統計数理 (1999)
第47巻 第2号 395-424

特集「情報量規準」
[統計ソフトウェア]

情報量統計学の技術的側面

統計数理研究所　石　黒　真木夫

(受付 1999年5月17日；改訂 1999年6月14日)

要　旨

情報量統計学を

1. データに応じた新しいモデルを開発し,
2. そのモデルをデータにあてはめ,
3. 得られた知見とそれを得た方法を発表する

という一連の流れであると考え, その各段階を支援する技術として,

1. プログラムの虫とり支援のサブルーチンプログラム 'bug' の紹介と使用法の簡単な説明,
2. 対数尤度最大化のためのサブルーチンプログラム 'DALL' の紹介と使用法の簡単な説明,
3. ソフトウエア配布技術としてオープンマーケットライセンス (OML) 条件のもとでの著作権表示の提案,

を束ねた.

キーワード: 虫とり, 対数尤度, 数値的最適化, 疑似ニュートン法, 著作権表示.

1. はじめに

Akaike(1973) による AIC の提案にはじまる情報量規準の研究の発展にともなって, 多くの統計的問題が統計的モデルの比較評価の問題として定式化でき, 統計的モデル開発は統計科学の専門家のみがかかわる事柄でなくデータ解析にたずさわるすべての研究者に開かれたものである, と喧伝されるようになった (Sakamoto et al. (1986)). この情報量統計学の原理的長所を現実のものにするために, 役立ついくつかの基盤的技術がある.

情報量統計学を

1. データに応じた新しいモデルを開発し,
2. そのモデルをデータにあてはめ,
3. 得られた知見とそれを得た方法を発表する

という一連の流れであると考えて, その各段階で有用な技術を紹介する.

情報量統計学では新しいモデルすなわち新しい尤度関数である. 新しいプログラムを組むことが必要となる. 実際にプログラムを書いて最初から正しく計算が出来ることはむしろまれである. 虫とりが大切な技術となる. これが第2節のテーマである.

EIC(Extended Information Criterion, Ishiguro et al.(1997)) など情報量規準の種類が増えているが, モデルあてはめに関しては

$$\text{AIC}= -2\ (\text{最大対数尤度})+ 2\ (\text{パラメータ数})$$

が使いこなせる事がまず目標になる. この場合最大対数尤度の計算が必須であるが, 解析的最大化が可能な型のモデルに発想の範囲を限定するのは得策ではない. 最大化の使いやすい道具を持っている事が必要である. 第3節で数値的最適化の技術を扱う.

また出来上がったモデルすなわちソフトウエアを公開するにあたってその著作権の扱いも技術的基盤として考えることが出来る. 第4節で考察する.

2, 3, 4節はそれぞれ独立しており別個に読むことが出来る. 2, 3節で紹介したサブルーチンの入手法を含む「まとめ」が5節である.

2. 虫とり

新しいデータを入手するごとに新しいモデルを作るという研究スタイルをとる場合プログラムの虫とりが必要になる. 虫とりにあたっては, 異常が検知される以前からの動きを調べることが役に立つ. 普通に行われるのは, プログラム中での中間結果を出力させて, その値の動きを追跡する方法であるが, このような出力の量があまりにも膨大になってしまうことがしばしばである. 異常が起こる (ほんの) 少し前からの値 (だけ) を出力するようにするには,

1. 異常が起きたときの計算結果を参照しながら計算を進めるようにプログラムを作っておき,
2. 異常な動作の記録のすこし前に「前方注意」の記述を書き加えることができて,
3. それをプログラムが読み取ってから精密な出力を出す

ようにしておけばよい.

ここで紹介する bug というサブルーチン (石黒 (1995)) はこれを実現する道具である. bug は2つの動作モードを持っている. 第1のモードでは呼ばれるたびに標準出力に出力する. 第2のモードでは bug は呼ばれるたびに標準出力を記録したファイルを読んで前回の出力値を探す. 自分自身の出力を探す過程でそのファイルにユーザーが書き込んだコマンドを読み, それに応じた動作をする. コマンドの中で特に重要なのは bug にメッセージを送るコマンドである. bug は受け取ったメッセージをメインプログラムに伝えることが出来る. メインプログラムをこのメッセージの値によってデバッグ用出力を出すように作っておけば目的が達成されることになる.

2.1 bug

利用法の詳細は (石黒 (1995)) に与えてあるので, 簡単な例を示す. 後述する URL でコードを配布しているので, 読者がそれを入手し, 手もとの計算機で実験しながら読んで下さること

を想定して書く.

bug ルーチンは,

```
call bug ( Lid , J , J1 , J2 , Cid , Iid , Rid , message )
```

の形で呼ぶ. message だけが出力で他はすべて入力変数である.

- 第 5 引数 Cid には英数字の文字列を入れる.
- Iid と Rid にはそれぞれ整数値と (real*8 の) 実数値を置く.
- Lid, J, J1, J2 は整数値の定数または変数である.

```
program bugged
 implicit real*8(a-h,o-z)
 call bug (1,0,0,0,'start',0,0.d0,message)
 call bug (1,0,0,0,'tutorial',511,1999.d0,message)
 if( message .eq. 123 ) then
    write(6,*) 'message received'
 end if
 do 1 j = 1,10
    call bug (2,j,7,8,'peekaboo',j,0.d0, message )
    call bug (1,j,10,10,'hello',0,0.d0,message)
1 continue
 stop
end
```

図 1. bug を呼ぶプログラムの例.

1. 図 1 に示すプログラムをコンパイルして実行すると, 見かけ上何もせずに終了する.
2. プログラムの第 3 行目,

   ```
   call bug (1,0,0,0,'start',0,0.d0,message)
   ```

   に現れる最初の「1」を「0」に替えて

   ```
   call bug (0,0,0,0,'start',0,0.d0,message)
   ```

   とする.
3. 修正したプログラムをコンパイルして走らせる. このセッションの記録をファイルに残すことが必要である.
4. 今度は, プロンプト

   ```
   start: Bug ? (<Y>es / with <M>ap / <N>o bug )
   ```

   が出るのでこれに 'y' で答えると次のような結果が得られる.

   ```
   DEBugging Bug, version 4-MPI

   start: Bug ? (<Y>es / with <M>ap / <N>o bug )
   y
   DEBB started with a Map
   ```

```
bug.map command list
LOOK     LETITGO  MESSAGE  LEVEL    BACK     QUIT
SKIP     DUMMY
COM: 0:LOOK
COM: 0:LETITGO      -10:
BUG: 0: 1:start   !   1:         0:  .00000000000000000000000000000000D+00:
BUG: 0: 1:tutorial!   2:       511:  .19990000000000000000000000000000D+04:
BUG: 0: 1:hello   :   3:         0:  .00000000000000000000000000000000D+00:
```

- 'bug' は，少くとも一度 Lid = 0 の設定で呼び出されないと活性化されない．虫とりに備えて 'bug' をあらかじめプログラムの要所に埋め込んでおくことが可能である．
- 'bug' の出力の一般型は

  ```
  BUG:id:Lid:Cid   :n: Iid:Cid
  ```

  である．並列計算機の MPI 環境では id にこの出力を発行するプロセッサーの「ランク」が入るが，本稿では並列計算の場合は扱わない．普通の計算機のランクは常に 0 である．n は出力順，その他は bug 呼出時に与えられる情報である．

5. 上記記録を納めたファイルを 'bug.map' と名付ける (ねばならない)．
6. 'bug.map' を編集して，

   ```
   BUG: 0: 1:hello   :   3:         0:  .00000000000000000000000000000000D+00:
   ```

   の一行上にコマンド行を付け加えて，

   ```
   COM: 0:QUIT
   BUG: 0: 1:hello   :   3:         0:  .00000000000000000000000000000000D+00:
   ```

   とする．

   - コマンドの一般型は

     ```
     COM:id:CCCC    d
     ```

     である．CCCC がコマンド，d は整数値である．QUIT のように d が付かないコマンドもある．

7. 再びプログラムを走らせて，プロンプト

   ```
   start: Bug ? (<Y>es / with <M>ap / <N>o bug )
   ```

   に 'm' と答えて何が起こるか見よ．

   - この操作を "map 付きで bug を起動する" という．
   - 回答 'n' も試みてみよ．

8. 上のように手をいれた 'bug.map' 付きで，bug を起動すると，

   ```
   BUG: 0: 1:start   !   1:         0:  .00000000000000000000000000000000D+00:
   BUG: 0: 1:tutorial!   2:       511:  .19990000000000000000000000000000D+04:
   COM: 0:QUIT
   ```

を出力し停止する.

- 'bug' は, 'QUIT' コマンドを受け付けた時点でプログラムを停止させる.
- これによって, 虫とりの際にむだな計算をしないようにできる.

9. 再び 'bug.map' を編集して,

```
BUG: 0: 1:start   :   1:          0:  .00000000000000000000000000000000D+00:
COM: 0:MESSAGE      123
BUG: 0: 1:tutorial:   2:        511:  .19990000000000000000000000000000D+04:
COM: 0:QUIT
BUG: 0: 1:hello   :   3:          0:  .00000000000000000000000000000000D+00:
```

として, map つきで bug を起動すると

```
BUG: 0: 1:start   !   1:          0:  .00000000000000000000000000000000D+00:
COM: 0:MESSAGE      123:
BUG: 0: 1:tutorial:   2:        511:  .19990000000000000000000000000000D+04:
message received
COM: 0:QUIT
```

となる.

