解説

# 回帰分析の変数選択における分枝限定法

吉澤　正

## 1. はじめに

重回帰分析において多数の説明変数の候補から対象とする目的変数を最も良く説明する変数の組合せを探す課題を一般に変数選択の問題とよんでいる．そこでは，大きく分けて“良く説明する”というモデルの良さの規準と，“組合せを探す”ための効率的算法が追求される．ここでは，とくに後者の課題のうち，分枝限定法的算法の一つを紹介したい．

良さの規準については，本誌1978年5月号の特集で種種の解説が載せられているように，情報量規準（AIC），予測平均二乗誤差(MSEP)，予測平方和(PSS)，Mallowsの$C_p$規準，自由度二重調整重相関係数などがさまざまな角度から検討されている．これら多くの規準は，予測平方和を除いて採用する説明変数の個数を一定にすれば，残差平方和(RSS, Residual Sum of Squares)と単調関係にある．したがって，ここでは残差平方和を目的関数として選択する説明変数の個数ごとに最良の組合せを探すことを目標としよう．すなわち，説明変数の全候補を$k$個とし，その中から$p$個選ぶ組合せは二項係数${}_kC_p$とおりあり，それらから RSS を最小にする組合せを最良部分集合とする．

従来からの段階的回帰（Stepwise Regression, たとえば，Jennrich［7］参照），あるいは変数増加法，減少法，増減法，減増法（奥野ほか［9］）などでは，必ずしも最良部分集合が得られない．といって，上述の$p$について1から$k$まで，すべてで$2^k-1$個にものぼる部分集合を総当りするのでは大変な時間がかかる．総当り法の算法についても，Garside［4］，Schatzoffほか［10］をはじめ，Furnival［2］などによって，その効率化が計られていて，大型電子計算機を使えば$k$が15ぐらいまでは実用になろう．

一方，Hocking と Leslie［5］，Beal ほか［1］，LaMotte と Hocking［8］，Furnival と Wilson［3］などは，総当り的算法を土台に進行しながらも，後述の残差平方和に関する基本不等式を上手に使って，一部の組合せについての回帰計算を省略する算法を工夫している．以下では，残差平方和の性質，掃き出し演算，ある種の総当り算法などについての必要最小限の準備に引き続いて，効率の良いとされている Furnival と Wilson による分枝限定法を紹介する

## 2. 残差平方和

$k$個の説明変数 $x_1, x_2, \cdots, x_k$ と一つの目的変数 $y$ について，$n$組のデータ $(x_{\alpha 1}, x_{\alpha 2}, \cdots, x_{\alpha k}, y_\alpha)$，$\alpha=1, 2, \cdots, n$ が用意されているとしよう．説明変数の添字集合を $I$，$I=\{1, 2, \cdots, k\}$ とする．$I$ の任意の部分集合 $J$ について，$y$ の重回帰式

(1)　$\hat{y}=b_0+\sum_{\forall j\in J} b_j x_j$

を最小二乗法で推定したときの残差平方和を RSS $(J)$ と書く．すなわち，

$$\mathrm{RSS}(J)=\sum_{\alpha=1}^{n}(\hat{y}_\alpha-y_\alpha)^2$$

ここで，$\hat{y}_\alpha$ は (1) の右辺の $x_j$ にデータを代入したときの値である．

残差平方和についてはつぎの不等式が成り立つ：

(2)　RSS$(J)\leqq$RSS$(K)$，$J\supset K$のとき．

すなわち，$I$の任意の部分集合$J$と$K$について，$J$が$K$を含むなら，$K$ の RSS は$J$のそれより小さくはなれない．この性質を RSS の基本不等式とよんでおこう．以下では，空集合を$\phi$と書くほか，通常の集合に関する記号を用いる．また，たとえば $K=\{3, 4\}$ のときのRSSを単にRSS (34) などと書く．具体例としての変数番号は1桁の数字を用いるので混乱はあるまい．なお，{3, 4}−{4}は{3}を表わすようにマイナス記号を用いる．

この基本不等式はつぎのように利用される．たとえば RSS (345) が 720，RSS (2) が 702，すなわち，

(3)　RSS(345)>RSS(2)

