Artificial Intelligence

8.d.不确定知识与推理

对应课本13-15章

时间概率模型

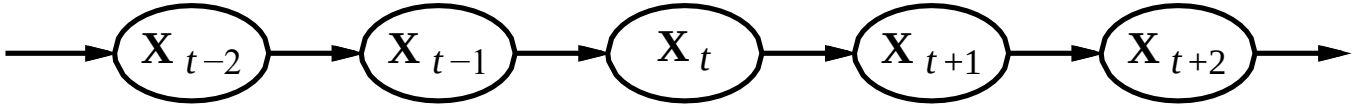
- 时间与不确定性
- 推理：过滤、预测、平滑
- 隐马尔可夫模型
- 卡尔曼过滤器
- 动态贝叶斯网络
- 粒子滤波

时间和不确定性

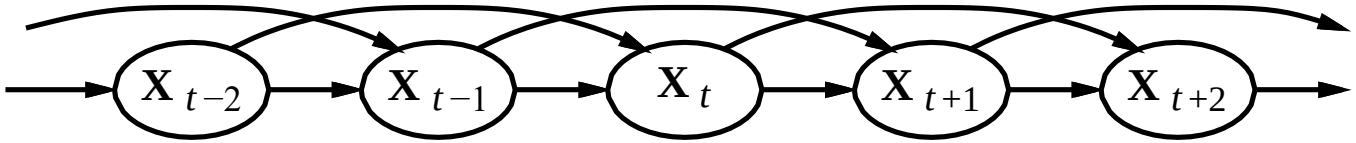
- 世界在随时间变化，如何跟踪和预测?
- 考虑糖尿病和车辆诊断两种情况
- 基本思想：在每个time step内记录状态和证据变量的值
- $\mathbf{X}_t$ = t 时刻不可观测的状态变量的集合，
  - 如：血糖含量、胃里的东西
- $\mathbf{E}_t$ = t 时刻可观测的证据变量的集合，
  - 如：血糖测量值、脉搏、吃下的食物
- 这样做实际上假设了时间是离散值（是吗？），根据问题的具体情况选择合适的时间步长
- 得到变量： $\mathbf{X}_{a:b} = \mathbf{X}_a, \mathbf{X}_{a+1}, \ldots, \mathbf{X}_{b-1}, \mathbf{X}_b$

马尔科夫过程（马尔科夫链）

- 考虑之前学过的贝叶斯网络，是否能够根据这些变量构造一个？但是贝叶斯网络下的父子关系如何确定？
- 马尔科夫假设：X_t 依赖于$X_{0:t-1}$的有界子集
- 一阶马尔可夫过程：$P(X_t|X_{0:t-1}) = P(X_t|X_{t-1})$



- 二阶马尔可夫过程：$P(X_t|X_{0:t-1}) = P(X_t|X_{t-2}, X_{t-1})$

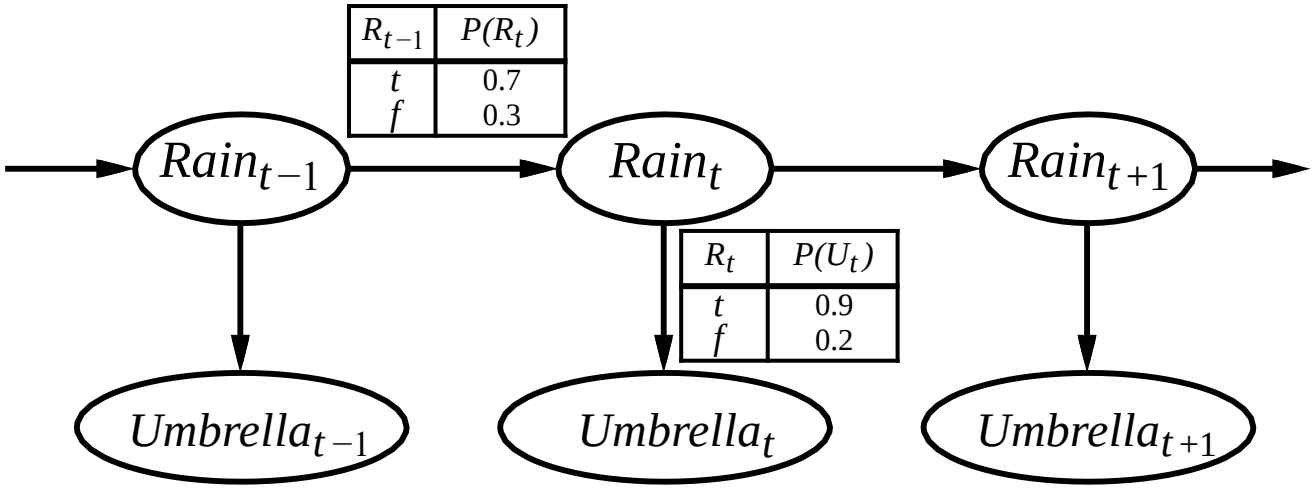


- 感知马尔科夫假设（Sensor Markov assumption）：

$$P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t)$$

- 平稳过程：转移模型$P(X_t|X_{t-1})$和感知模型$P(E_t|X_t)$对所有t固定

实例



- 一阶马尔科夫过程假设真实世界中不存在概率为1.0的事件
- 修正可能性：
  - 对马尔科夫过程升阶
  - 增加状态，如$Temp_t$, $Pressure_t$
- 例子：对于机器人的马达，如要计算位置和速度，可以增加状态时间t时候的电量$Battery_t$

推理任务

- 过滤任务filtering：$P(X_t|e_{1:t})$
  - 信念状态：输入理性主体的决策过程
- 预测任务prediction：$P(X_{t+k}|e_{1:t})$ for $k>0$
  - 对可能的动作序列进行评估
  - 形式上类似在没有证据的条件下进行过滤任务
- 平滑任务smoothing：$P(X_k|e_{1:t})$ for $0 \leq k < t$
  - 更好地估计过去的状态，对于学习来说很重要
- 序列预测Most likely explanation：$\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$
  - 语音识别、声道中去除噪音

过滤任务

- 目的：设计递归状态估计算法：

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

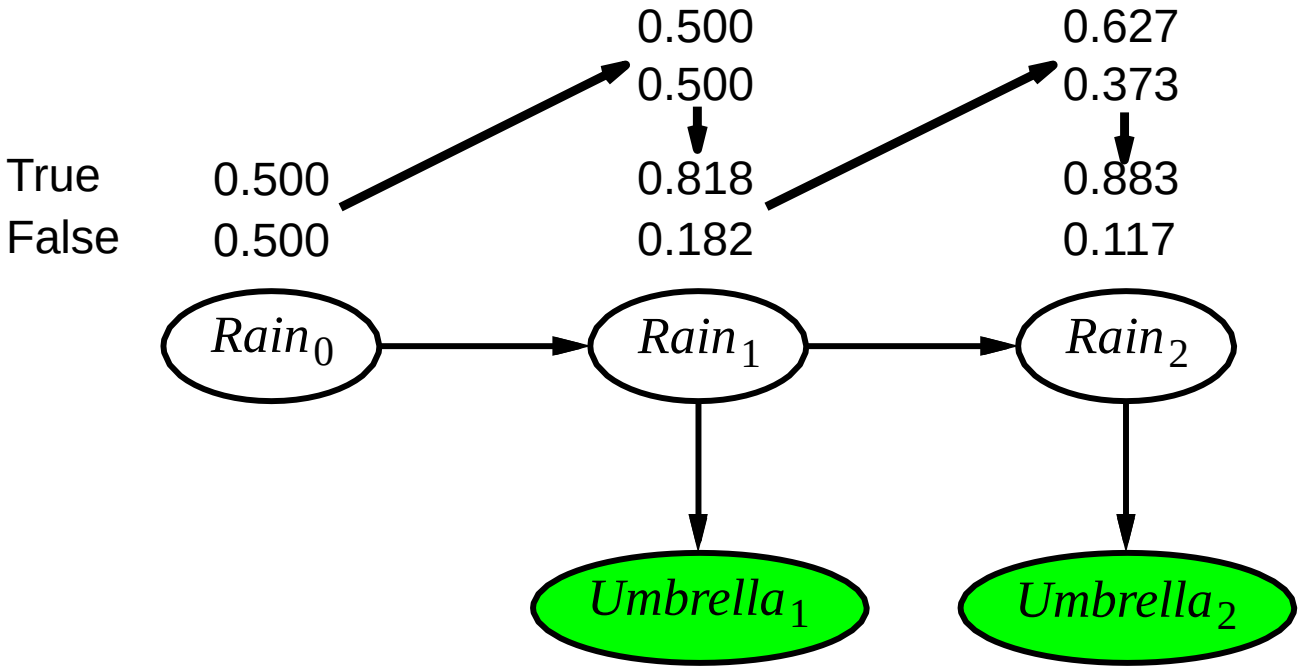
$$\begin{aligned}\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{e}_{1:t+1}) &= \mathrm{P}(\mathrm{X}_{t+1}|\mathrm{e}_{1:t}, \mathrm{e}_{t+1}) \\ &= \alpha\mathrm{P}(\mathrm{e}_{t+1}|\mathrm{X}_{t+1}, \mathrm{e}_{1:t})\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{e}_{1:t}) \\ &= \alpha\mathrm{P}(\mathrm{e}_{t+1}|\mathrm{X}_{t+1})\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{e}_{1:t})\end{aligned}$$

