USTC

Chapter 15
Temporal Probability Models

王子磊 (Zilei Wang)
Email: zlwang@ustc.edu.cn
http://vim.ustc.edu.cn

提纲

- 时间与不确定性 (Time and uncertainty)
- 推理：滤波 (filtering)、预测 (prediction) 与平滑 (smoothing)
- 隐马尔科夫模型 (Hidden Markov models, HMM)
- 卡尔曼滤波器 (Kalman filters)
- 动态贝叶斯网络 (Dynamic Bayesian networks)
- 粒子滤波 (Particle filtering)

时间与不确定性

- 世界是动态变化的，我们需要追踪和预测它
  - 糖尿病管理 vs 车辆诊断
- 基本思想：每个时间步拷贝状态和证据变量

$\mathbf{X}_t$ = set of unobservable state variables at time t
e.g., $BloodSugar_t$, $StomachContents_t$, etc.

$\mathbf{E}_t$ = set of observable evidence variables at time t
e.g., $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$

  - 这里假定时间是离散的，具体步长依赖于给定的问题
  - 记号：$\mathbf{X}_{a:b} = \mathbf{X}_a, \mathbf{X}_{a+1}, \ldots, \mathbf{X}_{b-1}, \mathbf{X}_b$

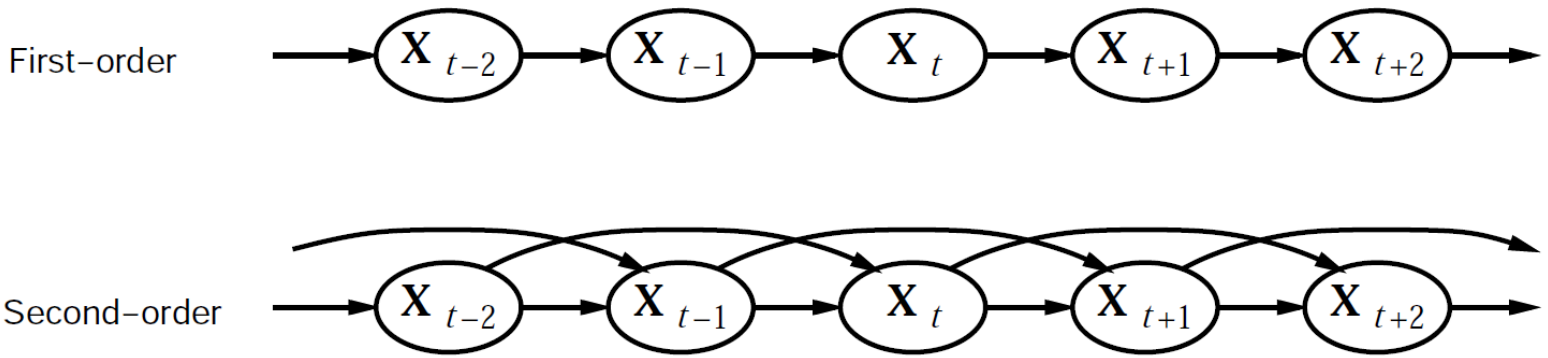
Markov 过程 (Markov 链)

- 针对状态和证据变量构建一个贝叶斯网络：父节点？
- 马尔科夫假设 (Markov assumption)： $\mathbf{X}_t$ 依赖于 $\mathbf{X}_{0:t-1}$ 的一个有限子集
  - 一阶马尔科夫过程 (first-order Markov process)

$$\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$$

  - 二阶马尔科夫过程 (second-order Markov process)

$$\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-2}, \mathbf{X}_{t-1})$$



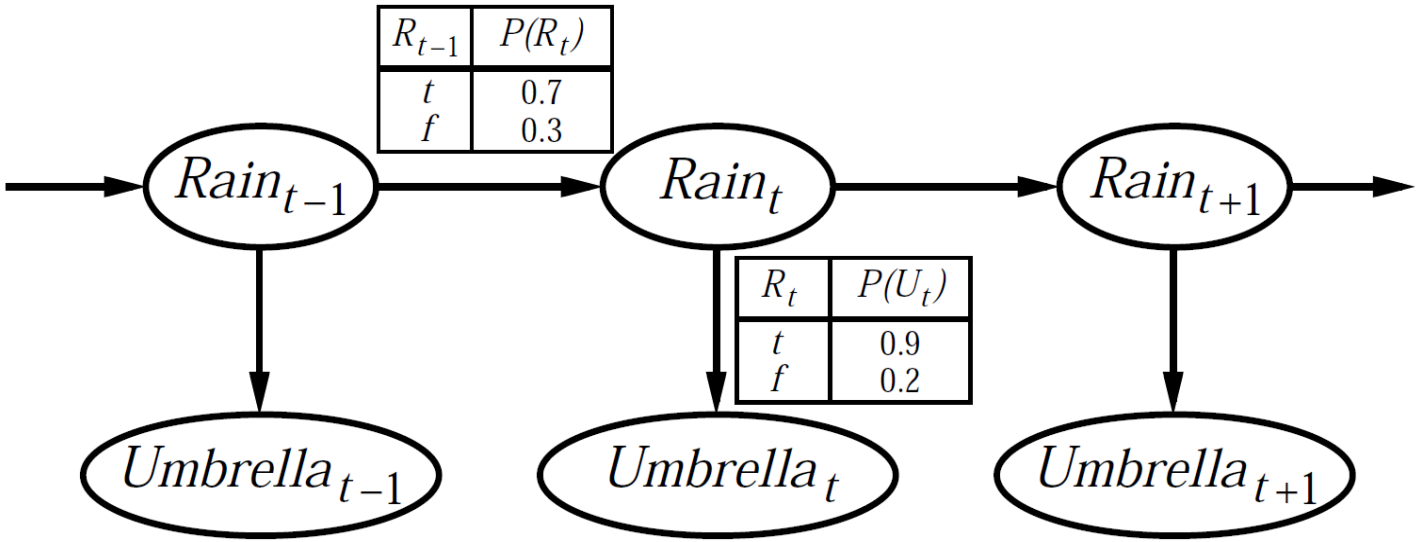
Markov 过程 (Markov 链)

- 传感器马尔科夫假设 (Sensor Markov assumption)

$$\mathbf{P}(\mathbf{E}_t|\mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$$

- 稳态过程 (Stationary process)：
  转移模型 $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ 和传感器模型 $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ 针对所有的时刻 t 是固定不变的

示例



- 一阶 Markov 假设在真实世界中并不是精确成立的
- 可能的处理方法：
  - 增加 Markov 过程的阶数
  - 增加状态，如添加 $Temp_t$, $Pressure_t$
- 如在机器人运动中，用 $Battery_t$ 来增强位置和速度

推理任务

- 滤波 (filtering)：$\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$
  - 置信状态 (belief state) —— 理性 Agent 决策过程的输入
- 预测 (prediction)：$\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$
  - 可能动作序列的评估
- 平滑 (smoothing)：$\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$
  - 过去状态的最佳估计，对学习很重要
- 最可能解释 (most likely explanation)：$\arg\max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$
  - 生成观察序列的最可能状态序列
  - E.g., 语音识别，噪声信道的解码

滤波 (Filtering)

- 目标：设计一个递归的状态估计算法

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, \mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t}))$$

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1}) && \text{(证据分解)}\\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) && \text{(贝叶斯规则)}\\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) && \text{(传感器 Markov 假设)}\end{aligned}$$

i.e., 预测 + 估计，这里预测通过在 $\mathbf{X}_t$ 上求和计算

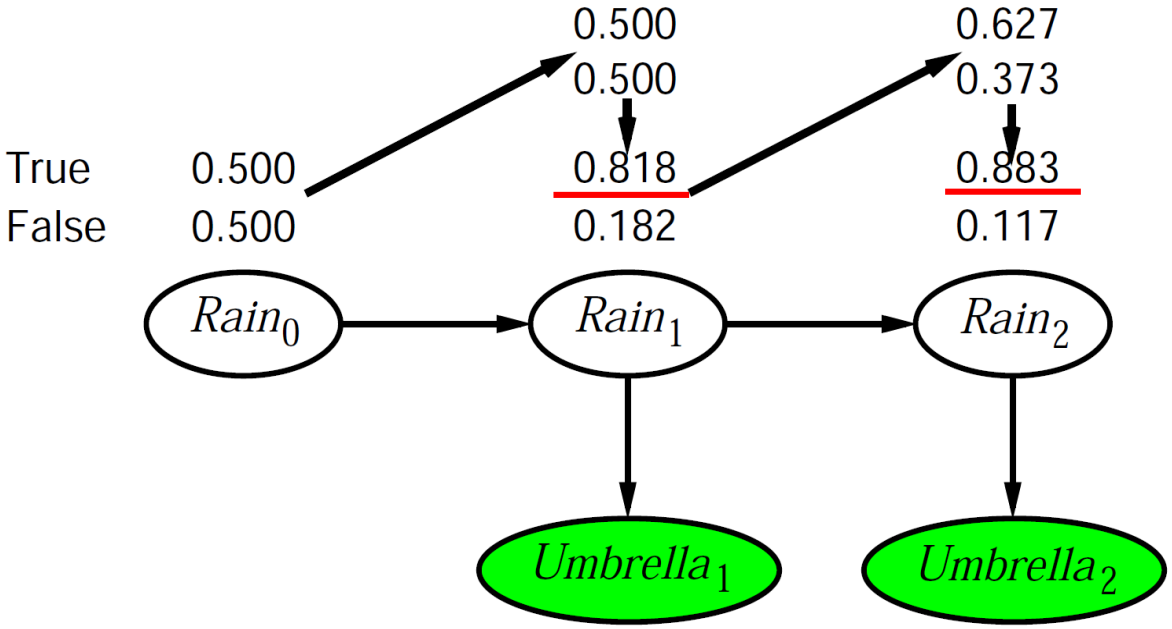
$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\textstyle\sum_{\mathbf{x}_t}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t, \mathbf{e}_{1:t})P(\mathbf{x}_t|\mathbf{e}_{1:t})\\ &= \alpha\underline{\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})}\textstyle\sum_{\mathbf{x}_t}\underline{\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)}P(\mathbf{x}_t|\mathbf{e}_{1:t})\end{aligned}$$

