

Methods of Maximizing the Likelihood

Maximum likelihood estimation requires maximization of the log likelihood
 $\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta} | \mathbf{Y})$.

In most cases, this means taking derivatives and solving likelihood equations

$$\underline{S}(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}^T} \ell(\boldsymbol{\theta}) = 0.$$

Sometimes we can do this analytically (yay!).

When an analytical solution doesn't exist, we have options:

- Standard optimization methods like Newton-Raphson
(or fancy ones like gradient descent).
- Profile likelihood. (later).
- EM algorithm (today).

"Expectation"
↓
"maximization"
↓

1 EM Algorithm

Approach solving the likelihood equation via viewing the observed data \mathbf{Y} as incomplete and that there is missing data \mathbf{Z} that would make the problem simpler if we had it.

↳ sometimes it is actually missing data

↳ others just additional data we wish we had.

Intuition for
data we
"wish we had"
"how we could
proceed."

Example (Two-Component Mixture): Suppose Y_1, \dots, Y_n are iid from the mixture density

$$f(y; \boldsymbol{\theta}) = p f_1(y; \boldsymbol{\mu}_1, \Sigma_1) + (1-p) f_2(y; \boldsymbol{\mu}_2, \Sigma_2),$$

where f_1 and f_2 are bivariate normal densities with mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and variance matrices Σ_1 and Σ_2 , respectively. Thus, the parameter vector $\boldsymbol{\theta} = (p, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2)$ and the likelihood is

$$L(p, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = \prod_{i=1}^n [p f_1(Y_i; \boldsymbol{\mu}_1, \Sigma_1) + (1-p) f_2(Y_i; \boldsymbol{\mu}_2, \Sigma_2)]$$

$$\Rightarrow \ell(p, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = \sum_{i=1}^n \log \{ p f_1(Y_i; \boldsymbol{\mu}_1, \Sigma_1) + (1-p) f_2(Y_i; \boldsymbol{\mu}_2, \Sigma_2) \}$$

... and we're stuck.

We cannot get nice expressions for $\hat{\boldsymbol{\mu}}_{\text{MLE}}$ or $\hat{\Sigma}_{k, \text{MLE}}$ $k=1, 2$.

Actually this log likelihood has maxima on boundary of the parameter space \Rightarrow not well-behaved.

```

library(mvtnorm) ## multivariate normal

p = .6
mu1 <- c(0, 0)
sig1 <- matrix(c(1, 0, 0, 1), ncol = 2)
mu2 <- c(1.5, 1.5)
sig2 <- matrix(c(1, .6, .6, 1), ncol = 2)

## sample from the mixture
n <- 50
z <- rbinom(n, 1, p)

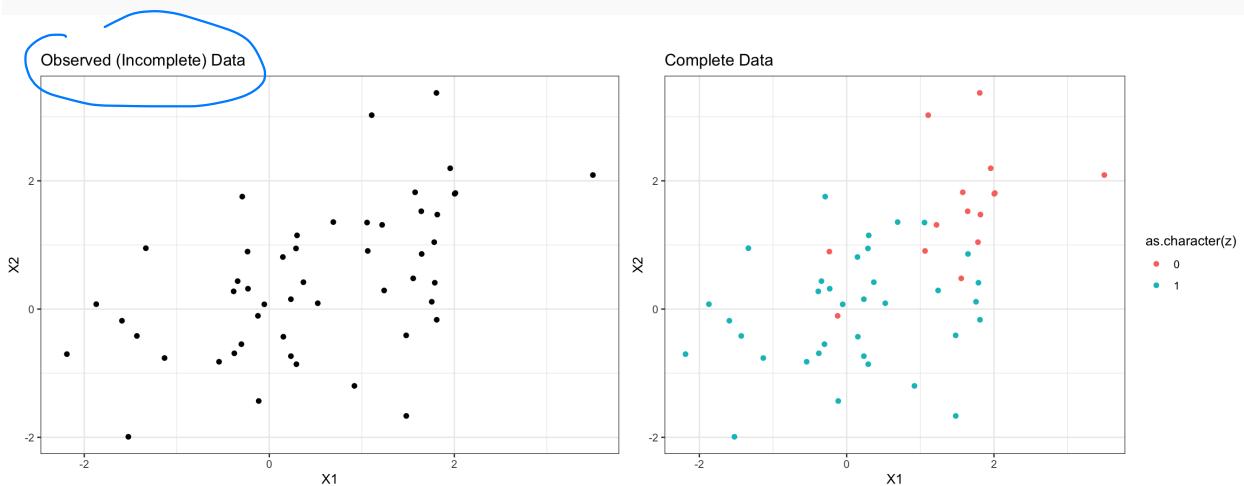
y1 <- rmvnorm(sum(z), mean = mu1, sigma = sig1)
y2 <- rmvnorm(n - sum(z), mean = mu2, sigma = sig2)
Y <- matrix(NA, nrow = n, ncol = 2) ## observed data
Y[z == 1, ] <- y1
Y[z == 0, ] <- y2

df <- data.frame(Y, z)

## plot data
ggplot(df) +
  geom_point(aes(X1, X2)) +
  ggtitle("Observed (Incomplete) Data")

ggplot(df) +
  geom_point(aes(X1, X2, colour = as.character(z))) +
  ggtitle("Complete Data")

