STAT542 Coding Assignment3

Chenzi Zhang, NetID chenziz2, UIN 654728837

3/27/2019

Contents

Reference to EM Algorithm on Wikipedia.org. Modified by Chenzi Zhang.

Define EM functions

Estep

```
Estep <- function(data, G, para) {
  # My Code
  pr = para$prob
  mu = para$mean
  Sinv = solve(para$Sigma)
  p = nrow(mu)
  tmp = NULL
  for(k in 1:G){
    tmp = cbind(tmp,
                apply(data, 1,
                      function(x) t(x - mu[, k]) %*% Sinv %*% (x - mu[, k])))
  }
  tmp = -tmp/2 + matrix(log(pr) + log((2*pi)^(-p/2))
                        + log(det(para$Sigma)^(-0.5)), nrow=n, ncol=G, byrow=TRUE)
  tmp = exp(tmp)
  tmp = tmp / apply(tmp, 1, sum)

  # Return the n-by -G probability matrix
  return(tmp)
}
```

Unknown parameters:

$$\theta = (\tau_{1:G}, \boldsymbol{\mu}_{1:G}, \boldsymbol{\Sigma}) \tag{1}$$

Multivariate normal distribution:

$$f(x) = (2\pi)^{-\frac{k}{2}} det(\Sigma)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)} \tag{2}$$

From the number of rows in mean (pxG) matrix, we get k=p, writing the pdf as following:

$$f(x_i;\mu_j^{(t)},\Sigma^{(t)}) = (2\pi)^{-\frac{p}{2}} det(\Sigma^{(t)})^{-\frac{1}{2}} e^{-\frac{1}{2}(x_i-\mu_j^{(t)})'\Sigma^{(t)^{-1}}(x_i-\mu_j^{(t)})} \tag{3}$$

The matrix of $T_{i,j}^{(t)}$ (nxG) is the output of my function:

$$T_{i,j}^{(t)} := P(Z_i = j|X_i = x_i;\theta^{(t)}) = \frac{\tau_j^{(t)} f(x_i;\mu_j^{(t)},\Sigma^{(t)})}{\sum_{j=1}^{G}\tau_j^{(t)} f(x_i;\mu_j^{(t)},\Sigma^{(t)})} \tag{4}$$

To get this matrix, we first calculate $log(\tau_j^{(t)} f(x_i;\mu_j^{(t)},\Sigma^{(t)}))$, which is:

$$log(\tau_j^{(t)} f(x_i;\mu_j^{(t)},\Sigma^{(t)})) = -\frac{1}{2}(x_i-\mu_j^{(t)})'\Sigma^{(t)}(x_i-\mu_j^{(t)}) + log(\tau_j^{(t)}) + log((2\pi)^{-\frac{k}{2}}) + log(det(\Sigma)^{-\frac{1}{2}}) \tag{5}$$

Then, use $e^{log(...)}$ to get $\tau_j^{(t)} f(x_i;\mu_j^{(t)},\Sigma^{(t)})$.

In addition, use `apply(tmp, 1, sum)` to conduct $\sum_{j=1}^{G}\tau_j^{(t)} f(x_i;\mu_j^{(t)},\Sigma^{(t)})$.

Therefore, I get the nxG matrix for $T_{i,j}^{(t)}$.

Mstep

```
Mstep <- function(data, G, para, post.prob) {
  # My Code
  new.pr <- apply(post.prob, 2, function(x) mean(x))

  new.nu <- NULL
  for(k in 1:G){
    tmp <- NULL
    for(j in 1:n){
      tmp <- rbind(tmp, data[j,]*post.prob[j,k])
    }
    tmp <- apply(tmp, 2, function(x) sum(x))
    new.nu <- cbind(new.nu, tmp/sum(post.prob[,k]))
  }

  new.Sigma <- NULL
  p <- nrow(new.nu)
  tmp <- matrix(0,p,p)
  for(j in 1:n){
    for(k in 1:G){
      tmp <- tmp + post.prob[j,k]*t(as.matrix((data[j,] - new.nu[,k]))) %*%
                    as.matrix((data[j,] - new.nu[,k]))
    }
  }
  new.Sigma <- tmp/sum(post.prob)
```

```
  # Return the updated parameters
  return(list(prob = new.pr, mean = new.nu, Sigma = new.Sigma))

}
```

