法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

采样

小象学院 ChinaHadoop.cn　邹博

主要内容

- 采样的意义
- 拒绝采样方法
- 马尔科夫链
  - 细致平稳条件
- Matropolis-Hastings算法
- Gibbs采样

思考：LDA的迭代

$$p\left(z_i = k \mid \vec{z}_{\neg i}, \vec{w}\right) \propto \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t}\left(n_{m,\neg i}^{(k)} + \alpha_k\right)$$

$$\begin{aligned}
p(z_i=k|\vec{z}_{\neg i}, \vec{w}) &= \frac{p(\vec{w}, \vec{z})}{p(\vec{w}, \vec{z}_{\neg i})} = \frac{p(\vec{w}|\vec{z})}{p(\vec{w}_{\neg i}|\vec{z}_{\neg i})p(w_i)} \cdot \frac{p(\vec{z})}{p(\vec{z}_{\neg i})} \\
&\propto \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{n}_{z,\neg i} + \vec{\beta})} \cdot \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{n}_{m,\neg i} + \vec{\alpha})} \\
&= \frac{\Gamma(n_k^{(t)} + \beta_t)\,\Gamma(\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t)}{\Gamma(n_{k,\neg i}^{(t)} + \beta_t)\,\Gamma(\sum_{t=1}^{V} n_k^{(t)} + \beta_t)} \cdot \frac{\Gamma(n_m^{(k)} + \alpha_k)\,\Gamma(\sum_{k=1}^{K} n_{m,\neg i}^{(k)} + \alpha_k)}{\Gamma(n_{m,\neg i}^{(k)} + \alpha_k)\,\Gamma(\sum_{k=1}^{K} n_m^{(k)} + \alpha_k)} \\
&= \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t} \cdot \frac{n_{m,\neg i}^{(k)} + \alpha_k}{[\sum_{k=1}^{K} n_m^{(k)} + \alpha_k] - 1} \\
&\propto \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t}(n_{m,\neg i}^{(k)} + \alpha_k)
\end{aligned}$$

Code

```python
def gibbs_sampling(z, m, i, nt, nd, nt_sum, nd_sum, term):
    topic = z[m][i]            # 当前主题
    nt[term][topic] -= 1       # 去除当前词
    nd[m][topic] -= 1
    nt_sum[topic] -= 1
    nd_sum[m] -= 1

    topic_alpha = topic_number * alpha
    term_beta = len(dic) * beta
    p = [0 for x in range(topic_number)] # p[k]:属于主题k的概率
    for k in range(topic_number):
        p[k] = (nd[m][k] + alpha) / (nd_sum[m] + topic_alpha)\
               * (nt[term][k] + beta) / (nt_sum[k] + term_beta)
        if k >= 1:           # 顺手转换成累加概率
            p[k] += p[k-1]
    gs = random.random() * p[topic_number-1]     # 采样
    new_topic = 0
    while new_topic < topic_number:
        if p[new_topic] > gs:
            break
        new_topic += 1

    nt[term][new_topic] += 1
    nd[m][new_topic] += 1
    nt_sum[new_topic] += 1
    nd_sum[m] += 1
    z[m][i] = new_topic     # 新主题
```

为什么要研究采样？

- 根据采样结果估算分布的参数，完成参数学习。
  - 前提：模型已经存在，但参数未知；
  - 方法：通过采样的方式，获得一定数量的样本，从而学习该系统的参数。
  - 例：投硬币试验中，进行N次试验，n次朝上，N-n次朝下——可以认为，是进行了N次(独立)抽样。
  - 假定朝上的概率为p，使用对数似然函数作为目标函数：

$$f(n|p)=\log\left(p^n(1-p)^{N-n}\right)\xrightarrow{\Delta}h(p)=\log\left(p^n(1-p)^{N-n}\right)$$

$$\frac{\partial h(p)}{\partial p}=\frac{n}{p}-\frac{N-n}{1-p}\xrightarrow{\Delta}0\Rightarrow p=\frac{n}{N}$$

附：Bernoulli版本的大数定理

- 一次试验中事件A发生的概率为p；重复n次独立试验中，事件A发生了n_A次，则p、n、n_A的关系满足：

  对于任意整数ε，

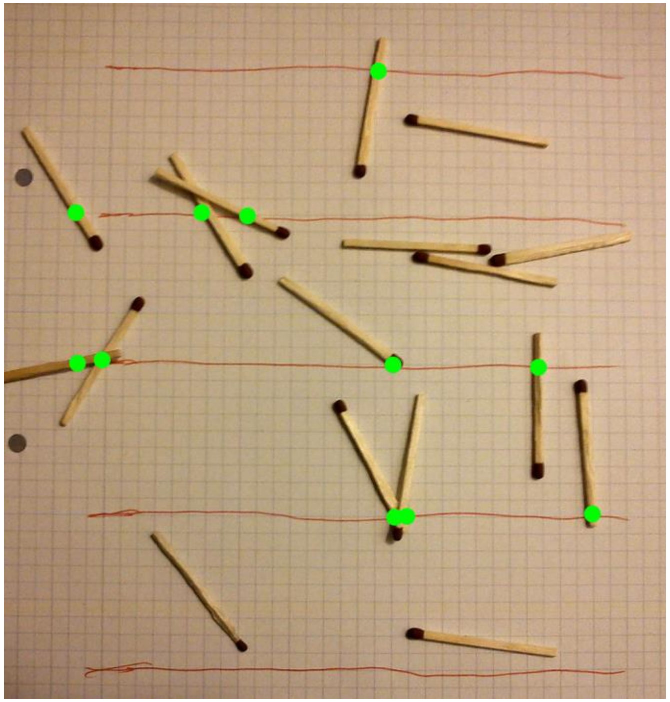
$$\lim_{n \to \infty} P\left\{ \left| \frac{n_A}{n} - p \right| < \varepsilon \right\} = 1$$

