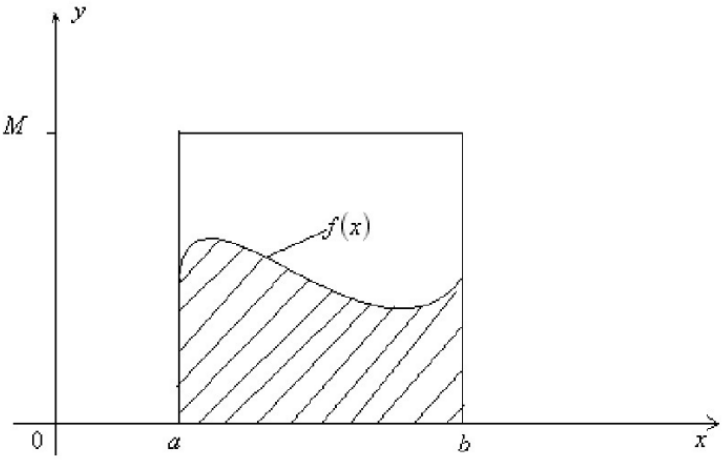
MCMC（Markov Chain Monte Carlo）

回顾:

1、Monte Carlo



（1）概率分布采样

如果我们很难求解出 f(x)的原函数，那么这个积分比较难求解。

我们可以采样[a,b]区间的 n 个值：x0,x1,...xn−1,用它们的均值来代表[a,b]区间上所有的 f(x)的值。这样我们上面的定积分的近似求解为:

$$\frac{b-a}{n}\sum_{i=0}^{n-1} f(x_i)$$

（分布均匀）

（2）分布不均问题

绝大部分情况，x 在[a,b]之间不是均匀分布的

Example：

我们通过 x 在[a,b]的概率分布函数 p(x)，可以求出Θ:

$$\theta=\int_a^b f(x)dx=\int_a^b \frac{f(x)}{p(x)}p(x)dx \approx \frac{1}{n}\sum_{i=0}^{n-1}\frac{f(x_i)}{p(x_i)}$$

