UNIVERSITY OF
COPENHAGEN

A Deeper Look into the EM Algorithm

Computational Statistics

Johan Larsson

Department of Mathematical Sciences, University of Copenhagen

March 31, 2026

Last Time

Likelihood Optimization

We introduced the Peppered Moths problem and solved the full likelihood using general-purpose and constrained optimization.

We talked about the barrier methods for constrained optimization.

The EM Algorithm

We introduced the EM algorithm as a general algorithm for likelihood optimization when the likelihood is complicated.

Recap: The EM Algorithm

The EM algorithm iteratively optimizes

$$\theta^{(t+1)} = \arg\min_{\theta} Q(\theta \mid \theta^{(t)}),$$

with

$$Q(\theta \mid \theta^{(t)}) = \mathrm{E}_{Z|X,\theta^{(t)}} \left(\log p(X, Z \mid \theta) \right).$$

The algorithm monotonically increases the observed data log-likelihood at each step.

Today

More on the EM Algorithm
We tackle the Peppered Moths example with the EM algorithm.

Fisher Information
After we have optimized the likelihood, we often want to estimate variance of estimates. For this we need the Fisher information.

Gaussian Mixture
We tackle an example of a mixture of Gaussians—the prototypical EM example.

Recap: The Peppered Moths

Alleles
C, I, T with frequencies p_C, p_I, p_T and

$$p_C + p_I + p_T = 1.$$

Genotypes
CC, CI, CT, II, IT, TT

Phenotypes
Black, Mottled, Light-Colored

Table 1: Genotype to Phenotype Map

	C	I	T
C	Black	Black	Black
I	Black	Mottled	Mottled
T	Black	Mottled	Light-Colored

Genotype Labels

Let each moth $i = 1, \dots, n$ have an (unobserved) genotype label

$$Z_i \in \{1, 2, 3, 4, 5, 6\} \quad \text{(CC, CI, CT, II, IT, TT)}.$$

Via the Hardy–Weinberg principle, the **genotype** probabilities are

$$p = (p_C^2,\ 2p_C p_I,\ 2p_C p_T,\ p_I^2,\ 2p_I p_T,\ p_T^2).$$

Genotype Counts

The genotype counts Y_k are the sufficient statistics for the complete data, and also latent, with

$$Y_k = \sum_{i=1}^{n} \mathbf{1}(Z_i = k), \quad Y = (Y_1, \dots, Y_6), \quad \sum_k Y_k = n.$$

Observed Data

What we actually observe are the **phenotype** counts

$$X_j = \sum_{k \in A_j} Y_k, \quad X = M(Y).$$

with partitions

$$A_1 = \{1, 2, 3\} \text{ (Black)}, \quad A_2 = \{4, 5\} \text{ (Mottled)}, \quad A_3 = \{6\} \text{ (Light)},$$

Here, $M : \mathbb{N}_0^6 \to \mathbb{N}_0^3$ is the cell-collapsing map

$$M(y) = (y_1 + y_2 + y_3,\ y_4 + y_5,\ y_6).$$

Multinomial Distribution
If $Y \sim \text{Mult}(p, n)$ with $p = (p_1, \dots, p_K)$ then

$$X = M(Y) \sim \text{Mult}(M(p), n).$$

Complete-Data Log-Likelihood

Using the genotype counts Y, the complete-data log-likelihood is

$$\ell(p \mid Y) = \sum_{k=1}^{6} Y_k \log p_k.$$

Optimization Variable

For our EM algorithm, we will optimize over the **allele frequencies**

$$\theta = (p_C, p_I), \qquad p_T = 1 - p_C - p_I.$$

Let's define the function π, that maps θ to the genotype probabilities

$$\pi(\theta) = \begin{bmatrix} p_C^2 \\ 2p_C p_I \\ 2p_C(1 - p_C - p_I) \\ p_I^2 \\ 2p_I(1 - p_C - p_I) \\ (1 - p_C - p_I)^2 \end{bmatrix}$$