という情報が得られているとしよう．(2)式より，

$$\mathrm{RSS}(3),\ \mathrm{RSS}(4),\ \mathrm{RSS}(5) \geqq \mathrm{RSS}(345)$$

よって，(3)式の情報から，

$$\mathrm{RSS}(2) < \mathrm{RSS}(3),\ \mathrm{RSS}(4),\ \mathrm{RSS}(5)$$

となり，$p=1$ での1変数の最良部分集合を求めるためには $\{3\}$, $\{4\}$, $\{5\}$ に対する RSS の評価は省略してよいことになる．さらに $p=2$ についても，

$$\mathrm{RSS}(23) \leqq \mathrm{RSS}(2),\ \{2,\ 3\} \supset \{2\}$$

などと，変数2を含む2変数部分集合のRSSは RSS(2) 以下，一方，$\{3,\ 4,\ 5\}$ に含まれる $\{3,\ 4\}$, $\{3,\ 5\}$, $\{4,\ 5\}$などの RSS はRSS(345)以上である．よって，(3)式の情報があれば，RSS(34), RSS(35), RSS(45) の評価も不要であることがわかる．もし，(3)式の情報は未確認で，

$$\mathrm{RSS}(345) \geqq \mathrm{RSS}(23)$$

が確認されれば，$\{3,\ 4,\ 5\}$ の2変数部分集合についての RSS の計算は省略されるが，1変数については省略できない．

残差平方和のこのような性質から，要素数の多い部分集合 $J$ について RSS($J$)が大きく，小変数の$K$については RSS($K$) が小さいものを早く探し出して，

$$\mathrm{RSS}(J) < \mathrm{RSS}(K),\ J \supset K \text{のとき}$$

となることがわかれば，RSS を評価しないですむ割合が増すことになる．

## 3. 掃き出し演算

つぎに，残差平方和の計算について触れておこう．目的変数 $y$ を形式的に $k+1$ 番目の説明変数 $x_{k+1}$ とし，添字集合 $I_+$ を，

$$I_+ = I \cup \{k+1\}$$

と置く．まず，変数間の偏差積和 $a_{ij}$ を求める．

$$a_{ij} = \sum_{\alpha=1}^{n} (x_{\alpha i} - \bar{x}_i)(x_{\alpha j} - \bar{x}_j),\quad \forall i,\ j \in I_+ \tag{4}$$

ここで，$\bar{x}_i\ (\forall i \in I_+)$ は変数 $x_i$ の標本平均である．

後述の算法の説明のために，つぎのような行列と列ベクトルを定義しておく：

$$A = (a_{ij}),\quad \forall i,\ j \in I;\quad \boldsymbol{c} = (a_{i,k+1}),\quad \forall i \in I \tag{5}$$

$$A_+ = (a_{ij}) = \begin{pmatrix} A & \boldsymbol{c} \\ \boldsymbol{c}' & a_{k+1,k+1} \end{pmatrix} \tag{6}$$

$$B_- = (b_{ij}) = \begin{pmatrix} -A^{-1} & A^{-1}\boldsymbol{c} \\ \boldsymbol{c}'A^{-1} & b_{k+1,k+1} \end{pmatrix} \tag{7}$$

ここで，$A^{-1}$ は $A$ の逆行列，$'$ は転置を意味し，

$$b_{k+1,k+1} = a_{k+1,k+1} - \boldsymbol{c}'A^{-1}\boldsymbol{c}$$

である．$b_{k+1,k+1}$ は RSS($I$)，$a_{k+1,k+1}$ は RSS($\phi$) にほかならない．また，$A^{-1}\boldsymbol{c}$ は全変数での回帰の偏回帰係数ベクトルになっている．以下では$A$の逆行列が存在するとして話を進める．

掃き出し演算にはいろいろな定義があるが，ここではつぎのように行列 $(a_{ij})$ を行列 $(\tilde{a}_{ij})$ に変換する演算を変数 $l$ を枢軸とする掃き出し演算とよび，swp[$l$] とも書く．

$$\begin{cases} \tilde{a}_{ll} = -1/a_{ll} \\ \tilde{a}_{il} = a_{il}/a_{ll},\quad \forall i \in I_+ - \{l\} \\ \tilde{a}_{lj} = a_{lj}/a_{ll},\quad \forall j \in I_+ - \{l\} \\ \tilde{a}_{ij} = a_{ij} - a_{il}\ a_{lj}/a_{ll},\quad \forall i,j \in I_+ - \{i,j\} \end{cases} \tag{8}$$