- 'bug' は 'MESSAGE' を受け取ってメインプログラムに渡すことが出来る.

10. 更に 'bug.map' を編集して,

```
BUG: 0: 1:start   :   1:          0:  .00000000000000000000000000000000D+00:
COM: 0:MESSAGE      123
COM: 0:LEVEL          2
BUG: 0: 1:tutorial:   2:        511:  .19990000000000000000000000000000D+04:
COM: 0:QUIT
BUG: 0: 1:hello   :   3:          0:  .00000000000000000000000000000000D+00:
```

として, map つきで bug を起動すると

```
BUG: 0: 1:start   !   1:          0:  .00000000000000000000000000000000D+00:
COM: 0:MESSAGE      123:
MAP: 0: 2::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
BUG: 0: 1:tutorial:   2:        511:  .19990000000000000000000000000000D+04:
message received
BUG: 0: 2:peekaboo:   3:          7:  .00000000000000000000000000000000D+00:
BUG: 0: 2:peekaboo:   4:          8:  .00000000000000000000000000000000D+00:
COM: 0:QUIT
```

となる.

- 'Lid'- 値が高い 'bug' の呼出しは, 適切な 'LEVEL' コマンドが発行されてからはじめて有効となる.
- これを活用することによって虫がどこに潜んでいるか分らないプログラムの虫を追い詰めていく作業が楽になる.
- 'bug' は, 'J', 'J1' および 'J2' の間に $J1 \leq J \leq J2$ なる関係が成り立つときのみ出力する.

11. 'bug.map' を編集して

```
BUG: 0: 1:tutorial:   2:       511:  .19990000000000000000000000000000D+04:
```

の行を

```
BUG: 0: 1:tutorial:   2:       511:  .20000000000000000000000000000000D+04:
```

として，map つきで bug を起動すると

```
COM: 0:LOOK
COM: 0:LETITGO      -10:
BUG: 0: 1:start   !   1:         0:  .00000000000000000000000000000000D+00:
COM: 0:MESSAGE      123:
MAP: 0: 2::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
BUG: 0: 1:tutorial!   2:       511:  .19990000000000000000000000000000D+04:
!! mismatch !!!!!!!!!!!! new !!!!!!!!!!!!!!! old !!!!!!!!!!!!!! diff !!!!!!!!!!
OLD: 0: 1:tutorial?   2:       511:  .20000000000000000000000000000000D+04:
DEB: 0:rid:     .1999000000D+04     .2000000000D+04     .5000000000D-03
!! letitgo = -10!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
MEM: QUIT on -10 LETITGO
```

となる．

- bug は，'bug.map' の記録が合わないのを検出して，プログラムの実行を停止することが出来る．
- この機能は，計算結果を変えずにより効率のよい計算法に変えようとするような作業をする場合に便利である．

12. さらに 'bug.map' の

```
COM: 0:LETITGO      -10:
```

の行の「-10」の値を「-3」にして，

```
COM: 0:LOOK
COM: 0:LETITGO       -3:
bug.map command list
LOOK     LETITGO  MESSAGE  LEVEL    BACK     QUIT
SKIP     DUMMY
BUG: 0: 1:start   :   1:         0:  .00000000000000000000000000000000D+00:
COM: 0:MESSAGE      123
COM: 0:LEVEL          2
BUG: 0: 1:tutorial:   2:       511:  .20000000000000000000000000000000D+04:
COM: 0:QUIT
BUG: 0: 1:hello   :   3:         0:  .00000000000000000000000000000000D+00:
```

として，map つきで bug を起動すると

```
COM: 0:LOOK
COM: 0:LETITGO       -3:
BUG: 0: 1:start   !   1:         0:  .00000000000000000000000000000000D+00:
COM: 0:MESSAGE      123:
MAP: 0: 2::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
BUG: 0: 1:tutorial!   2:       511:  .19990000000000000000000000000000D+04:
```

```
!! mismatch !!!!!!!!!!!! new !!!!!!!!!!!!!!! old !!!!!!!!!!!!!! diff !!!!!!!!!!
OLD: 0: 1:tutorial?   2:       511:  .200000000000000000000000000000000D+04:
DEB: 0:rid:      .1999000000D+04      .2000000000D+04      .5000000000D-03
!! letitgo =  -3!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
message received
BUG: 0: 2:peekaboo:   3:         7:  .000000000000000000000000000000000D+00:
BUG: 0: 2:peekaboo:   4:         8:  .000000000000000000000000000000000D+00:
COM: 0:QUIT
```

という結果を得る.

- 'LETITGO' の値を変えて，条件を緩めることによって 'bug.map' の記録が合わない場合にも計算の実行を継続させることが出来る.
- この機能は計算プログラムを他の機種に移植する場合に役立つ.

2.2 使用例

単純な例を示す. 単純すぎて現実の虫の例としては不適当かもしれない. しかし, 動作の本質はとらえている. 非常に扱いづらい虫を扱う場合をイメージしながら読んでいただきたい. もう少し真実味のある使用例が技術報告に与えてある.

図 2 に示すプログラムは,

```
      implicit real*8 (a-h,o-z)
      sum = 0.d0
      do 1 i = 1,500
         a = 1.1d0 ** i
         sum = sum + 1.d0 / (( a + 1.d0) ** 3 - a ** 3)
    1 continue
      write(6,*) 'sum =', sum
      stop
      end
```

図 2. 虫が潜むプログラム.

$$s = \sum_{i=1}^{500} \frac{1}{(1.1^i + 1)^3 - 1.1^{3i}} \tag{2.1}$$

を計算するものである.

実行の記録を図 3 に示す.

```
MI@sunmi% simple
sum = Infinity
 Note: the following IEEE floating-point arithmetic exceptions
 occurred and were never cleared; see ieee_flags(3M):
 Inexact;  Division by Zero;
```

図 3. 不正な計算結果.

sum を無限大にしてしまう虫がいる. 'bug' 呼出しを書き込んだ図 4 に示すプログラムを実行して図 5 の結果を得る.

```
      implicit real*8 (a-h,o-z)
      sum = 0.d0
      call bug(0,0,0,0,'start',0,0.d0,message)
      do 1 i = 1,500
         a = 1.1d0 ** i
         sum = sum + 1.d0 / (( a + 1.d0) ** 3 - a ** 3)
         call bug(1,mod(i,20),0,0,'sum',i, sum , message )
         if(message .eq. 1) then
            write(6,*) i, ( a + 1.d0) ** 3 - a ** 3, a ** 3
         end if
    1 continue
      write(6,*) 'sum =', sum
      stop
      end
```

図4. 'bug' 付きプログラム.

```
MI@sunmi% simple
DEBugging Bug, version-1

start: Bug ? (<Y>es/with <M>ap/<N>o bug)
y
 DEBB started
 COM: 0:LOOK
 COM: 0:LETITGO      -10:
bug.map command list
 LOOK      LETITGO  MESSAGE  LEVEL    BACK     QUIT
 SKIP      DUMMY
 BUG: 0: 1:start   :   1:         0: 0.00000000000000000000000000000000D+00:
 BUG: 0: 1:sum     :   2:        20: 0.88788420742434559507216818019514 9D+00:
 BUG: 0: 1:sum     :   3:        40: 0.91910501469537997376590965359355 3D+00:
 BUG: 0: 1:sum     :   4:        60: 0.91985207786480982505850079178344 5D+00:
 BUG: 0: 1:sum     :   5:        80: 0.91986878538747829381350129551719 9D+00:
 BUG: 0: 1:sum     :   6:       100: 0.91986915520666123757820287210051 9D+00:
 BUG: 0: 1:sum     :   7:       120: 0.91986916337998558823585426580393 7D+00:
 BUG: 0: 1:sum     :   8:       140: 0.91986916356058168187104229218675 8D+00:
 BUG: 0: 1:sum     :   9:       160: 0.91986916356457215648845249234000 2D+00:
 BUG: 0: 1:sum     :  10:       180: 0.91986916356466041921891019228496 6D+00:
 BUG: 0: 1:sum     :  11:       200: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  12:       220: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  13:       240: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  14:       260: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  15:       280: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  16:       300: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  17:       320: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  18:       340: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  19:       360: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  20:       380: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  21:       400:Infinity                                :
 BUG: 0: 1:sum     :  22:       420:Infinity                                :
```

```
 BUG: 0: 1:sum     :  23:       440:Infinity                                  :
 BUG: 0: 1:sum     :  24:       460:Infinity                                  :
 BUG: 0: 1:sum     :  25:       480:Infinity                                  :
 BUG: 0: 1:sum     :  26:       500:Infinity                                  :
sum = Infinity
```

図 5. 'bug' 記録を含む結果.