The Q Function

EM Q-function:

$$\begin{aligned} Q(\theta \mid \theta') &= \mathrm{E}_{Y|X,\theta'} \log p(Y \mid \theta) \\ &= \sum_{k=1}^{6} \mathrm{E}_{Y|X,\theta'}(Y_k) \log \pi_k(\theta) \\ &= \sum_{k=1}^{6} \frac{X_{j(k)} \pi_k(\theta')}{M(\pi(\theta'))_{j(k)}} \log \pi_k(\theta) \quad \text{where } k \in A_{j(k)}. \end{aligned}$$

Note that

$$\mathrm{E}_{Y|X,\theta'}(Y_k) = \frac{X_{j(k)} \pi_k(\theta')}{M(\pi(\theta'))_{j(k)}}$$

since $Y \mid X, \theta' \sim \mathrm{Mult}\left(X_{j(k)}, \left(\frac{\pi_k(\theta')}{M(\pi(\theta'))_{j(k)}} \right)_{k \in A_{j(k)}} \right)$.

E-step for Multinomial Model

```r
e_step_mult <- function(theta, x, group) {
  p <- pi_map(theta)
  x[group] * p / M(p, group)[group]
}
```

```r
pi_map <- function(p) {
  p[3] <- 1 - p[1] - p[2]
  c(
    p[1]^2,
    2 * p[1] * p[2],
    2 * p[1] * p[3],
    p[2]^2,
    2 * p[2] * p[3],
    p[3]^2
  )
}
```

```r
M <- function(y, group) {
  as.vector(
    tapply(y, group, sum)
  )
}
```

M-Step for the Multinomial Model

We need to maximize $Q(\theta \mid \theta')$ w.r.t. θ.

With $y = (n_{CC}, n_{CI}, n_{CT}, n_{II}, n_{IT}, n_{TT})^T$ a complete observation, it can be shown that the MLE is

$$\hat{p}_C = \frac{n_{CC} + (n_{CI} + n_{CT})/2}{n}$$

$$\hat{p}_I = \frac{n_{II} + (n_{CI} + n_{IT})/2}{n}$$

where $n = \mathbf{1}^T y$ is the total number of observations.

Here's an implementation of this.

```
m_step_mult <- function(y) {
  n <- sum(y)
  pC <- (y[1] + (y[2] + y[3]) / 2) / n
  pI <- (y[4] + (y[2] + y[5]) / 2) / n
  c(pC, pI)
}
```

EM Step Function Factory

A **function factory** returns a function. Here, we create a function that performs one EM step for the multinomial model.

```
create_em_mult_step <- function(x, group) {
  force(x)
  force(group)

  e_step <- function(theta) e_step_mult(theta, x, group)
  m_step <- function(y) m_step_mult(y)

  function(par) m_step(e_step(par))
}
```

We use `force()` to ensure that `x` and `group` are evaluated when the factory is called, not when the returned function is called.

EM Algorithm Implementation

```r
em <- function(par, em_step, eps = 1e-6, maxit = 20, cb = NULL) {
  for (i in seq_len(maxit)) {
    par0 <- par
    par <- em_step(par)

    if (!is.null(cb)) {
      cb() # Callback function for tracing, to be explained
    }

    if (sum((par - par0)^2) <= eps * (sum(par^2) + eps)) {
      break
    }
  }
  par
}
```

Peppered Moths EM Algorithm

First, we create the EM step function.

```
em_mult_step <- create_em_mult_step(
  x = c(85, 196, 341),
  group = c(1, 1, 1, 2, 2, 3)
)
```

Then we can run the EM algorithm.

```
em(c(0.3, 0.3), em_mult_step)
```

```
[1] 0.07084 0.18877
```

This corresponds to the MLE found via `optim()`.

```
moth_optim$par
```

```
[1] 0.07085 0.18872
```

Inside the EM Algorithm

The `tracer()` function in the `CSwR` package can be used to trace the iterations of an algorithm.[1]

```
library(CSwR)
em_tracer <- tracer("par", Delta = 0)
```

Each call to `tracer()` returns a S3 object of class tracer. The object has a field `tracer`, which is a function that will record the current value of given parameters each time it is called.

```
em(c(0.3, 0.3), em_mult_step, cb = em_tracer$tracer)
```

```
[1] 0.07084 0.18877
```

[1] Use `Delta` to control verbosity. 0: no output.

