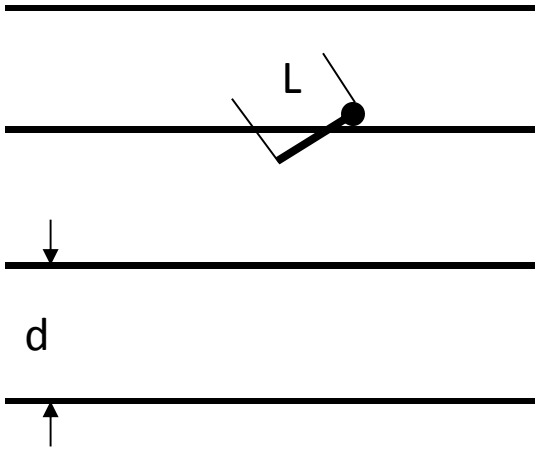
第5-3章 MCMC

- 引言
- Monte Carlo 近似计算
- 伪随机数生成
- 理论基础
- MCMC算法
- MCMC应用

Buffon投针实验(1777)



设针与平行线的夹角为ϕ，针的中点与最近的平行线的距离为X，相交条件为:

$$X \le \frac{L\sin\phi}{2}$$

$$p = \Pr\{X \le \frac{L\sin\phi}{2}\} = \frac{2}{d\pi}\int_0^{\pi}\left(\int_0^{\frac{L\sin\phi}{2}} dx\right)d\phi = \frac{2L}{d\pi}$$

Buffon实验

学者	年代	l/d	总次数	成功次数	π 的估计
Wolf	1850	0.8	5000	2532	3.15956
Smith	1855	0.6	3204	1218	3.1554
Fox	1884	0.75	1030	489	3.1595
Lazzarini	1907	0.833	3408	1808	3.14159292

Monte Carlo方法

- Von Neumann
- S.Ulam (1946)
- N.Metropolis (1953)
- Hasting (1970)

- Monte Carlo (Monaco), famous for its gambling casino.

Monte Carlo 近似计算

- 例: 求 π 的近似值
- 考虑在单位方块上均匀分布的二元随机变量 (ξ, η),

$$Pr(\xi^2 + \eta^2 < 1) = \frac{\pi}{4}$$

- 如果生成一组样本

$$(\xi_1, \eta_1), \cdots, (\xi_n, \eta_n),$$

- 其落在单位圆内的频率v_n就是 $\frac{\pi}{4}$ 的估计值

Monte Carlo 近似计算

- 利用 Chebechev 不等式, 立刻可以得到要保证误差不超过一个 给定值 ε 的概率小于 α , 应把 n 取多大

$$Pr(\left|v_n - \frac{\pi}{4}\right| \geq \epsilon) \leq \frac{D(v_1)}{n\epsilon^2}$$

Monte Carlo 积分近似计算

- Monte Carlo方法计算积分

$$I = \int_a^b f(x)p(x)dx = E_p[f(X)]$$

- 随机抽取p(x)的样本

$$X_1, X_2, \cdots, X_n \sim p(x)$$

- 积分近似

$$\hat{I}_n = \frac{1}{n}\sum_i f(x_i)$$

收敛性与收敛速度

- 大数定理

$$I_n \to E_p[f(X)], \quad n \to +\infty$$

- 当 $E_p[f^2(X)] < +\infty$ 时，

$$var(\hat{I}_n) = \frac{1}{n} \int_X \left(f(x) - E_p[f(X)]\right)^2 p(x) dx$$

Confidence Interval

- Sample variance

$$v_n = \frac{1}{n^2}\left(f(x_i) - \hat{I}_n\right)^2$$

- Convergence test

$$\frac{\hat{I}_n - E_p[f(X)]}{\sqrt{v_n}} \sim N(0,1)$$

With which the confidence interval can be given

积分计算的统计意义

- Bayesian分析中后验概率计算，通过极大后验概率进行分类

$$P(\theta|X) = \int_{\Theta} f(X|\theta)p(\theta)d\theta$$

- 缺失数据(Missing data)中似然概率的计算

$$P(y|\theta) = \int_{\mathcal{X}} f(x, y|\theta)dx$$

Importance Sampling

- 若 q(x)在p(x)非零点恒正，则

$$I = \int_a^b f(x)p(x)dx = \int_a^b f(x)\frac{p(x)}{q(x)}q(x)dx = E_q\left[\frac{f(x)p(x)}{q(x)}\right]$$

- 随机抽取q(x)的样本点 $\xi_1, \cdots, \xi_n \sim q(x),$

$$\hat{I}_n = \frac{1}{n}\left[\frac{f(\xi_1)p(\xi_1)}{q(\xi_1)} + \cdots + \frac{f(\xi_n)p(\xi_n)}{q(\xi_n)}\right]$$

Importance Sampling

- 它是无偏估计

$$E_q(\hat{I}_n) = \int_a^b f(x)p(x)dx$$

- 收敛条件

$$E_q\left[\left(\frac{f(X)p(X)}{q(X)}\right)^2\right] < +\infty$$

Choice of q(x)

- 方差

$$Var(\hat{I}_n) = \frac{1}{n}\left[\int_{\mathcal{X}} \left(\frac{f(x)p(x)}{q(x)}\right)^2 q(x)dx - I^2\right]$$

- 定理：当q(x)有如下表示时

$$q^*(x) = \frac{|f(x)|p(x)}{\int_{\mathcal{X}} |f(z)|p(z)dz}$$

上述方差最小

Choice of q(x)

- 证明：

$$var\left[\frac{f(X)p(X)}{q(X)}\right] = E_q\left[\frac{f^2(X)p^2(X)}{q^2(X)}\right] - \left(E_q\left[\frac{f(X)p(X)}{q(X)}\right]\right)^2$$

- 第二项与q无关，而对第一项由Jensen不等式(函数x^2)，

$$E_q\left[\frac{f^2(X)p^2(X)}{q^2(X)}\right] \geq \left(E_q\left[\frac{|f(X)|p(X)}{q(X)}\right]\right)^2$$
$$= \left(\int |f(x)|p(x)dx\right)^2$$

Choice of q(x)

- 直接验证得：等号成立时q(x)

$$q^*(x) = \frac{|f(x)|p(x)}{\int_{\mathcal{X}} |f(z)|p(z)dz} = C|f(x)|p(x)$$

Monte Carlo 近似计算

- 但是上面的分布密度函数q(.)含未知数c, 其实它正是我们所要求的积分。所以上面的最优Importance Sampling方法并不可行。
- 利用 MCMC 算法可以不需要知道c的取值而得到分布密度函数近似为g(.)的样本。这样，我们就可以实现前面所讲的Monte Carlo的近似计算。