- 也即预测+估计，通过对X_t求和进行预测

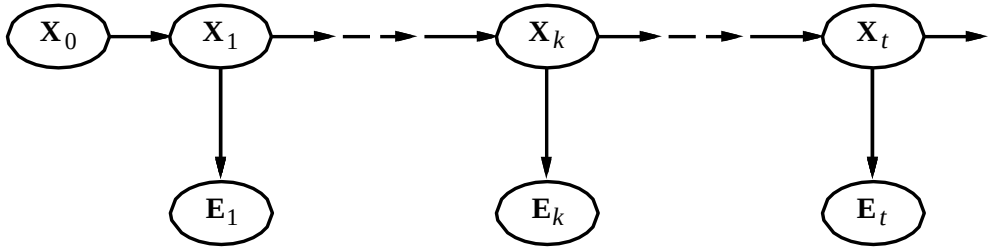
$$\begin{aligned}\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{e}_{1:t+1}) &= \alpha\mathrm{P}(\mathrm{e}_{t+1}|\mathrm{X}_{t+1})\Sigma_{\mathrm{x}_t}\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{x}_t, \mathrm{e}_{1:t})P(\mathrm{x}_t|\mathrm{e}_{1:t}) \\ &= \alpha\mathrm{P}(\mathrm{e}_{t+1}|\mathrm{X}_{t+1})\Sigma_{\mathrm{x}_t}\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{x}_t)P(\mathrm{x}_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

- 当 $\mathrm{f}_{1:t} = \mathrm{P}(\mathrm{X}_t|\mathrm{e}_{1:t})$ 时 $\mathrm{f}_{1:t+1} = \mathrm{Forward}(\mathrm{f}_{1:t}, \mathrm{e}_{t+1})$
  - 注意这里时间和空间都是常数（与 t 独立）

过滤任务实例


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

平滑任务



- 将证据变量$e_{1:t}$分为两部分$e_{1:k}$, $e_{k+1:t}$，则有：

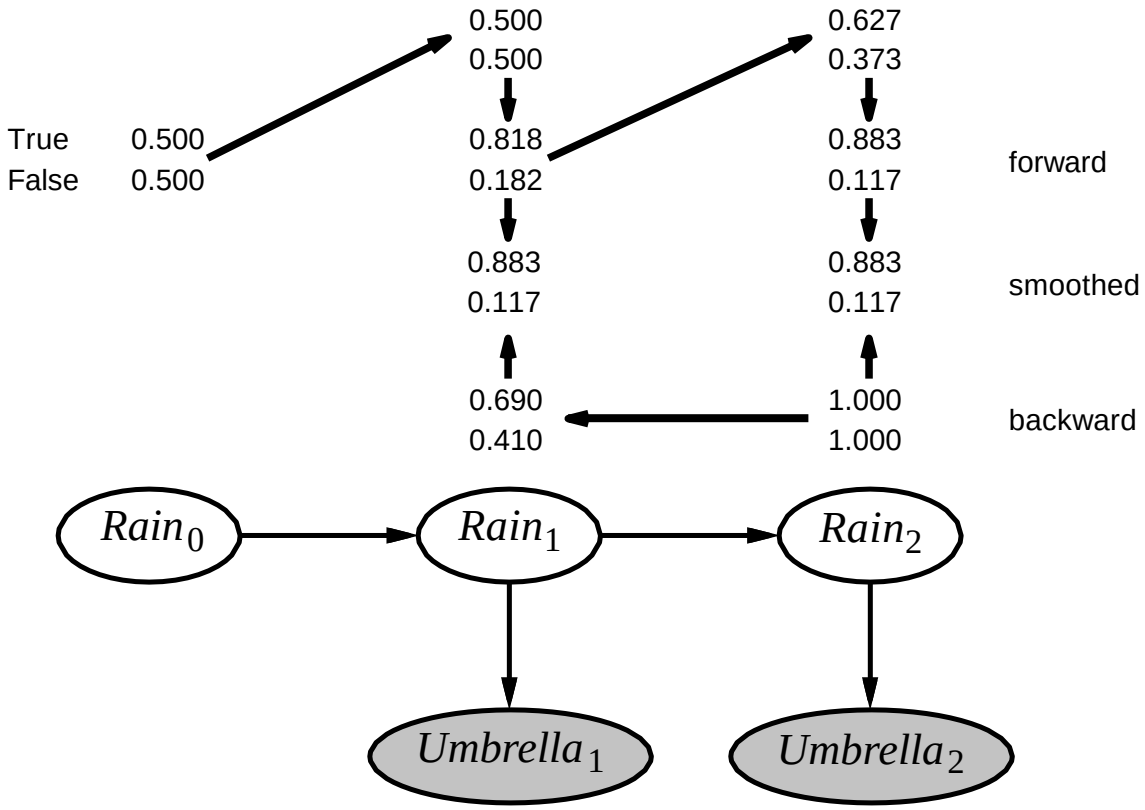
$$\begin{aligned} P(X_k|e_{1:t}) &= P(X_k|e_{1:k}, e_{k+1:t}) \\ &= \alpha P(X_k|e_{1:k})P(e_{k+1:t}|X_k, e_{1:k}) \\ &= \alpha P(X_k|e_{1:k})P(e_{k+1:t}|X_k) \\ &= \alpha f_{1:k} b_{k+1:t} \end{aligned}$$

- 通过后向递归计算得到后向消息：

$$\begin{aligned} P(e_{k+1:t}|X_k) &= \sum_{x_{k+1}} P(e_{k+1:t}|X_k, x_{k+1}) P(x_{k+1}|X_k) \\ &= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1}) P(x_{k+1}|X_k) \\ &= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1}) P(e_{k+2:t}|x_{k+1}) P(x_{k+1}|X_k) \end{aligned}$$

平滑任务例子

- 前后向算法Forward–backward algorithm：
  - 在路径上对消息进行缓存
  - 时间对 t 线性(polytree推理)、空间复杂度$O(t\,|\mathrm{f}|)$



序列预测

- 可能性序列≠一系列可能的状态
- 对于每个x_{t+1}可能性序列 = 对x_t增加新的一步的可能性序列

$$\max_{x1...xt} P(x_1, \ldots, x_t, X_{t+1}|e_{1:t+1})$$

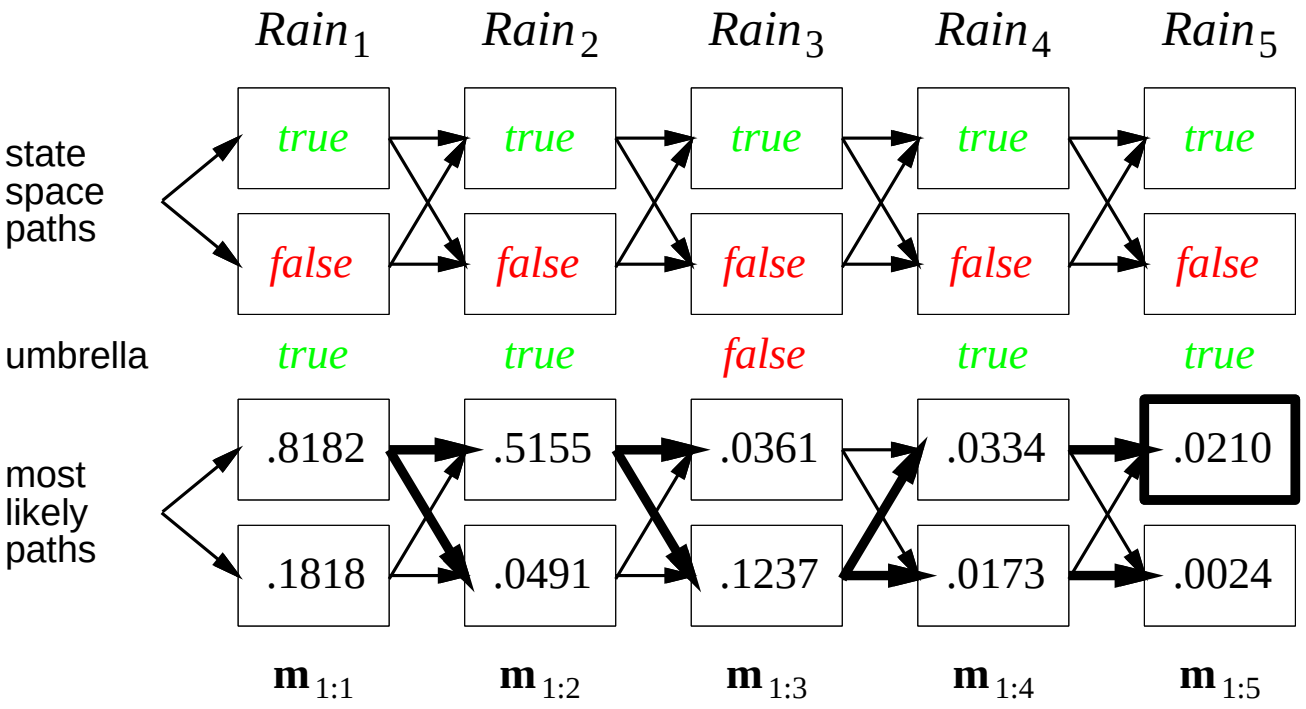
- 与滤波相同，除了$f_{1:t}$被替换为

$$m_{1:t} = \max_{x1...xt-1} P(x_1, \ldots, x_{t-1}, X_t|e_{1:t})$$

- 也就是说，$m_{1:t}(i)$给出状态 i 的最可能路径的概率。
- 更新将sum操作替换为max，给出Viterbi算法：

$$m_{1:t+1} = P(e_{t+1}|X_{t+1}) \max_{xt} (P(X_{t+1}|x_t)m_{1:t})$$

Viterbi算法实例



隐马尔科夫模型

- X_t是一个单一且离散的变量，其定义域为$\{1, \ldots, S\}$
- 转移矩阵 $\mathrm{T} = P(X = j \mid X = i)$，如$\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$
- 感知矩阵O_t对每一步来说，对角元素为$P\,(e_t \mid X_t = i)$
  - 比如如果$U_1 = \ true$，有$\mathrm{O}1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$
- 作为列向量的前向和后向消息

$$\mathrm{f}_{1:t+1} = \ \alpha \mathrm{O}_{t+1} \mathrm{T}^{\mathsf{T}} \mathrm{f}_{1:t}$$

$$\mathrm{b}_{k+1:t} = \ \mathrm{T}\mathrm{O}_{k+1} \mathrm{b}_{k+2:t}$$

- 正反向算法需要时间$O(S^2 t)$，需要空间$O(St)$

乡村舞蹈算法

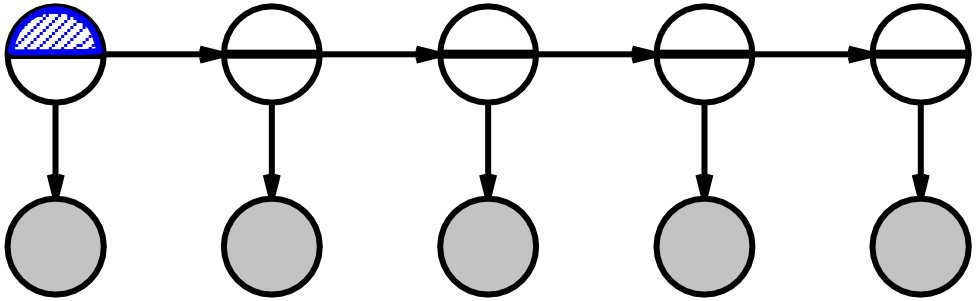
• 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