You can then summarize the collected values using `summary()`.

```
summary(em_tracer)
```

```
    par.1  par.2     .time
1 0.08039 0.2246 0.0000000
2 0.07119 0.1955 0.0001373
3 0.07085 0.1899 0.0002465
4 0.07084 0.1889 0.0003760
5 0.07084 0.1888 0.0005167
```

Tracing the EM Algorithm

```
em_tracer <- tracer(c("par0", "par"), Delta = 0)
p_hat <- em(
  c(0.3, 0.3),
  em_mult_step,
  eps = 0,
  cb = em_tracer$tracer
)
p_hat
```

```
[1] 0.07084 0.18874
```

```
em_trace <- summary(em_tracer)
```

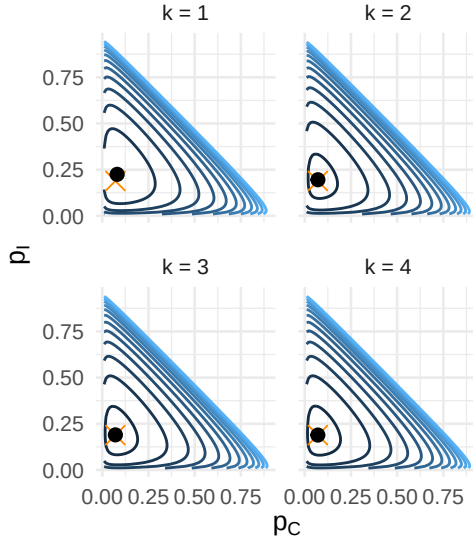

Figure 1: EM algorithm iterations for the Peppered Moths example. The orange cross marks the MLE. The surfaces are the current negative observed log-likelihoods, with the black dot indicating the EM step.

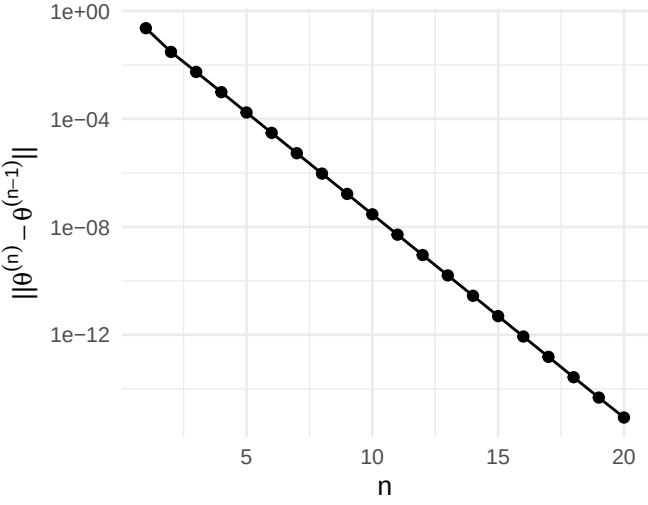

Figure 2: Parameter convergence of the EM algorithm applied to the Peppered moths problem.

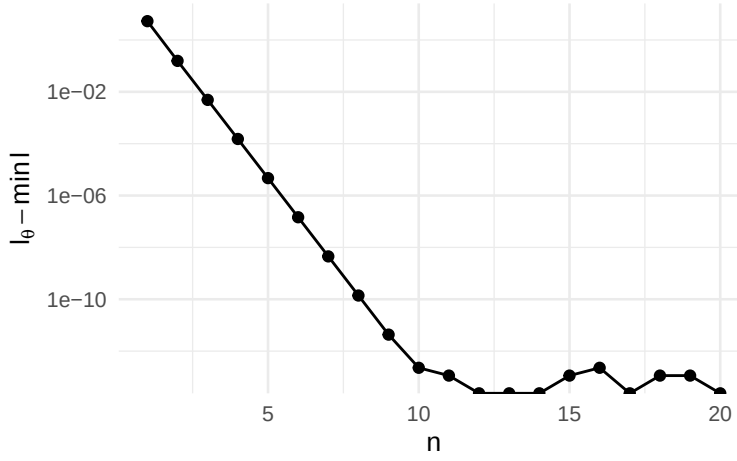

Figure 3: Log-likelihood convergence

Multi-Start

A common problem in likelihood optimization is local maxima.

