Activity: Fitting Gaussian mixture models

Group members:

The goal of this activity is to derive the iterative procedure used to fit Gaussian mixture models. For the purposes of this activity, we will restrict ourselves to the univariate case.

Gaussian mixture model

Suppose we observe data $X_1, ..., X_n$, and we assume that each observation i comes from one of k groups. Let $Z_i \in \{1, ..., k\}$ denote the group assignment. The one-dimensional Gaussian mixture model assumes that

$$P(Z_i = j) = \lambda_j \qquad j \in \{1, ..., k\}$$

and

$$X_i | (Z_i = j) \sim N(\mu_j, \sigma_j^2)$$

That is, the probability of belonging to group j is λ_j, and the distribution of each group is Gaussian with its own mean μ_j and variance σ_j^2.

Parameter estimation

How do we estimate the parameters in a Gaussian mixture model? As we have seen throughout the course, fitting a statistical model involves optimizing some function of the data (e.g., minimizing the residual sum of squares for a linear regression model).

In the case of the Gaussian mixture model, the quantity we **want** to optimize is called the **log-likelihood**, and it is given by

$$\ell(\lambda, \mu, \sigma) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} \lambda_j f(X_i | Z_i = j) \right)$$

where

$$f(X_i | Z_i = j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp \left\{ -\frac{1}{2\sigma_j^2} (X_i - \mu_j)^2 \right\}$$

is the $N(\mu_j, \sigma_j^2)$ density.

Example

The following code simulates data from a 2-component Gaussian mixture model, and fits the model with the `mixtools` package:

```
library(mixtools)

set.seed(8675309)

n <- 300
group_lambda <- c(0.75, 0.25)
group_mu <- c(0, 4)
group_sd <- c(1, 1)

z <- sample(c(1, 2), n, replace=T, prob=group_lambda)
x <- rnorm(n, mean=group_mu[z], sd=group_sd[z])

em_res <- normalmixEM(x, k=2)
```

The log-likelihood $\ell(\widehat{\lambda}, \widehat{\mu}, \widehat{\sigma})$ of the estimated parameters is provided by

```
em_res$loglik
```

To verify that this log-likelihood is calculated using the formula above, here is a function which calculates the log-likelihood for a univariate Gaussian mixture:

```
compute_loglik <- function(x, lambda, mu, sigma){
  indiv_logliks <- rep(0, length(x))
  for(i in 1:length(lambda)){
    indiv_logliks <- indiv_logliks + dnorm(x, mean=mu[i], sd=sigma[i]) *lambda[i]
  }

  sum(log(indiv_logliks))
}
```

1. **Question:** Use this `compute_loglik` function to compute $\ell(\widehat{\lambda}, \widehat{\mu}, \widehat{\sigma})$ for the fitted model, and verify the result agrees with `em_res$loglik`.

2. **Question:** `em_res$all.loglik` will return the log-likelihood at each iteration of the EM algorithm used to fit the Gaussian mixture model. Plot the values in `em_res$all.loglik`, and verify they are increasing.

The challenge with maximizing the log-likelihood

For other models in this course, we have optimized quantities like the log-likelihood directly (by taking a derivative, and either solving in closed form or using an iterative procedure like gradient descent or Newton's method). However, the log-likelihood for the Gaussian mixture model is **hard** to optimize directly.

Why? Because there is a *sum inside a log*:

$$\ell(\lambda, \mu, \sigma) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} \lambda_j f(X_i | Z_i = j) \right)$$

So what do we do instead?

Introducing a new quantity to maximize

Let $\theta = (\lambda, \mu, \sigma)$ be the collection of all parameters we are trying to estimate for the Gaussian mixture model. Let $\theta^{(t)}$ be our current estimates of these parameters, at iteration t, and let

$$\gamma_{ij}^{(t)} = P^{(t)}(Z_i = j | X_i, \theta^{(t)})$$

be the posterior probabilities calculated with $\theta^{(t)}$. Then define

$$Q(\theta | \theta^{(t)}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij}^{(t)} \log(\lambda_j f(X_i | Z_i = j)) - \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij}^{(t)} \log(\gamma_{ij}^{(t)})$$

Two important results can be shown (you do not need to prove these results!):

- $\ell(\lambda, \mu, \sigma) \geq Q(\theta | \theta^{(t)})$ (this is a result of a probability inequality called *Jensen's inequality*)
- Maximizing $Q(\theta | \theta^{(t)})$ helps us maximize $\ell(\lambda, \mu, \sigma)$

Example

Suppose we start our estimation procedure at the initial values $\lambda^{(0)} = (0.5, 0.5)$, $\mu^{(0)} = (1, 5)$, and $\sigma^{(0)} = (0.5, 0.5)$.