Pseudo-Random Numbers

- Truly random numbers can not be generated on a computer
- Pseudo-random numbers follows a well-defined algorithm, thus predictable and repeatable
- Have nearly all the properties of true random numbers

Linear congruence generator (LCG)

- One of the earliest and fastest algorithm:

$$X_{n+1} = (aX_n + c) \mod M$$

where $0 \leq X_n < M$, M is the modulus, a is multiplier, c is increment. All of them are integers. Choice of a, c, M must be done with special care.

http://random.mat.sbg.ac.at/links/rando.html

Choice of Parameters

Source	m	a	c	output bits of seed in *rand()* / *Random(L)*
Numerical Recipes	2^{32}	1664525	1013904223	
Borland C/C++	2^{32}	22695477	1	bits 30..16 in *rand()*, 30..0 in *lrand()*
glibc (used by GCC)[4]	2^{32}	1103515245	12345	bits 30..0
ANSI C: Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++	2^{32}	1103515245	12345	bits 30..16
Borland Delphi, Virtual Pascal	2^{32}	134775813	1	bits 63..32 of *(seed * L)*
Microsoft Visual/Quick C/C++	2^{32}	214013	2531011	bits 30..16
RtlUniform from Native API [5]	$2^{31} - 1$	2147483629	2147483587	
Apple CarbonLib	$2^{31} - 1$	16807	0	see MINSTD
MMIX by Donald Knuth	2^{64}	6364136223846793005	1442695040888963407	
VAX's **MTH$RANDOM**,[6] old versions of glibc	2^{32}	69069	1	
Random class in Java API	2^{48}	25214903917	11	bits 47...16
LC53[7] in Forth (programming language)	$2^{32} - 5$	$2^{32} - 333333333$	0	

$$x_{n+1} = (ax_n + c) \mod M$$

http://en.wikipedia.org/wiki/Linear_congruential_generator
http://crypto.mat.sbg.ac.at/results/karl/server/server.html

Other Modern Generators

- Mersenne Twister (MT19937)

  Extremely long period (2^{19937}-1), fast

  (http://www.math.keio.ac.jp/~matumoto/emt.html)

- Inversive congruential generator

$$X_{n+1} = (ax_n^{-1} + c) \mod M$$

  nonlinear, no lattice structure

http://en.wikipedia.org/wiki/Mersenne_twister
http://en.wikipedia.org/wiki/Inversive_congruential_generator

多项分布生成

- 设有多项分布

$$\{ p_1 , p_2 , \ldots , p_n \},$$

要生成以它为分布的随机变量 ξ，只要

  - 将 [0 , 1] 区间分割为 n 份, 使得各份的长度分别为 $p_1 , p_2 , \ldots , p_n$
  - 生成随机变量 u, 使 U 遵从 [0,1]上均匀分布.
  - 当 u 在上述区间的第 k 份， 取 ξ 为 k。

Non-Uniformly Distributed Random Numbers

- Let F(*x*) be the cumulative distribution function of a random variable *X*, then *x* can be generated from

$$x = F^{-1}(\xi)$$

where ξ~U(0,1) , and $F^{-1}(x)$ is the inverse function of F(*x*).

Proof the Inverse Method

- The Mapping from *x* to *ξ* is one-to-one.
- The probability for *ξ* between value *ξ* and d*ξ* is 1·d*ξ*, which is the same as the probability for *x* between value *x* and d*x*. Thus

$$d\xi = dF(x) = F'(x)dx = p(x)dx$$

Since $F^{-1}(\xi)=x$, or $\xi = F(x)$

Example 1, Exponential Distribution

- Density function,

$$p(x) = \exp(-x), x \geq 0.$$

- Distribution function,

$$F(x) = \int_0^x \exp(-y)dy = 1 - \exp(-x)$$

- So we generate *x* by

$$x = -\log(\xi), \quad \xi \sim U(0,1)$$

Example 2, Gaussian distribution

- Take 2D Gaussian distribution

$$p(x,y)dxdy = \frac{1}{2\pi}\exp(-\frac{x^2+y^2}{2})dxdy$$

- Work in polar coordinates,

$$p(r,\theta)rdrd\theta = \frac{1}{2\pi}\exp(-\frac{r^2}{2})rdrd\theta$$

Box-Muller Method

- The formula implies that the variable ϑ is distributed uniformly between 0 and 2π, $\frac{1}{2}r^2$ is exponentially distribution, we have

$$\begin{cases} x = \sqrt{-2\log(\xi_1)}\cos(2\pi\xi_2) \\ y = \sqrt{-2\log(\xi_1)}\sin(2\pi\xi_2) \end{cases}$$

ξ1 and ξ2 are two independent, uniformly distributed random numbers.

Von Neuman 取舍原则

- 对于比较复杂的或不常见的分布, 下面的 **Von Neuman 取舍原则** 提供了一个非常有效的方法。
- 若密度为$p_0(x)$的随机变量容易生成，现在要生成密度为$p(x)$的随机变量，而且

$$p(x) \leq cp_0(x)$$

Von Neuman 取舍原则

- 生成密度为$p_0(x)$的随机变量列

$$\{\xi_1, \xi_2, \cdots, \xi_n\}$$

- 对 $\{\xi_n\}$ 进行随机筛选, 留下的随机变量列就是密度为p(x)的随机变量列
- 筛选方法：独立生成U_1,…, U_n~U(0,1)，如果

$$U_k \geq \frac{p(\xi_k)}{cp_0(\xi_k)}$$

删除 ξ_k，否则保留 ξ_k .

Von Neuman 取舍原则的论证

- 留下的随机变量仍然是相互独立同分布的，它们中的每一个 (记为 η) 的分布是

$$
\begin{aligned}
&Pr(\eta < x) = Pr(\xi_k < x | \xi_k \text{ kept}) \\
&= Pr(\xi_k < x | U_k \leq \frac{p(\xi_k)}{cp_0(\xi_k)} \\
&= \frac{Pr\left(\xi_k < x, U_k \leq \frac{p(\xi_k)}{cp_0(\xi_k)}\right)}{Pr\left(U_k \leq \frac{p(\xi_k)}{cp_0(\xi_k)}\right)} \\
&= \frac{\int_{-\infty}^{x} \frac{p(y)}{cp_0(y)} p_0(y) dy}{\int_{-\infty}^{+\infty} \frac{p(y)}{cp_0(y)} p_0(y) dy} = \int_{-\infty}^{x} p(y) dy = F(y)
\end{aligned}
$$

马氏链的遍历性与遍历极限

- **马氏链的遍历性定理：**若有限状态Markov链的所有状态都是互通的(不可约)，f是状态空间上的有界实值函数且满足 $\sum_i |f(i)|\pi_i < +\infty,$ 则

$$Pr\left(\frac{f(\xi_1)+\cdots+f(\xi_n)}{n} \to \sum_i f(\xi_i)\pi_i\right) = 1$$

其中π是MC的不变分布，而且此极限与初分布无关.