We can solve this using a grid of starting values.[2]

```
theta_grid <- expand.grid(
  pC = seq(0.01, 0.49, length.out = 5),
  pI = seq(0.01, 0.49, length.out = 5)
)

res <- apply(theta_grid, 1, em, em_step = em_mult_step)
neg_logliks <- apply(res, 2, neg_noglik_obs)

which.min(neg_logliks)
```

```
[1] 9
```

[2] `neg_loglik_obs()` computes the negative observed log-likelihood.

Fisher Information

The Fisher Information

When doing statistical inference, we need to measure the uncertainty of estimates.

The Fisher information is a key quantity for this. It is defined as

$$i(\theta) = -\,\mathrm{E}\left(\frac{\partial^2}{\partial\theta^2}\log p(X \mid \theta)\right).$$

This is the expected curvature of the log-likelihood.

Complete, Observed, and Missing Information

In the presence of latent variables, we can define three types of Fisher information:

- **Complete-data information** $i_Y(\theta)$
- **Observed-data information** $i_X(\theta)$
- **Missing information** $i_{Y|X}(\theta) = i_Y(\theta) - i_X(\theta)$

Empirical Fisher Information

The empirical observed Fisher information is defined as

$$\hat{i}(\theta) = -\frac{\partial^2}{\partial \theta^2} \log p(X \mid \theta)$$

where $\ell_i(\theta)$ is the log-likelihood of observation i.

In the EM context, we typically do not have access to the individual log-likelihood contributions $\ell_i(\theta)$, but we can solve this through other means.

Three Ways of Computing the Observed Fisher Information

1. The **gradient identity**: For the score function from the Q function, then differentiate.
2. The **information identity**: Estimate complete and missing information, then subtract.
3. The **EM-mapping**: Differentiate the EM-mapping, which is a fixed point at the MLE.

First Way: Via the Gradient Identity

Recall that the Q function is

$$Q(\theta \mid \theta') = \mathrm{E}_{Y|X,\theta'} \log p(X, Y \mid \theta).$$

At convergence, $\theta' = \hat{\theta}$ (the MLE of the observed data log-likelihood).

If we differentiate once, we have

$$\nabla_\theta Q(\theta \mid \theta') = \mathrm{E}_{Y|X,\theta'} \nabla_\theta \log p(X, Y \mid \theta).$$

Setting $\theta' = \theta$ gives Fisher's (score) identity

$$\nabla_\theta Q(\theta \mid \theta) = \mathrm{E}_{Y|X,\theta} \nabla_\theta \log p(X, Y \mid \theta) = \nabla_\theta \log p(X \mid \theta).$$

Gradient of Q for the Multinomial Model

For the multinomial model, the Q function is

$$Q(\theta \mid \theta') = \sum_i \frac{x_{j(i)} \pi_i(\theta')}{M(\pi(\theta'))_{j(i)}} \log \pi_i(\theta),$$

where $j(i)$ is defined by $i \in A_{j(i)}$.

The gradient is

$$\nabla_\theta Q(\theta \mid \theta') = \sum_i \frac{x_{j(i)} \pi_i(\theta')}{M(\pi(\theta'))_{j(i)} \pi_i(\theta)} \nabla \pi_i(\theta).$$

When evaluated in θ it is

$$\nabla_\theta Q(\theta \mid \theta) = \sum_i \frac{x_{j(i)}}{M(\pi(\theta))_{j(i)}} \nabla \pi_i(\theta).$$

Numerical Validation of the Gradient Identity

```
theta_hat <- em(c(0.3, 0.3), em_mult_step, eps = 1e-20)
```

```
theta_hat
```

```
[1] 0.07084 0.18874
```

```
score(theta_hat)
```

```
             [,1]         [,2]
[1,] -1.456e-09 -7.168e-09
```

Numerical Fisher information

We could differentiate analytically, but a simpler way is to use numerical differentiation.

