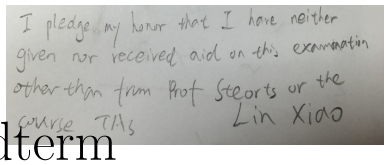


XIAO_Lin_Solutions_Midterm

Lin Xiao

September 30, 2015

1. Wine recommendation (20 pts total)
   An organization of wine enthusiasts gathers data on the wines its members have in their collections. They use the box-of-wines representation, so the data consists of a list of wine types for each member, like so:

   Aligheri,D: "L'Inferno 00", "Beata Beatrix 83", "Averno 01", "Ad Astra 02"
   Maro,PV: "L'Inferno 00", "Averno 01", "Ad Astra 02", "Maecenas Estates n.d."
   Montresor,E: "Valdemar Revenant 03", "L'Inferno 00"
   Fortunato,A: "Pym Amontillado 02", "Ligeia 99"
   Rumpole,J: "Chateau Plonk 97", "Chateau Plonk 98", "Chateau Plonk 99",
   "Chateau Plonk 00", "Chateau Plonk 01", "Chateau Plonk 02"

   There will, however, be many more than five members.
   We want to use this data to build a system to recommend new wines to people, based on the ones they already like.

(a) (1 points) Explain how to convert this data into a data frame with one row for each member, and a binary value for each feature. Be clear about what the features are, how they relate to wines, and how you would determine the number of features from the data.

[Ans]: We let names of rows to be members' names separately, column names to be the types of wines. For the same wine but produced in different years, we separate them as different columns. If the member has the a certain wine, the corresponding entry of the data frame would be 1, otherwise set it to be 0.

(b) (1 points) Explain how to find the k wine-drinkers most similar to a given member. Be explicit about how you would calculate similarity.

[Ans]: Suppose we are given a number i representing the member in ith row, then we can use the data frame above and l2 norm to calculate the distances from other rows to the given row. By setting the center as the given row we use scale function and then use sqrt(rowsums()) to calculate the l2 norm distance vector. After sorting the vector we get the first K wine-drinkers most similar to the given member. The smaller the distance, the more similar between the two members.

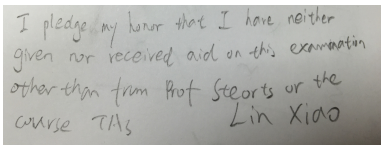
(c) (3 points) Raw term frequency in documents or classification suffers in the sense that all terms are considered equally important when it comes to assessing relevance on a `query.`[1] To attenuate the effect of terms that occur too often in the collection to be meaningful for relevance determination, we can scale down the weights of terms with high frequency. That is the inverse document frequency is the

$$\texttt{IDF(w)} = \texttt{log}\frac{\texttt{total number of documents}}{\texttt{document frequency of word w}}$$

What is the IDF of the wine "L'Inferno 00" among the 5 drinkers above? Do you think IDF would be useful in finding similar drinkers?

[Ans]: document frequency of word "L'Inferno 00" is 3 in this case, so we have

$$\texttt{IDF("L'Inferno 00")} = \texttt{log}\frac{5}{3}$$



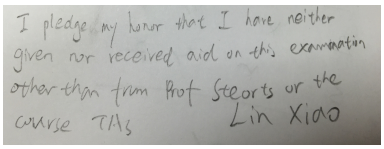
IDF would be useful in finding similar drinkers. For instance, if all members share the same type of wine, the IDF of that column would be 0 and by multiplying it we get a column with no difference between each two rows. The more members share a certain type of wine, the more help idf will provide to decrease the distance, and it assists in tossing out the common words. It might produce a more reasonable result.

(d) (5 points) Describe a procedure to recommend up to m wines to a customer based on the wines enjoyed by the k most-similar people in the data base. (Your procedure can be anything that we have talked about in class or used in homework or that is in the notes posted online for the class.) Explain how your procedure ensures that these wines are new to the customer, and how it selects m recommendations if it comes up with more than m candidates.

[Ans]: Fisrt we find out the k most similar wine-drinkers, and delete the columns where the entries of the given row is 1. Among the columns left, we can simply calculate the column sums and rank them in descending order. And we wrap up by selecting the first m wines as recommendations.

