统计计算 张楠 2019秋: The Metropolis–Hastings Algorithm

(返回 统计计算 张楠 2019秋)

MH(*Metropolis-Hastings*) 算法是一类常用的构造马氏链的方法, 其包括了: Metropolis抽样, Gibbs抽样, 独立抽样, 随机游动抽样等. MCMC方法的精髓在于构造合适的马氏链, 因此算法的主要 目的是对马氏链$\{X_t|t=0,1,2,\cdots\}$, 在给定一个X_t所处的状态下, 产生下一步的状态X_{t+1}. MH算法构造如下:

1. 构造合适的提议分布$g(\cdot|X_t)$.
2. 从$g(\cdot|X_t)$中产生Y;
3. 若Y被接受, 则$X_{t+1}=Y$, 否则 $X_{t+1}=X_t$.

提议分布(Proposal distribution)的选择要使得生产的马氏链的平稳分布为目标抽样分布f, 需要满足的正则化 条件包括不可约, 正常返, 非周期. 一个具有和目标分布相同支撑集的提议分布一般会满足这些正则化条件.

Metropolis-Hastings Sampler

MH抽样方法通过如下方式生产马氏链$\{X_0,X_1,X_2,\cdots\}$:

1. 构造合适的提议分布$g(\cdot|X_t)$(满足前述的正则化条件).
2. 从某个分布g中产生X_0;
3. 重复(直至马氏链达到平稳状态)

(1) 从$g(\cdot|X_t)$中产生Y.
(2) 从$U(0,1)$中产生U;
(3) 若 $U\leq \frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)}$ 则接受Y并令$X_{t+1}=Y$, 否则 $X_{t+1}=X_t$.
(4) 增加t, 返回到(1).

目录

注意到上述算法中接受概率为 $\alpha(X_t,Y)=min\left(1,\frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)}\right)$, 因此只需知道目标分布$f$正比于某个常数即可, 该常数可以未知.

显然, 通过MH算法构造的链满足马氏性, 因为X_{t+1}仅依赖于X_t. 而这样 的链是否是非周期不可约的则取决于提议分布的选取. 如果是非周期不可约的, 则由MH算法得到的链具有唯一的平稳分布. 我们来验证在MH算法下得到的马氏链的平稳分布为f. 事实上, 当$r\neq s$时, 转移核为

$$\begin{aligned}K(r,s)&=p(s|X_t=r)\approx P(X_{t+1}\in s\pm h,TA|X_t=r)/2h\\&=\int_{s-h}^{s+h}g(y|r)\alpha(r,y)dy/2h\to g(s|r)\alpha(r,s),h\to 0.\end{aligned}$$ 当$r=s$时,

$$\begin{aligned}K(r,r)&=p(r|X_t=r)\approx P(X_{t+1}\in r\pm h,TA|X_t=r)/2h\\&+P(X_{t+1}\notin r\pm h,\bar{TA}|X_t=r)\\&=\int_{r-h}^{r+h}\alpha(r,y)g(y|r)dy/2h+\int_{y\notin r\pm h}[1-\alpha(r,y)]g(y|r)dy\end{aligned}$$ 因此我们有

$$\to\alpha(r,r)g(r|r)+\int[1-\alpha(r,y)]g(y|r)dy,h\to 0.$$

$K(r,s)=\alpha(r,s)g(s|r)+I(r=s)\int[1-\alpha(r,y)]g(y|r)dy.$ 从而对$r=s$显然细致方程成立. 对任意$r\neq s$有

$$\begin{aligned}K(r,s)f(r)&=\alpha(r,s)g(s|r)=\min\{1,\frac{f(s)g(r|s)}{f(r)g(s|r)}\}g(s|r)f(r)\\&=min\{g(s|r)f(r),f(s)g(r|s)\}=\alpha(s,r)g(r|s)f(s)\\&=K(s,r)f(s).\end{aligned}$$

因此, f满足细致平衡方程. 从而f为平稳分布.

例1 使用MH抽样方法从Rayleigh分布中抽样. Rayleigh分布的密度为 $f(x)=\frac{x}{\sigma^2}e^{-x^2/(2\sigma^2)},x\geq 0,\sigma>0.$

取自由度为X_t的χ^2分布为提议分布, 则使用MH算法如下:

1. 令$g(\cdot|X)$为$\chi^2(df=X)$.
2. 从$\chi^2(1)$中产生X_0, 并存在x[1]中.
3. 对$i=2,\cdots,N$, 重复

(a) 从$\chi^2(df=X_t)=\chi^2(df=\mathrm{x}[i-1])$中产生$Y$.

