STAT 5361: Statistical Computing

Jun Yan

Department of Statistics
University of Connecticut

Outline I

1. Non-Stochastic Optimization
   - Some Background
   - Univariate Problems
   - Multivariate Problems

2. Combinatorial Optimization
   - Local Search
   - Simulated Annealing
   - Genetic Algorithm

3. EM and MM Algorithms
   - EM Algorithm
   - Majorization-Minimization Algorithm
     - Example: Penalized AFT model with induced smoothing

Non-stochastic Optimization

What we will learn:

- Some basics about statistical inference
- Univariate problems
    - Newton's method
    - Fisher scoring
    - Secant method
    - Fixed-point method
    - Connections of the above
- Multivariate problems
    - Newton's method
    - Newton-like methods

Background: likelihood inference

- Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be an i.i.d. sample from

$$f(\boldsymbol{x} \,|\, \boldsymbol{\theta}^*),$$

with the true parameter value $\boldsymbol{\theta}^*$ being unknown.

- The likelihood function is

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(\boldsymbol{x}_i \,|\, \boldsymbol{\theta}),$$

- The *maximum likelihood estimator* (MLE) of the parameter value is the maximizer of $L(\boldsymbol{\theta})$. MLE has the invariate property.
- Usually it is easier to work with the *log likelihood function*

$$l(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}).$$

- Typically, maximization of $l(\boldsymbol{\theta})$ is done by solving

$$l'(\boldsymbol{\theta}) = 0.$$

  - $l'(\boldsymbol{\theta})$ is called the *score function*.
  - For each possible parameter value $\boldsymbol{\theta}$, $l(\boldsymbol{\theta})$ is a random variable, because it depends on the observed values of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

- For any $\boldsymbol{\theta}$,

$$\mathbb{E}_{\boldsymbol{\theta}} \left\{ l'(\boldsymbol{\theta}) \right\} = 0,$$
$$\mathbb{E}_{\boldsymbol{\theta}} \left\{ l'(\boldsymbol{\theta}) l'(\boldsymbol{\theta})^T \right\} = -\mathbb{E}_{\boldsymbol{\theta}} \left\{ l''(\boldsymbol{\theta}) \right\}.$$

  where $\mathbb{E}_{\boldsymbol{\theta}}$ is the expectation with respect to $f(\boldsymbol{x} \mid \boldsymbol{\theta})$. The equality holds true under mild regularity conditions.

-

$$I(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}} \left\{ l'(\boldsymbol{\theta}) l'(\boldsymbol{\theta})^T \right\}$$

  is known as the *Fisher information*.

- The importance of the Fisher information $I(\boldsymbol{\theta})$ is that it sets the limit on how accurate an unbiased estimate of $\boldsymbol{\theta}$ can be.
- As $n \to \infty$, the asymptotic distribution of $\sqrt{n}(\widehat{\boldsymbol{\theta}}_{\mathrm{MLE}} - \boldsymbol{\theta}^*)$ is $N_p(\mathbf{0}, nI(\boldsymbol{\theta}^*)^{-1})$. Since θ^* is unknown, the asymptotic covariance matrix $I(\boldsymbol{\theta}^*)^{-1})$ needs to be estimated.
  - If $\dim(\boldsymbol{\theta}) = 1$, $I(\boldsymbol{\theta})$ is a nonnegative number.
  - If $\dim(\boldsymbol{\theta}) > 1$, $I(\boldsymbol{\theta})$ is a nonnegative definite matrix.
- The *observed Fisher information* is

$$-l''(\boldsymbol{\theta}).$$

The expected Fisher information $I(\boldsymbol{\theta})$ may not be easily computed. The observed Fisher information is a good approximation to $I(\boldsymbol{\theta})$ that improves as n gets bigger.

Besides MLEs, there are other likelihoods for parameter estimation. Suppose $\boldsymbol{\theta}$ has two parts: $\boldsymbol{\theta} = (\phi, \boldsymbol{\mu})$ and we are only interested in ϕ. The *profile likelihood* for ϕ is

$$L(\phi) := \max_{\boldsymbol{\mu}} L(\phi, \boldsymbol{\mu}).$$

- $\boldsymbol{\mu}$ is the nuisance parameter.
- Need to maximize $L(\phi, \boldsymbol{\mu})$ for every fixed ϕ. Also need to maximize $L(\phi)$.

A general method

Suppose $g(\boldsymbol{x})$ is a differentiable function, where

$$\boldsymbol{x} = (x_1, \ldots, x_n).$$

To find its maximum (or minimum), one method is to solve the equation

$$g'(\boldsymbol{x}) = 0$$
$$\text{where} \quad g'(\boldsymbol{x}) = \left(\frac{\partial g}{\partial x_1}, \ldots, \frac{\partial g}{\partial x_n}\right)^T.$$

Then the maximization is equivalent to solving

$$f(\boldsymbol{x}) = 0,$$

where $f = g'$.
For maximum likelihood estimation, g is the log likelihood function l, and $\boldsymbol{x}$ is the corresponding parameter vector $\boldsymbol{\theta}$.

Univariate Problems

Optimization: Univariate case

- Goal: optimize a real-valued function g with respect to its argument, a p dimensional vector $\boldsymbol{x}$. We will first consider the case $p = 1$.
- We will limit consideration mainly to smooth and differentiable functions.
- Root finding methods. Solving unconstrained nonlinear equations.
- Iterative algorithms: starting value, updating equation, stopping rule/convergence criterion.

Using bisection method to maximize

$$g(x) = \frac{\log x}{1 + x}.$$

or to sovle

$$f(x) = g'(x) = \frac{1 + \frac{1}{x} - \log x}{(1 + x)^2} = 0.$$

Newton's method

This is a fast approach to finding roots of a differentiable function $f(x)$. First, set initial value x_0. Then for $t = 0, 1, \ldots$, compute

$$x_{t+1} = x_t + h_t, \text{ with } h_t = -\frac{f(x_t)}{f'(x_t)}.$$

Continue the iterations until x_t converges.

- Also known as Newton-Raphson iteration.
- Need to specify x_0.
- If $f(x) = 0$ has multiple solutions, the end result depends on x_0.

Newton's method require computing the derivative of a function. Algorithms are available to find root(s) of a univariate function without having to compute its derivative. For example, in R, one can use `uniroot`. Newton's method can be applied to optimize g by applying to

$$f = g'.$$

- Both g' (*gradient*) and g'' (*Hessian*) are needed.
- Many variants of Newton's method avoid the computation of the Hessian, which can be difficult especially for multivariate functions.

To maximize

$$g(x) = \frac{\log x}{1+x},$$

first find

$$f(x) = g'(x) = \frac{1 + \frac{1}{x} - \log x}{(1+x)^2},$$
$$f'(x) = g''(x) = -\frac{3 + 4/x + 1/x^2 - 2\log x}{(1+x)^3}.$$

So in the Newton's method,

$$h_t = \frac{(x_t + 1)(1 + 1/x_t - \log x_t)}{3 + 4/x_t + 1/x_t^2 - 2\log x_t}.$$

Note that to solve $f(x) = 0$, one can instead solve $1 + 1/x - \log x = 0$. Treat this as a new f function. Then in the Newton's method,

$$h_t = x_t - \frac{x_t^2 \log x_t}{1 + x_t} \quad \Longrightarrow \quad x_{t+1} = 2x_t - \frac{x_t^2 \log x_t}{1 + x_t}.$$

To maximize the log likelihood $l(\theta)$, Newton's method computes

$$\theta_{t+1} = \theta_t - \frac{l'(\theta_t)}{l''(\theta_t)}$$

Example: consider $x_1, \dots, x_n \sim$ i.i.d. $N(\mu, \sigma^2)$.

Example: consider the model on shift

$$p(x \mid \theta) = p(x - \theta).$$

Given observations $x_1, \ldots, x_n$ i.i.d. $\sim p(x \mid \theta)$

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n} \log p(x_i - \theta) \\ l'(\theta) &= -\sum_{i=1}^{n} \frac{p'(x_i - \theta)}{p(x_i - \theta)} \\ l''(\theta) &= \sum_{i=1}^{n} \frac{p''(x_i - \theta)}{p(x_i - \theta)} - \sum_{i=1}^{n} \left\{ \frac{p'(x_i - \theta)}{p(x_i - \theta)} \right\}^2 . \end{aligned}$$

Note that we need to update θ in the iterations, not $x_1, \ldots, x_n$.

In R, to *minimize* a function, one can use

```
z = nlminb(x0, g, gr.g, hess.g)
```

here `x0` is the initial value, g is the function being minimized, `gr.g` its gradient and `hess.g` its Hessian. (Unconstrained and box-constrained optimization using PORT routines).

In the above function, the derivatives g' and g'' have to be analytically calculated. One can also use

```
z = nlminb(x0, g, gr.g)
```

without inputing the analytic expression of g'', or, even simpler,

```
z = nlminb(x0, g)
```

without inputing the analytic expressions of either derivatives. In these cases, numerical approximations of derivatives will be computed during the iterations.

Other functions for optimization in R include `optim`, `optimize`, `nlm` and `constrOptim`.

For profile likelihood

$$L(\phi) = \max_{\boldsymbol{\mu}} L(\phi, \boldsymbol{\mu})$$

we need to optimize $L(\phi, \boldsymbol{\mu})$ for each given ϕ.
In R, in order to get $\min_x g(x, y)$ for each fixed y, one can do the following. First, define `g(x,y)`, `gr.g(x,y)`, etc. Then call, say,

```
nlminb(x0, g, gr.g, hess.g, y=1),   or
nlminb(x0, g, gr.g, y=.3),   or
nlminb(x0, g, y=-1)
```

If one only wants minimization for $x \in [a, b]$, use,

```
nlminb(x0, ..., lower=a, upper=b)
```

Fisher scoring

This is a variant of Newton's method specific for MLE. Recall $-l''(\theta)$ is the observed Fisher information at θ. To maximize $l(\theta)$, an alternative is to replace $-l''(\theta_t)$ with $I(\theta_t)$ to get

$$\theta_{t+1} = \theta_t + \frac{l'(\theta_t)}{I(\theta_t)}.$$

To *minimize* $-l(\theta)$, one still can use

```
z = nlminb(x0, f, grf, fs)
```

where `f` is $-l(\theta)$, `grf` is $-l'(\theta)$, and `fs` is $I(\theta)$ instead of $-l''(\theta)$. Generally, use Fisher scoring in the beginning to make rapid improvements, and Newton's method for refinement near the end.

