从模拟退火到概率编程

From Simulated Annealing to Probabilistic Programming

王迪
北京大学

2025 年 1 月 22 日

喵喵花园

洛谷 P8212

- 给定有 N 边的凸多边形 P，构造一个有 K 边的凸多边形 Q，使得
  - Q 的 K 个顶点都在 P 的边界上
  - Q 的 K 个顶点平均划分 P 的周长
  - Q 的面积尽可能小
- 找出 Q 的最小面积
- $3 \leq N, K \leq 1000$

优化问题在 OI 中很常见

在计算机科学和人工智能中也是

x^* 是解空间中使得评价函数值最小的解

$f : \mathcal{X} \to \mathbb{R}$ 是解的评价函数，比如值越小则解越优

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

x 是一个可能的解

$\mathcal{X}$ 是问题的解空间

喵喵花园

作为一个优化问题

- 给定有 N 边的凸多边形 P，构造一个有 K 边的凸多边形 Q，使得
  - Q 的 K 个顶点都在 P 的边界上
  - Q 的 K 个顶点平均划分 P 的周长
  - Q 的面积尽可能小
- 找出 Q 的最小面积
- $3 \le N, K \le 1000$

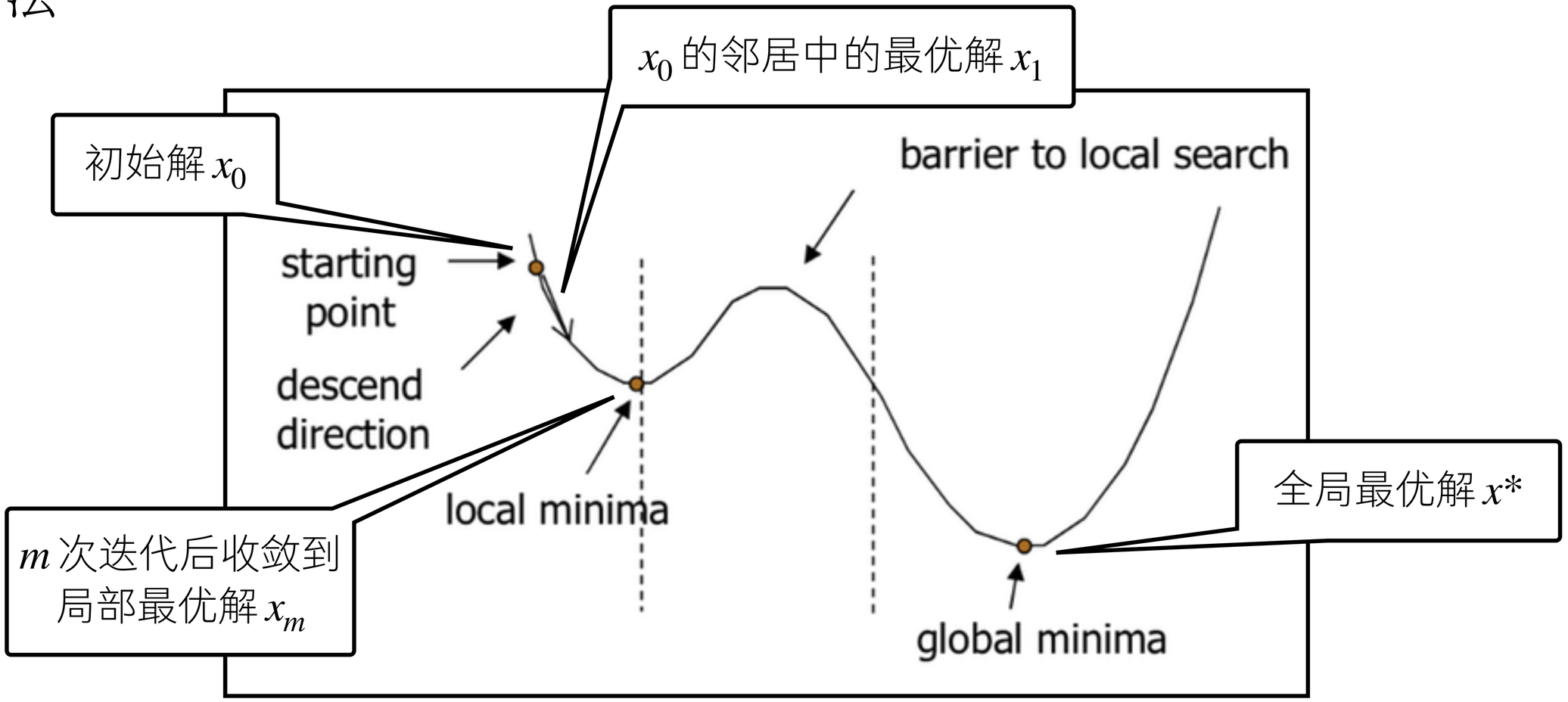
确定了**一个点**的位置就确定了多边形 Q

需要优化多边形 Q 的**面积**

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

贪心局部搜索

爬山法

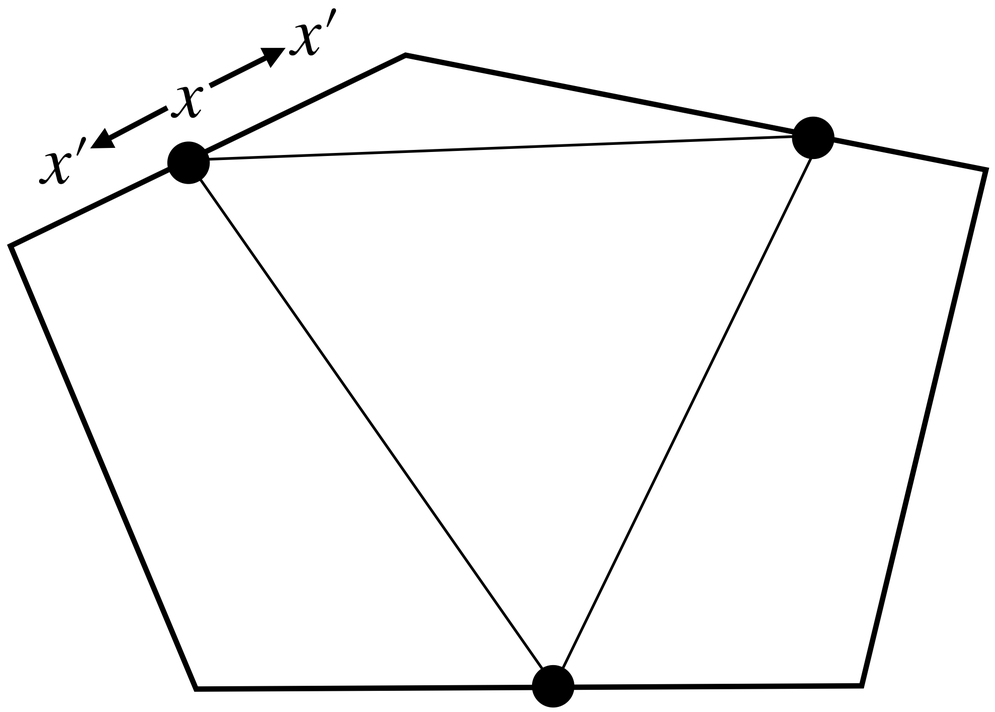


图片来源：https://codeforces.com/blog/entry/94437

喵喵花园

贪心局部搜索

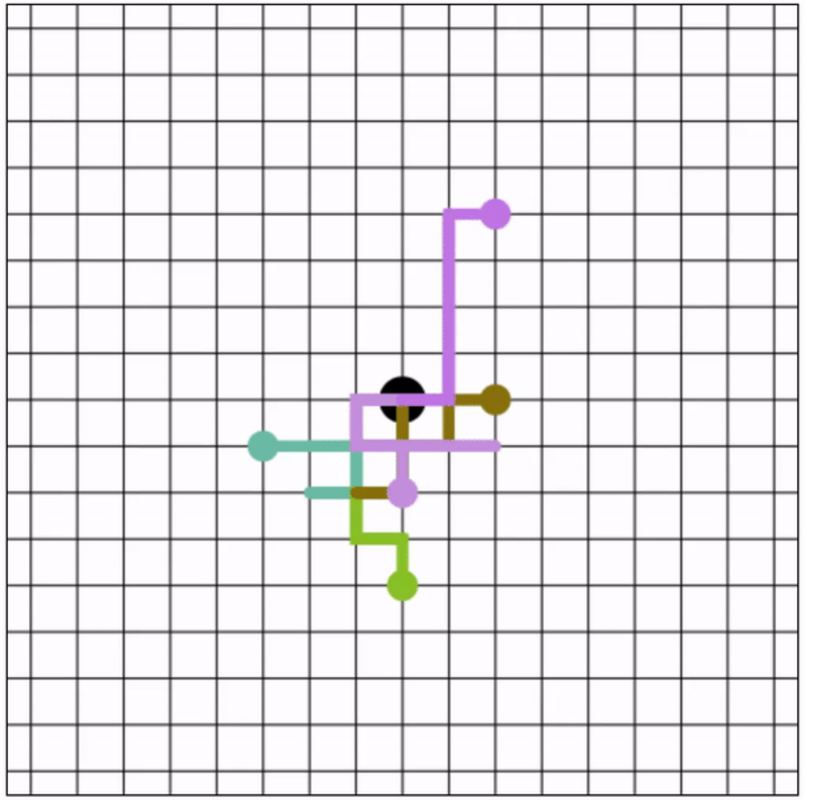
- 当前解 x：多边形 Q 的一个顶点
- 邻居 x'：顺时针或逆时针移动 x 一段距离
  - 步长太小：收敛慢，时间复杂度高
  - 步长太大：容易错过局部最优解
- 卡在局部最优了怎么办？



随机游走

阳寿法

- 从当前解 x 出发，随机从邻居中选择下一个解 x'
- 记录游走过程中遇到过的最优解
- （在一定的假设下）可以证明，随机游走**充分长**的时间后，一定能够找到最优解
- 无限猴子定理：给猴子一台打字机和**充分长**的时间，那么它一定能打出三体全集



二维网格随机游走

动图来源：https://en.wikipedia.org/wiki/File:Random_Walk_Simulator.gif

模拟退火 = 贪心局部搜索 + 随机游走
Simulated Annealing

- 在**爬山**的每一轮迭代中，**随机**从邻居中选择下一个解
- 当这个解比当前解更优时，接受它并开始下一轮迭代
- 当这个解比当前解更差时，以一定**概率**接受它
  - 在刚开始迭代时，这个概率可以高一些，以**探索**解空间
  - 在快结束迭代时，这个概率可以低一些，以**利用**当前解
- 探索（**exploration**）与利用（**exploitation**）间的平衡

模拟退火

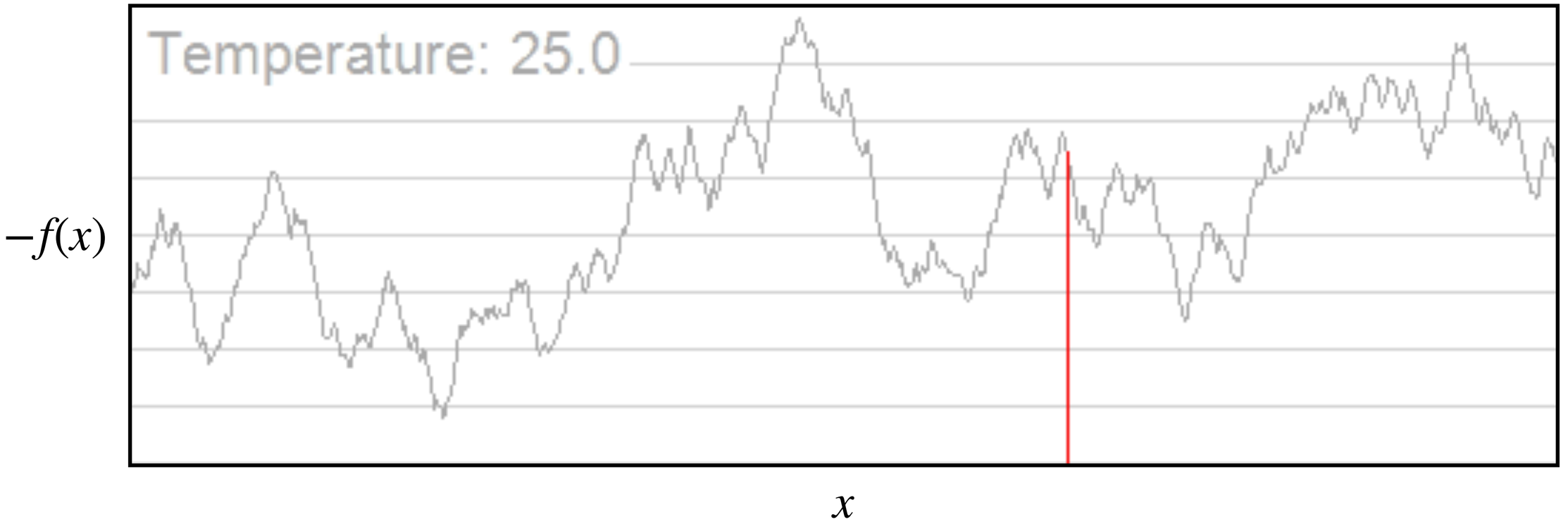
贪心局部搜索 + 随机游走

- $x_0 \leftarrow$ 初始解
- 迭代 m 次，其中第 i 次迭代在 x_{i-1} 的邻居中选出 x_i
  - $T \leftarrow \text{temparature}(1 - i/m)$ 温度 T 随着迭代次数增多而下降
  - $x' \leftarrow \text{neighbor}(x_{i-1}, T)$ 随机选择一个邻居，通过参数 T 控制探索-利用
  - 若 $f(x') < f(x_{i-1})$，则令 $x_i \leftarrow x'$
  - 若不然，则以概率 $e^{(f(x_{i-1})-f(x'))/T}$ 令 $x_i \leftarrow x'$，否则令 $x_i \leftarrow x_{i-1}$

接受概率与温度正相关，与解变差的程度负相关

模拟退火

贪心局部搜索 + 随机游走

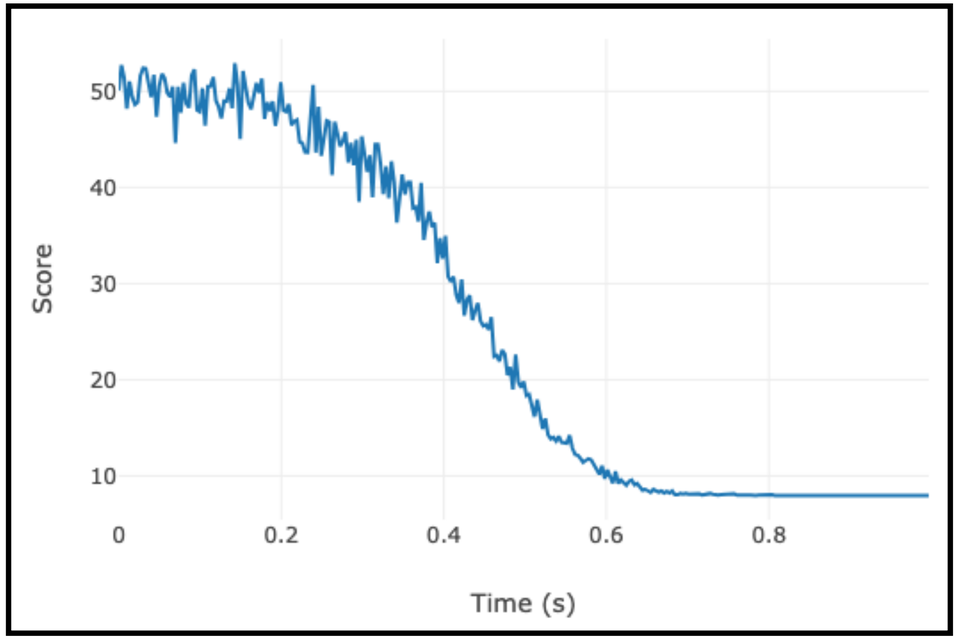


动图来源：https://en.wikipedia.org/wiki/File:Hill_Climbing_with_Simulated_Annealing.gif

喵喵花园

模拟退火

- 解空间 $\mathcal{X}$：多边形 P 上的点
- 评价函数 $f(x)$：以 x 为顶点的多边形 Q 的面积
- 温度控制：初始温度为 P 的周长，接下来每次迭代乘上一个常量（例如 0.999）
- 邻居选择 $\text{neighbor}(x, T)$
  - 随机把 x 顺/逆时针移动不超过 T 的距离
  - 开始时距离大，结束时距离小



图片来源：https://bminaiev.github.io/simulated-annealing

炸弹攻击

洛谷 P5544

- 给定二维平面上 N 个圆形建筑和 M 个点状敌人
- 画一个半径不超过 R 的圆 C，使得
  - C 不与任何的圆形建筑相交
  - C 覆盖尽可能多的点状敌人
- 找出 C 最多能覆盖多少敌人
- $0 \leq N \leq 10$，$0 < M \leq 10^3$

炸弹攻击
作为一个优化问题

- 给定二维平面上 N 个圆形建筑和 M 个点状敌人
- 画一个半径不超过 R 的圆 C，使得
  - C 不与任何的圆形建筑相交
  - C 覆盖尽可能多的点状敌人
- 找出 C 最多能覆盖多少敌人
- $0 \leq N \leq 10$，$0 < M \leq 10^3$

解空间 $\mathcal{X}$：半径不超过 R 且与所有建筑不相交的圆

评价函数 $f(x)$：圆 x 不能覆盖的敌人的数目

能模拟退火吗？

炸弹攻击
模拟退火

- 能模拟退火，但我们可以做得更好
- 解空间 $\mathcal{X}$：平面上不在建筑内的点
  - 在不碰到建筑的前提下半径越大越好，所以根据点的选择可直接计算半径
- 评价函数 $f(x)$：圆 x 不能覆盖的敌人数目 + 最近一个 x 不能覆盖的敌人的距离
  - 在覆盖敌人数目相同的解中，倾向于离不能覆盖的敌人近一些的解
  - 让评价函数更加**平滑**

方差

洛谷 P7962

- 给定数列 $1 \le a_1 \le a_2 \le \cdots \le a_n$
- 每次操作任选一个正整数 $1 < i < n$，把 a_i 变成 $a_{i-1} + a_{i+1} - a_i$
- 求若干次操作后，数列方差的最小值是多少
- $1 \le n \le 10^4$，$1 \le a_i \le 600$

方差

模拟退火 - 普通版

- 解空间 $\mathcal{X}$：多次操作 a 后可得到的数列
- 评价函数 $f(x)$：数列 x 的方差
- 邻居选择 $\text{neighbor}(x)$：随机对数列 x 进行一次操作（忽略温度 T）

- 如何提升效果？

方差

模拟退火 - Pro 版

- 一次操作等价于在差分序列上交换相邻项
  - $a_{i-1}, a_i, a_{i+1} \Rightarrow a_{i-1}, (a_{i-1}+a_{i+1}-a_i), a_{i+1}$
  - $(a_i-a_{i-1}), (a_{i+1}-a_i) \Rightarrow (a_{i+1}-a_i), (a_i-a_{i-1})$
- 解空间 $\mathcal{X}$：数列 a 的差分序列的所有排列
- 邻居选择 $\text{neighbor}(x)$：随机交换差分序列 x 中的两个元素
  - 一次交换等价于多次操作

方差

模拟退火 - Pro Max 版

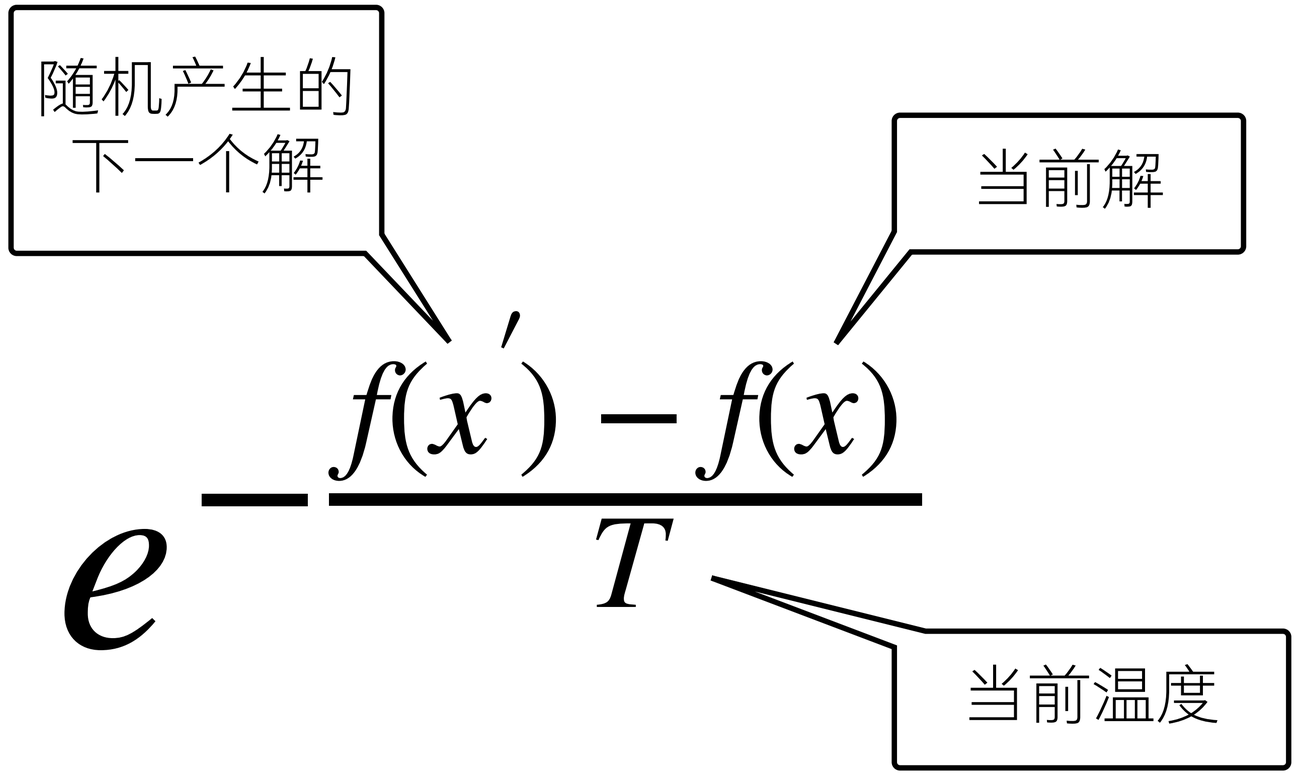
- 推式子（或找规律）可知，存在最优解的差分序列呈现先减后增趋势
- 解空间 $\mathcal{X}$：数列 a 的差分序列的所有先减后增的排列
- 邻居选择 $\text{neighbor}(x)$：随机将差分序列 x 一边「斜坡」上的值移动到另一边

- 总结：探索-利用平衡
  - 尽可能把每个解**利用**到极致，比如找出最优解应当满足的必要条件
  - 从而有更多资源**探索**解空间，在时限内有更大概率找到全局最优解

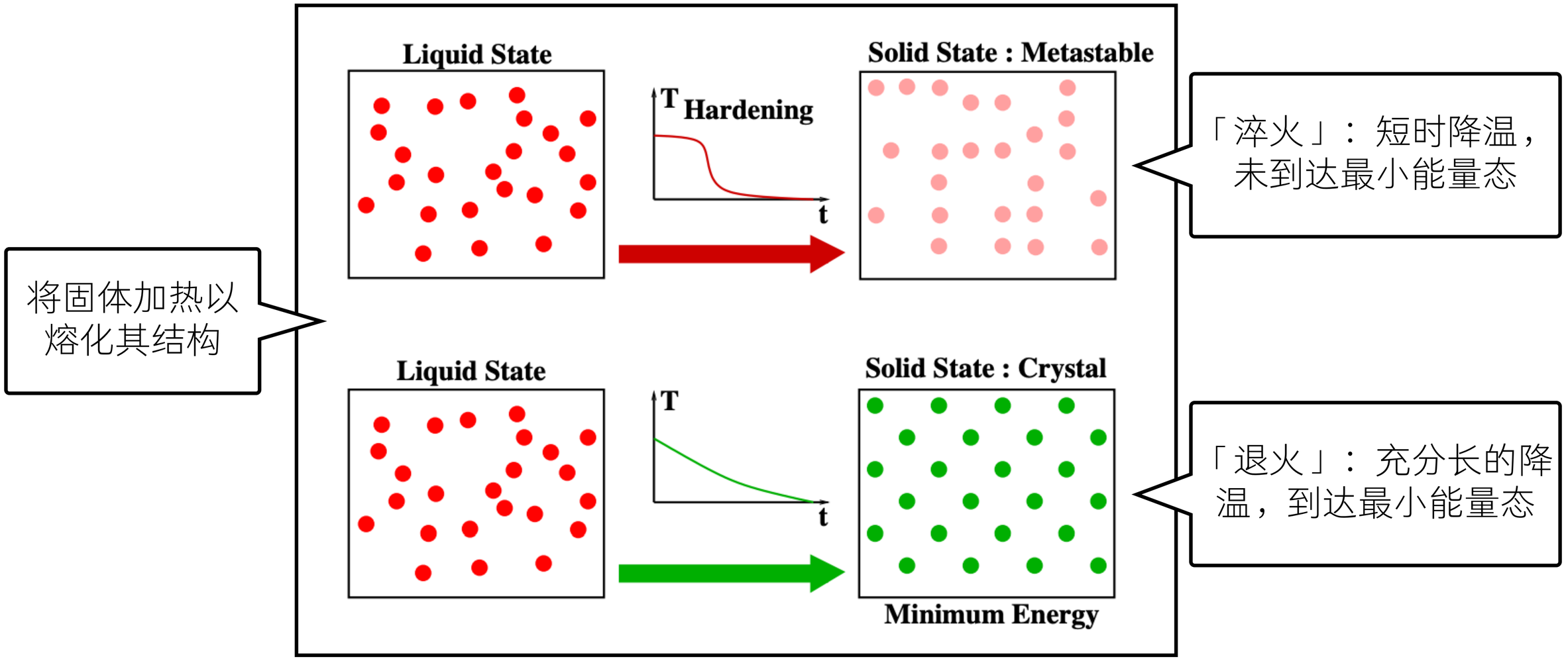
模拟退火

还记得那个接受概率公式吗？

这个概率怎么来的？



「退火」



图片来源：D. Delahaye, S. Chaimatanan, and M. Mongeau. 2019. Simulated Annealing: From Basics to Applications. *Handbook of Metaheuristics*.