この記録にコマンドを書き込んで 'bug.map' ファイル図 6 を作る.

```
start: Bug ? (<Y>es/with <M>ap/<N>o bug)
y
 DEBB started
 COM: 0:LOOK
 COM: 0:LETITGO      -10:
bug.map command list
 LOOK     LETITGO  MESSAGE  LEVEL    BACK     QUIT
 SKIP     DUMMY
 BUG: 0: 1:start   :   1:         0: 0.00000000000000000000000000000000D+00:
 BUG: 0: 1:sum     :   2:        20: 0.88788420742434559507216818019514 9D+00:
 BUG: 0: 1:sum     :   3:        40: 0.91910501469537997376590965359355 3D+00:
 BUG: 0: 1:sum     :   4:        60: 0.91985207786480982505850079178344 5D+00:
 BUG: 0: 1:sum     :   5:        80: 0.91986878538747829381350129551719 9D+00:
 BUG: 0: 1:sum     :   6:       100: 0.91986915520666123757820287210051 9D+00:
 BUG: 0: 1:sum     :   7:       120: 0.91986916337998558823585426580393 7D+00:
 BUG: 0: 1:sum     :   8:       140: 0.91986916356058168187104229218675 8D+00:
 BUG: 0: 1:sum     :   9:       160: 0.91986916356457215648845249234000 2D+00:
 BUG: 0: 1:sum     :  10:       180: 0.91986916356466041921891019228496 6D+00:
 BUG: 0: 1:sum     :  11:       200: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  12:       220: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  13:       240: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  14:       260: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  15:       280: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  16:       300: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  17:       320: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  18:       340: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum     :  19:       360: 0.91986916356466219557574959253543 1D+00:
 COM: 0:MESSAGE       1
 BUG: 0: 1:sum     :  20:       380: 0.91986916356466219557574959253543 1D+00:
 COM: 0:QUIT
 BUG: 0: 1:sum     :  21:       400:Infinity                                  :
 BUG: 0: 1:sum     :  22:       420:Infinity                                  :
 BUG: 0: 1:sum     :  23:       440:Infinity                                  :
 BUG: 0: 1:sum     :  24:       460:Infinity                                  :
 BUG: 0: 1:sum     :  25:       480:Infinity                                  :
 BUG: 0: 1:sum     :  26:       500:Infinity                                  :
sum = Infinity
```

図 6. 'bug.map'.

これをこの 'bug.map' つきで 'bug' を起動して図7の結果を得る.

```
MI@sunmi% simple

DEBugging Bug, version-1

start: Bug ? (<Y>es/with <M>ap/<N>o bug)
m
 DEBB started with a Map
bug.map command list
 LOOK      LETITGO   MESSAGE  LEVEL     BACK      QUIT
 SKIP      DUMMY
 COM: 0:LOOK
 COM: 0:LETITGO       -10:
 BUG: 0: 1:start    !   1:          0: 0.00000000000000000000000000000000D+00:
 BUG: 0: 1:sum      !   2:         20: 0.88788420742434559507216818019514 9D+00:
 BUG: 0: 1:sum      !   3:         40: 0.91910501469537997376590965359355 3D+00:
 BUG: 0: 1:sum      !   4:         60: 0.91985207786480982505850079178344 5D+00:
 BUG: 0: 1:sum      !   5:         80: 0.91986878538747829381350129551719 9D+00:
 BUG: 0: 1:sum      !   6:        100: 0.91986915520666123757820287210051 9D+00:
 BUG: 0: 1:sum      !   7:        120: 0.91986916337998558823585426580393 7D+00:
 BUG: 0: 1:sum      !   8:        140: 0.91986916356058168187104229218675 8D+00:
 BUG: 0: 1:sum      !   9:        160: 0.91986916356457215648845249234000 2D+00:
 BUG: 0: 1:sum      !  10:        180: 0.91986916356466041921891019228496 6D+00:
 BUG: 0: 1:sum      !  11:        200: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  12:        220: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  13:        240: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  14:        260: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  15:        280: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  16:        300: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  17:        320: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  18:        340: 0.91986916356466219557574959253543 1D+00:
 BUG: 0: 1:sum      !  19:        360: 0.91986916356466219557574959253543 1D+00:
 COM: 0:MESSAGE        1:
 BUG: 0: 1:sum      !  20:        380: 0.91986916356466219557574959253543 1D+00:
  380    6.0847228810955D+31    1.5404977927049D+47
  381    1.2169445762191D+32    2.0504025620902D+47
  382    1.6225927682921D+32    2.7290858101421D+47
  383    1.6225927682921D+32    3.6324132132992D+47
  384    2.4338891524382D+32    4.8347419869012D+47
  385    2.4338891524382D+32    6.4350415845655D+47
  386    4.8677783048764D+32    8.5650403490566D+47
  387  0.    1.1400068704594D+48
  388    6.4903710731685D+32    1.5173491445815D+48
  389  0.    2.0195917114380D+48
  390    6.4903710731685D+32    2.6880765679240D+48
  391  0.    3.5778299119068D+48
  392  0.    4.7620916127480D+48
  393  0.    6.3383439365675D+48
  394  0.    8.4363357795714D+48
```

```
 395  0.    1.1228762922610D+49
 396  0.    1.4945483449993D+49
 397  0.    1.9892438471941D+49
 398  0.    2.6476835606154D+49
 399  0.    3.5240668191790D+49
COM: 0:QUIT
Note: the following IEEE floating-point arithmetic exceptions
occurred and were never cleared; see ieee_flags(3M):
Inexact;  Division by Zero;
```

図7．虫の発見．

途中で $(a+1)^3 - a^3 = 0$ と計算されたための異常である事が分る．いわゆる桁落ちである．

$$\frac{x-y}{x^3-y^3} = \frac{1}{x^2+xy+y^2} \tag{2.2}$$

なる恒等式に

$$x = y+1 \tag{2.3}$$

を代入して得られる

$$\frac{1}{(y+1)^3-y^3} = \frac{1}{(y+1)^2+(y+1)y+y^2} \tag{2.4}$$

の右辺の形を利用して計算するようプログラムを修正すればこのような事は起こらない．

ノート． 'bug' は MPI(Message-Passing Interface) を利用する並列計算機の環境で使えるように設計されている．このサブルーチンはプログラム中どこに書き込んでもよい．もちろん虫によっては 'bug' を書き加えることに干渉されて姿を現す場所を変えたり，消えてしまう虫もいる．'bug' はそうでない場合の虫を扱う道具である．

3. 最適化

最適化の一般的な手法に関しては様々な文献，ソフトウエアが存在するが，我々が情報量規準を利用する場合に必要な最適化は対数尤度の最大化という特殊な問題，最尤法を解くことに使われる．最尤法であるがための特徴がある．対数尤度関数ではその大小のスケールを1のオーダーで見ればよいということが分っているのでこれを前提としたストッピングルールを設計することが出来る．また対数尤度関数が近似しようとしている平均対数尤度は上から有界であるということが分っている．これは特に病的でない場合には最大値が存在するとの仮定のもとで計算を進めてまちがいがないことを意味する．

3.1 DALL

ここで紹介するのは疑似ニュートン法として代表的な Davidon 法をインプリメントした DALL(Ishiguro and Akaike (1989)) というプログラムである．all, ball, call, fall, gall, hall, mall, pall, tall, wall などの英語の言葉の類推からドールと読むことにしている．

Davidon 法のロジック. 関数 ϕ が

$$\phi(\boldsymbol{x}) = \phi_0 + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_0)^T V_0^{-1}(\boldsymbol{x}-\boldsymbol{x}_0) \tag{3.1}$$

で表される形をしているとき, $\boldsymbol{x}$ における微分ベクトルは

$$\boldsymbol{g} = V_0^{-1}(\boldsymbol{x}-\boldsymbol{x}_0) \tag{3.2}$$

と計算され, 従って関数の最大値を与える点 $\boldsymbol{x}_0$ は

$$\boldsymbol{x} - V_0\boldsymbol{g} \tag{3.3}$$

で与えられる. 関数の形を表す V_0 (バリアンスという) が分っていればある点の微分ベクトル $\boldsymbol{g}$ から最適点がただちに求められるわけである. 逆に, 何箇所かの微分ベクトルが分っていれば V_0 を推定できることも明らかである. Davidon(1968) のバリアンス手続きはこの2つの性質を利用する. $\boldsymbol{x}$ における関数値を

$$PHI = \phi(x)$$

で表すことにすると, これと最大値の差は

$$\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_0)^T V_0^{-1}(\boldsymbol{x}-\boldsymbol{x}_0) \tag{3.4}$$

であり,

$$\frac{1}{2}\boldsymbol{g}^T V_0 \boldsymbol{g} \tag{3.5}$$

に等しい.

$$RHO = \boldsymbol{g}_* V_n \boldsymbol{g}_* \tag{3.6}$$

と定義する. RHO はバリアンスが V_n であるとの仮定のもとで得られる微分ベクトル $\boldsymbol{g}_*$ の位置における値と最適値との差を見積る量とみなすことができる.

DALL においては微分ベクトル $\boldsymbol{g}$ の第 i 成分の計算を数値計算

$$g_i = \frac{\phi(\boldsymbol{x}+\mathrm{step}(i)\boldsymbol{e}_i) - \phi(\boldsymbol{x}-\mathrm{step}(i)\boldsymbol{e}_i)}{2\mathrm{step}(i)}$$

によって行う. $\boldsymbol{e}_i$ は第 i 成分のみ1の単位ベクトルである. 本来の微分と数値微分の差が存在するという欠点があるが, その一方で数値微分をすることによって実際のプログラム作成の過程でおこり得る関数値の大小関係と微分値の間の矛盾を避けることが可能である. 関数値を計算するサブルーチンと微分値を計算するサブルーチンを別に組むこと自体わずらわしい作業であり, 研究の過程での作業量の節約は実用的な方法として必須の条件である. 又, 数値微分を採用できるもう一つの理由はわれわれが最適化する関数が対数尤度関数であり, その形, 特にパラメータのスケールに関してプログラム作成者の自由がきくし, 直観もきくという前提をおいていいと考えられるからである. なお, この微分ベクトルの計算に並列計算機を利用できると最適化に要する時間が大幅に短縮できる.

