Building Expectation Maximization Model in R

DS:630 – Project 1

Jinlian HowPanHieMeric & Pranay Katta

11/21/2017

11/21/2017
DS530: Project – Expectation Maximization Algorithm

Contents

Submitted by: Jinlian HowPanHieMeric & Pranay Katta

11/21/2017
DS530: Project – Expectation Maximization Algorithm

Executive Summary

This report outlines the methodology and result to building an Expectation Maximization Algorithm manually in R on a dataset. The goal was to complete an expectation maximization model from the beginning which consisted of initializing the problem through K-means and then running the expectation maximization model for each K value. The dataset consisted of 71 variables, inclusive of ID and 70 vectors, and 1077 observations.

Prior to running the model, the dataset was cleaned by removing null or NA values as well as outliers. Thus, the final dataset the model was run with consisted of 771 observations and 70 variables excluding the ID column, which is also removed. Prior to proceeding with initialization standardization of dataset is run to achieve a standardized distribution on an equal scale across all variables.

The Expectation Maximization (EM) Model was defined as a result of initializing values retrieved from K-mean algorithm for various K-value from 1 to 16, 18, 20, and 21 until convergence was reached. Convergence would be reached if it was lower than convergence threshold defined at 0.01. The model for each K-value would iterate up to 10 times or until convergence reached, whichever came first. Finally, all results would be compared for each K-value, wherein which K-value is the best, which would occur when the last new likelihood would be the least across all K-values with convergence reached prior to seeing a trend in the new likelihood results increasing again for additional K-values.

Time limitations in execution was a big obstacle, and therefore did not allow us to explore over K = 21 Nonetheless, a trend was apparent that as K increased final log likelihood decreased as well. In order to choose the best K value for the dataset we would choose the K value with the lowest log likelihood prior to getting higher ones with subsequent K-values. With available output, we choose K = 21 as the best K value, however, it is our estimate that a higher K-value had we been able to generate additional outputs.

Submitted by: Jinlian HowPanHieMeric & Pranay Katta

Methodology

Data exploration and cleaning

The dataset provided had 71 variables with 1077 observations. Prior to modeling the dataset was verified and cleaned so that no NA values and outliers were in dataset for the model to proceed. Following this we removed the ID column as it did not fit in the domain that would run through the Expectation Maximization (EM) Model. Once this was completed the final dataset that would go through the model consisted of 70 variables and 771 observations.

Data Preparation

Within the data exploration step, we looked to understand the summary of each variable to understand their range, quantiles and mean. This confirmed that the variables did not share a same distribution and mean and to better facilitate and run a successful k-means and then EM model we chose to standardize all our observations in the dataset. This would render all variable with a mean 0, as well as the dataset as a whole, and bring all observations onto one same standard level which would allow for optimal and preferred results in the algorithms.

Fixing the limits

The model created will run for various K-values and iterate a fixed number of times for each K-value, wherein K represented the number of clusters in the model. In building the model we fixed the starting K-value at 1, and the end K-value at 25, with a fixed iteration (T) limit of 10, wherein t = 1, .., 10. The latter would mean that if convergence was not reached the model would reiterate either until convergence was achieved or 10 iterations was reached, whichever came first. In order for the model to understand if convergence was reached we fixed our convergence limit to 0.01, which was calculated by:

|((newlikelihood - LogsumLoglikelihood_old)/LogsumLoglikelihood_old) * 100|

Thus, if the results from the convergence equation was greater than the fixed convergence limit (0.01) a new reiteration would begin, otherwise convergence would be considered as achieved.

