

Methods of Maximizing the Likelihood

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

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.

Example (Two-Component Mixture): Suppose Y_1, \dots, 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

$$L(p, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) =$$

```

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)

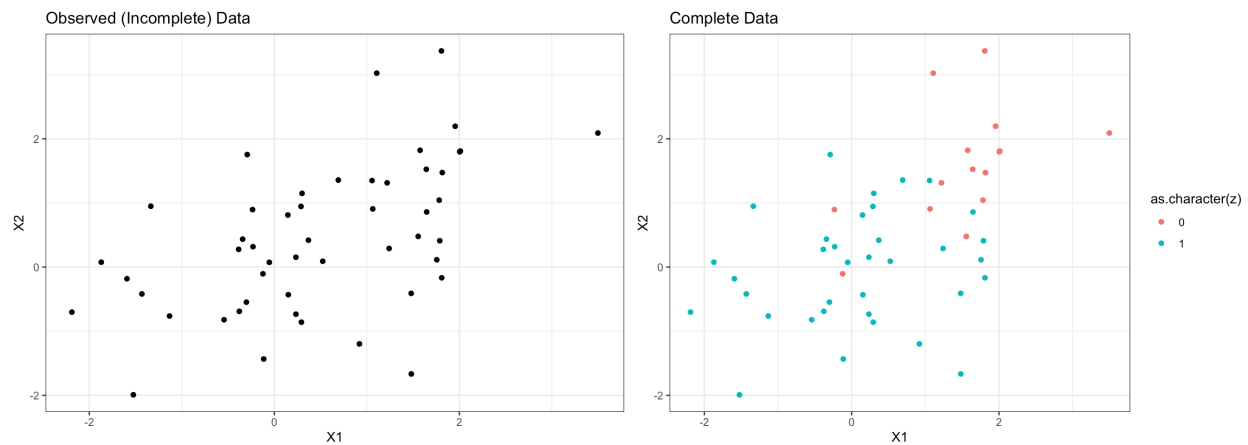
y1 <- rmvnorm(sum(z), mean = mu1, sigma = sig1)
y2 <- rmvnorm(n - sum(z), mean = mu2, sigma = sig2)
y <- matrix(NA, nrow = n, ncol = 2) ## observed data
y[z == 1, ] <- y1
y[z == 0, ] <- y2

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)

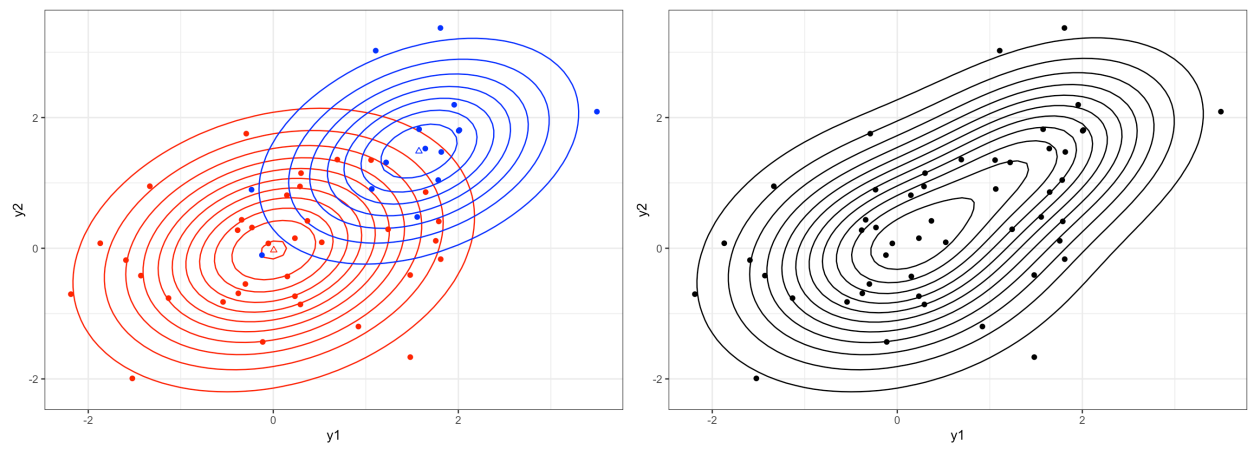
  out <- log(p * dmvnorm(data, mean = mu1, sigma = sig1) +
(1-p) * dmvnorm(data, mean = mu2, sigma =
sig2))
  return(sum(out))
}

## 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

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

Parameter	Truth	MLE3
Σ_{212}	0.6	0.0000

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

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.

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 \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:

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

Assume we observe $\mathbf{y} = (y_1, \dots, y_n)$, 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 $\Phi\left(\frac{x_1+x_2}{2}\right) \leq \frac{1}{2}\Phi(x_1) + \frac{1}{2}\Phi(x_2)$. Then

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

