第3章 采样方法

CHAPTER 3

上一章介绍了采样的概念。例如，想知道一所大学里所有男生的平均身高。但是因为学校里的男生可能有上万人之多，所以为每个人都测量一下身高存在很大困难，于是从每个学院随机挑选出100名男生来作为样本，这个过程就是采样。然而，本章将要讨论的采样则有另外一层含义。现实中的很多问题可能求解起来是相当困难的。这时就可能会想到利用计算机模拟的方法来帮助求解。在使用计算机进行模拟时，所说的采样，是指从一个概率分布中生成观察值的方法。而这个分布通常是由其概率密度函数来表示的。但即使在已知概率密度函数的情况下，让计算机自动生成观测值也不是一件容易的事情。

3.1 蒙特卡洛法求定积分

蒙特卡洛(Monte Carlo)法是一类随机算法的统称。它是20世纪40年代中期由于科学技术的发展，尤其是电子计算机的发明，而被提出并发扬光大的一种以概率统计理论为基础的数值计算方法。它的核心思想就是使用随机数(或更准确地说是伪随机数)来解决一些复杂的计算问题。现今，蒙特卡洛法已经在诸多领域展现出了超强的能力。本节，我们将通过蒙特卡洛法最为常见的一种应用——求解定积分，来演示这类算法的核心思想。

3.1.1 无意识统计学家法则

作为一个预备知识，先来介绍一下无意识统计学家法则(Law of the Unconscious Statistician，LOTUS)。在概率论与统计学中，如果知道随机变量 X 的概率分布，但是并不显式地知道函数 $g(X)$ 的分布，那么 LOTUS 就是一个可以用来计算关于随机变量 X 的函数 $g(X)$ 之期望的定理。该法则的具体形式依赖于随机变量 X 之概率分布的描述形式。

如果随机变量 X 的分布是离散的，而且我们知道它的 PMF 是 f_X，但不知道 $f_{g(X)}$，那么 $g(X)$ 的期望是

$$E[g(X)] = \sum_x g(x) f_X(x)$$

其中和式是在取遍 X 的所有可能之值 x 后求得。

如果随机变量 X 的分布是连续的，而且我们知道它的 PDF 是 f_X，但不知道 $f_{g(X)}$，那么 $g(X)$ 的期望是

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x)$$

简而言之，已知随机变量 X 的概率分布，但不知道 $g(X)$的分布，此时用 LOTUS 公式能计算出函数 $g(X)$的数学期望。其实就是在计算期望时，用已知的 X 的 PDF(或 PMF)代替未知的 $g(X)$的 PDF(或 PMF)。

3.1.2 投点法

投点法是讲解蒙特卡洛法基本思想的一个最基础也最直观的实例。这个方法也常常被用来求圆周率 π。现在我们用它来求函数的定积分。如图 3-1 所示，有一个函数 $f(x)$，若要求它从 a 到 b 的定积分，其实就是求曲线下方的面积。

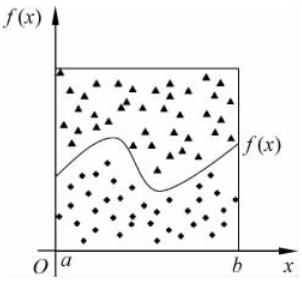

图 3-1 投点法求定积分

可以用一个比较容易算得面积的矩型罩在函数的积分区间上(假设其面积为 Area)。然后随机地向这个矩形框里面投点，其中落在函数 $f(x)$下方的点为菱形，其他点为三角形。然后统计菱形点的数量占所有点(菱形＋三角形)数量的比例为 r，那么就可以据此估算出函数 $f(x)$从 a 到 b 的定积分为 Area×r。

注意由蒙特卡洛法得出的值并不是一个精确值，而是一个近似值。而且当投点的数量越来越大时，这个近似值也越接近真实值。

3.1.3 期望法

下面来重点介绍利用蒙特卡洛法求定积分的第二种方法——期望法，有时也称为平均值法。

任取一组相互独立、同分布的随机变量$\{X_i\}$，X_i 在$[a,b]$上服从分布律 f_X，也就是说 f_X 是随机变量 X 的 PDF(或 PMF)。令 $g^*(x)=\dfrac{g(x)}{f_X(x)}$，则 $g^*(X_i)$也是一组独立同分布的随机变量，而且因为 $g^*(x)$是关于 x 的函数，所以根据 LOTUS 可得

$$E[g^*(X_i)]=\int_a^b g^*(x)f_X(x)\mathrm{d}x=\int_a^b g(x)\mathrm{d}x=I$$

由强大数定理

$$Pr\left(\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}g^*(X_i)=I\right)=1$$

若选

$$\bar{I}=\frac{1}{N}\sum_{i=1}^{N}g^*(X_i)$$

则 $\bar{I}$ 依概率 1 收敛到 I。平均值法就用 $\bar{I}$ 作为 I 的近似值。

假设要计算的积分有如下形式

$$I=\int_a^b g(x)\mathrm{d}x$$

其中，被积函数 $g(x)$在区间$[a,b]$上可积。任意选择一个有简便办法可以进行抽样的概率密度函数 $f_X(x)$，使其满足下列条件：

(1) 当 $g(x)\neq 0$ 时，$f_X(x)\neq 0$，$a\leqslant x\leqslant b$；

(2) $\int_a^b f_X(x)\mathrm{d}x=1$。

如果记

$$g^*(x)=\begin{cases}\dfrac{g(x)}{f_X(x)}, & f_X(x)\neq 0\\ 0, & f_X(x)=0\end{cases}$$

那么原积分式可以写成

$$I=\int_a^b g^*(x)f_X(x)\mathrm{d}x$$

因而求积分的步骤是：

(1) 产生服从分布律 f_X 的随机变量 X_i，$i=1,2,\cdots,N$；

(2) 计算均值

$$\bar{I}=\frac{1}{N}\sum_{i=1}^{N}g^*(X_i)$$

并用它作为 I 的近似值，即 $I\approx\bar{I}$。

如果 a，b 为有限值，那么 f_X 可取作为均匀分布

$$f_X(x)=\begin{cases}\dfrac{1}{b-a}, & a\leqslant x\leqslant b\\ 0, & \text{其他}\end{cases}$$

此时原来的积分式变为

$$I=(b-a)\int_a^b g(x)\frac{1}{b-a}\mathrm{d}x$$

因而求积分的步骤是：

(1) 产生$[a,b]$上的均匀分布随机变量 X_i，$i=1,2,\cdots,N$；

(2) 计算均值

$$\bar{I}=\frac{b-a}{N}\sum_{i=1}^{N}g(X_i)$$

并用它作为 I 的近似值，即 $I\approx\bar{I}$。

最后来看一下平均值法的直观解释。注意积分的几何意义就是$[a,b]$区间曲线下方的面积，如图 3-2 所示。

当在$[a,b]$随机取一点 x 时，它对应的函数值就是 $f(x)$，然后便可以用 $f(x)\cdot(b-a)$ 来粗略估计曲线下方的面积(也就是积分)，如图 3-3 所示，当然这种估计(或近似)是非常粗略的。

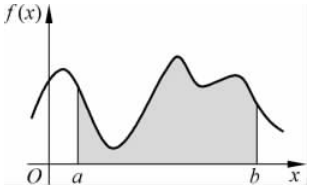

图 3-2　积分的几何意义

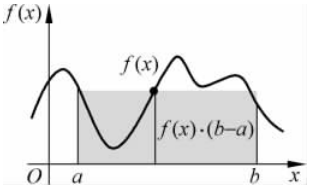

图 3-3　对积分值进行粗略估计

于是我们想到在$[a,b]$随机取一系列点 x_i 时(x_i 满足均匀分布)，然后把估算出来的面积取平均来作为积分估计的一个更好的近似值，如图 3-4 所示。可以想象，如果这样的采样

点越来越多，那么对于这个积分的估计也就越来越接近。

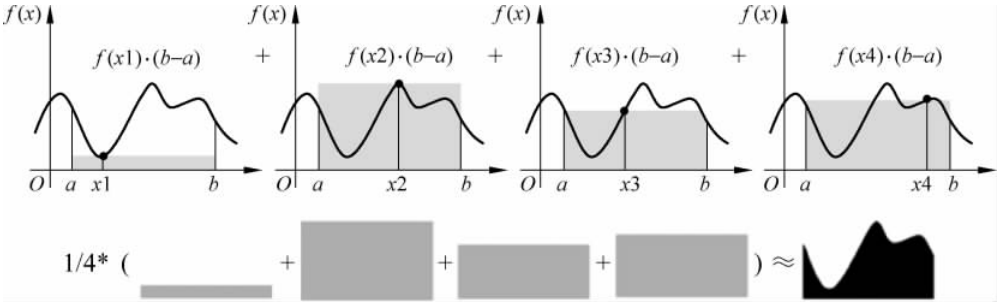

图 3-4　对积分值进行估计

按照上面这个思路，得到积分公式为

$$\bar{I} = (b-a)\frac{1}{N}\sum_{i=0}^{N-1} f(X_i) = \frac{1}{N}\sum_{i=0}^{N-1}\frac{f(X_i)}{\frac{1}{b-a}}$$

其中，$\frac{1}{b-a}$ 就是均匀分布的 PMF。这跟之前推导出来的蒙特卡洛积分公式是一致的。

3.2　蒙特卡洛采样

在 3.1 节中，通过求解定积分这个具体的例子来演示了蒙特卡洛方法的基本思想，而其中的核心就是使用随机数。当所求解问题可以转化为某种随机分布的特征数（如随机事件出现的概率，或者随机变量的期望值等）时，往往就可以考虑使用蒙特卡洛方法。通过随机抽样的方法，以随机事件出现的频率估计其概率，或者以抽样的数字特征估算随机变量的数字特征，并将其作为问题的解。这种方法多用于求解复杂的高维积分问题。