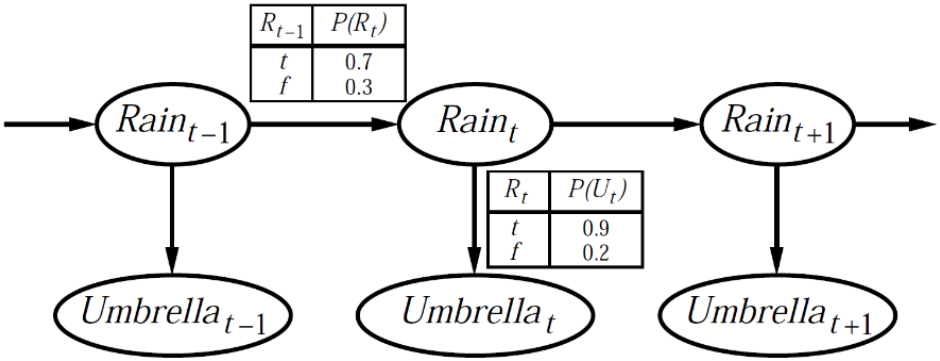
模型给定

⇨ $\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$ where $\mathbf{f}_{1:t} = \mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$

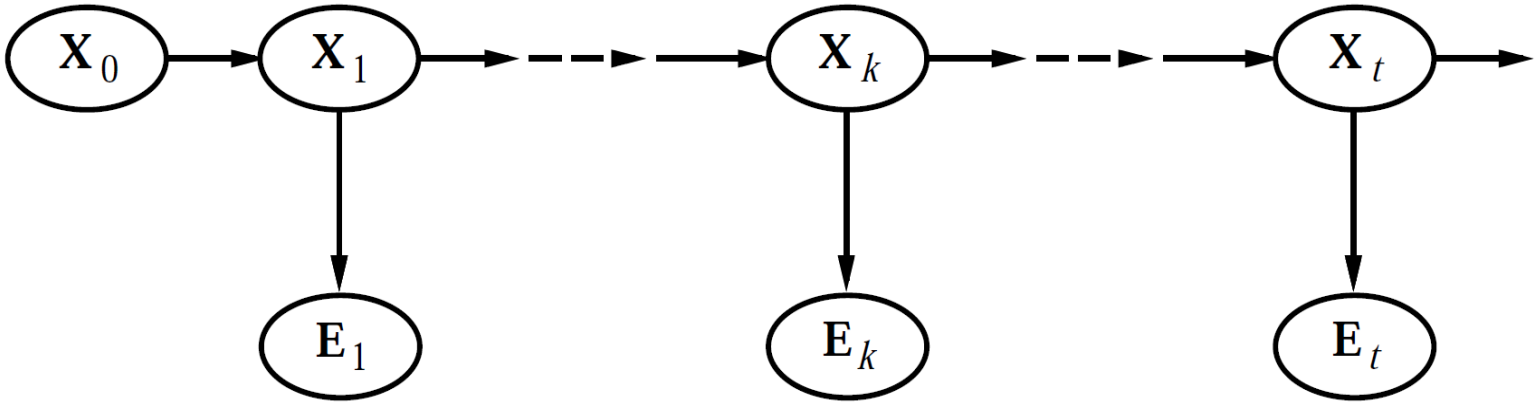
- 时间和空间复杂度都是常数的（与时刻 t 无关）

滤波示例


0.500
0.500
0.627
0.373
True
False
0.500
0.500
0.818
0.182
0.883
0.117
$Rain_0$
$Rain_1$
$Rain_2$
$Umbrella_1$
$Umbrella_2$


R_{t-1}
$P(R_t)$
t
f
0.7
0.3
$Rain_{t-1}$
$Rain_t$
$Rain_{t+1}$
R_t
$P(U_t)$
t
f
0.9
0.2
$Umbrella_{t-1}$
$Umbrella_t$
$Umbrella_{t+1}$

平滑 (Smoothing)

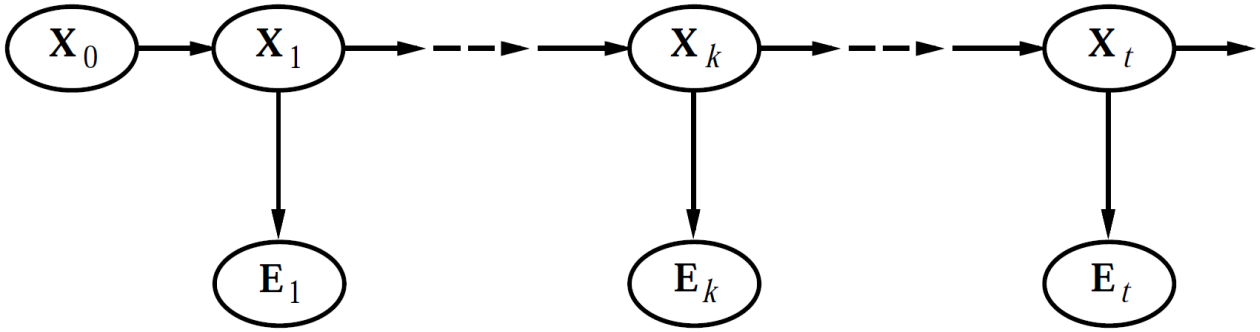


- 将证据变量 $\mathbf{e}_{1:t}$ 分解为 $\mathbf{e}_{1:k}$, $\mathbf{e}_{k+1:t}$:

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k},\mathbf{e}_{k+1:t}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k,\mathbf{e}_{1:k}) && \text{(贝叶斯规则)} \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) && \text{(条件独立性)} \\
&= \alpha\mathbf{f}_{1:k}\mathbf{b}_{k+1:t}
\end{aligned}$$

反向消息

平滑 (Smoothing)

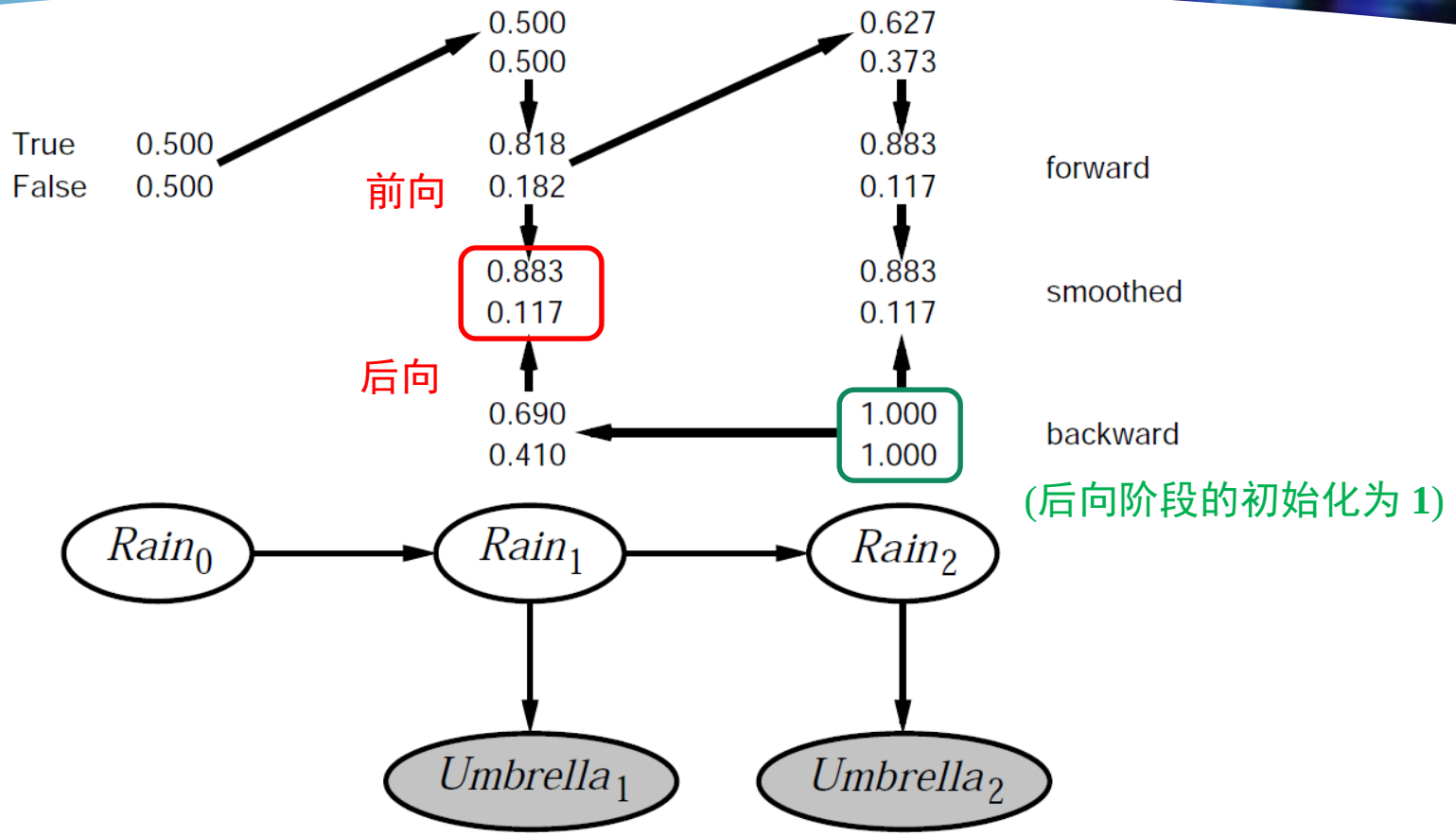


- 通过反向递归计算反向消息

$$\begin{aligned}\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}}\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k,\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)\\ &= \sum_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)\\ &= \sum_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)\end{aligned}$$

