第1章　课程介绍

这门课的英文名字我们应该叫做 Statistical Computing and Software Package. 统计计算不仅是统计学本科专业的一门重要基础课程, 而且越来越多的理工学院、商学院、经济学院、医学院等某些专业本科及研究生也都选修此课程. 近些年,随着计算机的快速发展和统计方法的丰富,统计计算方法已得到了很快的发展和重视, 产生了许多实用的且得到广泛应用的统计计算方法, 如 EM 算法、Bootstrap方法、MCMC方法等. 本门课程包含当今统计计算和计算统计中所涉及的广泛且十分有用的多个内容. 力求让学生们理解现有方法的机理及有用的原因,并让大家能有效地利用这些现代统计方法和统计软件进行更深层次的理论研究和应用工作. 该门课的主要内容实际上包扩如下三个方面：

- 统计计算-Statistical Computing
  - 计算机的储存与运算系统（computer number systems）
  - 算法与编程（algorithms and programming ）
  - 数值近似（numerical approximation）
  - 数值线性代数分析（numerical linear algebra analysis）
  - 非线性方程数值解及优化方法solution of nonlinear equations and optimization
  - 随机数的生成（generation of random numbers）.
- 计算统计-Computational Statistics
  - 蒙特卡洛模拟方法（Monte Carlo Methods）
  - EM 算法（EM algorithm）
  - Bootstrap方法
  - 函数估计（Estimation of Functions）
  - 参数及非参数的密度函数估计（Parametric and Nonparametric Estimation of Probability Density Functions）
- 统计软件-Statistical Software
  - R语言简介包括：数据输入输出,控制流,矩阵运算及常用函数
  - 数据描述性统计分析,参数估计及假设检验
  - R 作图
  - 线性模型
  - 多元分析

例1：极大似然估计 极大似然估计是统计推断的核心,学习MLE的理论表现和其解析形式的导出都需要若干时间,然而,当面临没有解析形式的复杂似然时, 多数人都不知如何处理. 对于极大似然估计, 对数似然函数 l,$\mathbf{x}$ 对应着参数向量 $\boldsymbol{\theta}$. 如果 $\hat{\boldsymbol{\theta}}$ 是MLE,则它最大化其对数似然,即 $\hat{\boldsymbol{\theta}}$ 是得分方程

$$\boldsymbol{l}'(\boldsymbol{\theta}) = \mathbf{0} \tag{1.1}$$

的解,其中 $\boldsymbol{l}'(\boldsymbol{\theta}) = \left(\frac{\mathrm{d}l(\boldsymbol{\theta})}{\mathrm{d}\theta_1}, \ldots, \frac{\mathrm{d}l(\boldsymbol{\theta})}{\mathrm{d}\theta_n}\right)^T$,$\mathbf{0}$ 是元素为0的列向量.

例2：LASSO

考虑如下的多重线性回归模型

$$y_i = \boldsymbol{Z}_i\boldsymbol{\beta} + \varepsilon_i, \quad \text{for } i = 1, 2, \ldots, n,$$

其中y_i, $\boldsymbol{Z}_i$, and $\boldsymbol{\beta}$分别为响应变量,协变量,回归系数, 而 ε_i 为 i.i.d. 服从$N(0, \sigma^2)$的随机误差. 为估计 $\boldsymbol{\beta}$ 如下的惩罚最小二乘估计现在颇为流行：

$$g(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \boldsymbol{Z}_i\boldsymbol{\beta})^2 + n\sum_{j=1}^{p}\gamma|\boldsymbol{\beta}^{(j)}|,$$

其中 $\boldsymbol{\beta}^{(j)}$ 表示 $\boldsymbol{\beta}$ 的 j-th个元素. $\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} g(\boldsymbol{\beta})$. 由于后面的惩罚项在零点不可导,标准的牛顿法此时无法使用.然而我们可将这样的方程转化为带有约束的非线性最优化问题,我们将介绍对这样问题的通用的解决方法.

例3：自助法（Bootstrap）

令 $\theta = T(F)$ 为我们所感兴趣的关于分布函数 F 的某一特征, 被表示为 F 的一函数. 比如,$T(F) = \int z dF(z)$ 是分布的期望. 令 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ 为观测数据,其可看作为随机变量 $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \sim$ i.i.d.F 的实现. 统计推断的问题通常是根据 $T(\widehat{F})$ 或某个 $R(\mathcal{X}, F)$ 提出来的,这里 $R(\mathcal{X}, F)$ 是依赖于数据和它们的未知分布函数 F 的统计函数. 举例来说,一个一般的检验统计量可以为 $R(\mathcal{X}, F) = [T(\widehat{F}) - T(F)]/S(\widehat{F})$,其中 S 为一估计 $T(\widehat{F})$ 的标准差的函数.

随机变量 $R(\mathcal{X}, F)$ 的分布可能难以处理或者根本就是未知的. 这个分布也许也依赖于未知分布 F. *bootstrap* 方法提供了关于 $R(\mathcal{X}, F)$ 的分布的一种近似,其是由观测数据的经验分布函数 (本身是 F 的估计) 所导出的.

例4：Monte Carlo 模拟指数加权移动平均法 假设 $X_i, i = 1, \ldots,$为一列同分布的随机变量. 指数加权移动平均 (exponentially weighted moving average) 法定义为

$$Z_i = (1 - \lambda)Z_{i-1} + \lambda X_i,$$

其中$Z_0 = E(X_i)$.我们通常关心的是所谓的run-length及其抽样性质

$$RL = \inf_t\{t : Z_t > h\}.$$

其中 $E(RL)$ 和 $\mathrm{var}(RL)$ 很多时候非常重要. 然而,它们的精确的性质往往非常难以得到,这个时候我们就求助于 Monte Carlo 模拟来帮助我们获得RL的分布信息.

例5: EM算法

我们收集到数据$Y = \{Y_1, \ldots, Y_m\}$, 其中我们假设$Y_i = (n_i, X_i)$ 来自于如下的模型

$$X_i \overset{\text{i.i.d.}}{\sim} \begin{cases} 0, & \text{with probability } 1-p \\ \text{Binomial}(n_i, r), & \text{with probability } p, \end{cases}$$

不难得到 X_i的密度函数为

$$f(X_i; p, r) = [1 - p + p\text{BIN}(X_i, n_i; r)]I(X_i = 0) + p\text{BIN}(X_i, n_i; r)I(X_i > 0).$$

直接求p, r的极大似然估计（使用牛顿法等）通常收敛会很慢，原因是似然函数中示性函数的存在（导致似然函数关于样本不连续），此时著名的EM算法将会很有效。

例6: 局部线性光滑方法

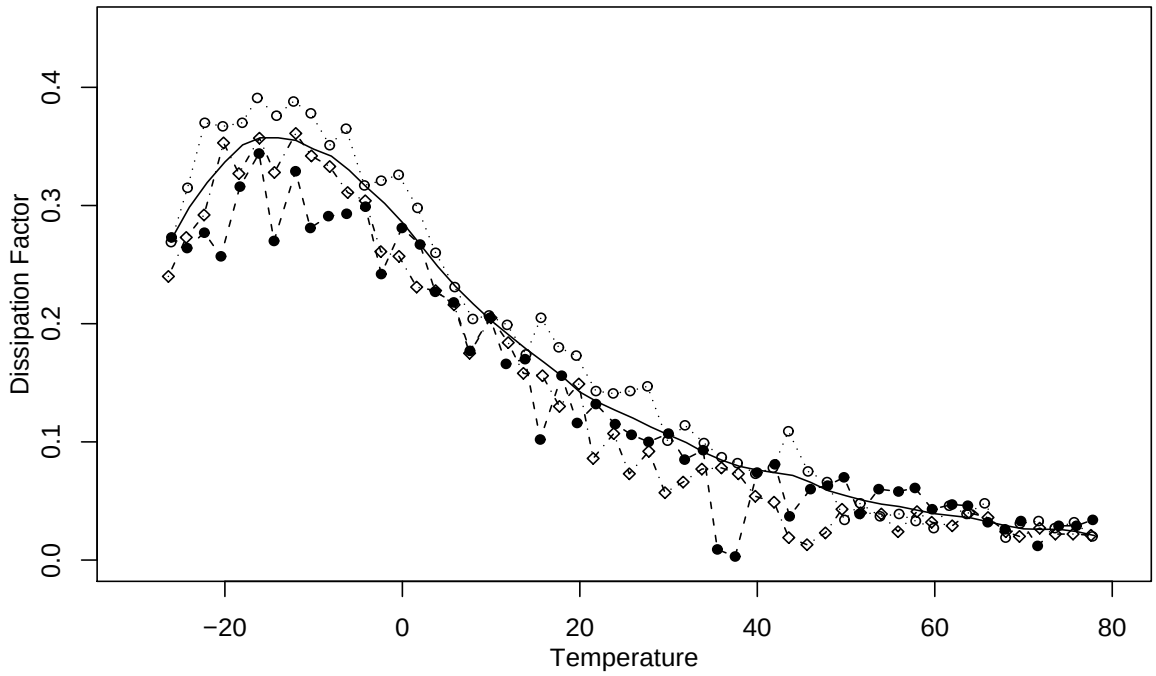

图 1.1: Three AEC curves (lines connecting points with three different symbols) and the estimate (solid curve) of the population profile function.

简单的非参数回归模型:

$$y_{ij} = g(x_{ij}) + \varepsilon_{ij},\ i = 1, 2, 3,\ j = 1, \ldots, n_i$$

其中 ε_{ij}为误差项. 我们可使用样条或局部线性多项式方法用已有数据来估计未知函数 $g(\cdot)$. 并且我们还将学习如何将很复杂的函数及高维数据有效地直观地用图形表示出来. （统计学科现在发展中的新出现的分支为图形统计,即 graphical statistics. Journal of computational and graphical statistics）

由于考虑到 R 的实用性及课时的限制,故我在这门课中省略了本领域过去或现在的某些重要研究内容, 如数值积分,随机数的产生,线性方程组的求解等. 我将重点讲述如下内容：

- R语言简介
  - R 环境, 基本运算及读写数据文件（1课时）
  - 数组矩阵运算与控制流语句（1课时）
  - 定义函数及实例（1课时）
- 数据描性分析
  - 描述性统计量（1课时）
  - 数据分布及作图表示（2课时）
- R 作图及假设检验（3课时）
- 非线性方程数值解及优化方法（9课时）
- 函数逼近和数值积分（6课时）
- R 中的回归分析和方差分析（6课时）
  - 多元线性回归分析,逐步回归（3课时）
  - 回归诊断, 方差分析（3课时）
- R 中的多元分析（6课时）
  - 多元数据特征的计算及多元数据的图形表示（2课时）
  - 判别分析, 聚类分析（2课时）
  - 主成分分析,因子分析,典型相关分析（2课时）
- 随机数生成及蒙特卡洛模拟方法（Monte Carlo Methods）（6课时）
- Bootstrap方法（3课时）
- 非参数函数及密度估计（3课时）

参考书:

薛毅, 陈立萍(2007), **统计建模与 *R* 软件**, 清华大学出版社.

Ross, S. M. (2006). *Simulation*, 4th ed. Academic Press (王兆军、陈广雷、邹长亮译, 人民邮电出版社, 2007)

Givens, G. H. and Hoeting, J. A. (2005), *Computing Statistics* (王兆军、刘民千、邹长亮、杨建峰译, 人民邮电出版社, 2009)

Gentle, J. E. (2009), *Computational Statistics*, Springer, New York.

Dalgaard, P. (2009), *Introductory Statistics with R*, Springer, New York.

第2章　R：语言简介

1. 基本运算

   $2+2$

   exp(−2)

   rnorm(15)

2. 赋值

   x<–2; x

   x+x

3. 向量运算

   weight<–c(60, 72, 57, 90, 95, 72)

   等价于 c(60, 72, 57, 90, 95, 72)–>weight; weight

   height<–c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)

   bmi<–weight/height^2; bmi

   y<–c(weight, 1, height)

   对于向量可作+, -, *, / 和乘方 ⋀ 运算,其含义是对应向量每一个元素进行运算. 注意分号后 bmi 是为了显示计算内容,因为 R 完成计算后赋值,并不显示相应的计算内容.

   %/% 表示整数除法, %% 表示求余数

   其他初等函数 log, exp, cos, tan, sqrt 亦适用于向量.

4. 典型统计量的计算

   min(x), max(x), range(x), which.min(), which.max()

   sum(x): 求和函数; prod(x)=$\prod_{i=1}^{n} x_i$: 求积函数; length(x): 求长度, n

   sum(x)/length(x) 等价于 mean(x), 即均值; median (x), 中位数;

   var(x), 方差,即

   var(x)=sum((x-mean(x))^2)/(length(x)-1); sd(x) 标准差

   sort(x), 次序统计量. 相应的下标由 order(x) 或 sort.list(x) 列出

5. 产生等差数列

   a:b 表示从 a 开始, 逐项加一或减一, 直到 b 为止. 所得数据格式同 a , 即

   $2.3:6$ 得 $2.3, 3.3, \ldots, 5.3$

注意 ":" 的运算优先级高于加减乘除, 即 $1:n-1$ 不同于$1:(n-1)$

seq(a, b, 2), 2 表示间隔, seq(a, b) 等价于 a:b, 即默认间隔为 1.

rep() 是重复函数,

s<–rep(x, times=3), times 可省略,即 s<–rep(x, 3);

rep(x, 1:3) or rep(x, c(1, 5, 10))

6. 字符向量

y<–c("er", "sdf", "dim"), 用单引号亦可,但输出总是双引号

y<–c(a="er", b="sdf", c="dim") 在定义向量时可以给元素加上名字; 元素名字也可后加, 即使用 names() 命令. y<–c("er", "sdf", "dim"); names(y)<-c("a", "b", "c"); y

可以用 paste 函数把它的自变量连成一个字符串,中间默认用空格分开,如

paste("Annals", "of", "Statistics"); paste("Annals", "of", "Statistics", sep="-")

7. 逻辑向量

x<–1:5

x>3

FALSE FALSE FALSE TRUE TRUE

逻辑运算符：<, <=, >, >=, ==, != , c1&c2, c1|c2, !c1

z<–c(TRUE, FALSE, F, T) 这里注意均是大写

判断一个逻辑向量是否都为真值的函数是 all, 如

all(c(1:6))>3;

判断是否其中有真值用 any.

8. 几种特殊符号表示

x<–c(0/1, 1/0, 0/0, NA); x. 该向量的后三元素分别为 Inf, NaN, NA: 无穷,不确定,缺失

我们可分别使用函数 is.infinite, is.nan, 和 is.na 来检测数据是否为该三类类型. 例如

is.infinite (x): F, T, F, F.

9. 向量下标运算

我们可用 x[i] 来访问向量中的第 i 个元素, 其中 x 是一个向量名或者一个取向量值的表达式. 注意 R 中初始下标默认是从 1 开始, 这同绝大多数其它统计软件是类似的, 不同于 C 语言.

x<–c(2, 5, 8); x[2]; (c(2, 5, 8))[2]; x[2]<–9

x[c(1, 3)]<–c(10, 11); x[1:3]<–1:3; x[c(1:5)]: 1, 2, 3, NA, NA

x[order(x)], 即 sort(x)

若 v 为与 x 等长的逻辑向量,x[v] 表示取出所有 v 为真的元素.

x<–c(1, 4, 7); x[x<5]

也可做负整数运算,即不要向量中某些特定的元素,这在编程中有的时候使用起来也较为方便.

x[-c(1, 3)]: 4. 但注意不能同时出现正负的 index, R 不能识别.

如果定义向量时给元素加上了名字,此时亦可用这些名字来访问相应元素, 如

y<–c(a="er", b="sdf", c="dim"); y["a"]

10. 矩阵及其运算

- 定义及初始化：可用 array(data=, dim=) 或 matrix(data=, nrow=, ncol=, byrow=) 这两种方法来定义矩阵数据.

  x<–array(1:20, c(4, 5)) 注意是按列来存放.

  x<–array(0, c(4, 5)) 常用于初始化. 等价于x<–array(, c(4, 5)); x[]=0.

  x<–matrix(1:20, 4, 5), 默认按列存放, 同上面的array. 若需要按行存放, 可

  x<–matrix(1:20, 4, 5, TRUE)

- 下标访问操作：

  a<–matrix(1:12, 3, 4)

  a[2, 2]; a[1, 2:3]; a[1,] 略写表示全选, 即 a[1, 1:4]

  亦可使用负整数的下标来去掉一个或若干个行与列. a[-1,]; a[, -(2:3)]

- 四则运算：加减和数乘满足我们一般矩阵运算的定义, 但数组相乘和相除实际上是数组对应位置的元素做运算.

- 矩阵运算

  转置：t(); 行列式: det(); 乘法 %*%; 内积：crossprod(x, y), 即 t(x)%*%y; x 的范数平方：crossprod(x); 外积: tcrossprod(x, y), 即x%*%t(y);

  diag(v): 当 v 是一个向量时, diag(v) 表示以 v 的元素为对角元素的对角阵; 当 v 是一个矩阵时, 其表示的是取 v 对角线元素的向量. 例：初始化一个单位矩阵. v<–rep(1, p); x<–diag(v); diag(diag(v)); 当然由于统计中单位矩阵过于常用, R 中直接使用 diag(p) 即可.

  solve(A, b) 或 sovle(A): 前者求解线性方程组 Ax=b, 后者求 A^{-1}.

  v<–matrix(1:9, 3, 3); v[9]<–10; solve(v, rep(1, 3))

特征值与特征向量：使用 eigen(A) 获得, 输出包括特征值 values 和特征向量 vectors. 若需要分别提取这些结果, 可使用 $ 符号, 也就是x<–eigen(A)$values 和 y<–eigen(A)$vectors.

矩阵的奇异值分解, 即$A_{p\times q} = U_{p\times r}D_{r\times r}V^T_{q\times r}$, 其中 U, V 是正交阵, D 为对角阵, 即 A 的奇异值. 使用函数 svd(A). 其中$d, $u, $v 分别对应 diag(D), U, V.

- 与矩阵操作有关的函数：

  取维数: dim(A); nrow(A); ncol(A)

  矩阵合并：rbind(), cbind() 这两个函数我们编写自己的程序时常用. cbind(x, y) 表示 (x, y), 要求他们的行数是相同的; rbind(x, y) 表示$(x, y)^T$, 要求他们的列数是相同的; 如x<–array(1:4, c(2, 2)); y<–array(5:11, c(2, 3)); cbind(x, y); rbind(x, t(y)). 注意可合并多与两个向量, 即 rbind(x, y, z, ...)

  拉直：as.vector(A)将矩阵A转化为向量, 即 vec(A).

  apply函数, 对矩阵的行或者列使用 mean等函数进行计算, 1 表示对行操作, 2 对列, 即 apply (A, 1, mean) 当然我们也可以使用 mean(A[i,]) 来实现, 只不过这时候我们需要对 i 再循环赋值才能得到所有的值.

11. 控制流

R 是一个表达式语言, 其任何一个语句可以看成是一个表达式. 表达式之间以分号分隔或用换行分隔. 表达式可以续行, 只要前一行不是完整的表达式, 则下一行为上一行的继续.

若干个表达式可以放在一起, 用{}, 组成一个表达式使用.

判断语句：if/else:

if(condition 1) statment1 else if (condition2) statment2 else statment3

if(x>=0) $x^{0.5}$ else NA; or ifelse(x>=0), $x^{0.5}$, NA)

x<—5 if(x>=0) z1<–$x^{0.5}$, z2<–$x^{0.25}$ else if (x<=-5) “x<=-5” else NA

选择语句：switch(statement, list)

x<–3; switch(x, 2, 5, 7); switch(2, 5, mean(1:10), sd(1:10))

当 list 是有名定义时, statement 等于变量名时, 返回变量名对应的值；否则为 null. 如y<–“name”; switch(y, name=“mike”, age=“25”, wife=“jane”)

循环语句：for, while, repeat

for(i in 1:n){expression}. 例如, 构造单位阵

p<–5; id<–array(0, dim=c(p, p))

for(i in 1:p){ for (j in 1:p){ if(i==j) id[i, j]=1 } }

while(condition) expression

repeat expression. 需用 break 来停止循环. 例如, 求 1000 以内的 Fibonacci 数

x<–1; x[2]<–1; i<–1

repeat{x[i+2]<–x[i]+x[i+1]; i<–i+1; if(x[i]+x[i+1]>=1000) break }

12. 列表与数据框.

- list 是一种特殊的对象集合, 它的元素也有下标区分, 但是各元素的类型可以是任意的, 比如列表里包扩实数向量, 矩阵或者字符向量等. 比如：

  lst<–list(journal.name="Annals of Statistics", no.issues=4, impact.factor=2.2, editor="Tony Cai", associate.editor=c("Peter Hall", "Bing Li", "Song Xi Chen")).

  引用: lst$journal.name, lst$no.issue, 或者 lst[[2]], lst[[5]], lst[["associate.editor"]][2]. 用元素名字在中括号中亦可, 如 lst[["name"]], 当然注意此时需要加双引号. 这里使用单中括号亦可, 但是所得的输出类型不同于元素类型, 仍为列表, 使用时要格外注意.

  list 在我们编写程序中最重要之处在于其能实现返回值为多元素的函数输出. 如前面刚介绍过的 eigen(), svd() 等. 我们在自己编程时, 经常希望函数的返回值为多种元素类型, 比如一个向量和一个矩阵, R 不同于 Fortran 等语言, 它不区分函数和子程序（subroutine）, 因此, 实现上述功能的方法就是采用 list, 或者数据框 data.frame. 我们将在稍后介绍如何定义函数及使用list.

- 我们先再来介绍一下 data.frame. 数据框是 R 的一种数据结构, 它通常是矩阵形式的数据, 但矩阵各列可以是不同类型的, 每列是一个变量, 每行一般是一个观测. 一般我们可以把数据看成是一种推广了的矩阵, 可以用矩阵的形式显示且用矩阵下标的操作方法来引用元素和子集. dataname<–data.frame(data1=c(1:5), data2=c(6:10), data3=array(1:10, dim=(5, 2)))

- 为了说明用法, 我们来看下面的简单例子.

  library(ISwR)

  数据集：intake. 这个数据已经存为 data.frame 格式. 除看成矩阵进行操作外, 数据框的引用操作也可同 list 类型. intake$pre; intake$post; intake[[1]][2]; intake[["pre"]][3:5]; d<–intake$pre

  intake$pre[intake$post>7000] [1] 8770

  intake$pre[intake$pre>7000&intake$pre<=8000] [1] 7515 7515

  可用 subset() 函数来方便地提取满足条件的子数据框

  subset(intake, pre>6000&post>5000)

  在原有数据框中添加新的变量, 最基本的方法是直接赋值

  intake$ratio<–intake$pre/intake$post

13. 读、写数据文件

- 读文本文档
  - read.table() 文本文档数据 eg1.txt. 读取格式: read.table(“eg1.txt”). 这时注意将第一行当做数据处理了, 这种情况添加 head=T, 即read.table(”eg1.txt”, head=T). 所读取的数据为一数据框, 可将其赋值给某一变量, 即

    da<–read.table(“eg1.txt”, head=T), 此时剩余操作同前面介绍对 data.frame的引用方式. 另外注意, 若数据没有第一列的数据标号, 则自动添加标号. 如果有标号, 注意标号列不能有header, 否则该列也会自动被识别为数据.

    read.table(“eg2.txt”, head=T)
  - scan() 常用于读取无 header 类型的数据, 其可直接读纯文本文件数据. scan(“eg3.txt”) 按行将数据读入. 我们可直接将读入的数据赋值给某向量 x<– scan(“eg3.txt”). 通常我们将读入的数据存为矩阵型式, 比如

    x<–matrix(scan(“eg3.txt”), ncol=3, byrow=T)

    相较于 read.table 其更加灵活, 重要的区别在于 scan() 可以指定变量的类型.

    x<–scan(“eg5.txt”, what=list(“”, 0, 0))

    我们也可直接用 scan() 从屏幕上直接读取数据, x<–scan()
- 读 SAS 数据等

  R 可以读其他统计软件格式的数据. 首先需要调入一个模块 library(foreign), 之后我们可使用read.spss, read.s, read.xport, read.dbf, read.mtp 来分别读取 SPSS, S-PLUS, SAS, FoxPro, Minitab 格式的数据, 常用命令同 scan, 即 read.spss(“filename”). 每个命令都有其一些特定的格式设置, 我们可直接通过 help(read.spss) 这样来查看帮助, 了解特定用法.
- 读 Excel 数据.

  读取 Excel 格式的数据文件(即.xls)可能在我们平时的应用中最为常见. 但是 R 不能够直接读取 .xls 文件数据. 我们需要将Excel表转化为其他格式. 其中之一是文本文件(制表符即空格分隔), 另一种是 .csv 文件(以逗号分隔). 这样, 我们可以采用 read.table 或者 scan() 来进行读取, 只需注意对于.csv文件, 我们需要加上 sep=“, ”.
- 链接嵌入的数据库

  R 软件内置了许多数据库, 一些其他的软件包中也包含了许多数据. 我们可用 data() 命令来列出基本软件包中所有可利用的数据集. 我们可直接在控制台输入数据集的名字来显示数据. 当然绝大部分数据都是 data.frame 形式的, 我们可利用前面介绍关于数据框的操作来进行运算和引用. R中包含的其他一些数据软件包有boot, cluster, ISwR, lattice, nlme, MASS, npart, survival, stats 等. 我们可用命令

data(package = .packages(all.available = TRUE)) 来列出全部的数据集. 想要使用这些软件包中的数据集, 我们只需要先使用 library(name), name 即是所对应的 package 的名称. 使用 data(pacakge=“pkname”) 即可列出该程序包中的所有数据集.

- 写数据文件

  write(x, file=“filename”, append=F), x 通常为矩阵, 注意我们通常需要使用 t(x) 来使得所得到的数据文件形式同矩阵一致, 我们需要输出的是 x 的转置, 并且指定列数, 即 write(t(x), file=“filename”, ncolumns=3, append=F)

  我们可以用 write.table 将数据框或列表写至文件, 格式如下

  write.table(x, file=“filename”, append=false, row.names=T, col.names=T)

  这里 x 可以是数据框或列表, 也可以是矩阵, 因此推荐使用 write.table 来写文件而取代 write.

14. 定义函数

R 语言中的函数定义格式如下：

function.name<–function(arg1, arg2, ...) expression

这里 arg1, arg2 是函数的参数. 在表达式中, 放在程序最后的信息是函数的返回值, 返回值可以是向量, 矩阵, 列表或者数据框.

例如,

f<–function(x, y){c1<–crossprod(x, y); c2<–tcrossprod(x, y); list(c1, c2)};

为了调用自己编写的函数, 需要将其调到内存中去, 我们可使用 source 命令. 假设上面的函数我们存为文件 exf.r, 那么我们在其他的 R 程序或者在主控台中调用该函数时只需先输入 source(“exf.r”) 即可.

另外, 当我们采用形如“name=object” 这样的形式给出被调用函数的参数的时候, 比如f(x=xx, y=yy), 则这些参数可以按照任意顺序给出. 实际上我们前面经常调用的 R的内部函数时经常如此操作, 这样我们很多时候只需要记住一些关键的参数名称即可：也就是说, 重要的我们按照顺序则可省略名字, 中间一些参数选择默认则不给出, 后面有些参数我们需要改变, 就按照“name=object” 这样的形式给出来即可.

第3章　R：描述性统计量, 绘图, 参数估计和假设检验

§3.1　单组数据的描述性统计分析

已知一组观测 $x_1, \ldots, x_n$, 它们是从所要研究的对象的全体中抽出的, 现我们就是要研究这 n 个观测值所构成的样本的数字特征, 即分析数据的集中位置, 分散程度和数据分布等.

1. 位置参数

- 均值. mean(x, trim=0, na.rm=F) trim 为 (0, 0.5)之间的一个值, 是在计算均值之前去掉 x 两端观察值的比例. 默认 na.rm=F, 当为 T 时允许数据中有缺失数据. 如果 x 是矩阵, mean() 的返回值是一个数 (矩阵中全部数值的平均), 可用 apply 函数来获得各行各列的均值. 当 x 是数据框时, 其返回值就是向量, 即按列求均值.

  ```
  x<–faithful; names(x)<–c(x1, x2)
  mean(x, trim=0.1);
  ```

- weighted.mean(x, w, na.rm=F) 计算数据的加权平均, 其中 w 为权. 例如：

  ```
  mean(x[[2]]); v1<–var(x[[2]][1:100]); v2<–var(x[[2]][-c(1:100)])
  weighted.mean(x[[2]], w=c(rep(1/v1, 100), rep(1/v2, 172))
  apply(x, 2, weighted.mean, w);
  ```

- sort(x, na.last=NA, decreasing=F, index.return=F) 次序统计量. na.last=F, 允许处理缺失数据, 放在最前, T 放在最后; decreasing, T 为降序排列; index.return=T 时返回一列表, 可通过 $x 访问排序后的向量值, $ix 访问对应的排序下标.

  ```
  library(ISwR); intake[[2]]; sort(intake[[2]], index.return=T)
  all(sort(intake[[2]],index.return=T)$ix==order(intake[[2]]))
  ```

- 中位数. median(x, na.rm=F).

  ```
  median(faithful[[2]])
  ```

- p 分位点. quantile(x, probs=, na.rm=F), 即计算样本(经验)分位点

$$q_p = \begin{cases} x_{([np]+1)}, & \text{当 } np \text{ 不是整数时,} \\ \frac{1}{2}(x_{(np+1)} + x_{(np)}), & \text{否则.} \end{cases}$$

  probs 为一向量, 即给出相应的百分位数.

  ```
  quantile(faithful[[2]], probs=c(1/4,1/2,3/4))
  ```

  注意 quantile() 函数在通过模拟来求某检验统计量的临界值时常用.

注意 weighted.mean, sort, median, quantile 均不能直接操作数据框, 只能对其中一列来运算.

2. 尺度参数及形状参数.

   - 方差, 标准差. var(x, na.rm=F); sd(x, na.rm=F)
   - 极差, 半极差, 标准误. max(x)-min(x); quantile(x, 0.75)-quantile(x, 0.25); sd(x)/sqrt(n)
   - 偏度系数. 总体：$\frac{E(X-EX)^3}{[E(X-EX)^2]^{3/2}}$. 刻画对称性, 关于均值对称偏度系数为零, 系数为正右侧数据更分散, 系数为负左侧更分散.

     样本：$\frac{n^2\mu_3}{(n-1)(n-2)s^3}$: n^2/((n-1)*(n-2))*sum((x-mean(x))^3)/sd(x)^3
   - 峰度系数. 总体: $\frac{E(X-EX)^4}{[E(X-EX)^2]^2}-3$; 峰度系数为正(负), 其形状比高斯分布更尖陡(平坦), 即高(低)峰度.

     样本：$\frac{n^2(n+1)\mu_4}{(n-1)(n-2)(n-3)s^4}-3\frac{(n-1)^2}{(n-2)(n-3)}$:

     n^2*(n+1)/((n-1)*(n-2)*(n-3))*sum((x-mean(x))^4)/sd(x)^4- 3*(n-1)^2/((n-2)*(n-3)).

     s 为样本标准差, μ_3、μ_4 分别为样本三阶、四阶中心距.

     R 的扩展统计包 fBasics 提供了函数 skewness(), kurtosis() 用来求样本的偏度和峰度.

     library(fBasics)

     skewness(faithful[[2]], method=c("moment", "fisher"))

     kurtosis(faithful[[2]], method=c("excess", "moment", "fisher"))

     参数 moment 指直接代入矩估计; fisher是上述的近似无偏的修正法.

3. 五数总括 fivenum(x, na.rm=T). 计算最小值, 0.25 分位数, 中位数, 0.75 分位数, 最大值. 其适用于向量数据, 但对数据框不适合, 因为其只能返回一列向量.

4. summary(x) 则既可对普通向量又可对数据框进行操作, 即可按数据框所分变量来分别输出主要的描述性统计量.

   x<-faithful; apply(x, 2, fivenum); summary(x)

5. 数据分布

   - 分布函数等. 我们以正态分布举例说明.

     pnorm(x, mean=0, sd=1, lower.tail=T, log.p=F) 正态分布函数. $\Phi(x)$

     qnorm(p, mean=0, sd=1, lower.tail=T, log.p=F) 求给定概率 p 时的下分位点, 即 $\Phi(x)=p$, 返回 x.

     rnorm(n, mean=0, sd=1); 生成 n 个正态随机数.

     dnorm(x, mean=0, sd=1, log=F) 正态概率密度函数. $\phi(x)$

     这里函数中的参数 x 和 p 均可为数值向量, 即返回多个所需要的值.

log, log.p 默认为 F, 为 T 时表示对应 log 值; lower.tail 默认为 T, 为 F 时对应上分位点.

pnorm(c(1, 2, 3)); qnorm(pnorm(c(1, 2, 3)))

dbinom(0:10,10,0.1); sum(dbinom(0:10,10,0.1))

其他分布函数也与此类似, 如 t, f, chisq, binom 等, 详细可参考 R 的帮助文档. 在分布名称前加上 d, p, q, r 分别代表概率密度函数、分布函数、分布函数反函数、产生随机数.

举两个简单的例子：

- 查找分布的分位数, 用于计算假设检验中分布的临界值或置信区间的置信限. 如, 显著性水平为 5% 的正态分布的双侧临界值是：qnorm(c(0.025, 0.975))
- 计算假设检验的 p 值. 如, 自由度为 1 的卡方检验统计量 3.84 其检验的 p 值(拒绝原假设的最小显著性水平)为 1-pchisq(3.84, 1); 又如容量为 14的 t-检验统计量为-2.43时, 双边 t 检验的 p 值为: 2*pt(-2.43, df=13)

- 直方图：将观测数据的取值范围分为若干个区间, 计算落在每个区间的频数或频率, 在每个区间上画一个矩形, 以估计总体的概率密度. 在 R 中用

  hist(x, breaks=, freq=T, main=, xlab=, ylab=, axes=T, plot=T, labels=F)

  breaks 规定直方图的组距, 或为向量或为一数（组的个数）, 推荐直接使用默认; freq=T/F (频数/密度); main, xlab, ylab 分别为直方图的标题, 横坐标, 纵坐标的名称; axes 指示是否包含坐标轴; plot 指示是否给出直方图（若 F 则列出绘图的结果）; labels 指示是否在每一个矩形上方给出频数或密度值; col: 直方图的颜色.

  以 Old Faithful Geyser Data 为例,

  hist(x$x2, breaks=5, freq=T, main="Histogram of waiting", xlab="waiting", col="blue", axes=T, labels=T)

- 核密度估计函数. 用给定样本估计密度函数.

  density(x[[2]]) 其他参数我们将在讲述非参数和密度估计时再统一进行介绍, 这里先使用默认参数. 我们可使用 plot(density(x[[2]])) 绘出密度曲线图. 有时我们希望将估计的密度曲线图, 直方图, 和某个给定分布的密度曲线图画在一张图上以直观的比较, 这时可使用 lines 命令. 例如,

  hist(x$x2, freq=F, col="yellow", axes=T, labels=T)

  lines(density(x$x2), col="blue", lwd=3)

  y<-(min(x[[2]])-5):(max(x[[2]])+5);

  lines(y, dnorm(y, mean(x[[2]]), sd(x[[2]])), col="red", lwd=3)

- 经验分布和拟合优度检验
  - R 语言中样本经验分布函数计算 $F_n(x) = n^{-1}\sum_{j=1}^{n} I_{\{X_j \le x\}}$.

plot(ecdf(x$x2), main="ECDF of waiting", verticals=F, do.p=F)

其中逻辑变量 verticals 指示是否画竖线, 逻辑变量 do.p 指示是否在点处画记号.

可将正态分布曲线和其画在一起, y<–40:100

lines(y, pnorm(y, mean(faithful[[2]]), sd(faithful[[2]])), col="blue").

– $F_n(x)$ 是总体分布函数 $F(x)$ 的估计. 经验分布拟合检验的分布是检验经验分布 $F_n(x)$ 与假设的总体分布函数 $F_0(x)$ 之间的差异. Kolmogorov-Smirnov 检验是个古老的但最常用的方法,

$$D = \sup_{-\infty<x<\infty} |F_n(x) - F_0(x)|.$$

R 中可用 ks.test() 函数来进行该检验. 仍以 faithful数据的 waiting 为例, 检验其是否为正态.

ks.test(faithful[[2]], "pnorm")

ks.test(faithful[[2]], "pnorm", mean(faithful[[2]]), sd(faithful[[2]]))

注意区分这两种情况, 前者检验是否为标准正态, 后者考虑检验是否为均值和方差相同的正态分布, 所以后者更科学, 其也等价于

ks.test((faithful[[2]]-mean(faithful[[2]]))/sd(faithful[[2]]), "pnorm")

– 另一种常用的拟合优度检验就是我们熟知的 pearson χ^2 检验, 在 R 中我们可通过函数 chisq.test() 来完成该检验.

- 正态性检验和 Q-Q 图

Shapiro-Wilk 检验是最常用且最有效的正态性检验之一, 其检验统计量形如

$$W = \left(\sum_{i=1}^{n} a_i X_{(i)}\right)^2 / \sum_{i=1}^{n} (x_i - \bar{x})^2 ,$$

其中 $X_{(i)}$ 是第 i 个次序统计量, a_i 和临界值的确定可参见

Royston, P. (1982) Algorithm AS 181: The W test for Normality. *Applied Statistics*, 31, 176–180.

在 R 中, 我们使用函数 shapiro.test() 来进行该检验. 仍以 faithful 数据为例

shapiro.test(faithful[[2]]). 结果:

W=0.9221, p-value = 1.016e-10. 我们在 0.05 的显著性水平下显然应该拒绝原价设, 这同前面使用 Kolmogorov-Smirnov 检验所得结果相同.

另外一种直观地判断鉴别样本分布是否近似为正态（或其它类型）分布就是我们所熟知的 QQ 图.

qqnorm(faithful[[2]], main="", xlab="", ylab="")

qqline(faithful[[2]])

- 盒子图

  盒子图能够直观简洁地展现数据分布的主要特征. 我们在 R 中使用 boxplot() 函数作盒子图. 在盒子图中, 上下四分位数分别确定中间箱体的顶部和底部, 箱体中间的粗线是中位数所在的位置. 由箱体向上下伸出的垂直部分为"触须" (whiskers), 表示数据的散布范围, 其为1.5倍四分位间距内距四分位点最远的数据点. 超出此范围的点可看作为异常点(outlier).

