# Methods of Maximizing the Likelihood

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

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

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

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 $\boldsymbol{Y}$ as incomplete and that there is missing data $\boldsymbol{Z}$ 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 $Y_1, \ldots, Y_n$ are iid from the mixture density

$$f(y; \boldsymbol{\theta}) = pf_1(y; \boldsymbol{\mu}_1, \Sigma_1) + (1-p)f_2(y; \boldsymbol{\mu}_2, \Sigma_2),$$

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

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

$$\Rightarrow \ell(p, \underline{\mu}_1, \underline{\mu}_2, \Sigma_1, \Sigma_2) = \sum_{i=1}^{n} \log \left\{ p f_1(\underline{y}_i; \underline{\mu}_1, \Sigma_1) + (1-p) f_2(\underline{y}_i; \underline{\mu}_2, \Sigma_2) \right\}$$

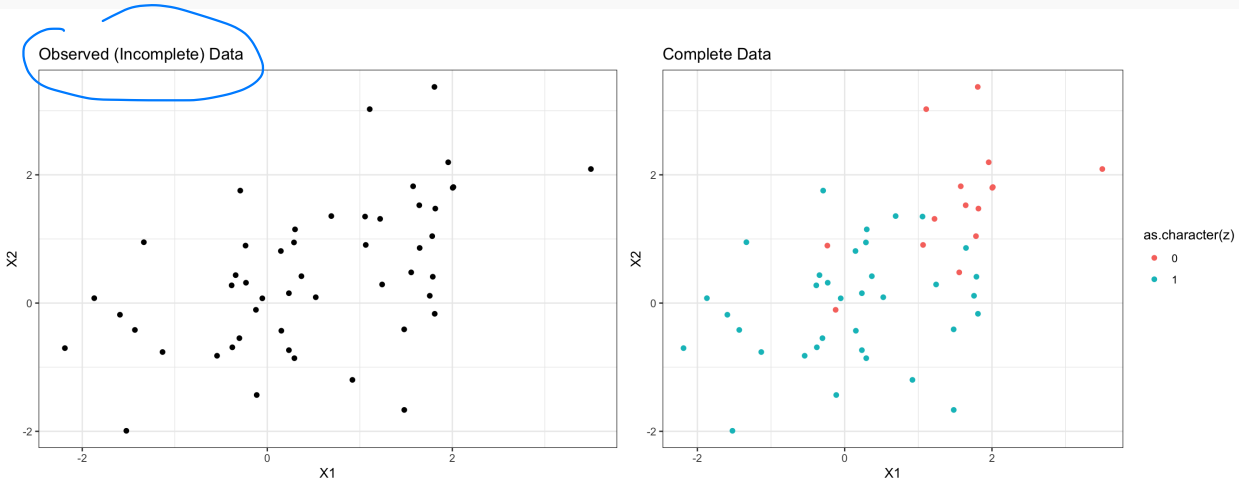
... and we're stuck.

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

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

```r
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")
```

simulate data.