• $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$

• $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$

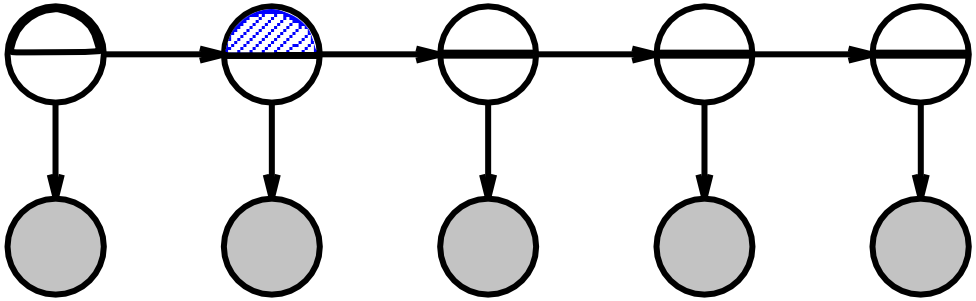
• $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

• 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



乡村舞蹈算法

- 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：
- $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$
- $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$
- $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$
- 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



乡村舞蹈算法

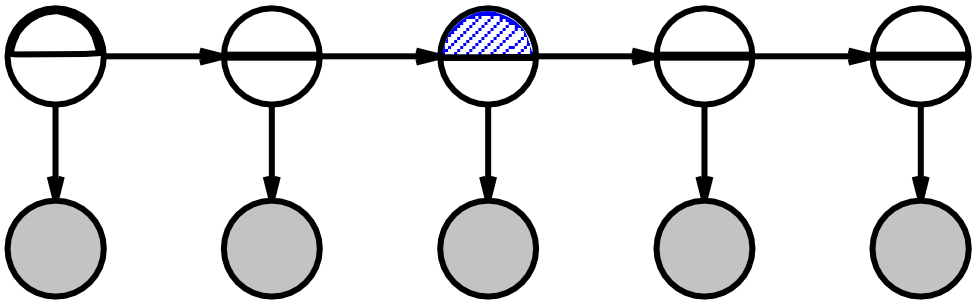
• 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

• $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$

• $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$

• $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

• 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



乡村舞蹈算法

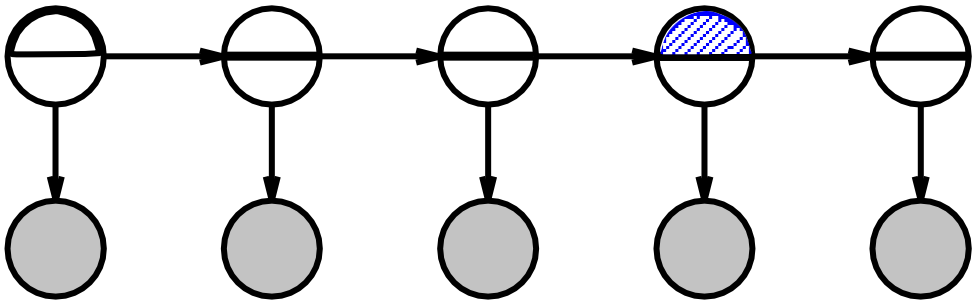
• 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

• $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$

• $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$

• $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

• 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



乡村舞蹈算法

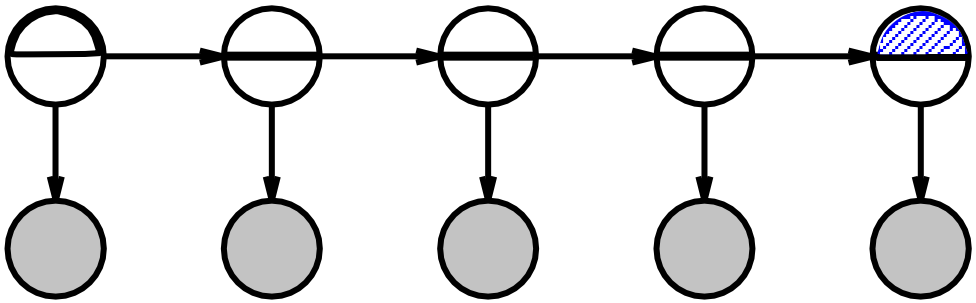
• 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

• $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$

• $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$

• $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

• 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$

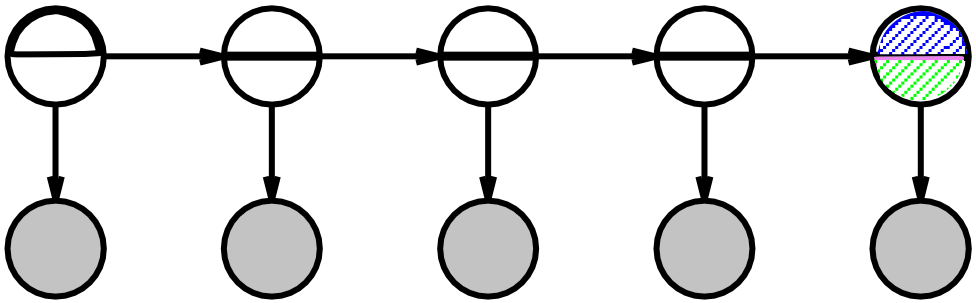


乡村舞蹈算法

- 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

- $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$
- $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$
- $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

- 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



乡村舞蹈算法

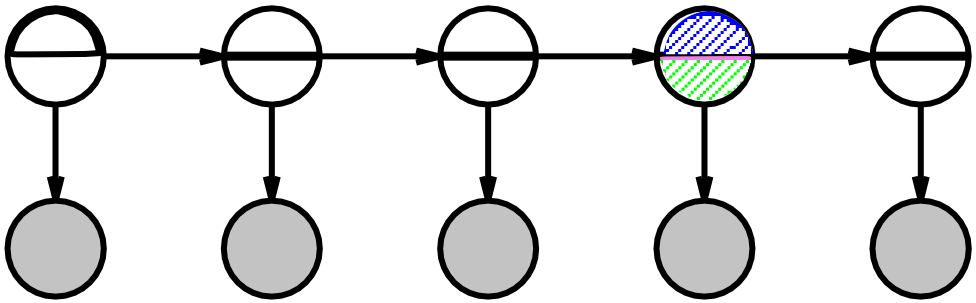
• 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

• $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$

• $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$

• $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

• 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



乡村舞蹈算法

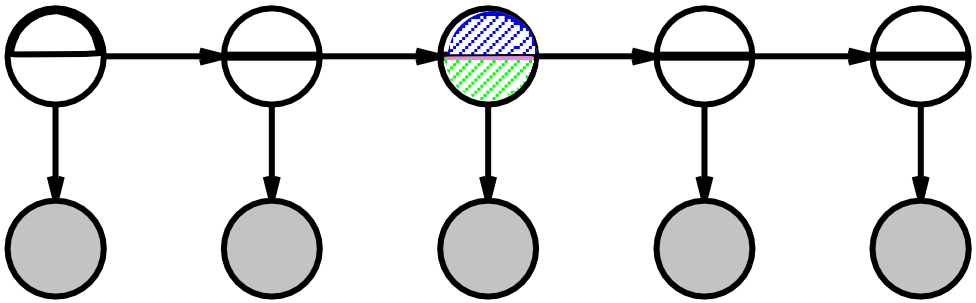
• 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

• $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$

• $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$

• $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

• 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



乡村舞蹈算法

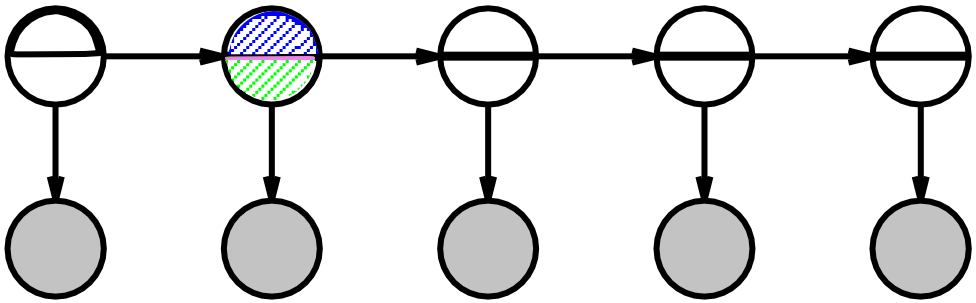
• 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：