MCMC 算法的思想

- 对于非周期的MC (例如$p_{ij}>0$时)，我们还有

$$p_{ij}(n) \to \pi_j, \qquad n \to +\infty$$

- **Markov Chain Monte Carlo** 算法的思想就是：设计一个马氏链, 使得它的极限分布π与f成比例, 于是当我们模拟马氏链足够多步后, 它的分布就近似于π.

Markov Chain Monte Carlo

- Goal: $p(z^{(m)}) = p^*(z)$ as $m \to \infty$
  - MCMCs that have this property are **ergodic.**

- Transition properties that provide **detailed balance** guarantee ergodic MCMC processess.
  - Also considered **reversible**.

$$p^*(z)T(z, z') = p^*(z')T(z', z)$$

Markov Chain Monte Carlo (MCMC)

- 设计一个 Markov 链，使其不变分布为我们关心的分布, (如高维分布, 或样本空间非常大的离散分布)。用这个 Markov 链的样本, 来对该分布作采样, 并用以作随机模拟。这样的方法, 统称为 **Markov Chain Monte Carlo (MCMC)方法。**
- 由于这种方法的问世, 使随机模拟在很多领域的计算中, 相对于决定性算法，显示出它的巨大的优越性。而有时随机模拟与决定性算法的结合使用, 会显出更多的长处.

MCMC 的应用

- 用于生成较复杂的随机数： 高维分布的随机向量。
- 求复杂空间上函数的极值： 模拟退火；
- 缺失数据的参数估计：Gibbs Sampler
- Bayesian 缺失数据问题

Gibbs Sampler

- 生成一元随机变量是并不困难的， 但是生成高维各分量不独立的随机向量就非常困难。
- Gibbs 采样法的思想是**通过条件分布**得到以给定分布π为不变分布的马氏链的转移概率.

Gibbs采样法

- 这里$\pi(x) = \pi(x_1, x_2, \cdots, x_m)$是一个m-维分布密度, 相应的马氏链的状态是m-维向量, 其转移矩阵是 (p_{xy}). 具体地, 取

$$p_{xy} = p(x, y) = \prod_{k=1}^{m} \pi(y_k | y_1, \cdots, y_{k-1}, x_{k+1}, \cdots, x_m)$$

- 意思是依次改变状态的m个分量, 每次只改变状态的一个分量, 使x变为y
- 容易验证: $\sum_x \pi(x) p_{xy} = \pi(y)$, 即 $\pi \times (p_{xy}) = \pi$

Gibbs 采样法的论证

$$\sum_x \pi(x)p(x,y) = \sum_x \pi(x) \prod_{k=1}^{m} \pi(y_k|y_1,\cdots,y_{k-1},x_{k+1},\cdots,x_m)$$

$$= \sum_{x_1,\cdots,x_m} \left[\sum_{x_1} \pi(x_1,\cdots,x_m)\right] \frac{\pi(y_1,x_2,\cdots,x_m)}{\sum_{x_1}\pi(x_1,\cdots,x_m)} \prod_{k=2}^{m} \pi(y_k|y_1,\cdots,y_{k-1},x_{k+1},\cdots,x_m)$$

$$= \sum_{x_2,\cdots,x_m} \pi(y_1,x_2,\cdots,x_m) \prod_{k=2}^{m} \pi(y_k|y_1,\cdots,y_{k-1},x_{k+1},\cdots,x_m)$$

$$= \cdots\cdots\cdots$$

$$= \pi(y_1,\cdots,y_m) = \pi(y)$$

构造 Markov 链轨道的方法

- 从已有时刻 n 的样本 ξ_n 求时刻 $n+1$ 的样本 ξ_{n+1}
  1. 先由 $\xi_n=(\xi_n^1,\xi_n^2,\ldots,\xi_n^s)$ 得到服从分布为

$$\pi(y_1|x_2,\cdots,x_m)=\frac{\pi(y_1,x_2,\cdots,x_m}{\pi(+\infty,x_2,\cdots,x_m)}$$

  的一元随机变量 ξ_{n+1}^1，其中 $x_k=\xi_n^k$

  2. 用同样的方法，再得到服从分布

$$\pi(y_2|y_1,x_3,\cdots,x_m)$$

  的样本 ξ_{n+1}^2，其中 $y_1=\xi_{n+1}^1$

构造 Markov 链轨道的方法

3. 依此下去…，得到 ξ_{n+1} 的所有分量.

4. 不断重复 1) — 3)， 我们就可以得到 上述马氏链的一 个样本轨道， 即一列随机向量 $\{\xi_n\}$, 其中 ξ_n 当 n 充分大时, 都可以近似地作为 π 的样本。

Metropolis Sampler

- Metropolis-Hastings Algorithm

- N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E.Teller. Equations of state calculations by fast computing machines. J.Chem.Phys.21: 1087-1091, 1953.
- W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. Biometrika. 57:97-109, 1970.

Metropolis 采样法概述 I

- 与Gibbs采样法一样，Metropolis方法也给出了在计算机上用 马氏链近似模拟遵从一个分布 π 的随机变量(向量) 的一个算法。
- Metropolis 提出了这种采样法, 称为**Metropolis采样法**。

Metropolis 采样法概述 II

- 它与 Gibbs 采样法的不同处在于, 对于Metropolis 采样法的转移概率如下：

$$p_{i,j} = \begin{cases} \overline{p}_{ij} \frac{\pi_j}{\pi_i} & \forall j \neq i, \\ 1 - \sum_{k \neq i} \overline{p}_{ik} \frac{\pi_k}{\pi_i} & j = i \end{cases}$$

Metropolis 采样法概述 III

- 其中 $\overline{P} = (\overline{p}_{ij})$ 是一个对称的互通转移矩阵, 称为**预选矩阵**，使用它是为了减少状态间的连接, 以加快 Markov 链的分布向不变分布收敛的速度。
- 由于预选矩阵是一个对称的互通转移矩阵, 所以

$$\sum_i \pi_i \left(\overline{p}_{ij} \frac{\pi_j}{\pi_i} \right) = \left(\sum_i \overline{p}_{ij} \right) \pi_j = \pi_j$$

Metropolis 采样法概述 IV

- **预选矩阵 $\overline{P} = (\overline{p}_{ij})$ 的选取:** 在许多情况下, 由于总的状态数非常多, 我们希望每次转移只能到达很少的几个状态, 但是保持不变分布仍然为 π.
- 我们通常选取非常稀疏的预选矩阵, 只要保证它是互通而且对称的转移概率阵就行了.
- 例如当 $\boldsymbol{\pi}$ 为多维分布, 从一个状态 $\boldsymbol{x}$ 出发,可规定它只能到达与它只有一个分量不同的状态.