模拟「退火」

最初真的是在模拟「退火」

- **The Metropolis Algorithm** 被誉为 20 世纪的 top 10 算法之一
- 固体处于具有能量 E_i 的状态 i，改变一个粒子的位置后处于具有能量 E_j 的状态 j
- 若能量差 $E_i - E_j$ 为正，则设置 j 为当前状态
- 若不然，则以概率 $e^{(E_i - E_j)/(k_B \cdot T)}$ 设置 j 为当前状态
  - k_B 为**玻尔兹曼常数**
- 如果降温足够慢（在每个温度进行多次状态转移），固体会到达平衡态
  - 热力学平衡态中，固体的状态服从**玻尔兹曼分布**

$$p_i \propto e^{-\frac{E_i}{k_B \cdot T}}$$

算法为什么正确?

- 「平衡态」：在温度 T 上状态转移**充分多**次，可证明当前状态 X 服从概率分布

$$\mathbb{P}(X = i) \propto e^{-\frac{E_i}{k_B \cdot T}}$$

- 当温度 T 趋近于零时，$-E/(k_B \cdot T)$ 对于 E 的变化愈发敏感
  - 即便 E_j 只比 E_i 大一点，都会有 $\mathbb{P}(X = j)$ 远小于 $\mathbb{P}(X = i)$
- 当温度 T 趋近于零时，当前状态 X 的概率分布聚集在最小能量态上
  - 在模拟退火算法中，一个**解**对应一个**状态**，**评价**函数对应**能量**函数
  - 寻找到**全局最优解**对应退火至**最小能量态**

到达「平衡态」

等等，这好像是个不得了的能力

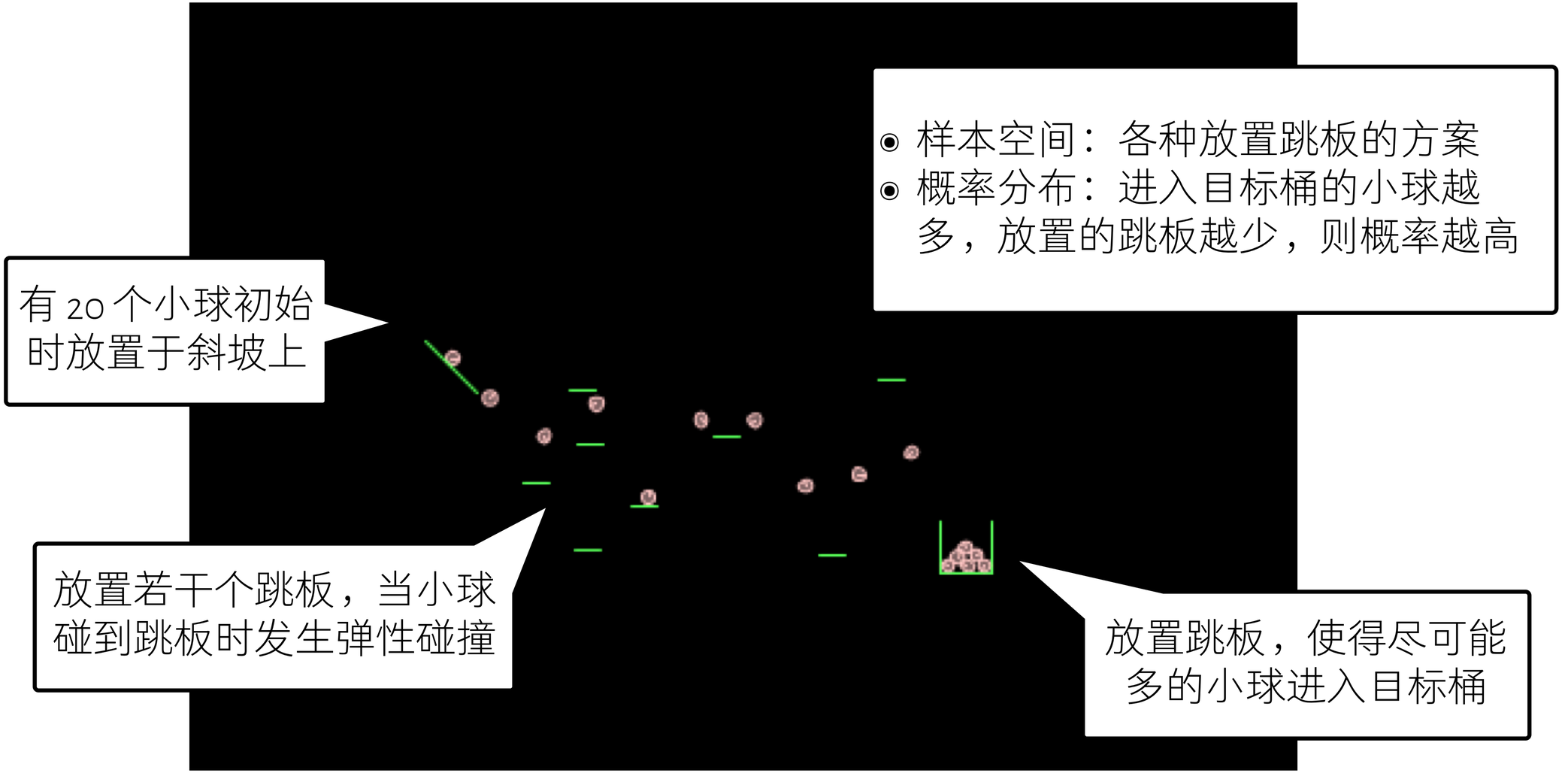
- 「平衡态」：在温度 T 上状态转移**充分多**次，可证明当前状态 X 服从概率分布

$$\mathbb{P}(X = i) \propto e^{-\frac{E_i}{k_B \cdot T}}$$

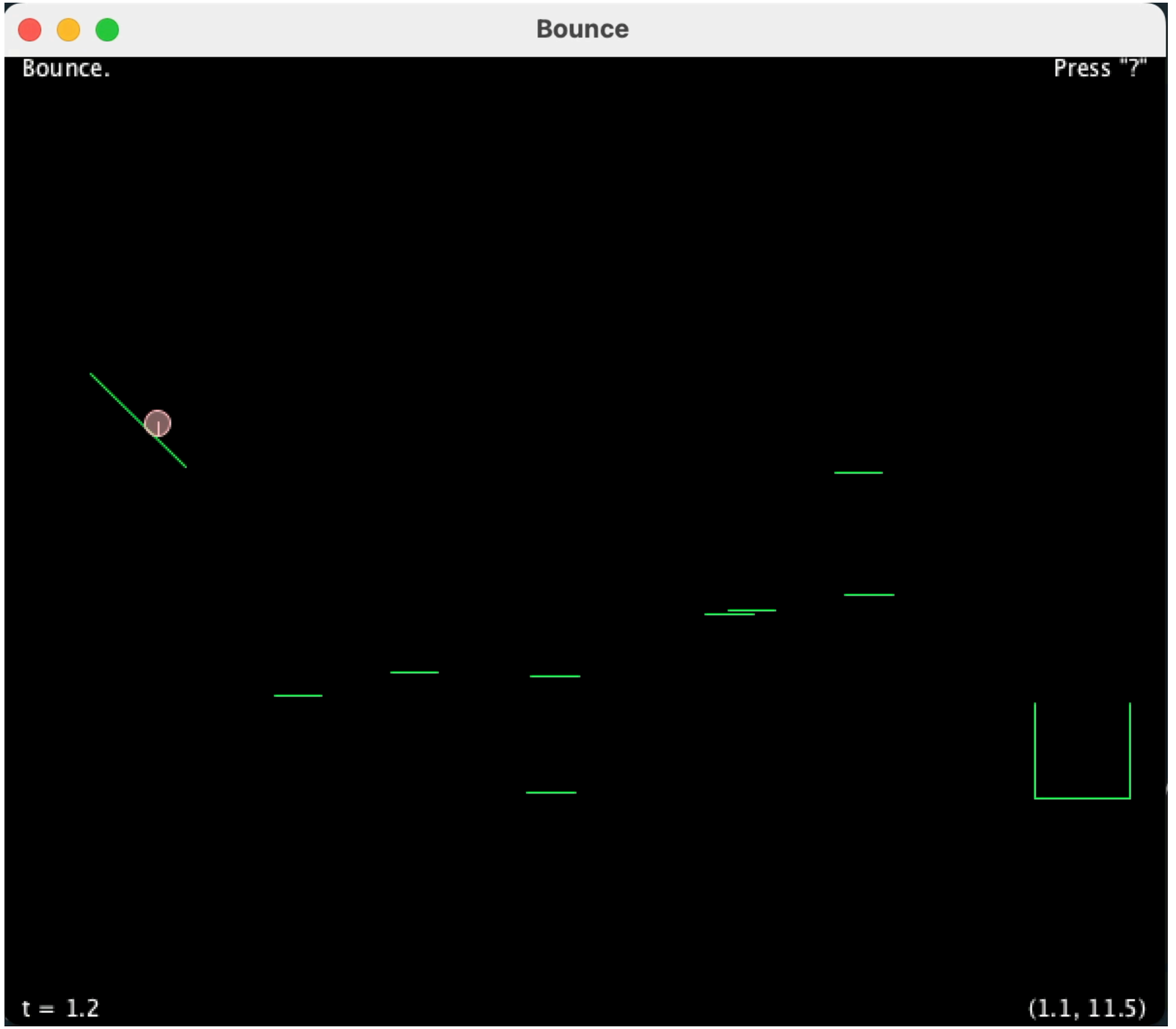
- 即使不知道概率分布 $\mathbb{P}$ 的具体形式，通过迭代算法可以达到对其采样的目的
  - **The Metropolis-Hastings Algorithm**
  - 推广转移概率 $e^{(E_i - E_j)/(k_B \cdot T)}$ 并得出一般形式，支持各种各样的概率分布

- 不过首先，能对未知概率分布采样有啥用吗？

机制设计
Mechanism Design

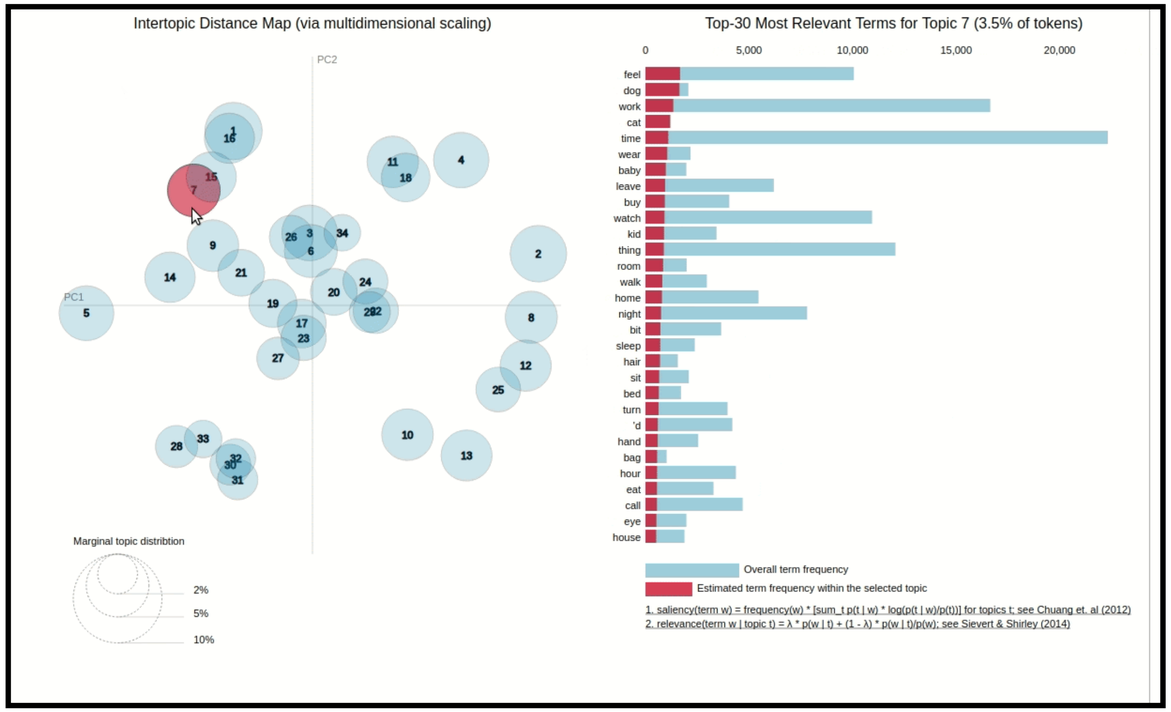



Bounce
Bounce.
Press "?"
t = 1.1
(6.8, 8.7)


Bounce
Bounce.
Press "?"
t = 1.2
(1.1, 11.5)

文本分类
Text Classification

- 样本空间：每个主题中的单词概率
- 概率分布：与给出的若干文本贴合度越高，则概率越高



文本分类可视化演示

有若干个主题，每个主题都有一个概率

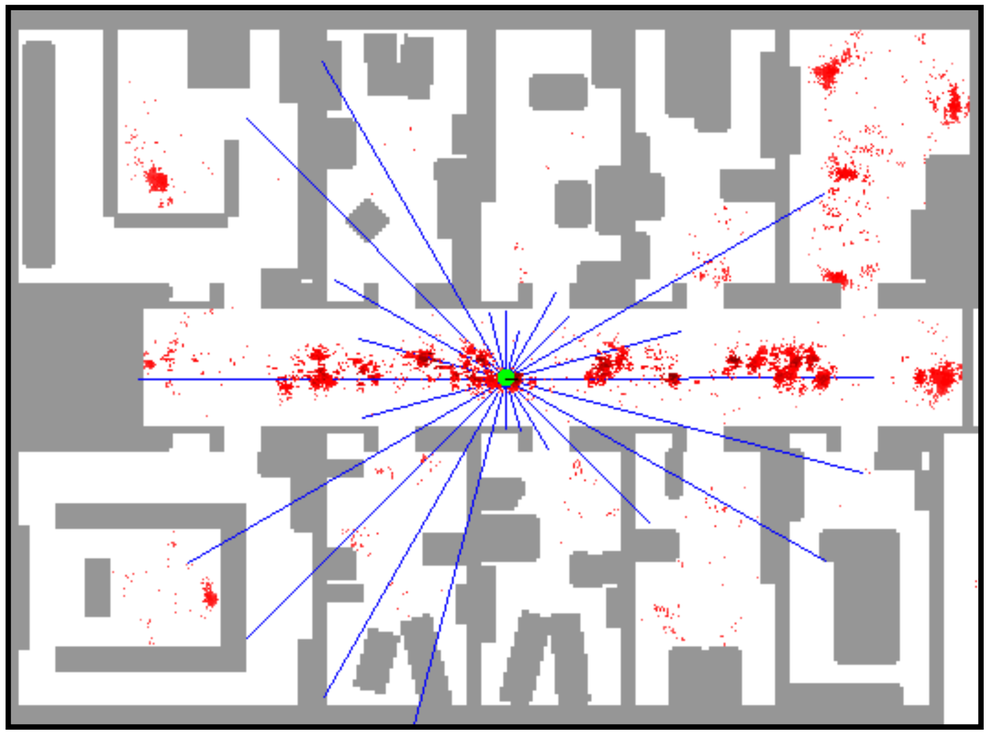
IAGO: 'Sblood, but you'll not hear me. If ever I did dream
Of such a matter, abhor me.
RODERIGO: Thou told'st me
Thou didst hold him in thy hate.
IAGO: Despise me
If I do not. Three great ones of the city,
In personal suit to make me his lieutenant,
Off-capped to him; and, by the faith of man,
I know my price, I am worth no worse a place.
But he, as loving his own pride and purposes,
Evades them with a bombast circumstance,
Horribly stuffed with epithets of war,
And in conclusion,
Nonsuits my mediators. For "Certes," says he,
"I have already chose my officer."
And what was he?
Forsooth, a great arithmetician,
One Michael Cassio, a Florentine,
A fellow almost damned in a fair wife,
That never set a squadron in the field,
Nor the division of a battle knows

对于每个主题，每个单词出现的概率可以不同

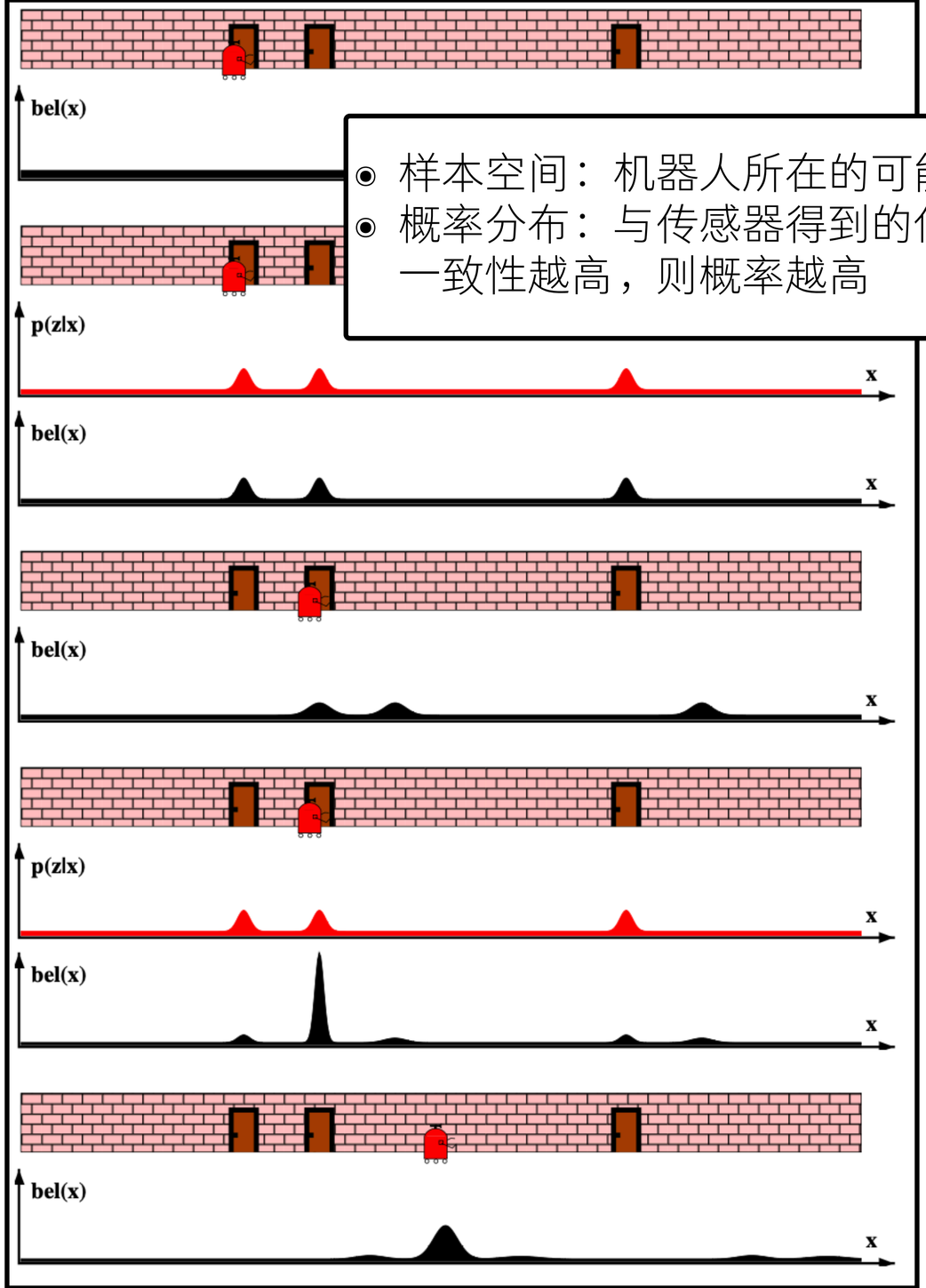
Topics					
negative	0.04	money	0.04	military	0.04
hate	0.02	worth	0.02	conflict	0.02
bad	0.01	purse	0.01	officer	0.02

动图来源：https://neptune.ai/blog/pyldavis-topic-modelling-exploration-tool-that-every-nlp-data-scientist-should-know
图片来源：https://www.ibm.com/think/topics/latent-dirichlet-allocation

概率定位
Probabilistic Localization



使用 24 个声呐传感器进行概率定位



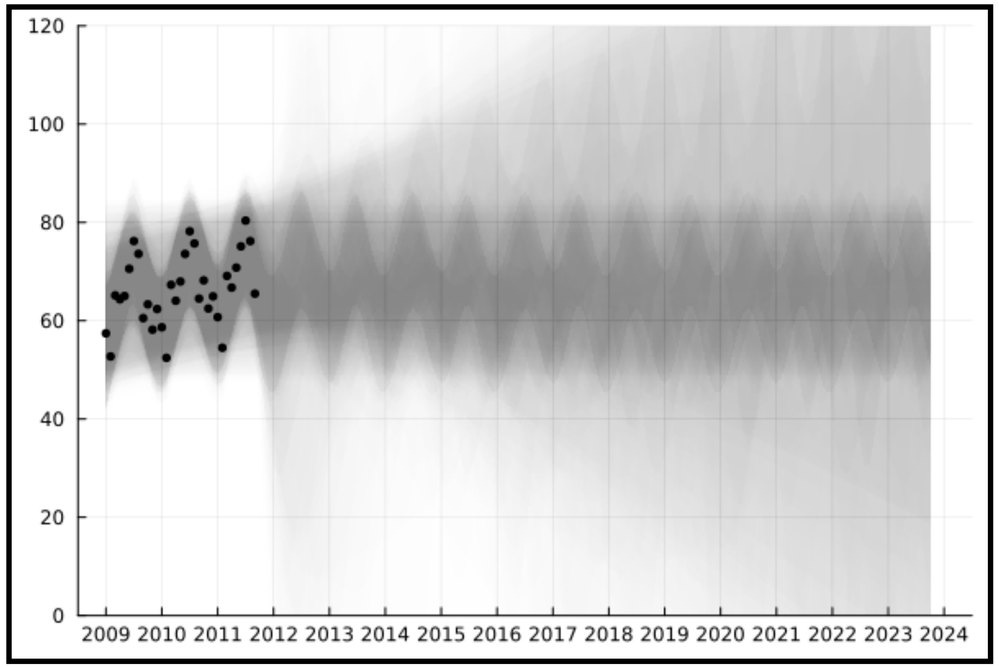
- 样本空间：机器人所在的可能位置
- 概率分布：与传感器得到的信息的一致性越高，则概率越高

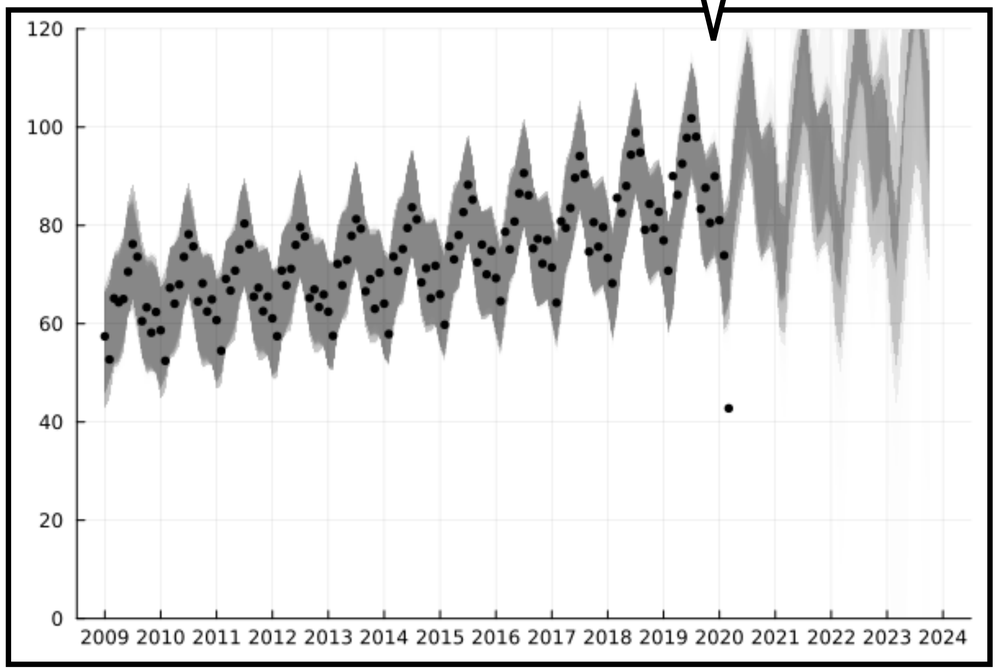
动图来源：https://rse-lab.cs.washington.edu/projects/mcl/
图片来源：S. Thrun, W. Burgard, and D. Fox. 2005. Probabilistic Robotics. *MIT Press.*

时间序列
Time Series

- 样本空间：平面上的所有月份-里程的走势折线
- 概率分布：折线拟合的数据点越多、越准确，则概率越高

对未来趋势的预测
对未来变化的适应



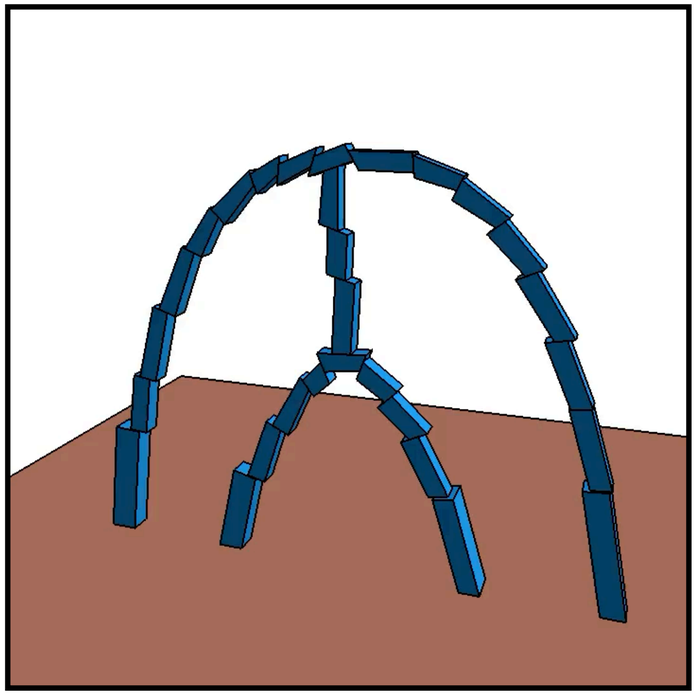


美国国内民航总里程（按月份）

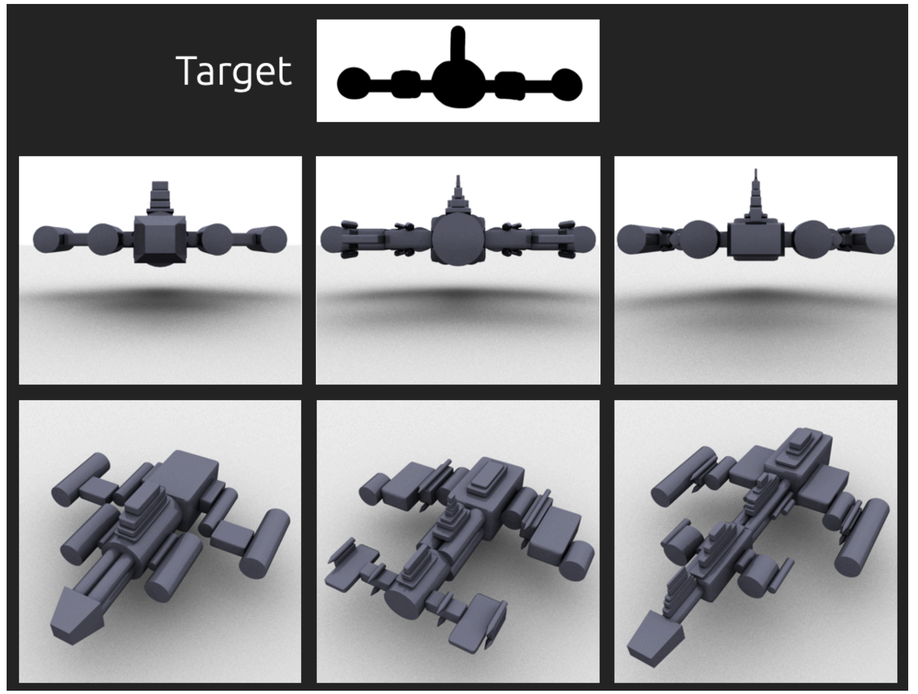
数据来源：https://fred.stlouisfed.org/series/RPM