• $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$

• $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$

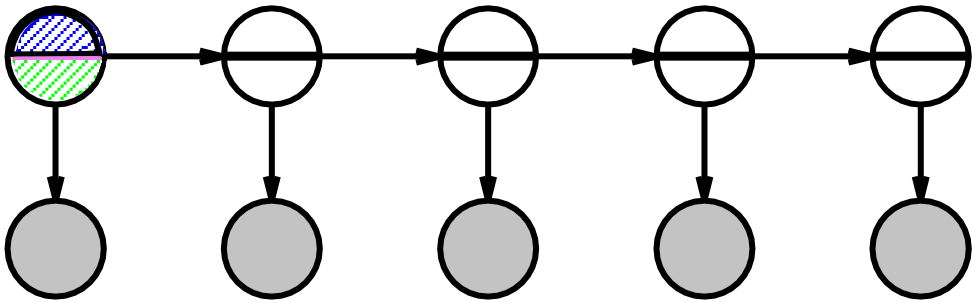
• $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$

• 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



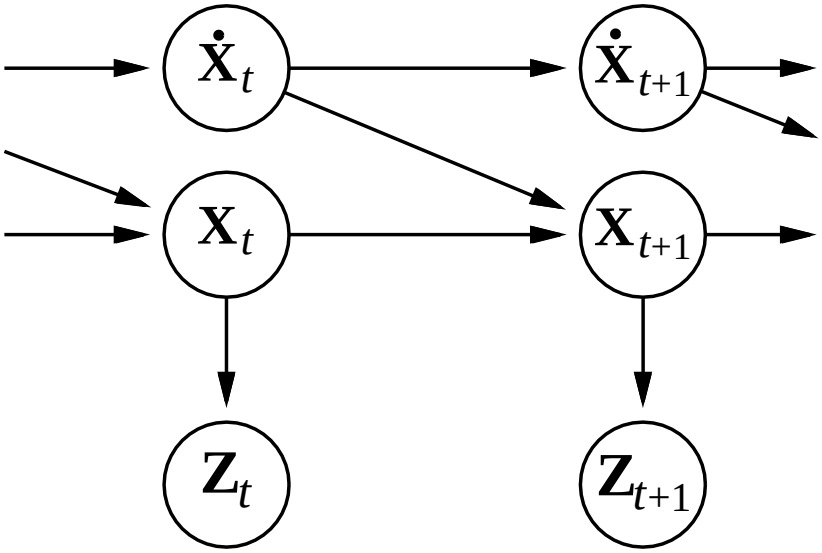
乡村舞蹈算法

- 通过向后运行前向算法，可以避免将所有前向消息在平滑任务中存储：
- $$f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$$
- $$O_{t+1}^1 f_{1:t+1} = \alpha T^T f_{1:t}$$
- $$\alpha^t (T^T)^{-1} O_{t+1}^{-1} f_{1:t+1} = f_{1:t}$$
- 算法：向前传计算的f_t，向后传$\mathrm{f}_i, \mathrm{b}_i$



卡尔曼Kalman滤波

- 考虑对基于一组连续变量描述的系统建模，例如追踪一只飞行的小鸟
  - $X_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$
  - 也可以是飞机，机器人，生态系统，经济，化工厂，行星...



- 高斯先验概率，线性高斯转移模型和感知模型

更新高斯分布

- 预测步骤：如果$\mathrm{P}(\mathrm{X}_t|\mathrm{e}_{1:t})$满足高斯分布，那么预测

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

- 也满足高斯分布，如果是高斯分布，那么更新满足的分布

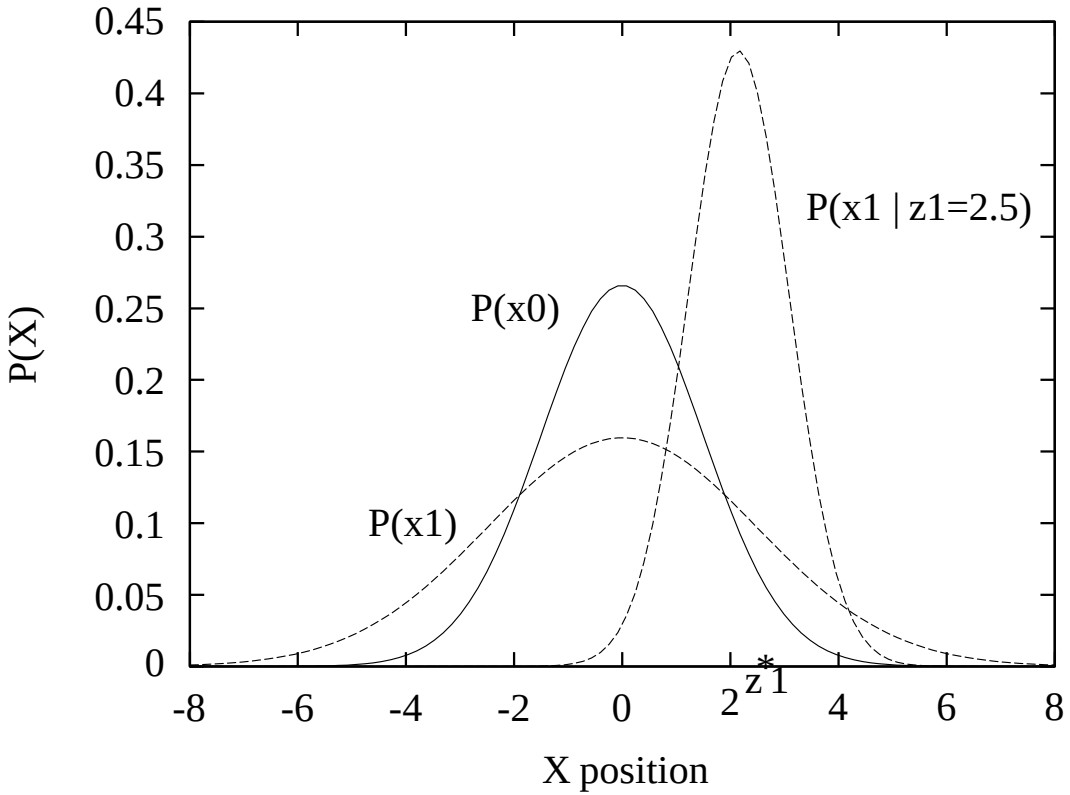
$$\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{e}_{1:t+1}) = \alpha\mathrm{P}(\mathrm{e}_{t+1}|\mathrm{X}_{t+1})\mathrm{P}(\mathrm{X}_{t+1}|\mathrm{e}_{1:t})$$

- 也满足高斯分布
- 对于任意的 t 来说$\mathrm{P}(\mathrm{X}_t|\mathrm{e}_{1:t})$都是$N(\mu_t, \Sigma_t)$的多元高斯分布
- 通常的处理过程是，对$t \to \infty$这样无限增长的后验分布进行描述

简单的一维分布

- 在X轴上实现高斯随机漫步，记为σ_x
- 相应的记感知模型为σ_z

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



通用卡尔曼滤波的更新过程

- 转移和感知模型：

$$P(\mathrm{x}_{t+1}|\mathrm{x}_t) = N(\mathrm{F}\mathrm{x}_t, \Sigma_x)(\mathrm{x}_{t+1})$$
$$P(\mathrm{z}_t|\mathrm{x}_t) = N(\mathrm{H}\mathrm{x}_t, \Sigma_z)(\mathrm{z}_t)$$

- F 是转移模型的矩阵，Σ_x 是转移模型的噪声协方差
- H 是感知模型的矩阵，Σ_z 是感知模型的噪声协方差
- 过滤任务的计算过程遵循以下更新：

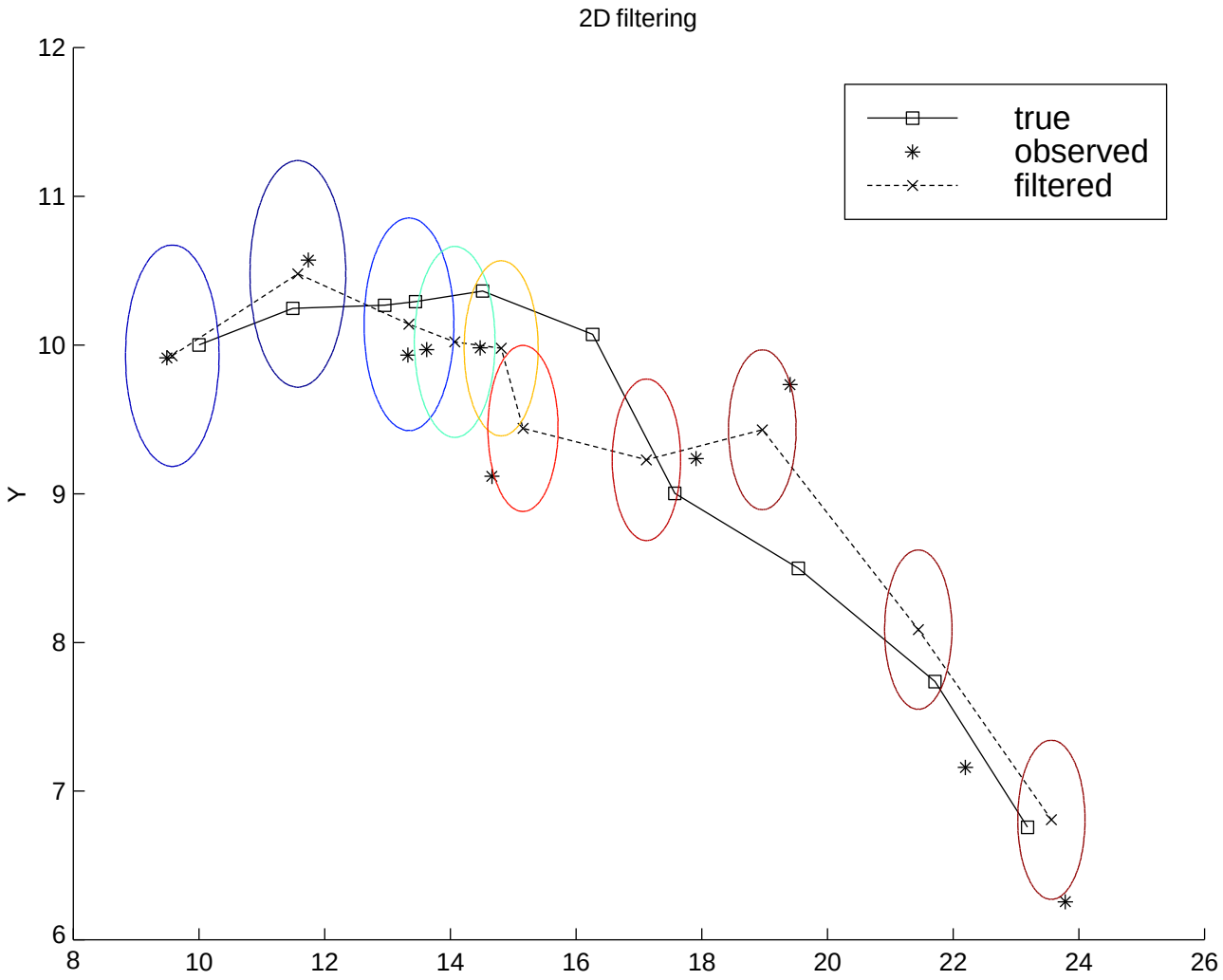
$$\mu_{t+1} = \mathrm{F}\mu_t + \mathrm{K}_{t+1}(\mathrm{z}_{t+1} - \mathrm{H}\mathrm{F}\mu_t)$$
$$\Sigma_{t+1} = (\mathrm{I} - \mathrm{K}_{t+1})(\mathrm{F}\Sigma_t\mathrm{F}^\mathsf{T} + \Sigma_x)$$

