第十五节 · 蒙特卡洛方法与 MCMC

司继春

上海对外经贸大学统计与信息学院

在此之前我们已经学习了一些随机数生成的方法，如分布函数的逆函数方法以及拒绝采样法。在这一节中，我们将介绍一般的蒙特卡洛（Monte Carlo）模拟法，以及给予马尔可夫链的马尔可夫链蒙特卡罗法，这些方法在科学计算、贝叶斯统计中都有广泛的应用。

1 蒙特卡洛积分

在科学计算中，很多问题都可以归结为一个积分计算的问题：

$$I=\int_D h(x)\,dx \tag{1}$$

由于函数 $h(x)$ 可能具有非常复杂的形式，因而其准确的积分值通常难以计算，这时我们可能需要在计算其数值积分。

为了计算该数值积分，第一种解决方案是根据黎曼积分的定义对以上积分进行定义。比如，如果 $D=[0,1]$，那么可以令 $b_i=i/M, i=0,1,...,M$，通过计算：

$$\tilde{I}=\frac{1}{M+1}\sum_{i=0}^{M}h(b_i)$$

对原积分进行逼近。此时，误差率应为 $O\left(M^{-1}\right)$，因而随着 M 的增大，误差会快速降低。然而，在一个更高维度的空间中，比如 $D=[0,1]^{10}$，为了达到 $O\left(M^{-1}\right)$，需要使用 $O\left(M^{10}\right)$ 个点，计算量非常大。

为了解决以上数值积分的问题，另外一种解决方案是使用大数定律。对于积分问题 (1)，可以从 D 上的均匀分布中抽取一系列样本 $(x_1,x_2,...,x_M)$，并使用：

$$\hat{I}=\frac{1}{M}\sum_{i=1}^{M}h(x_i)$$

对积分问题进行逼近。根据大数定律，有：

$$\hat{I}\xrightarrow{p}I$$

以及中心极限定理：

$$\sqrt{M}\left(\hat{I}-I\right) \overset{a}{\sim} N\left(0,\mathrm{Var}\left(h\left(x_i\right)\right)\right)$$

注意到使用这种方法的好处是，无论 D 的维度有多大，收敛的速度（至少理论上）都是 $O\left(M^{-1/2}\right)$，尽管这一速度比一维时的 $O\left(M^{-1}\right)$ 要慢，然而成功的避免了维数的诅咒。

注意以上 D 有有界的支撑集，如果 D 的支撑集时无界的，比如 $D=\mathbb{R}$，我们可以选取一个取值范围为 $\mathbb{R}$ 的密度函数 $f\left(x\right)$，并从 $f\left(x\right)$ 中进行抽样，得到样本 $\left(x_1,...,x_M\right)$，进而计算

$$\hat{I}=\frac{1}{M}\sum_{i=1}^{M}\frac{h\left(x_i\right)}{f\left(x_i\right)}$$

根据大数定律，有：

$$\hat{I}\xrightarrow{p}\mathbb{E}_f\left(\frac{h\left(x_i\right)}{f\left(x_i\right)}\right)=\int_{\mathbb{R}}\frac{h\left(x\right)}{f\left(x\right)}f\left(x\right)dx=\int_{\mathbb{R}}h\left(x\right)dx$$

以及中心极限定理：

$$\sqrt{M}\left(\hat{I}-I\right)\overset{a}{\sim}N\left(0,\mathrm{Var}\left(\frac{h\left(x_i\right)}{f\left(x_i\right)}\right)\right)$$

其中：

$$\begin{aligned}\mathrm{Var}\left(\frac{h\left(x_i\right)}{f\left(x_i\right)}\right)&=\mathbb{E}_f\left[\left(\frac{h\left(x_i\right)}{f\left(x_i\right)}\right)^2\right]-\left[\mathbb{E}_f\left(\frac{h\left(x_i\right)}{f\left(x_i\right)}\right)\right]^2\\&=\int_{\mathbb{R}}\frac{\left[h\left(x\right)\right]^2}{\left[f\left(x\right)\right]^2}f\left(x\right)dx-I^2\\&=\int_{\mathbb{R}}\frac{\left[h\left(x\right)\right]^2}{f\left(x\right)}dx-I^2\end{aligned}$$

以上便是经典的蒙特卡洛积分（Monte Carlo integration）方法。

2 重要性抽样

以上我们介绍了经典的蒙特卡洛积分方法，然而注意到，在一些情况下，以上算法可能时非常没有效率的。

例 1. 令 $\left(x_1,x_2\right)\in\left[-1,1\right]\times\left[-1,1\right]$，以及：

$$h\left(x,y\right)=\frac{1}{2}\exp\left\{-90\left(x_1-0.5\right)^2-45\left(x_2+0.1\right)^2\right\}+\exp\left\{-45\left(x_1+0.4\right)^2-60\left(x_2-0.5\right)^2\right\}$$

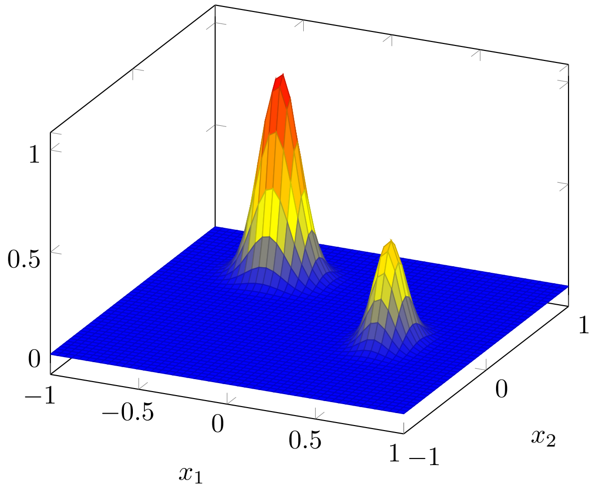

图 1: 例 (1) 函数示意图

如图 (1) 所示，该函数在定义域内有大量的区域取值几乎为 0，因而如果我们从 $[-1,1]\times[-1,1]$ 区域内取均匀分布，会有大量的抽样样本被浪费，尽管收敛速度仍然是 $O\left(M^{-1/2}\right)$，然而效率仍然不高。注意到以上函数是两部分相加的形式，因而我们不妨分开计算。不失一般性，我们以第一部分作为示例。以下程序给出了使用均匀分布抽样计算以上积分中第一部分的示例：