$A_+$ にこの演算を，$l=1,\ 2,\cdots,k$ について（適当な順で）反復適用すれば $B_-$ が得られる．もし，残差平方和 RSS($12\cdots p$) のみが必要なら，つぎの算法＊で掃き出し演算の一部を実行すればよい．

- $l=1,2,\cdots,p$ について
  - $i=l+1,\ l+2,\cdots,\ k+1$ について
    - $j=i,\ i+1,\cdots,\ k+1$ について
      - $a_{ij} \leftarrow a_{ij} - a_{li}\ a_{lj}/a_{ll}$

同様に，$B_-$ から出発して RSS($p+1\cdots k$) を求めるには，上記の手順と同一の反復構造で，

$$b_{ij} \leftarrow b_{ij} - b_{li}\ b_{lj}/b_{ll}$$

を実行すればよい．

そこで，次節以下では，掃き出し演算を，

- $i=l+1,\ l+2,\cdots,\ k$ について
  - $j=i,\ i+1,\cdots,\ k$ について
    - $\tilde{a}_{ij} = a_{ij} - a_{li}\ a_{lj}/a_{ll}$

逆掃き出し演算を，

- $i=l+1,\ l+2,\cdots,\ k+1$ について
  - $j=i,\ i+1,\cdots,\ k+1$ について
    - $\tilde{b}_{ij} = b_{ij} - b_{li}\ b_{lj}/b_{ll}$

と定義して使用する．また，これらの演算を単に枢軸$l$による演算とよんだり，枢軸のある行や列を枢軸行や枢軸列ともよぶことがある．なお，掃き出し演算についての詳しい解説は Jennrich［7］などを参照されたい．

## 4. 総当り法

総当り法には多くの算法があって興味深い話題も尽きないが，ここでは分枝限定法の説明に必要な辞書式順と変形家族式順の算法を述べておく．いずれの算法にしろ掃き出しおよび逆掃き出し演算の必要部分を逐次適用してゆくことを基本としている．

まず，記法について断っておこう．以下の具体例では 4.12のように，点の両側に数字列を並べる記法を用いる．点の右側の列は回帰に採用ずみの変数の部分集合を

＊ 算法の記述では，構造的プログラミングの精神で，反復構造を字下げと縦線で表現する．

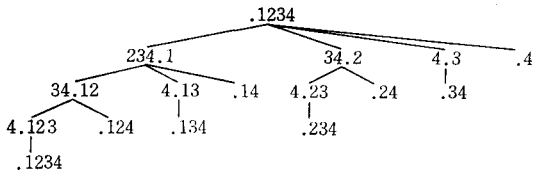



**図1　回帰木**

左側はこれから採用するために残すべき積和行列の行と列の番号を表わす.

例として，4.12の場合は，変数{1,2}についてのRSS（12）が得られており，かつ，つぎに変数4を枢軸として掃き出しを行なえば　.124へ移れるという状態を表わしている．あるいは，$A_+$ から始めて，swp[1]とswp[2]を行ない，その時点の $A_+$ から変数4と目的変数に対応する行と列を抽出した2×2の行列が得られている状態と考えてもよい．実際の算法では，前節で述べたように，残差平方和だけを求めるためには $A_+$ の上（あるいは下）三角部分に関して操作すればよい.

また，　.12345のように，数字列の一部に下線を附した記法を用いる．点の右側にのみ数字列があり，その数字列に対応する集合のRSSが得られており，かつ，下線付きの数字の変数に対して逆掃き出しを行なえる状態を示す．　.12345については，RSS(12345)が得られており，かつ $k$ を5とすれば，　$B_-$ から変数3と4および目的変数に対応する部分が残されていて，変数3と4を枢軸として逆掃き出しを行なえば，　.125 の状態にまで移ることができる.

さて，辞書式順の総当り法は，各段階の状態を木(tree)構造として図1のように表現して，木の節(node)あるいは葉のたどり方としてその算法を考えると理解しやすい．木構造については，通常の用語として，親・子（図では親が上），兄弟，子孫などを用いる．また，最上段の親のない節を根，子のない節を葉ともよぶ.

図1は $k$ が4の場合の一例であるが，その木を回帰木とよぶ．辞書式順はその回帰木を表1の回帰という見出しの欄に示す順にたどる．その欄で点の右側だけを見れば1，12，123,…となっていて，辞書式の名の由来がわかろう．回帰木についてのたどり方はつぎのように再帰的に表現することもできる.

