采样方法

无意识统计学家法则(LOTUS)

已知随机变量X的概率密度为$p(x)$，$g(X)$ 为X的函数，则：

$$E(g(X)) = \int_a^b g(x)p_X(x)dx \tag{1}$$

思想

采样方法背后的一般思想是得到从概率分布$p(z)$中独立抽取的一组变量$z(l)$， 其 中$l = 1, 2, \dots L$。这使得期望可以通过有限和的方式计算，即

$$\hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(z^{(l)}) \tag{2}$$

蒙特卡洛数值积分

对于一个连续函数f，要计算的积分有如下形式：

$$F = \int_a^b f(x)dx \tag{3}$$

转换：

$$F = \int_a^b \frac{f(x)}{q(x)}q(x)dx \tag{4}$$

根据$q(x)$采样X_i，f的蒙特卡洛积分公式为：

$$F^N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{q(X_i)} \tag{5}$$

这公式的作用相当于在对$f(x)$做积分，只不过不那么“精确”，即蒙特卡罗积分是**对理想积分的近似**。

那么这个近似是如何完成的？很简单，核心就是两个字：**采样(Sampling)**。对一个连续函数的采样方法是在该函数的定义域中随机挑N个值，并求出对应的N个$f(X_i)$，就得到了样本集合。再对这些样本集合做一些换算，就可以得到一个近似的积分了。对于蒙特卡罗积分，**采样样本越多，就越逼近真实的积分结果**，这是蒙特卡罗积分的最核心特性。

证明：

$$\begin{aligned}\mathbb{E}[F^N] &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[\frac{f(X_i)}{q(X_i)}] \\ &= \frac{1}{N}\sum_{i=0}^{N-1}\int_\omega \frac{f(x)}{q(x)}q(x)dx \\ &= F\end{aligned} \tag{6}$$

实际应用中，x可以选择均匀分布，$q(x) = \frac{1}{m}$，随机采样得到$\frac{f(x_i)}{q(x_i)}$，采样很多次后，求期望。

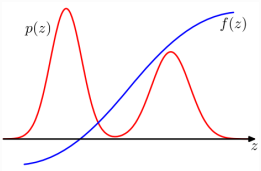
这样把$q(x)$看做是x在区间内的概率分布，而把前面的分数部份看做一个函数，然后在$q(x)$下抽取n个样本，当n足够大时，可以用采用均值来近似：

因此只要$q(x)$比较容易采到数据样本就行了。随机模拟方法的核心就是如何对一个概率分布得到样本，即抽样(sampling) 。

Monte Carlo principle

Monte Carlo 抽样计算随即变量的期望值是接下来内容的重点：X 表示随即变量，服从概率分布$p(x)$，那么要计算$f(x)$的期望，只需要我们不停从 $p(x)$ 中抽样x_i ，然后对这些$f(x_i)$取平均即可近似$f(x)$的期望。

$$E_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \tag{7}$$



逆变换采样

比较简单的一种情况是，我们可以通过PDF与CDF之间的关系，求出相应的CDF。或者我们根本就不知道PDF，但是知道CDF。此时就可以使用Inverse CDF的方法来进行采样。这种方法又称为逆变换采样（Inverse transform sampling）。

所以，通常可以通过对PDF进行积分来得到概率分布的CDF。然后我们再得到CDF的反函数，如果你想得到n个观察值，则重复下面的步骤n次：

- 从 Uniform(0,1) 中随机生成一个值（前面已经说过，计算机可以实现从均匀分布中采样），用U表示。
- 计算$x = CDF^{-1}(U)$的值，则x就是从PDF中得出的一个采样点。