过程化设计
Procedural Design

- 样本空间：所有潜在的可能的设计
- 概率分布：达成目标越好（比如越稳定、越对称等）则概率越高



从一个初始结构出发
设计具有稳定结构的积木结构



给定一个前视图
设计一个具有该前视图的飞船

动图和图片来源：D. Ritchie. 2016. Probabilistic Programming for Procedural Modeling and Design. *Ph.D. Thesis.*

验证码破解
Captcha Breaking

Facebook Captcha

Observed images

(W4kgvQ) (uV7FeWB) (MqhnpT)

Inference

Training traces	10^7	W4kgvQ	uV7IeWB	MqbnpT
	10^6	WA4ryvQ	uV7FoWB	MyphppT
	10^5	WpxSX1u	mTEmwWM	RfppXf
	10^4	bQTXhf1	zuTop3D	ytyCHO

验证码破解演示

- 样本空间：验证码字符串
- 概率分布：渲染后与图片越接近，则概率越高

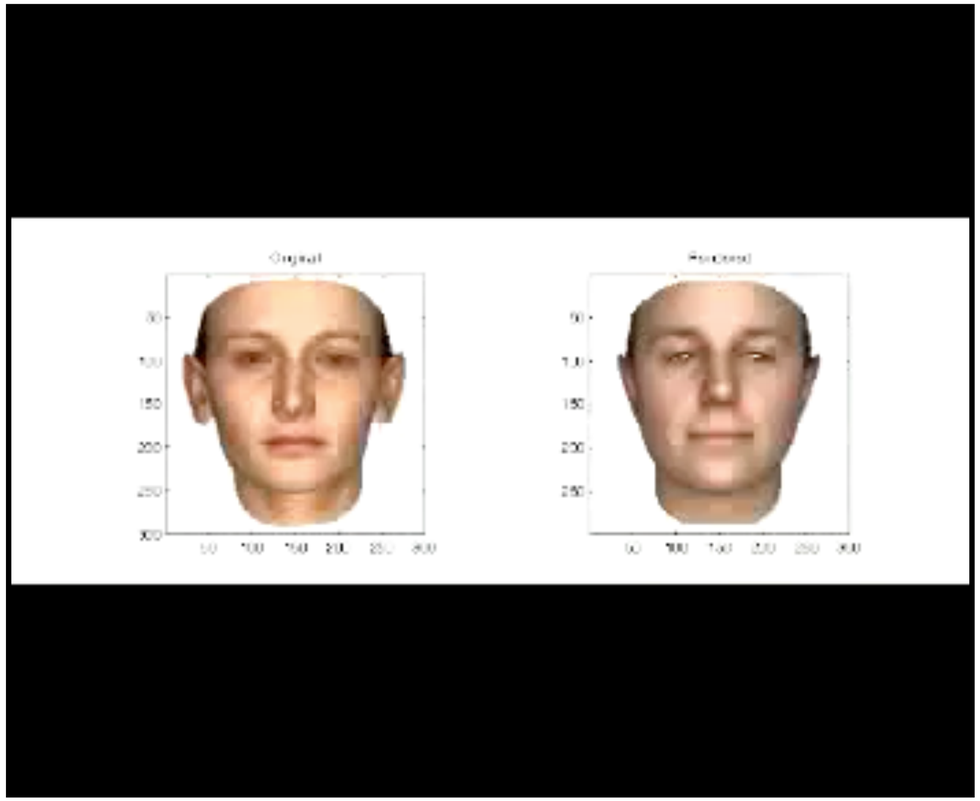
SMKBDF

渲染过程从字符串随机生成一张验证码图片

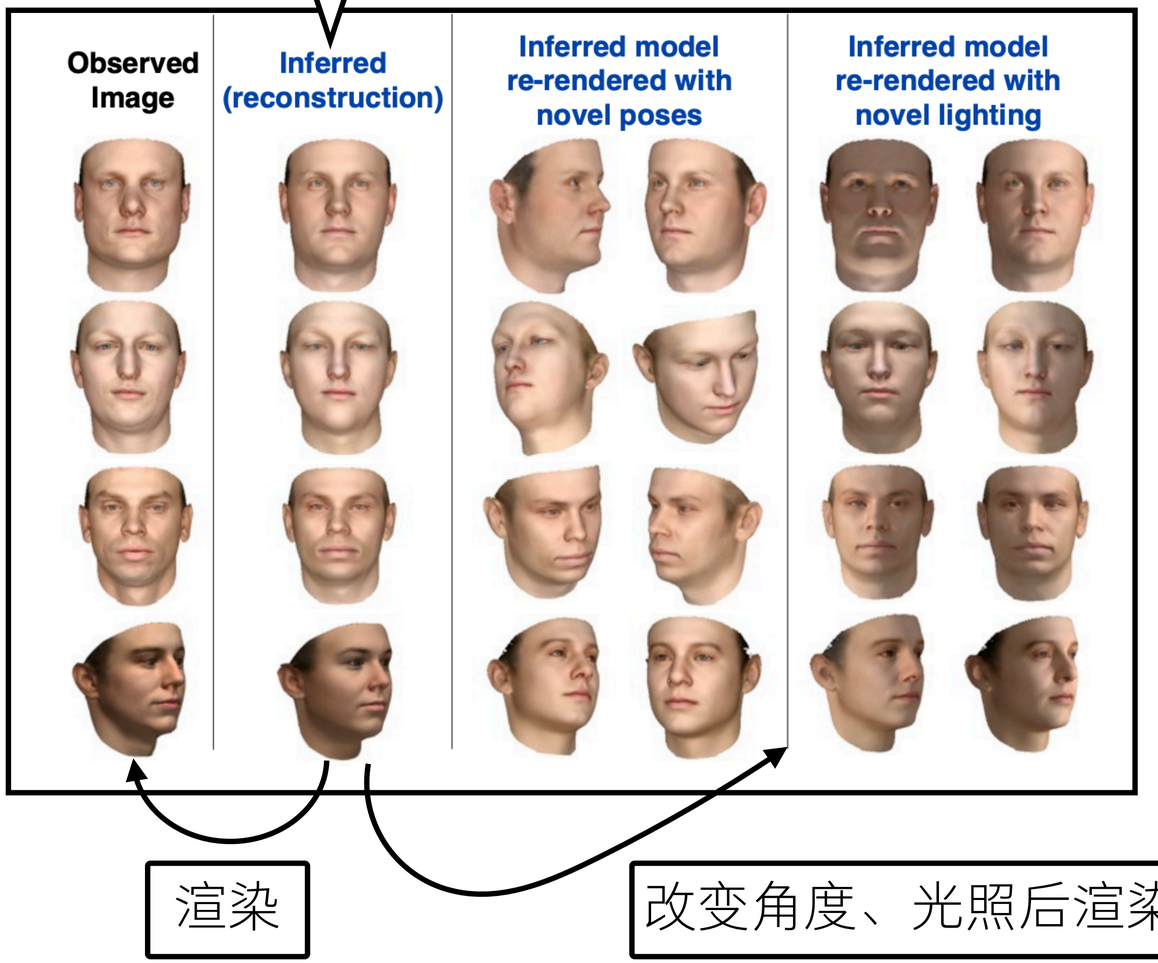
动图和图片来源：T. A. Le, A. G. Baydin, and F. Wood. 2017. Inference Compilation and Universal Probabilistic Programming. In *AISTATS'17*.

逆向图形学
Inverse Graphics

- 样本空间：人脸的 3D 结构、扫描的角度、光照情况等不同的组合
- 概率分布：渲染后与扫描图越接近，则概率越高



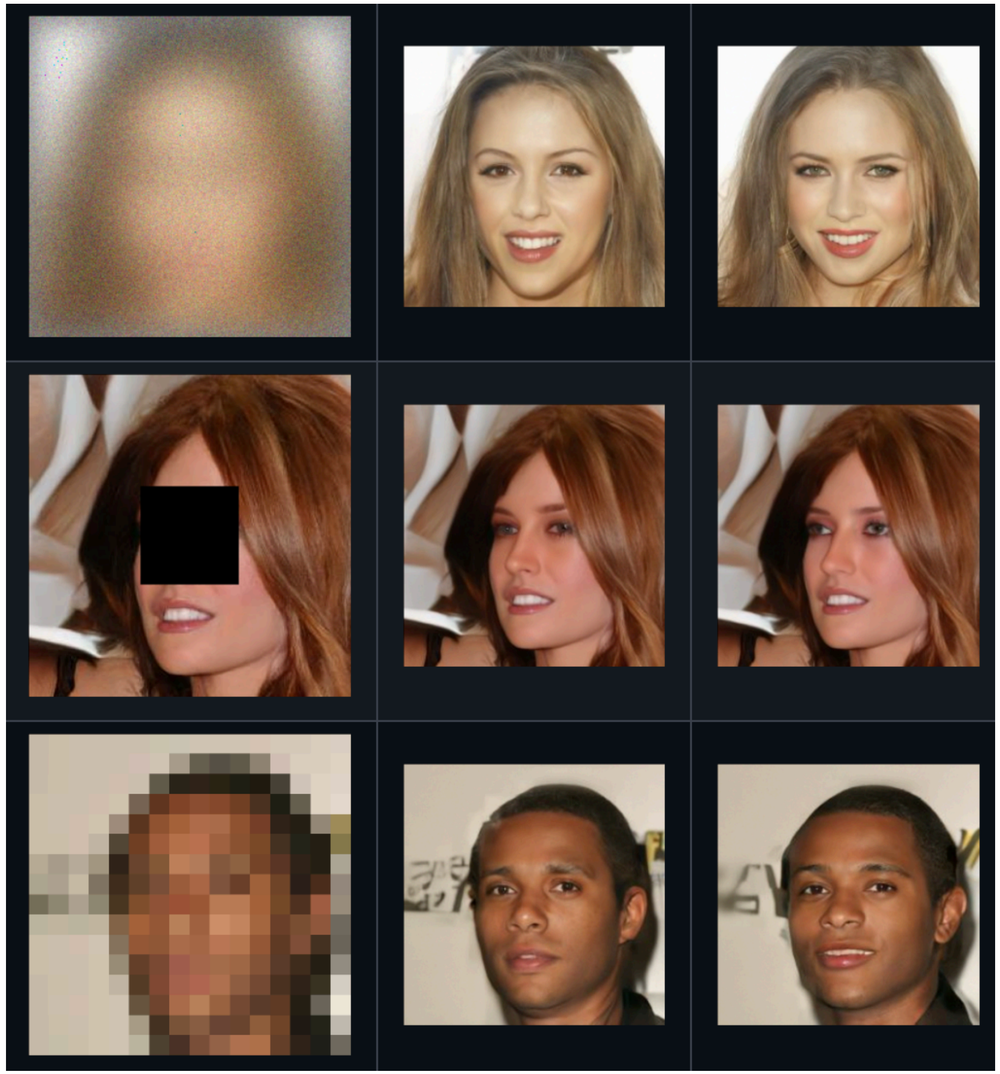
人脸扫描 3D 建模演示



动图和图片来源：T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. Mansinghka. 2015. Picture: A Probabilistic Programming Language for Scene Perception. In *CVPR'15*.

图片复原
Image Restoration

- 样本空间：所有可能的复原图
- 概率分布：与输入图越接近，则概率越高
- 其实，Stable Diffusion（图片生成模型）和 GPT（文本生成模型）的理论中都涉及概率分布



图片来源：G. Cardoso, Y. J. E. Idrissi, S. L. Corff, and E. Moulines. 2024. Monte Carlo guided Diffusion for Bayesian linear inverse problems. In *ICLR'24*.

以上问题是否有共性?

除了都可以建模成对未知概率分布采样之外?

- 如何描述一个未知的概率分布?
- 首先，要有一个**样本空间**
  - 放置跳板的方案、机器人所在位置、潜在的可能设计、人脸的 3D 结构
- 然后，要有一个概率高低的**衡量指标**
  - 进入桶的球多、与传感器信息一致、设计达到目标、人脸结构渲染图接近扫描图
- 还有吗?
  - 在解决问题前，我们可以先添加一些**先验信息**，表示「大概哪些样本概率更高」

概率论101
为了严谨地进行后面的讨论

- 样本空间 Ω：在随机过程/现象中可能出现的结果集合
- 样本 $\omega \in \Omega$ 通常表示不可再细分的结果
  - 例如：掷骰子的样本空间为 $\Omega = \{1,2,3,4,5,6\}$
- 事件 $A \subseteq \Omega$ 是样本空间的子集
  - 例如：掷骰子为偶数对应事件 $A = \{2,4,6\}$
- 事件域 $\mathcal{F}$ 是所有我们感兴趣的事件的集合
  - 当样本空间有限（或离散）时，通常可考虑所有 Ω 子集构成的集合

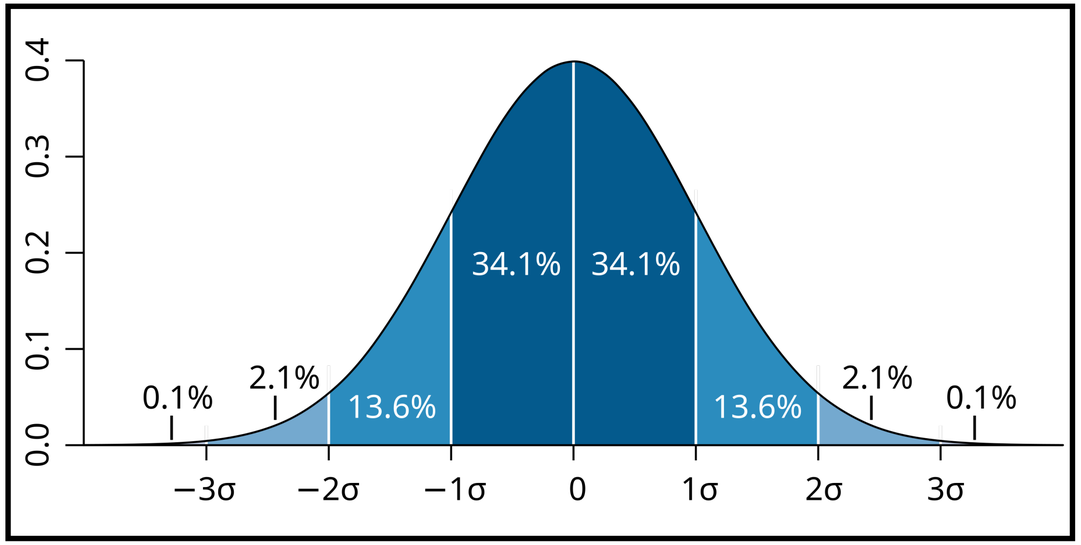
概率论 101
概率空间

- 概率分布 $\mathbb{P} : \mathscr{F} \to [0,1]$ 为事件到实数的映射，满足 $\mathbb{P}(\Omega) = 1$ 且
  - 对任意两两不交的事件 $A_1, A_2, \cdots$，都有 $\mathbb{P}(\bigcup_{i \geq 1} A_i) = \sum_{i \geq 1} \mathbb{P}(A_i)$
- 当样本空间 Ω 有限（或离散）且事件域包含所有 Ω 子集时，可以发现
  - 存在**概率质量函数** $p : \Omega \to [0,1]$，使得 $\mathbb{P}(A) = \sum_{\omega \in A} p(\omega)$
  - 例如：掷均匀骰子对应 $p(1) = p(2) = p(3) = p(4) = p(5) = p(6) = \frac{1}{6}$
- 概率空间即三元组 $(\Omega, \mathscr{F}, \mathbb{P})$
  - 在很多场景下，我们可以使用二元组 (Ω, p)

概率论 101

样本空间连续的情况呢?

- 连续样本空间很常见，例如实数 $\mathbb{R}$
  - 严格定义需要微积分和测度论
- 考虑一种特殊情形，可以用**概率密度函数** $p : \Omega \to [0, +\infty)$ 定义概率空间
- 例如：标准正态分布 $p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$
  - 事件 $[l, r]$ 的概率 $\mathbb{P}([l, r])$ 为 $p(x)$ 从 l 到 r 与横轴围成的面积



图片来源：https://commons.wikimedia.org/wiki/File:Standard_deviation_diagram.svg

NestedRandomness

TCO05 Qual 5, Div 1 - Level 3

- 考虑 $rand(N)$ 函数以均匀的概率返回 $0, \cdots, N-1$ 中的一个整数
- 计算嵌套调用 *nestings* 次后，返回 *target* 的概率
  - 调用一次是 $rand(N)$，调用两次是 $rand(rand(N))$，……
- 样本空间是什么？
- 概率分布是什么？

NestedRandomness

TCO05 Qual 5, Div 1 - Level 3

- 样本空间 $\Omega = \{(n_1, n_2, \cdots, n_{nestings}) \mid N > n_1 > n_2 > \cdots > n_{nestings}\}$
- 使用概率质量函数来定义概率分布：
  - $p(n_1, n_2, \cdots, n_{nestings}) = \frac{1}{N} \times \frac{1}{n_1} \times \frac{1}{n_2} \times \cdots \times \frac{1}{n_{nestings-1}}$
- 计算 $\sum_{n_1, n_2, \cdots, n_{nestings-1}} p(n_1, n_2, \cdots, n_{nestings-1}, target)$
  - 对应的事件为
    $A = \{(n_1, n_2, \cdots, n_{nestings-1}, target) \mid N > n_1 > n_2 > \cdots > n_{nestings-1} > target\}$
  - 本质上是计算 $\mathbb{P}(A)$

NestedRandomness

TCO05 Qual 5, Div 1 - Level 3

$$\mathbb{P}(A) = \sum_{N>n_1>n_2>\cdots>n_{nestings-1}>target} p(n_1, n_2, \cdots, n_{nestings-1}, target)$$

$$= \sum_{n_1=target+nestings-1}^{N-1} \sum_{n_2=target+nestings-2}^{n_1-1} \cdots \sum_{n_{nestings-1}=target+1}^{n_{nestings-2}-1} \frac{1}{N} \times \frac{1}{n_1} \times \frac{1}{n_2} \times \cdots \times \frac{1}{n_{nestings-1}}$$

$$= \frac{1}{N} \sum_{n_1=target+nestings-1}^{N-1} \frac{1}{n_1} \sum_{n_2=target+nestings-2}^{n_1-1} \frac{1}{n_2} \cdots \sum_{n_{nestings-1}=target+1}^{n_{nestings-2}-1} \frac{1}{n_{nestings-1}}$$

- 动态规划：$dp(i,j)$ 表示 $n_{i-1} = j$ 时，从 n_i 开始的求和式的值
  - 初始时 $dp(nestings, j) = 1$ 对 $j > target$，结束时答案为 $\frac{1}{N} \times dp(1,N)$
  - $dp(i,j) = \sum_{k=target+nestings-i}^{j-1} \frac{1}{k} \times dp(i+1,k) = dp(i,j-1) + \frac{1}{j-1} \times dp(i+1,j-1)$

概率论 101
事件的运算

- 事件的补集：$\overline{A}$ 表示「A 不发生」的事件
  - 例如：「掷骰子为偶数」的补集是「掷骰子为奇数」
- 事件的并集：$A \cup B$ 表示「A 发生或 B 发生」的事件
  - 例如：「掷骰子为偶数」与「掷骰子为奇数」的并集是整个样本空间
- 事件的交集：$A \cap B$ 表示「A 发生且 B 发生」的事件
  - 例如：「掷骰子为偶数」与「掷骰子为质数」的交集是 $\{2\}$

概率论 101
概率的运算

- $\mathbb{P}(\overline{A}) = 1 - \mathbb{P}(A)$
  - 推导：$\mathbb{P}(\overline{A}) = \mathbb{P}(\Omega \backslash A) = \mathbb{P}(\Omega) - \mathbb{P}(A) = 1 - \mathbb{P}(A)$
- $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$
  - 推导：$\mathbb{P}(A \cup B) = \mathbb{P}((A \backslash (A \cap B)) \cup (B \backslash (A \cap B)) \cup (A \cap B))$
  - 当 $A \cap B = \varnothing$，即「A 与 B **互斥**」时，$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$
- 什么情况下能有 $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$?

概率论 101

条件概率

- 如果事件 A 满足 $\mathbb{P}(A) > 0$，那么对任意事件 B，可按照如下方式定义「已知 A 发生的条件下 B 发生」的概率 $\mathbb{P}(B \mid A)$：

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}$$

- 例如：考虑掷均匀骰子，$A = \{2,4,6\}$ 表示「掷出偶数」，$B = \{1,2,3\}$ 表示「掷出小于 4」，则有 $\mathbb{P}(B \mid A) = (1/6)/(1/2) = 1/3$

- $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B \mid A) = \mathbb{P}(B)\mathbb{P}(A \mid B)$
  - 当 $\mathbb{P}(B \mid A) = \mathbb{P}(B)$（或 $\mathbb{P}(A \mid B) = \mathbb{P}(A)$）时，有 $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$
  - 此时称「A 与 B **独立**」

概率论 101
条件概率

- 注意，条件概率 $\mathbb{P}(B \mid A)$ **不代表**「因为 A 所以 B」或「先 A 后 B」
  - 例如：B 为「感染了病毒」，A 为「检测呈阳性」，$\mathbb{P}(B \mid A)$ 仍是有意义的
- 给定 $\mathbb{P}(A) > 0$，可证明函数 $B \mapsto \mathbb{P}(B \mid A)$ 也是一个概率分布
  - 所以条件概率也满足各种运算规则
- 例如：$\mathbb{P}(\overline{B} \mid A) = 1 - \mathbb{P}(B \mid A)$
  - 推论：若 A 与 B 独立，则 A 与 $\overline{B}$ 也独立
  - $\mathbb{P}(\overline{B} \mid A) = 1 - \mathbb{P}(B \mid A) = 1 - \mathbb{P}(B) = \mathbb{P}(\overline{B})$

QuizShow

TopCoder SRM 223, Div 1 - Easy

- 三名玩家，第 i 名玩家有分数 s_i，可以下注 $w_i \leq s_i$，然后回答问题，回答正确加 w_i 分，回答错误减 w_i 分，最后分数高的玩家获胜
- 事件「第 i 名玩家答对」相互独立，概率为 p_i
- 已知所有玩家的分数和 w_1、w_2，请问零号玩家下注多少能最大化获胜概率？

- 样空空间是什么？
- 概率分布是什么？

QuizShow

TopCoder SRM 223, Div 1 - Easy

- 样本空间 $\Omega = \{(b_0, b_1, b_2) \mid b_i = \top \text{ 或 } b_i = \bot\}$，$\top$ 表示答对，$\bot$ 表示答错
  - 事件「零号玩家答对」$B_0 = \{(\top, b_1, b_2) \mid b_i = \top \text{ 或 } b_i = \bot\}$
- 对第 i 名玩家有 $\mathbb{P}(B_i) = p_i$，且 B_i 相互独立
  - $\mathbb{P}(B_0 \cap B_1 \cap B_2) = p_1 p_2 p_3$，$\mathbb{P}(\overline{B_0} \cap B_1 \cap B_2) = (1 - p_1) p_2 p_3$，……
- 算法：枚举零号玩家的下注 w_0，然后通过枚举所有样本计算获胜概率

BirthdayOdds

TopCoder SRM 174, Div 1 - Easy

- 生日悖论：若房间里有至少 23 个人，则有大于 50 % 的概率有两人同月同日生
  - 应该有怎样的独立性假设？
  - 样本空间和概率分布是什么？

- 考虑有这么一个星球，它每年有 *daysInYear* 天
- 请问至少要多少人，才能促成「生日悖论」中的概率至少为 *minOdds* % ？

BirthdayOdds

TopCoder SRM 174, Div 1 - Easy

- 样本空间 $\Omega = \{(d_1, d_2, \cdots, d_k) \mid 1 \le d_i \le daysInYear\}$，其中 k 表示人数
  - 事件「第 i 个人出生在第 j 天」
    $B_{i,j} = \{(d_1, \cdots, d_{i-1}, j, d_{i+1}, \cdots, d_k) \mid d_1, \cdots, d_{i-1}, d_{i+1}, \cdots, d_k \le daysInYear\}$
- 独立性假设：对任意 $d_1, d_2, \cdots, d_k$，事件 B_{i,d_i} 相互独立
  - $\mathbb{P}(B_{i,d_i}) = 1/daysInYear$
- 考虑事件 A 为「所有人的生日都不相同」，A_i 为「前 i 个人的生日都不相同」
  - $\mathbb{P}(A_k) = \mathbb{P}(A)$，$\mathbb{P}(A_1) = 1$
  - $\mathbb{P}(\overline{A}) = 1 - \mathbb{P}(A)$ 即「有两人同月同日生」的概率