```



Let's try to maximize the likelihood

```

# loglikelihood of incomplete data--no knowledge of z
loglik_mixture <- function(par, data) {
  p <- plogis(par[1]) # p guaranteed to be in [0,1]
  mu1 <- c(par[2], par[3])
  sig1 <- matrix(c(exp(par[4]), par[5], par[5], exp(par[4])), nrow
    = 2)
  mu2 <- c(par[6], par[7])
  sig2 <- matrix(c(exp(par[8]), par[9], par[9], exp(par[8])), nrow
    = 2)
  # note: exponential guarantees the diagonal elements are
  # positive, but
  # nothing to guarantee matrices are positive definite. (Could do
  # square root)

  out <- log(p * dmvnorm(data, mean = mu1, sigma = sig1) +
    (1-p) * dmvnorm(data, mean = mu2, sigma = sig2))
  return(sum(out))
}

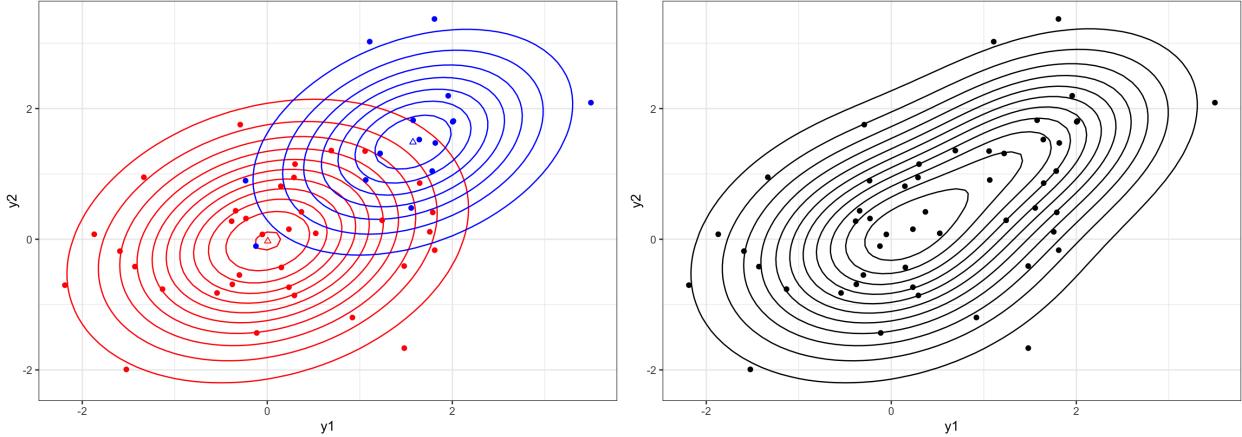
## optimize from different starting values
mle1 <- optim(c(0, -.2, -.2, .5, 0, 2, 2, .5, 0), loglik_mixture,
  data = y, control = list(fnscale = -1))
mle2 <- optim(c(.405, 0, 0, 0, 0, 1.5, 1.5, 0, .6), loglik_mixture,
  data = y, control = list(fnscale = -1))

```

Parameter	Truth	MLE1	MLE2
p	0.6	0.6771	0.6634
μ_{11}	0.0	0.0307	0.0050
μ_{12}	0.0	-0.0512	-0.0281
Σ_{111}	1.0	0.9757	0.9757
Σ_{112}	0.0	0.2178	0.2267
μ_{21}	1.5	1.5597	1.5744
μ_{22}	1.5	1.4815	1.4859
Σ_{211}	1.0	0.7161	0.7220
Σ_{212}	0.6	0.2679	0.2436

OK, not great not terrible.