§3.2 多组和分类数据的描述性统计分析

在对于多组数据的描述性统计量的计算和图形表示方面, 前面所介绍的部分方法不能够有效地使用, 例如许多函数都不能直接对数据框进行操作. 这时我们需要一些其他的函数配合使用.

1. 图形表示：

- 散点图：前面介绍的 plot, 可直接对数据框操作. 此时将绘出 数据框中 所对应的所有变量两两之间的散点图. 所做图框中第一行的散点图是以第一个变量为纵坐标, 分别以第二、三...个变量为横坐标的散点图. 这里数据举例说明.

  library(DAAG); plot(hills)

- 盒子图：前面介绍的 boxplot, 亦可直接对数据框操作, 其在同一个作图区域内画出各组数的盒子图. 但是注意, 此时由于不同组数据的尺度可能差别很大, 这样的盒子图很多时候表达出来不是很有意义. boxplot(faithful). 因此这样做比较适合多组数据具有同样意义或近似尺度的情形. 例如, 我们想做某一数值变量在某个因子变量的不同水平下的盒子图. 我们可采用类似如下的命令：

  boxplot(skullw∼age, data=possum), 亦可加上参数 horizontal=T, 将该盒子图横向放置.

  boxplot(possum$skullw∼possum$sex, horizontal=T)

- 条件散点图：当数据集中含有一个或多个因子变量时, 我们可使用条件散点图函数 coplot() 作出因子变量不同水平下的多个散点图, 当然该方法也适用于各种给定条件或限制情形下的作图. 其调用格式为

  coplot(formula, data)

  比如 coplot(possum[[9]]∼possum[[7]]|possum[[4]]), 或

  coplot(skullw∼taill|age, data=possum);

  coplot(skullw∼taill|age+sex, data=possum)

- 直方图：一个方法就是使用 mfrow 将绘图框分隔多个部分, 然后直接反复使用 hist() 对各个变量做直方图.

  par(mfrow=c(1, 2)); hist(faithful[[1]]); hist(faithful[[2]])

  另一种直接方法是使用 lattice 包中的直方图函数, histogram(). lattice 包是一个强大的绘图软件包, 我们以后还会对其中的常用函数加以介绍.

  x<–possum; histogram(∼x[[7]]|x[[4]])

- 密度曲线图：这里我们欲绘出某一数值变量在某个因子变量的不同水平下的密度曲线图, 可采用 lattice 包中的 densityplot, 举例如下：

  densityplot(∼skullw|age, data=possum); densityplot(∼skullw|sex, data=possum)

2. 描述性统计分析：

- 前面介绍的 summary() 函数可直接对数据框操作. 如library(DAAG); summary(cuckoos)
- tapply() 能够方便地对分组数据进行函数操作.

  a1<–tapply(cuckoos$length, cuckoos$species, mean);

  a2<–tapply(cuckoos$length, cuckoos$species, sd);

  a3<–tapply(cuckoos$length, cuckoos$species, median);

  cbind(mean=a1, std=a2, median=a3)

  a1<–tapply(cuckoos$length, cuckoos$species, mean);

  a2<–tapply(cuckoos$breadth, cuckoos$species, mean);

  a3<–tapply(cuckoos$id, cuckoos$species, mean);

  cbind(length.mean=a1, breadth.mean=a2, id.mean=a3)

  其它函数, 如 fivenum 等, 亦可, 不过注意此时输出地是一个 list.

  另外注意对有缺失的数据, 使用 na.rm=T.

- 为了计算不同因子变量所对应的各数值变量的描述性统计量, 我们还可方便地使用分组概括函数 aggregate. 该函数作用同 tapply 类似, 不同的在于其对数据框直接操作, 返回值也是数据框. 举例来说,

  aggregate(cuckoos, list(species=cuckoos$species), mean)

  注意这里第二个参数必须是一个列表形式的, 因此我们需要使用 list 函数将其转化. 我们当然也可以才用下面的调用形式来避免使用 list.

  aggregate(cuckoos, cuckoos[“species”], mean)

  我们前面讲过使用 subset 提取子数据框, 是指观测来提取, 即子数据框是在原数据框基础上删去部分观测; 而我们亦可提取一定的变量列来构造新的数据框, 比如

  cuckoos[c(“length”, “breadth”, “id”)]; 此时我们再使用 aggregate 函数

  aggregate(cuckoos[c(“length”, “breadth”, “id”)], list(species=cuckoos$species), mean)

- by() 函数同 aggregate 类似, 只不过对于 by, 它将数据框中的每列逐一处理, 这时那些不能对数据框直接操作的函数如 fivenum 或者 median 等函数亦可使用了.

  by(cuckoos[c("length", "breadth", "id")], list(species=cuckoos$species), fivenum)

  另外, 使用 tapply, aggregate, 及 by等函数中的用于计算的函数中要填写其他参数的, 直接在函数名的后面填写, 比如求分位数

  by(cuckoos[c("length", "breadth", "id")], list(species=cuckoos$species), quantile, probs=0.75)

§3.3 绘图

R 提供了非常多样的绘图功能. 我们可以通过 R 提供的两组演示例子进行了解：

demo(graphics): 二维; demo(persp): 三维.

在 R 的作图函数中, 一类是高水平作图, 另一类是低水平作图, 前者中的函数均可产生图形, 可有坐标轴, 以及图和坐标轴的说明文字等; 后者自身无法生成图形, 只能在前者生成的图形基础上增加新的图形.

高水平作图函数: 这其中包括我们前面在描述性统计分析中介绍过的各种具有特殊功能的绘图函数, 如：hist(), boxplot(), qqnorm()等. 我们下面对几种一般的高级绘图函数给予更详细的说明.

1. plot() 函数可绘出各种散点图和曲线图.

   (a) plot(x, y): 生成 y 关于 x 的散点图.

   (b) plot(x): 生成 x 关于下标的散点图.

   x <–faithful; names(x)=c("x1", "x2"); plot(x$x1, type="o"); plot(x$x1, x$x2);

   (c) plot(f): 其中 f 是因子变量. 生成 f 的直方图.

   (d) plot(f, y): 生成 y 关于 f 各水平的 boxplot, 也就是将不同水平所对应的 y 各自做盒子图.

   library(DAAG); plot(possum[[4]]); plot(possum[[4]], possum[[7]])

   (e) plot(df), 其中 df 是数据框.

2. contour(x, y, z) 绘出三维图形的等高曲线图; persp(x, y, z) 绘出三维图形的表面曲线.

   library(MASS); z<–kde2d(x[[1]], x[[2]]); contour(z). 这里我们使用了 MASS 程序包中的二维核密度估计函数 kde2d() 来估计二维数据的联合密度函数, 再利用该函数画出密度的等高曲线图.

   persp(z). 做出该估计的密度函数的三维图形的曲线图.

其中 persp(x, y, z) 常用于刻画二维密度曲线, 其中两个参数 theta=, phi=, 用于改变图形的观察角度, 使用中经常根据需要自己变动.

persp(z, theta=45, phi=30, xlab="x", expand=0.7)

3. 高水平绘图中的辅助命令

- add=T (默认 F) 表示所绘图在原图上加图.
- axes=F (默认 T) 表示所绘图形没有坐标轴. 我们可以用 xaxt="n" 或 yaxt="n" 来选择是否画横纵坐标. hist(cuckoos$length, axes=F); hist(cuckoos$length, axes=T, xaxt="n")
- main="" 图的主标题说明, sub=""图的副标题, xlab, ylab 分别是 x 轴, y 轴的说明.

  plot(cuckoos$length, cuckoos$breadth, main="length vs breadth", sub="", xlab="length", ylab="breadth")
- xlim, ylim 用于指定轴的上下限. 如:

  plot(cuckoos$length, cuckoos$breadth, ylim=c(14, 19))
- log="x", log="y", log="xy" 表示对 x, y 轴的数据取对数.
- type="" 表示绘图类型. 常用的有p (散点图), l (实线), b (所有点被实线链接), o (实线通过所有的点)

  plot(cuckoos$length, type="l")

绘图参数

除了低级作图命令之外, 图形的显示也可以用绘图参数来改良. 绘图参数可以作为图形函数的选项, 比如在 plot()函数中可以指定颜色等, 但不是所有参数都可以在绘图函数中来指定. 我们可通过使用函数 par() 来永久地改变绘图参数, 也就是说后来的图形都将按照函数 par() 指定的参数来绘制.

一些常用的参数设置如下, 其它相关参数可参见参见其帮助.

cex: 控制文字大小的值(cex.main; cex.sub; cex.lab; cex.axis); col: 符号的颜色; lty: 线的类型; mar：控制图形边空的4个值 c(bottom, left, top, right). mfrow=c(m, n): 将绘图窗口分割为 m 行和 n 列, 也就是可在一个窗口内画多个图形; pch: 符号的类型; lwd：控制连线宽度.

另外注意, 通常我们在修改 par 的参数之前, 先将默认值赋值给某变量, 如 op<–par(), 在某一个图做完之后, 可使用 par(op), 还原到默认状态.

低水平作图函数

1. points(x, y), lines(x, y) 分别是加点和加线函数, 即在已有的图上加点或者加线. x, y分别是横纵坐标数值或向量, 函数表示在对应的坐标（或坐标向量）处加点、加线.

   plot(x$x1, x$x2); lines(lowess(x[[1]], x[[2]]), lwd=3)

这里我们在两者的散点图上添加了一非线性拟合曲线, 调用了函数 lowess().

2. text() 该函数的作用是在图上加标记. 一般用法是 text(x, y, labels=), 即表示在对应的坐标（或坐标向量）(x, y)处添加标记, 其中 labels 默认为 label=1:length(x), 我们实际中经常用到的是labels=“”, 也就是在某处添加某个说明性的字符向量.

   text(3.0, 6.0, “nonlinear fit”)

3. abline() 函数可在图上加直线, 范围是整个绘图框. 常用方法如下：(i) abline(a, b) 绘出 $y = a + bx$ 的直线; (ii) abline(h=y), 表示画出过 y 点的水平直线; (iii) abline(v=x) 表示画出过 x 点的垂直直线.

   abline(30, 10, col=“red”); abline(h=75); abline(v=3.0)

4. polygon(x, y), 以数据 (x, y) 为坐标, 依次连接所有的点, 绘出一多边形.

5. 在图上加说明文字、标记或其他内容.

   - title(main=“Main Title”, sub=“sub title”) 其中主题目加在图的顶部, 子题目加在图的底部.
   - axis(side, ...) 是在坐标轴上加标记、说明或其它内容, 其中 side=1, 2, 3, 4 分别表示所加内容放在图的底部、左侧、顶部、右侧.

     比如, axis(1, seq(1.5, 6.5, 1.0), pos=40)
   - legend(x, y, legend) 在点 (x, y) 处添加图例, 说明的内容由 legend 给定.

     legend(3.5, 55, legend=c(“scatter plot”, “nonlinear fit”), lty=c(0, 1), pch=c(21, NA), lwd=c(1, 2))

     其中 lty=0, 1分别对应“无线”和实线, pch 分别对应空心点和无点.
   - rug(x), 在 x 轴上用短线画出 x 数据的位置. rug(x$x1)

6. 在使用 text(), legend() 等函数中, 我们不仅可以使用字符串类型的说明文字, 亦可通过使用函数 expression() 来加入各种数学公式或数学表达式. 在 expression() 函数中的表达式与 Latex 中的命令非常地类似, 很多都同 Latex是一致的, 具体用法可通过 help(symbol) 来进行查询. 举例如下：

   qqnorm(faithful[[2]]); qqline(faithful[[2]])

   比如在该图上添加对 qqline的注释,

   text(0.5, 60, expression(italic(y)==sigma*italic(x)+mu))

   这里我们若想在表达式中带入某个变量的值, 如 sigma, mu, 我们可以使用函数 substitute() 和 as.expression().

   text(1, 60, as.expression(substitute(italic(y)==sigma*italic(x)+mu,

list(sigma=sd(faithful[[2]]), mu=mean(faithful[[2]])))))

如果我们想只显示3位小数, 使用 round() 函数将上述命令中的 list 命令里修改为：

list(sigma=round(sd(faithful[[2]]),3), mu=round(mean(faithful[[2]]),3))

7. 另外当使用原数据不能够得到有意义的图形时, 可以对数值进行变换以得到有意义的图形, 例如常用的对数、倒数、指数以及著名的 Box-Cox 变换. 这里以常用的指数变换举例.

library(MASS); par(mfrow=c(1, 2));

plot(brain∼body, data=Animals); plot(log(brain)∼log(body), data=Animals)

由于尺度的影响, 左侧的散点图几乎没有体现出任何的信息, 而做了 log 变换后的右侧散点图则呈现出明显的线性关系.

§3.4 R 中的参数估计和假设检验

1. **参数估计**

- **矩估计**：假设总体 X 的分布函数含有 k 个未知参数 $\theta_1,\ldots,\theta_k$, 且分布的前 k 阶矩存在且都是 $\theta_1,\ldots,\theta_k$ 的函数. 求 θ_j 的矩估计的具体步骤如下：

  (i) 求出 $E(X^j)=\mu_j, j=1,\ldots,k$, 并假定

  $$\mu_j = g_j(\theta_1,\ldots,\theta_k), j=1,\ldots,k.$$

  (ii) 解上面的方程组得到

  $$\theta_j = h_j(\mu_1,\ldots,\mu_k), j=1,\ldots,k.$$

  (iii) 用样本的 k 阶原点矩 $m_k=\frac{1}{n}\sum_{i=1}^n X_i$ 来代替 μ_j 则得到 θ_j 的矩估计 $\widehat{\theta}_j = h_j(m_1,\ldots,,m_k)$.

  如果 h_j 可以显示表达, 则我们只需要适当的 R编程即可直接获得矩估计. 当然, 当方程组是非线性的时候, 我们根本无法得到显示的 h_j. 这时我们直接将 m_j 代入 (i), 则直接得到 k元的方程组. 解这样的方程求根问题一般采用迭代算法数值求解, 我们将会在稍后的章节中介绍一般的方法. R 中有不少解这种非线性方程组的 package, 比如 nleqlsv.

- **极大似然估计**: 设 $X_1,\ldots,X_n$ 为取自总体 X 的样本且其联合密度函数为 $L(\boldsymbol{\theta}) = \prod_{i=1}^n f(X_i|\boldsymbol{\theta})$. 极大似然估计即是求使得 $L(\boldsymbol{\theta})$ 或对数似然函数 $l(\boldsymbol{\theta})=\log(L(\boldsymbol{\theta}))$ 达到最大的参数 $\boldsymbol{\theta}$ 的值. 对于没有显示解的情形, 我们通常要使用优化方法来求极大似然估计值. 在 R 中, 对于单参数场合, 我们可以使用函数 optimize() 求极大似然估计值.

optimize(f=, interval=, maximum=F, tol=)

其中, f 是似然函数, interval 是参数 θ 的取值范围, maximum=F 是求最小值, tol 则是求值的精度. 对于多参数场合, 我们可以使用函数 nlm() 或者 optim() 求极大似然估计值. 这里仅给出 nlm 的法, 关于 opti()可参见帮助, 我们在下一章介绍了各种优化方法后会再介绍 opti. nlm 的基本用法是

nlm(f, p, steptol=1e-6, iterlim=100)

其中 f 是似然函数, p 为给定的初始值, steptol 为求值的精度, 而 iterlim 是我们所容许的最大的迭代次数. 该函数采用 Newton-Raphson 算法, 函数的返回值是一个列表, 包含极小值、极小点的估计值等.

- **区间估计**. 矩估计和极大似然估计都是点估计, 是参数的一个近似值, 因而它们都不能给出估计的误差范围, 亦没能指出这个误差范围以多大的概率包括未知参数. 而区间估计则能够给出的估计的可信程度. 最常用的区间估计主要是基于正态总体(或是某个参数分布)假设下的, 我们在这里介绍的内容也基于这一假设, 而基于非参数的区间估计的推断我们将在后面的章节中介绍.

(1) 正态总体的单样本均值和两样本均值差的区间估计.

R 中提供了函数 t.test 来处理方差未知情形下该类区间估计的问题. 而对于方差已知的情形, 其仅限于书本上的讨论, 在实际问题中并不常用, 因为我们通常在仅给定样本情况下假设方差或均值已知都是不合理的.

单样本的均值 μ 的置信区间是根据 $\frac{\bar{X}_n-\mu}{S_n/\sqrt{n}} \sim t(n-1)$ 得到的, 即 μ 的置信水平为 $1-\alpha$ 的置信区间是

$$\left(\bar{X}_n - \frac{S}{\sqrt{n}}t_{1-\frac{\alpha}{2}}(n-1), \bar{X}_n + \frac{S}{\sqrt{n}}t_{1-\frac{\alpha}{2}}(n-1)\right),$$

其中 $t_p(n)$ 为自由度为 n 的 t 分布的下侧 p 分位数.

对于两样本问题的均值差 $\mu_1-\mu_2$ 当两方差均未知时仅当 $\sigma_1^2=\sigma_2^2=\sigma^2$ 时(或 $\sigma_1^2=\theta\sigma_2^2, \theta$ 已知) 我们才可得到精确的置信区间. 其是根据 $\frac{\bar{X}_n-\bar{Y}_n-(\mu_1-\mu_2)}{S_w\sqrt{\frac{n_1+n_2}{n_1n_2}}} \sim t(n_1+n_2-2)$ 得到的, 其中 $S_w^2=\frac{(n_1-1)S_1^2+(n_2-1)S_2^2}{n_1+n_2-2}$, 即 $\mu_1-\mu_2$ 的置信水平为 $1-\alpha$ 的置信区间是

$$\bar{X}_n - \bar{Y}_n \pm S_w\sqrt{\frac{n_1+n_2}{n_1n_2}}t_{1-\frac{\alpha}{2}}(n_1+n_2-2)$$

当方差不等时, 我们通常采用 Welch 近似的方法, 所得到的置信区间形如.

$$\bar{X}_n - \bar{Y}_n \pm t_{1-\frac{\alpha}{2}}(v)S^*,$$

其中 v是近似的自由度, $S^*=\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}$.

t.test一般用法如下: t.test(x, y=null, alternative=c("two.sided", "less", "greater"), mu=0, paired=F, var.equal=F, conf.level=0.95)

由于置信区间和显著性水平假设检验通常来说是统一的, t.test 即可同时完成二者. 上面的命令中: 若仅有数据 x, 则进行单样本 t 检验; 若给出数据 y, 则进行两样本的 t 检验; alternative 用于指定所求置信区间的类型, 即置信区间, 置信下限, 置信上限; mu 仅在假设检验中起作用.

举例如下：我们考虑 possum 数据. 一般我们在使用基于正态假设构造置信区间前需要做正态性检验, 如果我们接受正态的假设, 则我们可认为构造出来的置信区间的精确程度应该很高; 而如果拒绝正态假设, 由统计大样本理论知道我们仍可使用同样的方法来构造, 但此时我们自己应该牢记可能构造出来的置信区间的覆盖率与我们实际的要求相差一定距离, 当然如果 n 足够大时, 这样的置信区间的覆盖率会接近我们想要的值.

先来看一个单样本的例子：

library(DAAG); shapiro.test(possum$hdlngth);

qqnorm(possum$hdlngth); qqline(possum$hdlngth)

t.test(x1); ci<–t.test(x1)$conf.int (提取 t.test 的置信区间的输出结果);

attributes(ci)<–NULL (表示去掉 ci 的属性使之成为一标准的二维向量);

ci<–t.test(x1, alternative=“less”, conf.level=0.99)$conf.int

假如我们想要得到置信水平为 0.01, 0.05, 0.10 的区间, 可以

alpha<–c(0.01, 0.05, 0.10); ci<–array(0,c(3,2)); for (i in 1:3){ x2<–t.test(x1, conf.level=1-alpha[i])$conf.int; attributes(x2)<–NULL; ci[i,]<–x2}; ci

再来看一个两样本的例子：

x1<-possum$hdlngth[possum$sex==“f”]; n1<–length(x1); var(x1)

x2<-possum$hdlngth[possum$sex==“m”]; n2<–length(x2); var(x2)

shapiro.test(x1); shapiro.test(x2);

t.test(x1,x2)$conf.int; t.test(x1,x2,var.equal=T)$conf.int

(2) 正态总体的单样本方差及两样本方差比的区间估计.

单样本的方差 σ^2 的置信区间是根据 $\frac{(n-1)S_n^2}{\sigma^2} \sim \chi^2(n-1)$ 得到的, 即 σ 的置信水平为 $1-\alpha$ 的置信区间是

$$\left(\frac{(n-1)S_n^2}{\chi^2_{1-\frac{\alpha}{2}}(n-1)}, \frac{(n-1)S_n^2}{\chi^2_{\frac{\alpha}{2}}(n-1)}\right),$$

R 中没有提供求单样本 σ^2 置信区间的函数, 需要我们自己编写, 留作作业.

对于两样本问题的方差比 σ_1^2/σ_2^2 的置信区间, 其是根据 $\frac{S_1^2/\sigma_1^2}{S_2^2/\sigma_2^2} \sim F(n_1-1, n-2-1)$ 得到的, 即 σ_1^2/σ_2^2 的置信水平为 $1-\alpha$ 的置信区间是

$$\left(\frac{S_1^2/S_2^2}{F_{\alpha/2}(n_1-1,n_2-1)}, \frac{S_1^2/S_2^2}{F_{1-\alpha/2}(n_1-1,n_2-1)}\right).$$

R 软件中提供了该两样本方差比区间估计的函数 var.test(), 其一般调用格式如

下：

var.test(x, y, ratio=1, alternative=c("two.sided", "less", "greater"), conf.level=0.95)

其中 ratio 只是在作假设检验的时候才有用. 仍以 possum 数据为例:

var.test(x1, x2)$conf.int

2. **假设检验**

在 R 中假设检验的作法与上述的区间估计基本是一致的, 使用同样的命令即可获得检验的结果. 这里不再赘述. 仅举例如下：

t.test(possum$hdlngth, mu=93.0);

t.test(possum$hdlngth, alternative="less", mu=93.0); 这里 less 表示单边检验 H_0 : $\mu \geq \mu_0$ vs $H_1 : \mu < \mu_0$.

t.test(x1, x2); t.test(t.test(x1, x2), var.equal=T);

t.test(x1, x2, alternative="less"); 这里 less 为单边检验 $H_0 : \mu_1 \geq \mu_2$ vs $H_1 : \mu_1 < \mu_2$.

另外, 我们有时要做成对数据的 t 检验, 即指两个样本的样本容量相同, 两组数据分别是同一观测个体在两种不同状况下得到的观测. 这时, 我们仅需要在 t.test() 函数中添加 paired=T, 即可, 举例来说：

shapiro.test(intake$pre); shapiro.test(intake$pre);

t.test(intake$pre, intake$post, paired=T)

检验两方差比例的检验 var.test(x1, x2, ratio=1)

另外, R 中还提供了关于二项分布比率检验的函数, binom.test(). 也就是假设 $X_1, \ldots, X_n$ 来自两点分布 BIN$(p, 1)$, 而 $T = \sum_{i=1}^{n} X_i \sim$ BIN(n, p). 检验 $H_0 : p = p_0$ vs $H_1 : p \neq p_0$. 具体用法参见帮助, 这里不详细介绍了.

第4章　非线性方程数值解及优化方法

统计学中的一个重要问题是MLE估计问题, 解决这样问题的关键是寻找似然方程的最优解. 在很多情况下, 我们不能直接得到似然方程的显式解, 需要通过数值分析的方法得到方程的解. 除极大似然外, 统计学中也有许多其它的优化问题, 如在 Bayes 决策问题中的最小风险、非线性最小二乘问题的求解问题等.

上述求解都属于如下的一般问题：

$$\arg\min_{\boldsymbol{\theta}\in D} g(\boldsymbol{\theta}), \tag{4.1}$$

其中 g 是参数向量 $\boldsymbol{\theta}$的函数,称之为**目标函数** (*objective function*), 而 $\boldsymbol{\theta}$ 我们有时亦称之为**决策变量** (*decision variable*). 决策变量的取值区域 D 称为可行集合 (feasible set) 或者 候选集合 (candidate set). 由于最大化一个函数等价于其负值的最小化 ($-g(\boldsymbol{\theta})$), 故区别最大与最小的意义不大. 于是作为惯例, 我们一般将考虑求取最小值的算法.

这里我们需要区分有约束和无约束两种优化问题. 当 D 就是 $g(\boldsymbol{\theta})$ 的定义域时, 问题4.1就是一**无约束优化问题** (*unconstrained optimization problem*); 否则, 即是**有约束优化问题** (*constrained optimization problem*). 另外, 在取值区域内 D 的某个特定子域内, 可能有某个局部最小值, 而在另外一个特定子域内存在另一个局部最小值. 我们之后称**全局最优** (*global optimum*) 即指在取值区域内 D 的最小值; 称**局部最优** (*local optimum*) 即指在取值区域内 D 的某个子域内的最小值.

在本章, 我们将主要考虑 g 关于 θ 为光滑且可微的情形, 而 g 在离散区域上的优化问题将在最后给予介绍.

§4.1　单变量方程求根问题

我们知道, 很多优化问题都等价于是一个方程求根问题. 因此, 我们首先来讨论一元方程求根问题的数值解法, 即对于给定的关于 x 的函数 g 寻找 x 使得 $g(x)=0$.

问题：求解函数 $\log x/(1+x)$ 的最大值? 其等价于求方程

$$g(x)=\frac{1+1/x-\log x}{(1+x)^2}=0 \Leftrightarrow 1+1/x-\log x=0$$

的解. 显然没有显式解。

我们介绍一些常用的迭代算法. 我们接下来假设 g 是一连续函数.

§4.1.1　二分法 (bisection method)

一、原理：如果 g 在区间 $[a_0,b_0]$ 上连续, 且 $g(a_0)g(b_0)\leqslant 0$, 则由中值定理知, 至少存在一个 $x^*\in[a_0,b_0]$,使得 $g(x^*)=0$. 我们称 $[a_0,b_0]$ 为有根区间.

取区间 $[a_0, b_0]$ 中点 $x_0 = (a_0 + b_0)/2$ 将它分为两半, 若 $g(x_0) = 0$, 则 x_0 就是 $g(x) = 0$ 的一个根; 否则由 x_0 分成的两个区间中必有一个是有根区间, 记为 $[a_1, b_1]$. 此时即把区间 $[a_0, b_0]$ 缩短至 $[a_1, b_1]$, 其长度为原区间的一半. 如此反复进行, 可得到一系列有根区间 $[a_0, b_0] \supset [a_1, b_2] \supset [a_2, b_2] \supset \cdots$, 其中每个区间都是前一个区间的一半, 因此 $[a_k, b_k]$ 的长度 $b_k - a_k = (b_0 - a_0)/2^k$. 当 $k \to \infty$ 时其趋于零, 就是说, 如果二分过程无限地继续下去, 这些区间将最终收缩于一点, 该点显然为所求的根 x^*. 当然在实际计算时, 我们无法完成这一无限过程, 且也无这种必要, 因为我们数值计算的结果是允许误差的. 将以上描述总结为如下算法:

算法 4.1.1. 二分法求方程根:

(0) 令 $k = 0$, 且找到一有根区间 $[a_0, b_0]$.

(1) 令 $k = k + 1$ 且 $x_k = (a_{k-1} + b_{k-1})/2$.

(2) 如果 $\operatorname{sgn}(g(x_k)) = \operatorname{sgn}(g(a_{k-1}))$ 则令 $a_k = x_k$; 否则 $b_k = x_k$.

(3) 如果达到某一收敛准则, 则停止并返回解 x_k. 否则返回第一步.

关于收敛准则: 绝对收敛和相对收敛. 绝对收敛的停止准测:

$$|x_{t+1} - x_t| < \epsilon, \tag{4.2}$$

其中常数 ϵ 是选定的可容忍精度. 对于上述二分法, 不难看出 $|x_{t+1} - x_t| = (b_t - a_t)/2 = (b_0 - a_0)/2^{t+1}$. 只要 t 足够大,则 (4.2) 就能成立.

相对收敛准测:

$$\frac{|x_{t+1} - x_t|}{|x_t|} < \epsilon \tag{4.3}$$

时停止迭代. 此准则可以在不必考虑 x 的单位的情况下达到指定的目标精度,如1%.

我们应该根据实际问题来选择应用绝对还是相对收敛准则. 如果 x 的刻度相对于 ϵ 很大（或很小）时,绝对收敛准则有时将不如我们所愿地停止迭代或迭代很快就停止了. 相对收敛准则对 x 的刻度做了校正,但当 $x^{(t)}$ 的值（或真值）与 0 非常接近时,它将变得不稳定,此时,我们可以通过 $\frac{|x_{t+1} - x_t|}{|x_t| + \eta} < \epsilon$ 来修正相对收敛准则.

在今后使用各种迭代算法的过程中, 我们通常还需要给出一个标记不收敛的停止准则. 也就是说, 不论收敛与否,N 步迭代后停止运算(当然这里 N 通常来说就是指我们所能容许的最多的迭代次数). 超过这样一个值, 即便它能够收敛, 我们也需要将其标示出来, 需要分析其如此多的迭代次数仍为达到收敛的原因.

此外, 使用收敛准则的时候, 可考虑一个或多个收敛度量, 比如 $|x_{t+1} - x_t|$, $|x_{t+1} - x_t| / |x_t|$, 或 $|g(x_{t+1})|$. 如果每一个都不单减或若干次迭代后出现了周期,则迭代停止.

最后, 总结二分法的优缺点：算法简单, 而且收敛性总能得到保证. 其收敛速度很慢, 即它相对于后面讨论的其它方法而言, 为达到要求的精度, 它需要更多次的迭代; 但相对于其它方法具有明显的优势, 也就是如果 g 在区间 $[a_0, b_0]$ 上连续, 则不论 g' 是否存在或是否容易导出, 其根都可以由括入根法找到, 因为它们不必考虑 g' . 故, 相对其它强烈依赖 g 的光滑性的方法, 二分法有其合理的一面; 另外, 如果 g 在初始区间内的根多于 1,则容易看到二分法将找到其中一个,而找不到其余的.

例 4.1.1. (**分位数的计算**) 设连续型随机变量 X 的分布函数为 $F(x)$, 对任给的 $\alpha \in (0,1)$, 我们求取 $F(x)$ 的 α 下分位点, 即求取 $g(x) = F(x) - \alpha = 0$ 的根. 我们需要找到一个有根区间来开始使用算法 4.1.1. 一个简单方便的做法是利用 $g(x)$ 的单调性任取一初始点 a. 比如通常我们可取 a 为均值. 如果 $g(a) < 0$, 那么我们寻找第一个使得 $g(a+k) > 0$ 的正整数 k. 在找到这样的 k 之后, 显然 $[a+k-1, a+k]$ 就是有根区间; 反之, 若 $g(a) > 0$, 我们寻找第一个使得 $g(a+k) < 0$ 的负整数 k, 而 $[a+k, a+k+1]$ 有根区间. 当然, 我们亦可将这个搜索的步长 1 改为其它任何有可能加快搜索到有根区间进程的步长, 比如, 有时可取该步长为 $F(x)$ 的标准差.

§4.1.2 不动点迭代法(Fixed-Point Method)

一、原理：一个函数的不动点就是指此点的函数值等于其自身的点. 用不动点方法求根就是要确定一个函数 f 使得 $g(x) = 0$ 当且仅当 $f(x) = x$. 这样就把求 g 的根的问题变换成求 f 的不动点问题,而利用更新方程 $x_{t+1} = f(x_t)$ 就是寻找不动点的最简单方法.

对于方程 $g(x) = 0$, 为要应用不动点迭代法, 必须先将它改写成 $x = f(x)$ 的形式,即需要针对所给的函数 $g(x)$ 构造合适的迭代函数 $f(x)$. 我们可以尝试任何合适的 f,但选取 $f(x) = g(x) + x$ 是显然的,此时,其更新方程为

$$x_{t+1} = x_t + g(x_t). \tag{4.4}$$

二、收敛：此算法的收敛依赖于 f 是否是收缩的(contractive). 要使 f 在区间 $[a,b]$ 上是收缩的,则它必须满足：

1. 只要 $x \in [a,b]$,$f(x) \in [a,b]$, 且
2. 对某个 $\lambda \in [0,1)$, $|f(x_1) - f(x_2)| \leqslant \lambda |x_1 - x_2|$, $\forall\ x_1, x_2 \in [a,b]$.

注意到第二个条件就是 Lipschitz 条件,称 λ 为 Lipschitz常数. 如果 f 在区间 $[a,b]$ 上是收缩的,则在此区间内存在惟一的不动点 x^*,且对于此区间内的任一初值,此算法都将收敛到此不动点. 此外,在上述条件下,我们有

$$|x_t - x^*| \leqslant \frac{\lambda^t}{1-\lambda} |x_1 - x_0|. \tag{4.5}$$

算法 4.1.2. 不动点迭代法求方程根:

(0) 提供初值 x_0.

(1) 令 $t = t + 1$ 计算迭代值 $x_t = f(x_{t-1})$.

(2) 如果达到某一收敛准则, 则停止并返回解 x_t. 否则返回第一步.

有时也称不动点迭代法为泛函迭代 (*functional iteration*). 很多情况下, 不动点迭代法收敛相当缓慢, 因此有很多改进的方法来加速迭代过程, 比如著名的 Aitken 方法. 这里就不赘述了. 我们后面将介绍的常用的 Newton 法和正割法都亦是不动点迭代的特殊情况.

§4.1.3 Newton法

- **原理**: 假设 g 是连续可微的且 $g'(x^*) \neq 0$. 在第 t 次迭代,此方法通过线性 Taylor 级数展开

$$0 = g(x^*) \approx g(x_t) + (x^* - x_t)g'(x_t)$$

来近似 $g(x^*)$.

因为 g 可由在点 x_t 的切线值近似, 故用此切线的根来近似 g 的根看来是合理的. 于是, 解上述关于 x^* 的方程, 我们有

$$x^* = x_t - \frac{g(x_t)}{g'(x_t)} = x_t + h_t.$$

此方程告诉我们, 对 x^* 的近似依赖于当前的估计值 x_t 和一个修正 h_t. 重复此过程,则 Newton 法的更新方程为

$$x_{t+1} = x_t + h_t,$$

其中 $h_t = -g(x_t)/g'(x_t)$. 如用二次 Taylor 级数 $g(x_t)+(x^*-x_t)g'(x_t)+(x^*-x_t)^2g''(x_t)/2$ 来近似 $g(x^*)$, 则可得到类似的更新方程.

算法 4.1.3. 牛顿法求方程根:

(0) 令 $t = 0$, 给定一个初值 x_0.

(1) 令 $t = t + 1$. 如果 $[g'(x_{t-1})]^{-1}$ 存在, $x_t = x_{t-1} - \frac{g(x_{t-1})}{g'(x_{t-1})}$.

(2) 如果达到某一收敛准则, 则停止并返回解 x_t.

(3) 如果迭代次数达到预先指定的次数 N, 或者$g'(x_t) = 0$, 则方法失败; 否则返回第一步继续迭代.

我们一般采用前述的绝对 (4.2) 或相对 (4.3) 收敛准则来判断是否达收敛.

- **牛顿法的几何解释**: 对于方程 $g(x) = 0$, 如果 $g(x)$ 是线性函数, 则它的求根是容易的. 牛顿法实质上是一种线性化方法, 其基本思想是将非线性方程 $g(x) = 0$ 逐步归结为某种线性方程来求解. 也就是说切线方程 $g(x_t) + (x - x_t)g'(x_t) = 0$.

牛顿法有明显的几何解释. 方程 $g(x)=0$ 的根 x^* 可解释为曲线 $y=g(x)$ 与 x 轴的交点的横坐标. 设 x_t 是根 x^* 的某个近似值, 过曲线 $y=g(x)$ 上横坐标为 x_t 的点 P_t 引切线, 并将该切线与 x 轴的交点的横坐标 x_{t+1} 作为 x^* 的新的近似值 (当然我们期望该点更靠近 $g(x)$ 的零点). 由于这种几何背景, Newton 法也称切线法.

- **收敛条件**: Newton 法的收敛性依赖于 g 的形状和初值的选取.

(1). 假设 g 具有二阶连续导数, $g'(x^*)\neq 0$, 且 x^* 为 g 的一个单根, 则存在 x^* 的一个邻域, 当初值为此邻域内任一点时, Newton 法都收敛到 x^*.

证明: 因为 $g'(x^*)\neq 0$ 且 g' 在 x^* 处连续, 则必存在 x^* 的一个邻域, 使得在此邻域内 $g'(x)\neq 0$. 我们仅在此邻域内考虑, 且定义 $\epsilon_t = x_t - x^*$.

由 Taylor 展开有

$$0=g(x^*)=g(x_t)+(x^*-x_t)g'(x_t)+(x^*-x_t)^2 g''(\xi)/2,$$

其中 $\xi\in(x_t,x^*)$ 之间. 移项后, 我们有

$$x_t+h_t-x^*=(x^*-x_t)^2\frac{g''(\xi)}{2g'(x_t)},$$

其中 h_t 是 Newton 更新增量. 由于上式左边等于 $x_{t+1}-x^*$, 故我们有

$$\epsilon_{t+1}=(\epsilon_t)^2\frac{g''(\xi)}{2g'(x_t)}. \tag{4.6}$$

现对某个 $\delta>0$, 考虑 x^* 的邻域 $\mathcal{N}_\delta(x^*)=[x^*-\delta,x^*+\delta]$. 记

$$c(\delta)=\max_{x_1,x_2\in\mathcal{N}_\delta(x^*)}\left|\frac{g''(x_1)}{2g'(x_2)}\right|. \tag{4.7}$$

因为当 $\delta\to 0$ 时, $c(\delta)\to\left|\frac{g''(x^*)}{2g'(x^*)}\right|$, 所以 $\delta\to 0$ 时, $\delta c(\delta)\to 0$. 我们取满足 $\delta c(\delta)<1$ 的 δ, 则由 (4.6) 式得

$$|c(\delta)\epsilon_{t+1}|\leqslant\left(c(\delta)\epsilon_t\right)^2. \tag{4.8}$$

假设初值满足 $|\epsilon_0|=|x_0-x^*|\leqslant\delta$, 则由 (4.8) 式得

$$|\epsilon_t|\leqslant\frac{(c(\delta)\delta)^{2^t}}{c(\delta)},$$

它当 $t\to\infty$ 时收敛到 0, 于是, $x_t\to x^*$.