代码 1: 简单的 Monte Carlo 方法

```
#!/usr/bin/python3
## file: MonteCarlo.py

import numpy as np
from numpy import random as nprd

##设定参数
M=10000
h=lambda x: 0.5*np.exp(-90*(x[0]-0.5)**2-45*(x[1]+0.1)**2)

##抽样
x=[(nprd.random()*2-1,nprd.random()*2-1) for i in range(M)]

##计算h(x)
sample=list(map(h,x))
integral=4*np.mean(sample)
se=4*np.std(sample)/np.sqrt(M)
subsample_001=list(filter(lambda x: x>0.01, sample))
```

```
print("Intgral=",integral)
print("s.e. of Integral=",se)
print("95% C.I.:",integral-1.96*se,"~",integral+1.96*se)
print("Ratio of >0.01 samples:",len(subsample_001)/M)
```

计算的积分值约为 0.026，以上程序还给出了 95% 的置信区间。注意到根据抽样计算，$[-1,1]\times[-1,1]$ 中只有约 5% 的区域函数值 >0.01，而最大值为 0.5，对样本是比较大的浪费。

为了解决这一问题，一个比较现实的解决方案是在抽样时给予函数值不为 0 的部分以更高的权重。不失一般性，如果我们希望计算积分：

$$I=\int h\left(x\right)\pi\left(x\right)dx=\mathbb{E}_{\pi}\left[h\left(x\right)\right]$$

其中 $\pi\left(x\right)$ 为一个密度函数，那么我们可以使用以下算法：

算法 1. 重要性抽样（Importance sampling）

1. 从一个工具密度函数 $g\left(x\right)$ 中抽取样本 $\left(x_1,...,x_M\right)$
2. 计算：
$$\hat{I}=\frac{1}{M}\sum_{i=1}^{M}h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}$$

注意到根据大数定律：

$$\hat{I}\xrightarrow{p}\mathbb{E}_g\left[h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right]=\int h\left(x\right)\frac{\pi\left(x\right)}{g\left(x\right)}g\left(x\right)dx=\int h\left(x\right)\pi\left(x\right)dx$$

而其方差：

$$\begin{aligned}\mathrm{Var}_g\left[h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right]&=\mathbb{E}_g\left[\left(h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)^2\right]-\left[\mathbb{E}_g\left(h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)\right]^2\\&=\mathbb{E}_g\left[\left(h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)^2\right]-I^2\end{aligned}$$

因而最终 $\mathrm{Var}\left(\hat{I}\right)$ 与 $g\left(\cdot\right)$ 的选取有关系。为了保证以上方差有限，必须要保证：

$$\mathbb{E}_g\left[\left(h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)^2\right]=\int h^2\left(x\right)\frac{\pi^2\left(x\right)}{g\left(x\right)}dx=\mathbb{E}_{\pi}\left[h^2\left(x\right)\frac{\pi\left(x\right)}{g\left(x\right)}\right]<\infty$$

这就要求工具密度函数 $g\left(x\right)$ 的尾部不应该比 $\pi\left(x\right)$ 还要薄，否则如果 $\pi\left(x\right)/g\left(x\right)$ 是无界的，那么就给了尾部以太多的权重，因而计算出的积分值非常不稳定。

理论上，应该存在一个使得以上方差最小的工具密度函数 $g\left(\cdot\right)$。以下定理给出了理论上最优的工具密度函数。

定理 1. 使得 $Var\left(\hat{I}\right)$ 最小的密度函数为：

$$g^*\left(x\right)=\frac{\left|h\left(x\right)\right|\pi\left(x\right)}{\int\left|h\left(x\right)\right|\pi\left(x\right)dx}$$

证明. 由于 I^2 与 $g\left(\cdot\right)$ 无关，因而最小化 $\mathrm{Var}\left(\hat{I}\right)$ 等价于最小化：

$$\mathbb{E}_g\left[\left(h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)^2\right]$$

根据 Jensen 不等式，有：

$$\mathbb{E}_g\left[\left(h\left(x_i\right)\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)^2\right]\geq\left[\mathbb{E}_g\left(\left|h\left(x_i\right)\right|\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)\right]^2=\left[\int\left|h\left(x\right)\right|f\left(x\right)dx\right]^2$$

可以验证，当 $g\left(x\right)=g^*\left(x\right)$ 时，达到了最小值。 □

以上定理表明，如果希望达到最小方差，我们必须给予 $\left|h\left(x\right)\right|$ 或者 $\pi\left(x\right)$ 比较大的区域以更大的权重。然而，一个悖论是，为了得到最优的工具密度函数 $g^*\left(x\right)$，必须首先知道 $\int\left|h\left(x\right)\right|\pi\left(x\right)dx$，而这正是我们感兴趣的积分。因而在绝大多数情况下，以上最优的工具密度函数并不可行。

重要性抽样还有另外一种算法：

算法 2. 重要性抽样的简单形式

1. 从一个工具密度函数 $g\left(x\right)$ 中抽取样本 $\left(x_1,...,x_M\right)$
2. 计算权重
$$\omega_i=\frac{\pi\left(x_i\right)}{g\left(x_i\right)}$$
3. 计算：
$$\check{I}=\frac{\sum_{i=1}^{M}h\left(x_i\right)\omega_i}{\sum_{i=1}^{M}\omega_i}$$

由于：

$$\frac{1}{M}\sum_{i=1}^{M}\omega_i\xrightarrow{p}\mathbb{E}_g\left(\frac{\pi\left(x_i\right)}{g\left(x_i\right)}\right)=\int\frac{\pi\left(x\right)}{g\left(x\right)}g\left(x\right)dx=\int\pi\left(x\right)dx=1$$

从而

$$\check{I}\xrightarrow{p}\mathbb{E}_g\left(h\left(x_i\right)\omega_i\right)=\int h\left(x\right)\pi\left(x\right)dx$$

