MATH38161 Multivariate Statistics and Machine Learning Coursework

Name: Josh Mottley

May 11, 2020

Introduction

For this report I will be using bold letters to represents vectors (e.g: $\boldsymbol{A}$), bold and underlined letters to show matrices (e.g: $\underline{\boldsymbol{B}}$). A full example may look like:

$$\boldsymbol{y} = \boldsymbol{\beta}\underline{\boldsymbol{X}}$$

Futhermore, note that the "Appendix" chapter does not need to be viewed, and only contains the whole code, including testings, written, and should only be viewed if futher detail is needed.

1 One-dimensional Gaussian mixture model and the inference with the EM algorithm

The analytical update formula in the EM algorithm, for estimation of the parameters of a K-component mixture of one-dimensional normal distribution, can be written down as the E (expectation) and M (Maximisation) step equations.

1.1 Expectation step

The E "expectation" step of the EM algorithm is given by using Bayes theorem to predict the probabilities of allocations for all samples x_i, i.e:

$$\underline{\boldsymbol{z}}_{ik}^{(b+1)} = \frac{\boldsymbol{\pi}_k F(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}{\sum_{j=1}^{K}[\boldsymbol{\pi}_j F(\boldsymbol{x}_i|\boldsymbol{\theta}_j)]} \tag{1}$$

where $\boldsymbol{\pi}$ is the weights of each group of $\underline{\boldsymbol{z}}_{ik}^{(b)}$, F is the probability density function, K is the number of assumed groups, $\boldsymbol{\theta}^{(b)}$ is the current parameter vector for each group k, $\boldsymbol{x}$ is the observed data, and $\underline{\boldsymbol{z}}_{ik}^{(b)}$ is the current probability distribution of the latent variable k.

First of, to calculate the weights $\boldsymbol{\pi}_k$ of each group of $\underline{\boldsymbol{z}}_{ik}^{(b)}$, we use:

$$\boldsymbol{\pi}_k = \frac{\sum_{i=1}^{n} \underline{\boldsymbol{z}}_{ik}^{(b)}}{n}$$

So now calculate the E step for 1-dimensional Gaussian mixture models using (1):

$$\begin{aligned}\underline{\boldsymbol{z}}_{ik}^{(b+1)} &= \frac{\boldsymbol{\pi}_k F(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\sigma^2}_k)}{\sum_{j=1}^{K}[\boldsymbol{\pi}_j F(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\sigma^2}_j)]} \\ &= \frac{\boldsymbol{\pi}_k \left(\frac{1}{\sqrt{2\pi\boldsymbol{\sigma}_k^2}}\right) \exp\left(-\frac{(\boldsymbol{x}_i-\boldsymbol{\mu}_k)^2}{2\boldsymbol{\sigma}_k^2}\right)}{\sum_{j=1}^{K} \boldsymbol{\pi}_j \left(\frac{1}{\sqrt{2\pi\boldsymbol{\sigma}_j^2}}\right) \exp\left(-\frac{(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^2}{2\boldsymbol{\sigma}_j^2}\right)}\end{aligned} \tag{2}$$

where μ is the mean vector, and σ^2 is the variance vector.

1.2 Maximisation step

The maximisation step involves computing the expected complete data log-likelihood function using the new weights, and then maximising the function to update the mixture model parameter $\boldsymbol{\theta}$, which will be the maximum likelihood parameter. This can be also be described by the following equations

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(b)}) = \sum_{i=1}^{n}\sum_{k=1}^{K} \underline{\boldsymbol{z}}_{ik}^{(b+1)} \log(\boldsymbol{\pi}_k F_k(\boldsymbol{x}_i)) \tag{3}$$

$$\boldsymbol{\theta}^{(b+1)} \leftarrow \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(b)}) \tag{4}$$

