COMP/INDR 421/521 HW07: Expectation-Maximization Clustering

Asma Hakouz 0063315

December 22, 2017

The purpose of this assignment was to implement the expectation-maximization (EM) clustering algorithm in R.

The problem consisted of the implementation and visualization of an expectation-maximization clustering algorithm, which was done in these stages:

1- **Data pre-processing and visualization**
2- **Initialization using k-means clustering**
3- **EM iterations**
4- **Results and visualization**

Following, more details will be discussed about each part accompanied by snippets from my source code.

- **Data pre-processing and visualization**

```
library(MASS)
## GENERATING DATA
# choose an arbitrary value for the seed which will be used for random numbers generation
set.seed(521)
# mean parameters
cluster_means <- array(c(2.5, 2.5,
                         -2.5, 2.5,
                         -2.5, -2.5,
                         2.5, -2.5,
                         0, 0), c(1, 2, 5))
# Coveriance matrices
cluster_sigma <- array(c(+0.8, -0.6, -0.6, +0.8,
                               +0.8, +0.6, +0.6, +0.8,
                               +0.8, -0.6, -0.6, +0.8,
                               +0.8, +0.6, +0.6, +0.8,
                               +1.6, 0.0, 0.0, +1.6), c(2, 2, 5))
# sample sizes
cluster_sizes <- c(50, 50, 50, 50, 100)

# generate random samples from multivariate (in our case it's bivariate) normal distributions
points <- c()
for(i in 1:5){
  points <- rbind(points, MASS::mvrnorm(n = cluster_sizes[i], cluster_means[,, i], cluster_sigma[,, i]))
}

# plot the generated data points.
plot(points[,1], points[,2], type = "p", col = rgb(0.2,0.4,0.1,0.9), lwd = 0.5,
     xlab = "x1", ylab = "x2", ylim = c(-6, max(points[,2])),
     xlim = c(-6, max(points[,1])), pch = 19)
X <- points
```

This step includes generating 5 data sample from different clusters from multivariate gaussian distributions with means and covariances and number of samples as follows

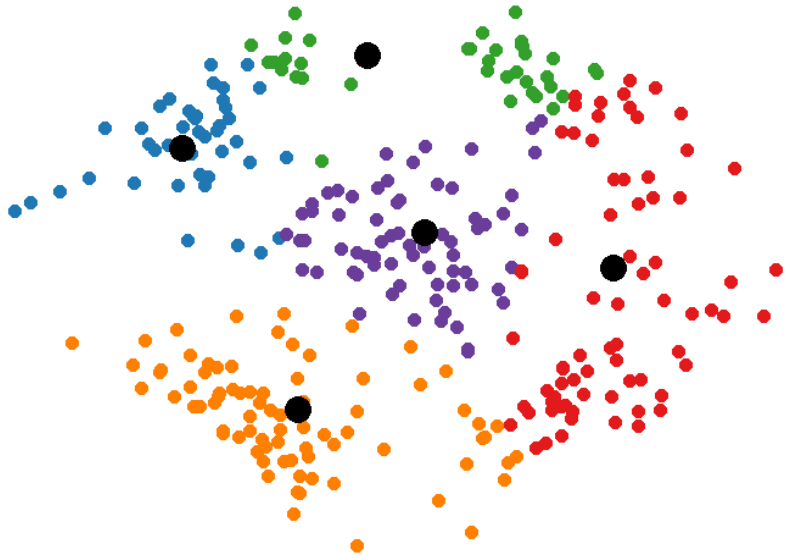
$$\mu_1 = \begin{bmatrix} +2.5 \\ +2.5 \end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix} +0.8 & -0.6 \\ -0.6 & +0.8 \end{bmatrix}, \quad N_1 = 50$$
$$\mu_2 = \begin{bmatrix} -2.5 \\ +2.5 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} +0.8 & +0.6 \\ +0.6 & +0.8 \end{bmatrix}, \quad N_2 = 50$$
$$\mu_3 = \begin{bmatrix} -2.5 \\ -2.5 \end{bmatrix}, \quad \Sigma_3 = \begin{bmatrix} +0.8 & -0.6 \\ -0.6 & +0.8 \end{bmatrix}, \quad N_3 = 50$$
$$\mu_4 = \begin{bmatrix} +2.5 \\ -2.5 \end{bmatrix}, \quad \Sigma_4 = \begin{bmatrix} +0.8 & +0.6 \\ +0.6 & +0.8 \end{bmatrix}, \quad N_4 = 50$$
$$\mu_5 = \begin{bmatrix} +0.0 \\ +0.0 \end{bmatrix}, \quad \Sigma_5 = \begin{bmatrix} +1.6 & +0.0 \\ +0.0 & +1.6 \end{bmatrix}, \quad N_5 = 100$$