实际应用中，所要面对的第一个问题就是如何抽样？注意，在计算机模拟时，这里所说的抽样其实是指从一个概率分布中生成观察值（observations）的方法。而这个分布通常是由其概率密度函数来表示的。前面曾经提过，即使在已知 PDF 的情况下，让计算机自动生成观测值也不是一件容易的事情。从本质上来说，计算机只能实现对均匀分布的采样。幸运的是，仍然可以在此基础上对更为复杂的分布进行采样。

3.2.1　逆采样

比较简单的一种情况是，可以通过 PDF 与 CDF 之间的关系，求出相应的 CDF。或者根本就不知道 PDF，但是知道 CDF。此时就可以使用 CDF 反函数（及分位数函数）的方法来进行采样。这种方法又称为逆变换采样（Inverse Transform Sampling）。

假设已经得到了 CDF 的反函数 $F^{-1}(u)$，如果想得到 m 个观察值，则重复下面的步骤 m 次：

（1）从 $U(0,1)$ 中随机生成一个值（计算机可以实现从均匀分布中采样），用 u 表示；

（2）计算 $F^{-1}(u)$ 的值 x，则 x 就是从目标分布 $f(x)$ 中得出的一个采样点。

面对一个具有复杂表达式的函数，逆变换采样法真有效吗？来看一个例子，假设现在希望从具有下面这个 PDF 的分布中采样

$$f(x)=\begin{cases}8x, & 0\leqslant x<0.25\\ \dfrac{8}{3}-\dfrac{8x}{3}, & 0.25\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

可以算得相应的 CDF 为

$$F(x)=\begin{cases}0, & x<0\\ 4x^2, & 0\leqslant x<0.25\\ \dfrac{8x}{3}-\dfrac{4x^2}{3}-\dfrac{1}{3}, & 0.25\leqslant x\leqslant 1\\ 1, & x>1\end{cases}$$

对于 $u\in[0,1]$，上述 CDF 的反函数为

$$F^{-1}(u)=\begin{cases}\dfrac{\sqrt{u}}{2}, & 0\leqslant x<0.25\\ 1-\dfrac{\sqrt{3(1-u)}}{2}, & 0.25\leqslant x\leqslant 1\end{cases}$$

下面在 R 中利用上面的方法采样 10000 个点，并以此来演示抽样的效果。

```
m <- 10000
u <- runif(m, 0, 1)
invcdf.func <- function(u) {
+ if (u >= 0 && u < 0.25)
+       sqrt(u)/2
+     else if (u >= 0.25 && u <= 1)
+       1 - sqrt(3 * (1 - u))/2
+ }
x <- unlist(lapply(u, invcdf.func))

curve(8 * x, from = 0, to = 0.25, xlim = c(0,1), ylim = c(0,2),
+ col = "red",xlab = "", ylab = "")
par(new = TRUE)
curve((8/3) - (8/3) * x, from = 0.25, to = 1, xlim = c(0,1), ylim = c(0,2),
+ col = "red",xlab = "", ylab = "")
par(new = TRUE)
plot(density(x), xlim = c(0,1), ylim = c(0,2),
+ col = "blue", xlab = "x", ylab = "density")
```

将所得结果与真实的 PDF 函数图形进行对照，如图 3-5 所示。可见由逆变换采样法得到的点所呈现处理的分布与目标分布非常吻合。

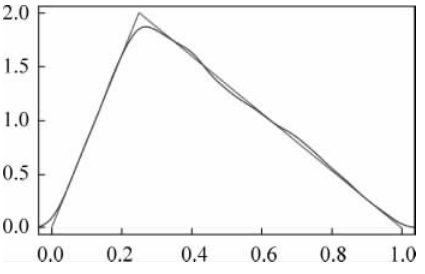

图 3-5　逆变换采样举例

下面再举一个稍微复杂一点的例子，已知分布的 PDF 如下

$$h(x)=\frac{2m^2}{(1-m^2)x^3},\quad x\in[m,1]$$

可以算得相应的 CDF 为

$$H(x)=\int_{-\infty}^{x}h(t)\mathrm{d}t\begin{cases}0, & x<m\\ \dfrac{1}{1-m^2}-\dfrac{m^2}{(1-m^2)x^2}, & x\in[m,1]\\ 1, & x>1\end{cases}$$

对于 $u\in[0,1]$,它的反函数为

$$H^{-1}(u)=\sqrt{\frac{m^2}{1-(1-m^2)u}}$$

同样,给出 R 中的示例代码如下。

```
invcdf <- function(u, m) {
    return(sqrt(m^2/(1 - (1 - m^2) * u)))
}

sample1 <- sapply(runif(1000), invcdf, m = .5)
```

下面这段代码利用 R 中提供的一些内置函数实现了已知 PDF 时基于逆变换方法的采样,将新定义的函数命名为 samplepdf()。当然,对于那些过于复杂的 PDF 函数(例如很难积分的),samplepdf()确实有力所不及的情况。但是对于标准的常规 PDF,该函数的效果还是不错的。

```
endsign <- function(f, sign = 1) {
    b <- sign
    while (sign * f(b) < 0) b <- 10 * b
    return(b)
}

samplepdf <- function(n, pdf, ..., spdf.lower = -Inf, spdf.upper = Inf) {
    vpdf <- function(v) sapply(v, pdf, ...) # vectorize
    cdf <- function(x) integrate(vpdf, spdf.lower, x)$value
    invcdf <- function(u) {
        subcdf <- function(t) cdf(t) - u
        if (spdf.lower == -Inf)
            spdf.lower <- endsign(subcdf, -1)
        if (spdf.upper == Inf)
            spdf.upper <- endsign(subcdf)
        return(uniroot(subcdf, c(spdf.lower, spdf.upper))$root)
    }
    sapply(runif(n), invcdf)
}
```

下面就用 samplepdf()函数来对上面给定的 h(x)进行采样,然后再跟之前所得结果进行对比。

```
h <- function(t, m) {
    if (t >= m & t <= 1)
        return(2 * m^2/(1 - m^2)/t^3)
    return(0)
```

```
}

sample2 <- samplepdf(1000, h, m = .5)

plot(density(sample1), xlim = c(0.4, 1.1), ylim = c(0, 4),
+ col = "red", xlab = "", ylab = "", main = "")
par(new = TRUE)
plot(density(sample2), xlim = c(0.4, 1.1), ylim = c(0, 4),
+ col = "blue", xlab = "x, N = 1000", ylab = "density", main = "")
text.legend = c("my_invcdf","samplepdf")
legend("topright", legend = text.legend, lty = c(1,1), col = c( "red", "blue"))
```

代码执行结果如图 3-6 所示。

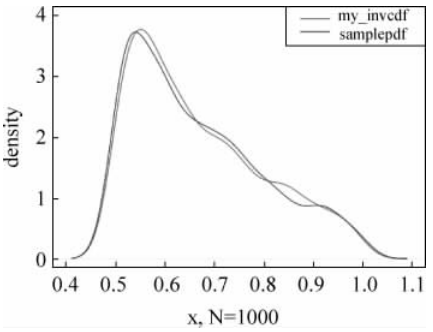

图 3-6　逆采样代码执行结果

3.2.2　博克斯-穆勒变换

博克斯-穆勒变换(Box-Muller Transform)最初由乔治·博克斯(George Box)与默文·穆勒(Mervin Muller)在 1958 年共同提出。博克斯是统计学的一代大师,统计学中的很多名词术语都以其名字命名。博克斯与统计学的家学渊源相当深厚,他的导师是统计学开山鼻祖皮尔逊的儿子,英国统计学家埃贡·皮尔逊(Egon Pearson),博克斯还是统计学的另外一位巨擘级奠基人费希尔的女婿。统计学中的名言“所有模型都是错的,但其中一些是有用的”也出自博克斯之口。

本质上来说,计算机只能生产符合均匀分布的采样。如果要生成其他分布的采样,就需要借助一些技巧性的方法。而在众多的“其他分布”中,正态分布无疑占据着相当重要的地位。下面这个定理,就为生成符合正态分布的采样(随机数)提供了一种方法,而且这也是很多软件或者编程语言的库函数中生成正态分布随机数时所采样的方法。

定理(Box-Muller 变换):如果随机变量 U_1 和 U_2 是独立同分布的,且 $U_1, U_2 \sim U[0,1]$,则

$$Z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

$$Z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2)$$

其中,Z_0 和 Z_1 独立且服从标准正态分布。

如何来证明这个定理呢?这需要用到一些微积分中的知识,首先回忆一下二重积分化

为极坐标下累次积分的方法

$$\iint_D f(x,y)\mathrm{d}x\mathrm{d}y = \int_\alpha^\beta \mathrm{d}\theta \int_{\rho_1(\theta)}^{\rho_2(\theta)} f(\rho\cos\theta,\rho\sin\theta)\rho\mathrm{d}\rho$$

假设现在有两个独立的标准正态分布 $X\sim N(0,1)$ 和 $Y\sim N(0,1)$，由于两者相互独立，则联合概率密度函数为

$$p(x,y) = p(x)\cdot p(y) = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{x^2}{2}}\cdot\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{y^2}{2}} = \frac{1}{2\pi}\mathrm{e}^{-\frac{x^2+y^2}{2}}$$

做极坐标变换，则 $x=R\cos\theta$，$y=R\sin\theta$，则有

$$\frac{1}{2\pi}\mathrm{e}^{-\frac{x^2+y^2}{2}} = \frac{1}{2\pi}\mathrm{e}^{-\frac{R^2}{2}}$$

这个结果可以看成是两个概率分布的密度函数的乘积，其中一个可以看成是$[0,2\pi]$上均匀分布，将其转换为标准均匀分布则有 $\theta\sim U(0,2\pi)=2\pi U_2$。

另外一个的密度函数为

$$P(R) = \mathrm{e}^{-\frac{R^2}{2}}$$

则其累计分布函数 CDF 为

$$P(R\leqslant r) = \int_0^r \mathrm{e}^{-\frac{\rho^2}{2}}\rho\mathrm{d}\rho = -\mathrm{e}^{-\frac{\rho^2}{2}}\Big|_0^r = -\mathrm{e}^{-\frac{r^2}{2}}+1$$

