

```
## Warning in tm_map.SimpleCorpus(doc.corpus, tolower): transformation drops
## documents
```

```r
doc.corpus <- tm_map(doc.corpus, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(doc.corpus, removePunctuation):
## transformation drops documents
```

```r
doc.corpus <- tm_map(doc.corpus, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(doc.corpus, removeNumbers): transformation
## drops documents
```

```r
doc.corpus <- tm_map(doc.corpus, removeWords, stopwords("english"))  # removes very common English word
```

```
## Warning in tm_map.SimpleCorpus(doc.corpus, removeWords,
## stopwords("english")): transformation drops documents
```

```r
doc.corpus <- tm_map(doc.corpus, stemDocument)  # stems words so that words like running, runs, runner
```

```
## Warning in tm_map.SimpleCorpus(doc.corpus, stemDocument): transformation
## drops documents
```

```r
doc.corpus <- tm_map(doc.corpus, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(doc.corpus, stripWhitespace): transformation
## drops documents
```

```r
inspect(doc.corpus[4])  # resulting text for one 'document'
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1
##
## [1] like box like fox like hous like mous like like anywher like green egg ham like sam
```

```r
inspect(doc.corpus[15])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1
##
## [1] hello hello like hat like hat goodby goodby dog car go away go away fast look dog go go dog go s
```

```r
DTM <- DocumentTermMatrix(doc.corpus) # creates a matrix of the words and their frequencies in each doc
DTM
```

```
## <<DocumentTermMatrix (documents: 19, terms: 59)>>
## Non-/sparse entries: 247/874
## Sparsity           : 78%
## Maximal term length: 7
## Weighting          : term frequency (tf)
```

```r
inspect(DTM[1:10,1:10])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 10)>>
## Non-/sparse entries: 77/23
## Sparsity           : 23%
## Maximal term length: 7
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs anywher box car egg green ham hous like mous sam
##   1        1   0   0   3     3   3    0    9    0   8
##   10       1   1   0   1     1   1    1    1    1   1
##   2        1   0   0   1     1   1    2    8    2   1
##   3        1   2   3   1     1   1    1    3    1   1
##   4        1   1   0   1     1   1    1    8    1   1
##   5        1   1   1   1     1   1    1    1    0   2
##   6        1   1   1   1     1   1    1    4    1   2
##   7        0   0   1   0     0   0    0    0    0   0
##   8        1   1   0   1     1   1    1    8    1   2
##   9        0   0   1   1     1   1    0    2    0   1
```

```
word_counts <- as.matrix(DTM) # You can now use this as it shows the frequency of each word
dim(word_counts) ## our corpus has 19 documents, and a total of 59 unique words
```

```
## [1] 19 59
```

```
head(word_counts) ## a visual inspection already shows some patterns in the word usage
```

```
