「区画」をとらない「率」推定 (最初の一歩篇)

久保拓弥 kubo@ees.hokudai.ac.jp

2000 年 11 月 21 日

1 従来の手法の問題点と対策

森林生態学では死亡率や開花率などいろいろな「率」をモデル化することがよくある．そのやりかたを類形化してみると……

- サイズや「明るさ」に関していくつかの「区画」を設け，観測された樹木個体をその区画に分ける．
- 区画ごとに死亡率だの開花率だのを計算する (よくある例：死亡率 = その区画で死んだ個体/その区画の個体数)．
- 区画の代表値を独立変数，区画の死亡率を従属変数として直線や曲線をあてはめる．

このやり方にはいくつかの問題点がある．

- 「区画」のとりかたに依存して「率」の推定値やあてはめの結果が変わる．
- 「(部分)/(全体)」として求めた「率」は等分散性が成立しないので，ふつーの最小二情報を適用できない．

後者の問題を解決するために「比率のあてはめならロジット変換 (ロジスティック回帰) をやればよい」とする人もいる．その場合でも前者の「区画依存性問題」は残る．さらにロジット変換「しか」使わないとすると，他にもっと「あてはまりのよい」モデルがあっても気づかない場合がある．

この「区画取り→率計算→あてはめ」というパターンを脱して，より妥当なモデリングを行うためには，まずこういう「率モデル」がどういう確率現象 (確率論的モデル) を取り扱っているかを理解しなければならない．とはいえ，これはちっとも難しいことではない．

自分がどういう確率論的モデルを使っているかよく自覚できたならば，次にはそれを利用してデータからパラメーターを推定できる．その方法がいくつかある．

ひとつは一般化線形モデル (GLM あるいは GLIM, Generalized Linear Model) に帰着させることである．筆者はこれをよく理解してないので，ここでは解説しない (…… 勉強中)．しかし一般化線形モデルを使いこなすことができるようになれば，freeware の統計学パッケイジ R の glm 関数を利用することで簡単にパラメーター推定ができるようになるだろう．

もうひとつの方法は尤度関数を自分で書いてしまって，mlfitting (この計算プログラムについては後述) などの最尤推定値を求める方法がある．ここではこちらの，そう，どちらかと言えば原始的な方法について解説する．

一般化線形化モデルもその基本は最尤法なので，R で一般化線形モデルを使おうと考えている人たちにとってもこの解説は有用だろう．一般化線形モデルだけでなく，言ってしまえば，たいていの統計学的手法はこの最尤法的な発想に基づいている．

2　データと確率論的モデル

ここでは樹木の死亡率を推定する問題を考えよう．そしてその率がサイズに依存していると仮定する．もちろん読者が自分の問題を考えるときには，死亡率でなくて開花率などでもよいし，その率を説明しようとする要因がいくつあっても同じことである．

たとえば次のようなデータがあったとしよう．

個体番号	サイズ	死亡してる?	(死亡確率)
1	x_1	M_1	$m(x_1)$
2	x_2	M_2	$m(x_2)$
$\vdots$			
i	x_i	M_i	$m(x_i)$
$\vdots$			
N	x_N	M_N	$m(x_N)$

つまり i 番目の個体のサイズ (まぁ例えば胸高直径とか) が x_i である，としている．ある年度の観測から次の観測までの間 (1 年間としましょうか) に死んだ個体 i の M_i は YES で死ななかったものは NO という記号がわりあてられる ($M_i \in \{$ YES, NO $\}$)．個体は 1 番から N 番までの N 個体.

つまり，上の表はよくあるデータセットを少し一般化してるだけなのである．

最後の列に $m(x_i)$ という関数が割り当てられている．これは今から考えてるいろいろな確率論的モデルにおいて具体的な形が与えられる死亡確率であり,「サイズ x_i の個体が 1 年間に死ぬ確率」という意味である．

3　尤度関数

さてこれからパラメーター推定のために用いる最尤法では「データが上の表のように観測された確率」というものを考える．これを尤度 (ゆーど, likelihood) という．この尤度が最大になるようにパラメーターを調節し (データは変更できないんでパラメーターの値を動かして尤度を変えてみる),