Metropolis-Hastings Algorithm

- Set an initial value X_1.
- If the chain is currently at $X_n = x$, randomly propose a new position $X_{n+1} = y$ according to a proposal density $q(x, y)$.
- Accept the proposed jump with probability

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

- If not accepted, $X_{n+1} = x$.

Why M-H Works?

- Transition probability

$$P(x \to y) = q(x,y)\alpha(x,y)$$
$$= q(x,y)\min\left(\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right)$$

- Detailed balance

$$\pi(x)P(x \to y) = \pi(y)P(y \to x)$$

$$q(x,y)\alpha(x,y)\pi(x) = q(y,x)\alpha(y,x)\pi(y)$$

Why M-H Works?

- Case 1: $q(x,y)\pi(x) = q(y,x)\pi(y).\ \ \alpha(x,y) = \alpha(y,x) = 1$

$$q(x,y)\alpha(x,y)\pi(x) = q(y,x)\alpha(y,x)\pi(y)$$

- Case 2:

$$q(x,y)\pi(x) > q(y,x)\pi(y).$$

$$\alpha(x,y) = \frac{q(y,x)\pi(y)}{q(x,y)\pi(x)}, \alpha(y,x) = 1$$

$$q(x,y)\alpha(x,y)\pi(x) = q(x,y)\frac{q(y,x)\pi(y)}{q(x,y)\pi(x)}\pi(x)$$

$$= q(y,x)\pi(y) = q(y,x)\alpha(y,x)\pi(y)$$

i.e.

$$\pi(x)P(x \to y) = \pi(y)P(y \to x)$$

Why M-H Works?

- Case 3: $q(x,y)\pi(x) < q(y,x)\pi(y).$

$$\alpha(x,y) = 1, \alpha(y,x) = \frac{q(x,y)\pi(x)}{q(y,x)\pi(y)}$$

$$\begin{aligned} q(y,x)\alpha(y,x)\pi(y) &= q(x,y)\frac{q(x,y)\pi(x)}{q(x,y)\pi(x)}\pi(y) \\ &= q(x,y)\pi(x) = q(x,y)\alpha(x,y)\pi(x) \end{aligned}$$

i.e.

$$\pi(y)P(y \to x) = \pi(x)P(x \to y)$$

MCMC应用于优化
模拟退火算法大意

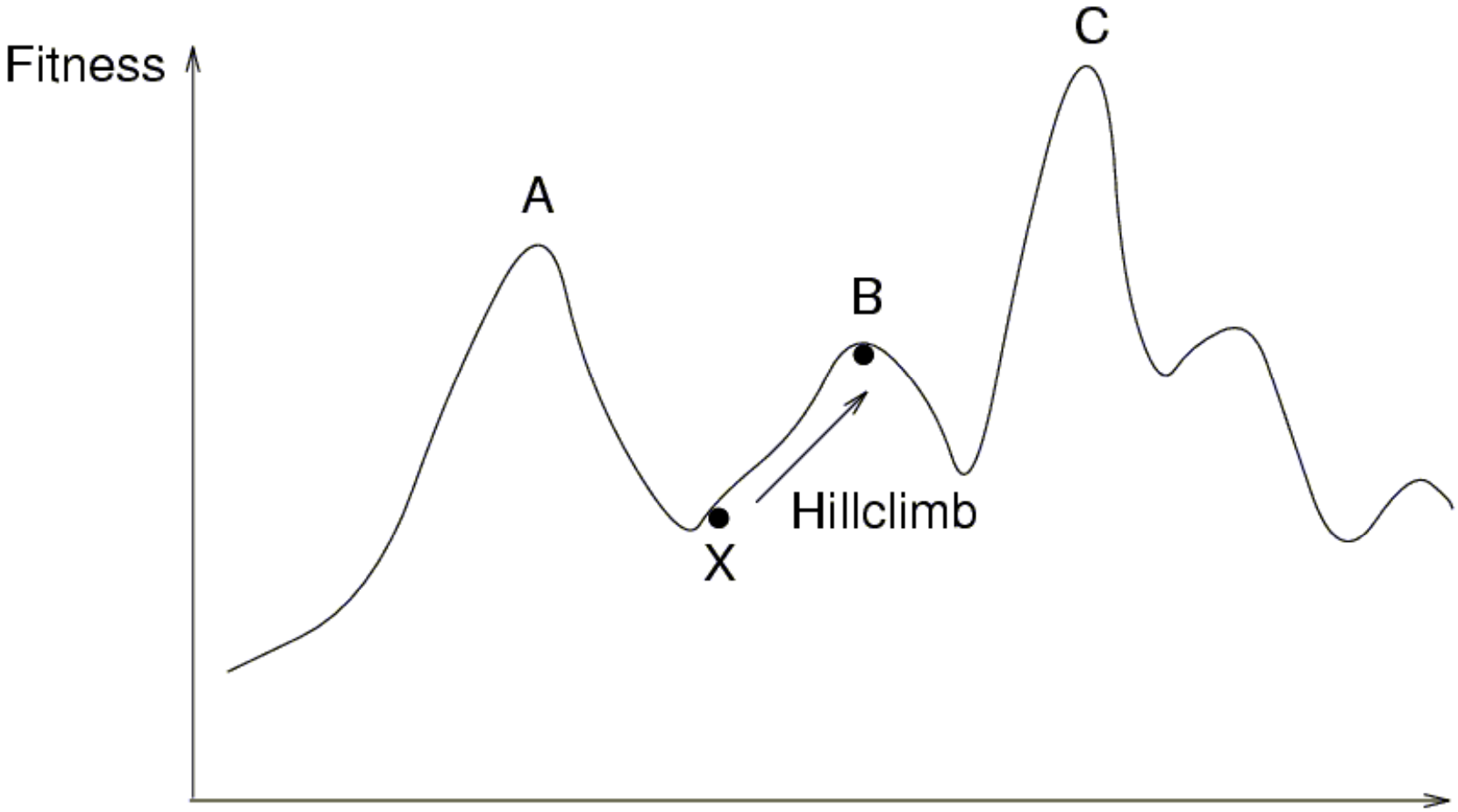
部分Slides从下面的PPT中摘录
www.intelligentmodelling.org.uk/AB/G52AIM/G52AIM-Lecture03.ppt