⇨ $\mathbf{b}_{k+1:t}$ = BACKWARD($\mathbf{b}_{k+2:t}$, $\mathbf{e}_{k+1}$)

平滑示例



- 前向-后向算法：缓存前向消息
  - 具有 t 线性的时间复杂性（多形树推理）和 $O(t|f|)$ 的空间复杂性

最可能解释 (most likely explanation)

- Most likely sequence ≠ sequence of most likely status !!!
- Most likely path to each $\mathbf{x}_{t+1}$

  = most likely path to some $\mathbf{x}_t$ plus one more step

$$\max_{\mathbf{x}_1 \ldots \mathbf{x}_t} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{X}_{t+1} | \mathbf{e}_{1:t+1})$$

$$= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} \left(\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} P(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{e}_{1:t}) \right)$$

- 与滤波是一样的，除了 $\mathbf{f}_{1:t}$ 被替换为

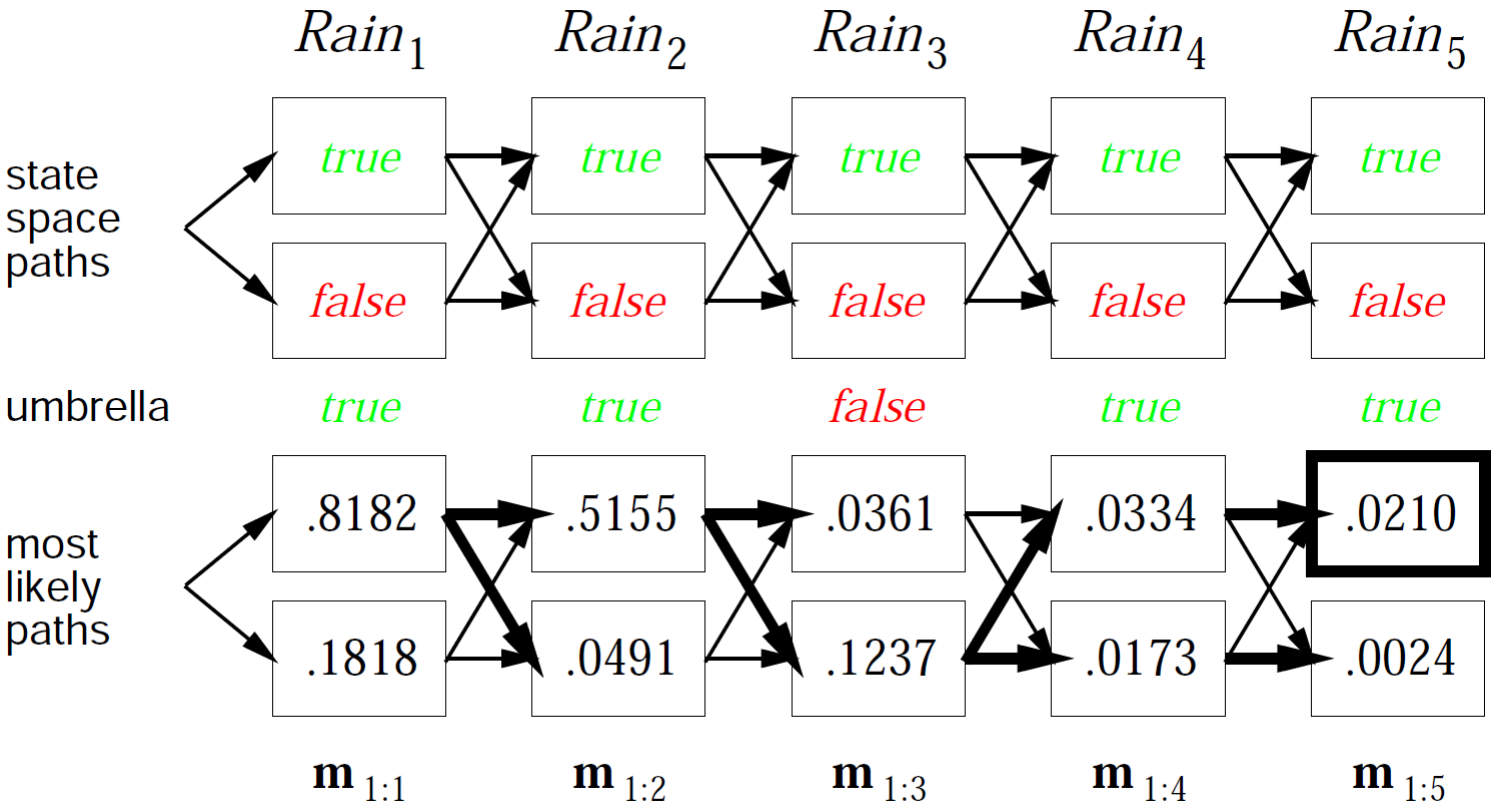
$$\mathbf{m}_{1:t} = \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{X}_t | \mathbf{e}_{1:t}),$$

  i.e., $\mathbf{m}_{1:t}(i)$ 给出了到状态 i 最可能路径的概率

- Viterbi 算法：用 max 代替 sum 进行更新

$$\mathbf{m}_{1:t+1} = \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} (\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \mathbf{m}_{1:t})$$

Viterbi 示例


$Rain_1$
$Rain_2$
$Rain_3$
$Rain_4$
$Rain_5$
state space paths
true
false
umbrella
true
true
false
true
true
most likely paths
.8182
.5155
.0361
.0334
.0210
.1818
.0491
.1237
.0173
.0024
$\mathbf{m}_{1:1}$
$\mathbf{m}_{1:2}$
$\mathbf{m}_{1:3}$
$\mathbf{m}_{1:4}$
$\mathbf{m}_{1:5}$

隐马尔科夫模型 (Hidden Markov Models, HMM)

- $\mathbf{X}_t$ 是单个离散随机变量（$\mathbf{E}_t$ 通常也是）
  $\mathbf{X}_t$ 的域是 $\{1, 2, \ldots, S\}$
- 转移矩阵：$\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i)$, e.g., $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$
- 传感矩阵：单一时间步为 $\mathbf{O}_t$，对角元素为 $P(e_t | X_t = i)$

  e.g., with $U_1 = true$, $\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$
- 前向和反向消息 (列向量)：

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t}$$
$$\mathbf{b}_{k+1:t} = \mathbf{T} \mathbf{O}_{k+1} \mathbf{b}_{k+2:t}$$

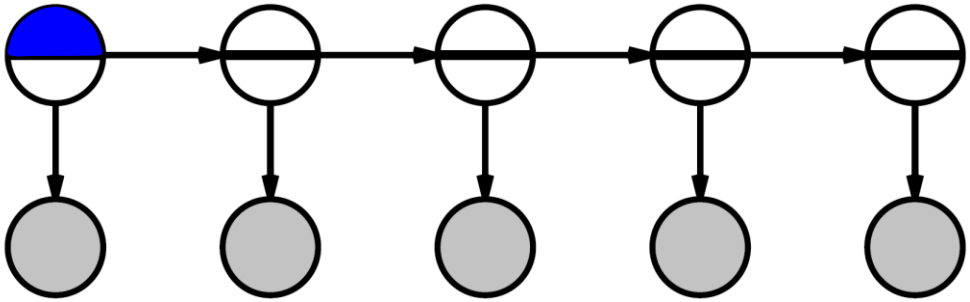
- 前向-反向算法的时间复杂度为 $O(S^2 t)$，空间复杂度为 $O(St)$

Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i$, $\mathbf{b}_i$

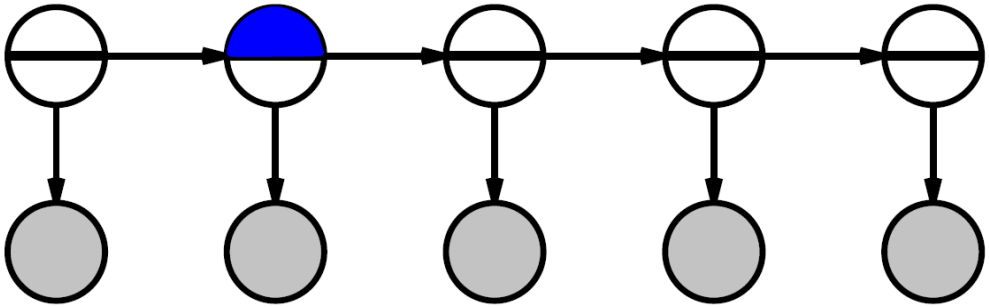


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i$, $\mathbf{b}_i$

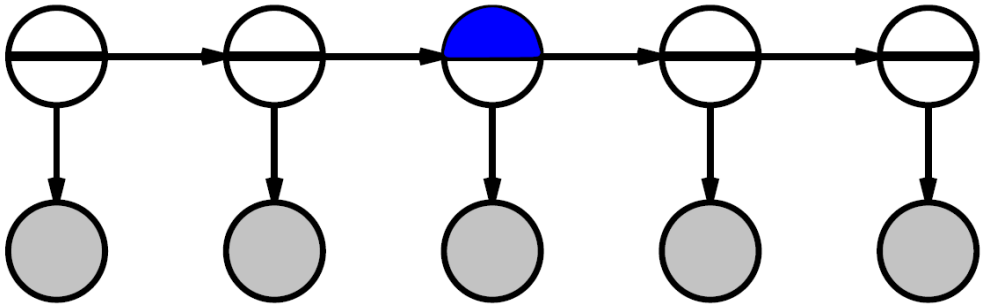


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1}\mathbf{T}^{\top}\mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top}\mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^{\top})^{-1}\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i, \mathbf{b}_i$

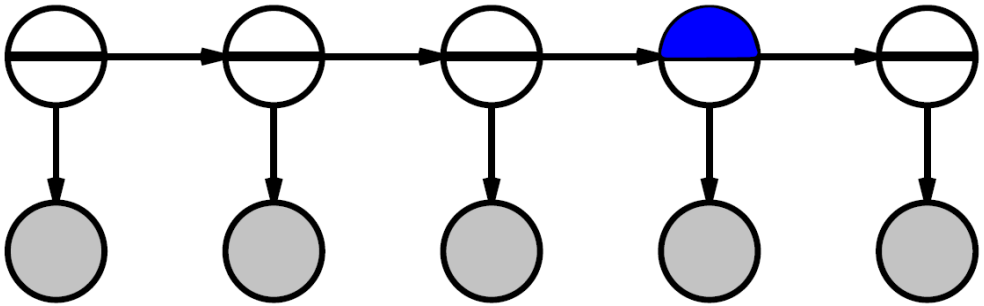


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1}\mathbf{T}^{\top}\mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top}\mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^{\top})^{-1}\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i, \mathbf{b}_i$

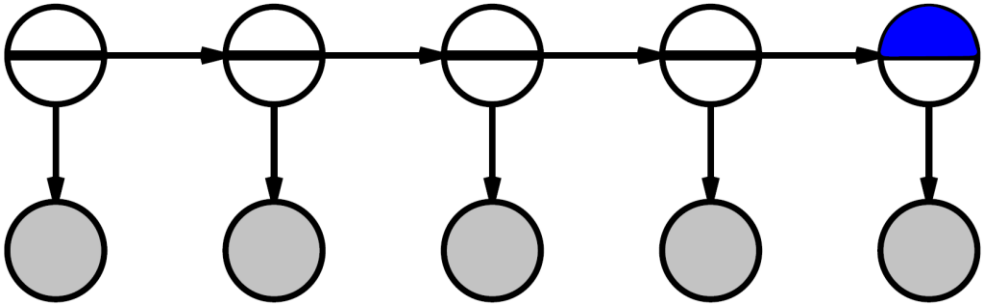


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i, \mathbf{b}_i$

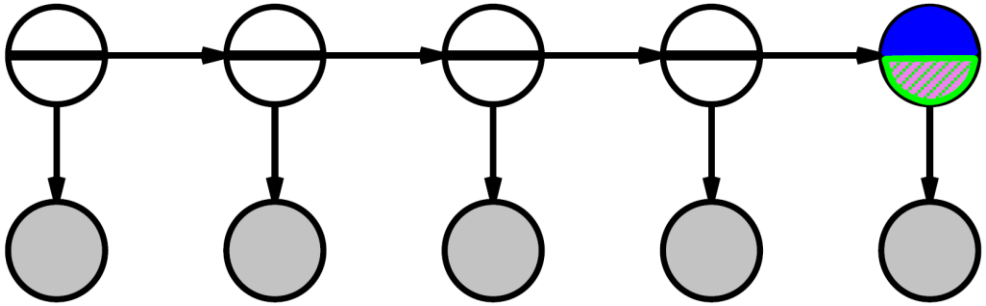


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i, \mathbf{b}_i$

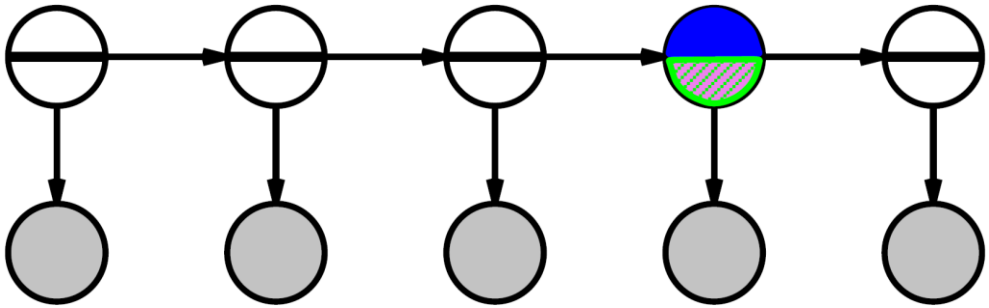


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i, \mathbf{b}_i$

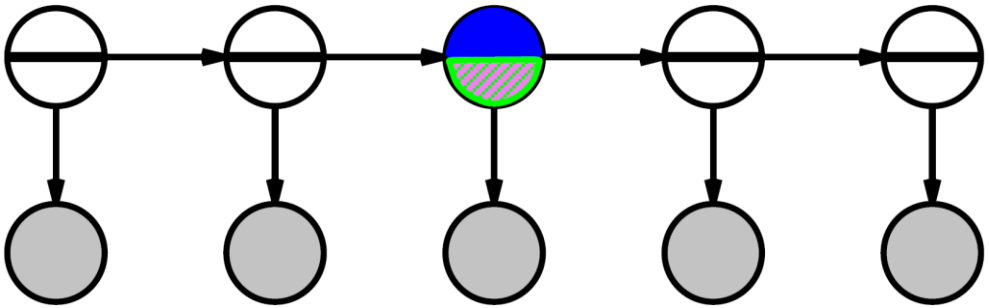


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i$, $\mathbf{b}_i$

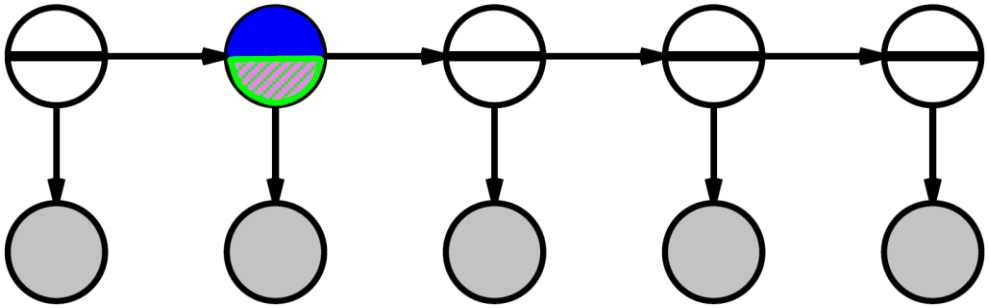


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i, \mathbf{b}_i$

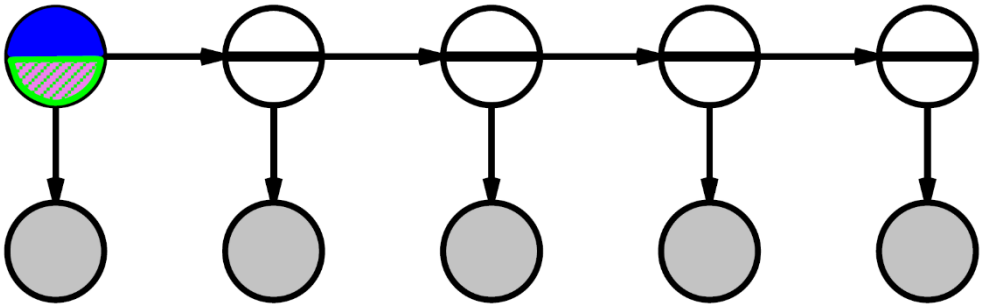


Country dance algorithm

- 通过反向执行前向算法，能够避免存储平滑处理中所有前向消息

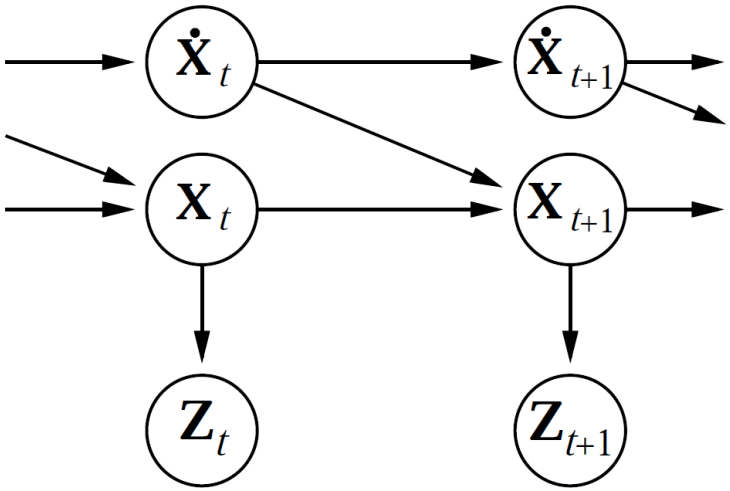
$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top} \mathbf{f}_{1:t} \\
\alpha' (\mathbf{T}^{\top})^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