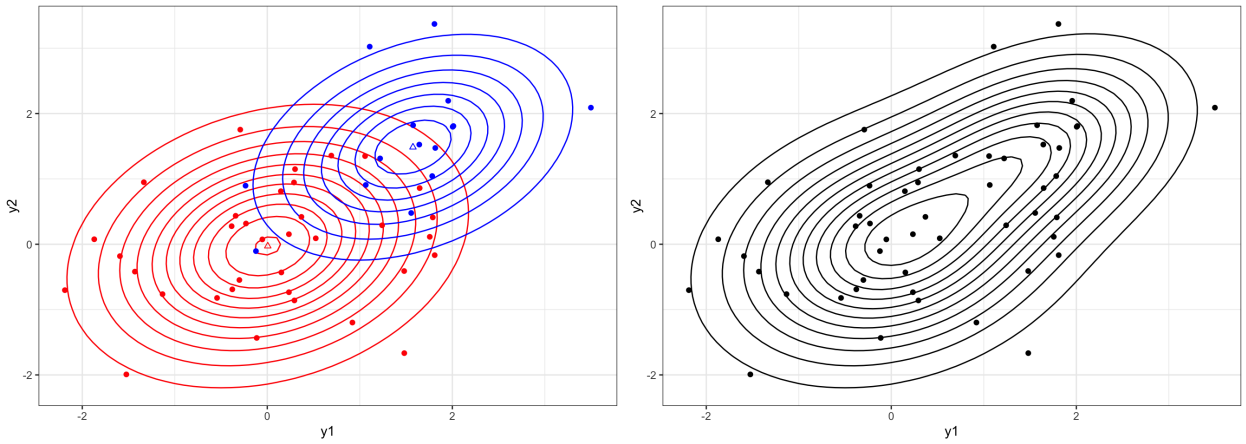

Let's try to maximize the likelihood

```r
# 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 |
| $\mu_{11}$ | 0.0 | 0.0307 | 0.0050 |
| $\mu_{12}$ | 0.0 | -0.0512 | -0.0281 |
| $\Sigma_{111}$ | 1.0 | 0.9757 | 0.9757 |
| $\Sigma_{112}$ | 0.0 | 0.2178 | 0.2267 |
| $\mu_{21}$ | 1.5 | 1.5597 | 1.5744 |
| $\mu_{22}$ | 1.5 | 1.4815 | 1.4859 |
| $\Sigma_{211}$ | 1.0 | 0.7161 | 0.7220 |
| $\Sigma_{212}$ | 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 | Yikes. |
| $\mu_{11}$ | 0.0 | 0.0000 | |
| $\mu_{12}$ | 0.0 | 0.0000 | |
| $\Sigma_{111}$ | 1.0 | 1.0000 | |
| $\Sigma_{112}$ | 0.0 | 0.0000 | |
| $\mu_{21}$ | 1.5 | 1.8067 | |
| $\mu_{22}$ | 1.5 | 3.3712 | |
| $\Sigma_{211}$ | 1.0 | 0.0000 | ← that's bad. |
| $\Sigma_{212}$ | 0.6 | 0.0000 | |

What would change if we were given the complete data, where $Z_i \overset{iid}{\sim} \text{Bern}(p)$?

now we know cluster assignment!

$$f_{Y,Z}(y,z;\underline{\theta}) = \left(p f_1(y,\underline{\mu}_1,\Sigma_1)\right)^z \left((1-p) f_2(y,\underline{\mu}_2,\Sigma_2)\right)^{1-z}$$

$$\Rightarrow \ell(p,\underline{\mu}_1,\underline{\mu}_2,\Sigma_1,\Sigma_2 \mid \underline{Y},\underline{Z}) = \sum_{i=1}^n \{z_i \log f_1(y_i;\underline{\mu}_1,\Sigma_1) + (1-z_i)\log f_2(y_i;\underline{\mu}_2,\Sigma_2) + z_i \log p + (1-z_i)\log(1-p)\}.$$

$$\frac{\partial \ell(\underline{\theta} \mid \underline{Y},\underline{Z})}{\partial \underline{\mu}_1} = \sum_{i=1}^n z_i \frac{\partial \log f_1(y_i;\underline{\mu}_1,\Sigma_1)}{\partial \underline{\mu}_1}$$

$$\log f_1(y_i;\underline{\mu}_1,\Sigma_1) = -\log 2\pi - \frac{1}{2}\log\det\Sigma_1 - \frac{1}{2}(y_i-\underline{\mu}_1)^T\Sigma_1^{-1}(y_i-\underline{\mu}_1)$$

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

plugging in:

$$\frac{\partial \ell(\underline{\theta} \mid \underline{Y},\underline{Z})}{\partial \underline{\mu}_1} = -\sum_{i=1}^n z_i \Sigma_1^{-1}(y_i-\underline{\mu}_1) \overset{\text{set}}{=} 0 \quad \overset{\text{solve}}{\Rightarrow} \quad \hat{\mu}_{1,MLE} = \frac{1}{n_{\{z_i=1\}}} \sum_{i=1}^n z_i y_i$$

So MLE is the sample mean of the observations from the first density (DUH).

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

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

Now $p$:

$$\frac{\partial \ell(\underline{\theta} \mid \underline{Y},\underline{Z})}{\partial p} = \frac{1}{p}\sum_{i=1}^n z_i - \frac{1}{(1-p)}\sum_{i=1}^n (1-z_i) \overset{\text{set}}{=} 0$$

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

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:

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

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 (based on current knowledge of model parameters), what is our estimate of the model parameters?

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

$\hat{\theta}^{(k)} \rightarrow$ belief weights $\rightarrow \hat{\theta}^{(k+1)}$

"E" "M"

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

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

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

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

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

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

