自然语言处理（2）主题模型 LDA （1 数学基础篇）

随风
大数据、人工智能

关注他

3 人赞同了该文章

1 前言

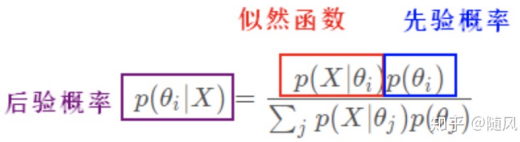
要想彻底搞明白 LDA 的实现原理，就需要具备一定的数学基础。LDA 用到的数学知识包括：

一个公式和两个概念：贝叶斯公式、共轭分布和共轭先验
一个函数：gamma 函数
四个分布：二项分布、多项分布、beta 分布、dirichlet 分布
一个采样：gibbs 采样

下面将分别对这几个部分作介绍。

2 一个公式和两个概念

先来看一下贝叶斯公式： θ_i 表示未知参数，X 表示样本



先验概率：在事件尚未发生前，对该事件发生概率的估计。利用过去历史资料计算出来得到的先验概率叫做客观先验概率；凭主观经验来判断而得到的先验概率叫做主观先验概率。
后验概率：通过调查或其它方式获取新的附加信息，利用贝叶斯公式对先验概率进行修正后而得到的概率。
似然函数：给定模型参数 θ 的条件下，样本数据服从这一概率模型的相似程度。
先验分布：反映在进行统计试验之前根据其他有关参数知识得到的分布。也就是说在观测获取样本之前，人们对 θ 已经有一些知识，此时这个 θ 的分布函数为 $H(\theta)$ ， θ 的概率密度函数为 $h(\theta)$ ，分别称为先验分布函数和先验概率密度函数，统称先验分布。
后验分布：根据样本 X 的分布以及 θ 的先验分布 $\pi(\theta)$ ，采用求解条件概率的方式可以计算出已知 X 的条件下， θ 的条件分布 $\pi(\theta|X)$ 。因为该分布是在获取样本 X 之后计算出来的，所以称为后验分布。
共轭分布和共轭先验：在贝叶斯概率理论中，如果后验概率 $p(\theta \mid x)$ 和先验概率 $p(\theta)$ 满足同样的分布律，那么，先验分布和后验分布被叫做**共轭分布**。同时，先验分布叫做似然函数的**共轭先验分布**。

3 一个函数

gamma 函数定义如下： $\Gamma(x)=\int_0^{+\infty}t^{x-1}e^{-t}dt$ ，gamma 函数可以看作阶乘在实数集上的拓展，对于正整数 n，具有如下性质： $\Gamma(n)=(n-1)!$ 。

4 四个分布

4.1 二项分布

二项分布是从伯努利分布推导出来的。伯努利分布，又称两点分布或0-1分布，是一个离散型的随机分布，其中的随机变量只有两种取值，非正即负{+,−}。而二项分布即重复 n 次伯努利试验，记为 X∽B(n,p) 。简言之，只做一次实验，是伯努利分布，重复做了 n 次，是二项分布。二项分布的概率密度函数为： $P(k)=C_n^kP^k(1-P)^{n-k},k=0,1,2,\dots,n$ ，其中 $C_n^k=\frac{n!}{k!(n-k)!}$ ，是二项分布的系数。

一个典型的例子就是抛硬币，我们做 n 次实验，有 k 次为正面的概率。

4.2 多项分布

多项分布是二项分布在多维上的推广，是指单次试验中随机变量的取值不再是 0-1 ，而是有多种离散值。比如掷骰子，有 6 个面，n 次试验结果服从 k=6 的多项分布。其中 k 个离散值的概率满足： $\sum_{i=1}^{k}P_i=1$ 。多项分布的概率密度函数为：

$$P(x_1=m_1,\dots,x_k=m_k;n;p_1,\dots,p_k)=\frac{n!}{m_1!m_2!\dots m_k!}p_1^{m_1}\dots p_k^{m_k}$$

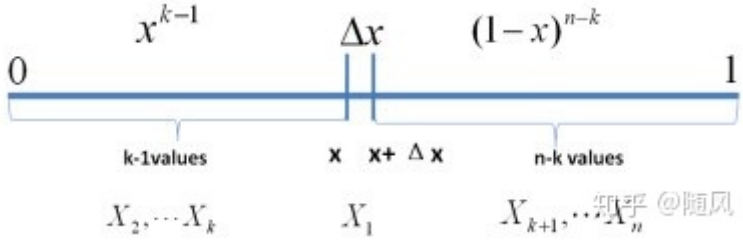
其中 m_i 表示随机变量 x_i 发生的次数， p_i 表示随机变量 x_i 发生的概率。 $\sum_{i=1}^{k}m_i=n$ ，表示所有随机变量发生次数的总和为 n。

4.3 beta 分布

用一句话来说，beta 分布可以看作是一个概率的概率分布。当你不知道一个事件的具体概率是多少时，它可以给出所有概率出现的可能性大小。

假设随机变量 $X_1, X_2, \ldots, X_n$ 服从 0~1 上的均匀分布，把这 n 个随机变量排序后得到顺序统计量 $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$ ，请问 $X_{(k)}$ 的分布是什么？为了解决这个问题，可以尝试计算 $X_{(k)}$ 落在区间 $[x, x+\Delta x]$ 的概率： $P(x \le X_{(k)} \le x+\Delta x)$ 。

首先把 $[0,1]$ 区间分成三段 $[0,x), [x,x+\Delta x], (x+\Delta x, 1]$，然后考虑一下简单的情形：假设 n 个数中只有 1 个落在了区间 [x, x+Δx] 内，由于这个区间内的数 $X_{(k)}$ 是第 k 大的，所以 $[0,x)$ 中应该有 k - 1 个数， $(x+\Delta x, 1]$ 这个区间中应该有 n - k 个数。如下图所示：



从而问题转换为下述事件 E：

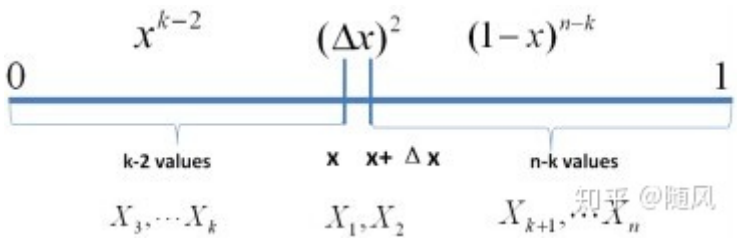
$$
\begin{aligned}
E = \{\ & X_1 \in [x, x+\Delta x], \\
& X_i \in [0, x) \quad (i = 2, \cdots, k), \\
& X_j \in (x+\Delta x, 1] \quad (j = k+1, \cdots, n)\}
\end{aligned}
$$

对于上述事件 E 有：

$$
\begin{aligned}
P(E) &= \prod_{i=1}^{n} P(X_i) \\
&= x^{k-1}(1-x-\Delta x)^{n-k}\Delta x \\
&= x^{k-1}(1-x)^{n-k}\Delta x + o(\Delta x)
\end{aligned}
$$

其中， $o(\Delta x)$ 表示 Δx 的高阶无穷小。显然，由于不同的组合，即 n 个数中有一个落在 $[x, x+\Delta x]$ 区间的有 n 种取法，余下 n - 1 个数中有 k - 1 个落在 $[0,x)$ 的有 C_{n-1}^{k-1} 种组合，所以和事件 E 等价的事件一共有 nC_{n-1}^{k-1} 个。

如果有 2 个数落在区间 $[x, x+\Delta x]$ 呢？如下图所示：



类似于事件 E，对于 2 个数落在区间 $[x, x+\Delta x]$ 的事件 E'：

$$E' = \{\ X_1, X_2 \in [x, x+\Delta x],\\ X_i \in [0, x) \quad (i = 3, \cdots, k),\\ X_j \in (x+\Delta x, 1] \quad (j = k+1, \cdots, n)\}$$

有：

$$P(E') = x^{k-2}(1-x-\Delta x)^{n-k}(\Delta x)^2 = o(\Delta x)$$

从上述的事件E、E' 中，可以看出，只要落在 $[x, x+\Delta x]$ 内的数字超过一个，对应事件的概率就是 $o(\Delta x)$。于是有：

$$\begin{aligned} &P(x \le X_{(k)} \le x+\Delta x)\\ &= n\binom{n-1}{k-1}P(E) + o(\Delta x)\\ &= n\binom{n-1}{k-1}x^{k-1}(1-x)^{n-k}\Delta x + o(\Delta x) \end{aligned}$$