- **Initialization using k-means clustering**
  - To initialize my EM algorithm from a good initial solution, I ran K-means clustering algorithm with K = 5 just for two iterations.

```
# K-means initialization
centroids <<- NULL
assignments <<- NULL
for(i in 1:2){
  if (is.null(centroids) == TRUE) {
    # for the first iteration; random initialization of centroids
    centroids <- X[sample(1:N, K),]
  } else {
    for (k in 1:K) {
      centroids[k,] <- colMeans(X[assignments == k,])
    }
  }
  #update_memberships,
  # check dist between all DPs and centroids
  D <- as.matrix(dist(rbind(centroids, X), method = "euclidean"))
  D <- D[1:nrow(centroids), (nrow(centroids) + 1):(nrow(centroids) + nrow(X))]
  #assignments <<- apply(D, MARGIN = 2, FUN = function(x) sort(x, index.return = TRUE)$ix[1])
  assignments <<- sapply(1:N, function(i) {which.min(D[,i])})

  colors <- c("#1f78b4", "#33a02c", "#e31a1c", "#ff7f00", "#6a3d9a", "#b15928", "#a6cee3",
              "#b2df8a", "#fb9a99", "#fdbf6f", "#cab2d6", "#ffff99")
  if (is.null(X) == FALSE && is.null(assignments) == TRUE) {
    par(mar = c(0, 0, 0, 0), oma = c(0, 0, 0, 0))
    plot(X[,1], X[,2], col = "lightgray", xlim = c(-6, 6), ylim = c(-6, 6), xlab = "", ylab = "",
         cex = 1.5, axes = FALSE, pch = 19)
  }
  if (is.null(X) == FALSE && is.null(assignments) == FALSE) {
    par(mar = c(0, 0, 0, 0), oma = c(0, 0, 0, 0))
    plot(X[,1], X[,2], col = colors[assignments], xlim = c(-6, 6), ylim = c(-6, 6), xlab = "",
         ylab = "", cex = 1.5, axes = FALSE, pch = 19)
  }
  if (is.null(centroids) == FALSE) {
    points(centroids[,1], centroids[,2], col = "black", pch = 19, cex = 3)
  }
}
```

the centroids after initialization where as follows



Then clusters covariances and means were estimated as the sample covariances and means in order to be given as an input to the E-step of the EM algorithm.

```
# estimate classes variances
sample_covariance <- array(sapply(X = 1:K, FUN = function(c) {
  matrix(c(mean((X[assignments == c, 1] - centroids[c, 1])^2),
           mean((X[assignments == c, 1] - centroids[c, 1])*(X[assignments == c, 2] - centroids[c, 2])),
           mean((X[assignments == c, 1] - centroids[c, 1])*(X[assignments == c, 2] - centroids[c, 2])),
           mean((X[assignments == c, 2] - centroids[c, 2])^2))}), dim=c(2,2,K))

# calculate prior probabilities
class_priors <- sapply(X = 1:K, FUN = function(c) {mean(assignments == c)})
```

- **EM iterations**

```
for(loop in 1:100){
  h_cluster <- sapply(1:N, function(t) {sapply(1:K, function(i) {
    (class_priors[i] * det(sample_covariance[,, i]) ** (-1/2.0) *
       exp(-1/2.0 *(t(as.matrix(X[t,] - centroids[i,])) %*% chol2inv(chol(sample_covariance[,, i])) %*%
                      as.matrix(X[t,] - centroids[i,])))/
      sum(sapply(1:K, function(i) {
        (class_priors[i] * det(sample_covariance[,, i]) ** (-1/2.0) *
           exp(-1/2.0 *(t(as.matrix(X[t,] - centroids[i,])) %*% chol2inv(chol(sample_covariance[,, i])) %*%
                          as.matrix(X[t,] - centroids[i,])))))
      }))
  })})

  class_priors <- sapply(1:K, function(j) {mean(h_cluster[j, ])})
  centroids <- t(sapply(1:k, function(j) {colSums(cbind(as.matrix(h_cluster[j,]), as.matrix(h_cluster[j,])) *
                                                    X)/ sum(h_cluster[j,])}))
  sample_covariance <- array(sapply(1:K, function(i){
    array(rowSums(sapply(1:N, function(t){
      h_cluster[i,t] * (X[t,] - centroids[i,]) %*% t(X[t,] - centroids[i,])
    })), c(2, 2)) / sum(h_cluster[i,])
  }), c(2, 2, K))

}
```

The E-step included estimating ‘soft’ indicating vectors (h) following the equation:

$$h_i^t = \frac{\pi_i |\mathbf{S}_i|^{-1/2} \exp[-(1/2)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T \mathbf{S}_i^{-1} (\boldsymbol{x}^t - \boldsymbol{m}_i)]}{\sum_j \pi_j |\mathbf{S}_j|^{-1/2} \exp[-(1/2)(\boldsymbol{x}^t - \boldsymbol{m}_j)^T \mathbf{S}_j^{-1} (\boldsymbol{x}^t - \boldsymbol{m}_j)]}$$

Then the M-step included estimating the means, covariances and priors for the clusters/components

$$\boldsymbol{m}_i^{l+1} = \frac{\sum_t h_i^t \boldsymbol{x}^t}{\sum_t h_i^t}$$

$$\mathbf{S}_i^{l+1} = \frac{\sum_t h_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i^{l+1})(\boldsymbol{x}^t - \boldsymbol{m}_i^{l+1})^T}{\sum_t h_i^t}$$

$$\pi_i = \frac{\sum_t h_i^t}{N}$$

And both E and M steps were repeated iteratively 100 times. Resulting in the following centroids:

```
> centroids
           [,1]       [,2]
[1,] -2.9945420  1.8493107
[2,] -0.3786763  3.4238423
[3,]  3.1044639 -0.1594590
[4,] -1.3630390 -2.5610210
[5,]  0.4381250  0.4366235
```

And the final clustering results, and centroid locations were as in the figure below. The dashed contours represent the original Gaussian densities I used to generate data points where their values are equal to 0.05. while the solid contours represent the Gaussian densities my EM algorithm finds where their values are equal to 0.05.