① Write down joint likelihood of the "complete" data $(Y, Z)$, $L_c(\underline{\theta} \mid \underline{Y}, \underline{Z})$ (don't actually have $Z$! need to maximize something that is only a function of $\underline{\theta}, \underline{Y}$.)

$\Rightarrow$ conditional expectation!

② E-step: compute conditional expectation of $\log L_c(\underline{\theta} \mid \underline{Y}, \underline{Z})$ given $\underline{Y}$, assuming parameter is $\underline{\theta}^{(v)}$ (actually current value in the iteration.)

$$Q(\underline{\theta}, \underline{\theta}^{(v)}, \underline{Y}) = E_{\underline{\theta}^{(v)}}\left[\log L_c(\underline{\theta} \mid \underline{Y}, \underline{Z}) \mid \underline{Y}\right] = \int \log L_c(\underline{\theta} \mid \underline{Y}, \underline{z}) f_{\underline{Z}|\underline{Y}}(\underline{z} \mid \underline{Y}, \underline{\theta}^{(v)}) \, d\underline{z}.$$

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

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

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

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

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

(initialize) start with $\hat{\underline{\theta}}^{(0)}$, for $v = 0, 1, 2, \ldots$

① E step: $Q(\underline{\theta}, \hat{\underline{\theta}}^{(v)}, \underline{Y}) = E_{\underline{\theta}^{(v)}}\left[\log L_c(\underline{\theta} \mid \underline{Y}, \underline{Z}) \mid \underline{Y}\right] = \sum_{i=1}^{n} \{ w_i^{(v)} \log f_1(y_i, \underline{\mu}_1^{(v)}, \underline{\Sigma}_1^{(v)}) + (1 - w_i^{(v)}) \log f_2(y_i, \underline{\mu}_2^{(v)}, \underline{\Sigma}_2^{(v)}) + w_i^{(v)} \log p^{(v)} + (1 - w_i^{(v)}) \log(1 - p^{(v)}) \}$

$$w_i^{(v)} = E_{\theta^{(v)}}(Z_i \mid \underline{Y}) = \frac{p^{(v)} f_1(y_i; \underline{\mu}_1^{(v)}, \underline{\Sigma}_1^{(v)})}{p^{(v)} f_1(y_i; \underline{\mu}_1^{(v)}, \underline{\Sigma}_1^{(v)}) + (1 - p^{(v)}) f_2(y_i; \underline{\mu}_2^{(v)}, \underline{\Sigma}_2^{(v)})}$$

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 $\ell\left(\hat{\boldsymbol{\theta}}^{(v+1)}\right) \geq \ell\left(\hat{\boldsymbol{\theta}}^{(v)}\right)$.

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

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

$y$ = observed data
$z$ = hidden

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

$\Rightarrow f_Y(y;\underline{\theta}) = \dfrac{f_{YZ}(y,z;\underline{\theta})}{f_{Z|Y}(z|y;\underline{\theta})}$ just rewritten (not clear why yet).

Assume we observe $\boldsymbol{y} = (y_1, \ldots, y_n)$, then

$L(\underline{\theta}|y) = f_Y(y;\theta) = \dfrac{f_{YZ}(y,z;\underline{\theta})}{f_{Z|Y}(z|y;\underline{\theta})}$ (if iid, product of univariate densities)

$\Rightarrow \ell(\underline{\theta}|y) = \log f_{YZ}(y,z;\underline{\theta}) - \log f_{Z|Y}(z|y;\underline{\theta}) = \underbrace{\ell_c(\underline{\theta}|y,z)}_{\text{log-likelihood of complete data } Y,Z} - \underbrace{\ell(\underline{\theta}|\{z|y\})}_{\text{conditional log-likelihood}}$ holds for any $z$! ⇒ take expected values wrt $z|y;\hat{\theta}^{(v)}$

$\int \ell(\theta|y) f_{Z|Y}(z|y,\hat{\underline{\theta}}^{(v)})\,dz = \int \log f_{YZ}(y,z;\underline{\theta}) f_{Z|Y}(z|y;\hat{\underline{\theta}}^{(v)})\,dz - \int \log f_{Z|Y}(z|y;\underline{\theta}) f_{Z|Y}(z|y;\hat{\underline{\theta}}^{(v)})\,dz$