应用Bernoulli版本的大数定理

- 一般的说，上述结论可以直接推广：频率的极限为概率：$p = {}^{n}\!/_{N}$
- 将上述二项分布扩展成多项分布，如K项分布：$p_i = \frac{n_i}{N}$
  - 从而得到K项分布的参数：$p = \left(\frac{n_1}{N}, \frac{n_2}{N} \cdots \frac{n_k}{N}\right)$
- 在主体模型LDA中，每个文档的主题分布和每个主题的词分布都是多项分布，如果能够通过采样的方式获得关于参数的一定数量的样本，即可估算主题分布和词分布的参数，即可完成参数学习。

例：Buffon's Needle

- 桌面上有距离为a的若干平行线，将长度为L的针随机丢在桌面上，则这根针与平行线相交的概率是多少？
  - 假定L＜a



概率计算

- 记投针中点到最近横线的距离为y，则y∈[0,a/2]，投针是随机的，y为均匀分布：

$$f(y)=\begin{cases}\frac{2}{a} & 0\le y\le\frac{a}{2}\\ 0 & elsewhere\end{cases}$$

- 假定横线向右为正向，记投针与横线正向的角为θ，则θ∈[0,π]为均匀分布。

$$f(\theta)=\begin{cases}\frac{1}{\pi} & 0\le\theta\le\pi\\ 0 & elsewhere\end{cases}$$

- 投针与横线有交点，即：$y\le\frac{L}{2}\sin\theta$

蒙特卡洛模拟

- 有交点的概率：

$$P(X)=\int_0^{\pi}\int_0^{\frac{L}{2}\sin\theta} f(y,\theta)dyd\theta=\int_0^{\pi}\int_0^{\frac{L}{2}\sin\theta} f(y)f(\theta)dyd\theta$$

$$=\int_0^{\pi}\int_0^{\frac{L}{2}\sin\theta}\frac{2}{a}\cdot\frac{1}{\pi}dyd\theta=\frac{2L}{a\pi}$$

- 如果做n次试验，得到k次相交。则频率是$\frac{k}{n}$
- 从而：$\frac{2L}{a\pi}\approx\frac{k}{n}\Rightarrow\pi\approx\frac{2Ln}{ak}$

Buffon's Needle

- 公式：$\frac{2L}{a\pi} \approx \frac{k}{n} \Rightarrow \pi \approx \frac{2Ln}{ak}$

试验者	时间	投掷次数	相交次数	π的试验值
Wolf	1850年	5000	2532	3.1596
Smith	1855年	3204	1218.5	3.1554
C.De Morgan	1860年	600	382.5	3.137
Fox	1884年	1030	489	3.1595
Lazzerini	1901年	3408	1808	3.141593
Reina	1925年	2520	859	3.1795

Code

```
double Buffon(double a, double L)    //横线之间的距离，针长度
{
    double y;          //到最近的横线的距离
    double theta;      //针的倾角
    int c = 0;         //相交次数
    int n = 1000000;       //实验次数
    for(int i = 0; i < n; i++)
    {
        y = Rand(a/2);
        theta = Rand(PI);
        if(y < L*sin(theta)/2)   //相交
            c++;
    }
    return 2 * (double)(n * L) / (double)(c * a);
}
```

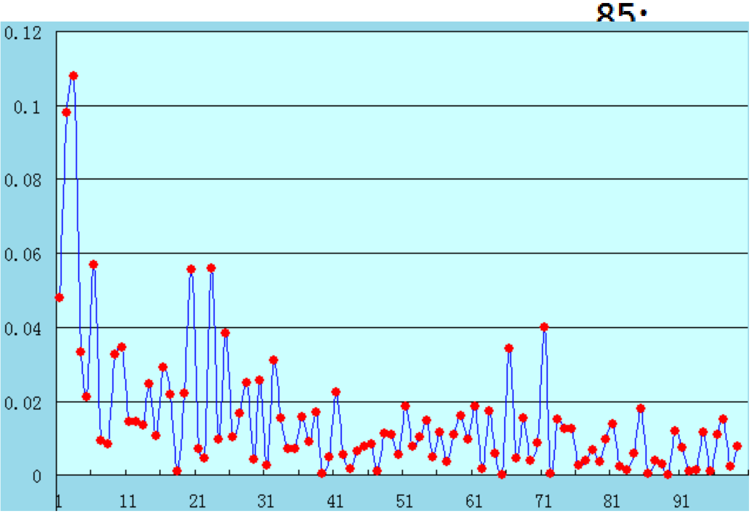
Code2

```
int _tmain(int argc, _TCHAR* argv[])
{
    double a = 100;      //横线的间隔
    double L;            //针的长度
    double pi;           //估算值
    double avg = 0;      //估算值的均值
    double count = 0;    //计算次数
    for(L = a; L > 1; L -= 1)
    {
        pi = Buffon(a, L);
        cout << pi << '\n';
        avg += pi;
        count++;
    }
    avg /= count;
    cout << avg << '\n';
    return 0;
}
```

```
double Buffon(double a, double L)    //横线的间隔、针的长度
{
    double X = a * 1000;     //取足够大的信纸
    double Y = a * 1000;
    int N = 100000;      //进行10万次投针试验
    int c = 0;
    double x1, x2, y1, y2;
    double d,y;
    for(int i = 0; i < N; i++)
    {
        x1 = Rand(X);
        y1 = Rand(Y);
        x2 = Rand(X);
        y2 = Rand(Y);
        d = sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
        y = (y2 - y1) * L / d + y1;
        if((int)(y1/a) != (int)(y/a))
            c++;
    }
    return 2 * L * N / (a * c);
}
```