- 木の根をたどる.
- 左の木から順に各副木をたどる.

ここで，一つの木の根を取り除いたとき，残りがいくつかの木に分かれるので，それらを副木とよぶ．図1では，1234.　を除くと，　234.1　を根とする副木から，　34.2，4.3，　.4　をそれぞれ根とする副木ができ，それらを順に処理する．234.1　を根とする副木をたどるには，その根をたどり，つぎにその根を除いた副木を左から順にたどることになり，結局表1の順が得られることがわかろう.

**表1　辞書式順総当り法**

| 相番号 | （2進） | 親行列 | 枢軸 | 保存行列 | 回帰 |
|---|---|---|---|---|---|
| 0 | 000 | 1234. | 1 | 234. | 234.1 |
| | | 234.1 | 2 | 34.1 | 34.12 |
| | | 34.12 | 3 | 4.12 | 4.123 |
| | | 4.123 | 4 | .123 | .1234 |
| 1 | 001 | 4.12 | 4 | .12 | .124 |
| 2 | 010 | 34.1 | 3 | 4.1 | 4.13 |
| | | 4.13 | 4 | .13 | .134 |
| 3 | 011 | 4.1 | 4 | .1 | .14 |
| 4 | 100 | 234. | 2 | 34. | 34.2 |
| | | 34.2 | 3 | 4.2 | 4.23 |
| | | 4.23 | 4 | .23 | .234 |
| 5 | 101 | 4.2 | 4 | .2 | .24 |
| 6 | 110 | 34. | 3 | 4. | 4.3 |
| | | 4.3 | 4 | .3 | .34 |
| 7 | 111 | 4. | 4 | . | .4 |

各節をたどるときは，初めの根1234.　は別にして，その親の節で保存された親行列のとり出し；後に必要な部分行列の保存；一つの変数を枢軸としての掃き出しという三つの仕事が行なわれる．たとえば，表1にそれらの仕事の結果を示したように，4.13では親の節234.1で保存された34.1をとり出し．4.1　を保存し，変数3で掃き出して4.13を作る．この方式では，長男の続く本家の系列はその親を順次に親行列とできる．一方，次男以下の処理ではあたかも分家を立てるように，すぐ上の兄の処理での親行列の一部を保存しておいて親行列とする．そこで，表1では，はじめの4段は本家，あとは分家をたてるたびに横線を引いてある．このように，直系の節の処理をまとめて相とよぶことにする.

表1では，相の番号欄の右にその2進数表現を付けてある．相の番号は0から $2^{k-1}-1$ までであることは分家の数として容易にわかろう．相番号の2進表現を $b_{k-1}\cdots b_2b_1$ としておくと各相の先頭の処理での枢軸が簡単に求まる．すなわち，2進表現の右のビットから見てはじめてゼロでないビットを $b_m$ とする．そのとき，枢軸は，

$$p=k-m+1$$

で求まる．たとえば，$k$ が4，相が2進数100のとき，$p$ は4－3＋1＝2となる．ただし，全ビットゼロの第0相は $m=k$，すなわち，$p=1$ とする.

保存行列用の記憶場所は，わかりやすくするなら，$k+1$個の配列を用意しておき，その節での枢軸番号＋1

番目の配列に保存するようにしておく．そうすれば，各相の先頭での親行列は枢軸と同じ番号の記憶場所に入っていることになる．保存すべき親行列の要素は，掃き出しの対象とする要素と完全に一致している．

以上の算法を電子計算機用プログラミングを念頭に整理するとつぎのように記述できよう．

**辞書式順総当り法の算法**

(6)式の$A_+$が求められているとする．保存行列用の$k+1$個の記憶場所を配列$S_1$, $S_2$, …, $S_{k+1}$とする．$A_+$は$S_1$に代入されているとし，配列$(a_{ij})$を親行列の格納場所として使用する．

- 第0相から$2^{k-1}$個の相につき以下を反復する．
  - 相内の初段の枢軸を相番号の2進表現から定めて$p$とする．
  - $S_p$から親行列を配列$(a_{ij})$に移す．
  - 枢軸$l$を$p$から$k$まで変えながら
    - 保存と掃き出しを行なう．すなわち，$i=l+1, l+2, \cdots, k+1$について
      - $j=i, i+1, \cdots, k+1$について
        - $S_{l+1}(i,j)\leftarrow a_{ij}$
        - $a_{ij}\leftarrow a_{ij}-a_{li}\ a_{lj}/a_{ll}$
    - それまでの最良部分集合を決める．