从而得到概率密度函数：

$$\begin{aligned} f(x) &= \lim_{\Delta x \to 0}\frac{P(x \le X_{(k)} \le x+\Delta x)}{\Delta x}\\ &= n\binom{n-1}{k-1}x^{k-1}(1-x)^{n-k}\\ &= \frac{n!}{(k-1)!(n-k)!}x^{k-1}(1-x)^{n-k} \quad x \in [0,1] \end{aligned}$$

考虑 gamma 函数的性质： $\Gamma(n) = (n-1)!$ ，取 $a = k, b = n-k+1$ ，得到：

$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1}$ ，令 $\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} = \frac{1}{B(a,b)}$ ，得 beta 分布：

$f(x) = \frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1}$ 。

我们在来看看 Beta 分布的期望：

$$E(Beta(\alpha,\beta)) = \int_0^1 xf(x)dx = \int_0^1 x\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}dx = \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha}(1-x)^{\beta-1}dx$$

由于 Beta 分布的积分值为 1，所以：

$\int_0^1 Beta(\alpha+1,\beta) = \int_0^1 \frac{\Gamma(\alpha+\beta+1)}{\Gamma(\alpha+1)\Gamma(\beta)}x^{\alpha}(1-x)^{\beta-1}dx = 1$ ，由此可得：

$$E(Beta(\alpha,\beta)) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)} = \frac{\alpha}{\alpha+\beta}$$

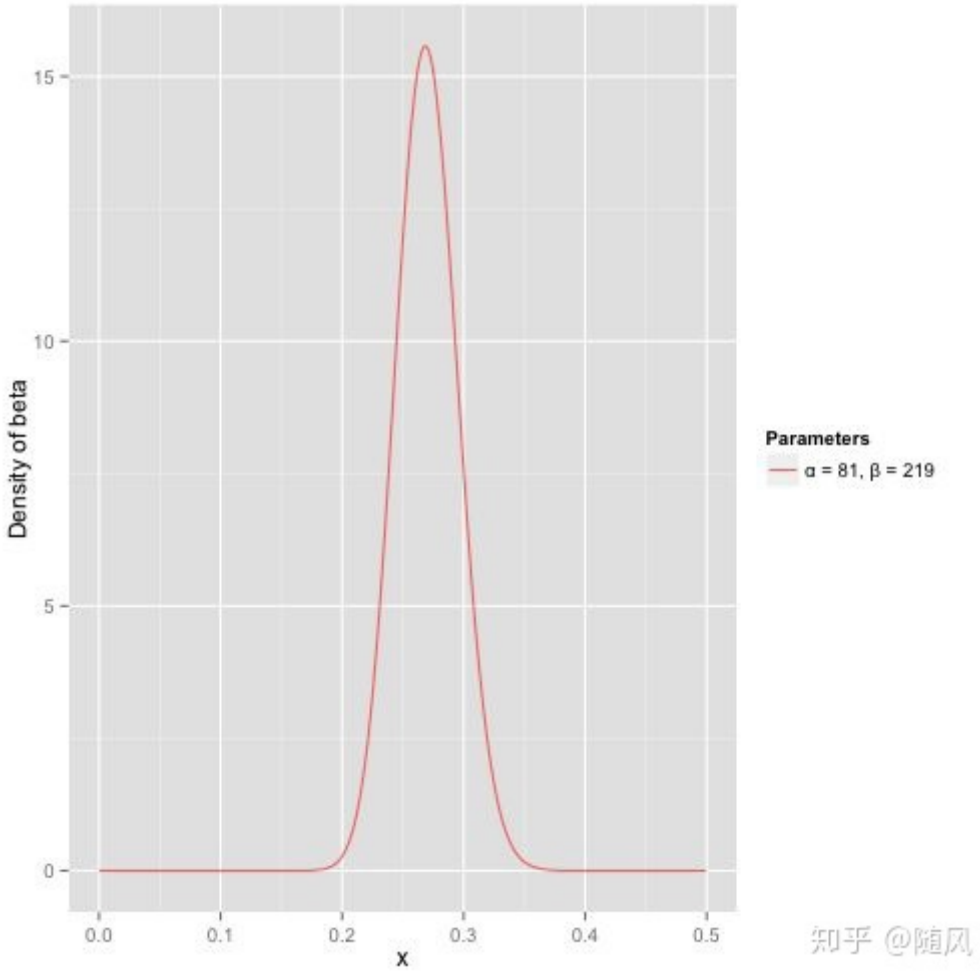
那么 beta 分布有什么用处呢?

我们举个例子，熟悉棒球运动的都知道有一个指标就是棒球击球率，就是用一个运动员击中的球数除以击球的总数，我们一般认为 0.266 是正常水平的击球率，而如果击球率高达 0.3 就被认为是非常优秀的。

现在有一个棒球运动员，我们希望能够预测他在这一赛季中的棒球击球率是多少。你可能就会直接计算棒球击球率，用击中的数除以击球数，但是如果这个棒球运动员只打了一次，而且还命中了，那么他就击球率就是 100%，这显然是不合理的，因为根据棒球的历史信息，我们知道这个击球率应该是 0.215 到 0.36 之间。

对于这个问题，我们可以用一个二项分布表示（一系列成功或失败），一个最好的方法来表示这些经验（在统计中称为先验信息）就是用 Beta分布 ，这表示在我们没有看到这个运动员打球之前，我们就有了一个大概的范围。

beta 分布是指一组定义在 (0,1) 区间的连续概率分布，具有两个参数：a，b>0 。接下来我们将这些先验信息转换为 beta 分布的参数，我们知道一个击球率应该是平均 0.27 左右，而他的范围是 0.21 到 0.35，那么根据这个信息，我们可以取 a=81，b=219。