举个具体例子吧，例如我想按照标准正态分布$N(0,1)$取10个随机数，那么我首先在(0,1)上按照均匀分布取10个点

0.4505 0.0838 0.2290 0.9133 0.1524 0.8258 0.5383 0.9961 0.0782 0.4427

然后，我去找这些值在CDF上对应的x，如下

-0.1243 -1.3798 -0.7422 1.3616 -1.0263 0.9378 0.0963 2.6636 -1.4175 -0.1442

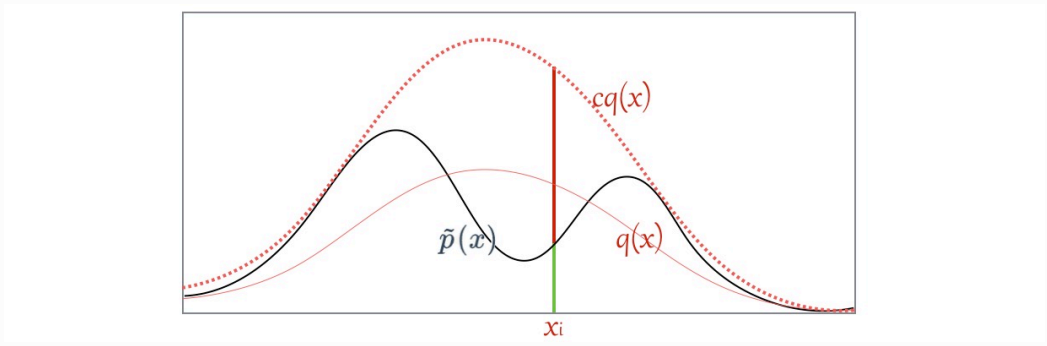
那么上述这些点，就是我按照正态分布取得的10个随机数。

缺点:

- 有时CDF不好求
- CDF的反函数不好求

接受拒绝采样

很多实际问题中，$p(x)$是很难直接采样的的，因此，我们需要求助其他的手段来采样。既然$p(x)$太复杂在程序中没法直接采样，那么我设定一个程序可抽样可抽样的分布$q(x)$比如高斯分布，然后按照一定的方法拒绝某些样本，达到接近$p(x)$分布的目的。其中$q(x)$叫做 proposal distribution 。



具体操作如下，设定一个方便抽样的函数 $q(x)$ ，以及一个常量c，使得 $p(x)$ 总在$cq(x)$ 的下方。

- x 轴方向：从 $q(x)$ 分布抽样得到$x(i)$；
- y 轴方向：对$x(i)$计算接受概率：$\alpha = \frac{p(x_i)}{cq(x_i)}$；
- 从均匀分布$(0,1)$中抽样得到u；
- 如果： $\alpha \geq u$ ， 则接受$x(i)$作为$p(x)$的抽样；否则，拒绝，重复以上过程

它的原理从直观上来解释也是相当容易理解的。在上图的例子中，从哪些位置抽出的点会比较容易被接受？显然，红色曲线和绿色曲线所示之函数更加接近的地方接受概率较高，也即是更容易被接受，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）的地方采到的点就会比较少，这也就保证了这个方法的有效性。

在高维的情况下，Rejection Sampling 会出现两个问题：

- 合适的q分布比较难以找到，
- 很难确定一个合理的 c 值。

这两个问题会导致拒绝率很高，无用计算增加。

重要性采样

$f(x)$为x的函数，$p(x)$为x的PDF，问题为求下式的积分：

$$E[f(x)] = \int_X f(x)p(x)dx \tag{8}$$

按照蒙特卡洛求定积分的方法，我们将从满足$p(x)$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

$$E[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{9}$$

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。这时我们该怎么做呢？于是想到做等量变换，将其转化为：

$$\int_X f(x)p(x)dx = \int_X f(x)\frac{p(x)}{q(x)}q(x)dx = \int_X f(x)w(x)q(x)dx \tag{10}$$

其中$w(x) = \frac{p(x)}{q(x)}$，称其为重要性权重。那么，根据$q(x)$ 采样x_i **(此过程可以使用逆变换采样)**，那么蒙特卡洛估计就是：

$$I_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x_i)w(x_i) \tag{11}$$

我们来考察一下上面的式子，$p(x)$和 $f(x)$是确定的，我们要确定的是$q(x)$。要确定一个什么样的分布才会让采样的效果比较好呢？ 直观的感觉是，样本的方差越小期望收敛速率越快。举个简单的例子比如一次采样是 0, 一次采样是 1000, 平均值是 500,这样采样效果很差，如果一次采样是 499, 一次采样是 501, 你说期望是 500,可信度还比较高。因此，我们很有必要研究相应的蒙特卡洛的方差：

$$var_{q(x)}(f(x)w(x)) = \mathbb{E}_{q(x)}(f^2(x)w^2(x)) - I^2(f) \tag{12}$$

上式中第二项不依赖于$q(x)$，因此我们只需要最小化第一项就可以。那么根据Jensen不等式可知具有下界即：

$$\mathbb{E}_{q(x)}(f^2(x)w^2(x)) \geq (\mathbb{E}_{q(x)}(|f(x|w(x)))^2 = (\int |f(x|p(x)dx)^2 \tag{13}$$

