4/3/2018

Manually Building Expectation Maximization Algorithm in R

Machine Learning

Abbas Navsariwala

Deenadayalamuthu

Rohan Thorat

***Output chart for convergence:***

***We ran the EM algorithm for convergence threshold equal to 0.000001, 0.0001 and 0.01 for k = 2 to 10. And we came to a conclusion that with a 4 digit convergence threshold i.e. (0.0001) there are 9 clusters(k=9) with maximum iterations 20.***

***The converged log-likelihood value for 9 clusters is 140623.3978***

***Log-likelihood for each iteration are as follows***

******

***Summary of the Project***

The main objective of this project was to build an Expectation Maximization Algorithm manually using R.

The process to build the Expectation Maximization algorithm involved the following steps:

* Running K-means Algorithm
* Obtaining Initial Parameter values from the K-means Algorithm for the Initialization step in EM algorithm
* Executing the EM Algorithm

***About the Dataset and Cleaning the Dataset***

We are using cluster.csv dataset for our project. The size of the dataset is 1077 observations and 71 dimensions. Before executing the K-means algorithms we first cleaned the dataset. The dataset contained some NA values which were removed from the dataset by completely eliminating the row. The resulting clean dataset’s size is 1047 observations and 71 dimensions. Later on dimension ID was removed which resulted in 70 dimensions.

***Running K-means Algorithm***

After cleaning the dataset, we first initialized the number of clusters we want to start with in the K-means Algorithm. To begin with we started with k=2(in our case we defined it as n\_clusters = 2). Then we ran the K-means with n\_clusters = 2. We obtained the Cluster mean values for each cluster in each dimension. This is the basis for the EM algorithm.

***Obtaining Initial Parameter values from the K-means Algorithm for the Initialization step in EM algorithm***

To begin with the EM algorithm we need some initial values for the parameters which were obtained from the K-means Algorithm.

* μk (in our case we defined it as mu\_initials)
* Covariance Σk  (in our case we defined it as sigma\_initials)
* πk  (in our case we defined it as Pi\_k\_initials)
* Nk  (which is nothing but the total data points for k cluster)