(2). 如果初值位于一个区间 $[a,b]$, 则需要验证下列一些条件. 如果

(i) 在区间 $[a,b]$ 上, $g'(x)\neq 0$,

(ii) 在区间 $[a,b]$ 上, $g''(x)$ 不变号,

(iii) $g(a)g(b) < 0$,

(iv) $|g(a)/g'(a)| < b-a$ 且 $|g(b)/g'(b)| < b-a$,

则对于此区间内的任一个初值 x_0, Newton 法都将收敛.

- 两种最常见的不收敛情形：(i) $g'(x_t)$=0; (ii) 初值距离真值太远.
- **收敛阶数**. 定义：收敛阶数是用来度量如 Newton 法等求根方法的收敛速度的一个量. 称一个方法的收敛阶数为 β, 如果 $\lim_{t\to\infty}\epsilon_t = 0$ 且

$$\lim_{t\to\infty}\frac{|\epsilon_{t+1}|}{|\epsilon_t|^{\beta}} = c, \tag{4.9}$$

其中常数 $c \neq 0$ 且 $\beta > 0$. 也就是指在收敛过程中迭代误差的下降速度.

注：

a. 在可能精确近似真值的意义下, 高阶收敛为优.

b. 某些高阶收敛方法是以付出稳健的代价而取得的, 某些速度较慢的方法会比其对应的快速算法更安全.

c. 对于 Newton 法, (4.6) 式指出

$$\frac{\epsilon_{t+1}}{(\epsilon_t)^2} = \frac{g''(\xi)}{2g'(x_t)}. \tag{4.10}$$

如果 Newton 法收敛, 则其连续性告诉我们, 此方程的右端收敛到 $\frac{g''(x^*)}{2g'(x^*)}$. 于是, Newton法二次收敛, 即 $\beta = 2$ 且 $c = \frac{g''(x^*)}{2g'(x^*)}$.

d. 对于二分法, 如果在其初始区间有解的话, 由于其每次迭代区间的长度均减半且 $\lim_{t\to\infty}|\epsilon_t| = 0$, 故它显示出具有类似线性收敛 ($\beta = 1$) 的特点. 然而, 距离 $x_t - x^*$ 不一定每次迭代都缩小, 且它们的比值可能是无界的, 于是, 对于任何 $\beta > 0$, $\lim_{t\to\infty}\frac{|\epsilon_{t+1}|}{|\epsilon_t|^{\beta}}$ 可能不存在. 这样, 二分法从形式上就不满足收敛阶数的定义.

- 当 g' 无解析形式或不易显示表出时, 我们有时可使用如下的数值近似来代替它,

$$g'(x_t) = \frac{g(x_t+h) - g(x_t)}{h}.$$

通常将这种方法称为离散牛顿法, 我们将看到后面将要介绍的正割法同该离散法机理是类似的.

- **Fisher得分法**：考虑前面的例子, 如果 $\widehat{\theta}$ 是 MLE, 则它最大化其对数似然, 即 $\widehat{\theta}$ 是得分方程 $l'(\theta) = 0$ 的解. 由数理统计知识我们知道当观测来自于指数分布族时, 我们有

$$I(\theta) = E[l'^2(\theta)] = -E(l''(\theta)).$$

因此可用 $-l''(\theta)$ 来近似 $I(\theta)$. 于是, 当 g 对应着MLE的优化问题时, 在Newton更新方程中, 我们可以用 $I(\theta)$ 来替换 $-l''(\theta)$, 此时其更新增量为 $h_t = l'(\theta_t)/I(\theta_t)$, 其中 $I(\theta_t)$ 为在 θ_t 点的期望 Fisher 信息量. 这样, 此更新方程为

$$\theta_{t+1} = \theta_t + l'(\theta_t)/I(\theta_t). \tag{4.11}$$

此方法被称为 Fisher**得分法**.

Fisher 得分法与 Newton 法具有相同的渐近性质, 但对于个别问题, 一个可能比另一个易于计算或分析. 一般来讲, Fisher 得分法在迭代之初效果明显, 而 Newton 法则在迭代结束前效果明显.

§4.1.4 正割法(或弦截法; Secant method)

- **原理** 在Newton法中, 其更新增量依赖其一阶导数 $g'(x_t)$. 如果计算此导数比较困难, 则可以用离散差分 $\frac{g(x_t)-g(x_{t-1})}{x_t-x_{t-1}}$ 来近似之. 称此方法为**正割法**(secant method), 其更新方程为

$$x_{t+1} = x_t - g(x_t)\frac{x_t - x_{t-1}}{g(x_t) - g(x_{t-1})},\ \forall\ t \geqslant 1. \tag{4.12}$$

此方法需要两个初值 x_0, x_1. 这样导出的迭代公式相当于牛顿迭代

$$x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)}$$

中的导数 $g'(x_t)$ 用差商 $\frac{g(x_t)-g(x_{t-1})}{x_t-x_{t-1}}$ 取代的结果.

算法 4.1.4. 正割法求方程根:

(0) 令 $t = 1$, 给定两个初值 x_0 和 x_1.

(1) 令 $t = t + 1$. $x_t = x_{t-1} - g(x_{t-1})\frac{x_{t-1}-x_{t-2}}{g(x_{t-1})-g(x_{t-2})}$.

(2) 如果达到某一收敛准则, 则停止并返回解 x_t.

(3) 如果迭代次数达到预先指定的次数 N; 否则返回第一步继续迭代.

- **正割法的几何解释:** 曲线 $y = g(x)$ 上横坐标为 x_t, x_{t-1} 的点分别为 P_t, P_{t-1}, 则弦线 $\bar{P}_t\bar{P}_{t-1}$ 的斜率等于差商值 $\frac{g(x_t)-g(x_{t-1})}{x_t-x_{t-1}}$, 其方程是

$$g(x_t) + \frac{g(x_t)-g(x_{t-1})}{x_t - x_{t-1}}(x - x_t) = 0.$$

因此, 按 4.12 求出的 x_{t+1} 实际上是弦线 $\bar{P}_t\bar{P}_{t-1}$ 与 x 轴交点的横坐标. 因此该算法亦称为弦截法. 注意到该法与牛顿法都是线性化方法, 但有本质上的区别. 后者在计算 x_{t+1} 时只用到前一步的 x_t 值, 而前者在求 x_{t+1} 时用到前面两步的结果 x_t, x_{t-1}, 因此使用该法需给出两个初始值.

- **收敛阶数**. 在类似于 Newton 法的条件下, 正割法也将收敛到根 x^*. 为求得其收敛阶数, 我们仅在某个合适的小区间 $[a,b]$ 内考虑, 且假设此区间包含 x_0, x_1 和 x^*, 且在此区间内 $g'(x) \neq 0, g''(x) \neq 0$. 记 $\epsilon_{t+1} = x_{t+1} - x^*$, 则可直接证得

$$\begin{aligned} \epsilon_{t+1} &= \left[\frac{x_t - x_{t-1}}{g(x_t) - g(x_{t-1})}\right]\left[\frac{g(x_t)/\epsilon_t - g(x_{t-1})/\epsilon_{t-1}}{x_t - x_{t-1}}\right][\epsilon_t \epsilon_{t-1}] \\ &= A_t B_t \epsilon_t \epsilon_{t-1}, \end{aligned}$$

其中当 $x_t \to x^*$ 且 g' 连续时, $A_t \to 1/g'(x^*)$.

为得到 B_t 的极限, 对 g 在 x^* 处进行Taylor展开：

$$g(x_t) \approx g(x^*) + (x_t - x^*)g'(x^*) + (x_t - x^*)^2 g''(x^*)/2,$$

于是,

$$g(x_t)/\epsilon_t \approx g'(x^*) + \epsilon_t g''(x^*)/2.$$

类似地, $g(x_{t-1})/\epsilon_{t-1} \approx g'(x^*) + \epsilon_{t-1} g''(x^*)/2$. 这样,

$$B_t \approx g''(x^*)\frac{\epsilon_t - \epsilon_{t-1}}{2(x_t - x_{t-1})} = g''(x^*)/2,$$

经仔细验证, 可证当 $x_t \to x^*$ 时, 上述近似是严格的. 于是,

$$\epsilon_{t+1} \approx d_t \epsilon_t \epsilon_{t-1}, \tag{4.13}$$

其中当 $t \to \infty$ 时, $d_t \to \frac{g''(x^*)}{2g'(x^*)} = d$.

为求得正割法的收敛阶数, 我们必须找到 β 满足：$\lim_{t\to\infty} \frac{|\epsilon_{t+1}|}{|\epsilon_t|^\beta} = c$, 其中 c 为常数. 为此, 先假设上式成立, 并用 ϵ_{t-1} 与 ϵ_{t+1} 之比来表达(4.13)式中的 ϵ_t, 且代入上式, 经整理后, 有

$$\lim_{t\to\infty} |\epsilon_t|^{1-\beta+1/\beta} = \frac{c^{1+1/\beta}}{d}. \tag{4.14}$$

因(4.14)式右端为正数, 故 $1 - \beta + 1/\beta = 0$, 其解为 $\beta = (1+\sqrt{5})/2 \approx 1.62$. 于是, 正割法的收敛阶数低于Newton法.

- 多根问题 很多问题中我们面对的方程具有多个根, 我们当然希望将他们全部找出来. 通用的处理方法是在前述的这些迭代方法中尝试多个初始点. 所以这个时候利用画图在很多情况下或有所帮助. 但是, 一般来说, 如果方程根的个数未知, 我们没有一种一般的方法能够保证找到全部的根.

§4.2 多变量非线性方程求解问题

如果方程的参数是一个 m 维向量, 而方程值是一个 n 维的向量, 多变量方程系统 $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ 也就是如下的形式:

$$g_1(x_1, x_2, \ldots, x_m) = 0, \ldots, g_n(x_1, x_2, \ldots, x_m) = 0,$$

其中每一个 g_i 都是关于向量 $\mathbf{x}$ 的一个标量取值的函数. 解这样的非线性方程组所需计算量明显要比前面的单变量单方程的问题大得多.

这样一个非线性方程组是否有解本身就是难以确定的. 一般来说, 不像线性方程组的问题, 我们不能够给出非线性方程组有解的的简单的充分必要条件. 当然, 如果 $n > m$, 则方程组很可能没有解. 甚至当 $n = m$ 时, 我们仍旧没有简单方法来确定解的存在.

解该类问题的方法并不像我们前面介绍的单变量问题那样具有许多不同方法. 我们这里仅考虑一种方法, 也就是类似于单变量问题中的牛顿法. 许多其它解非线性方程组的方法也基本上是基于牛顿法的一些变型. 我们这里不详细讨论, 而将会在下一节考虑我们的主要目标也就是优化问题的时候再详加介绍.

注意到牛顿法需要用到导数, 因此我们下面假设方程是可微的. 由上一节我们看到, 牛顿迭代法利用导数或者导数的近似来决定我们从某一个点向前移动的方向. 而如果我们考虑的是一个关于多元变量 $\mathbf{x}$的函数 g_i, 那么这时我们需要考虑多个方向上的斜率, 也就是梯度 $\nabla g_i(\mathbf{x})$. 而对于上述的非线性方程组 $\mathbf{g}(\mathbf{x}) = 0$, 我们则需要考虑所有函数所对应的梯度; 也就是, Jacobian 矩阵 $\mathbf{J}_g \equiv (\nabla \mathbf{g})^T = (\nabla g_1, \nabla g_2, \ldots, \nabla g_n)^T$, where $\nabla g_i = \left(\frac{\partial g_i(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial g_i(\mathbf{x})}{\partial x_m} \right)^T$.

由 $\mathbf{x}_t$ 点处的一阶泰勒展开, 我们有

$$0 = \mathbf{g}(\mathbf{x}^*) \approx \mathbf{g}(\mathbf{x}_t) + \mathbf{J}_g(\mathbf{x}_t)\,(\mathbf{x}^* - \mathbf{x}_t) \tag{4.15}$$

注意到 (4.15) 式左端实际上是 $\mathbf{g}(\mathbf{x}^*)$ 的线性Taylor级数近似,且求解(4.15)就相当于求此线性方程的根. 无论从哪个角度看,多元Newton迭代的增量都为 $\boldsymbol{h}_t = -\mathbf{J}_g^{-1}(\mathbf{x}_t)\mathbf{g}(\mathbf{x}_t)$. 由此我们有如下算法

算法 4.2.1. 牛顿法求解非线性方程组:

(0) 令 $t = 0$, 给定一个初值 $\mathbf{x}_0$.

(1) 令 $t = t + 1$. 求解线性方程组

$$\mathbf{J}_g(\mathbf{x}_{t-1})\,(\mathbf{x}_t - \mathbf{x}_{t-1}) = -\mathbf{g}(\mathbf{x}_{t-1}).$$

(2) 如果达到某一收敛准则, 则停止并返回解 $\mathbf{x}_t$.

(3) 如果迭代次数达到预先指定的次数 N 则停止; 否则返回 (1) 继续迭代.

前面讨论的关于单变量优化问题的一般原则也适应于多元情形, 尽管在形式上有些小的改变, 但收敛准则仍是类似的. 为构建收敛准则, 以 $D(\boldsymbol{u},\boldsymbol{v})$ 记两个 m 维向量间的距离. 两个显然的选择为 $D(\boldsymbol{u},\boldsymbol{v})=\sum_{i=1}^{m}|u_i-v_i|$ 和 $D(\boldsymbol{u},\boldsymbol{v})=\sqrt{\sum_{i=1}^{m}(u_i-v_i)^2}$. 则绝对和相对收敛准则由如下不等式给出：

$$D(\mathbf{x}_{t+1},\mathbf{x}_t)<\epsilon,\ \frac{D(\mathbf{x}_{t+1},\mathbf{x}_t)}{D(\mathbf{x}_t,\mathbf{0})}<\epsilon,\ \text{或}\ \frac{D(\mathbf{x}_{t+1},\mathbf{x}_t)}{D(\mathbf{x}_t,\mathbf{0})+\eta}<\epsilon.$$

当 $m=n$ 且 Jacobian 矩阵可逆时, 算法中的线性方程组亦可写作更新方程

$$\mathbf{x}_{t+1}=\mathbf{x}_t-\mathbf{J}_g^{-1}(\mathbf{x}_t)\mathbf{g}(\mathbf{x}_t). \tag{4.16}$$

但是需要注意的是, 这么写并不代表我们在编程的时候需要通过求逆来方程组. 事实上, 线性方程组有良好快速的迭代算法来求解而不需要通过求逆, 绝大多数的统计软件包中都包含了针对的函数或子程序来解 (solution of linear system). 这样表示在许多算法分析或者统计理论性质的研究中有其方便之处.

有时, 与在单变量牛顿法类似, 多元 Newton 法中我们也可用一个有限差分商的矩阵 $\mathbf{M}_t$ 近似 Jacobian 矩阵. 以 $\boldsymbol{e}_j$ 记第 j 个分量为 1 而其它均为 0 的 m-维向量. 一个最直接的方法是：令 $\mathbf{M}_t$ 的第 (i,j) 元等于

$$\mathbf{M}_t^{(ij)}=\frac{\partial g_i}{\partial x_j}=\frac{g_i\left(\mathbf{x}_t+h_t^{(ij)}\boldsymbol{e}_j\right)-g_i(\mathbf{x}_t)}{h_t^{(ij)}}, \tag{4.17}$$

其中 $h_t^{(ij)}$ 为常数. 对于所有的 (i,j) 和 t,取 $h_t^{(ij)}=h$ 最容易,但其收敛阶数 $\beta=1$. 另外,如果我们对于所有的 i,取 $h_t^{(ij)}=x_t^{(j)}-x_{t-1}^{(j)}$,则得到的收敛阶数类似于单变量的正割法,其中 $x_t^{(j)}$ 为 $\mathbf{x}_t$ 的第 j 个分量.

§4.3 无约束优化问题

我们现在考虑我们的主要问题, 优化问题 (4.1). 在一个多元优化问题中,假设 g 是 m 维向量 $\mathbf{x}=(x_1,\ldots,x_m)^T$ 的实值函数,我们要求其最值. 我们记其一个解, 即最值为

$$\mathbf{x}^*=\arg\min_{\mathbf{x}\in D}g(\mathbf{x}),$$

其中 $\mathbf{x}$ 是 m 维向量而 g 是一连续实值标量函数. 我们这里首先考虑 $D=\mathbb{R}^m$, 也就是连续无约束优化问题. 并且, 在这一节中我们亦假设函数 g 可微且我们通常假定是二次可微的.

优化问题 (4.1) 的数值求解通常是一个迭代过程, 也就是从函数上的某一个点移动到另一个. 基本步骤描述如下：

(1) 取初值 $\mathbf{x}_0$;

(2) 计算目标函数 $g(\mathbf{x}_t)$;

(3) 确定搜索方向 $\mathbf{d}_t$ (direction or step)和步长因子 λ_t (step factor), 使 $\mathbf{x}_{t+1} = \mathbf{x}_t + \lambda_t \mathbf{d}_t$ 满足 $g(\mathbf{x}_{t+1}) \le g(\mathbf{x}_t)$.

(4) 如果达到某一收敛准则, 则停止并返回解 $\mathbf{x}_{t+1}$; 否则, $t = t + 1$ 转 (2).

在这些步骤中, 关键是如何确定搜索方向和步长因子, 使得从 $\mathbf{x}_t$ 出发找到比 $\mathbf{x}_t$ 更接近 $\mathbf{x}^*$ 的点 $\mathbf{x}_{t+1}$. 确定搜索方向的不同方法对应于不同的迭代算法. 这里首先最重要的是确定搜索方向, 而我们在下一节中首先考虑这一方向已经确定, 而来确定我们要移动的步长因子.

§4.3.1 一维(步长)搜索法

在前述多变量优化问题 (4.1) 中, 经常要涉及到一维搜索问题. 即从给定点 $\mathbf{x}$ 出发, 沿某一确定的适当下降方向 $\mathbf{d}$, 求单变量 λ 的函数

$$q(\lambda) = g(\mathbf{x} + \lambda \mathbf{d})$$

的极小点 λ^*, 这就是一个一维搜索过程或者就是一个一维最优化的问题, 也是我们优化方法中最基本方法.

这里我们面临的一个问题是: 如何去协调分配确定好的搜索方向和寻找优良的步长因子的所需计算量. 一般来说, 如果计算量允许, 且 $q(\lambda)$ 的导数易于计算或显示表达, 则我们自然地会把求极小点的问题转化为求方程 $q'(x) = 0$ 的根. 这时我们即可方便地针对我们的需要和函数的类型, 使用在 4.1 节中介绍的各种方法, 这里就不再详细赘述; 相反地, 我们通常更关心搜索方向的确定, 因此, 在很多时候只要给出一个该步长最优值的较为合理的近似即可. 我们下面主要考虑两种常用的直接方法.

黄金分割法: 黄金分割法是求单峰函数在给定区间 $[a, b]$ 上极值的一种试探法. 对于 $[a, b]$ 上的单峰函数 $q(x)$, 记 x^* 为极小点. 对 $[a, b]$ 上任意两个试探点 $x_1 < x_2$, 若 $q(x_1) < q(x_2)$, 则 $x^* \in [a, x_2]$; $q(x_1) \ge q(x_2)$, $x^* \in [x_1, b]$. 这说明只要计算出搜索区间内任两个试探点的函数值, 就可以把搜索区间缩小. 可见, 反复多次比较试探点的函数值, 能够越来越精确地估计 x^* 的位置.

可用以下两条原则来选取 x_1 和 x_2 来简捷又迅速地找到 x^*.

(i) 对称原则. 由于计算前不能预测 $q(x_1)$ 和 $q(x_2)$ 哪个小, 故既有可能去掉 $[a, x_1]$, 也有可能去掉 $[x_2, b]$. 为稳妥, 可选取 x_1 和 x_2 使得每次去掉的区间长度相等, 即 $x_1 - a = b - x_2$. 只要给出 x_1, 就能算出 x_2, 且这两点在 $[a, b]$ 中处于对称位置 (关于 $[a, b]$ 中点).

(ii) 等比收缩原则.

给定某 x_1, 不妨假定 $q(x_1) \ge q(x_2)$, 则去掉的长度为 $x_1 - a$. 若由对称原则, 如果选择的 x_1 接近 $[a, b]$ 的中点, 那么这一次丢掉的区间长度大, 或者说这一次收缩较快. 但这不能保证下一次和今后每次都收缩较快. 我们通常希望区间的收缩稳定为好, 不要时快时慢. 一个常

用的方法是希望每次留下的区间是原来长度的 α 倍 ($\alpha < 1$). 这就是压缩区间长度的的等比收缩原则.

不妨设区间 $[a, b]$ 的长度为 1, 并假设第一次删去 $[x_2, b]$, 则根据等比收缩原则有

$$\frac{x_2 - a}{x_1 - a} = \frac{b - a}{x_2 - a},$$

即 $\alpha/(1-\alpha) = 1/\alpha$, 则我们有 $\alpha = \frac{-1+\sqrt{5}}{2} \approx 0.618$ ($0 < \alpha < 1$). 这就是著名的 0.618 法的由来. 使用该比率缩短搜索区间的迭代方法就被称为 0.618 法或黄金分割法 (在古代人们认为按 0.618 的比率分割线段是最协调的).

算法 4.3.1. 求单峰函数 $q(x)$ 极值的0.618法:

(0) 确定 $q(x)$ 的初始搜索区间 $[a, b]$ 和 某迭代精度 ϵ;

(1) 按以下公式计算初始试探点:

$$x_1 = a + (1-\alpha)(b-a),\ x_2 = a + \alpha(b-a),\ \alpha = 0.618;$$

(2) 计算 $q(x_1)$ 和 $q(x_2)$, 若 $q(x_2) > q(x_1)$, 则置 $b = x_2, x_2 = x_1, a = a$; 并计算新内点: $x_1 = a + (1-\alpha)(b-a)$; 否则置 $a = x_1, x_1 = x_2, b = b$, 并计算新内点: $x_2 = a + \alpha(b-a)$. 然后转向(3);

(3) 检验 $b - a < \epsilon$, 若成立, 停止迭代, 同时令 $x^* = (a+b)/2$; 否则转 (2).

0.618 法对函数除要求是单峰外, 不做其他要求, 甚至可以是不连续函数, 故该方法的适用面相当广泛, 比如在函数光滑中使用 cross-validation 来搜索最优带宽, 或在惩罚似然最小二乘中搜索惩罚参数, 都可以用该法来替代我们常用的最原始的方法, 也就是在 $[a, b]$ 区间内设置足够多的离散节点来找这些点中使得 $q(x)$ 达到最小的节点.

二次插值法: 插值法的基本思想是根据 $q(x)$ 在某些试探点的信息, 构造一个与 $q(x)$ 近似的函数 $\widehat{q}(x)$, 在一定的条件下可以期望 $\widehat{q}(x)$ 会接近 $q(x)$ 的最小点. 因此可取 $\widehat{q}(x)$ 的极小点作为新的试探点, 然后设法缩短搜索区间. 显然近似函数 $\widehat{q}(x)$ 应该比较简单, 以便容易地求出它的极小点.

假设已知函数 $q(x)$ 在三个点 x_1, x_2 和 x_3 (不妨假设 $x_1 < x_2 < x_3$) 的函数值满足两头大, 中间小, 即

$$q(x_1) > q(x_2),\ q(x_3) > q(x_2). \tag{4.18}$$

则我们通过 $(x_1, q(x_1))$, $(x_2, q(x_2))$ 和 $(x_3, q(x_3))$ 三个点来作抛物线

$$P(x) = a_2 x^2 + a_1 x + a_0\ (a_2 \neq 0).$$

该条抛物线看作是在区间 $[x_1, x_3]$ 内 $q(x)$ 的近似. 我们通过求取 $P(x)$ 的极小点 x_4 作为 $q(x)$ 极小点的新估计值. 不难求出并证明

$$x_4 = \frac{(x_2^2 - x_3^2)q(x_1) + (x_3^2 - x_1^2)q(x_2) + (x_1^2 - x_2^2)q(x_3)}{2[(x_2 - x_3)q(x_1) + (x_3 - x_1)q(x_2) + (x_1 - x_2)q(x_3)]} \tag{4.19}$$

是抛物线看 $P(x)$ 的极小点, 且满足 $x_1 < x_4 < x_3$.

我们则可通过 x_4 提供的信息进一步缩短搜索区间 $[x_1, x_3]$ 从而进行下一次近似迭代直至终止. 总结算法如下:

算法 4.3.2. 二次插值法:

(0) 确定 $q(x)$ 的初始搜索区间 $[x_1, x_3]$ 和某个 x_2 使得满足 (4.18);

(1) 利用 (4.19) 计算 x_4;

(2) 比较 x_2 和 x_4, 若 $x_4 > x_2$, 转 (3); 否则转 (4).

(3) 若 $q(x_2) \geq q(x_4)$, 则置 $x_1 = x_2, x_2 = x_4$, 即以原 $\{x_2, x_4, x_3\}$ 为搜索区间; 否则置 $x_3 = x_4$, 即以原 $\{x_1, x_2, x_4\}$ 为搜索区间;

(4) 若 $q(x_2) \geq q(x_4)$, 则置 $x_3 = x_2, x_2 = x_4$, 即以原 $\{x_1, x_4, x_2\}$ 为搜索区间; 否则置 $x_1 = x_4$, 即以原 $\{x_4, x_2, x_3\}$ 为搜索区间.

(5) 新的搜索区间 $\{x_1, x_2, x_3\}$ 的长度缩小了, 且满足 (4.18), 因此返回 (1) 在新的搜索区间上重新计算插值抛物线的极小点进行迭代直到某终止准则达到. [比如可采用 $q(x_2)$ 和 $q(x_4)$ 的绝对或相对距离, 或者 $|x_1 - x_3| < \epsilon$].

上面两种常用的直接搜索法都依赖于找到有极值的区间或者说找到满足两头大中间小的区间及初始值. 在实际问题中, 我们往往可以很容易的给定一个 x 的上界或下界, 不妨设为下界 x_1. 下面的算法可以帮助我们较为方便地初始化前面的两种算法.

算法 4.3.3. 进退法:

(0) 确定最优解 x^* 的一个下界, x_1, 且给定一个初始步长 δ;

(1) 若 $q(x_1 + \delta) < q(x_1)$, 置 $x_2 = x_1 + \delta$, 转 (2); 否则, x^* 落在左边, 置 $x_3 = x_1 + \delta$, 转 (3).

(2) 进: 置 $\delta = 2\delta$. 若 $q(x_2) < q(x_2 + \delta)$, 置 $x_3 = x_2 + \delta$ 并停止; 否则, $x_1 = x_2$, $x_2 = x_2 + \delta$, 返回重复 (2).

(3) 退: 置 $\delta = \frac{1}{2}\delta$. 若 $q(x_1 + \delta) < q(x_1)$, 置 $x_2 = x_1 + \delta$ 并停止; 否则, $x_3 = x_1 + \delta$, 返回重复 (3).

(4) 停止后返回 $\{x_1, x_2, x_3\}$ 即得到我们希望的有解区间及三个初始点.

§4.3.2 方向搜索及最速下降法

先回顾一下凸函数: 如果函数 g 在 $[a,b]$ 上连续, (a,b) 内可导且导数是非降的. 如果二阶导 g'' 存在, 则对 $x\in(a,b)$, $g''(x)\geq 0$. 对于可微函数, 偏导向量, 也就是梯度 $\nabla g(x)$, 刻画了该函数的局部信息,

$$\nabla g=\left(\frac{\partial g(\mathbf{x})}{\partial x_1},\ldots,\frac{\partial g(\mathbf{x})}{\partial x_m}\right)^T.$$

当 g 是凸函数且梯度存在时, 梯度向量中的每一维均是非降的. 另外, Hessian 矩阵, 其提供了关于稳定点的信息,

$$\begin{aligned}\mathbf{H}_g&=\nabla(\nabla g(\mathbf{x}))=\nabla^2 g(\mathbf{x})\\&=\frac{\partial g(\mathbf{x})}{\partial\mathbf{x}\partial\mathbf{x}^T}=\left(\frac{\partial^2 g(\mathbf{x})}{\partial x_i\partial x_j}\right)_{m\times m}\end{aligned}$$

注意这里 Hessian 矩阵是一个关于 $\mathbf{x}$ 的函数, 所以我们通常将其记作 $\mathbf{H}_g(\mathbf{x})$. 对于凸函数, 如果 Hessian 存在, 则其是半正定的 (严凸则正定), 即 $\mathbf{H}_g\geq 0$.

求解优化问题 (4.1) 的迭代算法的基本特点是: 从给定的初始点出发, 通过搜索一步步地接近问题的解, 且要求每一步目标函数值有所下降. 因此该类迭代算法统称下降算法(descent algorithm). 现在的问题是, 从任一点出发, 沿什么方向可使目标函数下降, 沿什么方向可使目标函数下降最快? 根据微积分知识, 我们知道使目标函数下降最快的方向是负梯度方向.

由 Taylor 展开

$$g(\mathbf{x}+\mathbf{d})=g(\mathbf{x})+\mathbf{d}^T\nabla g(\mathbf{x})+o(||\mathbf{d}||),$$

$g(\mathbf{x})+\mathbf{d}^T\nabla g(\mathbf{x})$ 作为 $g(\mathbf{x}+\mathbf{d})$ 的一次近似, $\mathbf{d}$ 取什么方向, 可使函数 $g(\mathbf{x}+\mathbf{d})$ 下降. 显然, 我们称一个向量 $\mathbf{d}$ 使得

$$\mathbf{d}^T\nabla g(\mathbf{x})<0$$

为一个 **下降方向**(descent direction). 如果 $\nabla g(\mathbf{x})\neq 0$, 我们可以将 $\mathbf{d}$ 表示为

$$\mathbf{R}\mathbf{d}=-\nabla g(\mathbf{x}),\tag{4.20}$$

其中 $\mathbf{R}$ 是某个正定阵. 选择不同的 $\mathbf{R}$ 我们则可得到不同的下降方向.

进一步地, $\mathbf{d}$ 取什么方向下降最快? 即 $\mathbf{d}$ 取什么方向时, 可使

$$g(\mathbf{x}+\mathbf{d})-g(\mathbf{x})\approx\mathbf{d}^T\nabla g(\mathbf{x})$$

取得最小值. 由 Cauchy-Schwarz 不等式知,

$$\mathbf{d}^T\nabla g(\mathbf{x})\geq-||\mathbf{d}||\cdot||\nabla g(\mathbf{x})||,$$

等号当且仅当 $\mathbf{d}=-\nabla g(\mathbf{x})/||\nabla g(\mathbf{x})||$ 时成立. 即 $\mathbf{d}$ 取负梯度方向时, $g(\mathbf{x}+\mathbf{d})$ 下降最快. 也就是说, 负梯度方向是 $g(\mathbf{x})$ 在点 $\mathbf{x}$ 的邻域内函数值下降最快的方向, 故称负梯度方向为最速下

降方向 (steepest descent direction). 即 (4.20) 中 $\mathbf{R}$ 取为单位阵. 我们又称以该负梯度为搜索方向的算法, 称为最速下降法.

确定从点 $\mathbf{x}_t$ 出发的搜索方向 $\mathbf{d}$ 后, 沿 $\mathbf{d}$ 方向上找一点 $\mathbf{x}_{t+1}=\mathbf{x}_t+\lambda\mathbf{d}$, $g(\mathbf{x}_t+\lambda\mathbf{d})$ 在 $\mathbf{x}_{t+1}$ 关于 λ 取最小值. 从 $\mathbf{x}_{t+1}$ 出发重复以上过程反复迭代, 即得问题 (4.1) 的解. 总结最速下降法算法如下：

算法 4.3.4. 最速下降法:

(0) 令 $t=0$, 给定一个初值 $\mathbf{x}_0$, 并计算 $g(\mathbf{x}_0)$;

(1) 求目标函数 $g(\mathbf{x})$ 在 $\mathbf{x}=\mathbf{x}_t$ 的梯度方向 $\nabla g_t=\nabla g(x_t)=\left(\frac{\partial g(\mathbf{x})}{\partial x_1},\ldots,\frac{\partial g(\mathbf{x})}{\partial x_m}\right)^T_{|\mathbf{x}=\mathbf{x}_t}$. 若 $||\nabla g_t||<\epsilon_1$, 则停止迭代, 取 $\mathbf{x}^*=\mathbf{x}_t$; 否则令 $\mathbf{d}_t=-\nabla g_t$;

(2) 求步长因子 λ_t 使

$$g(\mathbf{x}_t+\lambda_t\mathbf{d}_t)=\min_{\lambda>0}g(\mathbf{x}_t+\lambda\mathbf{d}_t);$$

(3) 取 $\mathbf{x}_{t+1}=\mathbf{x}_t+\lambda_t\mathbf{d}_t$;

(4) 若达到收敛条件, 则停止迭代, 返回 $\mathbf{x}^*=\mathbf{x}_{t+1}$; 否则置 $t=t+1$ 转 (1).

此算法的特点是: (1) 简单易行, 用一次函数作为目标函数 $g(x)$ 的近似; (2) 不管初始点如何选取, 保证每次迭代均使目标函数减少; (3) 收敛速度慢, 因为相邻两次迭代的搜索方向正交. 下面来看一个简单的例子来说明该方法的操作过程.

例 4.3.1. 用最速下降法求 $g(x_1,x_2)=\frac{1}{3}x_1^2+\frac{1}{2}x_2^2$ 的极小点.

解: 取初始点 $x_0=(3,2)^T$, $f(x_0)=5$. 在 x_0 的梯度方向

$$\nabla g(x_0)=(\frac{2}{3}x_1,x_2)^T=(2,2)^T$$

取搜索方向 $d_0=(-2,-2)^T$.

由 x_0 出发, 沿 d_0 方向的点 $x=x_0+\lambda d_0=(3-2\lambda,2-2\lambda)^T$. 考虑函数

$$g(x_0+\lambda d_0)=\frac{1}{3}(3-2\lambda)^2+\frac{1}{2}(2-2\lambda)^2$$

解得 $\lambda_0=\frac{6}{5}$. 故有 $x_1=x_0+\lambda_0 d_0=(\frac{3}{5},-\frac{2}{5})^T$, 且 $g(x_1)=\frac{1}{5}$. 以 x_1 为出发点重复以上步骤, 类似可得 $x_t=(\frac{3}{5^t},(-1)^t\frac{2}{5^t})^T\to(0,0)$ as $t\to\infty$.

§4.3.3 Newton 法和 Fisher 得分法

以 $\mathbf{x}_t$ 记第 t 步最优点的估计.

$$g(\mathbf{x}+\mathbf{d})\approx g(\mathbf{x})+\mathbf{d}^T\nabla g(\mathbf{x})+\frac{1}{2}\mathbf{d}^T\mathbf{H}_g(\mathbf{x})\mathbf{d}.$$

即 $g(\mathbf{x})+\mathbf{d}^T\nabla g(\mathbf{x})+\frac{1}{2}\mathbf{d}^T\mathbf{H}_g(\mathbf{x})\mathbf{d}$ 作为 $g(\mathbf{x}+\mathbf{d})$ 的二次近似. 当 $\mathbf{H}_g$ 为正定时, $\mathbf{d}^T\nabla g(\mathbf{x})+\frac{1}{2}\mathbf{d}^T\mathbf{H}_g(\mathbf{x})\mathbf{d}$ 有最小值满足

$$\nabla g(\mathbf{x})+\mathbf{H}_g(\mathbf{x})\mathbf{d}=0.$$

则取 $\mathbf{d}=-\mathbf{H}_g^{-1}(\mathbf{x})\nabla g(\mathbf{x})$ 为搜索方向, 得 $x_{t+1}=x_t-\mathbf{H}_g^{-1}(\mathbf{x}_t)\nabla g(\mathbf{x}_t)$. 该式即为牛顿法的迭代公式. 不难看出, 牛顿法是在 (4.20) 中 $\mathbf{R}$ 取为 Hessian 矩阵的下降法.

算法 4.3.5. 牛顿法:

(0) 令 $t=0$, 给定一个初值 $\mathbf{x}_0$. 设定两个精度 ϵ_1 和 ϵ_2;

(1) 计算 ∇g_t, 若 $D(\nabla g_t,\mathbf{0})<\epsilon_1$, 则停止迭代, 取 $\mathbf{x}^*=\mathbf{x}_t$ 为解. 否则, 求解线性方程组 $\mathbf{H}_t\mathbf{d}=-\nabla g_t$, 可得 $\mathbf{d}_t=-\mathbf{H}_t^{-1}\nabla g_t$;

(2) 令 $\mathbf{x}_{t+1}=\mathbf{x}_t+\mathbf{d}_t$;

(3) 若达到收敛条件 (如 $D(\mathbf{x}_{t+1},\mathbf{x}_t)<\epsilon_2$ 或 $|g(\mathbf{x}_{t+1})-g(\mathbf{x}_t)|<\epsilon_2$), 则停止迭代, 返回 $\mathbf{x}^*=\mathbf{x}_{t+1}$; 否则置 $t=t+1$ 转 (1).