那么下界达到即等号成立当且仅当

$$q^*(x) = \frac{|f(x|p(x)}{\int |f(x|p(x)dx} \tag{14}$$

尽管在实际中，上式很难拿取到，但是他告我们一个真理就是，当我们取样$p(x)$时候，应该是取$|f(x)|p(x)$相当大值，这样才有高效率的采样。 这表明重要性采样有可能比用原来的$p(x)$分布抽样更加有效。

采样在DL中的应用

重要性采样

语言模型：在训练阶段，我们的目标是使得训练集中每个词语w的交叉熵最小，也就是使得softmax层输出值的负对数取值最小。模型的损失函数可以写成：

$$J_\theta = -log\frac{exp(h^T v_w^{'})}{\sum_{w_i \in V} exp(h^T v_{w_i}^{'})} \tag{15}$$

为了便于推导，我们将J_θ改写为：

$$J_\theta = -h^T v_w^{'} + log \sum_{w_i \in V} exp(h^T v_{w_i}^{'}) \tag{16}$$

令 $-\mathcal{E}(w)$代替 $h^T v_w^{'}$，于是得到等式：

$$J_\theta = \mathcal{E}(w) + log \sum_{w_i \in V} exp(-\mathcal{E}(w_i)) \tag{17}$$

在反向传播阶段，我们可以将损失函数对于θ的偏导写为：

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \nabla_\theta log \sum_{w_i \in V} exp(-\mathcal{E}(w_i)) \tag{18}$$

因为$log\ x$的导数是$\frac{1}{x}$，则上式又可以改写为：

$$\begin{aligned}
\nabla_\theta J_\theta &= \nabla_\theta \mathcal{E}(w) + \frac{1}{\sum_{w_i \in V} exp(-\mathcal{E}(w_i))} \nabla_\theta \sum_{w_i \in V} exp(-\mathcal{E}(w_i)) \\
&= \nabla_\theta \mathcal{E}(w) + \frac{1}{\sum_{w_i \in V} exp(-\mathcal{E}(w_i))} \sum_{w_i \in V} \nabla_\theta exp(-\mathcal{E}(w_i)) \\
&= \nabla_\theta \mathcal{E}(w) + \frac{1}{\sum_{w_i \in V} exp(-\mathcal{E}(w_i))} \sum_{w_i \in V} exp(-\mathcal{E}(w_i)) \nabla_\theta (-\mathcal{E}(w_i)) \\
&= \nabla_\theta \mathcal{E}(w) + \sum_{w_i \in V} \frac{exp(-\mathcal{E}(w_i))}{\sum_{w_i \in V} exp(-\mathcal{E}(w_i))} \nabla_\theta (-\mathcal{E}(w_i))
\end{aligned} \tag{19}$$

注意：$\frac{exp(-\mathcal{E}(w_i))}{\sum_{w_i \in V} exp(-\mathcal{E}(w_i))}$ 就是词语w_i的$softmax$概率值$P(w_i)$。将其代入上面的等式中得到：

$$\begin{aligned}
\nabla_\theta J_\theta &= \nabla_\theta \mathcal{E}(w) + \sum_{w_i \in V} P(w_i) \nabla_\theta (-\mathcal{E}(w_i)) \\
&= \nabla_\theta \mathcal{E}(w) - \sum_{w_i \in V} P(w_i) \nabla_\theta (\mathcal{E}(w_i))
\end{aligned} \tag{20}$$

梯度值可以分解为两个部分：一部分与目标词语w正相关（等式右边的第一项），另一部分与其余所有词语负相关，按照各个词语的出现概率分配权重（等式右边的第二项）。我们可以发现，等式右边的第二项其实就是词表V中所有词语w_i的的期望值：

$$\sum_{w_i \in V} P(w_i) \nabla_\theta (\mathcal{E}(w_i)) = E_{w_i \sim P}[\nabla_\theta (\mathcal{E}(w_i))] \tag{21}$$

现在大多数基于采样方法的核心都是用简单的过程来近似计算后一项的值。

如果已知网络模型的分布$P(w)$，于是我们就可以从中随机采样m个词语$w_1, \cdots, w_m$ ，并用下面的公式计算期望值：

$$E_{w_i \sim P}[\nabla_\theta(\mathcal{E}(w_i))] \approx \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta(\mathcal{E}(w_i)) \tag{22}$$