BirthdayOdds

TopCoder SRM 174, Div 1 - Easy

$$
\begin{aligned}
\mathbb{P}(A_i) &= \sum_{d_1,\cdots,d_k | d_1,\cdots,d_i \text{ 両両不同}} \mathbb{P}(B_{1,d_1}) \cdots \mathbb{P}(B_{k,d_k}) \\
&= \sum_{d_1,\cdots,d_i \text{ 両両不同}} \mathbb{P}(B_{1,d_1}) \cdots \mathbb{P}(B_{i,d_i}) \\
&= \sum_{d_1,\cdots,d_{i-1} \text{ 両両不同}} \sum_{j \neq d_1,\cdots,d_{i-1}} \mathbb{P}(B_{1,d_1}) \cdots \mathbb{P}(B_{i-1,d_{i-1}}) \mathbb{P}(B_{i,j}) \\
&= \sum_{d_1,\cdots,d_{i-1} \text{ 両両不同}} \mathbb{P}(B_{1,d_1}) \cdots \mathbb{P}(B_{i-1,d_{i-1}}) \sum_{j \neq d_1,\cdots,d_{i-1}} \mathbb{P}(B_{i,j}) \\
&= \sum_{d_1,\cdots,d_{i-1} \text{ 両両不同}} \mathbb{P}(B_{1,d_1}) \cdots \mathbb{P}(B_{i-1,d_{i-1}}) \cdot \frac{daysInYear - (i-1)}{daysInYear} \\
&= \mathbb{P}(A_{i-1}) \cdot \frac{daysInYear - (i-1)}{daysInYear}
\end{aligned}
$$

BirthdayOdds

TopCoder SRM 174, Div 1 - Easy

- 也可以通过条件概率来推导
  - $\mathbb{P}(A_i) = \mathbb{P}(A_i \cap A_{i-1}) = \mathbb{P}(A_i \mid A_{i-1})\mathbb{P}(A_{i-1})$
  - $\mathbb{P}(A_i \mid A_{i-1})$ 表示前 $i-1$ 个人生日两两不同的条件下第 i 个人也不与他们生日相同的概率
  - 也就是 $\frac{daysInYear - (i-1)}{daysInYear}$
- 生日悖论：$\mathbb{P}(A) = \frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{365-(23-1)}{365} \approx 0.493$，$\mathbb{P}(\overline{A}) \approx 0.507$
- 算法：从小到大枚举人数 k，直到 $\prod_{i=1}^{k} \frac{daysInYear - (i-1)}{daysInYear}$ 达到 $(100 - minOdds)\,\%$

概率论 101
随机变量

- 给定概率空间 $(\Omega, \mathscr{F}, \mathbb{P})$，函数 $X : \Omega \to \mathbb{R}$ 被称为随机变量，如果
  - 对任意 $t \in \mathbb{R}$，都有 $\{\omega \in \Omega \mid X(\omega) \leq t\} \in \mathscr{F}$ 为概率空间中的事件
  - 若样本空间 Ω 有限（或离散）且事件域包含所有 Ω 子集，该要求是平凡的
  - $\mathbb{P}(X \leq t)$ 表示 $\mathbb{P}(\{\omega \in \Omega \mid X(\omega) \leq t\})$，$X \leq t$ 可以一般化为关于 X 的谓词
- 例如：考虑掷均匀骰子两次 $\Omega = \{(a, b) \mid a, b \in \{1,2,3,4,5,6\}\}$
  - $X : (a, b) \mapsto a + b$ 表示「掷出点数之和」的随机变量
  - $\mathbb{P}(X = 10) = \mathbb{P}(\{(a, b) \mid a + b = 10\}) = \mathbb{P}(\{(4,6), (5,5), (6,4)\}) = 1/12$

概率论 101
随机变量

- 若 X 是离散型随机变量，那么 X 的概率质量函数可定义为 $p_X(x_i) = \mathbb{P}(X = x_i)$
- 若 X 是连续型随机变量，那么一些情况下可以定义其概率密度函数 $p_X(x)$

- 回顾：事件 A 与 B 独立定义为 $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$
- 随机变量 X 与 Y 独立可以定义为对任意 $x, y \in \mathbb{R}$，都有
  - $\mathbb{P}(X \leq x, Y \leq y) = \mathbb{P}(X \leq x)\mathbb{P}(Y \leq y)$

概率论 101
随机变量

- 回顾：对于事件 B 在事件 A 发生的条件下的概率定义为 $\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}$
- 在已知随机变量 $X = x$ 的条件下，随机变量 Y 的条件概率分布记为 $\mathbb{P}(Y \mid X = x)$
  - 若 Y 是离散型随机变量：$p_{Y|X}(y_i \mid x) = \mathbb{P}(Y = y_i \mid X = x) = \frac{\mathbb{P}(X = x, Y = y_i)}{\mathbb{P}(X = x)}$
  - 若 Y 是连续型随机变量：一些情况下可以定义其概率密度函数 $p_{Y|X}(y \mid x)$
- 若 X 和 Y 的取值都是有限的，那么 $\mathbb{P}(Y \mid X)$ 可以用矩阵来描述

概率论101
马尔可夫链

- 马尔可夫链（Markov chain）是同一样本空间上的随机变量序列 $X_1, X_2, \cdots$，满足
  - 对任意 n 和 $x_1, x_2, \cdots, x_n$，都有条件概率分布
  - $\mathbb{P}(X_{n+1} \mid X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n)$ 等价于 $\mathbb{P}(X_{n+1} \mid X_n = x_n)$
- 例如：
  - 重复掷骰子，样本空间为点数序列，X_i 表示第 i 次掷出点数
  - 重复掷骰子，样本空间为点数序列，X_i 表示前 i 次掷出点数之和
  - 再想一些例子？

概率论 101
马尔可夫链

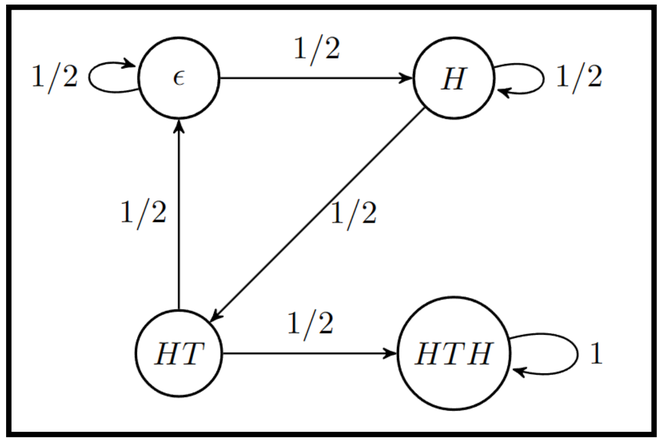
- 回顾：考虑 $rand(N)$ 函数以均匀的概率返回 $0,\cdots,N-1$ 中的一个整数
- 计算嵌套调用 *nestings* 次后，返回 *target* 的概率
- 构造马尔可夫链：随机变量 X_i 表示调用 i 次后的返回值
  - $X_0 = N$，$\mathbb{P}(X_{n+1} = k \mid X_n = x_n) = 1/x_n$ 若 $k < x_n$
- $\mathbb{P}(X_{n+1} = k) = \sum_{j=k+1} \mathbb{P}(X_{n+1} = k, X_n = j) = \sum_{j=k+1} \mathbb{P}(X_{n+1} = k \mid X_n = j)\mathbb{P}(X_n = j)$
- $\mathbb{P}(X_{n+1} = k) = \sum_{j=k+1} \frac{1}{j}\mathbb{P}(X_n = j)$，也可以进行动态规划

概率论 101
马尔可夫链

- 无限猴子定理：给猴子打字机和**充分长**的时间，那么它一定能打出三体全集
- 构造马尔可夫链：随机变量 X_i 表示打了 i 个字后，如果前面已经打出过三体，那么随机变量取值为三体的长度，否则取值为当前后缀能匹配三体前缀的长度
  - 可以证明，$\lim_{t\to\infty} \mathbb{P}(X_t = N) = 1$，其中 N 为三体的长度
- 如何构造条件概率分布 $\mathbb{P}(X_{t+1} \mid X_t)$?
  - 该概率分布与 t 无关，所以只需要考虑构造 $p(x'; x)$ 表示从匹配长度为 x 到匹配长度为 x' 的概率
  - 这是字符串匹配问题，可以用 KMP 算法或 AC 自动机构造状态机

概率论 101
马尔可夫链

- 考虑一个简化问题，字符集只有 H, T，希望打出的字符串是 HTH



$$P = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- 共有四种匹配状态，状态转移之间的有向边刻画了状态转移关系和概率
- 表达了条件概率分布 $\mathbb{P}(X_{t+1} \mid X_t)$，状态有限时可以用矩阵表示

图片来源：https://commons.wikimedia.org/wiki/File:Markov_Chain_for_String_Generation_Example.png

Guessing the Dice Roll

HDU 5955

- 有 n 个人，每人给出长度为 m 的点数序列，重复掷骰子直到掷出的点数序列的后缀与某个人的序列匹配，结束并认为那个人获胜
- 计算每个人获胜的概率

- 第一步：构造马尔可夫链 $X_1, X_2, \cdots$ 表示重复掷骰子过程中的匹配状态
  - 设 $a_1, a_2, \cdots, a_n$ 表示 n 个人获胜的状态，这些状态满足 $p(a_i; a_i) = 1$
- 第二步：计算 $\lim_{t\to\infty} \mathbb{P}(X = a_i)$，对第 i 个人计算匹配到其获胜的状态 a_i 的概率

Guessing the Dice Roll

HDU 5955

- 设转移矩阵为 P，初始状态的概率分布为行向量 ν，那么需要计算 $\lim_{t\to\infty} \nu P^t$
  - 与无限猴子定理的情况类似，可以证明极限存在
- 算法一：矩阵快速幂，直到收敛
- 算法二：观察到如果 $\pi = \lim_{t\to\infty} \nu P^t$，那么有 $\pi = \pi P$
  - 建立线性方程组，通过高斯消元求解 π
  - 比如状态 x 在掷出 1 到 6 点后的后继状态是 x_1' 到 x_6'，那么有方程
    - $\pi(x) = \frac{1}{6}\pi(x_1') + \cdots + \frac{1}{6}\pi(x_6')$

概率论101
贝叶斯公式

- 对于随机变量 X 和 Y，条件概率分布 $\mathbb{P}(X \mid Y)$ 和 $\mathbb{P}(Y \mid X)$ 有何关系？
- 事件 A 和 B 的版本：

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)} = \frac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(A)}$$

贝叶斯公式
（Bayes' rule）

- 离散随机变量 X 和 Y 的版本：

$$\mathbb{P}(Y = y \mid X = x) = \frac{\mathbb{P}(Y = y)\mathbb{P}(X = x \mid Y = y)}{\mathbb{P}(X = x)}$$

- 连续随机变量 X 和 Y 的版本：

$$p_{Y|X}(y \mid x) = \frac{p_Y(y)p_{X|Y}(x \mid y)}{p_X(x)}$$

病毒检测

- 回顾：B 为「感染了病毒」，A 为「检测呈阳性」，$\mathbb{P}(B \mid A)$ 为真阳性的概率
- 如果群体患病率为 $1\,\%$，检测准确率为 $95\,\%$，真阳性的概率是多少？

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(A)}$$

贝叶斯公式：$\mathbb{P}(B \mid A) = \dfrac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(A)}$

$$= \frac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(B)\mathbb{P}(A \mid B) + \mathbb{P}(\overline{B})\mathbb{P}(A \mid \overline{B})}$$

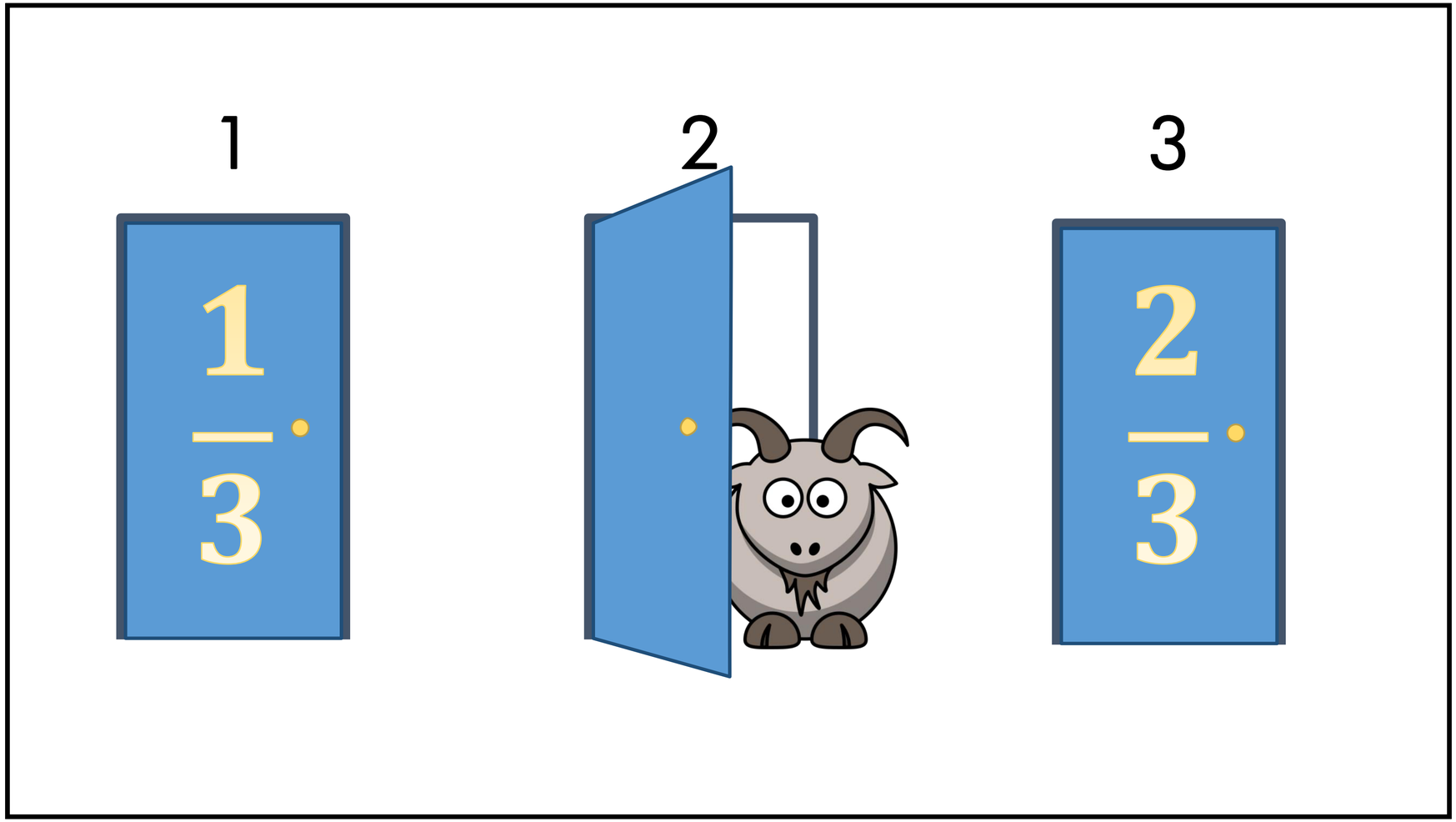
$\overline{B}$ 表示「没有感染病毒」

$$= \frac{1/100 \times 95/100}{(1/100 \times 95/100) + (99/100 \times 5/100)}$$

$$\approx 16.1\,\%$$

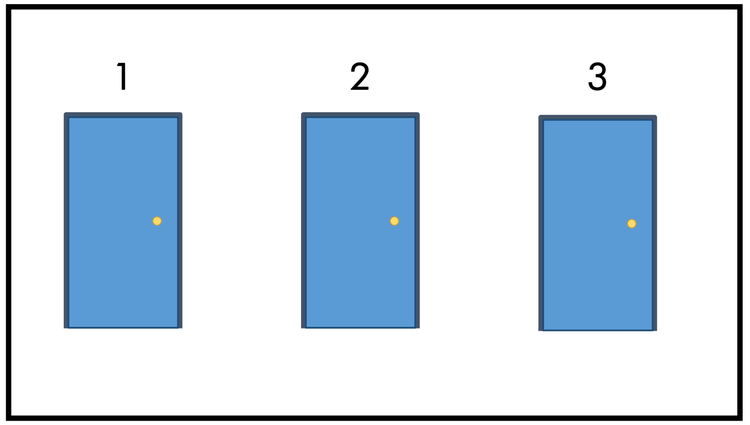
即便检测阳性，真的感染的概率其实也不高

三门问题



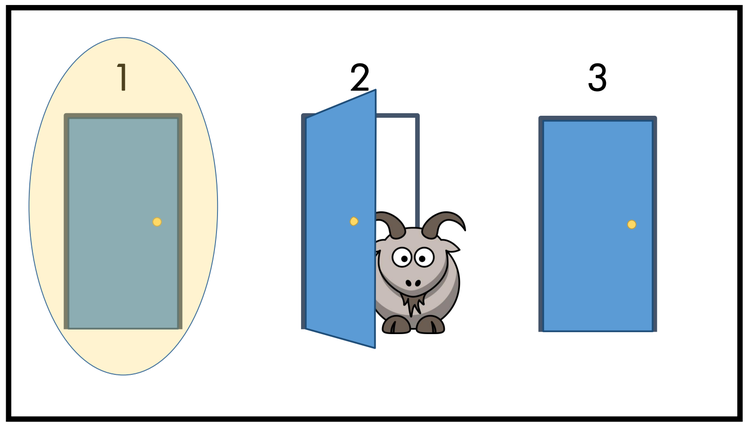
图片来源：Maria Gorinova's Advances in Programming Languages (Guest Lecture) slides on Probabilistic Programming.

三门问题



$$\mathbb{P}(\text{车在1号门}) = \frac{1}{3}$$

贝叶斯公式：$\mathbb{P}(B \mid A) = \dfrac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(A)}$



$$\mathbb{P}(\text{车在1号门} \mid \text{主持人开2号门})$$

$$= \frac{\mathbb{P}(\text{车在1号门})\mathbb{P}(\text{主持人开2号门} \mid \text{车在1号门})}{\mathbb{P}(\text{主持人开2号门})}$$

$$= \frac{\mathbb{P}(\text{车}1)\mathbb{P}(\text{主}2 \mid \text{车}1)}{\mathbb{P}(\text{车}1)\mathbb{P}(\text{主}2 \mid \text{车}1) + \mathbb{P}(\text{车}2)\mathbb{P}(\text{主}2 \mid \text{车}2) + \mathbb{P}(\text{车}3)\mathbb{P}(\text{主}2 \mid \text{车}3)}$$

$$= \frac{1/3 \times 1/2}{(1/3 \times 1/2) + (1/3 \times 0) + (1/3 \times 1)} = \frac{1}{3}$$

萨利·克拉克的审判

- 1996 年，萨利·克拉克的新生儿在出生两周后去世
- 一年之后历史重演，她的第二个新生儿也去世了
- 儿科医生罗伊·梅多斯出庭作证「两名新生儿接连由于自然原因死亡的概率是 **7300 万分之一**」，这段证词给萨利·克拉克定了罪

- 这个审判是否存在问题？

萨利·克拉克的审判

- 从经典逻辑的角度，似乎没有问题
  - 如果克拉克无罪，那么她的两个孩子不会都刚出生就死亡
  - **逆否命题**：如果她的两个孩子都刚出生就死亡，那么克拉克有罪
- 考虑概率分析，令 B 为「克拉克无罪」，A 为「两个孩子刚出生就死亡」

医生的证词只说明 $\mathbb{P}(A \mid B)$ 概率很低

贝叶斯公式：$\mathbb{P}(B \mid A) = \dfrac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(A)}$

但是「两个孩子刚出生就死亡」本来就及其罕见！

萨利·克拉克的审判

- 令 B 为「克拉克无罪」，$\overline{B}$ 为「克拉克有罪」，A 为「两个孩子刚出生就死亡」
- 考虑 $\mathbb{P}(\overline{B})$，它应该是个很小的概率，根据历史统计估算为 **5 亿分之一**

$$
\begin{aligned}
&\mathbb{P}(B \mid A) \\
&= \frac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(A)} = \frac{\mathbb{P}(B)\mathbb{P}(A \mid B)}{\mathbb{P}(B)\mathbb{P}(A \mid B) + \mathbb{P}(\overline{B})\mathbb{P}(A \mid \overline{B})} \\
&= \frac{(1 - 1/500000000) \times 1/73000000}{((1 - 1/500000000) \times 1/73000000) + (1/500000000 \times 1)} \\
&\approx 87.3\,\%
\end{aligned}
$$

医生的证词说明 $\mathbb{P}(A \mid B)$ 为 7300 万分之一

在有罪的条件下，估算 $\mathbb{P}(A \mid \overline{B}) = 1$

克拉克无罪的概率很大！

萨利·克拉克的审判

- 概率视角下，如何看待**逆否命题**？
  - $\mathbb{P}$(孩子没死 | 无罪) 接近 1，但是 $\mathbb{P}$(有罪 | 孩子死了) 接近 0!

- 相比于经典逻辑，似乎「贝叶斯式的逻辑」更适合我们的现实生活
  - 为命题赋予**置信度**，而不是单纯的真与假的二分法
  - 使用**贝叶斯公式**来更新我们对命题的置信度

贝叶斯推断
Bayesian Inference

对 H 的**先验**概率（**prior**）

若 H 发生则 E 发生的**似然度**（**likelihood**）

观察到 E 后，对 H 的**后验**概率（**posterior**）

$$\mathbb{P}(H \mid E) = \frac{\mathbb{P}(H)\mathbb{P}(E \mid H)}{\mathbb{P}(E)}$$

- 设 H 是我们关注但无法直接获取信息的事件，比如「感染了病毒」
  - 可理解为**假设**（Hypothesis）
- 设 E 是与 H 相关而且可以通过观察获取信息的事件，比如「检测呈阳性」
  - 可理解为**证据**（Evidence）

贝叶斯推断
Bayesian Inference

对 H 的**先验**概率（**prior**）

若 H 发生则 E 发生的**似然度**（**likelihood**）

观察到 E 后，对 H 的**后验**概率（**posterior**）

$$\mathbb{P}(H \mid E) = \frac{\mathbb{P}(H)\mathbb{P}(E \mid H)}{\mathbb{P}(E)}$$

- 通过观察证据 E 获取信息，从而调整我们对假设 H 是否成立的认知
- 按照随机变量来理解，若 X 表示无法直接观察的量，Z 表示可以观察的相关量：

观察 Z 的值为 z 后，对 X 的后验概率分布

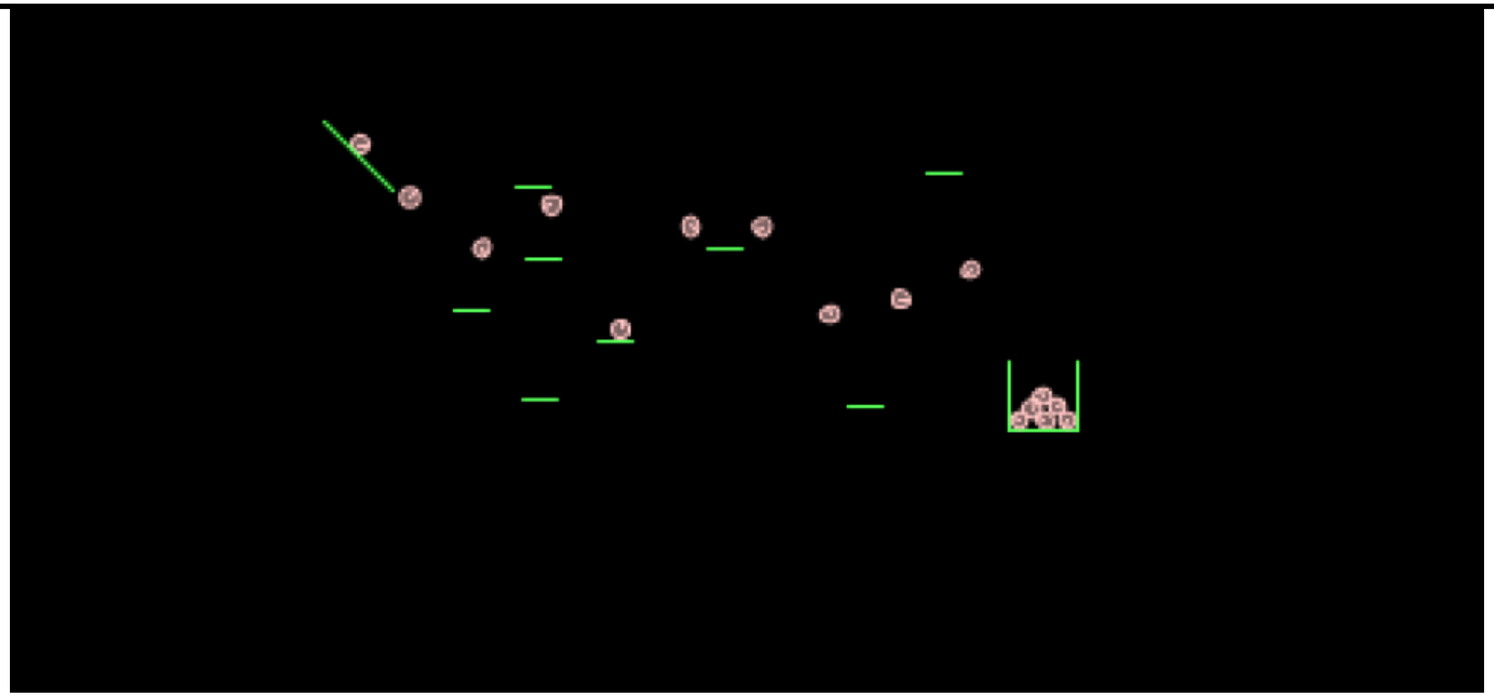
$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

机制设计

是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- X：放置跳板的方案，Z：进入目标桶的小球数
- 先验 $\mathbb{P}(X)$：随机在平面上放置
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定方案，有 z 个小球进入目标桶的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到 z 个小球进入目标桶，方案的条件概率分布