当初始点选择的不好时, 也就是说离极小点比较远, 牛顿法则不能够保证收敛或者收敛速度很慢, 其主要原因是可能在其中的几次迭代中未能保证下降. 为了克服这一缺点, 可考虑改进 $\mathbf{x}_{t+1}$ 的迭代公式, 即只把增量的方向设定为搜素方向 $\mathbf{d}_t=-\mathbf{H}_t^{-1}\nabla g_t$, 引入步长因子 λ_t, 并由一维搜索方法确定 λ_t. 即, 令 $x_{t+1}=x_t+\lambda_t\mathbf{d}_t$, 其中 λ_t 是一维搜索得到的步长因子, $\lambda_t=\arg\min g(\mathbf{x}_t+\lambda\mathbf{d}_t)$, 这样修改的算法称为阻尼牛顿法 (damp Newton method). 当取 $\lambda_t=1$, 即是牛顿法. 通常情况下, 阻尼牛顿法每次迭代比原始的牛顿法下降得更多, 所以通常收敛会更快. 当然, 有时根据计算量的考虑, 我们也不一定必须求得上述一维的优化的最优值, 只需要保证下降即可. 这时, 可使用类似前一节中的进退算法找到一个适当的值即可.

当函数 g 是非二次的, Hessian 矩阵很可能是非正定的. 这时, 我们可能无法从 $\mathbf{H}_t\mathbf{d}=-\nabla g_t$ 中解出 $\mathbf{d}_t$, 或者即便能求出来, 也不能保证 $\mathbf{d}_t$ 是一个下降方向. 因此, 为了在梯度不为零的点 $\mathbf{x}_t$ 找到一个下降方向, 我们可以强迫 $\mathbf{H}_t\mathbf{d}=-\nabla g_t$ 中的 $\mathbf{H}_t$ 恒取正定阵, 这样总能得到下降方向. 其中一种常用的方法是采用 $\mathbf{G}_t=\mathbf{H}_t+\mathbf{E}_t$, 其中 $\mathbf{E}_t$ 为一个对角阵. 这样的一个矩阵可通过强迫矩阵的Cholesky分解而得到.

同单变量情形一样, 在MLE问题中, 我们可以用在 $\boldsymbol{\theta}_t$ 点的期望Fisher信息量

$$\mathbf{I}(\boldsymbol{\theta}_t)=-E\left[\frac{\partial^2\mathbf{l}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\right]_{|\boldsymbol{\theta}_t}$$

替代在点 $\boldsymbol{\theta}_t$ 处的观测的信息量, 则此时得到所谓的多元Fisher得分法, 其更新方程为

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t+\mathbf{I}^{-1}(\boldsymbol{\theta}_t)\mathbf{l}'(\boldsymbol{\theta}_t), \tag{4.21}$$

此方法渐近等价于 Newton 法.

例 4.3.2. (Logistic 回归模型的极大似然估计)

logistic model 是一著名的广义线性模型, 现考虑其参数的MLE. 在广义线性模型中, 响应变量 y_i 独立来自某参数为 θ_i 的分布 $(i = 1, 2, \ldots, n)$. 虽然不同类型的响应用不同的分布来拟合, 但其分布通常均属于某指数分布族, 此分布族的形式为 $f(y|\theta) = \exp\{[y\theta - b(\theta)]/a(\phi) + c(y, \phi)\}$, 其中 θ 为自然或典则参数, 而 ϕ 为散度参数. 此分布族的两个最有用的性质为: $E\{Y\} = b'(\theta)$ 和 $\mathrm{var}\{Y\} = b''(\theta)a(\phi)$.

广义线性模型是对线性模型进行了两个方面的推广: 一是通过一个连接函数, 将响应变量的期望与线性自变量相联系; 二是对误差的分布给出一个误差函数. 这样的推广使许多线性模型的方法能够用于一个一般的问题, 尤其是在离散响应变量的情形下.

具体地, 假设我们有数据 $\{y_i, \mathbf{x}_i\}_{i=1}^n$, 其中 y_i 和 $\mathbf{x}_i \in \mathbb{R}^{p+1}$ 分别是响应变量和自变量. $E\{y_i|\mathbf{x}_i\}$ 由方程 $f(E\{y_i|\mathbf{x}_i\}) = \mathbf{x}_i^T\boldsymbol{\beta}$ 与 $\mathbf{x}_i$ 相关联, 其中 $\boldsymbol{\beta}$ 为参数向量, 称 f 为*连接函数*. 记列向量 $\mathbf{x}_i = (1, x_{i1}, \ldots, x_{ip})^T$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$. 这方面的详细内容可参见线性模型的诸多教材, 在这里不予赘述, 下面仅以 logistic 模型为例来说明其求解过程.

对于 logistic 回归的广义线性模型, 它是由属于指数分布族的 Binomial 分布而得到的. 此时响应的分布为 $y_i|\mathbf{x}_i \sim \mathrm{BIN}(\pi_i, m_i)$, $i = 1, \ldots, n$, 且相互独立. 则对于第 i 个观测,

$$f(\pi_i) = \log\{\pi_i/(1 - \pi_i)\} = \mathbf{x}_i^T\boldsymbol{\beta} \equiv \alpha_i,$$

$$y_i \sim \mathrm{BIN}(\pi_i, m_i).$$

$y_1, \ldots, y_n$ 的联合似然函数可表示为

$$L(\boldsymbol{\pi}; \mathbf{y}) = \prod_{i=1}^n C_{m_i}^{y_i} \pi_i^{y_i} (1 - \pi_i)^{m_i - y_i},$$

其中 $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)^T$, $\mathbf{y} = (y_1, \ldots, y_n)^T$. 将 $\pi_i = \frac{e^{\alpha_i}}{1+e^{\alpha_i}}$ 代入上式, 取对数并将与参数 $\boldsymbol{\beta}$ 无关项舍去, 我们得到如下的联合对数似然函数

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{i=1}^n \left[y_i \log \frac{e^{\alpha_i}}{1 + e^{\alpha_i}} + (m_i - y_i) \log \frac{1}{1 + e^{\alpha_i}} \right] \\ &= \sum_{i=1}^n [y_i\alpha_i - m_i \log(1 + e^{\alpha_i})] \\ &= \mathbf{y}^T\mathbf{X}\boldsymbol{\beta} - \sum_{i=1}^n m_i \log(1 + \exp\{\mathbf{x}_i^T\boldsymbol{\beta}\}), \end{aligned}$$

其中 $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$ 是 $n \times (p + 1)$ 矩阵.

现考虑利用 Newton 法求最大化此似然的 $\boldsymbol{\beta}$, 此时的得分函数为

$$\boldsymbol{l}'(\boldsymbol{\beta}) = \mathbf{X}^T\mathbf{y} - \sum_{i=1}^n m_i\pi_i\mathbf{x}_i = \mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}), \tag{4.22}$$

其中 $\boldsymbol{\mu} = E(\mathbf{y}) = (m_1\pi_1, \ldots, m_n\pi_n)^T$. 其 Hessian 矩阵为

$$\boldsymbol{l}''(\boldsymbol{\beta}) = \frac{\partial}{\partial \boldsymbol{\beta}}(\mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu})) = -\left(\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\beta}}\right)^T \mathbf{X} = -\mathbf{X}^T\mathbf{W}\mathbf{X}, \tag{4.23}$$

其中 $\mathbf{W}$ 为第 i 个对角元等于 $m_i\pi_i(1-\pi_i)$ 的对角阵.

于是, Newton法的更新方程为

$$\begin{aligned} \boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t - \boldsymbol{l}''(\boldsymbol{\beta}_t)^{-1}\boldsymbol{l}'(\boldsymbol{\beta}_t) && (4.24)\\ &= \boldsymbol{\beta}_t + \left(\mathbf{X}^T\mathbf{W}_t\mathbf{X}\right)^{-1}\left(\mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_t)\right), && (4.25) \end{aligned}$$

其中 $\boldsymbol{\mu}_t$ 为对应着 $\boldsymbol{\beta}_t$ 的 $\boldsymbol{\mu}$ 的值, $\mathbf{W}_t$ 为在 $\boldsymbol{\mu}_t$ 处取值的对角权重阵.

注意到 Hessian 阵 $l''(\boldsymbol{\beta}_t)$ 不依赖于 $\mathbf{y}$. 于是, Fisher信息阵等于观测的信息量, 即 $\mathbf{I}(\boldsymbol{\beta}) = E\{-\boldsymbol{l}''(\boldsymbol{\beta})\} = E\{\mathbf{X}^T\mathbf{W}\mathbf{X}\} = -\boldsymbol{l}''(\boldsymbol{\beta})$. 因此, 对于本例, Fisher 得分法等同于 Newton 法.

利用 Fisher 得分法来求广义线性模型的极大似然估计亦可看作是 **迭代加权最小二乘**(iterative weighted least quare; IWLS) 方法的应用. 记

$$\mathbf{e}_t = \boldsymbol{y} - \boldsymbol{\mu}_t,$$

$$\mathbf{z}_t = \mathbf{X}\boldsymbol{\beta}_t + \mathbf{W}_t^{-1}\mathbf{e}_t.$$

则 Fisher 得分法的更新方程可以写成

$$\begin{aligned} \boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t + \left(\mathbf{X}^T\mathbf{W}_t\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{e}_t \\ &= \left(\mathbf{X}^T\mathbf{W}_t\mathbf{X}\right)^{-1}\left[\mathbf{X}^T\mathbf{W}_t\mathbf{X}\boldsymbol{\beta}_t + \mathbf{X}^T\mathbf{W}_t\mathbf{W}_t^{-1}\mathbf{e}_t\right] \\ &= \left(\mathbf{X}^T\mathbf{W}_t\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}_t\mathbf{z}_t. && (4.26) \end{aligned}$$

从(4.26)可以看出, 由于 $\boldsymbol{\beta}_{t+1}$ 是 $\mathbf{z}_t$ 关于 $\mathbf{X}$ 的加权最小二乘的回归系数, 且其权重为 $\mathbf{W}_t$ (我们通常称其为工作权矩阵working weighted matrix) 的对角元, 而称 $\mathbf{z}_t$ 为**工作响应**(working response). 在每一步迭代, 都要重新计算一个新的工作响应和权向量, 且更新方程可由一个加权最小二乘拟合得到. 对于广义线性模型, IWLS 是下将介绍的处理非线性最小二乘问题的 Gauss-Newton 法的一种特殊情况.

§4.3.4 拟牛顿法 (Quasi Newton)

回顾下降法, 它们都由如下的方程系统来决定前进的方向

$$\mathbf{R}\mathbf{d} = -\nabla g(\mathbf{x}),$$

不同的方法区别在于采用了不同的 $\mathbf{R}$. 最速下降法使用 $\mathbf{R} = \mathbf{I}$, 而牛顿法使用 $\mathbf{R} = \mathbf{H}_g$. 前者简单, 但由于"锯齿" 效应导致收敛速度较慢; 后者收敛快, 但每次迭代都需要计算 $\mathbf{H}_g(\mathbf{x}_t)$ 并需要解线性系统, 运算量较大.

我们希望设计一种新方法, 尽可能保持牛顿法收敛快的优点, 但又不必计算 Hessian 阵及逆阵. 本节介绍的变尺度法 (variable metric), 其找到一个适当的 $\mathbf{R}$, 能达到上述目的. 由于其亦可看作是牛顿法的推广, 故也称之为拟牛顿法. 下面介绍其基本思想. 考虑基于 (4.20) 的迭代公式

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \lambda_t \mathbf{G}_t \nabla g_t.$$

为使 $\mathbf{G}_t$ 确实与 $\mathbf{H}_t^{-1}$ 近似且具有容易计算的特点, 要求 $\mathbf{G}_t$ 应满足一些条件.

- 为保证搜索方向是下降方向, 要求每次迭代的矩阵 $\mathbf{G}_t$ 都是正定的.
- 要求 $\mathbf{G}_t, t = 0, 1, \ldots$ 之间的迭代具有简单形式, 显然最简单的形式为

$$\mathbf{G}_{t+1} = \mathbf{G}_t + \mathbf{E}_t.$$

- 由于

$$\nabla g_{t+1} - \nabla g_t \approx \mathbf{H}_{t+1}(\mathbf{x}_{t+1} - \mathbf{x}_t), \tag{4.27}$$

所以我们自然也要求

$$\mathbf{G}_{t+1} \triangle g_t' = \triangle \mathbf{x}_t,$$

其中我们定义 $\triangle g_t' = \nabla g_{t+1} - \nabla g_t$, $\triangle \mathbf{x}_t = \mathbf{x}_{t+1} - \mathbf{x}_t$. 注意到这个条件有些类似于我们正割法中的基于差分的正割公式, 我们一般称此条件为拟牛顿条件.

根据 $\mathbf{G}_t$ 应满足的基本条件, 特别是拟牛顿条件, 可以构造出 $\mathbf{G}_t$ 的各种各样的方法, 从而形成各种变尺度算法. 下面介绍最具代表性的 DFP 算法中 $\mathbf{G}_t$ 的构造方法. 要求 $\mathbf{G}_{t+1} = \mathbf{G}_t + \mathbf{E}_t$ 满足拟牛顿条件, 得

$$(\mathbf{G}_t + \mathbf{E}_t) \triangle g_t' = \triangle \mathbf{x}_t, \tag{4.28}$$

因而有

$$\mathbf{E}_t \triangle g_t' = \triangle \mathbf{x}_t - \mathbf{G}_t \triangle g_t'.$$

若能选取向量 $\mathbf{u}_t$ 和 $\mathbf{v}_t$ 使其满足规范化条件

$$\mathbf{u}_t^T \triangle g_t' = \mathbf{v}_t^T \triangle g_t' = 1,$$

则满足 (4.28) 式的 $\mathbf{E}_t$ 可以简单地取以下形式:

$$\mathbf{E}_t = \triangle \mathbf{x}_t \mathbf{v}_t^T - \mathbf{G}_t \triangle g_t' \mathbf{u}_t^T,$$

故我们有

$$\mathbf{G}_{t+1} = \mathbf{G}_t + \triangle \mathbf{x}_t \mathbf{v}_t^T - \mathbf{G}_t \triangle g_t' \mathbf{u}_t^T.$$

一种简单的取 $\mathbf{u}_t$ 和 $\mathbf{v}_t$ 的方法是

$$\mathbf{u}_t^T = \frac{(\triangle g_t')^T \mathbf{G}_t}{(\triangle g_t')^T \mathbf{G}_t \triangle g_t'},\ \mathbf{v}_t = \frac{\triangle \mathbf{x}_t^T}{\triangle \mathbf{x}_t^T \triangle g_t'},$$

代入前式我们得到最终的 DFP 算法的关于 $\mathbf{G}_t$ 迭代公式

$$\mathbf{G}_{t+1} = \mathbf{G}_t + \frac{\triangle \mathbf{x}_t \triangle \mathbf{x}_t^T}{\triangle \mathbf{x}_t^T \triangle g_t'} - \frac{\mathbf{G}_t \triangle g_t' (\triangle g_t')^T \mathbf{G}_t}{(\triangle g_t')^T \mathbf{G}_t \triangle g_t'}.$$

算法 4.3.6. DFP拟牛顿法:

(0) 令 $t = 0$, 给定一个初值 $\mathbf{x}_0$; 设定三个精度 ϵ_1, ϵ_2, 和 ϵ_3;

(1) 计算 $g(\mathbf{x}_0)$ 和梯度 ∇g_0, 取 $\mathbf{G}_0 = \mathbf{I}$, $\mathbf{d}_0 = -\nabla g_0$;

(2) 作直线搜索求得 $\lambda_t = \arg\min g(\mathbf{x}_t + \lambda \mathbf{d}_t)$, 令 $\mathbf{x}_{t+1} = \mathbf{x}_t + \lambda_t \mathbf{d}_t$;

(3) 计算 $g(\mathbf{x}_{t+1})$ 和 ∇g_{t+1}. 若 $D(\nabla g_{t+1}, \mathbf{0}) < \epsilon_1$, 或 $|g(\mathbf{x}_{t+1}) - g(\mathbf{x}_t)| < \epsilon_2$, 或 $D(\mathbf{x}_{t+1}, \mathbf{x}_t) < \epsilon_3$, 则停止迭代, 返回 $\mathbf{x}^* = \mathbf{x}_{t+1}$; 否则继续执行 (4).

(4) 判别 $g(\mathbf{x}_{t+1}) \geq g(\mathbf{x}_t)$ 是否成立, 如果成立令 $\mathbf{x}_0 = \mathbf{x}_t$, 转 (1); 否则继续执行 (5).

(5) 计算 $\triangle \mathbf{x}_t$, $\triangle g_t'$, 及矩阵

$$\mathbf{G}_{t+1} = \mathbf{G}_t + \frac{\triangle \mathbf{x}_t \triangle \mathbf{x}_t^T}{\triangle \mathbf{x}_t^T \triangle g_t'} - \frac{\mathbf{G}_t \triangle g_t' (\triangle g_t')^T \mathbf{G}_t}{(\triangle g_t')^T \mathbf{G}_t \triangle g_t'}.$$

令 $\mathbf{d}_{t+1} = -\mathbf{G}_{t+1} \nabla g_{t+1}$, 并置 $t = t + 1$, 转 (2) 继续迭代.

在 DFP 算法中, 可以证明, 如果给定的初始矩阵 $\mathbf{G}_0$ 是正定阵, 则迭代过程中每个 $\mathbf{G}_t$ 都是正定的. 保证 $\mathbf{G}_t$ 正定, 也就保证了算法是下降算法. 但在实际数值计算中, 由于舍入误差的影响, 特别是直线搜索不精确的影响, 有可能会破坏了迭代矩阵 $\mathbf{G}_t$ 的正定性. 所以我们在算法 4.3.6 中采取了措施 (4).

一般地, 拟 Newton 法的收敛阶数比线性高, 但比二次低. 相对于Newton法而言, 其收敛阶数低于二次的原因是对 Hessian 阵的近似. 不过, 拟 Newton 法仍是快速且有效的, 而且在许多软件包中都采用该种方法. 拟Newton法的表现对初始矩阵 $\mathbf{G}_0$ 的选取较为敏感, 最直接的选择就是单位阵, 但当 $\mathbf{x}_t$ 各分量的尺度差异很大时, 此种选择经常不合适. 对于MLE, 如果期望的 Fisher 信息量可以计算, 则取 $\mathbf{G}_0 = \mathbf{I}(\boldsymbol{\theta}_0)$ 通常是一个较好的选择. 在一般情况下, 对于拟Newton法, 重新调整 $\mathbf{x}$ 各元素的刻度使其具有可比性是非常重要的. 这种调整将改进其表现并有效预防其停止准则仅依赖于那些刻度大的变量. 在刻度调整不好的多数问题中, 人们可能会发现对于拟 Newton 法的收敛点, 仅有部分分量 $x_t^{(i)}$ 与其相应的初值有别, 而其余分量均不变.

§4.3.5 Gauss-Newton 法: 计算最小二乘估计

对于某个非线性函数 f 和随机误差 ϵ_i, 我们有如下的非线性回归模型

$$y_i = f(\mathbf{x}_i, \boldsymbol{\theta}) + \epsilon_i. \tag{4.29}$$

非线性回归就是通过 n 次观测数据来估计函数 $f(\mathbf{x}_i, \boldsymbol{\theta})$ 中的未知参数 $\boldsymbol{\theta}$, 进而利用该非线性关系和估计的参数值来预测给定某个 $\mathbf{x}$ 的 y 值. 显然, 当 f 是特殊的线性关系时 $\mathbf{x}_i^T\boldsymbol{\theta}$, (4.29) 就变为线性回归模型.

在线性模型中, 我们常采用最小二乘准则来估计参数. 其实际上就是使误差平方和 $\sum_{i=1}^n \epsilon_i^2$ 在估计处达到最小值. 因而该方法完全可以推广应用到非线性回归的情况. 当我们采用最小二乘准则来估计模型 (4.29) 的参数时, 即得到如下的最优化问题

$$\widehat{\theta}_{\mathrm{LSE}} = \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}) \equiv \sum_{i=1}^n (y_i - f(\mathbf{x}_i, \boldsymbol{\theta}))^2,$$

其中 $\{y_i, \mathbf{x}_i\},\ i = 1, \ldots, n$ 为观测数据.

显然, 该优化问题就是一个无约束的非线性优化问题, 前述的各种优化方法都可用于该问题的求解, 尤其是牛顿法. 不过由于该目标函数形式比较特殊, 我们可开发出较为有效的算法来替代牛顿法从而避免 Hessian 矩阵. 我们下面讨论著名的 Gauss-Newton 算法, 其是非线性回归分析中应用十分广泛的算法.

记目标函数

$$g(\boldsymbol{\theta}) = \sum_{i=1}^n (y_i - f(\mathbf{x}_i, \boldsymbol{\theta}))^2 \equiv \mathbf{e}^T(\boldsymbol{\theta})\mathbf{e}(\boldsymbol{\theta}).$$

注意到最小二乘问题的梯度向量和 Hessian 矩阵均与残差向量 $\mathbf{e}(\boldsymbol{\theta}) = (e_1(\boldsymbol{\theta}), \ldots, e_n(\boldsymbol{\theta}))^T$ 的 Jacobian 矩阵, $\mathbf{J}_e(\boldsymbol{\theta})$ 有关,

$$\nabla g(\boldsymbol{\theta}) = 2(\mathbf{J}_e(\boldsymbol{\theta}))^T\mathbf{e}(\boldsymbol{\theta}),$$

$$\mathbf{H}_g(\boldsymbol{\theta}) = 2(\mathbf{J}_e(\boldsymbol{\theta}))^T\mathbf{J}_e(\boldsymbol{\theta}) + 2\sum_{i=1}^n e_i(\boldsymbol{\theta})\mathbf{H}_{e_i}(\boldsymbol{\theta}).$$

注意到 Hessian矩阵 $\mathbf{H}_g$ 较为复杂, 造成实际使用中的困难. 为简化计算, 我们将误差向量 $\mathbf{e}(\boldsymbol{\theta})$ 在点 $\boldsymbol{\theta}_t$ 泰勒展开:

$$\mathbf{e}(\boldsymbol{\theta}) \approx \mathbf{e}(\boldsymbol{\theta}_t) + \mathbf{J}_e(\boldsymbol{\theta}_t)(\boldsymbol{\theta} - \boldsymbol{\theta}_t).$$

由上式可得到目标函数 $g(\boldsymbol{\theta})$ 在 $\boldsymbol{\theta}_t$ 附近的近似表达式及 $g(\boldsymbol{\theta})$ 的梯度向量和 Hessian 矩阵的近似表达式:

$$g(\boldsymbol{\theta}) \approx [\mathbf{e}(\boldsymbol{\theta}_t) + \mathbf{J}_e(\boldsymbol{\theta}_t)(\boldsymbol{\theta} - \boldsymbol{\theta}_t)]^T[\mathbf{e}(\boldsymbol{\theta}_t) + \mathbf{J}_e(\boldsymbol{\theta}_t)(\boldsymbol{\theta} - \boldsymbol{\theta}_t)]$$

$$\nabla g(\boldsymbol{\theta}) \approx 2(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{e}(\boldsymbol{\theta}_t) + 2(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{J}_e(\boldsymbol{\theta}_t)(\boldsymbol{\theta} - \boldsymbol{\theta}_t)$$

$$\mathbf{H}_g(\boldsymbol{\theta}) \approx 2(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{J}_e(\boldsymbol{\theta}_t)$$

上式中 $\mathbf{H}_g(\boldsymbol{\theta})$ 里没有包含 $\mathbf{e}(\boldsymbol{\theta})$ 的二阶导数, 使用这种近似的 $\nabla g(\boldsymbol{\theta})$ 和 $\mathbf{H}_g(\boldsymbol{\theta})$ 的牛顿迭代算法称为*Gauss-Newton 法*. 而如果采用带惩罚因子的阻尼牛顿法, 则为通常软件包中所广泛采用的*修正的 Gauss-Newton 法*, 其迭代公式为

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \lambda_t[(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{J}_e(\boldsymbol{\theta}_t)]^{-1}(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{e}(\boldsymbol{\theta}_t).$$

这等价于 $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \lambda_t\mathbf{d}_t$, 其中 $\mathbf{d}_t$ 是方程组

$$(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{J}_e(\boldsymbol{\theta}_t)\mathbf{d}_t = -(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{e}(\boldsymbol{\theta}_t) \tag{4.30}$$

的解. 可以证明, 当 $(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{J}_e(\boldsymbol{\theta}_t)$ 是可逆时, $\mathbf{d}_t$ 是目标函数的下降方向. 也就是说, 如果 $\nabla g(\boldsymbol{\theta}_t) \neq 0$, 则

$$\mathbf{d}_t^T\nabla g(\boldsymbol{\theta}_t) = 2\mathbf{d}_t^T\mathbf{J}_t^T\mathbf{e}_t = -\mathbf{d}_t^T\mathbf{J}_t^T\mathbf{J}_t\mathbf{d}_t < 0.$$

故而该算法是有效的. 但当 $(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{J}_e(\boldsymbol{\theta}_t)$ 奇异或接近奇异时, 我们无法解出 $\mathbf{d}_t$, 这时, 可将 $g(\boldsymbol{\theta})$ 在 $\boldsymbol{\theta}_t$ 处的负梯度方向作为搜索方向. 总结算法如下:

算法 4.3.7. 修正的 Gauss-Newton 算法:

(0) 令 $t = 0$, 给定一个初值 $\boldsymbol{\theta}_0$, 计算 $g(\boldsymbol{\theta}_0)$;

(1) 计算残差向量 $\mathbf{e}_t \equiv \mathbf{e}(\boldsymbol{\theta}_t)$ 和 Jacobian 矩阵 $\mathbf{J}_t \equiv \mathbf{J}_e(\boldsymbol{\theta}_t) = \left(\frac{\partial \mathbf{e}(\boldsymbol{\theta}_t)}{\partial \theta_j}\right)_{n\times m}$;

(2) 解线性方程组 $\mathbf{J}_t^T\mathbf{J}_t\mathbf{d}_t = -\mathbf{J}_t^T\mathbf{e}_t$; 当 $\operatorname{rank}(\mathbf{J}_t^T\mathbf{J}_t) = m$ 时, 方程组有唯一解: $\mathbf{d}_t = -(\mathbf{J}_t^T\mathbf{J}_t)^{-1}\mathbf{J}_t^T\mathbf{e}_t$; 否则, 取 $\mathbf{d}_t = -\mathbf{J}_t^T\mathbf{e}_t$.

(3) 作直线搜索求得 $\lambda_t = \arg\min g(\boldsymbol{\theta}_t + \lambda\mathbf{d}_t)$, 令 $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \lambda_t\mathbf{d}_t$;

(4) 计算 $g(\boldsymbol{\theta}_{t+1})$. 若 $|g(\boldsymbol{\theta}_{t+1}) - g(\boldsymbol{\theta}_t)| < \epsilon_1$, 或 $D(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) < \epsilon_2$, 则停止迭代, 返回 $\widehat{\boldsymbol{\theta}}_{\mathrm{LSE}} = \boldsymbol{\theta}_{t+1}$; 否则转 (1) 继续迭代.

Gauss-Newton 法亦可看作是一种 IWLS. 其不去近似 g, 而是用 f 在点 $\boldsymbol{\theta}_t$ 的线性 Taylor 展开近似 f 本身. 由此线性近似替换 f 就成了一个线性最小二乘问题, 而解此问题就得到一个更新 $\boldsymbol{\theta}_{t+1}$. 特别地, 非线性模型(4.29) 可被近似为

$$y_i \approx f(\mathbf{x}_i, \boldsymbol{\theta}_t) + (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^T\nabla f_t(\mathbf{x}_i) + \epsilon_i,$$

以 z_{ti} 记取值为 $z_{ti} = y_i - f(\mathbf{x}_i, \boldsymbol{\theta}_t)$ 的工作残差, 且定义 $\mathbf{a}_{ti} = \nabla f_t(\mathbf{x}_i)$, 则近似问题可被描述成最小化下面线性回归模型

$$\mathbf{z}_t = \mathbf{A}_t(\boldsymbol{\theta} - \boldsymbol{\theta}_t) + \boldsymbol{\epsilon}, \tag{4.31}$$

的平方残差, 其中 $\mathbf{z}_t$, $\boldsymbol{\epsilon}$ 分别是第 i 个分量为 z_{ti}, ϵ_i 的列向量, 类似地, $\mathbf{A}_t$ 是第 i 行为 $(\mathbf{a}_{ti})^T$的矩阵. 显然, 当取最小二乘估计

$$(\boldsymbol{\theta} - \boldsymbol{\theta}_t) = \left(\mathbf{A}_t^T\mathbf{A}_t\right)^{-1}\mathbf{A}_t^T\mathbf{z}_t \tag{4.32}$$

时, 拟合(4.31)式的均方误差达到最小. 于是, 关于 $\boldsymbol{\theta}_t$ 的 Gauss-Newton 法的更新为

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \left(\mathbf{A}_t^T\mathbf{A}_t\right)^{-1}\mathbf{A}_t^T\mathbf{z}_t \\ &= \left(\mathbf{A}_t^T\mathbf{A}_t\right)^{-1}\mathbf{A}_t^T(\mathbf{A}_t\boldsymbol{\theta}_t + \mathbf{z}_t).\end{aligned}$$

由此可看出, $\boldsymbol{\theta}_{t+1}$ 是工作相应 $\mathbf{A}_t\boldsymbol{\theta}_t + \mathbf{z}_t$ 关于工作设计阵 $\mathbf{A}_t$ 的最小二乘的回归系数. 在每一步迭代, 都要重新计算一个新的工作响应和工作设计阵, 且更新方程可由一个最小二乘拟合得到.

相对于 Newton 法, Gauss-Newton 法的优点在于它不需要计算 Hessain 阵. 当 f 接近线性或模型拟合较好时, Guass-Newton 法的收敛速度很快, 但在其它一些情况, 特别是由于模型拟合不好而当残差很大时, 此方法收敛可能很慢或根本就不收敛 (即使初值很好). 对于这些情况, 现有多种改进的具有良好收敛性质的 Gauss-Newton 法, 其中最著名的就是常用的 Marquard 算法.

我们把矩阵 $\mathbf{J}_t^T\mathbf{J}_t$ 对角线上的元素都加上同一个正整数 μ, 则方程组 (4.30) 变成

$$(\mathbf{J}_t^T\mathbf{J}_t + \mu\mathbf{I})\mathbf{d} = -\mathbf{J}_t^T\mathbf{e}_t \tag{4.33}$$

由于 $\mathbf{J}_t^T\mathbf{J}_t$ 是非负定的, 则 $\mathbf{J}_t^T\mathbf{J}_t + \mu\mathbf{I}$ 一定是正定阵. 因此 (4.33) 一定有解. 这个解依赖于 μ, 我们记为 $\mathbf{d}_t(\mu)$. 特别地, 当 $\mu = 0$ 时, (4.33) 就是 (4.30) 式. 并且当 $\mathbf{J}_t^T\mathbf{J}_t$ 非奇异时, $\mathbf{d}_t(0)$ 存在, 即 Gauss-Newton 方向; $\mathbf{J}_t^T\mathbf{J}_t$ 奇异时, $\mathbf{d}_t(0)$ 不存在. 这时考虑 $\mu > 0$, 当 μ 值增大时, 假定 μ 已经大到与 $\mathbf{J}_t^T\mathbf{J}_t$ 的每个分量相比这些分量都可以会略不计时, 则 (4.33) 式可写为

$$\mu\mathbf{d}_t(\mu) \approx -\mathbf{J}_t^T\mathbf{e}_t, \quad \text{或} \quad \mathbf{d}_t(\mu) \approx -\frac{1}{\mu}\mathbf{J}_t^T\mathbf{e}_t.$$

这就是说, 当 μ 很大时, $\mathbf{d}_t(\mu)$ 将接近目标函数 $g(\boldsymbol{\theta})$ 在 $\boldsymbol{\theta}$ 处的负梯度方向. 可以想象, 当 μ 从零增加到无穷大时, $\mathbf{d}_t(\mu)$ 将从 $g(\boldsymbol{\theta})$ 在 $\boldsymbol{\theta}$ 处的 Gauss-Newton 方向连续地转向负梯度方向. 由以上分析, 我们可以构造如下迭代公式:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - (\mathbf{J}_t^T\mathbf{J}_t + \mu_t\mathbf{I})^{-1}\mathbf{J}_t^T\mathbf{e}_t$$

这就是 Marquard 算法的迭代公式, 这里的 μ_t 也常被称为阻尼因子, 因为当 $\mu \to \infty$ 时, $||\mathbf{d}_t(\mu_t)|| \to 0$ (也就是控制了 $\mathbf{d}_t(\mu_t)$ 的模长大小). 为使 $\mathbf{J}_t^T\mathbf{J}_t + \mu_t\mathbf{I}$ 正定, μ_t 需取得较大, 但我们看出这时步长 $||\mathbf{d}_t(\mu_t)||$ 越短, 前进速度越慢, 因此 μ_t 也不能选得太大. 这样每次迭代中就存在如何适当选取阻尼因子 μ_t 的问题. 不同的选取方法对应不同的算法, 我们下面给出的是最简单常用的一种方法, Levenberg-Marquard 算法, 其一边迭代一边调整阻尼因子的方法. 记 $\mathbf{A}_t = \mathbf{J}_t^T\mathbf{J}_t = (a_t^{(i,j)})_{m\times m}$.

算法 4.3.8. Levenberg-Marquard 算法:

(0) 令 $t = 0$, 给定一个初值 $\boldsymbol{\theta}_0$; 初始阻尼因子 $\mu_0 = a_0^{(1,1)}c_0$ (如取 $c_0 = 10^{-2}$ 或 10^{-3}) 及整数 m (如 $m = 2, 4$ 或 10); 计算 $g(\boldsymbol{\theta}_0)$;

(1) 计算残差向量 $\mathbf{e}_t \equiv \mathbf{e}(\boldsymbol{\theta}_t)$ 和 Jacobian 矩阵 $\mathbf{J}_t \equiv \mathbf{J}_e(\boldsymbol{\theta}_t) = \left(\frac{\partial \mathbf{e}(\boldsymbol{\theta}_t)}{\partial \theta_j}\right)_{n\times m}$, 以及梯度向量 $\mathbf{b}_t = -\mathbf{J}_t^T\mathbf{e}_t$;

(2) 解线性方程组

$$\left(\mathbf{J}_t^T\mathbf{J}_t + \mu_t a_t^{(1,1)}\mathbf{I}\right)\mathbf{d} = -\mathbf{b}_t,$$

得搜索方向 $\mathbf{d}_t$, 令 $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{d}_t$, 并计算 $g(\boldsymbol{\theta}_{t+1})$;

(3) 若 $|g(\boldsymbol{\theta}_{t+1}) - g(\boldsymbol{\theta}_t)| < \epsilon_1$, 或 $D(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) < \epsilon_2$, 则停止迭代, 返回 $\widehat{\boldsymbol{\theta}}_{\mathrm{LSE}} = \boldsymbol{\theta}_{t+1}$; 否则转 (4);

(4) 若 $g(\boldsymbol{\theta}_{t+1}) < g(\boldsymbol{\theta}_t)$, 则置 $\tilde{\mu}_t = \mu_t/m$ (即缩小阻尼因子), 转 (5); 否则转 (7);

(5) 解线性方程组 $(\mathbf{J}_t^T\mathbf{J}_t + \tilde{\mu}_t a_t^{(1,1)}\mathbf{I})\mathbf{d} = -\mathbf{b}_t$, 得搜索方向 $\tilde{\mathbf{d}}_t$, 并计算 $g(\tilde{\boldsymbol{\theta}}_{t+1})$

(6) 若 $g(\tilde{\boldsymbol{\theta}}_{t+1}) < g(\boldsymbol{\theta}_t)$, 则置 $\mu_{t+1} = \tilde{\mu}_t$, $\boldsymbol{\theta}_{t+1} = \tilde{\boldsymbol{\theta}}_{t+1}$; 否则置 $\mu_{t+1} = \mu_t$. 令 $t = t+1$, 转 (1);

(7) 若 $g(\boldsymbol{\theta}_{t+1}) \geq g(\boldsymbol{\theta}_t)$, 对 $i = 1, 2, \ldots,$, 计算 $\mu_t^{(i)} = m^i\mu_t$ (即增大阻尼因子), 并解线性方程组

$$(\mathbf{J}_t^T\mathbf{J}_t + \mu_t^{(i)} a_t^{(1,1)}\mathbf{I})\mathbf{d} = -\mathbf{b}_t,$$

设解为 $\mathbf{d}_t^{(i)}$, 令 $\boldsymbol{\theta}_{t+1}^{(i)} = \boldsymbol{\theta}_t + \mathbf{d}_t^{(i)}$, 并计算 $g(\boldsymbol{\theta}_{t+1}^{(i)})$, 直到 $g(\tilde{\boldsymbol{\theta}}_{t+1}^{(l)}) < g(\boldsymbol{\theta}_t)$ 成立时为止. 这时令 $\mu_{t+1} = \mu_t^{(l)}$, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_{t+1}^{(l)}$, 且 $t = t+1$, 转 (1);

在上面的算法中, 对于取定的阻尼因子 μ_t, 令 $\mu_t a_t^{(11)}$ 作为实际的阻尼因子. 这样做是考虑到不同的非线性函数和观测数据, 矩阵 $\mathbf{J}_t^T\mathbf{J}_t$ 中元素的数量级别可能有差别. 当然也可以用 $\mu_t a_t^{(ii)}, i = 1, 2, \ldots$. 或者也可以进一步地将迭代公式修改为

$$(\mathbf{J}_t^T\mathbf{J}_t + \mu_t \mathrm{diag}(\mathbf{J}_t^T\mathbf{J}_t))\mathbf{d} = -\mathbf{b}_t,$$

即对角线元素加上不同的阻尼因子, 该方法在实际操作中也相当有效.

