5.16 Bootstrap GMM　　0.01 暫定版

ここでは Bootstrap の応用例の１つである Bootstrap GMM についてプログラムしていきます。これは、GMM の t テストや Hansen テストなどにおいて、データの大きさにかかわらず、一般にその帰無仮説を棄却しやすくなるという特性を修正するために Bootstrap を導入するものです。やり方は、極めて簡単明瞭で、GMM をやる前にデータの段階でペアワイズ（ケースワイズ）に Bootstrap して、それぞれの Bootstrap サンプルに対する GMM による Estimates を求め、それにより通常の Bootstrap の扱いと同じように区間推定や仮説検定を修正しようというものです。ただし、簡単といっても、やり方がケースワイズに Bootstrap サンプルを作成し、それぞれの GMM Estimates を求めるわけですから、収束計算では共線性その他の理由により収束しないケースもでてくることもあり、そのままではいつもこの方法が利用できるとはかぎりません。

以下では、まずΓ分布の GMM 推定をしてみます。これは Lin[2001]12.1 を replicate したものです。まず、２ステップで係数だけを Grid Search で正確に求めてみます。通常は Newey-West タイプまたは Andrews タイプの Covariance 行列を求めるアルゴリズムが必要ですが、この推定ではそうした系列相関は考えないものとしています。

プログラム

```
new; cls;
let data[20,2]=
      20.5     12
      31.5     16
      47.7     18
      26.2     16
      44.0     12
      8.28     12
      30.8     16
      17.2     12
      19.9     10
      9.96     12
      55.8     16
      25.2     20
      29.0     12
      85.5     16
      15.1     10
      28.5     18
```

```
        21.4    16
        17.7    20
        6.42    12
        84.9    16
        ;
/* The data set above comes from Greene[1999] Example 11.4 */
/* This program replicates Lin[2001] Lesson 12.1 by GAUSS itself*/
x=data[.,1]/10;
start={0.01,0.01}; finish={10,3};
w=1;
b=g5search(&mom,start,finish);
print "Step 1: b=" b;
w=invpd(gmmv(b));
b=g5search(&mom,start,finish);
print "Step 2: b=" b;

proc gmmv(b);
    local rho,lambda,n,m,v,w;
    rho=b[1]; lambda=b[2];
    n=rows(x);
    m=zeros(n,4);
    m[.,1]=x-rho/lambda;
    m[.,2]=x^2-rho*(rho+1)/(lambda^2);
    m[.,3]=ln(x)-gradp(&lngam,rho)+ln(lambda);
    m[.,4]=1/x-lambda/(rho-1);
    m=m/n;
    v=m'm;
    retp(v);
endp;

proc mom(b);
    local rho,lambda,n,m;
    rho=b[1]; lambda=b[2];
    n=rows(x);
    m=zeros(n,4);
    m[.,1]=x-rho/lambda;
```

```
    m[.,2]=x^2-rho*(rho+1)/(lambda^2);
    m[.,3]=ln(x)-gradp(&lngam,rho)+ln(lambda);
    m[.,4]=1/x-lambda/(rho-1);
    m=meanc(m);
    retp(m'*w*m);
endp;

proc gmmm(b);
    local rho,lambda,n,m;
    rho=b[1]; lambda=b[2];
    n=rows(x);
    m=zeros(n,4);
    m[.,1]=x-rho/lambda;
    m[.,2]=x^2-rho*(rho+1)/(lambda^2);
    m[.,3]=ln(x)-gradp(&lngam,rho)+ln(lambda);
    m[.,4]=1/x-lambda/(rho-1);
    m=meanc(m);
    retp(m);
endp;

fn lngam(x)=ln(gamma(x));

proc g5search(&g,start,finish);
    local b,e,g:proc;
    print "1st Round";
    b=gsearch(&g,start,finish);
    e=0.5;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
    print "2nd Round";
    b=gsearch(&g,start,finish);
    e=0.1;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
    print "3rd Round";
    b=gsearch(&g,start,finish);
    e=0.01;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
```

```
    print "4th Round";
    b=gsearch(&g,start,finish);
    e=0.001;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
    print "5th Round";
    b=gsearch(&g,start,finish);
    retp(b);
endp;

proc gsearch(&f,start,finish);
    local iter,length,xstart,ystart,xstep,ystep,x,y,xmax,ymax,xend,yend,grid,k,i,j,f:proc;
    local fmax,temp,tol;
    iter=100; length=99; tol=1e-10;
    xstart=start[1]; ystart=start[2]; xend=finish[1]; yend=finish[2];
    fmax=0;
    k=1;
    do while k<=iter;
        print /rz "# of iterations=" k;;
        xstep=(xend-xstart)/length; ystep=(yend-ystart)/length;
        grid=(-1e256).*ones(length+1,length+1);
        x=xstart;
        i=1;
        do while i<=length+1;
            y=ystart;
            j=1;
            do while j<=length+1;
                grid[i,j]=f(x|y);
                j=j+1;
                y=y+ystep;
            endo;
            i=i+1;
            x=x+xstep;
        endo;
        xmax=xstart+(minindc(minc(grid'))-1)*xstep;
        ymax=ystart+(minindc(minc(grid ))-1)*ystep;
        temp=f(xmax|ymax);
```

```
        if abs(fmax-temp)<tol;
            break;
        endif;
        fmax=temp;
        print "            f=" fmax;
        xstart=xmax-xstep; xend=xmax+xstep;
        ystart=ymax-ystep; yend=ymax+ystep;
        k=k+1;
    endo;
    print "            f=" fmax;
    retp(xmax|ymax);
endp;
```