(b) 产生$U\sim U(0,1)$.

(c) 在$X_t=\mathrm{x}[i-1]$, 计算 $r(X_t,Y)=\frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)}$, 其中$f$为Rayleigh密度. $g(Y|X_t)$为$\chi^2(df=X_t)$的密度在Y处的值, $g(X_t|Y)$ 为$\chi^2(df=Y)$的密度在X_t处的值. 若$U\leq r(X_t,Y)$, 则接受Y, 令$X_{t+1}=Y$; 否则令$X_{t+1}=X_t$. 将X_{t+1}存在x[i]里.

(d) 增加t 在密度f中的常数可以在计算r中抵消, 因此 $r(x_t,y)=\frac{f(y)g(x_t|y)}{f(x_t)g(y|x_t)}=\frac{ye^{-y^2/2\sigma^2}}{x_te^{-x_t^2/2\sigma^2}}\times\frac{\Gamma(x_t/2)2^{x_t/2}y^{y/2-1}e^{-x_t/2}}{\Gamma(y/2)2^{y/2}x_t^{x_t/2-1}e^{-y/2}}$. 在此例中, 我们还是计算整个密度的某点的值来计算r. 下面的代码计算Rayleigh密度在某点的值:

```
f <- function(x, sigma) {
    if (any(x < 0)) return (0)
    stopifnot(sigma > 0)
    return((x / sigma^2) * exp(-x^2 / (2*sigma^2)))
}
```

下面我们产生$\sigma^2=4$的Rayleigh分布随机数. 使用的提议分布为自由度是$xt=\mathrm{x}[i-1]$的 $\chi^2(df=xt)$分布:

```
xt<-x[i-1]
y<-rchisq(1,df=xt)
```

在计算$r(X_{i-1},Y)$中,分子和分母分别用变量**num**和**den**表示.记数变量**k**记录了 候选点被拒绝的次数.

```
m <- 10000
sigma <- 4
x <- numeric(m)
x[1] <- rchisq(1, df=1)
k <- 0
u <- runif(m)
for (i in 2:m) {
    xt <- x[i-1]
    y <- rchisq(1, df = xt)
    num <- f(y, sigma) * dchisq(xt, df = y)
    den <- f(xt, sigma) * dchisq(y, df = xt)
    if (u[i] <= num/den) x[i] <- y else {
         x[i] <- xt
         k <- k+1     #y is rejected
```

```
            }
        }
    print(k)
```

大约40%的候选点被拒绝了. 因此产生链的这个方法有些效率不高. 我们使用样本对时间作图 (称为trace plot), 来观测其样本路径图:

```
index <- 5000:5500
y1 <- x[index]
plot(index, y1, type="l", main="", ylab="x")
```

注意在候选点被拒绝的时间点上链没有移动, 因此图中有很多短的水平平移.

本例中我们的目的是说明MH算法的应用, 对Rayleigh分布, 有更高效率的产生随机数方法. 比如Rayleigh分布的分位数 可以表示为 $x_q = F^{-1}(q) = \sigma[-2log(1-q)]^{1/2}, 0 < q < 1$. 因此可以使用逆变换方法生成随机数.

例2 比较Rayleigh分布的分位数和MH算法下得到样本分位数.(QQ图)

```
b <- 2001       #discard the burnin sample
y <- x[b:m]
a <- ppoints(100)
QR <- sigma * sqrt(-2 * log(1 - a))  #quantiles of Rayleigh
Q <- quantile(x, a)
qqplot(QR, Q, main="",
    xlab="Rayleigh Quantiles", ylab="Sample Quantiles")
hist(y, breaks="scott", main="", xlab="", freq=FALSE)
lines(QR, f(QR, 4))
```

从图上可以看出,样本分位数和理论分位数拟合较好, 直方图也显示出较好的拟合性.

The Metropolis Sampler

MH算法是Metropolis抽样方法的推广. 在Metropolis算法中, 提议分布是对称的. 即$g(\cdot|X_t)$ 满足 $g(X|Y) = g(Y|X)$, 因此接受概率为 $\alpha(X_t, Y) = \min\{1, \frac{f(Y)}{f(Xt)}\}$.