Note: k stands for n\_clusters

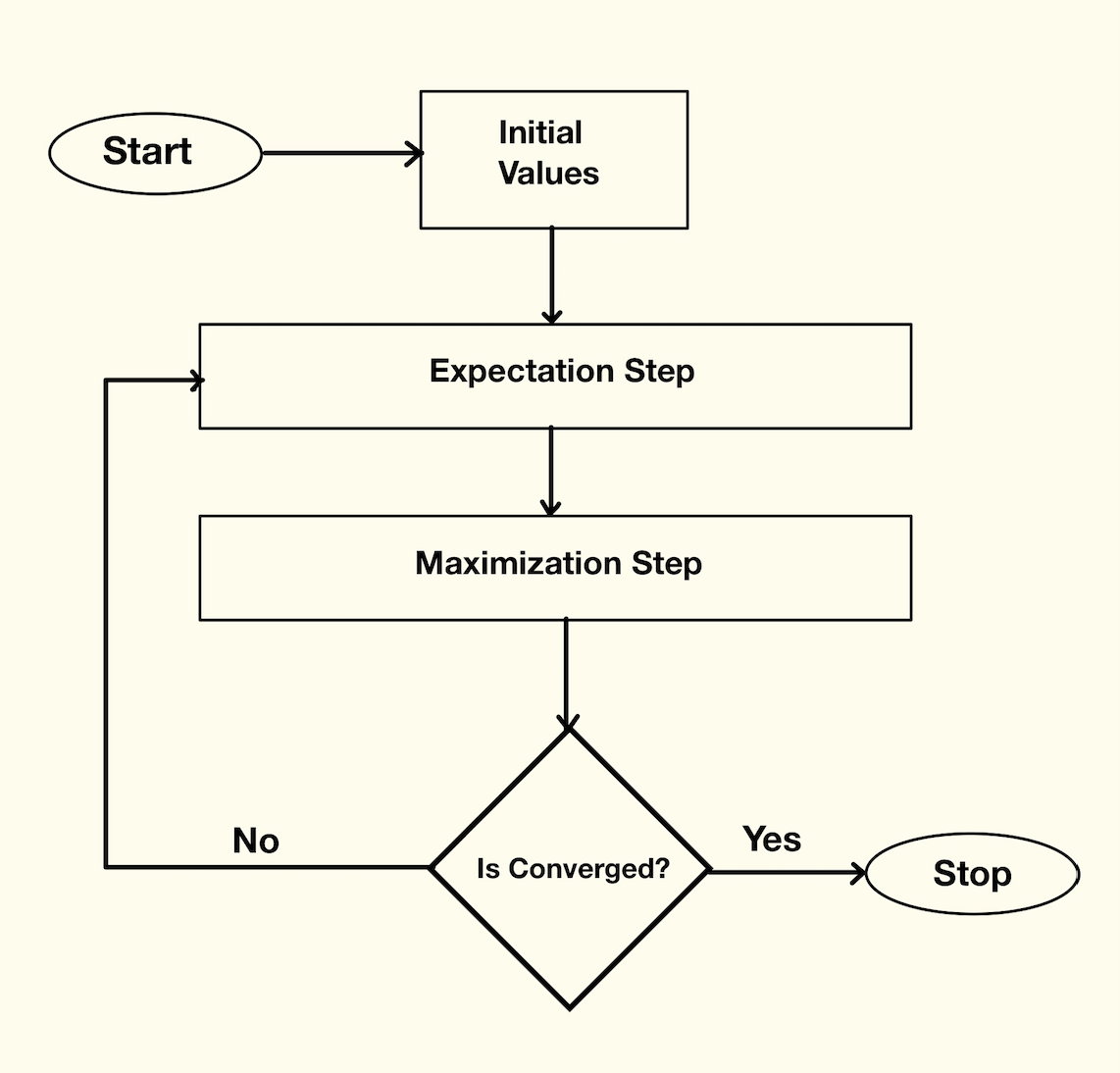
***Executing the EM Algorithm***

The Expectation-Maximization algorithm for Gaussian mixture is an iterative algorithm that starts from some initial parameter values and then proceeds to iteratively update either until convergence is detected or out of max iterations. Each iteration consists of an E-step and an M-step.

In our case convergence would be reached if log\_likelihood is lower than the threshold value that is, in our case, 0.0001. Keeping the time constraint in mind we are iterating the model 20 times.

EM Algorithm has 4 steps:

1. Determining the Initial GMM Parameters (μk, Covariance Σk , πk , Nk )
2. Expectation Step
3. Maximization Step
4. Check for Convergence



***Diving deep into the R Execution of the EM algorithm:***

**# loading required libraries**

**library(dplyr)**

**library(mvtnorm)**

**# Getting the dataset into R**

**proj\_data <- read.csv("~/cluster.csv", sep = ",", header = F)**

**# Exploring the dataset**

**head(proj\_data)**

**dim(proj\_data)**

**summary(proj\_data)**

**# Removing the first column from the dataset**

**proj\_data <- proj\_data[-1]**

**# Checking for missing/NA values**

**sum(is.na(proj\_data))**

**# Removing na values from the dataset**

**proj\_data <- na.omit(proj\_data)**

**dim(proj\_data)**

**# Setting the number of clusters for K-means**

**n\_clusters <- 9**

**# Getting the dimensions of the dataset**

**n\_dimensions <- ncol(proj\_data)**

**# K means clustering to get the initial parameters (mu\_initial, sigma\_initial)**

**cluster\_pred <- kmeans(x = proj\_data, centers = n\_clusters)**

**# Getting initial mu values**

**mu\_initials <- cluster\_pred[2]**

**mu\_initials <- mu\_initials$centers**

**# Adding the cluster prediction to the dataset**

**proj\_data\_df <- data.frame(cbind(proj\_data, cluster = cluster\_pred$cluster))**

**# Getting the initial covariance array**

**sigma\_initials <- array(rep(0),dim = c(n\_dimensions, n\_dimensions, n\_clusters))**

**for (i in 1:n\_clusters){**

**sigma\_initials[,,i] <- cov(proj\_data\_df[proj\_data\_df$cluster == i, -71])**

**}**

**# Getting the initial Pi\_k values**

**proj\_data\_df\_by\_cluster <- proj\_data\_df %>% group\_by(cluster)**

**Pi\_k\_initials <- proj\_data\_df\_by\_cluster %>% summarise(size = n()) %>% mutate(Pi\_k = size / sum(size))**

**Pi\_k\_initials <- Pi\_k\_initials$Pi\_k**

**# Empty data structures for comp & responsibility**

**comp <- matrix(rep(0), nrow = nrow(proj\_data), ncol = n\_clusters)**

**responsibility <- matrix(rep(0), nrow = nrow(proj\_data), ncol = n\_clusters)**

**#dim(comp)**

**# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Expectation Step \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #**