`stats::optimHess()`

```
-optimHess(
  theta_hat,
  neg_noglik_obs,
  score
)
```

```
      [,1] [,2]
[1,] 18491 1385
[2,]  1385 6817
```

`numDeriv::jacobian()`

```
library(numDeriv)
ihat <- -jacobian(
  score,
  theta_hat
)
ihat
```

```
      [,1] [,2]
[1,] 18488 1385
[2,]  1385 6817
```

Second Way: The Information Identity

Recall that with $\theta' = \theta$,

$$\nabla_\theta Q(\theta \mid \theta) = \nabla_\theta \log p(X \mid \theta).$$

Differentiating again (w.r.t to θ) gives

$$-\nabla^2_\theta Q(\theta \mid \theta') = -\operatorname{E}_{Y|X,\theta'} \nabla^2_\theta \log p(X, Y \mid \theta).$$

At $\theta' = \theta$, using $\log p(X, Y \mid \theta) = \log p(X \mid \theta) + \log p(Y \mid X, \theta)$, we have

$$-\nabla^2_\theta Q(\theta \mid \theta) = -\nabla^2_\theta \log p(X \mid \theta) - \operatorname{E}_{Y|X,\theta} \nabla^2_\theta \log p(Y \mid X, \theta).$$

Taking expectation also over X recovers the decomposition

$$i_Y(\theta) = i_X(\theta) + i_{Y|X}(\theta),$$

with

$$\begin{aligned} i_Y(\theta) &= -\operatorname{E}_{X,Y|\theta} \nabla^2_\theta \log p(X, Y \mid \theta), \\ i_X(\theta) &= -\operatorname{E}_{X|\theta} \nabla^2_\theta \log p(X \mid \theta), \\ i_{Y|X}(\theta) &= -\operatorname{E}_{X|\theta} \operatorname{E}_{Y|X,\theta} \nabla^2_\theta \log p(Y \mid X, \theta). \end{aligned}$$

The Information Identity

At the MLE $\hat{\theta}$,

$$\hat{i}_X = -\nabla^2_\theta \ell(\hat{\theta}) = \underbrace{-\nabla^2_\theta Q(\hat{\theta} \mid \hat{\theta})}_{\hat{i}_Y} - \underbrace{\nabla^2_\theta H(\hat{\theta} \mid \hat{\theta})}_{\hat{i}_{Y|X}}.$$

Thus we have the **information identity** (Louis identity):

$$\hat{i}_X = \hat{i}_Y - \hat{i}_{Y|X}.$$

Interpretation

- $\hat{i}_Y$ is the Fisher information for complete Y.
- $\hat{i}_{Y|X}$ is the information "lost" from not observing full X.

The Complete-Data Information

We can compute the complete-data information directly from the Q function, but how do we compute the missing information?

The second **Bartlett identity** can be used to show that

$$\nabla_\theta \nabla_\theta H(\hat{\theta} \mid \hat{\theta}) = -\nabla_{\theta'} \nabla_\theta H(\hat{\theta} \mid \hat{\theta}).$$

And since we also have $Q = \ell - H$ and because $\ell(\theta)$ does not depend on θ', we get

$$\nabla_{\theta'} \nabla_\theta Q(\hat{\theta} \mid \hat{\theta}) = -\nabla_{\theta'} \nabla_\theta H(\hat{\theta} \mid \hat{\theta}) = \nabla^2_\theta H(\hat{\theta} \mid \hat{\theta}).$$

Thus

$$\hat{\hat{\imath}}_{Y|X} = \nabla_{\theta'} \nabla_\theta Q(\hat{\theta} \mid \hat{\theta}).$$

Implementations of Q

First we implement the map Q as an R function.

```
Q <- function(theta, theta0, x = c(85, 196, 341)) {
  theta[3] <- 1 - theta[1] - theta[2]
  theta0[3] <- 1 - theta0[1] - theta0[2]
  group <- c(1, 1, 1, 2, 2, 3)
  (x[group] * pi_map(theta0) / M(pi_map(theta0), group)[group]) %*%
    log(pi_map(theta))
}
```

Implementation of Gradient of Q

Next, we implement the gradient for Q:

```
Q_grad <- function(theta, theta0, x = c(85, 196, 341)) {
  theta[3] <- 1 - theta[1] - theta[2]
  group <- c(1, 1, 1, 2, 2, 3)
  (x[group] *
    pi_map(theta0) /
    (M(pi_map(theta0), group)[group] * pi_map(theta))) %*%
    pi_grad(theta)
}
```

`pi_grad()` is the gradient of the genotype probabilities w.r.t. allele frequencies (defined in source code).