Random Walk Metropolis

随机游动Metropolis抽样方法是Metropolis方法的一个例子. 假设候选点Y从一个对称的提议分布 $g(Y|X_t) = g(|X_t - Y|)$ 中产生的. 则在每一次迭代中, 从$g(\cdot)$中产生一个增量Z, 然后$Y = X_t + Z$. 比如增量Z可以从标准正态分布中产生, 此时候选点$Y|X_t \sim N(X_t, \sigma^2), \sigma^2 > 0$.

随机游动Metropolis算法下得到的链, 其收敛性常常对刻度参数的选择比较敏感. 当增量的方差太大时, 大部分的候选点 会被拒绝, 此时算法的效率很低. 如果增量的方差太小, 则候选点就几乎都被接受, 因此此时随机游动Metropolis算法 下得到的链就几乎是随机游动了, 这也是效率较低. 一种选择刻度参数的方法是监视接受率, 拒绝率应该 在区间[0.15,0.5]之内才可以保证得到的链有较好的性质.

Reference

Robert G.O., Gelman, A., Gilks, W.R., 1996, Weak convergence and optimal scaling of random walk Metropolis algorithms, Annals of Applied Probability, 7:110-20

例3 随机游动Metropolis 使用提议分布$N(X_t, \sigma^2)$和随机游动Metropolis算法产生自由度为ν的t分布随机数. 并对不同的方差σ^2重复此过程.

t_ν的密度正比于$(1+x^2/\nu)^{-(\nu+1)/2}$, 因此 $\alpha(x_t,y)=\min\{1,\frac{f(y)}{f(x_t)}\}=\min\{1,\frac{(1+y^2/\nu)^{-(\nu+1)/2}}{(1+x_t^2/\nu)^{-(\nu+1)/2}}\}$. 下面我们仍然使用$dt$来计算$t$密度在给定点处的值.

```
rw.Metropolis <- function(n, sigma, x0, N) {
   # n: degree of freedom of t distribution
   # sigma:  standard variance of proposal distribution N(xt,sigma)
   # x0: initial value
   # N: size of random numbers required.
    x <- numeric(N)
    x[1] <- x0
    u <- runif(N)
    k <- 0
    for (i in 2:N) {
        y <- rnorm(1, x[i-1], sigma)
            if (u[i] <= (dt(y, n) / dt(x[i-1], n)))
            x[i] <- y  else {
                x[i] <- x[i-1]
                k <- k + 1
            }
        }
    return(list(x=x, k=k))
    }
n <- 4  #degrees of freedom for target Student t dist.
N <- 2000
sigma <- c(.05, .5, 2,  16)
x0 <- 25
rw1 <- rw.Metropolis(n, sigma[1], x0, N)
rw2 <- rw.Metropolis(n, sigma[2], x0, N)
rw3 <- rw.Metropolis(n, sigma[3], x0, N)
rw4 <- rw.Metropolis(n, sigma[4], x0, N)
#rate of candidate points rejected
print(c(rw1$k, rw2$k, rw3$k, rw4$k)/N)
```

上述四种方差的选择, 只有第三个链的拒绝率在区间[0.15,0.5]之间. 我们可以在不同的提议分布方差下, 检查所得链的收敛性.

```
par(mfrow=c(2,2))  #display 4 graphs together
   refline <- qt(c(.025, .975), df=n)
   rw <- cbind(rw1$x, rw2$x, rw3$x,  rw4$x)
   for (j in 1:4) {
       plot(rw[,j], type="l",
            xlab=bquote(sigma == .(round(sigma[j],3))),
            ylab="X", ylim=range(rw[,j]))
       abline(h=refline)
   }
   par(mfrow=c(1,1)) #reset to default
```

可以看出: $\sigma^2=0.05$时, 增量太小, 几乎每个候选点都被接受了, 链在2000次迭代后还没有收敛. $\sigma^2=0.5$时, 链的收敛较慢. $\sigma^2=2$时, 链很快收敛. 而当$\sigma^2=16$时, 接受的概率太小, 使得大部分候选点都被拒绝, 链虽然收敛了, 但是效率很低. (需要更多的运行时间才能得到指定个数个随机数).

