Homework 5

Kensuke Suzuki

November 28, 2018

Problem 1: Gaussian Quadrature

In this problem, I rule out u_i from the model. Define the function `llk_wou(Y,X,Z,par,node,method)` which returns (negative of) the log likelihood, given data (`X`, `Y`, and `Z`), parameter vector (`par`), number of node (`node`), and specified integration method (`method`).
In the first problem, I use the Gaussian quadrature method; `method=1` with 20 nodes. Using `qnwnorm( )` included in the CEtools, I draw 20 nodes for β_i from the normal distribution with mean β_0 and variance σ_β^2. This also generates the weighting vector $\mathbf{w}$ which I use later. I pick each draw of β_i, compute the the likelihood for each i, $L_i(\gamma|\beta_i, u_i)$, and stack it up for all draws. Numerical integration is completed by calculating the weighted average of the likelihood using the weights obtained above. Finally take log and sum over all i. Log likelihood is $-1.2571e+03$.

Problem 2: Monte Carlo

In the second problem, I use the Monte Carlo method; `method=2` with 100 nodes. I draw 100 nodes using `haltonNormshuddle( )` provided in the lecture. Analogous to the first problem, for each draw, I compute the likelihood $L_i(\gamma|\beta_i, u_i)$, stack it up for all draws, and compute the simple average. Finally take log and sum over all i. Log likelihood is $-1.2571e+03$.

Problem 3: MLE without u_i using `fmincon`

We use `fmincon` to estimate the parameters. Let parameter vector $\boldsymbol{\theta} = \begin{bmatrix}\gamma_0 & \beta_0 & \sigma_\beta^2\end{bmatrix}'$. We need to impose the parameter restrictions such that $\sigma_\beta^2 \geq 0$. We define $\mathbf{A} = \begin{bmatrix}0 & 0 & -1\end{bmatrix}$ and $b = 0$. When minimizing the negative of the log likelihood over the parameter vector $\boldsymbol{\theta}$, I have a constraint $\mathbf{A}\boldsymbol{\theta} \leq b$. Results are presented below:

Gaussian Quadrature

$$\text{Initial guess: } \begin{bmatrix}\gamma_0\\ \beta_0\\ \sigma_\beta^2\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix}, \quad \text{estimates: } \begin{bmatrix}\hat{\gamma}\\ \hat{\beta}\\ \hat{\sigma}_\beta^2\end{bmatrix} = \begin{bmatrix}-0.5056\\ 2.4832\\ 1.4054\end{bmatrix}, \quad \text{loglikelihood: } -536.2378$$

Monte Carlo

$$\text{Initial guess: } \begin{bmatrix}\gamma_0\\ \beta_0\\ \sigma_\beta^2\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix}, \quad \text{estimates: } \begin{bmatrix}\hat{\gamma}\\ \hat{\beta}\\ \hat{\sigma}_\beta^2\end{bmatrix} = \begin{bmatrix}-0.5056\\ 2.5578\\ 1.1816\end{bmatrix}, \quad \text{loglikelihood: } -536.5876$$

Matlab function `llk_wou( )`

```
function [llf,methodname] = llk_wou(Y,X,Z,par,node,method)
    % compute negative of llf

gamma = par(1);
betanot = par(2);
sigmab = par(3);
%unot = par(4);
unot = 0;
%sigmaub = par(5);
sigmaub = 0;
%sigmau = par(6);
sigmau = 0;

mu = [betanot unot];
Sigma = [sigmab sigmaub; sigmaub sigmau];

ui = 0;


if method == 1
    methodname = 'Gaussian Quadrature';
        % if method is Gaussian Quadrature
    [rcoef,w] = qnwnorm(node, betanot, sigmab);


for i = 1:length(rcoef)
    betai = rcoef(i,1);
        % pick ith draw of beta
    epsi = (betai * X + gamma * Z + ui  ) ;
    logitval = ( 1 + exp(-1 * epsi) ).^(-1);
        % compute the logistic CDF
    lkt = logitval.^Y .* (ones(20,100)-logitval).^(ones(20,100)-Y);
        % compute the contribution of each year
```

```
    lkii(i,:) = prod(lkt);
        % product over years
end
lki = w' * lkii;
    % numerical integration

llki = log(lki);

llf = -1 * sum(llki,2);

elseif method == 2
    methodname = 'Monte Carlo';
        % if method is MC

    norm = haltonNormShuffle(node, 1, 3);
    rcoef = repmat(betanot,node,1) + sigmab * norm';

for i = 1:length(rcoef)
    betai = rcoef(i,1);
        % pick ith draw of beta
    epsi = (betai * X + gamma * Z + ui  ) ;
    logitval = ( 1 + exp(-1 * epsi) ).^(-1);
        % compute the logistic CDF
    lkt = logitval.^Y .* (ones(20,100)-logitval).^(ones(20,100)-Y);
        % compute the contribution of each year
    lkii(i,:) = prod(lkt);
        % product over years
end
lki = sum(1/node * lkii);
    % numerical integration

llki = log(lki);

llf = -1 * sum(llki,2);

end


end
```