优化问题

- 考虑最简单的单变量函数的最大值求解问题

$$x^\star = \operatorname*{Argmax}_{x \in \Omega} F(x)$$

Hill Climbing



The Problem with Hill Climbing

- Gets stuck at local minima
- Possible solutions
  - Try several runs, starting at different positions
  - Increase the size of the neighbourhood (e.g. in TSP try 3-opt rather than 2-opt)

Simulated Annealing

- Motivated by the physical annealing process
- Material is heated and slowly cooled into a uniform structure
- Simulated annealing mimics this process
- The first SA algorithm was developed in 1953 (Metropolis)

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. Equations of State Calculations by Fast Computing Machines. Journal of Chemical Physics, 21(6):1087-1092, 1953.

Simulated Annealing

- Compared to hill climbing the main difference is that SA allows downwards steps
- Simulated annealing also differs from hill climbing in that a move is selected at random and then decides whether to accept it
- In SA better moves are always accepted. worse moves are not

Simulated Annealing

- Kirkpatrick (1982) applied SA to optimization problems

Kirkpatrick, S , Gelatt, C.D., Vecchi, M.P. 1983. Optimization by Simulated Annealing. Science, vol 220, No. 4598, pp 671-680

概率观点处理优化问题

- 若x_0是最大值，那么考虑非负函数

$$E = F(x_0) - F(x)$$

- 由热力学定律，有Boltzman分布

$$\pi_\beta = C\exp(-\beta E), \quad \beta = \frac{1}{kT}$$

  - 温度趋于无穷时，分布趋于均匀分布
  - 温度趋于0是，分布趋于单点分布

模拟退火

- 实质就是要构造适当的Markov链，使得相应Boltznann分布作为其极限分布

- 温度T有一个调整的过程，首先升温保证初始点几乎以均一的概率都能被访问到，而后逐渐降温，使得分布越来越集中到最大值点

转移概率构造
(Metropolis-Hastings Algorithm)

- 对应同一个温度下，考虑x到 y的移动,

$$p_{x,y}(\beta) = \begin{cases} 1, & \Delta E = F(y) - F(x) > 0 \\ \exp(-\beta \Delta E), & \Delta E = F(y) - F(x) < 0 \end{cases}$$

- 或者只允许x到其近邻的移动，

$$p_{x,y}(\beta) = \begin{cases} 1, & \Delta E = F(y) - F(x) > 0 \\ \exp(-\beta \Delta E), & \Delta E = F(y) - F(x) < 0 \end{cases}$$

To accept or not to accept?

- The law of thermodynamics states that at temperature, t, the probability of an increase in energy of magnitude, ΔE , is given by

$$Pr(\Delta E) = \exp\left(-\frac{\Delta E}{kT}\right)$$

- Where k is a constant known as Boltzmann's constant

To accept or not to accept - SA?

$$p = \exp(-\frac{c}{t}) > r$$

- where
  - c is change in the evaluation function
  - t the current temperature
  - r is a random number between 0 and 1

To accept or not to accept - SA?

- The probability of accepting a worse state is a function of both the temperature of the system and the change in the cost function
- As the temperature decreases, the probability of accepting worse moves decreases
- If t=0, no worse moves are accepted (i.e. hill climbing)

SA Cooling Schedule

- Starting Temperature
- Final Temperature
- Temperature Decrement
- Iterations at each temperature

Starting Temperature

- Must be hot enough to allow moves to almost neighbourhood state (else we are in danger of implementing hill climbing)
- Must not be so hot that we conduct a random search for a period of time
- Problem is finding a suitable starting temperature

Choosing Starting Temperature

- If we know the maximum change in the cost function we can use this to estimate
- Start high, reduce quickly until about 60% of worse moves are accepted. Use this as the starting temperature
- Heat rapidly until a certain percentage are accepted the start cooling

Choosing Final Temperature

- It is usual to let the temperature decrease until it reaches zero. However, this can make the algorithm run for a lot longer, especially when a geometric cooling schedule is being used
- In practice, it is not necessary to let the temperature reach zero because the chances of accepting a worse move are almost the same as the temperature being equal to zero

Choosing Final Temperature

- Therefore, the stopping criteria can either be a suitably low temperature or when the system is “frozen” at the current temperature (i.e. no better or worse moves are being accepted)

Temperature Decrement

- Theory states that we should allow enough iterations at each temperature so that the system stabilizes at that temperature
- Unfortunately, theory also states that the number of iterations at each temperature to achieve this might be exponential to the problem size

Temperature Decrement

- We need to compromise
- We can either do this by doing a large number of iterations at a few temperatures, a small number of iterations at many temperatures or a balance between the two

Temperature Decrement

- Linear

$$\text{Temp} = \text{Temp} - x$$

- Geometric

$$\text{Temp} = \text{Temp} \times \alpha$$

- Experience has shown that α should be between 0.8 and 0.99, with better results being found in the higher end of the range. Of course, the higher the value of α, the longer it will take to decrement the temperature to the stopping criterion

Iterations at each temperature

- A constant number of iterations at each temperature

- Another method, first suggested by (Lundy, 1986) is to only do one iteration at each temperature, but to decrease the temperature very slowly.