上式最右边的这个形式就是蒙特卡罗方法的一般形式。

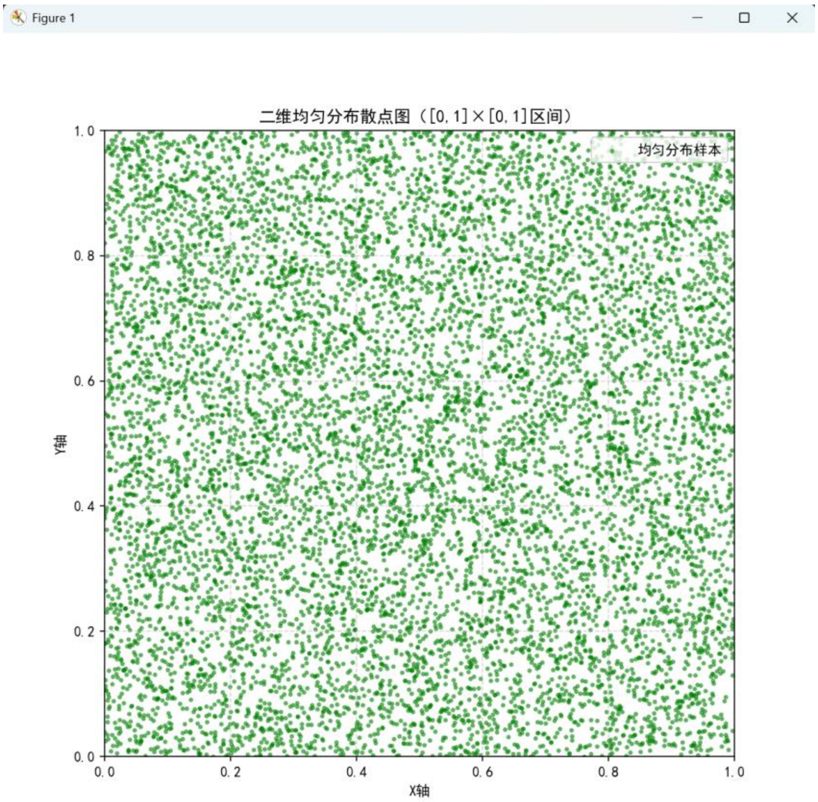
（3）概率分布采样及接受-拒绝采样

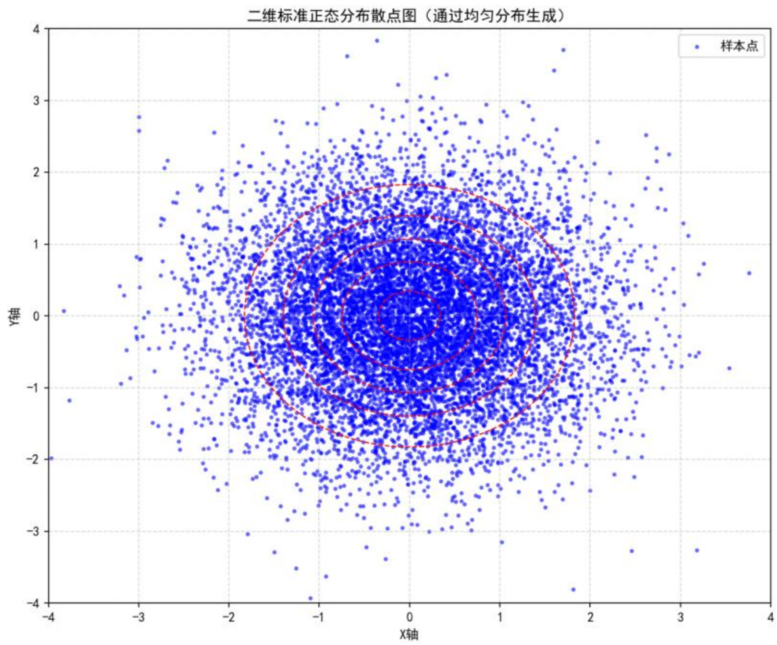
我们知道，如果求出了 x 的概率分布，我们可以基于概率分布去采样 n 个 x 的样本集。那么如何基于概率分布去采样这 n 个 x 的样本集？

常见的概率分布，比如 t 分布，F 分布，Beta 分布，Gamma 分布等，无论是离散的分布还是连续的分布，它们的样本都可以通过 uniform(0,1)先生成均匀随机样本，再通过对应公式转换而得。

Example：基于正态分布采样 10000 个 x 的样本集

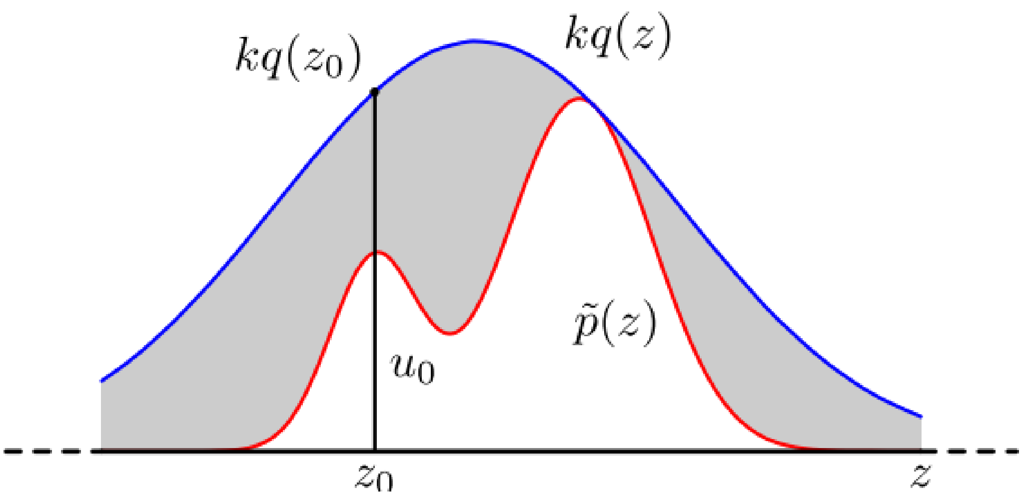
```python
import numpy as np
import matplotlib.pyplot as plt

# 设置随机种子（保证可重复性）
np.random.seed(42)

# 生成n=10000个Uniform(0,1)的样本
n = 10000
u1 = np.random.uniform(0, 1, n)
u2 = np.random.uniform(0, 1, n)

# Box-Muller变换：将均匀分布转换为标准正态分布
z0 = np.sqrt(-2 * np.log(u1)) * np.cos(2 * np.pi * u2)
z1 = np.sqrt(-2 * np.log(u1)) * np.sin(2 * np.pi * u2)

# 合并为二维正态分布样本（均值为0，协方差矩阵为单位矩阵）
samples = np.column_stack((z0, z1))

plt.figure(figsize=(10, 8))
# 设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

uniform_samples = np.column_stack((u1, u2))  # 二维均匀分布样本
# 绘制散点图
plt.scatter(samples[:, 0], samples[:, 1], alpha=0.5, s=5, color='blue', label='样本点')

# 添加理论等高线（红色虚线）
x = np.linspace(-4, 4, 100)
y = np.linspace(-4, 4, 100)
X, Y = np.meshgrid(x, y)
Z = (1 / (2 * np.pi)) * np.exp(-0.5 * (X**2 + Y**2))  # 二维标准正态分布密度函数
plt.contour(X, Y, Z, levels=5, colors='red', linestyles='dashed', linewidths=1, label='理论等高线')

# 美化图表
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
plt.title('二维标准正态分布散点图（通过均匀分布生成）')
plt.xlabel('X轴')
plt.ylabel('Y轴')
plt.xlim(-4, 4)
plt.ylim(-4, 4)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.show()

# 绘制二维均匀分布散点图
plt.figure(figsize=(8, 8))
plt.scatter(uniform_samples[:, 0], uniform_samples[:, 1], alpha=0.5, s=5, color='green', label='均匀分布样本')
plt.title('二维均匀分布散点图（[0,1]×[0,1]区间）')
plt.xlabel('X轴'); plt.ylabel('Y轴')
plt.xlim(0, 1); plt.ylim(0, 1)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.show()
```