First we want to find the complete log-likelihood part in (3), which is given by:

$$\begin{aligned} l(\boldsymbol{\mu}, \boldsymbol{\sigma}|\boldsymbol{x}_i) &= \sum_{i=1}^{n}\sum_{k=1}^{K} \log(\boldsymbol{\pi}_k F_k(\boldsymbol{x}_i)) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \left[\log \boldsymbol{\pi}_k - \frac{1}{2}\log 2\pi - \frac{1}{2}\log \boldsymbol{\sigma}_k^2 - \frac{1}{2\sigma^2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^2\right] \end{aligned} \tag{5}$$

Now we want to find Q, by substituting (5) into (3) to get:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(b)}) = \sum_{i=1}^{n}\sum_{k=1}^{K} \underline{\boldsymbol{z}}_{ik}^{(b+1)} \left[\log \boldsymbol{\pi}_k - \frac{1}{2}\log 2\pi - \frac{1}{2}\log \boldsymbol{\sigma}_k^2 - \frac{1}{2\sigma^2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^2\right] \tag{6}$$

Subbing (6) into (4) for 1-dimensional Gaussian, we get:

$$(\boldsymbol{\mu}_k^{(b+1)}, \boldsymbol{\sigma}_k^{(b+1)}) = \arg\max_{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k} \sum_{i=1}^{n} \underline{\boldsymbol{z}}_{ik}^{(b+1)} \left[\log \boldsymbol{\pi}_k - \frac{1}{2}\log 2\pi - \frac{1}{2}\log \boldsymbol{\sigma}_k^2 - \frac{1}{2\sigma^2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^2\right] \tag{7}$$

Take the derivatives $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ of (7), and set each to 0 and solve, leading to the predicition of the next parameters as:

$$\boldsymbol{\mu}_k^{(b+1)} = \frac{\sum_{i=1}^{n} \underline{\boldsymbol{z}}_{ik}^{(b)} \boldsymbol{x}_i}{\sum_{i=1}^{n} \underline{\boldsymbol{z}}_{ik}} \tag{8}$$

$$\boldsymbol{\sigma}_k^{(b+1)} = \frac{\sum_{i=1}^{n} \underline{\boldsymbol{z}}_{ik}^{(b)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(b)})^2}{\sum_{i=1}^{n} \underline{\boldsymbol{z}}_{ik}} \tag{9}$$

2 Implementation of 1-dimensional gaussian EM algorithm with R

For the full code regarding the implementation (i.e the function and testing), refer to Listing-3.

2.1 The EM function

The R function that performs EM estimation and returns the model parameters, as well as the probabilities of each sample that belongs to one of the K classes for 1-dimensional gaussion is given by:

Listing 1: EM function

```r
# EM algorithm function
# @param: x     : the data
# @param: K     : number of classes
# @param: it    : number of iterations
emAlgo<-function(x, K, it)
{
  # Initilise z_ik with a random soft allocation
  z_ik<-matrix(runif(K*length(x), 0, 1), ncol=K)
  # Normalise matrix (so sum of groups per sample = 1)
  correction<-z_ik[, 1]+ z_ik[, 2]
  z_ik[, 1]<-z_ik[, 1]/correction
  z_ik[, 2]<-z_ik[, 2]/correction
  # Initialise variables
  nk=0
  weights=0
  mean=0
  var=0
  # Iterate through the E and M steps
  for(iter in 1:it)
  {
    # Perform the M-step
    nk=colSums(z_ik)
    weights=nk/length(x)
    mean=apply(z_ik, 2, `%*%`, x)/nk
    var=colSums(z_ik*(outer(x, mean, "-")^2))/nk
    # Perform the E-step
    normal.d=t(sapply(x, dnorm, mean=mean, sd=sqrt(var)))
    z_ik=apply(normal.d*weights[col(normal.d)], 2, "/", rowSums(normal.d*weights[col(normal.d)]))
  }
  # Store in data frame
  output<-list(
    "parameters"=cbind(mean, sd=sqrt(var)),
    "z_ik"=z_ik
    )

  # Return Output
  return(output)
}
```