手続き.

(a) 点, $\boldsymbol{x}_*(=x_n - V_n g_n)$ における対数尤度

$$SPHI = \phi(\boldsymbol{x}_*) \tag{3.7}$$

と微分ベクトル $\boldsymbol{g}_*$ を計算する. 式 (3.1) が正しく, $V_n = V_0$ であれば $\boldsymbol{g}_* = 0$ となり, 手続きはここで終了する.

(b) $\boldsymbol{g}_* \neq 0$ であった事は $V_n \neq V_0$ を意味する. (3.1) 式が正しいものとすると V_0 は

$$\boldsymbol{x} - \boldsymbol{x}_* = V_0(\boldsymbol{g} - \boldsymbol{g}_*) \tag{3.8}$$

を満足することが分っているので, V_{n+1} が

$$\boldsymbol{x} - \boldsymbol{x}_* = V_{n+1}(\boldsymbol{g} - \boldsymbol{g}_*) \tag{3.9}$$

を満たすように選ぶ. このとき, $V_{n+1} = V_n + (LAMBDA - 1)V_n \boldsymbol{g}_* \boldsymbol{g}_*^T V_n / RHO$ という形を仮定する.

一般に, K 次元行列 A の形を調べるには K 個のベクトルの直交系 $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \ldots, \boldsymbol{b}_k\}$ を用いて, $\{A\boldsymbol{b}_1, A\boldsymbol{b}_2, A\boldsymbol{b}_3, \ldots, A\boldsymbol{b}_k\}$ を見れば良いが, $\boldsymbol{b}_1 = V_n \boldsymbol{g}_*$ として残りの $\{\boldsymbol{b}_2, \boldsymbol{b}_3, \ldots, \boldsymbol{b}_k\}$ をこれと直交するように選ぶと, $V_{n+1}\boldsymbol{b}_j = V_n \boldsymbol{b}_j$ $(j = 2, \ldots, K)$ となるのが明らかである.

$LAMBDA$ が 1 から大きく離れていると更新量が大きく, (3.1) 式が成り立つとする仮定からの外れが大きいときの挙動が悪くなるので, $LAMBDA < \alpha$ なら, $LAMBDA = \alpha$ とし, $LAMBDA > \beta$, なら $LAMBDA = \beta$ としている. DALL の 1999 年 5 月現在の版では $\alpha = 0.25$, $\beta = 4.0$ である.

(c) $LAMBDA - 1 = 0$ のとき $V_{n+1} = V_n$ となるので V_{n+1} をリセットする.

(d)

$$x_{n+1} = \begin{cases} x_* & (SPHI > PHI) \\ x_n & (SPHI \leq PHI) \end{cases}$$

(e) n を 1 増やして (a) に戻る.

3.2 例

2 次元問題. 以下の関数は特に具体的な対数尤度でなく模型的に作った関数である.

$$f(x, y) = -\frac{1}{2}\{r(x^2 + x + y)^2 + (x^2 + 3x + y)^2\} \tag{3.10}$$

$r = 40.0$ とした. 図 8 の左上に等高線を示してある. 頂点から値が 1 下がった位置と, 値が 10 下がった位置の 2 本だけ描いてある. B 点 (0, 0) で最大値 0 をとる. 初期値 A 点 (-2.5, -3.0) から出発した場合を示す. 最終的に落ち着いた結果を右上の NCOUNT= 70 の図に示した.

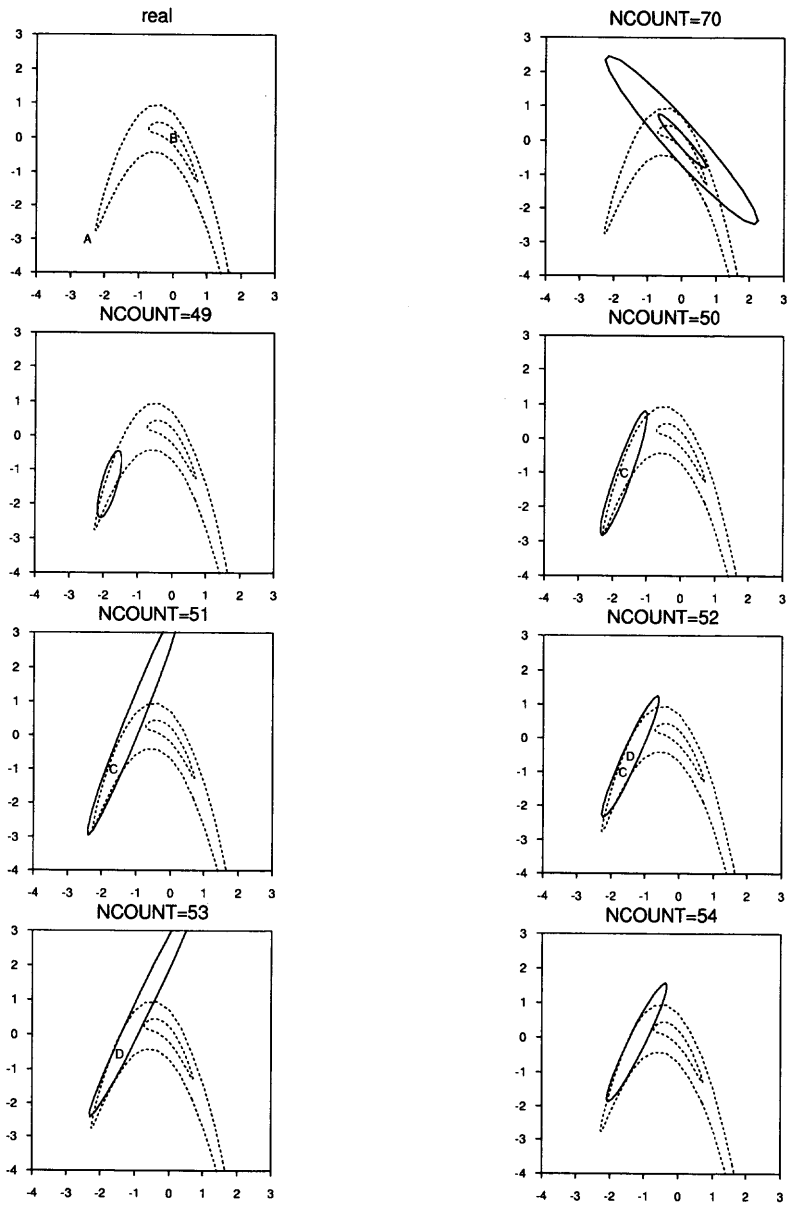

図 8. Davidon 法による最大値の探索過程.

疑似ニュートン法が想定しているのは目的関数が 2 次曲面で表されるということであり, 最大値から 1 だけ下がった等高線と 10 下がった等高線を描くと 2 重の楕円になる. DALL によって認識された関数が大局的な形に関しては大きくずれているが最大値の位置と関数の局所的な形をおおよそとらえているのが明らかである.

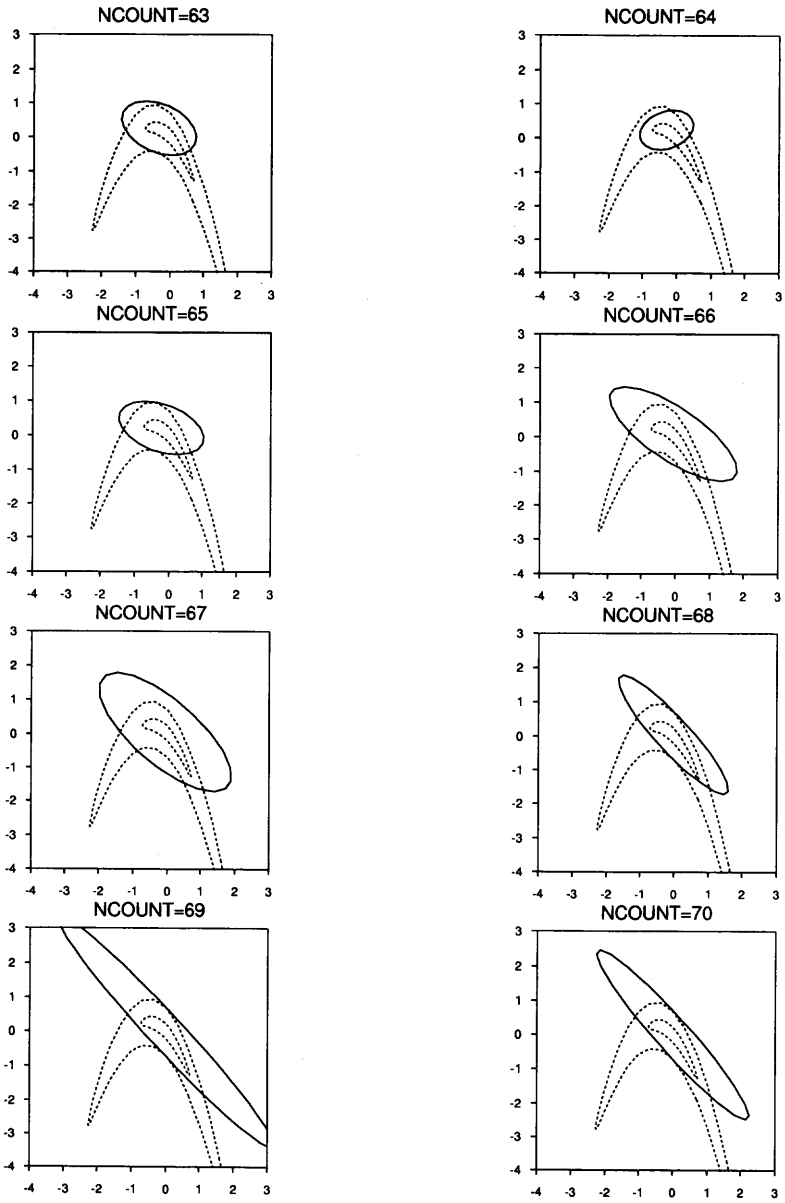
NCOUNT=63
NCOUNT=64
NCOUNT=65
NCOUNT=66
NCOUNT=67
NCOUNT=68
NCOUNT=69
NCOUNT=70

図 8. (つづき)