然而以上方法并不是无偏的，尽管在一些情况下，$\check{I}$ 可能会有比较好的表现。

算法2的好处在于，我们只需要计算 $\frac{\pi(x_i)}{g(x_i)}$ 的比例，而不需要两者都进行计算。比如，如果我们取 $g=g^*$，由于 g^* 的分母为一个常数，因而我们无需计算

其分母，只要计算其分子 $|h(x)|\pi(x)$ 即可。将 g^* 带入到上述算法中，得到：

$$
\begin{aligned}
\check{I} &= \frac{\sum_{i=1}^{M} h(x_i)\,\omega_i}{\sum_{i=1}^{M}\omega_i} \\
&= \frac{\sum_{i=1}^{M} h(x_i)\,\frac{\pi(x_i)}{g^*(x_i)}}{\sum_{i=1}^{M}\frac{\pi(x_i)}{g^*(x_i)}} \\
&= \frac{\sum_{i=1}^{M} h(x_i)\,\frac{\pi(x_i)}{|h(x_i)|\pi(x_i)}}{\sum_{i=1}^{M}\frac{\pi(x_i)}{|h(x_i)|\pi(x_i)}} \\
&= \frac{\sum_{i=1}^{M}\frac{h(x_i)}{|h(x_i)|}}{\sum_{i=1}^{M}\frac{1}{|h(x_i)|}}
\end{aligned}
$$

注意到分子上就是 $h(x_i)$ 等于正数的个数和负数的个数之差。同时，从 g^* 中抽样可以使用拒绝采样法，同样不需要计算分子 $|h(x)|\pi(x)$。然而，g^* 的最优性是在算法1的条件下得到的，并不适用于算法2，因而这一算法2并不是最优的，甚至有可能得到不稳定的结果。

例 2. 在例 (1) 中，我们使用简单的蒙特卡洛方法计算了函数

$$h_1(x_1,x_2) = \frac{1}{2}\exp\left\{-90(x_1-0.5)^2 - 45(x_2+0.1)^2\right\}$$

的积分。现在我们使用算法2计算以上积分。不是一般性，我们令

$$\pi(x,y) = \frac{1}{4}\cdot 1\{-1\le x_1\le 1\}\cdot 1\{-1\le x_2\le 1\}$$

从而：

$$
\begin{aligned}
\int_{[-1,1]\times[-1,1]} h_1(x_1,x_2)\,dxdy &= \int_{\mathbb{R}} [4h_1(x_1,x_2)]\cdot\frac{1}{4}\cdot 1\{-1\le x_1\le 1\}\cdot 1\{-1\le x_2\le 1\}\,dx_1dx_2 \\
&= \int_{\mathbb{R}} h(x_1,x_2)\cdot\pi(x_1,x_2)\,dx_1dx_2
\end{aligned}
$$

算法2的第一步是找到一个工具密度 $g(\cdot)$，并在该密度上进行抽样。在这里，我们不妨令：

$$
\begin{aligned}
g(x_1,x_2) = g^*(x_1,x_2) &= \frac{|h(x_1,x_2)|\,\pi(x_1,x_2)}{\int_{\mathbb{R}}|h(x_1,x_2)|\,\pi(x_1,x_2)\,dx_1dx_2} \\
&\propto \exp\left\{-90(x_1-0.5)^2-45(x_2+0.1)^2\right\}\cdot 1\{-1\le x_1\le 1\}\cdot 1\{-1\le x_2\le 1\} \\
&\triangleq l(x_1,x_2)
\end{aligned}
$$

我们使用拒绝采样法从 $g(x_1,x_2)$ 中进行抽样，为此我们需要使用一个工具密度

函数 $m(x_1,x_2)$ 使得存在一个常数 R 有：

$$R\cdot m(x_1,x_2)\geq l(x_1,x_2)$$

观察 $l(x_1,x_2)$ 的形式，我们不妨使用一个独立的联合正态的密度函数：

$$m\left(x_1,x_2;\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)=\frac{1}{2\pi\sigma_1\sigma_2}\exp\left\{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}\right\}$$

从而，R 需要满足：

$$R\left(\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)=\max_{x,y}\frac{l(x,y)}{m\left(x_1,x_2;\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)}$$

或者：

$$\ln R\left(\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)=\max_{x,y}\left[\ln l(x,y)-\ln m\left(x_1,x_2;\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)\right]$$

以上最大化问题的一阶条件为：

$$\begin{cases}x_1^*=\frac{90\sigma_1^2-2\mu_1}{180\sigma_1^2-2}\\ x_2^*=\frac{9\sigma_2^2+2\mu_2}{2-90\sigma_2^2}\end{cases}$$

二阶条件要求其海塞矩阵为负定矩阵：

$$\begin{bmatrix}-180+\frac{1}{\sigma_1^2} & 0\\ 0 & -90+\frac{1}{\sigma_1^2}\end{bmatrix}\leq 0$$

从而：

$$\begin{cases}\sigma_1^2\geq\frac{1}{180}\\ \sigma_2^2\geq\frac{1}{90}\end{cases}$$

注意到 R 为 $\left(\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)$ 的函数，由于 $\frac{1}{R}$ 度量了拒绝采样中接受的概率，因而我们需要最小化 R，根据包络定理：

$$\begin{cases}\frac{\partial\ln R}{\partial\mu_1}=-\frac{\partial\ln m\left(x_1^*,x_2^*;\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)}{\partial\mu_1}=0\\ \frac{\partial\ln R}{\partial\mu_2}=-\frac{\partial\ln m\left(x_1^*,x_2^*;\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)}{\partial\mu_2}=0\\ \frac{\partial\ln R}{\partial\sigma_1^2}=-\frac{\partial\ln m\left(x_1^*,x_2^*;\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)}{\partial\sigma_1^2}=0\\ \frac{\partial\ln R}{\partial\sigma_2^2}=-\frac{\partial\ln m\left(x_1^*,x_2^*;\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\right)}{\partial\sigma_2^2}=0\end{cases}$$

解得：

$$\begin{cases} \mu_1 = 0.5 \\ \mu_2 = -0.1 \\ \sigma_1^2 = \frac{1}{180} \\ \sigma_2^2 = \frac{1}{90} \end{cases}$$

