Lecture 8: Markov Chain Monte Carlo Methods (一)
(马尔科夫蒙特卡罗方法)

肖柳青 教授 博士

上海交通大学 数学科学学院

Contents

Chapter 1
Markov Chain Monte Carlo Methods

1.1 Introduction

MCMC(Markov Chain Monte Carlo) 方法的一般理论框架可以参看 Metropolis et al. (1953)以及 Hastings (1970), 以及其他各种介绍MCMC的专著. 本节我们介绍这种方法的基本思想和应用.

注意在前面介绍Monte Carlo方法估计积分

$$\int_A g(t)dt,$$

是把此积分表示成一个对某个概率密度$f(t)$下的期望. 从而积分估计问题转化成从目标概率密度$f(t)$中产生 随机样本.

在MCMC方法中, 首先建立一个马尔科夫链, 使得$f(t)$为其平稳分布. 则可以运行此马尔科夫链充分长时间直至 收敛到平稳分布, 那么从目标分布$f(t)$中产生随机样本, 就是从达到平稳状态的马尔科夫链中产生其样本路径. 我们将介绍几种建立这样的马尔科夫链的方法: Metropolis算法, Metropolis-Hastings算法, 以及Gibbs抽样方法. 一个好的链应该具有快速混合(rapid mixing)性质—从任意位置出发很快达到平稳分布.

1.1.1 Integration problems in Bayesian inference

Bayesian推断中的许多问题都是MCMC方法的应用. 从Bayesian的观点来看, 模型中的观测变量和参数都是随机变量. 因此, 样本$x = (x_1, \cdots, x_n)$和参数θ联合分布可以表示为

$$f_{x,\theta}(x, \theta) = f_{x|\theta}(x_1, \cdots, x_n)\pi(\theta).$$

从而根据Bayes定理, 可以通过样本$x = (x_1, \cdots, x_n)$的信息对θ的分布进行更新:

$$f_{\theta|x}(\theta|x) == \frac{f_{x|\theta}(x)\pi(\theta)}{\int f_{x|\theta}(x)\pi(\theta)d\theta}.$$

则在后验分布下, $g(\theta)$的期望为

$$Eg(\theta|x) = \int g(\theta) f_{\theta|x}(\theta|x)d\theta = \frac{\int g(\theta) f_{x|\theta}(x)\pi(\theta)d\theta}{\int f_{x|\theta}(x)\pi(\theta)d\theta}.$$

此积分为值为样本x的函数. 因此可以对$g(\theta)$进行推断. 比如$g(\theta) = \theta$时, 则$Eg(\theta|x) = E[\theta|x]$ 可以作为θ的估计.

对此类问题我们考虑一般形式:

$$Eg(Y) = \frac{\int g(t)\pi(t)dt}{\int \pi(t)dt}.$$

这里$\pi(\cdot)$为一个密度或者似然. 若π为一密度, 则此期望即为常见的期望定义: $Eg(Y) = \int g(t) f_Y(t)dt$. 若$\pi$为一似然, 则需要一个正则化常数才可以成为密度. 在贝叶斯分析中, $\pi(\cdot)$为一后验密度. 当$\pi(\cdot)$ 的正则化常数未知时, 此积分

也可以计算. (分子分母中的正则化常数抵消了). 这是非常好的一个性质, 因为在 很多问题下, 正则化常数很难计算.

另一方面, 在很多具体问题中此积分根本没有显示表示, 数值方法也很难计算, 特别是在高维时, 而MCMC方法对此类积分提供了 一个很好的计算方法.

1.1.2 Markov Chain Monte Carlo Integration

积分 $E[g(\theta|x)] = \int g(\theta) f_{\theta|x}(\theta|x) d\theta$ 的Monte Carlo估计为样本均值

$$\bar{g} = \frac{1}{m} \sum_{i=1}^{m} g(x_i),$$

其中$x_1, \cdots, x_m$为从分布$f_{\theta|x}(\theta|x)$中抽取的样本. 当$x_1, \cdots, x_m$独立时, 则可以根据大数律知当样本量n趋向于无穷时, $\bar{g}$收敛到$E[g(\theta|x)$.