画面表示

```
Step 1: b=
         2.2986424
        0.72227724
Step 2: b=
         3.5326904
         1.3076837
```

上のように、ステップ１での係数は Lin[2001]の結果の 2.3691 および 0.74112 に若干ズレが生じており、ステップ２での最終結果の 2.8971 および 0.8484 とはかなりのくるいが生じています。実際、上のプログラムでwに相当する部分を Lin[2001]のステップ１の結果の係数を与えて計算したものを用いても上のステップ２に近いような結果が得られます。また、そもそもの GPE２用に上のプログラムを若干書きなおすと、Lin[2001]の結果と１桁も違わない結果が replicate できます。したがって、GPE2 はこの場合のような複雑な表面の最小化には正確ではないソフトウエアと言わざるをえません。通常の Log-Likelihod の最大化では、そうは問題になることはないのですが、GPE2 に限らず、GAUSS の正規のライブラリである Maxlik や CML などを用いても、Criterion Function の最小化には特に細心の注意が必要です。

上の推定では、GMM 推定値として２つのパラメータからなる $\theta=(\lambda,\rho)$を考えます。その上で、４つのサンプルモーメント関数として、

$m1(\lambda,\rho)=1/N\ \Sigma[Xi-\rho/\lambda]$

$m2(\lambda,\rho)=1/N\ \Sigma[Xi^2-\rho(\rho+1)/\lambda^2]$

$m3(\lambda,\rho)=1/N\ \Sigma[\ln(Xi)-d\ln\Gamma(\rho)/d\rho+\ln(\lambda)]$

$m4(\lambda,\rho)=1/N\ \Sigma[1/Xi-\lambda/(\rho-1)]$

を考えて、初期値のWにもとづいて（ここではＩ行列、または同じことであるが１）、
Criterion Function：　Q(θ)＝m(θ)'W m(θ)
を最小化するθを求める。ここでは、θはλとρの２パラメータである。そこまでが第 1 ステップである。次に、そのθを用いて（上の計算ではｂ）、w=invpd(gmmv(b));というところで、$W=W(\theta^0)=[Var(m(\theta^0))]^{-1}$ を求めることで、この求められた W を使って上の Criterion Function を再び最小化するパラメータθを都合２ステップで求めている。