效果/措施

100:	3.13386	0.0077297	27:	3.13134	0.0102502
99:	3.14405	0.00246222	26:	3.18004	0.0384465
98:	3.12665	0.0149476	25:	3.13205	0.00954555
97:	3.13065	0.0109446	24:	3.19744	0.0558494
96:	3.14054	0.00105353	23:	3.13693	0.00465867
95:	3.12994	0.0116506	22:	3.1487	0.00711208
94:	3.14308	0.00148422	21:	3.19708	0.0554843
93:	3.14274	0.00114863	20:	3.11964	0.0219545
92:	3.13426	0.0073304	19:	3.14257	0.000980949
91:	3.12973	0.0118637	18:	3.16344	0.021852
90:	3.14164	4.27422e-005	17:	3.17075	0.0291618
89:	3.14471	0.00311699	16:	3.13112	0.0104772
88:	3.14538	0.00379203	15:	3.11688	0.0247095
87:	3.14113	0.000458235	14:	3.15493	0.0133369
86:	3.12352	0.0180682	13:	3.15611	0.0145132
85:	3.14739	0.00579779	12:	3.12704	0.0145568
	3.14019	0.00140574	11:	3.10691	0.0346868
	3.13936	0.00223149	10:	3.10897	0.0326233
	3.12774	0.0138511	9:	3.13316	0.00843338
	3.13201	0.0095843	8:	3.15085	0.00925414
	3.13799	0.0035994	7:	3.19854	0.0569452
	3.1348	0.00679642	6:	3.16289	0.0212961
	3.14567	0.004076	5:	3.10849	0.0331065
	3.13895	0.00264318	4:	3.03375	0.107842
	3.12892	0.0126686	3:	3.23974	0.0981482
	3.15424	0.0126498	2:	3.09358	0.0480118
74:	3.15666	0.0150673	AVG:	3.14202	0.000430937