这个 CDF 函数的反函数可以写成

$$F^{-1}(u) = \sqrt{-2\log(1-u)}$$

根据逆变换采样的原理，如果有个 PDF 为 $P(R)$的分布，那么对齐 CDF 的反函数进行均匀采样所得的样本分布将符合 $P(R)$的分布，而如果 u 是均匀分布的，那么 $U_1=1-u$ 也将是均匀分布的，于是用 U_1 替换 $1-u$，最后可得

$$X = R\cdot\cos\theta = \sqrt{-2\log U_1}\cdot\cos(2\pi U_2)$$

$$Y = R\cdot\sin\theta = \sqrt{-2\log U_1}\cdot\sin(2\pi U_2)$$

结论得证。最后来总结一下利用 Box-Muller 变换生成符合高斯分布的随机数的方法：

(1) 产生两个随机数 $U_1, U_2\sim U[0,1]$；

(2) 用它们来创造半径 $R=\sqrt{-2\log(U_1)}$ 和和夹角 $\theta=2\pi U_2$；

(3) 将(R,θ)从极坐标转换到笛卡儿坐标：$(R\cos\theta, R\sin\theta)$。

3.2.3 拒绝采样与自适应拒绝采样

读者已经看到逆变换采样的方法确实有效。但其实它的缺点也是很明显的，那就是有些分布的 CDF 可能很难通过对 PDF 的积分得到，再或者 CDF 的反函数也很不容易求。这时可能需要用到另外一种采样方法，这就是下面即将要介绍的拒绝采样(Reject Sampling)。

图 3-7 阐释了拒绝采样的基本思想。假设对 PDF 为 $p(x)$的函数进行采样，但是由于种种原因(例如这个函数很复杂)，对其进行采样是相对困难的。但是另外一个 PDF 为$q(x)$的函数则相对容易采样，例如采用逆变换方法可以很容易对对它进行采样，甚至 $q(x)$就是一个均匀分布(别忘了计算机可以直接进行采样的分布就只

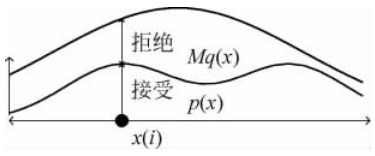

图 3-7 拒绝采样的原理

有均匀分布)。那么,当我们将 $q(x)$与一个常数 M 相乘之后,可以实现如图 3-7 所示关系,即 $M \cdot q(x)$将 $p(x)$完全"罩住"。

然后重复如下步骤,直到获得 m 个被接受的采样点:

(1) 从 $q(x)$中获得一个随机采样点 x_i;

(2) 对于 x_i 计算接受概率(acceptance probability)

$$\alpha = \frac{p(x_i)}{Mq(x_i)}$$

(3) 从 $U(0,1)$中随机生成一个值,用 u 表示;

(4) 如果 $\alpha \geqslant u$,则接受 x_i 作为一个来自 $p(x)$的采样值,否则就拒绝 x_i 并回到第一步。

当然可以采用严密的数学推导来证明拒绝采样的可行性。但它的原理从直观上来解释也是相当容易理解的。可以想象一下在图 2-7 的例子中,从哪些位置抽出的点会比较容易被接受。显然,红色曲线(位于上方)和绿色曲线(位于下方)所示之函数更加接近的地方接受概率较高,即是更容易被接受,所以在这样的地方采到的点就会比较多,而在接受概率较低(即两个函数差距较大)的地方采到的点会比较少,这也就保证了这个方法的有效性。

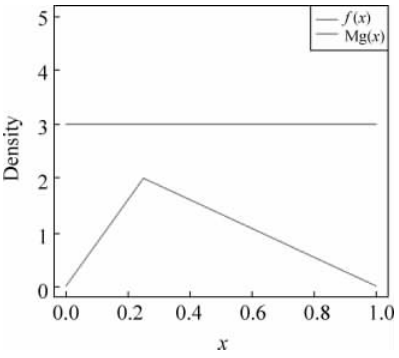

图 3-8　拒绝采样示例

还是以本章前面给出的那个分段函数 $f(x)$为例来演示拒绝采样方法。如图 3-8 所示,所选择的参考分布是均匀分布(当然也可以选择其他的分布,但采用均匀分布显然是此处最简单的一种处理方式)。而且令常数 $M=3$。

下面给出 R 中的示例代码,易见此处采样点数目为 10 000。

```
f.x <- function(x) {
+ if (x >= 0 && x < 0.25)
+ 8 * x
+ else if (x >= 0.25 && x <= 1)
+ 8/3 - 8 * x/3
+ else 0
+ }

g.x <- function(x) {
+ if (x >= 0 && x <= 1)
+ 1
+ else 0
+ }

M <- 3
m <- 10000
n.draws <- 0
draws <- c()
x.grid <- seq(0, 1, by = 0.01)
```

```
while (n.draws < m) {
 + x.c <- runif(1, 0, 1)
 + accept.prob <- f.x(x.c)/(M * g.x(x.c))
 + u <- runif(1, 0, 1)
 + if (accept.prob >= u) {
 +     draws <- c(draws, x.c)
 +     n.draws <- n.draws + 1
 +     }
 + }
```

上述代码的执行结果如图 3-9 所示，可见采样结果是非常理想的。

下面的例子演示了对(表达式非常复杂的)$beta(3,6)$分布进行拒绝采样的效果。这里采用均匀分布作为参考分布。而且这里的 $Mq(x)$所取之值就是 $beta(3,6)$分布的极大值，它的函数图形应该是与 $beta(3,6)$的极值点相切的一条水平直线。

```
sampled <- data.frame(proposal = runif(50000,0,1))
sampled$targetDensity <- dbeta(sampled$proposal, 3,6)

maxDens = max(sampled$targetDensity, na.rm = T)
sampled$accepted = ifelse(runif(50000,0,1)
 + < sampled$targetDensity / maxDens, TRUE, FALSE)

hist(sampled$proposal[sampled$accepted], freq = F,
 + col = "grey", breaks = 100, main = "")
curve(dbeta(x, 3,6),0,1, add = T, col = "red", main = "")
```

图 3-10 给出了采样 50 000 个点后的密度分布情况，可见采样分布与目标分布 $beta(3,6)$非常吻合。

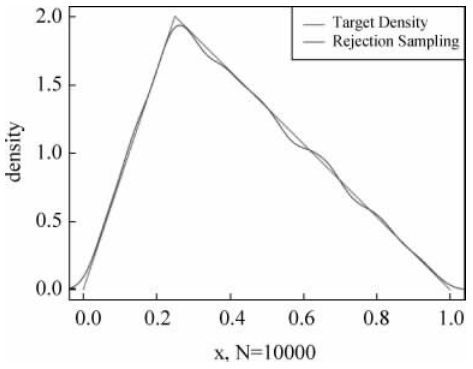

图 3-9　程序执行结果

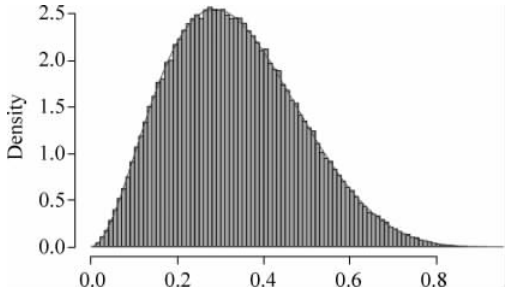

图 3-10　拒绝采样举例

拒绝采样的方法确实可以解决我们的问题。但是它的一个不足涉及其采样效率的问题。针对上面给出的例子而言，我们选择了离目标函数最近的参考函数，就均匀分布而言，已经不能有更进一步的方法了。但即使这种，在这个类似钟形的图形两侧其实仍然会拒绝掉很多很多采样点，这种开销相当浪费。最理想的情况下，参考分布应该跟目标分布越接近

越好，从图形上来看就是包裹的越紧实越好。但是这种情况的参考分布往往又不那么容易得到。在满足某些条件的时候也确实可以采用所谓的改进方法，即自适应的拒绝采样(Adaptive Rejection Sampling)。

拒绝采样的弱点在于当被拒绝的点很多时，采样的效率会非常不理想。同时我们也知道，如果能够找到一个跟目标分布函数非常接近的参考函数，那么就可以保证被接受的点占大多数(被拒绝的点很少)。这样一来便克服了拒绝采样效率不高的弱点。如果函数是 log-concave 的话，那么就可以采用自适应的拒绝采样方法。什么是 log-concave 呢？还是回到之前介绍过的贝塔分布，用下面的代码来绘制 beta(2,3)的概率密度函数图像，以及将 beta(2,3)的函数取对数之后的图形。

```
> integrand <- function(x) {(x^1) * ((1 - x)^2)}
> integrate(integrand, lower = 0, upper = 1)
0.08333333 with absolute error < 9.3e-16
> f <- function(x,a,b){log(1/0.08333) + (a - 1) * log(x) + (b - 1) * log(1 - x)}
> curve(f(x, 2, 3))
> curve(dbeta(x, 2, 3))
```

上述代码的执行结果如图 3-11 所示，其中(a)是 beta(2,3)的概率密度函数图形，(b)是将 beta(2,3)的函数取对数之后的图形，你可以发现结果是一个凹函数(concave)。那么 beta(2,3)就满足 log-concave 的要求。