2.2 Analysis of data

Now if we test this by applying the function with 10000 iterations (to try to get convergence) to the following vector of $n = 10$ observations with $K = 3$:

$x = c(4.54, 1.57, 1.41, 1.77, 1.43, 0.07, 0.05, 4.19, 0.02, 1.32)$

Resulting in the following output:

Listing 2: Complete R code output

```
> print(out$parameters)
             mean         sd
[1,] 1.50000000 0.15697133
[2,] 0.03333333 0.03858612
[3,] 4.36500000 0.17500000
> print(out$z_ik)
               [,1]          [,2]          [,3]
 [1,] 1.651920e-81  0.000000e+00  1.000000e+00
 [2,] 1.000000e+00  0.000000e+00  1.609491e-56
 [3,] 1.000000e+00 1.126038e-276  5.147652e-63
 [4,] 1.000000e+00  0.000000e+00  2.815409e-48
 [5,] 1.000000e+00 8.586318e-285  3.301200e-62
 [6,] 6.127144e-19  1.000000e+00 3.667272e-132
 [7,] 1.330678e-19  1.000000e+00 1.540945e-133
 [8,] 7.799149e-64  0.000000e+00  1.000000e+00
 [9,] 4.636914e-21  1.000000e+00 1.753860e-137
[10,] 1.000000e+00 1.677989e-241  1.249908e-66
```

Now look at the density graph of the data given (x):