但是在一些问题中, 从分布$f_{\theta|x}(\theta|x)$中抽取样本是非常困难的, MCMC方法就是为此目的而诞生的, 其第一个"MC", Markov Chain, 就表示从目标分布中抽样, 第二个"MC", Monte Carlo, 则表示在抽取到的样本下, 应用Monte

Carlo积分方法对积分进行计算.

MCM方法的理论依据在于下述的极限定理:

Theorem 1. 设$\{X_t, t \geq 0\}$为一不可约的,非周期的状态空间为Ω的马氏链, π为平稳分布, π_0是初始分布, 则

$$\pi_t \to \pi, \quad t \to \infty.$$

这里π_t表示马氏链在t时刻的边际分布. 此定理说明了当此马氏链运行充分长时间后($t = n$, n很大), 则在时刻n, X_n的分布为π, 且和初始分布无关.

Theorem 2. *(Markov chain Law of Large Numbers)*若$X_1, X_2, \cdots$为一遍历的平稳分布为π的马氏链值*(*其中每个X_t可以是多维的*)*, 则X_n依分布收敛到 分布为π的随机变量X, 且对任意函数f, 当$E_\pi|f(X)|$存在时, 且$n \to \infty$时, 则

$$\bar{f}_n = \frac{1}{n}\sum_{t=1}^{n} f(X_t) \to E_\pi f(X), \quad a.s.$$

下面我们计算遍历均值$\bar{f}_n$的方差. 记$f^t = f(X_t)$, $\gamma_k = cov(f^t, f^{t+k})$, 则$f^t$的方差为$\sigma^2 = \gamma_0$, k步相关系数$\rho_k = \gamma_k/\sigma^2$. 从而

$$\begin{aligned}
\tau_n^2 &= nVar_\pi(\bar{f}_n) = nVar(\frac{1}{n}\sum_{t=1}^{n} f^t) = \frac{1}{n}Var_\pi(\sum_{t=1}^{n} f^t) \\
&= \frac{1}{n}\sum_{t=1}^{n} Var_\pi(f^t) + 2\frac{1}{n}\sum_{k=1}^{n-t} cov(f^t, f^{t+k}) \\
&= \sigma^2 + 2\sum_{k=1}^{n-1}\frac{n-k}{n}\gamma_k = \sigma^2[1 + 2\sum_{k=1}^{n-1}\frac{n-k}{n}\rho_k].
\end{aligned}$$

可以证明

$$\tau_n^2 \to \tau^2 = \sigma^2[1 + 2\sum_{k=1}^{\infty}\rho_k]$$

. 因此遍历均值$\bar{f}_n$的方差为

$$Var_\pi(\bar{f}_n) = \frac{\sigma^2}{n}[1 + 2\sum_{k=1}^{n-1}\frac{n-k}{n}\rho_k].$$

Theorem 3. 若一个链是一致的几何遍历(转移概率以速度$\lambda^t, 0 < \lambda < 1$收敛)的, 以及 f相对于平稳分布π是平方可积的, 则有

$$\sqrt{n}\frac{\bar{f}_n - E_\pi f(x)}{\tau} \to N(0,1), \quad n \to \infty$$

这个定理对构造平稳分布π的参数的置信区间提供了理论依据.

这些定理说明了:

- 不可约+非周期+正常返$\Longrightarrow$ 唯一的平稳分布.
- LLN: 马氏链的实值函数的遍历均值以概率1收敛到极限分布下的均值.
- CLT: 遍历均值作合适的变化依分布收敛到标准正态分布.

因此, 我们可以建立一个以π为平稳分布的马氏链, 则在运行此链足够长时间后, 该马氏链就会达到 平稳状态, 此时马氏链的值就相当于从分布π中抽取样本. 从而可以根据遍历定理对积分进行估计. MCMC方法的优点在于:

1. 适用于广泛且困难的问题;
2. 问题维数的增加通常不会降低其收敛速度或者使其更复杂.