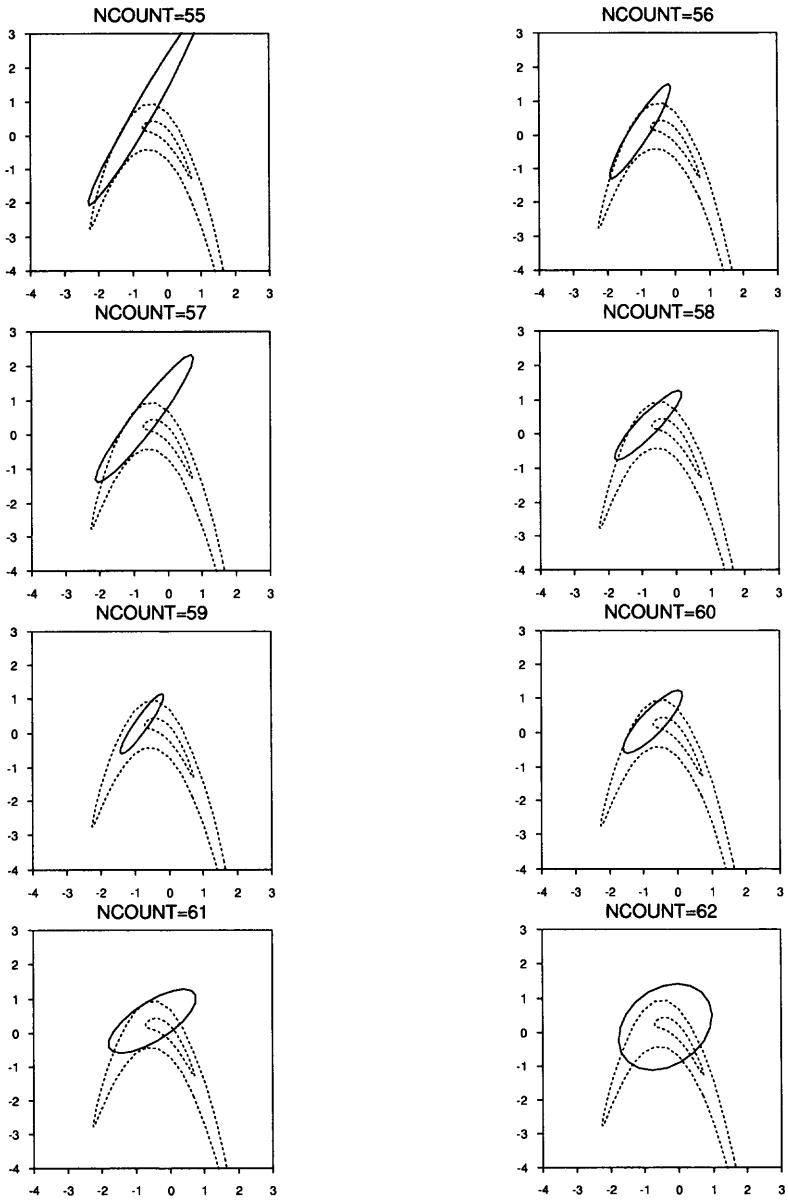
NCOUNT=55
NCOUNT=56
NCOUNT=57
NCOUNT=58
NCOUNT=59
NCOUNT=60
NCOUNT=61
NCOUNT=62

図8. (つづき)

この時の DALL の印字出力の一部を図 9 に示す.

```
D2:Hessian reset count = 11
D2:value=          -7.60
D2:point
     -0.19124D+01 -0.15012D+01
B3: NCOUNT          PHI        SPHI-PHI        RHO   LAMBDA-1    ISPHI
B3:    47          -7.60          0.1857      -2.2604    -0.50       0
B3:    48          -7.41          0.6270      -0.0819     2.73       0
B3:    49          -6.79          0.2991      -0.2897     3.00       0
B3:    50          -6.49          1.1112      -0.9496     3.00       0
B3:    51          -5.38         -1.0834     -49.6538    -0.75       0
B3:    52          -5.38          1.3701      -1.0523     3.00       0
B3:    53          -4.01         -8.7070    -390.8665    -0.75       0
B3:    54          -4.01          1.0407      -0.5745     2.85       0
B3:    55          -2.97         -2.2384     -73.4456    -0.75       1
B3:    56          -2.97          0.7719      -0.4114     2.26       0
B3:    57          -2.19          0.2893      -4.6355    -0.75       0
B3:    58          -1.90          0.4668      -0.3263    -0.75       0
B3:    59          -1.44          0.0776      -0.0683     3.00       0
B3:    60          -1.36          0.2411      -0.1684     3.00       0
B3:    61          -1.12          0.4728      -0.2189     3.00       0
B3:    62          -0.65          0.0453      -3.1124    -0.75       0
B3:    63          -0.60          0.3932      -0.0996    -0.71       0
B3:    64          -0.21          0.0271      -0.0233     3.00       0
B3:    65          -0.18          0.0763      -0.0422     2.51       0
B3:    66          -0.10          0.0882      -0.0166     0.92       0
B3:    67          -0.02          0.0025      -0.0915    -0.75       0
B3:    68          -0.01          0.0123      -0.0015     3.00       0
B3:    69           0.00         -0.0001      -0.0065    -0.52       0
B3:    70           0.00          0.0014       0.0000
B2:Bill ended with abs(RHO) = 0.00003 <  0.00010 = rhomin: code =  0
D2:val. dif.=          7.60
D2:NCOUNT=    70
D1:Bill ended with abs(RHO) < rhomin.
D2:==========  Summary =========
D1:value=           0.00
D1:point
     -0.97975D-06 -0.80570D-03
D2:final gradient
      0.34650D-01  0.33076D-01
D1:val. dif.=         20.28
D2:global profile
    -40.00      -30.00      -20.00      -10.00        0.00       10.00
     1 |                     *XXXXX                    |           |
     2 |                     |    XXXXX*               |           |
     3 |                     |       XXXXX*X           |           |
     4 |                XXXX*XXXXX                     |           |
     5 |           XX*XXXX   |                         |           |
     6 |      *XX    |       |                         |           |
     7 |       *XXXX |       |                         |           |
     8 |            XXXX*XXXXXX                        |           |
     9 |             |       XXXXXX*XXXXXX             |           |
    10 |             |       |           XXXXXX*XX     |           |
    11 |             |       |           |       XX*   |           |
D1:Hessian reset count =  11
D1:end code =70.00
D1:============= end of  maximization ===========
```

図 9. DALL の出力.

図 8 に書き込まれているのは NCOUNT=n における「モデル」

$$\phi(\boldsymbol{x}) = \phi_* + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_*)^T V_n^{-1}(\boldsymbol{x}-\boldsymbol{x}_*) \tag{3.11}$$

の等高線である.