尤度が最大になるパラメーターの値を最尤推定値 (maximum likelihood estimate) という．ところで,

いま考えている例では死亡確率 $m(x_i)$ は関数なので最尤推定量 (maximum likelihood estimator) と呼ばれる．名前はともかく, 推定の最終段階ではこの最尤推定量を求める計算問題になってしまうのである．

さて，その「観測データが実現する確率」である尤度は尤度関数という形式で書かれる．いま考えている事例だとこのようになる．

$$L = \prod_{i=1}^{N} f(m(x_i))$$

というカケ算の形になっている．この $f(m(x_i))$ は何かというと，

$$f(m(x_i)) = \begin{cases} m(x_i) & \text{if } M_i = \mathsf{YES} \\ 1.0 - m(x_i) & \text{if } M_i = \mathsf{NO} \end{cases},$$

つまり「死んでいたら死ぬ確率 $m(x_i)$，生き残っていたら死なない確率 $1.0 - m(x_i)$ を示しなさい」というほどの意味である．

ここでちょっとごく単純化した例を考えてみよう．いまもしサンプルが 3 個体 だとして ($N = 3$, こんな数でパラメーター推定するもんじゃないが)，1 番目以外の個体では「死んだ」が観測されたとしよう．すると上の尤度関数は

$$L = (\text{死なない確率}) \times (\text{死ぬ確率}) \times (\text{死ぬ確率})$$

となり，尤度 L が「観測データをあらわしている確率」だとわかるのではないかと思う．

「最尤原理」なる大げさな名前のついた最尤推定の大前提がある．これは尤度 L を最大化するようなパラメーターが「最も尤も (もっとも) らしい」ハズ，とういようなことを言ってるようだ．ともかく $m(x_i)$ なんかをいろいろと変えてみて，L が一番大きくなるような $m(x_i)$ (記号としては $\hat{m}(x_i)$) を探しましょう，という発想なのである．

4 最尤推定量

さて尤度関数がカケ算のかたちでは取り扱いに面倒なので，ふつうはこれの両辺の対数をとった対数尤度

$$\log L = \sum_{i=1}^{N} \log f(m(x_i))$$

を計算する．対数 log は単調な増加関数なので，対数尤度を最大化するものもやはり最尤推定量 $\hat{m}(x_i)$ である．ということで，尤度 L のかわりに対数尤度 $\log L$ を最大化する $\hat{m}(x_i)$ を求める問題にすりかえてしまおう．

ところで，ここまでは「サイズ x_i によって決まる死亡確率」$m(x_i)$ の関数形を具体的に決めていないので，これ以上計算を進めようがない．以下ではいくつかの簡単な $m(x_i)$ とその最尤推定の例を示してみる．

4.1 例: 死亡率が定数の場合

これはとてつもなく簡単で，死亡確率が一定の値，

$$m(x_i) = c$$

つまりサイズ x_i などとは無関係に死亡確率が決まっている場合である．この場合は対数尤度 $\log L$ を最大化するようなパラメーター c を決めてやればよい．尤度 L は c の関数 $L(c)$ となる．

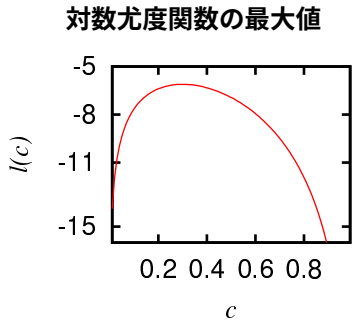
対数尤度を $l(c)$ と記すことにすると，

$$l(c) = \log L(c) = \sum_{i=1}^{N} \log f(c)$$

となるわけだが，$f(c)$ は「死んでたら c，生きてたら $1-c$」という値をとるので，

$$l(c) = (\text{死んだ個対数}) \times \log c + (\text{生きてた個対数}) \times \log(1-c)$$

というところまで簡略化される．はしょって言うと，右辺は極値をもつ関数形になっており，



となっている．極大値が最大値になっている．上のグラフの縦軸は対数なので，最大値において「観測データが実現する確率」とやらである尤度はケタちがいに高くなりそう，ということがわかる．

この最大値である極大値を求めるには，上の対数尤度関数の両辺をパラメーター c で偏微分してゼロとなる c ($\hat{c}$ としよう) を求めてやればよい．まずは両辺を微分して，

$$\frac{\partial l(c)}{\partial c} = \frac{(\text{死んだ個対数})}{c} - \frac{(\text{生きてた個対数})}{1-c}$$