Continuing the example on $p(x \mid \theta) = p(x - \theta)$, to use Fisher scoring, we need to compute

$$I(\theta) = -\mathbb{E}_\theta[l''(\theta)].$$

We already know

$$l''(\theta) = \sum_{i=1}^n \frac{p''(x_i - \theta)}{p(x_i - \theta)} - \sum_{i=1}^n \left\{ \frac{p'(x_i - \theta)}{p(x_i - \theta)} \right\}^2 .$$

Therefore

$$I(\theta) = -n\mathbb{E}_\theta \left[\frac{p''(X - \theta)}{p(X - \theta)} - \left\{ \frac{p'(X - \theta)}{p(X - \theta)} \right\}^2 \right].$$

Since under parameter θ, X has density $p(x-\theta)$, the last $\mathbb{E}_\theta[...]$ equals

$$\begin{aligned}
&\int \left[\frac{p''(x-\theta)}{p(x-\theta)} - \left\{ \frac{p'(x-\theta)}{p(x-\theta)} \right\}^2 \right] p(x-\theta)\ \mathrm{d}x \\
&= \int p''(x-\theta)\ \mathrm{d}x - \int \frac{[p'(x-\theta)]^2}{p(x-\theta)}\ \mathrm{d}x \\
&= \frac{\mathrm{d}^2}{\mathrm{d}\theta^2} \int p(x-\theta)\ \mathrm{d}x - \int \frac{\{p'(x)\}^2}{p(x)}\ \mathrm{d}x \\
&= \frac{\mathrm{d}^2}{\mathrm{d}\theta^2} 1 - \int \frac{\{p'(x)\}^2}{p(x)}\ \mathrm{d}x = -\int \frac{\{p'(x)\}^2}{p(x)}\ \mathrm{d}x.
\end{aligned}$$

Therefore,

$$I(\theta) = n \int \frac{\{p'(x)\}^2}{p(x)}\ \mathrm{d}x.$$

So, in this case, Fisher information is a constant.

Secant method

Approximating $f'(x_t)$ by

$$\frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}},$$

the Newton's method turns into the secant method

$$x_{t+1} = x_t - \frac{f(x_t)(x_t - x_{t-1})}{f(x_t) - f(x_{t-1})}.$$

- Need to specify x_0 and x_1.

Fixed point iteration

Compute

$$x_{t+1} = x_t + \alpha f(x_t), \quad t = 0, 1, \dots,$$

where $\alpha \neq 0$ is a tuning parameter so that

$$|1 + \alpha f'(x)| \le \lambda < 1, \text{ all } x$$

or more generally,

$$|x - y + \alpha[f(x) - f(y)]| \le \lambda |x - y|, \quad \text{any } x, y$$

with $0 \le \lambda < 1$ a constant, i.e., $F(x) = x + \alpha f(x)$ is a *contraction*.
If such α exists, the speed of convergence depends on λ. The smaller λ is, the faster the convergence.

Some Details on Fixed point iteration

Definition: A fixed point of a function is a point whose evaluation by that function equals to itself, i.e., $x = G(x)$.
Fixed point iteration: the natural way of hunting for a fixed point is to use $x_{t+1} = G(x_t)$.
Definition: A function G is contractive on $[a, b]$ if

(1). $G(x) \in [a, b]$ whenever $x \in [a, b]$,
(2). $|G(x_1) - G(x_2)| \leq \lambda |x_1 - x_2|$ for all $x_1, x_2 \in [a, b]$ and some $\lambda \in [0, 1)$.

Theorem: if G is contractive on $[a, b]$, then there is a unique fixed point $x^* \in [a, b]$, and the fixed point iteration convergence to it when starting inside the interval.
Convergence:
$|x_{t+1} - x_t| = |G(x_t) - G(x_{t-1})| \leq \lambda |x_t - x_{t-1}| \leq \lambda^t |x_1 - x_0| \to 0$, as $t \to \infty$. It follows that $\{x_t\}$ convergent to a limit x^*.

Connection between fixed-point method and Newton methods

Root-finding problems using fixed point iteration: for solving $f(x) = 0$, we can simply let $G(x) = x + \alpha f(x)$, where $\alpha \neq 0$ is a constant.
Required Lipschitz condition: $|x - y + \alpha[f(x) - f(y)]| \leq \lambda|x - y|$, for some $\lambda \in [0, 1)$ and for all $x, y \in [a, b]$. This holds if $|G'(x)| \leq \lambda$ for some $\lambda \in [0, 1)$ and for all $x \in [a, b]$, i.e., $|1 + \alpha f'(x)| \leq \lambda$. (use mean value theorem.)
Newton methods: $G(x) = x - f(x)/f'(x)$. So it is as if α_t is chosen adaptively as $\alpha_t = -1/f'(x_t)$. This leads to a faster convergence order (quadratic).

Convergence Order

Define $\varepsilon_t = x_t - x^*$. A method has convergence order β if

$$\lim_{t\to\infty} \varepsilon_t = 0, \qquad \lim_{t\to\infty} \frac{|\varepsilon_{t+1}|}{|\varepsilon_t|^\beta} = c,$$

for some constant $c \neq 0$ and $\beta > 0$.

- For Newton's method, $\beta = 2$. (If it converges.)
- For Secant method, $\beta \approx 1.62$.
- For fixed point iteration, $\beta = 1$.

Convergence Order

For Newton's method: suppose f has two continuous derivatives and $f'(x^*) \neq 0$. There then exists a neighborhood of x^* within which $f'(x) \neq 0$ for all x. By Taylor expansion,

$$0 = f(x^*) = f(x_t) + f'(x_t)(x^* - x_t) + \frac{1}{2}f''(q)(x^* - x_t)^2,$$

for some q between x^* and x_t. Rearranging terms, we find that

$$\frac{\varepsilon_{t+1}}{\varepsilon_t^2} = \frac{f''(q)}{2f'(x_t)} \to \frac{f''(x^*)}{2f'(x^*)}.$$

Convergence Order

For fixed point iteration: let G be a continuous function on the closed interval $[a, b]$ with $G : [a, b] \to [a, b]$ and suppose that G' is continuous on the open interval (a, b) with $|G'(x)| \le k < 1$ for all $x \in (a, b)$. If $g'(x^*) \neq 0$, then for any $x_0 \in [a, b]$, the fixed point iteration converges linearly to the fixed point x^*. This is because

$$|x_{t+1} - x^*| = |G(x_t) - G(x^*)| = |G'(q)||x_t - x^*|,$$

for some q between x_t and x^*. This implies that

$$\frac{|\varepsilon_{t+1}|}{|\varepsilon_t|} \to |G'(x^*)|.$$

Stopping Rules

- Absolute convergence criterion:

$$\|x_{t+1} - x_t\| < \epsilon,$$

  where ϵ is a chosen tolerance.

- Relative convergence criterion

$$\frac{\|x_{t+1} - x_t\|}{\|x_t\|} < \epsilon.$$

  If x_t close to zero,

$$\frac{\|x_{t+1} - x_t\|}{\|x_t\| + \epsilon} < \epsilon.$$

- Many other rules depending on the algorithm.

Multivariate Problems

Newton's method

Let g now be a function in $\boldsymbol{x} = (x_1, \ldots, x_p)^T$.
The generalization is straightforward: to maximize $g(\boldsymbol{x})$, set

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - [g''(\boldsymbol{x}_t)]^{-1} g'(\boldsymbol{x}_t).$$

- $g''(\boldsymbol{x})$ is a $p \times p$ matrix with the (i, j)th entry equal to
$$\frac{\partial^2 g(\boldsymbol{x})}{\partial x_i \partial x_j};$$
- $g'(\boldsymbol{x})$ is a $p \times 1$ vector with the ith entry equal to
$$\frac{\partial g(\boldsymbol{x})}{\partial x_i}.$$

Newton-like methods

For MLE, the generalization is straightforward. Simply use

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + I(\boldsymbol{\theta}_t)^{-1} l'(\boldsymbol{\theta}_t).$$

These methods in each iteration approximate $g''(\boldsymbol{x}_t)$ by some $p \times p$ matrix $\boldsymbol{M}_t = \boldsymbol{M}_t(\boldsymbol{x}_t)$ to get

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \boldsymbol{M}_t^{-1} g'(\boldsymbol{x}_t). \tag{1}$$

Of course, $\boldsymbol{M}_t$ should be easier to compute than $g''(\boldsymbol{x}_t)$.
Fisher scoring is a Newton-like method, because it uses

$$\boldsymbol{M}_t = -I(\boldsymbol{\theta}_t)$$

in place of $-l''(\boldsymbol{\theta}_t)$.

Steepest ascent methods

In (1), set

$$\boldsymbol{M}_t = -\alpha_t^{-1} \boldsymbol{I}_p,$$

so that

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \alpha_t g'(\boldsymbol{x}_t),$$

where $\alpha_t > 0$ is the step size at t which can shrink to ensure ascent. If at step t, the original step turns out to be downhill, the updating can backtrack by halving α_t.

Discrete Newton and Fixed-point methods

In fixed-point methods, usually $\boldsymbol{M}_t$ is fixed to be $\boldsymbol{M}$, e.g., $g''(\boldsymbol{x}_0)$. This amounts to applying univariate scaled fixed-point iteration to each component.

In discrete Newton methods, $g''(\boldsymbol{x}_t)$ is approximated as follows. Compute matrix $\boldsymbol{M}_t$ with the $(i,j)^{\text{th}}$ entry equal to

$$\boldsymbol{M}_t(i,j) = \frac{g_i'(\boldsymbol{x}_t + h_t(i,j)\boldsymbol{e}_j) - g_i'(\boldsymbol{x}_t)}{h_t(i,j)}$$

for some constant $h_t(i,j) \neq 0$, where

$$g_i'(\boldsymbol{x}) = \frac{\partial g(\boldsymbol{x})}{\partial x_i}.$$

If $h_t(i,j)$ is small, then

$$\boldsymbol{M}_t(i,j) \approx \frac{\partial^2 g(\boldsymbol{x}_t)}{\partial x_i \partial x_j}$$

and so $\boldsymbol{M}_t$ approximates $g''(\boldsymbol{x}_t)$.

However, since $g''(\boldsymbol{x}_t)$ is symmetric, instead of using $\boldsymbol{M}_t$ directly, use the symmetric

$$(\boldsymbol{M}_t + \boldsymbol{M}_t^T)/2$$

as the approximation of $g''(\boldsymbol{x}_t)$.

- No need to calculate second order derivatives
- Inefficient: all p^2 entries have to be updated each time.

Quasi-Newton methods

These methods aim to achieve the following goals.

- Each step satisfies the secant condition

$$g'(\boldsymbol{x}_{t+1}) - g'(\boldsymbol{x}_t) = \boldsymbol{M}_{t+1}(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t).$$

- No need to compute second order derivatives.
- Maintain symmetry of $\boldsymbol{M}_t$.
- Aim to update $\boldsymbol{M}_t$ efficiently.

There is a unique rank-one method that satisfies the secant condition and maintains the symmetry of $\boldsymbol{M}_t$: after getting

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \boldsymbol{M}_t^{-1} g'(\boldsymbol{x}_t)$$

compute

$$\begin{aligned} \boldsymbol{z}_t &= \boldsymbol{x}_{t+1} - \boldsymbol{x}_t, \quad \boldsymbol{y}_t = g'(\boldsymbol{x}_{t+1}) - g'(\boldsymbol{x}_t), \\ \boldsymbol{v}_t &= \boldsymbol{y}_t - \boldsymbol{M}_t \boldsymbol{z}_t, \quad c_t = \frac{1}{\boldsymbol{v}_t^T \boldsymbol{z}_t}. \end{aligned}$$

Then update

$$\boldsymbol{M}_{t+1} = \boldsymbol{M}_t + c_t \boldsymbol{v}_t \boldsymbol{v}_t^T.$$

Note $\boldsymbol{M}_{t+1} - \boldsymbol{M}_t$ is of rank one. We can verify the secant condition is satisfied by multiplying both sides by $\boldsymbol{z}_t$.