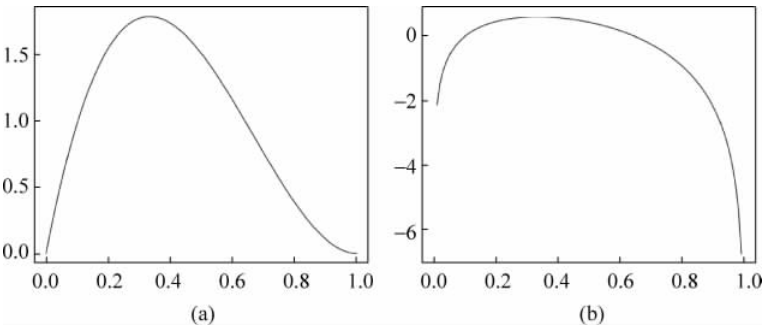

图 3-11　贝塔函数与其取对数后的函数图形

然后在对数图像上找一些点做图像的切线，如图 3-12 所示。因为取对数后的函数是凹函数，所以每个切线都相当于一个超平面，而且对数图像只会位于超平面的一侧。

同时给出用以绘制图 3-12 的代码，要知道 R 语言的一个强项就是绘图。

```
log_f <- function(x,a,b){log(1/0.08333) + (a - 1) * log(x) + (b - 1) * log(1 - x)}
g <- function(x,a,b){(a - 1)/x - (b - 1)/(1 - x)}

log_f_y1 <- log_f(0.18, 2, 3)
log_f_y2 <- log_f(0.40, 2, 3)
log_f_y3 <- log_f(0.65, 2, 3)
log_f_y4 <- log_f(0.95, 2, 3)

g1 <- g(0.18, 2, 3)
```

```
b1 <- log_f_y1 - g1 * 0.18
y1 <- function(x) {g1 * x + b1}

g2 <- g(0.40, 2, 3)
b2 <- log_f_y2 - g2 * 0.40
y2 <- function(x) {g2 * x + b2}

g3 <- g(0.65, 2, 3)
b3 <- log_f_y3 - g3 * 0.65
y3 <- function(x) {g3 * x + b3}

g4 <- g(0.95, 2, 3)
b4 <- log_f_y4 - g4 * 0.95
y4 <- function(x) {g4 * x + b4}

curve(log_f(x, 2, 3), col = "blue", xlim = c(0,1), ylim = c( - 7, 1))
curve(y1, add = T, lty = 2, col = "red", to = 0.38)
curve(y2, add = T, lty = 2, col = "red", from = 0.15, to = 0.78)
curve(y3, add = T, lty = 2, col = "red", from = 0.42)
curve(y4, add = T, lty = 2, col = "red", from = 0.86)

par(new = TRUE)

xs = c(0.18, 0.40, 0.65, 0.95)
ys = c(log_f_y1, log_f_y2, log_f_y3, log_f_y4)
plot(xs, ys, col = "green", xlim = c(0,1), ylim = c( - 7, 1), xlab = "", ylab = "")
```

再把这些切线转换回原始的 beta(2,3)图像中，显然原来的线性函数会变成指数函数，它们将对应图 3-13 中的一些曲线，这些曲线会被原函数的图形紧紧包裹住。特别是当这些的指数函数变得很多很稠密时，以彼此的交点作为分界线，其实相当于得到了一个分段函数。这个分段函数是原函数的一个逼近。用这个分段函数来作为参考函数再执行拒绝采样，自然就完美地解决了之前的问题。

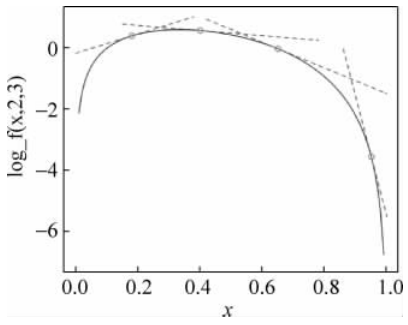

图 3-12　做取对数后的图形的切线

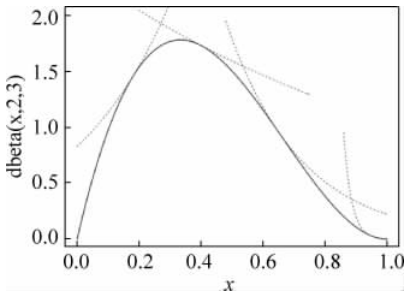

图 3-13　切线取指数函数后的变换结果

下面是用来画出图 3-13 的 R 语言代码。

```
e_y1 <- function(x) {exp(g1 * x + b1)}
e_y2 <- function(x) {exp(g2 * x + b2)}
e_y3 <- function(x) {exp(g3 * x + b3)}
e_y4 <- function(x) {exp(g4 * x + b4)}

curve(dbeta(x, 2, 3), col = "blue", ylim = c(0, 2.0))
curve(e_y1(x), add = T, lty = 3, col = "red")
curve(e_y2(x), add = T, lty = 3, col = "red", from = 0.2, to = 0.75)
curve(e_y3(x), add = T, lty = 3, col = "red", from = 0.48)
curve(e_y4(x), add = T, lty = 3, col = "red", from = 0.86)
```

这无疑是一种绝妙的想法。而且这种想法，在前面其实已经暗示过。在上一部分最后一个例子中，我们其实就是选择了一个与原函数相切的均匀分布函数来作为参考函数。我们当然会想去选择更多与原函数相切的函数，然后用这个函数的集合来作为新的参考函数。只是由于原函数的凹凸性无法保证，所以直线并不是一种好的选择。而自适应拒绝采样(Adaptive Rejection Sampling，ARS)所采用的策略则非常巧妙地解决了我们的问题。当然函数是 log-concave 的条件必须满足，否则就不能使用 ARS。

下面给出一个在 R 中进行自适应拒绝采样的例子。显然，该例子要比之前的代码简单许多。因为 R 中 ars 包已经提供了一个现成的用于执行自适应拒绝采样的函数，即 ars()。关于这个函数在用法上的一些细节，读者还可以进一步参阅 R 的帮助文档，这里不再赘言。此次我们需要指出：ars()函数中两个重要参数，一个是对原分布的 PDF 取对数，另外一个则是对 PDF 的对数形式再进行求导(在求导时我们忽略了前面的系数项)，其实也就是为了确定切线。

```
f <- function(x,a,b){(a - 1) * log(x) + (b - 1) * log(1 - x)}
fprima <- function(x,a,b){(a - 1)/x - (b - 1)/(1 - x)}
mysample <- ars(20000, f, fprima, x = c(0.3,0.6), m = 2, lb = TRUE, xlb = 0,
 + ub = TRUE, xub = 1, a = 1.3, b = 2.7)
hist(mysample, freq = F)
curve(dbeta(x,1.3,2.7), add = T, col = "red")
```

上述代码的执行结果如图 3-14 所示。

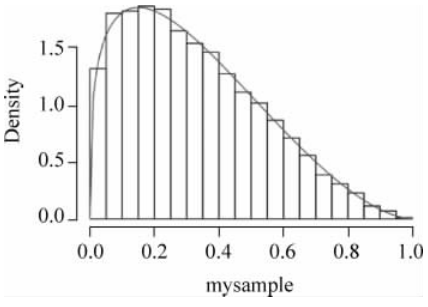

图 3-14　自适应采样代码执行结果

3.3 矩阵的极限与马尔科夫链

先来看一个例子。社会学家经常把人按其经济状况分成3类：下层(lower-class)、中层(middle-class)、上层(upper-class)，用1、2、3分别代表这三个阶层。社会学家们发现决定一个人的收入阶层的最重要的因素就是其父母的收入阶层。如果一个人的收入属于下层类别，那么他的孩子属于下层收入的概率是0.65，属于中层收入的概率是0.28，属于上层收入的概率是0.07。从父代到子代，收入阶层的变化的转移概率如图3-15所示。

		子代		
	State	1	2	3
父代	1	0.65	0.28	0.07
	2	0.15	0.67	0.18
	3	0.12	0.36	0.52

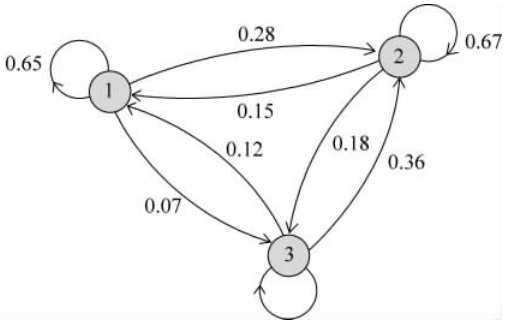

图3-15 阶层变化的转移概率

使用矩阵的表示方式，转移概率矩阵记为

$$\boldsymbol{P}=\begin{bmatrix}0.65 & 0.28 & 0.07\\ 0.15 & 0.67 & 0.18\\ 0.12 & 0.36 & 0.52\end{bmatrix}$$

假设当前这一代人处在下、中、上层的人的比例是概率分布向量$\boldsymbol{\pi}_0=[\pi_0(1),\pi_0(2),\pi_0(3)]$，那么他们的子女的分布比例将是$\boldsymbol{\pi}_1=\boldsymbol{\pi}_0P$，孙子代的分布比例将是$\boldsymbol{\pi}_2=\boldsymbol{\pi}_1P=\boldsymbol{\pi}_1P^2=\cdots$，第$n$代子孙的收入分布比例将是$\boldsymbol{\pi}_n=\boldsymbol{\pi}_{n-1}P=\cdots=\boldsymbol{\pi}_0P^n$。假设初始概率分布为$\boldsymbol{\pi}_0=[0.21,0.68,0.11]$，则可以计算前$n$代人的分布状况如下：

第n代人	下层	中层	上层
0	0.210	0.680	0.110
1	0.252	0.554	0.194
2	0.270	0.512	0.218
3	0.278	0.497	0.225
4	0.282	0.490	0.226
5	0.285	0.489	0.225
6	0.286	0.489	0.225
7	0.286	0.489	0.225
8	0.289	0.488	0.225
9	0.286	0.489	0.225
10	0.286	0.489	0.225
…	…	…	…

发现从第7代人开始，这个分布就稳定不变了，这个是偶然的吗？我们换一个初始概率

分布$\pi_0=[0.75,0.15,0.1]$试试看，继续计算前 n 代人的分布状况如下

第 n 代人	下层	中层	上层
0	0.75	0.15	0.1
1	0.522	0.347	0.132
2	0.407	0.426	0.167
3	0.349	0.459	0.192
4	0.318	0.475	0.207
5	0.303	0.482	0.215
6	0.295	0.485	0.220
7	0.291	0.487	0.222
8	0.289	0.488	0.225
9	0.286	0.489	0.225
10	0.286	0.489	0.225
…	…	…	…

