R と Matlab による最尤推定のコードの作成†

1. 最尤法とは?

簡単に言うと尤度関数を最大にするように未知パラメーターの値を決める事。以下では観測されたデータを$\{y_n, \ldots, y_2, y_1\}$とし、そのベクトルを $\mathbf{Y} = [y_1,\ldots,y_n]'$、未知パラメーターのベクトルを $\boldsymbol{\theta} = [\theta_1,\ldots,\theta_q]'$とする。また尤度関数を $L(\theta)$と表すとする(尤度関数は未知パラメーターの関数)。

(1) データ(の分布)が連続型の場合

n 個の確率変数$\{y_n, \ldots, y_2, y_1\}$の同時密度関数 $f(y_n, \ldots, y_2, y_1;\boldsymbol{\theta})$ を未知パラメーターの関数とみなしたものが尤度関数。つまり

$$L(\boldsymbol{\theta}) = f(y_n, \ldots, y_2, y_1;\boldsymbol{\theta})。$$

これを最大化するようなパラメーター$\boldsymbol{\theta}$ の値が最尤推定値。通常、尤度関数ではなくその対数をとった対数尤度関数、$\log L(\boldsymbol{\theta})$、を最大化するようなパラメーターを求める(そちらの方が計算が簡単なので)。

(例) $y_i, i=1,\ldots,n$ は独立に平均 μ、分散 σ^2 の正規分布に従っているとする。この場合、推定したい未知パラメーターは $\boldsymbol{\theta}= [\mu, \sigma^2]$。これを最尤法で推定する。まず y_i の密度関数は

$$f(y_i;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(y_i-\mu)^2}{2\sigma^2}\right]$$

であり、さらに$y_i, i=1,\ldots,n$は独立なので、$\{y_n, \ldots, y_2, y_1\}$の同時密度関数は$y_i$それぞれの密度関数を掛け合わせたもの、すなわち、

$$f(y_n,...,y_1;\mu,\sigma^2) = f(y_n;\mu,\sigma^2)\cdots f(y_2;\mu,\sigma^2)f(y_1;\mu,\sigma^2)$$

である。これより、対数尤度関数は

$$\begin{aligned}
\log L(\theta) &= \log f(y_n,...,y_1;\mu,\sigma^2) \\
&= \log f(y_n;\mu,\sigma^2) + \cdots + \log f(y_2;\mu,\sigma^2) + \log f(y_1;\mu,\sigma^2) \\
&= \sum_{i=1}^{n} \log f(y_i;\mu,\sigma^2) \\
&= \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(y_i-\mu)^2}{2\sigma^2}\right]\right) \\
&= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2
\end{aligned}$$

となる。ちなみにこれを最大化する μ と σ^2 の値は

$$\hat{\mu}_{MLE} = n^{-1}\sum_{i=1}^{n} y_i\ , \qquad \hat{\sigma}^2_{MLE} = n^{-1}\sum_{t=1}^{n}(y_i-\mu_{MLE})^2$$

---

†この資料は私のゼミおよび講義で R の使用法を説明するために作成した資料です。ホームページ上で公開しており、自由に参照して頂いて構いません。ただし、内容について、一応検証してありますが、間違いがあるかもしれません。間違いがあった場合でもそれによって生じるいかなる損害、不利益について責任は負いかねますのでご了承ください。

である。

(2) データ(の分布)が離散型の場合

$\{y_n, \ldots, y_2, y_1\}$ の同時確率関数を $p(y_n,\ldots,y_2,y_1;\boldsymbol{\theta})$とするとこれをパラメーターの関数として見たものが尤度関数。

(例) $y_i, i=1,\ldots,n$ はそれぞれ独立に確率 α $(0<\alpha<1)$で 1 を取るベルヌーイ分布に従っているとする。この場合、推定したい未知パラメーターは $\boldsymbol{\theta}=\alpha$。これを最尤法で推定する。まず y_i の確率関数は

$$p(y_i;p)=\alpha^{y_i}(1-\alpha)^{1-y_i}$$

であり、さらに$y_i, i=1,\ldots,n$は独立なので、$\{y_n, \ldots, y_2, y_1\}$の同時確率関数は$y_i$のそれぞれの確率関数を掛け合わせたもの、すなわち、

$$p(y_n,\ldots,y_1;\alpha)=p(y_n;\alpha)\cdots p(y_2;\alpha)p(y_1;\alpha)$$

である。これより、対数尤度関数は

$$\begin{aligned}\log L(\theta) &= \log p(y_n,\ldots,y_1;\alpha)\\ &= \log p(y_n;\alpha)+\cdots+\log p(y_2;\alpha)+\log p(y_1;\alpha)\\ &= \sum_{i=1}^{n}\log p(y_i;\alpha)\\ &= \log\alpha\sum_{i=1}^{n}y_i+\log(1-\alpha)\sum_{i=1}^{n}(1-y_i)\end{aligned}$$