$\ell(\theta|y) \int f_{Z|Y}(z|y;\hat{\underline{\theta}}^{(v)})\,dz = \ldots$

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

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

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

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

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

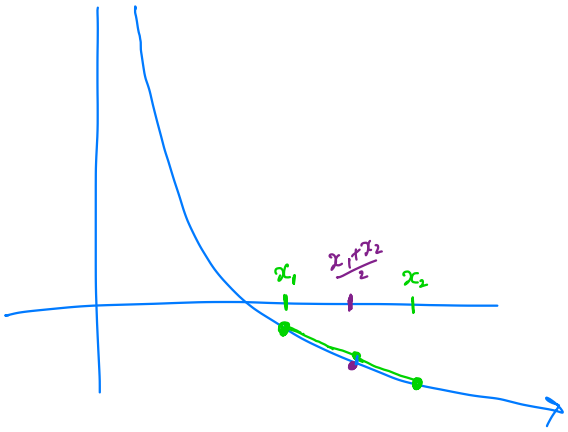
Recall: Jensen's Inequality. A function $\Phi$ is convex if $\Phi(\frac{x_1+x_2}{2}) \leq \frac{1}{2}\Phi(x_1) + \frac{1}{2}\Phi(x_2)$. Then

$$\Phi(\mathrm{E}[g(X)]) \leq \mathrm{E}[\Phi(g(X))], \iff \Phi\left(\int g(x) f(x)\, dx\right) \leq \int \Phi(g(x)) f(x)\, dx$$

where $g$ is a real-valued integrable function.

where $X \sim f$.

Fact: $-\log$ is convex




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

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

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

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

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

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

$$= 0$$

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

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

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

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

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

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

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

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

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

$\leq Q(\hat{\underline{\theta}}^{(v+1)}, \hat{\underline{\theta}}^{(v+1)}) - H(\hat{\underline{\theta}}^{(v+1)}, \hat{\underline{\theta}}^{(v)}) = \ell(\hat{\theta}^{(v+1)} | y)$ ✓

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

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

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

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

To get the conditional density,

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

$$f_{z|y}(z_i \mid y_i; \hat{\underline{\theta}}^{(v)}) = \frac{f_{yz}(y_i, z_i; \hat{\underline{\theta}}^{(v)})}{f_y(y_i; \hat{\underline{\theta}}^{(v)})} \qquad \text{complete density / observed density.}$$

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

$z_i$ can only take value 0 or 1

$$P(z_i = 1 \mid y = y_i) = \frac{\hat{p}^{(v)} f_1(y_i; \hat{\mu}_1^{(v)}, \hat{\Sigma}_1^{(v)})}{\hat{p}^{(v)} f_1(y_i; \hat{\mu}_1^{(v)}, \hat{\Sigma}_1^{(v)}) + (1-\hat{p}^{(v)}) f_2(y_i; \hat{\mu}_2^{(v)}, \hat{\Sigma}_2^{(v)})} := \hat{w}_i^{(v)}; \text{ Then}$$

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

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

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

take derivatives, set = 0, solve.

↳ 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 $\hat{\boldsymbol{\theta}}_{\text{EM}}$, the parameter estimate which optimizes the algorithm.

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

## 1.2 Variance Estimation for EM estimates

→ likelihood is well-behaved.