このようなスコアー関数を得る．これがゼロとなるような $\hat{c}$ を計算してみると，

$$\frac{(\text{死んだ個対数})}{\hat{c}} - \frac{(\text{生きてた個対数})}{1-\hat{c}} = 0$$

から，

$$\frac{1-\hat{c}}{\hat{c}} = \frac{(\text{生きてた個対数})}{(\text{死んだ個対数})}$$

となり，最終的には，

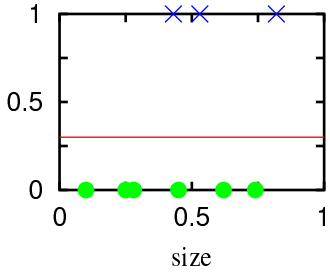
$$\hat{c} = \frac{(\text{死んだ個対数})}{(\text{生きてた個対数}) + (\text{死んだ個対数})}$$

となる．なんのことはない，いろいろともってまわった計算などやって，結局のところ比率の計算をやっているのである．

つまり死亡確率 $m(x_i)$ がサイズ x_i などに影響されない定数 c であると考えている確率論的死亡モデルの場合には，定数である死亡率 c の最尤推定値 $\hat{c}$ は (死んだ数) / (全個対数) でよい，と正当化され「最も尤もらしい」モデルだとわかった．

なお，このときに最大化された対数尤度は $\hat{l}(\hat{c}) = (\text{死んだ個対数}) \times \log \hat{c} + (\text{生きてた個対数}) \times \log(1-\hat{c})$ となり，この値が大きい (ゼロに近い) ほど「モデルの当てはまりが良い」と評価される．

定数死亡率モデルをあてはめた例



• が生きていた個体，× は死んだ個体をあらわしている．上のように 10 個体中 3 個体が死んでいる場合，定数死亡率 $c = 0.3$ となる．最大化された対数尤度は

$$3 \times \log(0.3) + 7 \times \log(0.7) \approx -6.11$$

となる．これが「定数死亡率という確率論的モデルへのデータのあてはまりのよさ」をあらわしている (ゼロに近いほどあてはまりがよい)．

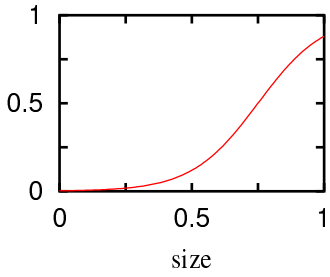
4.2 例: 死亡率の関数がロジスティック型

もうちょっと複雑な死亡モデルの例として,

$$m(x_i) = \frac{1.0}{1.0 + \exp(a - bx_i)}$$

という死亡モデルを考えてみよう．何やらややこしそうな関数型だけど下のように図示してみるとわかるように……

ロジスティック型の関数の例 $(a = 6, b = 8)$



このようなよくある曲線で死亡率を表すモデルである．

この場合でも,「死亡率は定数」モデルと同じように，尤度関数・対数尤度関数を書いていけばよい．尤度関数は

$$L(\mathbf{x}|a, b) = \prod_{i=1}^{N} f(m(x_i))$$

対数尤度関数は

$$l(\mathbf{x}|a, b) = \sum_{i=1}^{N} \log f(m(x_i))$$

あるいは

$$l(\mathbf{x}|a, b) = \sum_{\text{死んだ個体}} \log(m(x_i)) + \sum_{\text{生残した個体}} \log(1 - m(x_i))$$

となる．この対数尤度 $l(\mathbf{x}|a, b)$ を最大化するようなパラメーター $\hat{a}$ と $\hat{b}$ を決めなくてはならない．パラメーターがふたつあるので，$\partial l(\mathbf{x}|a, b)/\partial a = 0$ と $\partial l(\mathbf{x}|a, b)/\partial b = 0$ の連立方程式を解けばよいのである．

しかしながら，一般にこのような連立方程式は記号操作では解が得られない．そこで，計算機に数値計算させることによって最尤推定値 $\hat{a}$ と $\hat{b}$ の近似値を求めることになる．(2002.09 月後記: モ

デルが線形である場合については，複数のパラメーターの推定値を簡単に得られる logistic.pm という Perl モジュールを開発した．計算方法は以下で説明しているものと同じ．この文書をおいてるペイジを参照).