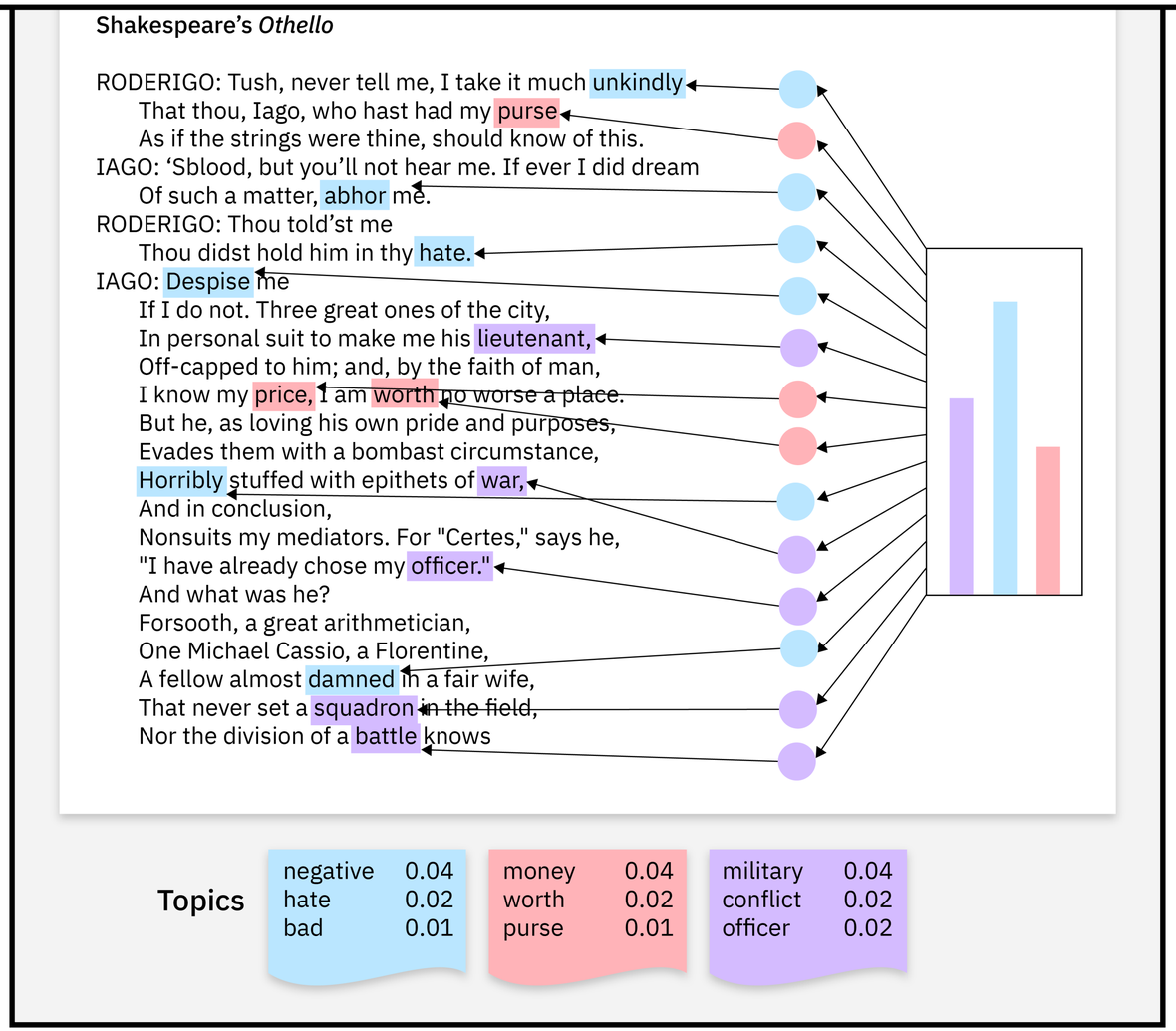


文本分类
是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- 只从单词出现的角度来考虑文本分类，即「词袋」（bag of words）
- 设单词表长度为 K

- X：每个主题中的单词概率，Z：文本
- 先验 $\mathbb{P}(X)$：随机的长度为 K 的非负向量，其元素和为一
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定每个主题的单词概率，生成文本 z 的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到文本 z，每个主题的单词概率的条件概率分布

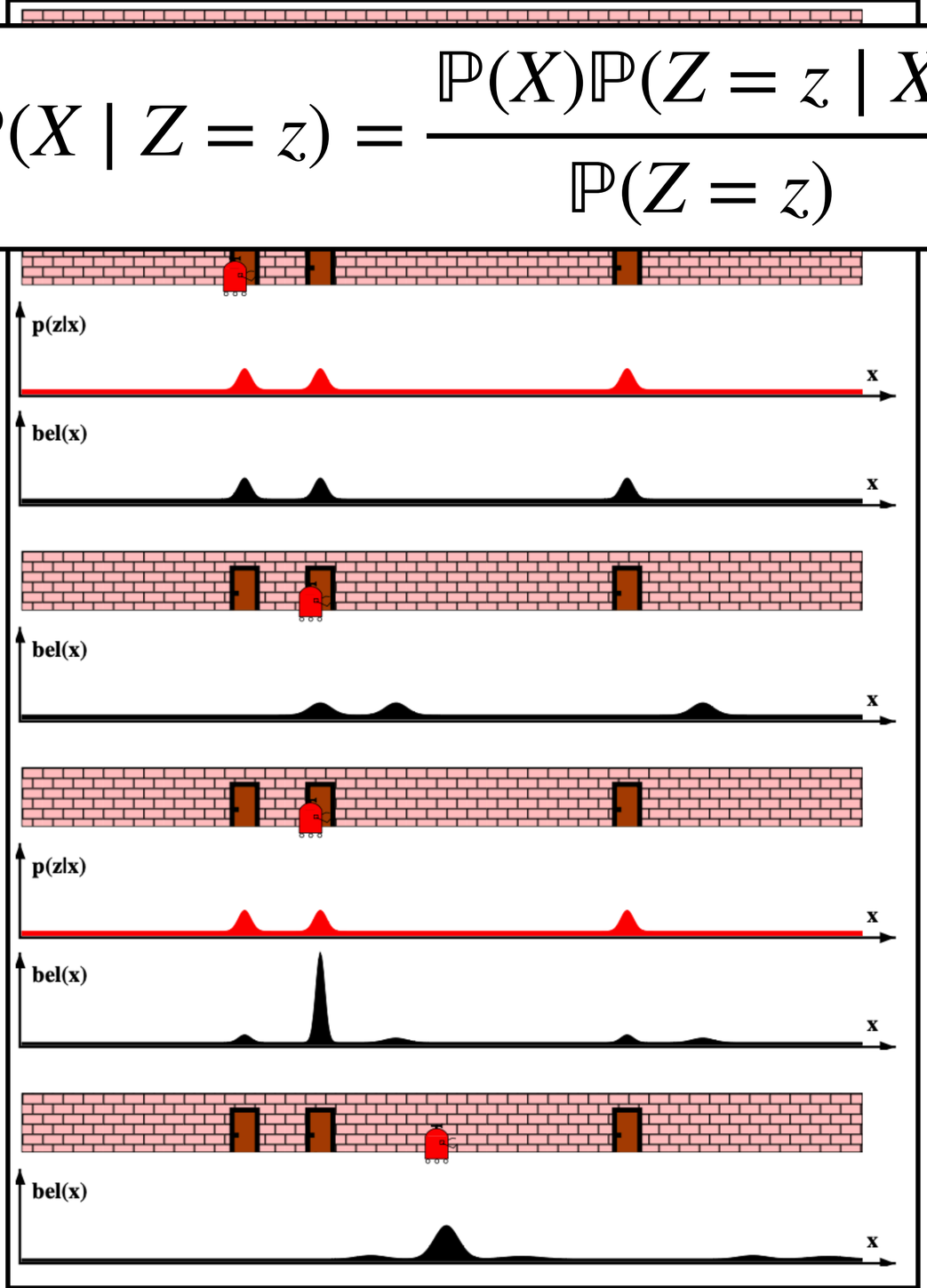


图片来源：https://www.ibm.com/think/topics/latent-dirichlet-allocation

概率定位
是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- X：机器人的位置，Z：传感器返回的信息
- 先验 $\mathbb{P}(X)$：在考虑的空间里的随机分布
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定机器人的位置，传感器返回信息 z 的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到传感器信息 z，机器人位置的条件概率分布

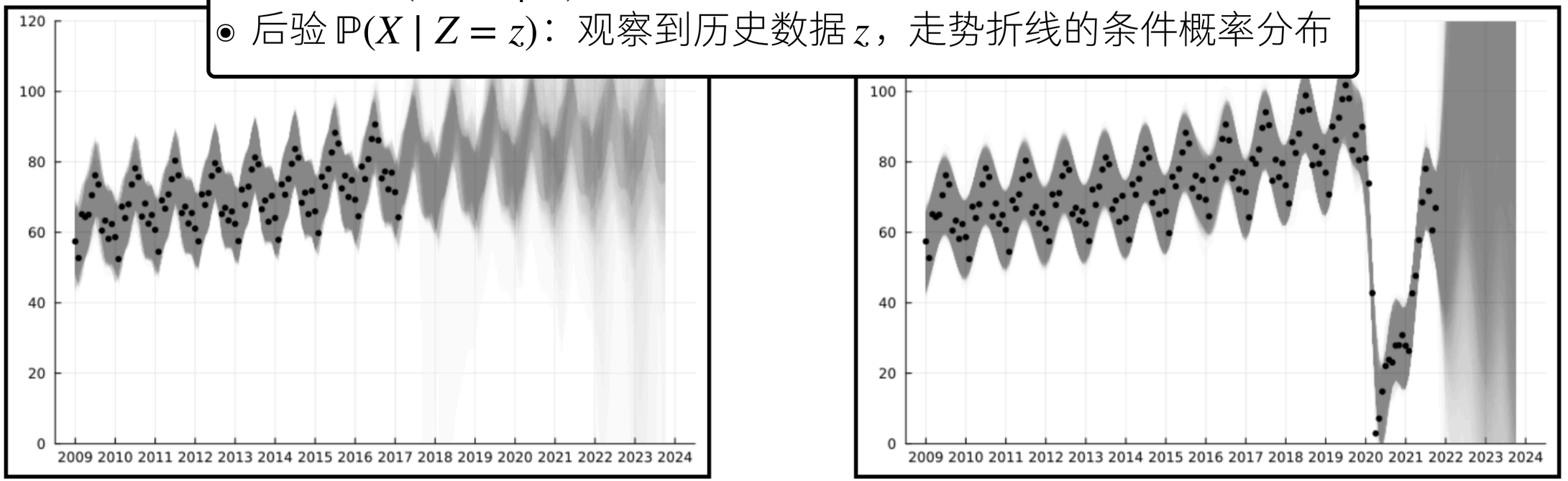


图片来源：S. Thrun, W. Burgard, and D. Fox. 2005. Probabilistic Robotics. *MIT Press.*

时间序列
是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- X：走势折线，Z：历史数据
- 先验 $\mathbb{P}(X)$：通过一些方式（比如高斯过程）给出走势折线的分布
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定走势折线，拟合历史数据 z 的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到历史数据 z，走势折线的条件概率分布



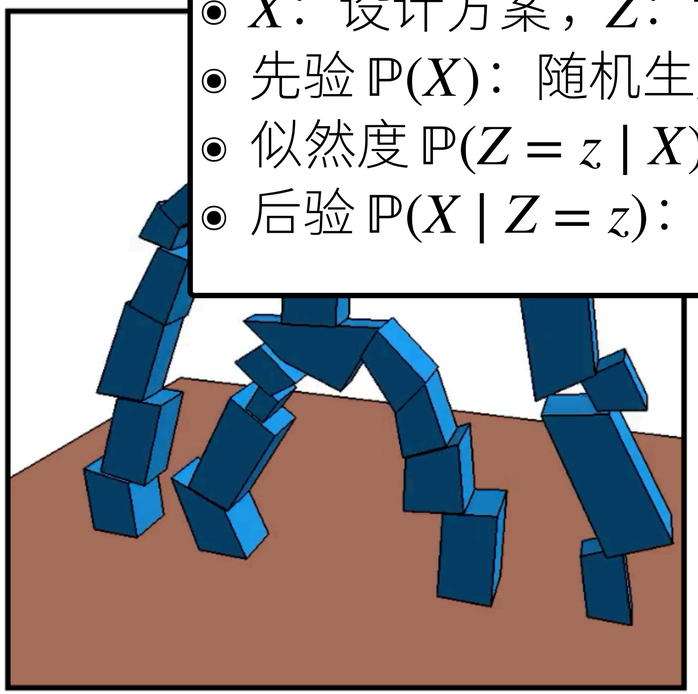
美国国内民航总里程（按月份）

数据来源：https://fred.stlouisfed.org/series/RPM

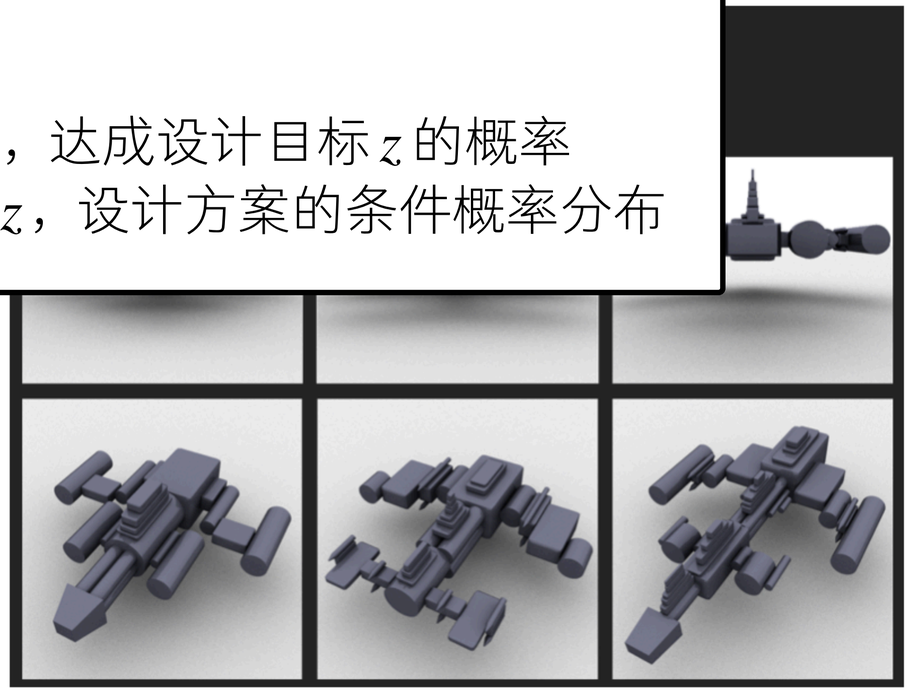
过程化设计
是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- X：设计方案，Z：设计目标
- 先验 $\mathbb{P}(X)$：随机生成设计方案
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定设计方案，达成设计目标 z 的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到设计目标 z，设计方案的条件概率分布



从一个初始结构出发
设计具有稳定结构的积木结构



给定一个前视图
设计一个具有该前视图的飞船

图片来源：D. Ritchie. 2016. Probabilistic Programming for Procedural Modeling and Design. *Ph.D. Thesis.*

验证码破解
是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- X：字符串，Z：验证码图片
- 先验 $\mathbb{P}(X)$：随机产生的字符串
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定字符串，渲染后接近验证码图片 z 的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到验证码图片 z，字符串的条件概率分布

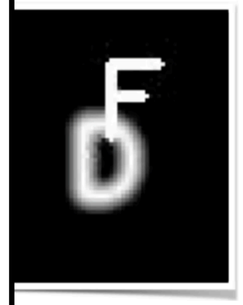
Facebook Captcha

Observed images

(W4kgvQ)

Inference

	10^7	W4kgvQ		
Training traces	10^6	WA9opvQ		
	10^5	WpzyTcQ	mTYnIwd	NmnMmC3
	10^4	ZVRYTZ8	wB3Fc6	EqK4hhc



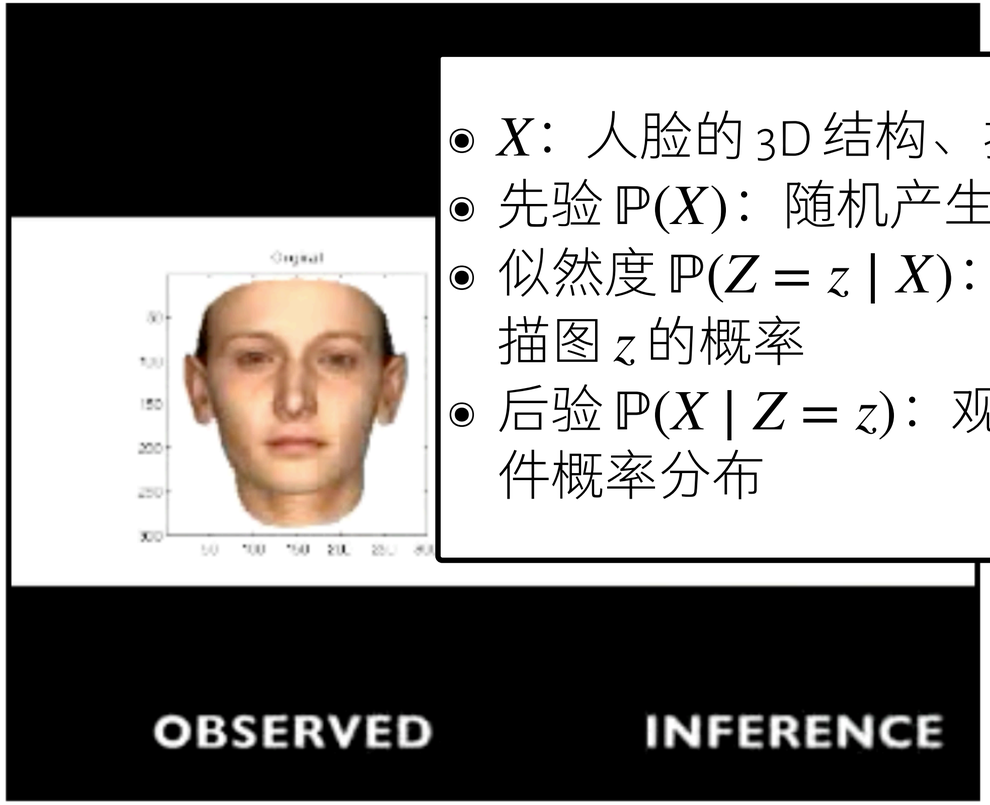
→ SMKBDF

渲染过程从字符串随机生成一张验证码图片

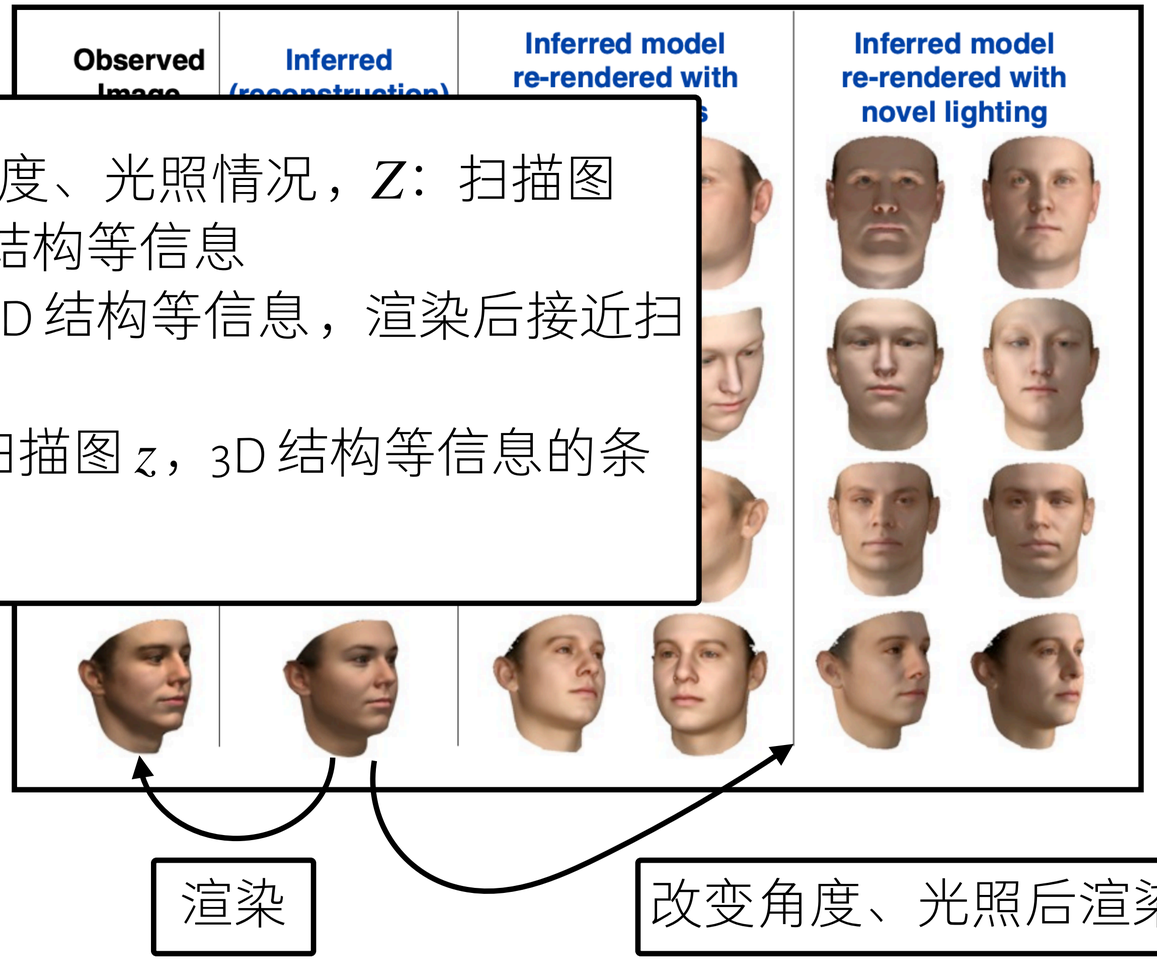
图片来源：T. A. Le, A. G. Baydin, and F. Wood. 2017. Inference Compilation and Universal Probabilistic Programming. In *AISTATS'17*.

逆向图形学
是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$



- X：人脸的 3D 结构、扫描角度、光照情况，Z：扫描图
- 先验 $\mathbb{P}(X)$：随机产生的 3D 结构等信息
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定 3D 结构等信息，渲染后接近扫描图 z 的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到扫描图 z，3D 结构等信息的条件概率分布

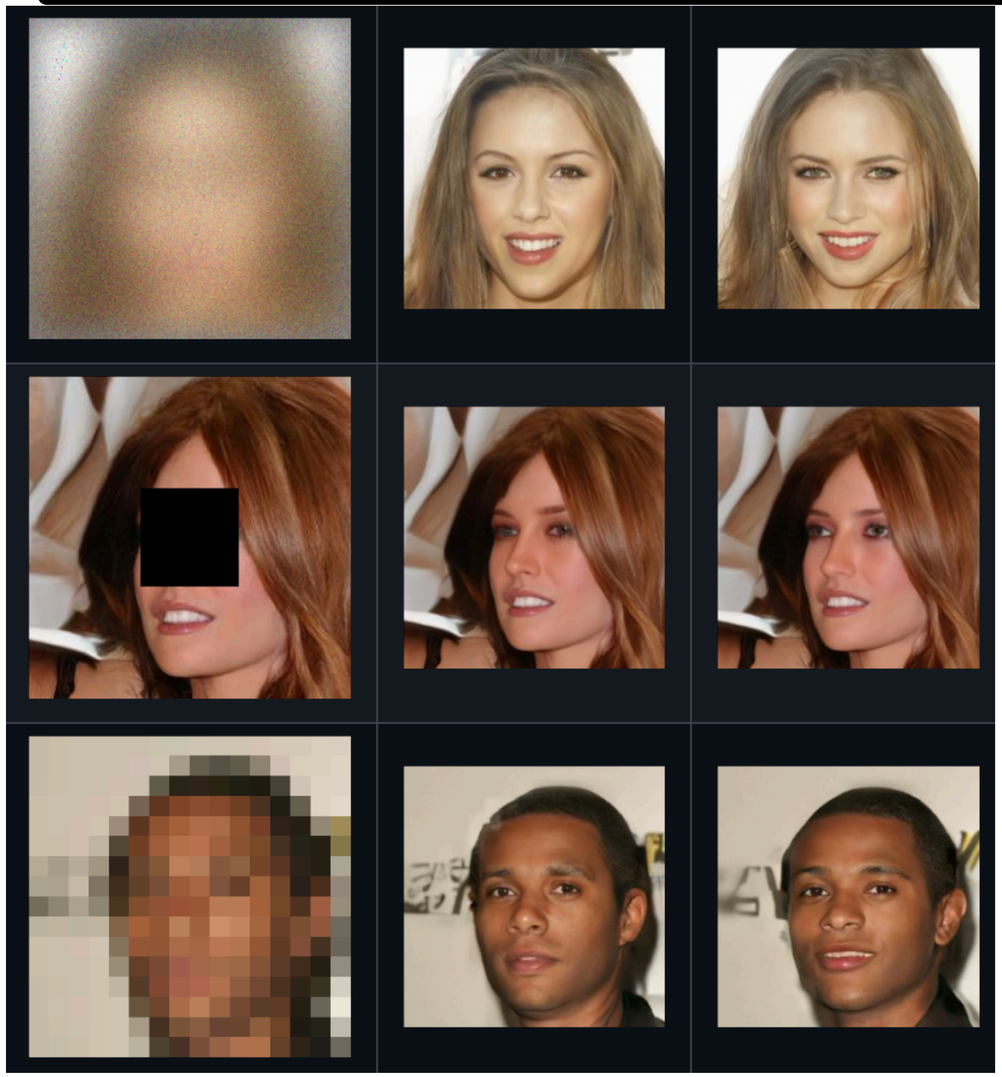


图片来源：T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. Mansinghka. 2015. Picture: A Probabilistic Programming Language for Scene Perception. In *CVPR'15*.

图片复原
是贝叶斯推断

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- X：复原图，Z：输入图
- 先验 $\mathbb{P}(X)$：随机产生的复原图（可以使用 Stable Diffusion 模型）
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定复原图，破坏后接近输入图 z 的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到输入图 z，复原图的条件概率分布



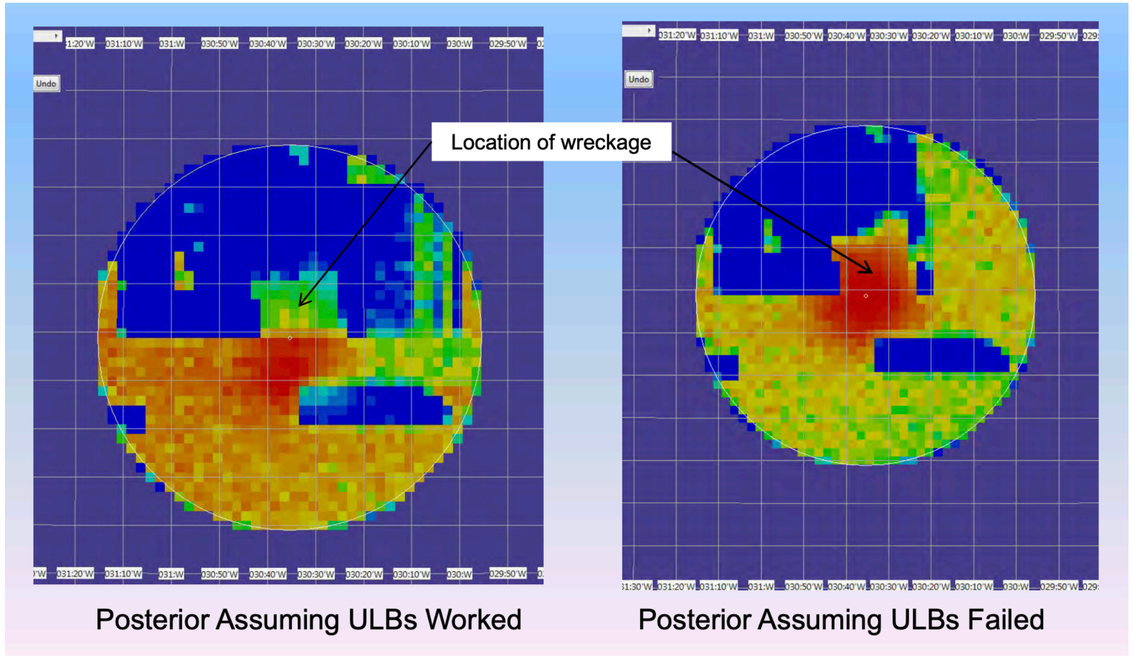
图片来源：G. Cardoso, Y. J. E. Idrissi, S. L. Corff, and E. Moulines. 2024. Monte Carlo guided Diffusion for Bayesian linear inverse problems. In *ICLR'24*.

