CS 410: Text Information Systems
University of Illinois at Urbana-Champaign
Fall 2022

Technology Review

Submitted by: **Samuel Kirby**
slkirby2@illinois.edu

November 4, 2022

Background

The goal of clustering and clustering algorithms is to organize observations into groups in an unsupervised fashion, and have been widely used for many years in applications ranging from customer segmentation to anomaly detection (such as fraud). The essential notion is that the members of a cluster share more characteristics than those of other clusters; e.g., some sort of similarity measure is required. Interestingly, clustering analysis was initially used within the social sciences, including anthropology and psychology, dating back to the 1930's.[1] The popular K-Means clustering algorithm dates to the 1950s, when it was developed for the domain of signal processing at Bell Labs.[2]

Today, there are many types of clustering algorithms, with different applications and attributes ranging from scalability to large datasets, tolerance of outliers, and the dimensionality of data.[3]

This review will focus on two approaches: the widely-used K-means algorithm, as well as an EM (expectation-maximization) approach.

The popularity of the K-means algorithm is driven by its scalability and efficiency given large datasets. However, it does have some disadvantages. For example, the K-means algorithm is sensitive to the "starting points" of the K centroids. For this reason, "smart" starting points are sometimes used. In addition, K-means is also sensitive to outliers because it assigns each observation to the closest cluster – in other words, it uses a hard decision rule. Further, each data point influences the mean equally, which could affect the specificity of the algorithm in cases where data is more ambiguous.

This is where Expectation Maximization may offer an advantage in some applications as it has the ability to generate "fuzzy" or soft clusters. It accomplishes this through the use of mixtures of cluster

---

[1] Cluster analysis - Wikipedia
[2] k-means clustering - Wikipedia

[3] Table 22 | A Comprehensive Survey of Clustering Algorithms | SpringerLink

distributions. As a result, EM text cluster does not actually assign observations to a single cluster; rather, it generates the probabilities of cluster membership.

The EM (Expectation-Maximization) model was introduced in 1977 in a paper in the Journal of the Royal Statistical Society.[4] EM models in general, as discussed in this course, are used to find the maximum likelihood of parameters given the presence of "hidden variables" that are unobserved. To do this, EM models alternate between an "E" (expectation) step, which calculates likelihood assuming that the hidden variables were observed, followed by the "M" (maximization) step that maximizes the likelihoods found in the E step. The process is then repeated, and the parameters found in the M step are used to initialize another E step.

Evaluation of K-Means and EM Text Clustering in R

Because of its maturity, there are ample libraries for K-Means based text clustering in R. I could not locate similar libraries for the EM approach, however I found an article that walked through the process with code examples. Using this example, I was able to conduct a side-by-side comparison using the BBC dataset of 2226 articles spanning five categories. However, the categories themselves will not be used as these are unsupervised algorithms.[5]

The remaining pages of this review represent the R Markdown document illustrating the implementation of these two clustering approaches and their output, using a sample dataset of more than 2,000 BBC articles.

Other Sources:

- Text Clustering using K-means. Complete guide on a theoretical and… | by Kajal Yadav | Towards Data Science
- A Comprehensive Survey of Clustering Algorithms | SpringerLink
- Expectation-Maximization for News Clustering | Welcome to my Website (alangewerc.com)
- Introductory tutorial to text clustering with R (rstudio-pubs-static.s3.amazonaws.com)

[4] Microsoft PowerPoint - Lecture3_statnlp_web.ppt (columbia.edu)
[5] BBC News Classification | Kaggle

Technology Review: K-Means and EM Text Clustering

CS410, Text Information Systems Fall 2022

Sam Kirby, slkirby2@illinois.edu 2022-11-04

EM-based text clustering approach

Code source: http://www.alangewerc.com/blog/EM-Clustering/#:~:text=The%20goal%20of%20EM%20clustering,different%20distributions%20in%20different%20clusters

Data source: https://www.kaggle.com/c/learn-ai-bbc

```
library(reshape2) # for melt and cast functions
```

```
## Warning: package 'reshape2' was built under R version 4.2.2
```

```
library(ggplot2) # for plotting functions
library(repr) # to resize the plots
```

```
## Warning: package 'repr' was built under R version 4.2.2
```

```
library(NLP) # natural language preprocessing
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(tm) # text mining library
```

```
## Warning: package 'tm' was built under R version 4.2.2
```

```
# for k-means
library(tm)
library(dbscan)
```

```
## Warning: package 'dbscan' was built under R version 4.2.2
```

```
##
## Attaching package: 'dbscan'
```

```
## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

