Numerical Maximization and MLE

April 27, 2012

Numerical maximization of likelihood functions

- Grid search
- Steepest ascent
- Newton-Raphson

Based on selected parts of CH 5 of Hamilton.

The basic idea

How can we estimate parameters when we cannot maximize likelihood analytically?

We need to

- Be be able to evaluate the likelihood function for a given set of parameters
- Find a way to evaluate a sequence of likelihoods conditional on difference parameter vectors so that we can feel confident that we have found the parameter vector that maximizes the likelihood

Example: AR(1) process

Log likelihood of AR(1) process is given by

$$
\begin{aligned}
L(\theta) \quad = \quad & -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma^2/\left(1-\phi^2\right)) \\
& -\frac{1}{2}\frac{-\{y_1 - [c/(1-\phi)]\}^2}{\sigma^2} \\
& -\frac{T-1}{2}\log(2\pi) - \frac{T-1}{2}\log(\sigma^2) \\
& -\frac{1}{2}\sum_{t=2}^{T}\left[\frac{-\{y_t - [c+\phi y_{t-1}]\}^2}{\sigma^2}\right]
\end{aligned}
$$

Grid Search

Divide range of parameters into grid and evaluate all possible combinations

- The only method guaranteed to find the global optimum

Take example of AR(1) process

$$x_t = \rho x_{t-1} + u_t : \Theta = \{\rho, \sigma_u^2\}$$

Define grid points

- $\rho : \{-1, -.95, -0.90, ..., 0, ..., 0.90, 0.95, 1\}$
- $\sigma_u^2 : \{0, 0.05, 0.10, ...2.45, 2.5\}$

Evaluate $\ln \mathcal{L}(x^t \mid \Theta)$ for all grid combinations of ρ and σ_u^2

Grid Search: Find the X's

$\rho \backslash \sigma_u^2$	0	0.5	1	1.5	2	2.5
-1	x	x	x	x	x	x
-0.5	x	x	x	x	x	x
0	x	x	x	x	x	x
0.5	x	x	x	x	x	x
1	x	x	x	x	x	x

Grid Search in MatLab

```
RHO=[-1:0.05:1]; SIG2U=[0:0.05:2.5];
G=zeros(length(RHO),length(SIG2U))
for r=1:length(RHO)
    for s=1:length(SIG2U)
      rho=RHO(r);
      sig2u=SIG2U(s);
      [L]=loglikeAR1(x,rho,sig2u)
G(r,s)=L;
    end
end
```

Grid Search in MatLab cont'd

```
function [L]=loglikeAR1(x,rho,sig2u)
T=length(x);
L=0;
p=1;
a=x(1,2:T)-rho*x(1,1:T-1);
for tt=1:T-1
  L=L -.5*(p*log(2*pi)+log(det(sig2u))+...
   ...+a(1,tt)*inv(sig2u)*a(1,tt)');
end
L=L + log(1-rho^2)-log(1-rho^2)*(x(1))^2;
```

Surface plot of grid in MatLab

```
figure(1);
surf(G);
[C1,I1] = max(G);
[C2,I2] = max(C1);
gmax=max(max(G))
rhomax=RHO(I1(1,1))
sig2umax=SIG2U(I2)
```

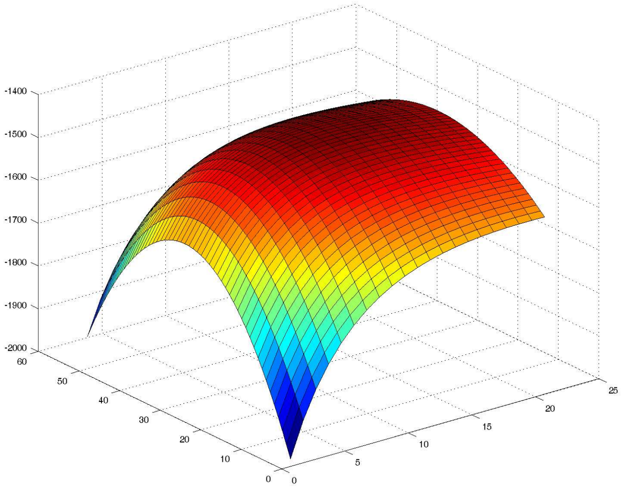

Figure: Estimated posterior densities of structural parameters

Grid search

Pros:

- With a fine enough grid, grid search always finds the global maximum (if parameter space is bounded)

Cons:

- Computationally infeasible for models with large number of parameters

Steepest Ascent method

A blind man climbing a mountain. How to do it:

1. Make initial guess of $\Theta = \Theta^{(0)}$
2. Find direction of "steepest ascent" by computing the gradient

$$\mathbf{g}(\Theta) \equiv \frac{\partial \mathcal{L}(Z \mid \Theta)}{\partial \Theta}$$

which is a vector which can be approximated element by element

$$\frac{\partial \mathcal{L}(Z \mid \Theta^{(0)})}{\partial \theta_i}$$

$$\approx \frac{\mathcal{L}(Z \mid \theta_j = \theta_j^{(0)} + \varepsilon : j = i; \theta_j = \theta_j^{(0)} \text{ otherwise}) - \mathcal{L}(Z \mid \Theta^{(0)})}{\varepsilon}$$

for each θ_j in $\Theta = \{\theta_1, \theta_2, ...\theta_J\}$.

Steepest Ascent method cont'd

3. Take step proportional to gradient, i.e. in the direction of "steepest ascent" by setting new value of parameter vector as $\Theta^{(1)} = \Theta^{(0)} + s\mathbf{g}(\Theta)$

4. Repeat Steps 2 and 3 until convergence.

Steepest Ascent in MatLab

```
rho=.2;sig2u=1.4;
theta=[rho;sig2u;];
eps=1e-3;
s=1e-3;
gr=zeros(2,1);
diff=1;
tol=1e-20;
```

Steepest Ascent in MatLab cont'd

```
while diff > tol;
 for j=1:length(theta);
 epsvec=zeros(2,1);epsvec(j)=eps;
 L=loglikeAR1(x,theta(1),theta(2));
 thetadelta=theta+eps;
 Ldelta=loglikeAR1(x,thetadelta(1),thetadelta(2));
 gr(j)=(Ldelta-L)/eps;
 end
 thetast=theta+gr*s;
 diff=max(max(abs(thetast-theta)));
 theta=thetast;
end
```

Steepest Ascent method

Pros:

- Feasible for models with a large number of parameters

Cons:

- Can be hard to calibrate even for simple models to achieve the right rate of convergence
  - Too small steps and “convergence” is achieved to soon
  - Too large step and parameters may be sent off into orbit.
- Can converge on local maximum. (How could a blind man on $K2$ find his way to Mt Everest?)

Newton-Raphson

Newton-Raphson is similar to steepest ascent, but also computes the step size

- Step size depends on second derivative
- May converge faster than steepest ascent
- Requires concavity, so is less robust when shape of likelihood function is unknown