结果，到第 9 代人的时候，分布又收敛了。最为奇特的是，两次给定不同的初始概率分布，最终都收敛到概率分布$\pi=[0.286,0.489,0.225]$，也就是说收敛的行为和初始概率分布π_0无关。这说明这个收敛行为主要是由概率转移矩阵 P 决定的。计算一下 P^n，

$$P^{20}=P^{21}=\cdots=P^{100}=\cdots=\begin{bmatrix}0.286 & 0.489 & 0.225\\ 0.286 & 0.489 & 0.225\\ 0.286 & 0.489 & 0.225\end{bmatrix}$$

我们发现，当 n 足够大的时候，这个 P^n 矩阵的每一行都是稳定地收敛到$\pi=[0.286,0.489,0.225]$这个概率分布。自然地，这个收敛现象并非是这个例子所独有，而是绝大多数“马尔科夫链”的共同行为。

为了求得一个理论上的结果，再来看一个更小规模的例子(这将方便后续的计算演示)，假设在一个区域内，人们要么是住在城市，要么是住在乡村。下面的矩阵表示人口迁移的一些规律(或倾向)。例如，第 1 行第 1 列就表示，当前住在城市的人口，明年将会有 90%的人选择继续住在城市。

$$\begin{array}{c} \\ \text{下一年住在城市} \\ \text{下一年住在乡村}\end{array}\begin{array}{c}\begin{array}{cc}\text{当前住在}\\ \text{城市}\end{array}\quad\begin{array}{c}\text{当前住在}\\ \text{乡村}\end{array}\\ \begin{pmatrix}0.90 & 0.02\\ 0.10 & 0.98\end{pmatrix}\end{array}=\boldsymbol{A}$$

作为一个简单的开始，来计算一下现今住在城市的人中 2 年后会住在乡村的概率有多大。分析可知，当前住在城市的人，1 年后，会有 90%继续选择住在城市，另外 10%的人则会选择搬去乡村居住。然后又过了 1 年，上一年中选择留在城市的人中又会有 10%的人搬去乡村。而上一年中搬到乡村的人中将会有 98%的选择留在乡村。这个分析过程如图 3-16 所示，最终可以算出现今住在城市的人中 2 年后会住在乡村的概率$=0.90\times0.10+0.10\times0.98$。

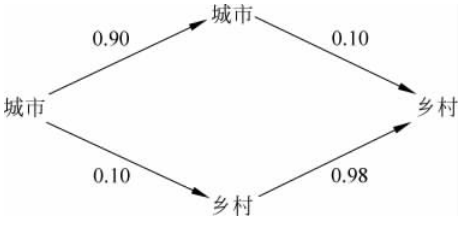

图 3-16　两年后人口的转移

其实上述计算过程就是在做矩阵的平方。在下面给出的矩阵乘法中，你会发现结果矩阵中第 2 行第 1 列的计算就是在执行上面的操作。在此基础，我们还可以继续计算 n 年后的情况，也就是计算矩阵 $\boldsymbol{A}$ 自乘 n 次后的结果。

$$\boldsymbol{A}^2 = \begin{bmatrix} 0.90 & 0.02 \\ 0.10 & 0.98 \end{bmatrix}\begin{bmatrix} 0.90 & 0.02 \\ 0.10 & 0.98 \end{bmatrix} = \begin{bmatrix} (\boldsymbol{A}^2)_{11} & (\boldsymbol{A}^2)_{12} \\ (\boldsymbol{A}^2)_{21} & (\boldsymbol{A}^2)_{22} \end{bmatrix}$$

如果假设最开始的时候，城乡人口的比例为 7∶3，可以用一个列向量来表示它，即 $\boldsymbol{P}=[0.7,0.3]^{\mathrm{T}}$，想知道最终城乡人口的比例为何，则就是要计算。如果最初城乡人口的比例为 9∶1，结果又如何呢？这些都要借助矩阵的极限，对角化操作以及马尔科夫链等概念来辅助计算。

来辨析三个概念：随机过程、马尔科夫过程、马尔科夫链。这三个概念中，都涉及对象和它们对应的状态这两个要素。在刚刚给出的例子中，研究的对象就是人，人的状态分为住在城市或者住在乡村两种。

三者之中，最宽泛的概念就是随机过程，限制最多的就是马尔科夫链。对于马尔科夫链，必须满足两个条件：①当前状态仅跟上一个状态有关；②总共的状态数是有限的。如果状态数可以是无限多个，那样的问题就称为马尔科夫过程。但在马尔科夫过程中仍然要求，时间是离散的，例如前面的例子是以“年”为单位的。如果时间允许是连续的，那样的问题就称为随机过程。本书仅仅讨论马尔科夫链。

在某个时间点上，对象的状态为 s_1，下一个时刻，它的状态以某种概率转换到其他状态（也包含原状态 s_1），这里所说的“以某种概率转换”最终是通过状态转移矩阵（或称随机矩阵）的形式来给出的。转移矩阵的定义如下：

令 $\boldsymbol{A}\in\boldsymbol{M}_{n\times n}(\mathbb{R})$，假设：

（1）$\boldsymbol{A}_{ij}\geqslant 0$；

（2）对于所有的 $1\leqslant j\leqslant n$，$\sum_{i=1}^{n}\boldsymbol{A}_{ij}=1$。

那么，$\boldsymbol{A}$ 称为一个转移矩阵（或随机矩阵）。矩阵 $\boldsymbol{A}$ 的列向量被称为概率向量。

此外，如果矩阵的某个次幂仅包含正数项，该转移矩阵称为是正则的。这里，“某个”的意思就是存在一个整数 n 使得对于所有的 i,j，有 $(\boldsymbol{A}^n)_{ij}>0$。

从状态转移矩阵中，（结合之前的例子）可以看出 $\boldsymbol{A}_{ij}$ 元素给出的信息就是（在一个单位时间间隔内）对象从状态 j 转移到状态 i 的概率。令 $\boldsymbol{P}=(p_0,p_1,\cdots,p_n)^{\mathrm{T}}$ 是一个向量，如果对于所有的 i，有 $p_i\geqslant 0$ 以及 $\sum p_i=1$，那么 $\boldsymbol{P}$ 就称为一个概率向量（probability vector）。所以可以看出，任意一个转移矩阵中的某一列都是一个概率向量。

定理：令 $\boldsymbol{A}$ 是一个 $n\times n$ 的正则转移矩阵。那么：

（1）1 一定是矩阵 $\boldsymbol{A}$ 的一个特征值，并且 1 的几何重数等于 1，除此之外，所有其他的特征值之绝对值都小于 1；

（2）$\boldsymbol{A}$ 可以对角化，并且 $\lim_{m\to\infty}\boldsymbol{A}^m$ 存在；

（3）$\boldsymbol{L}=\lim_{m\to\infty}\boldsymbol{A}^m$ 是一个转移矩阵；

（4）$\boldsymbol{AL}=\boldsymbol{LA}=\boldsymbol{L}$；

（5）矩阵 $\boldsymbol{L}=[\boldsymbol{v},\boldsymbol{v},\cdots,\boldsymbol{v}]$，即 $\boldsymbol{L}$ 的每一列都一样，都是 $\boldsymbol{v}$。而且 $\boldsymbol{v}$ 就是矩阵 $\boldsymbol{A}$ 相对于

特征值 1 的特征向量；

(6) 对于任意概率向量 $\boldsymbol{w}$，都有 $\lim_{m\to\infty}(\boldsymbol{A}^m\boldsymbol{w})=\boldsymbol{v}$。

这个定理非常非常奇妙的地方就是它解答了之前那个令人困扰的问题！原问题是：如果假设最开始的时候，城乡人口的比例为 7∶3，可以用一个列向量来表示它 $\boldsymbol{P}=[0.7,0.3]^{\mathrm{T}}$，欲知道最终城乡人口的比例为何，则就是要计算 $\boldsymbol{A}^nP$，如果最开始的城乡人口的比例为 9∶1，结果又如何。上述定理中的最后一条就表明，当 n 趋近于无穷大的时候，$\boldsymbol{A}^nP$ 就等于 $\boldsymbol{v}$，而且与 P 是无关的。更精妙的地方还在于，这个定理还告诉我们 v 是一个概率向量，而且它就是特征值 1 所对应的特征向量。

这个定理的证明已经超出了本书的范围，但可以用之前给出的例子来验证一下它。注意到如果想要计算 $\boldsymbol{A}^nP$，其实就是要先设法计算矩阵 $\boldsymbol{A}$ 自乘 n 次的结果，这时为了计算方便应该先将矩阵 $\boldsymbol{A}$ 对角化。为此，先求矩阵 $\boldsymbol{A}$ 的特征多项式，通过其特征多项式便知道矩阵 $\boldsymbol{A}$ 有两个特征值，一个是 1，一个是 0.88。根据定理，1 必然是该矩阵的特征值。更进一步，特征值 1 对应的特征向量是 $[1,5]^{\mathrm{T}}$，特征值 0.88 对应的特征向量是 $[1,-1]^{\mathrm{T}}$。所以知道矩阵对角化时所用的 $\boldsymbol{Q}$ 和 $\boldsymbol{Q}^{-1}$ 分别为

$$\boldsymbol{Q}=\begin{bmatrix}1 & 1\\ 5 & -1\end{bmatrix},\quad \boldsymbol{Q}^{-1}=-\frac{1}{6}\begin{bmatrix}-1 & -1\\ -5 & 1\end{bmatrix}$$

于是可知矩阵 $\boldsymbol{A}$ 的对角化结果如下：

$$\begin{bmatrix}1 & 0\\ 0 & 0.88\end{bmatrix}=\boldsymbol{D}=\boldsymbol{Q}^{-1}\boldsymbol{A}\boldsymbol{Q}$$

所以有

$$\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^{-1}\Rightarrow\boldsymbol{A}^m=\boldsymbol{Q}\boldsymbol{D}^m\boldsymbol{Q}^{-1}$$

$$\lim_{m\to\infty}\boldsymbol{A}^m=\lim_{m\to\infty}\boldsymbol{Q}\boldsymbol{D}^m\boldsymbol{Q}^{-1}=\boldsymbol{Q}\left(\lim_{m\to\infty}\boldsymbol{D}^m\right)\boldsymbol{Q}^{-1}$$