```
library(proxy)
```

```
## Warning: package 'proxy' was built under R version 4.2.2

##
## Attaching package: 'proxy'

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

library(colorspace)
```

Helper function

```
logSum <- function(v) {
   m = max(v)
   return ( m + log(sum(exp(v-m))))
}
```

Initialize Model Parameters Randomly

```
initial.param <- function(vocab_size, K=4, seed=123456){
  set.seed(seed)
  rho <- matrix(1/K,nrow = K, ncol=1) # assume all clusters have the same siz
e (we will update this later on)
  mu <- matrix(runif(K*vocab_size),nrow = K, ncol = vocab_size)    # initiate
Mu
  mu <- prop.table(mu, margin = 1)   # normalization to ensure that sum of ea
ch row is 1
  return (list("rho" = rho, "mu"= mu))
}
```

E Step

```
E.step <- function(gamma, model, counts, soft = TRUE){
  # Model Parameter Setting
  N <- dim(counts)[2] # number of documents
  K <- dim(model$mu)[1]

  # E step:
  for (n in 1:N){
    for (k in 1:K){
      ## calculate the posterior based on the estimated mu and rho in the "lo
g space"
      gamma[n,k] <- log(model$rho[k,1]) +  sum(counts[,n] * log(model$mu[k,])
)
    }
    # normalisation to sum to 1 in the log space
    logZ = logSum(gamma[n,])
    gamma[n,] = gamma[n,] - logZ
```

```
  }

  # converting back from the log space
  gamma <- exp(gamma)
  # if it is hard EM, we need to select the K with highest gamma
  if(soft == FALSE){
      z_star = max.col(gamma, 'first') # gets the "argmax" class for each obs
ervation
  }
  # if we are doing Hard-EM, we return "z_star", which are the classes assign
ed for each n,
  # if we are doing soft-EM, we return "gamma", which is the matrix with post
erior probabilities for each k,n
  if(soft==FALSE){
      return(z_star)
  }else{return(gamma)}
}
```

M-Step

```
M.step <- function(gamma.z_star, model, counts, soft = TRUE){
  # Model Parameter Setting
  N <- dim(counts)[2]   # number of documents
  W <- dim(counts)[1]   # number of words i.e. vocabulary size
  K <- dim(model$mu)[1] # number of clusters
  # to avoid NaN where all elements are zeros when calculating the new  means
  eps = matrix(1e-10, nrow = W, ncol = N)

  if(soft== TRUE){ # Soft-EM
      gamma = gamma.z_star
      for (k in 1:K){
          ## recalculate the estimations:
          model$rho[k] <- sum(gamma[,k])/N  # the relative cluster size
          model$mu[k,] <- ((counts%*%gamma[,k])+eps[,k])/sum((counts%*%gamma[
,k])+eps[,k]) # new means

    }
  }else{ # Hard-EM
      z_star = gamma.z_star
      for (k in 1:K){
          ## recalculate the estimations:
          ## recalculate the estimations:
          model$rho[k] <- sum(z_star==k)/N  # the relative cluster size
          model$mu[k,] <- rowSums(counts[,z_star==k]+eps[,z_star==k])/sum(row
Sums(counts[,z_star==k]+eps[,z_star==k]))
          # new means
    }

  }

  # Return the result
```

```
  return (model)
}
```

Training Objective Function

```
train_obj <- function(model, counts) {
  N <- dim(counts)[2] # number of documents
  K <- dim(model$mu)[1]

  nloglike = 0
  for (n in 1:N){
    lprob <- matrix(0,ncol = 1, nrow=K)
    for (k in 1:K){
      lprob[k,1] = sum(counts[,n] * log(model$mu[k,]))
    }
    nloglike <- nloglike - logSum(lprob + log(model$rho))
  }

  return (nloglike)
}
```

EM for Document Clustering

```
EM <- function(counts, K=4, max.epoch=10, seed=123456, soft = TRUE){

  # Model Parameter Setting
  N <- dim(counts)[2] # number of documents
  W <- dim(counts)[1] # number of unique words (in all documents)