如果使用算法1，我们可以直接先从 $m\left(x_1, x_2; \mu_1, \mu_2, \sigma_1^2, \sigma_2^2\right)$ 中进行抽样根据算法1进行计算。以下代码给出了使用算法1计算该积分的过程：

代码 2: 重要性抽样代码示例

```python
#!/usr/bin/python3
## file: MonteCarlo.py

import numpy as np
from numpy import random as nprd

##设定参数
M =10000
h1=lambda x: 0.5*np.exp(-90*(x[0]-0.5)**2-45*(x[1]+0.1)**2)
domain=lambda x:(x[0]>=-1)*(x[1]>=-1)*(x[0]<=1)*(x[1]<=1)
pai=lambda x: 1/4*domain(x)
h=lambda x: 4*h1(x)
mu1=0.5
mu2=-0.1
sigma1=np.sqrt(1/180)
sigma2=np.sqrt(1/90)
m=lambda x: 1/(2*np.pi*sigma1*sigma2)* \
      np.exp(-1*(x[0]-mu1)**2/(2*sigma1**2)\
            -(x[1]-mu2)**2/(2*sigma2**2))

#从m(x)中采样
x=[(nprd.normal(mu1,sigma1),nprd.normal(mu2,sigma2))\
    for i in range(M)]

## 计算积分
H=list(map(lambda x:h(x)*pai(x)/m(x),x))
integral=np.mean(H)
se=np.std(H)/np.sqrt(M)
print("Intgral=",integral)
print("s.e. of Integral=",se)
```

```
print("95% C.I.:",integral-1.96*se,"~",integral+1.96*se)
```

可以发现，使用重要性抽样大大降低了标准误。而如果使用算法2，我们可以先从 $x_i \sim N\left(0.5, \frac{1}{180}\right)$，$y_i \sim N\left(-0.1, \frac{1}{90}\right)$ 的样本中采样，使用拒绝采样法从 $g\left(x_1, x_2\right) = g^*\left(x_1, x_2\right)$ 中采样，进而计算：

$$\check{I} = \frac{\sum_{i=1}^{M} \frac{h(x_{1i}, x_{2i})}{|h(x_{1i}, x_{2i})|}}{\sum_{i=1}^{M} \frac{1}{|h(x_{1i}, x_{2i})|}} = \frac{\sum_{i=1}^{M} \frac{h_1(x_{1i}, x_{2i})}{|h_1(x_{1i}, x_{2i})|}}{\sum_{i=1}^{M} \frac{1}{|4h_1(x_{1i}, x_{2i})|}} = \frac{4}{\frac{1}{N}\sum_{i=1}^{M} \frac{1}{|h_1(x_{1i}, x_{2i})|}}$$

然而注意到，在分母上，如果不限制积分区域，那么：

$$\mathrm{Var}\left(\frac{1}{\left|h_1\left(x_1, x_2\right)\right|}\right) \to \infty$$

即使积分区域被限制在 $[-1,1] \times [-1,1]$ 的区域内，该方差也非常的大，因而会造成积分计算的不稳定。实际上，根据计算的结果，该方法存在严重偏差。

3 马尔可夫链蒙特卡洛

在此之前我们介绍了拒绝采样法以及重要性抽样两种方法，其中拒绝采样法可以帮助我们从一个一般的密度函数（如 $\pi(x)$）中抽取随机数，而重要性抽样可以帮助我们计算形如

$$\int h(x)\,\pi(x)\,dx = \mathbb{E}_\pi\left[h(x)\right]$$

的积分。而我们知道，为了计算以上积分，我们也可以通过在 $\pi(x)$ 中抽取独立同分布的样本 $(x_1, ..., x_M)$，根据大数定律，有：

$$\frac{1}{M}\sum_{i=1}^{M} h(x_i) \xrightarrow{p} \int h(x)\,\pi(x)\,dx$$

然而无论是拒绝采样法还是重要性抽样，都要求我们对密度函数 $\pi(x)$ 有足够多的了解，比如在拒绝采样中，我们必须找到能够覆盖 $\pi(x)$ 的常数和工具密度函数，而在重要性抽样中，同样需要 $h(x)$ 和 $g(x)$ 的知识以保证方差有界等。

而马尔可夫链蒙特卡洛方法则另辟蹊径，不寻求从 $\pi(x)$ 中抽取独立同分布的样本，而是构造一个遍历的马尔可夫链 $\{x_t, t = 1, 2, ...\}$，使得该马尔可夫链的平稳分布为 $\pi(x)$。根据遍历性定理，仍然有：

$$\frac{1}{M}\sum_{t=1}^{M} h(x_t) \xrightarrow{p} \int h(x)\,\pi(x)\,dx$$

定义 1. 马尔可夫链蒙特卡洛（**Markov chain Monte Carlo, MCMC**）方法指的是，模拟一个遍历的马尔可夫链，使得该马尔可夫链的平稳分布为 $\pi(x)$。

因而接下来要解决的问题是，如何才能构造这样的马尔可夫链。回想一般状态马尔可夫链，为了构造一条马尔可夫链，我们需要构造一个转移概率函数 $P(x,A)$，或者一个转移核 $K(x,y)=f_{x_{n+1}|x_n=x}(y|x)$。

3.1 Metropolis-Hastings 算法

在所有的 MCMC 算法中，Metropolis-Hastings 算法是其中最为常用的通用算法。该算法需要设定一个条件密度 $q(y|x)$，我们称之为工具分布（instrumental distribution）。该条件密度需要比较容易的从中进行抽样，并且要求如果 $q(y|x)>0$，那么必须有 $q(x|y)>0$。在实际使用中，该条件密度经常选为对称的，即 $q(x|y)=q(y|x)$，或者与目标分布 $\pi(x)$ 比较相近的密度函数。有了 $q(y|x)$ 之后，Metropolis-Hastings 算法如下：

算法 3. （Metropolis-Hastings 算法）

1. 给定现在的状态 x_t，产生一个随机数 $y_t\sim q(y|x)$
2. 令：
$$\rho(x,y)=\min\left\{\frac{\pi(y)}{\pi(x)}\frac{q(x|y)}{q(y|x)},1\right\}$$
生成一个 $U\sim U(0,1)$，根据以下规则生成 x_{t+1}：
$$x_{t+1}=\begin{cases}y_t & if\ U<\rho(x_t,y_t)\\ x_t & else\end{cases}$$