Problem 4: MLE of full model using `fmincon`

In this problem, I estimate the full model. Define the function `llk_wu(Y,X,Z,par,node,method)` which returns (negative of) the log likelihood of the full model, given data (`X`, `Y`, and `Z`), parameter vector (`par`), number of node (`node`), and specified integration method (`method`). Since I only invoke Monte Carlo method, `method=2`.
In this function, I draw 100 nodes using `haltonNormshuddle( )`. For this time, I draw β_i and u_i from the bivariate normal distribution with mean $\mu = [\beta_0, u_0]'$ and variance-covariance matrix

$\Sigma = \begin{bmatrix} \sigma_\beta & \sigma_{u\beta} \\ \sigma_{u\beta} & \sigma_u \end{bmatrix}$. I use `chol( )` to make Cholesky decomposition of Σ to simulate from the joint distribution—bivariate normal. Implementation of numerical integration is same as in Problem 2. For optimization, I use `fmincon`. In addition to the nonnegative restrictions on variances, σ_β^2 and σ_u^2, I need to restrict the variance-covariance matrix Σ to be positive definite. Therefore, rather than optimizing over $\sigma_{u\beta}$, I optimize over the correlation coefficient ρ with restriction $-1 \leq \rho \leq 1$ and recover $\sigma_{ub} = \rho\sqrt{\sigma_\beta^2}\sqrt{\sigma_u^2}$. As I have learned in the lecture, I may constrain ρ by imposing upper and lower bound. But since there is no special reason here, I simply write the constraint by linear equation.
Parameter vector over which I optimize is $\boldsymbol{\theta} = \begin{bmatrix} \gamma_0 & \beta_0 & u_0 & \sigma_\beta^2 & \rho & \sigma_u^2 \end{bmatrix}$. Constraints can be expressed by $\mathbf{A}\boldsymbol{\theta} \leq \mathbf{b}$ where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

I will present the initial values and estimated values for $\sigma_{u\beta}$ which is recovered from the ones for ρ. Our initial guess on ρ is 0.9 and estimated values are $\hat{\rho} = 0.4536$.

$$\text{Initial guess: } \begin{bmatrix} \gamma_0 \\ \beta_0 \\ u_0 \\ \sigma_\beta^2 \\ \sigma_{ub} \\ \sigma_u^2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0.9783 \\ 1 \end{bmatrix}, \quad \text{estimates: } \begin{bmatrix} \hat{\gamma} \\ \hat{\beta} \\ \hat{u} \\ \hat{\sigma}_\beta^2 \\ \hat{\sigma}_{ub} \\ \hat{\sigma}_u^2 \end{bmatrix} = \begin{bmatrix} -0.6815 \\ 3.1877 \\ 1.4710 \\ 1.9226 \\ 0.8068 \\ 1.6457 \end{bmatrix}, \quad \text{loglikelihood: } -464.0001$$

Matlab function `llk_wu( )`

```
function [llf ,methodname] = llk_wu(Y,X,Z,par ,node ,method)
    % compute negative of llf

gamma = par(1);
betanot = par(2);
sigmab = par(3);
unot = par(4);
%sigmaub = par(5);
rho = par(5);
sigmau = par(6);

sigmaub = sigmab^(1/2) * sigmau^(1/2) * rho;

mu = [betanot unot];
Sigma = [sigmab sigmaub; sigmaub sigmau];
U = chol(Sigma);

```

```
if method == 2
    methodname = 'Monte Carlo';
        % if method is MC

    norm = haltonNormShuffle(node, 2, 2);
    rcoef = repmat(mu,node, 1) + (U' * norm)';

for i = 1:length(rcoef)
    betai = rcoef(i,1);
    ui = rcoef(i,2);
        % pick ith draw of beta
    epsi = (betai * X + gamma * Z + ui   ) ;
    logitval = ( 1 + exp(-1 * epsi) ).^(-1);
        % compute the logistic CDF
    lkt = logitval.^Y .* (ones(20,100)-logitval).^(ones(20,100)-Y);
        % compute the contribution of each year
    lkii(i,:) = prod(lkt);
        % product over years
end
lki = sum(1/node * lkii);
    % numerical integration

llki = log(lki);

llf = -1 * sum(llki,2);

end


end
```