例4 随机游动Metropolis算法下的随机数分位数和理论分位数比较

在上例中, 由于理论分布是已知的, 我们可以比较样本分位数和理论分位数.

```
 a <- c(.05, seq(.1, .9, .1), .95)
  Q <- qt(a, n)
  rw <- cbind(rw1$x, rw2$x, rw3$x, rw4$x)
  mc <- rw[501:N, ]
  Qrw <- apply(mc, 2, function(x) quantile(x, a))
  print(round(cbind(Q, Qrw), 3))
  xtable::xtable(round(cbind(Q, Qrw), 3)) #latex format
```

例5 (贝叶斯推断: 一个简单的投资模型) 一般, 不同的投资所得的回报是不独立的. 为了减少风险, 因此使用投资组合以保证有价证券的回报是负相关的. 这里不讨论回报的相关性, 而是对每天组合里的每个证券的收益 进行排序. 假设有5种股票被跟踪记录了250的交易日每天的表现, 在每一个交易日, 收益最大的股票被标记出来. 用X_i表示股票i在250个交易日中胜出的天数, 则记录到得的频数$(x_1,\cdots,x_5)$为随机变量$(X_1,\cdots,X_5)$ 的观测. 基于历史数据, 假设这5种股票在任何给定的一个交易日能胜出的先验机会比率为$1:(1-\beta):(1-2\beta):2\beta:\beta$, 这里$\beta\in(0,0.5)$ 是一个未知的参数. 在有了当前这250个交易日的数据后, 使用Bayes方法对此比例进行更新.

根据前述, $(X_1,\cdots,X_5)$ 在给定β的条件下服从多项分布, 概率向量为 $p=\left(\frac{1}{3},\frac{1-\beta}{3},\frac{1-2\beta}{3},\frac{2\beta}{3},\frac{\beta}{3}\right)$ 因此后验分布为 $P(\beta|x_1,\cdots,x_5)=\frac{250!}{x1!\cdots x5!}p_1^{x1}p_2^{x2}p_3^{x3}p_4^{x4}p_5^{x5}$. 我们不能直接从此后验分布中产生随机数. 一种估计β的方法是产生一个链, 使其平稳分布为此后验分布, 然后从产生的链中抽样来估计β. 我们这里使用随机游动Metropolis算法, 在提议分布为均匀分布下, 产生目标后验分布的随机数. 此时, 接受的概率为 $\alpha(X_t,Y)=\min\{1,\frac{f(Y)}{f(X_t)}\}$. 其中 $\frac{f(Y)}{f(Xt)}=\frac{(1/3)^{x1}((1-Y)/3)^{x2}((1-2Y)/3)^{x3}((2Y)/3)^{x4}(Y/3)^{x5}}{(1/3)^{x1}((1-Xt)/3)^{x2}((1-2Xt)/3)^{x3}((2Xt)/3)^{x4}(Xt/3)^{x5}}$. 此式可以进一步简化.

我们首先生成观测数据:

```
b <- .2          #actual value of beta
w <- .25         #width of the uniform support set
m <- 5000        #length of the chain
burn <- 1000     #burn-in time
days <- 250
x <- numeric(m)  #the chain

# generate the observed frequencies of winners
i <- sample(1:5, size=days, replace=TRUE,
        prob=c(1, 1-b, 1-2*b, 2*b, b))
win <- tabulate(i)
print(win)
```

下面使用随机游动Metropolis算法生产随机数:

```
prob <- function(y, win) {
    # computes (without the constant) the target density
    if (y < 0 || y >= 0.5)
        return (0)
    return((1/3)^win[1] *
        ((1-y)/3)^win[2] * ((1-2*y)/3)^win[3] *
            ((2*y)/3)^win[4] * (y/3)^win[5])
}
# Random Walk Metropolis algorithm
u <- runif(m)          #for accept/reject step
v <- runif(m, -w, w)   #proposal distribution
x[1] <- .25
for (i in 2:m) {
    y <- x[i-1] + v[i]
    if (u[i] <= prob(y, win) / prob(x[i-1], win))
        x[i] <- y
    else
            x[i] <- x[i-1]
}
```

链的路径图显示链已经收敛到目标分布.

```
par(mfrow=c(1,2))
plot(x, type="l")
abline(h=b, v=501, lty=3)
xb <- x[- (1:501)]
hist(xb, prob=TRUE, xlab=bquote(beta), ylab="X", main="")
z <- seq(min(xb), max(xb), length=100)
lines(z, dnorm(z, mean(xb), sd(xb))
```

从而产生的随机数在丢弃初始的burn-in部分后可以用来估计β. β的估计, 样本频率和MCMC估计的多项分布概率由如下代码给出:

```
print(win)
print(round(win/days, 3))
print(round(c(1, 1-b, 1-2*b, 2*b, b)/3, 3))
xb <- x[(burn+1):m]
print(mean(xb))
print(sd(xb))
```