  # Initialization
  model <- initial.param(W, K=K, seed=seed)
  gamma <- matrix(0, nrow = N, ncol = K)

  print(train_obj(model,counts))
  # Build the model
  for(epoch in 1:max.epoch){

    # E Step
    gamma_kmax <- E.step(gamma, model, counts, soft = soft)
    # M Step
    model <- M.step(gamma_kmax, model, counts, soft = soft)

    print(train_obj(model, counts))
  }
  # Return Model
  if(soft==TRUE){
      return(list("model"=model,"gamma"=gamma_kmax))}else{
      return(list("model"=model,"k_max"=gamma_kmax))
  }
}
```

Read and pre-process data

```
# reading the data
read.data <- function(file.name='bbc-text.csv', spr.ratio=0.90) {

  # Read the data
  text_df <- read.csv(file.name, colClasses = c('factor', 'character'))
  ## the terms before the first '\t' are the lables (the newsgroup names) and all the remaining text after '\t'
  ##are the actual documents
  docs <- text_df
  colnames(docs) <- c("doc_id", "text")
  docs$doc_id <- rownames(docs)
  # store the labels for evaluation
  labels <-  text_df$category

  library(tm)
  # create a corpus
  docs <- DataframeSource(docs)
  corp <- Corpus(docs)

  # Preprocessing:
  corp <- tm_map(corp, removeWords, stopwords("english")) # remove stop words
    #(the most common word in a language that can be find in any document)
  corp <- tm_map(corp, removePunctuation) # remove pnctuation
  corp <- tm_map(corp, stemDocument) # perform stemming (reducing inflected and derived words to their root form)
  corp <- tm_map(corp, removeNumbers) # remove all numbers
  corp <- tm_map(corp, stripWhitespace) # remove redundant spaces
  # Create a matrix which its rows are the documents and colomns are the words.
  dtm <- DocumentTermMatrix(corp)
  ## reduce the sparcity of out dtm
  dtm <- removeSparseTerms(dtm, spr.ratio)
  ## convert dtm to a matrix
  word.doc.mat <- t(as.matrix(dtm))

  # Return the result
  return (list("docs" = docs, "word.doc.mat"= word.doc.mat, "labels" = labels))
}
# Reading documents
## Note: sample.size=0 means all read all documents!
data <- read.data(file.name='bbc-text.csv', spr.ratio= .99)
# word-document frequency matrix
counts <- data$word.doc.mat
```

Run soft-EM algorithm and visualize

```
#  run soft-EM algorithm on the provided data
res_soft <- EM(counts, K=5, max.epoch=10, seed = 200)
```

```
## [1] 3047174
## [1] 2712009
## [1] 2668739
## [1] 2654916
## [1] 2650543
## [1] 2649232
## [1] 2648148
## [1] 2647383
## [1] 2646302
## [1] 2645564
## [1] 2644978

# visualization
## find the cluster with the maximum probability (since we have soft assignme
nt here)
label.hat.soft <- apply(res_soft$gamma, 1, which.max)

#  run hard-EM algorithm on the provided data
res_hard <- EM(counts, K=4, max.epoch=10, soft = FALSE, seed = 200)

## [1] 3049437
## [1] 2714663
## [1] 2667961
## [1] 2646901
## [1] 2641906
## [1] 2640816
## [1] 2640574
## [1] 2640488
## [1] 2640407
## [1] 2640395
## [1] 2640395