1.1.3 Markov Chain

离散的状态空间

假设$\{X_t, t \geq 0\}$为一齐次马氏链, 转移概率为$P_{ij} = P(X_{t+1} = j | X_t = i)$, 平稳分布为$\pi$. 则从相反的方向看此马氏链有

$$P(X_t = j | X_{t+1}, X_{t+2}, \ldots) = P(X_t = j | X_{t+1}),$$

即反方向来看, 仍为马氏链(reversed markov chain, 逆向马氏链). 记$P^*_{ij}(t)$为反方向看马氏链从$t+1$时刻转移到t时刻的转移概率, 则

$$\begin{aligned} P^*_{ij}(t) &= P(X_t = j | X_{t+1} = i) = \frac{P(X_{t+1} = i | X_t = j) P(X_t = j)}{P(X_{t+1} = i} \\ &= \frac{P_{ji} \pi_t(j)}{\pi_{t+1}(i)} \end{aligned}$$

这里的π_t表示马氏链在t时刻的边际分布. 从而逆向马氏链一般不再是齐次的. 如果当$t \to \infty$时其收敛到平稳分布, 或者$\pi_0 = \pi$, 则 逆向马氏链就具有了平稳

性, 此时

$$P^*_{ij}(t) = \frac{P_{ji}\pi(j)}{\pi(i)} = P^*_{ij}.$$

若$P^*_{ij} = P_{ij}$, 此时称此马氏链为**可逆的马氏链**(reversible markov chain). 可逆马氏链的可逆性经常 表示为(细致平衡方程, detailed balance equations)

$$\pi(i)P_{ij} = \pi(j)P_{ji}.$$

从而如果一个目标分布π满足此细致平衡方程, 则容易验证其就是平稳分布:

$$\begin{aligned} & \sum_j \pi(i)P_{ij} = \sum_j \pi(j)P_{ji} \\ \Longleftrightarrow \quad & \pi(i)\sum_j P_{ij} = \sum_j \pi(j)P_{ji} \\ \Longleftrightarrow \quad & \pi(i) = \sum_j \pi(j)P_{ji} \qquad \text{平稳分布的定义} \end{aligned}$$

例 设$\{p_j, j \in \Omega, \sum_j p_j = 1\}$ 为我们感兴趣的目标分布, 以及Q为任一不

可约的状态 空间为Ω的马氏链的转移概率矩阵, 且满足对称性条件

$$Q_{ij} = Q_{ji}, \quad i, j \in \Omega$$

按如下方式定义一个马氏链:

1. 从时刻t的状态i转移到下个时刻的状态, 由转移核Q生成一个候选的状态j.

2. 以概率$min\{1, \frac{p_j}{p_i}\}$接受下一时刻的状态为$X_{t+1} = j$, 否则$X_{t+1} = i$.

我们计算一下此种方式下得到的马氏链的转移概率:

(a). $i \neq j$:

$$\begin{aligned} P_{ij} &= P(X_{t+1} = j, TA | X_t = i) = P(X_{t+1} = j | X_t = i) P(TA) \\ &= Q_{ij} min\{1, \frac{p_j}{p_i}\} \end{aligned}$$

"TA" - the event "transition is accepted"

(b). $i = j$:

$$P_{ii} = P(X_{t+1} = i, TA | X_t = i) + P(X_{t+1} \neq i, \overline{TA} | X_t = i)$$

$$= \quad Q_{ii} + \sum_{j:j\neq i} Q_{ij}[1 - min\{1, \frac{p_j}{p_i}\}].$$

则我们可以验证此转移概率和目标分布满足细致平衡方程: 不妨$p_j > p_i (i = j$显然成立), 则

$$p_i P_{ij} = p_i Q_{ij} min\{1, \frac{p_j}{p_i}\} = p_j Q_{ji} min\{1, \frac{p_i}{p_j}\} = p_j P_{ji}.$$

因此这样定义的马氏链为可逆马氏链, 且平稳分布为我们的目标分布.