贝叶斯推断
Bayesian Inference

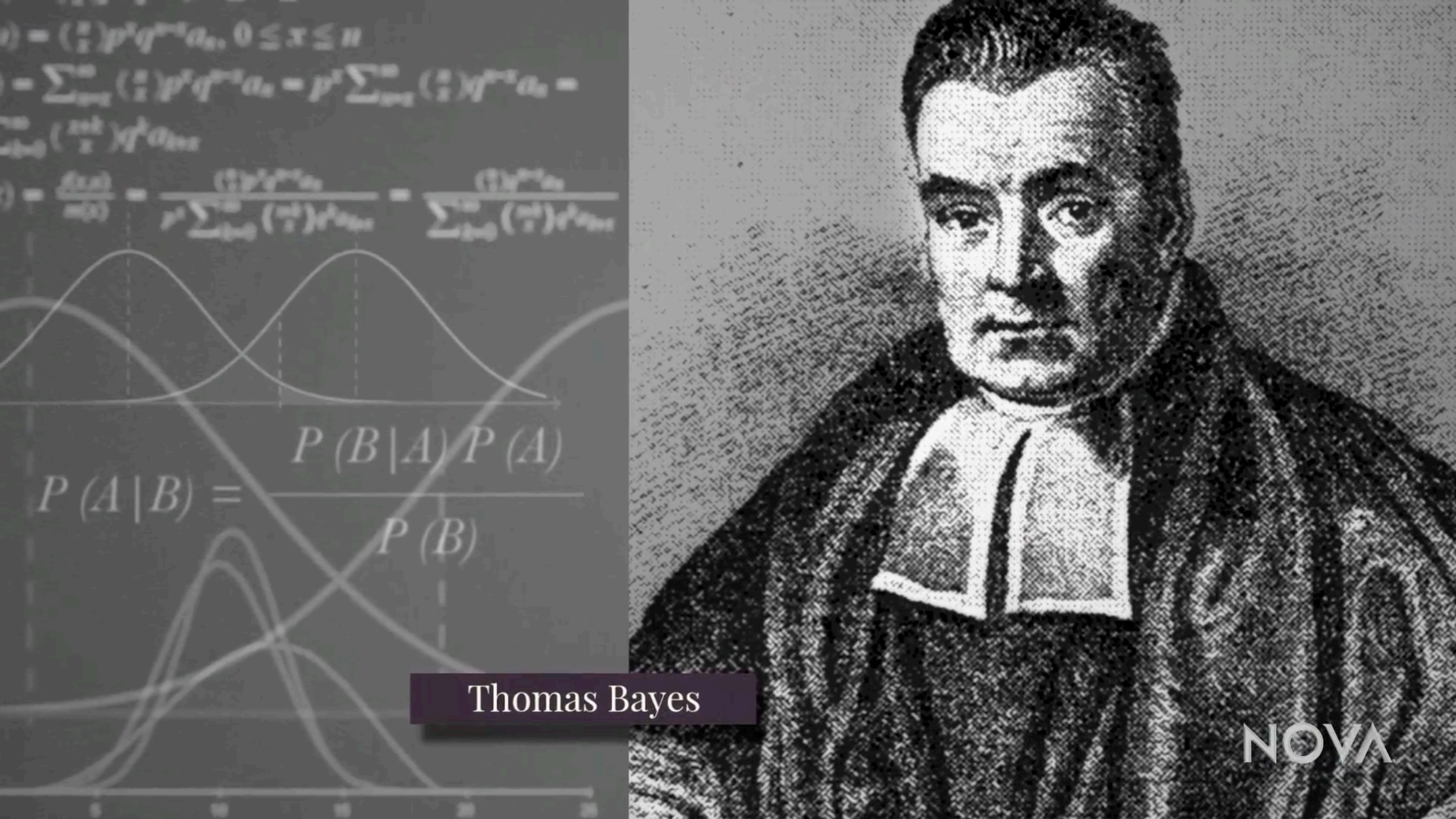
观察 Z 的值为 z 后，
对 X 的后验概率分布

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- 在一些场景中，通过贝叶斯推断建议下一步去观察什么，进而更新对 X 的认知
- 例如：贝叶斯搜索（物理层面的）
  - 1968 年：US Submarine Scorpion
  - 1988 年：SS Central America
  - 2008 年：Fosset
  - 2011 年：AF 447（法航 447 空难）



图片来源：L. D. Stone. 2017. Bayesian Search for Missing Aircraft. *Online.*


$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$
Thomas Bayes
NOVA

贝叶斯推断
Bayesian Inference

观察 Z 的值为 z 后，
对 X 的后验概率分布

$$\mathbb{P}(X \mid Z = z) = \mathbb{P}(X) \times \frac{\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- 我们应该去**观察**什么样的 Z（换句话说，收集什么样的**证据**）？
- 直觉：我们的观察应该尽可能大地改变先验概率 $\mathbb{P}(X)$
  - 当先验概率很低的时候，$\mathbb{P}(Z = z \mid X)/\mathbb{P}(Z = z)$ 很大的观察
  - 当先验概率很高的时候，$\mathbb{P}(Z = z \mid X)/\mathbb{P}(Z = z)$ 很低的观察
- 换句话说，$\mathbb{P}(Z = z \mid X)/\mathbb{P}(Z = z)$ 接近于 1 的则是没啥**信息量**的观察

萨根标准：超凡主张
须有超凡证据

贝叶斯视角下的星座特质

- 考虑 x 为我的星座，z 是星座学告诉我的性格特质，然后我发现很符合
  - 「他们也很实际，能使爱幻想与图实际的性格共存且并荣。 做事周到细心、谨慎有条理，非常理性，甚至冷酷，有特殊的评论力。 喜欢一点点的分析、批判，做事很投入。 好学、好奇、求知欲旺盛……」
- 也就是说，$\mathbb{P}(Z = z \mid X = x) \approx 1$
- 但是，观察本身的概率 $\mathbb{P}(Z = z) \approx 1$，也就是描述对于大多数人多少都符合
- 那么，比值 $\mathbb{P}(Z = z \mid X)/\mathbb{P}(Z = z) \approx 1$，是**信息量**很少的观察！

贝叶斯视角下的医生诊断

- 医生诊断需要选择「性价比」高的观察
  - 「性能」：观察带来的**信息量**
  - 「价格」：进行观察的时间消耗、检查成本、对病人的伤害等
- 例子：考虑两种疾病 H_1 和 H_2，已知 H_1 下症状 E_1 的概率 $\mathbb{P}(E_1 \mid H_1)$
- 考虑下列两条信息，应该选择哪一条？
  - 在另外一种疾病 H_2 下症状 E_1 的概率 $\mathbb{P}(E_1 \mid H_2)$
  - 找疾病 H_1 下的另一个症状 E_2 及其概率 $\mathbb{P}(E_2 \mid H_1)$

贝叶斯视角下的医生诊断

- 回顾可以衡量信息量的比值：$\mathbb{P}(E \mid H)/\mathbb{P}(E)$，其中 H 为疾病，E 为症状

$$\frac{\mathbb{P}(E_1 \mid H_1)}{\mathbb{P}(E_1)} = \frac{\mathbb{P}(E_1 \mid H_1)}{\mathbb{P}(H_1)\mathbb{P}(E_1 \mid H_1) + \mathbb{P}(H_2)\mathbb{P}(E_1 \mid H_2)}$$

已知

- 获取 $\mathbb{P}(E_1 \mid H_2)$ 可以帮助我们评估 E_1 作为证据的「排他性」
- 如果 $\mathbb{P}(E_1 \mid H_2) \approx 0$，那么后验概率 $\mathbb{P}(H_1 \mid E_1) = \mathbb{P}(H_1) \times \frac{\mathbb{P}(E_1 \mid H_1)}{\mathbb{P}(E_1)} \approx 1$
- **优先找其它能解释证据的假设，而不是只找当前假设能对应上的证据！**

到达「平衡态」

这确实是个不得了的能力

$$\mathbb{P}(X \mid Z = z) = \frac{\mathbb{P}(X)\mathbb{P}(Z = z \mid X)}{\mathbb{P}(Z = z)}$$

- 在贝叶斯推理中，往往 $\mathbb{P}(Z = z)$ 是很难求（或者没法求）的
- 即使不知道 $\mathbb{P}(X \mid Z = z)$ 的具体形式，通过迭代算法可以达到对其采样的目的
  - **The Metropolis-Hastings Algorithm**，即将介绍
- 核心：给定 z 后，有 $\mathbb{P}(X \mid Z = z) \propto \mathbb{P}(X)\mathbb{P}(Z = z \mid X)$，右侧式子往往可以计算
  - 对于不同的 x_1, x_2，比值 $\frac{\mathbb{P}(X = x_1 \mid Z = z)}{\mathbb{P}(X = x_2 \mid Z = z)} = \frac{\mathbb{P}(X = x_1)\mathbb{P}(Z = z \mid X = x_1)}{\mathbb{P}(X = x_2)\mathbb{P}(Z = z \mid X = x_2)}$ 可以计算

The Metropolis-Hastings (MH) Algorithm
一种基于采样的贝叶斯推断算法

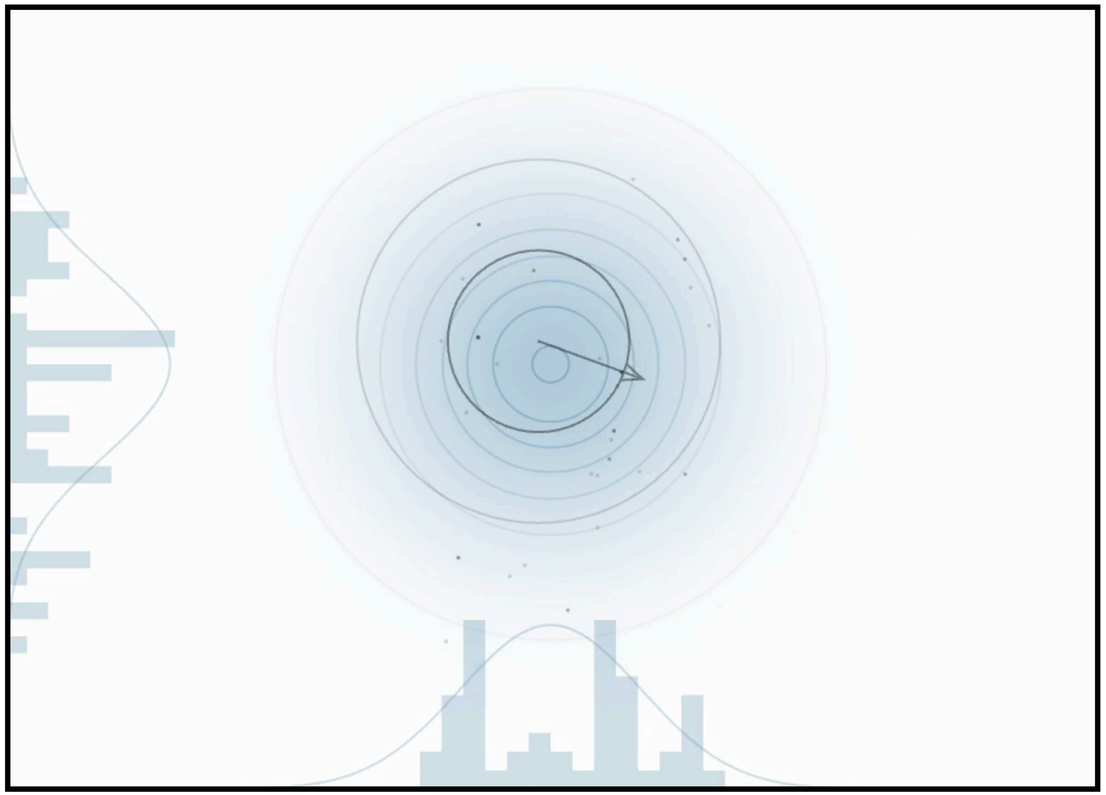
- 我们需要对未知分布 $p(x)$ 进行采样，其中 $x \in \Omega$ 表示样本
- 虽然分布是未知的，但是支持计算 $f(x)$ 并满足 $p(x) \propto f(x)$
  - 这样对于任意样本 x_1 和 x_2，可以计算比值 $p(x_1)/p(x_2) = f(x_1)/f(x_2)$
- $x_0 \leftarrow$ 初始样本
- 迭代，其中第 i 次迭代从 x_{i-1} 通过**提议分布** $q(x'; x_{i-1})$ 采样 x'
  - 计算**接受概率** $\alpha(x'; x_{i-1}) = \min(1, \frac{p(x') \cdot q(x_{i-1}; x')}{p(x_{i-1}) \cdot q(x'; x_{i-1})})$
  - 以概率 $\alpha(x'; x_{i-1})$ 令 $x_i \leftarrow x'$，否则令 $x_i \leftarrow x_{i-1}$

可理解为邻居选择的推广

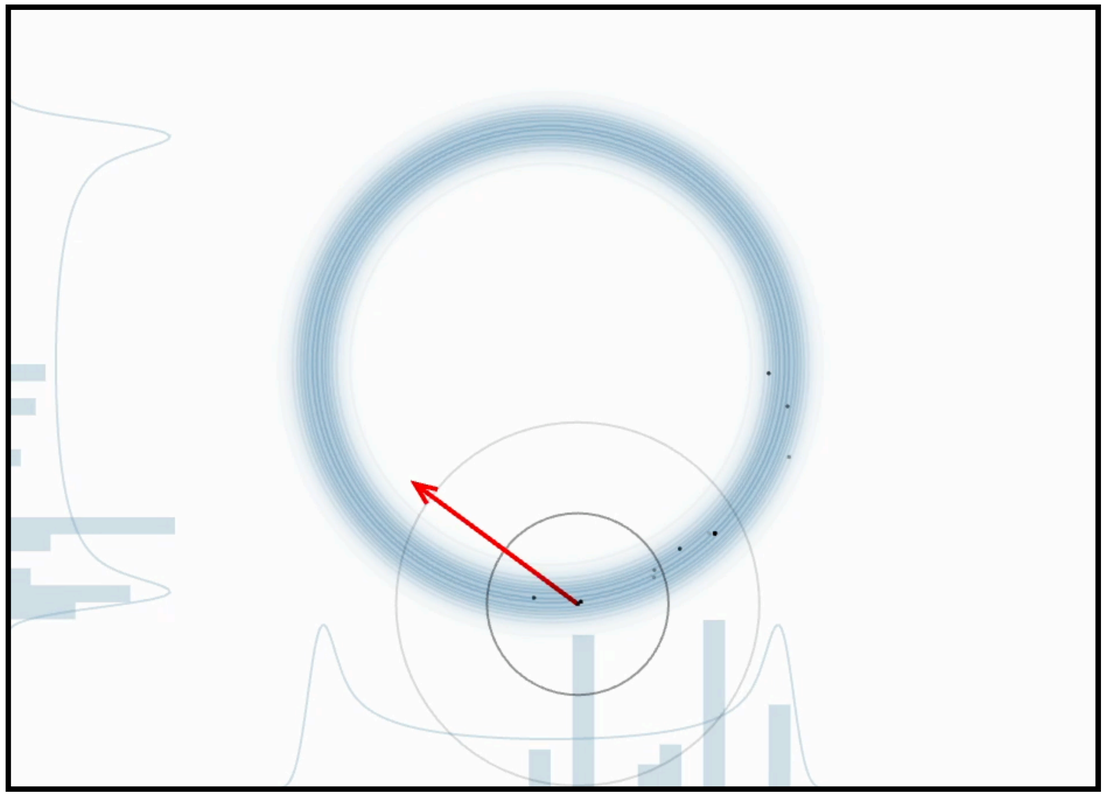
当 q 具备对称性且 p 是玻尔兹曼分布，退化为模拟退火中的形式

MH 算法演示

MH 算法 + 随机游走提议分布



二维标准正态分布
采样效果不错



二维「甜甜圈」分布
采样效果较差

动图来源：https://chi-feng.github.io/mcmc-demo/

MH 算法本质是在构造马尔可夫链

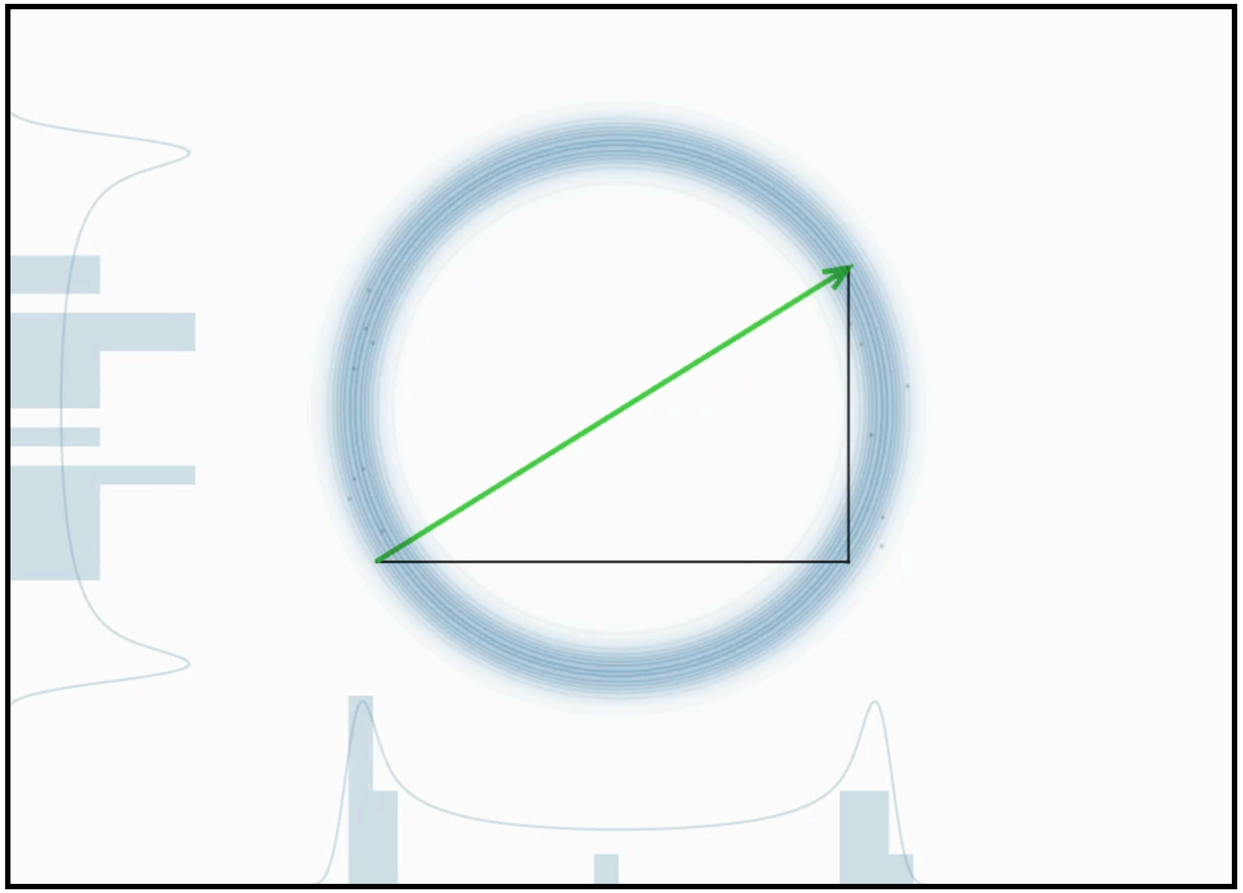
Markov Chain Monte Carlo (MCMC)

- 构造马尔可夫链 $X_1, X_2, \cdots$，随机变量 X_i 表示第 i 次迭代时的样本
- 在有限（或离散）样本空间下推导马尔可夫链的转移概率
  - $\mathbb{P}(X_{n+1} = k \mid X_n = x_n) = q(k; x_n) \cdot \alpha(k; x_n)$，若 $k \neq x_n$
  - $\mathbb{P}(X_{n+1} = k \mid X_n = k) = q(k; k) + \sum_{j \neq k} q(j; k) \cdot (1 - \alpha(j; k))$
  - 可证明，若把目标分布 p 看作向量 π，$\mathbb{P}(X_{n+1} \mid X_n)$ 看作矩阵 P，有 $\pi P = P$
- 通过模拟马尔可夫链转移完成采样，**充分长**时间后逼近目标分布

其它 MCMC 算法

Gibbs Sampling

- 考虑二维的样本空间 $\mathbb{R} \times \mathbb{R}$ 和目标概率分布 $p(x, y) \propto f(x, y)$
- 假设条件概率分布 $p(x \mid y)$ 和 $p(y \mid x)$ 都是容易采样的
- 那么从 (x_i, y_i) 出发，可以先采样 $p(x \mid y_i)$ 得 x_{i+1}，再采样 $p(y \mid x_{i+1})$ 得 y_{i+1}，从而得到 (x_{i+1}, y_{i+1})
- 相当于每次迭代都能接受新样本

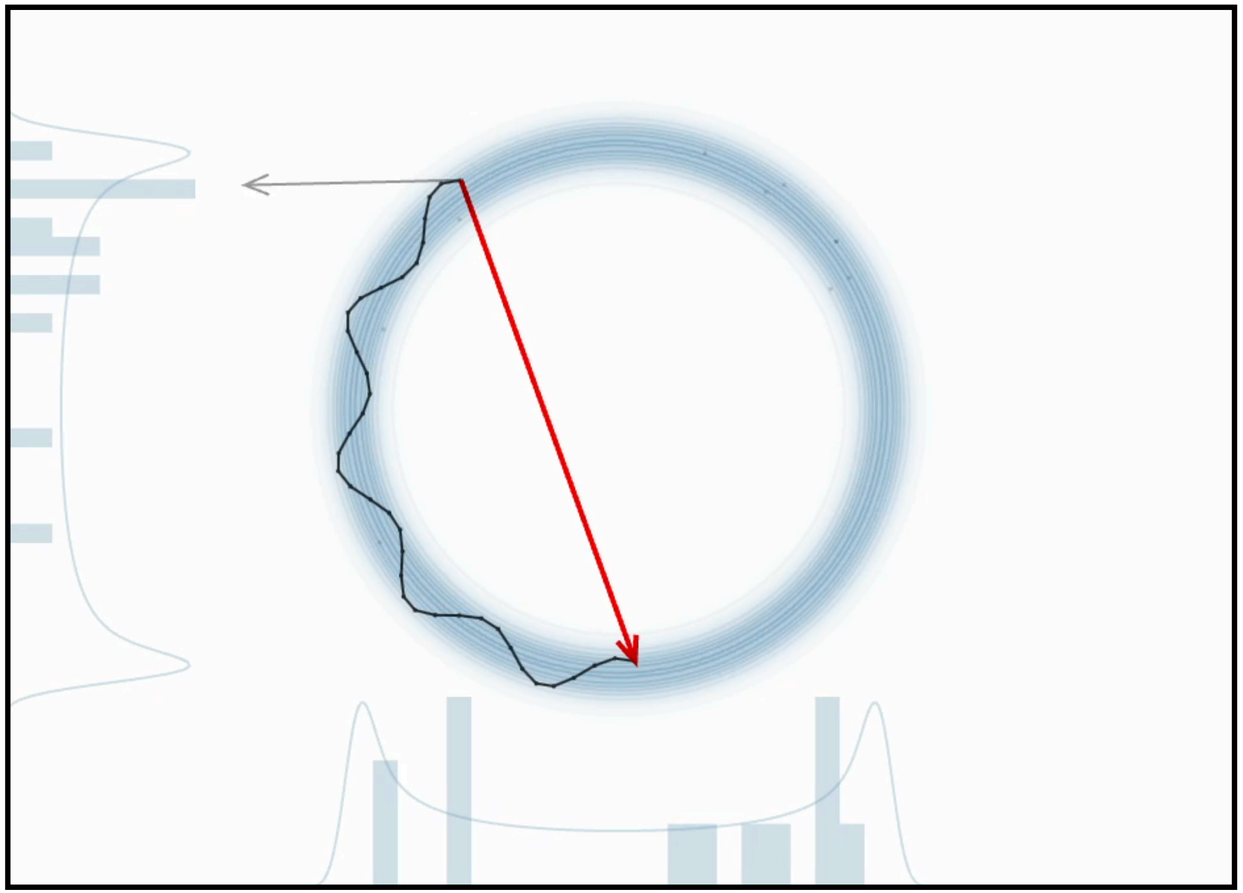


动图来源：https://chi-feng.github.io/mcmc-demo/

其它 MCMC 算法

Hamiltonian Monte Carlo

- 考虑二维的样本空间 $\mathbb{R} \times \mathbb{R}$ 和目标概率分布 $p(x, y) \propto f(x, y)$
- 可以想像 $(x, y, -\log p(x, y))$ 构成了三维空间中的一个面，每个样本对应面上的一个点
- 再想像每次给样本一个随机的动量，然后模拟其在面上运动一段时间，则可以比随机游走探索得更远，并且倾向于概率高的位置



动图来源：https://chi-feng.github.io/mcmc-demo/

还有很多 MCMC 算法！

- 为什么会有多种算法？
  - 不存在一个算法能在所有的贝叶斯推断问题上都达到好的采样效果
- 对于一个贝叶斯推断问题，实现不同的 MCMC 算法感觉很重复工作呢？
  - 确实！
- 怎么办？
  - 将贝叶斯推断问题分成**建模**和**求解**两部分
  - 概率编程！