##     Terms
## Docs anywher egg green ham like sam hous mous box car eat fox let may see
##    1       1   3     3   3    9   8    0    0   0   0   0   0   0   0   0
##    2       1   1     1   1    8   1    2    2   0   0   0   0   0   0   0
##    3       1   1     1   1    3   1    1    1   2   3   7   2   1   2   1
##    4       1   1     1   1    8   1    1    1   1   0   0   1   0   0   0
##    5       1   1     1   1    1   2    1    0   1   1   4   1   1   0   0
##    6       1   1     1   1    4   2    1    1   1   1   1   1   0   0   1
##     Terms
## Docs tree will train dark rain say boat goat tri good thank big black blue
##    1    0    0     0    0    0   0    0    0   0    0     0   0     0    0
##    2    0    0     0    0    0   0    0    0   0    0     0   0     0    0
##    3    2    1     0    0    0   0    0    0   0    0     0   0     0    0
##    4    0    0     0    0    0   0    0    0   0    0     0   0     0    0
##    5    1    3     6    0    0   0    0    0   0    0     0   0     0    0
##    6    1    1     1    5    2   1    0    0   0    0     0   0     0    0
##     Terms
## Docs dog goodby hat hello littl one red three white yellow around two
##    1   0      0   0     0     0   0   0     0     0      0      0   0
##    2   0      0   0     0     0   0   0     0     0      0      0   0
##    3   0      0   0     0     0   0   0     0     0      0      0   0
##    4   0      0   0     0     0   0   0     0     0      0      0   0
##    5   0      0   0     0     0   0   0     0     0      0      0   0
##    6   0      0   0     0     0   0   0     0     0      0      0   0
##     Terms
## Docs water hot night now parti play sun work away fast light look stop
##    1     0   0     0   0     0    0   0    0    0    0     0    0    0
##    2     0   0     0   0     0    0   0    0    0    0     0    0    0
##    3     0   0     0   0     0    0   0    0    0    0     0    0    0
##    4     0   0     0   0     0    0   0    0    0    0     0    0    0
##    5     0   0     0   0     0    0   0    0    0    0     0    0    0
##    6     0   0     0   0     0    0   0    0    0    0     0    0    0
##     Terms
## Docs sleep time top day get
##    1     0    0   0   0   0
##    2     0    0   0   0   0
##    3     0    0   0   0   0
##    4     0    0   0   0   0
##    5     0    0   0   0   0
##    6     0    0   0   0   0
```

```r
tail(word_counts)
```

```
##     Terms
## Docs anywher egg green ham like sam hous mous box car eat fox let may see
##   14       0   0     0   0    0   0    2    0   0   0   0   0   0   0   0
##   15       0   0     1   0    2   0    0    0   0   1   0   0   0   0   0
##   16       0   0     0   0    0   0    0    0   0   0   0   0   0   0   0
##   17       0   0     0   0    0   0    0    0   0   1   0   0   0   0   0
##   18       0   0     0   0    0   0    0    0   0   1   0   0   0   0   0
##   19       0   0     1   0    3   0    0    0   0   0   0   0   0   0   0
##     Terms
## Docs tree will train dark rain say boat goat tri good thank big black blue
##   14    0    0     0    0    0   0    1    0   0    0     0   0     0    0
##   15    0    0     0    0    0   0    0    0   0    0     0   0     0    0
##   16    0    1     0    0    0   0    0    0   0    0     0   0     0    0
##   17    1    0     0    0    0   0    0    0   0    0     0   1     0    0
##   18    6    2     0    0    0   0    0    0   0    0     0   0     0    0
##   19    0    0     0    0    0   0    0    0   0    0     0   2     1    1
##     Terms
## Docs dog goodby hat hello littl one red three