There are several rank-two method satisfying the secant condition and the symmetry requirement. The Broyden class updates $\boldsymbol{M}_t$ as follows. After getting $\boldsymbol{x}_{t+1}$ and $\boldsymbol{z}_t$, $\boldsymbol{y}_t$ as above, compute

$$\boldsymbol{d}_t = \frac{\boldsymbol{y}_t}{\boldsymbol{z}_t^T \boldsymbol{y}_t} - \frac{\boldsymbol{M}_t \boldsymbol{z}_t}{\boldsymbol{z}_t^T \boldsymbol{M}_t \boldsymbol{z}_t}.$$

Then update

$$\boldsymbol{M}_{t+1} = \boldsymbol{M}_t - \frac{\boldsymbol{M}_t \boldsymbol{z}_t (\boldsymbol{M}_t \boldsymbol{z}_t)^T}{\boldsymbol{z}_t^T \boldsymbol{M}_t \boldsymbol{z}_t} + \frac{\boldsymbol{y}_t \boldsymbol{y}_t^T}{\boldsymbol{z}_t^T \boldsymbol{y}_t} + \delta_t (\boldsymbol{z}_t^T \boldsymbol{M}_t \boldsymbol{z}_t) \boldsymbol{d}_t \boldsymbol{d}_t^T,$$

where δ_t is a parameter.

A popular method to solve unconstrained nonlinear optimization problems is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, with $\delta_t \equiv 0$.

In R, `optim` can be called to minimize a function, using the BFGS method or its variant, the "L-BFGS-B" method. "L" stands for limited memory, and "B" stands for box constraint.

Approximating Hessian for MLE

For MLE, the Hessian $l''(\widehat{\boldsymbol{\theta}}_{\mathrm{MLE}})$ is critical because it provides estimates of standard error and covariance.

- Quasi-Newton methods may provide poor approximation because it is based on the idea of using poor approximation of the Hessian to find the root for $l'(\boldsymbol{\theta}) = 0$.
- Use the discrete multivariate Newton method with

$$\boldsymbol{M}_t(i,j) = \frac{l_i'(\boldsymbol{\theta}_t + h_{ij}\boldsymbol{e}_j) - l_i'(\boldsymbol{\theta}_t - h_{ij}\boldsymbol{e}_j)}{2h_{ij}}.$$

Gauss-Newton method

Example: nonlinear regression. In many cases, we want to maximize

$$g(\boldsymbol{\theta}) = -\sum_{i=1}^{n}(y_i - f_i(\boldsymbol{\theta}))^2,$$

where each $f_i(\boldsymbol{\theta})$ is differentiable. Recall that for linear regression:

$$y_i = \boldsymbol{x}_i^T\boldsymbol{\theta} + \varepsilon, \quad i = 1, \dots, n$$

the LS estimator of $\boldsymbol{\theta}$ maximizes $g(\boldsymbol{\theta})$ with $f_i(\boldsymbol{\theta}) = \boldsymbol{x}_i^T\boldsymbol{\theta}$ and equals

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

where

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

The Gauss-Newton method applies a similar idea to the nonlinear case. Let $\boldsymbol{\theta}^*$ be the (unknown) maximizer of $g(\boldsymbol{\theta})$. Given any candidate $\boldsymbol{\theta}$, the function

$$h(\boldsymbol{u}) = -\sum_{i=1}^{n} (y_i - f_i(\boldsymbol{\theta} + \boldsymbol{u}))^2.$$

is maximized by $\boldsymbol{\beta} = \boldsymbol{\theta}^* - \boldsymbol{\theta}$. Of course, $\boldsymbol{\beta}$ is unknown. However, if $\boldsymbol{\theta}$ is near $\boldsymbol{\theta}^*$, then $\boldsymbol{\beta} \approx 0$, so by Taylor expansion of $h(\boldsymbol{u})$, it may be close to the maximizer of

$$-\sum_{i=1}^{n} (y_i - f_i(\boldsymbol{\theta}) - f_i'(\boldsymbol{\theta})^T \boldsymbol{u})^2.$$

Treat $y_i - f_i(\boldsymbol{\theta})$ the same way as y_i in the linear regression, and $f_i'(\boldsymbol{\theta})$ the same way as $\boldsymbol{x}_i$, then

$$\boldsymbol{\theta}^* - \boldsymbol{\theta} \approx (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{z}$$

where

$$\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{\theta}) = \begin{pmatrix} y_1 - f_1(\boldsymbol{\theta}) \\ \vdots \\ y_n - f_n(\boldsymbol{\theta}) \end{pmatrix}, \quad \boldsymbol{A} = \boldsymbol{A}(\boldsymbol{\theta}) = \begin{pmatrix} f_1'(\boldsymbol{\theta})^T \\ \vdots \\ f_n'(\boldsymbol{\theta})^T \end{pmatrix}.$$

The update rule thus is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + (\boldsymbol{A}_t^T \boldsymbol{A}_t)^{-1} \boldsymbol{A}_t^T \boldsymbol{z}_t,$$

where $\boldsymbol{z}_t = \boldsymbol{z}(\boldsymbol{\theta}_t)$ and $\boldsymbol{A}_t = \boldsymbol{A}(\boldsymbol{\theta}_t)$.
In R, the Gauss-Newton method is the default method of the function `nls`.

Practical Issues

- Initial value
- Convergence criteria: based on a distance measures for vectors
- Multiple local optima
  - Multiple initial values
  - Compare the objective function value at convergences
- Trade-off between efficiency and robustness

Combinatorial Optimization

Motivation:

- There are hard optimization problems for which most methods including what we have discussed so far are useless.
- These problems are usually combinatorial in nature. Maximization requires a discrete search of a very large space.
- Problem: maximize $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$, where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\boldsymbol{\Theta}$ consists of N elements.
  - Each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is called a candidate solution.
  - Usually N is a very large number depending on problem size p.
  - The difficulty of a particular size-p problem can be characterized by the number of operations required to solve it in the worst case scenario using the best known algorithm.
  - Suppose a problem is $\mathcal{O}(p!)$. If it requires 1 minute to solve for $p = 20$, it would take 12.1 years for $p = 25$ and 207 million years for $p = 30$.

What can we do:

- We have to take some sacrifices: we will abandon the global algorithms and focus on algorithms that can find a good local solution.
- Heuristic strategies.

What we will learn:

- Local search methods
- Simulated annealing
- Genetic algorithms

Motivating Example: Subset regression

Suppose $x_1, \dots, x_p$ are predictors that can be used to construct a linear model for Y. Each candidate model is

$$Y = \sum_{j=1}^{s} \beta_{i_j} x_{i_j} + \varepsilon,$$

where $1 \le i_1 < i_2 < \ldots < i_s \le p$, $s \ge 0$. The Akaike information criterion (AIC) for the candidate model is

$$\mathsf{AIC} = n \log \frac{\mathsf{RSS}}{n} + 2s$$

where n is the sample size and RSS is the residual sum of squares of model. The best model is the one that minimizes AIC.
To parameterize the model, set $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)$, such that $\theta_i = 1$ if x_i is included as a predictor, and 0 otherwise. The set $\boldsymbol{\Theta}$ of all possible $\boldsymbol{\theta}$ has 2^p values.

Local search

First, define a neighborhood $\mathcal{N}(\boldsymbol{\theta})$ for each $\boldsymbol{\theta}$, so that it

- contains candidate solutions that are "near" $\boldsymbol{\theta}$, and
- reduces the number of changes to the current $\boldsymbol{\theta}$.

For example, if $\boldsymbol{\Theta} = \{\boldsymbol{\theta} = (\theta_1, \dots, \theta_p) : \text{each } \theta_i = 0, 1\}$, one may define $\mathcal{N}(\boldsymbol{\theta})$ to be the set of $\boldsymbol{\theta}'$ which are different from $\boldsymbol{\theta}$ in at most one coordinate.

After neighborhoods are defined, at iteration t, choose $\boldsymbol{\theta}_{t+1}$ from $\mathcal{N}(\boldsymbol{\theta}_t)$ according to a certain rule. For example,

- steepest ascent: $\boldsymbol{\theta}_{t+1} = \arg\max_{\boldsymbol{\theta} \in \mathcal{N}(\boldsymbol{\theta}_t)} f(\boldsymbol{\theta})$;
- ascent algorithm: $\boldsymbol{\theta}_{t+1} \in \mathcal{N}(\boldsymbol{\theta}_t)$ uphill from $\boldsymbol{\theta}_t$, i.e.,

$$f(\boldsymbol{\theta}_{t+1}) \geq f(\boldsymbol{\theta}_t).$$

To avoid trapping into a local maximum, two often used variants are

- random-starts local search: repeatedly run an ascent algorithm to termination from a large number of randomly chosen starting points.
- steepest ascent/mildest descent: set to the least unfavorable $\boldsymbol{\theta}_{t+1} \in \mathcal{N}(\boldsymbol{\theta}_t)$;
    - if $\boldsymbol{\theta}_t$ is a local maximum $\boldsymbol{\theta}_{t+1}$ is the one with least decrease;
    - otherwise, $\boldsymbol{\theta}_{t+1}$ is the one with the largest increase.

To select a linear model to minimize AIC (or maximize $-$AIC),

- randomly select a set of predictors
- at iteration t, decrease the AIC by adding a predictor to the set of selected predictors or deleting a predictor that has been selected; continue the iterations until the AIC can not be decreased;
- repeat Steps 1 and 2 many times, and choose the set of predictors at termination that has the lowest AIC.

A salesman must visit each of p cities exactly once and return to his starting city, using the shortest total travel distance.
A candidate solution is

$$\boldsymbol{\theta} = (i_1, i_2, \ldots, i_p, i_1)$$

where $i_1, \ldots, i_p$ is a permutation of 1, …, p. The objective function to be minimized is

$$f(\boldsymbol{\theta}) = d(i_1, i_2) + d(i_2, i_3) + \cdots + d(i_{p-1}, i_p) + d(i_p, i_1).$$

There are $(p-1)!/2$ all possible routes, since the point of origin and direction of travel are arbitrary. For a traveling plan to visit 20 cities, that amounts to more than 6×10^{16} possible routes.

One could define $\mathcal{N}(\boldsymbol{\theta})$ as the set of sequences that only differ from $\boldsymbol{\theta}$ at two entries. In other words, each sequence in $\mathcal{N}(\boldsymbol{\theta})$ is obtained by exchanging two entries of $\boldsymbol{\theta}$.
For example, if

$$\boldsymbol{\theta} = (1, 2, 5, 4, 6, 3, 1)$$

then

$$(1, 2, 3, 4, 6, 5, 1)$$

is a neighbor of $\boldsymbol{\theta}$, because it only has 3 and 5 exchanged; while

$$(1, 2, 4, 3, 6, 5, 1)$$

is not a neighbor of $\boldsymbol{\theta}$, because it has 5, 4, and 3 rotated.

Simulated annealing

In most cases, the simulated annealing algorithm can be thought of as a randomized local search algorithm. It uses a “temperature” parameter to control the randomness of the search. The algorithm starts with a high temperature and cools down gradually so that a global optimum may be reached. This is analogous to the annealing of metal or glass.

In the following description, a global *minimum* of f is being searched. The algorithm is run in stages, such that for the iterations within a stage, the temperature is a constant τ_j.

Suppose iteration t belongs to stage j.

