Optimization

Review: Maximum likelihood

Parameter	θ, a p-vector
Log likelihood	$l(\theta) = \log \mathsf{Pr}(X \mid \theta)$
MLE	$\hat{\theta} = \arg\max_\theta l(\theta)$
Score function	$\dot{l} = (\partial l/\partial\theta_1 \ldots \partial l/\partial\theta_p)'$
Likelihood eqns	$\dot{l}(\theta) = 0$
Hessian	$\ddot{l} = \left(\partial^2 l/\partial\theta_i\partial\theta_j\right)$ [$p \times p$ matrix]
Fisher information	$I(\theta) = -(\mathsf{E}\,\ddot{l}_X)(\theta) = \mathsf{E}\{\dot{l}_X(\dot{l}_X)'\}(\theta)$
Est'd Fisher info	$I(\hat{\theta})$
Observed info	$-\ddot{l}(\hat{\theta})$

When $\theta^\star$ is a local maximum of l, $\dot{l}(\theta^\star) = 0$ and $\ddot{l}(\theta^\star)$ is negative definite.

Saddle point: $\dot{l}(\theta^\star) = 0$ but $\ddot{l}(\theta^\star)$ is neither negative definite nor positive definite.

Optimization

Scalar case

Find $\hat{\theta}$ such that $\dot{l}(\hat{\theta}) = 0$.

Newton's method

$$\hat{\theta}^{(s+1)} = \hat{\theta}^{(s)} - \dot{l}(\hat{\theta}^{(s)}) \,/\, \ddot{l}(\hat{\theta}^{(s)})$$

Quadratic convergence

$$|\hat{\theta}^{(s+1)} - \hat{\theta}| \;\leq\; c\,|\hat{\theta}^{(s)} - \hat{\theta}|^2$$

$\rightarrow$ The number of significant digits nearly doubles at each step (in the neighborhood of $\hat{\theta}$).

What can go wrong?

- Bad starting point
- May not converge to the global maximum

Warning: Usually, in discussions of optimization, authors consider *minimization* (say of $-l(\theta)$) rather than *maximization*. In the following, I've chosen to rewrite things in terms of maximization. In doing that, I may have screwed everything up!

Generalization to higher dimensions

$\hat{\theta}^{(s+1)} = \hat{\theta}^{(s)} + \alpha^{(s)} d^{(s)}$

where $\alpha^{(s)}$ is the **step size**

and $d^{(s)}$ is the **direction** (a p-vector)

Determining the direction

$\hat{\theta}^{(s+1)}$ is acceptable if $l(\hat{\theta}^{(s+1)}) > l(\hat{\theta}^{(s)})$.

We generally pick $d^{(s)} = -R^{-1}\, \dot{l}(\theta^{(s)})$ where R is a negative definite matrix. (For minimizing, pick a pos def R)

Choosing a step size

- Fix $\alpha^{(s)} = \alpha$
- Linear search
  - Find an acceptable $d^{(s)}$
  - Find $\alpha^\star$ maximizing $l(\hat{\theta}^{(s)} + \alpha d^{(s)})$
    - Start at a large value of α and decrease it
    - Approximate $l(\hat{\theta}^{(s)} + \alpha d^{(s)})$ by a polynomial in α.
- Step halving
  If $l(\hat{\theta}^{(s+1)}) < l(\hat{\theta}^{(s)})$, then halve $\alpha^{(s)}$ until $l(\hat{\theta}^{(s+1)}) > l(\hat{\theta}^{(s)})$

Survey of basic methods

1. **Steepest ascent**: $R = -I$

$$\begin{aligned} d^{(s)} &= \dot{l}(\hat{\theta}^{(s)}) \\ \hat{\theta}^{(s+1)} &= \hat{\theta}^{(s)} + \alpha^{(s)}\, \dot{l}(\hat{\theta}^{(s)}) \end{aligned}$$

Crude: gets you where you're going fast, but doesn't converge too quickly once you get there

2. **Newton-Raphson**: $R = \ddot{l}(\hat{\theta}^{(s)}) =$ observed info

$$\begin{aligned} \alpha^{(s)} &= 1 \text{ for all } s \\ d^{(s)} &= -[\ddot{l}(\hat{\theta}^{(s)})]^{-1}\, \dot{l}(\hat{\theta}^{(s)}) \\ \hat{\theta}^{(s+1)} &= \hat{\theta}^{(s)} - [\ddot{l}(\hat{\theta}^{(s)})]^{-1}\, \dot{l}(\hat{\theta}^{(s)}) \end{aligned}$$

- Approximate by linear function
- Need very good starting points; the algorithm is perfectly happy to go down rather than up
- **Suggestion**: Use steepest ascent for a few steps, then switch to Newton-Raphson

Steepest ascent vs steepest descent

Consider optimizing the function $f(x_1, x_2) = \sin(x_1 x_2)$

Function: `f <- function(x) sin(prod(x))`
Gradient: `g <- function(x) x*cos(prod(x))`

Steepest ascent function:

```
sa <-
function(x,F=f, G=g, step=0.1,
         nstep=1000,tol1=1e-8,tol2=1e-6)
{
  for(i in 1:nstep) {
    xn <- x + step*G(x)

    if(all(abs(x-xn) < tol1*(abs(x)+tol2)))
      break

    x <- xn
  }

  c(i,xn,F(xn),G(xn))
}
```

```
> sa(rep(pi/2,2),step= 0.1)
   43  1.25  1.25  1.00  0.00  0.00

> sa(rep(pi/2,2),step= -0.1)
   10  2.17  2.17 -1.00  0.00  0.00
```