相についての反復では，相番号の2進表現$b_{k-1}\cdots b_2b_1$を同時に保持し，それに1ずつ加えてゆく．$b_{k-1}\cdots b_2b_1$に1を加えるには，右端の$b_1$から順に見て，1が続く限りそのビットをゼロに変え，はじめてゼロになるビットを1に変える(くり上げを入れる)．前述したようにその時点でつぎの枢軸がわかる．

算法の下段の“それまでの最良部分集合を決める”ための処理には，保存行列の記憶と同時に掃き出しずみの変数番号を記憶しておき，掃き出しの時点でどの部分集合についてのRSSが得られたのか判明するようにしておく必要がある．

つぎに，図2と表2を使って，変形家族式順の総当り法を説明しよう．ここでは，実質的には$k=4$の場合に相当するが，分枝限定法の説明にそのまま使えるように，$k$を5とし，.12345の状態から変数5を常に含むすべての部分集合を総当りする場合を考える．変形家族式順での図2の木のたどり方は，

- その根の子供を末弟から長兄への順にたどる†．
- 長兄を根とする副木をたどる．以下次兄の木から末弟の木までを順にたどる．

と再帰的に表現される．これを実行すれば，表2の回帰の欄に示す順になる．図2の木を限定木とよんでおく．

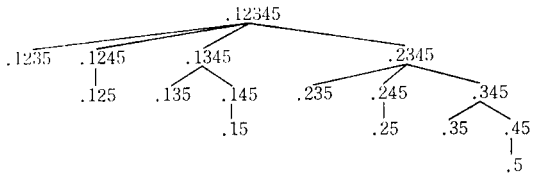



**図2　限定木**

**表2　変形家族式順総当り法**

| 相番号 | (2進) | 親行列 | 枢　軸 | 回　帰 |
|---|---|---|---|---|
| 0 | 000 | .12345 | 1 | .2345 |
| | | | 2 | .1345 |
| | | | 3 | .1245 |
| | | | 4 | .1235 |
| 1 | 001 | .1245 | 4 | .125 |
| 2 | 010 | .1345 | 3 | .145 |
| | | | 4 | .135 |
| 3 | 011 | .145 | 4 | .15 |
| 4 | 100 | .2345 | 2 | .345 |
| | | | 3 | .245 |
| | | | 4 | .235 |
| 5 | 101 | .245 | 4 | .25 |
| 6 | 110 | .345 | 3 | .45 |
| | | | 4 | .35 |
| 7 | 111 | .45 | 4 | .5 |

この方式では，兄弟の節の処理を一つの相とする．各相での枢軸の決め方は辞書式順と一致している．各節での処理では，同一の相内では親行列が一定であり，逆掃き出し演算を行なった後，その結果を保存する．保存すべき要素は逆掃き出しの対象となった要素だけで，本節のはじめに断ったように，回帰欄の下線のある数字に対応する説明変数と目的変数に関する行と列である．保存の方法は，辞書式順の場合と同じように，枢軸＋1番目の配列に保存する方式でよい．親行列は枢軸の値と同じ番目の配列からもってこられる．

## 5. 分枝限定法

以上の準備にもとづいて，Furnival と Wilson [3] による分枝限定法の説明に入ろう．この算法は，$k$個の説明変数の部分集合を第$k$変数を含むものと含まないものと二つのクラスに分け，後者については図1のような回帰木の辞書式順，前者については図2のような限定木の変形家族式順のたどり方を並行して進めてゆく．

† 長兄から末弟の順にたどる算法が家族式とよばれるのにふさわしかろう．

木の各節をたどるときには，2節で述べた残差平方和の性質によって，掃き出しをする必要があるかどうかを判定する．回帰木による辞書式順と限定木による変形家族式順を組み合わせると，両者の同一の相で回帰の行なわれる部分集合は，表1と表2を比べて（あるいは，両表を結合して分枝限定法の手順例を示す表3を見れば），限定木での親行列で得られている部分集合に含まれることがわかる．2節で述べた例は，表3のRSSの値とは異なるが，表1と2の第6相での判定の仕方を示しているので改めて読み直してほしい．一般には，つぎのように判定を行なう．