数値計算による求解では，まず a と b に適当な値を初期値として与える．そしてそれらを何らかの方法で変化させながら，次第に最尤推定値 $\hat{a}$ と $\hat{b}$ に近づけていく，という方法が取られる．

以下では mlfitting(2000 年 11 月版) における最尤推定値を得る数値解法の概略を説明する．mlfitting では数値微分による Newton-Rapthson 法を採用している．そのため利用者は対数尤度関数を指定するだけで数値計算を実行できる (スコアー関数など計算しなくてもよい).

mlfitting では数値微分によって求めた対数尤度関数の微分値 (すなわちスコアー関数の値) がゼロになるように Newton-Raphson 法でパラメータセットを変化させている．すなわち，スコアー関数を

$$u_i(\mathbf{x}|\mathbf{p}) = \frac{\partial l(\mathbf{x}|\mathbf{p})}{\partial p_i}$$

と定義 ($i = 1, 2, ..., N$)，すると $\mathbf{u}(\mathbf{x}|\mathbf{p}) = \mathbf{0}$ を満たすパラメーターセット p を求めればよい．

最尤推定量付近でスコアー関数 u とパラメーターセット u の関係を調べてみよう．ここで p の近傍における u の Taylor 展開は

$$\mathbf{u}(\mathbf{x}|\mathbf{p} + \delta\mathbf{p}) = \mathbf{u}(\mathbf{x}|\mathbf{p}) + \mathbf{J}\delta\mathbf{p} + O(\delta\mathbf{p}^2)$$

となる．ただし J は

$$J_{i,j} = \frac{\partial u_i(\mathbf{x}|\mathbf{p})}{\partial p_i \partial p_j}$$

を要素とする行列である．上の展開式で $\mathbf{u}(\mathbf{x}|\mathbf{p} + \delta\mathbf{p})$ をゼロと置き，また $O(\delta\mathbf{p}^2)$ が小さいものとして無視すると，$\delta\mathbf{p}$ は

$$\delta\mathbf{p} = -\mathbf{J}^{-\mathbf{1}}\mathbf{u}(\mathbf{x}|\mathbf{p})$$

を満たすことがわかる．

数値最尤推定プログラム mlfitting ではパラメーターセット $\mathbf{p}$ にてきとうな初期値を与えてやり，多次元版の Newton-Rathon 法

$$\mathbf{p}^{\mathrm{new}} = \mathbf{p}^{\mathrm{old}} + \delta\mathbf{p}$$

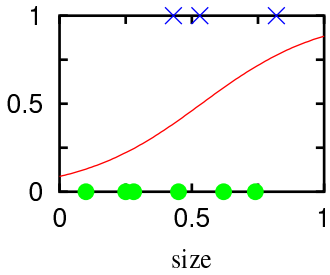
を繰り返すことによってスコアー関数をゼロ ($\mathbf{u}(\mathbf{x}|\mathbf{p}) = \mathbf{0}$) にするような $\hat{\mathbf{p}}$ を求めている．上述したように偏微分は数値微分法で計算し，また $\mathbf{J}$ の逆行列は Gauss-Jordan 法によって求めている．

数値最尤推定プログラム mlfitting の利用者が注意しなければならない点は以下のとおりである．

- 対数尤度関数を定式化してやり Plugin 関数のコードに C++ で正確に書く (mlfitting は対数尤度関数を計算する部分を実行時に動的に読み込む Plugin モジュールとしている)
- データ x のファイルを準備する．
- パラメーター設定ファイルを書き，各パラメーターごとに以下の量を決めなければならない．
  - 初期値
  - パラメーターの上限値・下限値
  - 数値微分の幅
  - 加減速値 (これは上述の Newton-Raphson 法の式の最後の項にかかる係数で，どの程度の「速さ」で p_i を動かすかを決定する)
- いったん最尤解が得られたら，初期値や加減速値を変えて同じ値が得られるかどうか確認してみる．

このような点に注意しつつ mlfitting を使用すると，たとえば以下の図のように複雑な関数型を仮定する死亡モデルであっても最尤推定法を適用できる．

最尤推定されたロジスティック型関数



(とりあえずここまで)