在该算法中，每一步都进行一次接受/拒绝抽样，其中 $\rho(x,y)$ 一般称之为接受率。形象的理解是，该算法在每一步 x_t 时都尝试转移到新的状态 y_t，当：

- y_t 的目标密度函数值 $\pi(y_t)$ 更大，或者
- 从 y_t 转移回当前状态 x_t 比较容易

时，x_t 以更大的概率转移到新的状态 y_t；如果没有转移，那么在 $t+1$ 时刻保持 t 时刻的状态不动，并在 $t+2$ 时刻重新进行尝试。

此外，除了以上的接受算法之外，还有其他的接受算法。比如 Barker (1965) 提出，可以使用：
$$\rho_B(x,y)=\frac{\pi(y)q(x|y)}{\pi(y)q(x|y)+\pi(x)q(y|x)}$$
作为接受率。而 Stein 提出了一个更一般的接受率：
$$\rho_S(x,y)=\frac{d(x,y)}{\pi(x)q(y|x)} \tag{2}$$
其中 $d(x,y)=d(y,x)$ 且 $\rho_S(x,y)\le 1$。

为了证明以上过程所产生的马尔可夫链的平稳分布为 $\pi\left(x\right)$，我们必须要证明:

$$\int \pi\left(x\right) K\left(x,y\right) dx = \pi\left(y\right)$$

实际上，如果 $\pi\left(x\right) K\left(x,y\right) = \pi\left(y\right) K\left(y,x\right)$ 满足，那么:

$$\int \pi\left(x\right) K\left(x,y\right) dx = \int \pi\left(x\right) \frac{\pi\left(y\right) K\left(y,x\right)}{\pi\left(x\right)} dx = \int \pi\left(y\right) K\left(y,x\right) dx = \pi\left(y\right)$$

从而我们仅仅需要验证 $\pi\left(x\right) K\left(x,y\right) = \pi\left(y\right) K\left(y,x\right)$ 是否满足。

然而要注意，在 Metropolis-Hastings 算法中引入的工具分布 $q\left(y|x\right)$ 并不是该马尔可夫链实际的转移核 $K\left(x,y\right)$，因为在 Metropolis-Hastings 算法中还有拒绝的步骤。实际上，在 Metropolis-Hastings 算法中，转移核可以写为:

$$K\left(x,y\right) = \rho\left(x,y\right) q\left(y|x\right) + \left[1 - \int \rho\left(x,y\right) q\left(y|x\right) dy\right] \delta_x\left(y\right)$$

其中 $\delta_x\left(y\right)$ 为 Dirac 函数，即满足:

$$\delta_x\left(y\right) = \begin{cases} +\infty & y = x \\ 0 & y \neq x \end{cases}$$

以及

$$\int_{\mathbb{R}} \delta_x\left(y\right) dy = 1$$

即在 x 处的质量函数。该转移核意味着转移规则是以概率 $\rho\left(x,y\right)$ 取的从 $q\left(y|x\right)$ 中抽样得到的值，而以概率 $1 - \int \rho\left(x,y\right) q\left(y|x\right) dy$ 取 x 不变，即 Metropolis-Hastings 算法的过程。

可以验证:

$$\begin{aligned} \pi\left(x\right) \rho\left(x,y\right) q\left(y|x\right) &= \pi\left(x\right) \min\left\{\frac{\pi\left(y\right)}{\pi\left(x\right)} \frac{q\left(x|y\right)}{q\left(y|x\right)}, 1\right\} q\left(y|x\right) \\ &= \min\left\{\pi\left(y\right) q\left(y|x\right), \pi\left(x\right) q\left(y|x\right)\right\} \\ &= \pi\left(y\right) \rho\left(y,x\right) q\left(x|y\right) \end{aligned}$$

以及:

$$\pi\left(x\right) \left[1 - \int \rho\left(x,y\right) q\left(y|x\right) dy\right] \delta_x\left(y\right) = \pi\left(y\right) \left[1 - \int \rho\left(y,x\right) q\left(x|y\right) dx\right] \delta_y\left(x\right)$$

因而在 M-H 算法中，$\pi\left(x\right) K\left(x,y\right) = \pi\left(y\right) K\left(y,x\right)$ 满足，因而 $\pi\left(x\right)$ 是该马尔可夫链的平稳分布。

更进一步，该马尔可夫链的收敛性需要使用一般状态空间的遍历性定理，在此不做讨论。我们不做证明的引入如下定理:

定理 2. 对于算法3，如果：

1. $P\left[\frac{\pi(y_t)}{\pi(x_t)}\frac{q(x_t|y_t)}{q(y_t|x_t)} \geq 1\right] < 1$
2. 记 $D = \{x : \pi(x) > 0\}$，对于所有的 $(x, y) \in D \times D$，有：$q(y|x) > 0$

满足，那么对于所有满足 $\int |h(x)| \pi(x)\, dx < \infty$ 的函数 $h(x)$，有：

$$\frac{1}{M}\sum_{t=1}^{M} h(x_t) \xrightarrow{a.s.} \int h(x)\pi(x)\, dx$$

其中第一条假设要求以正的概率，$X_{t+1} = X_t$ 成立；第二条要求每一次进行转移时，工具分布 $q(y|x)$ 可以遍历 $\pi(x)$ 取值的所有可能性。此外，第一条假设还意味着，工具密度 $q(y|x)$ 不可以等于目标密度 $\pi(y)$。

接下来，我们将分别介绍 MCMC 算法的两种特殊形式。

3.2 独立的 Metropolis-Hastings 算法

该方法仍然建立在算法3的基础上，只不过在该方法中选取工具分布 $p(y|x) = g(y)$，即直接选取一个密度函数，与 t 时刻的状态 x_t 无关，因而每一次抽取的 y_t 都是独立的。