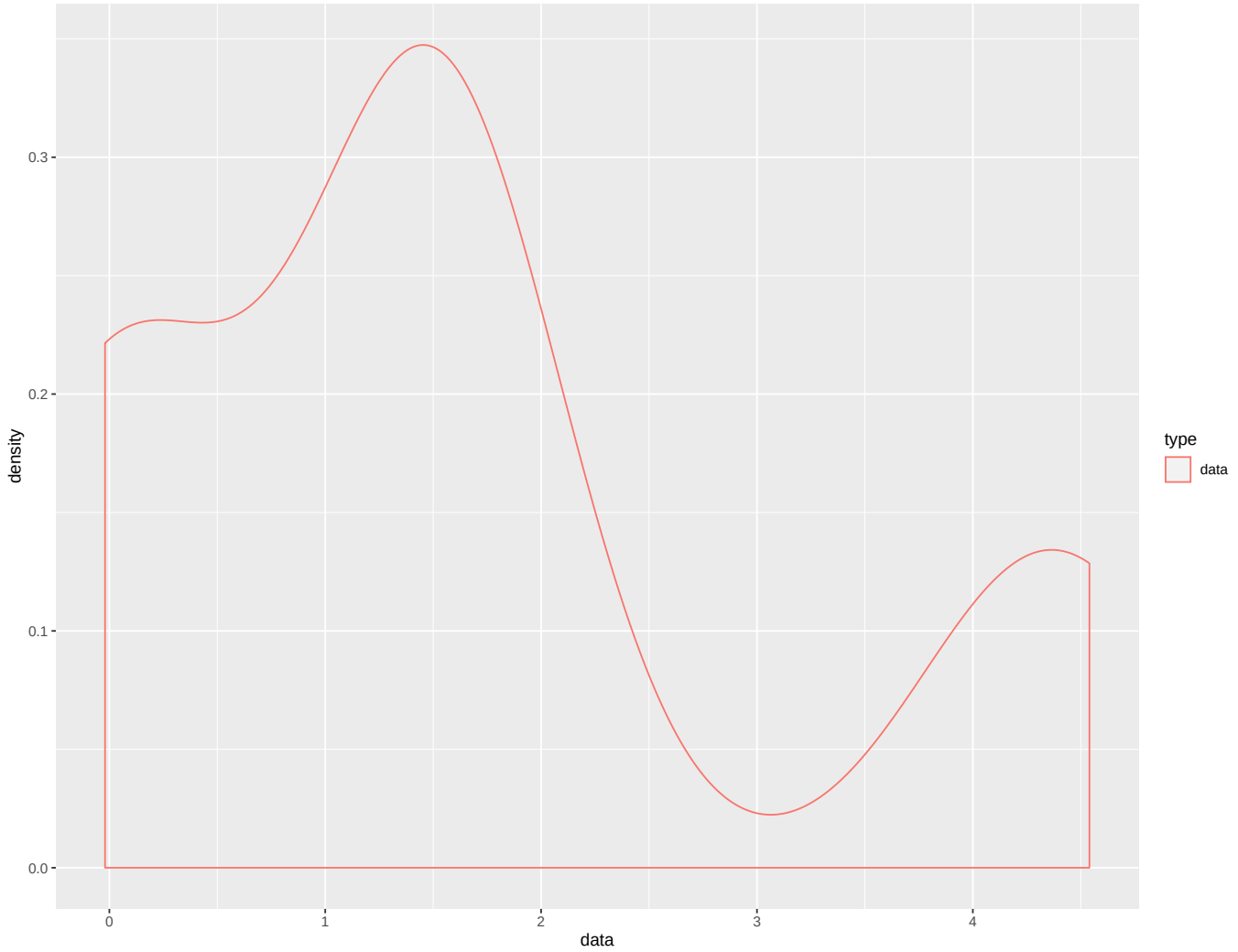

Figure 1: Plot of density of example data

Using the assumption that there are 3 groups, we can see 3 Gaussian distributions, with rough means between $(0-1)$, $(1-2)$ and $(4-5)$. This aligns with the results of the parameters returned from the function at 0.03333333, 4.36500000, 1.50000000. Futhermore, we can see from the graph that the right data have a wider spread than the middle, which can also been seen by the standard

deviation of the group with mean 4.36500000 being higher than the sd of the group with mean 1.50000000. Altogether, when comparing the graph of the data, against the results after testing the functions I can confidently say that the EM function is successful at determining the parameters of a mixed Gaussian model, with the downside being that it can take many iterations of the algorithm to successfully converge to the parameters.

3 Appendix

Listing 3: Complete R code (Function and testing)

```
# Delete previous variables
rm(list=ls())
dev.off()

# Includes
library("ggplot2")

# EM algorithm function
# @param: x     : the data
# @param: K     : number of classes
# @param: it    : number of iterations
emAlgo<-function(x, K, it)
{
  # Initilise z_ik with a random soft allocation
  z_ik<-matrix(runif(K*length(x), 0, 1), ncol=K)
  # Normalise matrix (so sum of groups per sample = 1)
  correction<-z_ik[, 1]+ z_ik[, 2]
  z_ik[, 1]<-z_ik[, 1]/correction
  z_ik[, 2]<-z_ik[, 2]/correction
  # Initialise variables
  nk=0
  weights=0
  mean=0
  var=0
  # Iterate through the E and M steps
  for(iter in 1:it)
  {
    # Perform the M-step
    nk=colSums(z_ik)
    weights=nk/length(x)
    mean=apply(z_ik, 2, `%*%`, x)/nk
    var=colSums(z_ik*(outer(x, mean, "-")^2))/nk
    # Perform the E-step
    normal.d=t(sapply(x, dnorm, mean=mean, sd=sqrt(var)))
    z_ik=apply(normal.d*weights[col(normal.d)], 2, "/", rowSums(normal.d*weights[col(normal.d)]))
  }
  # Store in data frame
  output<-list(
    "parameters"=cbind(mean, sd=sqrt(var)),
    "z_ik"=z_ik
    )

  # Return Output
  return(output)
}

#############################################################
# Test function
#############################################################

# Create data
x = c(4.54,1.57,1.41,1.77,1.43,0.07,0.05,4.19,-0.02,1.32)
y = factor(c(rep("data", length(x))))

# Plot data
df <- data.frame(
  type=y,
  data=x
)
ggplot(df, aes(x=data, color=type)) +
```

```
  geom_density()

# Test function
out<-emAlgo(x, 3, 10000)

# Output possible values
print(out$parameters)
print(out$z_ik)
```