Setting up function for Q:

$$
\begin{aligned}
Q(\theta|\theta^{(t)}) &= \mathbb{E}_{\mathbf{Z}|\mathbf{X},\theta^{(t)}}[logL(\theta;\mathbf{x},\mathbf{Z})] \\
&= \sum_{i=1}^{n} \mathbb{E}_{\mathbf{Z}_i|\mathbf{X},\theta^{(t)}}[logL(\theta;\mathbf{x}_i,\mathbf{z}_i)] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{G} P(Z_i = j|X_i = x_i;\theta^{(t)})logL(\theta;\mathbf{x}_i,\mathbf{z}_i) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{G} T_{i,j}^{(t)}[log\tau_j - \frac{1}{2}log|\Sigma| - \frac{1}{2}(\mathbf{x}_i-\mu_j)^T\Sigma^{-1}(\mathbf{x}_i-\mu_j) - \frac{p}{2}log(2\pi)]
\end{aligned}
$$

To maximize function Q, partial derivative function Q by parameters we want to update.
New τ:

$$
\begin{aligned}
\tau^{(t+1)} &= \arg\max_{\tau} Q(\theta|\theta^{(t)}) \\
&= \arg\max_{\tau} \left\{ \sum_{j=1}^{G}\sum_{i=1}^{n} T_{i,j}^{(t)} log\tau_j \right\}
\end{aligned}
$$

So the new τs are:

$$
\tau_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} T_{j,i}^{(t)}
$$

New μ, $\mathbf{\Sigma}$:

$$
\begin{aligned}
(\mu_j^{(t+1)}, \mathbf{\Sigma}^{(t)}) &= \arg\max_{\mu_j,\Sigma} Q(\theta|\theta^{(t)}) \\
&= \arg\max_{\mu_j,\Sigma} \sum_{i=1}^{n} T_{i,j}^{(t)} \left\{ -\frac{1}{2}log|\Sigma| - \frac{1}{2}(\mathbf{x}_i-\mu_j)^T\Sigma^{-1}(\mathbf{x}_i-\mu_j) - \frac{p}{2}log(2\pi) \right\}
\end{aligned}
$$

So the new μs are:

$$
\mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{i,j}^{(t)}\mathbf{x_i}}{\sum_{i=1}^{n} T_{i,j}^{(t)}}
$$

The new Σ is:

$$
\mathbf{\Sigma}^{(t+1)} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{G} T_{i,j}^{(t)}(\mathbf{x}_i-\mu_j)^T(\mathbf{x}_i-\mu_j)}{\sum_{i=1}^{n}\sum_{j=1}^{G} T_{i,j}^{(t)}}
$$

myEM

This is a iteration process.

```
myEM <- function(data ,T ,G ,para) {
  for(t in 1: T) {
    post.prob <- Estep(data , G, para)
```

```
    para <- Mstep(data , G, para, post.prob)
  }
  return (para)
}
```

Application

```
library(mclust)
```

```
## Package 'mclust' version 5.4.1
## Type 'citation("mclust")' for citing this R package in publications.
```

```
n <- nrow(faithful)
Z <- matrix (0, n, 2)
Z[sample(1:n, 120), 1] <- 1
Z[, 2] <- 1 - Z [, 1]
ini0 <- mstep(modelName ="EEE", faithful, Z)$parameters

# Output from my EM alg
para0 <- list(prob = ini0$pro, mean = ini0$mean,
                Sigma = ini0$variance$Sigma)
myEM(data = faithful, T = 10, G = 2, para = para0)
```

```
## $prob
## [1] 0.4412524 0.5587476
##
## $mean
##                [,1]      [,2]
## eruptions  3.447154  3.519868
## waiting   70.003255 71.602911
##
## $Sigma
##           eruptions   waiting
## eruptions  1.296635  13.89774
## waiting   13.897741 183.51292
```

```
# Output from mclust
Rout <- em(modelName = "EEE", data = faithful,
             control = emControl (eps = 0, tol = 0, itmax = 10),
             parameters = ini0)$parameters
list(Rout$pro, Rout$mean, Rout$variance$Sigma)
```

```
## [[1]]
## [1] 0.4412524 0.5587476
##
## [[2]]
##                [,1]      [,2]
## eruptions  3.447154  3.519868
## waiting   70.003255 71.602911
##
## [[3]]
##           eruptions   waiting
## eruptions  1.296635  13.89774
```

```
## waiting   13.897741 183.51292
```

The output from my EM algorithm is totally similar to the output from `mclust`.