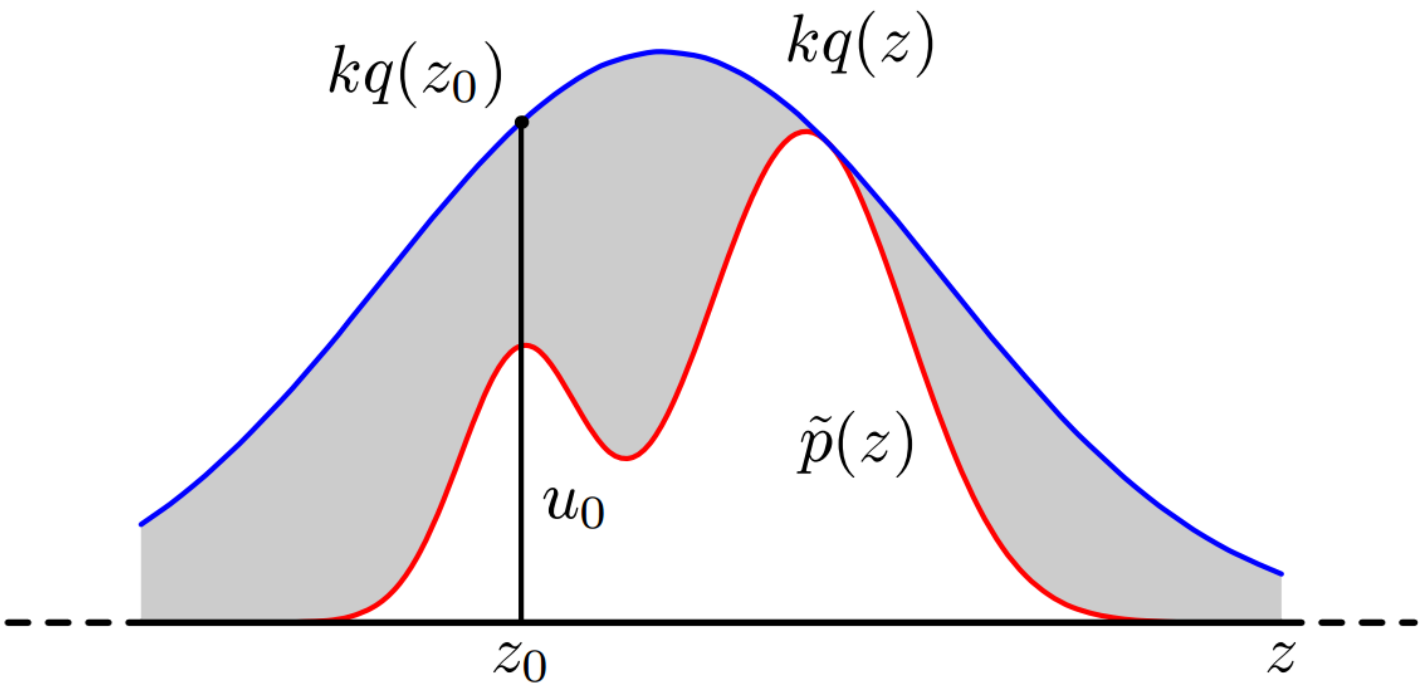


采样算法

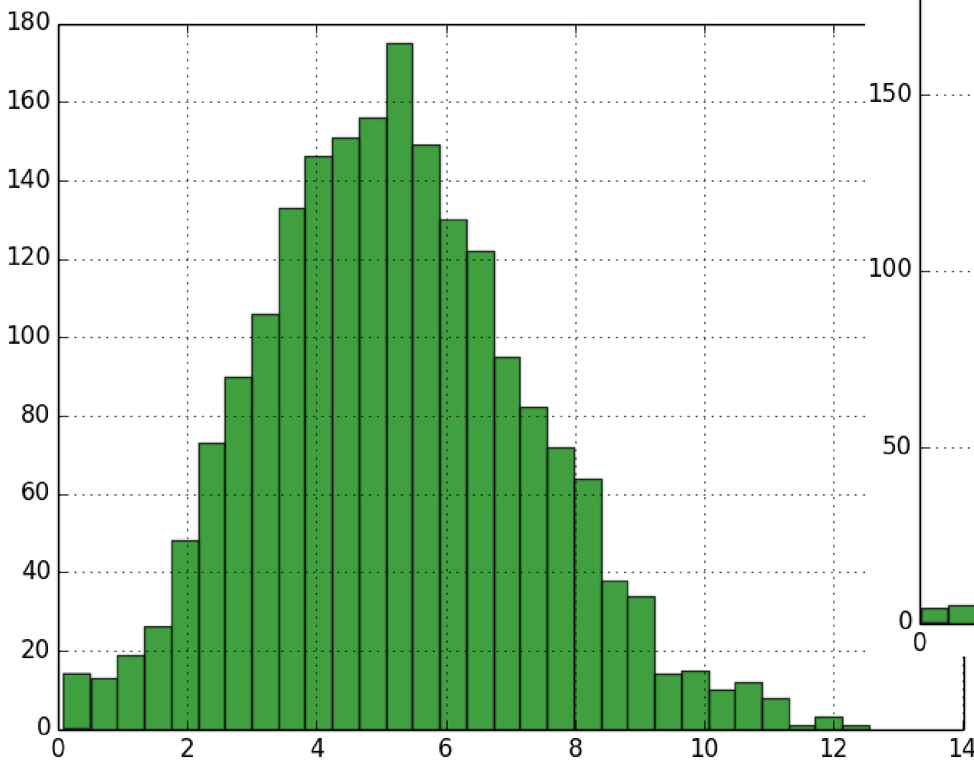
- 现需要对概率密度函数$f(x)$的参数进行估计，若已知的某概率密度函数$g(x)$容易采样获得其样本，可以如何估计$f(x)$的参数？
  - 若离散分布，则$f(x)$为概率分布律。

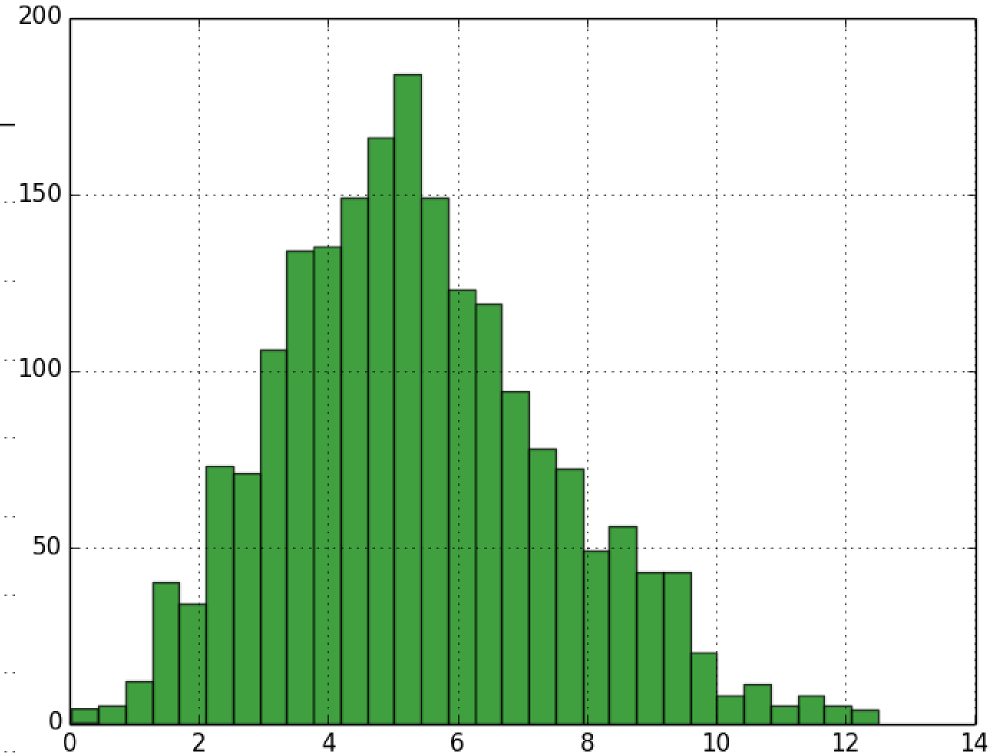
Rejection sampling

- 如：根据均匀分布得到正态分布。



MCMC采样：模拟Poisson分布





Matropolis-Hastings算法

- 假定t时刻$X^{(t)}=x^{(t)}$，采取如下策略采样$X^{(t+1)}$
- 在给定$x^{(t)}$的条件分布$g(x|x^{(t)})$中采样一个值x^*
- 计算M-H率：$R\left(x^{(t)},x^*\right)=\dfrac{f(x^*)g(x^{(t)}|x^*)}{f(x^{(t)})g(x^*|x^{(t)})}$ $\quad\begin{cases}f\left(x^{(t)}\right)>0\\ g\left(x^*|x^{(t)}\right)>0\end{cases}$
- 则t+1时刻的X值$x^{(t+1)}$为：

$$X^{(t+1)}=\begin{cases}x^*, & t\le R\left(x^{(t)},x^*\right), t\sim Uniform(0,1)\\ x^{(t)}, & otherwise\end{cases}$$

- 最终得到的序列收敛于f(x)分布。

分析MH率 $R\left(x^{(t)}, x^*\right)=\dfrac{f(x^*)g(x^{(t)}\mid x^*)}{f(x^{(t)})g(x^{(t)}\mid x^*)}$

- 两样本x_1, x_2，不妨假定$f\left(x_2\right)g\left(x_1 \mid x_2\right)\geq f\left(x_1\right)g\left(x_2 \mid x_1\right)$
- 则：