NCOUNT = 51 の図では NCOUNT=50 までの情報に基づいて得られたモデルが描かれている. $\boldsymbol{x}_{51}$ の位置を C で示す. このモデルの頂点の位置 $\boldsymbol{x}_*$ における関数値を調べて見ると $\boldsymbol{x}_{51}$ よりも下がってしまうということが分り, モデルが不正確であることが判明する. モデルを作り直して, NCOUNT = 52 のモデルが作られる. $\boldsymbol{x}_{52} = \boldsymbol{x}_{51}$ である. このモデルの頂点

$\boldsymbol{x}_*(D)$ はC に比べて関数値を改善するので $\boldsymbol{x}_{53} = \boldsymbol{x}_*$ とする．このような経過をたどって最終的に NCOUNT=70 の結果に至る．この図9の最後に出ている星印とXで書かれたグラフはもともとの点AからBへの直線上での関数の形を書いたものである．実際にモデルが頂点を発見する過程でたどった点 $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots,$ における関数値でないので必ずしも単調に増加しない．このグラフの形によって関数が2次曲面に近い形をしているのか，それとも難しい形をしているのか，ある程度イメージをつかむ事が出来る．

1次元問題． 1次元の関数の最大値を見つける例である．DALL の挙動を横から見ることになる．この例では本物の対数尤度を最大化する．n 回の実験で m 回成功したという結果から成功率 p を推定する二項分布の対数尤度

$$\ell(p) = m \log p + (n-m) \log(1-p) \tag{3.12}$$

の最大化の問題である数値的最適化を使わなくても解析的に $\hat{p} = m/n$ であることは簡単に示され，$m = 12$, $n = 76$ の場合，$\hat{p} = 12/76$, ほぼ 0.16 である．

真の値からはずれた初期値から次第に最大値を捜し当てる過程が見られる．ここに見られるように局所的な情報にもとづく放物線をあてはめその最大値の位置で改めて情報を得て放物線を推定し直すという形で推定が進んでいることが見える．特に図中に印はつけてないが $\ell(p)$ のグラフ（破線）と放物線が接しているのが $\boldsymbol{x}_{\mathrm{NCOUNT}}$ の位置である．

制約ガウス分布のあてはめ． ガウス分布のあてはめと制約ガウス分布のあてはめを例としてプログラムの書き方を紹介する．データ $x_1, \ldots, x_n$ が与えられているものとする．正規分布モデルの対数尤度は

$$l_x(\mu_x, \sigma_x^2) = -\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma_x^2 - \frac{1}{2\sigma_x^2} \sum_{i=1}^{n_x} (x_i - \mu_x)^2 \tag{3.13}$$

で与えられる．データ $y_1, \ldots, y_n$ に対しては

$$l_y(\mu_y, \sigma_y^2) = -\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma_y^2 - \frac{1}{2\sigma_y^2} \sum_{i=1}^{n_y} (y_i - \mu_x)^2 \tag{3.14}$$

で与えられる．μ_x と σ_x^2 の最尤推定量は

$$\hat{\mu}_x = \frac{1}{n} \sum_{i=1}^{n} x_i$$

と

$$\hat{\sigma}_x^2 = \frac{1}{n_x} \sum_{i=1}^{n_x} (x_i - \hat{\mu}_x)^2,$$

で与えられる．μ_y, σ_y^2 に関しても同様．

データ $x = \{0.73, -0.06, 1.04, 2.29, 0.51, -0.45, 1.03, 0.44, 0.02, 0.11, -2.42\}$, が与えられた時の最尤推定値は

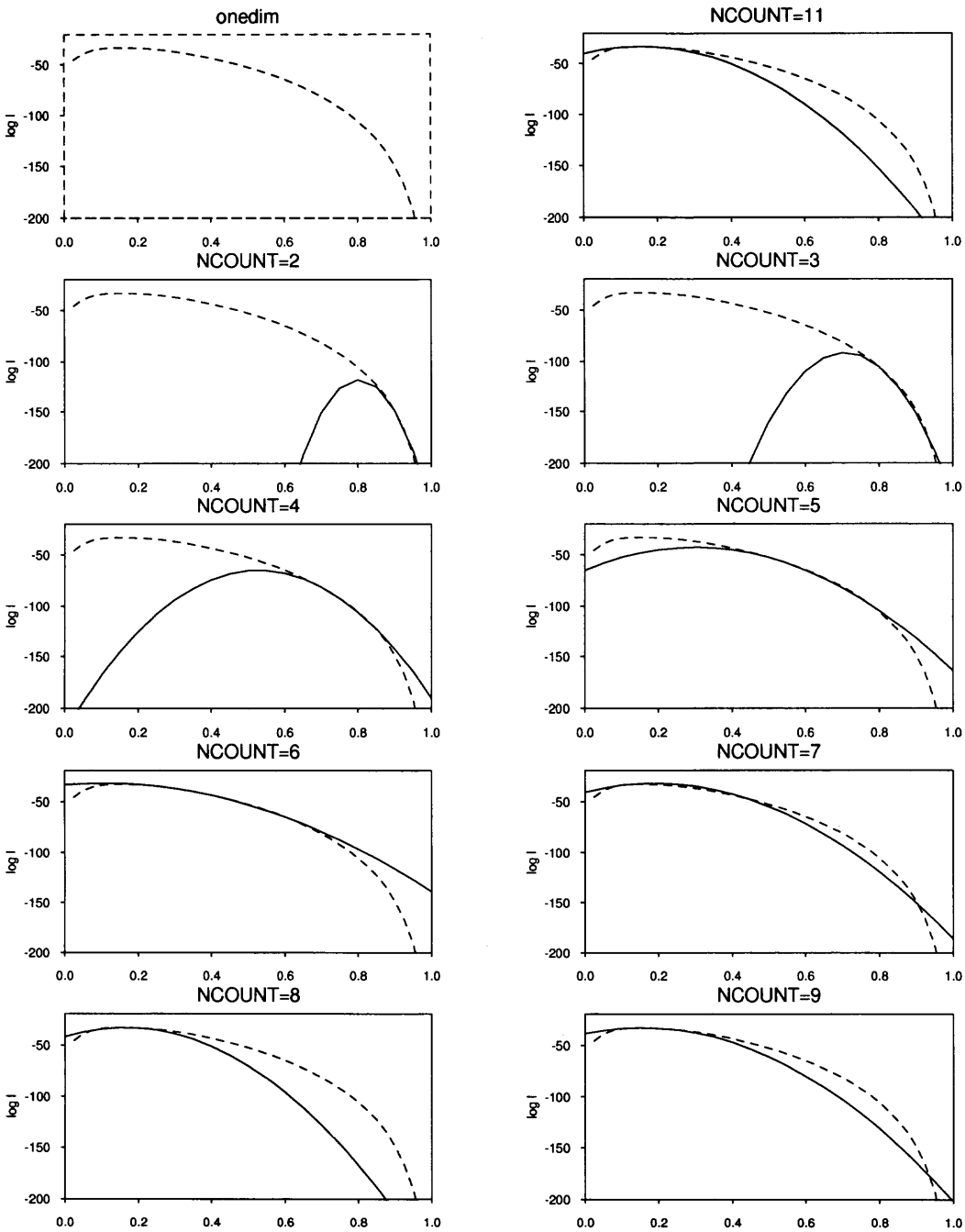

図 10. 二項分布の対数尤度の数値的最大化.

$$\hat{\mu}_x = 0.2945$$
$$\hat{\sigma}_x^2 = 1.2267$$

であり, データ $y = \{0.10, 0.56, -1.11, -0.48, 3.46, -2.39, 0.36, 4.56\}$, が与えられた時の最尤推定値は

$$\hat{\mu}_y = 0.6325$$
$$\hat{\sigma}_y^2 = 4.6491$$

である. μ_x と μ_y に関して, $\mu_x = \mu_y$ という制約を加えたモデルの対数尤度は

$$\begin{aligned} l_{x+y}(\mu_{x+y}, \sigma_x^2, \sigma_y^2) &= -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma_x^2 - \frac{1}{2\sigma_x^2}\sum_{i=1}^{n_x}(x_i - \mu_{x+y})^2 \\ &\quad -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma_y^2 - \frac{1}{2\sigma_y^2}\sum_{i=1}^{n_y}(y_i - \mu_{x+y})^2 \\ &= l_x(\mu_{x+y}, \sigma_x^2) + l_y(\mu_{x+y}, \sigma_y^2) \end{aligned} \tag{3.15}$$

で与えられる. これを最大化するパラメータの値を解析的に求めるには三次方程式を解く必要があり, やっかいな問題となる.

これらの結果を DALL で求めてみよう. 図 11 のサブルーチン群を用意する.

```
c
      subroutine llx(p,m,f,ig)
      implicit real*8 (a-h,o-z)
      dimension p(m)
      common /i721/ n
      common /r721/ x(11)
      data pi /3.14159265d0/
      ig=0
      if(p(2) .gt. 0.d0) then
         sum=0.d0
         do 1 i=1,n
            sum=sum+(x(i)-p(1))**2
    1    continue
         f= -n*0.5*( dlog(pi*2)+dlog(p(2)) ) - 0.5*sum/p(2)
         ig=1
      end if
      return
      end
c
      subroutine lly(p,m,f,ig)
```

```
      implicit real*8 (a-h,o-z)
      dimension p(m)
      common /i722/ n
      common /r722/ x(8)
      data pi /3.14159265d0/
      ig=0
      if(p(2) .gt. 0.d0) then
         sum=0.d0
         do 1 i=1,n
            sum=sum+(x(i)-p(1))**2
    1    continue
         f= -n*0.5*( dlog(pi*2)+dlog(p(2)) ) - 0.5*sum/p(2)
         ig=1
      end if
      return
      end

c
      subroutine llxy(p,m,f,ig)
      implicit real*8 (a-h,o-z)
      dimension p(m),px(2),py(2)
      px(1)=p(1)
      px(2)=p(2)
      call ll1(px,2,fx,ig)
      if(ig .eq. 0) return
      py(1)=p(1)
      py(2)=p(3)
      call ll2(py,2,fy,ig)
      if(ig .eq. 0) return
      f=fx+fy
      return
      end
c
      block data ex72
      implicit real*8 (a-h,o-z)
      common /i721/ n1
      common /r721/ x1(11)
      common /i722/ n2
      common /r722/ x2(8)
      data n1/11/
      data x1/ 0.73, -0.06,  1.04,  2.29,  0.51,
     *        -0.45,  1.03,  0.44,  0.02,  0.11,
     *        -2.42 /
      data n2/8/
      data x2/ 0.10,  0.56, -1.11, -0.48,  3.46,
     *        -2.39,  0.36,  4.56 /
      end
```

図 11. ガウス分布，制約ガウス分布の対数尤度を計算するサブルーチン．

サブルーチンllx, lly, llxy, は大きさmのアレイpを通して与えられたパラメータの値に対する対数尤度を計算し, 変数fを通して返すように作る. 対数尤度が計算できた時にはigの値を正とする. 対数尤度が定義されないパラメータが与えられた場合はig=0とする. 式(3.15)に対応して, llxyはllxとllyを呼ぶ形で作られている.