§4.3.6 迭代重加权最小二乘和最小L-P范回归

考虑如下的最小L-P范回归问题：对于$0 < p < 2$,

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} |y_i - f(\mathbf{x}_i, \boldsymbol{\theta})|^p,$$

其中 $f(\mathbf{x}_i, \boldsymbol{\theta})$ 是某个线性或非线性函数. 处理这样的最小化问题的最常用的方法就是迭代重

加权最小二乘（iteratively reweighted least squares; IRLS）:

$$\begin{aligned}\sum_{i=1}^{n}|y_i-f(\mathbf{x}_i,\boldsymbol{\theta})|^p &= \sum_{i=1}^{n}(y_i-f(\mathbf{x}_i,\boldsymbol{\theta}))^2\times|y_i-f(\mathbf{x}_i,\boldsymbol{\theta})|^{p-2}\\ &\equiv \sum_{i=1}^{n}\omega_i(\boldsymbol{\theta})(y_i-f(\mathbf{x}_i,\boldsymbol{\theta}))^2,\\ &\approx \sum_{i=1}^{n}\omega_i(\boldsymbol{\theta}_t)(y_i-f(\mathbf{x}_i,\boldsymbol{\theta}))^2.\end{aligned}$$

再结合前面所介绍的Gauss-Newton计算非线性最小二乘的方法, 我们则有

$$g(\boldsymbol{\theta})\equiv\sum_{i=1}^{n}|y_i-f(\mathbf{x}_i,\boldsymbol{\theta})|^p\approx\mathbf{e}^T(\boldsymbol{\theta})\mathbf{W}(\boldsymbol{\theta}_t)\mathbf{e}(\boldsymbol{\theta}),$$

其中 $\mathbf{W}(\boldsymbol{\theta}_t)=\mathrm{diag}\{\omega_1(\boldsymbol{\theta}_t),\ldots,\omega_n(\boldsymbol{\theta}_t)\}$. 仍将误差向量 $\mathbf{e}(\boldsymbol{\theta})$ 在点 $\boldsymbol{\theta}_t$ 泰勒展开:

$$\mathbf{e}(\boldsymbol{\theta})\approx\mathbf{e}(\boldsymbol{\theta}_t)+\mathbf{J}_e(\boldsymbol{\theta}_t)(\boldsymbol{\theta}-\boldsymbol{\theta}_t).$$

由上式可得到 $g(\boldsymbol{\theta})$ 的梯度向量和 Hessian 矩阵的近似表达式:

$$\nabla g(\boldsymbol{\theta})\approx 2(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{W}(\boldsymbol{\theta}_t)\mathbf{e}(\boldsymbol{\theta}_t)+2(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{W}(\boldsymbol{\theta}_t)\mathbf{J}_e(\boldsymbol{\theta}_t)(\boldsymbol{\theta}-\boldsymbol{\theta}_t)$$

$$\mathbf{H}_g(\boldsymbol{\theta})\approx 2(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{W}(\boldsymbol{\theta}_t)\mathbf{J}_e(\boldsymbol{\theta}_t).$$

相应地，迭代重加权最小二乘方法的迭代公式可写为

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t-\lambda_t[(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{W}(\boldsymbol{\theta}_t)\mathbf{J}_e(\boldsymbol{\theta}_t)]^{-1}(\mathbf{J}_e(\boldsymbol{\theta}_t))^T\mathbf{W}(\boldsymbol{\theta}_t)\mathbf{e}(\boldsymbol{\theta}_t).$$

注意到，该方法存在一个明显的小缺陷，就是在某些情况下，对某些 i，残差 $y_i-f(\mathbf{x}_i,\boldsymbol{\theta}_t)$可能很小，即接近零。此时我们的权 $\omega_i(\boldsymbol{\theta}_t)$的数值计算就可能出现问题. 一种常用而有效的解决方案就是重新定义权为

$$\omega_i(\boldsymbol{\theta}_t)=\frac{1}{\epsilon+|y_i-f(\mathbf{x}_i,\boldsymbol{\theta})|^{2-p}},$$

其中$\epsilon>0$是一个给定的较小常数.

§4.3.7 LASSO

考虑如下的多重线性回归模型

$$y_i=\mathbf{x}_i^T\boldsymbol{\beta}+\varepsilon_i,\quad i=1,2,\ldots,n,$$

其中y_i, $\mathbf{x}_i$, and $\boldsymbol{\beta}$ 分别为响应变量, 协变量, 回归系数, 而 ε_i 为 i.i.d. 服从 $N(0,\sigma^2)$ 的随机误差. 为估计 $\boldsymbol{\beta}$ 如下的 *least absolute shrinkage and selection operator* (LASSO) 现在颇为流行：

$$g(\boldsymbol{\beta})=\sum_{i=1}^{n}(y_i-\boldsymbol{x}_i^T\boldsymbol{\beta})^2+\lambda\sum_{j=1}^{p+1}|\boldsymbol{\beta}_j|,$$

其中 $\boldsymbol{\beta}_j$ 表示 $\boldsymbol{\beta}$ 的第 j 个元素. $\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} g(\boldsymbol{\beta})$. 我们注意到该方法相当于是在 最小二乘的误差平方和的目标函数和的基础上加上了一个关于 $\boldsymbol{\beta}$ 的 L_1 范数的惩罚项. 这里 λ 是惩罚参数, 或者有时也叫做正则化参数 (regularization parameter), 其用于控制惩罚的程度. 不难发现, 如果我们将 $||\boldsymbol{\beta}||_1$ 替换为 $||\boldsymbol{\beta}||_2$, 即 L_2 范数的惩罚项,

$$\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2 + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta},$$

则估计就是我们熟知的岭回归估计. 使用 LASSO 的估计参数 $\widehat{\boldsymbol{\beta}}$ 同岭估计类似, 实现估计偏差和方差之间的某种平衡, 得到的估计在很多情形下比最小二乘估计的均方法误差小很多; 不同的是, 由于后面的惩罚项在零点不可导, 这样的方法有一个特点就是其估计可以精确为零. 因此,该方法通过选取适当的 λ, 我们可以一方面进行变量选择, 而同时得到非零参数的估计.

而正是由于零点不可导的问题, 标准的牛顿法或最速下降法等依赖于梯度或是 Hessian 矩阵的方法此时无法使用. 然而我们可将其近似而转化为我们可以处理的最优化问题. 假设 $\boldsymbol{\beta}_t$ $\boldsymbol{\beta}$ 现在的估计, 如果 β_{tj} 非常接近于零, 则我们就令 $\widehat{\beta}_j = 0$; 否则, 当 $\beta_{tj} \neq 0$, 采用如下的近似

$$|\beta_j|' = \mathrm{sgn}(\beta_j) \approx \beta_j/|\beta_{tj}|,$$

则有

$$|\beta_j|'' \approx 1/|\beta_{tj}|.$$

因此我们有关于 LASSO 惩罚的局部二次近似函数

$$\begin{aligned}|\beta_j| &\approx |\beta_{tj}| + \frac{\beta_{tj}}{|\beta_{tj}|}(\beta_j - \beta_{tj}) + \frac{1}{2|\beta_{tj}|}(\beta_j - \beta_{tj})^2\\ &= |\beta_{tj}| + \frac{1}{2|\beta_{tj}|}(\beta_j^2 - \beta_{tj}^2)\end{aligned}$$

因此,

$$\lambda||\boldsymbol{\beta}||_1 = \lambda\sum_{i=1}^{p+1}|\beta_j| = \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\Sigma}(\boldsymbol{\beta}_t)\boldsymbol{\beta} + c,$$

其中 c 为与 $\boldsymbol{\beta}$ 无关的一个常数, $\boldsymbol{\Sigma}(\boldsymbol{\beta}_t) = \mathrm{diag}\left\{\frac{1}{|\beta_{t1}|}, \ldots, \frac{1}{|\beta_{tp+1}|}\right\}$. 由此最终我们将得到关于惩罚最小二乘目标函数 $g(\boldsymbol{\beta})$ 的局部二次近似, 其由两个二次函数组成

$$g(\boldsymbol{\beta}) \approx (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\Sigma}(\boldsymbol{\beta}_t)\boldsymbol{\beta}.$$

对上式关于 $\boldsymbol{\beta}$ 求导并令其为零则可得到迭代更新方程

$$\boldsymbol{\beta}_{t+1} = \left(\mathbf{X}^T\mathbf{X} + \frac{\lambda}{2}\boldsymbol{\Sigma}(\boldsymbol{\beta}_t)\right)^{-1}\mathbf{X}^T\mathbf{y}.$$

这样, 给定一个初值 $\boldsymbol{\beta}_0$ (比如说, 最小二乘估计), 我们则可利用上式不断迭代直到最后达到某个收敛准则.

对于非线性回归或是广义线性模型, LASSO 同样是很有效的, 也就是说这时考虑的目标函数是某个非线性函数同LASSO惩罚的和, 即

$$g(\boldsymbol{\beta}) = l(\boldsymbol{\beta}, \mathbf{y}) + \lambda \sum_{j=1}^{p+1} |\boldsymbol{\beta}_j|,$$

其中对于广义线性模型, $l(\boldsymbol{\beta}, \mathbf{y})$ 是负的联合对数似然函数, 而对于非线性回归, 则 $l(\boldsymbol{\beta}, \mathbf{y})$ 是误差平方和函数. 这时, 前面两节中介绍的各种算法经过修改均可方便地使用, 只要记住后面的 L_1 惩罚在每一次迭代过程中都如前述般进行二次近似即可.

§4.3.8 EM算法

EM算法最早由Dempster，Laird 和Rubin(1977)所提出，它是受缺失思想以及考虑给定已知项下缺失项的条件分布而激发产生的一种迭代优化策略. 该算法能非常简单地执行并且能通过稳定、上升的步骤可靠地找到最优值. 其收敛性由 Wu (1983)给出，在那之后，其在许多统计问题中得到了良好的、普及性的应用.

我们假设由随机变量$\mathbf{Y}$生成的观测数据连同来自随机变量$\mathbf{Z}$的缺失或未观测数据，而用$\mathbf{X} = (\mathbf{Y}, \mathbf{Z})$记产生完全数据的随机变量. 给定观测数据$\mathbf{y}$，我们希望最大化某似然函数$L(\boldsymbol{\theta}|\mathbf{x})$. 通常采用该似然函数会难以处理，EM算法避开了直接考虑$L(\boldsymbol{\theta}|\mathbf{x})$. 注意到缺失数据可能不是真的缺少了：它们可能仅是一个简化问题的概念上的策略. 在这种情形，$\mathbf{Z}$通常称为*潜数据*. 优化有时可以通过引入这个新要素到问题中而得到简化，我们后面会举说明该方法. 在某些情形，我们必须利用创造力来虚构有效的潜变量；而在一些其它情形，有时亦有自然的选择.

设$f_{\mathbf{Y}}(\mathbf{y}|\boldsymbol{\theta})$和$f_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\theta})$分别表示观测数据和完全数据的密度. EM算法迭代寻求关于$\boldsymbol{\theta}$最大化$L(\boldsymbol{\theta}|\mathbf{x})$. 设$\boldsymbol{\theta}_t$表示在迭代$t$时估计的最大值点，$t = 0, 1, \ldots$. 定义$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$为观测数据$\mathbf{Y} = \mathbf{y}$条件下完全数据的联合对数似然的期望. 即，

$$\begin{aligned} Q(\boldsymbol{\theta}|\boldsymbol{\theta}_t) &= \mathrm{E}\big\{ \log L(\boldsymbol{\theta}|\mathbf{x}) | \mathbf{y}, \boldsymbol{\theta}_t \big\} \\ &= \mathrm{E}\big\{ \log f_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\theta}) | \mathbf{y}, \boldsymbol{\theta}_t \big\} \\ &= \int [\log f_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\theta})] f_{\mathbf{Z}|\mathbf{Y}}(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}_t) d\mathbf{z}, \end{aligned}$$

其中上式强调一旦我们给定$\mathbf{Y} = \mathbf{y}$，$\mathbf{Z}$是$\mathbf{X}$的唯一的随机部分.

EM从$\boldsymbol{\theta}_0$开始，然后在两步之间交替：E表示期望，M表示最大化. 该算法概括如下：

1. **E 步**：计算$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$.
2. **M 步**：关于$\boldsymbol{\theta}$最大化$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$. 设$\boldsymbol{\theta}_{t+1}$等于$Q$的最大值点.

3. 返回E步，直到满足某停止规则为止.

例 4.3.3.（**简单的指数密度**）为理解EM的符号，考虑一个普通的例子，其中$Y_1, Y_2 \sim$ i.i.d. $\mathrm{Exp}(\theta)$. 假定$y_1 = 5$是观测到的，但y_2的值是缺失的. 完全数据对数似然函数为$\log L(\theta|\mathbf{x}) = \log f_{\mathbf{X}}(\mathbf{x}|\theta) = 2\log\{\theta\} - \theta y_1 - \theta y_2$. 取$\log L(\theta|\mathbf{X})$的条件期望得到$Q(\theta|\theta_t) = 2\log\{\theta\} - 5\theta - \theta/\theta_t$，因为由独立性得$\mathrm{E}\{Y_2|y_1, \theta_t\} = \mathrm{E}\{Y_2|\theta_t\} = 1/\theta_t$. 容易发现$Q(\theta|\theta_t)$ 关于θ的最大值点是$2/\theta - 5 - 1/\theta_t = 0$的根. 对$\theta$求解得到更新方程$\theta_{t+1} = \frac{2\theta_t}{5\theta_t+1}$. 注意到这儿E步和M步不需要在每次迭代时重新导出：由某初始值开始对更新公式的反复应用可给出收敛到$\hat{\theta} = 0.2$估计.

例 4.3.4.（**失效时间**）假设一种灯泡的寿命服从某个参数为θ的指数分布. 为了估计θ，我们测试了 n 个灯泡直至他们失效，记失效的时间是 $y_1, \ldots, y_n$; 另外还有一组独立的实验测试了m个灯泡, 但是单个灯泡的失效时间没有被记录, 只记录下来在时刻 t时共有 r个灯泡失效. 不妨记这些缺失的失效时间数据为 $z_1, \ldots, z_m$. 则我们有如下的联合似然函数

$$\log L(\theta|\mathbf{x}) = n(\log\theta - \theta\bar{y}) + \sum_{i=1}^{m}(\log\theta - \theta z_i)$$

取条件期望得到

$$\mathrm{E}\big\{\log L(\boldsymbol{\theta}|\mathbf{x})|\mathbf{y}, \boldsymbol{\theta}_t\big\} = (n+m)\log\theta - \theta\left[n\bar{y} + (m-r)(t + \frac{1}{\theta_t}) + r(\frac{1}{\theta_t} - th_t)\right],$$

其中

$$h_t = \frac{\exp\{-t\theta_t\}}{1 - \exp\{-t\theta_t\}}.$$

则我们在第 $t+1$个M-步确定最大值 θ_{t+1}

$$\theta_{t+1} = (n+m)\left[n\bar{y} + (m-r)(t + \frac{1}{\theta_t}) + r(\frac{1}{\theta_t} - th_t)\right]^{-1}.$$

例 4.3.5.（**混合分布**）设 Y 的密度函数为

$$f(y_j|\boldsymbol{\psi}) = \sum_{i=1}^{g}\pi_i f_i(y_j|\theta_i),$$

其中 $\pi_i > 0$, $\sum_{i=1}^{g}\pi_i = 1$, $f_i(\cdot|\cdot)$ 是密度函数. 这里记 $\boldsymbol{\psi} = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\xi}^T)^T$ 和 $\boldsymbol{\xi} = (\theta_1, \ldots, \theta_g)^T$.

在给定观测$\mathbf{y}$下对$\boldsymbol{\psi}$的对数似然函数可写为

$$\log L(\mathbf{y}|\boldsymbol{\psi}) = \sum_{j=1}^{n}\log f(y_j|\psi) = \sum_{j=1}^{n}\log\left[\sum_{i=1}^{g}\pi_i f_i(y_j|\theta_i)\right]$$

下面考虑使用EM算法对上式优化求解$\widehat{\boldsymbol{\psi}}$. 在该问题中，看上去没有缺失数据，我们这里人为地构造缺失数据$\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)^T$，其中$\mathbf{z}_j = (z_{1j}, \ldots, z_{gj})^T$是一$g$维向量: 如果 y_j 来自于混合

分布的第i个分布，则 $z_{ij}=1$, 否则 $z_{ij}=0$. 由此可知 $z_1,\ldots,z_n$是独立同分布地来自于多点分布 $\mathrm{Mult}_g(1,\pi)$. 则我们可进一步地考虑如下完全数据 $\mathbf{x}=(\mathbf{y},\mathbf{z})$的对数似然函数

$$\log L(\boldsymbol{\psi}|\mathbf{x})=\sum_{i=1}^{g}\sum_{j=1}^{n}z_{ij}\left[\log\pi_i+\log f_i(y_j|\theta_i)\right].$$

下面来进行E-步，也就是求$L(\mathbf{x}|\boldsymbol{\psi})$的条件期望

$$\mathrm{E}\big\{\log L(\boldsymbol{\psi}|\mathbf{x})|\mathbf{y},\boldsymbol{\psi}_t\big\}.$$

注意到由Bayes公式我们可得到

$$\mathrm{E}\big\{Z_{ij}|\mathbf{y},\boldsymbol{\psi}_t\big\}=P(Z_{ij}=1|\mathbf{y},\boldsymbol{\psi}_t)=\tau_i(y_j|\boldsymbol{\psi}_t),$$

其中

$$\begin{aligned}\tau_i(y_j|\boldsymbol{\psi}_t)&=\pi_{ti}f_i(y_j|\theta_{ti})/f(y_j|\boldsymbol{\psi}_t)\\&=\pi_{ti}f_i(y_j|\theta_{ti})/\sum_{h=1}^{g}\pi_{th}f_h(y_j|\theta_{th}).\end{aligned}$$

则条件期望可写为

$$Q(\boldsymbol{\psi}|\boldsymbol{\psi}_t)=\sum_{i=1}^{g}\sum_{j=1}^{n}\tau_i(y_j|\boldsymbol{\psi}_t)\left[\log\pi_i+\log f_i(y_j|\theta_i)\right].$$

下面进行M-步. 不难得到

$$\pi_{t+1,i}=\sum_{j=1}^{n}\tau_i(y_j|\boldsymbol{\psi}_t)/n$$

而 $\boldsymbol{\psi}$中的剩余部分 $\boldsymbol{\xi}$的第 $(t+1)$ 个估计则是通过对下式的最大化求得的

$$\xi_{t+1}=\arg\max\sum_{i=1}^{g}\sum_{j=1}^{n}\tau_i(y_j|\boldsymbol{\psi}_t)\log f_i(y_j|\theta_i).$$

第5章　R 中的回归分析

§5.1　相关性及其度量

变量之间相互关系大致可分为两种类型,即函数关系和相关关系. 函数关系是指变量之间存在的相互依存关系,它们之间的关系可以用某一方程（函数）表达出来；相关关系是指两个变量的数值变化存在不完全确定的依存关系,它们之间的数值不能用方程表示出来,但可用某种相关性度量来刻画. 相关关系是相关分析的研究对象,而函数关系是回归分析的研究对象.

散点图是一种最为有效最为简单的相关性分析工具. 若通过散点图可以基本明确它们之间存在直线关系，则可通过线性回归进一步确定它们之间的函数关系, R中使用函数 plot( ) 可以方便地画出两个样本的散点图，从而直观地了解对应随机变量之间的相关关系和相关程度. 这方面我们在第三章中进行了详细介绍, 这里不再重复.

设两个随机变量X与Y的观测值为$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, 则它们之间的（样本）相关系数为

$$\gamma(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

可以证明，当样本个数 n 充分大时，样本相关系数可以作为总体 X 和 Y 的相关系数的估计. 除此之外，还有Spearman秩相关系数和Kendall相关系数. R 中提供了函数 cor() 来计算这三种相关系数.

cor(x, y, method=c(“pearson”,“kendall”,“spearman”)).

cor(faithful[[1]], faithful[[2]], method=“kendall”).

进一步地，若 X, Y 服从二元独立正态分布，则

$$T = \frac{\gamma(X, Y)\sqrt{n-2}}{\sqrt{1 - \gamma(X, Y)^2}} \sim t(n-2).$$

由此可以对 X 和 Y 进行Pearson相关性检验：若 $T > t_\alpha(n-2)$, 则认为 X 和 Y 的观测之间存在显著的相关性. 此外，还可根据Spearman秩相关系数和Kendall相关系数进行相应的Spearman秩检验和Kendall秩检验.

在R软件中，cor.test()提供了上述三种检验方法，其调用格式用

cor.test(x, y, alternative=c(“two.sided”,“less”,“greater”),

method=c(“pearson”,“kendall”,“spearman”), conf.level=0.95, $\cdots$)

其中 x, y 是长度相同的向量; alternative是备择假设, 默认值为“two.sided”; method是选择检验方法, 默认值为Pearson检验; coef.level是置信水平, 默认值为 0.95.

library(ISwR); intake; cor.test(intake[[1]], intake[[2]])

§5.2 多元线性回归分析

考虑如下的多重线性回归模型

$$y_i = \mathbf{x}_i^T\boldsymbol{\beta} + \varepsilon_i, \quad i = 1, 2, \ldots, n,$$

其中y_i, $\mathbf{x}_i$, and $\boldsymbol{\beta}$ 分别为响应变量, 协变量, 回归系数, 而 ε_i 为 i.i.d. 服从 $N(0, \sigma^2)$ 的随机误差.

1. 基本命令. 在 R 中，由函数 lm() 可以非常方便地求出回归方程. 函数 lm() 基本的调用格式为:

   lm(formula=, data=)

   其中 formula 是回归模型, data是数据框.

   nyc<–read.table("nyc.txt")

   lm.reg<–lm(formula = Price∼Food+Decor+Service+East, data=nyc); lm.reg

   该函数的返回值是拟合结果的对象, 本质上是一个具有类属性值 lm 的列表, 有model, coefficients, residuals 等成员. 直接运行该函数的结果非常简单, 为了获得更多的信息, 可以使用对 lm() 类有特殊操作的通用函数. 常用的函数包括:

   - summary()—提取模型拟合详细结果
   - coefficients()—提取模型系数, 可用简写形式 coef()
   - deviance()—计算残差平方和
   - formula()—提取模型公式
   - plot()—绘制模型诊断的几种图形
   - predict()—作预测
   - residuals()— 计算残差
   - step()— 作逐步回归分析

   例如：coef(lm.reg); sqrt(deviance(lm.reg)/160). 其它函数的使用我们后面将会逐一看到.

2. 显著性检验. 检验分为回归系数的显著性检验和回归方程的显著性检验.

   - 回归系数的显著性检验. 使用检验统计量

   $$T_j = \frac{\widehat{\beta}_j}{\widehat{\sigma}\sqrt{c_{jj}}} \sim t(n-p-1),$$

   其中 $\widehat{\sigma}^2 = SSE/(n-p-1)$, c_{jj} 是 $(\mathbf{X}^T\mathbf{X})^{-1}$ 的对角线上的第 j 个元素, 而 $\widehat{\sigma}\sqrt{c_{jj}}$ 即为 $\widehat{\beta}_j$ 的标准差.

- 回归方程的显著性检验则使用常用的 F 检验, 即

$$F = \frac{SSR/p}{SSE/(n-p-1)} \sim F(p, n-p-1),$$

其中 SSR 和 SSE 分别是回归平方和、残差平方和.

我们可使用 summary(lm.reg) 可清晰地得到这两种检验的检验统计量和相应的 p 值;

3. 参数的区间估计.

$$\frac{\widehat{\beta}_j - \beta_j}{\mathrm{sd}(\widehat{\beta}_j)} \sim t(n-p-1)$$

β_j 的区间估计为

$$\left[\widehat{\beta}_j - \mathrm{sd}(\widehat{\beta}_j)t_{\alpha/2}(n-p-1), \widehat{\beta}_j + \mathrm{sd}(\widehat{\beta}_j)t_{\alpha/2}(n-p-1)\right]$$

confint(object, parm, level=0.95, $\cdots$)

说明：object是指回归对象, parm 是指要求指出所求区间估计的参数, 默认值为所有的回归参数, 若取指定的, 可使用 parm=2, 即表示求第二个估计参数的置信区间, level是指置信水平, 默认 0.95.

confint(lm.reg); confint(lm.reg, parm=2)

4. 预测. 当多元线性回归方程经过检验是显著的，且其中每一个回归系数均显著时（不显著则先剔除）, 这时可用此回归方程作预测. 给定 $\mathbf{x}_0$, 将其代入回归方程, 得预测值 $\widehat{y}_0 = \mathbf{x}_0^T\widehat{\boldsymbol{\beta}}$. 相应的置信水平为 $1-\alpha$ 的预测区间为

$$\widehat{y}_0 \pm t_{\alpha/2}(n-p-1)\widehat{\sigma}\sqrt{1 + \mathbf{x}_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_0}.$$

R 中可以利用 predict() 函数求预测值和预测区间. 仍以 nyc 的数据为例, 由于 Service 的回归系数不显著, 因此我们将其在回归变量中剔除, 重新进行回归计算.

lm.reg<–lm(formula = Price∼Food + Decor + East, data=x1)

假定给定新的 $\mathbf{x}_0 = (20, 20, 1)$, 我们使用如下命令进行预测

x0<–data.frame(Food=20, Decor=20, East=1)

predict(lm.reg, x0, interval="prediction", level=0.95)

其中, x0 必须先定义为数据框才可使用 predict函数, 而interval="prediction" 表示给出相应的预测区间, 如果去掉这个参数, 则只给出预测值 y_0.

5. 共线性诊断. 我们通常把 $\mathbf{X}^T\mathbf{X}$ 变换为主对角线为 1 的矩阵, 然后求出特征值和特征向量. 如果有 r 个特征值近似等于 0, 则回归设计阵中有 r 个共线性关系. 度量多重共线性的一个重要指标是矩阵 $\mathbf{X}^T\mathbf{X}$ 的条件数

$$\kappa(\mathbf{X}^T\mathbf{X}) = \frac{\lambda_{\max}(\mathbf{X}^T\mathbf{X})}{\lambda_{\min}(\mathbf{X}^T\mathbf{X})},$$

其中 $\lambda_{\max}(\mathbf{X}^T\mathbf{X})$ 和 $\lambda_{\min}(\mathbf{X}^T\mathbf{X})$ 分别表示矩阵 $\mathbf{X}^T\mathbf{X}$ 的最大、最小特征值. 直观上, 该指标刻画了 $\mathbf{X}^T\mathbf{X}$ 的特征值差异的大小. 经验来说, 若该值小于100, 则认为共线性的程度很小. R 中使用 kappa() 函数来求取此条件数.

xx<–nyc[c(“Food”,“Decor”,“East”)]; cor(xx); kappa(cor(xx))

这里我们注意到这里中心标准化后的矩阵 $\mathbf{X}^T\mathbf{X}$ 即是原矩阵的相关系数矩阵, 因此直接使用 cor() 函数则可直接求得.

6. 回归方程的选择. 选择一个“最优”回归就是从可供选择的所有变量中选出对 Y 有显著影响的变量逐步建立方程, 使得在方程中不含对 Y 无显著影响的变量. 几种常用的最优回归方程选择方法包括: “一切子集回归”, “向前法”, “向后法”, “逐步回归”. R 中提供了函数 step() 来实现这些回归方法. 其基本调用命令如下:

step(object, direction = c(“both”, “backward”, “forward”), k=2)

该函数通过选择最小的AIC 或是 BIC 信息统计量, 来达到删除或增加变量的目的. 表达式中如取 $k = \log(n)$, 则表示以 BIC 为准则.

nyc1<–subset(nyc,is.na(Service)==F). 这一步是因为使用 step() 函数不能有缺失值, 因此首先提取出 165 行的子数据框.

lm.reg<–lm(formula = Price∼Food+Decor+Service+East, data=nyc1);

step(lm.reg); step(lm.reg, k=log(165))

对于一切子集回归等其它方法可使用软件包 leaps 中的函数 regsubsets().

7. 残差分析.

R 中可分别用 residuals(), rstandard(), 和 rstudent() 来分别求取残差, 标准化残差, 学生化残差

$$\widehat{\epsilon}_i = y_i - \widehat{y}_i, \quad \frac{\widehat{\epsilon}_i}{\widehat{\sigma}\sqrt{1-h_{ii}}}, \quad \frac{\widehat{\epsilon}_i}{\widehat{\sigma}_{(i)}\sqrt{1-h_{ii}}},$$

其中 h_{ii} 帽子矩阵 $\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ 的对角线上的第 i 个元素, $\sigma^2_{(i)} = \frac{1}{n-p-2}\sum_{j\neq i}(y_i - \mathbf{x}_j^T\widehat{\boldsymbol{\beta}}_{(i)})^2$, 而 $\widehat{\boldsymbol{\beta}}_{(i)}$ 是删去第 i 个样本点后用余下的 $n-1$ 个样本点求得的回归系数.

在实际应用, 通过对残差进行正态检验, 我们可粗略判断回归模型的正态性假设是否成立.

y.res<-residuals(lm.reg); shapiro.test (y.res)

而通过画出回归值 $\widehat{y}_i$ 和 $\widehat{\epsilon}_i$ 的散点图, 或者 $\widehat{y}_i$ 和标准化残差的散点图, 来直观地检验所建立的回归模型是否合适, 这样的图我们一般称之为残差图. R 中通过形如命令 plot(lm.reg) 可以最简单直接地得到我们最需要的几个图形.

§5.3 logistic 回归

回顾一下例4.3.2 中所讲述的 logistic 广义线性回归模型,

$$f(\pi_i) = \log\{\pi_i/(1-\pi_i)\} = \mathbf{x}_i^T\boldsymbol{\beta} \equiv \alpha_i,$$

$$y_i \sim \mathrm{BIN}(\pi_i, m_i).$$

R 语言提供了拟合和计算广义线性模型的函数 glm(), 其基本调用格式为

glm(formula=, family=, data=)

我们这里以作业中的例子举例说明. 对于 logistic 回归模型, 上面的命令中取 family=binomial, 而其它参数同 lm().

library(ISwR); lg.reg<–glm(mal∼age+log(ab), family=binomial, data=malaria); summary(lg.reg)

注意到在回归结果中一些与线性模型的结果的区别之处: 首先, residuals 换为了 Deviance. Deviance 的概念就是针对广义线性模型中残差的定义不合适这一问题所引入的. 回顾一下我们的对数似然函数

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \log \pi_i + (m_i - y_i)\log(1-\pi_i) \right].$$

Deviance 的定义就是在饱和模型下和 logistic 回归模型下的两倍极大对数似然的差

$$\begin{aligned} D =& 2\left\{ \sum_{i=1}^{n} \left[y_i \log \frac{y_i}{m_i} + (m_i - y_i)\log\left(1 - \frac{y_i}{m_i}\right) \right] \right\} \\ & - 2\left\{ \sum_{i=1}^{n} \left[y_i \log \frac{\widehat{y}_i}{m_i} + (m_i - y_i)\log\left(1 - \frac{\widehat{y}_i}{m_i}\right) \right] \right\} \\ =& 2\left\{ \sum_{i=1}^{n} \left[y_i \log \frac{y_i}{\widehat{y}_i} + (m_i - y_i)\log\left(\frac{m_i - y_i}{m_i - \widehat{y}_i}\right) \right] \right\} \equiv \sum_{i=1}^{n} d_i^2, \end{aligned}$$

其中 $\widehat{y}_i = m_i\widehat{\pi}_i = m_i \log \frac{e^{\mathbf{x}_i^T\hat{\boldsymbol{\beta}}}}{1+e^{\mathbf{x}_i^T\hat{\boldsymbol{\beta}}}}$. 上式中我们对于饱和模型, 也就是仅假设 $y_i \sim \mathrm{BIN}(\pi_i, m_i)$, 利用了其极大似然估计 $\tilde{\pi}_i = y_i/m_i$ 这一显然的结果. 这个基于似然比的 Deviance D 以及其组成元素 d_i 显然能够刻画 $\widehat{y}_i$ 和 y_i 之间的差异从而衡量回归方程的好坏. 在显示结果中的 Deviance residuals一项即指 $\mathrm{sgn}(y_i - \widehat{y}_i)d_i$.

其次, t 检验统计量这里换为了 z 检验统计量, 这是一正态近似的检验, 这里不详细介绍了. 我们注意到变量 age 对应的回归系数不显著. 对于广义似然模型, 我们仍旧可以使用变量选择的方法, 方法同前面线性模型类似, 如:

summary(step(lg.reg)); 显然, 逐步回归的最终模型中剔除了变量 age.

我们再来看一个 $m_i \geq 1$ 的例子.

mf<–read.table("MichelinFood.txt", header=TRUE); mf

names(mf)<–c("Food", "In", "Notin", "mi", "prop")

plot(Food∼prop, data=mf)

m1<–glm(cbind(In, Notin)∼Food, family=binomial, data=mf); summary(m1)

§5.4 非线性回归

R 软件提供了函数 nls() 来计算非线性模型的参数估计, 其默认方法使用我们前面介绍的 Gauss-Newton 法. 其基本使用格式为

nls(formula=, data=, start=, control=nls.control())

其中 start 是初值点, 以列表的形式给出, control 用于指定算法中的精度等参数.

nfit<–read.table("nfit.txt", header=T)

plot(Y∼X, data=nfit)

g<–function(x, a1, a2, a3) {y<–(a1∗x^2+a2∗exp(a3∗x))/x;y }

ncol<–nls.control(maxiter = 50, tol = 1e-08)

nls.sum<–nls(Y∼g(X, a1, a2, a3), data=nfit, start=list(a1=0.5, a2=0.5, a3=0.5), control=ncol)

其中的 nls.control() 函数是用于指定参数, 其中最常用的是 maxiter 控制迭代次数, tol 控制相对误差的大小. 关于估计参数的假设检验亦可通过 summary() 来实现. summary(nls.sum) 最后给出数据的散点图和相应的拟合曲线

xfit<–seq(-2.5, 6.0, 100); yfit<–predict(nls.sum, data.frame(X=xfit))

plot(Y∼X, data=nfit); lines(xfit, yfit)

§5.5 LASSO

R 软件提供了几个软件包都可以用来计算 LASSO 的估计, 我们这里只简要介绍其中常用的一个 lasso2. 下载并加载该软件包. 我们可通过帮助来查看该 package 的详细内容. help(package=lasso2). 其中函数 l1ce() 是用于进行 LASSO 估计的. 其使用方法同 lm() 类似,

区别在于这时需要指定参数 bound.

```
l1<–l1ce(Price∼Food + East + Service + Decor, data = nyc, bound=0.5); summary(l1)
```

该包中还提供了 gcv() 函数用于对 bound 进行选择.

第6章　统计模拟

本章介绍从某一目标分布 f 中随机抽取变量 $\mathbf{X}_1,\ldots,\mathbf{X}_n$ 并用模拟方法来估计所感兴趣的量, 如模型的期望值, 事件的概率等. 譬如, 考虑我们前面讲过的 Kolmogorov-Smirnov 检验

$$D=\sup_{-\infty<x<\infty}|F_n(x)-F_0(x)|.$$

我们考虑如何能够通过模拟的方法来获得该检验的近似临界值 $\xi_\alpha, 0<\alpha<1$. 由极限理论知识我们可知, 样本分位点 $\widehat{\xi}_\alpha$ 是 ξ_α 的强相合估计, 因此我们可随机生成 N 次样本容量为 n 的来自 $F_0(x)$ 的样本并对每次模拟生成的样本计算一个 $D_i, i=1,\ldots,N$. 最后求取 $\{D_1,\ldots,D_N\}$ 的样本分位数, 则得到 ξ_α 的估计值. 这就是一个典型的模拟过程. 我们之后会详加阐述.

§6.1　随机数的产生

在统计模拟中, 我们首先需要产生各种概率分布的随机数, 如我们上面的例子中来自 $F_0(x)$ 的随机样本. 大多数概率分布的随机数的产生都基于均匀分布 $U(0,1)$ 的随机数. 由于 $U(0,1)$ 分布的随机数在大多数统计计算软件中都有现成的程序可调用, 我们这里略去关于它的产生方法, 而介绍利用均匀随机数产生其它非均匀随机数的一般方法, 即产生服从各种不同分布随机变量的方法, 我们常把产生各种随机变量的随机数这一过程称为*对随机变量进行模拟*, 或称为*对随机变量进行抽样*, 且称产生某个随机变量的随机数的方法为*抽样法*.

一般情况下, n 个独立随机变量可由 n 次重复抽样获得.

§6.1.1　逆变换法

该方法有时也称之为*直接抽样法*. 设随机变量 X 的分布函数为 $F(x)$, 定义

$$F^{-1}(y)=\inf\{x:F(x)\geq y\},\ 0\leq y\leq 1. \tag{6.1}$$

则我们有如下结论: 假设随机变量 U 服从 $(0,1)$ 上的均匀分布, 则 $X=F^{-1}(U)$ 的分布函数为 $F(x)$. 由 (6.1) 和均匀分布的分布函数可得

$$P(X\leq x)=P(F^{-1}(U)\leq x)=P(U\leq F(x))=F(x).$$

由该结论, 要产生来自 $F(x)$ 的随机数, 只要先产生来自 $U(0,1)$ 的随机数 u, 然后由 (6.1) 计算 $F^{-1}(u)$ 即可.

例 6.1.1. 设 $X\sim U(a,b)$, 则其分布函数

$$F(x)=\begin{cases}0, & x<a\\ \frac{x-a}{b-a}, & a\leq x<b\\ 1, & x\geq b.\end{cases}$$

从而有 $F^{-1}(u)=a+(b-a)u, 0\le u\le 1$. 若由 $U(0,1)$ 抽得 u, 则 $a+(b-a)u$ 来是自 $U(a,b)$ 的一个随机数.

类似本例, 我们亦可简单地产生密度函数为 $\lambda e^{-\lambda x}$ $(x>0)$ 的指数分布随机数.

例 6.1.2. 设 $X_1,\dots,X_n$ 是来自 $F(x)$ 的一个样本, $Y_1=\min(X_1,\dots,X_n)$, 我们想要产生 Y_1 的随机数. 自然地, 这可以通过产生 n 个来自 $F(x)$ 的独立随机变量并求最小值来得到, 但是我们没必要这样做. 我们知道,

$$F_{Y_1}(y)=1-[1-F(y)]^n\equiv G(y)$$

则易得

$$Y_1=G^{-1}(U)=F^{-1}(1-(1-U)^{1/n})$$

应用逆变换法, 则可直接产生来自 $G(y)$ 的随机数.