需要注意的时，尽管每一次产生的 y_t 都是独立的，但是 x_t 并非独立的，因为算法3中还有拒绝-接受的过程。

例 3. 在例 (2) 中，我们使用重要性抽样计算了函数

$$h_1(x_1, x_2) = \frac{1}{2}\exp\left\{-90(x_1 - 0.5)^2 - 45(x_2 + 0.1)^2\right\}$$

的积分。现在，我们使用独立的 Metropolis-Hastings 算法计算积分：

$$h_1(x_1, x_2) = \frac{1}{2}h(x_1, x_2)\exp\left\{-90(x_1 - 0.5)^2 - 45(x_2 + 0.1)^2\right\}$$

当

$$h(x_1, x_2) = 1\{-1 \leq x_1 \leq 1\} \cdot 1\{-1 \leq x_2 \leq 1\}$$

时即例 (2) 中的积分，同样，我们也可以令 $u(x)$ 为其他函数，比如我们令

$$h(x_1, x_2) = \sin^2(10x_1) + \ln|1 + 10x_2|$$

首先，我们不妨将以上函数改写为：

$$\begin{aligned}
h_1(x_1,x_2) &= \frac{1}{2}h(x_1,x_2)\exp\left\{-\frac{(x_1-0.5)^2}{2\times\frac{1}{180}}-\frac{(x_2+0.1)^2}{2\times\frac{1}{90}}\right\}\\
&= \frac{\pi}{2\pi}\frac{\sqrt{\frac{1}{180}}\sqrt{\frac{1}{90}}}{\sqrt{\frac{1}{180}}\sqrt{\frac{1}{90}}}h(x_1,x_2)\exp\left\{-\frac{(x_1-0.5)^2}{2\times\frac{1}{180}}-\frac{(x_2+0.1)^2}{2\times\frac{1}{90}}\right\}\\
&= \pi\sqrt{\frac{1}{180}}\sqrt{\frac{1}{90}}h(x_1,x_2)\frac{1}{2\pi\sqrt{\frac{1}{180}}\sqrt{\frac{1}{90}}}\exp\left\{-\frac{(x_1-0.5)^2}{2\times\frac{1}{180}}-\frac{(x_2+0.1)^2}{2\times\frac{1}{90}}\right\}\\
&\triangleq \frac{\pi}{90\sqrt{2}}h(x_1,x_2)\cdot\pi(x_1,x_2)
\end{aligned}$$

其中

$$\pi(x_1,x_2)=\frac{1}{2\pi\sqrt{\frac{1}{180}}\sqrt{\frac{1}{90}}}\exp\left\{-\frac{(x_1-0.5)^2}{2\times\frac{1}{180}}-\frac{(x_2+0.1)^2}{2\times\frac{1}{90}}\right\}$$

为一个联合正态的密度函数。我们需要计算积分：

$$\int h_1(x_1,x_2)\,dx_1dx_2=\frac{\pi}{90\sqrt{2}}\int\int h(x_1,x_2)\cdot\pi(x_1,x_2)\,dxdy$$

其中 $\pi(x_1,x_2)$ 为一个密度函数。接下来我们使用独立的 Metropolis-Hastings 算法从 $\pi(x_1,x_2)$ 中进行抽样，并选取工具分布

$$p(y_1,y_2|x_1,x_2)=\frac{1}{2\pi}\exp\left\{-\frac{y_1^2}{2}-\frac{y_2^2}{2}\right\}$$

即标准正态分布。在得到样本 $\{(x_{1t},x_{2t}),t=1,2,...\}$ 后，计算：

$$\hat{I}=\frac{\pi}{90\sqrt{2}}\frac{1}{M}\sum_{t=1}^{M}h(x_{1t},x_{2t})$$

注意实际上，在这里我们可以直接从联合正态分布中进行抽样，不过在这里为了展示，仍然使用 MCMC 算法从 $\pi(x_1,x_2)$ 中进行抽样。代码如下：

代码 3: 独立的 Metropolis-Hastings 算法示例

```python
#!/usr/bin/python3
## file: MonteCarlo.py

import numpy as np
from numpy import random as nprd

```

```
##设定参数
M =10000
pai_cons=90*np.sqrt(2)/(2*np.pi)
pai=lambda x: pai_cons* \
      np.exp(-90*(x[0]-0.5)**2-45*(x[1]+0.1)**2)
domain=lambda x:(x[0]>=-1)*(x[1]>=-1)*(x[0]<=1)*(x[1]<=1)
q=lambda y: 1/(2*np.pi)*np.exp(-1*y[0]**2/2-y[1]**2/2)
h=lambda x: domain(x)
h2=lambda x: np.sin(10*x[0])**2+np.log(abs(1+x[1]*10))
#从均匀分布中采样
def sample_q():
    return (nprd.normal(),nprd.normal())


##独立的MCMC算法，输入：
##      N_samples   :  抽样次数
##        pai(x)    :  目标密度函数
##        q(y)      :  工具密度函数
##     q_sampler    :  给定x，从q中抽样的函数
##        x0        :  初始值
def MH_independent(N_samples, pai, q, q_sampler, x0):
    X=[]
    x=x0
    for i in range(N_samples):
        y=q_sampler()
        rho=min(1,pai(y)*q(x)/(pai(x)*q(y)))
        if nprd.uniform()<rho:
            X.append(y)
            x=y
        else:
            X.append(x)
    return X

## 计算积分
x=MH_independent(M, pai, q, sample_q, (0,0))
## 第一个积分
H=list(map(h,x))
## 取h后面80%的样本
subH=H[int(M*0.2):]
integral=np.pi/(90*np.sqrt(2))*np.mean(subH)
```

```
print("Intgral1=",integral)
## 第二个积分
H2=list(map(h2,x))
subH2=H2[int(M*0.2):]
integral2=np.pi/(90*np.sqrt(2))*np.mean(subH2)
print("Intgral2=",integral2)
```

注意在该算法中，接受率为：

$$\rho(x,y)=\min\left\{\frac{\pi(y)}{\pi(x)}\frac{g(x)}{g(y)},1\right\}$$

为了降低拒绝率或者增加接受率，我们需要挑选一个与目标密度函数 $\pi(x)$ 比较接近的密度函数 $g(x)$。实际上，该算法与拒绝采样法非常相近，不过一般情况下比拒绝采样法更加有效。