概率编程
Probabilistic Programming

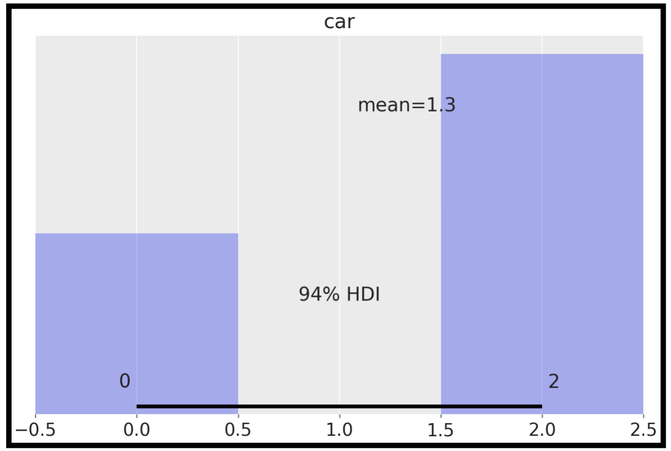
- **建模**：将概率模型 $\mathbb{P}(X, Z)$ 实现为**概率程序**，表达先验 $\mathbb{P}(X)$ 和似然度 $\mathbb{P}(Z \mid X)$
  - 概率程序 = 普通程序 + **采样**（sample） + **观察**（observe）
- **求解**：概率编程系统实现多种推断算法，自动对**概率程序**进行贝叶斯推断
  - 各种 MCMC 算法
  - 还有一些不基于采样的算法，比如变分推断（variational inference）

- 我们使用 PyMC 概率编程库进行接下来的讲解

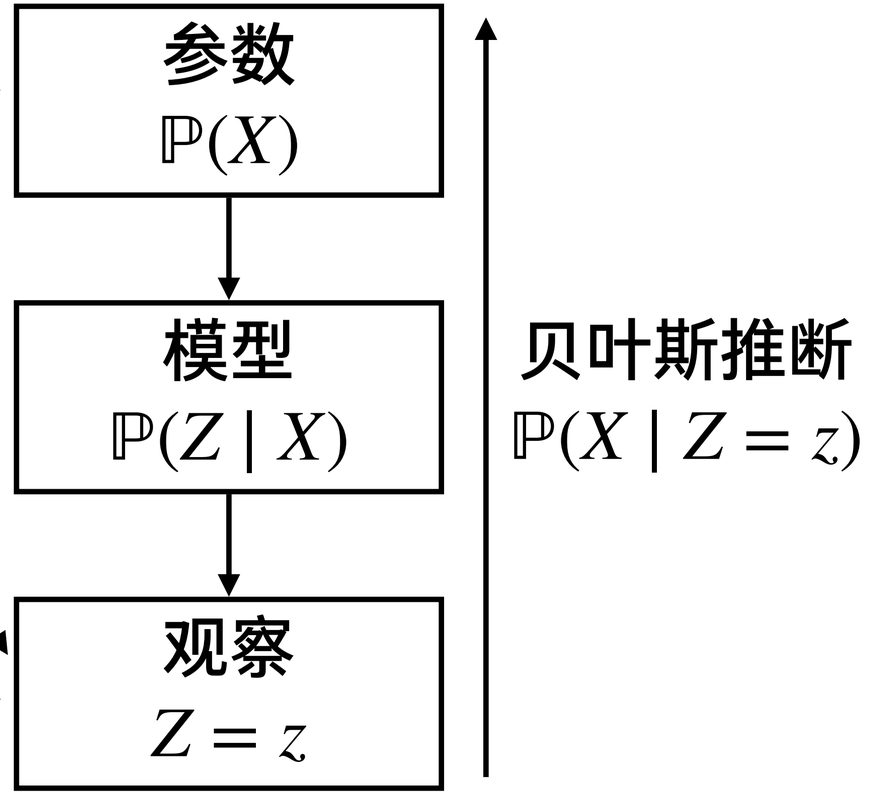
三门问题

使用概率编程求解

```
with pm.Model() as model:
  car = pm.Categorical("car", p=[1/3,1/3,1/3])

  pick = pm.Categorical("pick", p=[1/3,1/3,1/3], observed=0)
  p_host = pm.Data("p_host", [[0,1/2,1/2], [0,0,1], [0,1,0]])
  host = pm.Categorical("host", p=p_host[car], observed=1)

  idata = pm.sample(10000)
```



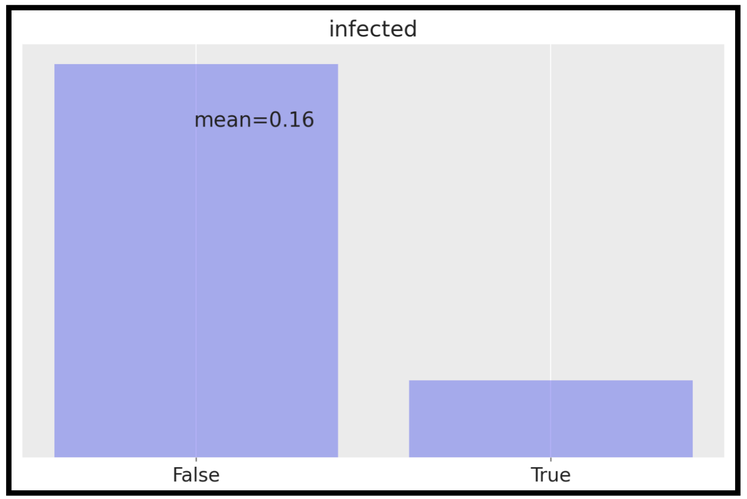
后验概率分布表明，车出现在另一扇门后的概率为 2/3



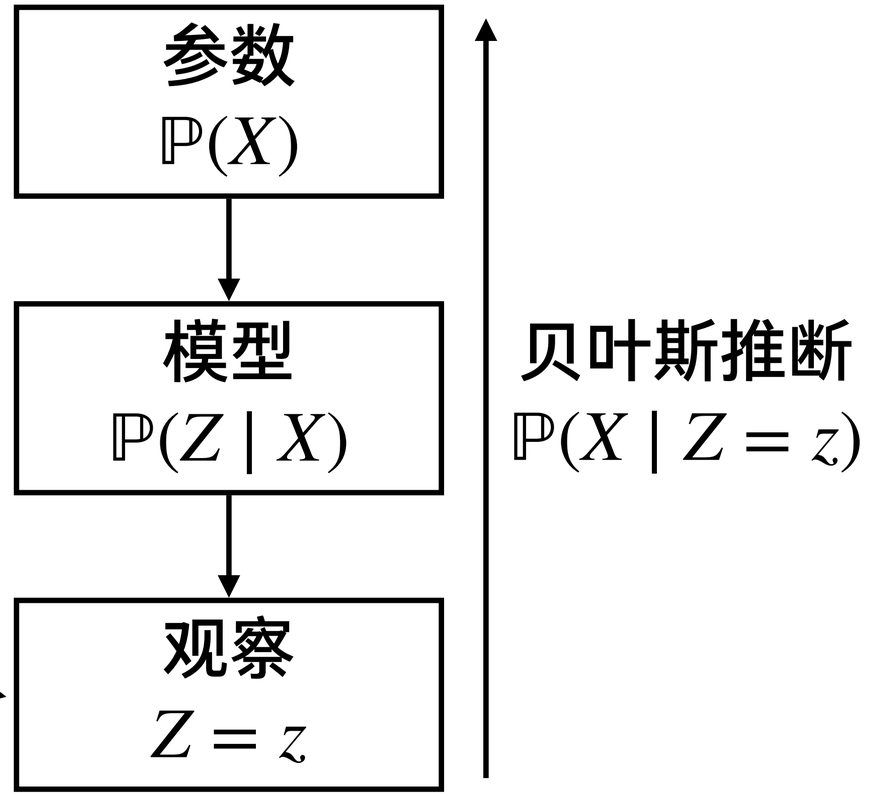
病毒检测

使用概率编程求解

```
with pm.Model() as model:
  infected = pm.Bernoulli("infected", 0.01)

  p_pos = pm.Data("p_pos", [0.05,0.95])
  pos = pm.Bernoulli("pos", p_pos[infected], observed=True)

  idata = pm.sample(10000)
```



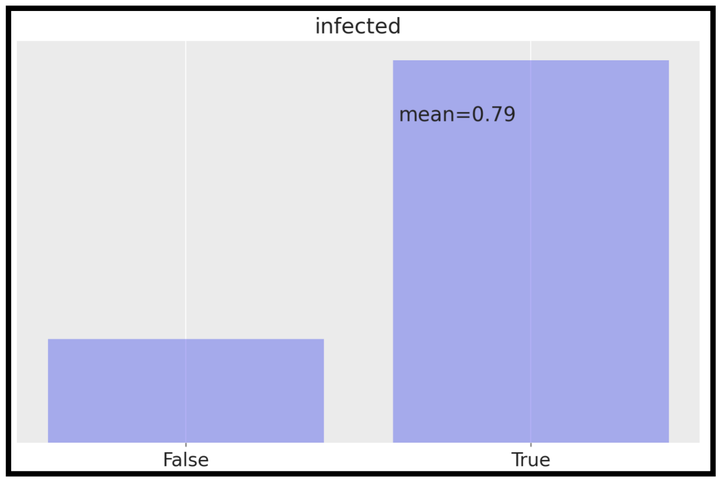
后验概率分布表明，真阳性的概率远低于假阳性



病毒检测，测两次

使用概率编程求解

```
with pm.Model() as model:
  infected = pm.Bernoulli("infected", 0.01)

  p_pos = pm.Data("p_pos", [0.05,0.95])
  pos1 = pm.Bernoulli("pos1", p_pos[infected], observed=True)
  pos2 = pm.Bernoulli("pos2", p_pos[infected], observed=True)

  idata = pm.sample(10000)
```



参数
$\mathbb{P}(X)$

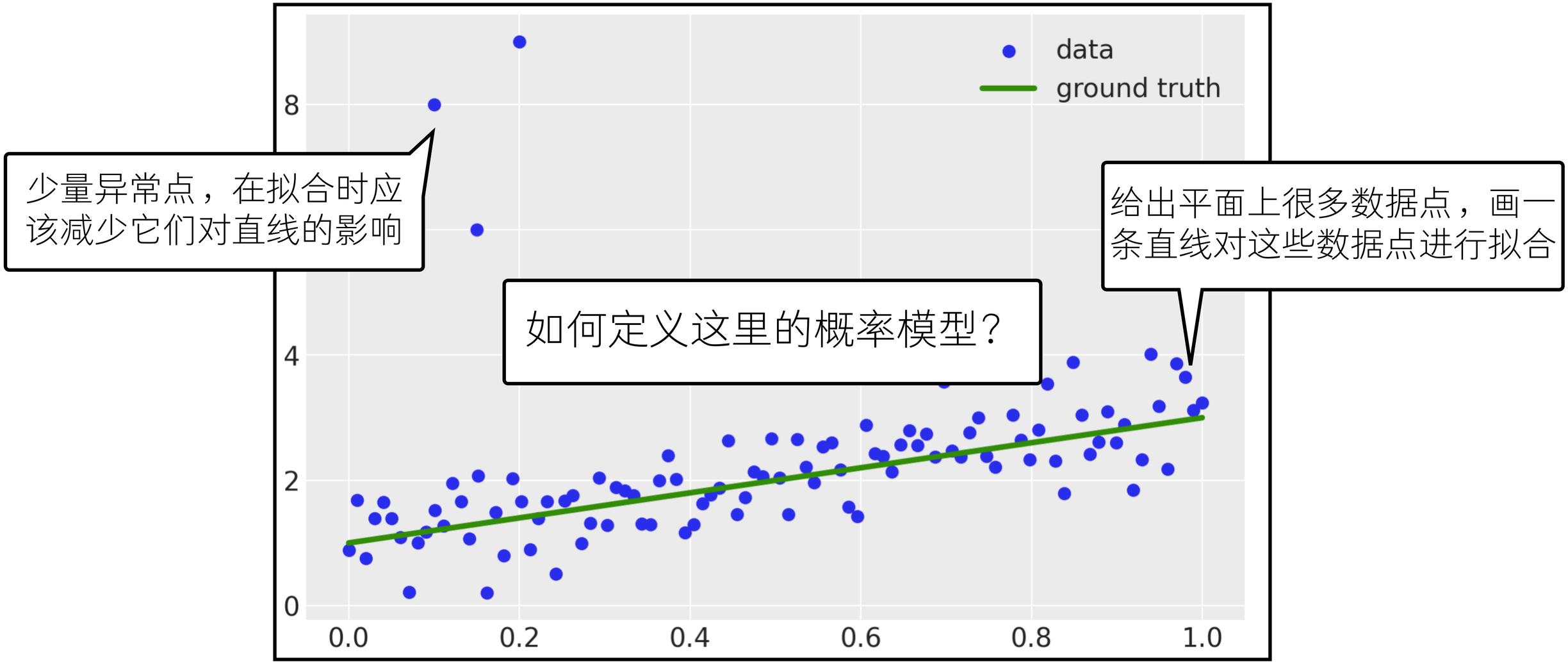
模型
$\mathbb{P}(Z \mid X)$

观察
$Z = z$

贝叶斯推断
$\mathbb{P}(X \mid Z = z)$

后验概率分布表明，测两次都阳性的话，真阳性的概率就很高了

线性回归


data
ground truth
8
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0
少量异常点，在拟合时应该减少它们对直线的影响
如何定义这里的概率模型？
给出平面上很多数据点，画一条直线对这些数据点进行拟合

线性回归

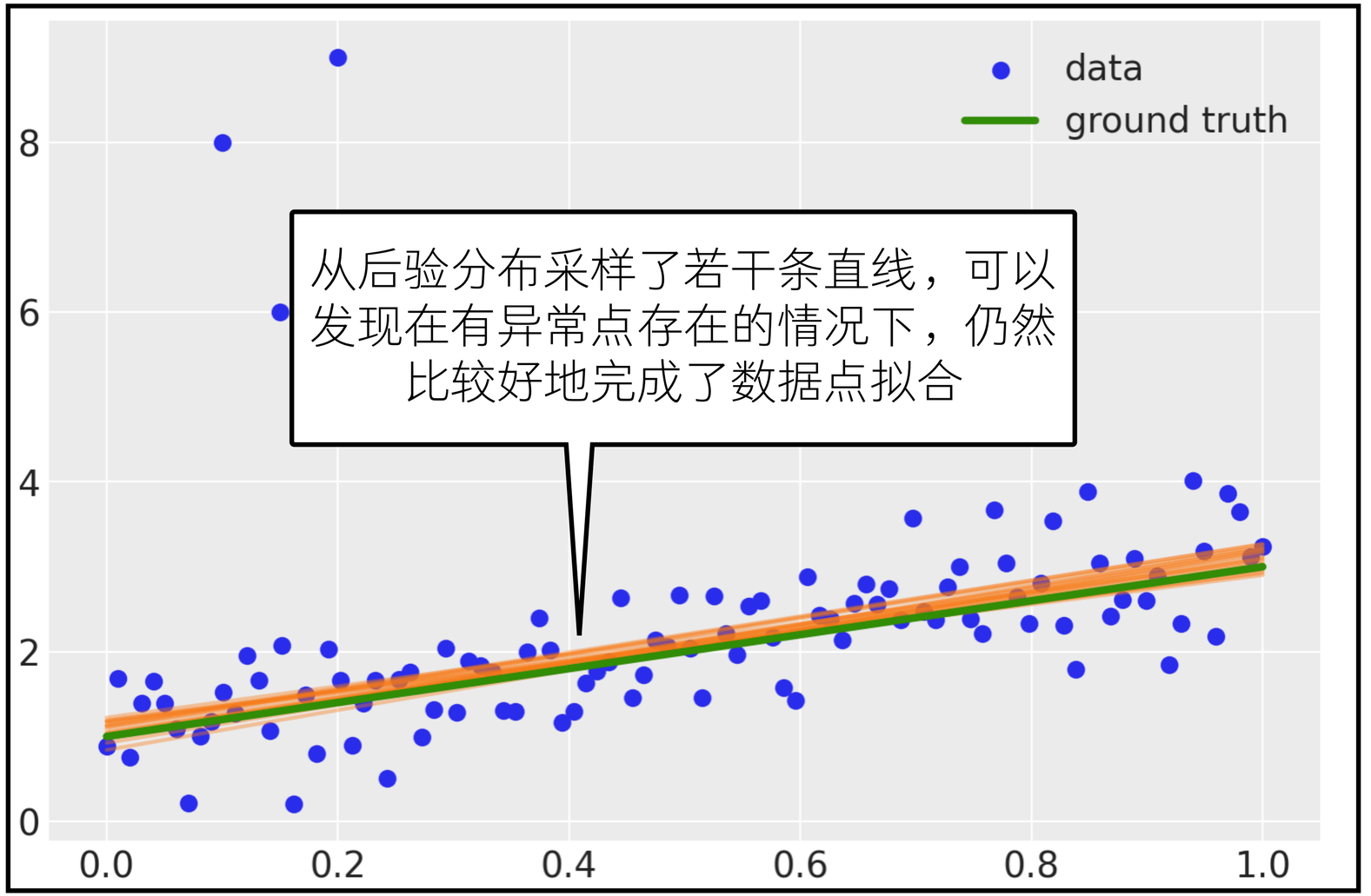
参数：
- 直线截距 b（intercept）
- 直线斜率 k（slope）
- 数据扰动 σ（sigma）

```
with pm.Model() as model:
  intercept = pm.Normal("intercept", mu=0, sigma=1)
  slope = pm.Normal("slope", mu=0, sigma=1)
  sigma = pm.HalfCauchy("sigma", beta=10)

  x = pm.Data("x", xdata)
  mu = pm.Deterministic("mu", intercept + slope * x)
  y = pm.StudentT("y", mu=mu, sigma=sigma, nu=3, observed=ydata)

  idata = pm.sample(10000)
```

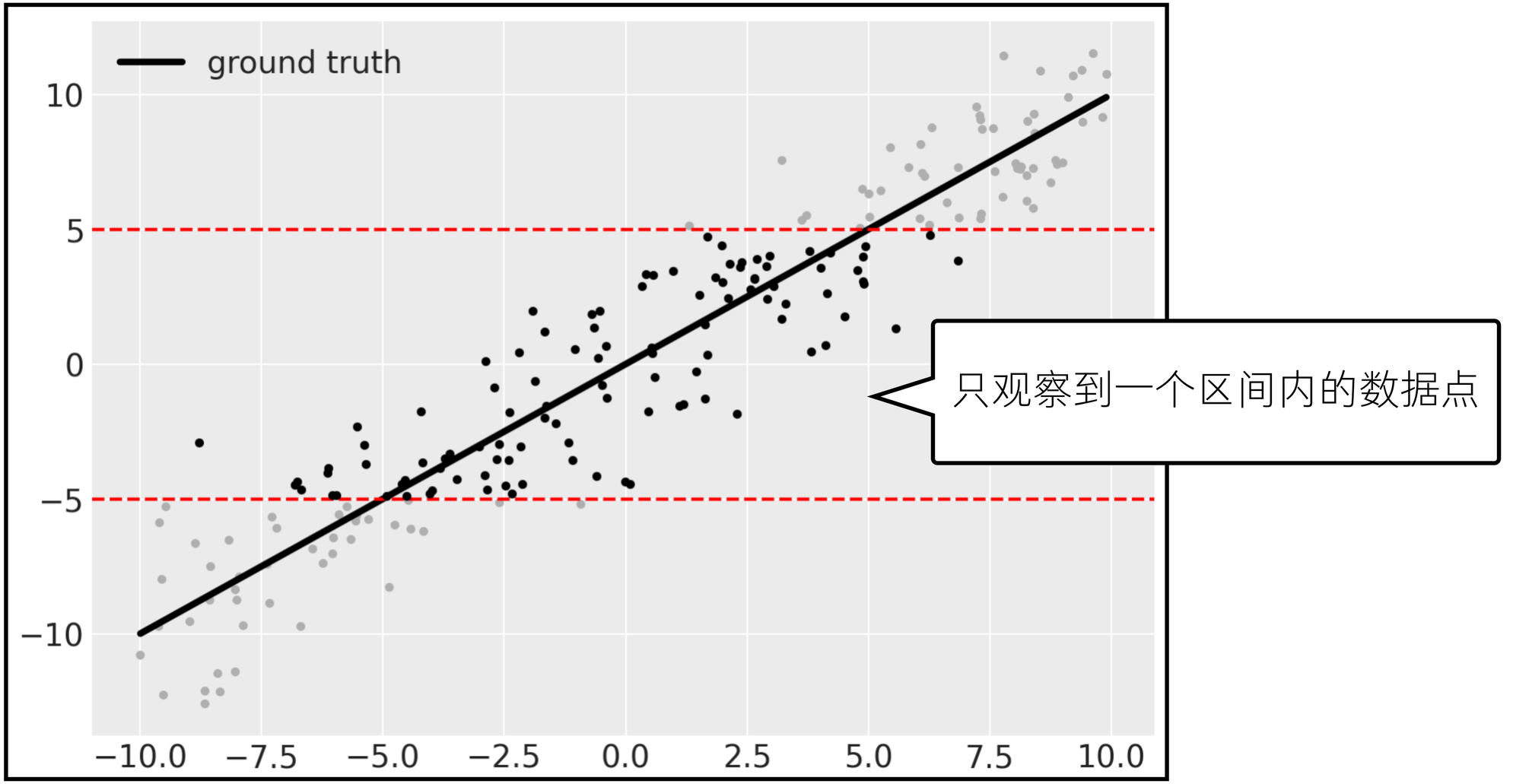
给定输入数据 x，观察对应的输出数据 y，建模方式为 y 等于 $kx+b$ 加上由 σ 控制的随机扰动

通过采样算法获得关于直线参数的后验概率分布

线性回归


data
ground truth
从后验分布采样了若干条直线，可以发现在有异常点存在的情况下，仍然比较好地完成了数据点拟合
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0

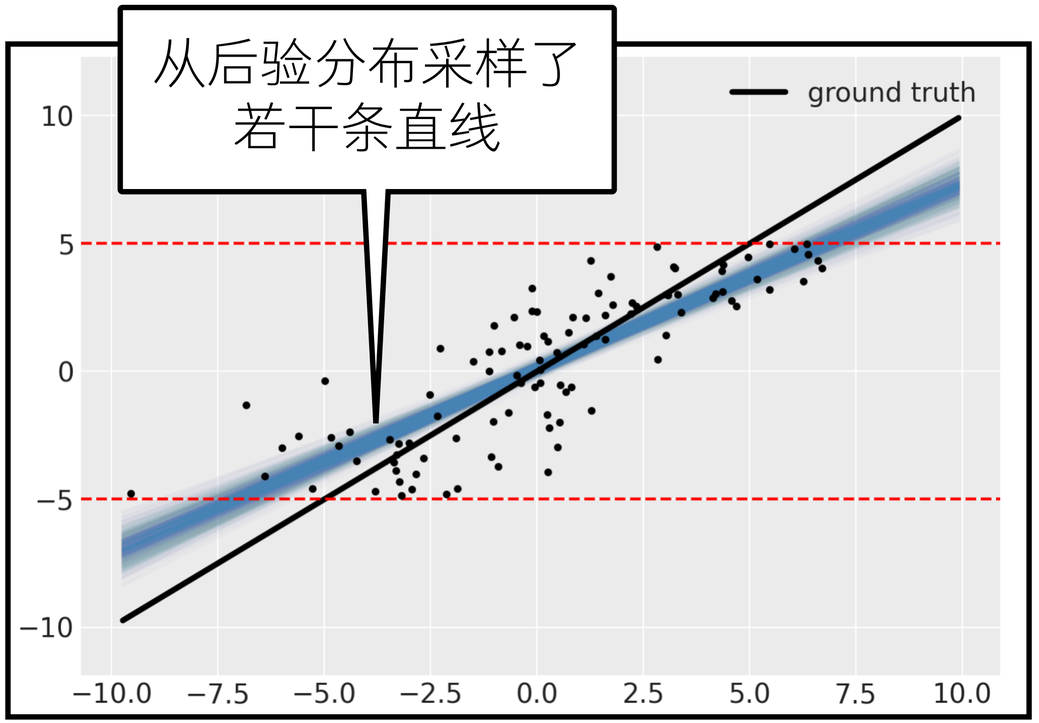
线性回归，但是有未知数据



线性回归，但是有未知数据

```
with pm.Model() as model:
  intercept = pm.Normal("intercept", mu=0, sigma=1)
  slope = pm.Normal("slope", mu=0, sigma=1)
  sigma = pm.HalfCauchy("sigma", beta=10)

  x = pm.Data("x", xdata)
  mu = pm.Deterministic("mu", intercept + slope * x)
  y = pm.StudentT("y", mu=mu, sigma=sigma, nu=3,
observed=ydata)

  idata = pm.sample(10000)
```

之前的线性回归模型

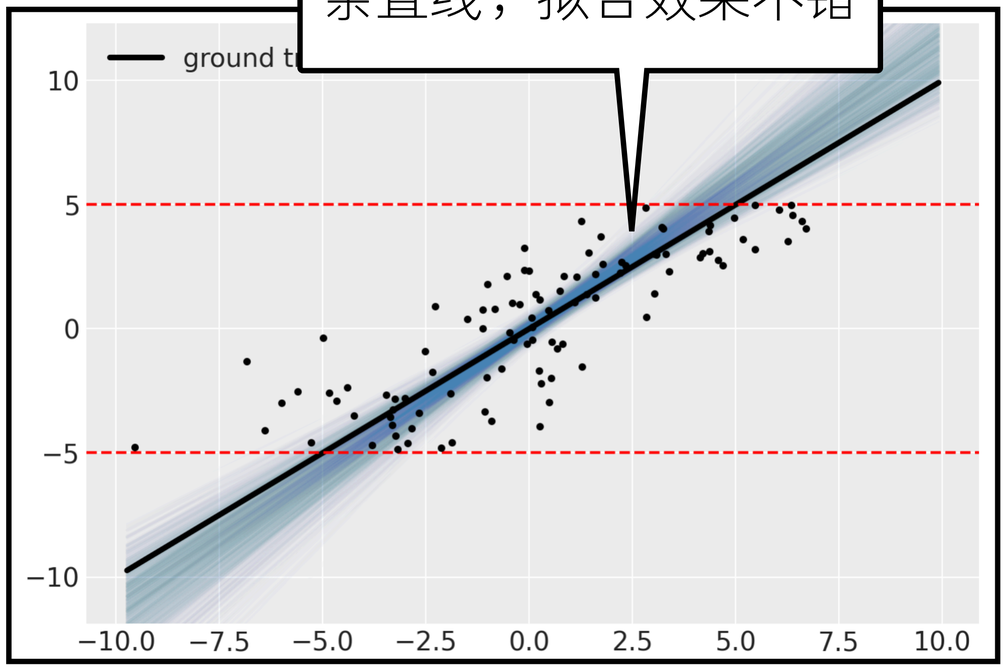


拟合效果一般，没有考虑
到限制区间外的数据

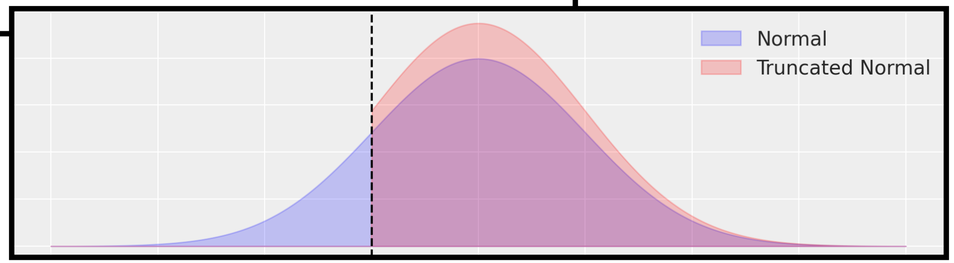
线性回归，但是有未知数据

```python
with pm.Model() as model:
  intercept = pm.Normal("intercept", mu=0, sigma=1)
  slope = pm.Normal("slope", mu=0, sigma=1)
  sigma = pm.HalfCauchy("sigma", beta=10)

  x = pm.Data("x", xdata)
  mu = pm.Deterministic("mu", intercept + slope * x)
  y_dist = pm.StudentT.dist(mu=mu, sigma=sigma, nu=3)
  y = pm.Truncated("y", y_dist, lower=bounds[0],
upper=bounds[1], observed=ydata)

  idata = pm.sample(10000)
```