连续的状态空间

若Ω是连续的, 则对任意的$x, y \in \Omega$, 设$\{X_t\}$为齐次马氏过程, 记转移概率分布为

$$P_{xy} = P(X_{t+1} \leq y | X_t = x) = P(X_1 \leq y | X_0 = x), \quad x, y \in \Omega.$$

若P_{xy}对y绝对连续, 则可以得到条件密度:

$$p_{xy} = \frac{\partial P_{xy}}{\partial y}$$

称为**转移核**. m步转移概率为

$$P_{xy}(m) = P(X_{t+m} \leq y | X_t = x), \quad x, y \in \Omega.$$

从而m步转移核为

$$p_{xy}(t+m) = \frac{\partial P_{xy}(m)}{\partial y}$$

Chapman-Kolmogorov 等式为

$$P_{xy}(t+m) = \int P_{zy}(m) p_{xz}(t) dz,$$

(离散场合:$P_{ij}(m+t) = \sum_{s \in \Omega} P_{is}(m) P_{sj}(t)$)

X_t的边际密度为

$$\pi_t(y) = \int p_{xy} \pi_{t-1}(x) dx.$$

此时细致平衡方程为

$$\pi(y) p_{yx} = \pi(x) p_{xy}.$$

两边积分, 即有

$$\begin{aligned} & \int \pi(x)p_{yx}dx = \int \pi(x)p_{xy}dx \\ \Longleftrightarrow \quad & \pi(y)\int p_{yx}dx = \int \pi(x)p_{xy}dx \\ \Longleftrightarrow \quad & \pi(y) = \int \pi(x)p_{xy}dx \qquad \text{平稳分布的定义} \end{aligned}$$

1.2 The Metropolis-Hastings Algorithm

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.

1.2.1 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)
   (a) 从$g(\cdot|X_t)$中产生Y.
   (b) 从$U(0,1)$中产生U;
   (c) 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 $X_{t+1} = X_t$.

(d) 增加t, 返回到(a).

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,

$$\begin{aligned} K(r,r) &= p(r|X_t = r) \approx P(X_{t+1} \in r \pm h, TA|X_t = r)/2h \\ &+ P(X_{t+1} \notin r \pm h, T\bar{A}|X_t = r) \\ &= \int_{r-h}^{r+h} \alpha(r,y)g(y|r)dy/2h + \int_{y \notin r \pm h} [1 - \alpha(r,y)]g(y|r)dy \\ &\to \alpha(r,r)g(r|r) + \int [1 - \alpha(r,y)]g(y|r)dy, h \to 0. \end{aligned}$$

因此我们有

$$K(r,s) = \alpha(r,s)g(s|r) + I(r = s) \int [1 - \alpha(r,y)]g(y|r)dy.$$

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$\chi^2(df = X_t) = \chi^2(df = \text{x}[i-1])$中产生$Y$.
   (b) 产生$U \sim U(0, 1)$.
   (c) 在$X_t = \text{x}[i-1]$, 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

$$r(x_t, y) = \frac{f(y)g(x_t|y)}{f(x_t)g(y|x_t)} = \frac{ye^{-y^2/2\sigma^2}}{x_t e^{-x_t^2/2\sigma^2}} \times \frac{\Gamma(x_t/2)2^{x_t/2}x_t^{y/2-1}e^{-x_t/2}}{\Gamma(y/2)2^{y/2}y^{x_t/2-1}e^{-y/2}}.$$

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:

↑Code

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

↓Code

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:

↑Code

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

↓Code

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.

↑Code

```
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)
```

↓Code

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:

↑Code

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

↓Code

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)

↑Code

```
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))
```

↓Code

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.

1.2.2 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

$$\alpha(X_t, Y) = \min\{1, \frac{f(Y)}{f(X_t)}\}.$$

1.2.3 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.[1]

[1] 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.

↑Code

```
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)
```

↓Code

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.

↑Code

```
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
```

↓Code

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.

↑Code

```
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
```

↓Code

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

$$P(\beta|x_1, \cdots, x_5) = \frac{250!}{x_1! \cdots x_5!} p_1^{x_1} p_2^{x_2} p_3^{x_3} p_4^{x_4} p_5^{x_5}.$$

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

$$\frac{f(Y)}{f(X_t)} = \frac{(1/3)^{x_1}((1-Y)/3)^{x_2}((1-2Y)/3)^{x_3}((2Y)/3)^{x_4}(Y/3)^{x_5}}{(1/3)^{x_1}((1-X_t)/3)^{x_2}((1-2X_t)/3)^{x_3}((2X_t)/3)^{x_4}(X_t/3)^{x_5}}.$$

此式可以进一步简化.

我们首先生成观测数据:

Code

```
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)
```

↓Code

下面使用随机游动Metropolis算法生产随机数:

↑Code

```
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]
}
```

↓Code

链的路径图显示链已经收敛到目标分布.