然后把值带进去就能算出最终结果如下

$$\lim_{m\to\infty}\boldsymbol{A}^m=\begin{bmatrix}1/6 & 1/6\\ 5/6 & 5/6\end{bmatrix}$$

之前计算过特征值 1 对应的一个特征向量是 $[1,5]^{\mathrm{T}}$，特征向量乘以一个系数仍然是特征向量(注意要求最后的特征向量同时是一个概率向量，所以会得到 $[1/6,5/6]^{\mathrm{T}}$，可见上述计算与定理所揭示的结果是完全一致的。

矩阵的极限，其实就是在讨论 $\lim_{m\to\infty}\boldsymbol{A}^m$ 的存在性，也就是把矩阵 $\lim_{m\to\infty}\boldsymbol{A}^m$ 自乘 m 次后，如果结果矩阵中每个元素的极限都存在，就说这个矩阵的极限是存在的。而矩阵极限是否存在可以由下面的定理保证。

定理　矩阵极限存在的充要条件：

(1) $|\lambda|<1$ 或者 $\lambda=1$，其中 λ 是 $\boldsymbol{A}$ 的任意特征值。

(2) 如果 $\lambda=1$，那么它对应的几何重数等于代数重数。

定理　矩阵极限存在的充分条件：

(1) $|\lambda|<1$ 或者 $\lambda=1$，其中 λ 是 $\boldsymbol{A}$ 的任意特征值。

(2) 矩阵 $\boldsymbol{A}$ 是可对角化的。

3.4 查普曼-柯尔莫哥洛夫等式

还是先从一个例子来谈起。图 3-15 中左侧是状态转移矩阵，其中每个位置都表示从一个状态转移到另外一个状态的概率。例如，从状态 1 转移到状态 2 的概率是 0.28，所以在第一行第二列的位置，给出的数值是 0.28。

在马尔科夫链中，随机变量在一个按时间排序的数组 $T_1, T_2, \cdots, T_n$ 中，根据状态转移矩阵让我们可以非常直观地得知当前时刻某一状态在下一时刻变到任意状态的概率，如图 3-17 所示。

现在的问题是能不能做更进一步的预测。例如，当前时刻 T_1 时的状态为 a，能否知道在下下时刻 T_3 时状态为 b 的概率是多少呢？这其实也相当容易做到，如图 3-18 所示。

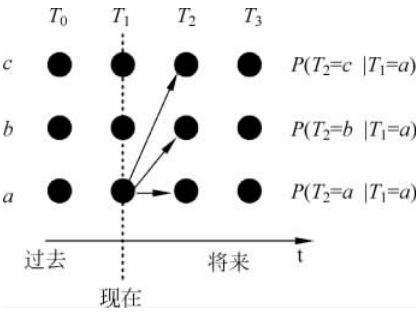

图 3-17 按时间排序的状态转移情况

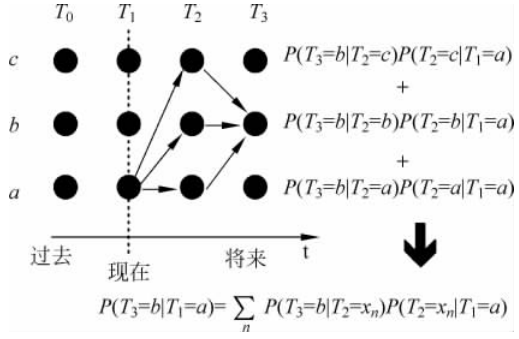

图 3-18 计算下下时刻某状态的概率

这个过程用转移矩阵的自乘来表示是非常直观且方便的。矩阵 $\boldsymbol{P}$ 中第 1 行就表示，当前时刻状态 a 在下一时刻变到状态 a、b、c 的概率，矩阵 $\boldsymbol{P}$ 中第 2 列就表示，由状态 a、b、c，在下一时刻转移到状态 b 的概率。那么根据矩阵的乘法公式，$\boldsymbol{P}^2$ 中第 1 行第 2 列的位置就表示“当前时刻 T_1 时的状态为 a，在下下时刻 T_3 时状态为 b 的概率”。也就是说，如果想得到跨越 2 个时刻的转移矩阵，就把跨越 1 个时刻的转移矩阵乘以跨越 1 个时刻的转移矩阵即可。同理，如果想跨越 3 个时刻的转移矩阵，就把跨越 2 个时刻的转移矩阵乘以跨越 1 个时刻的转移矩阵即可。更普遍地有 $\boldsymbol{P}^{m+n}=\boldsymbol{P}^m\boldsymbol{P}^n$，而这个关系就被称为查普曼-柯尔莫哥洛夫等式(Chapman-Kolmogorov Equation)。

下面是查普曼-柯尔莫哥洛夫等式的一个表述：令 T 是一个离散状态空间中的 n 步马尔科夫链，其状态转移矩阵为

$$\boldsymbol{P}^n = [p^n(j,k)]_{j,k\in S}$$

其中，$p^n(j,k)=P(T_{m+n}=k \mid T_m=j)=p^n_{j,k}$ 是 n 步状态转移概率。那么，$\boldsymbol{P}^{m+n}=\boldsymbol{P}^m\boldsymbol{P}^n$，或等价地有

$$p^{n+m}_{i,j} = \sum_{k\in S} p^n_{i,k} p^m_{k,j}$$

最后给出查普曼-柯尔莫哥洛夫等式的证明。

首先考虑在 n 时刻，系统正处在什么状态。给定 $T_0=i$，$p^n_{i,k}=P(T_n=k \mid T_0=i)$ 就表示(在时刻 n)系统处于 k 状态的概率。但是，此后再给定 $T_n=k$，在第 n 时刻之后的情况就与

过去的历史彼此独立了。于是，再过 m 个时间单位后，处在状态 j 的概率是 $p_{k,j}^{m}$ 将满足乘积

$$p_{i,k}^{n}p_{k,j}^{m}=P(T_{n+m}=j,T_n=k\mid T_0=i)$$

将所有 k 的情况进行加总就会得到要证明之结论。一个更加严格的证明如下

$$\begin{aligned}p_{i,j}^{n+m}&=P(T_{n+m}=j\mid T_0=i)\\&=\sum_{k\in S}P(T_{n+m}=j,T_n=k\mid T_0=i)\\&=\sum_{k\in S}\frac{P(T_{n+m}=j,T_n=k,T_0=i)}{P(T_0=i)}\\&=\sum_{k\in S}\frac{P(T_{n+m}=j\mid T_n=k,T_0=i)P(T_n=k,T_0=i)}{P(T_0=i)}\\&=\sum_{k\in S}\frac{p_{k,j}^{m}P(T_n=k,T_0=i)}{P(T_0=i)}\\&=\sum_{k\in S}p_{i,k}^{n}p_{k,j}^{m}\end{aligned}$$

上述证明中运用了马尔科夫链的性质来得到

$$P(T_{n+m}=j\mid T_n=k,T_0=i)=P(T_{n+m}=j\mid T_n=k)=P(T_m=j\mid T_0=k)=p_{k,j}^{m}$$

最终，结论得证。

3.5　马尔科夫链蒙特卡洛

在以贝叶斯方法为基础的各种机器学习技术中，常常需要计算后验概率，再通过最大后验概率(MAP)的方法进行参数推断和决策。然而，在很多时候，后验分布的形式可能非常复杂，这个时候寻找其中的最大后验估计或者对后验概率进行积分等计算往往非常困难，此时可以通过采样的方法来求解。这其中非常重要的一个基础就是马尔科夫链蒙特卡洛(Markov chain Monte Carlo，MCMC)。

3.5.1　重要性采样

前面已经详细介绍了基于随机算法进行定积分求解的技术。这里主要用到其中的平均值法。这里仅做简单回顾。

在计算定积分 $\int_a^b g(x)\mathrm{d}x$ 时，会把 $g(x)$ 拆解成两项的乘积，即 $g(x)=f(x)p(x)$，其 $f(x)$ 是某种形式的函数，而 $p(x)$ 是关于随机变量 X 的概率分布(也就是 PDF 或 PMF)。如此一来，上述定积分就可以表示为求 $f(x)$ 的期望，即

$$\int_a^b g(x)\mathrm{d}x=\int_a^b f(x)p(x)\mathrm{d}x=E[f(x)]$$

当然，$g(x)$的分布函数可能具有很复杂的形式，仍然无法直接求解，这时就可以用采样的方法去近似。这时积分可以表示为

$$\int_a^b g(x)\mathrm{d}x=E[f(x)]=\frac{1}{n}\sum_{i=1}^{n}f(x_i)$$

在贝叶斯分析中，蒙特卡洛积分运算常常被用来在对后验概率进行积分时做近似估计。比如要计算

$$I(y)=\int_a^b f(y\mid x)p(x)\mathrm{d}x$$

便可以使用下面这个近似形式

$$\hat{I}(y)=\frac{1}{n}\sum_{i=1}^{n}f(y\mid x_i)$$

不难发现，在利用蒙特卡洛法进行积分求解时，非常重要的一个环节就是从特定的分布中进行采样。这里的“采样”意思也就是生成满足某种分布的观测值。之前已经介绍过“逆采样”和“拒绝采样”等方法。

采样的目的很多时候都是为了做近似积分运算，前面的采样方法（逆采样和拒绝采样）都是先对分布进行采样，然后再用采样的结果近似计算积分。下面要介绍的另外一种方法“重要性采样”（Importance sampling）则两步并做一步，直接做近似计算积分。

现在的目标是计算下面的积分

$$E[f(x)]=\int f(x)p(x)\mathrm{d}x$$

按照蒙特卡洛求定积分的方法，将从满足 $p(x)$ 的概率分布中独立地采样出一系列随机变量 x_i，然后便有

$$E[f(x)]\approx\frac{1}{n}\sum_{i=1}^{n}f(x_i)$$

但是现在的困难是对满足 $p(x)$ 的概率分布进行采样非常困难，毕竟实际中很多 $p(x)$ 的形式都相当复杂。这时该怎么做呢？于是想到做等量变换，于是有

$$\int f(x)p(x)\mathrm{d}x=\int f(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x$$