Fitted results:



This seems pretty good... can we break this with initialization?

```
# Centered the second mixture component at a data point, and shrink
# variance, so normal is super-concentrated around that point.
loglik_mixture(c(.6, 0, 0, 0, 0, y[30, 1], y[30, 2], -50, 0), data =
y)

## [1] -137.7964
```

```
mle3 <- optim(c(.6, 0, 0, 0, 0, y[30, 1], y[30, 2], -50, 0),
loglik_mixture, data = y, control = list(fnscale = -1))
```

Parameter	Truth	MLE3
p	0.6	0.9873 <i>yikes.</i>
μ_{11}	0.0	0.0000
μ_{12}	0.0	0.0000
Σ_{111}	1.0	1.0000
Σ_{112}	0.0	0.0000
μ_{21}	1.5	1.8067
μ_{22}	1.5	3.3712
Σ_{211}	1.0	0.0000 <i>← that's bad.</i>
Σ_{212}	0.6	0.0000

What would change if we were given the complete data, where $Z_i \stackrel{iid}{\sim} \text{Bern}(p)$?
 now we know cluster assignment!

$$f_{YZ}(y, z; \theta) = (p f_1(y; \mu_1, \Sigma_1))^z ((1-p) f_2(y; \mu_2, \Sigma_2))^{1-z}$$

$$\Rightarrow l(p, \mu_1, \mu_2, \Sigma_1, \Sigma_2 | y, z) = \sum_{i=1}^n \left\{ z_i \log f_1(y_i; \mu_1, \Sigma_1) + (1-z_i) \log f_2(y_i; \mu_2, \Sigma_2) + z_i \log p + (1-z_i) \log (1-p) \right\}$$

$$\frac{\partial l(\theta | y, z)}{\partial \mu_1} = \sum_{i=1}^n z_i \frac{\partial \log f_1(y_i; \mu_1, \Sigma_1)}{\partial \mu_1}$$

$$\log f_1(y_i; \mu_1, \Sigma_1) = -\log 2\pi - \frac{1}{2} \log \det \Sigma_1 - \frac{1}{2} (y_i - \mu_1)^T \Sigma_1^{-1} (y_i - \mu_1)$$

$$\Rightarrow \frac{\partial \log f_1(y_i; \mu_1, \Sigma_1)}{\partial \mu_1} = -\sum_{i=1}^n (y_i - \mu_1)$$

plugging in:

$$\frac{\partial l(\theta | y, z)}{\partial \mu_1} = -\sum_{i=1}^n z_i \sum_{i=1}^n (y_i - \mu_1) \stackrel{\text{set}}{=} 0 \quad \text{solve} \Rightarrow \hat{\mu}_{1, \text{MLE}} = \frac{1}{n \sum_{i=1}^n z_i} \sum_{i=1}^n z_i y_i$$

so MLE is the sample mean of the observations from the first density (Data).

The other Gaussian parameter estimates are also exactly what you think:

$$\hat{\mu}_{2, \text{MLE}} = \frac{1}{n \{z_i=0\}} \sum_{i=1}^n (1-z_i) y_i, \quad \hat{\Sigma}_{1, \text{MLE}} = \frac{1}{n \{z_i=1\}} \sum_{i=1}^n z_i (y_i - \hat{\mu}_{1, \text{MLE}})^T (y_i - \hat{\mu}_{1, \text{MLE}}), \quad \text{similar for } \hat{\Sigma}_{2, \text{MLE}}.$$

Now p :

$$\frac{\partial l(\theta | y, z)}{\partial p} = \frac{1}{p} \sum_{i=1}^n z_i - \frac{1}{(1-p)} \sum_{i=1}^n (1-z_i) \stackrel{\text{set}}{=} 0$$

$$\sum_{i=1}^n z_i - p \sum_{i=1}^n z_i = np - p \sum_{i=1}^n z_i \Rightarrow \hat{p}_{\text{MLE}} = \frac{\sum_{i=1}^n z_i}{n}$$

also exactly what we expect!

So, if we knew which mixture component the data came from, our life would be easy...