1. Sample a candidate $\boldsymbol{\theta}^* \in \mathcal{N}(\boldsymbol{\theta})$ according to a *proposal density* $g_t(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t)$; for different t, the proposal density g_t can be different.
2. Let $\Delta = f(\boldsymbol{\theta}^*) - f(\boldsymbol{\theta}_t)$.
   - a) If $\Delta \leq 0$, then set $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^*$.
   - b) If $\Delta > 0$, then set $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^*$ with probability $e^{-\Delta/\tau_j}$, and $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$ otherwise. This can be done as follows. Sample $U \sim$ Unif$(0,1)$. Then

$$\boldsymbol{\theta}_{t+1} = \begin{cases} \boldsymbol{\theta}^* & \text{if} \quad U \leq e^{-\Delta/\tau_j} \\ \boldsymbol{\theta}_t & \text{otherwise.} \end{cases}$$

3. Repeat steps 1 and 2 a total of m_j times.
4. Update $\tau_{j+1} = \alpha(\tau_j)$, $m_{j+1} = \beta(m_j)$ and move to stage $j+1$, where α and β are two deterministic functions that govern how to cool down the temperature and how long to stay at a given temperature.

Suppose I is an image consisting of "pixels" $I(i,j)$, $i,j = 1, \dots, N$. The image is corrupted by noise $Z(i,j)$ and only $J = I + Z$ is observed. To reconstruct I, one way is to minimize a function

$$f(I) = \sum_{i,j} |J(i,j) - I(i,j)|^2 + K(I),$$

where $K(I)$ is a function that has large values if I has many "irregularities". The idea is that, as long as the noise is not too strong, the real image should be similar to J; on the other hand, irregularities observed in J are likely to be due to noise and should be reduced.
One way to use the simulated annealing to minimize $f(I)$ is as follows. In each iteration, only one pixel of I can be updated. That means two I and I' are neighbors only when they are different at just one pixel. Then at each iteration, choose a pixel $I(i,j)$ and select a candidate value for it. Update $I(i,j)$ according to the rule in step 2 and move to the next iteration.

Some guidelines:

- R function

  ```
  optim
  ```

  can implement some simple versions of simulated annealing;
- the temperature τ_j should slowly decrease to 0;
- the number m_j of iterations at each temperature τ_j should be large and increasing in j;
- reheating strategies that allow sporadic, systematic, or interactive temperature increases to prevent getting stuck in a local minimum at low temperatures can be effective.

Genetic algorithms

First, each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is a string of alphabets:

$$\boldsymbol{\theta} = (\theta_1, \dots, \theta_C),$$

where each θ_i is a symbol from a finite set, such as $\{0, 1\}$, $\{0, 1, 2\}$, $\{$'a', 'b', ...$\}$.

Genetic algorithms regard the maximization of $f(\boldsymbol{\theta})$ over $\boldsymbol{\Theta}$ as a process of natural selection, with $f(\boldsymbol{\theta})$ being a measure of fitness and $\boldsymbol{\theta}$ the genetic code, or "chromosome". It assumes that fitness is a result of some "good" pieces of $\boldsymbol{\theta}$ and by inheriting these pieces plus some mutations fitness is enhanced.

In a genetic algorithm, at each iteration t, at least two candidate solutions $\boldsymbol{\theta}_{t,1}$, ..., $\boldsymbol{\theta}_{t,P}$ have to be tracked. Each iteration consists of several steps.

1. **Selection.** Randomly select from $\boldsymbol{\theta}_{t,1}, \ldots, \boldsymbol{\theta}_{t,P}$ to form a set of pairs. The selection should be based on a *fitness function* $\phi(\boldsymbol{\theta})$. In general, larger values of $f(\boldsymbol{\theta})$ result in larger values of $\phi(\boldsymbol{\theta})$, and higher chance for $\boldsymbol{\theta}$ to be selected.
   There are many selection mechanisms:
   a) select one parent $\boldsymbol{\theta}$ with probability proportional to $\phi(\boldsymbol{\theta})$ and the other parent completely at random;
   b) select each parent independently with probability proportional to $\phi(\boldsymbol{\theta})$;

   ϕ must be selected carefully: using $\phi(\boldsymbol{\theta}_{t,i}) = f(\boldsymbol{\theta}_{t,i})$ may result in rapid convergence into a local maximum. A common choice is

$$\phi(\boldsymbol{\theta}_{t,i}) = \frac{2r_i}{P(P+1)}$$

   where r_i is the *rank* of $f(\boldsymbol{\theta}_{t,i})$ in $f(\boldsymbol{\theta}_{t,1}), \ldots, f(\boldsymbol{\theta}_{t,P})$.

2. **Breeding.** For each pair, $(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b)$, generate one or more "offspring" $\boldsymbol{\theta}' = c(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b) \in \boldsymbol{\Theta}$, where c is a random operator.
   Typically, c is a "crossover". If

   $$\boldsymbol{\theta}_a = (\theta_{a1}, \dots, \theta_{aC}),$$
   $$\boldsymbol{\theta}_b = (\theta_{b1}, \dots, \theta_{bC}),$$

   then a crossover works as follows,

   a) randomly choose a position $1 \le d \le C$; and
   b) combine $(\theta_{a1}, \dots, \theta_{ad})$ and $(\theta_{b,d+1}, \dots, \theta_{bC})$ to form

   $$\boldsymbol{\theta}' = (\theta_{a1}, \dots, \theta_{ad}, \theta_{b,d+1}, \dots, \theta_{bC}).$$

   If necessary, also take

   $$\boldsymbol{\theta}'' = (\theta_{a,d+1}, \dots, \theta_{aC}, \theta_{b1}, \dots, \theta_{bd})$$

   to be another offspring.

3. **Mutation.** Make some random modifications to each offspring. Typically, if an offspring chromosome is

$$\boldsymbol{\theta} = (\theta_1, \dots, \theta_C)$$

then for $i = 1, \dots, C$, with a small probability $0 < p < 1$ (mutation rate), change θ_i to a different value.

The offspring produced at iteration t are taken as the candidate solutions for iteration $t + 1$.

Initialization.

Usually the first generation consists of completely random individuals.

- Large values of the size of a generation, P, are preferred. For binary encoding of $\boldsymbol{\theta}$, one suggestion is to have $C \leq P \leq 2C$, where C is the chromosome length. In most real applications, population sizes have ranged between 10 and 200.
- Mutation rates are typically very low, in the neighborhood of 1%.

Termination.

A genetic algorithm is usually terminated after a maximum number of iterations. Alternatively, the algorithm can be stopped once the genetic diversity in the current generation is sufficiently low.

In many problems, like the traveling salesman problem, it is natural to write $\boldsymbol{\theta}$ as a permutation of $1, 2, \ldots, p$.

- Standard crossover usually produces invalid sequences

For example, if

$$\boldsymbol{\theta}_a = \{7, 5, 2, 6, 3, 1, 9, 4, 8\}$$
$$\boldsymbol{\theta}_b = \{9, 1, 2, 3, 8, 6, 7, 5, 4\}$$

and if the crossover point is between 2nd and 3rd positions, then it produces

$$\{7, 5, 2, 3, 8, 6, 7, 5, 4\}$$

an invalid traveling route.

A remedy is order crossover: pick a number of positions from one parent and get the values at those positions, say $i_1, \ldots, i_s$. Next identify those values from the 2nd parent. Suppose the set of values are found at $j_1 < j_2 < \ldots < j_s$. Put i_1 at j_1, i_2 at j_2, ..., i_s at j_s while keeping the values of the 2nd on other locations unchanged.
Example: Consider parents (752631948) and (912386754) and random positions $(4, 6, 7)$. The offsprings are (612389754) and (352671948).
The drawback of the operation is that it destroys some important structures of the parent routes, in particular, the links.

Edge-recombination crossover uses a different idea. It generates an offspring whose edges belong to those of the parent routes. By edge it means a link into or out of a city in a route. For example, the above two parents have the following links, the order of numbers in each link is unimportant.

$$(1,2),(1,3),(1,9),(2,3),(2,5),(2,6),(3,6),(3,8),$$
$$(4,5),(4,8),(4,9),(5,7),(6,7),(6,8),(7,8)$$

First, choose one of the initial cities of the parents as the initial city of the offspring. At k-th step, if $i_1, \ldots, i_k$ have been chosen as the first k cities on the route, then choose among cities that are linked to i_k *and* are different from $i_1, \ldots, i_{k-1}$ as the $(k+1)$st city of the offspring.

EM and MM Algorithms

EM Optimizations

What it is for

- inference when there is missing information
  - incomplete observations
  - auxiliary parameters that are unobserved but can facilitate inference on the main parameter.
  - Some important applications of the EM algorithm are in problems where one has what we might call pseudo missing data. We never had any chance of obtaining such data, but we could pretend they are missing and use EM algorithm to facilitate the computation of MLEs.

Why it is important

- widely useful
- simple to implement
- numerically stable

Suppose a population consists of several sub-populations, each following a distribution $p(x \mid \theta_k)$ and having population fraction q_k. Typically,

$$\boldsymbol{q} = (q_1, \ldots, q_K), \quad \boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$$

are unknown, and the goal is to estimate them.
Let $x_1, \ldots, x_n$ be a sample. If for each x_i we know it comes from the z_i^{th} sub-population, then, letting

$$\boldsymbol{x} = (x_1, \ldots, x_n), \quad \boldsymbol{z} = (z_1, \ldots, z_n),$$

the likelihood function of $(\boldsymbol{q}, \boldsymbol{\theta})$ is

$$L(\boldsymbol{q}, \boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{z}) = \prod_{i=1}^{n} [q_{z_i} \times p(x_i \mid \theta_{z_i})].$$

However, in many cases, $\boldsymbol{z}$ is unobservable:

- cluster analysis
- the "sub-populations" may be constructs that facilitate inference or decision making, so they are nonexistent in reality.

The likelihood function then becomes

$$L(\boldsymbol{q}, \boldsymbol{\theta} \mid \boldsymbol{x}) = \sum_{\boldsymbol{z}} L(\boldsymbol{q}, \boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{z}).$$

The question is how to get the MLE of $\boldsymbol{q}$ and $\boldsymbol{\theta}$ efficiently.
In this example, $\boldsymbol{y} = (\boldsymbol{x}, \boldsymbol{z})$ is the complete data. When only $\boldsymbol{x}$ is observable, $\boldsymbol{z}$ is called the missing data.

Suppose a population follows a distribution $p(y \mid \theta)$, where θ is the unknown parameter. Suppose a random sample is collected from $p(y \mid \theta)$. However, when an observation y_i is greater than a cut-off C, it is truncated and recorded as C, such as in clinical trials. In this case, we only know $y_i \geq C$ but not its exact value. On the other hand, if $y_i \leq C$, then it is fully observed. Therefore, the observed data is

$$x_1 = \min(y_1, C), \ldots, x_n = \min(y_n, C).$$

In this case, $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the complete data and $\boldsymbol{x} = (x_1, \ldots, x_n)$. The question is how to estimate θ based on $\boldsymbol{x}$.

Suppose $Z_1, \ldots, Z_n$ and $X_1, \ldots, X_n$ are time series related by

$$X_k = f(Z_k \,|\, \theta),$$

where the transformation f is parametrized by θ. Z_k may include not only variables of interest, but also some "noise" that interact with the variables. Suppose the joint distribution of $Z_1, \ldots, Z_n$ has the parametric form $h(z_1, \ldots, z_n \,|\, \gamma)$, however, the values of θ and γ are both unknown. If $X_1, \ldots, X_n$ are observed but $Z_1, \ldots, Z_n$ are only partially observed or completely hidden, how to estimate θ and γ?