white yellow around two
##   14   5      0   0     0     0   0   0     1     0      2      0   0
##   15   5      2   2     2     0   0   1     0     0      0      0   0
##   16   3      0   0     0     0   0   0     0     0      0      0   1
##   17   4      0   0     0     0   0   0     0     0      0      0   0
##   18   2      0   0     0     0   0   0     0     0      0      0   0
##   19  11      1   3     1     1   0   1     0     1      1      0   0
##     Terms
## Docs water hot night now parti play sun work away fast light look stop
##   14     0   2     2   1     1    3   4    3    0    0     0    0    0
##   15     0   0     0   2     0    0   0    0    2    1     2    1    2
##   16     0   0     3   1     0    4   0    0    0    0     0    0    0
##   17     0   0     0   2     0    0   1    0    0    1     0    2    0
##   18     0   0     0   2     0    1   0    1    0    0     0    1    1
##   19     0   0     0   1     4    0   0    0    0    0     0    0    0
##     Terms
## Docs sleep time top day get
##   14     0    0   0   0   0
##   15     0    0   0   0   0
##   16     3    2   1   0   0
##   17     0    2   0   2   3
##   18     0    0   2   0   1
##   19     0    0   0   0   0
```

```
# initial parameters
V = dim(word_counts)[2] # number of words in the vocabulary
N = dim(word_counts)[1] # number of documents in the corpus
gamma_pi_1 = 5 # hyper parameter for pi
gamma_pi_0 = 5 # hyper parameter for pi
gamma_theta = rep(1, V) # vector of hyper parameters for the vector theta

library(MCMCpack)
```

```
## Loading required package: coda

## Warning: package 'coda' was built under R version 3.5.2

## Loading required package: MASS

## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2019 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

## ##
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
set.seed(1)
# randomly initialize the label assignments
pi <- rbeta(1, gamma_pi_0, gamma_pi_1)
L <- rbinom(N, 1, pi)
theta_0 <- rdirichlet(1, gamma_theta)
theta_1 <- rdirichlet(1, gamma_theta)
```

```
## Write the code for the gibbs sampler here
## it might take a long time to run
p <- 0
q <- 0
c_0 <- NA
c_1 <- NA
N <- c_0 + c_1
l <- matrix(rep(NA),nrow=500,ncol=19)
L_0 <- matrix(rep(NA),nrow=19, ncol=59)
L_1 <- matrix(rep(NA),nrow=19, ncol=59)
for(i in 1:nrow(l)){
        l[i,] <-as.matrix(L)
        {
             for(j in 1:19){
                if(L[j] == "1"){
                    p <- p + 1
                        }
                else if(L[j] == "0"){
                        q <- q + 1
                  }
                        c_0 <- q
                        c_1 <- p
                        N <- c_0 + c_1
        }
```

```
                {
                 for(k in 1:19){
                  l_0 <- ((c_0 + gamma_pi_0 - 1) / (N + gamma_pi_0 + gamma_pi_1 - 1))*
                          prod(theta_0^word_counts[k,])
                  l_1 <- ((c_1 + gamma_pi_1 - 1) / (N + gamma_pi_0 + gamma_pi_1 - 1))*
                          prod(theta_1^word_counts[k,])
                  prob1 <- l_1 / (l_0 + l_1)
                        L[k] <- rbinom(1,1,prob = prob1)
                        if(L[k] == "1"){
                                L_1[k,] <- as.numeric(word_counts[k,])
