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

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

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)

- EM algorithm.

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, other times just additional data we wish we had.

Intuition for what data we "wish we had" & how to 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

$$L(p, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = \prod_{i=1}^{n} \left[pf_1(y_i; \underline{\mu}_1, \Sigma_1) + (1-p) f_2(y_i; \underline{\mu}_2, \Sigma_2) \right]$$

$$\Rightarrow \ell(p, \underline{\mu}_1, \underline{\mu}_2, \Sigma_1, \Sigma_2) = \sum_{i=1}^{n} \log \left\{ pf_1(y_i; \underline{\mu}_1, \Sigma_1) + (1-p) f_2(y_i; \underline{\mu}_2, \Sigma_2) \right\}$$

... and we're stuck.

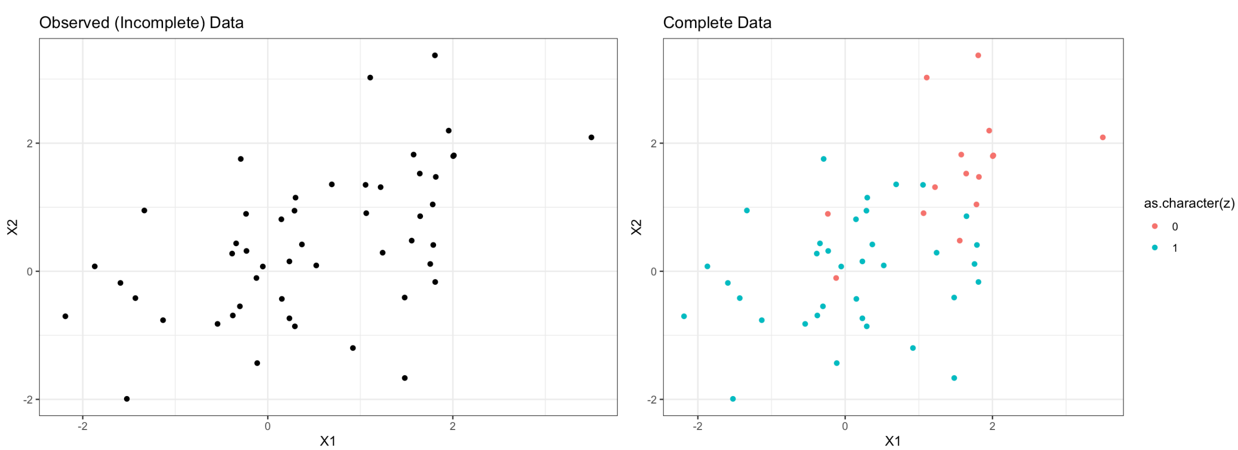
We cannot get nice expressions for $\hat{\mu}_{k,MLE}$ or $\hat{\Sigma}_{k,MLE}$ $k=1,2$.

Actually, this log-likelihood has maxima on boundary of the 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.



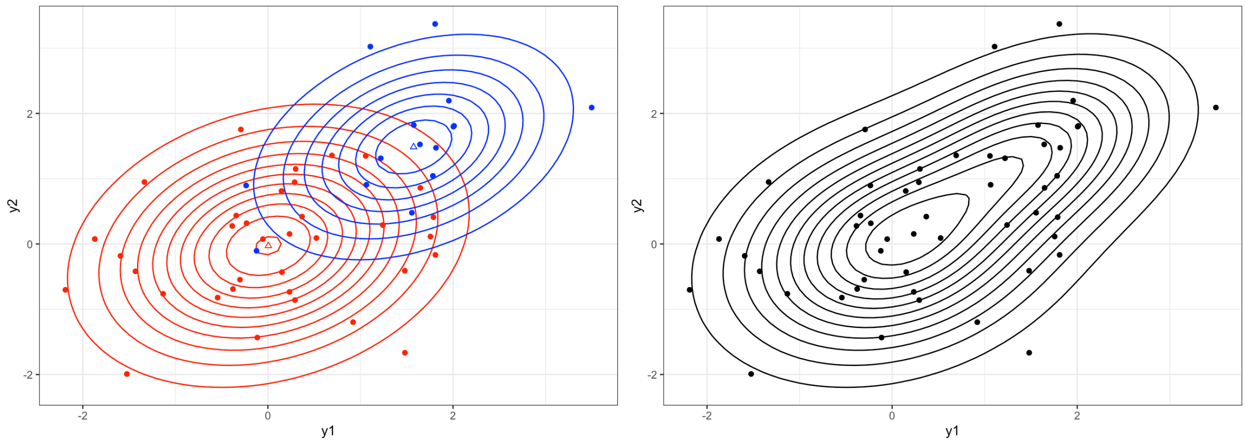
Let's try to maximize the likelihood

true cluster id.

```
        # 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

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
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

Yikes!

makes no sense

Parameter	Truth	MLE3
Σ_{212}	0.6	0.0000

makes no sense.

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

now we know cluster assignments!

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

$$\Rightarrow \ell(p, \mu_1, \mu_2, \Sigma_1, \Sigma_2 \mid Y, Z) = \sum_{i=1}^{n} \{ z_i \log f_1(y_i; \mu_1, \Sigma_1) + (1-z_i) \log f_2(y_i; \mu_2, \Sigma_2) + z_i \log p + (1-z_i)\log(1-p) \}$$

$$\frac{\partial \ell(\theta \mid Y, Z)}{\partial \mu_1} = \sum_{i=1}^{n} z_i \frac{\partial \log f_1(y_i; \mu_1, \Sigma_1)}{\partial \mu_1}$$

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

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

plugging in above:

$$\frac{\partial \ell(\theta \mid Y, Z)}{\partial \mu_1} = -\sum_{i=1}^{n} z_i \Sigma_1^{-1}(y_i - \mu_1) \overset{set}{=} 0$$

$$\sum_{i=1}^{n} z_i y_i = \sum_{i=1}^{n} z_i \mu_1 \Rightarrow \hat{\mu}_{MLE} = \frac{1}{n_{\{z_i=1\}}} \sum_{i=1}^{n} z_i y_i$$

$\Rightarrow$ MLE is the sample mean of the observations from the first density (DUH!).

$$\hat{\mu}_{2,MLE} = \frac{1}{n_{\{z_i=0\}}} \sum_{i=1}^{n} (1-z_i) y_i \qquad \hat{\Sigma}_{1,MLE} = S_1 = \frac{1}{n_{\{z_i=1\}}} \sum_{i=1}^{n} z_i (y_i - \mu_1)(y_i - \mu_1)^T, \qquad \hat{\Sigma}_{2,MLE} = S_2.$$

Now p:

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

also what we would expect!

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

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