The Hidden Markov Model (HMM) is a typical case. Let $S_1, \ldots, S_n$ be a discrete Markov chain with a finite number of states. Conditional on $S_1, \ldots, S_n$, the observations $X_1, \ldots, X_n$ are independent, such that, given $S_k = s$, $X_k \sim N(\mu_s, \sigma_s^2)$.

$$\begin{tikzcd}
S_1 \arrow[r] \arrow[d] & S_2 \arrow[r] \arrow[d] & S_3 \arrow[r] \arrow[d] & S_4 \arrow[r] \arrow[d] & \cdots \\
X_1 & X_2 & X_3 & X_4 &
\end{tikzcd}$$

In this case,

$$Z_k = (S_k, W_k),$$

where $W_1, \ldots, W_n \sim N(0, 1)$ are independent of $Z_1, \ldots, Z_n$, such that $X_k = \mu_{S_k} + \sigma_{S_k} W_k$. Typically, μ_k, σ_k and the transition probabilities of S_k are unknown and only X_k are observed.

Positron emission tomography (PET) is a tool to study the metabolic activity in organs. To generate a PET scan, some radioactive material is administered into the organ. Then the emissions of the material are recorded using a PET scanner, which consists of an array of detectors surrounding the patient's body. The region of interest is divided into "voxels". The number of photons Y_{ij} coming from voxel i and received by detector j is assumed to follow a Poisson distribution

$$Y_{ij} \sim \text{Poisson}(\theta_i a_{ij})$$

where the coefficient a_{ij} can be determined accurately beforehand, and θ_i is the emission density of the radioactive material in voxel i, which is used to quantify the metabolic activity therein.

If for each pair of voxel and detector, we can observe Y_{ij}, then θ_j's can be estimated by MLE. However, in reality, each detector j receives photons from all the voxels, and hence only the sum

$$X_j = \sum_i Y_{ij}$$

is observable. The goal is to use $\boldsymbol{X} = (X_j)$ instead of $\boldsymbol{Y} = (Y_{ij})$ to estimate $\boldsymbol{\theta} = (\theta_j)$.

Suppose the complete data

$$\boldsymbol{Y} \sim p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})$$

while the observed data is $\boldsymbol{X} = M(\boldsymbol{Y})$, where M is a many-to-one mapping. Then

$$\underbrace{p_{\boldsymbol{X}}(\boldsymbol{x} \,|\, \boldsymbol{\theta})}_{L(\boldsymbol{\theta} \,|\, \boldsymbol{x})} = \int_{M(\boldsymbol{y})=\boldsymbol{x}} \underbrace{p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})}_{L(\boldsymbol{\theta} \,|\, \boldsymbol{y})} \, \mathrm{d}\boldsymbol{y}.$$

If only $\boldsymbol{X} = \boldsymbol{x}$ is observed, then the MLE $\widehat{\theta}$ satisfies

$$L'(\widehat{\theta} \,|\, \boldsymbol{x}) = 0$$

i.e.,

$$\int_{M(\boldsymbol{y})=\boldsymbol{x}} L'(\widehat{\theta} \,|\, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} = 0.$$

Write log-likelihood function $l(\boldsymbol{\theta} \mid \boldsymbol{y}) = \ln L(\boldsymbol{\theta} \mid \boldsymbol{y})$, so that

$$[L(\boldsymbol{\theta} \mid \boldsymbol{y})]' = [e^{l(\boldsymbol{\theta} \mid \boldsymbol{y})}]' = [l(\boldsymbol{\theta} \mid \boldsymbol{y})]' e^{l(\boldsymbol{\theta} \mid \boldsymbol{y})} = [l(\boldsymbol{\theta} \mid \boldsymbol{y})]' L(\boldsymbol{\theta} \mid \boldsymbol{y}).$$

Therefore,

$$\int_{M(\boldsymbol{y})=\boldsymbol{x}} L'(\widehat{\theta} \mid \boldsymbol{y})\, \mathrm{d}\boldsymbol{y} = 0$$

is the same as

$$\int_{M(\boldsymbol{y})=\boldsymbol{x}} [l(\widehat{\theta} \mid \boldsymbol{y})]' L(\widehat{\theta} \mid \boldsymbol{y}) \ \mathrm{d}\boldsymbol{y} = 0.$$

In principle, Newton's method can be used to find a solution. However, since $\widehat{\theta}$ appears in two places in the integration, the implementation is often messy and because of that, numerically unstable.

We may try to “decouple” the two $\boldsymbol{\theta}$’s in the integration to get simpler iterations. An iterative procedure could go as follows. At step t, if we have $\boldsymbol{\theta}_t$, then find $\boldsymbol{\theta}_{t+1}$ to solve

$$\int_{M(\boldsymbol{y})=\boldsymbol{x}} [l(\boldsymbol{\theta}_{t+1} \mid \boldsymbol{y})]' L(\boldsymbol{\theta}_t \mid \boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} = 0,$$

or, to optimize the function

$$\int_{M(\boldsymbol{y})=\boldsymbol{x}} l(\boldsymbol{\theta} \mid \boldsymbol{y}) L(\boldsymbol{\theta}_t \mid \boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}.$$

This is a function in $\boldsymbol{\theta}$. Since $\boldsymbol{\theta}_t$ is different at each step, the function is different. If $\boldsymbol{\theta}_t$ converge, then the limit is a solution to the original MLE.

The explicit numerical integration in the above function is inconvenient. To replace it, note that, since $\boldsymbol{\theta}_t$ is fixed, the above way to find $\boldsymbol{\theta}_{t+1}$ is equivalent to optimizing

$$\int_{M(\boldsymbol{y})=\boldsymbol{x}} l(\boldsymbol{\theta} \mid \boldsymbol{y}) \frac{L(\boldsymbol{\theta}_t \mid \boldsymbol{y})}{L(\boldsymbol{\theta}_t \mid \boldsymbol{x})} \, \mathrm{d}\boldsymbol{y}.$$

For any $\boldsymbol{\theta}_t$,

$$\frac{L(\boldsymbol{\theta}_t \mid \boldsymbol{y})}{L(\boldsymbol{\theta}_t \mid \boldsymbol{x})} = \frac{p_{\boldsymbol{Y}}(\boldsymbol{y} \mid \boldsymbol{\theta}_t)}{p_{\boldsymbol{X}}(\boldsymbol{x} \mid \boldsymbol{\theta}_t)} = p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}_t),$$

the conditional density of $\boldsymbol{Y}$ given $\boldsymbol{X} = \boldsymbol{x}$ under $p_{\boldsymbol{Y}}(\boldsymbol{y} \mid \boldsymbol{\theta}_t)$. Then the integral is simply $\mathbb{E}\left[l(\boldsymbol{\theta} \mid \boldsymbol{Y}) \mid \boldsymbol{x}, \boldsymbol{\theta}_t\right]$. In turns out that in order to find suitable $\boldsymbol{\theta}_{t+1}$, we need to maximize this function.

EM (Expectation-Maximization) algorithm.

Define

$$Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}') = \mathbb{E}\left[l(\boldsymbol{\theta} \mid \boldsymbol{Y}) \,|\, \boldsymbol{x}, \boldsymbol{\theta}'\right] = \mathbb{E}\left[\ln L(\boldsymbol{\theta} \mid \boldsymbol{Y}) \,|\, \boldsymbol{x}, \boldsymbol{\theta}'\right].$$

- Initialize $\boldsymbol{\theta}_0$.
- At iteration $t = 1, 2, \ldots$, with $\boldsymbol{\theta}_t$ given, set $\boldsymbol{\theta}_{t+1}$ to maximize $Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t)$.

The "E step" of the algorithm refers to the computation of $Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t)$, and the "M step" refers to the maximization of $Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t)$. Stopping criteria are usually based upon $|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t|$ or $|Q(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t) - Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_t)|$.

If $\boldsymbol{Y} = (\boldsymbol{X}, \boldsymbol{Z})$, where $\boldsymbol{Z}$ is the missing data, then, given $\boldsymbol{X} = \boldsymbol{x}$, the only unknown part of $\boldsymbol{Y}$ is $\boldsymbol{Z}$, so

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}') &= \mathbb{E}\left[\ln L(\boldsymbol{\theta} \mid \boldsymbol{Y}) \mid \boldsymbol{x}, \boldsymbol{\theta}'\right] \\ &= \mathbb{E}\left[\ln L(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{Z}) \mid \boldsymbol{x}, \boldsymbol{\theta}'\right] \\ &= \begin{cases} \displaystyle\sum_{\boldsymbol{z}} p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}') \ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}), & \text{if } \boldsymbol{Z} \text{ is discrete} \\ \displaystyle\int p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}') \ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{z}, & \text{if } \boldsymbol{Z} \text{ is continuous} \end{cases} \end{aligned}$$

Gaussian mixture clustering

Suppose a population is a mixture of Gaussian distributions $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with population fractions q_k, $k = 1, \ldots, K$. The goal is to estimate $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, q_k.
Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ be an iid sample from the population. Let z_i be the index of the Gaussian distribution from which $\boldsymbol{x}_i$ is sampled. Suppose that

$$\boldsymbol{z} = (z_1, \ldots, z_n)$$

cannot be observed. The observed data thus is

$$\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n).$$

Suppose at iteration t of the EM algorithm, one has obtained

$$\boldsymbol{\theta}_t = (\boldsymbol{\mu}_{t1}, \dots, \boldsymbol{\mu}_{tK}, \boldsymbol{\Sigma}_{t1}, \dots \boldsymbol{\Sigma}_{tK}, q_{t1}, \dots, q_{tK}).$$

Then, since $(\boldsymbol{x}_i, z_i)$ are independent, and z_i are discrete, for any candidate

$$\boldsymbol{\theta} = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K, q_1, \dots, q_K),$$

one has

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}_t) &= \sum_{\boldsymbol{z}} p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}_t) \ln p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} \underbrace{p(z_i = k \mid \boldsymbol{x}_i, \boldsymbol{\theta}_t)}_{w_{ik}} \ln p(z_i = k, \boldsymbol{x}_i \mid \boldsymbol{\theta}). \end{aligned}$$

To maximize $Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t)$, note that w_{ik} has nothing to do with $\boldsymbol{\theta}$ so it should be computed in the first place. By Bayes rule,

$$w_{ik} = \frac{p(z_i = k, \boldsymbol{x}_i \,|\, \boldsymbol{\theta}_t)}{p(\boldsymbol{x}_i \,|\, \boldsymbol{\theta}_t)} = \frac{p(z_i = k, \boldsymbol{x}_i \,|\, \boldsymbol{\theta}_t)}{\sum_{s=1}^K p(z_i = s, \boldsymbol{x}_i \,|\, \boldsymbol{\theta}_t)},$$

with

$$\begin{aligned} p(z_i = k, \boldsymbol{x}_i \,|\, \boldsymbol{\theta}_t) &= p(z_i = k \,|\, \boldsymbol{\theta}_t) p(\boldsymbol{x}_i \,|\, z_i = k, \boldsymbol{\theta}_t) \\ &= q_{tk} \phi(\boldsymbol{x}_i \,|\, \boldsymbol{\mu}_{tk}, \boldsymbol{\Sigma}_{tk}), \end{aligned} \tag{1}$$

where $\phi(\boldsymbol{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the density of $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ at $\boldsymbol{x}$. Once w_{ik}'s are computed,

$$Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t) = \sum_{k=1}^K \sum_{i=1}^n w_{ik} \ln p(z_i = k, \boldsymbol{x}_i \,|\, \boldsymbol{\theta}),$$

which looks a lot simpler.