但是为了实现从概率值分布P中采样，我们必须先计算得到P，而这个过程正是我们想绕开的。于是，我们用另一种类似于P但是采样更方便的分布Q来代替。在语言建模的任务中，直接把训练集的$unigram$分布作为Q不失为良策。这就是经典的重要性采样的做法：它使用蒙特卡洛方法得到分布Q来模拟真实的分布P。可是，被采样到的词语w仍然需要计算其概率值$P(w)$。

上面把P_{w_i}赋值为$\nabla_\theta(\mathcal{E}(w_i))$ 的权重，这里我们把权重值改为与Q相关的一个因子。这个因子是 $\frac{r(w_i)}{R}$。其中:

$$r(w) = \frac{exp(-\mathcal{E}(w))}{Q(w)},\ R = \sum_{j=1}^{m} r(w_j) \tag{23}$$

于是期望的估计值公式可以写为:

$$E_{w_i \sim P}[\nabla_\theta(\mathcal{E}(w_i))] \approx \sum_{i=1}^{m}\frac{r(w_i)}{R}\nabla_\theta(\mathcal{E}(w_i)) \tag{24}$$

若是采样的数量越少，估计的分布与真实分布差别越大。如果样本数量非常少，在训练过程中网络模型的分布P可能与$unigram$的分布Q差异很大，会导致模型发散，因此我们需要调整到合适的样本数量。Bengio和Senécal的论文中介绍了一种快速选择样本数量的方法。最终的运算速度比传统的softmax提升了19倍。

注:

$$\frac{1}{R}\sum_{i=1}^{m} r(w_i)\nabla_\theta(\mathcal{E}(w_i)) = \sum_{i=1}^{m}\frac{exp(-\mathcal{E}(w_i))}{\sum_{i=1}^{m} exp(-\mathcal{E}(w_i))}\nabla_\theta(-\mathcal{E}(w_i)) \tag{25}$$

噪声对比估计

噪声对比估计是Mnih和Teh发明的一种比重要性采样更稳定的采样方法，因为某些情况下重要性采样存在导致分布Q与P分道扬镳的风险。NCE不是直接估计某个词语的概率值。相反，它借助一个辅助的损失值，从而实现了正确词语概率值最大化这一目标。

NCE的想法：训练一个模型来区分目标词语与噪声。于是待解决的问题就由预测正确的词语简化为一个二值分类器任务，分类器试图将正确的词语与其它噪声样本中区分开来。对于每个词语w_i，它的前n个词语$w_{t-1}, ..., w_{t-n+1}$表示为w_i的语境c_i。然后从含有噪声的分布Q中生成k个噪声样本$\tilde{w}_{ik}$。参照重要性采样的方法，这里也可以从训练数据的$unigram$分布中采样。由于分类器需要用到标签数据，我们把语境 c_i 对应的所有正确的词语w_i标记为正样本$(y=1)$，其余的噪声词语$\tilde{w}_{ik}$作为负样本$(y=0)$。

接着，用逻辑回归模型来训练样本数据:

$$J_\theta = -\sum_{w_i \in V}[logP(y=1|w_i, c_i) + kE_{\tilde{w}_{ik}\sim Q}[logP(y=0|\tilde{w}_{ij}, c_i)]] \tag{26}$$

由于计算所有噪声样本的期望$E_{\tilde{w}_{ik}\sim Q}$ 仍需要对词表V中的词语求和，得到标准化的概率值。于是可以采用蒙特卡洛方法来估算:

$$\begin{aligned} J_\theta &= -\sum_{w_i \in V}[logP(y=1|w_i, c_i) + k\sum_{j=1}^{k}\frac{1}{k}\ logP(y=0|\tilde{w}_{ij}, c_i)] \\ &= -\sum_{w_i \in V}[logP(y=1|w_i, c_i) + \sum_{j=1}^{k} logP(y=0|\tilde{w}_{ij}, c_i)] \end{aligned} \tag{27}$$

实际上，我们是从两个不同的分布中采样数据：正样本是根据语境c从训练数据集P_{train}中采样，而负样本从噪声分布Q中采样获得。因此，无论是正样本还是负样本，其概率值都可以表示成上述两种分布带权重的组合，权重值对应于来自该分布的样本值：

$$P(y,w|c) = \frac{1}{k+1}P_{train}(w|c) + \frac{k}{k+1}Q(w) \tag{28}$$

于是，样本来自于$Ptrain$的概率值可以表示为条件概率的形式：

$$\begin{aligned} P(y=1|w,c) &= \frac{\frac{1}{k+1}P_{train}(w|c)}{\frac{1}{k+1}P_{train}(w|c) + \frac{k}{k+1}Q(w)} \\ &= \frac{P_{train}(w|c)}{P_{train}(w|c) + kQ(w)} \end{aligned} \tag{29}$$

