

Numerical Maximization and MLE

Econometric Methods II

Universitat Pompeu Fabra

Code and slides available at

http://www.kris-nimark.net/TS_UPF_2010.html

June 8, 2010

Numerical maximization of likelihood functions

- ▶ Grid search
- ▶ Steepest ascent
- ▶ Newton-Raphson
- ▶ Simulated annealing

Based on selected parts of CH 5 of Hamilton and article by Goffe, Ferrier and Rogers (1994)

The basic idea

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

We need to

- ▶ 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

Some notation

What is a likelihood function?

$$\hat{\Theta} = \arg \max_{\Theta} \mathcal{L}(Z | \Theta)$$

- ▶ Any function of Θ that is proportional to $p(Z | \Theta)$
- ▶ So why the formulation $\mathcal{L}(Z | \Theta)$? Bayes' theorem:

$$\begin{aligned} p(Z | \Theta) p(\Theta) &= p(\Theta | Z) p(Z) \\ &\Leftrightarrow \\ p(Z | \Theta) &= p(\Theta | Z) \frac{p(Z)}{p(\Theta)} \end{aligned}$$

$\mathcal{L}(Z | \Theta) \propto p(Z | \Theta) \propto p(\Theta | Z)$ but does not necessarily integrate to 1.

Example: AR(1) process

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

Use Markov property

$$\begin{aligned} p(x_1, x_2, \dots, x_T) &= p(x_1)p(x_2 | x_1) \cdots p(x_T | x_{T-1}) \\ &= p(x_1)p(u_2) \cdots p(u_T) \end{aligned}$$

to write log likelihood function

$$\begin{aligned} \ln \mathcal{L}(x^t \mid \Theta) &= -0.5 \sum_{t=2}^T \left[\ln(2\pi) + \ln |\sigma_u^2| + u_t' (\sigma_u^2)^{-1} u_t \right] \\ &\quad + \ln \left[(1 - \rho^2) \sigma_u^2 \right] - \left(\frac{1 - \rho^2}{\sigma_u^2} \right) x_1^2 \end{aligned}$$

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

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

Evaluate $\ln \mathcal{L}(x^t | \Theta)$ 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)
```

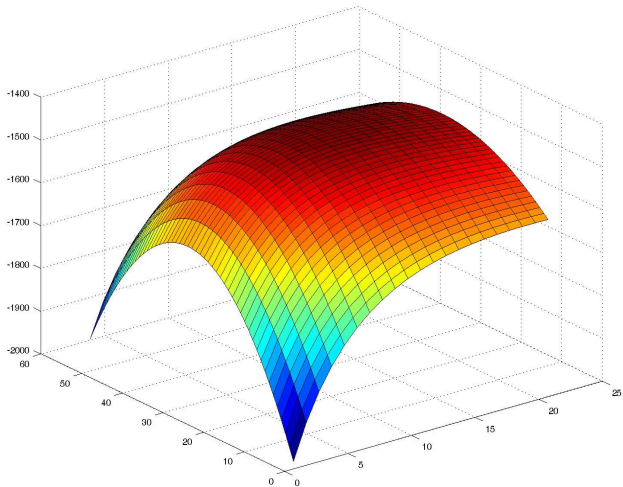


Figure: Estimated posterior densities of structural parameters

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

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

which is a vector which can be approximated element by element

$$\begin{aligned} & \frac{\partial \mathcal{L}(Z | \Theta^{(0)})}{\partial \theta_i} \\ \approx & \frac{\mathcal{L}(Z | \Theta^{(0)}) - \mathcal{L}(Z | \theta_j = \theta_j^{(0)} + \varepsilon : j = i; \theta_j = \theta_j^{(0)} \text{ otherwise})}{\varepsilon} \end{aligned}$$

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

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 $\Theta^{(1)} = \Theta^{(0)} + \mathbf{sg}(\Theta)$
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 too 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

Simulated Annealing Goffe et al (1994)

- ▶ Language is from thermodynamics
- ▶ Combines elements of grid search with (strategically chosen) random movements in the parameter space
- ▶ Has a good record in practice, but cannot be proven to reach global max quicker than grid search.

Simulated Annealing: The Algorithm

Main inputs: $\Theta^{(0)}$, temperature T , boundaries of Θ , temperature reduction parameter r_T (and the function to be max/minimized $f(\Theta)$).

1. $\theta'_j = \theta_j^{(0)} + r \cdot v_j$ where $r \sim U[-1, 1]$ and v_i is an element of the step size vector V .
2. Evaluate $f(\Theta')$ and compare with $f(\Theta^{(0)})$. If $f(\Theta') > f(\Theta^{(0)})$ set $\Theta^{(1)} = \Theta'$. If $f(\Theta') < f(\Theta^{(0)})$ set $\Theta^{(1)} = \Theta'$ with probability $e^{(f(\Theta') - f(\Theta^{(0)}))/T}$ and $\Theta^{(1)} = \Theta^{(0)}$ with probability $1 - e^{(f(\Theta') - f(\Theta^{(0)}))/T}$.
3. After N_s loops through 1 and 2 step length vector V is adjusted in direction so that approx 50% of all moves are accepted.
4. After N_T loops through 1 and 3 temperature is reduced so that $T' = r_T \cdot T$ so that fewer downhill steps are accepted.

Simulated Annealing in MatLab

```
theta=[0.7,.8;]';  
LB=[1,2;]';  
UB=[-1,0;]';  
sa_t= 5; %starting temperature  
sa_rt=.5;  
sa_nt=5;  
sa_ns=20;  
[xopt]=simannb( 'loglikeAR1data', theta, LB, UB,  
sa_t, sa_rt, sa_nt, sa_ns, 1);
```

Code has three components

1. The main program that defines starting values for simulated annealing algorithm etc
2. A function that translates Θ into a state space system
3. A function that evaluates $\mathcal{L}(Z | \Theta)$

Point 2 and 3 are done by `loglikeAR1data.m`

Simulated annealing output:

Initial loss function value:

1.4267e+003

Simulated annealing output:

```
No. of evaluations
41
Current temperature
5
Current optimum function value
1.3994e+003
No. of downhill steps
4
No. of accepted uphill steps
0
No. of rejections
36
Current parameter values
0.6125
0.5288
Current optimum vector
0.6125
0.8805

Elapsed time is 0.327341 seconds.
```

Simulated annealing output:

```
No. of evaluations
401
Current temperature
2.5000
Current optimum function value
1.3906e+003
No. of downhill steps
8
No. of accepted uphill steps
3
No. of rejections
189
Current parameter values
0.5246
0.1151
Current optimum vector
0.5246
0.9565

Elapsed time is 3.162982 seconds.
```

Simulated annealing: Final output

```
No. of evaluations
3201
Current temperature
1.5259e-004
Current optimum function value
1.3904e+003
No. of downhill steps
0
No. of accepted uphill steps
0
No. of rejections
200
Current parameter values
0.5095
0.8301
Current optimum vector
0.5095
0.9467
Elapsed time is 24.990692 seconds.
```

Simulated annealing achieved termination after 3201 evals