Iterations at each temperature

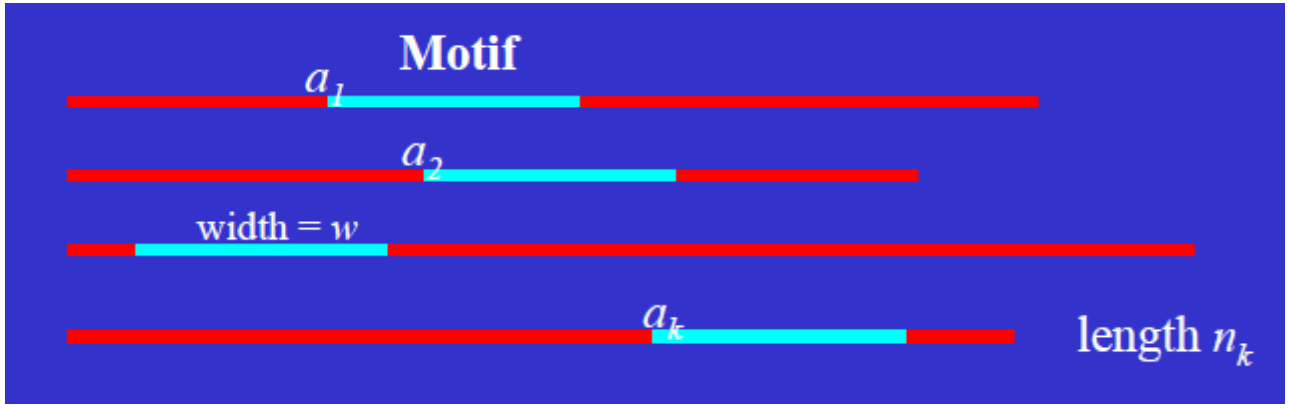
- An alternative is to dynamically change the number of iterations as the algorithm progresses
- At lower temperatures it is important that a large number of iterations are done so that the local optimum can be fully explored
- At higher temperatures, the number of iterations can be less

MCMC for Parameters Estimation with Missing Data

Motif Finding Problem

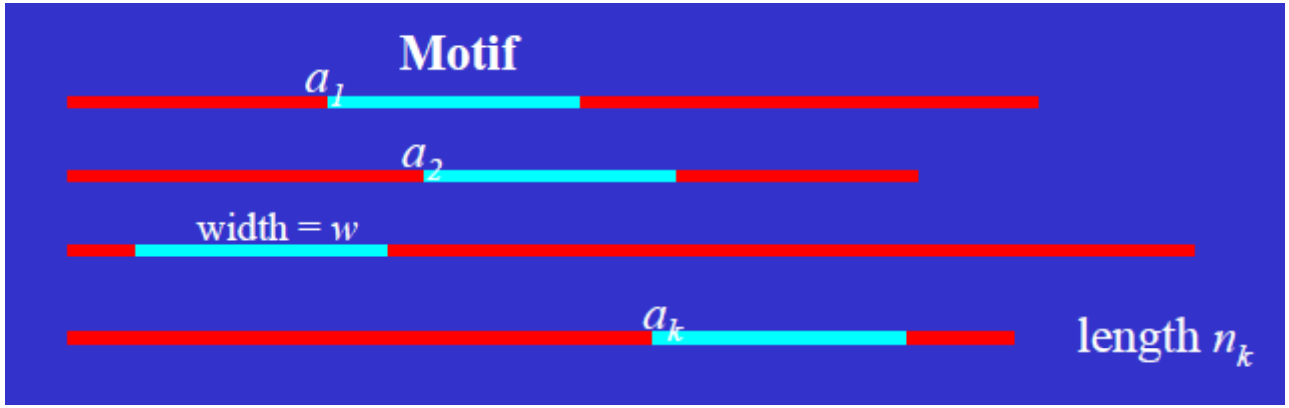
- Object: finding the best “common” pattern
- Missing data: alignment variable

$$A = \{a_1, \cdots, a_K\}$$



Motif Finding Problem

- Alignment variable: $A = \{a_1, \cdots, a_K\}$
- Parameters: $\theta = \{\vec{p}_0, \vec{p}_1, \cdots, \vec{p}_w\}$



Gibbs Sampler

- Iterative sampling
  - Draw from

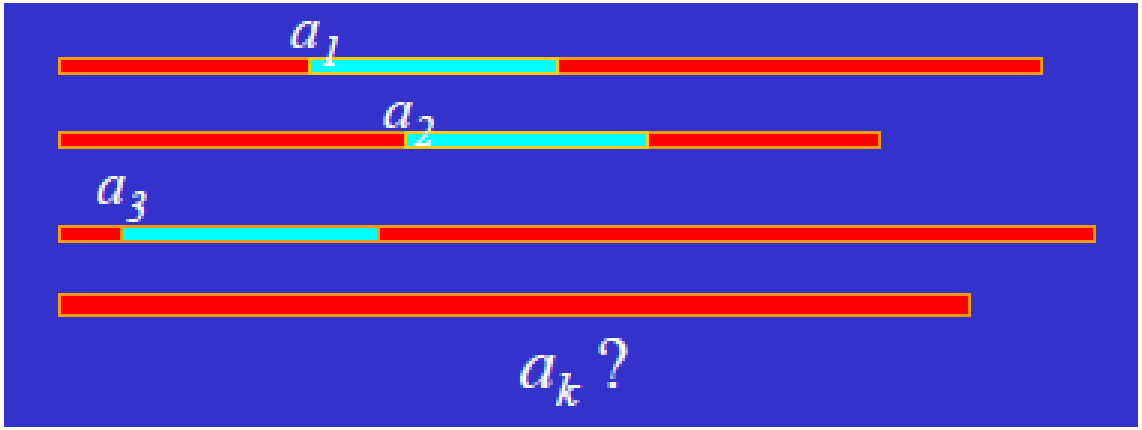
$$P(\theta|A, Data)$$

  - Draw from

$$P(A|\theta, Data)$$

Implementation: Predictive Updating

- Pretend k-1 sequences are aligned
- Predict the kth sequence (stochastically)



Predictive Updating

- Compute predictive frequencies of each position i in motif
  - c_{ij} : Counts of nucleotide type j at position i.
  - c_{0j} : Count of nucleotide type j at non-site position.

$$q_{ij} = \frac{c_{ij} + b_j}{\sum_k c_{kj} + \sum_k b_k}$$

- Sample from the predictive distribution of a_k

$$P(a_k = l + 1) \propto \prod_{i=1}^{w} \frac{q_{i,R_{k,(l+i)}}}{q_{0,R_{k,(l+i)}}}$$

Gibbs Sampler Algorithm

- Initialize by choosing random starting positions
- Iterate the following steps many times:
  - Randomly or systematically choose a sequence, say *sequence k, to exclude.*
  - Carry out Predictive Update step to update
- Stop when changes are small, or other convergence criterion met.

