MTH210: Lab 11

EM algorithm for Gaussian Mixture Model

1. Go through the `classDemonstration.R` file that runs the Gaussian Mixture Model EM algorithm for the `eruptions` on the Old Faithful dataset.

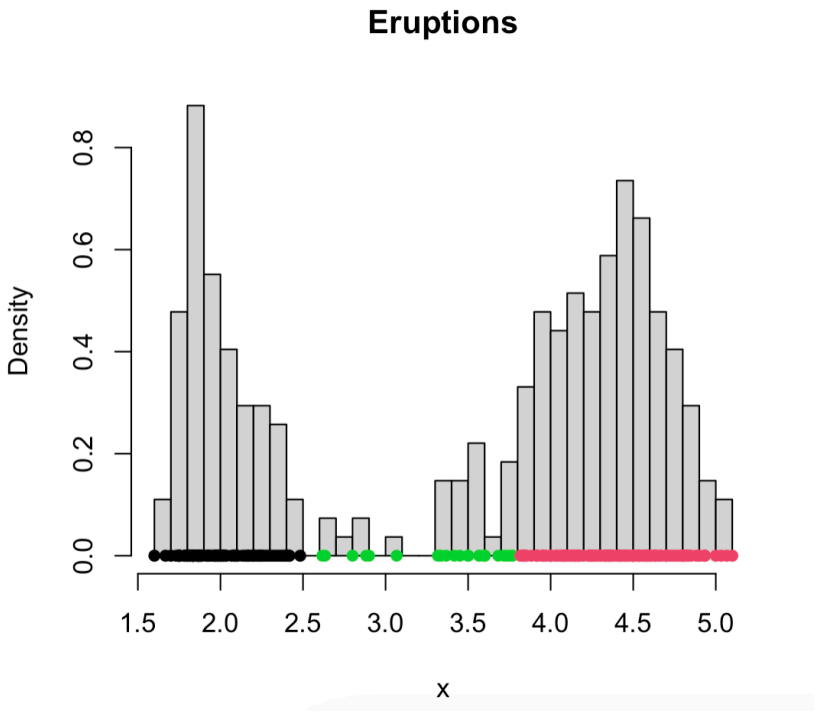
2. Now, change the code so that it is a general function that takes any 1-dimensional data vector `x` and number fo desired clusters `C` as an argument. Your function should look like:

```
# Function outputs estimated Zs (class allocations)
# and estimates of mu, sigma^2, and pis
GMMoneDim <- function(x, C = 2)
{
  ...
  rtn <- list(Zs, mu, sig2, pis)
}
```

   Once done, call `GMMoneDim` for `eruptions`:

```
erup2 <- GMMoneDim(faithpul$eruptions, C = 2)
erup3 <- GMMoneDim(faithpul$eruptions, C = 3)
```

   `erup2` should match your results from the previous exercise. Using `erup3` reproduce the following allocation plot:



3. In class, we discussed how to choose C using the Bayesian Information Criterion (BIC). Suppose we obtain a vector of estimates of $\theta = (\mu_1, \ldots, \mu_C, \sigma_1^2, \ldots, \sigma_C^2, \pi_1, \ldots, \pi_C)$. Denote these estimates by $\hat{\theta}$. Recall that the log likelihood for a given C is

$$\log f(x|\theta) = \log \left\{ \sum_{c=1}^{C} \pi_c f_c(x \mid \mu_c, \sigma_c^2) \right\}$$

First, write a function that calculates the log-likelihood for the observed data, given a particular value of θ:

```
log.like <- function(x, mu, sig2, pis)
{
  n <- length(x)
  track <- 0
  for(i in 1:n)
  {
    track <- track + ...
  }
  ...
  return(...)
}
```

Now, the BIC for an estimated $\hat{\theta}$ is

$$\text{BIC}(\hat{\theta}) = -\log f(\mathbf{x}|\hat{\theta}) + K\log(n)\,,$$

where K is the number of parameters. In this 1-dimensional example $K = 3C - 1$. The choice of C is the one that yields the lowest BIC. For the `eruptions` data, amongst $C = 2, 3, 4$, what value of C is best?

4. Repeat the above for the `waiting` component of `faithful` dataset.

5. Now, we will implement EM algorithm for a general p-dimensional problem. Below is the theory for this. Go through the theory and the details carefully.

   Fundamentally, the steps are exactly the same, however, there are some additional nuances in the M-step and the practical implementations are important to consider.