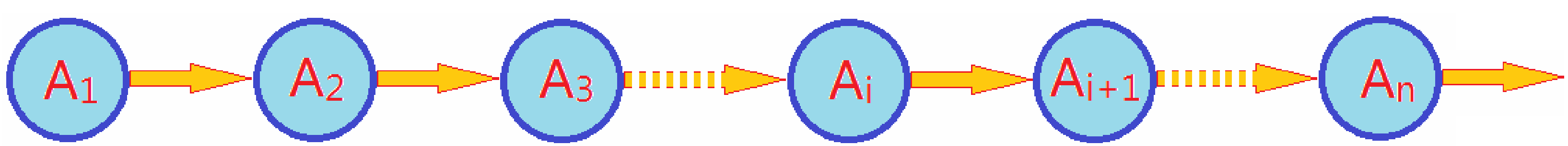
$$\begin{aligned}
& p\left(X^{(t)}=x_1, X^{(t+1)}=x_2\right) \\
& =f\left(x_1\right)\cdot g\left(x_2 \mid x_1\right)\cdot R\left(x_1, x_2\right) \\
& =f\left(x_1\right)\cdot g\left(x_2 \mid x_1\right)
\end{aligned}$$

$$\begin{aligned}
& p\left(X^{(t)}=x_2, X^{(t+1)}=x_1\right) \\
& =f\left(x_2\right)\cdot g\left(x_1 \mid x_2\right)\cdot R\left(x_2, x_1\right) \\
& =f\left(x_2\right)\cdot g\left(x_1 \mid x_2\right)\cdot\frac{f\left(x_1\right)\cdot g\left(x_2 \mid x_1\right)}{f\left(x_2\right)\cdot g\left(x_1 \mid x_2\right)} \\
& =f\left(x_1\right)\cdot g\left(x_2 \mid x_1\right)
\end{aligned}$$

- 若在某个时刻t，使得$x^{(t)}$满足f(x)的分布，则后面的采样值服从f(x)的分布。

重述采样

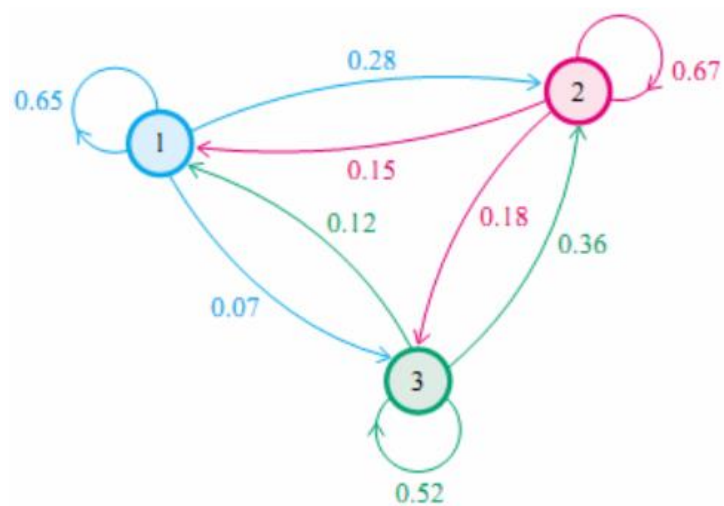
- 采样：给定概率分布p(x)，如何在计算机中生成它的若干样本？
- 方法：马尔科夫链模型
- 考虑某随机过程π，它的状态有n个，用1~n表示。记在当前时刻t时位于i状态，它在t+1时刻位于j状态的概率为P(i,j)=P(j|i)：即状态转移的概率只依赖于前一个状态。



举例

- 假定按照经济状况将人群分成上、中、下三个阶层，用1、2、3表示。假定当前处于某阶层只和上一代有关，即：考察父代为第i阶层，则子代为第j阶层的概率。假定为如下转移概率矩阵：

$$
\begin{array}{cc}
P & \text{子代} \\
\text{父代} & \begin{bmatrix} 0.65 & 0.28 & 0.07 \\ 0.15 & 0.67 & 0.18 \\ 0.12 & 0.36 & 0.52 \end{bmatrix}
\end{array}
$$



概率转移矩阵

- 显然，第n+1代中处于第j个阶层的概率为：

$$\pi\left(X_{n+1}=j\right)=\sum_{i=1}^{n}\pi\left(X_{n}=i\right)\cdot P\left(X_{n+1}=j \mid X_{n}=i\right)$$

- 因此，矩阵P即贝叶斯网络中描述的(条件)概率转移矩阵。
  - 第i行元素表示：在上一个状态为i时的分布概率，即：每一行元素的和为1。

初始概率π= [0.21, 0.68, 0.1]的迭代结果

第n代	第1阶层	第2阶层	第3阶层
0	0.21	0.68	0.11
1	0.252	0.554	0.194
2	0.27	0.512	0.218
3	0.278	0.497	0.225
4	0.282	0.49	0.226
5	0.285	0.489	0.225
6	0.286	0.489	0.225
7	0.286	0.489	0.225
8	0.286	0.488	0.225
9	0.286	0.489	0.225
10	0.286	0.489	0.225

初始概率π= [0.75, 0.15, 0.1]的迭代结果

第n代	第1阶层	第2阶层	第3阶层
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	0.22
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

马尔科夫随机过程的平稳分布

- 初始概率不同，但经过若干次迭代，π最终稳定收敛在某个分布上。
- 转移概率矩阵P的性质，而非初始分布的性质。事实上，上述矩阵P的n次幂，每行都是(0.286,0.489,0.225)，n>20
- 如果一个非周期马尔科夫随机过程具有转移概率矩阵P，且它的任意两个状态都是连通的，则 $\lim_{n\to\infty} P_{ij}^{n}$ 存在，记做 $\lim_{n\to\infty} P_{ij}^{n} = \pi(j)$。