MCMC for Bayesian Missing Data Problem

Bayesian Inference

- Now the parameter is also a random variable, with a prior probability $p(\theta)$
- Using Bayes rule, the posterior probability

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int_\Theta p(D|\theta)p(\theta)d\theta}$$

Posterior $\propto$ "Likelihood" $\times$ Prior

Conjugate Prior

- The posterior distribution is in the same family as the prior distribution
- 采用共轭先验的原因是为了简化积分计算

- http://en.wikipedia.org/wiki/Conjugate_prior

Conjugate Prior

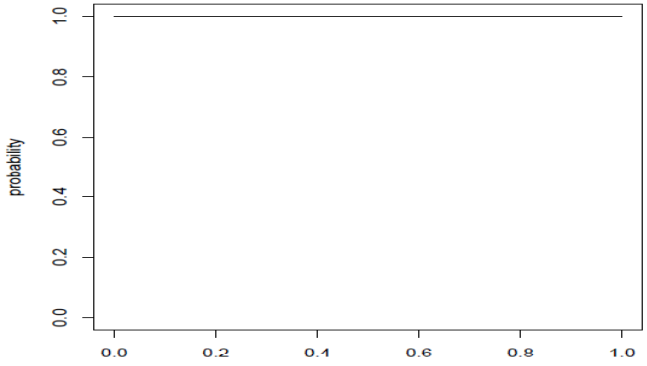
- Example: n次独立实验，每次实验成功的概率为p, 总成功次数X服从二项分布，共轭先验Beta分布

$$X \sim B(n,p), Pr(k) = C_n^k p^k (1-p)^{n-k}$$

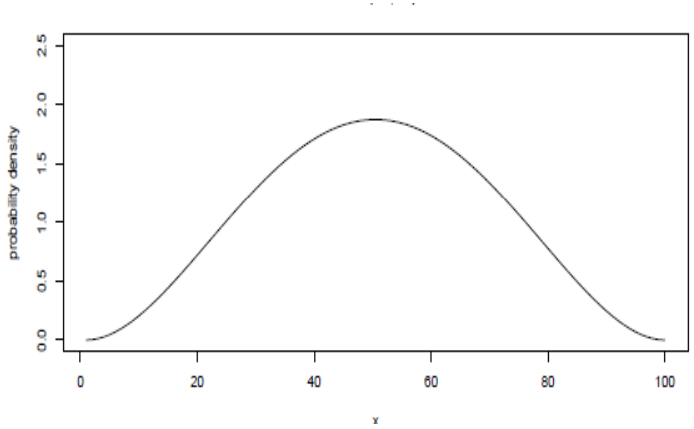
$$p \sim Beta(\alpha,\beta), Pr(p) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}$$

$$Pr(p|X) \sim Beta(\alpha+k, \beta+n-k)$$

Conjugate Prior



Beta(1,1)



Beta(3,3)

Conjugate Prior

- Normal distribution with known variance

$$X \sim N(\mu, \sigma^2)$$

- Normal prior distribution

$$\mu \sim N(\mu_0, \sigma_0^2)$$

- Posterior distribution

$$N\left(\frac{\sigma_0^2}{\sigma^2 + \sigma_0^2}x + \frac{\sigma^2}{\sigma^2 + \sigma_0^2}\mu_0, \frac{\sigma_0^2\sigma^2}{\sigma_0^2 + \sigma^2}\right)$$

MCMC for Bayesian Inference

- Draw samples from the posterior distribution

$$\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(m)} \sim P(\theta|D)$$

- Use these samples to compute some objectives.

$$E[f(\theta)] \approx \frac{1}{m}[f(\theta^{(1)}) + f(\theta^{(2)}) + \cdots + f(\theta^{(m)})]$$

Prediction

- Given data D, compute probability for $\tilde{D}$
- Predictive Distribution

$$P(\tilde{D}|D) = \int P(\tilde{D}|\theta)P(\theta|D)d\theta$$

$$(\quad \neq p(\tilde{D}|\hat{\theta})!!! \quad)$$

Back to Motif Finding Problem

- Given K sequences, $R = (R_1, R_2, \cdots, R_K)$ each containing exact one motif (width w)
- Alignment variable $A = (a_1, \cdots, a_K)$ is missing
- Product multinomial model
  - Background Model: $\theta_0 = (\theta_{01}, \cdots, \theta_{0d})$
  - Motif Model: $\Theta = [\theta_1, \theta_2, \cdots, \theta_w]$
- Prior: $\text{Dirichlet}(N_0\alpha_{0,1}, \cdots, N_0\alpha_{0,d})$

Gibbs Sampler

- Iterating
  - Drawing Θ from $\pi(\Theta|A, Data)$
  - Drawing A from $\pi(A|\Theta, Data)$
- Such a procedure constructs a Markov chain whose equilibrium distribution is

$$\pi(\Theta, A|Data)$$

Collapse Gibbs Sampler

- One can integrate out parameters

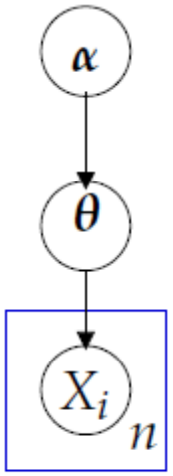
$$\pi(A|Data) = \int \pi(A, \theta|Data)p(\theta)d\theta$$

- Directly sample A.

- Lawrence, Liu, Neuwald, Collapsed Gibbs Sampler algorithm in Multiple Sequence Alignment. 1993,1994,1995

Dirichilet-Multinomial distribution

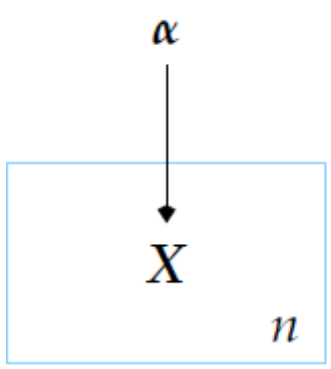
$$\begin{aligned}
\boldsymbol{\theta} \mid \boldsymbol{\alpha} &\sim \text{DIR}(\boldsymbol{\alpha}) \\
X_i \mid \boldsymbol{\theta} &\sim \text{CATEGORICAL}(\boldsymbol{\theta}), \quad i = 1, \ldots, n
\end{aligned}$$



$$\begin{aligned}
\mathrm{P}(\boldsymbol{X}, \boldsymbol{\theta} \mid \boldsymbol{\alpha}) &= \mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \mathrm{P}(\boldsymbol{X} \mid \boldsymbol{\theta}) \\
&= \left(\frac{1}{C(\boldsymbol{\alpha})} \prod_{j=1}^{m} \theta_j^{\alpha_j - 1} \right) \left(\prod_{j=1}^{m} \theta_j^{N_j(X)} \right) \\
&= \frac{1}{C(\boldsymbol{\alpha})} \prod_{j=1}^{m} \theta_j^{N_j(X) + \alpha_j - 1}, \text{ so} \\
\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{\alpha}) &\propto \mathrm{P}(\boldsymbol{X}, \boldsymbol{\theta} \mid \boldsymbol{\alpha}) \\
&\propto \prod_{j=1}^{m} \theta_j^{N_j(X) + \alpha_j - 1}, \text{ so} \\
\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{\alpha} &\sim \text{DIR}(\boldsymbol{N} + \boldsymbol{\alpha})
\end{aligned}$$