↑Code

```
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))
```

↓Code

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:

↑Code

```
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))
```

↓Code

1.2.4 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[2] 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

$$pN(\mu_1, \sigma_1^2) + (1-p)N(\mu_2, \sigma_2^2)$$

[2] 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

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

$$\frac{f(y)g(x_t)}{g(y)f(x_t)} = \frac{x_t^{a-1}(1-x_t)^{b-1}\prod_{j=1}^n [yf_1(z_j) + (1-y)f_2(z_j)]}{y^{a-1}(1-y)^{b-1}\prod_{j=1}^n [x_t f_1(z_j) + (1-x_t)f_2(z_j)]}.$$

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

$$0.2N(0,1)+0.8N(5,1)$$

↑Code

```
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]
    }
```

↓Code

链的路径图和丢掉100个burn-in样本后的直方图代码如下

↑Code

```
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.

1.3 Single-component Metropolis Hastings Algorithms

当状态空间为多维时, 不整体更新X_t, 而是对其分量进行逐个更新, 即称为Single-component Metropolis Hastings, 或者 Component-wise Metropolis Hastings, 或者 Gibbs within Metropolis. 这样做 更方便和更有效率. 记

$$X_t = (X_{t,1}, \ldots, X_{t,k}),$$

$$X_{t,-i} = (X_{t,1}, \ldots, X_{t,i-1}, X_{t,i+1}, \ldots, X_{t,k}).$$

分别表示在第t步链的状态, 以及在第t步除第i个分量外其他分量的状态. $f(x) = f(x_1, \cdots, x_k)$ 为目标分布, $f(x_i|x_{-i}) = \frac{f(x)}{\int f(x_1,\cdots,x_k)dx_i}$表示$X_i$对其他分量的条件密度.

则逐分量的MH算法更新X_t是由k步构成: 令$X_{t,i}$表示在第t次迭代 后X_t第i个分量的状态, 则在第$t+1$步迭代的第i步中, 使用MH算法更新$X_{t,i}$. 做法如下: 对$i = 1, \cdots, k$, 从第i个提议分布$q_i(\cdot|X_{t,i}, X^*_{t,-i})$中产生$Y_i$, 这里

$$X^*_{t,-i} = (X_{t+1,1}, \ldots, X_{t+1,i-1}, X_{t,i+1}, \ldots, X_{t,k}).$$

然后以概率

$$\alpha(X^*_{t,-i}, X_{t,i}, Y_i) = \min\{1, \frac{f(Y_i|X^*_{t,-i})q_i(X^*_{t,i}|Y_i, X^*_{t,-i})}{f(X_{t,i}|X^*_{t,-i})q_i(Y_i|X_{t,i}, X^*_{t,-i})}$$

若Y_i被接受, 则令$X_{t+1,i} = Y_i$; 否则令$X_{t+1,i} = X_{t,i}$.

1.4 Application: Logistic regression

例 7 考虑54位老年人的智力测试成绩(Wechesler Adult Intelligence Scale, WAIS, 0-20分). 研究的兴趣在于发现老年痴呆症.

我们采用如下简单的logistic回归模型:

$$Y_i \sim Bin(1, \pi_i), \quad log\frac{\pi_i}{1-\pi_i} = \beta_0 + x_i\beta_1, \ i = 1, \ldots, 54.$$

则似然函数为

$$\begin{aligned} f(\mathbf{y}|\beta_0, \beta_1) &= \prod_{i=1}^{n} \left(\frac{e^{\beta_0+x_i\beta_1}}{1+e^{\beta_0+x_i\beta_1}}\right)^{y_i} \left(\frac{1}{1+e^{\beta_0+x_i\beta_1}}\right)^{1-y_i} \\ &= exp\{n\bar{y}\beta_0 + \beta_1 \sum_{i=1}^{n} x_i y_i - log(1+e^{\beta_0+x_i\beta_1})\}. \end{aligned}$$

考虑β_0, β_1的先验分布π为独立的正态分布:

$$\beta_j \sim N(\mu_{\beta_j}, \sigma_j^2), j = 0, 1.$$

其中$\mu_{\beta_j}=0,\sigma_j^2$很大, 以表示接近无信息先验. 从而后验分布为

$$\begin{aligned} f(\beta_0,\beta_1|\mathbf{y}) &\propto f(\mathbf{y}|\beta_0,\beta_1)\pi(\beta_0,\beta_1) \\ &\propto exp\{\sum_{i=1}^{n}[(\beta_0+\beta_1 x_i)y_i - log(1+e^{\beta_0+x_i\beta_1})] \\ &\quad -\frac{(\beta_0-\mu_{\beta_0})^2}{2\sigma_0^2}-\frac{(\beta_1-\mu_{\beta_1})^2}{2\sigma_1^2}\}. \end{aligned}$$

从而我们需要从此分布中产生随机数.

我们考虑如下三种抽样方法.

1. 独立抽样

提议分布取为

$$\beta' \sim q = N(\beta, diag\{\bar{s}_{\beta_0}^2, \bar{s}_{\beta_1}^2\})$$

则算法如下

对$t=1,\cdots,T$:

1. 令$\beta=(\beta_0^{(t-1)},\beta_1^{(t-1)})$.

2. 从提议分布$N(\beta, diag\{\bar{s}^2_{\beta_0}, \bar{s}^2_{\beta_1}\})$产生候选点$\beta'$.

3. 计算接受概率

$$\alpha(\beta, \beta') = \min\{1, \frac{f(\mathbf{y}|\beta'_0, \beta'_1)q(\beta'_0, \beta'_1)}{f(\mathbf{y}|\beta_0, \beta_1)q(\beta_0, \beta_1)}\}$$

4. 以概率$\alpha(\beta, \beta')$接受β', 并令$\beta^{(t)} = \beta'$; 否则令$\beta^{(t)} = \beta$. R程序代码如下

↑Code