                        }
                        else if (L[k] == "0"){
                                L_0[k,] <- as.numeric(word_counts[k,])
                        }
                 }
            }

        }
                 p <- 0
                 q <- 0
                 vector_c_1 <- colSums(L_1,na.rm = T)
                 vector_c_0 <- colSums(L_0,na.rm = T)
                 f0 <- gamma_theta + vector_c_0
                 f1 <- gamma_theta + vector_c_1
                 theta_0 <- rdirichlet(1, f0)
                 theta_1 <- rdirichlet(1, f1)
}


# current classifications after the final iteration
l[500,]
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
```

```
# theta_0
theta_0
```

```
##             [,1]       [,2]       [,3]       [,4]      [,5]       [,6]
## [1,] 0.03150037 0.01718447 0.03979378 0.04014783 0.1020881 0.04034596
##            [,7]       [,8]       [,9]      [,10]      [,11]      [,12]
## [1,] 0.0380123 0.02317899 0.01587424 0.01488148 0.02534166 0.01300102
##             [,13]      [,14]       [,15]     [,16]      [,17]      [,18]
## [1,] 0.003540989 0.01043471 0.009497136 0.0217605 0.03768227 0.02103185
##             [,19]       [,20]      [,21]       [,22]       [,23]
## [1,] 0.005274559 0.002613212 0.00875899 0.002511041 0.003748503
##             [,24]       [,25]      [,26]      [,27]       [,28]       [,29]
## [1,] 0.008069599 0.002128548 0.01573071 0.04049989 0.008515304 0.008805333
##            [,30]      [,31]      [,32]      [,33]      [,34]       [,35]
## [1,] 0.08523952 0.01930591 0.01252324 0.01259223 0.01129925 0.009042571
##             [,36]       [,37]       [,38]      [,39]       [,40]
## [1,] 0.003394425 0.009420006 0.007103596 0.01028125 0.009152162
##             [,41]      [,42]       [,43]      [,44]      [,45]       [,46]
## [1,] 0.008260051 0.01466074 0.002250937 0.02258551 0.01988223 0.007136419
##            [,47]       [,48]      [,49]       [,50]       [,51]      [,52]
## [1,] 0.01442425 0.008795865 0.01032464 0.007290938 0.009056848 0.01040333
##             [,53]       [,54]       [,55]      [,56]      [,57]       [,58]
## [1,] 0.004933814 0.009830953 0.006547936 0.00935006 0.01039289 0.004720134
##            [,59]
## [1,] 0.01784493
```

```
# theta_1
theta_1
```

```
##             [,1]       [,2]       [,3]       [,4]      [,5]       [,6]
## [1,] 0.01513624 0.02277757 0.02949296 0.04395065 0.1202265 0.05154394
##             [,7]       [,8]       [,9]      [,10]      [,11]      [,12]
## [1,] 0.04286421 0.01453835 0.01275953 0.04753028 0.07112485 0.03421029
##           [,13]       [,14]      [,15]      [,16]      [,17]      [,18]
## [1,] 0.0137797 0.005333264 0.01532219 0.02387482 0.05089472 0.01347695
##            [,19]      [,20]      [,21]     [,22]       [,23]       [,24]
## [1,] 0.02068597 0.02237944 0.02440264 0.0101956 0.006567119 0.008467679
##             [,25]      [,26]        [,27]       [,28]       [,29]
## [1,] 0.008789996 0.00797958 0.0001160498 0.004890037 0.003915797
##            [,30]     [,31]       [,32]       [,33]       [,34]        [,35]
## [1,] 0.04488203 0.0138327 0.006611487 0.002312808 0.001489496 0.0004511357
##             [,36]      [,37]       [,38]       [,39]     [,40]       [,41]
## [1,] 