Collapse Model

$$
\begin{aligned}
\boldsymbol{\theta} \mid \boldsymbol{\alpha} &\sim \text{DIR}(\boldsymbol{\alpha}) \\
X_i \mid \boldsymbol{\theta} &\sim \text{CATEGORICAL}(\boldsymbol{\theta}), \quad i = 1, \ldots, n
\end{aligned}
$$



$$
\begin{aligned}
\mathrm{P}(\boldsymbol{X}|\boldsymbol{\alpha}) &= \int \mathrm{P}(\boldsymbol{X}, \boldsymbol{\theta}|\alpha)\, d\boldsymbol{\theta} = \int \mathrm{P}(\boldsymbol{X}|\boldsymbol{\theta})\, \mathrm{P}(\boldsymbol{\theta}|\boldsymbol{\alpha})\, d\boldsymbol{\theta} \\
&= \int \left(\prod_{j=1}^{m} \theta_j^{N_j} \right) \left(\frac{1}{C(\boldsymbol{\alpha})} \prod_{j=1}^{m} \theta_j^{\alpha_j - 1} \right) d\boldsymbol{\theta} \\
&= \frac{1}{C(\boldsymbol{\alpha})} \int \prod_{j=1}^{m} \theta_j^{N_j + \alpha_j - 1}\, d\boldsymbol{\theta} \\
&= \frac{C(\boldsymbol{N} + \boldsymbol{\alpha})}{C(\boldsymbol{\alpha})}, \text{ where } C(\boldsymbol{\alpha}) = \frac{\prod_{j=1}^{m} \Gamma(\alpha_j)}{\Gamma(\sum_{j=1}^{m} \alpha_j)}
\end{aligned}
$$

Dirichilet-Multinomial Posterior

$$\mathrm{P}(X|\boldsymbol{\alpha}) = \frac{C(\boldsymbol{N}+\boldsymbol{\alpha})}{C(\boldsymbol{\alpha})} \quad \text{where} \quad C(\boldsymbol{\alpha}) = \frac{\prod_{j=1}^{m}\Gamma(\alpha_j)}{\Gamma(\alpha_\bullet)} \text{ and } \alpha_\bullet = \sum_{j=1}^{m}\alpha_j$$

$$\mathrm{P}(X|\boldsymbol{\alpha}) = \left(\frac{\prod_{j=1}^{m}\Gamma(N_j+\alpha_j)}{\Gamma(n+\alpha_\bullet)}\right)\left(\frac{\Gamma(\alpha_\bullet)}{\prod_{j=1}^{m}\Gamma(\alpha_j)}\right)$$

$$= \left(\prod_{j=1}^{m}\frac{\Gamma(N_j+\alpha_j)}{\Gamma(\alpha_j)}\right)\left(\frac{\Gamma(\alpha_\bullet)}{\Gamma(n+\alpha_\bullet)}\right)$$

$$= \frac{\alpha_1}{\alpha_\bullet} \times \frac{\alpha_1+1}{\alpha_\bullet+1} \times \ldots \times \frac{\alpha_1+N_1-1}{\alpha_\bullet+N_1-1}$$
$$\times \frac{\alpha_2}{\alpha_\bullet+N_1} \times \frac{\alpha_2+1}{\alpha_\bullet+N_1+1} \times \ldots \times \frac{\alpha_2+N_2-1}{\alpha_\bullet+N_1+N_2-1}$$
$$\times \ldots$$
$$\times \frac{\alpha_m}{\alpha_\bullet+n-N_m-1} \times \frac{\alpha_m+1}{\alpha_\bullet+n-N_m} \times \ldots \times \frac{\alpha_m+N_m-1}{\alpha_\bullet+n-1}$$

Original Gibbs Sampler algorithm

Step0. choose an arbitrary starting point

$\mathbf{A_0=(a_{1,0},a_{2,0},\ldots,a_{N,0},Q_0);}$

Step2. Generate $A_{t+1}=(a_{1,t+1},a_{2,t+1},\ldots,a_{N,t+1},Q_{t+1})$ as follows:

Generate $\mathbf{a_{1,t+1}\sim\pi(a_1 \mid a_{2,t},\ldots,a_{N,t}, Q_t, B);}$

Generate $\mathbf{a_{2,t+1}\sim\pi(a_2 \mid a_{1,t+1}, a_{3,t} \ldots,a_{N,t}, Q_t, B);}$

…

Generate $\mathbf{a_{N,t+1}\sim\pi(a_N \mid a_{1,t+1}, a_{2,t+1} \ldots,a_{N-1,t+1}, Q_t, B);}$

Generate $\mathbf{Q_{t+1}\sim\pi(a_N \mid a_{1,t+1}, a_{2,t+1} \ldots,a_{N,t+1}, B);}$

Step3. Set **t=t+1**, and go to step 1

Collapsed Gibbs Sampler

Step0. choose an arbitrary starting point

$\mathbf{A_0=(a_{1,0},a_{2,0},\ldots,a_{N,0})}$;

Step2. Generate $A_{t+1}=(a_{1,t+1},a_{2,t+1},\ldots,a_{N,t+1})$ as follows:

Generate $\mathbf{a_{1,t+1}\sim \pi(a_1 \mid a_{2,t},\ldots,a_{N,t}, B)}$;

Generate $\mathbf{a_{2,t+1}\sim \pi(a_2 \mid a_{1,t+1}, a_{3,t} \ldots,a_{N,t}, B)}$;

…

Generate $\mathbf{a_{N,t+1}\sim \pi(a_N \mid a_{1,t+1}, a_{2,t+1} \ldots,a_{N-1,t+1}, B)}$;

Generate $\mathbf{Q_{t+1} \sim \pi(Q \mid a_{1,t+1}, a_{2,t+1} \ldots,a_{N,t+1}, B)}$;

Step3. Set **t=t+1**, and go to step 1

References

- Jun S. Liu. Monte Carlo Strategies in Scientific Computing. (2001). Springer.
- Christian P. Robert and George Casella. Monte Carlo Statistical Methods. Second Edition. (2004). Springer.