As in (1), $p(z_i = k, \boldsymbol{x}_i \mid \boldsymbol{\theta}) = q_k n(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Since

$$n(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{(2\pi)^{-d/2}}{\sqrt{|\boldsymbol{\Sigma}_k|}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)'\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)\right\},$$

then

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}_t) &= \sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik} \ln[(2\pi)^{-d/2} q_k] - \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik} \ln|\boldsymbol{\Sigma}_k| \\ &\quad - \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)'\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k) \\ &= I_1 - \frac{I_2}{2} - \frac{I_3}{2}. \end{aligned}$$

From last page,

$$I_1 = \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} \ln[(2\pi)^{-d/2} q_k], \quad I_2 = \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} \ln |\boldsymbol{\Sigma}_k|$$

$$I_3 = \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)' \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)$$

First, only I_3 contains $\boldsymbol{\mu}_k$, which is a quadratic form. To *minimize* it, observe that I_3 is the sum of K quadratic forms, each involving a single $\boldsymbol{\mu}_k$

$$I_{3k} = \sum_{i=1}^{n} w_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)' \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k).$$

We only need to find $\boldsymbol{\mu}_k$ to minimize each I_{3k}. From the property of sample mean, $\boldsymbol{\mu}_k$ must be the mean of a weighted sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, each $\boldsymbol{x}_i$ having weight w_{ik}. So

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} w_{ik} \boldsymbol{x}_i}{\sum_{i=1}^{n} w_{ik}}.$$

Next, only I_2 and I_3 contain $\boldsymbol{\Sigma}_k$, and $I_2 + I_3$ is the sum of K terms of the following form, each involving a single $\boldsymbol{\Sigma}_k$,

$$S_k = \sum_{i=1}^{n} w_{ik} \ln |\boldsymbol{\Sigma}_k| + \sum_{i=1}^{n} w_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)' \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)$$

and so we only need to find $\boldsymbol{\Sigma}_k$ to minimize each S_k. Now that $\boldsymbol{\mu}_k$ is equal to the weighted mean of $\boldsymbol{x}_i$, to *minimize* S_k, $\boldsymbol{\Sigma}_k$ must be the sample variance of the weighted sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. So

$$\boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^{n} w_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)'}{\sum_{i=1}^{n} w_{ik}}.$$

Finally, only I_1 contains q_k. Since

$$\begin{aligned} I_1 &= \sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik} \ln[(2\pi)^{-d/2} q_k] \\ &= \sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik} \ln[(2\pi)^{-d/2}] + \sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik} \ln q_k \\ &= -(d/2)\ln(2\pi) \sum_{k=1}^{K}\sum_{i=1}^{n} w_{ik} + \sum_{k=1}^{K}\left(\sum_{i=1}^{n} w_{ik}\right) \ln q_k \end{aligned}$$

it suffices to minimize

$$\sum_{k=1}^{K} \underbrace{\left(\sum_{i=1}^{n} w_{ik}\right)}_{W_k} \ln q_k$$

Note $q_1, \ldots, q_K$ must satisfy $q_1 + \cdots + q_K = 1$. Therefore, the maximization can be obtained by finding a solution for $\mathcal{L}'(q_1, \ldots, q_K) = 0$, i.e.

$$\frac{\partial \mathcal{L}(q_1, \ldots, q_K)}{\partial q_k} = 0$$

where

$$\mathcal{L}(q_1, \ldots, q_K) = \sum_{k=1}^{K} W_k \ln q_k - \lambda \left\{ \sum_{k=1}^{K} q_k - 1 \right\}$$

with λ a Lagrange multiplier. Then

$$q_k = \frac{W_k}{\sum_{k=1}^{K} W_k}.$$

Since for each i, $\sum_{k=1}^{K} w_{ik} = 1$,

$$q_k = \frac{1}{n} \sum_{i=1}^{n} w_{ik}.$$

The above steps consist a basic EM Gaussian clustering algorithm. It can be summarized as follows.

- Given

  $$\boldsymbol{\theta}_t = (\boldsymbol{\mu}_{t1}, \ldots, \boldsymbol{\mu}_{tn}, \boldsymbol{\Sigma}_{t1}, \ldots \boldsymbol{\Sigma}_{tK}, q_{t1}, \ldots, q_{tK}),$$

  for $k = 1, \ldots, K$, compute

  $$w_{ik} = p(z_i = k \,|\, \boldsymbol{x}_i, \boldsymbol{\theta}_t), \quad i = 1, \ldots, n$$

  then set $q_{t+1,k}$ equal to the mean of w_{1k}, ..., w_{nk}, $\boldsymbol{\mu}_{t+1,k}$ and $\boldsymbol{\Sigma}_{t+1,k}$ equal to the weighted average and weighted covariance of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, with weights w_{1k}, ..., w_{nk}, respectively.

For high dimensional data, there are many variants of the algorithm which put constraints on $\boldsymbol{\Sigma}_k$ to improve computational and estimation efficiency. The strongest constraint (and computationally the simplest) is

$$\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_K = \sigma^2 \boldsymbol{I},$$

with σ^2 being the only parameter. Others include

1. $\boldsymbol{\Sigma}_k = \sigma_k^2 \boldsymbol{I}$, $k = 1, \ldots, K$, with σ_k being the parameters;
2. $\boldsymbol{\Sigma}_k$ are diagonal matrices that may be equal to or different from each other;
3. $\boldsymbol{\Sigma}_1 = \sigma_k^2 \boldsymbol{\Sigma}$, with the parameters being σ_k^2 and $\boldsymbol{\Sigma}$.

If Newton's method is used to directly maximize the log likelihood function $l(\boldsymbol{\theta} \,|\, \boldsymbol{x}) = \ln L(\boldsymbol{\theta} \,|\, \boldsymbol{x})$, then one would have to differentiate the function

$$l(\boldsymbol{\theta} \,|\, \boldsymbol{x}) = \sum_{i=1}^{n} l_i(\boldsymbol{\theta})$$

where, for every $i = 1, \ldots, n$,

$$\begin{aligned} l_i(\boldsymbol{\theta}) = \ln p(\boldsymbol{x}_i \,|\, \boldsymbol{\theta}) &= \ln \left\{ \sum_{z_i} p(\boldsymbol{x}_i, z_i \,|\, \boldsymbol{\theta}) \right\} \\ &= \ln \left\{ \sum_{k=1}^{K} p(\boldsymbol{x}_k \,|\, z_i = k, \boldsymbol{\theta}) p(z_i = k \,|\, \boldsymbol{\theta}) \right\} \\ &= \ln \left\{ \sum_{k=1}^{K} \frac{(2\pi)^{-\frac{d}{2}} q_k}{\sqrt{|\boldsymbol{\Sigma}_k|}} e^{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)' \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)} \right\}. \end{aligned}$$

The iterations apparently are quite involved.

Computation

The R package

```
mclust
```

provides a large number of clustering methods, including EM based methods. Use

```
install.packages("mclust")
```

to install it on your computer if it is not yet installed. To use, first execute

```
library("mclust")
```

which loads all the functions in the package into an R session.

`mclustBIC`

computes the Bayes information criterion associated with different Gaussian mixture models, which are characterized by number of clusters and constraints on the covariance matrices.

`mclustModel`

estimate the parameters for the best model determined via `mclustBIC`.

`mclustModelNames`

lists the names of models used in the `mclust` package.

`mclust2Dplot`

plots 2-D data given parameters of a Gaussian mixture model for the data. Some MATLAB routines, such as `EM-GM.m` can perform some of the functions of the R package.

EM for Exponential Families

Many important parametric distributions belong to an exponential family, which has the form

$$p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = c_1(\boldsymbol{y}) c_2(\boldsymbol{\theta}) \exp\left\{\boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y})\right\},$$

where $\boldsymbol{\theta}$ is a vector of parameters and $\boldsymbol{s}(\boldsymbol{y})$ a vector of sufficient statistics.

1. The normal densities $N_d(\boldsymbol{\theta}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma}$ being fixed consist an exponential family, because

$$p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = \frac{e^{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\theta})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\theta})}}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}$$

$$= \underbrace{\frac{e^{-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}}}{(2\pi)^{d/2}}}_{c_1(\boldsymbol{y})} \underbrace{\frac{e^{-\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}}}{|\boldsymbol{\Sigma}|^{1/2}}}_{c_2(\boldsymbol{\theta})} \exp\Big\{\boldsymbol{\theta}^T \underbrace{(\boldsymbol{\Sigma}^{-1}\boldsymbol{y})}_{\boldsymbol{s}(\boldsymbol{y})}\Big\}.$$

1. Poisson distribution Poisson(θ), where $\theta > 0$, has distribution function

$$p(y \,|\, \theta) = \frac{e^{-\theta} y^{\theta}}{y!}, \quad y = 0, 1, \ldots$$

If we define

$$c_1(y) = \frac{1}{y!} \quad c_2(\theta) = e^{-\theta} \quad s(y) = \ln y,$$

then $p(y \,|\, \theta) = c_1(y) c_2(\theta) e^{\theta s(y)}$. Therefore, Poisson($\theta$) also form an exponential family.

1. Gamma distribution Gamma(a, r), where $a > 0$, $r > 0$, has density

$$p(y \mid a, r) = \begin{cases} \frac{e^{-y/r} y^{a-1}}{r^a \Gamma(a)} & y > 0 \\ 0 & y \leq 0. \end{cases}$$

Indeed, if we define $b = 1/r$, $\theta = (a, b)$, and

$$c_1(y) = \frac{1}{y} \quad c_2(\boldsymbol{\theta}) = c_2(a, b) = \frac{b^a}{\Gamma(\theta_1)} \quad \boldsymbol{s}(y) = (\ln y, -y)$$

Then

$$c_1(y) c_2(\boldsymbol{\theta}) \exp(\boldsymbol{\theta}^T \boldsymbol{s}(y)) = \frac{1}{y} \frac{1}{b^a \Gamma(a)} e^{a \ln y - by} = p(y \mid a, r).$$

1. Beta distribution Beta(a, b), where $a > 0,\ b > 0$, has density

$$p(y \mid a, b) = \begin{cases} \dfrac{y^{a-1}(1-y)^{b-1}}{B(a, b)}, & 0 < y < 1 \\ 0 & \text{otherwise} \end{cases}$$

If we let $\boldsymbol{\theta} = (a, b)$,

$$c_1(y) = \frac{1}{y(1-y)} \quad c_2(\boldsymbol{\theta}) = \frac{1}{B(a, b)} \quad \boldsymbol{s}(y) = (\ln y, \ln(1-y))$$

then

$$c_1(y)c_2(\boldsymbol{\theta})\exp(\boldsymbol{\theta}^T\boldsymbol{s}(y)) = \frac{1}{y(1-y)}\frac{1}{B(a, b)}e^{a\ln y + b\ln(1-y)} = p(y \mid a, b).$$

Suppose $\boldsymbol{x} = M(\boldsymbol{y})$ is observed and we want to estimate $\boldsymbol{\theta}$. The EM algorithm in this case is significantly simplified, with no explicit evaluation of $Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}')$. It works as follows. Given $\boldsymbol{\theta}_t$, set $\boldsymbol{\theta}_{t+1}$ such that

$$\mathbb{E}\left[s(\boldsymbol{Y}) \,|\, \boldsymbol{\theta}_{t+1}\right] = \mathbb{E}\left[s(\boldsymbol{Y}) \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t\right]. \tag{2}$$