如果把其中的 $f(x)\frac{p(x)}{q(x)}$ 看成是一个新的函数 $h(x)$，则有

$$\int f(x)p(x)\mathrm{d}x=\int h(x)q(x)\mathrm{d}x\approx\frac{1}{n}\sum_{i=1}^{n}f(x_i)\frac{p(x_i)}{q(x_i)}$$

其中 $\frac{p(x_i)}{q(x_i)}$ 被称为是 x_i 的权重或重要性权重（Importance Weights）。所以这种采样的方法就被称为是重要性采样（Importance Sampling）。

如图 3-19 所示，在使用重要性采样时，并不会拒绝掉某些采样点，这与在使用拒绝采样时不同。此时，所有的采样点都将为我们所用，但是它们的权重是不同的。因为权重为 $\frac{p(x_i)}{q(x_i)}$，所以在图 3-19 中，当 $p(x_i)>q(x_i)$ 时，采样点 x_i 的权重就大（深色线在浅色线上方时），反之就小。重要性采样就是通过这种方式来从一个“参考分布”$q(x)$ 中获得“目标分布”$p(x)$ 的。

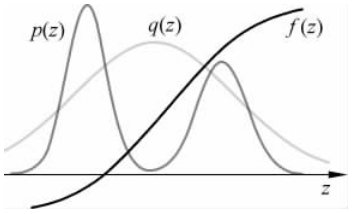

图 3-19　重要性采样原理

3.5.2　马尔科夫链蒙特卡洛的基本概念

马尔科夫链蒙特卡洛方法是一类用于从一个概率分布中进行采样的算法，该类方法以构造马尔科夫链为基础，而被构造的马尔科夫链分布应该同需要之分布等价。

MCMC 构造马尔科夫链，使其稳态分布等于要采样的分布，这样就可以通过马尔科夫链来采样。这种等价如何理解是深入探讨具体操作方法之前，需要先攻克的一个问题。在此之前，希望读者对马尔科夫链已经有了一个比较清晰的认识。

现在，用下面的式子来表示每一步（时刻推进）中从状态 s_i 到状态 s_j 的转移概率

$$p(i,j) = p(i \to j) = P(X_{t+1} = s_j \mid X_t = s_i)$$

这里的一步是指时间从时刻 t 过渡到下一时刻 $t+1$。

马尔科夫链在时刻 t 处于状态 j 的概率可以表示为 $\pi_j(t)=P(X_t=s_j)$。这里用向量 $\pi(t)$来表示在时刻 t 各个状态的概率向量。在初始时刻，需要选取一个特定的 $\pi(0)$，通常情况下可以使向量中一个元素为 1，其他元素均为 0，即从某个特定的状态开始。随着时间的推移，状态会通过转移矩阵，向各个状态扩散。

马尔科夫链在 $t+1$ 时刻处于状态 s_i 的概率可以通过 t 时刻的状态概率和转移概率来求得，并可通过查普曼-柯尔莫哥洛夫等式得到

$$\pi_i(t+1)= P(X_{t+1} = s_i) = \sum_k P(X_{t+1} = s_i \mid X_t = s_k)P(X_t = s_k)$$

$$= \sum_k p(k \to i)\pi_k(t) = \sum_k p(i \mid k)\pi_k(t)$$

用转移矩阵写成矩阵的形式如下

$$\pi(t+1) = \pi(t)\boldsymbol{P}$$

其中，转移矩中的元素 $p(i,j)$表示 $p(i\to j)$。因此，$\pi(t)=\pi(0)\boldsymbol{P}^t$。此外，$p_{i,j}^n$表示矩阵 $\boldsymbol{P}^n$ 中第 ij 个元素，即 $p_{i,j}^n=P(X_{t+n}=s_j \mid X_t=s_i)$。

一条马尔科夫链有一个平稳分布 π^*，是指给定任意一个初始分布 $\pi(0)$，经过有限步之后，最终都会收敛到平稳分布 π^*。平稳分布具有性质 $\pi^*=\pi^*\boldsymbol{P}$。可以结合本章前面谈到的矩阵极限的概念来理解马尔科夫链的平稳分布。

一条马尔科夫链拥有平稳分布的一个充分条件是对于任意两个状态 i 和 j，其满足细致平衡（Detailed Balance）：$p(j\to i)\pi_j=p(i\to j)\pi_i$。可以看到，此时

$$(\pi\boldsymbol{P})_j = \sum_i \pi_i p(i \to j) = \sum_i \pi_j p(j \to i) = \pi_j \sum_i p(j \to i) = \pi_j$$

所以 $\pi=\pi\boldsymbol{P}$，明显满足平稳分布的条件。如果一条马尔科夫链满足细致平衡，就说它是可逆的。

最后来总结一下 MCMC 的基本思想。在拒绝采样和重要性采样中，当前生成的样本点与之前生成的样本点之间是没有关系的，它的采样都是独立进行的。然而，MCMC 是基于马尔科夫链进行的采样。这就表明，当前的样本点生成与上一时刻的样本点是有关的。如图 3-20 所示，假设当前时刻生成的样本点是 x，下一次时刻采样到 x'的（转移）概率就是 $p(x \mid x')$，或者写成 $p(x\to x')$。我们希望的是这个过程如此继续下去，最终的概率分布收敛到 $\pi(x)$，也就是要采样的分布。

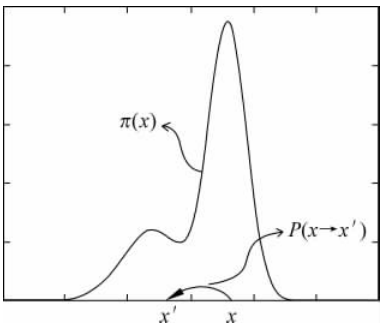

图 3-20　MCMC 的基本原理

易见，转移概率 $p(x \mid x')$与采样分布 $\pi(x)$之间必然存在某种联系，因为希望依照 $p(x \mid x')$来进行转移采样最终能收敛到 $\pi(x)$。那这个关系

到底是怎么样的呢？首先，给定所有可能的 x，然后求从 x 转移到 x' 的概率，所得之结果就是 x' 出现的概率，如果用公式表示即

$$\pi(x') = \int \pi(x) p(x' \mid x) \mathrm{d}x$$

这其实也就是查普曼-柯尔莫哥洛夫等式所揭示的关系。如果马尔科夫链是平稳的，那么基于该马尔科夫链的采样过程最终将收敛到平稳状态

$$\pi^*(x') = \int \pi^*(x') p(x \mid x') \mathrm{d}x'$$

而平稳状态的充分条件又是满足细致平衡

$$\pi(x) p(x' \mid x) \mathrm{d}x = \pi(x) p(x \mid x')$$

可见细致平衡的意思是说从 x 转移到 x' 的概率应该等于从 x' 转移到 x 的概率，在这样的情况下，采样过程将最终收敛到目标分布。也就是说，设计的 MCMC 算法，只要验证其满足细致平衡的条件，那么用这样的方法做基于马尔科夫链的采样，最终就能收敛到一个稳定状态，即目标分布。

实际应用中，有两个马尔科夫链蒙特卡洛采样算法十分常用，它们是 Metropolis-Hastings(梅特罗波利斯-黑斯廷斯)算法与 Gibbs Sampling 采样(吉布斯)，可以证明吉布斯采样是梅特罗波利斯-黑斯廷斯算法的一种特殊情况，而且两者都满足细致平衡的条件。

3.5.3 Metropolis-Hastings 算法

对给定的概率分布，如果从中直接采样比较困难，那么 Metropolis-Hastings 算法就是一个从中获取一系列随机样本的 MCMC 方法。这里所说的 Metropolis-Hastings 算法和模拟退火方法中所使用的 Metropolis-Hastings 算法本质上是一回事。用于模拟退火和 MCMC 的原始算法最初是由 Metropolis 提出的，后来 Hastings 对其进行了推广。Hastings 提出没有必要使用一个对称的参考分布(proposal distribution)。当然达到均衡分布的速度与参考分布的选择有关。

Metropolis-Hastings 算法的执行步骤如下。

1. 初始化 x^0
2. 对于 $i=0$ 到 $N-1$

   $u \sim U(0,1)$

   $x^* \sim q(x^* \mid x^{(i)})$

   如果 $u < \alpha(x^*) = \min\left[1, \dfrac{\pi(x^*) p(x \mid x^*)}{\pi(x) p(x^* \mid x)}\right]$

   $x^{(i+1)} = x^*$

   否则

   $x^{(i+1)} = x^{(i)}$

Metropolis-Hastings 算法的执行步骤是先随便指定一个初始的样本点 x^0，u 是来自均匀分布的一个随机数，x^* 是来自 p 分布的样本点，样本点是否从前一个位置跳转到 x^*，由 $\alpha(x^*)$ 和 u 的大小关系决定。如果接受，则跳转，即新的采样点为 x^*，否则就拒绝，即新的采

用点仍为上一轮的采样点。这个算法看起来非常简单，难免让人疑问照此步骤来执行，最终采样分布能收敛到目标分布吗？根据之前的讲解，可知要想验证这一点，就需要检查细致平衡是否满足。下面是具体的证明过程，不再赘言。

$$\begin{aligned}\pi(x)p(x^*\mid x)\alpha(x^*)&=\pi(x)p(x^*\mid x)\min\left[1,\frac{\pi(x^*)p(x\mid x^*)}{\pi(x)p(x^*\mid x)}\right]\\&=\min[\pi(x)p(x^*\mid x),\pi(x^*)p(x\mid x^*)]\\&=\pi(x^*)p(x\mid x^*)\min\left[1,\frac{\pi(x)p(x^*\mid x)}{\pi(x^*)p(x\mid x^*)}\right]\\&=\pi(x^*)p(x\mid x^*)\alpha(x)\end{aligned}$$