The Independence Sampler

MH算法的另一个特殊情形是独立抽样(Independence Sampler). 独立抽样中的提议分布不依赖于链的前一步状态值. 因此$g(Y|X_t) = g(Y)$, 接受概率为$\alpha(X_t, Y) = \min\{1, \frac{f(Y)g(X_t)}{f(X_t)g(Y)}\}$. 独立抽样方法容易实施, 而且在提议分布和目标分布很接近时也趋于表现很好, 但是当提议分布和目标分布差别很大时, 其表现就较差. Robert<ref>Roberts, G.O., Markov Chain concepts related to sampling algorithms. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, Markov Chain Monte Carlo in Pratice, Pages 45-58. Chapman & Hall, 1996 </ref> 讨论了独立抽样的收敛性, 并且说道:“ 独立抽样算法作为单独的算法很少是有用的”. 但是不管怎么样, 我们 仍然用下例来说明这种方法的应用.

例6 (独立抽样) 假设从一个正态混合分布 $pN(\mu_1, \sigma_1^2) + (1-p)N(\mu_2, \sigma_2^2)$ 中观测到一个样本$(z_1, \cdots, z_n)$. 问题是估计p.

显然, 混合正态的密度为 $f(z) = pf_1(z) + (1-p)f_2(z)$, 其中$f_1, f_2$分别为两个正态的密度.

提议分布的支撑应该和p的取值范围$(0,1)$相同, 最明显的选择就是$Beta$分布. 在没有先验信息的情况下, 可以使用$Beta(1,1)$作为提议分布.($Beta(0,1)$为$U(0,1)$) 候选点Y被接受的概率为 $\alpha(X_t, Y) = \min\{1, \frac{f(Y)g(X_t)}{f(X_t)g(Y)}\}$. 其中$g$为$Beta$提议分布密度. 因此, 若提议分布为$Beta(a,b)$, 则$g(y) \propto y^{a-1}(1-y)^{b-1}$, Y被接受的概率为$\min\{1, f(y)g(x_t)/g(y)f(x_t)\}$, 其中$\frac{f(y)g(x_t)}{g(y)f(x_t)} = \frac{x_t^{a-1}(1-x_t)^{b-1}\prod_{j=1}^{n}[yf_1(z_j)+(1-y)f_2(z_j)]}{y^{a-1}(1-y)^{b-1}\prod_{j=1}^{n}[x_tf_1(z_j)+(1-x_t)f_2(z_j)]}$. 下面我们进行模拟, 提议分布取为$U(0,1)$. 观测数据从下述正态混合中产生 $0.2N(0,1) + 0.8N(5,1)$

```
m <- 5000 #length of chain
xt <- numeric(m)
a <- 1   # a<-5          #parameter of Beta(a,b) proposal dist.
b <- 1   # b<-2          #parameter of Beta(a,b) proposal dist.
p <- .2            #mixing parameter
n <- 30            #sample size
mu <- c(0, 5)      #parameters of the normal densities
sigma <- c(1, 1)
# generate the observed sample
i <- sample(1:2, size=n, replace=TRUE, prob=c(p, 1-p))
x <- rnorm(n, mu[i], sigma[i])
# generate the independence sampler chain
u <- runif(m)
y <- rbeta(m, a, b)      #proposal distribution
xt[1] <- .5
for (i in 2:m) {
    fy <- y[i] * dnorm(x, mu[1], sigma[1]) +
            (1-y[i]) * dnorm(x, mu[2], sigma[2])
    fx <- xt[i-1] * dnorm(x, mu[1], sigma[1]) +
            (1-xt[i-1]) * dnorm(x, mu[2], sigma[2])
    r <- prod(fy / fx) *
           (xt[i-1]^(a-1) * (1-xt[i-1])^(b-1)) /
             (y[i]^(a-1) * (1-y[i])^(b-1))
    if (u[i] <= r) xt[i] <- y[i]
    else  xt[i] <- xt[i-1]
    }
```

链的路径图和丢掉100个burn-in样本后的直方图代码如下

```
plot(xt, type="l", ylab="p")
hist(xt[101:m], main="", xlab="p", prob=TRUE)
print(mean(xt[101:m]))
```

链的时间状态图显示链混合的很好, 很快收敛到平稳分布. 为比较提议分布的不同选择, 我们在提议分布取为$Beta(5,2)$来重复上述过程, 可以看出, 此时产生的链效率较低.

本页面最后编辑于2019年11月27日 (星期三) 11:47。