```
wais-read.table (' wais. txt'  ,header=TRUE)
y<-wais[,2]; x<-wais[,1]
m <- 55000    #length of chain
mu.beta<-c(0,0); sigma.beta<-c(100,100)
prop.s<-c(0.1,0.1)  #proposal distribution standard variance
beta <- matrix(nrow=m, ncol=2)
acc.prob <- 0
current.beta<-c(0,0)
for (t in 1:m){
    prop.beta<- rnorm(2, current.beta, prop.s )
    cur.eta<-current.beta[1]+current.beta[2]*x
    prop.eta<-prop.beta[1]+prop.beta[2]*x
    loga <-(sum(y*prop.eta-log(1+exp(prop.eta)))
```

```
                -sum(y*cur.eta-log(1+exp(cur.eta)))
                +sum(dnorm(prop.beta, mu.beta,s.beta,log=TRUE))
                -sum(dnorm(current.beta,mu.beta,s.beta,log=TRUE)))
    u<-runif(1)
    u<-log(u)
    if( u < loga) {
           current.beta<-prop.beta
           acc.prob <- acc.prob+1
       }
    beta[t,]<-current.beta
 }
acc.prob<-acc.prob/m
acc.prob
```

↓Code

我们可以画一下链的样本路径图, 以及遍历均值图:

↑Code

```
# convergence diagnostics plot
  erg.mean<-function( x ){ # compute ergodic mean
        n<-length(x)
```

```
        result<-cumsum(x)/cumsum(rep(1,n))
  }
burnin<-15000
idx<-seq(1,m,50)
idx2<-seq(burnin+1,m)
par(mfrow=c(2,2))
plot(idx,beta[idx,1],type="l",xlab="Iterations",ylab="Values of beta0")
plot(idx,beta[idx,2],type="l",xlab="Iterations",ylab="Values of beta1")

ergbeta0<-erg.mean(beta[,1])
ergbeta02<-erg.mean(beta[idx2,1])
ylims0<-range(c(ergbeta0,ergbeta02))
ergbeta1<-erg.mean(beta[,2])
ergbeta12<-erg.mean(beta[idx2,2])
ylims1<-range(c(ergbeta1,ergbeta12))
plot(idx , ergbeta0[idx], type='l', ylab='Values of beta0', xlab='Iterations',
     main='(c) Ergodic Mean Plot of beta0', ylim=ylims0)
lines(idx2, ergbeta02[idx2-burnin], col=2, lty=2)
plot(idx, ergbeta1[idx], type='l', ylab='Values of beta1', xlab='Iterations',
     main='(d) Ergodic Mean Plot of beta1', ylim=ylims1)
lines(idx2, ergbeta12[idx2-burnin], col=2, lty=2)
```