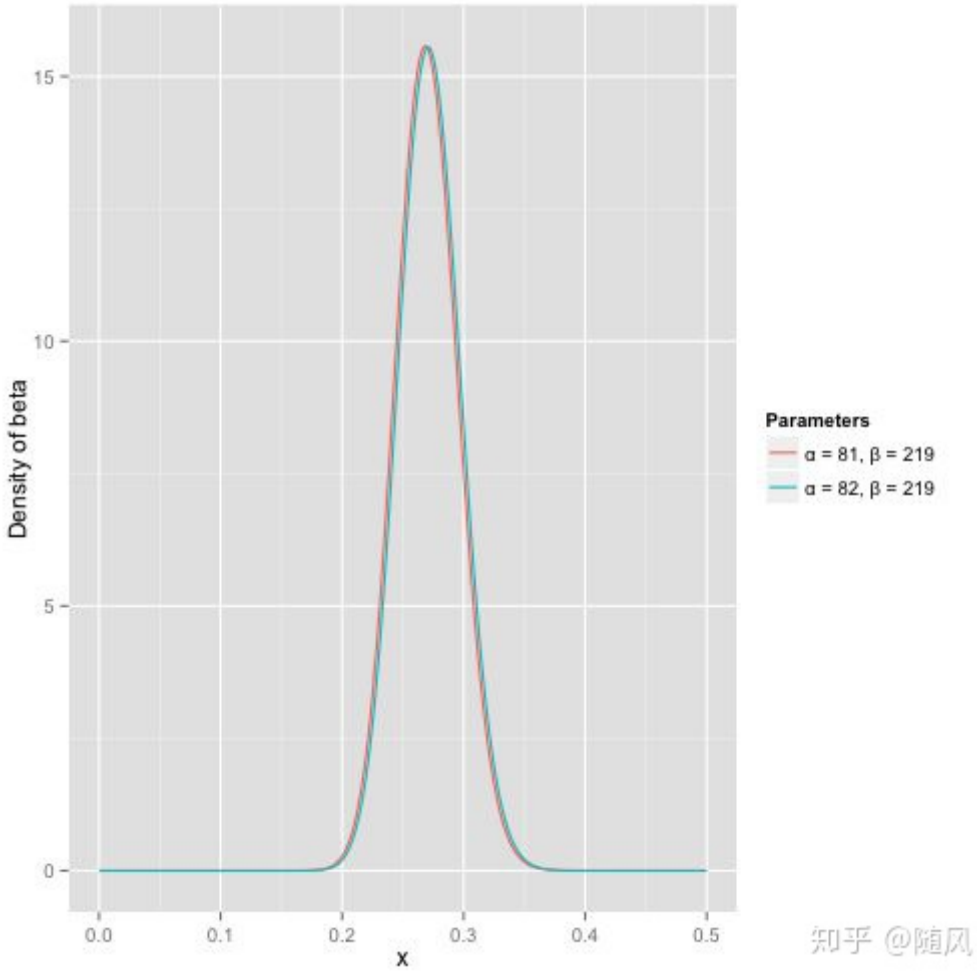
之所以取这两个参数是因为：

beta 分布的均值是 $\frac{a}{a+b}=0.27$

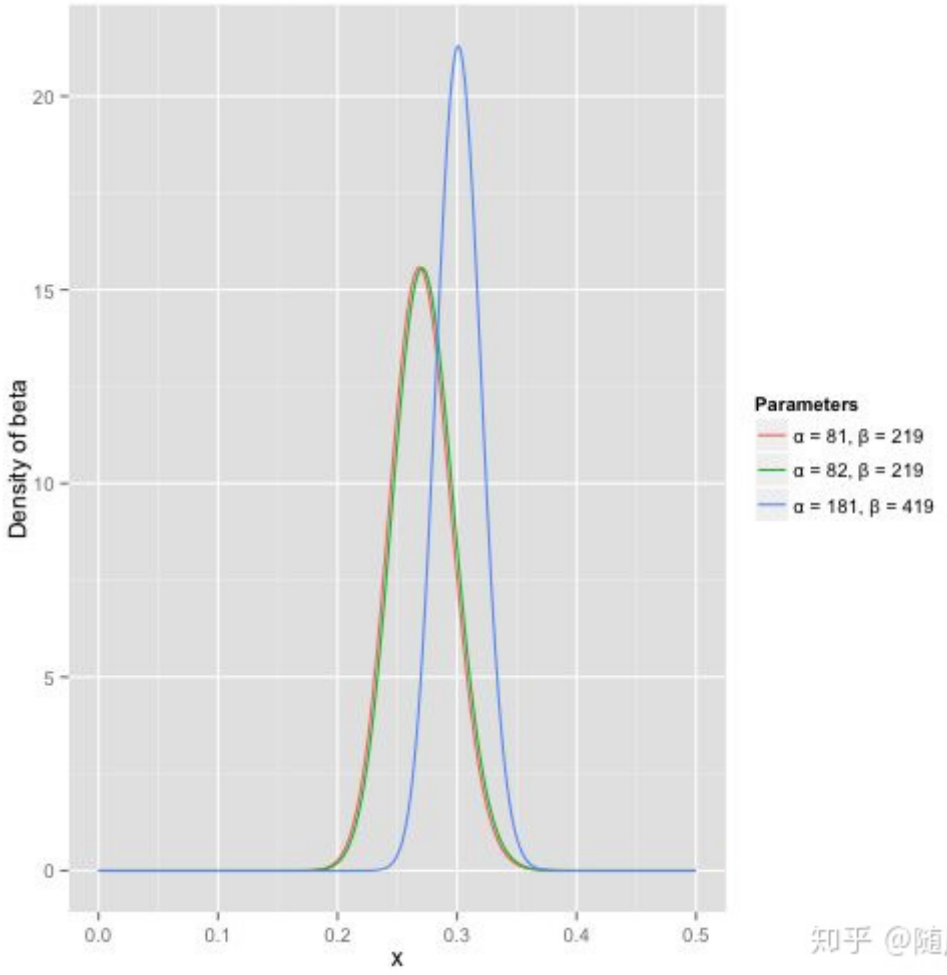
从图中可以看到这个分布主要落在了 $(0.2, 0.35)$ 间，这是从经验中得出的合理的范围。

其中，x 轴就表示各个击球率的取值，y 值就是这个击球率所对应的概率。也就是说 beta 分布可以看作是一个概率的概率分布。

那么有了先验信息后，现在我们考虑一个运动员只打一次球，那么他现在的数据就是"1 中；1 击"。这时候我们就可以更新我们的分布了，让这个曲线做一些移动去适应我们的新信息。beta 分布在数学上就给我们提供了这一性质，他与二项分布是共轭先验的。所谓共轭先验就是先验分布是 beta 分布 ，而后验分布同样是 beta 分布。在这里 a=81，b=219。所以，a增加了 1（击中了一次）。b 没有增加（没有漏球）。这就是我们的新的 beta 分布beta(81+1，219)，我们跟原来的比较一下：



可以看到这个分布其实没多大变化，这是因为只打了 1 次球并不能说明什么问题。但是如果我们得到了更多的数据，假设一共打了 300 次，其中击中了 100 次，200 次没击中，那么这一新分布就是：beta(81+100，219+200)。



注意到这个曲线变得更加尖，并且平移到了一个右边的位置，表示比平均水平要高。一个有趣的事情是，根据这个新的 beta 分布，我们可以得出他的数学期望为：0.3，这一结果要比直接的估计要小 。你可能已经意识到，我们事实上就是在这个运动员在击球之前可以理解为他已经成功了 81 次，失败了 219 次这样一个先验信息。

因此，对于一个我们不知道概率是什么，而又有一些合理的猜测时，beta 分布能很好的作为一个表示概率的概率分布。

下面来解释一下二项分布与 beta 分布的关系。看下面这个问题：

随机变量 $X \sim U(0,1)$ ，把这 n 个随机变量排序后得到顺序统计量 $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$ ，需要猜测 $X_{(k)}$ 的分布。
观测数据 $Y \sim U(0,1)$ ， Y 中有 m_1 个比 $X_{(k)}$ 小，有 m_2 个比 $X_{(k)}$ 大。
请问 $P(X_{(k)}|Y)$ 的分布是什么?

根据 Y 中有 m_1 个比 $X_{(k)}$ 小，有 m_2 个比 $X_{(k)}$ 大，可知， $X_{(k)}$ 是 $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m \sim U(0,1)$ 中排序后的第 $k+m_1$ 个数。根据前面的结论这一事件服从 beta 分布，从而可知 $X_{(k)}$ 在观测样本 Y 的条件下的概率密度函数：$beta(X_{(k)}|k+m_1, n-k+1+m_2)$ 。

上述过程描述如下：

为了猜测 $X_{(k)}$ 的分布，在获取观测数据之前，我们对这一事件的认知是：
$f(X_{(k)}) = beta(X_{(k)}|k, n-k+1)$ ，称为变量 $X_{(k)}$ 的先验分布。
为了获取观测数据 Y ，我们做了 m 次伯努利实验（一次实验就是按照 0-1 上的均匀分布随机生成一个数），所以变量 Y 服从二项分布： $Y \sim B(m, X_{(k)})$ 。
得到 $X_{(k)}$ 的后验分布： $f(X_{(k)}|Y) = beta(X_{(k)}, k+m_1, n-k+1+m_2)$

这正是贝叶斯方法的思考过程：先验分布 $\pi(\theta)$ + 样本信息 $x \Rightarrow$ 后验分布 $\pi(\theta|x)$

根据上述，这种观测到的数据符合二项分布，参数的先验分布和后验分布都是 beta 分布的情况，就是 **beta 分布与二项分布共轭**，换言之， **beta 分布是二项分布的共轭先验分布。**这意味着，如果我们为二项分布的参数 $X_{(k)}$ 选取的先验分布是 beta 分布，那么以 $X_{(k)}$ 为参数的二项分布用贝叶斯公式估计得到的后验分布仍然服从 beta 分布。

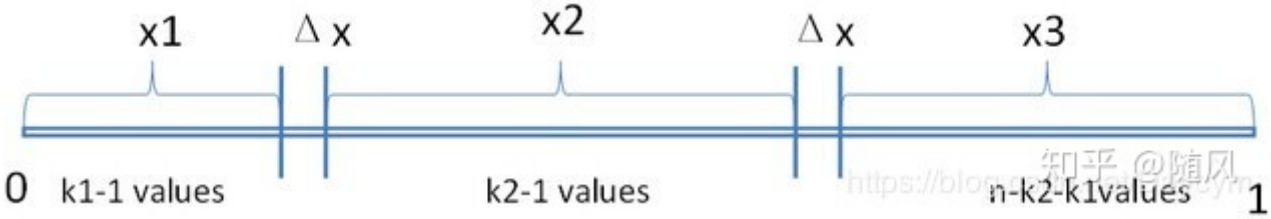
4.4 狄利克雷分布（dirichlet 分布）

dirichlet 分布可以看作是 beta 分布在多维变量上的推广，概率密度函数定义如下：

$$f(x_1, x_2, \ldots, x_k; \alpha_1, \alpha_2, \ldots, \alpha_k) = \frac{1}{B(\alpha)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$$

其中 $B(\alpha) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)}, \sum x_i = 1, \alpha = (\alpha_1, \ldots, \alpha_k)$