となる。

最尤法はモデルが複雑になると尤度関数、すなわち同時密度関数を求めるのが難しくなるが、それ以外は考え方はどの場合も上記とまったく同じ。対数尤度関数さえ求まれば(もっというとパラメーターの関数として対数尤度関数を計算するやり方さえ分かれば、明示的に求まらなくてもいい)、最尤法で推定できる。

2. Matlab による最尤法のプログラム

Matlab は自分で関数をつくる事ができる。その自作した関数に対して “fminunc”関数を用いて、その関数の値を最小(大)化する変数(パラメーター)の値を求める。

例として、先ほどの正規分布の平均 μ と分散 σ^2 を推定するプログラムの書き方を説明する。これには関数ファイルを作る必要があるのでまず関数ファイルについて簡単に説明する。例えば

$$y=f(x),\quad f(x)=3x^2$$

という関数のファイルを作ってみよう。まず、スクリプトファイルを新しく開く(m-file と呼ばれる)。そこに

```
function    y = func(x)
 y = 3*x^2;
```

と書いて、func.m という名前でどこかわかりやすい場所に保存する(m ファイルの拡張子は.m)。これで 新たに “func” という関数が Matlab で使えるようになった。一般に

```
function　出力変数名 = 関数名(入力変数名)
出力変数名 = 具体的な関数の形
```

というように関数を記述する。

この関数を使うには Matlab のコマンドウィンドウで「パスの設定」をクリックし、もし検索パスのところに、先ほどの func.m を置いたディレクトリがなければ、「フォルダーを追加」でそのディレクトリを追加する(追加したら「保存」、「閉じる」)。コマンドウィンドウ上で

```
>> func(5)
```

と入力してエンターを押すと

```
ans =
     75
```

と表示され、この関数が使用可能である事がわかる。また

```
>> y= func(3)
```

のように計算結果に名前(ここでは y)を付ければ、y は

```
y =
    27
```

となる。マトラブでは足し算(+)引き算(-)割り算(/)掛け算(*)を用いる事ができる。さらに行列の演算などもできる。これらについてここでは詳しくは説明しない。マトラブは R と並んで有名なソフトウエアであり、Web 上を検索すればその説明が多く見つかる。詳しくは Web で検索して資料を発見して欲しい。

次にこの関数を最小化する点を求めてみよう。実際にはこれは 0 である事はすぐにわかるが、目的関数がこのように簡単でない場合、数値計算で解かなくてはならないので、その練習としてやってみる。これには “fminunc” 関数を使う。fminunc は関数の制約なしの非線形最小化を行う関数である。制約なし関数とは入力変数の動く範囲に制約をおかない、すなわちこの場合 x は $-\infty < x < \infty$の範囲の値で、この範囲において $f(x)$ が最小値を取る x の値を求める事になる。先ほどの関数 func の最小値を求めるには(ここで 100 とは計算を始める時の x の初期値である)

```
>> fminunc( 'func',100)
```

と入力すると(間にいろいろ出るのは無視して)答として

```
ans =
        2.856749503621359e-09
```

と出力される。最後の e-09 は 10^{-9} を表している。実際の値の 0 に非常に近い値が出ているのがわかる。これを用いて尤度関数の最大化を行う。尤度関数を最大化するという事はその負の値を最小化する事と等しいことに注意しよう。

ここでは先ほどの正規分布の平均 μ と分散 σ^2 を推定してみよう。この時、問題となるのは先ほども述べたように、fminunc はある目的関数を最小化する引数の値を求める関数なので、目的関数は対数尤度関数の負の値とする事と、無制約の最小化を行うので、パラメーターに非負制約等がある場合、例えば分散などは $\sigma^2 = \exp(h)$ などとし、h ($-\infty < h < \infty$) の関数として、目的関数を記述する必要がある事、である。