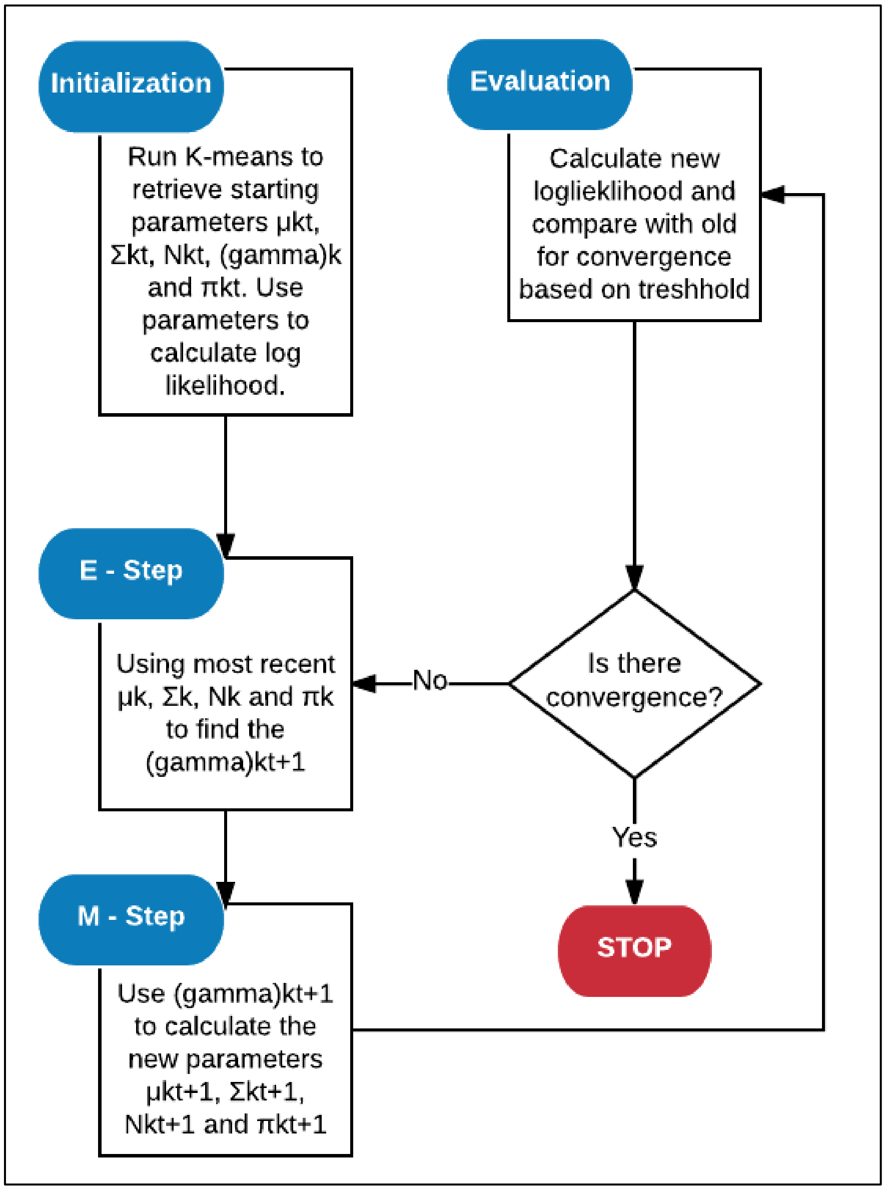
Defining the process

In order to facilitate running the model we predefined all our equations and parts of the model, as well as the results which would be necessary from each different parts of the initialization phase. This consists of finding the gamma $\bigcirc_k$, μ_k, covariance Σ_k, total data points in k N_k and mixing coefficients π_k for K = 1, … k. We are using the K-means algorithm as an initializing point to retrieve μ_k, Σ_k**,** N_k and π_k all above values as the initial values for all k's in K.

These latter three values would then be used to calculate the new $\bigcirc_k$, in the first Estep which would subsequently be used to calculate the new means μ_k, covariance Σ_k, total data points in k N_k and mixing coefficients π_k for K = 1, … k under the Maximization (M) step. This would conclude one iteration.

At end of each iteration (T) the log likelihood would be evaluated and compared to previous to check for convergence.

Following this the new μ_k, Σ_k, N_k and π_k would be used for the new iteration at Step E. The flow chart below demonstrates the process.