假设随机变量 $X_1, X_2, \ldots, X_n$ 服从 0~1 上的均匀分布，把这 n 个随机变量排序后得到顺序统计量 $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$ ，请问 $(X_{(k_1)}, X_{(k_1+k_2)})$ 的联合分布是什么？为了简化计算，取 x_3 满足 $x_1 + x_2 + x_3 = 1$ ，但只有 x_1, x_2 是变量，如下图所示：



从而有：

$$P(X_{(k_1)} \epsilon (x_1, x_1 + \Delta x), X_{(k_1+k_2)} \epsilon (x_2, x_2 + \Delta x))$$
$$= n(n-1) C_{n-2}^{k_1-1} C_{n-k_1-1}^{k_2-1} x_1^{k_1-1} x_2^{k_2-1} x_3^{n-k_1-k_2} (\Delta x)^2$$

$$= \frac{n!}{(k_1-1)!(k_2-1)!(n-k_1-k_2)!} x_1^{k_1-1} x_2^{k_2-1} x_3^{n-k_1-k_2} (\Delta x)^2$$

继而得到 $(X_{(k_1)}, X_{(k_1+k_2)})$ 的联合分布的概率密度函数：

$$f(x_1,x_2,x_3) = lim_{(\Delta x)^2 \to 0} \frac{P(X_{(k_1)} \epsilon (x_1, x_1+\Delta x), X_{(k_1+k_2)} \epsilon (x_2, x_2+\Delta x))}{(\Delta x)^2}$$

$$= \frac{n!}{(k_1-1)!(k_2-1)!(n-k_1-k_2)!} x_1^{k_1-1} x_2^{k_2-1} x_3^{n-k_1-k_2}$$

$$= \frac{\Gamma(n+1)}{\Gamma(k_1)\Gamma(k_2)\Gamma(n-k_1-k_2+1)} x_1^{k_1-1} x_2^{k_2-1} x_3^{n-k_1-k_2}$$

令 $\alpha_1 = k_1, \alpha_2 = k_2, \alpha_3 = n-k_1-k_2+1$ ，得：

$$f(x_1,x_2,x_3) = \frac{\Gamma(\alpha_1+\alpha_2+\alpha_3)}{\Gamma(\alpha_1)\Gamma(\alpha_2)\Gamma(\alpha_3)} x_1^{k_1-1} x_2^{k_2-1} x_3^{n-k_1-k_2}$$

这就是 3 维形式的 dirichlet 分布： $dir(x_1,x_2,x_3|\alpha_1,\alpha_2,\alpha_3)$ 。

对于 Dirichlet 分布的期望，也有和 Beta 分布类似的性质：

$$E(Dirichlet(\bar{\alpha})) = (\frac{\alpha_1}{\sum_{k=1}^K \alpha_k}, \frac{\alpha_2}{\sum_{k=1}^K \alpha_k}, \dots, \frac{\alpha_K}{\sum_{k=1}^K \alpha_k})$$

下面来解释一下多项分布与 dirichlet 分布的关系。看下面这个问题：

随机变量 $X \sim U(0,1)$ ，排序后得到顺序统计量 $X_{(1)}, X_{(2)}, \dots, X_{(n)}$ 。
令 $p_1 = X_{(k_1)}, p_2 = X_{(k_1+k_2)}, p_3 = 1-p_1-p_2$ （这里 p_3 不是变量，只是为了表达方便），现在需要猜测 $\bar{p} = (p_1, p_2, p_3)$ 。
观测数据 $Y \sim U(0,1)$ ， Y 中落到 $[0,p_1), [p_1,p_2), [p_2,1]$ 三个区间的个数分别为 m_1, m_2, m_3 ，且 $m_1+m_2+m_3 = m$ 。
问后验分布 $P(\bar{p}|Y)$ 是什么?

为了方便讨论，我们记 $\bar{m} = (m_1, m_2, m_3), \bar{k} = (k_1, k_2, n-k_1-k_2+1)$ ，根据已知条件， Y 中落到 $[0,p_1), [p_1,p_2), [p_2,1]$ 三个区间的个数分别为 m_1, m_2, m_3 ，可知 p_1, p_2 分别是这 $m+n$ 个数中第 k_1+m_1 大，第 k_2+m_2 大的数，于是后验分布为：

$$P(\bar{p}|Y) = dir(\bar{k}|k_1+m_1, k_2+m_2, n-k_1-k_2+1+m_3)$$

即 $dir(\bar{p}|\bar{k}+\bar{m})$ 。

同样的，按照贝叶斯推理的逻辑，可将上述过程描述如下：

我们要猜测参数 $\bar{p}=(p_1,p_2,p_3)$ ，其先验分布为 $dir(\bar{p}|\bar{k})$
观测数据 Y 中落到 $[0,p_1),[p_1,p_2),[p_2,1]$ 三个区间的个数分别为 m_1,m_2,m_3 ，所以 $\bar{m}=(m_1,m_2,m_3)$ 服从多项分布 $multi(\bar{m}|\bar{p})$
在给定数据 $\bar{m}$ 后， $\bar{p}$ 的后验分布变为： $dir(\bar{p}|\bar{k}+\bar{m})$

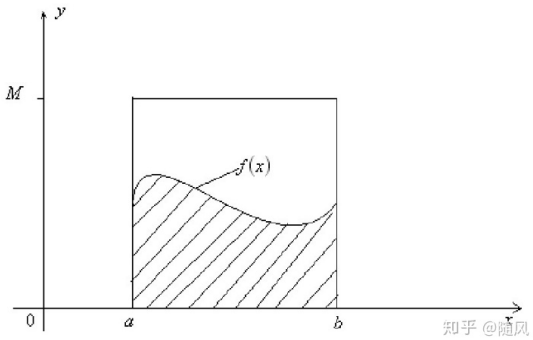
对于这种观测到的数据符合多项分布，**参数的先验分布和后验分布都是 dirichlet 分布的情况，就是 dirichlet 分布-多项分布共轭。换言之，至此已经证明了 dirichlet 分布就是多项分布的共轭先验分布。**

5 一个采样

5.1 蒙特卡罗方法（Monte Carlo method）

最早的蒙特卡罗方法都是为了求解一些不太好解的求和或者积分问题。比如积分：

$\theta=\int_a^b f(x)dx$ ，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。当然我们可以通过蒙特卡罗方法来模拟求解近似值。如何模拟呢？假设我们函数图像如下图：



一个简单的近似求解方法是在 $[a,b]$ 之间随机的采样 n 个点： $x_0,x_1,...,x_{n-1}$ ，用它们的均值来表示 $f(x)$ ，这样我们上面的定积分的近似求解为： $\frac{b-a}{n}\sum_{i=0}^{n-1}f(x_i)$ 。

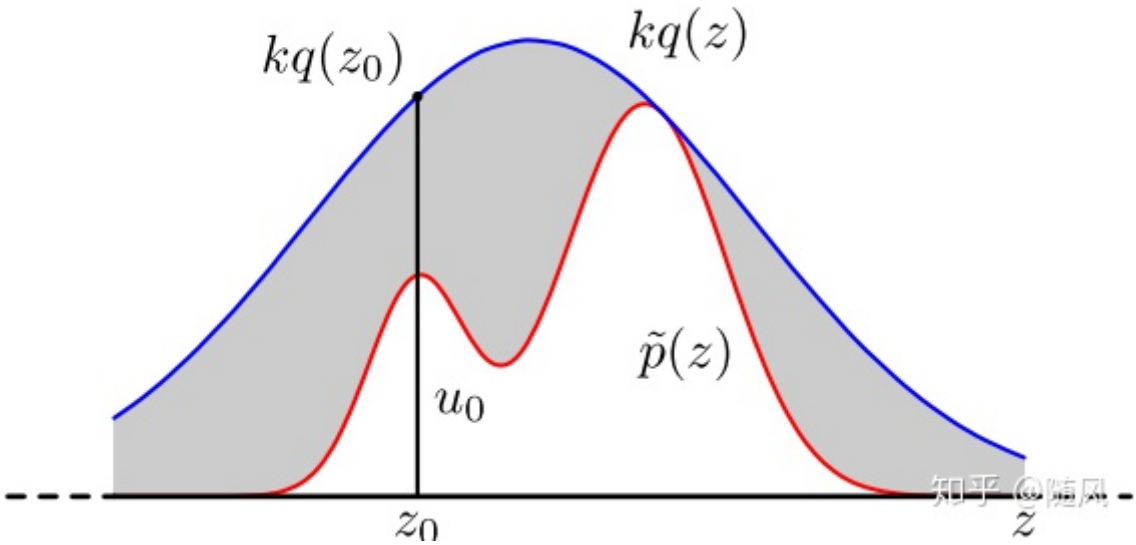
虽然上面的方法可以在一定程度上求解出近似值，但是它隐含了一个假定，即 x 在 $[a,b]$ 上是均匀分布的，而绝大部分情况是不成立的。如果我们用上面的方法，则模拟求出的结果很可能和真实值相差甚远。