不过很多时候，我们的 x 的概率分布不是常见的分布，这意味着我们没法方便的得到这些非常见的概率分布的样本集。这时我们引入接受-拒绝采样：

设定一个方便采样的常用概率分布函数 q(x)，以及一个常量 k，使得 p(x)总在 kq(x)的下方。



整个过程中，我们通过一系列的接受拒绝决策来达到用 q(x)模拟 p(x)概率分布的目的。

但是此方法在高维当中难以实现，所以学要借助马尔科夫链来解决。

2、Markov Chain

（1）定义：假设某一时刻状态转移的概率只依赖于它的前一个状态。

Example：

共有三种状态：A, B 和 C。每一个状态都以一定的概率转化到下一个状态。

这样我们得到了马尔科夫链模型的状态转移矩阵为:

通过矩阵相乘的方法，可以得到某初始状态运动 n 次后状态的概率

（2）**状态转移矩阵的收敛性**

假设 A,B,C 的概率分布为[0.3,0.4,0.3],初始状态为 t0，将其带入这个状态转移矩阵计算 t1,t2,t3...的状态:

```
import numpy as np

matrix = np.matrix([[0.9,0.075,0.025],[0.15,0.8,0.05],[0.25,0.25,0.5]], dtype=float)

vector1 = np.matrix([[0.3,0.4,0.3]], dtype=float)

for i in range(100):

    vector1 = vector1*matrix

    print "Current round:" , i+1

    print vector1
```

部分输出如下:

```
Current round: 1
[[ 0.695    0.1825   0.1225]]
Current round: 2
[[ 0.6835    0.22875   0.08775]]
Current round: 3
[[ 0.6714   0.2562   0.0724]]
Current round: 4
[[ 0.66079    0.273415    0.065795]]
。。。。。。。
Current round: 55
[[ 0.62500001   0.31249999   0.0625     ]]
Current round: 56
[[ 0.62500001   0.31249999   0.0625     ]]
Current round: 57
[[ 0.625    0.3125   0.0625]]
。。。。。。。
Current round: 99
[[ 0.625    0.3125   0.0625]]
Current round: 100
[[ 0.625    0.3125   0.0625]]
```

后面经过测试发现，尽管采用了不同初始概率分布，最终状态的概率分布趋于同一个稳定的概率分布[0.625   0.3125   0.0625]，也就是说马尔科夫链模型的状态转移矩阵收敛到的稳定概率分布与我们的初始状态概率分布无关。

（3）周期和非周期马尔科夫链

周期马尔科夫链：

如果存在一个正整数 d>1，使得从任意状态 i 出发，只有经过 d 的倍数步才能返回到状态 i，那么称状态 i 是周期为 d 的。

- **特点**：
  - 周期性意味着返回到某个状态的步数总是某个固定数的倍数。
  - 周期马尔科夫链可能不会收敛到一个唯一的平稳分布，或者可能根本不收敛。

非周期马尔科夫链:

如果马尔科夫链的所有状态都不是周期的，即从任意状态出发，可以在任意步数后返回到该状态，那么称这个马尔科夫链是非周期的。

- **特点**:
    - 非周期性意味着返回到某个状态的步数没有固定的周期限制。
    - 非周期马尔科夫链更容易分析，因为它们通常具有更好的收敛性质。

（4）马尔科夫链的细致平稳条件

非周期马尔科夫链的状态转移矩阵 P 和概率分布 π(x)对于所有的 i,j 满足:

$$\pi(i)P(i,j) = \pi(j)P(j,i)$$

对于任意两个状态 i 和 j，从状态 i 转移到状态 j 的概率流必须等于从状态 j 转移到状态 i 的概率流。

如果假定我们可以得到我们需要采样样本的平稳分布所对应的马尔科夫链状态转移矩阵，那么我们就可以用马尔科夫链采样得到我们需要的样本集，进而进行蒙特卡罗模拟。

MCMC 及 M-H 采样

在实际问题中，我们很难直接找到对应的马尔科夫链状态转移矩阵 P，不是所有的马尔科夫链都有平稳分布，只有满足细致平稳条件才行。我们希望从目标分布 π(x) 中采样，但直接从这个分布中采样可能很困难。因此，我们使用一个易于采样的提议分布 Q 来辅助采样（类似蒙特卡罗）。然而，直接使用 Q 可能不满足细致平衡条件，即：