- 各相のはじめに，限定木側の親行列でのRSSをその相の限界値とする．
- 相内の各段で，回帰木側で求めようとする部分集合の要素数を調べ，同じ要素数でのそれまでの最良部分集合のRSSをその相の限界値と比較する．相の限界値のほうが大きいか等しいなら，その段での掃き出しを省略できる．

その段の処理だけ省略してつぎの段に移ってもよいが，つぎのようにして飛び越しを行なうことができる．たとえば，第2節の例のように表1と表2で第6相の第1段が省略できるとすれば，そこで保存される情報を使う第7相も省略できる．一般に，省略できると判定されたときの枢軸を$l$とすると，$2^{k-l-1}$個の相を省略できる．この数は，限定木でのその節からの分家数（本家を加えて）を数えることによって得られる．

さて，分枝限定法の概略を記述しておこう．

**分枝限定法の算法**

説明変数の個数を$k$とする．目的変数を加えた(6)の積和行列$A_+$，および，(7)の$B_-$が準備されているとする．回帰木での保存行列用記憶場所を配列$S_1$，$S_2$,…，$S_k$，限定木での保存用記憶場所を配列$T_1$，$T_2$,…，$T_k$，作業用配列$A, B$，相の2進カウンター$b_{k-2}\cdots b_2 b_1$を用意する．なお，図1や前節の辞書式順での変数の個数は，ここでの$k$から1を引いたものになることに注意されたい．

- 説明変数の番号を変更する．
- 必要な初期設定を行なう．相をゼロに設定し，変更後の変数で第$k$変数の行と列を除く$A_+$と$B_-$の要素をそれぞれ$S_1$と$T_1$に格納する．目的変数に対応する行と列は第$k$行と列になる．
- 第0相より順次，相が尽きるまで以下を反復
  - 相内の初段の枢軸を$p$とする．
  - 親行列を決める．$S_p$と$T_p$をそれぞれ$A$と$B$に移す．
  - 枢軸$l$を$p$から$k-1$まで変えながら
    - 相の限界値による判定の準備を行なう．
    - 判定の結果，必要なら
      - 回帰木の処理．$S_{l+1}$に$A$の第$l+1$以上の行と列の要素を保存し，$A$を枢軸$l$で掃き出す．
      - ▶ 限定木の処理．$B$を枢軸$l$で逆掃き出しを行ない，結果を$T_{l+1}$に格納する．
      - それまでの最良部分集合を更新する．
    - 判定の結果，必要でないなら，つぎの相の処理へ．
  - つぎの相を決める．
- 最終結果を印刷して終り．

以下に，個々の処理で必要と思われる事項に説明を追加しておこう．

“説明変数の番号を変更する．”

- $B_-$の要素について

$$-b^2_{i,k+1}/b_{i,i},\quad \forall i \in I$$

を下降順に並べて，その順に変数番号をつけ直す．上記の量の最大のものが1番となる．

このことは，全説明変数から一つの変数のみを除いたときの残差平方和RSS（$I-\{i\}$）について下降順に並べることにほかならない．すなわち，

$$\mathrm{RSS}(I-\{i\})-\mathrm{RSS}(I)=-b^2_{i,p+1}/b_{i,i}$$

である．あるいは，全変数での偏回帰係数の有意性の検定に用いる通常の$t$値を用いるといってもよい．

“相の限界値による判定の準備を行なう．”

相の限界値は配列$B$の$(k,k)$要素にある．回帰木側でつぎに回帰の行なわれる説明変数の部分集合の要素数を$q$とする．要素数$q$でのそれまでの最良部分集合のRSSを$Rq$とする．そこで，

$$Rq \leqq B(k,k)$$

なら，飛び越し，そうでなければ，その段の処理を行なう．

“それまでの最良部分集合を更新する．”

要素数$q$ごとにそれまでの最良部分集合とそのRSS，$Rq$を保持する．親行列になるものが，すでに枢軸となった変数番号とその個数の情報をもつように処理しておく必要がある．

“回帰木の処理”と“限定木の処理”

前節で述べたように，処理の対象となるのは各行列の上三角部分であり，つぎのように書ける．

- $i=l+1$，$l+2$,…，$k$について
  - $j=i$，$i+1$,…，$k$について
    - $S_{l+1}(i,j) \leftarrow A(i,j)$
    - $A(i,j) \leftarrow A(i,j)-A(l,i)\times A(l,j)/A(l,l)$
    - $T_{l+1}(i,j) \leftarrow B(i,j)-B(l,i)\times B(l,j)/B(l,l)$