马尔科夫随机过程的平稳分布

- 事实上，下面两种写法等价：

$$\lim_{n\to\infty} P_{ij}^{n} = \pi(j) \qquad \lim_{n\to\infty} P^{n} = \begin{bmatrix} \pi(1) & \pi(2) & \dots & \pi(n) \\ \pi(1) & \pi(2) & \dots & \pi(n) \\ \vdots & \vdots & \ddots & \vdots \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{bmatrix}$$

- 同时，若某概率分布πP=π，说明
  - 该多项分布π是状态转移矩阵P的平稳分布；
  - 线性方程xP=x的非负解为π，而P^n唯一，因此π是线性方程xP=x的唯一非负解。

马尔科夫随机过程与采样

- 上述平稳分布的马尔科夫随机过程对采样带来很大的启发：对于某概率分布π，生成一个能够收敛到概率分布π的马尔科夫状态转移矩阵P，则经过有限次迭代，一定可以得到概率分布π。
- 该方法可使用Monte Carlo模拟来完成，即MCMC (Markov Chain Monte Carlo)。

细致平稳条件

- 从稳定分布满足πP=π可以抽象出如下定义：
- 如果非周期马尔科夫过程的转移矩阵P和分布π(x)满足

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

- 则π(x)是马尔科夫过程的平稳分布。上式又被称作细致平稳条件 (detailed balance condition)。
  - P(i,j)为矩阵P的第i行第j列，其意思为前一个状态为i时，后一个状态为j的概率：即P(j|i)，因此，有时也写成P(i→j)
  - 细致平稳的理解：根据定义，对于任意两个状态i，j，从i转移到j的概率和从j转移到i的概率相等。可直观的理解成每一个状态都是平稳的。

细致平稳条件和平稳分布的关系

- 根据马尔科夫过程的定义：$\pi(j) = \sum_{i=1}^{n} \pi(i) \cdot P(i, j)$
- 根据细致平稳条件：

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

- 得：

$$\pi(j) = \sum_{i=1}^{n} \pi(j) \cdot P(j, i)$$

- 从而：

$$\pi = \pi \cdot P$$

设定接受率

- 假定当前马尔科夫过程的转移矩阵为Q，对于给定分布p，一般的说，$p(i)q(i,j) \neq p(j)q(j,i)$
- 通过加入因子α的方式，使得上式满足细致平稳条件 $p(i)q(i,j)\alpha(i,j) = p(j)q(j,i)\alpha(j,i)$
- 满足等式的因子α有很多，根据对称性，可以取：$\alpha(i,j) = p(j)q(j,i)$，$\alpha(j,i) = p(i)q(i,j)$
- 根据接受率α改造转移矩阵Q：

$$p(i)\underline{\underline{q(i,j)\alpha(i,j)}} = p(j)\underline{\underline{q(j,i)\alpha(j,i)}}$$

MCMC：Metropolis-Hastings算法

- 根据需要满足的细致平稳条件

$$p(i)q(i,j)\alpha(i,j)=p(j)q(j,i)\alpha(j,i)$$

- 若令$\alpha(j,i)=1$，则有：$p(i)q(i,j)\alpha(i,j)=p(j)q(j,i)$
- 从而：

$$\alpha(i,j)=\frac{p(j)q(j,i)}{p(i)q(i,j)}$$

- 将接受率置为恒小于1，从而

$$\alpha(i,j)=\min\left(\frac{p(j)q(j,i)}{p(i)q(i,j)},1\right)$$

Metropolis-Hastings算法

- 初始化马尔科夫过程初始状态$I=i_0$
- 对t=0,1,2,3…
  - 第t时刻马尔科夫过程初始状态i_t,采样$q=q(j|i_t)$
  - 从均匀分布中采样$u\in[0,1]$
  - 如果 $$u<\alpha(i,j)=\min\left(\frac{p(j)q(j,i)}{p(i)q(i,j)},1\right)$$

    则接受状态j，即$i_{t+1}=j$

    否则，不接受状态j，即$i_{t+1}=i$

改造MCMC算法

- MCMC有一定的拒绝率。
- 若需要采样二维联合分布p(x,y)，固定x，得

$$p(x_1,y_1)\alpha_{x_1}(y_1,y_2)=p(x_1,y_2)\alpha_{x_1}(y_2,y_1)$$
$$\Rightarrow p(x_1)p(y_1\,|\,x_1)\alpha_{x_1}(y_1,y_2)=p(x_1)p(y_2\,|\,x_1)\alpha_{x_1}(y_2,y_1)$$
$$\Rightarrow \alpha_{x_1}(y_1,y_2)=p(y_2\,|\,x_1)\text{ , }\alpha_{x_1}(y_2,y_1)=p(y_1\,|\,x_1)$$
$$\Rightarrow \alpha_{x_1}(y_{cur},y_{other})=p(y_{other}\,|\,x_1)\text{ , }\alpha_{x_1}(y_{other},y_{cur})=p(y_{cur}\,|\,x_1)$$

  - 若固定y，可得到对偶的结论。

二维Gibbs采样算法

- 由 $\begin{cases} \alpha_{x_1}(y_{cur}, y_{other}) = p(y_{other} \mid x_1) \\ \alpha_{y_1}(x_{cur}, x_{other}) = p(x_{other} \mid y_1) \end{cases}$
- 很容易得到二维Gibbs采样算法：
  - 随机初始化(X,Y)=(x_0,y_0)
  - 对t=0,1,2…，循环采样：

$$\begin{cases} y_{t+1} = p(y \mid x_t) \\ x_{t+1} = p(x \mid y_{t+1}) \end{cases}$$