既然已经从理论上证明 Metropolis-Hastings 算法的确实可行，下面举一个简单的例子来看看实际中 Metropolis-Hastings 算法的效果。稍微有点不一样的地方是，这里并未出现 $\pi(x)p(x'|x)$ 这样的后验概率形式，作为一个简单的示例，下面只演示从柯西分布中进行采样的做法。

柯西分布也叫作柯西-洛伦兹分布，它是以奥古斯丁·路易·柯西与亨德里克·洛伦兹名字命名的连续概率分布，其概率密度函数为

$$f(x;x_0,\gamma)=\frac{1}{\pi\gamma\left[1+\left(\frac{x-x_0}{\gamma}\right)^2\right]}=\frac{1}{\pi}\left[\frac{\gamma}{(x-x_0)^2+\gamma^2}\right]$$

其中，x_0 是定义分布峰值位置的位置参数，γ 是最大值一半处的一半宽度的尺度参数。$x_0=0$ 且 $\gamma=1$ 的特例称为标准柯西分布，其概率密度函数为

$$f(x;0,1)=\frac{1}{\pi[1+x^2]}$$

下面是在 R 中进行基于 Metropolis-Hastings 算法对柯西分布进行采样的示例代码。

```
rm(list = ls()) # # 清除全部对象
set.seed(201912)
reps = 40000

# target density: the cauchy distribution
cauchy <- function(x, x0 = 0, gamma = 1){
  out <- 1/(pi * gamma * (1 + ((x - x0)/gamma)^2))
  return(out)
}

chain <- c(0)
for(i in 1:reps){
    proposal <- chain[i] + runif(1, min = -1, max = 1)
    accept <- runif(1)< cauchy(proposal)/cauchy(chain[i])
    chain[i + 1]<- ifelse(accept == T, proposal, chain[i])
}

plot(chain, type = "l")
plot(density(chain[1000:reps]), xlim = c(-5,5), ylim = c(0,0.4), col = "red")
den <- cauchy(seq(from = -5, to = 5, by = 0.1), x = 0, gamma = 1)
lines(den~seq(from = -5, to = 5, by = 0.1), lty = 2, col = "blue")
```

图 3-21 所示为每次采样点的数值分布情况。

为了更清晰明确地看到采样结果符合预期分布，这里也绘制了分布的密度图并与实际的分布密度进行对照，如图 3-22 所示，深色实线是采样分布的密度图，而浅色虚线则是实际柯西分布的密度图，可见两者吻合地相当好。

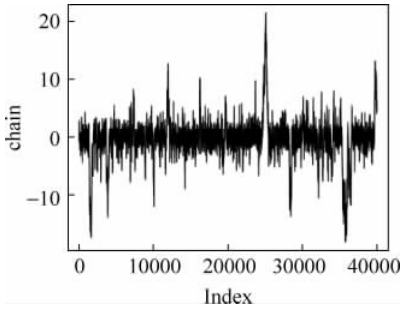

图 3-21 采样点数值分布情况

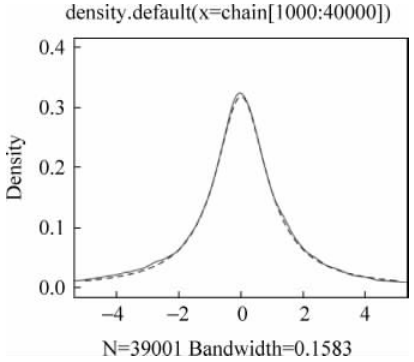

图 3-22 密度图对比

当然，作为一个简单例子的开始，上面的例子中并没有涉及 $p(y|x)$，接下来的这个例子则会演示涉及后验概率的基于 Metropolis-Hastings 算法的 MCMC。这里将用 Metropolis-Hastings 算法从瑞利分布(Rayleigh Distribution)中采样，瑞利分布的密度为

$$f(x)=\frac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}},\quad x\geqslant 0,\sigma>0$$

然后取自由度为 X_t 的卡方分布为参考分布。实现代码如下。

```
rm(list = ls()) # # 清除全部对象

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

m <- 40000
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
        }
    }
```

然后要验证一下，生产的采样数据是否真的符合瑞利分布。注意在 R 中使用瑞利分布的相关函数，需要加装 VGAM 包。下面的代码可以让读者直观地感受到采样结果的分布情况。

```
> curve(drayleigh(x, scale = 4, log = FALSE), from = -1, to = 15, xlim = c(-1,15), ylim =
c(0,0.2),
 + lty = 2, col = "blue",xlab = "", ylab = "")
> par(new = TRUE)
> plot(density(x[1000:m]) , xlim = c(-1,15), ylim = c(0,0.2),col = "red")
```

从图 3-23 中不难看出，采样点分布确实符合预期。当然，这仅仅是一个演示用的小例子。显然，它并不高效。因为采用自由度为 X_t 的卡方分布来作为参考函数，大约有 40%的采样点都被拒绝掉了。如果换做其他更加合适的参考函数，可以大大提升采样的效率。

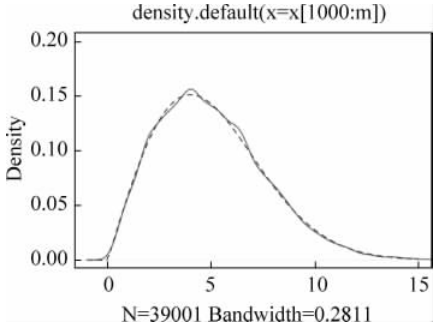

图 3-23　采样点分布

3.5.4　Gibbs 采样

Gibbs 采样是一种用以获取一系列观察值的 MCMC 算法，这些观察值近似于来自一个指定的多变量概率分布，但从该分布中直接采样较为困难。Gibbs 采样可以被看成是 Metropolis-Hastings 算法的一个特例。而这个特殊之处就在于，使用 Gibbs 采样时，通常是对多变量分布进行采样的。比如说，现在有一个分布 $p(z_1,z_2,z_3)$，可见它是定义在三个变量上的分布。这种分布在实际中还有很多，比如一维的正态分布也是关于其均值和方差这两个变量的分布。在算法的第 t 步，选出了三个值 $z_1^{(t)},z_2^{(t)},z_3^{(t)}$。这时，既然 $z_2^{(t)}$ 和 $z_3^{(t)}$ 是已知的，那么可以由此获得一个新的 $z_1^{(t+1)}$，并用这个新的值替代原始的 $z_1^{(t)}$，而新的 $z_1^{(t+1)}$ 可以由下面这个条件分布来获得

$$p(z_1 \mid z_2^{(t)},z_3^{(t)})$$

接下来同样的道理再用一个新的 $z_2^{(t+1)}$ 来替代原始的 $z_2^{(t)}$，而这个新的 $z_2^{(t+1)}$ 可以由下面这个条件分布来获得

$$p(z_2 \mid z_1^{(t+1)},z_3^{(t)})$$

可见刚刚获得的新值 $z_1^{(t+1)}$ 已经被直接用上了。同理，最后用 $z_3^{(t+1)}$ 来更新 $z_3^{(t)}$，而新的值由下面这个条件分布来获得

$$p(z_3 \mid z_1^{(t+1)}, z_2^{(t+1)})$$

按照这个顺序如此往复即可。下面给出更为完整的吉布斯采样的算法描述。

1. 初始化$\{z_i: i=1,\cdots,M\}$
2. 对于 $\tau=1,\cdots,T$：
   - 采样 $z_1^{(\tau+1)} \sim p(z_1 \mid z_2^{(\tau)}, z_3^{(\tau)}, \cdots, z_M^{(\tau)})$
   - 采样 $z_2^{(\tau+1)} \sim p(z_2 \mid z_1^{(\tau+1)}, z_3^{(\tau)}, \cdots, z_M^{(\tau)})$

     $\vdots$
   - 采样 $z_j^{(\tau+1)} \sim p(z_j \mid z_1^{(\tau+1)}, \cdots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)} \cdots, z_M^{(\tau)})$

     $\vdots$
   - 采样 $z_M^{(\tau+1)} \sim p(z_M \mid z_1^{(\tau+1)}, z_2^{(\tau+1)}, \cdots, z_{M-1}^{(\tau+1)})$

当然如果要从理论上证明 Gibbs 采样确实可以得到预期的分布，就应该考察它是否满足细致平衡。但是前面也讲过，Gibbs 采样是 Metropolis-Hastings 算法的一个特例。所以其实甚至无需费力去考察其是否满足细致平衡，如果能够证明吉布斯采样就是 Metropolis-Hastings 算法，而 Metropolis-Hastings 算法是满足细致平衡，其实也就得到了我们想要的。下面是证明的过程，其中 $\boldsymbol{x}_{-k}$ 表示一个向量$(x_1,\cdots,x_{k-1},x_{k+1},\cdots,x_D)$，也就是从中剔除了 x_k 元素。

令 $\boldsymbol{x}=x_1,\cdots,x_D$。

当采样第 k 个元素，$p_k(\boldsymbol{x}^* \mid \boldsymbol{x})=\pi(x_k^* \mid \boldsymbol{x}_{-k})$

当采样第 k 个元素，$\boldsymbol{x}_{-k}^*=\boldsymbol{x}_{-k}$

$$\frac{\pi(\boldsymbol{x}^*)p(\boldsymbol{x} \mid \boldsymbol{x}^*)}{\pi(\boldsymbol{x})p(\boldsymbol{x}^* \mid \boldsymbol{x})}=\frac{\pi(\boldsymbol{x}^*)\pi(x_k \mid \boldsymbol{x}_{-k}^*)}{\pi(\boldsymbol{x})\pi(x_k^* \mid \boldsymbol{x}_{-k})}=\frac{\pi(x_k^* \mid \boldsymbol{x}_{-k}^*)\pi(x_k \mid \boldsymbol{x}_{-k}^*)}{\pi(x_k \mid \boldsymbol{x}_{-k})\pi(x_k^* \mid \boldsymbol{x}_{-k})}=1$$

以 Gibbs 采样为基础实现的 MCMC 在自然语言处理中的 LDA 里有重要应用。如果读者正在或者后续准备研究 LDA 的话，这些内容将是非常重要的基础。