怎么解决这个问题呢？ 如果我们可以得到 x 在 $[a,b]$ 上的概率分布函数 $p(x)$ ，那么我们可以这样求解： $\theta = \int_a^b f(x)dx = \int_a^b \frac{f(x)}{p(x)}p(x)dx \approx \frac{1}{n}\sum_{i=0}^{n-1}\frac{f(x_i)}{p(x_i)}$

上式最右边就是蒙特卡罗方法的一般形式。假设 x 在 $[a,b]$ 上服从均匀分布，带入蒙特卡罗积分，可以得到： $\frac{1}{n}\sum_{i=0}^{n-1}\frac{f(x_i)}{1/(b-a)} = \frac{b-a}{n}\sum_{i=0}^{n-1}f(x_i)$ 。也就是说均匀分布的情况是蒙特卡洛方法的一个特例。

由蒙特卡洛方法可知，只要我们知道 x 的概率分布 $p(x)$ ，我们就可以通过采样的方法求和。对于一些常见的分布（如均匀分布、高斯分布），都可以采用上面的方法求解。不过很多时候， x 的概率分布不是常见的分布，这意味着我们没办法得到这些非常见的概率分布的样本集。那这个问题怎么解决呢？（注意，这里 x 的概率密度函数已知，只是不方便采样；并不是在概率密度函数未知的条件下采样）

这里介绍一种方法：接受-拒绝采样，既然 $p(x)$ 太复杂在程序中没法直接采样，那么我们设定一个可采样的分布 $q(x)$ （比如高斯分布），然后按照一定的方法拒绝某些样本，以达到接近 $p(x)$ 分布的目的。



具体采样过程如下，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，如上图。

首先，采样得到 $q(x)$ 的一个样本 z_0 ，然后，从均匀分布 $(0, kq(z_0))$ 中采样一个值 u ，如果 u 落在了上图中的灰色区域，则拒绝这次抽样，否则接受这个样本 z_0 ，重复以上过程得到 n 个样本 $z_0, z_1, \ldots, z_{n-1}$ ，则最后的蒙特卡罗方法求解结果为： $\frac{1}{n}\sum_{i=0}^{n-1}\frac{f(z_i)}{p(z_i)}$ 。

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。

接受-拒绝采样只能满足我们一部分需求，在很多时候我们还是很难得到概率分布的样本集。比如：

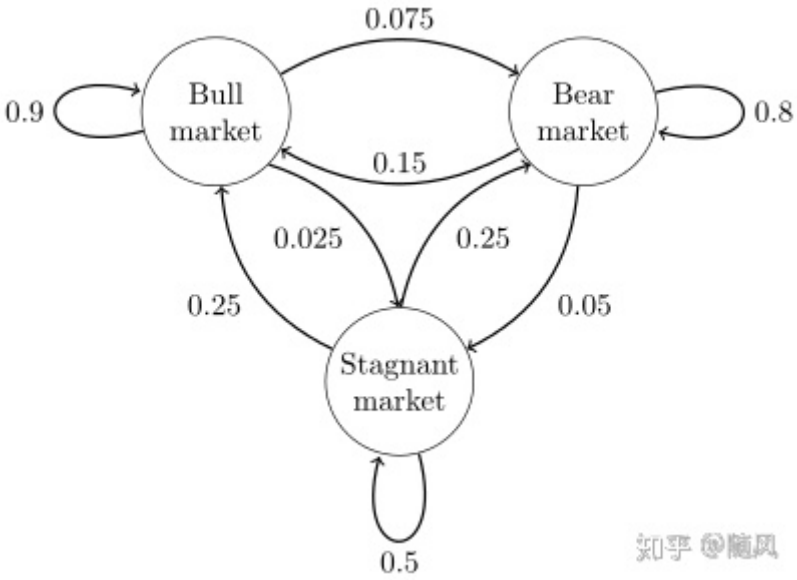
对于一些二维分布 $p(x,y)$ ，有时候我们只能求得条件概率分布 $p(x|y)$ ，却很难得到二维概率分布 $p(x,y)$ 。
对于一些高维的非常见概率分布 $p(x_1,x_2,\ldots,x_n)$ ，我们要找到一个合适的 $q(x_1,x_2,\ldots,x_n)$ 和 k 非常困难。

从上面可以看出，要想将蒙特卡罗方法作为一个通用的采样模拟求和的方法，必须解决如何方便得到各种复杂概率分布对应的采样样本集问题。下面就要用到马尔科夫链来解决这个问题。

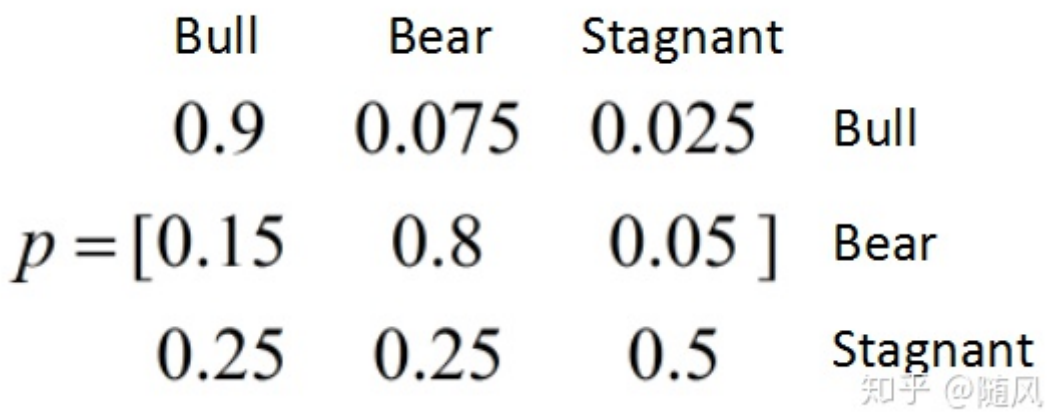
5.2 马尔科夫链（Markov Chain）

马尔科夫链的一个重要性质是，它假设某一时刻状态转移的概率只依赖于它的前一个状态。举个形象的比喻，假如每天的天气是一个状态的话，那么今天是不是晴天只依赖于昨天的天气，而和前天的天气没有任何关系。当然这么说可能有些武断，但是这样做可以大大简化模型的复杂度，因此马尔科夫链在很多时间序列模型中得到广泛的应用，甚至作为整个强化学习的基本框架（加入动作的马尔科夫过程称为马尔科夫决策过程）。

假设序列状态是 $\ldots x_{t-2},x_{t-1},x_t$ ，那么 $t+1$ 时刻的状态 x_{t+1} 只依赖于 t 时刻的状态 x_t ： $P(x_{t+1}|x_t,x_{t-1},\ldots)=P(x_{t+1}|x_t)$ 。我们来看下图这个马尔科夫链模型的具体的例子：



这个马尔科夫链是表示股市模型的，共有三种状态：牛市（Bull market），熊市（Bear market）和横盘（Stagnant market）。每一个状态都以一定的概率转化到下一个状态。比如，牛市以 0.025 的概率转化到横盘的状态。这个状态概率转移图可以以矩阵的形式表示：



下面来看一下这个状态转移矩阵 P 的性质。假设当前股市的概率分布为：[0.3,0.4,0.3]，即 30% 概率的牛市，40% 概率的熊市与 30% 概率的横盘。然后这个状态作为序列概率分布的初始状态 t_0 ，将其带入这个状态转移矩阵计算 $t_1, t_2, ...$ 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=
vector = np.matrix([[0.3, 0.4, 0.3]], dtype=float)
for i in range(100):
    vector = vector * matrix
    print("Current round:", i + 1)
    print(vector)
```

结果:

```
Current round: 57
[[0.62499999 0.31250001 0.0625    ]]
Current round: 58
[[0.62499999 0.31250001 0.0625    ]]
Current round: 59
[[0.62499999 0.3125     0.0625    ]]
Current round: 60
[[0.625  0.3125 0.0625]]
Current round: 61
[[0.625  0.3125 0.0625]]
```

可以发现，从第 60 轮开始，我们的状态概率分布就不变了，一直保持在[0.625 0.3125 0.0625]，即 62.5% 的牛市，31.25% 的熊市与 6.25% 的横盘。那么这个是巧合吗?

我们现在换一个初始概率分布试一试，现在我们用 [0.7,0.1,0.2] 作为初始概率分布。

结果：