- 算法：前向传递计算 $\mathbf{f}_t$，反向传递 $\mathbf{f}_i, \mathbf{b}_i$



卡尔曼滤波 (Kalman filters)

- 用于建模连续变量描述的系统
  - E.g., 跟踪鸟的飞行 $\mathbf{X}_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$
  - 飞机、机器人、生态系统、经济、星球……



- 假设：高斯先验、线性高斯的转移模型和传感器模型

更新高斯分布

- 单步预测：如果 $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ 是高斯分布，那么预测

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) = \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t})\, d\mathbf{x}_t$$

也是高斯分布。如果 $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$ 是高斯分布，那么更新后的分布

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

也是高斯分布

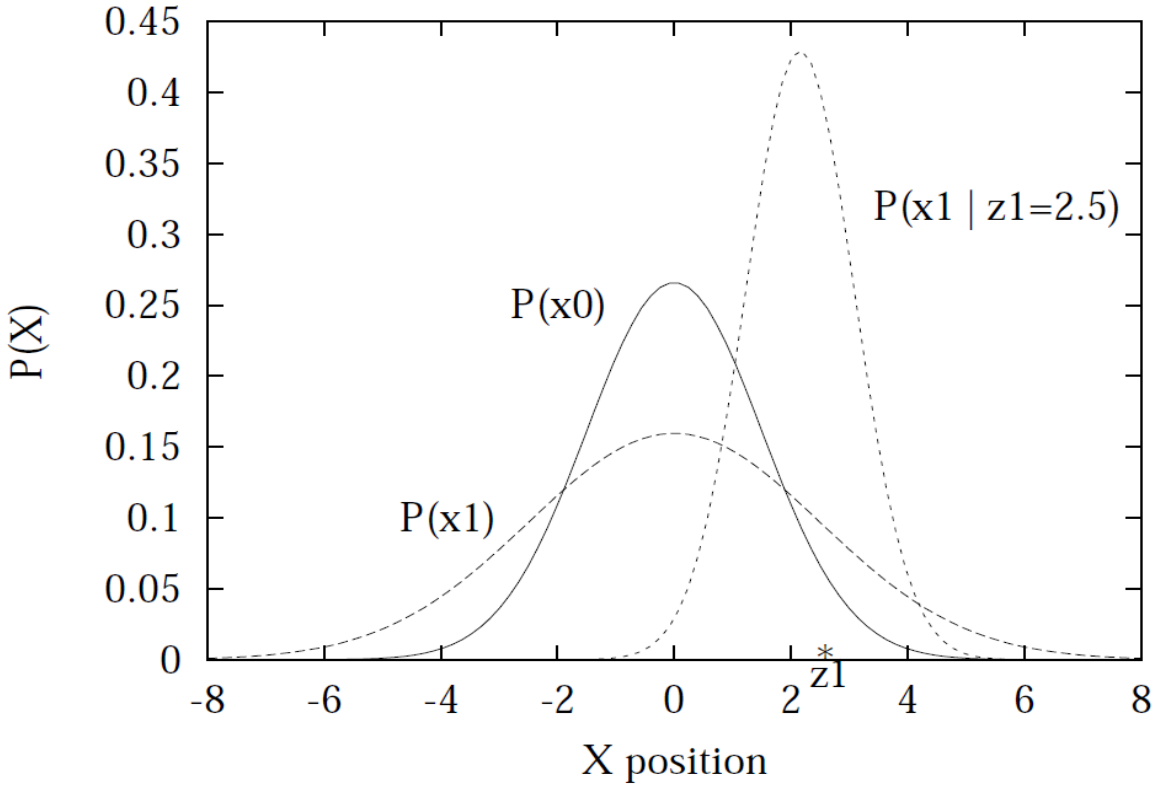
⟹ $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ 是一个多变量高斯分布 $N(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ for all t

- 通用（非线性，非高斯）过程：后验描述随着 $t \to \infty$ 将是无穷无尽 (unboundedly) 地增长

简单的 1-D 示例

- 在 X 轴上的高斯随机游走，转移方差为 σ_x，传感方差为 σ_z

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)z_{t+1} + \sigma_z^2\mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \qquad \sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}$$



一般的 Kalman 更新

- 转移和传感模型：

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t) = N(\mathbf{F}\mathbf{x}_t, \boldsymbol{\Sigma}_x)(\mathbf{x}_{t+1})$$
$$P(\mathbf{z}_t|\mathbf{x}_t) = N(\mathbf{H}\mathbf{x}_t, \boldsymbol{\Sigma}_z)(\mathbf{z}_t)$$

$\mathbf{F}$ 是转移矩阵，$\boldsymbol{\Sigma}_x$ 是转移噪声协方差

$\mathbf{H}$ 是传感矩阵，$\boldsymbol{\Sigma}_z$ 是传感器噪声协方差

- 滤波器采用的更新公式为

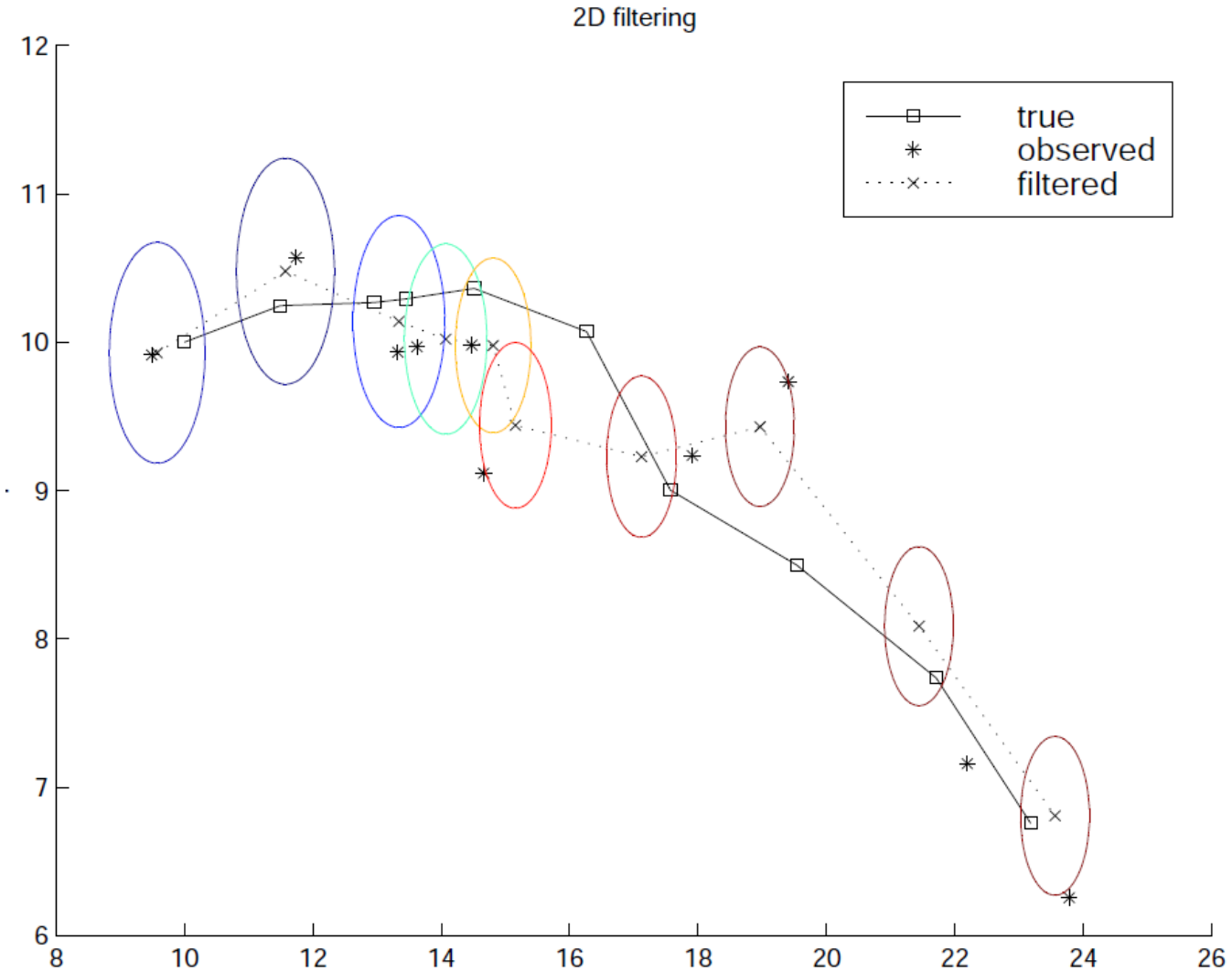
$$\boldsymbol{\mu}_{t+1} = \mathbf{F}\boldsymbol{\mu}_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\boldsymbol{\mu}_t)$$
$$\boldsymbol{\Sigma}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1})(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)$$

这里的 $\mathbf{K}_{t+1} = (\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top + \boldsymbol{\Sigma}_z)^{-1}$