- 卡尔曼滤波增益矩阵：

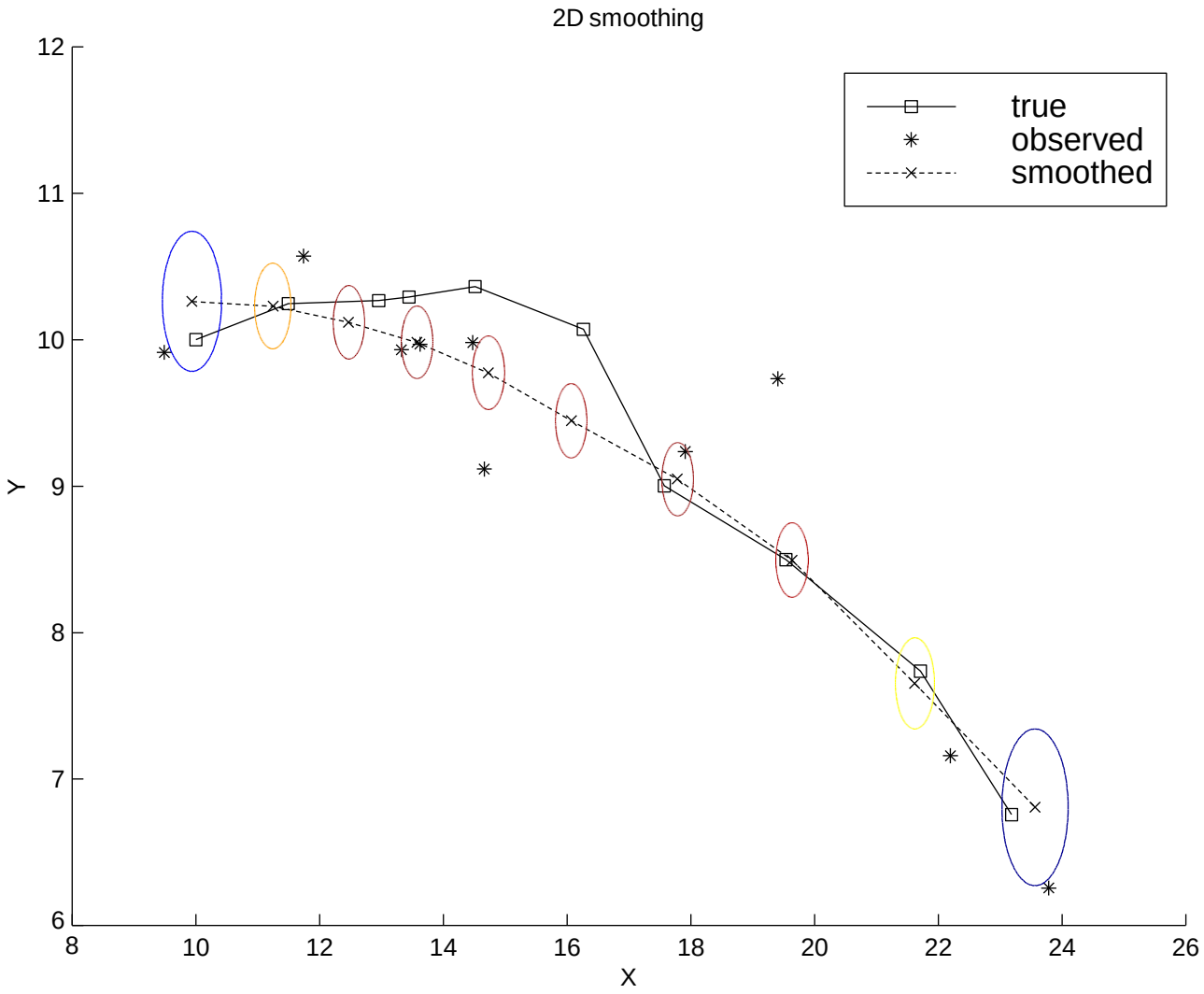
$$\mathrm{K}_{t+1} = (\mathrm{F}\Sigma_t\mathrm{F}^\mathsf{T} + \Sigma_x)\mathrm{H}^\mathsf{T}(\mathrm{H}(\mathrm{F}\Sigma_t\mathrm{F}^\mathsf{T} + \Sigma_x)\mathrm{H}^\mathsf{T} + \Sigma_z)^{-1}$$

- Σ_t 和 K_t 与观测序列无关，因此可以离线计算

2维跟踪的例子：过滤

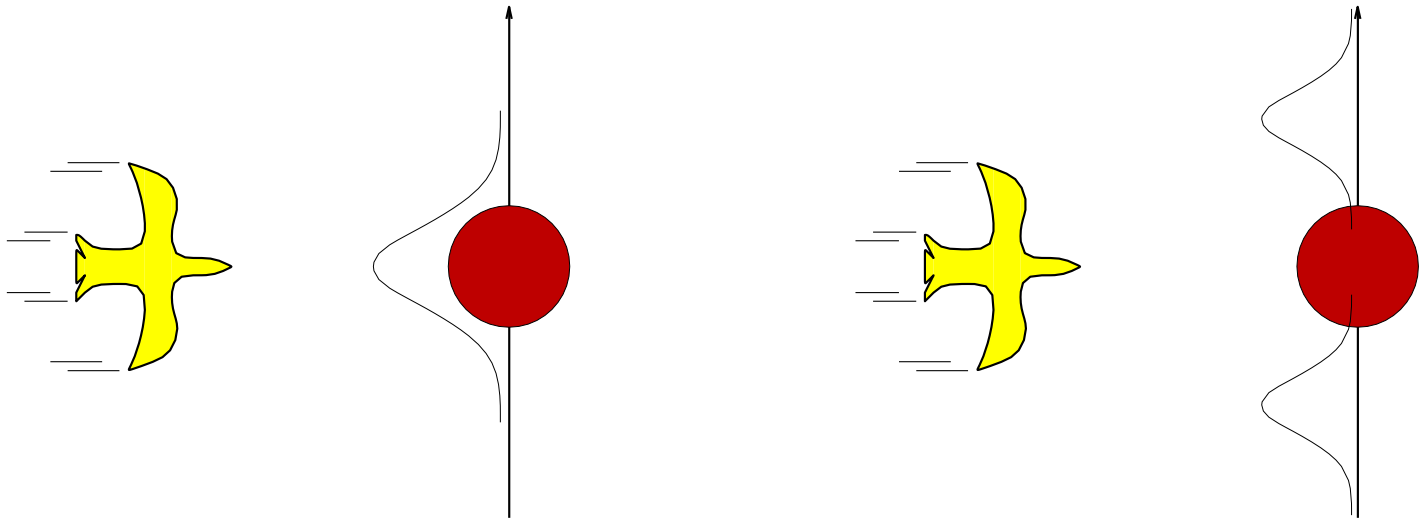

2D filtering
true
observed
filtered
Y
12
11
10
9
8
7
6
8
10
12
14
16
18
20
22
24
26

2维跟踪的例子：平滑


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

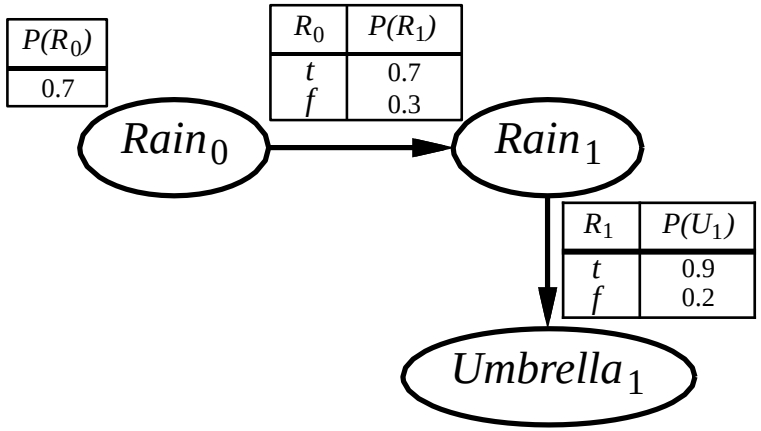
模型失效

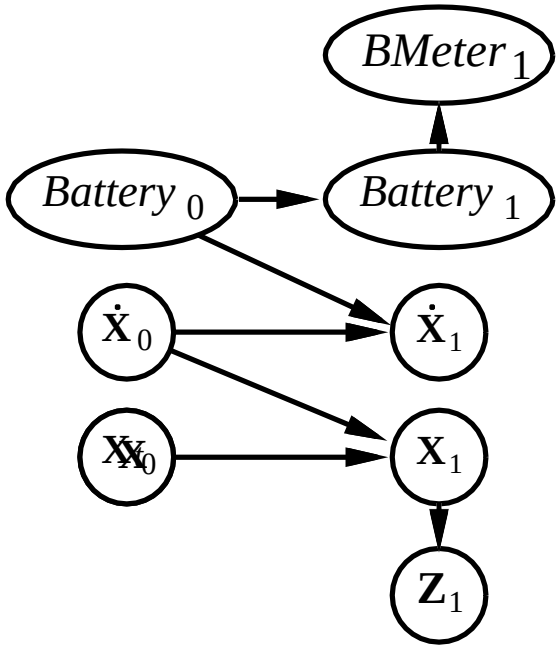
- 如果转换模型是非线性的，则不能应用
- 扩展的卡尔曼滤波模型转变为$x_t = \mu_t$附近的局部线性模型
- 如果系统在局部不平滑，则会失败