Matlab Main Code

```
% Empirical Method HW5 %
% Ken Suzuki (Penn State)
% kxs974@psu.edu

clear all
delete HW5log.txt
diary('HW5log.txt')
diary on

% Load Data
load('hw5.mat')

X = data.X;
Y = data.Y;
Z = data.Z;
```

```

addpath('../CEtools/');

%% Problem 1
% parameter value
betanot = 0.1;
sigmab = 1;
gamma = 0;

% method
method = 1;

% number of nodes
node = 20;

% set parameter vector
par = [gamma betanot sigmab];

[llf,methodname] = llk_wou(Y,X,Z,par,node,method);

llf = -1 * llf;
% display result
disp('Problem 1')
disp(methodname)
disp('Loglikelihood is')
disp(llf)

%% Problem 2
% parameter value
betanot = 0.1;
sigmab = 1;
gamma = 0;

% method
method = 2;

% number of nodes
node = 100;

% set parameter vector
par = [gamma betanot sigmab];

[llf,methodname] = llk_wou(Y,X,Z,par,node,method);
llf = -1 * llf; % take negative

% display result
disp('Problem 2')
disp(methodname)
```

```
disp('Loglikelihood is')
disp(llf)

%% Problem 3

clear par

% number of node
node = 20;

% method: GC
method = 1;

% define function to be minimzied (function of par)
llkwou_min = @(par) llk_wou(Y,X,Z,par,node,method);

% fmincon

% for constraint
A = [0, 0, -1];
b = 0

%[paraGQ, lfGQ] = fminsearch(llkwou_min, [1 1 1] );
[paraGQ, lfGQ] = fmincon(llkwou_min, [1; 1; 1], A, b );
lfGQ = -1 * lfGQ;

% number of node
node = 100;

% method: GC
method = 2;

% define function to be minimzied (function of par)
llkwou_min = @(par) llk_wou(Y,X,Z,par,node,method);

% fminsearch
[paraMC, lfMC] = fmincon(llkwou_min, [1; 1; 1], A, b );
lfMC = -1 * lfMC;


% display result
disp('Problem 3-1 (Gaussian Quadrature)')
disp('Initial guesses are')
disp('   gamma        beta       sigmab')
disp([1 1 1])
disp('   gamma        beta       sigmab')
disp(paraGQ')
disp('Maximized log-likelihood is:')
```

```
disp(lfGQ)


% display result
disp('Problem 3-2 (Monte Carlo)')
disp('Initial guesses are')
disp('    gamma         beta       sigmab')
disp([1 1 1])
disp('    gamma         beta       sigmab')
disp(paraMC')
disp('Maximized log-likelihood is:')
disp(lfMC)


%% Problem 4

clear A
clear b

% method: MC
method = 2;

% number of node
node = 100;

% initial values for parameter vector
gamma = -0.5056;
betanot = 2.5579;
sigmab = 1.1816;
unot = 1;
%sigmaub = 0.9;
rho = 0.9;
sigmau =1;
%intpar = [gamma betanot sigmab unot sigmaub sigmau];
intpar = [gamma betanot sigmab unot rho sigmau];

%define function to be minimized
llkwu_min = @(par) llk_wu(Y,X,Z,par,node,method);

% for constraint
A = [0 0 -1 0 0  0 ; ...
     0 0  0 0 0 -1; ...
     0 0  0 0 -1 0; ...
     0 0  0 0  1 0];
b = [0; 0;  1; 1];

% fmincon
[paraMC, lfMC] = fmincon(llkwu_min, intpar', A, b );
```

```
sigmaub = paraMC(3)^(1/2) * paraMC(6)^(1/2) * paraMC(5);
paraMC_cov = paraMC;
paraMC_cov(5) = sigmaub;

sigmaubint = intpar(3)^(1/2) * intpar(6)^(1/2) *intpar(5);
intpar_cov = intpar;
intpar_cov(5) = sigmaubint ;

lfMC = -1 * lfMC;

% display result
disp('Problem 4 (Monte Carlo)')
disp('Initial guesses are')
disp('    gamma        betanot    sigmab     unot        sigmaub    sigmau')
disp(intpar_cov)
disp('Estimated parameters')
disp('    gamma        betanot    sigmab     unot        sigmaub    sigmau')
disp(paraMC_cov')
disp('Maximized log-likelihood is:')
disp(lfMC)

diary off
```