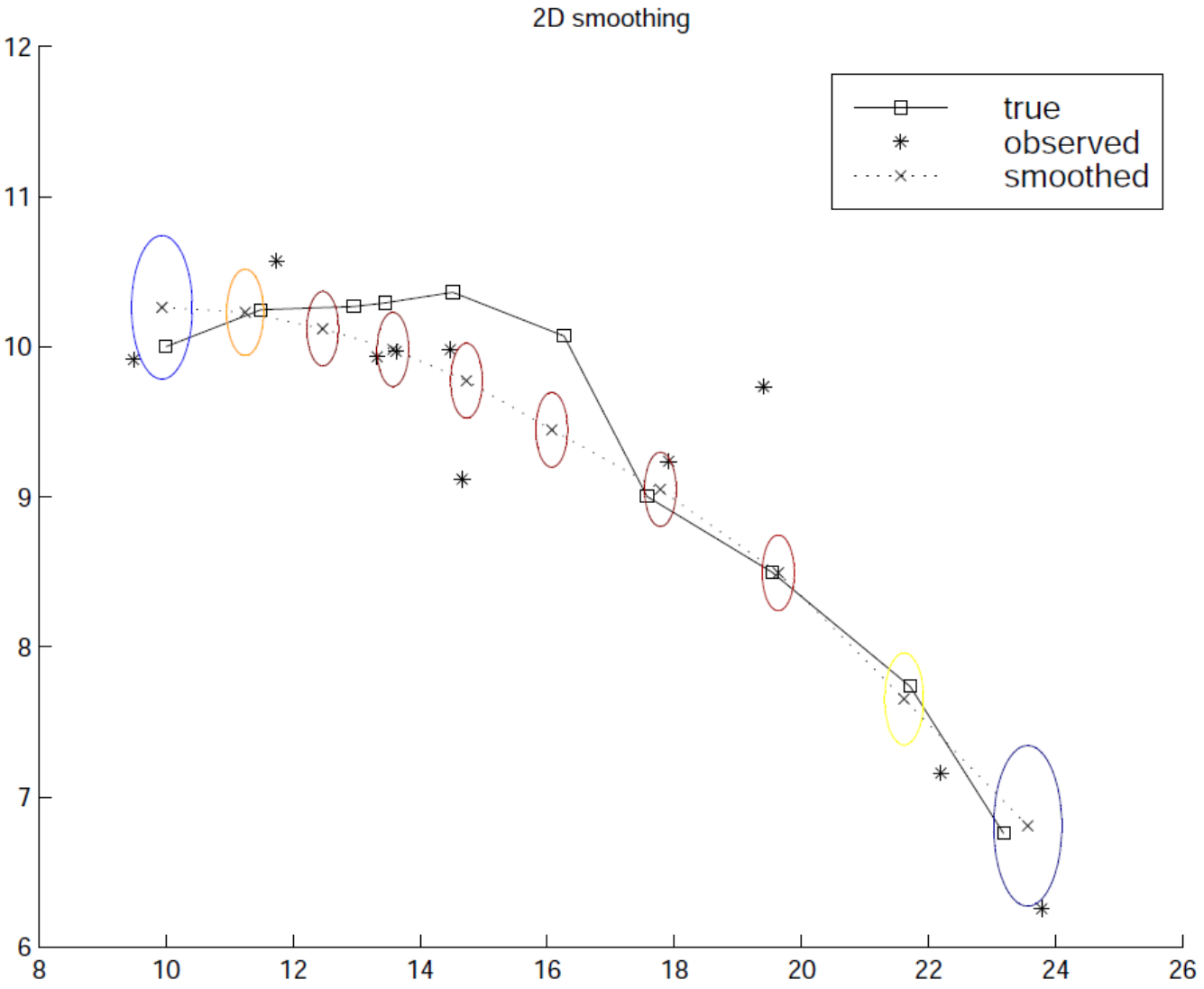
称作是卡尔曼增益矩阵 (Kalman gain matrix)

- $\boldsymbol{\Sigma}_t$ 和 $\mathbf{K}_t$ 是独立于观测的，可以离线计算

2-D 跟踪示例：滤波

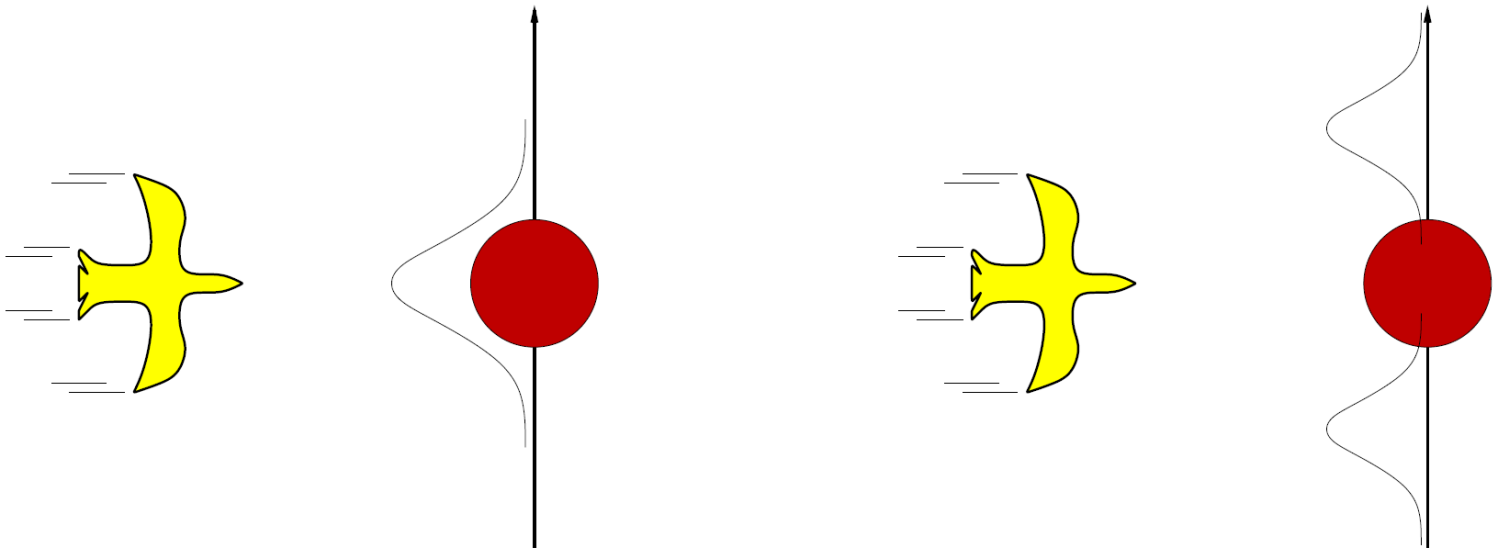

2D filtering
true
observed
filtered
12
11
10
9
8
7
6
8
10
12
14
16
18
20
22
24
26

2-D 跟踪示例：平滑


2D smoothing
true
observed
smoothed
12
11
10
9
8
7
6
8
10
12
14
16
18
20
22
24
26

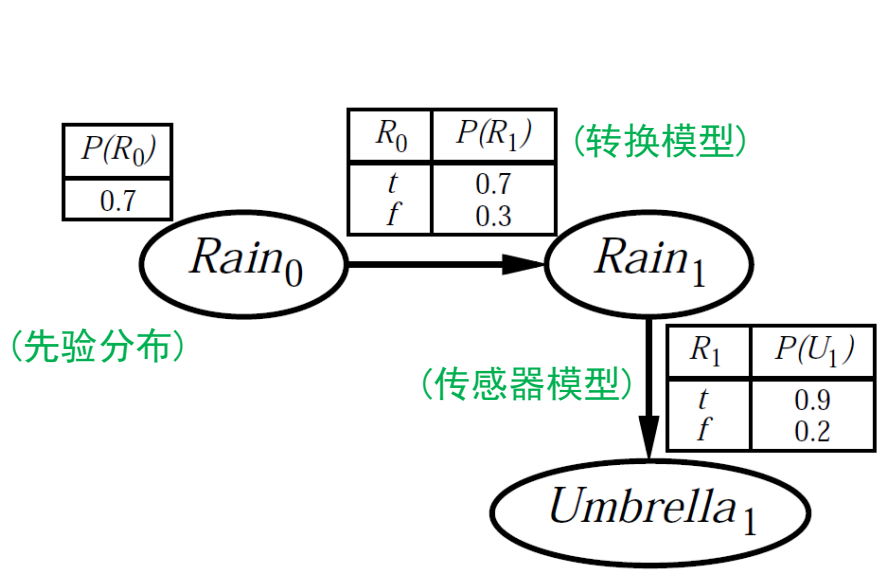
Where it breaks

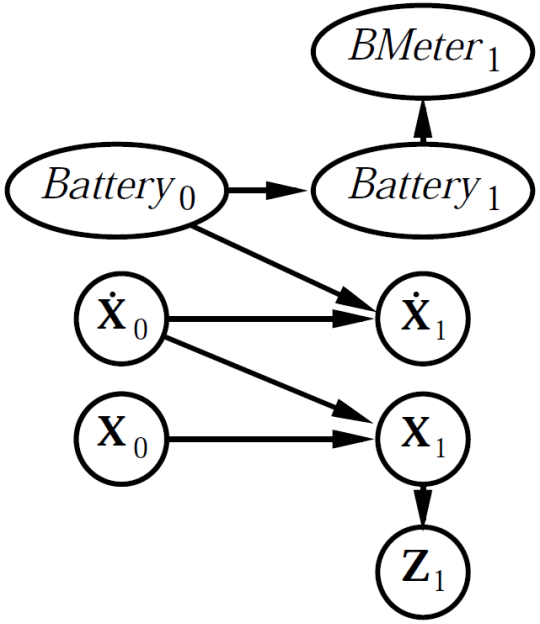
- 如果转移模型是非线性的，Kalman 滤波是不适用的
- 扩展的卡尔曼滤波器 (Extended Kalman Filter)
  - 转移模型在 $\mathbf{x}_t = \boldsymbol{\mu}_t$ 周边是局部线性的 (Locally linear)
  - 对局部不平滑的系统，难以处理