表 3　分枝限定法の例

| 相番号 | 2進 | 枢軸 | 回帰行列 | 回帰 | 木RSS | 限定行列 | 回帰 | 木RSS | 相の限界値 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 000 | *1 | 1234. | 234.1 | 810 | .12345 | .2345 | 701 | |
| | | *2 | 234.1 | 34.12 | 627 | | .1345 | 633 | |
| | | *3 | 34.12 | 4.123 | 607 | | .1245 | 625 | |
| | | *4 | 4.123 | .1234 | 603 | | .1235 | 605 | |
| 1 | 001 | 4 | 4.12 | .124 | 626 | .1245 | .125 | 626 | 625 |
| 2 | 010 | 3 | 34.1 | 4.13 | 641 | .1345 | .145 | 668 | 633 |
| | | 4 | 4.13 | .134 | 638 | | .135 | 633 | |
| 3 | 011 | 4 | 4.1 | .14 | 678 | .145 | .15 | 677 | 668 |
| 4 | 100 | *2 | 234. | 34.2 | 997 | .2345 | .345 | 702 | 701 |
| | | 3 | 34.2 | 4.23 | 953 | | .245 | 701 | |
| | | 4 | 4.23 | .234 | 701 | | .235 | 807 | |
| 5 | 101 | 4 | 4.2 | .24 | 701 | .245 | .25 | 814 | 701 |
| 6 | 110 | *3 | 34. | 4.3 | 993 | .345 | .45 | 704 | 702 |
| | | 4 | 4.3 | .34 | 702 | | .35 | 809 | |
| 7 | 111 | *4 | 4. | .4 | 705 | .45 | .5 | 815 | 704 |

*印は掃き出しを実行した段を示す.

実際のプログラムでは，$\{S_1,\cdots,S_k\}$ と $\{T_1,\cdots,T_k\}$ はそれぞれにまとめて一つの1次元配列とし，格納時の先頭のアドレスを保持すれば記憶場所の節約になる．

“つぎの相を決める”.

飛び越しのないときは，2進カウンターに1を加え，飛び越しのあったときは $2^{k-l-1}$ を加える．$l$ は枢軸の番号である．すなわち，

- 飛び越しのないときは $\lambda=1$，あったときは $\lambda=k-l$ とする．
- $m=\lambda, \lambda+1,\cdots,k-2$ について
  - $b_m=0$ なら，この反復を終り，$b_m$ を1にしてつぎの相の処理へ移る．
  - $b_m=1$ なら，$b_m$ を0に変える．
- 最終結果の印刷の処理へ移る

## 6. 数 値 例

最後に，数値例を示しておこう．$k$ を5とした例を表3に示す．残差平方和は600以上999以下になるように変換してある．変数番号の変更処理はすんでいるとする．第0相は，それまでの最良部分集合は何も得られていないので，必ず実行される．第1相では，相の限界値は限定木の親行列 .1245から625となり，変数3個のそれまでの最良部分集合 {1,2,3} のRSS，607と比べ，限界値のほうが大きいので飛び越す．この場合は枢軸が4であるから $2^{k-4-1}=1$ で，つぎの相へ移るにすぎない．第2相の初段では，それまでの2変数の最良部分集合 {1,2} の RSS，627を限界値633(=RSS (1345)) と比べ，飛び越してよいことがわかる．枢軸が3であるから $2^{k-3-1}=2$相分飛び越して第4相へ移る．ここで，限界値701=RSS (2345) はそれまでの1変数の最良部分集合 {1} の RSS，810 より小さい．そこで掃き出しを行なう．以下，つぎの段は飛び越せて，第6相へ移り，第6相の初段は実行して，第6相の2段目で飛び越して，第7相へゆき，そこを実行して終る．表3の枢軸欄で*印をつけた段のみが実行され，全体15段中7段で掃き出しが行なわれたことになる．

この例では，1変数の最良部分集合が {4} であり，そのために，最後の相まで実行する必要を生じた．たとえば，第4相以下での枢軸は2，3，および4だけであるので，第4相の実行に先だって，2，3，4の変数について当初と同じような原理で変数番号を変更するとよい．この場合，限定木での親行列 .2345で

$$-B(i,k)^2/B(i,i),\quad i=2,3,4$$

を下降順に並べて，最大のものを改めて2，最小のものを4番とする．表3での RSS を使うなら，第4相の限定木側での3個の RSS，702，701，807 が第1相でのように大きいほうが上にくるように番号を変更することになる．そうすると，これまでの第4変数が第4相初段