The code below calculates the posterior probabilities $\gamma_{ij}^{(0)}$:

```
calc_probs <- function(x, lambda, mu, sigma){
  (lambda[2] * dnorm(x, mu[2], sigma[2]))/(
    lambda[2] * dnorm(x, mu[2], sigma[2]) + lambda[1] * dnorm(x, mu[1], sigma[1])
  )
}

lambda <- c(0.5, 0.5)
mu <- c(1, 5)
sigma <- c(1, 1)

probs <- calc_probs(x, lambda, mu, sigma)
```

Now we wish to update our parameter estimates. Let's start with updating μ_1, to get $\mu_1^{(1)}$. The code below plots both ℓ and Q as a function of μ_1, when we keep $\lambda_1^{(0)} = \lambda_2^{(0)} = 0.5$, $\mu_2^{(0)} = 5$, and $\sigma_1^{(0)} = \sigma_2^{(0)} = 1$.

```
compute_q <- function(x, lambda, mu, sigma, probs){
  sum(probs*log(lambda[2]*dnorm(x, mean=mu[2], sd=sigma[2])) +
    (1-probs)*log(lambda[1]*dnorm(x, mean=mu[1], sd=sigma[1]))) -
    sum(probs*log(probs)) - sum((1-probs)*log(1-probs))
}

mu1_vals <- seq(-2, 2, 0.01)

loglik_vals <- sapply(mu1_vals,
                      function(mu1){compute_loglik(x, lambda, c(mu1, 5), sigma)})

q_vals <- sapply(mu1_vals,
                 function(mu1){compute_q(x, lambda, c(mu1, 5), sigma, probs)})

plot(mu1_vals, loglik_vals, type="l", xlab = "mu1", ylab="")
lines(mu1_vals, q_vals, col="red")
legend("bottom",
       legend=c("True log-likelihood (l)","Lower bound (Q)"),
       col=c("black", "red"), lty=c(1, 1))
```

3. **Question:** Make the plot, and verify that $Q \leq \ell$ at all points.

4. **Question:** The current guess for μ_1 is $\mu_1^{(0)} = 1$. How do Q and ℓ compare at $\mu_1 = 1$?

5. **Question:** Approximately which value of μ_1 maximizes Q? Is the log-likelihood ℓ also higher at this value of μ_1 than at $\mu_1^{(0)} = 1$?

Likewise, we can make a similar plot for μ_2. The code below plots both ℓ and Q as a function of μ_2, when we keep $\lambda_1^{(0)} = \lambda_2^{(0)} = 0.5$, $\mu_1^{(0)} = 1$, and $\sigma_1^{(0)} = \sigma_2^{(0)} = 1$.

```
mu2_vals <- seq(2, 6, 0.01)

loglik_vals <- sapply(mu2_vals,
                      function(mu2){compute_loglik(x, lambda, c(1, mu2), sigma)})

q_vals <- sapply(mu2_vals,
                 function(mu2){compute_q(x, lambda, c(1, mu2), sigma, probs)})

plot(mu2_vals, loglik_vals, type="l", xlab = "mu2", ylab="")
lines(mu2_vals, q_vals, col="red")
legend("bottom",
       legend=c("True log-likelihood (l)","Lower bound (Q)"),
       col=c("black", "red"), lty=c(1, 1))
```

6. **Question:** The current guess for μ_2 is $\mu_2^{(0)} = 5$. How do Q and ℓ compare at $\mu_2 = 5$?

7. **Question:** Approximately which value of μ_2 maximizes Q? Is the log-likelihood ℓ also higher at this value of μ_2 than at $\mu_2^{(0)} = 5$?

Updating μ

The update rule for μ is

$$\mu_j^{(1)} = \frac{\sum_{i=1}^{n} X_i P^{(0)}(Z_i = j|X_i)}{\sum_{i=1}^{n} P^{(0)}(Z_i = j|X_i)}$$

The code below calculates the updated means $\mu^{(1)}$ for the 2-component Gaussian mixture:

```
update_mu <- function(x, probs){
  mu1 <- sum(x*(1-probs))/sum(1-probs)
  mu2 <- sum(x*probs)/sum(probs)
  return(c(mu1, mu2))
}

update_mu(x, probs)
```

8. **Question:** Run the code. How do the values $\mu_1^{(1)}$ and $\mu_2^{(1)}$ compare to your answers in questions 5 and 7?

Doing the calculus

We have defined

$$Q(\theta|\theta^{(t)}) = \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(t)} \log(\lambda_j f(X_i|Z_i = j)) - \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(t)} \log(\gamma_{ij}^{(t)})$$

Substituting the $N(\mu_j, \sigma_j^2)$ pdf, we have

$$Q(\theta|\theta^{(t)}) = \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(t)} \log\left(\lambda_j \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{-\frac{1}{2\sigma_j^2}(X_i - \mu_j)^2\right\}\right) - \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(t)} \log(\gamma_{ij}^{(t)})$$

9. **Question:** Show that

$$\frac{\partial Q}{\partial \mu_j} = \frac{1}{\sigma_j^2}\sum_{i=1}^{n} \gamma_{ij}^{(t)}(X_i - \mu_j)$$

10. **Question:** To solve for μ_j, we set $\frac{\partial Q}{\partial \mu_j} = 0$. Verify that the value of μ_j which solves this equation is

$$\mu_j = \frac{\sum_{i=1}^{n} \gamma_{ij}^{(t)} X_i}{\sum_{i=1}^{n} \gamma_{ij}^{(t)}}$$

That is, we have derived our update rule for the means!