# visualization
## find the choosen cluster (since we have hard assignment here)
label.hat.hard <- res_hard$k_max
```

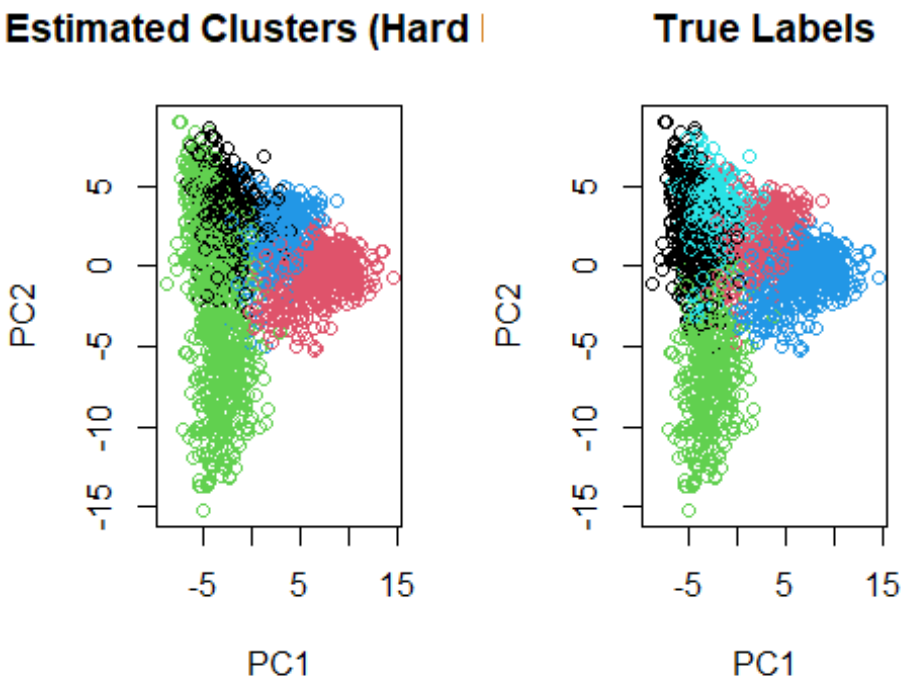
Visualize with PCA

```
counts<-scale(counts)
##--- Cluster Visualization ----------------------------------------------
cluster.viz <- function(doc.word.mat, color.vector, title=' '){
  p.comp <- prcomp(doc.word.mat, scale. = TRUE, center = TRUE)
  plot(p.comp$x, col=color.vector, pch=1,  main=title)
}
```

Visualize hard clusters

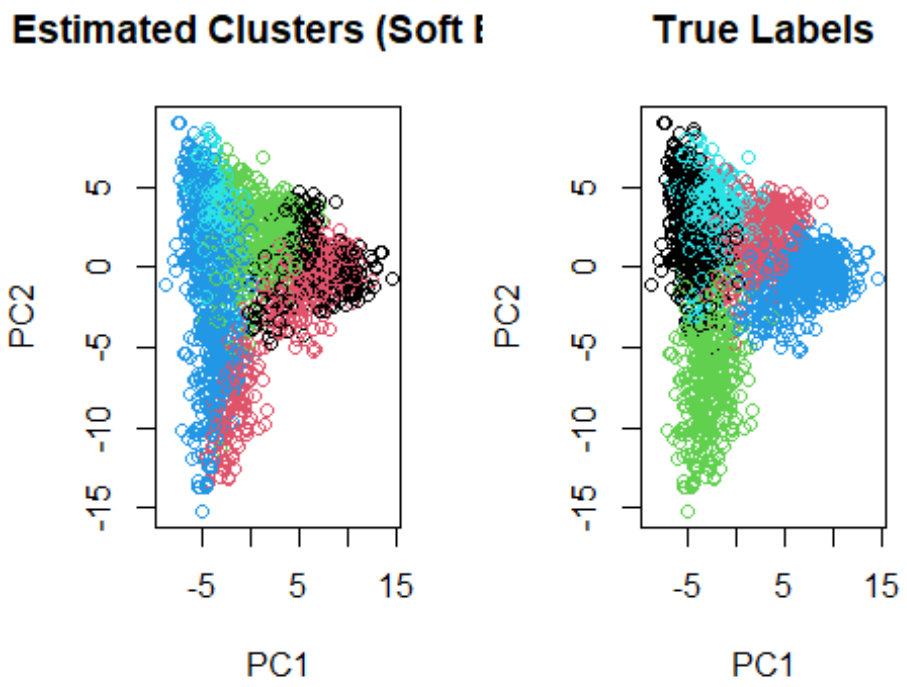
```
options(repr.plot.width=15, repr.plot.height=8)
par(mfrow=c(1,2))
## visualize the stimated clusters
counts <- scale(counts)
```

```
cluster.viz(t(counts), label.hat.hard, 'Estimated Clusters (Hard EM)')
cluster.viz(t(counts), data$labels, 'True Labels')
```



Visualize soft clusters