动态贝叶斯网络DBN

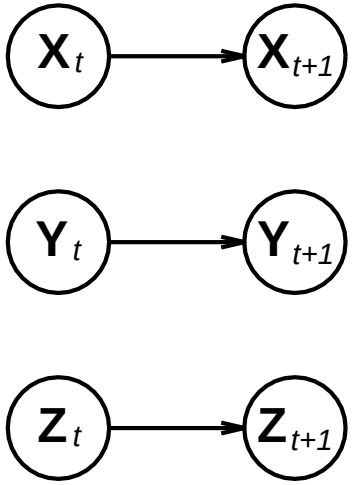
- X_t，E_t在一个重复的贝叶斯网络中包含多个变量

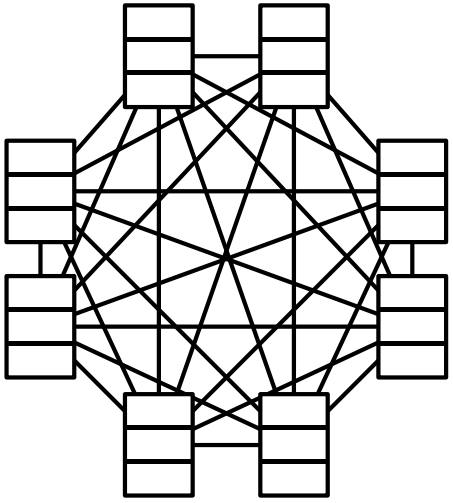




DBNs vs. HMMs

- 每个HMM都是一个单变量DBN
- 每个离散的DBN都是一个HMM

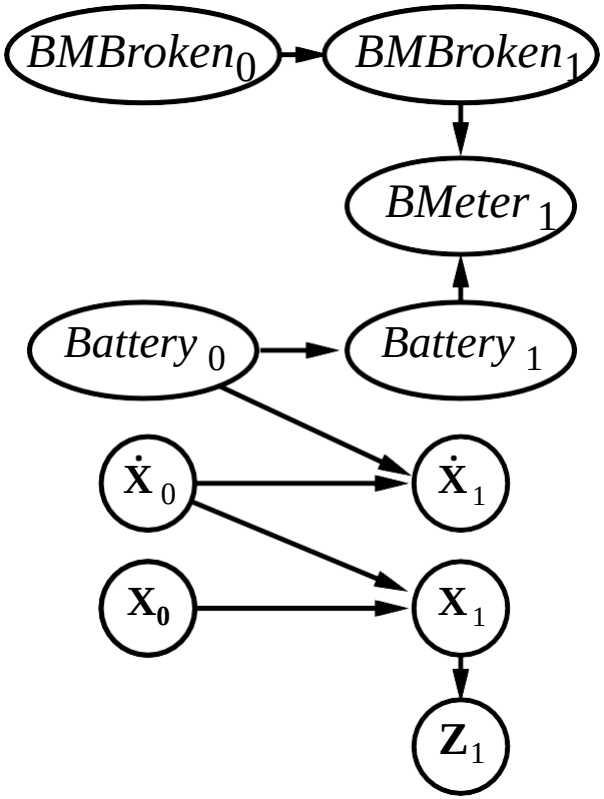


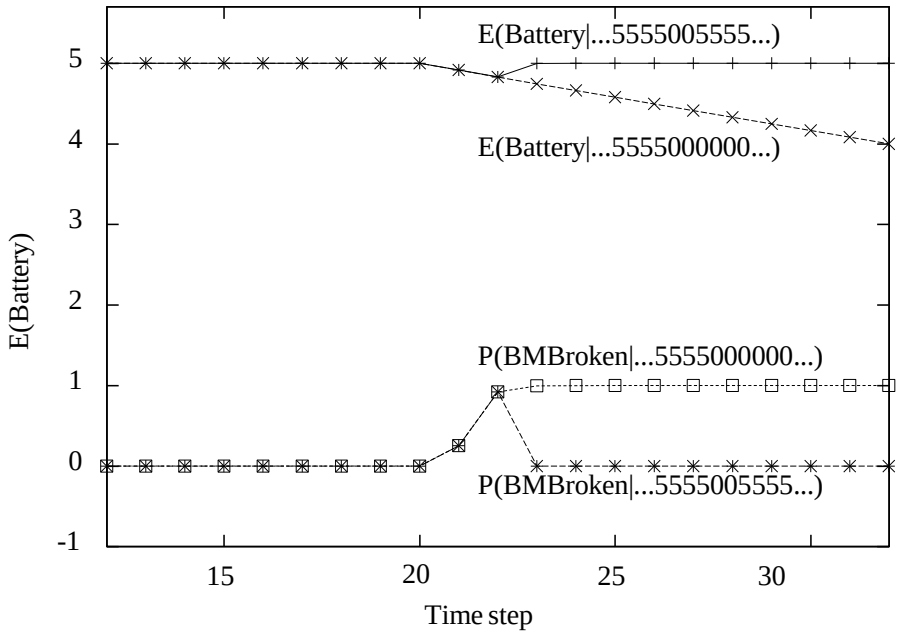


- 稀疏性依赖于指数更少的参数
  - 比如20个状态变量，每个有3个父对象
  - DBN有$20 \times 2^3 = 160$个参数，HMM有$2^{20} \times 2^{20} \approx 10^{12}$

DBNs vs 卡尔曼滤波器

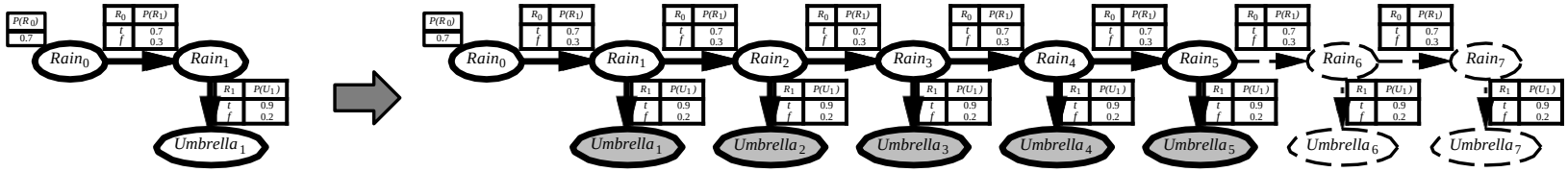
- 每个卡尔曼滤波模型都是一个DBN，但只有少数DBN是KFs
- 现实世界需要非高斯后验
- 贾跃亭和我的钥匙在哪里？电池的电量是多少？





DBNs中的精确推理

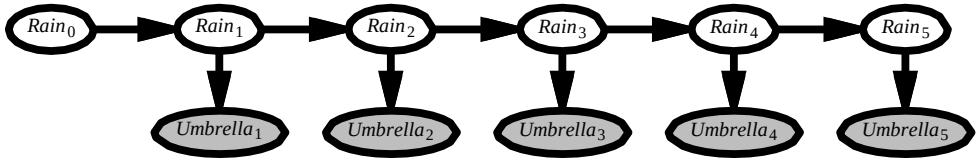
- 朴素方法：展开网络并运行任何精确的算法



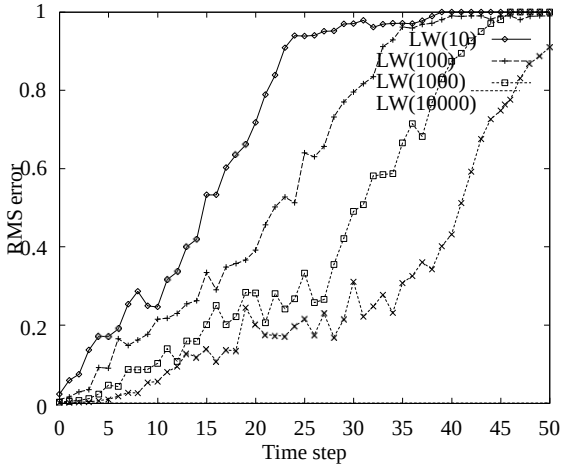
- 问题：每次更新的推理成本随 t 增长
- Rollup滤波：添加切片$t+1$，切片 t 使用变量消除最大因子是
- 最大的因子是$O(d^{n+1})$,更新的时间复杂度是$O(d^{n+2})$
- 作为对比，HMM的更新时间复杂度是$O(d^{2n})$