の枢軸となり，この段だけ実行して，残りはすべて省略できることになる．現実のプログラムでは，変数番号の変更を初段の枢軸番号が $k-1$ や $k-2$ 以外の相について実行するとよい．

また，実際のプログラムに追加すべき機能はいろいろあろうが，たとえば，最良部分集合を1個だけでなく，最良5位まで求めるというような処理は簡単に追加できる．

## あとがき

最近，わが国でも普及しているBMDなどの最新版BMDPなどにこの種のプログラムが入れられるようになり，関心のある方も多くなっているようである．紙数の都合もあって，説明の不十分な点もあろうが，本解説が読者のいささかのお役に立てば幸いである．

## 参考文献


[1] Beal, E.M. et al. (1967). The discarding of variables in multivariate analysis. *Biometrika* **54**, 357-365.

[2] Furnival, G.M. (1971). All possible regressions with less computation. *Technometrics* **13**, 403-408.

[3] Furnival, G.M. & Wilson, R.W.Jr. (1974). Regressions by leaps and bounds. *Technometrics* **16**, 499-511.

[4] Garside, M.J. (1965). The best subset in multiple regression analysis. *Applied Statistics* **14**, 196-200.

[5] Hocking, R.R. & Leslie, R.N.(1967). Selection of the best subset in regression analysis. *Technometrics* **9**, 531-540.

[6] Hocking, R.R. (1977). Selection of the best subset of regression variables. Enslein, K., Ralston, A. & Wilf, H.S. (eds.). *Statistical Methods for Digital Computers*, John Wiley & Sons, 39-57.

[7] Jennrich, R.I.(1977). Stepwise Regression. *ibid.*, 58-75.

[8] LaMotte, L.R. & Hocking, R.R. (1970). Computational efficiency in the selection of regression variables. *Technometrics* **12**, 83-93.

[9] 奥野忠一ほか(1971). 多変量解析法. 日科技連出版社.

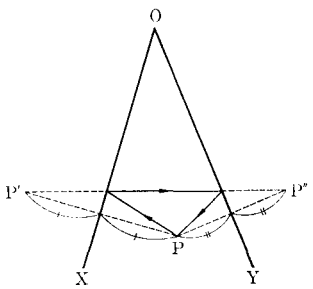
[10] Shatzoff, M. et al.(1968). Efficient calculations of all possible regressions. *Technometrics* **10**, 768-779.



（よしざわ・ただし　山梨大学工学部計算機科学科）


# 数理パズルを楽しもう (17)

**問題**　図のように，点Pの両側にOXとOYのガケがあります．良雄君が，点Pから出発して，両側のガケぶちを見たあと，出発点に戻る計画をたてました．点Pの鏡像を使うと，いちばんの近道は簡単に求められます．しかし，これではガケから落ちてしまって，実用的ではありません．最短路を求める実用的な方法を捜して下さい．




〔2月号(102ページ)の解答〕　マッチ棒で直角を作る問題は，すでに第14回に出題した．また，マッチ棒による直線も正三角の敷きつめから作れるので，図aの状態までは簡単である．問題は，点Aと点Bをマッチ棒で結ぶことである．まず，点Bを出発点として，点Aのおよその方向に5本のマッチ棒を直線に並べる．つぎに，その直線の先端から，点Aを通るようにマッチ棒を置く(図bの太線のマッチ)．これから，マッチ棒による菱形をつぎつぎと作り，点Aから菱形の対辺に向かってマッチ棒を順次に置けば，5本目の先端がちょうど点Aと一致する直線となる．なお，少し工夫すると，15本のマッチ棒だけでも，目的の図形は作れる．

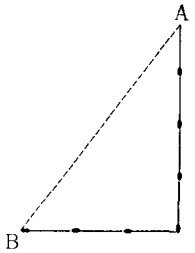



図 a

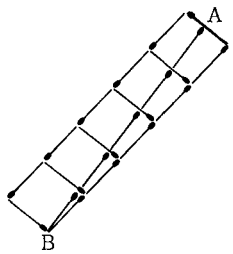



図 b


（中村義作・信州大学工学部）