なお、目的関数の最小化を推定するアルゴリズムとしては、３節の最適化の章で行なった簡単な Grid Search をさらに改良して初期値をアップデートしていく５重のグリッドサーチをほどこしている。最終２桁については、区間の切り方と実際の計算で用いられる桁との差異から若干バックワードに iteration がなるところもあるが、微分を用いたアルゴリズムよりも、この方がより小さな Criterion Function を見つけることができます。（なお、アルゴリズム的には、そのラウンドの最小値とパラメータの組を保存しておいて、その中の最小値を見つけることによってバックワードさせない方法も簡単に付け加えられるのだが、Light 版では動かなくなってしまうのでそのままにしている。）

次に、このΓ分布の GMM 推定にｔテストと Hansen テストを付け加えた上で、さらにデータを Bootstrap させてこれらのテストをやってみます。まずは、ｔテストと Hansen テストの procedure を付け加えます。

```
proc ttest(b);
    local m,v,g,varb,se,t;
    /* This procedure requires gmmm and gmmv to get m and v. */
    m=gmmm(b);
    v=gmmv(b);
    g=gradp(&gmmm,b);
    varb=invpd(g'*w*g)*g'*w*v*w'*g*invpd(g'*w*g); /* w is also given outside. */
    se=sqrt(diag(varb));
    t=b./se;
    print/lz "                    b                    SE                    t";;
    print b~se~t;
    retp(t);
endp;

proc hansen(b);
    local m,v,q,df;
    /* This procedure requires gmmm and gmmv to get m and v. */
    m=gmmm(b);
    v=gmmv(b);
```

```
    q=m'*invpd(v)*m;
    df=rows(m)-rows(b);
    print "Hansen Test for the Moment Restrictions";
    print "Chi2=" q;; print/rz "(df=" df ")"; ; print "P-value=" cdfchic(q,df);
    retp(q);
endp;
```

上の２つの procedure を一番最初に示したプログラム中のどこかに置いて、冒頭の Step 2 の print 文の後あたりに、

```
    call ttest(b);
    call hansen(b);
```

として呼び出せば、関係する係数、SE、それに t 値が表示され、Hansen テストの方には自由度とともにχ2 の値が示されます。

結果表示

```
                b                SE                t
        3.5326904      0.0033993216         1039.2339
        1.3076837        0.16759630         7.8025806
Hansen Test for the Moment Restrictions
Chi2=        2.4783708 (df=                2 )P-value=       0.28962004
```

さらに、これをデータレベルで Bootstrap するには、data の入力の後の冒頭部分で

```
        data=data[.,1]/10;
        n=rows(data);
        index=ceil(n*rndu(n,1));
        x=data[index];
```

というふうにランダムな index を作成しておいてから、もとになるデータをその index にもとづいてシャッフルして、ここで使う Bootstarp サンプルｘを作成してやります。これを任意の回数 do while ループで繰り返してやれば、Bootstrap が基本的になされます。