3. **Fisher scoring**: $R = (\mathsf{E}\,\ddot{l})(\hat{\theta}^{(s)}) =$ expected info

$$\begin{aligned} \alpha^{(s)} &= 1 \text{ for all } s \\ d^{(s)} &= -[(\mathsf{E}\,\ddot{l})(\hat{\theta}^{(s)})]^{-1}\, \dot{l}(\hat{\theta}^{(s)}) \end{aligned}$$

4. **Modifications of N-R**

(when the Hessian is not available or is not negative definite)

(a) **Greenstadt** (1967)
Replace $\ddot{l}(\theta)$ with a negative definite approximation if $\ddot{l}(\theta)$ is no longer negative definite.

(b) **Levenberg-Marquadt** (1963)
Replace $\ddot{l}(\theta)$ by $\ddot{l}(\theta)+\lambda I_p$ if $\ddot{l}(\theta)$ fails to be negative definite.

(c) **Quasi-Newton** (aka "variable metric methods" or "secant methods")
Approximate Hessian in a way that

- is easier to calculate
- has better global convergence properties

Estimating the Hessian

Davidson-Fletcher-Powell QNR algorithm

Let $y^{(s)} = \dot{l}(\hat{\theta}^{(s)}) - \dot{l}(\hat{\theta}^{(s-1)})$

Let $\sigma^{(s)} = \hat{\theta}^{(s)} - \hat{\theta}^{(s-1)}$

Take

$$H^{(s+1)} = H^{(s)} + \frac{\sigma^{(s)}(\sigma^{(s)})'}{(\sigma^{(s)})'\sigma^{(s)}} - \frac{H^{(s)}y^{(s)}(y^{(s)})'H^{(s)}}{(y^{(s)})'H^{(s)}y^{(s)}}$$

Use the starting point $H^{(0)} = I$

Caution: This sort of method may not give a very good estimate of the Hessian. Quite crude estimates (*eg*, I_p) often suffice for an optimization algorithm, but will result in extremely poor estimates of the variance matrix of $\hat{\theta}$.

Numerical derivatives

Let e_i be a unit vector with $e_{ij} = 1$ if $i = j$.

We estimate the gradient using

$$[\dot{l}(\theta)]_i = (\partial l/\partial\theta_i)(\theta) \approx [l(\theta + \delta_i e_i) - l(\theta - \delta_i e_i)]/(2\delta_i)$$

In calculating derivatives using this formula, I generally start with some medium size δ and then repeatedly halve it until the estimated derivative stabilizes.

(There are fancier, faster and more stable methods, but I won't discuss them here.)

We can estimate the Hessian by applying the above formula twice:

$$[\ddot{l}(\theta)]_{ij} \approx [l(\theta + \delta_i e_i + \delta_j e_j) - l(\theta + \delta_i e_i - \delta_j e_j)$$

$$-l(\theta - \delta_i e_i + \delta_j e_j) + l(\theta - \delta_i e_i - \delta_j e_j)] \,/\, (4\delta_i\delta_j)$$

For the diagonal elements, we get (taking $\delta_i \to \delta_i/2$)

$$[\ddot{l}(\theta)]_{ii} \approx [l(\theta + \delta_i e_i) - 2\, l(\theta) + l(\theta - \delta_i e_i)]/\delta_i^2$$

Optimization in R/Splus

R

`nlm` is the general function for "nonlinear minimization." It can make use of a gradient and/or Hessian, and it can give an estimate of the Hessian at the minimum. See `demo(nlm)` for examples.

Use the function `optimize` for minimizing a function of a single variable, and `uniroot` to find a zero of such a function.

`D` and `deriv` do symbolic differentiation.

Splus

I'm hardly familiar with Splus anymore, but I think `optimize`, `uniroot`, `D`, and `deriv` also exist there.

For general optimization, look at the functions `ms`, `nlmin`, and `nlminb`.

Stupid example using nls

```
> f <- function(x) sin(prod(x))

> nlm(f,rep(pi/2,2),hessian=TRUE)
$minimum
[1] -1

$estimate
[1] 2.170803 2.170803

$hessian
          [,1]     [,2]
[1,] 4.712386 4.713326
[2,] 4.713326 4.712386

> f2 <- function(x) -f(x)

> nlm(f2,rep(pi/2,2),hessian=TRUE)
$minimum
[1] -1

$estimate
[1] 1.253314 1.253314

$hessian
          [,1]     [,2]
[1,] 1.570796 1.571109
[2,] 1.571109 1.570796
```

Another stupid example

Consider $x \sim$ multinomial(n, p)
where $p = [\ (2+\theta)/4,\ (1-\theta)/2,\ \theta/4\ ]$.

The negative log likelihood is the following

```
> nll <- function(theta,x)
+         -sum(x*log(c((2+theta)/4,
+              (1-theta)/2, theta/4)))
```

Suppose we observe the data $x = (100, 94, 6)$.

Using the `nlm` function, we get:

```
> nlm(nll, 0.1, typsize=0.01,
+     hessian=TRUE, x=c(100,94,6))
$minimum
    161.6
$estimate
    0.104
$hessian
    689.7
```

Thus we estimate $\hat{\theta} = 0.10$ with an estimated SE $\approx 1/\sqrt{689.7} \approx 0.04$.