动态贝叶斯网络 (Dynamic Bayesian Networks, DBN)

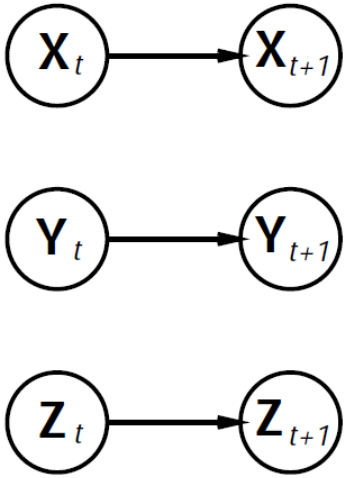
- 在一个可复制的贝叶斯网络中，$\mathbf{X}_t, \mathbf{E}_t$ 有任意数量的状态变量

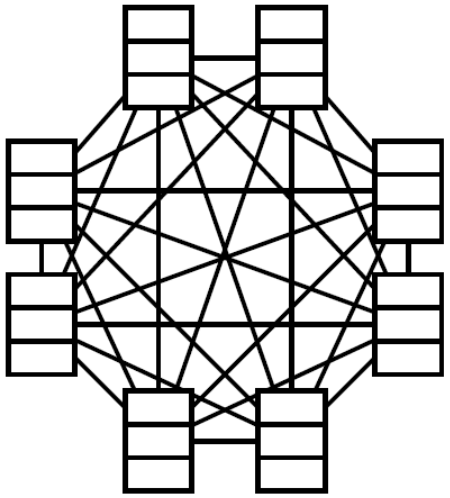




DBNs vs. HMMs

- 每个 HMM 是一个单变量的 DBN，每个离散的 DBN 是一个 HMM



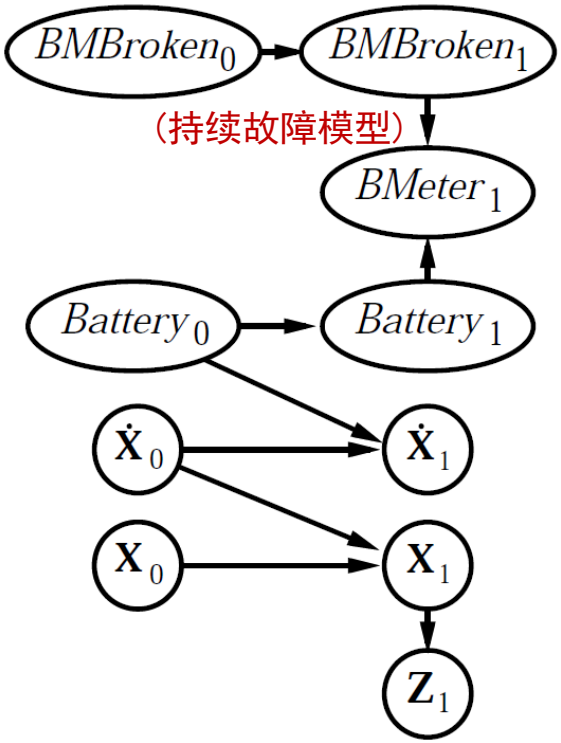


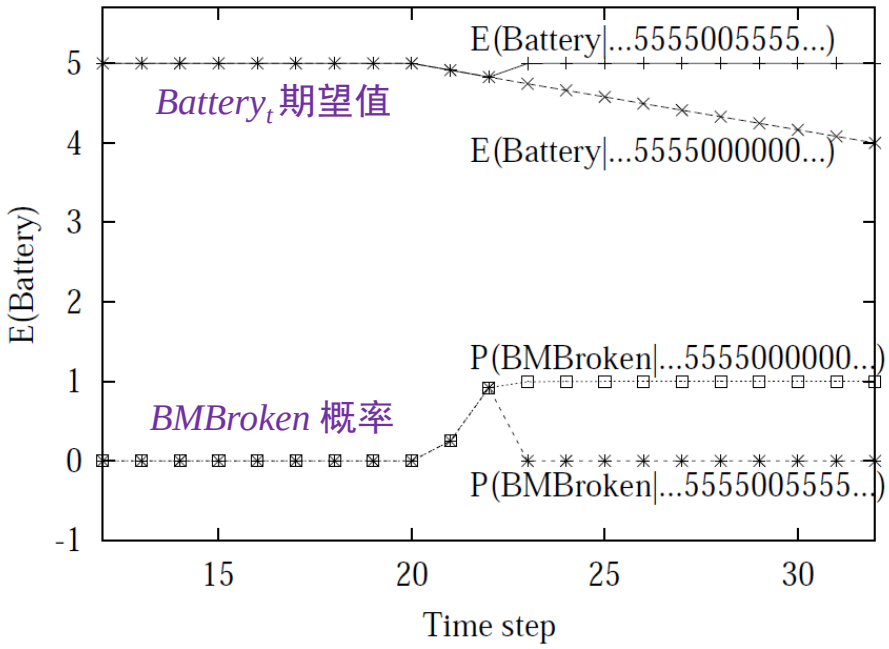
- 稀疏依赖性 (Sparse dependences) ➔ 参数量指数级减少
  - E.g., 20 个布尔状态变量，每个有 3 个父节点
  - DBN：$20 \times 2^3 = 160$ 个参数，HMM：$2^{20} \times 2^{20} \approx 10^{12}$ 个参数

➡ DBN 能够充分利用时序概率模型中的稀疏性

DBNs vs. Kalman filters

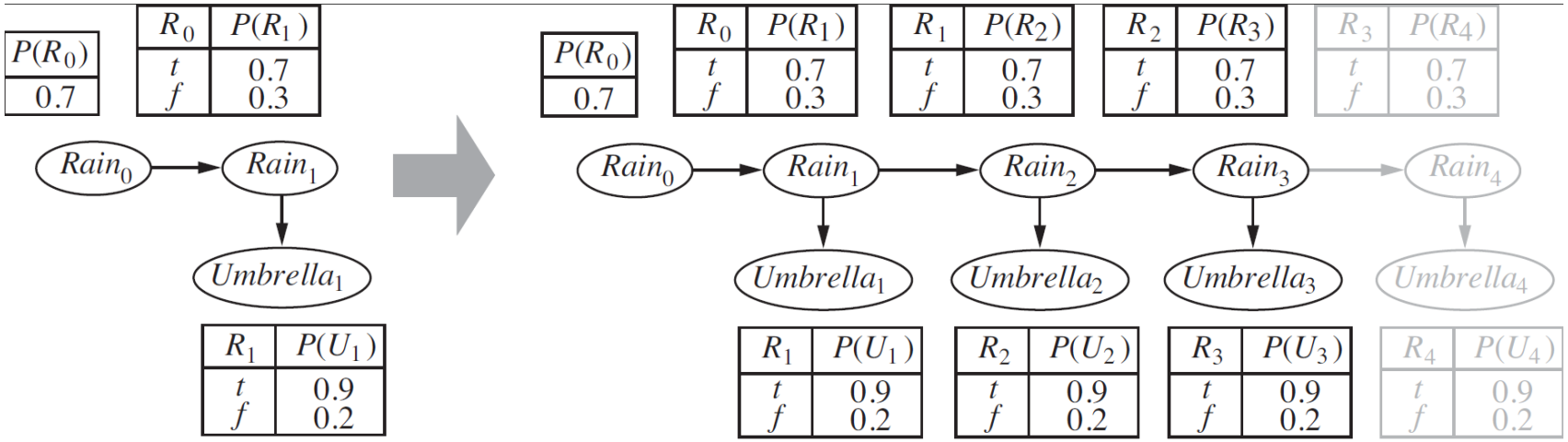
- 每个卡尔曼滤波器 (KF) 是一个 DBN，但很少 DBN 是 KF
  - 很多真实问题通常需要的是非高斯后验概率
  - E.g.，钥匙放置位置问题