To get the result, we start with the evaluation

$$\begin{aligned} Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t) &= \int [\ln p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})] p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t) \,\mathrm{d}\boldsymbol{y} \\ &= \int [\ln c_1(\boldsymbol{y}) + \ln c_2(\boldsymbol{\theta}) + \boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y})] p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t) \;\mathrm{d}\boldsymbol{y} \\ &= \int [\ln c_1(\boldsymbol{y})] p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t) \;\mathrm{d}\boldsymbol{y} \\ &\qquad + \int [\ln c_2(\boldsymbol{\theta})] p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t) \;\mathrm{d}\boldsymbol{y} + \int \boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y}) p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t) \;\mathrm{d}\boldsymbol{y} \\ &= k(\boldsymbol{\theta}_t) + \ln c_2(\boldsymbol{\theta}) + \boldsymbol{\theta}^T \mathbb{E}\left[s(\boldsymbol{Y}) \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t \right]. \end{aligned}$$

To maximize $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}_t)$ as a function of $\boldsymbol{\theta}$, it suffices to solve

$$(\ln c_2(\boldsymbol{\theta}))' + \mathbb{E}\left[s(\boldsymbol{Y}) \mid \boldsymbol{x}, \boldsymbol{\theta}_t\right] = 0,$$

or

$$\frac{c_2'(\boldsymbol{\theta})}{c_2(\boldsymbol{\theta})} + \mathbb{E}\left[s(\boldsymbol{Y}) \mid \boldsymbol{x}, \boldsymbol{\theta}_t\right] = 0.$$

The following equality is basic for exponential families. For *any* $\boldsymbol{\theta}$,

$$\frac{c_2'(\boldsymbol{\theta})}{c_2(\boldsymbol{\theta})} = -\mathbb{E}\left[\boldsymbol{s}(\boldsymbol{Y}) \mid \boldsymbol{\theta}\right] = -\int \boldsymbol{s}(\boldsymbol{y}) p_{\boldsymbol{Y}}(\boldsymbol{y} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{y}. \tag{3}$$

Assuming the result is correct for now, it is seen that $\boldsymbol{\theta}_{t+1}$ should be set such that $\mathbb{E}\left[s(\boldsymbol{Y}) \mid \boldsymbol{\theta}_{t+1}\right] = \mathbb{E}\left[s(\boldsymbol{Y}) \mid \boldsymbol{x}, \boldsymbol{\theta}_t\right]$, i.e., as in (2).

To get (3), notice $\int p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = 1$ for any $\boldsymbol{\theta}$, i.e.

$$c_2(\boldsymbol{\theta}) \int c_1(\boldsymbol{y}) \exp\{\boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y})\} \, \mathrm{d}\boldsymbol{y} = 1.$$

Then

$$\ln c_2(\boldsymbol{\theta}) = -\ln \left\{ \int c_1(\boldsymbol{y}) \exp\{\boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y})\} \, \mathrm{d}\boldsymbol{y} \right\}.$$

Differentiate both sides to get

$$\begin{aligned} \frac{c_2'(\boldsymbol{\theta})}{c_2(\boldsymbol{\theta})} &= -\frac{\int c_1(\boldsymbol{y}) \boldsymbol{s}(\boldsymbol{y}) \exp\{\boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y})\} \, \mathrm{d}\boldsymbol{y}}{\int c_1(\boldsymbol{y}) \exp\{\boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y})\} \, \mathrm{d}\boldsymbol{y}} \\ &= -c_2(\boldsymbol{\theta}) \int c_1(\boldsymbol{y}) \boldsymbol{s}(\boldsymbol{y}) \exp\{\boldsymbol{\theta}^T \boldsymbol{s}(\boldsymbol{y})\} \, \mathrm{d}\boldsymbol{y} \\ &= -\int \boldsymbol{s}(\boldsymbol{y}) p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{y}, \end{aligned}$$

as claimed.

Convergence of EM

We first show that each step of the algorithm increases the observed-data log likelihood $l(\boldsymbol{\theta} \mid \boldsymbol{x}) = \ln p_{\boldsymbol{X}}(\boldsymbol{x} \mid \boldsymbol{\theta})$, that is

$$l(\boldsymbol{\theta}_{t+1} \mid \boldsymbol{x}) \geq l(\boldsymbol{\theta}_t \mid \boldsymbol{x}).$$

Once this is done, one can say $l(\boldsymbol{\theta}_t \mid \boldsymbol{x})$ always climbs up to a local maximum of $l(\boldsymbol{\theta} \mid \boldsymbol{x})$.

- In general, there is no guarantee that the limit is the (global) maximum.
- EM has slower convergence than the Newton's method, sometime substantially.
- Nevertheless, the ease of implementation and the stable ascent of EM are often very attractive.

Convergence of EM

By $\boldsymbol{X} = M(\boldsymbol{Y})$, $p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = p(\boldsymbol{x}, \boldsymbol{y} \,|\, \boldsymbol{\theta}) = p_{\boldsymbol{X}}(\boldsymbol{x} \,|\, \boldsymbol{\theta}) p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})$, so

$$p_{\boldsymbol{X}}(\boldsymbol{x} \,|\, \boldsymbol{\theta}) = \frac{p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})}{p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})}.$$

Because $\boldsymbol{x}$ is fixed, $p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})$ only depends on $\boldsymbol{\theta}$. To simplify notation, write $f(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})$. Then

$$\ln p_{\boldsymbol{X}}(\boldsymbol{x} \,|\, \boldsymbol{\theta}) = \ln p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta}) - \ln f(\boldsymbol{y} \,|\, \boldsymbol{\theta}).$$

Convergence of EM

Integrate both sides with respect to the density $f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_t)$. From

$$\begin{aligned}&\int [\ln p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})] \, f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_t) \; \mathrm{d}\boldsymbol{y}\\ &= \int [\ln p_{\boldsymbol{Y}}(\boldsymbol{y} \,|\, \boldsymbol{\theta})] \, p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_t) \; \mathrm{d}\boldsymbol{y} = Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t),\end{aligned}$$

It follows

$$\ln p_{\boldsymbol{X}}(\boldsymbol{x} \,|\, \boldsymbol{\theta}_t) = Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t) - H(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t), \tag{4}$$

where

$$H(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t) = \int [\ln f(\boldsymbol{y} \,|\, \boldsymbol{\theta})] f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_t) \; \mathrm{d}\boldsymbol{y}.$$

Convergence of EM

To show $\ln p_{\boldsymbol{X}}(\boldsymbol{x} \,|\, \boldsymbol{\theta}_{t+1}) \geq \ln p_{\boldsymbol{X}}(\boldsymbol{x} \,|\, \boldsymbol{\theta}_t)$, by (4), their difference equals

$$[Q(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t) - Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_t)] - [H(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t) - H(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_t)].$$

By the M-step, $Q(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t) \geq Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_t)$. On the other hand, by the Jensen's inequality,

$$\begin{aligned} H(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t) - H(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_t) &= \int \ln \left[\frac{f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_{t+1})}{f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_t)} \right] f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_t) \,\mathrm{d}\boldsymbol{y} \\ &\leq \ln \left[\int \frac{f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_{t+1})}{f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_t)} f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_t) \,\mathrm{d}\boldsymbol{y} \right] = \ln \left[\int f(\boldsymbol{y} \,|\, \boldsymbol{\theta}_{t+1}) \,\mathrm{d}\boldsymbol{y} \right] = 0, \end{aligned}$$

thus showing the likelihood increases at each iteration.

Variants of EM

- E step is hard: Monte Carlo (MCEM)
- M step is hard: Instead of choosing $\boldsymbol{\theta}_{t+1}$ to maximize $Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_t)$, one simply chooses any $\boldsymbol{\theta}_{t+1}$ for which

$$Q(\boldsymbol{\theta}_{t+1} \,|\, \boldsymbol{\theta}_t) > Q(\boldsymbol{\theta}_t \,|\, \boldsymbol{\theta}_t),$$

  then the resulting algorithm is called generalized EM, or GEM.
  - In GEM, $l(\boldsymbol{\theta}_t \,|\, \boldsymbol{x})$ is still increasing.
  - An example: ECM algorithm, which replaces the M-step of the EM algorithm with several computationally simpler conditional maximization (CM) steps.

Variance Estimation

- Louis' method (1982, JRSSB)
- Supplemented EM (SEM) algorithm (Meng and Rubin, 1991, JASA)
- Oakes' method (1999, JRSSB)
- Bootstrapping
- Empirical information
- Numerical Differentiation

Example: Hidden Markov Model

- Hidden state $\mathbf{s} = (s_t)$, $t = 1, \ldots, T$, follows a Markov chain with initial state distribution π_k, $k = 1, \ldots, M$, and transition probability matrix $\{a_{k,l} = \Pr(s_{t+1} = l | s_t = k)\}$. (Note this matrix is row stochastic.)
- Given the state s_t, the observation u_t is independent of other observations and states
- Given state k, the observation follows distribution $b_k(u)$. For example, $b_k(u|\gamma_k)$ can be $N(\mu_k, \Sigma_k)$; or discrete.
- The observed data is $\mathbf{u} = (u_t)$, $t = 1, \ldots, T$.

Likelihood

- Parameters: θ contains π_k's, a_{kl}'s, γ_k's
- Missing (latent, unobserved) states: s_t, $t = 1, \ldots, T$.
- Observed data: u_t, $t = 1, \ldots, T$
- Complete data: $(\mathbf{s}, \mathbf{u})$, where $\mathbf{s} = (s_1, \ldots, s_T)$ and $\mathbf{u} = (u_1, \ldots, u_T)$.
- Complete data likelihood

$$\pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1}, s_t} \prod_{t=1}^{T} b_{s_t}(u_t | \gamma_{s_t})$$

- Complete data log-likelihood

$$\log p(\mathbf{s}, \mathbf{u} | \theta) = \log \pi_{s_1} + \sum_{t=2}^{T} \log a_{s_{t-1}, s_t} + \sum_{t=2}^{T} \log b_{s_t}(u_t | \gamma_{s_t})$$

E-Step

Taking expectation of $\log p(\mathbf{s}, \mathbf{u}|\theta')$ with respect to $p(\mathbf{s}|\mathbf{u}, \theta)$:

$$
\begin{aligned}
& E\Big(\log p(\mathbf{s}, \mathbf{u})|\theta')|\mathbf{u}, \theta) \Big) \\
= & \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{u}, \theta) \log p(\mathbf{s}, \mathbf{u}|\theta') \\
= & \sum_{k=1}^{M} L_k(1) \log \pi'_k + \sum_{t=2}^{T} \sum_{k=1}^{M} \sum_{l=1}^{M} H_{k,l}(t) \log a'_{k,l} \\
& + \sum_{t=1}^{T} \sum_{k=1}^{M} L_k(t) \log b_{s_t}(u_t|\gamma'_k)
\end{aligned}
$$

where

$$L_k(t) = \Pr(s_t = k|\mathbf{u}) = \sum_{\mathbf{s}: s_t = k} p(\mathbf{s}|\mathbf{u})$$

and

$$H_{k,l}(t) = \Pr(s_t = k, s_{k+1} = l|\mathbf{u}) = \sum_{\mathbf{s}: s_t = k, s_{t+1} = l} \Pr(\mathbf{s}|\mathbf{u}).$$