Consider the complete log-likelihood:

$$\ell(p, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2 | \mathbf{Y}, \mathbf{Z}) = \sum_{i=1}^n \{ Z_i \log f_1(Y_i; \boldsymbol{\mu}_1, \Sigma_1) + (1 - Z_i) \log f_2(Y_i; \boldsymbol{\mu}_2, \Sigma_2) \\ + Z_i \log p + (1 - Z_i) \log(1 - p) \}.$$

We could consider the Z_i 's as "weights" which represent our *current* believe in which density each datum come from.

If we were given this information (the complete data) our belief weights are 0s and 1s.

Instead have current belief based on model parameters.

Given what our belief is in the weights of the data, what is our estimate of the model parameters?

This seems circular. (and it is \rightarrow iterative procedure).



$$\hat{\mu}_1^{(k+1)} = \frac{\sum_{i=1}^n w_i^{(k)} y_i}{\sum_{i=1}^n w_i^{(k)}}$$

$$\hat{\mu}_2^{(k+1)} = \frac{\sum_{i=1}^n (1 - w_i^{(k)}) y_i}{\sum_{i=1}^n (1 - w_i^{(k)})}$$

$$\hat{\Sigma}_1^{(k+1)} = \frac{1}{\sum_{i=1}^n w_i^{(k)}} \sum_{i=1}^n w_i^{(k)} (y_i - \hat{\mu}_1^{(k)}) (y_i - \hat{\mu}_1^{(k)})$$

$$\hat{\Sigma}_2^{(k+1)} = \frac{1}{\sum_{i=1}^n (1 - w_i^{(k)})} \sum_{i=1}^n (1 - w_i^{(k)}) (y_i - \hat{\mu}_2^{(k)}) (y_i - \hat{\mu}_2^{(k)})$$

$$\hat{p}^{(k+1)} = \frac{\sum_{i=1}^n w_i^{(k)}}{n}$$

This is the basic intuition for the EM algorithm. We will view our data \mathbf{Y} as incomplete and imagine there is missing data \mathbf{Z} that would make the problem simpler if we had it.

The EM algorithm then follows:

① Write down joint likelihood of the "complete" data (\mathbf{y}, \mathbf{z}) , $L_c(\underline{\theta} | \mathbf{y}, \mathbf{z})$

*don't actually have \mathbf{z} !
need to maximize something
that is only a function of $\underline{\theta}, \mathbf{y}$.
⇒ conditional expectation!*

② E-step: compute conditional expectation of $\log L_c(\underline{\theta} | \mathbf{y}, \mathbf{z})$ given \mathbf{y} , assuming parameter is $\underline{\theta}^{(v)}$

$$Q(\underline{\theta}, \underline{\theta}^{(v)}, \mathbf{y}) = E_{\underline{\theta}^{(v)}} [\log L_c(\underline{\theta} | \mathbf{y}, \mathbf{z}) | \mathbf{y}] = \int \log L_c(\underline{\theta} | \mathbf{y}, \mathbf{z}) f_{\mathbf{z}|\mathbf{y}}(\mathbf{z} | \mathbf{y}, \underline{\theta}^{(v)}) d\mathbf{z}$$

*actually current value
in the iterations.*

③ M-step: Maximize $Q(\underline{\theta}, \underline{\theta}^{(v)}, \mathbf{y})$ wrt $\underline{\theta}$ ($\underline{\theta}^{(v)}$ fixed).

we will prove why this is what we want in a bit.

$$\text{i.e. } \underline{\theta}^{(v+1)} = \arg \max_{\underline{\theta}} Q(\underline{\theta}, \underline{\theta}^{(v)}, \mathbf{y}).$$

repeat ② & ③ until convergence (values of $\underline{\theta}^{(v)}$ and $\underline{\theta}^{(v+1)}$ not changing much).

Example (Two-Component Mixture, Cont'd): The EM algorithm for the two-component Gaussian mixture model is

(initialized) Start with $\hat{\underline{\theta}}^{(0)}$, for $v = 0, 1, 2, \dots$

$$\begin{aligned} \text{① E step: } Q(\underline{\theta}, \hat{\underline{\theta}}^{(v)}, \mathbf{y}) &= E_{\underline{\theta}^{(v)}} [\log L_c(\underline{\theta} | \mathbf{y}, \mathbf{z}) | \mathbf{y}] = \sum_{i=1}^n w_i^{(v)} \log f_1(y_i; \mu_1^{(v)}, \Sigma_1^{(v)}) + (1-w_i^{(v)}) \log f_2(y_i; \mu_2^{(v)}, \Sigma_2^{(v)}) + w_i^{(v)} \log p + \\ w_i^{(v)} &= E_{\underline{\theta}^{(v)}}(z_i | \mathbf{y}) = \frac{p^{(v)} f_1(y_i; \mu_1^{(v)}, \Sigma_1^{(v)})}{p^{(v)} f_1(y_i; \mu_1^{(v)}, \Sigma_1^{(v)}) + (1-p^{(v)}) f_2(y_i; \mu_2^{(v)}, \Sigma_2^{(v)})} \end{aligned}$$

② M-step : see page 7.