Numerical Differentiation of Q

```
iY <- -hessian(Q, theta_hat, theta0 = theta_hat)
iY
```

```
      [,1] [,2]
[1,] 19242 1680
[2,]  1680 8271
```

```
iYX <- jacobian(function(theta) Q_grad(theta_hat, theta), theta_hat)
iX <- iY - iYX
```

Define $\Phi : \Theta \mapsto \Theta$ by

$$\Phi(\theta') = \arg\max_{\theta} Q(\theta \mid \theta').$$

A global maximum of the likelihood is a fixed point of Φ, $\Phi(\hat{\theta}) = \hat{\theta}$.

Since the limit of the EM algorithm, $\hat{\theta}$, is a fixed point and other identities above, it can be shown that

$$\nabla_{\theta}\Phi(\hat{\theta})^T = \hat{i}_{Y|X}\left(\hat{i}_Y\right)^{-1}.$$

Hence

$$\hat{i}_X = \left(I - \hat{i}_{Y|X}\left(\hat{i}_Y\right)^{-1}\right)\hat{i}_Y = \left(I - \nabla_{\theta}\Phi(\hat{\theta})^T\right)\hat{i}_Y.$$

EM Step Fixed Point

```
Phi <- create_em_mult_step(c(85, 196, 341), c(1, 1, 1, 2, 2, 3))
theta_hat
```

```
[1] 0.07084 0.18874
```

Let's check if this is fixed point.

```
test_that("Fixed point", {
  expect_equal(Phi(theta_hat), theta_hat, tol = 1e-8)
})
```

```
Test passed
```

Differentiating the EM Map

```
DPhi <- jacobian(Phi, theta_hat)
iX <- (diag(2) - t(DPhi)) %*% iY
iX
```

```
      [,1] [,2]
[1,] 18488 1385
[2,]  1385 6817
```

Let's check that all of our methods agree.

```
test_that("Methods 1, 2, and 3 are equivalent", {
  expect_equal(iX, ihat, tol = 1e-6)
  expect_equal(iX, iY - iYX, tol = 1e-6)
})
```

```
Test passed
```

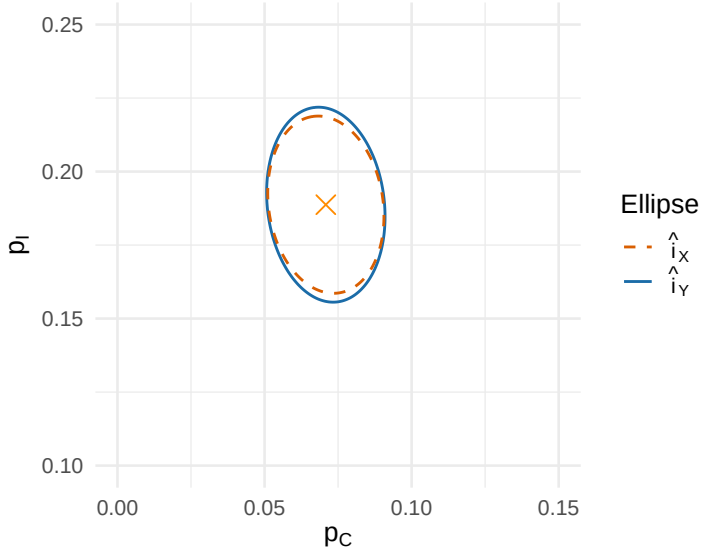

Figure 4: 95% Fisher information ellipses for the observed data (solid) and complete data (dashed) for the Peppered moths example. The MLE is indicated by the cross.