Initialization
Run K-means to retrieve starting parameters µkt, Σkt, Nkt, (gamma)k and πkt. Use parameters to calculate log likelihood.
Evaluation
Calculate new loglieklihood and compare with old for convergence based on treshhold
E - Step
Using most recent µk, Σk, Nk and πk to find the (gamma)kt+1
No
Is there convergence?
Yes
M - Step
Use (gamma)kt+1 to calculate the new parameters µkt+1, Σkt+1, Nkt+1 and πkt+1
STOP

Results

We ran our model against our dataset for K = 1, …., 16, 18, 20, and 21 for iterations T = 1, …, 10. As K increased, as estimated the length for iterations to complete and check for convergence would increase as well. We will be evaluating our results for K = 1, …., 16, 18, 20, and 21, while realizing that better results could've been achieved with additional K-values log likelihood outputs, as added comparison would've been possible however, as noted above the time limitation in execution prevented us from exploring further. Nonetheless, below are our log likelihood results, note when convergence was achieved the cell is highlighted in green:

	Log likelihood for each iteration (all negative values)										
K	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	76545.09	**76545.09**									
2	71068.49	72971.96	71152.71	71102.09	71076.58	71063.75	**71057.69**				
3	66873.44	71434.88	66910.07	66762.55	66736.97	66721.21	66697.69	66651.64	**66645.53**		
4	64719.44	70628.60	64756.98	64538.30	64474.30	**64468.47**					
5	62702.79	69704.21	62575.83	62337.19	62223.32	62187.50	62154.10	62120.54	62101.03	62086.60	62079.05
6	61054.21	70544.46	61343.42	60895.01	60841.26	60822.74	60814.24	60805.91	**60800.84**		
7	59181.66	69460.18	59565.76	58811.27	58654.95	58538.74	58464.99	58413.07	58365.75	58346.71	58338.08
8	57727.53	70360.42	58594.80	57629.38	57346.84	57256.75	57233.02	57223.44	**57217.90**		
9	57023.77	70438.03	57969.29	56759.29	56523.01	56461.61	56421.66	56389.69	56367.94	56364.10	56361.88
10	55655.84	69610.51	56267.71	55495.97	55374.83	55256.37	55108.76	55035.74	54995.99	54981.29	54963.74
11	54648.72	69419.78	55724.33	54421.74	54218.13	54091.70	53971.61	53933.51	53904.87	53849.54	53760.05
12	53015.15	68641.67	53651.70	52851.81	52668.00	52648.85	52637.56	52625.08	52617.95	52612.80	52609.89
13	52402.49	68946.59	53566.15	52203.15	52098.49	52039.57	52017.52	52005.40	51996.74	51988.14	51978.77
14	51151.93	69368.51	52514.60	50975.53	50812.85	50784.08	50740.98	50719.77	50708.65	50694.44	50691.32
15	51340.08	68843.83	52366.10	51077.63	50910.56	50779.26	50775.36	50773.87	50772.62	**50772.61**	
16	50767.13	69282.17	51606.18	50319.21	50227.47	50140.67	50097.10	50088.27	50067.46	50048.71	50041.85
18	48370.40	68802.56	49404.79	48116.95	48031.14	47964.12	47940.30	47919.72	47913.99	47912.05	47909.62
20	46461.61	68632.54	47512.45	46231.94	46134.78	46025.86	45995.62	45976.16	**45973.81**		
21	46267.51	68052.55	46967.76	45851.77	45725.78	45661.44	45639.30	45627.96	45613.66	45607.70	45603.23

Conclusion

We were able to accomplish output for up to K = 1, …., 16, 18, 20, and 21 only due to the time execution limitations. As we can see above though, as K increased the last log likelihood value was decreasing as well, either at convergence or at the 10th iteration. Based on results obtained K=21 is the best K-value for best estimate of parameters, as it had the lowest log likelihood at its 10th iteration with -45603.23. But seeing the trend we can assume further increases in K-value would've yielded a lower log likelihood for a better K-value overall for the dataset and estimates of parameters.

In the future, fixing the threshold limit higher would be recommended to reach convergence for more K values, as in this case multiple K values did not reach convergence by the 10th iteration. In addition, when re-running the model we would look to run it on multiple machines in parallel to reduce execution time and get all desired output. Another way to identify best K-value quickest would be to do limits of K-value ranges, such as K = 1, 5, 10, 20, 25, 30, etc. and identify when there is an increased change in log likelihood to then calculate for K-values in between the limit ranges.