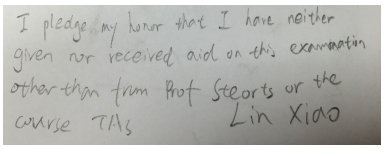
(e) (10 points) Code up your algorithm in the previous part and describe the results. Generate data to test your algorithm such that you have 50 wines choices and 15 customers. Test your procedure on you data and explain the results.

```r
# First we generate the data

dat <- matrix(rep(NA,15*50),ncol=50)
for(i in 1:15){
  for(j in 1:50){# Suppose the probability of a member having a specific wine is 0.7
    dat[i,j] <- rbinom(1,1,0.7)
  }
}

# Create a general function
# q is the one asking for recommendations
# k is the number of wine-drinkers that are most similar to q
# m is the number of recommended wines
# mat is the data frame, row for drinkers and column for wines
# Outputs would be two sets, with and without using IDF
# And it will show the number of wine to be recommended and their distance value
# Notice that there will be times
# When there are same values within distances or values for wines
# We just return the exact amount m
Wine <- function(q, k, m, mat){

  # First calculate the l2 norm distance, setting the center to be qth row
  dist.l2 <- sqrt(rowSums((scale(mat[-q,],center = mat[q,],scale = F))^2))

  # If exactly or more than k people share the same minimum distance
  # Simply return the first k
  if(k <= sum(dist.l2 == min(dist.l2))){
    print(cbind("index" = which(dist.l2 == min(dist.l2))[1:k],
                "values" = rep(min(dist.l2),k)))

    # Return them in order, choosing the first k
    k_members <- which(dist.l2 == min(dist.l2))[1:k]

    # Else sort the distance and return the first k
```




```
  }else{
    sorted <- sort.int(dist.l2,index.return = T)$x
    # Return the index
    sortedi <- sort.int(dist.l2,index.return = T)$ix
    print(cbind("index" = sortedi[1:k],"values" = sorted[1:k]))
    k_members <- sortedi[1:k]
  }

  # Create IDF
  IDF <- log(15/colSums(mat))

  # Adjusted by IDF
  mat_IDF <- t(t(mat)*IDF)



  # Using IDF, repeat the above steps

  dist.l2 <- sqrt(rowSums((scale(mat_IDF[-q,],center = mat_IDF[q,],scale = F))^2))

  if(k <= sum(dist.l2 == min(dist.l2))){
    print(cbind("IDF_index" = which(dist.l2 == min(dist.l2))[1:k],
                "IDF_values" = rep(min(dist.l2),k)))
    k_members <- which(dist.l2 == min(dist.l2))[1:k]
  }else{
    sorted <- sort.int(dist.l2,index.return = T)$x
    sortedi <- sort.int(dist.l2,index.return = T)$ix
    print(cbind("IDF_index" = sortedi[1:k],"IDF_values" = sorted[1:k]))
    k_members <- sortedi[1:k]
  }


  # Mark the columns where qth row equals to 0, which are the wines that q doesn't have

  flag <- which(mat[q,] == 0)

  # In order to let the wines q has rank low, we set the columns as all 0

  mat[,-flag] <- 0

  # Return the first m indices

  print(c("Wine's Num" = sort.int(colSums(mat, na.rm = T), decreasing = T,
                                  index.return = T)$ix[1:m]))

}


# Select the 2nd row as query, 3 most similar people to recommend 4 wines
# Using the dat we have simulate
Wine(2,3,4,dat)
```

```
##      index   values
```




```
## [1,]    11 4.123106
## [2,]     9 4.242641
## [3,]     2 4.472136
##      IDF_index IDF_values
## [1,]         7   1.745764
## [2,]         8   1.886757
## [3,]         1   1.889643
## Wine's Num1 Wine's Num2 Wine's Num3 Wine's Num4
##           5          15          19          23
```

From the results we can see that there is a difference between using IDF and not using IDF, which means tossing out common words will make a difference that cannot be ignored.

2. k-Means Clustering (20 pts total)

(a) (1 points) Explain what the k-means clustering algorithm is. Do not write any code or pseudo code, but give a precise verbal description which someone could turn into code (and do not copy the notes or words from the notes) Write the description in your own words.