由于不知道P_{train}（待计算项），我们就用P来代替：

$$P(y=1|w,c) = \frac{P(w|c)}{P(w|c) + kQ(w)} \tag{30}$$

当然，样本为负样本的概率值就是$P(y=0|w,c) = 1 - P(y=1|w,c)$ 。值得注意的是，已知c求词语w出现的概率值$P(w|c)$的计算方法实际上就是$softmax$的定义：

$$P(w|c) = \frac{exp(h^T v'_w)}{\sum_{w_i \in V} exp(h^T v'_{w_i})} \tag{31}$$

因为分母只与h相关，h的值与c相关（假设V不变），那么分母可以简化为$Z(c)$来表示。$softmax$就变为下面的形式：

$$P(w|c) = \frac{exp(h^T v'_w)}{Z(c)} \tag{32}$$

为了求解$Z(c)$，还是需要对V中所有词语出现的概率值求和。NCE则用了一个小技巧巧妙地绕开：即把标准化后的分母项$Z(c)$当作模型的待学习参数。Mnih和Teh、Vaswani等在论文中都把$Z(c)$的值固定设为1，他们认为这样不会对模型的效果造成影响。Zoph则认为，即使训练模型，最终得到$Z(c)$的值也是趋近于1，并且方差很小。若是我们把上面$softmax$等式中的$Z(c)$项改为常数1，等式就变为：

$$P(w|c) = exp(h^T v'_w) \tag{33}$$

再把上面的式子代入求解$P(y=1|w,c)$，得到：

$$P(y=1|w,c) = \frac{exp(h^T v'_w)}{exp(h^T v'_w) + kQ(w)} \tag{34}$$

继续把上式代入逻辑回归的目标函数中，得到：

$$J_\theta = -\sum_{w_i \in V}[log\frac{exp(h^T v'_w)}{exp(h^T v'_w) + kQ(w)} + \sum_{j=1}^{k} log(1 - \frac{exp(h^T v'_{\tilde{w}_{ij}})}{exp(h^T v'_{\tilde{w}_{ij}}) + kQ(\tilde{w}_{ij})})] \tag{35}$$

NCE方法有非常完美的理论证明：随着噪声样本k的数量增加，NCE导数趋近于$softmax$函数的梯度。

Jozefowicz认为**NCE与IS的相似点**不仅在于它们都是基于采样的方法，而且相互之间联系非常紧密。NCE等价于解决二分类任务，他认为IS问题也可以用一个代理损失函数来描述：IS相当于用$softmax$和交叉熵损失函数来优化解决多分类问题。他觉得IS是多分类问题，可能更适用于自然语言的建模，因为它迭代更新受到数据和噪声样本的共同作用，而NCE的迭代更新则是分别作用。事实上，Jozefowicz等人选用IS作为语言模型并且取得了最佳的效果。

负采样

负采样(Negative Sampling)可以被认为是NCE的一种近似版本。我们之前也提到过，随着样本数量k的增加，NCE近似于softmax的损失。由于NEG的目标是学习高质量的词向量表示，而不是降低测试集的perplexity指标，于是NEG对NCE做了简化。

NEG也采用逻辑回归模型，使得训练集中词语的负对数似然最小。再回顾一下NCE的计算公式：

$$P(y=1|w,c)=\frac{exp(h^Tv_w^{'})}{exp(h^Tv_w^{'})+kQ(w)} \tag{36}$$

NEG与NCE的关键区别在于NEG以尽可能简单的方式来估计这个概率值。为此，上式中计算量最大的$kQ(w)$ 项被置为1，于是得到：

$$P(y=1|w,c)=\frac{exp(h^Tv_w^{'})}{exp(h^Tv_w^{'})+1} \tag{37}$$

当$k=|V|$并且Q是均匀分布时，$kQ(w)=1$成立。此时，NEG等价于NCE。我们将$kQ(w)$设置为1，而不是其它常数值的原因在于，$P(y=1|w,c)$可以改写为$sigmoid$函数的形式：

$$P(y=1|w,c)=\frac{1}{1+exp(-h^Tv_w^{'})} \tag{38}$$

如果我们再把这个等式代入之前的逻辑回归损失函数中，可以得到：

$$\begin{aligned} J_\theta &= -\sum_{w_i\in V}[log\frac{1}{1+exp(-h^Tv_w^{'})}+\sum_{j=1}^{k}log(1-\frac{1}{1+exp(-h^Tv_{\tilde{w}_{ij}}^{'})})] \\ &= -\sum_{w_i\in V}[log\frac{1}{1+exp(-h^Tv_w^{'})}+\sum_{j=1}^{k}log\frac{1}{1+exp(h^Tv_{\tilde{w}_{ij}}^{'})}] \\ &= -\sum_{w_i\in V}[log\,\sigma(h^Tv_w^{'})+\sum_{j=1}^{k}log\,\sigma(-h^Tv_{\tilde{w}_{ij}}^{'})] \end{aligned} \tag{39}$$