The EM algorithm find 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).

⟹ 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 $\underline{Y} = \underline{y}_1, \ldots, \underline{y}_n$
2. Resample data with replacement to obtain bootstrap samples (sample from empirical dsn).

$\underline{Y}^{*b} = \underline{y}_1^{*b}, \ldots, \underline{y}_n^{*b}$ for $b = 1, \ldots, B$ ← # of bootstrap samples

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

3. Use $\hat{\theta}_{EM}^{*b}$'s to get bootstrap CI for $\hat{\theta}_{EM}$.
   - ↳ percentile bootstrap.
   - ↳ t-based
   - ↳ BCA bootstrap.

Straight forward, easy.

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

↳ could be done in parallel to ease the burden.

## 2. Louis's Method

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

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

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

$\uparrow$

$n\underline{I}(\underline{y},\underline{\theta})$ from before.

observed information = "complete information" − "missing information"

assuming integration and differentiation order are interchangable

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

Somewhat similar in form to Fisher information (except for dsn expectation is wrt).

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

So to get CIs, need $Q''(\theta,\theta)$ and $H''(\theta,\theta)$:

$Q''(\theta,\theta)$ is the curvature of the optimization surface $\Rightarrow$ can often get numerically.

$$H''(\theta,\theta) = \text{Var}\left[\frac{\partial \log f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y};\underline{\theta})}{\partial \underline{\theta}}\right] \text{ wrt } f_{\underline{z}|\underline{y}}$$ (again use fact ②)

because $H''(\theta,\theta) = \int \frac{\partial^2 \log f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y};\underline{\theta})}{\partial \underline{\theta}} f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y};\underline{\theta})\, d\underline{z}$

A MC estimate of $H''(\theta,\theta)$ is thus the sample variance of $\frac{d \log f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y};\underline{\theta})}{d\underline{\theta}}$

of a sample of $\underline{z}$'s imputed from $f_{\underline{z}|\underline{y}}$

Other options see Givens & Hoeting Section 4.2.3

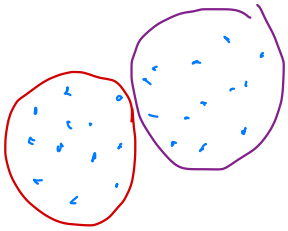
- Supplemental EM
- Empirical Information
- Numerical differentiation to get Hessian.

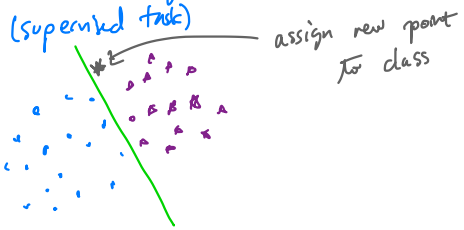
# 1.3 Another way to cluster: K-means

**Goal of clustering:**

Find an optimal grouping for which the observations within each group are "similar" but clusters are "dissimilar" to each other.

cluster analysis (unsupervised) vs. Discriminant Analysis (classification) (supervised task)






goal: discover structure

goal: prediction

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

hierarchical: proceed by going from $n \longrightarrow 1$ clusters (or $1 \rightarrow n$)

non-hierarchical: model-based (finite mixture of Normals)

algorithmic (K-means)

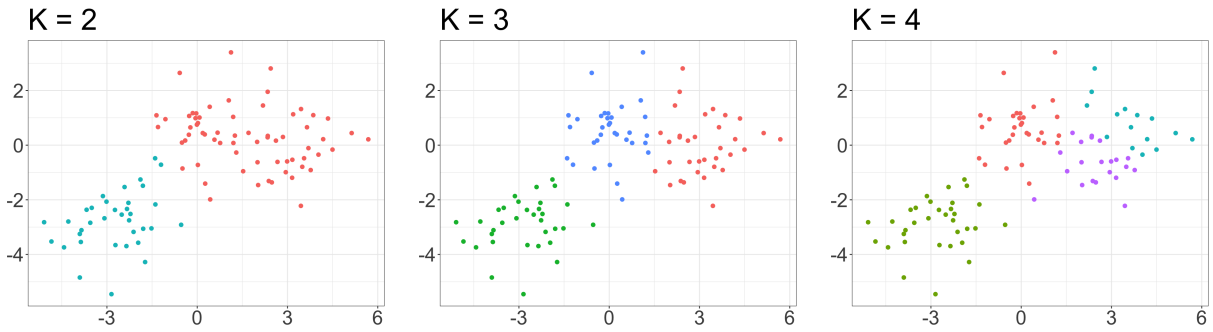
} both work for set # of clusters; but can be applied to different #s.

Not an exhaustive list.