3.3 随机游走的 Metropolis-Hastings 算法

在以上的独立的 Metropolis-Hastings 算法中，如果选择的工具密度 $g(\cdot)$ 与目标密度函数 $\pi(\cdot)$ 差距很大，拒绝率一般比较高，算法的效率较低。一个自然的想法时，我们可以充分使用现有的位置信息更新下一次的位置，当马尔可夫链运行足够长的时间时，马尔可夫链倾向于停留在 $\pi(\cdot)$ 比较高的区域，因而在现有的位置附近更新下一次的尝试 y_t，可以大大提高接受率。为此，一个解决方案是使用随机游走，即令：

$$y_t=x_t+\epsilon_t$$

其中 $\epsilon_t\sim g(\cdot)$，我们要去密度函数 $g(\cdot)$ 依 y 轴对称，即 $g(x)=g(-x)$。此时的工具密度函数为：

$$q(y|x)=g(|y-x|)=q(x|y)$$

因而此时工具密度函数时对称的，此时接受率：

$$\rho(x,y)=\min\left\{\frac{\pi(y)}{\pi(x)}\frac{q(x|y)}{q(y|x)},1\right\}=\min\left\{\frac{\pi(y)}{\pi(x)},1\right\}$$

与 $q(y|x)$ 无关。

例 4. 下面我们使用随机游走的 Metropolis-Hastings 算法，并取 $\epsilon_{1,t}\sim N\left(0,0.2^2\right)$，$\epsilon_{2,t}\sim N\left(0,0.2^2\right)$，令

$$\begin{aligned}y_{1t}&=x_{1t}+\epsilon_{1,t}\\y_{2t}&=x_{2t}+\epsilon_{2,t}\end{aligned}$$

并使用 Metropolis-Hastings 算法进行抽样，代码如下：

代码 4: 随机游走的 Metropolis-Hastings 算法示例

```
#!/usr/bin/python3
## file: MonteCarlo.py

import numpy as np
from numpy import random as nprd

##设定参数
M =10000
pai=lambda x: np.exp(-90*(x[0]-0.5)**2-45*(x[1]+0.1)**2)
domain=lambda x:(x[0]>=-1)*(x[1]>=-1)*(x[0]<=1)*(x[1]<=1)
h=lambda x: domain(x)
h2=lambda x: np.sin(10*x[0])**2+np.log(abs(1+x[1]*10))
#从均匀分布中采样
def sample_q(x):
    return (x[0]+0.2*nprd.normal(),x[1]+0.2*nprd.normal())


##独立的MCMC算法, 输入:
##      N_samples   : 抽样次数
##        pai(x)    : 目标密度函数
##     q_sampler(x): 给定x, 从q中抽样的函数
##        x0        : 初始值
def MH_RW(N_samples, pai, q_sampler, x0):
    X=[]
    x=x0
    for i in range(N_samples):
        y=q_sampler(x)
        rho=min(1,pai(y)/pai(x))
        if nprd.uniform()<rho:
            X.append(y)
            x=y
        else:
            X.append(x)
    return X

## 计算积分
x=MH_RW(M, pai, sample_q, (0,0))
## 第一个积分
H=list(map(h,x))
```

```
## 取h后面80%的样本
subH=H[int(M*0.2):]
integral=np.pi/(90*np.sqrt(2))*np.mean(subH)
print("Intgral1=",integral)
## 第二个积分
H2=list(map(h2,x))
subH2=H2[int(M*0.2):]
integral2=np.pi/(90*np.sqrt(2))*np.mean(subH2)
print("Intgral2=",integral2)
```

接下来我们使用该算法计算一个 Logistic 回归的后验分布问题。

例 5. 假设 d_i 为一个 0/1 变量，且给定自变量 w_{1i}, w_{2i}，$d_i=1$ 的概率为：

$$P\left(d_i=1|w_{1i},w_{2i},\beta\right)=\frac{\exp\left\{\beta_0+\beta_1 w_{1i}+\beta_2 w_{2i}\right\}}{1+\exp\left\{\beta_0+\beta_1 w_{1i}+\beta_2 w_{2i}\right\}}=F\left(\beta_0+\beta_1 w_{1i}+\beta_2 w_{2i}\right)$$

因而：

$$P\left(d_i||w_{1i},w_{2i},\beta\right)=\left[F\left(\beta_0+\beta_1 w_{1i}+\beta_2 w_{2i}\right)\right]^{d_i}\left[1-F\left(\beta_0+\beta_1 w_{1i}+\beta_2 w_{2i}\right)\right]^{1-d_i}$$

现在任意给定一个先验分布，比如：$\beta_j\sim N\left(0,1\right), j=0,1,2$，那么 β 的后验分布为：

$$\pi\left(\beta|d,w_1,w_2\right)\propto\prod_{i=1}^{N}P\left(d_i|w_{1i},w_{2i},\beta\right)\cdot\pi_0\left(\beta_0\right)\pi_1\left(\beta_1\right)\pi_2\left(\beta_2\right)$$

由于我们很难得到上述后验分布的函数形式，因而很难计算后验均值、后验中位数等特征，因而我们转而使用 MCMC 算法对其后验分布 $\pi\left(\beta|d,w_1,w_2\right)$ 进行采样。我们使用随机游走的 Metropolis-Hastings 算法，取 $\epsilon_{j,t}\sim N\left(0,0.1^2\right), j=0,1,2$，代码如下：