基于 (6.1) 定义的逆变换法对于生成离散分布的随机变量依然是有效的. 设 ξ 为一离散变量, 其概率分布为 $P(\xi=x_i)=p_i, i=1,2,\dots$. 不妨假设 $x_1<x_2<\cdots$. 记 ξ 的分布函数为

$$F(x)=P(\xi\le x)=\sum_{x_i\le x}p_i.$$

则根据逆变换法, 产生离散分布 $F(x)$ 随机数如下：首先产生 $(0,1)$ 上的均匀随机数 u, 取 $\xi=x_i$, 若 $F(x_{i-1})<u\le F(x_i), i=2,3,\dots$; 取 $\xi=x_1$, 若 $u<F(x_1)$. 则不难看出 $\xi\sim F(x)$.

例 6.1.3. 设 $\xi\sim \mathrm{BIN}(n,p)$, 考虑生成其随机数. 两种最直接的方法是：(1) 将 ξ 表示为 n 个两点分布变量的独立和. 因此, 我们可以先产生 n 个 $\mathrm{BIN}(1,p)$ 分布的随机数, 然后相加即可得到一个 $\mathrm{BIN}(n,p)$ 的随机数. 即, 先产生 n 个 $U(0,1)$ 分布的随机数 $u_1,\dots,u_n$; 计算 $x_i=I_{\{u_i\le p\}}, i=1,\dots,n$; 计算 $\sum\limits_{i=1}^{n}x_i$. (2) 直接应用逆变换, 生成 $U(0,1)$ 分布的随机数 u, 若 $\sum_{i=0}^{k-1}C_n^ip^i(1-p)^{n-i}<u\le\sum_{i=0}^{k}C_n^ip^i(1-p)^{n-i}$, 则 $\xi=k$. 事实上, 第二种方法还可以如下更简便地进行操作

(i) 生成 $U(0,1)$ 分布的随机数 u;

(ii) $c=p/(1-p)$, $i=0$, $a=(1-p)^n$, $F=a$;

(iii) 若 $u<F$, 令 $X=i$, 停止;

(iv) 否则, $a=[c(n-i)/(i+1)]a$, $F=F+a$, $i=i+1$, 转 (iii).

例 6.1.4. 生成离散均匀分布随机数, 即随机变量 ξ 的概率分布为

$$P(\xi=k)=\frac{1}{n}\quad (k=1,2,\dots,n).$$

这时使用逆变换法不需要像在前例中那样进行搜索. 事实上, 当 $\frac{j-1}{n}<u\le\frac{j}{n}$ 时, 取 $\xi=j$. 即 $j-1<nu\le j$ 时, $\xi=j$, 等价地, 取 $\xi=\lceil nu\rceil$ 即可.

直接抽样法是一个常用的方法, 对连续型和离散型的随机变量均有效. 对于连续型随机变量, 应用该法必须求得其分布函数的反函数, 但很多分布函数的反函数不能用初等函数表给出, 如常用的正态, t 以及 $Gamma$ 分布等. 抽样分布不能精确表出时, 我们当然可以用前面介绍的数值求根法就解方程 $u = F(\xi)$, 得到 ξ 的随机数. 但这样显然效率很低, 下面我们再介绍几种其它的一般抽样方法.

§6.1.2 筛选抽样法

假设我们要从 $f(x)$ 中抽样, 如果可以将 $f(x)$ 表示成 $f(x) = c \cdot h(x) \cdot g(x)$, 其中 $h(x)$ 是一个密度函数且易于抽样, 而 $0 \le g(x) \le 1$, $c \ge 1$ 是常数, 则 X 的抽样可如下进行:

(1) 由 $U(0,1)$ 抽取 u, 由 $h(y)$ 抽取 Y;

(2) 若 $u \le g(Y)$, 则 $X = Y$, 停止;

(3) 若 $u > g(Y)$, 回到 (1).

由上式表示的方法我们称之为**筛选抽样** (*Acceptance/Rejection Sampler*). 筛选抽样法不是对所产生的随机数都录用, 而是建立一个检验条件, 利用这一检验条件进行筛选, 得到所需的随机数. 由于其很灵活、计算简单且使用方便, 得到了非常广泛的应用. 其理论依据如下: 假设 u 和 Y 分别服从 $U(0,1)$ 和 $h(y)$, 则在 $u \le g(Y)$ 的条件下, Y 的密度函数为

$$p_Y(x|u \le g(Y)) = f(x).$$

该结论可如下简单地证明得到: 由 Bayes 公式且注意到 $1/c = \int g(x)h(x)dx$, 有

$$\begin{aligned} P(X \le z) &= P(Y \le z | u \le g(Y)) \\ &= \frac{P(Y \le z, u \le g(Y))}{P(u \le g(Y))} = \frac{\int_{-\infty}^{z}\int_{-\infty}^{g(y)} f_U(x)h(y)dxdy}{\int_{-\infty}^{\infty}\int_{-\infty}^{g(y)} f_U(x)h(y)dxdy} \\ &= \frac{\int_{-\infty}^{z}\int_{-\infty}^{g(y)} h(y)dxdy}{\int_{-\infty}^{\infty}\int_{-\infty}^{g(y)} h(y)dxdy} \\ &= \frac{\int_{-\infty}^{z} g(y)h(y)dy}{\int_{-\infty}^{\infty} g(y)h(y)dy} = \int_{-\infty}^{z} cg(y)h(y)dy = \int_{-\infty}^{z} f(y)dy \end{aligned}$$

应该指出, 筛选抽样法有两点很重要. 一方面, $h(x)$ 易于抽样, 另一方面, 我们不能保证抽几次可以得到一个 $f(x)$ 的随机数. 这方面涉及到抽样法效率问题, 也就是说平均抽几次可得到一个随机数, 我们在这里不作详细讨论.

选取 $h(x)$ 有多种方法. 一种直观的方法是: 如果存在一个函数 $M(x)$, 满足 $f(x) \le M(x)$, 且 $c = \int M(x)dx < \infty$, 令 $h(x) = M(x)/c$, 则 $f(x) = c \cdot h(x) \cdot f(x)/M(x)$. 如果 $h(x)$ 易于抽样, 则由如下筛选抽样法:

(1) 由 $U(0,1)$ 抽取 u, 由 $h(Y)$ 抽取 y;

(2) 若 $u \le f(Y)/M(Y)$, 则 $X = Y$, 停止;

(3) 若 $u > f(Y)/M(Y)$, 回到 (1).

该种方法, 对 X 的取值空间有限时特别简单有效, 这时总可取 h 为均匀分布. 比如, $X \sim f(x), -\infty < a \le x \le b < \infty$, 并设 $M = \sup f(x)$ 存在, 则可取 $h(x) = 1/(b-a)$, $c = M(b-a)$, $g(x) = f(x)/M$, 上面的筛选抽样法简化为:

(1) 由 $U(0,1)$ 抽取 u_1 和 u_2;

(2) 计算 $Y = a + u_2(b-a)$;

(3) 若 $u_1 \le g(Y) = f(a + u_2(b-a))/M$, 则 $X = Y$, 停止;

(4) 若 $u_1 > g(Y)$, 回到 (1).

例 6.1.5. 生成参数为 (a,b) 的 $Beta$ 分布的随机数, 即

$$X \sim f(x) = \frac{1}{\beta(a,b)} x^{a-1}(1-x)^{b-1},\ 0 < x < 1,$$

不难得到, 当 $x = \frac{a-1}{a+b-2}$ 时, $f(x)$ 取最大值 D:

$$D = \frac{1}{\beta(a,b)} \left(\frac{a-1}{a+b-2}\right)^{a-1} \left(\frac{b-1}{a+b-2}\right)^{b-1}.$$

由筛选法我们则可得到抽样过程如下:

(1) 由 $U(0,1)$ 抽取 u_1 和 u_2;

(2) 如果 $u_2 \le f(u_1)/D$ 时, 则 $X = u_1$, 否则转到 (1).

例 6.1.6. 生成尺度参数为 1 的 $Gamma$ 分布的随机数, 即

$$X \sim f(x) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x},\ x > 0,$$

$0 < \alpha < 1$ 已知. 注意到

$$\frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} \le M(x) = \begin{cases} x^{\alpha-1}/\Gamma(\alpha), & 0 < x \le 1 \\ e^{-x}/\Gamma(\alpha), & x > 1. \end{cases}$$

因为 $c = \int_0^\infty M(x)dx = (1/\alpha + 1/e)/\Gamma(\alpha) < \infty$, 取

$$h(x) = \begin{cases} x^{\alpha-1}/(1/\alpha + 1/e), & 0 < x \le 1 \\ e^{-x}/(1/\alpha + 1/e), & x > 1. \end{cases}$$

则 $g(x)=\begin{cases} e^{-x}, & 0<x\le 1 \\ x^{\alpha-1}, & x>1. \end{cases}$ 于是, 该分布的随机数可如下生成:

(1) 由 $U(0,1)$ 抽取 u;

(2) 用逆变换法由 $h(y)$ 抽取 Y;

(3) 当 $Y\in(0,1]$时, 如果 $u\le e^{-Y}$, 则 $X=Y$, 否则转到 (1);

(4) 当 $Y>1$时, 如果 $u\le Y^{\alpha-1}$, 则 $X=Y$, 否则转到 (1);

例 6.1.7. 生成半正态分布的随机数, 即

$$f(x)=\begin{cases} \sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}, & x\ge 0 \\ 0, & x<0. \end{cases}$$

注意到

$$\begin{aligned} f(x)&=\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}=\sqrt{\frac{2e}{\pi}}e^{-x}e^{-\frac{(x-1)^2}{2}} \\ &\equiv c\cdot h(x)\cdot g(x), \end{aligned}$$

其中 $h(x)=e^{-x}$ $(x>0)$ 为指数分布的密度函数; $g(x)=e^{-\frac{(x-1)^2}{2}}$ 的值域属于$[0,1]$; $c=\sqrt{\frac{2e}{\pi}}>1$. 注意检验条件为

$$u\le g(y)=e^{-\frac{(y-1)^2}{2}}\iff -\log u\ge\frac{(y-1)^2}{2}.$$

于是, 该分布的随机数可如下生成:

(1) 由 $U(0,1)$ 抽取 u_1 和 u_2;

(2) 计算 $Y=-\log u_2$. 如果 $-\log u_1\ge\frac{(Y-1)^2}{2}$, 则令 $X=Y$, 否则转到 (1).

§6.1.3 复合抽样法

该方法一般也称为合成法. 如果 X 的密度函数 $f(x)$ 难于抽样, 而 X 关于 Y 的条件密度函数 $p_{X|Y}(x|y)$ 以及 Y 的密度函数 $g(y)$ 均易于抽样, 则可如下产生 X 的随机数.

(1) 由 Y 的分布 $g(y)$ 抽取 Y_i;

(2) 由条件分布 $p_{X|Y}(x|Y=Y_i)$ 抽取 X_i.

不难证明（留作习题）, 由上所得的 X_i 服从 $f(x)$.

例 6.1.8. 设 X 的密度函数为 $f(x)=\sum_{i=1}^{n}\alpha_i f_i(x)$, 其中 $\alpha_i>0$, $\sum_{i=1}^{n}\alpha_i=1$, $f_i(x)$ 是密度函数. 令 $\alpha_0=0$, 由合成法, X 的随机数可以如下抽取:

(1) 由 $U(0,1)$ 抽取 u;

(2) 确定 k, 使得 $\sum_{i=0}^{k-1}\alpha_i<u\le\sum_{i=0}^{k}\alpha_i$;

(3) 由 $f_k(x)$ 抽取 X.

例如, $f(x)=(1+2x)/6$, $0<x<2$, 其分布函数为 $F(x)=(x+x^2)/6$, 若用逆变换法, 需解方程; 考虑合成法, 将 $f(x)$ 分解, $f_1(x)=1/2, f_2(x)=x/2, 0<x<2$, $\alpha_1=1/3$, $\alpha_2=2/3$, 由合成法, 结合逆变换法, 我们可给出具体抽样步骤为:

(1) 由 $U(0,1)$ 抽取 u_1 和 u_2;

(2) 计算 $X=\begin{cases}2u_2, & u_1\le 1/3\\ 2\sqrt{u_2}, & u_1>1/3.\end{cases}$

§6.1.4 随机向量的抽样法

在用 Monte Carlo 等方法接应用问题时, 随机向量的抽样也是经常用到的. 若随机向量各分量相互独立, 则它等价于多个一元随机变量的抽样. 因此, 下面我们讨论非独立随机向量的抽样.

设 $X_1,\dots,X_k$ 的概率密度函数为

$$f(x_1,\dots,x_k)=f_1(x_1)f_2(x_2|x_1)\cdots f_k(x_k|x_1,\dots,x_{k-1}),$$

其中 $f_1(x_1)$ 为 X_1 的边际分布密度函数, $f_i(x_i|x_1,\dots,x_{i-1})$ 为给定 $X_1,\dots,X_{i-1}$ 下 X_i 的条件分布密度函数. 由此, 随机向量 $X_1,\dots,X_k$ 可如下产生:

(1) 产生 $X_1\sim f_1(x_1)$;

(2) 以 X_1 为参数, 产生 $X_2\sim f_2(x_2|x_1)$;

(3) 依此类推, 产生 X_i, 最后输出向量 $(X_1,\dots,X_k)^T$.

例 6.1.9. 设 X_1,X_2 的联合密度函数为

$$f(x_1,x_2)=\begin{cases}6x_2, & x_1+x_2\le 1,\ x_1\ge 0,\ x_2\ge 0\\ 0, & \text{其它.}\end{cases}$$

将 $f(x_1,x_2)$ 写成 $f_1(x_1)f_2(x_2|x_1)$ 的形式, 其中

$$f_1(x_1)=\int_0^{1-x_1}f(x_1,x_2)dx_2=3(1-x_1)^2,\ 0\le x_1\le 1$$

$$f_2(x_2|x_1)=\frac{f(x_1,x_2)}{f_1(x_1)}=2x_2(1-x_1)^{-2},\ 0\le x_2\le 1-x_1.$$

计算边际分布函数和条件分布函数为

$$F_1(x_1) = 1 - (1 - x_1)^3,\ 0 \le x_1 \le 1$$

$$F_2(x_2|x_1) = x_2^2(1 - x_1)^{-2},\ 0 \le x_2 \le 1 - x_1.$$

因而由逆变换法, 我们可首先由 $U(0,1)$ 抽取 u_1 和 u_2, 解方程使得

$$\begin{cases} 1 - (1 - X_1)^3 = u_1 \\ X_2^2(1 - X_1)^{-2} = u_2. \end{cases}$$

则可得 $X_1 = 1 - (1 - u_1)^{1/3}$, $X_2 = (1 - u_1)^{1/3}u_2^{1/2}$.

需要注意的是由于联合密度函数分解表示不唯一(事实上有 $k!$ 种), 而不同的表示所得的方程可能是不一样的, 从而抽样的难易程度也不一样.

最后, 对于我们常用的多维正态分布, 我们举例详述如下.

例 6.1.10. 生成 $\mathbf{X} = (X_1, \ldots, X_k)^T$ 服从多元正态分布 $N_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. 由于 $\boldsymbol{\Sigma}$ 是正定阵, 故可作 Cholesky 分解, 即, 存在下三角矩阵 $\mathbf{C}$ 使得 $\boldsymbol{\Sigma} = \mathbf{C}\mathbf{C}^T$. 于是, 显然若生成 $\mathbf{Z} = (Z_1, \ldots, Z_k) \sim N_k(\mathbf{0}, \mathbf{I}_k)$, 则可得 $\mathbf{X} = \mathbf{C}\mathbf{Z} + \boldsymbol{\mu} \sim N_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. 在 R 中我们可方便地使用 chol($\boldsymbol{\Sigma}$) 以获得 Cholesky 分解.

例 6.1.11. 生成 $\mathbf{T} = (T_1, \ldots, T_k)^T$ 服从多元 t 分布. 假设 $\mathbf{X} \sim N_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ 且 S 独立地服从自由度为 v 的卡方分布, 即 χ_v^2. 则多元 t 分布 $T(\boldsymbol{\mu}, \boldsymbol{\Sigma}, v)$ 的定义为

$$\mathbf{T}_v = \frac{\mathbf{X}}{\sqrt{S/v}} + \mu$$

其位置向量为 $\boldsymbol{\mu}$, 尺度矩阵为 $\boldsymbol{\Sigma}$, 自由度为 v. 同一元时类似, 当 v 较小时, 该分布具有较厚的尾部, 我们通常可以利用这一分布对 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$ 进行稳健估计. 我们可通过分别独立地生成 $\mathbf{X}$ 和 S 来产生该分布的随机向量.

§6.2 随机模拟计算

模拟即是指把某一现实的或抽象的系统的某种特征或部分状态, 用另一系统 (称为模拟模型) 来代替或模仿. 由于模拟方法是利用随机数进行模拟计算, 故此方法称为随机模拟方法, 也成为计算机模拟方法. 该方法最早被物理学家冯诺依曼所使用, 其以著名赌城 Monte Carlo 来命名这种方法, 沿用至今. 最早的随机模拟方法的想法实际上可以追溯到我们所熟知的蒲丰实验问题, 详见概率论书, 这里不予详述. 我们下面主要介绍如何使用 Monte Carlo 来求解确定性问题.

我们这里先来回顾一下本章开始时介绍的一个使用随机模拟的用于计算临界值的例子, 在前面的描述中, 我们试图通过模拟来获取近似的临界值, 事实上, 我们直接可以通过模拟来做检验.

例 6.2.1. (Monte Carlo 检验)

考虑 Kolmogorov-Smirnov 检验

$$D = \sup_{-\infty<x<\infty} |F_n(x) - F_0(x)|.$$

我们考虑如何能够通过模拟的方法来构造近似的检验. 这就是所谓的 Monte Carlo 检验. 具体步骤为

(1) 随机生成 N 次样本容量为 n 的来自 $F_0(x)$ 的样本;

(2) 对每次模拟生成的样本计算一个 $D_i, i = 1, \ldots, N$;

(3) 用 $\{D_i\}_{i=1}^N$ 的经验分布函数来近似检验统计量的分布函数, 因此求检验的 p 值或拒绝域都可通过该经验分布函数来获得.

比如, 估计 p 值, 假设 r 是 $\{D_i\}_{i=1}^N$ 大于观测值 D^* 的个数, 则 r/N 就是 p 值的一无偏估计; 如果估计拒绝域, 即显著性水平为 α 临界点, 则为 $D_{(\lceil N(1-\alpha)\rceil)}$, 其中 $D_{(i)}$ 为样本 $\{D_i\}_{i=1}^N$ 的第 i 个次序统计量. □

例 6.2.2. (**质量控制**) 考虑一个逐个生产产品的过程, 并假设这些产品的质量指标是可测量的. 当过程处于“可控”状态时, 这些测量值（适当标准化后）来自标准正态分布. 进一步假设当过程处于“失控”状态时, 这些值的分布从标准正态变成其它分布.

为了帮助监测过程何时进入“失控”状态, 指数加权移动平均控制图是一种经常使用的方法. 设 $X_1, X_2, \ldots$ 是一列观测值, 对确定的值 λ, $0 < \lambda < 1$, 定义序列 $S_n, n \geqslant 0$, 如下：

$$\begin{aligned} S_0 &= 0, \\ S_n &= (1-\lambda)S_{n-1} + \lambda X_n, \quad n \geqslant 1. \end{aligned}$$

由于当过程处于可控状态时, 所有的 X_n 的均值都是 0, 因此, 容易证明, 在此条件下, 指数加权移动平均值 S_n 的均值也是 0. 对于给定的 λ 值, 由指数加权移动平均控制图规则确定一个常数 B, 当 $|S_n|$ 超过 B 时, 认为过程“失控”. 也就是说, 过程失控的时刻

$$N = \min\{n : |S_n| > B\}$$

是随机的.

很明显, 即使过程仍在正常运行状态（也就是说即使产生的数据仍是服从标准正态分布的数据）, $|S_n|$ 也有可能超过 B, 即认为过程失控. 于是, 为了确保这种情况不经常发生, 需要慎重地选取 λ 和 B 的值, 使得当 $X_n, n \geqslant 1$ 确实是来自标准正态分布时, $E[N]$ 的值较大. 在过程可控的条件下, 假设 800 作为 $E[N]$ 的值是可以接受的. 进一步假设, 如取 $\lambda = 0.1$ 和 $B = 0.8$, 则可使 $E[N]$ 的值大致是 800. 我们如何检验这一假设?

检验上述结论的一个方法就是进行统计模拟. 即我们产生服从标准正态的随机变量 $X_n, n \geqslant 1$ 直到 $|S_n|$ 的值超过 0.8 (在 $|S_n|$ 的定义中取 $\lambda = 0.1$). 如果用 N_1 表示直到 $|S_n|$ 的值超过 0.8 时已产生的服从正态分布的数据的个数, 则对于第一次模拟运行, 我们得到输出变量 N_1, 我们可以接着进行模拟, 则所有模拟的输出数据的平均值就是 $E[N]$ 的估计. 同理, 关于 N 的分布的其它信息我们亦可通过这样的模拟得到, 比如其分位点, 方差等; 反之, 假如我们想要知道的是 B 等于多少则 $E[N]$ 能够近似等于 800? 这时我们可以将其看为一个求方程根的问题, 即 $E[N]$ 是一个关于 B 的函数, 不妨记为 $g(B)$, 我们需要求解 $g(B) - 800 = 0$. 注意这里 $g(\cdot)$ 是需要通过上面类似的模拟来进行求值的. □

通过这两个例子我们可看出随机模拟的基本思路:

(1) 针对实际问题建立一个简单且便于实现的概率统计模型, 使所求的解恰好是所建模型的概率分布或某个数字特征, 如 f_D 的分位点 ξ_α;

(2) 对该模型的随机变量建立抽样方法, 通过计算机进行模拟实验, 抽取足够的随机数, 对有关的事件进行统计分析;

(3) 对模拟结果加以分析, 求得估计及精度;

(4) 必要时, 对模型进行改进以降低估计方差提高运算效率.

我们下面以最常用的计算定积分 $\theta = \int_a^b f(x)dx$ 为例, 介绍一些抽样方法和改进精度的方法. 很多统计问题, 如计算概率, 各阶矩等, 都可归纳为定积分的近似计算问题.

§6.2.1 样本均值法

对 $\theta = \int_a^b f(x)dx$, 设 $g(x)$ 是 (a, b) 上的一个密度函数, 改写

$$\theta = \int_a^b \frac{f(x)}{g(x)} g(x)dx = E_g\left[\frac{f(X)}{g(X)}\right].$$

可见, 任一积分均可以表示为某个随机变量 (函数) 的期望. 由矩估计法, 若有 n 个来自 $g(x)$ 的观测值, 则可给出 θ 的一个矩估计, 这就是样本均值法的原理.

特别地, 如果 (a, b) 有限, 可取 $g(x) = 1/(b-a)$. 设 $u_1, \ldots, u_n$ 是来自 $U(a, b)$ 的随机数, 则 θ 的一个估计为

$$\widehat{\theta} = \frac{1}{n}\sum_{i=1}^{n} \frac{f(u_i)}{g(u_i)} = \frac{b-a}{n}\sum_{i=1}^{n} f(u_i).$$

计算步骤如下:

(1) 独立地生成 n 个 $U(0, 1)$ 的随机数 $u_1, \ldots, u_n$;

(2) 计算 $x_i = a + (b-a)u_i$ 和 $f(u_i), i = 1, \ldots, n$;

(3) 计算估计 $\widehat{\theta}$.

由大数定律我们知道, $\widehat{\theta}$ 是 θ 的一相合估计, 亦易见其也是无偏估计. 我们也可计算其方差,

$$\begin{aligned}\mathrm{var}(\widehat{\theta}) &= \frac{1}{n}\left[(b-a)^2\int_a^b f^2(x)\frac{1}{b-a}dx - \theta^2\right]\\ &= \frac{1}{n}\left[(b-a)\int_a^b f^2(x)dx - \theta^2\right].\end{aligned}$$

一般地, 记 $k(x) = f(x)/g(x)$, 考虑估计量 $\widehat{\theta} = \frac{1}{n}\sum_{i=1}^n k(X_i)$, 其中 X_i 服从分布 $g(x)$. 则由中心极限定理知 $\widehat{\theta}$ 近似地服从正态分布, 均值 $E_g(k(X))$, 标准差 $\sqrt{\mathrm{var}_g[k(X)]/n}$. 进一步地, 实际中我们还可以用$\sqrt{v/n}$ 估计所谓的 *Monte Carlo* 标准误, 其中

$$v = \frac{1}{n-1}\sum_{i=1}^n\left[k(X_i) - \widehat{\theta}\right]^2$$

是通常所用的 $\mathrm{var}_g[k(X)]$ 的无偏估计. 通过该标准误，我们可以近似地决定所需使用的模拟次数：假设我们想要以$1-\alpha$的把握保证$\widehat{\theta}$的估计精确到误差范围是$\epsilon > 0$ 以内. 仍由极限正态分布，我们知

$$\frac{\widehat{\theta} - \theta}{\sqrt{\mathrm{var}_g[k(X)]/n}} \approx N(0,1).$$

因此我们要求$\epsilon/\sqrt{\mathrm{var}_g[k(X)]/n} \approx z_{\alpha/2}$，也就是说$n \approx z_{\alpha/2}^2\mathrm{var}_g([k(X)])/\epsilon^2$.

由上我们看到, 模拟的效率强烈依赖于 $g(\cdot)$ 的选择, 我们的问题就是, 如何选取 $g(\cdot)$ 使得 $\widehat{\theta}$ 的方差小. 我们来看下面的例子

例 6.2.3. (**Cauchy分布的尾部概率**) 假设我们感兴趣的是一个标准Cauchy分布$C(0,1)$随机变量大于2的概率, 即

$$p = \int_2^\infty \frac{1}{\pi(1+x^2)}dx.$$

一个最简单直接的方法是使用Monte Carlo估计量

$$\widehat{p}_1 = \frac{1}{n}\sum_{j=1}^n I(X_j > 2),$$

其中$X_1,\ldots,X_n$是来自于$C(0,1)$的iid样本，因为$p = \int_{-\infty}^\infty I(x>2)\frac{1}{\pi(1+x^2)}dx$. 而这一估计的方差为$p(1-p)/n$(约为$0.127/n$; $p \approx 0.15$).

一个直接的改进方法是利用$C(0,1)$的对称性，

$$\widehat{p}_2 = \frac{1}{2n}\sum_{j=1}^n I(|X_j| > 2).$$

此时我们是将p 写作$\int_{-\infty}^{\infty}\frac{1}{2}I(|x|>2)\frac{1}{\pi(1+x^2)}dx$. 显然$\widehat{p}_2$仍是无偏的，且不难验证此时的方差为$p(1-2p)/2n=0.052/n$.

上述两种方法的效率通常无法令人满意，主要原因是那些落在我们感兴趣的区域外的随机变量从某种角度来说对于估计p没有起到太大作用. 假如我们重新将p记作

$$p=\frac{1}{2}-\int_0^2\frac{1}{\pi(1+x^2)}dx,$$

则上面的积分可看作是$h(X)=2/\pi(1+x^2)$的期望，其中$X\sim U(0,2)$. 那么自然地我们可以得到一个新的估计

$$\widehat{p}_3=\frac{1}{2}-\frac{1}{n}\sum_{j=1}^{n}h(U_j),$$

其中$U_j\sim U(0,2)$. 此时$\widehat{p}_3$的方差为$[E(h^2)-E^2(h)]/n\approx 0.0285/n$.

最后，一种效率更高但不是非常直接的方法是记

$$p=\int_0^{1/2}\frac{y^{-2}}{\pi(1+y^{-2})}dy,$$

上面的积分可看作是$4^{-1}h(Y)=1/2\pi(1+y^2)$的期望，其中$Y\sim U(0,1/2)$. 相应地，

$$\widehat{p}_4=\frac{1}{4n}\sum_{j=1}^{n}h(Y_j),$$

其中$Y_j\sim U(0,1/2)$. 此时$\widehat{p}_4$的方差约为$0.95\times 10^{-4}/n$.

§6.2.2 重要抽样方法 (Importance Sample)

从理论上看, 因 $\mathrm{var}(\widehat{\theta})=\frac{1}{n}\left[E\left(\frac{f(X)}{g(X)}\right)^2-\theta^2\right]$, 若 $f(x)\geq 0$, 取 $g(x)=f(x)/\theta$, 则有 $\mathrm{var}(\widehat{\theta})=0$. 更一般地, 由 Cauchy-Schwarz 不等式我们有

$$\begin{aligned}\int\frac{f^2(x)}{g(x)}dx&=\int\frac{f^2(x)}{g(x)}dx\int g(x)dx\\&\geq(\int|f(x)|dx)^2.\end{aligned}$$

因此, 显然我们可取 $g(x)=\frac{|f(x)|}{\int|f(x)|dx}$ 以达到$\widehat{\theta}$ 方差的下界. 一般来说, 由于我们欲求的是 $\theta=\int f(x)dx$, 所以 $\int|f(x)|dx$ 通常是未知的, 那么这样取下解是无法实现的. 然而, 它也给我们一个启示, 即, $g(x)$ 与 $f(x)$ 形状接近应能降低估计的方差. 这便是重要抽样法的基本思想.

直观地来看, 使用均匀分布给出的 $\widehat{\theta}$, 各 $f(u_i)$ 对 $\widehat{\theta}$ 的贡献是不同的, $f(u_i)$ 大则贡献大, 但在抽样时, 这种差别未能体现出来, 因此效率不会很高, 要达到同样的精度需要的抽样次数多. 而重要抽样法则希望贡献率大的随机数出现的概率大, 贡献小的随机数出现的概率小, 从而提高抽样效率.

例 6.2.4. 考虑一个简单积分 $\theta = \int_0^1 e^x dx$. 此处 θ 的精确值 $e-1$ 是可求的, 为说明重要抽样法, 我们采用 Monte Carlo 方法估计 θ.

首先考虑样本均值法, 即产生 n 个 $U(0,1)$ 随机数 $u_1, \ldots, u_n$, 则 $\widehat{\theta}_1 = \frac{1}{n}\sum_{i=1}^n e^{u_i}$. 我们知道 $\widehat{\theta}_1$ 无偏, 且方差 $\mathrm{var}(\widehat{\theta}) \approx \frac{0.242}{n}$.

由重要抽样法的思想, 我们要选一个与 e^x 相似的密度函数. 由 Taylor 展开式, $e^x = 1 + x + \frac{x^2}{2} + \cdots$, 取 $g(x) = \frac{2}{3}(1+x)$ 为一密度函数(即取线性近似, 前面的系数是归一化常数). 设 $x_1, \ldots, x_n$ 是来自 $g(x)$ 的随机数 (产生均匀随机数 $u_1, \ldots, u_n$, 由逆变换法得 $x_i = \sqrt{3u_i + 1} - 1$ 则 $x_i \sim g(x)$), 则 θ 的估计可取

$$\widehat{\theta}_2 = \frac{1}{n}\sum_{i=1}^n \frac{f(x_i)}{g(x_i)} = \frac{3}{2n}\sum_{i=1}^n \frac{e^{x_i}}{1+x_i}.$$

当然 $\widehat{\theta}_2$ 也是无偏估计, 其方差$\mathrm{var}(\widehat{\theta}_2) \approx \frac{0.0269}{n}$. 显然 $\widehat{\theta}_2$ 明显优于 $\widehat{\theta}_1$. □

例 6.2.5. (**小尾部概率的模拟**) 假设我们感兴趣的是近似标准正态分布$N(0,1)$的尾部概率, 也就是

$$1 - \Phi(t) = \int_t^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy.$$

我们自然地通过Monte Carlo 估计$n^{-1}\sum_{j=1}^n I(X_i \geq t)$. 如前面的例6.2.3所述，该方法的效率不高, 而更为严重的问题是, 如果t较大，比如取4.5，我们知道$P(Z > 4.5)$值很小, 则如果我们直接使用上面的模拟方法，重复10,000次，通常来说我们会得到所有的示性函数$I(X_i > 4.5)$均为零. 这种问题当然是由我们现在所关心的事件就是以小概率事件所导致的，对于这种问题，我们一般都是需要更大的模拟次数才能够完成的. 但是，如果我们能够适当地结合重要抽样法，则会极大程度地改进我们的估计. 我们令$Y \sim TE(4.5, 1)$，其中TE表示在4.5处左截断的指数分布, 密度为

$$g_Y(y) = e^{-(y-4.5)},\ y > 4.5.$$

我们预期$g(y)$的形状与$\phi(y)$差不多(指数衰减), 则我们从我们从$g_Y(y)$中抽取n个随机变量$Y_1, \ldots Y_n$，使用

$$\frac{1}{n}\sum_{j=1}^n \frac{\phi(Y_j)}{g_Y(Y_j)},$$

而这里$g_Y(y)$的抽取只需先抽取一来自Exp(1)的随机样本Z，令$Y = Z + 4.5$即可(为什么？)

§6.2.3 Rao-Blackwellization 方法

该方法的得名是来自于著名的Rao-Blackwell定理：记W是参数$\tau(\theta)$的任一无偏估计, 而T是θ的充分统计量. 则$g(T) = E(W \mid T)$是一个一致优于W的无偏估计. 这里主要用到了两个基本等式：假设X和Y是两个随机变量,

$$EX = E[E(X|Y)], \quad \mathrm{var}X = \mathrm{var}[E(X|Y)] + E[\mathrm{var}(X|Y)].$$

直接地, 如果$\delta(X)$是一个传统的Monte Carlo估计量, 而$\delta(X)$在给定某一随机变量Y的条件下的期望, $\delta^*(Y) = E(\delta(X)|Y)$, 可较为容易求得则其是一个优于$\delta$的估计（从平方损失来看, 方差小, 而偏差均为零）. 当然这里注意的是Y没有必要是充分统计量.

例 6.2.6. (筛选抽样法中的应用)我们来考虑使用Monte Carlo方法计算期望

$$E(f(x)) = \int f(x)g(x)dx,$$

其中$g(x)$是我们欲抽样的密度函数. 相应地, 最直接地Monte Carlo方法就是抽取m个来自于$g(x)$的随机变量$Z_1, \ldots, Z_m$, 然后用$m^{-1}\sum_{i=1}^m f(Z_i)$作为估计. 而假设我们不容易从$g(x)$中直接抽样, 现考虑能够使用筛选抽样法, 也就是我们能够找到一个较为容易抽样的密度函数$h(x)$使得$g(x) \le ch(x)$, 则我们从$h(x)$中抽取随机数X_i并独立地抽取均匀随机数U_i, 如果$U_i \le W_i = g(X_i)/[ch(X_i)]$, 则接受$X_i$, 否则拒绝$X_i$. 相应地, 假设我们总共想要抽取$m$个随机数来估计$E(f(x))$, 而使用筛选抽样法共抽取了$n$个$X_1, \ldots, X_n$来达到这一目的（$n$个里面有$m$个接受了, 而最后一个是接受的）, 此时Monte Carlo估计变为

$$\frac{1}{m}\sum_{i=1}^n I(U_i \le W_i)f(X_i).$$

当然, 在这一过程中没有被接受的X_i的信息就完全被舍去了, 因为它们对上面的估计的贡献是0. 当接受率较低的时候, 上面的模拟方法不够有效. 现考虑可否利用Rao-Blackwellization方法进行改进. 此时考虑条件期望

$$\frac{1}{m}E[\sum_{i=1}^n I(U_i \le W_i)f(X_i) \mid n, X_1, \ldots, X_n]$$

$$= \frac{1}{m}E[\sum_{i=1}^n I(U_i \le W_i) \mid n, W_1, \ldots, W_n]f(X_i),$$

其仍然是无偏估计而降低了抽样方差（这一降低显然是由于我们利用了全部的抽样信息所致）. 此时我们的问题转化为求取这一条件期望, 而事实上也就是一个条件概率$p_i = P(U_i \le W_i \mid n, W_1 = w_1, \ldots, W_n = w_n)$, 也就是给定条件：成功的概率是$W_i = w_i$以及$n$次试验中共有$m$次成功而最后一次是成功的, 第$i$次抽样$X_i$被接受的概率. 假设$q_{jk}$是成功概率分别为$w_1, \ldots, w_k$的$k$次Bernoulli试验共有$j$次成功的概率, $q_{jk}^{(i)}$是上述事件与事件第i次试验是成功的交集的概率. 注意到共有m次成功的而最后一次是成功的, 因此

$$p_n = 1,\ p_i = q_{m-1,n-1}^{(i)}/q_{m-1,n-1}, 1 \le i < n.$$

q_{jk}和$q_{jk}^{(i)}$的计算均可迭代进行. 设置初值$q_{0,1} = 1 - w_1, q_{1,1} = w_1$, q_{jk}的迭代如下：

$$q_{0,k} = (1 - w_k)q_{0,k-1},$$

$$q_{j,k} = w_k q_{j-1,k-1} + (1 - w_k)q_{j,k-1}, \quad 1 \le j \le k-1,$$

$$q_{k,k} = w_k q_{k-1,k-1}.$$

而计算联合概率$q_{jk}^{(i)}$可先设置初值$q_{ji}^{(i)} = w_i q_{j-1,i-1}$, 通过下式迭代完成

$$q_{jk}^{(i)} = w_k q_{j-1,k-1}^{(i)} + (1 - w_k) q_{j,k-1}^{(i)},\ k > i.$$

§6.2.4 分层抽样法

另一种利用贡献率大小来降低估计方差的方法是分层抽样法 (stratified sampling). 它首先把样本空间分成一些小区间 $D_1, \ldots, D_m$, 且 D_i 互不相交, $\bigcup D_i = D$, 然后在个小区间内的抽样数由其贡献大小决定, 即, 定义 $p_i = \int_{D_i} f(x)dx$, 则, D_i 内的抽样数应与 p_i 成正比. 比如, 对 θ 贡献大的 D_i 抽样多, 则可提高抽样效率.