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

First specify how many clusters ($K$), then K-means assigns each observation to one of $K$ clusters.

Eg. clustering $n=100$ observations into $K$ clusters using $p=2$ features.




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

e.g. if obs $i$ is in cluster $k$, $i \in C_k$

1. $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \ldots, n\}$

   each obs. belongs to a cluster

2. $C_k \cap C_{k'} = \emptyset \quad \forall \; k \neq k'$

   clusters are non-overlapping.

**Idea:** "Good" clustering is one for which within-cluster-variation is as small as possible.

↳ clustred data are "similar"

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. call this $W(C_k)$.

Then we want to

$$\underset{C_1,\dots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

i.e. partition data into K clusters s.t. total within cluster variation is minimized.

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

many ways.

Most common: squared Euclidean distance.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

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

objective function.

$$\underset{C_1,\dots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

This is very hard to solve exactly, $\approx K^n$ ways to partition n obs into K clusters!

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

↳ "pretty good solution"

1. Randomly assign a number from 1 to K to each observation (initialize cluster assignments).

2. Iterate until cluster assignments stop changing:

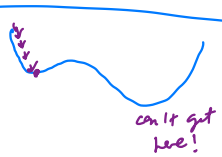
   (a) for each of the K clusters, compute cluster centroid → vector of the p feature means for each cluster.

   (b) Assign each obs to closest centroid cluster.

↑ euclidean distance.

This algorithm is guaranteed to decrease the value of objective function at each step.

Clustering depends on initial (random) cluster assignment.




⇒ run the algorithm multiple times from different initial configurations and choose clustering w/ smallest objective function.

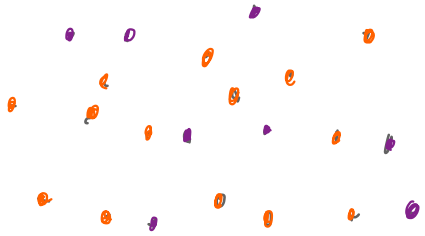
Still need K...

k-means illustration:

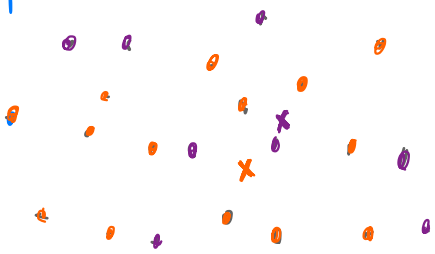
(1)

pick K=2.

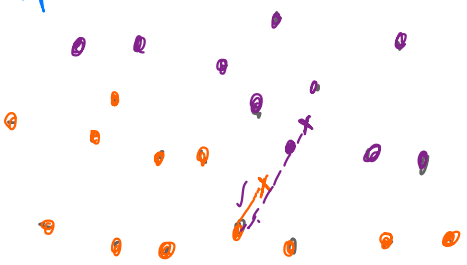
(2) Randomly assign



(3) compute medoids.



(4) Update assignments



repeat (3) & (4) until clusters stable.

Questions about the algorithm:

1. How do we define distance?

   Normally Euclidean.
   Could choose
   - Minkowski $d(\underline{x}_1, \underline{x}_2) = \left( \sum_{i=1}^{p} |x_{1i} - x_{2i}|^q \right)^{1/q}$
   - Mahalanobis

4, Should we scale the data?
Most times yes, unless no.

2. How do we choose starting values?

   randomly usually (shotgun approach).

   Maybe using another method? like hierarchical?

3. How do we choose K?

   look at between SS vs. within SS? (elbow plot of ratio)

   Another way: Dunn index → compares to a "null" clustering.

   No one right way.

Compared to the Gaussian mixture problem,

finite. ↓ also fixed # clusters

"soft assignment" "hard assignment"

In the Gaussian mixture, we get probabilities of assignment, not just assignment.

↓ model-based (making assumptions!)

K-means sensitive to starting values.

In R, kmeans function will fit this algorithm.