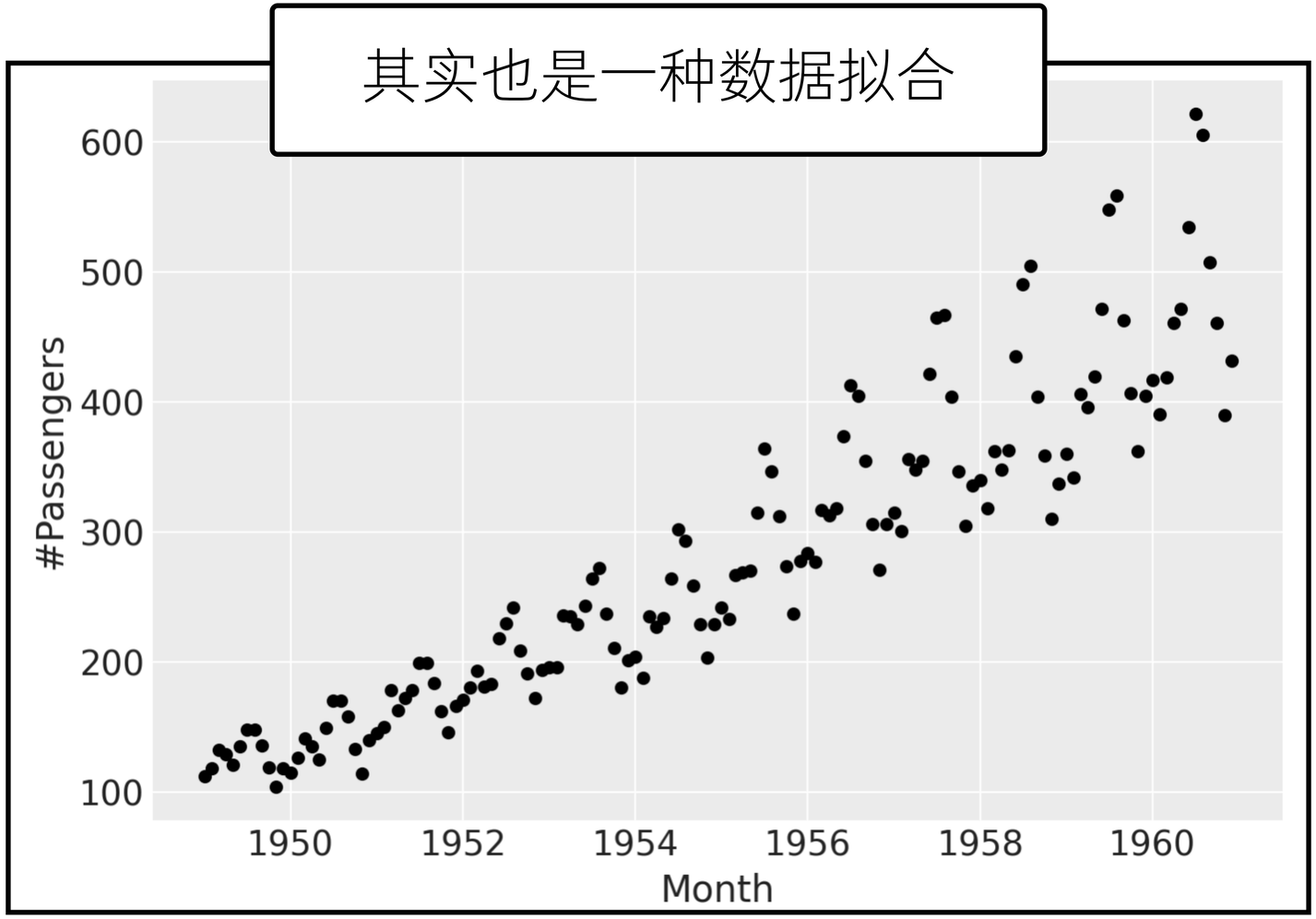
从后验分布采样了若干条直线，拟合效果不错



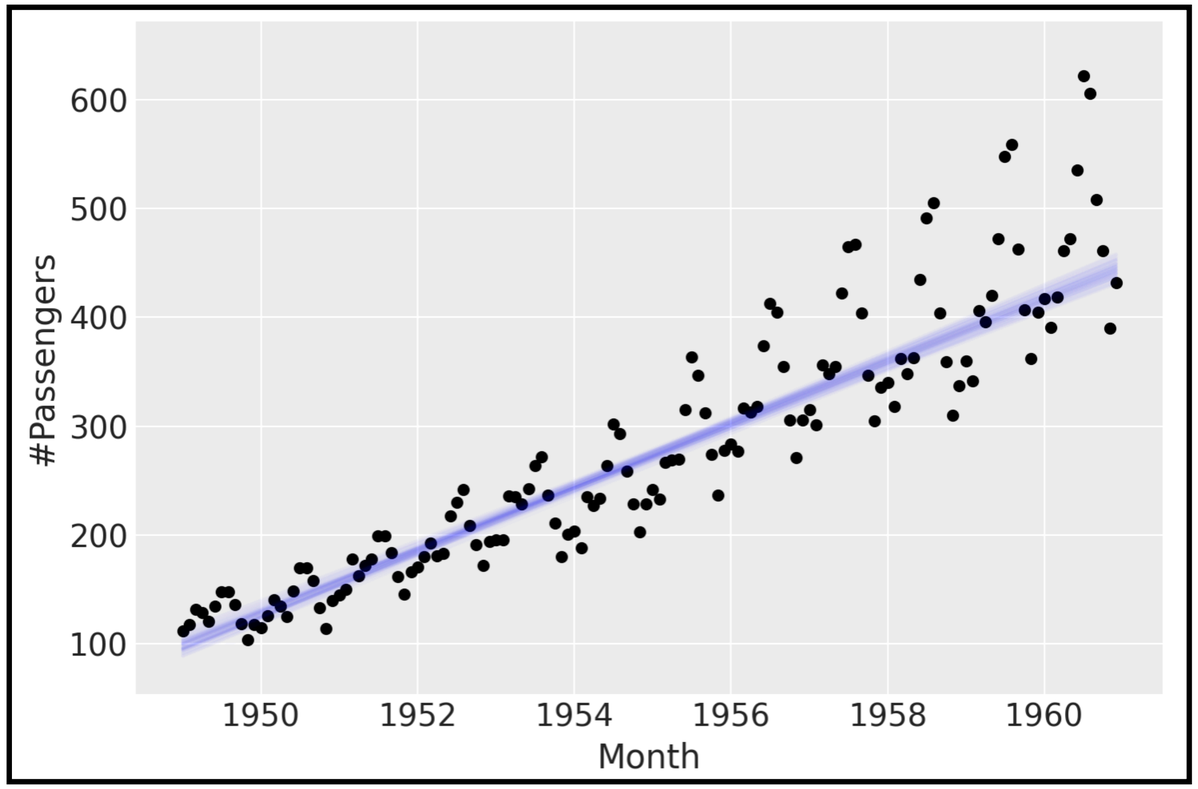
根据限制区间进行截断，并调整似然度计算



时间序列，简化版

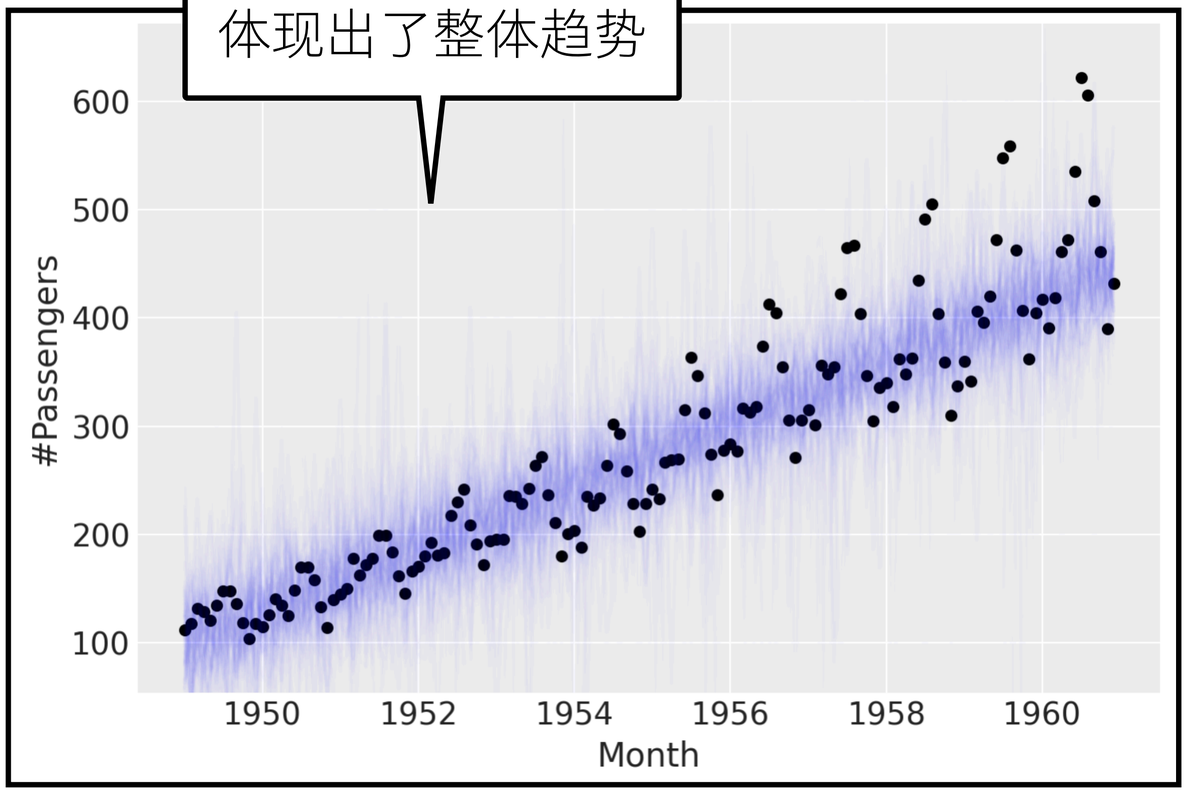

其实也是一种数据拟合
600
500
400
300
200
100
#Passengers
1950
1952
1954
1956
1958
1960
Month

时间序列，简化版



使用之前的线性回归模型
对趋势的后验分布采样

效果较差，基本只
体现出了整体趋势



使用之前的线性回归模型
对每个点预测的后验分布采样

时间序列，简化版

- 注意到，该时间序列问题有较强的周期性
  - 每一年中，随着月份/季节会呈现固定的增长或下降趋势
- 使用 Fourier Series：$s(t) = \sum_{n=1}^{N} (\alpha_n \cos(\frac{2\pi nt}{P}) + \beta_n \sin(\frac{2\pi nt}{P}))$
  - P 是周期长度（比如一年的天数），N 是度量周期变化快慢的参数
  - 对于线性模型 $y = kt + b$，调整为 $y = (kt + b) \cdot (1 + s(t))$
  - 系数 α_n, β_n 是新的模型参数，同样使用贝叶斯推断来采样其后验分布

时间序列，简化版

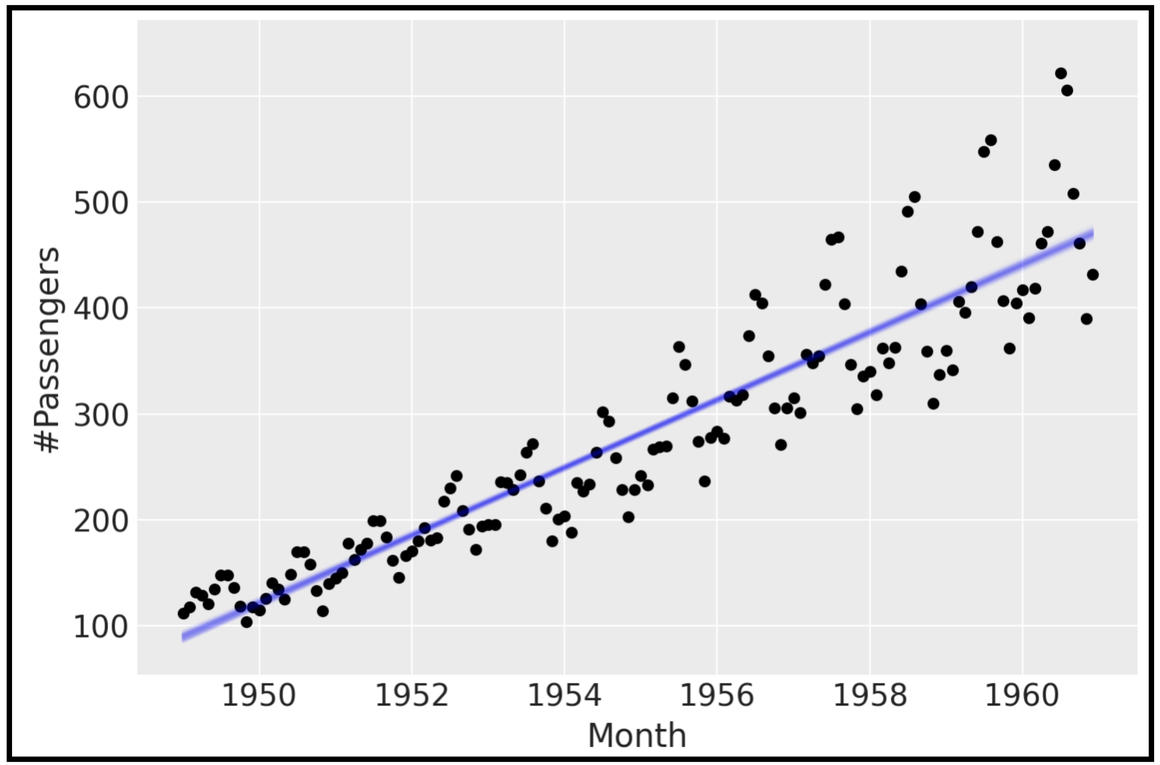
```python
with pm.Model() as model:
  intercept = pm.Normal("intercept", mu=0, sigma=1)
  slope = pm.Normal("slope", mu=0, sigma=1)
  sigma = pm.HalfCauchy("sigma", beta=10)

  features = pm.Data("features", fourier_features)
  fourier = pm.Normal("fourier", mu=0, sigma=0.1, shape=(2*N,))
  seasonality = pm.Deterministic("seasonality", pm.math.dot(fourier, features.T))

  t = pm.Data("t", tdata)
  trend = pm.Deterministic("trend", intercept + slope * t)
  mu = pm.Deterministic("mu", trend * (1 + seasonality))
  y = pm.StudentT("y", mu=trend, sigma=sigma, nu=3, observed=ydata)

  idata = pm.sample(10000)
```

$$[\cos(\frac{2\pi \cdot 1t}{P}), \sin(\frac{2\pi \cdot 1t}{P}), \cdots, \cos(\frac{2\pi \cdot Nt}{P}), \sin(\frac{2\pi \cdot Nt}{P})]$$

$$[\alpha_1, \beta_1, \cdots, \alpha_N, \beta_N]$$

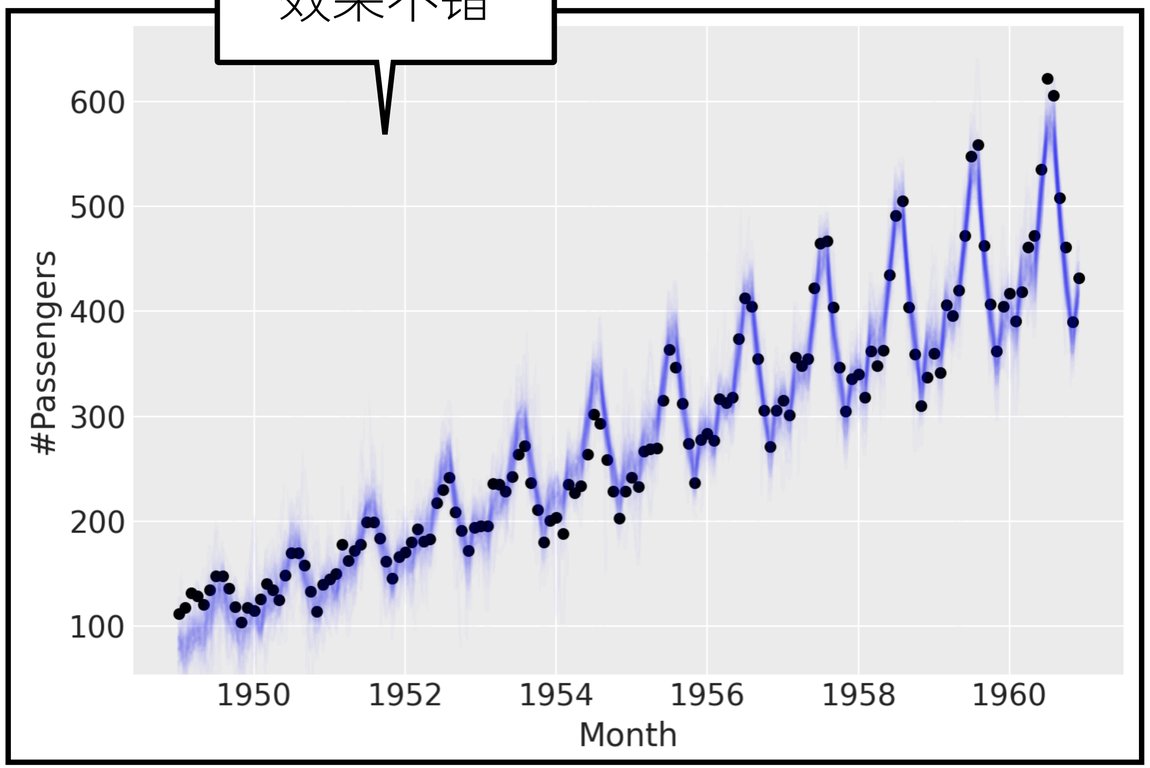
$$y = (kt + b) \cdot (1 + s(t))$$

时间序列，简化版



使用调整后的模型
对趋势的后验分布采样

效果不错

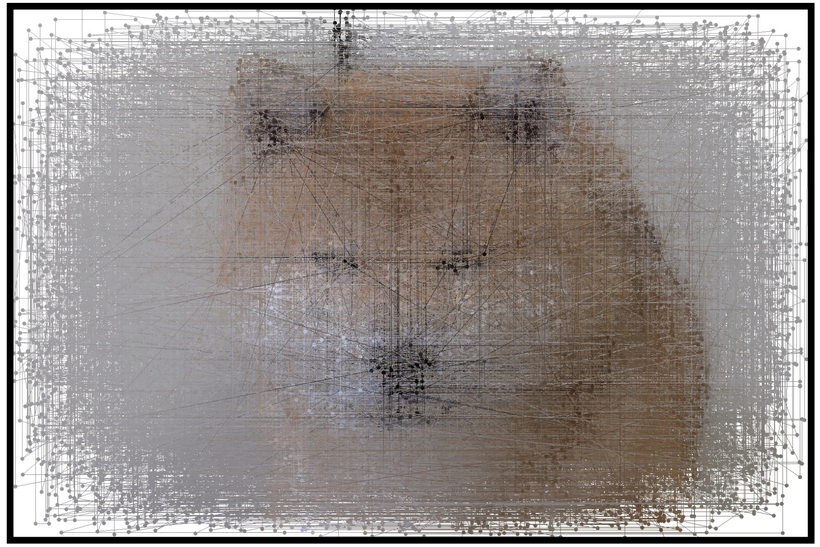


使用调整后的模型
对每个点预测的后验分布采样

过程化设计之图片生成

设计目标
（给定原图）

设计方案



设计方案

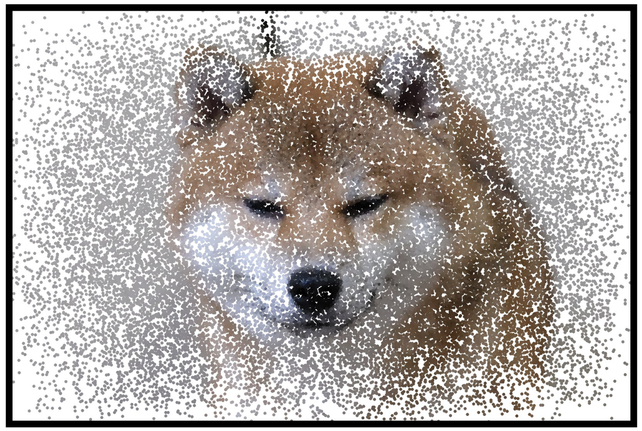
过程化设计之图片生成

- 如何建模成贝叶斯推断问题？
  - X：图片中的像素点，Z：像素点是否在原图中
  - 先验 $\mathbb{P}(X)$：均匀分布，图片边缘的概率可以稍微低一些
  - 似然度 $\mathbb{P}(Z = z \mid X)$：给定像素点，概率与它的「暗度」呈正比
  - 后验 $\mathbb{P}(X \mid Z = z)$：观察到像素点在原图的情况，像素点的条件概率分布
- 只考虑原图中的像素点，所以 z 就是 $true$
- 本质上是采样根据先验和「暗度」的乘积决定的概率分布

过程化设计之图片生成

```
with pm.Model() as model:
  x = pm.BetaBinomial("x", n=h-1, alpha=2, beta=2)
  y = pm.BetaBinomial("y", n=w-1, alpha=2, beta=2)
  u = pm.Deterministic("u", x * w + y)
  p = pm.Data("p", darkness)
  likelihood = pm.Bernoulli("likelihood", p=p[u], observed=True)

  idata = pm.sample(10000)
```

原图中越暗的地方，概率越高



相邻的样本间连线

语言模型概率编程

先来看看语言模型的大致原理

是否可以与概率编程结合？

输入： 提示或上下文 → 语言模型 → **输出：** 下一个词的概率

<句子开始>

<句子开始> **我**

<句子开始> 我 **爱**

<句子开始> 我 爱 **火锅**

<句子开始> 我 爱 火锅 **<句子结束>**

0.1：我，0.1：你，0.05：她们，……

0.2：爱，0.1：是，0.1：把，……

0.4：你，0.4：她，0.1：火锅，……

0.3：<句子结束>，0.1：面，……

语言模型概率编程

- X：文本，Z：各种约束
- 先验 $\mathbb{P}(X)$：语言模型生成的文本
- 似然度 $\mathbb{P}(Z = z \mid X)$：给定文本，满足 z 给出的约束的概率
- 后验 $\mathbb{P}(X \mid Z = z)$：观察到对文本的约束 z，文本的条件概率分布

约束：填空

Infilling

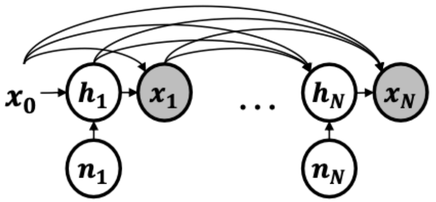
Query

Prompt:
"To tell the truth, every**[BLANK]** he**[BLANK]** to**[BLANK]** another**[BLANK]**."

Posterior Sample

"To tell the truth, every day I heave a sigh of relief to myself that another night has gone without incident."

Probabilistic Graphical Model



Language Model Probabilistic Program (*Python pseudocode*)

```
def infilling(prompt):
  # Initialize
  parts = prompt.split("[BLANK]")
  s = parts[0]
  x = new_context(s)
  # Generate for each blank
  for part in parts[1:]:
    n = sample(geom(0.5)) + 1
    for _ in range(n):
      s += sample(llm(x))
    for t in tokenize(part):
      s += observe(llm(x), t)
  return s
```

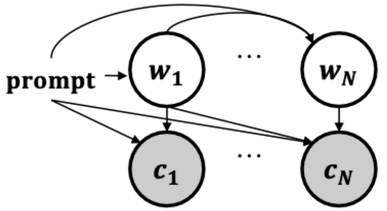
约束：词长

Hard Constraints

Prompt:
"The Fed says"
Constraint: (use only short words)

```
def constraint(p):
  return len(p.split()[-1]) <= 5
```

"The Fed says it will taper, but rate hikes are still years away."



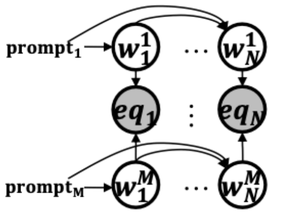

```
def constraints(prompt, constraint):
  # Initialize
  s = ""
  x = new_context(prompt)

  # Generate until EOS
  while True:
    tok = sample(llm(x))
    if tok == EOS:
      break
    s += tok
    condition(constraint(s))
  return s
```

约束：交叉

Prompt Intersection

Prompts:
"My favorite physicist is probably"
"My favorite writer is probably"

" Richard Feynman. I really admire how he communicates complex ideas so clearly."




```
def prompt_intersect(prompts):
  # Initialize
  xs = [new_context(p)
        for p in prompts]
  s  = ""
  # Generate until EOS
  while True:
    tok = sample(llm(xs[0]))
    for x in xs[1:]:
      observe(llm(x), tok)
    if tok == EOS:
      break
    s += tok
  return s
```

A. K. Lew, T. Zhi-Xuan, G. Grand, and V. K. Mansinghka. 2023. Sequential Monte Carlo Steering of Large Language Models using Probabilistic Programs. In *arXiv*.

从模拟退火到概率编程

模拟退火算法

模拟退火的正确性

概率论 101

贝叶斯公式

贝叶斯推理

MCMC 算法及其变种

概率编程语言