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 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

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

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

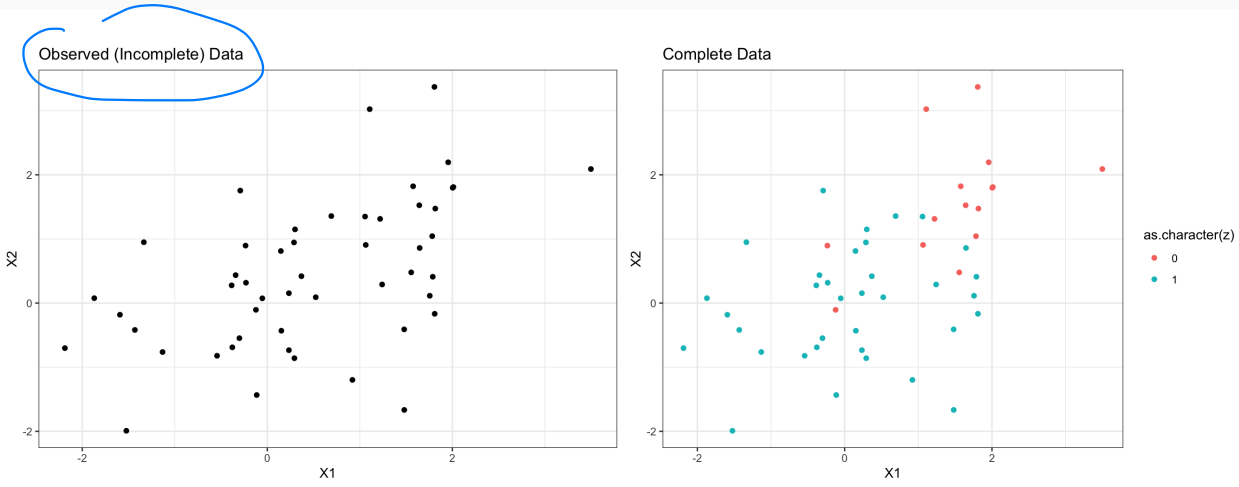
... and we're stuck.

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.



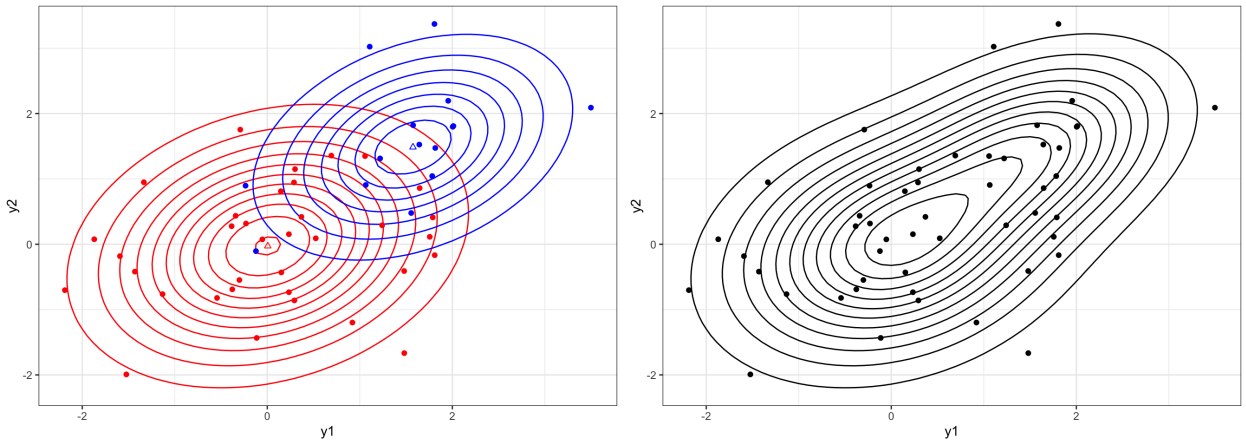
Let's try to maximize the likelihood

```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
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	Yikes.
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	← that's bad.
Σ_{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!

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

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

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

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

$$\Rightarrow \frac{\partial \log f_1(y_i; \mu_1, \Sigma_1)}{\partial \underline{\mu}_1} = -\Sigma_1^{-1}(y_i - \mu_1)$$

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).

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.

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

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:

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

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

We know $f_{Z|Y}(z|y;\boldsymbol{\theta}) = \frac{f_{YZ}(y,z;\boldsymbol{\theta})}{f_Y(y|\boldsymbol{\theta})}$.

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

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

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

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

$$\Phi(\mathrm{E}[g(X)]) \leq \mathrm{E}[\Phi(g(X))],$$

where g is a real-valued integrable function.

Step 2: Find a $\boldsymbol{\theta}^{k+1}$ that will optimize Q.

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

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

1.2 Variance Estimation for EM estimates

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

There are several options to estimate the variance.

1. Bootstrapping

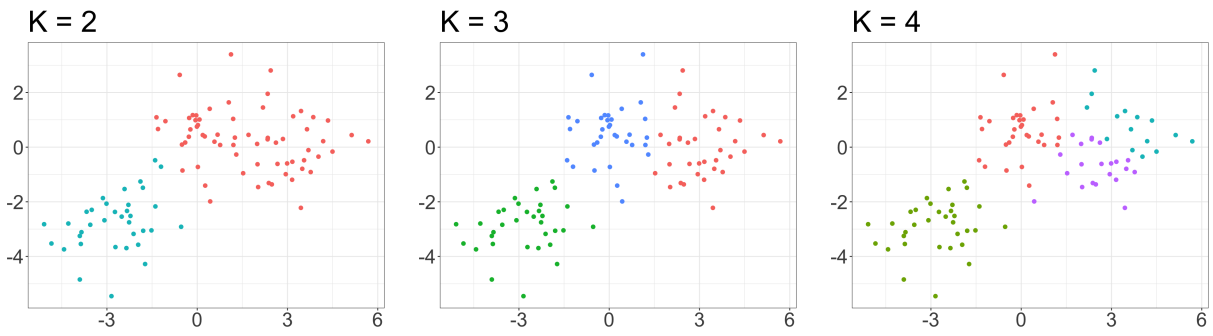
2. **Louis's Method**

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 $C_1, \ldots, C_K$ 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,