而且仅当$k=|V|$并且Q是均匀分布时，NEG才等价于NCE。在其它情况下，NEG只是近似于NCE，也就是说前者不会直接优化正确词语的对数似然，所以不适合用于自然语言建模。NEC更适用于训练词向量表示。

马尔可夫过程

MCMC(Markov Chain Monte Carlo)的基础理论为马尔可夫过程，在MCMC算法中，为了在一个指定的分布上采样，根据马尔可夫过程，首先从任一状态出发，模拟马尔可夫过程，不断进行状态转移，最终收敛到平稳分布。

参考链接

4.1 马尔可夫链

设X_t表示随机变量X在离散时间t时刻的取值。若该变量随时间变化的转移概率仅仅依赖于它的当前取值，即

$$P(X_{t+1}=s_j|X_0=s_0,X_1=s_1,\dots,X_t=s_t)=P(X_{t+1}=s_j|X_t=s_i) \tag{40}$$

也就是说状态转移的概率只依赖于前一个状态，称这个变量为马尔可夫变量。这个性质称为马尔可夫性质，具有马尔可夫性质的随机过程称为马尔可夫过程。

马尔可夫链指的是在一段时间内随机变量X的取值序列$(X_0,X_1,\dots,X_m)$，它们满足如上的马尔可夫性质。

4.2 转移概率

$$
\begin{aligned}
\pi_i^{(t+1)} &= P(X_{t+1} = s_i) \\
&= \sum_k P(X_{t+1} = s_i | X_t = s_k) \cdot P(X_t = s_k) \\
&= \sum_k P_{k,i} \cdot \pi_k^{(t)}
\end{aligned} \tag{41}
$$

假设状态的数目为n，则有：

$$
\begin{aligned}
\pi_{t+1} &= \pi_t \cdot P \\
\pi_t &= (\pi_1^{(t)}, \pi_2^{(t)}, \cdots, \pi_n^{(t)})
\end{aligned} \tag{42}
$$

4.3 马尔可夫链的平稳分布

对于马尔可夫链，需要注意以下的两点：

- 1、周期性：即经过有限次的状态转移，又回到了自身；
- 2、不可约：即两个状态之间相互转移；

如果一个马尔可夫过程既没有周期性，又不可约，则称为各态遍历的。

马氏链定理：如果一个非周期马氏链具有转移概率矩阵 P, 且它的任何两个状态是连通的，那么$\lim_{n\to\infty} p_{ij}^n$存在且与$i$无关，记 $\lim_{n\to\infty} p_{ij}^n = \pi$， 我们有

$$
\lim_{n\to\infty} P^n = \begin{bmatrix}
\pi(1) & \cdots & \pi(2) & \cdots & \pi(j) & \cdots \\
\pi(1) & \cdots & \pi(2) & \cdots & \pi(j) & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\pi(1) & \cdots & \pi(2) & \cdots & \pi(j) & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots
\end{bmatrix} \tag{43}
$$

$$
\pi_{(j)} = \sum_{i=0}^{\infty} \pi_{(i)} p_{i,j} \tag{44}
$$

$$
\pi = [\pi(1), \pi(2), \cdots, \pi(j), \cdots], \quad \sum_{i=0}^{\infty} \pi_i = 1 \tag{45}
$$

1. 举例：设转移概率矩阵为：

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

使用该状态转移概率矩阵,从任意一个随机概率分布开始,不断迭代进行状态转移,最终为稳定收敛到概率分布(0.28650138, 0.48852158, 0.22497704)

演示代码如下：

```
import numpy as np
import scipy as sp

# transition probability matrix
P = np.array([[0.65,0.28,0.07], [0.15, 0.67, 0.18], [0.12,0.36,0.52]])
```

```
6
7  # initial probability
8  pi = np.random.uniform(0.0,1.0,3)
9  # normalize to sum(pi) = 1
10 pi = pi / sum(pi)
11
12 # start to iteration and print distrubition which can show convergence
13 iter = 0
14 print iter,pi
15 while iter < 50:
16     pi = np.matmul(pi,P)
17     iter += 1
18     print iter,pi
```