将二维Gibbs采样推广到高维

- 随机初始化 $\left(X_1, X_2 \cdots X_n\right)=\left(x_1^{(0)}, x_2^{(0)} \cdots, x_n^{(0)}\right)$
- 对t=1,2…，循环采样直至收敛(burn-in)：

$$\begin{cases} x_1^{(t+1)} = p\left(x_1 \mid x_2^{(t)}, x_3^{(t)}, \cdots, x_n^{(t)}\right) \\ x_2^{(t+1)} = p\left(x_2 \mid x_1^{(t+1)}, x_3^{(t)}, \cdots, x_n^{(t)}\right) \\ \cdots \\ x_i^{(t+1)} = p\left(x_i \mid x_1^{(t+1)}, \cdots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \cdots, x_n^{(t)}\right) \\ \cdots \\ x_{n-1}^{(t+1)} = p\left(x_{n-1} \mid x_1^{(t+1)}, x_2^{(t+1)}, \cdots, x_{n-2}^{(t+1)}, x_n^{(t)}\right) \\ x_n^{(t+1)} = p\left(x_n \mid x_1^{(t+1)}, x_2^{(t+1)}, \cdots, x_{n-1}^{(t+1)}\right) \end{cases}$$

  - 显然，主题模型LDA中采样即采取以上策略。

用采样改造EM算法

Repeat until convergence {

(E-step) For each i, set

$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta).$$

(M-step) Set

$$\theta := \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}$$

}

用采样改造EM算法

- 在EM算法中，E-Step求出隐变量的条件概率，从而给出期望Q，M-Step将目标函数Q求极大值，期望Q为：

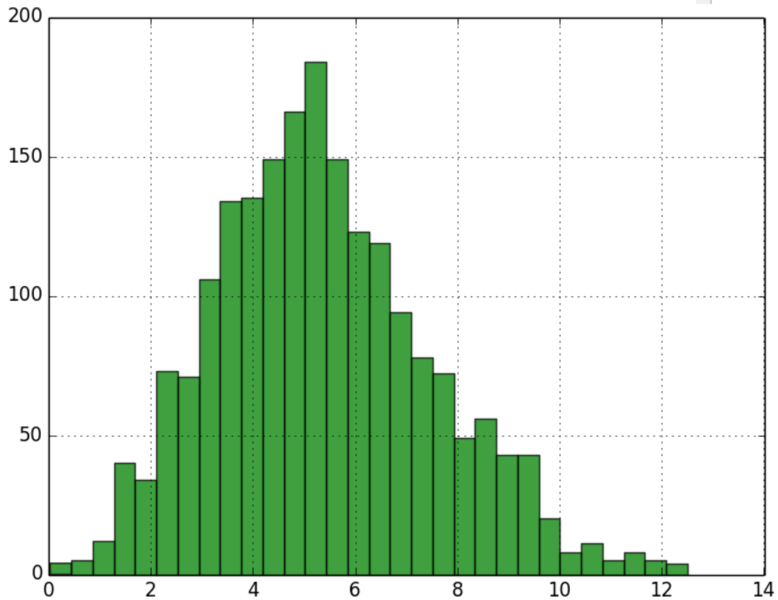
$$Q\left(\theta,\overline{\theta}\right)=\int p\left(Z \mid X,\overline{\theta}\right)\ln p\left(Z,X \mid \theta\right)dZ$$

- 显然，这仍然可以使用采样的方式近似得到：

$$Q\left(\theta,\overline{\theta}\right)\approx\frac{1}{L}\sum_{i=1}^{L}\ln p\left(Z^{(l)},X \mid \theta\right)$$

  - 这种EM算法称为MC-EM算法(Monte Carlo EM)
- 极限情况：若MC-EM算法的期望Q的估计，仅采样一个样本，则称之为随机EM算法(stochastic EM algorithm)。