where g is a real-valued integrable function.

Step 2: Find a θ^{k+1} that will optimize Q .

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

The EM algorithm allows us to obtain $\hat{\theta}_{EM}$, 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**

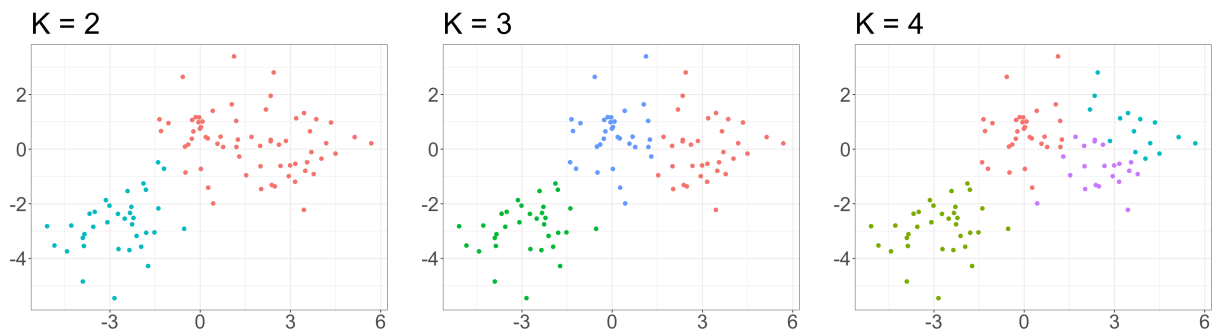
2. Louis's Method

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, \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,

2 Profile Likelihood

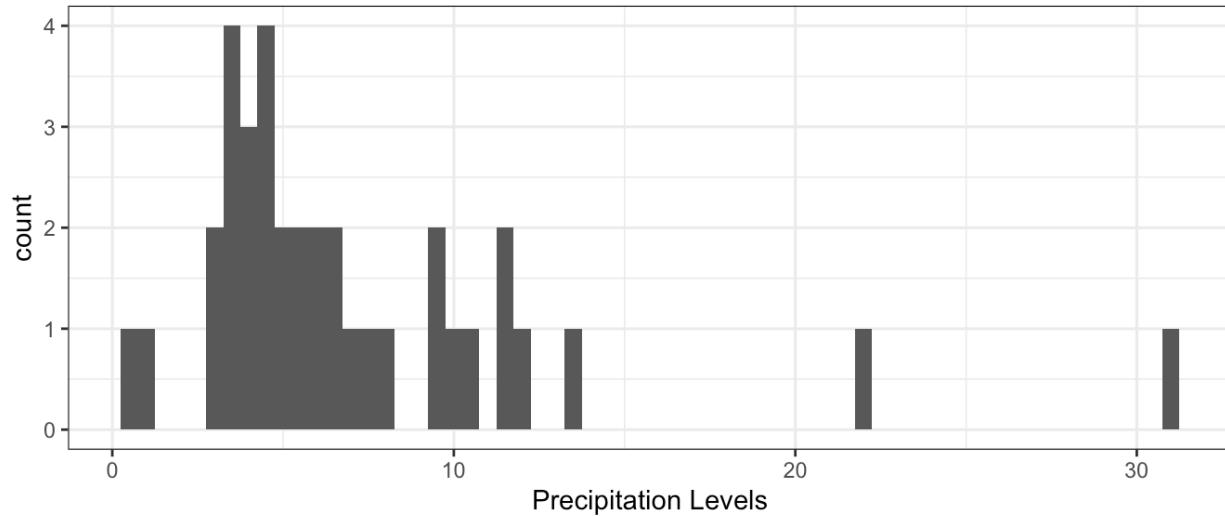
The term “profile likelihood” can mean multiple things.

2.1 Analytical Methods via Profile Likelihoods

In certain problems it is possible to maximize the log likelihood for part of $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^\top, \boldsymbol{\theta}_2^\top)^\top$ without actually knowing the value of the other part.

The **profile likelihood** is the usual likelihood with the known function of part of the parameter vector inserted for that parameter, making the likelihood only a function of one part of the vector.

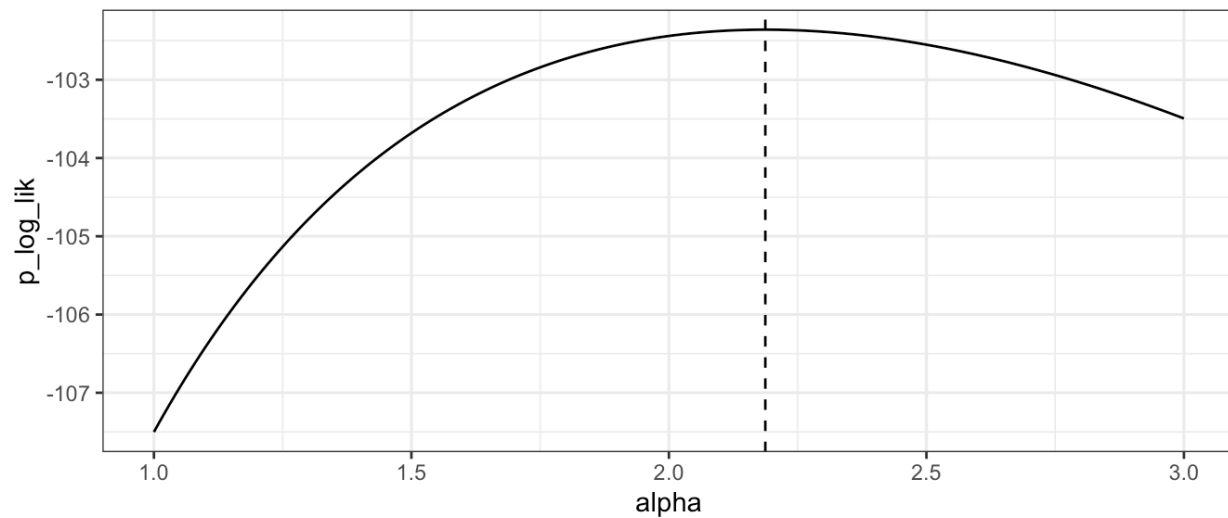
Example (Hurricane Data, Cont'd): For 36 hurricanes that had moved far inland on the East Coast of the US in 1900-1969, maximum 24-hour precipitation levels during the time they were over mountains.



We modeled the precipitation levels with a gamma distribution, which has log likelihood

$$\ell(\alpha, \beta) = -n \log \Gamma(\alpha) - n\alpha \log \beta + (\alpha - 1) \sum \log Y_i - \frac{\sum Y_i}{\beta}$$