Appendix 1: Code

```
library(MASS)
library(mvtnorm)

#if the percentage limit is less than 0.01 with any iteration EM exits with convergence limit
convergence_limit <- 0.01

max_iterations <- 10
#K value to start with
start_k_value <- 1
#(1 is default)

#No of K's to run for
end_k_value <- 25

#reading ydata.txt
data_file <- read.csv('/Users/JinLian/Google Drive/Saint Peters/Fall 17/630 - Machine Learning/Project/Project 1/cluster.csv', header= FALSE,numerals = "no.loss")

### STEP1 EXPLOR THE DATA #############

summary(data_file)

##NOTICING THE PRESENCE OF OUTLIERS AND NA'S########

x_col = data_file[,-1] #REMOVING THE FIRST COLUMN
head(x_col)
########### removing NA'S#########
x_col <- na.omit(x_col)
x_col <- x_col[complete.cases(x_col),]
dim(x_col)


#removing outliers
x_col <- subset(x_col, x_col$V6 >=2 & x_col$V6<=5.5)
dim(x_col)
x_col <- subset(x_col, x_col$V9 >=1.5 & x_col$V9<=5)
dim(x_col)
x_col <- subset(x_col, x_col$V11 >=0.8 & x_col$V11<=2)
dim(x_col)
x_col <- subset(x_col, x_col$V12<=1.5)
dim(x_col)
```

```
#View(x_col)


###################kmeans##############
########## function to scale our data. we defined it and put it into a variable called
standardize###################
standardize <- function(x) {
  return ((x - mean(x)) / sd(x))  }
###We only scale the domain and not the range.after removing the first variable id,now our
first variable in prc
x_col <- as.data.frame(lapply(x_col[1:70],standardize))
summary(x_col$V5) #checking
set.seed(1234)

# finding cluster covariance during initialization
cluster_covariance<-function(datak,meank){
  sumk<-0
  for(i in 1:(dim(datak)[1])){
    sumk <- sumk + (datak[i,]-meank) * (t(datak[i,])-meank)
  }
  return (sumk/dim(datak)[1])
}

#log_likelihood during initialization
log_likelhood <- function(datak,pik,meank,covariancek){
  sum <- 0
  for(i in 1:dim(datak)[1]){
    normk    <-    dmvnorm(as.matrix(datak[i,]),mean    =    as.matrix(meank[]),sigma    =
diag(covariancek))
    sum <- sum + normk * pik
  }
  return (sum)
}

#numerator of the gamma during initialization
gamma_numerator <- function(datak,pik,meank,covariancek){
  norm_numerator <- NULL
  for(i in 1:dim(datak)[1]){
    normk    <-    dmvnorm(as.matrix(datak[i,]),mean    =    as.matrix(meank[]),sigma    =
diag(covariancek))
    normk <- normk * pik
    norm_numerator <- rbind(norm_numerator,normk)
  }
  return (norm_numerator)
```

```
}

#denominator of gamma initialization
gamma_denominator <- function(datak,pik,meank,covariancek){
  norm_denominator <- NULL
  sumk <- 0
  for(i in 1:dim(datak)[1]){
    normk <- dmvnorm(as.matrix(datak[i,]),mean = as.matrix(meank[]),sigma =
diag(covariancek))
    normk <- normk * pik
    sumk <- sumk + normk
  }
  return (sumk)
}

init_likelihood <- function(){
  sumk <- sumlog <- 0
  for(j in 1:dim(newdf)[1]){
    data_ll <- newdf[j,2:71]
    sumk <- 0
    for(i in 1:k){
      mean_ll <- get(paste("mean",i,sep = ""))
      cov_ll <- get(paste("cov",i,sep = ""))
      pi_ll <- get(paste("pi",i,sep = ""))

      sumk <- sumk + pi_ll  * dmvnorm(as.matrix(data_ll),mean = as.matrix(mean_ll[]),sigma =
diag(cov_ll))
    }
      sumlog <- sumlog + log(sumk)
  }
    return(sumlog)
}


#function to find new_means for a given cluster after initialization
find_new_means <- function(data,gammak){
  sumk <- 0
  for(i in 1:dim(data)[1]){
    sumk <- sumk + data[i,] * gammak[i]
  }
  return (sumk)
}
#function to find new covariance for a given cluster after initialization
find_new_covariance <- function(gammak,datak,meank,Nk){
  sumk <- 0
  for(i in 1:dim(datak)[1]){