Your Turn: Implement the EM algorithm for the two-component mixture model on our example data.

1.1 Convergence of the EM algorithm

We will show that $\ell(\hat{\theta}^{(v+1)}) \geq \ell(\hat{\theta}^{(v)})$.

In other words, each step of the EM algorithm leads to an improvement in the log-likelihood value.

If the likelihood is well-behaved, it will achieve the MLE, otherwise it will achieve a local maximum
 ↳ bounded, unimodal
 ↳ if there is one).

$y = \text{observed data}$

$z = \text{hidden}$

We know $f_{Z|Y}(z|y; \theta) = \frac{f_{YZ}(y, z; \theta)}{f_Y(y|\theta)}$. def'n of conditional density.
 true for any y, z

$$\Rightarrow f_y(y; \theta) = \frac{f_{YZ}(y, z; \theta)}{f_{Z|Y}(z|y; \theta)} \quad \text{just rewritten (not clear why yet).}$$

Assume we observe $\mathbf{y} = (y_1, \dots, y_n)$, then

$$L(\theta | \mathbf{y}) = f_y(\mathbf{y}; \theta) = \frac{f_{YZ}(y_1, z_1; \theta)}{f_{Z|Y}(z_1 | y_1; \theta)} \quad (\text{if iid, product of univariate densities})$$

$$\Rightarrow \ell(\theta | \mathbf{y}) = \log f_{YZ}(y_1, z_1; \theta) - \log f_{Z|Y}(z_1 | y_1; \theta) = \underbrace{\ell_c(\theta | \mathbf{y}, \mathbf{z})}_{\text{log-likelihood of complete data } Y, Z} - \underbrace{\ell(\theta | \{\mathbf{z} | \mathbf{y}\})}_{\text{conditional log-likelihood}} \quad \text{holds for any } z! \quad \rightarrow \text{take expected values}$$

$$\int \ell(\theta | \mathbf{y}) f_{Z|Y}(z | y_1; \hat{\theta}^{(v)}) dz = \int \log f_{YZ}(y_1, z; \theta) f_{Z|Y}(z | y_1; \hat{\theta}^{(v)}) dz = \int \log f_{Z|Y}(z | y_1; \theta) f_{Z|Y}(z | y_1; \hat{\theta}^{(v)}) dz \quad \text{wrt } z | y_1; \hat{\theta}^{(v)}$$

$$\ell(\theta | \mathbf{y}) \int f_{Z|Y}(z | y_1; \hat{\theta}^{(v)}) dz = \dots$$

$$\ell(\theta | \mathbf{y}) = Q(\theta, \hat{\theta}^{(v)}) - H(\theta, \hat{\theta}^{(v)}).$$

So, in order to show that $\ell(\hat{\theta}^{(v+1)}) \geq \ell(\hat{\theta}^{(v)})$, this is the same as

$$Q(\hat{\theta}^{(v+1)}, \hat{\theta}^{(v)}) - H(\hat{\theta}^{(v+1)}, \hat{\theta}^{(v)}) \geq Q(\hat{\theta}^{(v)}, \hat{\theta}^{(v)}) - H(\hat{\theta}^{(v)}, \hat{\theta}^{(v)})$$

Step 1: Show that $H(\underline{\theta}, \hat{\theta}^{(v)})$ is maximized when $\underline{\theta} = \hat{\theta}^{(v)}$.

$$\text{i.e. } H(\underline{\theta}^{(v)}, \hat{\theta}^{(v)}) \geq H(\underline{\theta}, \hat{\theta}^{(v)}) \text{ for any } \underline{\theta}.$$