以上に注意して、まず対数尤度関数のファイルを作る。これは

```
function y = normalLLF(T,Y)
m=T(1); v=exp(T(2)); n= length(Y);
f = -0.5*n*log(2*pi)-0.5*n*log(v)-(1/(2*v))*(Y-m)'*(Y-m);
y= -f;
end
```

のように入力する。(変数名 y, 関数名 normalLLF、入力変数名 T, Y は任意)。ここで 入力変数として T は $T = [\mu, \log(\sigma^2)]$ を考えており、T(1)はその 1 番目の要素、T(2)はその 2 番目の要素であり、2 行目は T(1)と exp(T(2))にそれぞれ、m と v という名前をつけるという意味である。つまり m が μ で v が σ^2 に相当する。分散は非負の値なので、もとの変数を変換して非負の領域しか動かないようにしている事に注意。また観測値は n × 1 ベクトル $\mathbf{Y} = [y_1, \ldots, y_n]'$ として与えられることを想定しており、3 行目の “ Y – m ” は $\begin{bmatrix} y_1 - \mu \\ \vdots \\ y_n - \mu \end{bmatrix}$ というベクトル、“ (Y – m)’ ”はその転置行列を表している。よって

“ (Y-m)’*(Y-m) ” は

$$[y_1-\mu \quad \cdots \quad y_n-\mu]\begin{bmatrix} y_1-\mu \\ \vdots \\ y_n-\mu \end{bmatrix}=\sum_{i=1}^{n}(y_i-\mu)^2$$

を表している。また$\sum_{i=1}^{n}(y_i-\mu)^2$の計算は、Y –m のそれぞれの成分を 2 乗して和をとるという操作であるので、より直接的に

```
sum((Y-m).^2)
```

として計算する事もできる。ここで A.^k はベクトル(または行列)A のそれぞれの成分を成分ごとに k 乗する事を表しており、sum(b) はベクトル b の成分の和を計算するコマンドである。よって、先ほどのプログラムの中で“ (Y-m)'*(Y-m) ” の部分は“ sum((Y-m).^2) ”で置き換えても計算結果は同じになる。

最尤法用の関数を書くときは、最初の引数を未知パラメーターのベクトル、2 つ目の引数を観測値とする(ようにするとよい)。

次にこの関数を最小化する未知パラメーターの値を求めよう。これには “fminunc” 関数を使う。fminunc は関数の制約なしの非線形最小化を行う関数。新たにスクリプトファイルを作成して、nomalMLE.m として保存しよう。このファイルでは以下のように記述する。

```
function t = normalMLE(t0,Y)
s0 =[t0(1),log(t0(2))]';
options = optimset('LargeScale','off','DerivativeCheck','off',...
  'GradObj','off','TolX',1e-6,'Display','off',...
  'Diagnostics','off','MaxIter',1000000);
s=fminunc('normalLLF',s0,options,Y);
t = [s(1),exp(s(2))]';
```

(2行目と3行めの最後は “ … ” で終わっている事に注意。これはコマンドが次の行に続く事を意味する)。この関数 nomalMLE(t0,Y) は観測値ベクトルYが与えられたもとで、初期値t0より計算を始めて、目的関数 normalLLF(T,Y) を最小化するパラメーターベクトルTの値を返す関数になっている。基本的には 1行目の「function t = normalMLE(t0,Y)」、2行目の初期値に関する「s0 =[t0(1),log(t0(2))]';」という部分、および最後の行の「t=fminunc('normalLLF',t0,options,Y);」の部分を適宜変えれば自作の(対数)尤度関数について同じことができる。真ん中のoptions のところは最適化をどのように行うかを指定するもので基本的には変更する必要はない(推定がうまくいかない時などに変更する)。fminunc関数とoptimset関数について詳しくはコマンド画面上で help

fminunc および help optimsetと入力して(エンターキーを押して)ヘルプを見る。

この関数を使って正規分布のパラメーターを最尤推定してみよう。まず平均 $\mu = 2$, 分散 $\sigma^2 = 5$ の正規分布に従う標本を発生させる。これは "randn" 関数を用いる。randn 関数は randn(k,j)で標準正規分布に従う k 行 j 列の行列を発生させる。

```
>>Y=sqrt(5)*randn(100,1)+2
```