[**Ans**]:
K-means is an typical algorithm based on distance. What we have to do is basically to minimize the enlarged criterion W.
1.We start with an initial guess by picking k points at random as the centers in a graph that has N points in it.
2.Secondly we calculate the distances between these N points and the K points we specified before, and classify the N points into K categories by choosing the closest point for every of the N points.
3.Then we recalculate the mean of each cluster, by simply taking the average of every category.
4.And now we repeat the 2nd and 3rd steps. Remember to update the enlarged criterion, and repeat the steps until any two consecutive Ws' difference is smaller than what we set. Then we return the clusters.

(b) (2 points) Can k-means ever give results which contain more or less than k clusters?

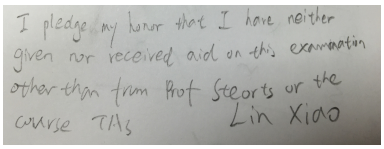
[**Ans**]:
No, because we have fixed the number of clusters. During iterations there won't be any change in the number of centers, and it's not possible to have 0 cluster involved.

(c) (2 points) Explain what the sum-of-squares is for k-means.

[**Ans**]:
Since the dissimilarities are squared L2 norm distances, sum of squares measures the total within-cluster variations — that is to sum up all k clusters of the squared distances between all the points in one cluster and the mean of this cluster. It is also the enlarged criterion W.

(d) (5 points) The next pages show the results of clustering the same data with k means, with k running from 2 to 6; also a plot of the sum-of-squares versus k. How many clusters would you guess this data has, and why? Does it matter whether the plot is an average over many runs of the algorithm?

[**Ans**]:
I would say 3(or 4 since they have almost the same value for slope), because according to the Elbow Method/Criterion, the point at which the line decreases the most abruptly should be the best number of cluster. And also, from the perspective of CH index, the number of clusters is involved in the calculation, if it gets bigger, it will result in smaller CH. Yes it matters, sometimes the algorithm converges



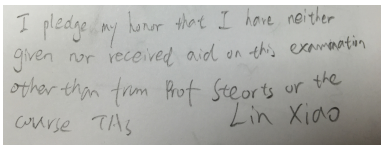
slowly for a certain number of clusters, and bad choice of initial values will have bad results, we have to iterate many times and average over them in order to attenuate the effects brought by bad choices.

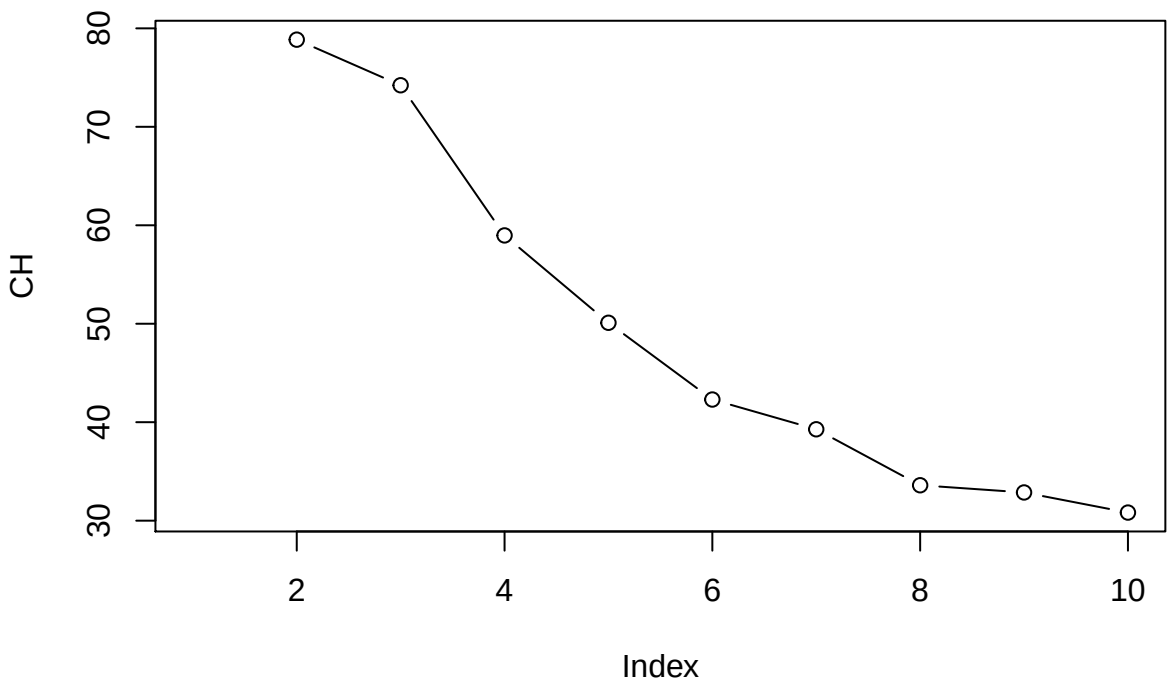
(e) (10 points) Fit k-means clustering to the wine data http://artax.karlin.mff.cuni.cz/r-help/library/HDclassif/html/wine.html using kmeans() in R. Choose the cluster pretending that you do not know that class exist. Evaluate different k using CH index. What do you find? Hint: Recall that you have learnt what a training dataset is. Think about how you might split the data into a training and a testing set for this part of the problem to evaluate k using the CH index.

```
require(HDclassif)
```

```
## Loading required package: HDclassif
## Loading required package: MASS
```