DBNs的可能性加权

- 加权样本集近似于置信状态

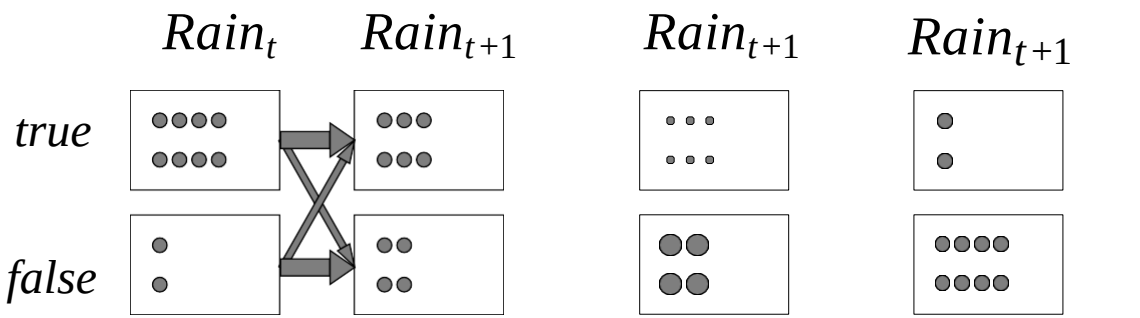


- LW的采样不在意证据
  - ⇒ “同意”的百分比随t呈指数下降
  - ⇒所需样本数随t呈指数增长



粒子滤波

- 基本思想：确保样本(粒子)的总体能够跟踪状态空间的高可能性区域
- 重复出现的粒子与真实可能性e_t成正比



- 广泛用于跟踪非线性系统，特别是在视觉任务中
- 也用于移动机器人的同步定位和绘图
  - 该场景下会用到10^5-维状态空间

粒子滤波

- 假设在时间 t 时是连续的：$N(\mathrm{x}_t|\mathrm{e}_{1:t})/N = P(\mathrm{x}_t|\mathrm{e}_{1:t})$
- 前向传播：x_{t+1}的规模是

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

- 对于e_{t+1}根据可能性对样本进行加权

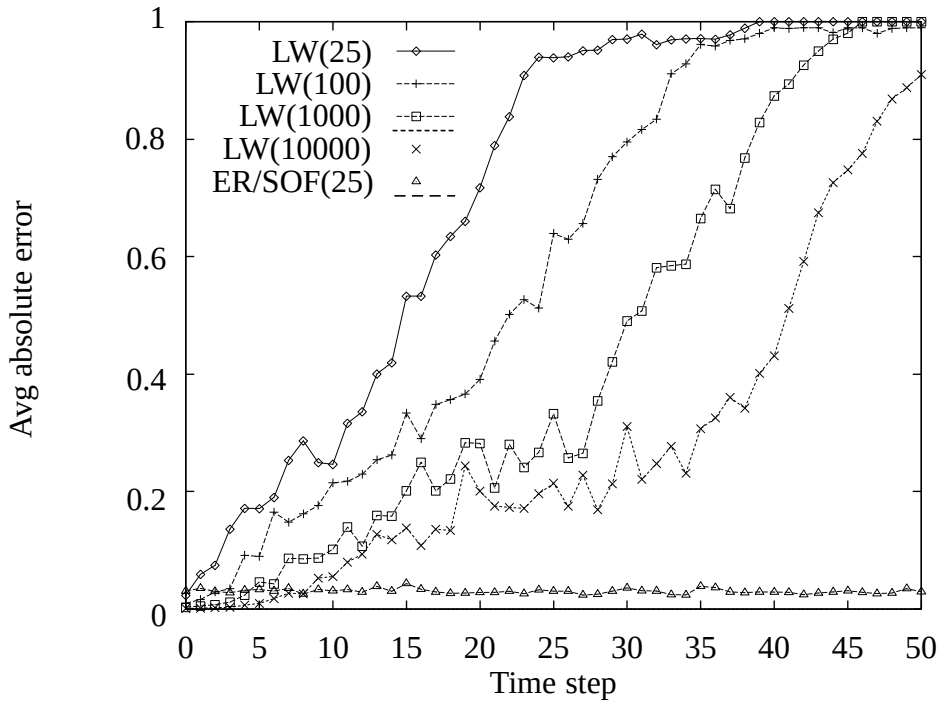
$$W(\mathrm{x}_{t+1}|\mathrm{e}_{1:t+1}) = P(\mathrm{e}_{t+1}|\mathrm{x}_{t+1}) N(\mathrm{x}_{t+1}|\mathrm{e}_{1:t})$$

- 重新对 W 采样以获得成比例的规模

$$\begin{aligned} N(\mathrm{x}_{t+1}|\mathrm{e}_{1:t+1})/N &= aW(\mathrm{x}_{t+1}|\mathrm{e}_{1:t+1}) = aP(\mathrm{e}_{t+1}|\mathrm{x}_{t+1}) N(\mathrm{x}_{t+1}|\mathrm{e}_{1:t}) \\ &= aP(\mathrm{e}_{t+1}|\mathrm{x}_{t+1}) \Sigma_{\mathrm{x}_t} P(\mathrm{x}_{t+1}|\mathrm{x}_t) N(\mathrm{x}_t|\mathrm{e}_{1:t}) \\ &= a^{\mathrm{t}} P(\mathrm{e}_{t+1}|\mathrm{x}_{t+1}) \Sigma_{\mathrm{x}_t} P(\mathrm{x}_{t+1}|\mathrm{x}_t) P(\mathrm{x}_t|\mathrm{e}_{1:t}) \\ &= P(\mathrm{x}_{t+1}|\mathrm{e}_{1:t+1}) \end{aligned}$$

粒子滤波的性能

- 粒子滤波的近似误差随时间保持有界，至少在经验上理论分析是困难的



总结

- 时间模型使用随时间重复出现的状态和感知变量
- 马尔可夫做了平稳性假设，所以需要
  - 转移模型$\mathrm{P}(\mathrm{X}_t|\mathrm{X}_{t-1})$
  - 感知模型$\mathrm{P}(\mathrm{E}_t|\mathrm{X}_t)$
- 任务包括过滤，预测，平滑，序列预测
  - 所有这些都是递归地完成的，每一步的代价都是常数
- 隐马尔可夫模型具有单个离散状态变量，可用于语音识别
- 卡尔曼滤波器允许n个状态变量，满足线性高斯分布，更新计算量为$O(n^3)$
- 动态贝叶斯网络包含HMMs和卡尔曼滤波器，准确计算更新量比较棘手
- 粒子滤波是一种对于DBNs的有效近似

语音检测

- 将语音用于概率推理
- 语音的发音
- 单词的发音
- 单词的顺序

语音作为概率推理

- 语音信号是嘈杂的、可变的、模糊的
- 给定语音信号，最有可能的单词序列是什么?
  - 也就是说给定$Words$来最大化可能性 $P(Words|signal)$
- 使用贝叶斯规则:

$$P(Words|signal) = \alpha P(signal|Words)P(Words)$$

- 按声学模型+语言模型分解，得到:
  - $Words$是隐状态序列
  - $signal$是观测序列

发音

- 人类所有的语言都是由40-50种发音组成的，由发音器官(嘴唇、牙齿、舌头、声带、气流)的结构决定。
- 在单词和信号之间形成中间隐藏状态

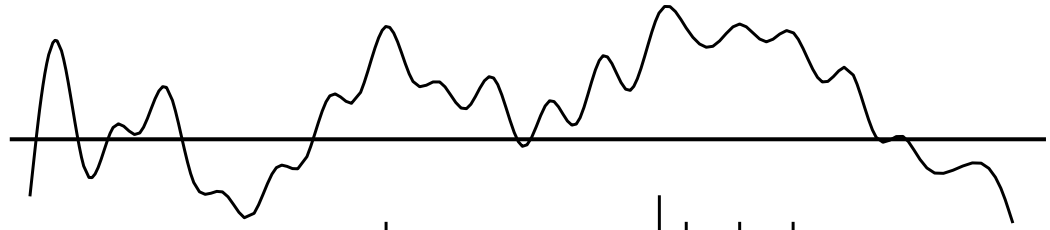
  ⇒音频模型=语音模型+电话模型
- ARPAbet是为美式英语设计的发音表

[iy]	beat	[b]	bet	[p]	pet
[ih]	bit	[ch]	Chet	[r]	rat
[ey]	bet	[d]	debt	[s]	set
[ao]	bought	[hh]	hat	[th]	thick
[ow]	boat	[hv]	high	[dh]	that
[er]	Bert	[l]	let	[w]	wet
[ix]	roses	[ng]	sing	[en]	button
⋮	⋮	⋮	⋮	⋮	⋮

语音

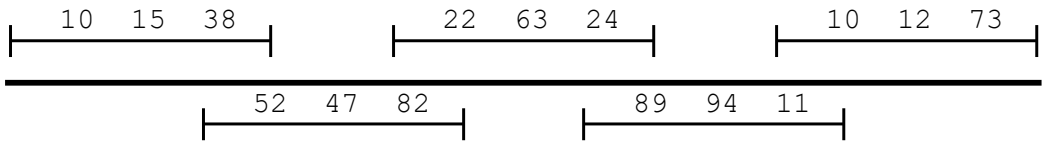
- 原始信号是麦克风位移随时间的函数
- 处理成重叠的30ms帧，每帧都通过选取的特征来描述