$$\pi(i)Q(i,j) \neq \pi(j)Q(j,i)$$

MCMC 初步：

我们可以对上式做一个改造，使细致平稳条件成立。方法是引入一个 α(i,j)，使上式可以取等号：

$$\pi(i)Q(i,j)\alpha(i,j) = \pi(j)Q(j,i)\alpha(j,i)$$

α(i,j) 满足下两式：

$$\alpha(i,j) = \pi(j)Q(j,i)$$
$$\alpha(j,i) = \pi(i)Q(i,j)$$

通过引入 α(i,j)，我们可以构造一个新的状态转移矩阵 P，它满足细致平衡条件：

$$P(i,j) = Q(i,j)\alpha(i,j)$$

α(i,j)一般称之为接受率，取值在[0,1]之间。

由于 α(xt,x*)可能非常的小，比如 0.1，导致我们大部分的采样值都被拒绝转移，采样效率很低，计算量大，因此引入 M-H 采样。

M-H 采样：

这个算法首先由 Metropolis 提出，被 Hastings 改进，因此被称之为 Metropolis-Hastings 采样或 M-H 采样。

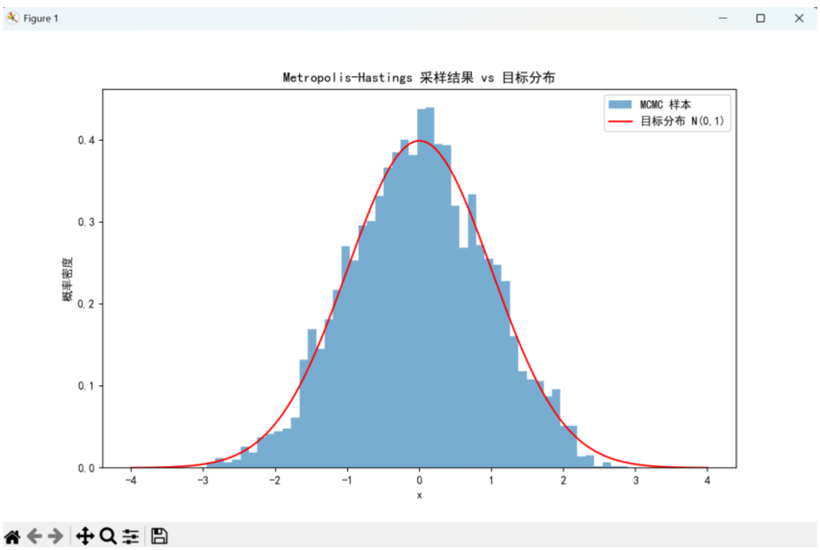
由于 α(i,j)太小了，比如为 0.1，而 α(j,i)为 0.2。即：

$$\pi(i)Q(i,j) \times 0.1 = \pi(j)Q(j,i) \times 0.2$$

如果两边同时扩大 5 倍，细致平稳条件仍然满足，这样我们的接受率可以做如下改进，即：

$$\alpha(i,j) = min\{\frac{\pi(j)Q(j,i)}{\pi(i)Q(i,j)}, 1\}$$

（使用 min 函数取这个比率和 1 中的较小值。这是因为接受概率 α(i,j) 不能大于 1，否则会有超过 100% 的概率接受候选样本）

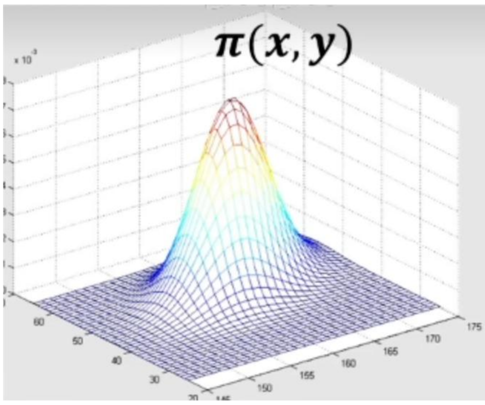


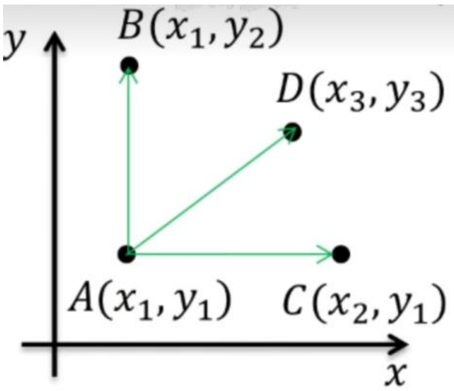
```python
import numpy as np
import matplotlib.pyplot as plt

# 目标分布：标准正态分布
def target_dist(x):
    return np.exp(-x**2 / 2) / np.sqrt(2 * np.pi)

# 提议分布 Q：生成候选样本（对称的）
def proposal_distribution(x_t, sigma=1):
    return np.random.normal(x_t, sigma)

# Metropolis-Hastings 算法
def metropolis_hastings(n1, n2):
    samples = []
    current_x = np.random.uniform(-5, 5)  # 初始状态

    for _ in range(n1 + n2):
        # 从提议分布生成候选样本
        x_s = proposal_distribution(current_x)

        # 计算接受概率
        alpha = min(1, target_dist(x_s) / target_dist(current_x))  # Q 对称时简化为 π(x_s)/π(x_t)

        # 接受或拒绝转移
        u = np.random.uniform(0, 1)
        if u < alpha:
            current_x = x_s
        samples.append(current_x)

    return samples[n1:]  # 去除 burn-in 期

# 参数设置
n1 = 1000  # burn-in 样本数
n2 = 5000  # 实际采样数
samples = metropolis_hastings(n1, n2)

# 设置中文字体（在绘制图表前配置）
plt.rcParams['font.sans-serif'] = ['SimHei']  # Windows系统中文支持
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题
# 可视化
plt.figure(figsize=(10, 6))
plt.hist(samples, bins=50, density=True, alpha=0.6, label="MCMC 样本")
x_range = np.linspace(-4, 4, 1000)
plt.plot(x_range, target_dist(x_range), 'r', label="目标分布 N(0,1)")
plt.title("Metropolis-Hastings 采样结果 vs 目标分布")
plt.xlabel("x")
plt.ylabel("概率密度")
plt.legend()
plt.show()
```

但是 M-H 采样有两个缺点：一是需要计算接受率，在高维时计算量大。因此引入 Gibbs 采样。

Gibbs 采样

在 M-H 采样中我们通过引入接受率使细致平稳条件满足。现在我们换一个思路。





从二维的数据分布开始，假设 π(x,y)是一个二维联合数据分布，观察第一个特征维度相同的两个点 A,B，容易发现下面两式成立：

$$\pi(A)=\pi(x_1,y_1)=\pi(x_1)\pi(y_1|x_1)$$
$$\pi(B)=\pi(x_1,y_2)=\pi(x_1)\pi(y_2|x_1)$$

再做个变换：

$$\pi(A)\pi(y_2|x_1)=\boxed{\pi(x_1)\pi(y_1|x_1)\pi(y_2|x_1)}$$
$$\pi(B)\pi(y_1|x_1)=\boxed{\pi(x_1)\pi(y_2|x_1)\pi(y_1|x_1)}$$

我们发现：

$$\pi(A)\boxed{\pi(y_2|x_1)}=\pi(B)\pi(y_1|x_1)$$

类似于细致平稳条件：

$$\pi(i)P(i,j)=\pi(j)P(j,i)$$

这样我们就得到状态转移概率：

$$\pi(A)P(A \to B) = \pi(B)P(B \to A)$$

如果用条件概率分布作为马尔科夫链的状态转移概率，则任意两个点之间的转移满足细致平稳条件。

基于上面的发现，我们可以这样构造分布 π(x,y)的马尔可夫链对应的状态转移矩阵 P：

$$P(A \to B) = \pi(y_2|x_1)$$

$$P(A \to C) = \pi(y_1|x_2)$$

$$P(A \to D) = 0$$

（只允许在平行坐标轴方向上采样）