```
apply(beta[(burnin+1):m,],2,mean)
apply(beta[(burnin+1):m,],2,sd)
```

↓Code

注意到在独立抽样方法产生的链中, β_0和β_1有很强的负相关性:

`cor(beta[(burnin+1):m,1],beta[(burnin+1):m,2])=-0.954`.

这是链收敛很慢的原因. 在高度相关的空间上使用独立的提议分布导致链的混合效率低下.

MH算法: 多元正态提议分布

在独立抽样中, 我们使用的提议分布是相互独立的, 从而导致链的混合效率低下. 因此, 自然而然的我们 可以考虑非独立的提议分布. 提议分布相关阵应该和后验分布的相关阵类似. 为此, 可以考虑利用 Fisher 信息阵$H(\beta)$, 提议分布取为

$$\beta' \sim q = N(\beta, c_\beta^2 [H(\beta)]^{-1}).$$

其中c_β为一调节参数, 以使算法达到预设的接受率. 由似然函数, 容易计算得到Fisher信息阵为

$$H(\beta) = X^T diag(h_i) X + \Sigma_\beta^{-1},$$

其中Σ_β为β的先验协方差矩阵, $h_i = exp(\beta_0 + \beta_1 x_i)/(1 + exp(\beta_0 + \beta_1 x_i))^2$. $X = (1_n, x)$为一$2 \times n$的矩阵.

从而MH算法如下

对$t = 1, \cdots, T$:

1. 令$\beta = (\beta_0^{(t-1)}, \beta_1^{(t-1)})$.

2. 计算Fisher信息阵

$$diag(h_i) = diag\{\frac{exp(\beta_0 + \beta_1 x_i)}{(1 + exp(\beta_0 + \beta_1 x_i))^2}, \}$$

$$H(\beta) = X^T diag(h_i) X + \Sigma_\beta^{-1}, \quad S_\beta = c_\beta^2 [H(\beta)]^{-1}.$$

2. 从提议分布$N(\beta, S_\beta)$产生候选点β'.

3. 计算接受概率

$$\alpha(\beta,\beta')=\min\{1,\frac{f(\mathbf{y}|\beta_0',\beta_1')\pi(\beta_0',\beta_1')}{f(\mathbf{y}|\beta_0,\beta_1)\pi(\beta_0,\beta_1)}\frac{q(\beta|\beta',S_{\beta'})}{q(\beta'|\beta,S_\beta)}\}$$

4. 以概率$\alpha(\beta,\beta')$接受β', 并令$\beta^{(t)}=\beta'$; 否则令$\beta^{(t)}=\beta$. R程序代码如下

↑Code

```
calculate.loglike<-function(b,X=x,Y=y){
 x<-X; y<-y
 n<-length(x); X<-cbind( rep(1,n), x )
 precision<-700
 eta<-b[1]+b[2]*x
 logq <- log(1+exp(eta))
 logq[eta>precision]<-eta[eta>precision]
 loglike<- sum(  y*eta  - logq )
 eta[eta>precision]<-precision
 h  <- 1/((1+exp(-eta))*(1+exp(eta)))
 H  <- t(X) %*% diag( h ) %*% X
 return( list(loglike=loglike, H=H) )
}
#-------------------------------------
```

```
library(MASS)
y=wais[,2]
x=wais[,1]
prop.sd=0.3
m=2500
beta0=c(0,0)
n<-length(y)
X<-cbind(rep(1,n), x )
mu.beta<-c(0,0)
s.beta<-c(100,100)
c.beta<- prop.sd
beta <- matrix(nrow=Iterations, ncol=2)
acc.prob <- 0
current.beta<-beta0
for (t in 1:m){
 cur<-calculate.loglike( current.beta )
 cur.T<-(1/c.beta^2)*(cur$H+diag(1/s.beta^2))
 prop.beta<- mvrnorm( 1, current.beta, solve(cur.T))
 prop<-calculate.loglike( prop.beta )
 prop.T  <- (1/c.beta^2)* (prop$H+diag(1/s.beta^2))
```