```
Current round: 53
[[0.62500002 0.31249999 0.0625    ]]
Current round: 54
[[0.62500001 0.31249999 0.0625    ]]
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: 58
[[0.625  0.3125 0.0625]]
Current round: 59
[[0.625  0.3125 0.0625]]
```

可以看出，尽管这次我们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]， 也就是说我们的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。这是一个非常好的性质，也就是说，如果我们得到了这个稳定概率分布对应的马尔科夫链模型的状态转移矩阵，则我们可以用任意的概率分布样本开始，带入马尔科夫链模型的状态转移矩阵，这样经过一些序列的转换，最终就可以得到符合对应稳定概率分布的样本。

同时，对于一个确定的状态转移矩阵 P ，它的 n 次幂 P^n 在当 n 大于一定的值的时候也是确定的。

```
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=
for i in range(10):
    matrix = matrix * matrix
    print("Current round:", i + 1)
    print(matrix)
```

结果：

```
Current round: 4
[[0.62803724 0.30972343 0.06223933]
 [0.61944687 0.3175772  0.06297594]
 [0.6223933  0.3148797  0.062727  ]]
Current round: 5
[[0.62502532 0.31247685 0.06249783]
 [0.6249537  0.31254233 0.06250397]
 [0.62497828 0.31251986 0.06250186]]
```

```
Current round: 6
[[0.625  0.3125 0.0625]
 [0.625  0.3125 0.0625]
 [0.625  0.3125 0.0625]]
Current round: 7
[[0.625  0.3125 0.0625]
 [0.625  0.3125 0.0625]
 [0.625  0.3125 0.0625]]
```

我们发现，当 $n \geq 6$ 以后， P^n 的值稳定不再变化，而且每一行都为 [0.625 0.3125 0.0625]，这和我们前面的最终状态概率分布是相同的。令 π 表示最终状态概率分布，这里 $\pi = [0.625, 0.3125, 0.0625]$ ，写成数学形式有：

$$P^n = \begin{matrix} \pi(1) & \pi(2) & \pi(3) \\ [\pi(1) & \pi(2) & \pi(3)] \\ \pi(1) & \pi(2) & \pi(3) \end{matrix}$$

π 是方程 $\pi = P\pi$ 的唯一非负解，其中 $\pi = [\pi_1, \pi_2, \pi_3], \sum \pi = 1$ 。通常称 π 为马尔科夫链的平稳分布。

如果我们得到了某个平稳分布所对应的马尔科夫链状态转移矩阵，我们就很容易采样出这个平稳分布的样本集。马尔科夫链的采样过程：

输入马尔科夫链状态转移矩阵 P ，设定状态转移次数阈值 n_1 ，需要的样本个数 n_2
从任意简单概率分布采样得到初始状态值 x_0
for t=0 to $n_1 + n_2$ - 1：从条件概率分布 $P(x|x_t)$ 中采样得到样本 x_{t+1}
样本集 $(x_{n1}, x_{n_1+1}, \ldots, x_{n_1+n_2-1})$ 即为我们需要的平稳分布对应的样本集

以上面股市为例，实现采样：

```
import numpy as np

sample_union = ['bull', 'bear', 'stagnant']
init_status = [0.3, 0.4, 0.3]
P = [[0.9, 0.075, 0.025], [0.15, 0.8, 0.05], [0.25, 0.25, 0.5]]
n1 = 30
n2 = 10
sample_result = []
```

```
for i in range(n1 + n2):
    print("Current round " + str(i) + ":", init_status)
    sample = np.random.choice(sample_union, size=None, replace=True, p=init_status)
    index = sample_union.index(sample)
    init_status = P[index]
    if i >= n1:
        sample_result.append(sample)