具体来说, 仍考虑积分 $\theta = \int_0^1 f(x)dx$, 将 $[0,1]$ 分成 m 个小区间, 各区间断点记为 $a_i, 0 = a_0 < a_1 < \cdots < a_m = 1$, 则

$$\theta = \int_0^1 f(x)dx = \sum_{i=1}^m \int_{a_{i-1}}^{a_i} f(x)dx \equiv \sum_{i=1}^m I_i.$$

记 $l_i = a_i - a_{i-1}$ 为第 i 个区间的长度, 在每个小区间上的积分值 I_i 可用平均值法估计出来, 然后将之相加即可给出 θ 的一个估计. 具体步骤描述如下:

(1) 产生 $U(0,1)$ 的随机数 $u_{ij}, j = 1, \ldots, n_i,\ i = 1, \ldots, m$;

(2) 计算 $x_{ij} = a_{i-1} + l_i u_{ij}, j = 1, \ldots, n_i,\ i = 1, \ldots, m$;

(3) 计算 $\widehat{I}_i = \frac{l_i}{n_i} \sum_{j=1}^{n_i} f(x_{ij})$;

(4) $\widehat{\theta} = \sum_{i=1}^m \widehat{I}_i$.

显然由于 $E(\widehat{I}_i) = I_i$, 因而 $\widehat{\theta}$ 是 θ 的无偏估计, 其方差为

$$\mathrm{var}(\widehat{\theta}) = \mathrm{var}\left\{ \sum_{i=1}^m \frac{l_i}{n_i} \sum_{j=1}^{n_i} f(x_{ij}) \right\} = \sum_{i=1}^m \frac{l_i^2}{n_i} \sigma_i^2, \tag{6.2}$$

其中

$$\sigma_i^2 = \int_{a_{i-1}}^{a_i} \frac{f^2(x)}{l_i} dx - \left(\frac{I_i}{l_i} \right)^2. \tag{6.3}$$

分层法有两个主要问题: 其一是怎样划分抽样区间, 简单而常用的方法是将抽样区间等分; 另一个问题是, 在抽样区间划分好后, 如何确定抽样次数. 不难证明, 在 n 固定下, 当

$$n_i = n l_i \sigma_i / \sum_{i=1}^m l_i \sigma_i \tag{6.4}$$

时, (6.2) 所得到的估计方差达到最小, 为 $\frac{1}{n}(\sum_{i=1}^{m} l_i\sigma_i)^2$. 但是由于 σ_i^2 总是未知的, 因而 (6.4) 通常无法直接应用. 即便如此, 分层抽样法还是有其作用的. 这里指出, 即使取简单的分配: $n_i = nl_i/\sum_i l_i = nl_i/(b-a)$, 也有 $\mathrm{var}(\widehat{\theta}_s) \le \mathrm{var}(\widehat{\theta}_m)$, 这里 $\widehat{\theta}_s$ 和 $\widehat{\theta}_m$ 分别表示用分层抽样法和平均值法得到的估计.

事实上, 我们以 $n_i = nl_i/(b-a)$ 代入 (6.2), 有

$$\mathrm{var}(\widehat{\theta}_s) = \frac{b-a}{n}\sum_{i=1}^{m} l_i\sigma_i^2.$$

又由 Cauchy-Schwarz 不等式, 有

$$\begin{aligned}\theta^2 = (\sum_{i=1}^{m} I_i)^2 &= \left(\sum_{i=1}^{m} \frac{I_i}{\sqrt{l_i}}\sqrt{l_i}\right)^2 \\ &\le \sum_{i=1}^{m}\frac{I_i^2}{l_i}\sum_{i=1}^{m} l_i = (b-a)\sum_{i=1}^{m}\frac{I_i^2}{l_i}.\end{aligned}$$

据此, 在 (6.3) 式中两端各乘以 l_i 并相加, 即有

$$\begin{aligned}\sum_{i=1}^{m} l_i\sigma_i^2 &= \int_a^b f^2(x)dx - \sum_{i=1}^{m}\frac{I_i^2}{l_i} \\ &\le \int_a^b f^2(x)dx - \theta^2/(b-a),\end{aligned}$$

即 $\mathrm{var}(\widehat{\theta}_s) \le \mathrm{var}(\widehat{\theta}_m)$.

例 6.2.7. 我们再看例6.2.4. 简单起见, 我们将区间 [0,1] 分为 $[0, 0.5]$ 和 $[0.5, 1]$, 并且使用简单的分配, 即在两个小区间内各使用 $n/2$ 次抽样, 则由分层抽样的估计方差公式得

$$\mathrm{var}(\widehat{\theta}_s) = \frac{1}{n}\sum_{i=1}^{2}\frac{1}{2}\sigma_i^2 \approx \frac{1}{2n}(0.03492 + 0.09493) = \frac{0.06492}{n}$$

与例 6.2.4 中平均值法的结果相比较, 即便是最简单的选取区间和抽样数都能够显著地减小估计的方差. □

§6.2.5 关联抽样法

假设我们对用模拟方法估计 $\theta = E[X]$ 感兴趣, 且假设已产生均值是 θ 的同分布的随机变量 X_1 和 X_2, 则

$$\mathrm{Var}\left(\frac{X_1+X_2}{2}\right) = \frac{1}{4}[\mathrm{Var}(X_1) + \mathrm{Var}(X_2) + 2\mathrm{Cov}(X_1, X_2)].$$

因此, 如果 X_1 和 X_2 不是独立的, 而是负相关, 则上式对于减小方差是有用的.

如何安排 X_1 和 X_2 以使它们负相关呢? 下面的结论给出了一个达到负相关的充分条件: 如果 X 是一个随机变量, $f(x)$ and $g(x)$ 同是单增或单减的函数. 如果随机变量 $f(X)$ 和 $g(X)$ 的二阶矩存在, 则

$$\mathrm{Cov}(f(X), g(X)) \geq 0.$$

如果 $f(x)$ 单增而 $g(x)$ 单减, 或者反之, 则上面的不等号变为"$\leq$".

证明: 考虑与 X 同分布的随机变量 Y. 若 $f(x)$ and $g(x)$ 同是单增或单减的函数, 则 $[f(X)-f(Y)][g(X)-g(Y)] \geq 0$. 因此,

$$\begin{aligned} 0 &\leq E\{[f(X)-f(Y)][g(X)-g(Y)]\} \\ &= E[f(X)g(X)] + E[f(Y)g(Y)] - E[f(X)]E[g(Y)] - E[f(Y)]E[g(X)] \\ &= 2\mathrm{Cov}[f(X), g(X)]. \end{aligned}$$

显然, 若 $f(x)$ 单增而 $g(x)$ 单减, 上面的证明只需要将符号改变一下即可.

例 6.2.8. (对偶均匀随机变量法) 考虑估计 $\theta = \int_a^b h(x)g(x)dx$, 其中 $g(x)$ 是一密度函数, 其分布函数为 $G(x)$. $h(x)$ 相当于前述的 $f(x)/g(x)$, 这里假设其是一单调函数. 如果 $U_1, \ldots, U_n$ 是来自 $U(0,1)$ 的独立随机数, 则 $1-U_1, \ldots, 1-U_n$ 也是来自 $U(0,1)$ 的独立随机数. 我们令 $X_i = G^{-1}(U_i)$, $X_i^* = G^{-1}(1-U_i)$, 则 $h(X_i)$ 和 $h(X_i^*)$ 都是 θ 的无偏估计, 根据上面的结论, 我们知道他们是负相关的, 因此估计量

$$\frac{1}{2n}\sum_{i=1}^{n}\{h(X_i) + h(X_i^*)\}$$

的方差比样本平均值法 $\frac{1}{2n}\sum_{i=1}^{2n} h(X_i)$ 要小.

注意到, 在这种情况下, 在利用 $U_1, \ldots, U_m$ 计算出 X_i 之后, 我们可以利用 $1-U_1, 1-U_2, \ldots, 1-U_m$ 计算出 X_i, 这将比使用新产生的 n 个随机数后用样本均值法的效果来得好 (估计的方差小). 并且, 我们还可以收到事半功倍的效果: 不但估计结果的方差较小（至少在 h 是单调函数时）, 而且可以节省下生成第二个随机数集的时间.

更一般地，假设X_i 是来自某一对称中心为μ的对称分布的随机变量，则一般我们可使用

$$\frac{1}{2n}\sum_{i=1}^{n}[h(X_i) + h(2\mu - X_i)],$$

来代替$\frac{1}{2n}\sum_{i=1}^{2n} h(X_i)$, 很多情况下都会得到较大的改进.

§6.3 Bootstrap 法

§6.3.1 Bootstrap 的基本原则

令 $\theta = T(F)$ 为我们所感兴趣的关于分布函数 F 的某一特征, 被表示为 F 的一函数. 比

如, $T(F)=\int zdF(z)$ 是分布的期望. 令 $X_1,\ldots,X_n$ 为观测数据. 在本节中, 我们用 $X\sim F$ 表示 X 服从密度函数为 f 的分布, 其对应的累积分布函数为 F. 令 $\mathbf{X}=\{X_1,\ldots,X_n\}$ 表示整个数据集.

如果 $\widehat{F}_n$ 是观测数据的经验分布函数, 则 θ 的一估计为 $\widehat{\theta}=T(\widehat{F}_n)$. 比如, 当 θ 是一元总体均值, 则估计就是样本均值, $\widehat{\theta}=\int zd\widehat{F}_n(z)=\sum_{i=1}^n X_i/n$. 统计推断的问题通常是根据 $T(F)$ 或某个 $R(\mathbf{X},F)$ 提出来的, 这里 $R(\mathbf{X},F)$ 是依赖于数据和它们的未知分布函数 F 的统计函数. 举例来说, 一个一般的检验统计量可以为 $R(\mathbf{X},F)=[T(\widehat{F}_n)-T(F)]/S(\widehat{F}_n)$, 其中 S 为估计 $T(\widehat{F}_n)$ 的标准差的函数. 这就是我们经常建立的某种枢轴量.

随机变量 $R(\mathbf{X},F)$ 的分布可能难以处理或者根本就是未知的. 这个分布也许也依赖于未知分布 F. *Bootstrap* 方法提供了关于 $R(\mathbf{X},F)$ 的分布的一种近似, 其是由观测数据的经验分布函数 (本身是 F 的估计) 所导出的. 该方法是美国当代著名统计学家 Efron 在 1979 年的 Annals of Statistics 提出来的, 发表后引起很大反响, 许多学者针对不同模型不同问题对该方法作出了大量的讨论, 其可看作是近三十年来最富有统计思想、最实用的工作之一. 由于该方法中需要对观测数据进行重抽样, 因此经常被看作是一种计算统计的方法.

记 $R(\mathbf{X},F)$ 的分布为 G_n, 与大样本方法对 G_n 进行极限逼近不同, Bootstrap 法着眼于模拟 G_n. 但样本是从 F 中抽出的, 且 $R(\mathbf{X},F)$ 中经常含有关于 F 的量, 如上面枢轴量中的 $T(F)$, 如何做呢? 一个方法就是用 $X_1,\ldots,X_n$ 的经验分布函数来代替未知的 F. 由 Glivenko 定理, 当 n 很大时二者很接近. 则我们以 $\widehat{F}_n$ 为总体分布, 从中抽出 iid 样本 $\mathbf{X}^*=\{X_1^*,\ldots,X_n^*\}$. 这里 $\mathbf{X}^*$ 表示一*伪-数据 Bootstrap 样本* (pseudo-Bootstrap sample). 注意 $\mathbf{X}^*$ 的元素是独立同分布于 $\widehat{F}_n$ 的随机变量. Bootstrap的思想就是考察 $R(\mathbf{X}^*,\widehat{F}_n)$ 的分布, 也就是在 R 中使用 $\mathbf{X}^*$ 所得到的随机变量的分布, 不妨记为 G_n^*, 并用该分布作为 G_n 的近似. 在某些特殊情况下, 我们有可能通过解析方法推导或估计 $R(\mathbf{X}^*,\widehat{F}_n)$ 但是通常所使用的是后面所描述的模拟方法.

§6.3.2 非参数和参数化 Bootstrap

我们可从观测数据的经验分布函数中随机抽取 B 个独立的 Bootstrap 伪-数据集. 将他们定义为 $\mathbf{X}_i^*=\{X_{i1}^*,\ldots,X_{in}^*\}, i=1,\ldots,B$. 之后, 计算基于这 B 个的 R 值, 即

$$\{R_1^*,\ldots,R_B^*\}\equiv\{R(\mathbf{X}_1^*,\widehat{F}_n),\ldots,R(\mathbf{X}_B^*,\widehat{F}_n)\}.$$

则 $\{R_1^*,\ldots,R_B^*\}$ 的经验分布函数, 记为 $\widehat{G}_B^*$, 可用于近似 $R(\mathbf{X}^*,\widehat{F}_n)$ 的分布 G_n^* 进而近似 G_n, 并进行推断. 但误差相应产生, 这里用 $\widehat{G}_B^*$ 近似 G_n 有两种误差, 一是 $\widehat{G}_B^*$ 逼近 G_n^* 的误差, 注意因为 $\widehat{G}_B^*$ 的分布依赖于样本 $\mathbf{X}$, 这误差本身就是随机的, 但由于 B 可以去非常大, 这一误差可使之任意小 (只要适当增大 B 即可), 因而不是主要的; 另一个是 G_n^* 逼近 G_n 的误差. 在适当条件下, 一般可以证明 G_n^* 和 G_n 有相同的极限. 因此, 只要 n 足够大, 这个误差也可很小.

Bootstrap 使得我们的分析和推断不需要进行参数假设, 它为那些不大可能得到解析方案的问题提供了解决方法, 并且可以给出比应用传统标准参数理论 (尤其是基于统计大样本近

似方法) 所得到的结果更加精确的回答.

前面所描述的典型的非参数 Bootstrap 方法是从 $\widehat{F}_n$ 中抽取独立同分布的 $X_1^*,\ldots,X_n^*$ 来生成伪-数据集 $\mathbf{X}^*$. 当数据可以被模型化或者本身就是来自于一参数分布的时候, 即 $X_1,\ldots,X_n \sim$ i.i.d. $F(x,\boldsymbol{\theta})$, 我们可采用 F 的另一种 Bootstrap 抽样方法. 假设我们通过观测数据来获得 $\widehat{\boldsymbol{\theta}}$ 以估计 $\boldsymbol{\theta}$. 则我们可通过抽取 $X_1^*,\ldots,X_n^* \sim$ i.i.d. $F(x,\widehat{\boldsymbol{\theta}})$ 来生成*参数化 Bootstrap* 伪-数据集 $\mathbf{X}^*$. 当模型已知或者我们有信心模型能够很好地匹配数据的时候, 参数化 Bootstrap 将会成为一个强有力的工具, 它能够对那些难以处理的问题给出推断, 并且其产生的置信区间会比用标准极限理论或者是前述的非参数 Bootstrap 所得到的精确很多. 当然, 需要指出的是, 如果模型不能够很好的拟合数据的生成机制, 参数化 Bootstrap 方法就会得到错误的推断结果.

§6.3.3 几个常见的应用

1. **Bootstrap 方差估计**

   在 Efron (1979) 年的文章中, 为了引出 Bootstrap, 考虑计算样本中位数 $\widehat{\xi}_{\frac{1}{2}}$ 的方差. 经典的极限理论告诉我们

$$\mathrm{var}[\widehat{\xi}_{\frac{1}{2}}] = \frac{1}{4nf^2(\xi_{\frac{1}{2}})} + o(n^{-1}).$$

   然而, $f(x)$ 通常很难被估计准确, 即使是在大样本的情形下. Bootstrap 方法则为我们提供了直接进行方差估计而避免去估计 $f(x)$.

   一般地, 假设我们想要估计的是某估计 $T(\widehat{F}_n)$ 的方差, 根据前面介绍的 Bootstrap 的基本思想, 我们自然是通过求取

$$\widehat{\mathrm{var}}(T) = \frac{1}{B-1}\sum_{i=1}^{B}[T(\widehat{F}_{ni}^*) - \bar{T}^*]^2,$$

   其中 $\widehat{F}_{nj}^*$ 表示第 j 个 Bootstrap 伪-数据集的经验分布, $\bar{T}^*$ 是基于伪样本 $\{\mathbf{X}_1^*,\ldots,\mathbf{X}_B^*\}$ 的 $T(\widehat{F}_{ni}^*)$ 的样本均值, 即 $\bar{T}^* = \frac{1}{B}\sum\limits_{i=1}^{B} T(\widehat{F}_{ni}^*)$. 一般来说, 使用 $B = 500$ 已可达到令人满意的结果.

   注意到, Bootstrap 不是对于所有问题都能起到效果的. 通常来说, 如果我们感兴趣的是 $T(\widehat{F}_n)$ 的某些可以通过样本 $\mathbf{X}$ 直接进行估计的数字特征, 则是否使用 Bootstrap 所得结果没有区别. 比如说, 如果感兴趣的 $T(\widehat{F}_n)$ 是样本均值, 则上面的 Bootstrap 的方差估计将收敛到 $\widehat{\mathrm{var}}(X)/n$（$B\to\infty$）, 其中 $\widehat{\mathrm{var}}(X)$ 是 $\mathbf{X}$ 的样本方差 (有偏的). 在这种情况下, Bootstrap 没有带来更多的帮助. 事实上, 假设 $\bar{X}_i^*$ 是一个 Bootstrap 估计, 则

$$\widehat{\mathrm{var}}(\bar{X}) = \mathrm{var}_*(\bar{X}_i^*) = \frac{1}{n^2}\sum_{i=1}^{n}\frac{1}{n}\sum_{i=1}^{n}(X_i-\bar{X})^2 = \widehat{\mathrm{var}}(X)/n.$$

2. **Bootstrap 偏差修正**

当 $T(F)=\theta$ 时, 在 Bootstrap 分析中我们感兴趣的量是 $R(\mathbf{X},F)=T(\widehat{F}_n)-T(F)$. 这个量代表的是 $T(\widehat{F}_n)=\widehat{\theta}$ 的估计误差, 其均值即是偏差 $E\{\widehat{\theta}\}-\theta$. 这个偏差的 Bootstrap 估计是 $\bar{\theta}^*-\widehat{\theta}$, 其中 $\bar{\theta}^*=\sum_{j=1}^{B}T(\widehat{F}_{ni}^*)/B$.

3. **Bootstrap 置信区间**

一般来说, 一种常用的构造参数 θ 置信区间的方法是找到某一个（枢轴）量, 其中包含 θ 以及某一个统计量 T, 不妨记为 $g(T,\theta)$. 之后重新安排使得我们有如下的概率等式

$$P\left(g_{\alpha/2}\le g(T,\theta)\le g_{1-\alpha/2}\right)=1-\alpha.$$

当（枢轴）量的分布不易求取的时候, 我们可以通过 Bootstrap 的方法来估计或者说近似分位点 $g_{\alpha/2}$ 和 $g_{1-\alpha/2}$.

而对于构造期望等参数的置信区间问题,（枢轴）量经常可以写成形如 $T-\theta$ 的形式. 这种情况下上面的概率表达式就是我们所常见的形式

$$P\left(T-g_{1-\alpha/2}\le\theta\le T-g_{\alpha/2}\right)=1-\alpha.$$

最简单的方法便是利用 T^*-T_0 的抽样分布作为 $T-\theta$ 的分布的近似, 其中这里我们用 T^* 和 T_0 分别表示利用 Bootstrap 伪样本和观测数据所计算得到的 T 值. 也就是说, 我们使用 $g(T^*,T_0)$ 代替 $g(T,\theta)$. 当 T^*-T_0 的抽样分布不易导出时, 我们即可采用 Bootstrap 模拟来近似之. 这就是如下的 Bootstrap percentile 置信区间方法

Bootstrap percentile interval

假设 $T(\widehat{F}_n)$ 为 θ 的一点估计, 我们可从观测数据的经验分布函数中随机抽取 B 个独立的 Bootstrap 伪-数据集. 将他们定义为 $\mathbf{X}_i^*$. 之后, 计算基于这 B 个的 $T(\widehat{F}_n^*)-T(\widehat{F}_n)$ 值, 如果使用我们前面的符号即是 $\{R_1^*,\ldots,R_B^*\}$. 那么利用 $\{R_1^*,\ldots,R_B^*\}$ 的经验分布函数 $\widehat{G}_B^*$, 可求得 $\left(g_{\alpha/2}^*,g_{1-\alpha/2}^*\right)$, 其中 g_p^* 是 $\{R_1^*,\ldots,R_B^*\}$ 的第 $\lceil Bp\rceil$ 个次序统计量. 则按照前述的 Bootstrap 思想, $\left(T(\widehat{F}_n)-g_{1-\alpha/2}^*,T(\widehat{F}_n)-g_{\alpha/2}^*\right)$ 即为我们欲求的区间估计的一种近似.

Bootstrap-t interval 一些经验指出, 在正态假设下所推导出来的基于 t 统计量的置信区间通常在分布是非正态的时候也能有较好的效果. 因此, 对于位置参数的区间估计我们常常可以用如下的近似区间

$$\left[\bar{X}-t_{1-\alpha/2}(n-1)s/\sqrt{n},\bar{X}-t_{\alpha/2}(n-1)s/\sqrt{n}\right],$$

这里 s^2 是通常的样本方差（无偏的）.

形如上式, 我们称下面这种类型的置信区间为 *Bootstrap-t interval*,

$$\left[T(\widehat{F}_n)-\widehat{t}_{1-\alpha/2}S(\widehat{F}_n),T(\widehat{F}_n)-\widehat{t}_{\alpha/2}S(\widehat{F}_n)\right],$$

其中 $\widehat{t}_p$ 是从标准化的统计量

$$R(\mathbf{X}^*, \widehat{F}_n) = \frac{T(\widehat{F}_n^*) - T(\widehat{F}_n)}{S(\widehat{F}_n^*)}$$

中所估计出来的分位点. 这个置信区间实际上就是通过考虑用 Bootstrap 重抽样样本所得到的 $R(\mathbf{X}^*, \widehat{F}_n)$ 的分布 G_n^* 来近似

$$R(\mathbf{X}, F) = \frac{T(\widehat{F}_n) - T(F)}{S(\widehat{F}_n)}$$

的分布 G_n.

很多问题下, $S(\widehat{F}_n^*)$ 都没有简单的表达式, 我们自然也可以用前面介绍的 Bootstrap 方差估计来近似之. 注意 $\widehat{t}_p$ 是需要通过 Bootstrap 由标准化的统计量来重抽样得到的, 因此此时两个 Bootstrap 层叠在一起, 这时计算量相对会大很多.

注意到, 前面关于 Bootstrap 构造置信区间的讨论与假设检验也密切相关. 一个原假设下的参数值落在置信度为 $(1-\alpha)100\%$ 的置信区间外, 则以 α 的显著性水平拒绝原假设. 当然我们亦可类似地计算出基于 Bootstrap 的 p-值 估计.

§6.3.4 基于回归模型的 Bootstrap 方法

考虑如下典型多重回归模型, $Y_i = \boldsymbol{x}_i^T\boldsymbol{\beta} + \epsilon_i, i = 1, \ldots, n$, 其中假设 ϵ_i 是均值为零方差为常数的独立同分布随机变量. 这里, $\boldsymbol{x}_i$ 和 $\boldsymbol{\beta}$ 分别是 p-维的协变量和参数. 一种简单但是错误的 Bootstrap 方法描述如下. 我们从响应值集合中重抽样来构成一个新的伪-数据, 也就是对于每一个观测的 $\boldsymbol{x}_i$ 有 Y_i^*, 从而可得到一个新的回归数据集. 然后可以由这些伪-数据来计算 Bootstrap 参数向量估计 $\widehat{\boldsymbol{\beta}}^*$. 重复重抽样和估计的步骤很多次后, $\widehat{\boldsymbol{\beta}}^*$ 经验分布可用于推断 $\boldsymbol{\beta}$. 这样做错误的原因是 $Y_i|\boldsymbol{x}_i$ 不是独立同分布的—它们具有不同的边际分布. 因此, 用这种方生成 Bootstrap 回归数据集是不恰当的.

为了确定一个正确的 Bootstrap 方法, 我们必须要找到合适的独立同分布的变量. 模型中的 ϵ_i 是独立同分布的. 因此, 更恰当的策略是如下所描述的 *Bootstrap 残差法*.

我们先由观测数据拟合回归模型, 然后获得响应 $\widehat{Y}_i$ 和残差 $\widehat{\epsilon}_i$. 从拟合残差集合中有放回随机抽取得到 Bootstrap 残差集合 $\widehat{\epsilon}_1^*, \ldots, \widehat{\epsilon}_n^*$. (注意实际上 $\widehat{\epsilon}_i^*$ 不是独立的, 尽管通常来说它们近似独立.) 生成一个伪-响应 Bootstrap 集合, $Y_i^* = \widehat{Y}_i + \widehat{\epsilon}_i^*, i = 1, \ldots, n$. 对 $\boldsymbol{x}$ 回归 Y^* 从而获得 Bootstrap 参数估计 $\widehat{\boldsymbol{\beta}}^*$. 重复多次该过程可得到 $\widehat{\boldsymbol{\beta}}^*$ 的经验分布函数, 然后我们用它进行推断. 由线性模型知识我们知道, 若采用 $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ 则所得到的 $\widehat{\boldsymbol{\beta}}^*$ 定满足$E(\widehat{\boldsymbol{\beta}}^*) = \boldsymbol{\beta}$.

对于设计好的实验或者 $\boldsymbol{x}_i$ 值是预先固定的数据, 即所谓的 fixed design 这种方法是最适合的. 对于其他模型, 如 AR(1), 非参数回归, 和广义线性模型的简单 Bootstrap 方法的核心都

是 Bootstrap 残差的策略.

Bootstrap 残差法依赖于选定的模型是否能够给予观测数据适当的拟合, 以及残差具有常数方差的假设. 如果我们对这些条件的成立没有足够信心的话, 我们则可能需要其它的 Bootstrap 方法.

假设数据是从一观察研究中得到的, 其中响应变量和协变量都是从一群个体中随机选出并测量得到的. 在这种情形下, 我们可将数据 $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ 视作是从响应–协变量联合分布中得到的随机变量 $\boldsymbol{Z}_i = (\boldsymbol{X}_i, Y_i)$ 的观测值. 为了 Bootstrap, 随机有放回地从观测数据 $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ 中抽取样本 $\boldsymbol{Z}_1^*, \ldots, \boldsymbol{Z}_n^*$. 对所得到的伪随机数据集拟合回归模型以获得 Bootstrap 参数估计 $\widehat{\boldsymbol{\beta}}^*$. 多次重复这些步骤, 然后如第一种方法中介绍的进行推断. 这种情形的 Bootstrap 方法有时也被称作为成对 *Bootstrap*.

如果回归模型的适当性, 残差方差的稳定性, 或者其它回归假设有疑问的话, 成对 Bootstrap 对不满足这些假设的情形要比 Bootstrap 残差方法更加稳健. 在协变量不是固定的情形下, 即所谓的 random design, 成对 Bootstrap 更加直接地匹配了原始数据的生成机制.

例 6.3.1. 考虑模型数据 R 中的数据集 faithful. 以 waiting 为响应变量, eruptions 为自变量, 建立简单线性回归模型 $y = \alpha + \beta x + \epsilon$. 通过 qqplot 和 shapiro 我们会发现 waiting 不能假设服从正态分布. 现考虑如下几种构造 β 的置信区间.

- 标准的基于正态假设下的置信区间

$$\left[\widehat{\beta} - \mathrm{sd}(\widehat{\beta}) t_{1-\alpha/2}(n-2), \widehat{\beta} + \mathrm{sd}(\widehat{\beta}) t_{1-\alpha/2}(n-2)\right],$$

  其中 $\mathrm{sd}(\widehat{\beta}) = \widehat{\sigma}\sqrt{c_{22}}$ 为 $\widehat{\beta}$ 的标准差, c_{jj} 是 $(\mathbf{X}^T\mathbf{X})^{-1}$ 的对角线上的第 j 个元素.

- 通过 Bootstrap 残差法进行抽样并构造相应的 Bootstrap-t 置信区间和 Bootstrap percentile 置信区间. 即 计算残差 $\widehat{\epsilon}_i = y_i - \widehat{\alpha} - \widehat{\beta} x_i$. 从拟合残差集合中有放回随机地抽取得到 Bootstrap 残差集合 $\widehat{\epsilon}_1^*, \ldots, \widehat{\epsilon}_n^*$, 再计算获得 Bootstrap 响应$y_i^* = \widehat{y}_i + \widehat{\epsilon}_i^*$. 以 $\{x_i, y_i^*\}_{i=1}^n$ 为一组 Bootstrap 回归样本.

  (1) Bootstrap percentile 置信区间. 对第 j 次生成的 Bootstrap 回归样本计算 $\widehat{\beta}_j^* - \widehat{\beta}, j = 1, \ldots, B$. 之后求得 $(t_{\alpha/2}^*, t_{1-\alpha/2}^*)$, 其中 t_p^* 是 $\{\widehat{\beta}_1^*, \ldots, \widehat{\beta}_B^*\}$ 的第 $\lceil Bp \rceil$ 个次序统计量. 则 $\left(\widehat{\beta} - t_{\alpha/2}^*, \widehat{\beta} - t_{1-\alpha/2}^*\right)$ 为我们所求.

  (2) Bootstrap-t 置信区间. 同上但每次不仅计算 $\widehat{\beta}_j^*$, 还需要计算 $\widehat{\sigma}_j^*$, 之后计算标准化统计量 $(\widehat{\beta}_j^* - \widehat{\beta})/(\widehat{\sigma}_j^*\sqrt{c_{22}})$. 之后类似前种方法求得相应的 $(t_{\alpha/2}^*, t_{1-\alpha/2}^*)$, 则

$$\left(\widehat{\beta} - \widehat{\sigma}\sqrt{c_{22}} t_{1-\alpha/2}^*, \widehat{\beta} - \widehat{\sigma}\sqrt{c_{22}} t_{\alpha/2}^*\right)$$

  为我们所求.

- 通过成对 Bootstrap法进行抽样, 并构造相应的 Bootstrap-t 置信区间和 Bootstrap percentile 置信区间. 该方法的构造完全类似于通过 Bootstrap 残差法, 只不过这时的特别之处在于: 此时 c_{22} 对每次的 Bootstrap 回归样本不固定了, 均需要重新进行计算.

另外一个问题随之而来, 我们如何衡量判断这些构造置信区间方法的好坏呢? 从极限理论角度来说, 前述这些方法都是 $O_p(n^{-1/2})$ 精确的（即真实的覆盖率以 $O_p(n^{-1/2})$ 的速度收敛至 $1-\alpha$）, 这时的我们通常很难从理论上给出它们的好坏的判断. 但是随机模拟的方法可以从一定程度上帮助我们解决这个问题. 这时我们只需要对参数模型 $y_i = \alpha + \beta x_i + \epsilon_i$ 中的系数赋值（通常来说赋值的大小对各置信区间相互比较的结果不产生显著影响）, 对误差项的分布作不同的假设（当然也包括异方差等情形）, 并给定某个适当的生成 x_i 的机制, 比如随机生成或者是从某个区间内等距节点得到. 之后从这样的模型中每次生成 n 个数据（当然主要是生成协变量和误差项所对应的随机数以及计算 y 值）, 作为原始观测数据, 然后通过上面的各种方法构造不同的置信区间, 判断真实的 β 是否落在置信区间内（这个真实的 β 当然就是我们模拟开始前所给定的参数值）. 重复整个过程 N 次后, 比较各种方法包含了真实值的次数（或概率）即可（当然是以模拟估计的覆盖率和 $1-\alpha$ 的差异最小者为最好）.

§6.4 经验似然简介

假定q维$X_1, X_2, ..., X_n \overset{\text{i.i.d.}}{\sim} X$, 期望为$\mu$. 问题: 如何构造关于$\mu$的较好的置信区间? 假定$X_1, X_2, ..., X_n \overset{\text{i.i.d.}}{\sim} f(x, \mu)$ $(\mu \in \Theta)$. 定义μ的似然函数以及似然比函数

$$L_p(\mu) = \prod_{i=1}^{n} f(X_i, \mu), \quad R_p(\mu) = L_p(\mu) / \sup_{\mu \in \Theta} L_p(\mu).$$

Wilks (1938)证明了, 在正规条件下,

$$-2 \ln R_p(\mu_0) \xrightarrow{\mathcal{L}} \chi^2_q, \quad n \to \infty.$$

基于上面的结果, 我们可以构造一个置信水平为$(1-\alpha)$的似然比置信区间如下

$$\{\mu : -2 \ln R_p(\mu) < \chi^2_{q,1-\alpha}\},$$

其中$\chi^2_{q,1-\alpha}$ 是χ^2_q分布的$(1-\alpha)$分位数. 参数似然方法的优点是简便易行而且精度高; 缺点是强烈依赖于参数假设,如果这个假设错误, 那么推断结果会与真实情况相差很远.

对于分布函数$F(x)$, 令$p_i = F(\{X_i\})$. 分布$F(x)$的经验似然定义为

$$L(F) = \prod_{i=1}^{n} p_i.$$

这里p_i满足$p_i \geq 0$且$\sum_{i=1}^n p_i \leq 1$. 最大化$L_n(F)$ 的分布是经验分布

$$F_n = \frac{1}{n}\sum_{i=1}^n \delta_{x_i},$$

其中δ_x 是单点分布. 从而

$$L(F) \leq L(F_n) = n^{-n}.$$

经验分布F_n是F的非参数极大似然估计(NPMLE).

定义μ的(剖面)似然函数

$$L(\mu) = \sup_F\{L(F) : \int xdF = \mu\}.$$

容易知道$\sup_\mu L(\mu) = n^{-n} = L(\bar{X})$. 从而我们定义$\mu$的经验似然比函数

$$R(\mu) = L(\mu)/\sup L(\mu) = n^n L(\mu).$$

记$l_n(\mu) = -2\ln R(\mu)$. Owen(1988, 1990)证明了,如果X的协方差矩阵Σ 满秩, 那么

$$l(\mu_0) \xrightarrow{\mathcal{L}} \chi^2_q, \quad n \to \infty.$$

即,一个非参数的Wilks 定理成立. 基于上面的结果, 我们可以构造一个置信水平为$(1-\alpha)$的经验似然置信区间如下

$$C_{\alpha,n} = \{\mu : l(\mu) \leq \chi^2_{q,1-\alpha}\}.$$

我们现在来考虑求解$R(\mu)$，也就是要在$p_i \geq 0$, $\sum_i p_i = 1$ 和$\sum_i p_i X_i = \mu$的约束下最大化$\prod np_i$, 或者等价地，$\sum_{i=1}^n \log(np_i)$. 将约束记为$\sum_i p_i(X_i - \mu) = 0$，使用Lagrange乘子法得

$$G = \sum_{i=1}^n \log(np_i) - n\lambda\sum_{i=1}^n p_i(X_i - \mu) - \gamma\left(\sum_{i=1}^n p_i - 1\right),$$

其中λ、γ是Lagrange乘子. 求偏导后可得$\gamma = n$，且

$$p_i = \frac{1}{n\{1 + \lambda(X_i - \mu)\}}.$$

最后我们有

$$l(\mu) = 2\sum_{i=1}^n \log\{1 + \lambda(\mu)(X_i - \mu)\},$$

其中$\lambda(\mu)$ 是下面方程的解

$$\sum \frac{X_i - \mu}{1 + \lambda(X_i - \mu)} = 0. \tag{6.5}$$

求解上面的等式或者最小化$l(\mu)$(关于λ)可由牛顿法完成.

第7章　非参数密度估计和函数估计

§7.1　概述

我们观测到一组关于 X 和 Y 的数据 $\{(x_i, y_i)\}_{i=1}^n$. 假设我们认为两个变量有如下的函数关系

$$y_i = m(x_i) + \epsilon_i, \tag{7.1}$$

其中 ϵ_i 可看作是随机扰动. 我们的目的是估计 $m(x)$. 一般来说有两种方法, 一种是前面讲义中常见的参数方法, 也就是假定该函数的形式是已知的, 并且可写成带参数的形式 $m(x, \theta)$, 这里 θ 为仅有的未知量. 因此, 只要估计出 θ, 我们即得到了 m. 线性或非线性回归就属于这种方法.

这种参数方法有很多优点, 特别是表达清晰简单, 分析容易. 但缺点也很明显, 由于需要假定参数模型本身的形式, 因此对假设本身的准确性有较高的要求, 如果函数模型本身假设的不适当, 显然我们无法得到较准确或者说较有用的函数估计. 这时, 人们可采用非参数方法. 在非参数方法中, 并不假定也不固定 $m(x)$ 的形式, 也不设置参数, 我们一般仅假设 $m(x)$ 满足一定的光滑特性, 函数在每一点的值都由数据决定. 显然, 原始数据的散点图我们通常可看到由于随机扰动的影响数据有很大的波动, 极不光滑. 因此要去除干扰使图形光滑.

最简单直接的方法就是取多点平均, 也就是每一点 $m(x)$ 的值都去取离 x 最近的多个数据点的相应的 Y 值的平均. 显见, 如果用来平均的点越多, 所得的曲线越光滑. 当然, 如果用 n 个数据点来平均, 则 $m(x)$ 为常数, 这时它最光滑, 但失去了大量的信息, 拟合的残差也很大. 所以说, 这就存在了一个平衡的问题, 也就是说, 要决定每个数据点在估计 $m(x)$ 的值时要起到的作用问题. 直观上, 和 x 点越近的数据对决定 $m(x)$ 的值所应起越大的作用, 这就需要加权平均. 因此, 如何加权（或如何选择权函数）来光滑及光滑到何种程度即是我们这里所关心的核心问题. 我们把本章中所考虑的方法也称为光滑法（smoothing method）.

§7.2　核密度估计

为了引出利用核方法进行密度估计, 我们回顾一下直方图估计. 一般说来, 对于观测数据 $x_1, \ldots, x_n$, 选择两个适当的常数 x_0 和 $h > 0$, 把 $(-\infty, \infty)$ 分成k个小区间 $D_i = [x_0 + (i-1)h, x_0 + ih], i = 0, \pm 1, \pm 2, \ldots$, 并以 n_i 记 $x_1, \ldots, x_n$ 落在 D_i 的个数. 我们以 D_i 为底, $\frac{n_i}{nh}$ 为高做一矩形. 对于所有的 i 得到的许多矩形就是一个直方图. 直方图的形状当然依赖于区间的选择. 根据这样的定义, 由于直方图中所有的矩形面积和为 1, 因此它是一个密度函数. 形

式上, $f(x)$ 的直方图（密度）估计可写为

$$\widehat{f}_D(x) = \frac{1}{nh}\sum_{i=1}^{n}\sum_{j=1}^{k}[I(x_i \in D_j)I(x \in D_j)].$$

可见, 带宽越大（即区间个数越少）, 则光滑得越好（干扰去掉的多）, 但可能失去有用信息, 而且残差大（拟合不好）; 反之, 如果带宽太小（即区间个数太多）, 则残差小, 但可能光滑得不好, 造成过分拟合. 显然, 因为直方图估计是离散的, 很难用它来更好地描述连续的密度函数. 因此我们引出如下的核密度估计.