```
options(repr.plot.width=15, repr.plot.height=8)
par(mfrow=c(1,2))
## visualize the stimated clusters
counts <- scale(counts)
cluster.viz(t(counts), label.hat.soft, 'Estimated Clusters (Soft EM)')
cluster.viz(t(counts), data$labels, 'True Labels')
```



K-Means Approach

Source: Adapted from: https://rstudio-pubs-static.s3.amazonaws.com/445820_c6663e5a79874afdae826669a9499413.html

Load Data

```
truth.K <- 5

# Creating the empty dataset with the formatted columns

dataframe <- read.csv('bbc-text.csv')
```

Preprocessing data

```
corpus = tm::Corpus(tm::VectorSource(dataframe$text))

corpus.cleaned = corpus

corpus.cleaned <- tm::tm_map(corpus.cleaned, tm::removeWords, tm::stopwords('
english')) # Removing stop-words

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

corpus.cleaned <- tm::tm_map(corpus, tm::stemDocument, language = "english")
# Stemming the words
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::stemDocument, language = "engli
sh"):
## transformation drops documents
```

Text Representation

```
# Building the feature matrices
tdm <- tm::DocumentTermMatrix(corpus.cleaned)
tdm.tfidf <- tm::weightTfIdf(tdm)
# We remove A LOT of features. R is natively very weak with high dimensional
matrix
tdm.tfidf <- tm::removeSparseTerms(tdm.tfidf, 0.999)
# There is the memory-problem part
# - Native matrix isn't "sparse-compliant" in the memory
# - Sparse implementations aren't necessary compatible with clustering algori
thms
tfidf.matrix <- as.matrix(tdm.tfidf)
# Cosine distance matrix (useful for specific clustering algorithms)
dist.matrix = proxy::dist(tfidf.matrix, method = "cosine")
```

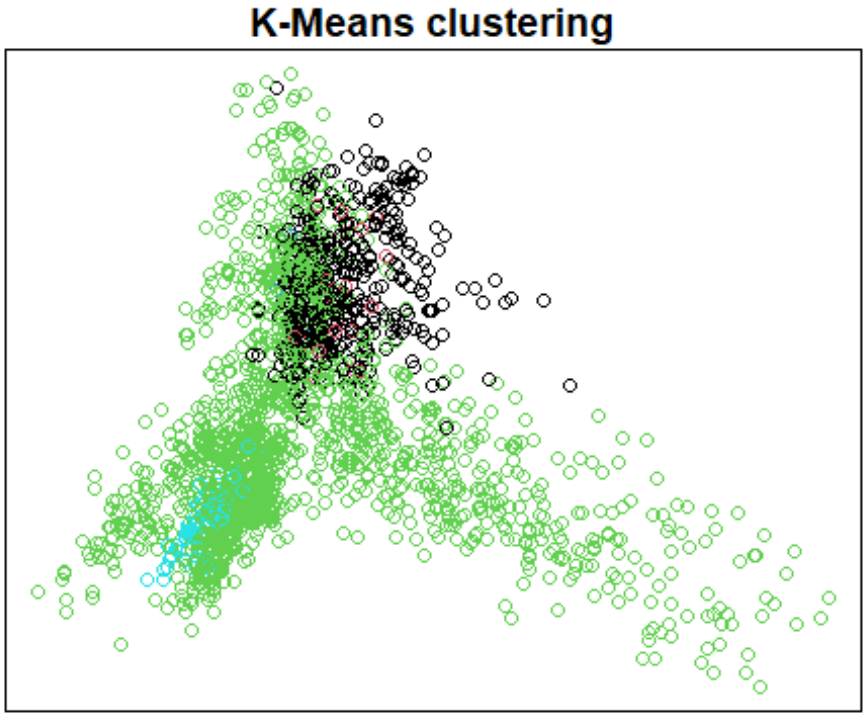
Perform K-means clustering

```
clustering.kmeans <- kmeans(tfidf.matrix, truth.K)
```

Plot results

```
master.cluster <- clustering.kmeans$cluster

points <- cmdscale(dist.matrix, k = 2) # Running the PCA
palette <- colorspace::diverge_hcl(truth.K) # Creating a color palette
previous.par <- par(mfrow=c(1,1), mar = rep(1.5, 4)) # partitionning the plot
space
plot(points,
     main = 'K-Means clustering',
     col = as.factor(master.cluster),
     mai = c(0, 0, 0, 0),
     mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n',
     xlab = '', ylab = '')
```


K-Means clustering