```
gamma_prof_loglik <- function(alpha, data) {  
  beta <- mean(data) / alpha  
  sum(dgamma(data, alpha, scale = beta, log = TRUE))  
}  
  
## get maximum profile likelihood estimate  
alpha_mple <- optim(1, gamma_prof_loglik, data = hurr_rain,  
method = "BFGS", control = list(fnscale = -1))  
  
## plot profile likelihood  
data.frame(alpha = seq(1, 3, length.out = 200)) |>  
  rowwise() |>  
  mutate(p_log_lik = gamma_prof_loglik(alpha, hurr_rain)) |>  
  ggplot() +  
  geom_line(aes(alpha, p_log_lik)) +  
  geom_vline(aes(xintercept = alpha_mple$par), lty = 2)
```



2.2 Numerical Methods via Profile Likelihoods

The log likelihood can be maximized over one portion of the partition $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^\top, \boldsymbol{\theta}_2^\top)^\top$ for any fixed value of the other, even if that maximization cannot be expressed as an explicit function.

We can define a profile likelihood as

$$L^p(\boldsymbol{\theta}_2) = \max_{\boldsymbol{\theta}_1} L(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2).$$

The profile likelihood and log profile likelihood behave in many ways like true likelihood functions:

1. The estimate of $\boldsymbol{\theta}_2$ found by maximizing $L^p(\boldsymbol{\theta}_2)$ is the MLE of $\boldsymbol{\theta}_2$.
2. A likelihood ratio test statistics formed with the profile likelihood has a limiting χ^2 distribution.
3. A profile likelihood confidence region is a valid approximate confidence region for $\boldsymbol{\theta}_2$.

Where does this confidence region come from?

However, these are *not* full likelihood functions.