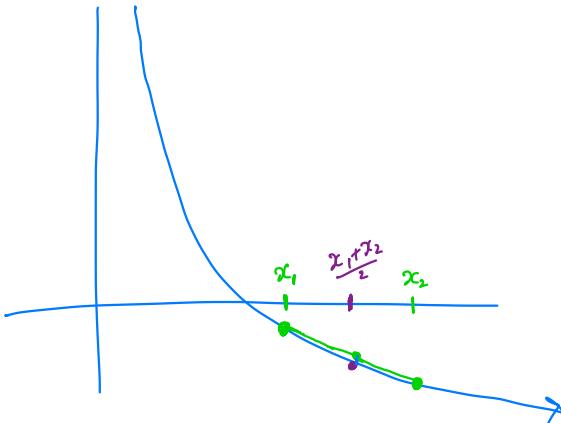
Recall: Jensen's Inequality. A function Φ is convex if $\Phi\left(\frac{x_1+x_2}{2}\right) \leq \frac{1}{2}\Phi(x_1) + \frac{1}{2}\Phi(x_2)$. Then

$$\Phi(E[g(X)]) \leq E[\Phi(g(X))], \Leftrightarrow \mathbb{E}\left(\int g(z)f(z)dz\right) \leq \int \mathbb{E}(g(z))f(z)dz$$

where g is a real-valued integrable function.

where $X \sim f$.

Fact: $-\log$ is convex



Consider $H(\underline{\theta}^{(v)}, \hat{\theta}^{(v)}) - H(\underline{\theta}, \hat{\theta}^{(v)})$ WTS this is positive $\forall \underline{\theta}$.

$$H(\underline{\theta}, \hat{\theta}^{(v)}) = \int \log(f_{z|y}(z|y; \underline{\theta})) f_{z|y}(z|y; \hat{\theta}^{(v)}) dz$$

$$\Rightarrow H(\underline{\theta}^{(v)}, \hat{\theta}^{(v)}) - H(\underline{\theta}, \hat{\theta}^{(v)}) = \int (\log f_{z|y}(z|y; \hat{\theta}^{(v)}) - \log f_{z|y}(z|y; \underline{\theta})) f_{z|y}(z|y; \hat{\theta}^{(v)}) dz$$

$$= \int -\log\left(\frac{f_{z|y}(z|y; \underline{\theta})}{f_{z|y}(z|y; \hat{\theta}^{(v)})}\right) f_{z|y}(z|y; \hat{\theta}^{(v)}) dz.$$

$$\geq -\log \left\{ \int \frac{f_{z|y}(z|y; \underline{\theta})}{f_{z|y}(z|y; \hat{\theta}^{(v)})} f_{z|y}(z|y; \hat{\theta}^{(v)}) dz \right\}$$

$$= -\log \left\{ \int f_{z|y}(z|y; \underline{\theta}) dz \right\}$$

$$= 0$$

$$\Rightarrow H(\underline{\theta}^{(v)}, \hat{\theta}^{(v)}) \geq H(\underline{\theta}, \hat{\theta}^{(v)}) \quad \forall \underline{\theta} //$$

This allows us to only focus on maximizing Q when performing the EM algorithm!

Step 2: Find a $\hat{\theta}^{(t+1)}$ that will optimize Q .

Recall goal is to $\hat{\theta}^{(t+1)}$ s.t. $l(\hat{\theta}^{(t+1)} | y) \geq l(\hat{\theta}^{(t)} | y)$.

And $l(\theta | y) = Q(\theta, \hat{\theta}^{(t)}) - H(\theta, \hat{\theta}^{(t)})$.

Let $\hat{\theta}^{(t+1)} = \underset{\theta}{\operatorname{argmax}} Q(\theta, \hat{\theta}^{(t)})$.

We know $H(\hat{\theta}^{(t+1)}, \hat{\theta}^{(t)}) \leq H(\hat{\theta}^{(t)}, \hat{\theta}^{(t)})$ because true all θ .

+ $Q(\hat{\theta}^{(t+1)}, \hat{\theta}^{(t)}) \geq Q(\hat{\theta}^{(t)}, \hat{\theta}^{(t)})$ by optimization.

$$\begin{aligned} \text{So } l(\hat{\theta}^{(t+1)} | y) &= Q(\hat{\theta}^{(t+1)}, \hat{\theta}^{(t)}) - H(\hat{\theta}^{(t+1)}, \hat{\theta}^{(t)}) \\ &\leq Q(\hat{\theta}^{(t+1)}, \hat{\theta}^{(t)}) - H(\hat{\theta}^{(t+1)}, \hat{\theta}^{(t)}) = l(\hat{\theta}^{(t+1)} | y) \checkmark \end{aligned}$$