- 模拟声音信号：



- 采样量化数字信号：

- 帧与特点：



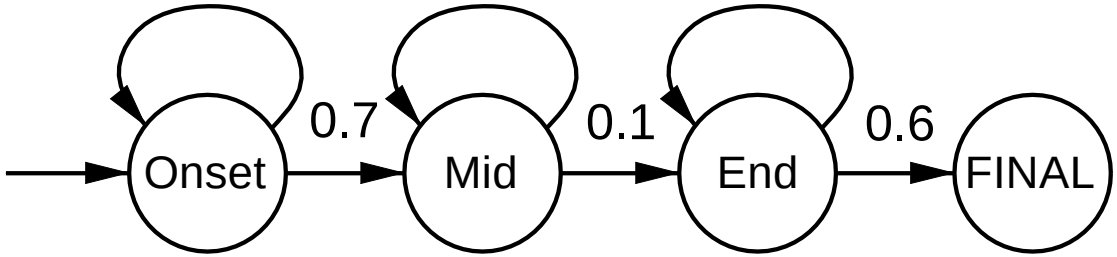
- 帧特征通常是功率谱中的共振峰

发音模型

- 帧特征通过$P(features|phone)$描述
  - 一个在[0 . . . 255]的整数(使用矢量量化)
  - 或高斯混合函数的参数
- 三态发音：每个发音有三个阶段(开始、中期、结束)
  - [t]静默开始，中段爆发，嘶嘶结束
  - $\Rightarrow P(features|phone, phase)$
- 上下文发音：每个发音由n^2个不同的发音组合而成，组合方式取决于它左边和右边的发音
  - “star”的[t] 可以写为[t(s,aa)](不同于“tar”)
- 上下文发音用于处理协同发音效果很有效，因为发音器往往具有惯性，无法在位置之间进行即时切换
- 例如“eighth”中的[t]有舌头抵住门牙

发音模型的例子

- HMM中的[m]的发音

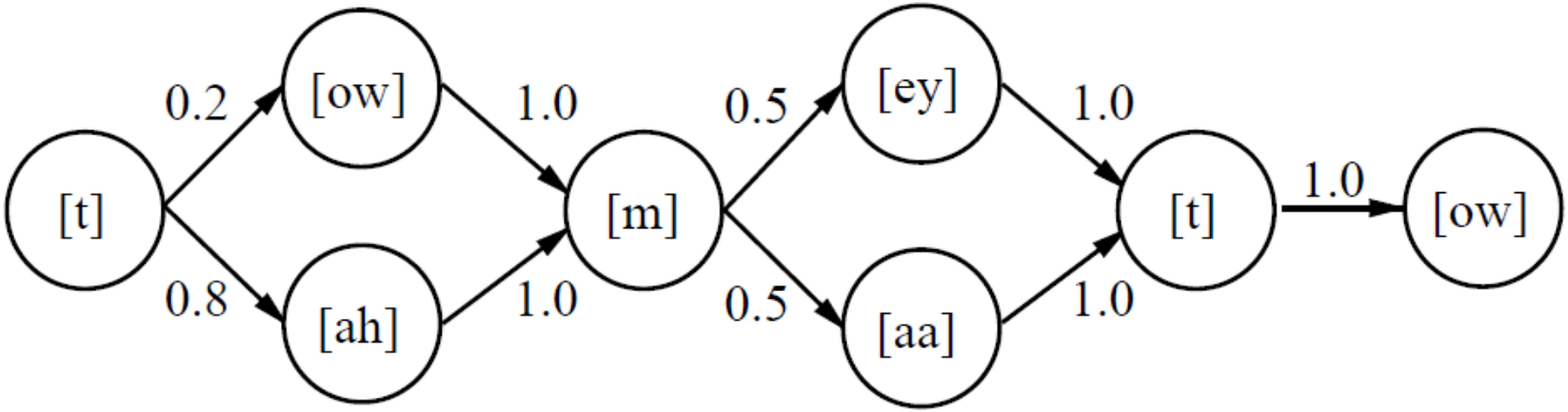


- 输出HMM发音的可能性：

Onset:	Mid:	End:
C1: 0.5	C3: 0.2	C4: 0.1
C2: 0.2	C4: 0.7	C6: 0.5
C3: 0.3	C5: 0.1	C7: 0.4

单词发音模型

- 每个单词都被描述为电话序列的分布
- 分布表示为HMM转换模型



$P([towmeytow]|\text{“tomato”}) = P([towmaatow]|\text{“tomato”}) = 0.1$

$P([tahmeytow]|\text{“tomato”}) = P([tahmaatow]|\text{“tomato”}) = 0.4$

- 结构是手动创建的，从数据中学习转换概率

孤立单词

- 发音模型和文字模型可以修复孤立单词的可能性$P(e_{1:t}|word)$

$$P(word|e_{1:t}) = \alpha P(e_{1:t}|word)P(word)$$

- 先验概率$P(word)$可以通过简单地计算单词频率得到
- $P(e_{1:t}|word)$可以递归计算，定义$l_{1:t} = P(X_t, e_{1:t})$
- 并使用递归更新$l_{1:t+1} = Forward(l_{1:t}, e_{t+1})$
- 则可以得到$P(e_{1:t}|word) = \Sigma_{x_t} l_{1:t}(x_t)$
- 经过训练的孤立单词听写系统的准确率达到95-99%

连续发音

- 连续发音不仅仅是一系列孤立的单词识别问题
  - 相邻词高度相关
  - 最有可能的单词序列≠单词最可能组成的序列形式
  - 分词：语言中很少有间隔
  - 连接词，如“然后”
- 连续语音系统在正常情况下可以达到60-80%的准确率

语言模型

- 一个单词序列的先验概率由链式法则给出：

$$P(w_1 \cdots w_n) = \prod_{i=1}^{n} P(w_i | w_1 \cdots w_{i-1})$$

- Bigram模型：

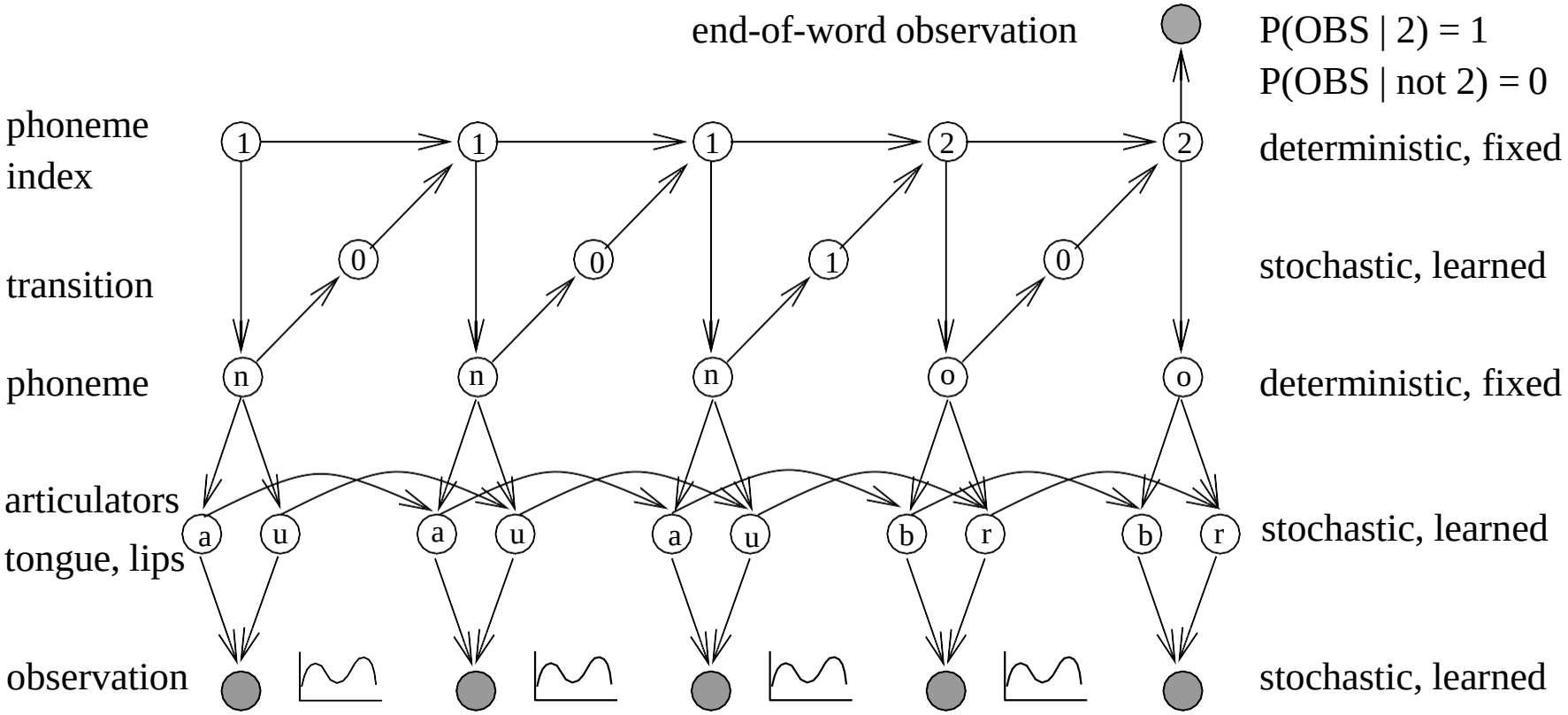
$$P(w_i | w_1 \cdots w_{i-1}) = P(w_i | w_{i-1})$$

- 通过计算大型文本语料库中的所有单词对进行训练
- 更复杂的模型（三元组、语法等）会有所帮助

结合HMM

- 语言+单词+发音的状态组合模型的状态是通过以下三部分组成的：
  - 我们所用的单词
  - 该单词的发音
  - 该发音中的状态标记
- Viterbi 算法可以用来找出最有可能的发音状态序列
- 考虑所有可能的单词序列和边界的分割并不总是给出最可能的单词序列，因为每个单词序列是许多状态序列的总和
- Jelinek在1969年发明了一种显示最可能的单词序列的方法A*
  - 其时间复杂度能达到 $\log P(w_i|w_{i-1})$

用于语音识别的DBNs



Also easy to add variables for, e.g., gender, accent, speed.
Zweig and Russell (1998) show up to 40% error reduction over HMMs

总结

- 从20世纪70年代中期开始，语音识别就被定义为概率推理
- 证据=语音信号，隐变量=单词和电话序列“上下文”效应(协同发音等)是通过增加状态来处理的
- 人类语言的可变性(速度、音色等)和背景噪声使得真实环境下的连续语音识别成为一个有待解决的问题