print(sample_result)
```

结果：

```
Current round 35: [0.9, 0.075, 0.025]
Current round 36: [0.9, 0.075, 0.025]
Current round 37: [0.15, 0.8, 0.05]
Current round 38: [0.25, 0.25, 0.5]
Current round 39: [0.9, 0.075, 0.025]
['bull', 'bull', 'bull', 'bull', 'bull', 'bull', 'bear', 'stagnant', 'bull', 'bull']
```

关于 n_1 的选取，其实我们是希望状态概率分布达到稳定后再进行采样，所以一般情况下尽量选取一个大一点的值。

如果假定我们可以得到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需要的样本集，进而进行蒙特卡罗方法。但是一个重要的问题是，随意给定一个平稳分布 π ，如何得到它所对应的马尔科夫链状态转移矩阵 P 呢？MCMC采样将通过迂回的方式解决上面这个问题。

5.3 MCMC 采样和 M-H 采样

在解决从平稳分布 π ，找到对应的马尔科夫链状态转移矩阵 P 之前，我们还需要先看看马尔科夫链的**细致平稳**条件。

如果马尔科夫链的状态转移矩阵 P 和概率分布 π 对所有的 (i, j) 满足：$\pi(i)P(i,j) = \pi(j)P(j,i)$ ，则称概率分布 π 是状态转移矩阵 P 的**细致平稳分布**。满足细致平稳条件的 π 一定也是平稳分布，证明很简单：

$\sum_{i=1}^{\infty}\pi(i)P(i,j) = \sum_{i=1}^{\infty}\pi(j)P(j,i) = \pi(j)\sum_{i=1}^{\infty}P(j,i) = \pi(j)$ ，用矩阵表示即为：

$\pi P = \pi$ 。

因此，只要能找到使概率分布 π 满足细致平稳分布的矩阵 P 即可。但是，仅从细致平稳条件还是很难找到合适的矩阵 P 。比如我们的目标平稳分布是 π ，随机找一个马尔科夫链状态转移矩

阵 Q ，它是很难满足细致平稳条件的，即： $\pi(i)Q(i,j) \neq \pi(j)Q(j,i)$ 。

那么如何使这个等式成立呢？这里就要用到 MCMC 采样。作为一种随机采样方法，马尔科夫链蒙特卡罗（Markov Chain Monte Carlo，MCMC）在机器学习，深度学习以及自然语言处理等领域都有广泛的应用，是很多复杂算法求解的基础。

一般情况下，目标平稳分布 π 和某一个马尔科夫链状态转移矩阵 Q 不满足细致平稳条件，即： $\pi(i)Q(i,j) \neq \pi(j)Q(j,i)$ 。

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 $\alpha(i,j)$ ，即： $\pi(i)Q(i,j)\alpha(i,j) = \pi(j)Q(j,i)\alpha(j,i)$ 。

问题是什么样的 $\alpha(i,j)$ 可以使等式成立呢？其实很简单，只要满足下两式即可：

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

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

这样，就得到了平稳分布 π 对应的马尔科夫链状态转移矩阵 P ，满足： $P(i,j) = Q(i,j)\alpha(i,j)$ 。

也就是说，我们的目标矩阵 P 可以通过任意一个马尔科夫链状态转移矩阵 Q 乘以 $\alpha(i,j)$ 得到。可能有人会问，这样乘积的结果 P 还是马尔科夫链状态转移矩阵吗？答：不是，这里只是做了一个近似矩阵 P ，毕竟真实的满足平稳分布 π 的马尔科夫链状态转移矩阵 P 我们求不出来。 $\alpha(i,j)$ 一般称之为接受率，取值在 [0,1] 之间，可以理解为一个概率值。即目标矩阵 P 可以通过任意一个马尔科夫链状态转移矩阵 Q 以一定的接受率获得。MCMC 的采样过程：

1、输入一个任意选定的马尔科夫链状态转移矩阵 Q ，平稳分布 π ，设定状态转移次数阈值 n_1 ，需要的样本个数 n_2

2、从任意简单概率分布采样得到初始状态值 x_0

3、for t=0 to $n_1 + n_2$ - 1：

从条件概率分布 $Q(x|x_t)$ 中采样得到样本 x_*
从均匀分布采样 $u \sim uniform[0,1]$
如果 $u < \alpha(x_t, x_*) = \pi(x_*)Q(x_*, x_t)$ ，则接受转移，即 $x_{t+1} = x_*$
否则不接受转移 $x_{t+1} = x_t$

样本集 $(x_{n1}, x_{n_1+1}, \ldots, x_{n_1+n_2-1})$ 即为我们需要的平稳分布对应的样本集。

上面这个过程就是 MCMC 采样的原理，但是这个采样算法还是比较难在实际中应用，为什么呢？问题在上面的第三步，接受率。由于 $\alpha(x_t, x_*)$ 可能非常小，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。有可能我们采样了上百万次，马尔可夫链还没有收敛，也就是上面这个 n_1 要非常大，这让人难以接受，怎么办呢？下面来看 M-H采样。

M-H采样是 Metropolis-Hastings 采样的简称，M-H 采样解决了我们上一节 MCMC 采样接受率过低的问题。回到细致平稳条件： $\pi(i)Q(i,j)\alpha(i,j) = \pi(j)Q(j,i)\alpha(j,i)$ 。我们采样效率低的原因是 $\alpha(i,j)$ 太小了，比如 0.1，而 $\alpha(j,i)$ 为 0.2。即：
$\pi(i)Q(i,j) * 0.1 = \pi(j)Q(j,i) * 0.2$ ，这时我们可以看到，如果两边同时扩大五倍，接受率提高到了 0.5，但是细致平稳条件仍然是满足的，即：
$\pi(i)Q(i,j) * 0.5 = \pi(j)Q(j,i) * 1$ ，这样我们的接受率可以做如下改进，即：

$\alpha(i,j) = min[\frac{\pi(j)Q(j,i)}{\pi(i)Q(i,j)}, 1]$ 。通过这个微小的改造，我们就得到了可以在实际应用中使用的 M-H 采样算法，过程如下：

1、输入一个任意选定的马尔科夫链状态转移矩阵 Q ，平稳分布 π ，设定状态转移次数阈值 n_1 ，需要的样本个数 n_2

2、从任意简单概率分布采样得到初始状态值 x_0

3、for t=0 to $n_1 + n_2$ - 1：

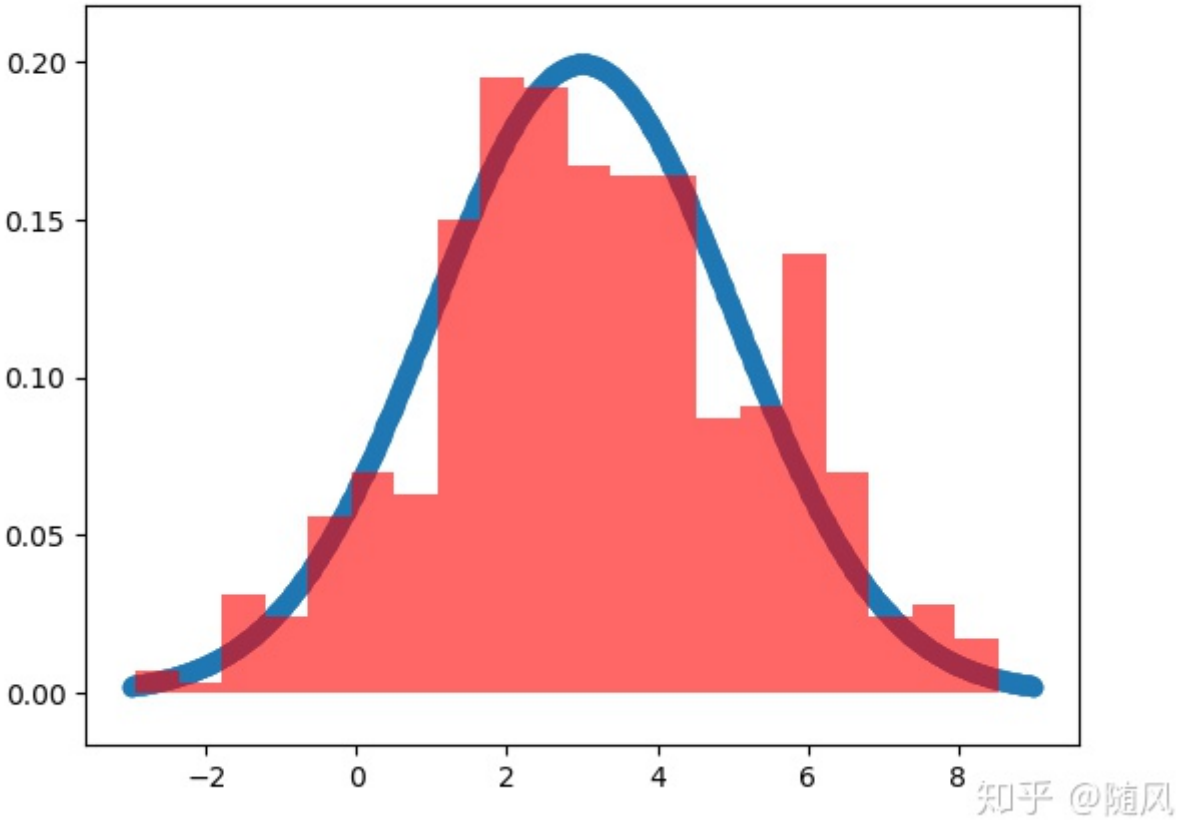
从条件概率分布 $Q(x|x_t)$ 中采样得到样本 x_*
从均匀分布采样 $u \sim uniform[0,1]$
如果 $u < \alpha(x_t, x_*) = min[\frac{\pi(x_*)Q(x_*, x_t)}{\pi(x_t)Q(x_t, x_*)}, 1]$ ，则接受转移，即 $x_{t+1} = x_*$
否则不接受转移 $x_{t+1} = x_t$

样本集 $(x_{n1}, x_{n_1+1}, \ldots, x_{n_1+n_2-1})$ 即为我们需要的平稳分布对应的样本集。

如果我们选择的马尔科夫链状态转移矩阵 Q 是对称矩阵，即满足 $Q(i,j) = Q(j,i)$ ，这时接受率可以进一步简化为： $\alpha(i,j) = min[\frac{\pi(j)}{\pi(i)}, 1]$ 。

下面举个具体例子来实现 M-H 采样。在例子里，我们的目标平稳分布是一个均值为 3，标准差为 2 的正态分布，选择的马尔可夫链状态转移矩阵 $Q(i,j)$ 的条件转移概率是以 i 为均值，方差为 1 的正态分布在位置 j 的概率值。

```
import random
import matplotlib.pyplot as plt
from scipy.stats import norm
import numpy as np


def pi(theta):
    # loc 表示均值, scale 表示标准差
    pro = norm.pdf(theta, loc=3, scale=2)
    return pro


def Q(i, j):
    pro = norm.pdf(j, loc=i, scale=1)
    return pro


n1 = 3000
n2 = 500
x = [0 for i in range(n1 + n2)]

for i in range(n1 + n2):
    if i == n1 + n2 - 1:
        break
    x_t = x[i]
    x_star = norm.rvs(loc=x_t, scale=1, size=1, random_state=None)[0]
    u = random.uniform(0, 1)
    alpha = min(1, ((pi(x_star) * Q(x_star, x_t)) / (pi(x_t) * Q(x_t, x_star))))
    if u < alpha:
        x[i + 1] = x_star
    else:
        x[i + 1] = x[i]