Example (Two-Component Mixture, Cont'd):

$$Q(\theta, \hat{\theta}^{(v)}) = \int \log f_{yz}(y, z; \theta) f_{z|y}(z|y; \hat{\theta}^{(v)}) dz.$$

for the Gaussian mixture, we know the complete log-likelihood:

$$\log f_{yz}(y, z; \theta) = \sum_{i=1}^n \{ z_i \log f_1(y_i; \mu_1, \Sigma_1) + (1-z_i) \log f_2(y_i; \mu_2, \Sigma_2) + z_i \log p + (1-z_i) \log (1-p) \}$$

To get the conditional density,

$$f_{z|y}(z|y; \hat{\theta}^{(v)}) = \prod_{i=1}^n f_{z|y}(z_i|y_i; \hat{\theta}^{(v)}).$$

$$f_{z|y}(z_i|y_i; \hat{\theta}^{(v)}) = \frac{f_{yz}(y_i; z_i; \hat{\theta}^{(v)})}{f_y(y_i; \hat{\theta}^{(v)})} \quad \begin{matrix} \text{complete density} \\ \text{observed density.} \end{matrix}$$

$$= \frac{\left[\hat{p}^{(v)} f_1(y_i; \hat{\mu}_1, \hat{\Sigma}_1) \right]^{z_i} \left[(1-\hat{p}^{(v)}) f_2(y_i; \hat{\mu}_2, \hat{\Sigma}_2) \right]^{1-z_i}}{\hat{p}^{(v)} f_1(y_i; \hat{\mu}_1, \hat{\Sigma}_1) + (1-\hat{p}^{(v)}) f_2(y_i; \hat{\mu}_2, \hat{\Sigma}_2)}$$

$$p(z_i=1 | y=y_i) = \frac{\hat{p}^{(v)} f_1(y_i; \hat{\mu}_1, \hat{\Sigma}_1)}{\hat{p}^{(v)} f_1(y_i; \hat{\mu}_1, \hat{\Sigma}_1) + (1-\hat{p}^{(v)}) f_2(y_i; \hat{\mu}_2, \hat{\Sigma}_2)} \stackrel{\text{define } \hat{w}_i^{(v)}, \text{ then}}{=} \hat{w}_i^{(v)}$$

$$\Rightarrow z_i | y_i, \hat{\theta}^{(v)} \sim \text{Bern}(\hat{w}_i^{(v)}).$$

$$\Rightarrow Q(\theta, \hat{\theta}^{(v)}) = \sum_{i=1}^n E_{\substack{z_i|y_i \\ \theta=\hat{\theta}^{(v)}}} [\log f_{yz}(y_i, z_i; \theta)] \quad \text{and} \quad E_{\substack{z_i|y_i \\ \theta=\hat{\theta}^{(v)}}} [g(z_i)] = g(1)\hat{w}_i^{(v)} + g(0)(1-\hat{w}_i^{(v)}).$$

$$= \sum_{i=1}^n \hat{w}_i^{(v)} [\log f_1(y_i; \mu_1, \Sigma_1) + \log p] + (1-\hat{w}_i^{(v)}) [\log f_2(y_i; \mu_2, \Sigma_2) + \log (1-p)]$$

take derivatives, solve.
set $\hat{w}_i^{(v)} = 0$.

↳ Which yields the intuitive update formulas from before!

So "plugging in the weights" makes sense from an optimization standpoint in this example.

In general can't always separate $E + M$ in this way for Q .

The EM algorithm allows us to obtain $\hat{\theta}_{EM}$, the parameter estimate which optimizes the algorithm.

If likelihood is "nice" = $\hat{\theta}_{MLE}$

1.2 Variance Estimation for EM estimates

\rightarrow likelihood is well-behaved.

The EM algorithm finds the MLE, but it does not automatically produce an estimate of the covariance matrix. Why not?

Couldn't we just look at the curvature of the optimized surface?

No. We are optimizing Q , which is not the log-likelihood! ($Q - H$).

\Rightarrow we can't rely on information results without computing the Hessian of the log-likelihood.
(which may be hard)

There are several options to estimate the variance.