1. 稳定分布与特征向量的关系:

稳定分布π是一个(行)向量，它的元素都非负且和为1，不随施加P操作而改变，定义为 $\pi P = \pi$。

那么：$P^T \pi^T = \pi^T$ ，

对比定义可以看出,这两个概念是相关的,并且$\pi = \frac{e}{\sum_i e_i}$是由($\sum_i \pi_i = 1$)归一化的转移矩阵$P$的左特征向量$e$的倍数，其特征值为1.

操作上:

1. 对P的转置进行特征值分解得到特征向量和特征值.
2. 最大的特征值应为1,其对应的特征向量为矩阵的第i列
3. 对特征向量进行归一化,可以得到该状态转移矩阵的稳定分布

演示代码如下:

```
1  # evd
2  w,v=np.linalg.eig(P.transpose())
3
4  # there must be an eigenvalue w[i] should be 1.
5  # all abs(eigenvalue) <= 1
6  # correspond eigenvector is the column of v[:,i]
7  # the eigenvector should be normalized to 1 as it should be probability distribution
8  idx = np.where(abs(w-1.0)<1e-9)[0][0]
9
10 print w[idx]
11 print v[:,idx]/sum(v[:,idx])
```

实际上nxn矩阵特征向量的一种求法就是用一个随机向量不断迭代与该矩阵相乘。

马氏链的收敛定理非常重要，所有的 MCMC(Markov Chain Monte Carlo) 方法都是以这个定理作为理论基础的。

对于给定的概率分布$p(x)$,我们希望能有便捷的方式生成它对应的样本。由于马氏链能收敛到平稳分布， 于是一个很的漂亮想法是：如果我们能构造一个转移矩阵为P的马氏链，使得该马氏链的平稳分布恰好是$p(x)$, 那么我们从任何一个初始状态x_0出发沿着马氏链转移, 得到一个转移序列$x_0, x_1, \ldots, x_n, x_{n+1}, \ldots$， 如果马氏链在第$n$步已经收敛了，于是我们就得到了$p(x)$的样本$x_n, x_{n+1}, \ldots$。

从初始概率分布 π_0 出发，我们在马氏链上做状态转移，记X_i的概率分布为π_i，则有

$$
\begin{aligned}
X_0 &\sim \pi_0(x) \\
X_i &\sim \pi_i(x) \\
\pi_i(x) &= \pi_{i-1}(x)P = \pi_0(x)P^n
\end{aligned} \tag{46}
$$

由马氏链收敛的定理，概率分布$\pi_i(x)$将收敛到平稳分布π_x。假设到第n步的时候马氏链收敛，则有

$$
\begin{aligned}
X_0 &\sim \pi_0(x) \\
X_1 &\sim \pi_1(x) \\
&\cdots \\
X_n &\sim \pi_n(x) = \pi(x) \\
X_{n+1} &\sim \pi(x) \\
X_{n+2} &\sim \pi(x) \\
&\cdots
\end{aligned} \tag{47}
$$

所以 $X_n, X_{n+1}, X_{n+2}, \cdots \sim \pi(x)$都是同分布的随机变量，当然他们并不独立。如果我们从一个具体的初始状态 x_0 开始，沿着马氏链按照概率转移矩阵做跳转，那么我们得到一个转移序列 $x_0, x_1, x_2, \cdots x_n, x_{n+1} \cdots$ 由于马氏链的收敛行为，$x_n, x_{n+1}, \cdots$ 都将是平稳分布 $\pi(x)$ 的样本。

4.4 细致平稳条件

又称可反转马尔可夫链。可反转马尔可夫链类似于应用贝叶斯定理来反转一个条件概率：

$$
\begin{aligned}
Pr(X_n = i | X_{n+1} = j) &= \frac{Pr(X_n = i, X_{n+1} = j)}{Pr(X_{n+1} = j)} \\
&= \frac{Pr(X_n = i)\,\mathrm{Pr}(X_{n+1} = j | X_n = i)}{Pr(X_{n+1} = j)}
\end{aligned} \tag{48}
$$