プログラム（所要約１日）

```
new; cls;
let data[20,2]=
      20.5      12
      31.5      16
      47.7      18
      26.2      16
      44.0      12
      8.28      12
```

```
        30.8    16
        17.2    12
        19.9    10
        9.96    12
        55.8    16
        25.2    20
        29.0    12
        85.5    16
        15.1    10
        28.5    18
        21.4    16
        17.7    20
        6.42    12
        84.9    16
        ;
/* The data set above comes from Greene[1999] Example 11.4 */
data=data[.,1]/10;
x=data;
start={0.01,0.01}; finish={10,3};
w=1;
b=g5search(&mom,start,finish);
print "Step 1: b=" b;
w=invpd(gmmv(b));
b=g5search(&mom,start,finish);
print "Step 2: b=" b;
t1=ttest(b);
h1=hansen(b);

rndseed 1000;
nboot=200;
i=1; tboot=zeros(nboot,rows(b)); hboot=zeros(nboot,1);
do while i<=nboot;
    print;
    print/lz "Bootstrap" i;
    n=rows(data);
    index=ceil(n*rndu(n,1));
```

```
    x=data[index];
        w=1;
        b=g5search(&mom,start,finish);
        w=invpd(gmmv(b));
        b=g5search(&mom,start,finish);
        tboot[i,.]=ttest(b)';
        hboot[i]=hansen(b);
    i=i+1;
endo;
i=1;
do while i<=rows(b);
    tboot[.,i]=sortc(tboot[.,i],1);
    i=i+1;
endo;
tboot=tboot';
hboot=sortc(hboot,1);
print "Bootstrap t test     (One-tail value below)";
tcrit=tboot[.,ceil(nboot*0.99)]~tboot[.,ceil(nboot*0.95)]~tboot[.,ceil(nboot*0.9)];
print "              abs(t)                1%                5%                10%";;
print t1~tcrit;
print "Boostrap mean of t" meanc(tboot');
print "Bootstrap Hansen test";
hcrit=hboot[ceil(nboot*0.99)]~hboot[ceil(nboot*0.95)]~hboot[ceil(nboot*0.9)];
print "              chi2                1%                5%                10%";;
print h1~hcrit;
print "Bootstrap Mean of Hansen statistic" meanc(hboot);

proc gmmv(b);
    local rho,lambda,n,m,v,w;
    rho=b[1]; lambda=b[2];
    n=rows(x);
    m=zeros(n,4);
    m[.,1]=x-rho/lambda;
    m[.,2]=x^2-rho*(rho+1)/(lambda^2);
    m[.,3]=ln(x)-gradp(&lngam,rho)+ln(lambda);
    m[.,4]=1/x-lambda/(rho-1);
```

```
    m=m/n;
    v=m'm;
    retp(v);
endp;

proc mom(b);
    local rho,lambda,n,m;
    rho=b[1]; lambda=b[2];
    n=rows(x);
    m=zeros(n,4);
    m[.,1]=x-rho/lambda;
    m[.,2]=x^2-rho*(rho+1)/(lambda^2);
    m[.,3]=ln(x)-gradp(&lngam,rho)+ln(lambda);
    m[.,4]=1/x-lambda/(rho-1);
    m=meanc(m);
    retp(m'*w*m);
endp;

proc gmmm(b);
    local rho,lambda,n,m;
    rho=b[1]; lambda=b[2];
    n=rows(x);
    m=zeros(n,4);
    m[.,1]=x-rho/lambda;
    m[.,2]=x^2-rho*(rho+1)/(lambda^2);
    m[.,3]=ln(x)-gradp(&lngam,rho)+ln(lambda);
    m[.,4]=1/x-lambda/(rho-1);
    m=meanc(m);
    retp(m);
endp;

fn lngam(x)=ln(gamma(x));

proc ttest(b);
    local m,v,g,varb,se,t;
    m=gmmm(b);
```

```
    v=gmmv(b);
    g=gradp(&gmmm,b);
    varb=invpd(g'*w*g)*g'*w*v*w'*g*invpd(g'*w*g); /* w is given outside. */
    se=sqrt(diag(varb));
    t=b./se;
    print/lz "                b                SE                t";;
    print b~se~t;
    retp(t);
endp;

proc hansen(b);
    local m,v,q,df;
    m=gmmm(b);
    v=gmmv(b);
    q=m'*invpd(v)*m;
    df=rows(m)-rows(b);
    print "Hansen Test for the Moment Restrictions";
    print "Chi2=" q;; print/rz "(df=" df ")";; print "P-value=" cdfchic(q,df);
    retp(q);
endp;

proc g5search(&g,start,finish);
    local b,e,g:proc;
    print "1st Round";
    b=gsearch(&g,start,finish);
    e=0.5;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
    print "2nd Round";
    b=gsearch(&g,start,finish);
    e=0.1;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
    print "3rd Round";
    b=gsearch(&g,start,finish);
    e=0.01;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
    print "4th Round";
```

```
    b=gsearch(&g,start,finish);
    e=0.001;
    start=(b[1]-e*b[1])|(b[2]-e*b[2]); finish=(b[1]+e*b[1])|(b[2]+e*b[2]);
    print "5th Round";
    b=gsearch(&g,start,finish);
    retp(b);
endp;

proc gsearch(&f,start,finish);
    local iter,length,xstart,ystart,xstep,ystep,x,y,xmax,ymax,xend,yend,grid,k,i,j,f:proc;
    local fmax,temp,tol;
    iter=100; length=99; tol=1e-10;
    xstart=start[1]; ystart=start[2]; xend=finish[1]; yend=finish[2];
    fmax=0;
    k=1;
    do while k<=iter;
    print /rz "# of iterations=" k;;
    xstep=(xend-xstart)/length; ystep=(yend-ystart)/length;
    grid=(-1e256).*ones(length+1,length+1);
        x=xstart;
        i=1;
        do while i<=length+1;
            y=ystart;
            j=1;
            do while j<=length+1;
                grid[i,j]=f(x|y);
                j=j+1;
                y=y+ystep;
            endo;
            i=i+1;
            x=x+xstep;
        endo;
        xmax=xstart+(minindc(minc(grid'))-1)*xstep;
        ymax=ystart+(minindc(minc(grid ))-1)*ystep;
        temp=f(xmax|ymax);
        if abs(fmax-temp)<tol;
```

```
            break;
        endif;
        fmax=temp;
        print "            f=" fmax;
        xstart=xmax-xstep; xend=xmax+xstep;
        ystart=ymax-ystep; yend=ymax+ystep;
        k=k+1;
    endo;
    print "            f=" fmax;
    retp(xmax|ymax);
endp;
```