M-step

If b_k's are multivariate normals, the M-step has closed-form solution:

$$
\begin{aligned}
\mu_k &= \frac{\sum_{t=1}^{T} L_k(t) u_t}{\sum_{t=1}^{T} L_k(t)} \\
\Sigma_k &= \frac{\sum_{t=1}^{T} L_k(t)(u_t - \mu_k)(u_t - \mu_k)^\top}{\sum_{t=1}^{T} L_k(t)} \\
a_{k,l} &= \frac{\sum_{t=1}^{T-1} H_{k,l}(t)}{\sum_{t=1}^{T-1} L_k(t)} \\
\pi_k &= \frac{\sum_{t=1}^{T} L_k(t)}{\sum_{k=1}^{M} \sum_{t=1}^{T} L_k(t)}
\end{aligned}
$$

Forward and Backward Probability

- This is for efficient computation of $L_k(t)$'s and $H_{k,l}(t)$'s
- Computation of order M^2T and memory requirement of order MT.
- Forward probability

$$\alpha_k(t) = \Pr(u_1, \dots, u_t, s_t = k)$$

- Backward probability

$$\beta_k(t) = \Pr(u_{t+1}, \dots, u_T | s_t = k)$$

Forward and Backward Algorithms

- Forward probability recursion

$$\alpha_k(1) = \pi_k b_k(u_1), \qquad 1 \le k \le M,$$

$$\alpha_k(t) = b_k(u_t) \sum_{l=1}^{M} \alpha_l(t-1) a_{l,k}, \qquad 1 < t \le T, \quad 1 \le k \le M.$$

- Backward probability recursion

$$\beta_k(T) = 1, \qquad 1 \le k \le M,$$

$$\beta_k(t) = \sum_{l=1}^{M} a_{k,l} b_l(u_{t+1}) \beta_l(t+1), \qquad 1 \le t < T, \quad 1 \le k \le M.$$

Fast Algorithm for the E-step Quantities

$$L_k(t) = \Pr(s_t = k|\mathbf{u}) = \frac{\alpha_k(t)\beta_k(t)}{p(\mathbf{u})}$$

$$H_{k,l}(t) = \Pr(s_t = k, s_{k+1} = l|\mathbf{u}) = \frac{\alpha_k(t)a_{k,l}b_l(u_{t+1})\beta_l(t+1)}{p(\mathbf{u})}$$

$$p(\mathbf{u}) = \sum_{k=1}^{M} \alpha_k(t)\beta_k(t), \qquad \forall t$$

Majorization-Minimization Algorithm

Majorization-Minimization Algorithm

- The MM algorithm is not an algorithm, but a prescription for constructing optimization algorithms.
- The EM algorithm is a special case.
- An MM algorithm operates by creating a surrogate function that majorizes (minorizes) the objective function. When the surrogate function is optimized, the objective function is driven downhill (uphill).
- Majorization-Minimization, Minorization-Maximization.

Majorization-Minimization Algorithm

- It may generate an algorithm that avoids matrix inversion.
- It can linearize an optimization problem.
- It can conveniently deal with equality and inequality constraints.
- It can solve a non-differentiable problem by iteratively solving smooth problems.

Majorization-Minimization Algorithm

- A function $g(\theta \mid \theta^s)$ is said to majorize the function $f(\theta)$ at θ^s if

  $$f(\theta^s) = g(\theta^s \mid \theta^s),$$

  and

  $$f(\theta) \le g(\theta \mid \theta^s)$$

  for all θ.
- Let

  $$\theta^{s+1} = \arg\min_{\theta} g(\theta \mid \theta^s).$$

- It follows that

  $$f(\theta^{s+1}) \le g(\theta^{s+1} \mid \theta^s) \le g(\theta^s \mid \theta^s) = f(\theta^s).$$

  The strict inequality holds unless $g(\theta^{s+1} \mid \theta^s) = g(\theta^s \mid \theta^s)$ and $f(\theta^{s+1}) = g(\theta^{s+1} \mid \theta^s)$.
- Therefore, by alternating between the majorization and the minimization steps, the objective function is monotonically decreasing and thus its convergence is guaranteed.

Majorization-Minimization Algorithm

- We can use any inequalities to construct the desired majorized/minorized version of the objective function. There are several typical choices:
  - Jensen's inequality: for a convex function $f(x)$,

$$f(E(X)) \leq E(f(X)).$$

  - Convexity inequality: for any $\lambda \in [0, 1]$,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$

  - Cauchy-Schwarz inequality.
  - Supporting hyperplanes.
  - Arithmetic-Geometric Mean Inequality.

Example: Penalized AFT model with induced smoothing

AFT model

- Let $\{T_i, C_i, \mathbf{x}_i\}$, $i = 1, \ldots, n$ be n independent copies of $\{T, C, \mathbf{X}\}$, where T_i and C_i are log-transformed failure time and log transformed censoring time, $\mathbf{x}_i$ is a $p \times 1$ covariate vector, and given $\mathbf{X}$, C and T are independent.
- The Accelerated failure time model has the form

$$T_i = \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i, \qquad i = 1, \ldots, n.$$

- The observed data are $(Y_i, \delta_i, \mathbf{x}_i)$, where $Y_i = \min(T_i, C_i)$, $\delta_i = I(T_i < C_i)$.

AFT model

- For a case-cohort study, The weight-adjusted estimating equation with induced smoothing is

$$\tilde{U}_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\Delta_i(\mathbf{x}_i - \mathbf{x}_j)\Phi\left(\frac{e_j(\boldsymbol{\beta}) - e_i(\boldsymbol{\beta})}{r_{ij}}\right) = 0,$$

  where $e_j(\boldsymbol{\beta}) = Y_j - \mathbf{x}_j^{\mathrm{T}}\boldsymbol{\beta}$ and
  $r_{ij}^2 = n^{-1}(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}_j) = n^{-1}\|\mathbf{x}_i - \mathbf{x}_j\|^2$.
- Define $H(x) = x\Phi(x) + \phi(x)$, and

$$\tilde{L}_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\Delta_i r_{ij} H\left(\frac{e_j(\boldsymbol{\beta}) - e_i(\boldsymbol{\beta})}{r_{ij}}\right).$$

  Then solving the estimating equation $\tilde{U}_n(\boldsymbol{\beta}) = 0$ is equivalent to minimizing the objective function $\tilde{L}_n(\boldsymbol{\beta})$.
- The objective function $\tilde{L}_n(\boldsymbol{\beta})$ is convex in $\boldsymbol{\beta}$.

AFT model

- We propose the penalized AFT method, which amounts to minimize

$$\tilde{P}L_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\Delta_i r_{ij} H\left(\frac{e_j(\boldsymbol{\beta}) - e_i(\boldsymbol{\beta})}{r_{ij}}\right) + \lambda P(\boldsymbol{\beta}),$$

where $P(\boldsymbol{\beta})$ is some penalty function and λ is the tuning parameter.

AFT model

- To solve this problem, the key is to identity a surrogate function of $\tilde{L}_n(\boldsymbol{\beta})$ that is easy to work with. In view of the form of the objective function, an immediate idea is to majorize $H(x)$.
- Because $H(x)$ is twice differentiable and $H''(x) = \phi(x) < \phi(0) \approx 0.4$ for any $x \in \mathbb{R}$, it follows that for any $x^0 \in R$,

$$H(x \mid x^0) \leq G(x \mid x^0) \equiv H(x^0) + (x - x^0)\Phi(x^0) + \frac{\phi(0)}{2}(x - x^0)^2.$$

- Note that in a standard optimization method based on quadratic approximation, e.g., Newton-Raphson, the $\phi(0)$ term is replaced with $\phi(x^0)$, which, however, does not guarantee the majorization.

AFT model

- Write

$$\tilde{L}_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} H(a_{ij} + \mathbf{z}_{ij}^{\mathrm{T}}\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta}),$$

where $w_{ij} = \Delta_i r_{ij}$, $a_{ij} = (Y_j - Y_i)/r_{ij}$ and $\mathbf{z}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/r_{ij}$. Then

$$\begin{aligned}\tilde{L}_n(\boldsymbol{\beta} \mid \boldsymbol{\beta}^s) &\le \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} G(a_{ij} + \mathbf{z}_{ij}^{\mathrm{T}}\boldsymbol{\beta} \mid a_{ij} + \mathbf{z}_{ij}^{\mathrm{T}}\boldsymbol{\beta}^s) + \lambda P(\boldsymbol{\beta}) \\ &= \frac{1}{2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{A}\boldsymbol{\beta} - \mathbf{b}^{s^{\mathrm{T}}}\boldsymbol{\beta} + \lambda P(\boldsymbol{\beta}) + \text{const} \\ &\equiv \tilde{L}_n^*(\boldsymbol{\beta} \mid \boldsymbol{\beta}^s) + \text{const},\end{aligned}$$

where

$$\mathbf{A} = \frac{\phi(0)}{n}\sum_{i=1}^{n}\sum_{i=1}^{n} w_{ij}\mathbf{z}_{ij}\mathbf{z}_{ij}^{\mathrm{T}}, \qquad \mathbf{b}^s = \mathbf{A}\boldsymbol{\beta}^s - \frac{1}{n}\sum_{i=1}^{n}\sum_{i=1}^{n} w_{ij}\Phi(a_{ij} + \mathbf{z}_{ij}^{\mathrm{T}}\boldsymbol{\beta}^s)\mathbf{z}_{ij}$$

- Minimizing $\tilde{L}_n^*(\boldsymbol{\beta} \mid \boldsymbol{\beta}^s)$ is a Lasso-type problem, for which a coordinate descent algorithm can be applied.

MM for Penalized AFT

```
Initialize $\boldsymbol{\beta}^0 \in \mathbb{R}^p$. Compute
    $\mathbf{A} = \frac{\phi(0)}{n} \sum_{i=1}^{n} \sum_{i=1}^{n} w_{ij} \mathbf{z}_{ij} \mathbf{z}_{ij}^{\mathrm{T}}.$
Set $k \leftarrow 0$.
repeat
    (1). Majorization step:
        $\mathbf{b}^{k+1} \leftarrow \mathbf{A}\boldsymbol{\beta}^k - \frac{1}{n} \sum_{i=1}^{n} \sum_{i=1}^{n} w_{ij} \Phi(a_{ij} + \mathbf{z}_{ij}^{\mathrm{T}} \boldsymbol{\beta}^k) \mathbf{z}_{ij}.$
    (2). Minimization step:
    Initialize $\tilde{\boldsymbol{\beta}}^0 = \boldsymbol{\beta}^k$.
    while $\|\tilde{\boldsymbol{\beta}}^s - \tilde{\boldsymbol{\beta}}^{s+1}\| / \|\tilde{\boldsymbol{\beta}}^s\| \geq \epsilon$ and $s < s_{max}$ do
        (i) For $j = 1, \ldots, p$,
            $\tilde{\beta}_j^s \leftarrow \frac{1}{a_{jj}} \mathcal{S}(b_j^{k+1} - \mathbf{a}_j^{\mathrm{T}} \tilde{\boldsymbol{\beta}}^s + a_{jj} \tilde{\beta}_j^s, \lambda).$
        (ii) $\tilde{\boldsymbol{\beta}}^{s+1} \leftarrow \tilde{\boldsymbol{\beta}}^s$.
        (iii)$s \leftarrow s + 1$.
    end while
        $\boldsymbol{\beta}^{k+1} \leftarrow \tilde{\beta}^{s+1}.$
    Set $k \leftarrow k + 1$.
until convergence, i.e., $\|\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^k\| / \|\boldsymbol{\beta}^k\| \leq \epsilon$.
```