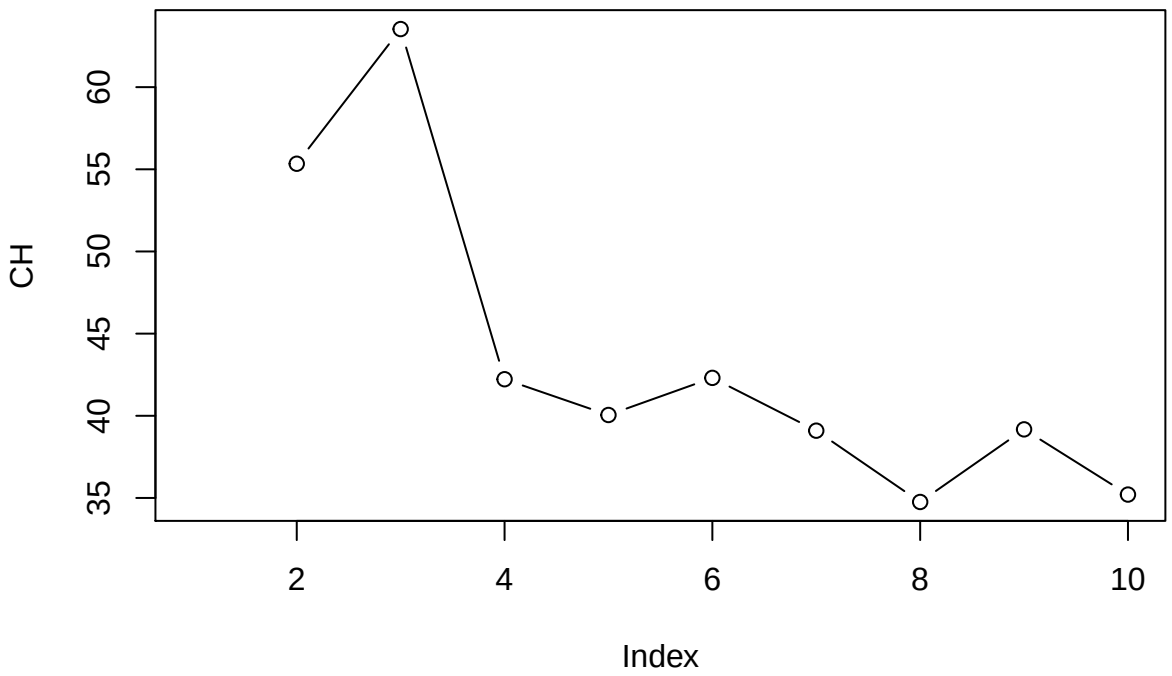
```
data(wine)

# Get rid of the class column and normalize the wine data
wine<-scale(wine[,-1])
n<-nrow(wine)

# We split the data with proportion of 3:1, 3 for training data
train_ind<-sample(seq_len(nrow(wine)),floor(0.75*nrow(wine)))

wine_training<-wine[train_ind,]
wine_test<-wine[-train_ind,]

## Inputs are the number of clusters, clustered object and the clean data frame
## Outputs are the plots, the index and the value of the maximum CH

Index_CH <- function(num_cluster,datfr){
  CH <- c()

  for(i in 2:num_cluster){
    k_mean<-kmeans(datfr, centers = i, iter.max = 1000, algorithm = "Lloyd")

    # The definition of CH index
    CH[i]<-(n-i)*k_mean$betweenss/((i-1)*sum(k_mean$withinss))
  }

  # Plot the CH index with respect to the number of clusters
  plot(CH,type = "b")

  # Then return the index and the value of the maximum CH
  print(c("Index" = 1+which.max(CH[2:num_cluster]),"Max" = max(CH[2:num_cluster])))
}

Index_CH(10, wine_training)
```