代码 5: Logistic 回归的 Metropolis-Hastings 算法示例

```
#!/usr/bin/python3
## file: MonteCarlo.py

import numpy as np
from numpy import random as nprd

##设定参数
M =20000
#真值
```

```
beta_0=1
beta_1=1
beta_2=-1
#样本量
N=200
#Logistic函数
Logistic=lambda x: 1.0/(1+np.exp(-1*x))

##产生数据
def gen_logit(N):
    Data=[]
    for n in range(N):
        x1=nprd.normal()*1.414+1
        x2=nprd.chisquare(2)
        d_star=beta_0+beta_1*x1+beta_2*x2
        p_star=Logistic(d_star)
        d=(1 if nprd.uniform()<p_star else 0)
        Data.append((d,x1,x2))
    return Data

#计算接受率
def rho(beta_x,beta_y,Data):
    log_post_pai_x=(-1*beta_x[0]**2-beta_x[1]**2-beta_x[2]**2)/2
    log_post_pai_y=(-1*beta_y[0]**2-beta_y[1]**2-beta_y[2]**2)/2
    log_ratio=log_post_pai_y-log_post_pai_x
    for data in Data:
        w_b_x=beta_x[0]+data[1]*beta_x[1]+data[2]*beta_x[2]
        w_b_y=beta_y[0]+data[1]*beta_y[1]+data[2]*beta_y[2]
        F_b_x=Logistic(w_b_x)
        F_b_y=Logistic(w_b_y)
        log_post_pai_x=np.log((F_b_x if data[0]==1 else 1-F_b_x))
        log_post_pai_y=np.log((F_b_y if data[0]==1 else 1-F_b_y))
        log_ratio+=(log_post_pai_y-log_post_pai_x)
    return min(1,np.exp(log_ratio))

#随机游走
def q_sampler(beta):
    return [b+nprd.normal(0,0.1) for b in beta]

##独立的MCMC算法，输入：
```

```
##      N_samples   :  抽样次数
##         rho(x,y,Data)     :  计算接受率
##     q_sampler(x):  给定x, 从q中抽样的函数
##         x0            :  初始值
##        data           :  数据
def MH_RW(N_samples, rho, q_sampler, x0, data):
    X=[]
    x=x0
    for i in range(N_samples):
        y=q_sampler(x)
        if nprd.uniform()<=rho(x,y,data):
            X.append(y)
            x=y
        else:
            X.append(x)
    return X

## 从后验抽样:
data=gen_logit(N)
beta_post=MH_RW(M, rho, q_sampler, [0,0,0], data)
beta0_post=[b[0] for b in beta_post]
sub_beta0=beta0_post[int(M*0.2):]
beta1_post=[b[1] for b in beta_post]
sub_beta1=beta1_post[int(M*0.2):]
beta2_post=[b[2] for b in beta_post]
sub_beta2=beta2_post[int(M*0.2):]
#后验均值
mean_beta0=np.mean(sub_beta0)
mean_beta1=np.mean(sub_beta1)
mean_beta2=np.mean(sub_beta2)
print("Mean beta0=",mean_beta0)
print("Mean beta1=",mean_beta1)
print("Mean beta2=",mean_beta2)
```

3.4 MCMC 算法的诊断

在使用 MCMC 算法得到对目标分布的抽样之后，一个很自然的问题时该抽样对目标分布 $\pi(\cdot)$ 的收敛性。为此我们需要对算法的收敛性进行诊断。

首先，为了检查马尔可夫链 $\{x_t, t=1,...,M\}$ 是否长时间停留在某一个点，也可以将该序列的时序图画出，观察其是否长时间不转移。理想的抽样应该具

有比较高的接受率，从而不会出现长时间不转移的情况。

其次，我们可以把采样的结果 $\{x_t, t = 1, ..., M\}$ 的直方图画出，观察该分布是否与目标分布想吻合。特别的，对于一个收敛良好的马尔可夫链，如果后验分布为连续分布，那么直方图应该也较为连续；否则，如果马尔可夫链长时间停留在某一点处，即拒绝率很高，那么直方图可能会出现不连续的情况。

为了检查积分：

$$\int h(x)\pi(x)\,dx = \mathbb{E}_\pi[h(x)]$$

的收敛性，我们也可以将序列：

$$m_t = \frac{1}{t}\sum_{\tau=1}^{t} h(x_t)$$

即前 t 个抽样的平均值，将其时序图画出，如果该积分是收敛的，应该可以观察到随着抽样次数的增加，m_t 应该时收敛的。

最后，如果所产生的马尔可夫链是遍历的，那么所计算的积分的方差：

$$\begin{aligned}\mathrm{Var}\left(\frac{1}{M}\sum_{t=1}^{M} h(x_t)\right) &= \frac{\sigma^2}{M}\left[1 + 2\sum_{\tau=1}^{m-1}\left(1 - \frac{\tau}{m}\right)\rho_\tau\right] \\ &\approx \frac{\sigma^2}{M}\left[1 + 2\sum_{\tau=1}^{\infty}\rho_\tau\right]\end{aligned}$$

其中 $\sigma^2 = \mathrm{Var}[h(x_t)]$，而

$$\rho_j = \mathrm{Corr}(h(x_t), h(x_{t+j}))$$

为自相关函数。注意到，如果 x_t 是独立同分布的，那么方差应为：σ^2/M，因而由于 MCMC 算法得到的抽样并非独立抽样，其方差扩大了 $[1 + 2\sum_{\tau=1}^{\infty}\rho_\tau]$ 倍。我们一般将

$$M^* = \frac{M}{1 + 2\sum_{\tau=1}^{\infty}\rho_\tau}$$

称为有效样本量。从而，我们可以通过画出自相关函数图，观察自相关收敛到 0 的速度，如果收敛到 0 的速度比较快，那么方差扩大的倍数也比较小，得到的积分更加可靠。

习题

练习 1. 令 $x \in [0, 1]$，$h(x) = [\cos(50x) + \sin(20x)]^2$，使用蒙特卡洛方法计算其数值积分：

$$\int_{[0,1]} h(x)\,dx$$

练习 2. 证明例 (2) 中，如果不限制积分区域，有:

$$\operatorname{Var}\left(\frac{1}{|h_1(x_i, y_i)|}\right) = \infty$$

练习 3. 证明式 (2) 也满足 $\pi(x) K(x, y) = \pi(y) K(y, x)$。

练习 4. 写程序使用 MCMC 算法从 Beta 分布中抽样，并画出直方图。

练习 5. 如果 $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$，且 $\epsilon \sim N(0, \sigma^2)$，适当设定模型，并使用 MCMC 算法给出所有参数的后验抽样，并画出直方图。

参考文献

[1] Casella, G., Berger, R.L., 2002. Statistical inference. Duxbury Pacific Grove, CA.

[2] Shao, J., 2007. Mathematical Statistics, 2nd ed. Springer, New York.