```
 loga <-( prop$loglike-cur$loglike
 +sum(dnorm(prop.beta,mu.beta,s.beta,log=TRUE))
 -sum(dnorm(current.beta,mu.beta,s.beta,log=TRUE))
 + as.numeric(0.5*log( det(prop.T) )
 - 0.5 * t(current.beta - prop.beta) %*% prop.T %*% (current.beta - prop.beta))
 - as.numeric(0.5*log( det(cur.T ) )
 - 0.5 * t(prop.beta - current.beta) %*% cur.T  %*% (prop.beta- current.beta ))
   u<-runif(1)
   u<-log(u)
   if( u < loga ) {
        current.beta<-prop.beta
        acc.prob <- acc.prob+1
        }
    beta[t,]<-current.beta
    }
print(acc.prob/m)
```

↓Code

在计算$exp(\eta)$时, 由于在R中, $exp(710) = Inf$, 因此我们对$\eta > 700$, 取近似$log(1 + exp(\eta)) \approx \eta$.

对链的收敛诊断可以看出, 链混合的效率很高.

逐分量的MH算法

在MH算法中, 按分量进行逐个更新, 其优势在与应用方便, 不需要考虑调节参数. 算法如下:

对$t = 1, \cdots, T$:

1. 令$\beta = (\beta_0^{(t-1)}, \beta_1^{(t-1)})^T$.
2. 从提议分布$N(\beta_0, \bar{s}^2_{\beta_0})$产生候选点$\beta_0'$.
3. 令$\beta' = (\beta_0', \beta_1^{(t-1)})^T$, 计算接受概率

$$\alpha_0(\beta, \beta') = \min\{1, \frac{f(\mathbf{y}|\beta_0', \beta_1)\pi(\beta_0', \beta_1)}{f(\mathbf{y}|\beta_0, \beta_1)\pi(\beta_0, \beta_1)}\}$$

4. 以概率$\alpha_0(\beta, \beta')$接受$\beta = \beta'$; 否则保持其值不变.
5. 从提议分布$N(\beta_1, \bar{s}^2_{\beta_1})$产生候选点$\beta_1'$.
6. 令$\beta' = (\beta_0, \beta_1')^T$, 计算接受概率

$$\alpha_1(\beta, \beta') = \min\{1, \frac{f(\mathbf{y}|\beta_0, \beta_1')\pi(\beta_0, \beta_1')}{f(\mathbf{y}|\beta_0, \beta_1)\pi(\beta_0, \beta_1)}\}$$

7. 以概率$\alpha_1(\beta, \beta')$接受$\beta = \beta'$; 否则保持其值不变.

8. 令$\beta^{(t)} = \beta$.

R实现代码如下

↑Code

```
y<-wais[,2]
x<-wais[,1]
m<-10000
beta0<-c(0,0)            #initial value
mu.beta<-c(0,0)          # prior
s.beta<-c(100,100)       # prior
prop.s<-c(1.75,0.2)      # sd of proposal normal
beta <- matrix(nrow=m, ncol=2)
acc.prob <-c(0,0)
current.beta<-beta0
for (t in 1:m){
  for (j in 1:2){
    prop.beta<- current.beta
    prop.beta[j]<- rnorm( 1, current.beta[j], prop.s[j] )
    cur.eta <-current.beta[1]+current.beta[2]*x
    prop.eta<-prop.beta[1]+prop.beta[2]*x
```

```
    if(sum(prop.eta>700)>0) {print(t); stop;}
    if(sum(cur.eta >700)>0) {print(t); stop;}
    loga <-(sum(y*prop.eta-log(1+exp(prop.eta)))
             -sum(y*cur.eta-log(1+exp(cur.eta)))
             +sum(dnorm(prop.beta,mu.beta,s.beta,log=TRUE))
             -sum(dnorm(current.beta,mu.beta,s.beta,log=TRUE)))
    u<-runif(1)
    u<-log(u)
    if(u< loga){
         current.beta<-prop.beta
         acc.prob[j] <- acc.prob[j]+1
            }
    }
    beta[t,]<-current.beta
}
print(acc.prob/m)
```

↓Code

链的收敛诊断显示了相比于独立的MH算法, 混合的效率要高的多.