とすると 100 行 1 列の $N(2, 5)$ に従う標本を発生させる事ができる(ちなみにY=sqrt(5)*randn(100,1)+2;のように最後にセミコロンを付けると発生させたデータは表示されない)。ちなみにこの場合の最尤推定値は明示的に求められ、ここで発生させた標本に対しては $\hat{\mu}_{MLE}$ = 2.275227181685753, $\hat{\sigma}^2_{MLE}$ = 6.688297716368413 となった(これは実際に発生させた標本に応じて異なるので、ここで説明されたのと同じようにやっても必ずしも同じにならない事に注意)。

では、先ほどの関数を用い数値計算によって最尤推定してみよう。最適化の初期値として t0 = [0, 1]' を用いる。

```
>> t=normalMLE(t0,Y)
t =
    2.275229004800967
    6.688292405451275
```

上記の実際の最尤推定値に非常に近いことがわかる。

Matlab による最尤推定の基本は以上のようになる。対数尤度関数を記述する関数のファイルさえ作れば、あとはほぼ先ほどの m ファイルを少し書き換えてあげれば(初期値の部分や目的関数名を変更する等)実行できる。

3. R による最尤法のプログラム

R でも基本的には同じである。R による関数の作成、使用、再使用については「R における自分で作成した関数の使い方」を参照のこと。ここでは先ほどの Matlab を用いて行ったことと同じことを R で行ってみる。

下記の説明は R の初心者にはわかりずらい部分がある。特に実際の関数の書き方についてはほどんど何も説明していない。R の関数の書き方については Web で検索するとたくさん参考になる資料を見つけることができるので、それらを見るなりして覚えて欲しい。例えば

http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html

などには初心者の人にもわかりやすく、たくさん有用なことが書いてあるので参照してほしい。

R のコンソール上で「ファイル」→「新しいスクリプト」を開き以下のように入力して、testMLE.R

という名前にしてファイルを保存し、以下のように入力する。

```
normalLLF = function(Y){
n=length(Y);
function(T){m=T[1]; v=exp(T[2]);
f=-0.5*n*log(2*pi)-0.5*n*log(v)-(1/(2*v))*sum((Y-m
)^2);
y=-f; return(y)}}

normalMLE = function(t0,Y){
s0=c(t0[1],log(t0[2]));
s=optim(s0,normalLLF(Y),gr=NULL,method=c("BFGS"))
return( t=c( s$par[1],exp(s$par[2]) ) ) }
```

上記のプログラムは Matlab と似ているが、もちろん要所要所で違っている。ただ 2 つの関数 normalLLF と normalMLE の使い方はほぼ同じである。最初の normalLLF というのが対数尤度関数を計算する関数、次の normalMLE というのが与えられた観測ベクトル Y と初期値 t0 に対して μ と σ^2 を推定する最尤法のプログラムである。これらの関数を R に読み込ますには「R における自分で作成した関数の使い方」でやったように source コマンドを用いてもいいが、より簡単にコンソール上の

というアイコンを用いてもよい。具体的には先ほど入力した関数式を全て選択した状態で上記のアイコンを押す。すると選択した部分が全て R に読み込まれる(R のコンソール上に入力される)。

先程と同じように正規乱数を発生させて、それを観測値として期待値と分散を推定してみよう。R では正規乱数は rnorm(n,m,s) という関数で発生させられる。ここで n は発生させる標本数、m は期待値の値、s は標準偏差の値である(分散ではない事に注意)。期待値 2、分散 5 の正規乱数を 100 個発生させるには

```
> Y=rnorm(100,2,sqrt(5))
```

とする。実際に発生させて、先ほどと同様に最尤推定値を計算してみると、$\hat{\mu}_{MLE}$ = 2.064848, $\hat{\sigma}^2_{MLE}$ = 5.059784 となる。それでは先ほど同様、作成したプログラムで推定してみると(初期値としては t0=c(0,1) を使用)

```
> t=normalMLE(t0,Y)
> t
```

```
  [1] 2.064848 5.059784
```
となる。正確な最尤推定値と(少なくとも小数点以下6桁までは)一致していることがわかる。ちなみにこの発生させたデータを保存してマトラブに読み込ませて、マトラブで作成したプログラムで推定すると

```
  >> t = normalMLE([0,1]',Y)
  t =
     2.064847922236464
     5.059784325721777
```
となった。同じ結果である。

このようにして R で最尤推定を行うプログラムを書くことができる。対数尤度関数を計算する関数と、初期条件のところをちょっと変えれば上記のプログラムは他のモデルにも使用できる。