   Suppose for some C classes/clusters/components, we observe $X_1, \ldots, X_n$ from a mixture of multivariate normal distributions. The joint distribution of the observed $\mathbf{x}$ is:

$$f(\mathbf{x}|\theta) = \sum_{c=1}^{C} \pi_c f_1(\mathbf{x}|\mu_c, \Sigma_c)$$

   where $\mu_c \in \mathbb{R}^p$, $\Sigma_c \in \mathbb{R}^{p\times p}$ and

$$f_c(\mathbf{x} \mid \mu_c, \sigma_c^2) = \left(\frac{1}{2\pi}\right)^{p/2} \frac{1}{|\Sigma_c|^{1/2}} \exp\left\{-\frac{(\mathbf{x}-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}-\mu_c)}{2}\right\}.$$

   Similar to the one dimensional case, set up the EM algorithm for this p-dimensional case, and then implement this on the Old Faithful dataset.

   Introduce latent variable Z_i such that

$$Z_i = c \text{ if } X_i \text{ is in class } c\,.$$

   The E-step is to find q:

$$q(\theta|\theta_{(k)}) = \mathbb{E}\left[\log f(\mathbf{x}, \mathbf{z}|\theta) \mid X = \mathbf{x}, \theta = \theta_{(k)}\right]$$

   Let's solve this a bit more:

$$
\begin{aligned}
q(\theta|\theta_{(k)}) &= \mathbb{E}_{Z|x}\left[\log f(x,z|\theta) \mid X=x,\theta_{(k)}\right] \\
&= \mathbb{E}_{Z|x}\left[\sum_{i=1}^{n}\log f(x_i,z_i|\theta) \mid X=x,\theta_{(k)}\right] \\
&= \sum_{i=1}^{n}\mathbb{E}_{Z_i|x_i}\left[\log f(x_i,z_i|\theta) \mid X=x_i,\theta_{(k)}\right] \\
&= \sum_{i=1}^{n}\sum_{c=1}^{C}\log\left\{f_c(x_i|\mu_c,\Sigma_c)\pi_c\right\}\ \frac{f_c(x_i|\mu_{c,k},\Sigma_{c,k})\pi_{c,k}}{\sum_{j=1,2}f_j(x_i|\mu_{j,k},\Sigma_{j,k})\pi_{j,k}}\,.
\end{aligned}
$$

For the E-step above, we need to find

$$
\Pr(Z=c|X=x_i,\theta=\theta_{(k)}) = \frac{f(x_i|Z=c,\theta=\theta_{(k)})\Pr(Z=c)}{f(x_i)} = \frac{f_c(x_i|\mu_{c,k},\Sigma_{c,k})\pi_{c,k}}{\sum_{j=1}^{C}f_j(x_i|\mu_{j,k},\Sigma_{j,k})\pi_{j,k}} := \gamma_{i,c,k}\,.
$$

And then in the M-step, we do the maximization step:

$$
q(\theta|\theta_{(k)}) = \text{const} - \frac{1}{2}\sum_{i=1}^{n}\sum_{c=1}^{C}\log|\Sigma_c|\gamma_{i,c,k} - \sum_{i=1}^{n}\sum_{c=1}^{C}\frac{(x_i-\mu_c)\Sigma_c^{-1}(x_i-\mu_c)^T}{2}\gamma_{i,c,k} + \sum_{i=1}^{n}\sum_{c=1}^{C}\log\pi_c\gamma_{i,c,k}\,.
$$

Taking derivative with respect to μ_c

$$
\begin{aligned}
\frac{\partial q}{\partial\mu_c} &= -\sum_{i=1}^{n}\Sigma_c^{-1}(x_i-\mu_c)\gamma_{i,c,k} \overset{set}{=} 0 \\
\Rightarrow \mu_{c,(k+1)} &= \frac{\sum_{i=1}^{n}\gamma_{i,c,k}\ x_i}{\sum_{i=1}^{n}\gamma_{i,c,k}}
\end{aligned}
$$

Taking derivatives with respect to a matrix, Σ_c is tricky. But the following properties will be useful to remember for compatible matrices A, B, C

- $\text{tr}(ACB) = \text{tr}(CAB) = \text{tr}(BC)$
- for scalar x, $x^TAx = \text{tr}(x^TAx) = \text{tr}(x^TxA)$
- $\dfrac{\partial}{\partial A}\text{tr}(AB) = B^T$
- $\dfrac{\partial}{\partial A}\log(|A|) = A^{-T}$