メインルーチン図12は初期値を与える部分とDALLを呼ぶ部分, 結果を印刷する部分で構成される.

```
      program gauss

      parameter  (ipmax=4)
      implicit  real*8( a-h,o-z )
      dimension a1(ipmax),a2(ipmax),a3(ipmax)
      dimension g(ipmax), vd(ipmax,ipmax+5),step(ipmax)
      external llx,lly,llxy
c
      myid = 0

      do 1 i=1,ipmax
 1    step(i) = 0.001d0

      ipx=2
      ax(1) = 0.d0
      ax(2) = 1.d0
      call dall(llx,ax,ipx,f,g,vd,endc,step,0,3)
      if(myid .eq. 0) then
         write(6,*) ' meanx(mle)=',ax(1)
         write(6,*) ' variancex(mle)=',ax(2)
         write(6,*) ' maximum likelihood=',f
         write(6,*) ' number of parameter=',ipx
         aic= -2*f + 2*ipx
         write(6,'('' AICx='',F10.2)') aic
      end if

      ipy=2
      ay(1) = 0.d0
      ay(2) = 1.d0
      call dall(lly,a2,ipy,f,g,vd,endc,step,0,3)
      if(myid .eq. 0) then
         write(6,*) ' meany(mle)=',a2(1)
         write(6,*) ' variancey(mle)=',ay(2)
         write(6,*) ' maximum likelihood=',f
         write(6,*) ' number of parameter=',ipy
         aic= -2*f + 2*ipy
         write(6,'('' AICy='',F10.2)') aic
      end if

      ipxy=3
      axy(1) = (ax(1) + ay(1))*0.5d0
      axy(2) = ax(2)
      axy(3) = ay(2)
```

```
call dall(llxy,a3xy, ipxy, f,g,vd,endc,step,0,3)
if(myid .eq. 0) then
   write(6,*) ' meanxy(mle)=',axy(1)
   write(6,*) ' variancedx1(mle)=',axy(2)
   write(6,*) ' variancey(mle)=',axy(3)
   write(6,*) ' maximum likelihood=',f
   write(6,*) ' number of parameter=',ipxy
   aic= -2*f + 2*ipxy
   write(6,'('' AICxy='',F10.2)') aic
end if

stop
end
```

図 12. ガウス分布，制約ガウス分布をあてはめるプログラム.

図 12 に見られるように dall を呼ぶ時の一般形は

```
call dall(ll, p, m, f, g, vd, endc, sstep, lim, iout)
```

である. 各変数の意味を表 1 にまとめておく.

図 12 のプログラムでは $iout = 3$ となっている. このプログラムを走らせた時の出力を図 13 に示す.

表 1.

変数	内容
ll	対数尤度を計算するサブルーチンの名
p, m	パラメータベクトル
f	対数尤度
g, vd	作業領域
endc	終了コード *.0 が正常終了. 整数部分は数値微分の回数.
lim	数値微分回数制限 0 で無制限
iout	印字出力レベル 0= 無印字, 3= 最大出力

```
D1:==========  log likelihood maximization =========
D1:number of processor =  1
D1:limit =  0

D2:Hessian reset count =  0
D2:value=         -17.33
D2:point
       0.00000D+00  0.10000D+01
B3: NCOUNT           PHI        SPHI-PHI        RHO  LAMBDA-1   ISPHI
B3:    2          -17.33          0.5980     -0.0019     0.08      0
B2:Bill ended with abs(log LAMBDA)=  0.08 <  0.20 = ramlim: code = 70
D2:val. dif.=           0.60
D2:NCOUNT=     2
```

```
D2:Hessian reset count =  1
D2:value=          -16.73
D2:point
       0.29455D+00  0.11927D+01
B3: NCOUNT            PHI        SPHI-PHI        RHO  LAMBDA-1   ISPHI
B3:    5           -16.73          0.0022      0.0000
B2:Bill ended with abs(RHO) = 0.00001 <  0.00010 = rhomin: code =  0
D2:val. dif.=            0.00
D2:NCOUNT=     5
D1:Bill ended with abs(RHO) < rhomin.
D2:==========  Summary =========
D1:value=          -16.73
D1:point
       0.29455D+00  0.12249D+01
D2:final gradient
      -0.17764D-11  0.67384D-02
D1:val. dif.=            0.60
D2:global profile
    -17.50    -17.25    -17.00    -16.75    -16.50    -16.25    -16.00
     1 !      *XX!         !         !         !         !         !
     2 !         XX*X      !         !         !         !         !
     3 !         !    X*X  !         !         !         !         !
     4 !         !        X*X        !         !         !         !
     5 !         !         ! X*      !         !         !         !
     6 !         !         !    *    !         !         !         !
     7 !         !         !      *  !         !         !         !
     8 !         !         !        *!         !         !         !
     9 !         !         !         *         !         !         !
    10 !         !         !         !*        !         !         !
    11 !         !         !         !*        !         !         !
D1:Hessian reset count =   1
D1:end code = 4.00
D1:============= end of  maximization ===========
meanx(mle)=   0.29454545454562
variancex(mle)=    1.2248781111300
maximum likelihood=   -16.732202448716
number of parameter=  2
AICx=     37.46

D1:==========  log likelihood maximization =========
D1:number of processor =  1
D1:limit =  0

D2:Hessian reset count =  0
D2:value=          -27.55
D2:point
       0.00000D+00  0.10000D+01
B3: NCOUNT            PHI        SPHI-PHI        RHO  LAMBDA-1   ISPHI
B3:    2           -27.55          5.8549     -1.0351     0.61      0
```

```
B3:    3          -21.69          1.3469      -0.7011     1.85      0
B3:    4          -20.35          1.4235      -0.4922     0.99      0
B3:    5          -18.92          0.7238      -0.2774     1.14      0
B3:    6          -18.20          0.4243      -0.1402     0.95      0
B3:    7          -17.77          0.1914      -0.0559     0.83      0
B3:    8          -17.58          0.0684      -0.0149     0.62      0
B3:    9          -17.51          0.0151      -0.0019     0.39      0
B3:   10          -17.50          0.0015      -0.0001
B2:Bill ended with abs(RHO) = 0.00006 <  0.00010 = rhomin: code =  0
D2:val. dif.=          10.05
D2:NCOUNT=    10
D1:Bill ended with abs(RHO) < rhomin.
D2:==========  Summary =========
D1:value=         -17.50
D1:point
       0.63250D+00  0.46284D+01
D2:final gradient
      -0.44409D-10  0.38577D-02
D1:val. dif.=          10.05
D2:global profile
    -30.00    -27.50    -25.00    -22.50    -20.00    -17.50    -15.00
     1 !         *XXXXXXXX !         !         !         !         !
     2 !         !         XXXXXXXX*XXXX       !         !         !
     3 !         !         !         ! XXXX*XX !         !         !
     4 !         !         !         !         XX*X      !         !
     5 !         !         !         !         !   X*    !         !
     6 !         !         !         !         !      *  !         !
     7 !         !         !         !         !       * !         !
     8 !         !         !         !         !        *!         !
     9 !         !         !         !         !         *         !
    10 !         !         !         !         !         *         !
    11 !         !         !         !         !         *         !
D1:Hessian reset count =   0
D1:end code =10.00
D1:============= end of  maximization ===========
meany(mle)=   0.63250000002579
variancey(mle)=    4.6284090071286
maximum likelihood=   -17.498215714685
number of parameter=  2
AICy=     39.00

D1:==========  log likelihood maximization =========
D1:number of processor =  1
D1:limit =  0

D2:Hessian reset count =  0
D2:value=         -34.38
D2:point
       0.46352D+00  0.12249D+01  0.46284D+01
B3: NCOUNT           PHI       SPHI-PHI       RHO  LAMBDA-1   ISPHI
```

```
B3:    2          -34.38          0.0698      -0.0023   -0.47      0
B3:    3          -34.31          0.0009      -0.0003    1.09      0
B3:    4          -34.31          0.0003       0.0000
B2:Bill ended with abs(RHO) = 0.00000 <  0.00010 = rhomin: code =  0
D2:val. dif.=           0.07
D2:NCOUNT=     4
D1:Bill ended with abs(RHO) < rhomin.
D2:==========  Summary =========
D1:value=         -34.31
D1:point
       0.34827D+00  0.12291D+01  0.47291D+01
D2:final gradient
       0.22816D-04  0.19078D-02  0.14178D-03
D1:val. dif.=           0.07
D2:global profile
    -34.50    -34.25    -34.00    -33.75    -33.50    -33.25    -33.00
     1 !    *    !         !         !         !         !         !
     2 !    *    !         !         !         !         !         !
     3 !     *   !         !         !         !         !         !
     4 !     *   !         !         !         !         !         !
     5 !     *   !         !         !         !         !         !
     6 !      *  !         !         !         !         !         !
     7 !      *  !         !         !         !         !         !
     8 !      *  !         !         !         !         !         !
     9 !      *  !         !         !         !         !         !
    10 !      *  !         !         !         !         !         !
    11 !       * !         !         !         !         !         !
D1:Hessian reset count =   0
D1:end code = 4.00
D1:============= end of  maximization ===========
meanxy(mle)=   0.34826773312855
variancex(mle)=    1.2290783247532
variancey(mle)=    4.7290641711741
maximum likelihood=   -34.312209336033
number of parameter=  3
AIC3=     74.62
```

図 13. ガウス分布， 制約ガウス分布のあてはめ.