1. Bootstrapping (we will discuss bootstrapping in depth later in the course).

A simple procedure is easy to imagine:

1. Find $\hat{\theta}_{EM}$ from the original data $\mathcal{Y} = Y_1, \dots, Y_n$

2. Resample data with replacement to obtain bootstrap samples (sample from empirical dsn).
 $\mathcal{Y}^{*b} = Y_1^{*b}, \dots, Y_n^{*b}$ for $b = 1, \dots, B$ $\xleftarrow{\text{# of bootstrap samples}}$

$$\mathcal{Y}^{*b} = Y_1^{*b}, \dots, Y_n^{*b} \quad \text{for } b = 1, \dots, B$$

Find $\hat{\theta}_{EM}^{*b}$ for each \mathcal{Y}^{*b} .

3. Use $\hat{\theta}_{EM}^{*b}$'s to get bootstrap CI for $\hat{\theta}_{EM}$.

\hookrightarrow percentile bootstrap.

\hookrightarrow t-based

\hookrightarrow BCa bootstrap.

Straight forward, easy.

Can be computationally expensive (need EM $b=1, \dots, B$ times).

\hookrightarrow could be done in parallel to ease the burden.

2. Louis's Method

$$\ell(\underline{\theta}|\underline{y}) = Q(\underline{\theta}, \underline{\theta}^{(v)}) - H(\underline{\theta}, \underline{\theta}^{(v)}) \quad \text{true for any } \underline{\theta}^{(v)}$$

So in particular $\ell(\underline{\theta}|\underline{y}) = Q(\underline{\theta}, \underline{\theta}) - H(\underline{\theta}, \underline{\theta})$

$$\Rightarrow -\ell''(\underline{\theta}|\underline{y}) = -Q''(\underline{\theta}, \underline{\theta}) + H''(\underline{\theta}, \underline{\theta}). \quad (\text{derivatives wrt to } \underline{\theta} \text{ first argument})$$

\nearrow

$$n \bar{I}(\underline{y}, \underline{\theta})$$

from before.

observed information = "complete information" - "missing information"

$$-\bar{Q}''(\underline{\theta}, \underline{\theta}) = -E_{z|y} [\ell''(\underline{\theta}|\underline{y}, z)] = - \int \ell''(\underline{\theta}|\underline{y}, z) f_{z|y}(z|\underline{y}; \underline{\theta}) dz$$

Some what similar in form to Fisher Information (except for dsn expectation is wrt).

similar statement about $H''(\underline{\theta}, \underline{\theta})$.

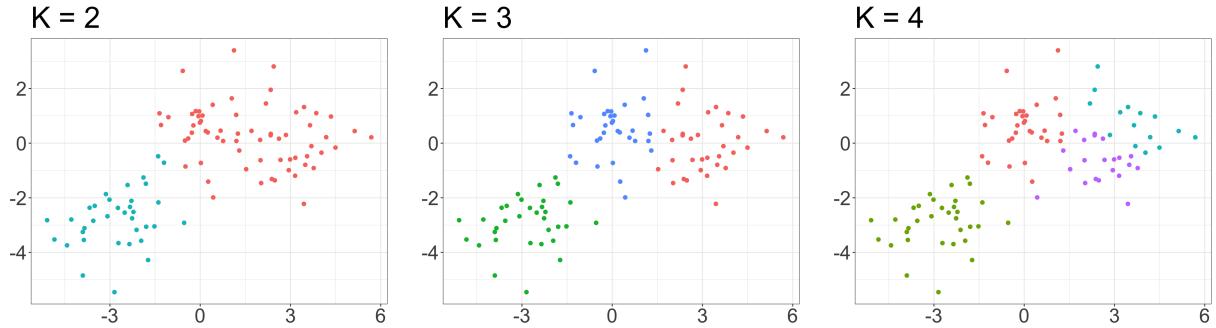
assuming integration
and differentiation order
are interchangeable

1.3 Another way to cluster: K-means

Goal of clustering:

Methods for clustering include hierarchical and non-hierarchical, algorithmic and model-based.

K-means is a simple and elegant approach to partition a data set into K distinct, non-overlapping clusters.



The K -means clustering procedure results from a simple and intuitive mathematical problem. Let C_1, \dots, C_K denote sets containing the indices of observations in each cluster. These satisfy two properties:

1.

2.

Idea:

The *within-cluster variation* for cluster C_k is a measure of the amount by which the observations within a cluster differ from each other.

To solve this, we need to define within-cluster variation.

This results in the following optimization problem that defines K -means clustering:

A very simple algorithm has been shown to find a local optimum to this problem:

Questions about the algorithm:

1. How do we define distance?
2. How do we choose starting values?
3. How do we choose k ?

Compared to the Gaussian mixture problem,