    sumk <- sumk + gammak[i] * (datak[i,]-meank) * (t(datak[i,]-meank))
  }
  return (sumk/Nk)
}

#function to find new Nk's for a given cluster after initialization
find_new_Nk <- function(gammak){
  sumk <- 0
  for(i in 1:k){
    sumk <- sumk + gammak[i]
  }
  return(sumk)
}

#function to find new pi's for a given cluster after initialization
new_pi <- function(datak,data){
  return(dim(datak)[1]/dim(data)[1])
}

find_gamma <- function(i){
  numerator <- get(paste("numerator",i,sep = ""))
  gammak <- NULL
  #browser()
  for(j in 1:dim(newdf)[1]){
    sum <- 0
    for(m in 1:k){
      temp <- get(paste("numerator",m,sep = ""))
      sum <- sum + temp[j]
    }
    gammak <- rbind(gammak,numerator[j]/sum)
  }
  return (gammak)
}


#Initialization starts here
sumLoglikelihood <- 0

#function to find new gamma for a given cluster after initialization
find_new_gamma <- function(){
  for(i in 1:k){
    numerator_name <- paste("numerator_new",i,sep = "")
    pik <- get(paste("new_pi",i,sep = ""))
    meank <- get(paste("new_mean",i,sep = ""))
    covariancek <- get(paste("new_covariance",i,sep = ""))
    clusterk <- get(paste("cluster",i,sep = ""))
    numerator <- gamma_numerator(newdf[,2:71],pik,meank,covariancek)
    assign(numerator_name,numerator,envir = .GlobalEnv)
  }
  for(i in 1:k){
    numerator <- get(paste("numerator_new",i,sep = ""))
    gamma_new_name <- paste("gamma_new",i,sep = "")
    gammak <- NULL
    for(j in 1:dim(newdf)[1]){
      sum <- 0
      for(m in 1:k){
        temp <- get(paste("numerator_new",m,sep = ""))
        sum <- sum + temp[j]
      }
      gammak <- rbind(gammak,numerator[j]/sum)
    }
    assign(gamma_new_name,gammak,envir = .GlobalEnv)
  }
}

#clears all the new values that have to computed
clearValues <- function(){
  for(i in 1:k){
    pik <- (paste("new_pi",i,sep = ""))
    meank <- (paste("new_mean",i,sep = ""))
    covariancek <- (paste("new_covariance",i,sep = ""))

    assign(pik,NULL,envir = .GlobalEnv)
    assign(meank,NULL,envir = .GlobalEnv)
    assign(covariancek,NULL,envir = .GlobalEnv)
  }
}

#function to find new_values for a given cluster after initialization
compute_new_values <- function(){
  for(i in 1:k){
  gamma_value <- get(paste("gamma_new",i,sep = ""))

  #getting Nk's
  N_names <- paste("N",i,sep = "")
  N_value <- sum(gamma_value)
  assign(N_names,N_value,envir = .GlobalEnv)

  #new means
  new_means_names <- paste("new_mean",i,sep = "")
  means_value <- find_new_means(newdf[,2:71],gamma_value)/get(N_names)
  assign(new_means_names,means_value,envir = .GlobalEnv)

  #new covariance
  new_covariance_names <- paste("new_covariance",i,sep = "")

assign(new_covariance_names,find_new_covariance(gamma_value,newdf[,2:71],means_value,
N_value),envir = .GlobalEnv)

  #new pis
  new_pi_names <- paste("new_pi",i,sep = "")
  pi_value <- N_value/dim(newdf)[1]
  assign(new_pi_names,pi_value,envir = .GlobalEnv)
  }
}

#function to find new likelihood with new means, covarinace, pi's
new_likelihood <- function(){
  sumk <- sumlog <- 0
  for(j in 1:dim(newdf)[1]){
    #take individual data points
    data_ll <- newdf[j,2:71]
    sumk <- 0
    for(i in 1:k){
      mean_ll <- get(paste("new_mean",i,sep = ""))
      cov_ll <- get(paste("new_covariance",i,sep = ""))
      pi_ll <- get(paste("new_pi",i,sep = ""))

      sumk <- sumk + pi_ll  * dmvnorm(as.matrix(data_ll),mean = as.matrix(mean_ll[]),sigma =
diag(cov_ll))
    }
    sumlog <- sumlog + log(sumk)
  }
  return(sumlog)
}

#Iterating through given K values

for(iteration in start_k_value:end_k_value){