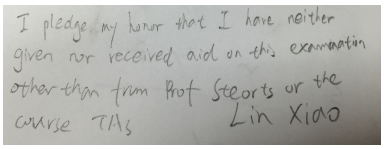

```
##    Index      Max
##  2.00000 78.84999
```

```
Index_CH(10, wine_test)
```




```
##    Index      Max
##  3.00000 63.53311
```

From the output and the plot of training data, because we are spliting data using sampling, CH reaches the max with either 2 or 3 clusters depending on the training set. It's also true for test data. Although we can



see from the class column of the wine data that we do have three classes and what we are trying to do is getting a close result, usually we can't match the output between training and test set. It seems that in this case, k means is not a stably good solution for clustering.

3. Page Rank (15 pts total)

(a) (5 points) The page rank vector can be calculated as the Eigen vector of the matrix $A = \frac{1-d}{n}E + dLM^{-1}$ corresponding to an Eigen value of 1. As noted in class, calculating the page rank using this method, can be computationally expensive. Provide pseudocode for calculating the the page rank using the stationary distribution of p by repeatedly multiplying by A by p^t, t = 0,...,T until the result stops changing (meaning that you have arrived at p). What are some reasonable initial values that you could use to start the iterative process with i.e $p^{(0)}$?

[Ans]:
input: constant d,
link matrix L,
column sums of link matrix M,
initial vector $p^{(0)}$,
a limit vector value δ
output: vector $p(t)$
n $\leftarrow$ column numbers of L
E $\leftarrow$ $n \times n$ matrix(all entries = 1)
A $\leftarrow$ $\frac{1-\mathrm{d}}{\mathrm{n}}\mathrm{E} + \mathrm{dLM}^{-1}$
t $\leftarrow$ 1
$p^{(t)} \leftarrow \mathrm{A}p^{(0)}$
while abs$(p^{(t)} - p^{(t-1)}) \geq \delta$ **do**
t $\leftarrow$ t+1
$p^{(t)} \leftarrow \mathrm{A}p^{(t-1)}$
return $p^{(t)}$
$p^{(0)}$ is subject to $\sum p_i^{(0)} = 1$, normallly we let all components to be equal:$(\frac{1}{n}, ..., \frac{1}{n})'$

(b) (5 points) Noting the formula for the matrix A in (a) what happens when $lim_{d\text{ß}1}$ A?

[Ans]:
When $lim_{d\text{ß}1}$, $A = LM^{-1}$, pagerank becomes brokenrank, stationary distribution may be nonunique when the Markov Chains are not strongly connected. That is to say, the BrokenRank vector p exists but is ambiguously defined.

(c) (5 points) Assume that you are asked to build your own search engine across a database of pages based on page rank and text-mining techniques you have seen in the course. Your task is to construct a metric that will take into account the page rank of each page in the database and similarity to some query that a user might input. Describe in detail how this metric could be constructed. Note: In this case there is no one unique correct answer.

[Ans]:

1. Find the documents that contain all words in the query that a user inputs.

2. Sort them by page rank and only keep the top K documents.

3. Subset the similarity matrix, saving the rows correspond to the top K documents.

4. Sort them by similarity to the query in ascending order (since it records the distance between the documents and the query) and return the top k (e.g., 50) documents as the final results. Normally k=K/100