DBNs 中的精确推理

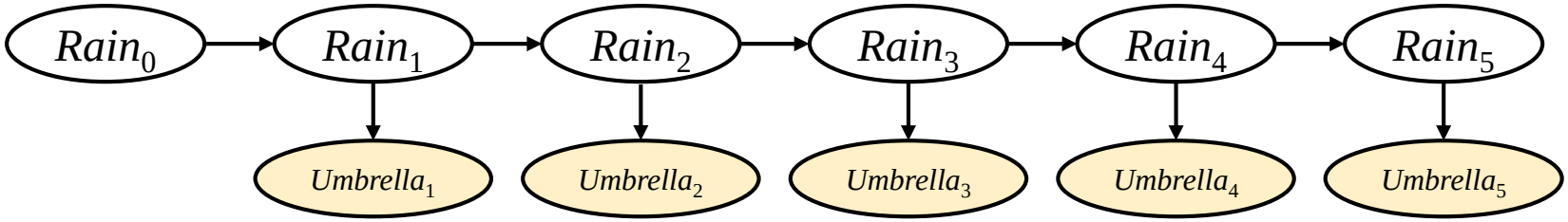
- 朴素方法：摊开 (unrolling) 网络，然后运行任一的精确推理算法



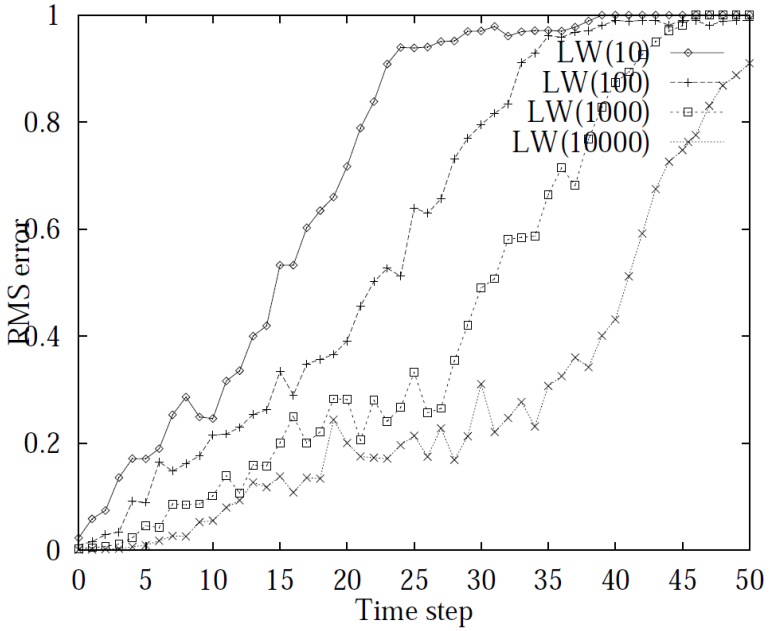
- 问题：每次更新的推理代价随着 t 的变化而提高
- 回卷滤波 (Rollup filtering)：添加 t+1 时间段，用变量消元法对 t 时间段进行求和
  - 最大因子为 $O(d^{n+1})$，更新代价为 $O(d^{n+2})$　　(HMM更新代价为 $O(d^{2n})$)

DBNs 的似然加权

- 加权样本集近似于置信状态

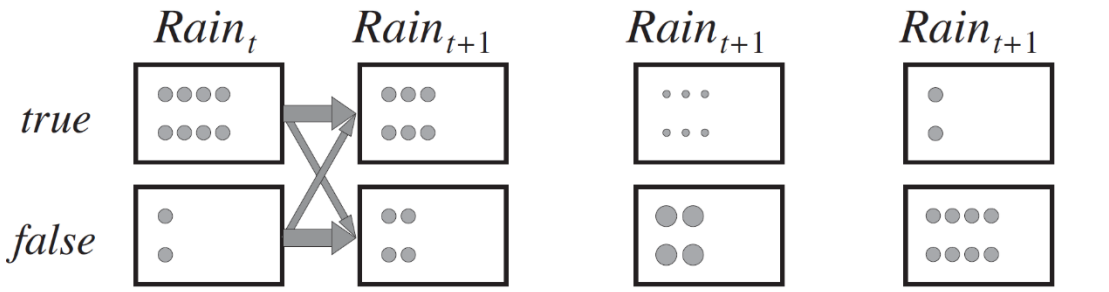


- LW 采样不关注证据！
  - ⇨ “一致”样本的比值随着 t 呈指数级下降
  - ⇨ 要求的样本数随着 t 呈指数级上升



粒子滤波 (Particle filtering)

- 基本思想：确保跟踪状态空间中高似然区域的样本 (“粒子”) 数量
- 复制粒子：正比于对 $\mathbf{e}_t$ 的似然度



- 应用
  - 广泛应用于非线性系统的跟踪，尤其是在视觉中
  - 也应用于移动机器人的 SLAM (simultaneous localization and mapping)

粒子滤波 (Particle filtering)

- 每个时间步的更新循环：
  - 对于每个样本， 通过转移模型 $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t)$，在给定当前状态值 $\mathbf{x}_t$ 的条件下对下一个状态进行采样，进行样本的前向传播
  - 对于每个样本，用它与新证据的似然值 $P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1})$ 作为权值
  - 对样本进行重采样以生成一个新的 N 样本总体，每个新样本从当前总体样本中进行选择，选中概率与权值成正比

```
function PARTICLE-FILTERING(e, N, dbn) returns a set of samples for the next time step
  inputs: e, the new incoming evidence
          N, the number of samples to be maintained
          dbn, a DBN with prior P(X_0), transition model P(X_1|X_0), sensor model P(E_1|X_1)
  persistent: S, a vector of samples of size N, initially generated from P(X_0)
  local variables: W, a vector of weights of size N

  for i = 1 to N do
      S[i] ← sample from P(X_1 | X_0 = S[i])    /* step 1 */
      W[i] ← P(e | X_1 = S[i])                  /* step 2 */
  S ← WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S, W)        /* step 3 */
  return S
```

粒子滤波 (Particle filtering)

- 假设在时间 t 是一致的：$N(\mathbf{x}_t|\mathbf{e}_{1:t})/N = P(\mathbf{x}_t|\mathbf{e}_{1:t})$
- 前向传递：$\mathbf{x}_{t+1}$的数量是

$$N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) = \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) N(\mathbf{x}_t|\mathbf{e}_{1:t})$$

- 根据对 $\mathbf{e}_{t+1}$ 的似然计算采样权重：

$$W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t})$$

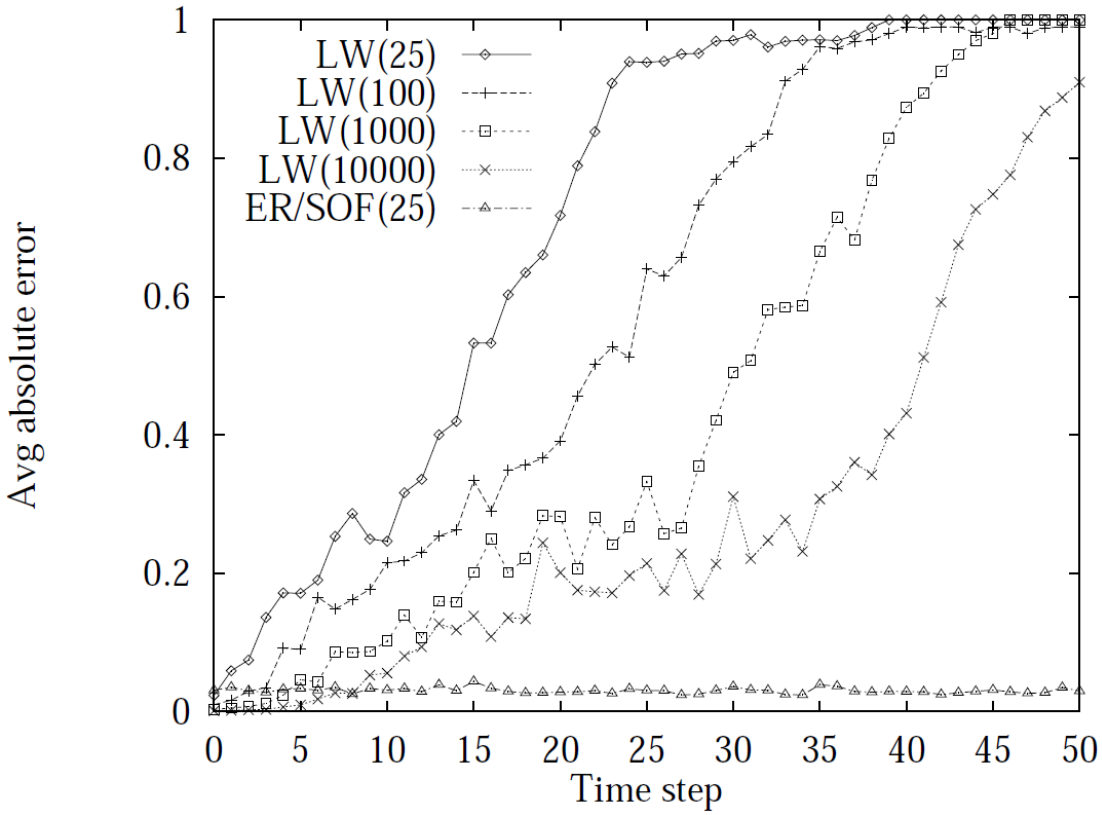
- 重采样获取正比于 W 的样本数：

$$\begin{aligned} N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1})/N &= \alpha W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) \\ &= \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) N(\mathbf{x}_t|\mathbf{e}_{1:t}) \\ &= \alpha' P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t}) \\ &= P(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) \end{aligned}$$

粒子滤波能够通过常数数目的样本保持实现对真实后验概率的良好近似

粒子滤波性能

- 粒子滤波的近似在时间上维持一个有界误差
  - 经验-理论分析是困难的



总结

- 时间模型在时间上复制状态和传感器变量
- Markov 假设和稳态假设
  - 转移模型 $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$
  - 传感器模型 $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$
- 推理任务有滤波、预测、平滑和最可能序列
- HMM 是单个离散状态变量 (如用于语音识别中)
- Kalman 滤波器允许有 n 个状态变量，但要求是线性高斯的
- DBNs 涵盖了 HMM、Kalman滤波器等，但其精确更新是不可追踪的
- 粒子滤波是 DBNs 的一个很好的近似算法

谢谢聆听！