**# Defining expectation function**

**expectation <- function(x, mu\_initials, sigma\_initials, Pi\_k\_initials)**

**{**

**for (j in 1:n\_clusters){**

**comp[,j] <- dmvnorm(x, as.matrix(mu\_initials[j,]), as.matrix(sigma\_initials[,,j])) \* Pi\_k\_initials[j]**

**}**

**sum\_of\_comps <- rowSums(comp)**

**responsibility <- comp / sum\_of\_comps**

**ln\_sum\_of\_comps <- log(sum\_of\_comps, base = exp(1))**

**sum\_of\_ln\_sum\_of\_comps <- sum(ln\_sum\_of\_comps)**

**list("log\_likelihood" = sum\_of\_ln\_sum\_of\_comps,**

**"responsibility\_df" = responsibility)**

**}**

**# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Maximisation Step \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #**

**# Creating empty data structures to store the values**

**Mu\_new <- matrix(rep(0), nrow = n\_clusters, ncol = n\_dimensions)**

**N\_k\_new <- rep(0, n\_clusters)**

**Pi\_k\_new <- rep(0, n\_clusters)**

**sigma\_t <- matrix(0, nrow = n\_dimensions, ncol = n\_dimensions)**

**sigma\_t1 <- matrix(0, nrow = n\_dimensions, ncol = n\_dimensions)**

**Sigma\_new <- array(rep(0),dim = c(n\_dimensions, n\_dimensions, n\_clusters))**

**# Defining maximisation function**

**maximisation <- function(x, responsibility\_df){**

**for (k in 1:n\_clusters){**

**N\_k\_new[k] <- sum(responsibility\_df[,k])**

**Pi\_k\_new[k] <- N\_k\_new[k] / nrow(x)**

**}**

**for (k in 1:n\_clusters){**

**Mu\_t <- ((1/N\_k\_new[k]) \* (responsibility\_df[,k] \* x))**

**for (i in 1:n\_dimensions){**

**Mu\_new [k, i] <- sum(Mu\_t[,i])**

**}**

**}**

**for (k in 1:n\_clusters){**

**for (j in 1:nrow(x)){**

**difference\_transposed <- t(x[j,] - Mu\_new[k,])**

**difference <- x[j,] - Mu\_new[k,]**

**co\_efficients <- responsibility\_df[j, k] / N\_k\_new[k]**

**sigma\_t <- co\_efficients \* (difference\_transposed %\*% as.matrix(difference))**

**sigma\_t1 <- sigma\_t1 + sigma\_t**

**}**

**Sigma\_new[,,k] <- sigma\_t1**

**}**

**list("Mu\_new" = Mu\_new,**

**"Sig\_new" = Sigma\_new,**

**"Pi\_k\_New" = Pi\_k\_new)**

**}**

**# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Convergence Step \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #**

**for (i in 1:20){**

**if (i == 1){**

**# Initialization step**

**e\_step <- expectation(proj\_data, mu\_initials, sigma\_initials, Pi\_k\_initials)**

**m\_step <- maximisation(proj\_data, e\_step[["responsibility\_df"]])**

**current\_log\_likelihood <- e\_step[["log\_likelihood"]]**

**loglikelihood\_vector <- e\_step[["log\_likelihood"]]**

**}**

**else{**

**# Repeatation for E and M steps until convergence**

**e\_step <- expectation(proj\_data, m\_step[["Mu\_new"]], m\_step[["Sig\_new"]], m\_step[["Pi\_k\_New"]])**

**m\_step <- maximisation(proj\_data, e\_step[["responsibility\_df"]])**

**loglikelihood\_vector <- c(loglikelihood\_vector, e\_step[["log\_likelihood"]])**

**loglikelihood\_diff <- abs((current\_log\_likelihood - e\_step[["log\_likelihood"]]))**

**# Check for Convergence**

**if(loglikelihood\_diff < 0.0001){**

**break**

**}**

**else{**

**current\_log\_likelihood <- e\_step[["log\_likelihood"]]**

**}**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**