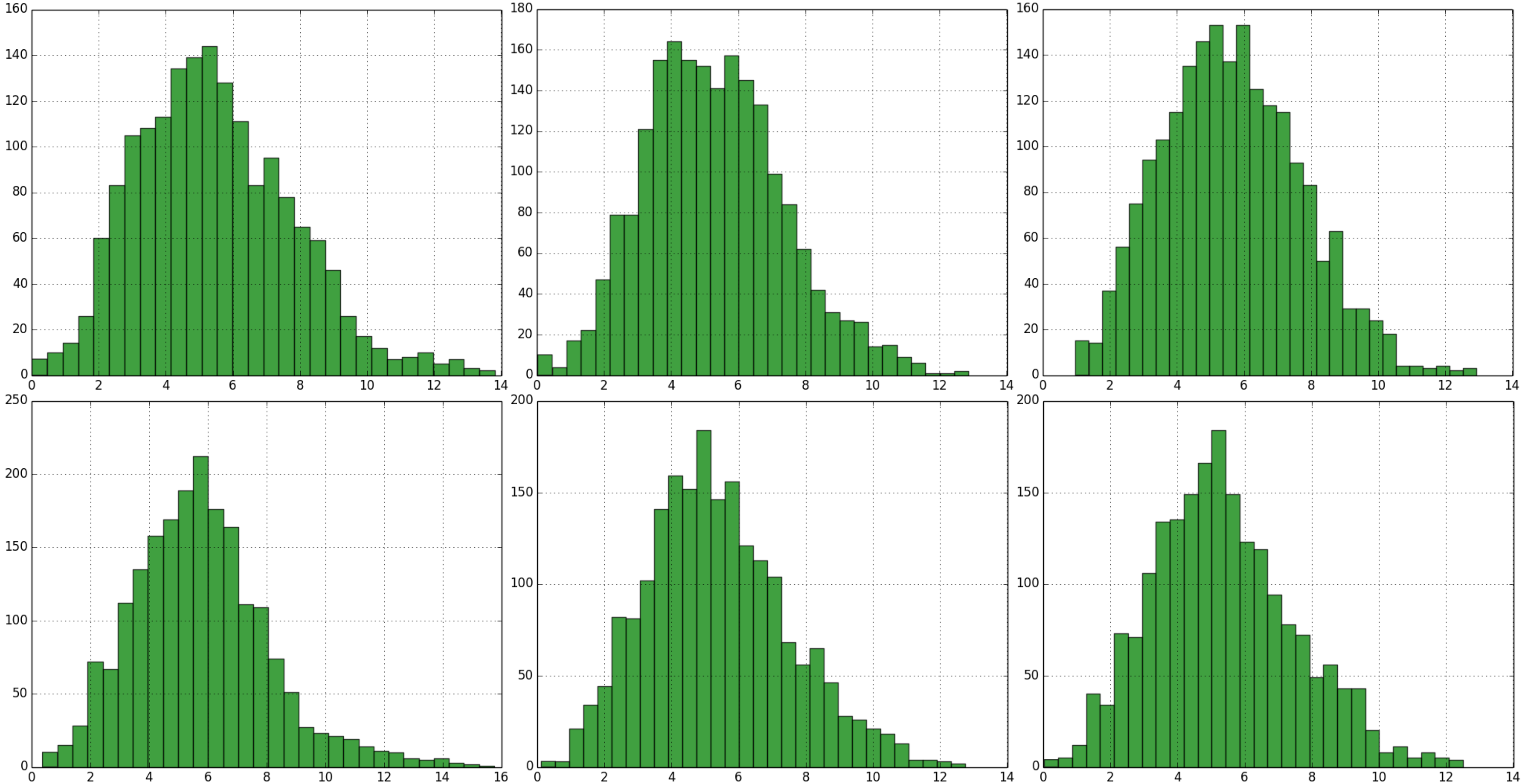
Code




```python
if __name__ == "__main__":
    s = []
    x1 = -1
    while x1 < 0:
        x1 = np.random.randn()
    for i in range(10000):
        x2 = np.random.randn() + x1
        while x2 < 0:
            x2 = np.random.randn() + x1
        f1 = f(x1)
        f2 = f(x2)
        g1 = g(x1, x2, 1)   # N(x2,1)中，x1的概率密度
        g2 = g(x2, x1, 1)   # N(x1,1)中，x2的概率密度
        mh = f2 * g1 / (f1 * g2)
        t = np.random.random()
        if t < mh:
            x1 = x2
            add(s, x2)
    plt.hist(s, 30, color='g', alpha=0.75)
    plt.grid(True)
    plt.show()
```

MCMC收敛性：2000

2000	3000	4000
6000	8000	10000



参考文献

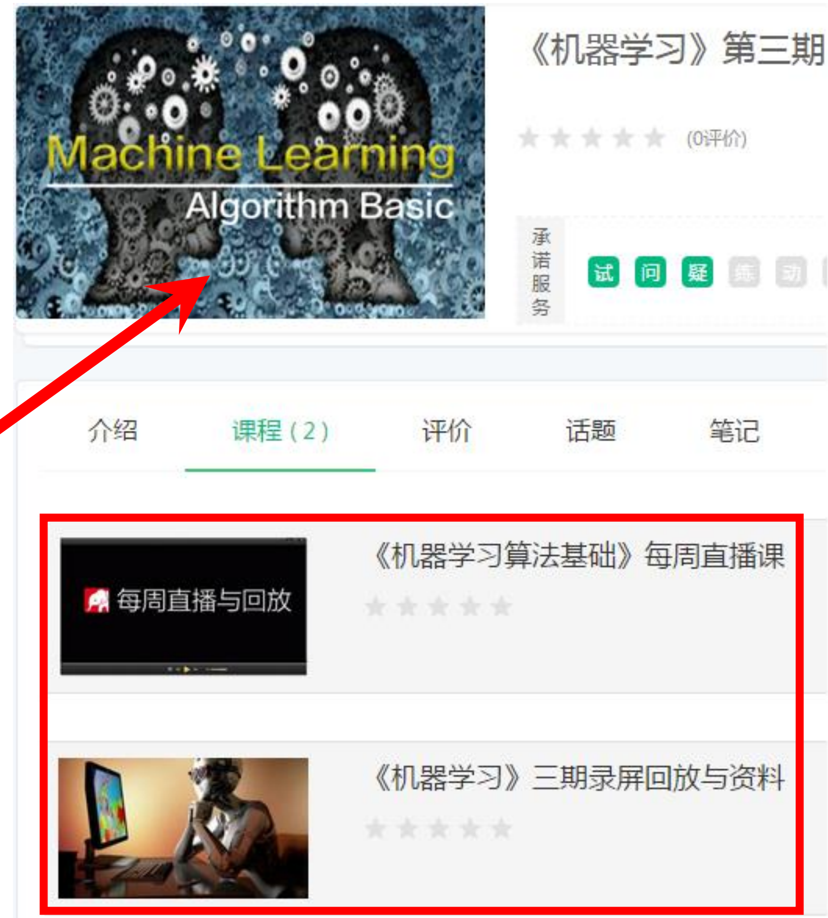
- Christopher M. Bishop. *Pattern Recognition and Machine Learning Chapter 10*. Springer-Verlag, 2006
- https://www.zybuluo.com/Hederahelix/note/101859
- http://thexbar.me/2014/11/07/reject-sample/
- http://www.cnblogs.com/daniel-D/p/3388724.html

我们在这里

- http://wenda.ChinaHadoop.cn
  - 视频/课程/社区
- 微博
  - @ChinaHadoop
  - @邹博_机器学习
- 微信公众号
  - 小象
  - 大数据分析挖掘

课程资源

- 直播课的入口
- 录播视频和讲义资料



感谢大家！

恳请大家批评指正！