0.001132626 0.01533022 0.001124467 0.005756879 0.0115504 0.008817999
##             [,42]      [,43]       [,44]       [,45]      [,46]      [,47]
## [1,] 0.006632936 0.01049671 0.008201529 0.002400119 0.01106299 0.01762298
##            [,48]      [,49]       [,50]       [,51]       [,52]       [,53]
## [1,] 0.01388999 0.02171526 0.003515475 0.003071531 0.003944235 0.002897017
##             [,54]      [,55]       [,56]       [,57]       [,58]
## [1,] 0.002888668 0.01190176 0.003166205 0.005165753 0.001391386
##             [,59]
## [1,] 0.009516301
```

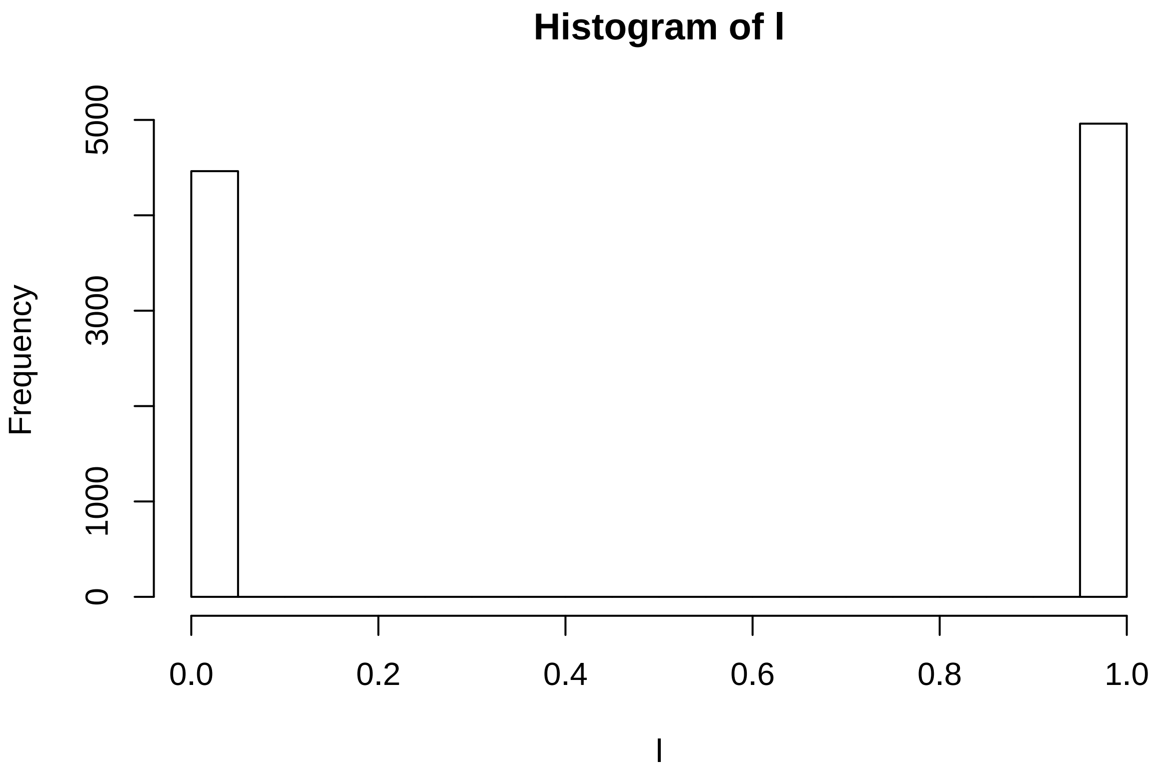
```
# Discard the first handful of iterations until you see the vector `L` reaches stability.
head(l,10)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    0    1    0    1    1    1    1    0    0     0     1     0     1
##  [2,]    0    1    1    0    0    0    1    0    1     0     0     0     1
##  [3,]    0    0    1    0    1    1    1    0    1     0     0     0     1
##  [4,]    0    1    1    1    1    1    1    1    1     1     0     0     0
##  [5,]    1    1    1    1    1    1    1    1    1     1     0     0     0
##  [6,]    1    1    1    1    1    1    1    1    1     1     0     0     0
##  [7,]    1    1    1    1    1    1    1    1    1     1     0     0     0
##  [8,]    1    1    1    1    1    1    1    1    1     1     0     0     1
##  [9,]    1    1    1    1    1    1    1    1    1     1     0     0     0
## [10,]    1    1    1    1    1    1    1    1    1     1     0     0     0
##       [,14] [,15] [,16] [,17] [,18] [,19]
##  [1,]     0     1     1     0     1     1
##  [2,]     1     0     0     0     0     0
##  [3,]     1     0     0     0     0     0
##  [4,]     1     0     0     0     0     0
##  [5,]     1     0     0     0     0     0
##  [6,]     1     0     0     0     0     0
##  [7,]     1     0     0     0     0     0
##  [8,]     1     0     0     0     0     0
##  [9,]     1     0     0     0     0     0
## [10,]     0     0     0     0     0     0
```

```r
# After 4 iteration it becomes stable. I will discard first four.
l <- l[-c(1:4),]
head(l)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    1    1    1    1    1    1    1    1    1     1     0     0     0
## [2,]    1    1    1    1    1    1    1    1    1     1     0     0     0
## [3,]    1    1    1    1    1    1    1    1    1     1     0     0     0
## [4,]    1    1    1    1    1    1    1    1    1     1     0     0     1
## [5,]    1    1    1    1    1    1    1    1    1     1     0     0     0
## [6,]    1    1    1    1    1    1    1    1    1     1     0     0     0
##      [,14] [,15] [,16] [,17] [,18] [,19]
## [1,]     1     0     0     0     0     0
## [2,]     1     0     0     0     0     0
## [3,]     1     0     0     0     0     0
## [4,]     1     0     0     0     0     0
## [5,]     1     0     0     0     0     0
## [6,]     0     0     0     0     0     0
```

```r
# histogram
hist(l)
```



```r
# column means
colMeans(l)
```

```
##  [1] 0.99193548 0.98387097 1.00000000 0.99193548 0.99798387 1.00000000
##  [7] 1.00000000 1.00000000 1.00000000 1.00000000 0.00000000 0.00000000
## [13] 0.01008065 0.02620968 0.00000000 0.00000000 0.00000000 0.00000000
## [19] 0.00000000
```