x_axis = np.linspace(-3, 9, 1000)
plt.scatter(x_axis, norm.pdf(x_axis, loc=3, scale=2))
num_bins = 20
plt.hist(x[-n2:], num_bins, normed=1, facecolor='red', alpha=0.6)
plt.show()
```



M-H 采样解决了使用蒙特卡罗方法需要的任意概率分布样本集的问题，因此在实际生产环境得到了广泛的应用。

但是在大数据时代，M-H 采样面临着两大难题：

(1)、我们的数据特征非常的多，M-H 采样由于接受率计算式 $\frac{\pi(j)Q(j,i)}{\pi(i)Q(i,j)}$ 的存在，在高维时需要的计算时间非常可观，算法效率很低。同时 $\alpha(i,j)$ 一般小于1，有时候辛苦计算出来却被拒绝了。能不能做到不拒绝转移呢?

(2)、由于特征维度非常大，很多时候甚至很难求出目标的各特征维度联合概率分布，但是可以方便求出各个特征之间的条件概率分布。这时候我们能不能只有各维度之间条件概率分布的情况下方便的采样呢?

Gibbs 采样解决了上面两个问题，因此在大数据时代，MCMC 采样基本是 Gibbs 采样的天下。

5.4 Gibbs 采样

在上一节中，我们讲到了细致平稳条件：如果非周期马尔科夫链的状态转移矩阵 P 和概率分布 π 对于所有的 i,j 满足： $\pi(i)P(i,j)=\pi(j)P(j,i)$ ，则称概率分布 π 是状态转移矩阵

P 的平稳分布。在 M-H 采样中我们通过引入接受率 $\alpha(i,j)$ 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 $\pi(x,y)$ 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 $A(x_1,y_1),B(x_1,y_2)$ ，容易发现下面两式成立：

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

由于两式的右边相等，因此我们有： $\pi(x_1,y_1)\pi(y_2|x_1)=\pi(x_1,y_2)\pi(y_1|x_1)$ 。也就是： $\pi(A)\pi(y_2|x_1)=\pi(B)\pi(y_1|x_1)$ 。

观察上式再观察细致平稳条件的公式，**我们发现在 $x=x_1$ 这条直线上，如果用条件概率分布 $\pi(y|x_1)$ 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！**同样的道理，在 $y=y_1$ 这条直线上，如果用条件概率分布 $\pi(x|y_1)$ 作为马尔科夫链的状态转移概率，则任意两点之间的转移也满足细致平稳条件。假如有一点 $C(x_2,y_1)$ ，可以得到：$\pi(A)\pi(x_2|y_1)=\pi(C)\pi(x_1|y_1)$ 。

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

$$P(A\to B)=P(y_1,y_2)=\pi(y_2|x_1),P(A\to C)=P(x_1,x_2)=\pi(x_2|y_1)$$

利用上面得到的状态转移矩阵 P ，我们就得到了二维 Gibbs 采样，这个采样需要两个维度之间的条件概率。具体过程如下：

1、输入 x,y 的条件概率分布 $P(x|y),P(y|x)$ ，设定状态转移次数阈值 n_1 ，需要的样本个数 n_2

2、随机初始化状态值 (x_0,y_0)

3、for t=0 to n_1+n_2 - 1：

从条件概率分布 $P(y|x_t)$ 中采样得到样本 y_{t+1}
从条件概率分布 $P(x|y_{t+1})$ 中采样得到样本 x_{t+1}

样本集 $(x_{n1},y_{n1}),\dots,(x_{n1+n2-1},y_{n1+n2-1})$ 即为我们需要的平稳分布对应的样本集。

整个采样过程，我们通过轮换坐标轴的方式采样： $(x_1,y_1)\to(x_1,y_2)\to(x_2,y_2)\dots$ 当然，坐标轴轮换不是必须的，我们也可以每次随机选择一个坐标轴进行采样。不过常用的Gibbs 采样的实现都是基于坐标轴轮换的。

这个算法推广到多维的时候也是成立的。比如一个 n 维的概率分布 $\pi(x_1, x_2, \ldots, x_n)$ ，我们可以通过在 n 个坐标轴上轮换采样，来得到新的样本。对于轮换到的任意一个坐标轴 x_i 上的转移，马尔科夫链的状态转移概率为 $P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ ，即固定 n-1 个坐标轴，在某一个坐标轴上移动。具体的算法过程如下：

1、输入对应的所有特征的条件概率分布，设定状态转移次数阈值 n_1 ，需要的样本个数 n_2

2、随机初始化状态值 $(x_1^0, x_2^0, \ldots, x_n^0)$

3、for t=0 to $n_1 + n_2$ - 1：

从条件概率分布 $P(x_1|x_2^t, x_3^t, \ldots, x_n^t)$ 中采样得到样本 x_1^{t+1}
从条件概率分布 $P(x_2|x_1^{t+1}, x_3^t, \ldots, x_n^t)$ 中采样得到样本 x_2^{t+1}
……
从条件概率分布 $P(x_n|x_1^{t+1}, x_2^{t+1}, \ldots, x_{n-1}^{t+1})$ 中采样得到样本 x_n^{t+1}

样本集 $(x_1^{n1}, x_2^{n1}, \ldots, x_n^{n1}), \ldots, (x_1^{n1+n2-1}, x_2^{n1+n2-1}, \ldots, x_n^{n1+n2-1})$ 即为我们需要的平稳分布对应的样本集。

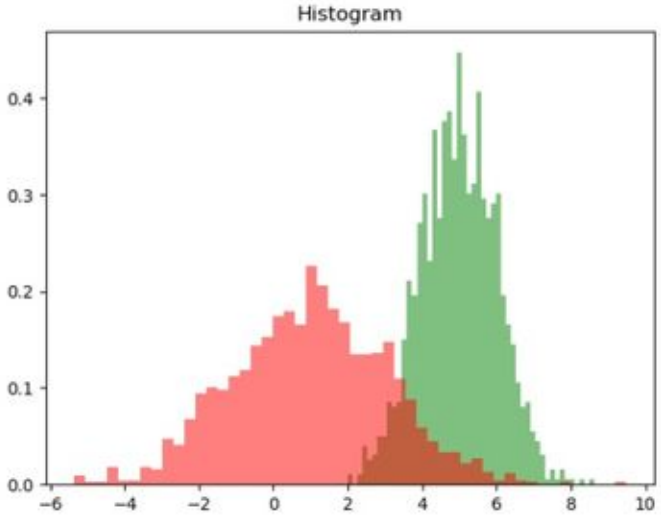
同样的，轮换坐标轴不是必须的，我们可以随机选择某一个坐标轴进行状态转移，只不过常用的 Gibbs 采样的实现都是基于坐标轴轮换的。

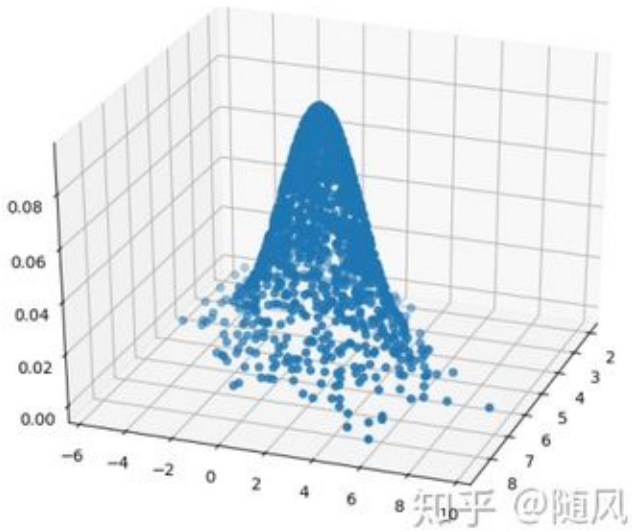
下面举个具体例子来实现 Gibbs 采样。假设我们要采样的是一个二维正态分布 $norm(u, \Sigma)$ 其中： $u = (u_1, u_2) = (5, 1)$ ， $\Sigma = [[\sigma_1^2, p\sigma_1\sigma_2], [p\sigma_1\sigma_2, \sigma_2^2]] = [[1, 1], [1, 4]]$ 采样过程中需要的状态转移条件分布为：

$$P(x|y) = norm(u_1 + p\sigma_1/\sigma_2(x_2 - u_2), (1 - p^2)\sigma_1^2)$$
$$P(y|x) = norm(u_2 + p\sigma_2/\sigma_1(x_1 - u_1), (1 - p^2)\sigma_2^2)$$

```
import math
import random

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.stats import multivariate_normal

samplesource = multivariate_normal(mean=[5, 1], cov=[[1, 1], [1, 4]])


def p_x_to_y(x, u1, u2, delta1, delta2):
    return (random.normalvariate(u2 + p * delta2 / delta1 * (x - u1), math.sqrt((1 - p
```

```
def p_y_to_x(y, u1, u2, delta1, delta2):
    return (random.normalvariate(u1 + p * delta1 / delta2 * (y - u2), math.sqrt((1 - p


n1 = 10000
n2 = 1500
x_res = []
y_res = []
z_res = []
u1 = 5
u2 = 1
delta1 = 1
delta2 = 2

p = 0.5
y = u2

for i in range(n1 + n2):
    x = p_y_to_x(y, u1, u2, delta1, delta2)
    y = p_x_to_y(x, u1, u2, delta1, delta2)
    z = samplesource.pdf([x, y])
    x_res.append(x)
    y_res.append(y)
    z_res.append(z)

num_bins = 50
plt.hist(x_res[-n2:], num_bins, normed=1, facecolor='green', alpha=0.5)
plt.hist(y_res[-n2:], num_bins, normed=1, facecolor='red', alpha=0.5)
plt.title('Histogram')
plt.show()

fig = plt.figure()
ax = Axes3D(fig, rect=[0, 0, 1, 1], elev=30, azim=20)
ax.scatter(x_res[-n2:], y_res[-n2:], z_res[-n2:], marker='o')
plt.show()
```


Histogram
0.4
0.3
0.2
0.1
0.0
−6
−4
−2
0
2
4
6
8
10


0.08
0.06
0.04
0.02
0.00
−6
−4
−2
0
2
4
6
8
10
2
3
4
5
6
7
8
知乎 @随风