以上就是反转的马尔可夫链。因而，如果存在一个 π，使得：$\pi_i p_{ij} = \pi_j p_{ji}$那么这个马尔可夫链就是可反转的. 这个条件也被称为细致平稳(detailed balance)条件. 对于所有的i求和：

$$
\sum_i \pi_i p_{ij} = \pi_j \tag{49}
$$

所以，对于可反转马尔可夫链，π 总是一个平稳分布。

注: 细致平稳条件为马尔可夫链有平稳分布的充分条件

设$p(x)$为目标分布, 马尔科夫状态转移概率矩阵为Q, 如果此时细致平稳条件不成立,即

$$
p(i)Q(i,j) \neq p(j)Q(j,i) \tag{50}
$$

可以对Q进行改造以满足细致平稳条件, 具体方法为引入α, 使下式成立.

$$
p(i)Q(i,j)\alpha(i,j) = p(j)Q(j,i)\alpha(j,i) \tag{51}
$$

当下面两式成立时，上式可成立：

$$
\begin{aligned}
\alpha(i,j) &= p(j)Q(j,i) \\
\alpha(j,i) &= p(i)Q(i,j)
\end{aligned} \tag{52}
$$

这样，我们就得到了我们的分布π(x)对应的马尔科夫链状态转移矩阵P，满足：

$$
Q'(i,j) = Q(i,j)\alpha(i,j) \tag{53}
$$

称$\alpha(i,j)$为接受概率。

于是我们把原来具有转移矩阵Q的一个很普通的马式链改造为了具有转移矩阵Q'的马式链，而Q'恰好满足细致平稳条件，由此马式链Q'的平稳分布就是$p(x)$!

为了使$Q(i,j)\alpha(i,j)$满足细致平稳条件. 一般来说$Q(i,j)\alpha(i,j)$也是不方便直接采样的. 实际的做法是采用拒绝采样方法,把$\alpha(i,j)$看作一个状态转移的接受概率. 从(0,1)均匀分布中做一个采样得到u,如果$u < \alpha(i,j)$则接受$Q(i,j)$采样出样本的状态转移,否则拒绝，保持原状态。

有了状态转移概率,现在我们可以开始做采样了

- $p(x)$为样本的目标概率分布
- t时刻马式链状态为$X_t = x_t$，采样$y \sim q(x|x_t)$
- 从(0, 1)均匀分布U采样u
- 如果$u < \alpha(x_t, y) = p(y)q(x_t|y)$，则接受转移$x_t \rightarrow y$，即$X_{t+1} = y$
- 否则拒绝转移，即$X_{t+1} = x_t$

U分布的选择,对于连续随机变量,个人觉得高斯分布是个常用的选择.也就是t时刻的值为以t-1时刻的值为均值的高斯分布.

4.5 Metroplis-Hasting算法

M-H算法主要是解决接受率过低的问题, 回顾MCMC采样的细致平稳条件:

$$p(i)Q(i,j)\alpha(i,j) = p(j)Q(j,i)\alpha(j,i) \tag{54}$$

我们采样效率低的原因是 $\alpha(i,j)$太小了，比如为0.1，而 $\alpha(j,i)$为0.2. 即:

$$p(i)Q(i,j) \times 0.1 = p(j)Q(j,i) \times 0.2 \tag{55}$$

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，但是细致平稳条件却仍然是满足的，即:

$$p(i)Q(i,j) \times 0.5 = p(j)Q(j,i) \times 1 \tag{56}$$

这样我们的接受率可以做如下改进，即:

$$\alpha(i,j) = min(\frac{p(j)Q(j,i)}{p(i)Q(i,j)}, 1) \tag{57}$$

在MCMC的基础上,只需将第4步中的计算接受率的公式改为上式即可.

MCMC因为提议分布和接受/拒绝机制使得其能构造出一个满足细致平稳的转移内核（也就是Markov Chain里面的转移概率矩阵）。这个Markov Chain是遍历的，他的稳态分布刚好是我们想要的后验分布。然后配上遍历定理，我们就能为所欲为了。

5 Gibbs Sampling

吉布斯采样（Geman and Geman, 1984）是一个简单的并且广泛应用的马尔科夫链蒙特卡罗 算法，可以被看做Metropolis-Hastings算法的一个具体的情形。

Gibbs sampling 的特点是，每个step更新变量中的一个维度，比如，$X_i = x_{i1}, x_{i2}, x_{i3}, \cdots$, 则$X_i = x_{i1}, x_{i2}, x'_{i3}, \cdots$, 仅更新第三个维度。多个维度交替更新。

注意Gibbs Sampler并不满足细致平稳，他只满足global balance。