k <- iteration
```

```
print(paste("running K-Means with K value -",k))

clust <- kmeans(x_col,centers = k,iter.max = 10)
#clust
#####################################################
newdf<-cbind(clust$cluster,x_col)

#Execution starts from here

for(i in 1:k){
  #create k clusters
  cluster_name <- paste("cluster",i,sep = "")
  a <- subset(newdf,clust$cluster==i)
  a <- a[,2:71]
  assign(cluster_name,a)
  #create k data_frames with each mean of Kth cluster in it
  mean_name <- paste("mean",i,sep = "")
  b <- apply(a, 2, mean)
  assign(mean_name,b)
  #create k data_frames with each pi value of Kth cluster in it
  pi_name <- paste("pi",i,sep = "")
  c <- dim(a)[1]/dim(newdf)[1]
  assign(pi_name,c)
  #create k data_frames with each co variance of Kth cluster in it
  cov_name <- paste("cov",i,sep = "")
  #a is data in kth cluster and b is mean of the kth cluster
  d <- cluster_covariance(a,b)
  assign(cov_name,d)
  numerator_name <- paste("numerator",i,sep = "")
  assign(numerator_name,gamma_numerator(newdf[,2:71],c,b,d))
}

#LogsumLogLikelihood_old contains likelihood of initilization
LogsumLoglikelihood_old <- init_likelihood()

for(i in 1:k){
  #getting gamma values
  #find gammas for the initialized values
  gamma_names <- paste("gamma",i,sep = "")
  nums <- paste("numerator",i,sep = "")
  denoms <- paste("denominator",i,sep = "")
  #gamma_value <- get(nums)/get(denoms)
  gamma_value <- find_gamma(i)
  assign(gamma_names,gamma_value)
  #getting Nk's
  N_names <- paste("N",i,sep = "")
  N_value <- sum(gamma_value)
  assign(N_names,N_value)
  cluster_value <- get(paste("cluster",i,sep = ""))

  #new means
  new_means_names <- paste("new_mean",i,sep = "")
  means_value <- find_new_means(cluster_value,gamma_value)/get(N_names)
  assign(new_means_names,means_value)

  #new covariance
  new_covariance_names <- paste("new_covariance",i,sep = "")

assign(new_covariance_names,find_new_covariance(gamma_value,newdf[,2:71],means_value,
N_value))

  #new pis
  new_pi_names <- paste("new_pi",i,sep = "")
  pi_value <- N_value/dim(newdf)[1]
  assign(new_pi_names,pi_value)
}


newlikelihood <- 0
count <- 0

#setting gamma_new to null
for(i in 1:k){
  gamma_new_name <- paste("gamma_new",i,sep = "")
  assign(gamma_new_name,NULL)
}

#EM starts here
repeat{
  #Evaluate log likelihood
  newlikelihood <- new_likelihood()

  print(paste("new log likelihood for iteration",count + 1,"is",newlikelihood))
  print(paste("old log likelihood is",LogsumLoglikelihood_old))

  #convergence <- abs(newlikelihood / LogsumLoglikelihood_old)
```

```
    convergence <- abs((newlikelihood - LogsumLoglikelihood_old )/LogsumLoglikelihood_old) *
100
  if(convergence <= convergence_limit || max_iterations < count) {
    if(max_iterations < count){
      print("max iterations reached")
    }else{
      print("convergence limit reached")
      print(paste("percentage decreased value",abs((newlikelihood - LogsumLoglikelihood_old
)/LogsumLoglikelihood_old) * 100))
    }
    break
  }

  print(paste("absabs((newlikelihood - LogsumLoglikelihood_old )/LogsumLoglikelihood_old) *
100 =",
              abs((newlikelihood - LogsumLoglikelihood_old )/LogsumLoglikelihood_old) * 100,
              "is greater than convergence limit",convergence_limit))
  count <- count + 1

  #Expectation Step
  #finds_new_gammas using new mean, covariance, pi
  find_new_gamma()

  clearValues()

  #Maximization Step
  # find new mean, covariance, pi and Nk
  compute_new_values()

  #copying the new likelihood to old likelihood
  LogsumLoglikelihood_old <- newlikelihood
}
}
```