对于数据 $x_1, \ldots, x_n$, **核密度估计**（keneral density estimation）有如下形式

$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{7.2}$$

这是一个加权平均, 而**核函数**（kernal function）$K(\cdot)$ 是一个权函数. 核函数的形状和值域控制着用来估计 $f(x)$ 在点 x 的值时所用数据点的个数和利用的程度. 我们通常考虑的核函数为关于原点对称的且其积分为 1, 所以有时我们也称为一核密度函数. 下面四个核函数可能是我们最常用的：

$$\text{Uniform} : \frac{1}{2}I(|t| \le 1)$$

$$\text{Epanechikov} : \frac{3}{4}(1 - t^2)I(|t| \le 1)$$

$$\text{Quartic} : \frac{15}{16}(1 - t^2)^2 I(|t| \le 1)$$

$$\text{Gaussian} : \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{1}{2}t^2\right\}$$

直观来看, 核密度估计的好坏依赖于核函数和带宽 h 的选取. 如果利用高斯函数, 由 $\widehat{f}_h(x)$ 的表达式可看出, 如果 x_i 离 x 越近, $\frac{x-x_i}{h}$ 越接近于零, 这时密度值 $\phi\left(\frac{x-xi}{h}\right)$ 越大. 因为正态密度的值域为整个实轴, 所以所有的数据都用来估计 $\widehat{f}_h(x)$ 的值, 只不过离 x 点越近的点对估计的影响越大. 当 h 小时只有特别接近 x 的点才起较大作用, h 越大, 则远一些的点的作用也增加. 如果用均匀核函数,

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{2}I\left(\left|\frac{x - x_i}{h}\right| \le 1\right)$$

作密度函数, 则只有 $\frac{x-x_i}{h}$ 的绝对值小于 1（或者说离 x 的距离小于带宽 h 的点）才用来估计 $f(x)$ 的值, 不过所有起作用的数据的权重都相同. 不同的是, 如果使用形如 Epanechikov 核函数, 不但有截断（即离 x 的距离大于带宽 h 的点则不起作用）, 并且起作用的数据他们的权重也随着与 x 的距离增大而变小. 一般说来, 核函数的选取对和核估计的好坏的影响远小于带宽 h 的选取, 在文献中使用最多的是高斯核函数和 Epanechikov 核函数.

注意, 根据大小为 n 的一组样本在每个观测样本点都计算核密度估计需要对 K 进行 $O(n^2)$ 次计算. 因此, 算 $\widehat{f}_h$ 的计算量随 n 的增加而迅速增加. 然而对多数实际问题, 像作密度曲线图, 就不必在每个点 x_i 上计算估计. 实际的方法是在 x 值的格子点（就是将 x 轴划分为许多个等距小区间）上计算 $\widehat{f}_h(x)$, 然后在格子点间线性内插. 100-200 个格子点通常足够使 $\widehat{f}_h$ 的图形看上去比较光滑了.

上面的方法对于多元数据同样适用, 这时仅需要使用所谓的multivariate kernel 函数即可, 通常得到如下的估计量

$$\widehat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|\mathbf{H}|}K(\mathbf{H}^{-1}(\mathbf{x}-\mathbf{x}_i)).$$

我们一般都会采用简单的带宽阵或$K(\cdot)$, 比如$\mathbf{H} = \mathrm{diag}\{h_1,\ldots,h_d\}$.

$$\widehat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{nh_1\cdots h_d}\sum_{i=1}^{n}\prod_{j=1}^{d}K_j\left(\frac{x_j - x_{ij}}{h_j}\right).$$

精确地来讲, (7.2) 的估计量称为*固定带宽核密度估计*, 因为 h 是常数. 上世纪90年代中期, 有学者亦指出, 如果 h 依 i 而变化, 即所谓的 variable bandwidth, 则有一些很好的性质, 但我们这里不对这种方法给予详细的讨论, 有兴趣的同学可参见著作 Fan and Gijbels (1996). 带宽值的选择对估计量 $\widehat{f}_h$ 有很大的影响. 如果 h 太小, 那么密度估计偏向于把概率密度分配得太局限于观测数据附近, 致使估计密度函数有很多错误的峰值. 如果 h 太大, 那么密度估计就把概率密度贡献散得太开. 在很大的邻域里求平均会光滑掉 f 的一些重要特征的. 下面一节讨论如何选取 h.

§7.3 带宽的选取

要更好地理解带宽的选择, 一定的理论上的分析是必须的. 我们必须首先考虑如何评价密度估计量的性质. 我们来看*均方误差* $\mathrm{MSE}(\widehat{f}_h(x))$ 和所谓的*积分均方误差*（mean integrated square error） $\mathrm{MISE}(h)$. 它们的定义是

$$\mathrm{MISE}(h) = \int \mathrm{MSE}(\widehat{f}_h(x))dx = \int \mathrm{var}\{\widehat{f}_h(x)\} + \Big(\mathrm{bias}\{\widehat{f}_h(x)\}\Big)^2 dx, \tag{7.3}$$

其中且 $\mathrm{bias}\{\widehat{f}_h(x)\} = \mathrm{E}\{\widehat{f}_h(x)\} - f(x)$. 显然, $\mathrm{MSE}(\widehat{f}_h(x))$ 是对于评价估计 $\widehat{f}_h(x)$ 好坏的一个逐点准则, 而 $\mathrm{MISE}(h)$ 可看成是在每点 x 处对局部均方误差的累积, 也就是一个全局的评判准则.

另外一个概念是*积分平方误差*（integrated square error）

$$\mathrm{ISE}(h) = \int (\widehat{f}(x) - f(x))^2 dx.$$

注意, ISE(h) 是观测数据的函数, 因此它在观测样本的条件下总结了 $\widehat{f}$ 的表现. 在不考虑样本的情况下, 如果我们想讨论估计量的一般性质, 那么在所有可能观测的样本上对 ISE(h) 进行平均是比较合理的, 即我们又得到了 MISE(h) = E{ISE(h)}（假设期望和积分可交换, 则两个 MISE(h)就统一在一起了）, 其中期望是关于分布 f 的. 因此 MISE(h) 又可以看成是误差（即 ISE(h) ）关于抽样密度的整体度量的平均值.

进一步地, 我们还需要对核函数和 f 作一定的假设. 设 K 是对称连续的核密度函数, 均值为零, 方差 $0 < \sigma_K^2 < \infty$. 这里 f 有二阶有界连续导数且 $\int [f''(t)]^2 dt < \infty$, 也就是对其光滑性做一定的假设. 假设当 $n \to \infty$ 时 $nh \to \infty, h \to 0$, 我们将进一步分析该表达式. 要计算 (7.3) 中的偏差项, 注意到应用变量替换有

$$\begin{aligned} \mathrm{E}\{\widehat{f}_h(x)\} &= \frac{1}{h}\int K\left(\frac{x-u}{h}\right) f(u)du \\ &= \int K(t) f(x-ht)dt. \end{aligned} \tag{7.4}$$

然后在 (7.4) 中用Taylor级数展开

$$f(x-ht) = f(x) - htf'(x) + h^2t^2 f''(x)/2 + o(h^2) \tag{7.5}$$

替换并注意到 K 关于零点对称可得

$$\mathrm{E}\{\widehat{f}_h(x)\} = f(x) + h^2\sigma_K^2 f''(x)/2 + o(h^2).$$

因此

$$\left(\mathrm{bias}\{\widehat{f}_h(x)\}\right)^2 = h^4\sigma_K^4[f''(x)]^2/4 + o(h^4),$$

且该表达式对 x 积分可得

$$\int \left(\mathrm{bias}\{\widehat{f}_h(x)\}\right)^2 dx = h^4\sigma_K^4 \int [f''(x)]^2 dx/4 + o(h^4).$$

计算 (7.3) 中的方差项可采用类似的方法：

$$\begin{aligned} \mathrm{var}\{\widehat{f}_h(x)\} &= \frac{1}{n}\mathrm{var}\left\{\frac{1}{h}K\left(\frac{x-x_i}{h}\right)\right\} \\ &= \frac{1}{nh}\int K^2(t) f(x-ht)dt - \frac{1}{n}\left[\mathrm{E}\left\{\frac{1}{h}K\left(\frac{x-x_i}{h}\right)\right\}\right]^2 \\ &= \frac{1}{nh}\int K^2(t)[f(x)+o(1)]dt - \frac{1}{n}[f(x)+o(1)]^2 \\ &= \frac{1}{nh} f(x)\int K^2(x)dx + o\left(\frac{1}{nh}\right). \end{aligned}$$

将其对 x 积分得

$$\int \mathrm{var}\{\widehat{f}_h(x)\}dx = \frac{\int K^2(x)dx}{nh} + o\left(\frac{1}{nh}\right).$$

因此 $\mathrm{MISE}(h) = \mathrm{AMISE}(h) + o\left(\frac{1}{nh} + h^4\right)$, 其中

$$\mathrm{AMISE}(h) = \frac{\int K^2(x)dx}{nh} + \frac{h^4\sigma_K^4 \int [f''(x)]^2 dx}{4}$$

称作*渐进均方积分误差*. 如果当 $n \to \infty$ 时 $nh \to \infty, h \to 0$, 则 $\mathrm{MISE}(h) \to 0$.

要关于 h 最小化 $\mathrm{AMISE}(h)$, 我们必须把 h 设在某个中间值, 这样可以避免 $\widehat{f}_h$ 有过大的偏差（太过光滑）或过大的方差（即过于光滑）. 关于 h 最小化 $\mathrm{AMISE}(h)$ 表明最好是精确地平衡 $\mathrm{AMISE}(h)$ 中偏差项和方差项的阶数. 显然最优的带宽是

$$h = \left(\frac{\int K^2(x)dx}{n\sigma_K^4 \int [f''(x)]^2 dx}\right)^{1/5}, \tag{7.6}$$

但该结果用处并不很大, 因为它依赖于未知密度 f. 注意最优带宽有 $h = O(n^{-1/5})$, 这种情况下 $\mathrm{MISE} = O(n^{-4/5})$. 该结果显示了随着样本量的增加带宽缩小的速度, 但对给定的数据集来说它并未指明带宽具体取多少对密度估计是合适的. 下面给出出几种带宽选择策略. 在实际应用中, 它们的表现随着 f 的性质以及观测数据的不同也有所不同, 通常没有一个绝对最好的方法.

拇指法则: 简便起见, 我们定义 $R(g) = \int g^2(z)dz$. 针对最小化 AMISE 得到的最优带宽中含有未知量 $R(f'')$, Silverman 提出一种初等的方法, *rule of thumb* （拇指法则, 即根据经验的方法）: 把 f 用方差和估计方差相匹配的正态密度替换. 这就等于用 $R(\phi'')/\widehat{\sigma}^5$ 估计 $R(f'')$, 其中 ϕ 为标准正态密度函数. 若取 K 为高斯密度核函数而 σ 使用样本方差 $\widehat{\sigma}$, Silverman 拇指法则得到

$$h = \left(\frac{4}{3n}\right)^{1/5} \widehat{\sigma}. \tag{7.7}$$

但通常推荐考虑半极差（IQR）, 因为 IQR 是更加稳健的散度度量. Silverman 建议在(7.7)中用 $\tilde{\sigma} = \min\{\widehat{\sigma}, \mathrm{IQR}/(\Phi^{-1}(0.75) - \Phi^{-1}(0.25))\} \approx \min\{\widehat{\sigma}, \mathrm{IQR}/1.35\}$ 替换 $\widehat{\sigma}$, 其中 Φ 是标准正态累积分布函数. 作为产生近似带宽的一种方法, Silverman 的拇指法则是很有效的, 这种带宽通常作为更加复杂的plugin方法中非参数估计的带宽.

Plug-in 方法: 该方法即是所谓的代入法, 其考虑在最优带宽 (7.6) 中使用某适当的估计 $\widehat{R}(f'')$ 来代替 $R(f'')$. 在众多的方法中, 最简单且最常用的即是 Sheather and Jones (1991; JRSSB) 所提出的 $\widehat{R}(f'') = R(\hat{f}'')$. 而 $\hat{f}''$ 的基于核的估计量为

$$\begin{aligned}\hat{f}''(x) &= \frac{\partial^2}{\partial x^2}\left\{\frac{1}{nh_0}\sum_{i=1}^{n} L\left(\frac{x - x_i}{h_0}\right)\right\} \\ &= \frac{1}{nh_0^3}\sum_{i=1}^{n} L''\left(\frac{x - x_i}{h_0}\right),\end{aligned}$$

其中 h_0 为带宽, L 为用来估计 f'' 的核函数（不一定与 K 相同）. 再对其平方并对 x 积分后即可得到 $R(\hat{f}'')$.

估计 f 的最优带宽和估计 f'' 或 $R(f'')$ 的最优带宽是不同的. 根据理论上以及经验上的考虑, Sheather and Jones 建议用简单的拇指法则计算带宽 h_0, 该带宽用来估计 $R(f'')$, 这是最优带宽表达式 (7.6) 中唯一未知的. 然后通过 (7.6) 计算带宽 h 并产生最后的核密度估计.

交叉核实（cross-validation）: 该方法是一种基于数据（data-driven）方法, 其基本思想是考虑最小化积分平方误差 ISE(h). 把积分平方误差重新写成

$$\begin{aligned}\mathrm{ISE}(h) &= \int \widehat{f}_h^2(x)dx - 2\mathrm{E}\{\widehat{f}_h(X)\} + \int f^2(x)dx \\ &= R(\widehat{f}_h) - 2\mathrm{E}\{\widehat{f}_h(X)\} + R(f).\end{aligned}$$

该表达式的最后一项 $R(f)$ 不依赖于 $\widehat{f}$, 因此其对选择 h 没有影响. 中间项可以用 $\frac{2}{n}\sum_{i=1}^n \widehat{f}_{-i}(x_i)$ 来估计, 其中

$$\widehat{f}_{-i}(x_i) = \frac{1}{h(n-1)}\sum_{j\neq i} K\left(\frac{x_i - x_j}{h}\right)$$

表示在 x_i 点处核密度估计量用除 x_i 外所有数据估计的密度.

因此, 通过关于 h 最小化

$$\mathrm{CV}(h) = R(\widehat{f}) - \frac{2}{n}\sum_{i=1}^n \widehat{f}_{-i}(X_i) \tag{7.8}$$

通常可得到较好的带宽. 事实上, 可以证明, 这样选出来的 h, 不妨记为 $\widehat{h}_{CV} = \arg\min \mathrm{CV}(h)$, 按下面的意义是渐进最优的: 当 $n \to \infty$ 时,

$$\frac{\mathrm{ISE}(\widehat{h}_{CV})}{\inf_{h>0}\mathrm{ISE}(h)} \xrightarrow{wp1} 1.$$

§7.4 基于核光滑的非参数回归

我们继续讨论本章开始的模型 (7.1). 假定 $E(\epsilon_i) = 0$, $E(\epsilon_i^2) = \sigma^2 < \infty$, 且 ϵ_i 相互独立. 如果假定 X 和 Y 都是随机的, 他们有联合分布 $f(x,y)$, 而 X 的边际分布为 $f(x)$. 我们可以认为 $m(x)$ 是 Y 在给定了 $X = x$ 之后的条件期望

$$m(x) = E(Y|X=x) = \int yf(y|x)dy = \frac{\int yf(x,y)dy}{f(x)}. \tag{7.9}$$

据此, 将 $f(x)$ 和 $f(x,y)$ 的核密度估计

$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

$$\widehat{f}(x,y) = \frac{1}{nhh_y}\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) K_y\left(\frac{y - y_i}{h_y}\right)$$

代入到 (7.9) 则得到著名的 Nadaraya-Watson 核估计为

$$\widehat{m}_{NW}(x)=\frac{\frac{1}{nh}\sum_{i=1}^{n}K(\frac{x-x_i}{h})y_i}{\frac{1}{nh}\sum_{i=1}^{n}K(\frac{x-x_i}{h})},$$

其中注意使用 $\int K_y(u)du=1$ 和 $\int uK_y(u)du=0$. 这里的核函数 $K(\cdot)$ 和密度估计时的核函数没什么不同, 带宽 h 在估计 $m(x)$ 曲线时的性质也在密度估计时类似.

在回归中, 也不一定要求分母是 $f(x)$ 的估计, 特别当 X 不是随机的时候. 因此, 更一般的对 $m(x)$ 的估计可写为

$$\widehat{m}(x)=\sum_{i=1}^{n}W_i(x)y_i$$

这里 W_i 为依赖于 $\{x_i\}_{i=1}^n$ 的权函数. 我们称形如这样表达的非参数回归方法为 **线性光滑方法**. 显然, Nadaraya-Watson 属于这一类方法, 而不同的光滑方法就依赖于 W_i 的选取方法.

通过一些简单的运算, 我们可证明 $m_{NW}(x)$ 是加权最小二乘问题（或称为局部最小二乘）的解, 即

$$\widehat{m}_{NW}(x)=\arg\min_{\beta_0}\sum_{i=1}^{n}(y_i-\beta_0)^2K\left(\frac{x-x_i}{h}\right).$$

也就是说, $\widehat{m}_{NW}$ 可看作是用常数来局部近似 $m(x)$. 然而, 局部常数只在非常小的邻域内才较为合理. 这就给了我们一些新的启示, 是否可以使用更高阶的多项式呢？对于任意给定的 x, 其附近的点 x_i 的函数值可由Taylor 展开逼近

$$m(x_i)\approx m(x)+m'(x)(x_i-x)+\cdots+m^{(p)}(x)(x_i-x)^p/p!$$

这就引出了著名的**局部多项式估计**（local polynomial regression estimator）

$$\arg\min_{\boldsymbol{\beta}}\sum_{i=1}^{n}\left[y_i-\beta_0-\cdots-\beta_p(x_i-x)^p\right]^2K\left(\frac{x_i-x}{h}\right).$$

令 $\boldsymbol{X}$ 为一设计阵

$$\begin{pmatrix}1 & x_1-x & & (x_1-x)^p\\ \vdots & \vdots & \cdots & \vdots\\ 1 & x_n-x & & (x_n-x)^p\end{pmatrix}$$

且令

$$\mathbf{W}=h^{-1}\mathrm{diag}\left[K\left(\frac{x_1-x}{h}\right),\ldots,K\left(\frac{x_n-x}{h}\right)\right]$$

为一权矩阵; 则如果 $\boldsymbol{X}^T\mathbf{W}\boldsymbol{X}$ 是可逆的, 则可计算得到

$$\widehat{\boldsymbol{\beta}}=(\boldsymbol{X}^T\mathbf{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{W}\mathbf{y}$$

则 $\widehat{m}_p(x)$ 即为截距项的估计 $\widehat{\beta}_0$. 而更一般地, $q! \times \widehat{\beta}_q$ 是 $m(x)$ 的 q 次导数 $m^{(q)}(x)$ 的估计. Fan 证明了局部线性估计（local linear estimator）具有很多最优的性质, 且由于其简单性, 在当今的许多应用中使用最为广泛. 简单的计算可得局部线性估计具有如下的显示表达式

$$\widehat{m}_1(x) = \frac{1}{nh}\sum_{i=1}^{n} \frac{[s_2(x,h) - s_1(x,h)(x_i - x)]K\left(\frac{x_i - x}{h}\right) y_i}{s_2(x,h)s_0(x,h) - s_1(x,h)^2},$$

其中

$$s_r(x,h) = \frac{1}{nh}\sum_{i=1}^{n}(x_i - x)^r K\left(\frac{x_i - x}{h}\right).$$

注: 使用局部常数的Nadaraya-Watson估计和局部线性方法到底有什么区别呢？研究这个问题需要从理论角度来考虑. Fan (1992) 详细地阐述了该问题. 理论上来说，Nadaraya-Watson估计的极限偏差和方差分别是

$$[m''(x) + \frac{2m'(x)f'(x)}{f(x)}]b_n, \quad V_n,$$

其中$b_n = \frac{1}{2}\int u^2 K(u)du h^2$, $V_n = \frac{\sigma^2(x)}{f(x)nh}\int K^2(u)du$. 而局部线性方法的极限偏差和方差分别是

$$m''(x)b_n, \quad V_n.$$

由此我们看出局部线性方法通过多使用的一个参数，在保证方差未变的情形下，降低了偏差，使得其极限偏差不依赖于m的导数，更重要的是不依赖于设计点的密度f, 这使得该方法从估计的稳健性以及参数选取的角度也都具有相当的优势。因此在实际问题中强烈推荐使用局部线性光滑方法.

对于局部多项式估计, 我们同样要面对选取带宽 h 的问题. 如同核密度估计, plug-in 和 cross-validation 是最常用的两种方法. 关于 plug-in, 由于其同样涉及到复杂的 MISE 的表达及近似, 这里就不详加叙述了. 而 CV 法很容易表达. 我们仍旧采用前面的 leave-one-out 的想法, 得到如下的准则

$$\mathrm{CV}(h) = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - \widehat{m}_p^{(-i)}(x_i)\right]^2,$$

其中表示 $\widehat{m}_p^{(-i)}(x_i)$ 在 x_i 点处估计量用除 x_i 外所有数据所得的局部多项式估计. 最小化 $\mathrm{CV}(h)$ 则得到基于 CV 的最优带宽.

§7.5 基于样条的非参数回归

我们知道，使用一个全局的多项式函数来逼近那些在不同位置具有不同的光滑程度的函数曲线是不合适的，因为它在所有位置处都有任一阶导数. 克服这种缺限的一种直接方法是使用前面介绍的局部多项式方法，也就是在每点处都使用某个低维多项式来逼近函数; 而另

一种增强灵活性的方法就是允许逼近函数的导数在某些位置上不连续. 这样就得到了某种分段多项式或者也称为样条(spline), 这种方法也就称为*样条光滑方法*. 而我们一般称那些导数不连续的位置点为*节点(knot)*.

§7.5.1 多项式样条

假定我们现在欲使用样条函数来逼近m. 令$t_1,\ldots,t_J$ 是一列固定的节点使得 $-\infty<t_1<\cdots<t_J<\infty$. 为研究方便，我们不妨假设$0\le t_1<\cdots<t_J\le 1$ K阶样条，就是一最高阶为$K-1$次（所以有时也称为$K-1$次样条），具有$K-2$阶连续导数的分段多项式函数，且在节点处$K-1$阶导数不连续. 例如, 三次样条(cubic spline), $K=4$, 就是指某二阶连续可导函数s使得该函数在每一个小区间 $(-\infty,t_1],[t_1,t_2],\cdots,[t_{J-1},t_J],\ [t_J,\infty)$ 内是一个三次多项式. 注意到所有的K阶样条构成了一个$(J+K)$维的线性空间. 两种最著名的样条基是:

- Truncated power basis: $(x-t_j)_+^{K-1}, j=1,\ldots,J,\ 1,x,x^2,x^3,\ldots,x^{K-1}$;
- B-spline basis: 定义另外$2K$个节点, $t_{-(K-1)},\ldots,t_{-1},t_0,t_{J+1},\ldots,t_{J+K}$为$t_{-(K-1)}=\cdots=t_{-1}=t_0=0$和$t_{J+1}=\cdots=t_{J+K}=1$. K阶B-spline则定义为

$$N_{i,K}(x)=\frac{x-t_i}{t_{i+K-1}-t_i}N_{i,K-1}(x)+\frac{t_{i+K}-x}{t_{i+K}-t_{i+1}}N_{i+1,K-1}(x),\quad i=-(K-1),\ldots,J$$

其中初始值$N_{i,1}(x)=I(x\in[t_i,t_{i+1}))$.

通常来说B-spline基计算起来更加稳定，这是由于各组基之间的相关性比较小，而Truncated power basis从统计角度来说更加直观，因为删掉$(x-t_j)_+^{K-1}$等同于时删除节点，因此从变量选择或是节点选择来说具有一定的优势.

下面考虑如何用这些样条基来拟合数据. 此时原来的非参数回归模型可近似写为

$$y_i\approx s(x_i)+\epsilon_i,$$

其中$s(x)=\sum_{j=1}^{J+K}\theta_jB_{j,\lambda}(x)$, $B_j(\cdot)$就是上面的样条基函数, $\lambda=(t_1,\ldots,t_J)^T$. 此时显然我们可通过最小二乘的方法来确定参数$\boldsymbol{\theta}=(\theta_1,\ldots,\theta_{J+K})^T$，即

$$\min_{\boldsymbol{\theta}}\sum_{i=1}^{n}\left[y_i-\sum_{j=1}^{J+K}\theta_jB_{j,\lambda}(x_i)\right]^2.$$

事实上，定义设计矩阵$\boldsymbol{X}_\lambda=(B_{j,\lambda}(x_i))_{i=1,\ldots,n;j=1,\ldots,J+K}$, 则我们可以直接将上面的最小化问题转化为我们熟知的线性模型最小二乘，也就是当$\boldsymbol{X}_\lambda$满秩时，我们有解

$$\widehat{\boldsymbol{\theta}}_\lambda=(\boldsymbol{X}_\lambda^T\boldsymbol{X}_\lambda)^{-1}\boldsymbol{X}_\lambda^T\mathbf{y},$$

其中$\mathbf{y}=(y_1,\ldots,y_n)$. 我们的样条估计为$\widehat{m}(x)=\sum_{j=1}^{J+K}\widehat{\boldsymbol{\theta}}_\lambda B_{j,\lambda}(x)$.

上面所描述的样条方法强烈依赖于结点个数J的选取，也依赖于他们的位置. 节点个数过多，$\widehat{m}(x)$就会抖动较为强烈（方差较大），反之会较为平滑，更像是在全支撑上的一个低阶多项式. 而一般来说，我们应该在那些曲线有较大变化的位置上放置我们的节点. 因此一个自动选取的方法是我们渴望得到的.

常用的方法就是节点删除法. 这里仅考虑使用Truncated power basis. 想法很直接, 先考虑使用较多的节点，比如$[n/2]$或者$[n/3]$并且节点就选取为样本点x_i, 如，$t_j = x_{(2j)}, j = 1, \ldots, [n/2]$. 接下来从全模型开始，考虑剔除$(x - t_j)_+^{K-1}, j = 1, \ldots, J$中的一些，也就是类似向后回归的思路. 记$\widehat{\theta}_j$为最小二乘估计, $SE(\widehat{\theta}_j)$是估计的标准误. 则我们删除$|\widehat{\theta}_j|/SE(\widehat{\theta}_j)$最小的那个，记为$j_0$. 重复整个过程, 每一次我们都计算残差平方和$RSS_k$($k$表示已删除$k$个节点), 则我们使用类似$C_p$准则来选取$k$，即

$$C_k = RSS_k + \alpha(J + K - k)\widehat{\sigma}^2,$$

其中$\widehat{\sigma}^2$是在全模型下得到的方差估计. $\alpha = 2$即是C_p准则，而一般在这里推荐使用$\alpha = 3$.

§7.5.2 光滑样条

另外一种能够自动确定节点的样条方法就是所谓的光滑样条（smoothing spline）. 为了引出该种方法，我们下面先来考虑一种非常直接的最小二乘，也就是

$$\min \sum_{i=1}^{n} (y_i - m(x_i))^2.$$

很容易看到，如果我们不对m作任何限制的话，上式的解就是所有的能够使得$y_i = m(x_i)$的函数m. 从统计角度来说，这种方法几乎没有任何意义，因为它根本就没有分析数据，得到了一个非常复杂的函数模型(就是像原始的数据一样，没有提取任何信息). 从统计建模的角度来说，这种方法属于典型的过参数化(over-parameterizes), 导致估计参数有非常大的方差.

在上面的方法中我们缺少了什么呢？事实上，就是我们没有对过参数化问题给予适当的惩罚，也就是未对m做任何的限制. 一种方便且流行的度量光滑性的惩罚就是$\int [m''(x)]^2 dx$. 这样就得到了如下的惩罚最小二乘(penalized least square)

$$\min \sum_{i=1}^{n} (y_i - m(x_i))^2 + \lambda \int [m''(x)]^2 dx,$$

其中$\lambda > 0$是一个惩罚（光滑）参数. 不难看出$\lambda = 0$时，就是我们上面所说的没有惩罚的解，而当$\lambda = \infty$时，我们得到的解是$m(x) = \alpha + \beta x$. 随着λ从零变到无穷大，我们的估计就从最复杂的模型变到最简单的直线回归模型. 因此，模型的复杂程度，或者说方法的有效性（从均方误差角度）就是被λ有效地控制了. 相应的解我们记为$\widehat{m}_\lambda$.

可以证明，$\widehat{m}_\lambda$是一个定义在$[x_{(1)}, x_{(n)}]$的cubic spline（参见Eubank 1999）, 且其是关于y的

线性表达：

$$\widehat{m}_\lambda(x) = n^{-1}\sum_{i=1}^{n} W_i(x,\lambda;x_1,\ldots,x_n)y_i,$$

其中W_i不依赖于响应$\mathbf{y}$. 剩下的就是如何选取参数λ. 此时，当然直接的想法是使用CV,

$$CV(\lambda) = n^{-1}\sum_{i=1}^{n}[y_i - \widehat{m}_{\lambda,(-i)}(x_i)]^2,$$

其中$\widehat{m}_{\lambda,(-i)}$就是去掉第$i$个观测之后的光滑样条估计. 当然，这种方法计算量较大，因此在光滑样条参数选取问题中，我们另外一种常用的方法是所谓的广义交叉核实(generalized cross-validation; GCV). 由上我们注意到拟合值向量可写为

$$(\widehat{m}_\lambda(x_1),\ldots,\widehat{m}_\lambda(x_n))^T = H(\lambda)\mathbf{y},$$

其中$H(\lambda)$是一个$n\times n$的矩阵，仅依赖于X (就是由上面的W_i决定的). 则GCV准则就是要找到λ最小化

$$GCV(\lambda) = \frac{\mathrm{MASE}(\lambda)}{[\frac{1}{n}\mathrm{tr}(\mathbf{I}-H(\lambda))]^2},$$

其中$\mathrm{MASE}(\lambda) = n^{-1}\sum_{i=1}^{n}[y_i - \widehat{m}_\lambda(x_i)]^2$. $CV(\lambda)$ 和$GCV(\lambda)$在MISE的意义下选择λ是相合的.

注: 光滑样条的方法可以很容易地推广到异方差情形下，也就是说$\mathrm{var}(y_i), i=1,\ldots,n$是不一定相同的. 在这种情形下，我们仅需要稍微修改最小化方程即可

$$\min\sum_{i=1}^{n} w_i(y_i - m(x_i))^2 + \lambda\int[m''(x)]^2dx,$$

其中$w_i = \mathrm{var}^{-1}(y_i)$, 这样也就是得到了一个加权惩罚最小二乘, 前面所介绍的方法都可以很容易地直接推广过来.

§7.5.3 高维非参数回归

§7.5.3.1 可加模型（additive model）

简单线性回归是基于模型 $\mathrm{E}\{Y|x\} = \beta_0 + \beta_1 x$ 的. 前面章节中的一元非参光滑将其推广为 $\mathrm{E}\{Y|x\} = s(x)$ ，其中 s 为某光滑函数. 现在我们试图类推到有 p 个预测变量的情形. 多元回归使用模型 $\mathrm{E}\{Y|\mathbf{x}\} = \beta_0 + \sum_{k=1}^{p}\beta_k x_k$, 其中 $\mathbf{x} = (x_1,\ldots,x_p)^T$. 对光滑的推广是可加模型

$$\mathrm{E}\{Y|\mathbf{x}\} = \alpha + \sum_{k=1}^{p} s_k(x_k), \tag{7.10}$$

其中 s_k 是 k 个预测变量的光滑函数. 因此，总模型是由对平均响应具有可加影响的一元效应构成. 注意到如果未加任何其他限制，上面模型中的α是一个自由参数，也就是说它无法与其他的$s_k(\cdot)$区分开来（即可识别性; identifiability）. 因此我们不妨假设$E(s_k(x_k)) = 0, k=1,\ldots,p.$

我们如何来拟合这种模型呢? 注意到关系

$$s_k(x_k) = \mathrm{E}\{Y - \alpha - \sum_{j\neq k} s_j(x_j)|x_k\},\ k = 1,\ldots,p. \tag{7.11}$$

这个式子立即就给出我们一个直观的迭代算法来求取$s_1,\ldots,s_p$. 也就是，对于给定的α和其它函数$s_j, j \neq k$, 我们可利用$\{x_{ik}, z_{ik}\}_{i=1}^n$构造某种适当的非参数光滑方法来估计$s_k$, 其中$z_{ik} = y_i - \alpha - \sum_{j\neq k} s_j(x_{ij})$. 当然，为了满足可识别性的约束，在获得了$s_k(\cdot)$的估计$\widehat{s}_k(\cdot)$之后，我们可将其中心化

$$\widehat{s}_k^*(\cdot) = \widehat{s}_k(\cdot) - \frac{1}{n}\sum_{i=1}^n \widehat{s}_k(x_{ik}). \tag{7.12}$$

这种迭代方法称为**后退拟合算法***(backfitting)*. 令 $\mathbf{y} = (y_1,\ldots,y_n)^T$ 且对每个 k，令 $\hat{\mathbf{s}}_k^{(t)}$ 表示在第 t 次迭代中 $s_k(x_{ik})$ 估计值, $i = 1,\ldots,n$, 构成的向量. 每个观测上估计光滑值的 n 维向量按如下步骤更新：

1. 令 $\hat{\alpha}$ 为 n 维向量 $(\bar{y},\ldots,\bar{y})^T$. 令 $t = 0$，其中 t 表示迭代次数.

2. 令 $\hat{\mathbf{s}}_k^{(0)}$ 代表在观测数据上对逐个坐标光滑的初步猜测. 一种合理的初步猜测是令 $\hat{\mathbf{s}}_k^{(0)} = (\hat{\beta}_k x_{1k}^*,\ldots,\hat{\beta}_k x_{nk}^*)^T$, $k = 1,\ldots,p$, 其中 $\hat{\beta}_k$ 是 y 对预测变量回归时的线性回归系数, $x_{ik}^* = x_{ik} - \bar{x}_k$.

3. 依次对 $k = 1,\ldots,p$, 令

$$\hat{\mathbf{s}}_k^{(t+1)} = \mathrm{smooth}_k(\mathbf{r}_k), \tag{7.13}$$

其中

$$\mathbf{r}_k = \mathbf{y} - \hat{\alpha} - \sum_{j<k}\hat{\mathbf{s}}_j^{(t+1)} - \sum_{j>k}\hat{\mathbf{s}}_j^{(t)} \tag{7.14}$$

且 $\mathrm{smooth}_k(\mathbf{r}_k)$ 表示通过对预测变量的第 k 个坐标值，即 $x_{1k},\ldots,x_{nk}$, 光滑 $\mathbf{r}_k$ 的元素并求在 x_{ik} 的光滑值所得到的向量. 之后再使用(7.12)得到中心化后的估计作为$\hat{\mathbf{s}}_k^{(t+1)}$.

4. 增加 t 并转入第3步.

当 $\hat{\mathbf{s}}_k^{(t)}$ 变化都不大时算法终止—也许是当

$$\sum_{k=1}^p \left(\hat{\mathbf{s}}_k^{(t+1)} - \hat{\mathbf{s}}_k^{(t)}\right)^T \left(\hat{\mathbf{s}}_k^{(t+1)} - \hat{\mathbf{s}}_k^{(t)}\right) \Big/ \sum_{k=1}^p \left(\hat{\mathbf{s}}_k^{(t)}\right)^T \left(\hat{\mathbf{s}}_k^{(t)}\right)$$

非常小时. 注意到此时求出的是在开始给定的n个观测点所对应的光滑估计，而如果想得到任意给定的$\mathbf{x}$所对应的估计，只需要使用(7.11)和(7.12)再作一次光滑即可.

§7.5.3.2 部分线性模型（partially linear model）

假设相应变量Y线性依赖于向量$\boldsymbol{Z}=(Z_1,\ldots,Z_q)^T$, 同时非线性依赖于协变量$X$, 则如下的模型可能足够描述数据

$$Y=g(X)+\boldsymbol{Z}^T\beta+\epsilon,$$

其中$g(\cdot)$是一未知一元函数, β是q维参数向量. 我们将该类模型称为partially linear model，有时也称为partial spline model, 因此最早是出现在spline的文献中的. 注意到，该类模型也正是可加模型的一种推广.

拟合这样的模型就是较为容易的了，我们可以考虑用任何一种常用的光滑方法. 假设我们的观测是$\{x_i,\boldsymbol{z}_i,y_i\}_{i=1}^n$是从$(X,\boldsymbol{Z},Y)$中得到的$n$组观测. 则使用光滑样条思想我们可有

$$\min\sum_{i=1}^n(y_i-g(x_i)-\boldsymbol{Z}_i^T\boldsymbol{\beta})^2w_i+\lambda\int[g''(x)]^2dx,$$

其中w_i可以用来描述异方差的影响.

使用局部多项式方法来拟合数据就是要求解如下的最小化问题

$$\min\sum_{i=1}^n(y_i-\sum_{j=0}^p\theta_j(x_i-x)^p-\boldsymbol{Z}_i^T\boldsymbol{\beta})^2K\left(\frac{x_i-x}{h}\right),$$

也就是，同时求解$(\theta_0,\ldots,\theta_p)^T$和$\boldsymbol{\beta}$. 不难看出，这最终就转化为$p+q$维的加权最小二乘, 因此可以获得显示解$\widehat{g}(x)$和$\widehat{\boldsymbol{\beta}}$. 当然需要注意到的是, 在这样一个解法中，我们得到的β的估计仅仅用到了x点附近的观测, 这对于一个全局的系数来说是不够有效的，因此我们可以最终再用$y_i-\widehat{g}(x_i)$对$\boldsymbol{Z}_i$做线性回归来获得更有效的$\boldsymbol{\beta}$的估计.