ノート. 数値的最適化の Davidon 法による扱いは局大値を持つ関数を 2 次局面で近似するという問題とみることが出来る. これは真の関数の形が 2 次局面からずれている場合に収束が悪くなるということを意味しているが, 逆に 2 次局面でない形で関数を近似する手法を導入することによってより能率のよい最適化の手法が得られる可能性を示している.

4. ソフト配布技術

ソフトウエアを公開するということが重要であるが, ソフトウエアは他の著作物と違って他の研究者あるいはユーザーがそれを再利用・改良するという使い方が大きな比重を占めるためそれに留意した配布の仕方を考える必要がある.

4.1 Open Market Licence

これは著作権表示に添えられている著作権者の権利行使に関する方針を明らかにした文章を転載することを条件にそのソフトウエアの改変及び再配布を許可するというものである. 統計数理研究所によって提案されている. 日本語版, 英語版があるが, 英語版は以下のものである.

Open Market Licence for software(version:OML-SW-E-1996)

0. In the following, 'Copyright_OML notice' means 'the copyright notice with a statement saying that the said software is made public under this Open Market Licence for software'.
1. On the condition that 'Copyright_OML notice' and attached statements are copied as they are, all or any portion of the said software can be copied or redistributed.
2. Modification of the software is permitted, provided that the exact place of the modification, the date and the name of the modifier are shown conspicuously.
3. It is forbidden to apply for a patent on the software which utilizes the said software.
4. The said software is distributed without any warranty. The copyright holder takes no responsibility for any damage caused by the use of the said software.
5. Any interactively controlled application made utilizing the said software has to show 'Copyright_OML notice' and attached statements conspicuously when it is started up.

このような方法は再配布を許すことによって著作権の放棄のように見えるかもしれないが, 著作権という権利の行使には様々な形体があり得る. ある条件のもとで自由に使わせるというのも著作権の行使法の一つであり, その意味で権利の放棄ではない.

4.2 著作権表示

Open Market Licence は適当な著作権表示を組み合わせることによって力を発揮する. たとえば, 次のような著作権表示が考えられる.

Copyright_OML 199x Some Body

The source code of this xxxxx can be obtained from YYYYY without any charge. On the conditions that terms of the OPEN MARKET LICENCE for software (version:OML-SW-E-1996, ftp://ftp.ism.ac.jp/pub/ISMLIB/OML) are observed and that the copyright holder nor YYYYY take no responsibility on any result from the use, xxxxx can be used or modified freely. On the

condition that this note is attached as it is, xxxxx can be redistributed. Appropriate reference must be made at times of publishing results obtained using this xxxxx.

有料での再配布も許していると解釈されるが, 上の著作権表示が付されたソフトを売ろうとする者はそのソフトのオリジナルが YYYYY で無償で配布されていることを告知せざるを得ない. しかし, 有償再配布の禁止と等価ではない. 元の版より使いやすいインターフェースが用意されているような場合にその付加価値を認めて購入するという事は大いにあり得る. この方法はソフトウエアの

- 流通を容易にし,
- 改良をうながす

方向にはたらくと期待される.

例示した著作権表記のポイントは「The original code of this subroutine can be obtained from YYYYY without any charge.」の一文にある. この「YYYYY」をどう用意するかが問題である.

4.3 URL

インターネットの普及によって上記「YYYYY」を個人的に維持することも全く不可能というわけでは無くなった. しかしその信頼性を保つのは容易ではない. ある程度以上の組織の仕事と考えるのが自然であり, 個人的に技術を身につければ解決される類の問題ではないから本来この稿の範囲を越えるが, その組織の仕様を考えておくことは無駄ではないだろう.

この「組織」の任務は,

1. ソフトウエアが確実・容易に入手可能であること
2. OML の条項が確実に容易に読めること

を「恒久的に」保証することである.

具体的にはネットワークに接続されたサーバーのメンテナンスと言っていいだろう. ハードウエアの整備, 更新などに責任を持つ「役職」を作る必要があるだろう. 当然, かなりの経費がかかる「事業」と考える必要がある. 著作権表示に「YYYYY」と書き込むということは, かなりの資源を消費することがらであるということになる. ということは, 資源を提供する側でなんらかの基準にもとづいた審査が必要ということになる.

以上をまとめると, 「YYYYY」の実態は,

- ハードウエアの管理者
- 審査委員会

ということになる. ある分野に「中核的研究機関」のようなものがあるとすると, その分野のソフトウエアの配布に関してはその機関が「YYYYY」を用意するのが自然な考え方であろうと思われる.

5. まとめ

プログラムを作り, 虫をとり, 公表する過程をサポートする 3 つの道具

1. 最適化サブルーチン DALL
2. 虫とりサブルーチン bug
3. ソフトウエア公開条件 Open Market Licence による著作権表示

を紹介した. Ishiguro et al.(1999) に DALL と bug の詳細に関しての情報が含まれているので参考にされたい. ソフトウエアの広い流通のための技術と考えると, 著作権表示は英文を正文とするのが適当と考えてつたない英語にしてある. 改良していただければありがたい.

科学計算の分野における FORTRAN の地位は当面揺らがないと考えられるので, 紹介したサブルーチンはいずれも FORTRAN 版で説明したが, C 版も用意されている. これらのサブルーチンは

`http://www.ism.ac.jp/software/ismlib/soft.html`

で入手できる. OML に関する統計数理研究所の考え方に関しては

`http://www.ism.ac.jp/software/ismlib/licence.html`

を参照されたい. この稿で紹介すべきであったもう一つの技術として「積分技術」があるが, 紙幅の制約もあり触れることが出来なかった. 特に MCMC に関する Ogata(1990), 伊庭 (1996) と下平 (1997) の文献を挙げておく.

参考文献

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle, *2nd International Symposium on Information Theory* (eds. B. N. Petrov and F. Csaki), 267-281, Akademiai Kiado, Budapest. (Reproduced in *Breakthroughs in Statistics*, Vol.1 (eds. S. Kotz and N. L. Johnson), Springer, New York, (1992).)

Davidon, W.C. (1968). Variance algorithm for minimization, *Computer Journal*, **10**, 406-410.

伊庭幸人 (1996). マルコフ連鎖モンテカルロ法とその統計学への応用, 統計数理, **44**, 49-84.

石黒真木夫 (1995). DEBB, a debugging bug, マニュアル, 統計計算技術報告, RSC-26, 統計数理研究所, 東京.

Ishiguro, M. and Akaike, H. (1989). DALL: Davidon's algorithm for log likelihood maximization, *Comput. Sci. Monographs*, No.25, The Institute of Statistical Mathematics, Tokyo.

Ishiguro, M., Sakamoto, Y. and Kitagawa, G. (1997). Bootstrapping log likelihood and EIC, an extension of AIC, *Ann. Inst. Statist. Math.*, **49**, 411-434.

Ishiguro, M., Kato, H. and Akaike, H. (1999). ARdock, an auto-regressive model analyzer, *Comput. Sci. Monographs*, No.30, The Institute of Statistical Mathematics, Tokyo.

Ogata, Y. (1990). A Monte Carlo method for an objective Bayesian procedure, *Ann. Inst. Statist. Math.*, **42**, 403-433.

Sakamoto, Y., Ishiguro, M. and Kitagawa, G. (1986). *Akaike Information Criterion Statistics*, Reidel, Dordrecht.

下平英寿 (1997). ベイズ的方法における MCMC の利用, 応用統計学会第 19 回シンポジウム予稿集, 1-10.

 Proceedings of the Institute of Statistical Mathematics Vol. 47, No. 2, 395-424 (1999)

Technical Aspects of Information Criterion Statistics

Makio Ishiguro

(The Institute of Statistical Mathematics)

Thinking that information criterion statistics consists of steps:

1. development of new model;
2. fitting of the model to data;
3. publication of the findings and the methods employed,

the author composes this article as a package of

1. introduction of debugging support subroutine 'bug' and its tutorial manual;
2. introduction of subroutine program 'DALL' for the numerical maximization of log likelihood and its tutorial manual; and
3. proposals of the use of copyright notice under the Open Market Licence(OML) condition as a part technology for software distribution.
4. 'bug' and DALL are distributed from http://www.ism.ac.jp/software/ismlib/soft.html The information about OML can be obtained from http://www.ism.ac.jp/software/ismlib/ismlib.e.html

Key words: Debugging, log likelihood, numerical optimization, quasi-Newtonian method, copyright notice.