4.8 特殊関数とライブラリ Optmum と CO との対応関係 工事中

MaklikとOptmumの対応関係

簡単に言うと、Optmumは目的関数の最小化のみの数学機能として働きます。Maklikはその名前からも明らかなようにLikelihoodの最大化計算に特化していて付随する各種統計や行列を計算してくれる関数やオプションが数多く用意されているものです。したがって、Likelihoodを最大化するべきときはMaxlikを使うべきです。また、Maxlikの方の開発がOptmumよりも発展した形となっているので、Likelihoodの計算に限らず最大化最小化どちらの計算にも用いることができます。古いプログラムはOptmumに頼っていることが多々ありますが、新規にプログラムするときは、Maxlikに統一すべきです。両者を交互して用いることができる力を養うことは、特に大学者のプログラムを模倣する際に重要となります。

	Maklik	Optmum
手続き	library maklik; maxset; start={1,1}; call maxlik(data,0,&f,start);	library optmum; optset; start={1,1}; call optmum(&f,start);
目的関数	proc f(b,data); . . . . retp(最大化すべき値またはベクトル); endp;	proc f(b); . . . . retp(最小化すべき値); endp;

以下では残差平方和を最小化してパラメータを求めてみます。

プログラム

```
new; cls;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];
library maxlik,optmum;
/* Maxlik */
    maxset;
    start={1,1};
    call maxlik(data,0,&f1,start);
```

```
/* Optmum */
    optset;
    start={1,1};
    call optmum(&f2,start);


    proc f1(b,data);
        local y, x, e;
        y=data[.,1];
        x=ones(28,1)~data[.,2];
        e=y-x*b;
        retp(-e'e);       @ This could be a vector of -e^2 . @
    endp;

    proc f2(b);
        local y, x, e;
        y=data[.,1];
        x=ones(28,1)~data[.,2];
        e=y-x*b;
        retp(e'e);
    endp;
```

これを通常のLikelihoodの最大化計算にして、{　　}=の形でアウトプットを利用できる形にすれば、次のようになります。目的関数の名前やアウトプットの各変数の名前は何でもよいのですが、アウトプットのうちの１つにｆという変数があれば、その他では使えなくなりますから、目的関数はｆのままではいけません。また、上では残差平方和を最小化したかったので、Maxlikの方の目的関数はマイナス値で、Optmumの方の目的関数はそのまま正の値を用いましたが、今度はLikelihoodの「最大化」ですから符号関係は逆転します。すなわち、Maxlikはその名の通りLikelihoodの最大化ですからそのままの値で、Optmumの方の目的関数はマイナスをつけて、それを最小化します。すなわち、最大化することになるのです（厳密にはこの通りですが、たとえ符号関係を間違えても、時間はかかる傾向にありますが、一応収束して極値を計算してたいていは同じ結果になります）。

Maklik	Optmum
{x,f,g,cov,retcode}=maxprt(maxlik(data,0,&ll,start));	
または	{x,f,g,retcode}=optmum(&ll,start);
{x,f,g,cov,retcode}=maxlik(data,0,&ll,start);	
目的関数	
proc ll(b,data);	proc ll(b);
．．．．	．．．．
retp(最大化すべき値またはベクトル);	retp(最小化すべき値);
endp;	endp;

Optmum→Maxlik**へのプログラムの変更方法**

１）library optmum; → library maxlik;

optset; → maxset;

ただし、pgraphなど他のライブラリがカンマで区切ってついていることもある

２）{x,f,g,retcode}=optmum(&ll,start); →

{x,f,g,cov,retcode}=maxprt(maxlik(data,0,&ll,start));

ポインタ＆の後の目的関数の名前やインプットおよびアウトプットの変数名は上と異なっても構わない（ただし、ｘやｆ、ｇといった変数が重なっていないか注意）

３）proc ll(b); → proc ll(b,data); と２値のインプット関数に変更

４）最後に下線から始まるグローバル変数をマニュアルを見て適宜変更する（わからない場合は、@と@または/* */で囲んで無効にしてまう）

上の方法は最適化のアップグレードの方法だが、反対にリソースが限られている場合にはダウングレードも可能である。ただし、収束計算に成功する確率は下がる。しかしながら、教育的には有効かもしれない。

Optmum→**標準関数**QNewton**へのプログラムの変更方法**

１）library optmum; → 消去

optset; → qnewtonset;

２）{x,f,g,retcode}=optmum(&ll,start); → {x,f,g,retcode}=QNewton(&ll,start);

３）proc ll(b); → そのまま

４）グローバル変数は消去または@と@または/* */で囲んで無効にする。関係なし。

ただし、この方法はBFGSのみによる最適化である。他方法のオプションは用意されていない。しかしながら、Optmumで書かれたプログラムにグローバル変数の設定がついていない場合には、この方法でも動く可能性は高い。

条件付ではない最適化問題では、この他、全章で扱った２値関数のdataとパラメータの順序を逆転させた目的関数をestimateという関数で最適化するGPE2の方法もある。この場合には、いくつかの収束計算のオプションが用意されている。１値または２値のどちらの目的関数を最適化することに気をつければ､Web上に公開されている最適化アルゴリズムによってその関数を最適化することも可能である。ただし、その場合には、その最適化アルゴリズムはプロトタイプであることに留意すべきである。パラメータの数が定数項があればそれも含めて２個または１個の場合の計算には、１値の目的関数を最大化または最小化するGrid Searchがどのようなアルゴリズムよりも確実である。

CMLとCOの対応関係

同様にして、条件付最適化のケースもCMLとCOの間に同じような関係があります。新規にプログラムする場合にはCMLを使うべきです。特に、CMLはGAUSSのライブラリの中でも芸術の域に達したものと言えます。

CML	CO