Convergence of the EM Algorithm

We have already shown that the EM algorithm monotonically increases the observed data log-likelihood:

$$\ell(\theta^{(t+1)}) \geq \ell(\theta^{(t)}).$$

We have also just shown that the EM algorithm converges to a fixed point of the EM mapping:

$$\theta^{(t+1)} = \Phi(\theta^{(t)}), \quad \hat{\theta} = \Phi(\hat{\theta}).$$

By the Taylor expansion of Φ around $\hat{\theta}$, we have

$$\theta^{(t+1)} - \hat{\theta} \approx \nabla\Phi(\hat{\theta})(\theta^{(t)} - \hat{\theta})$$

So the error at step $t+1$ is approximately a linear transformation of the error at step t.

And the rate of convergence is determined by the matrix $\nabla\Phi(\hat{\theta})$.

Convergence: Connection to Fisher Information

Let $\rho(A)$ be the spectral radius of a matrix A, i.e., the largest absolute value of its eigenvalues.

This means that the convergence rate of the EM algorithm is governed by

$$\rho(\nabla \Phi(\hat{\theta})) = \rho\left(\hat{i}_{Y|X}\left(\hat{i}_{Y}\right)^{-1}\right).$$

If $\rho(\nabla \Phi(\hat{\theta}))$ is close to 0, the EM algorithm converges quickly, and if it is close to 1, the EM algorithm converges slowly.

Relation to Missing Information

If there is little missing information, i.e., $\hat{i}_{Y|X}$ is small, then $\rho(\nabla \Phi(\hat{\theta}))$ is small and the EM algorithm converges quickly.

Convergence Estimation

The EM algorithm converges linearly near the optimum:

$$\|\theta^{(t+1)} - \hat{\theta}\| \approx \rho \|\theta^{(t)} - \hat{\theta}\|, \qquad 0 < \rho < 1.$$

Rate can be estimated by fitting a linear model to the log of the parameter differences:

$$\log\|\theta^{(t+1)} - \hat{\theta}\| \approx \log \rho + \log\|\theta^{(t)} - \hat{\theta}\| \implies \log\|\theta^{(t)} - \hat{\theta}\| \approx t \log \rho + \text{constant}.$$

```
log_lm <- lm(log(par_norm_diff) ~ n, data = em_trace)
exp(coefficients(log_lm)["n"]) # n: iteration (t above)
```

```
     n
0.1755
```

It is small in this case, implying fast convergence.

The Generalized EM Algorithm

Key Idea

Relax the M-step: instead of fully maximizing $Q(\theta \mid \theta^{(t)})$, we just require an **increase**:

$$Q(\theta^{(t+1)} \mid \theta^{(t)}) \geq Q(\theta^{(t)} \mid \theta^{(t)}).$$

Full maximization may be **hard or computationally expensive**. GEM allows **simpler updates** (e.g., gradient step, coordinate-wise update).

Properties

E-step is unchanged and the **log-likelihood is still non-decreasing**:

$$\ell(\theta^{(t+1)}) \geq \ell(\theta^{(t)}).$$

Still converges to a **stationary point** of the likelihood.

Trade-Off

We trade slower convergence for faster iterations.

Exercise

Implement the EM algorithm for a missing data problem:

```
x <- c(1, 0, 1, NA, 0, NA, 1, 1, 0)
```

Assume that $X_i \sim \text{Bernoulli}(\theta)$ i.i.d. and that the data are missing at random. The complete data log-likelihood is

$$\ell_C(\theta) = \sum_{i=1}^{n} \left(X_i \log(\theta) + (1 - X_i) \log(1 - \theta) \right).$$

Step 1 (Derive)

- E-step: Compute $Q(\theta \mid \theta') = \mathrm{E}_{X_{\text{miss}} \mid X_{\text{obs}}, \theta'} \left(\log p(X_{\text{obs}}, X_{\text{miss}} \mid \theta) \right)$.
- M-step: Maximize $Q(\theta \mid \theta')$ over θ.

Step 2 (Implement)

Implement the EM algorithm and run it from two different starting points. Record the path of $\theta^{(t)}$ and the log-likelihood values $\ell(\theta^{(t)})$.

Summary

We introduced the EM algorithm, using the Peppered Moths example throughout

We showed three ways to compute the Fisher information

We covered Gaussian mixtures and how to optimize over their parameters using the EM algorithm.