Using these properies and taking derivative w.r.t Σ_c^{-1} (for convenience), we get

$$
\begin{aligned}
\frac{\partial q}{\partial \Sigma_c^{-1}} &= -\frac{n}{2}\Sigma_c \gamma_{i,,c,k} - \frac{\partial}{\partial \Sigma_c^{-1}}\left[\sum_{i=1}^{n} \frac{\text{tr}((x_i-\mu_c)^T(x_i-\mu_c)\Sigma_c^{-1})}{2}\gamma_{i,c,k}\right] \\
&= -\frac{n}{2}\Sigma_c \gamma_{i,,c,k} - \left[\sum_{i=1}^{n} \frac{(x_i-\mu_c)^T(x_i-\mu_c)\gamma_{i,c,k}}{2}\right] \overset{set}{=} 0 \\
\Rightarrow \Sigma_{c,(k+1)} &= \frac{\sum_{i=1}^{n} \gamma_{i,c,k}(x_i-\mu_{c,(k+1)})(x_i-\mu_{c,(k+1)})^T}{\sum_{i=1}^{n}\gamma_{i,c,k}}\,.
\end{aligned}
$$

The optimization for π_c is the same as the univariate case. Hence, we get the following updates:

$$
\begin{aligned}
\mu_{c,(k+1)} &= \frac{\sum_{i=1}^{n} \gamma_{i,c,k}\ x_i}{\sum_{i=1}^{n}\gamma_{i,c,k}} \\
\Sigma_{c,(k+1)} &= \frac{\sum_{i=1}^{n} \gamma_{i,c,k}(x_i-\mu_{c,(k+1)})(x_i-\mu_{c,(k+1)})^T}{\sum_{i=1}^{n}\gamma_{i,c,k}} \\
\pi_{c,(k+1)} &= \frac{1}{n}\sum_{i=1}^{n}\gamma_{i,c,k}
\end{aligned}
$$

Here are some practical things to keep in mind:

- **Multivariate normal density**: Given a starting value, the algorithm updates the components using the above update equations. Notice that we need to evaluate the density of a multivariate normal. You can use `mvtnorm` package and `dmvnorm` function for that.
- **Local solutions**: Further note that the target log-likelihood is not concave. Thus, there will be multiple local maximas. The EM algorithm is an MM algorithm, so it does converge to a local maxima, but it need not converge to a global maxima. In order to try to ensure that we reach a reasonable maxima, it will be useful to start from multiple starting points and see where you converge. Then note down the log-likelihood and compare the log-likehood with other runs from different starting points. The optima with the largest log-likelihood should be the chosen maxima.
- **Starting values**: In addition to testing this on multiple starting values, one must also make sure the starting values are realistic. That is, if most of the data is centered near (50,100), then it is unreasonable to start at (0,0), and infact such starting values will lead to numerical inaccuracies. Additionally, if your starting values for the μ_c are the same for all c, then all normal distribution components are centered at the same spot and this leads to only one cluster, even if you have specified $C > 1$.
- **Lack of positive definiteness**: In the update for Σ_c, note that we are calculating a covariance matrix. Particularly when C is chosen to be larger than the actual number of clusters, an iteration value of $\Sigma_{c,(k+1)}$ may yield singular matrices. To correct for this, a common solution is to set

$$
\Sigma_{c,(k+1)} = \frac{\sum_{i=1}^{n} \gamma_{i,c,k}(x_i-\mu_{c,(k+1)})(x_i-\mu_{c,(k+1)})^T}{\sum_{i=1}^{n}\gamma_{i,c,k}} + \text{diag}(\epsilon, p)\,.
$$

That is, we add a diagonal matrix with small entries ϵ in order to make sure it's positive definite.

- In order to declare convergence, we will monitor the change in the log-likelihood values.

6. Having understood the challenged and the theoretical framework from before, your task is to write a function `GMMforD` that takes a matrix input and the number of clusters C as arguments:

```
# Now I recommend the following
# mu is a list
# Sigma is a list
GMMforD <- function(x, C = 2)
{
  p <- dim(x)[2]
  ...
  ...
  return(Zs, mu, Sigma, pis)
}
```

7. Run `GMMforD` for the bivariate Faithful dataset for $C = 2$. Plot the data, and color-code the clusters

8. Use BIC to obtain the best value of C among $2, 3, 4$ for this dataset.