上では Grind Search で、細かく Criterion Function の最小値を与えるパラメータを調べることを nboot=200;として 200 回繰り返して、それぞれの Bootstrap サンプルに対するｔ値と Hansen テストの値を tboot と hboot の 200×１の列ベクトルに格納しておいて、最初の Bootstrap をしないもともとのデータのｔ値の絶対値と Hansen テストの値を表示した横にそれぞれの上位１、５、10%の Bootstrap 統計量を上で格納した列から計算して、表示させています。なお、この場合のｔ値の表示は片側の値の表示に便宜上してあります。両側が必要ならば、１、5、10%を２で割ってその番目の値を表示させるようにしてください。

画面表示

```
Bootstrap t test     (One-tail value below)
          abs(t)              1%              5%             10%
       1039.2339       4831.5558       3029.5486       2182.9714
       7.8025806       20.553320       15.009164       13.227915
Boostrap mean of t
       1063.9087
       9.4279685

Bootstrap Hansen test
          chi2              1%              5%             10%
       2.4783708       18.277234       15.610026       13.481645
Bootstrap Mean of Hansen statistic          5.8085277
```

なお、上では便宜上、ｔ値の Bootstrap Mean と１%、５%、10%の値を示していますが、読者の関心にあわせて、Bootstrap の章で行なったのと同じように、補正されたｔ値を求めたり、そもそもの計算で使われる s.e.を変更したりするのに Bootstrap を使うこともできま

す。様々な使い方ができます。上の計算方法はその一面を捉えた一例にしかすぎません。

技術的補足と実践的な展開

この章は未完成です。なぜなら、上の例は２つのパラメータを求める計算を Grid Search で行なっているにすぎないからです。通常の３つ以上のパラメータを求める計算にも、同様に、Grid Search を用いなければ、特に２ステップの GMM の解法のケースでは一般に解が相当にズレます。GAUSS の最新の Maxlik と Matlab の両方によって、だいたい同じ解が求められたのでよいという問題ではけしてありません。Criterion Function を最小化する GMM の解法では、他の Log-Likelihood を最大化する非線形問題よりもまして、最適化には細心の注意が必要です。３変数以上の多変数の Grid Search が、技術的には必須であると私は考えます。特にこのことは、Bootstrap してみるとその必要性が強く再認識されることになります。つまり、時として解が通常の微分を使った方法では求められないという事態が生じるのです。それを If 文で取り除くことも可能でしょう、しかしながらそれは本来の Bootstrap の考え方から乖離してしまいます。「たまたま、ある一定の初期値から解が見つかる」あるいは「初期値によって、解が異なることもある」では困るのです。理論的には、近年日本人研究者を中心にして GMM は発展を見せました。しかしながら、実証的には、微分系の最適化アルゴリズムで、危ういことをやっている今の現実には深い憂慮を示さざるをえません。いずれにせよ、Multi-dimensional Grid Search の技術が早急に必要となります。