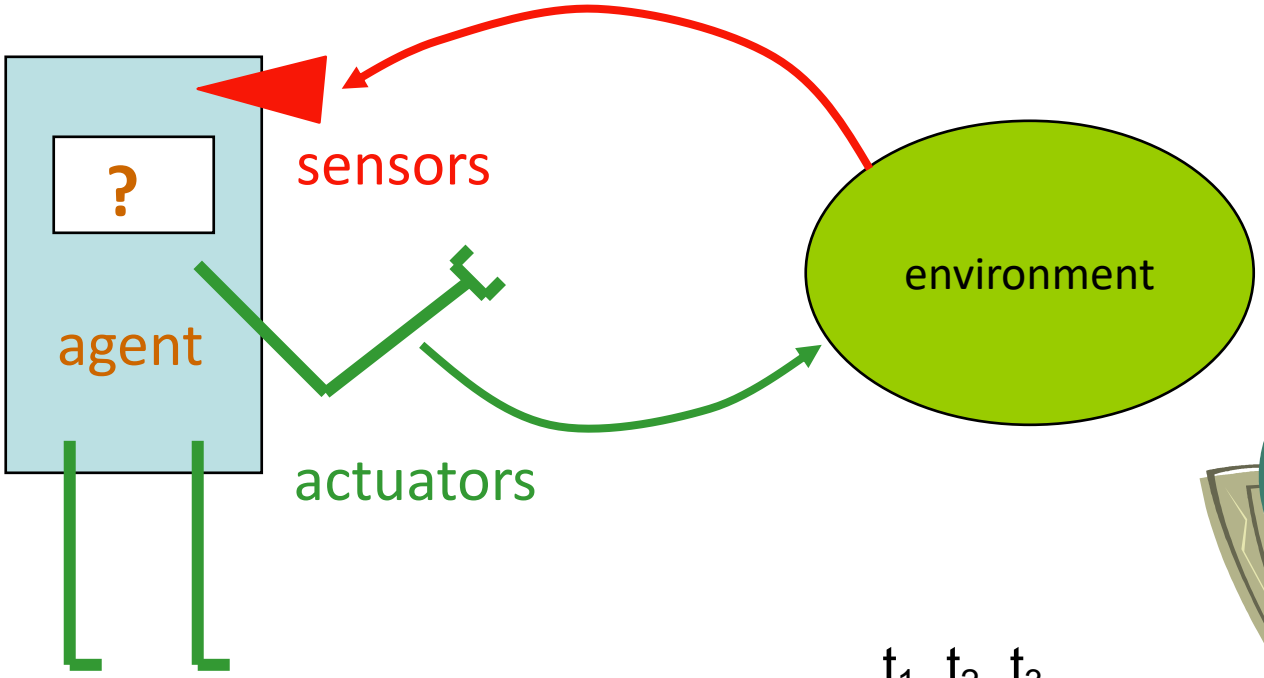
Non-Standard-Datenbanken

Dynamische Bayessche Netze

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Temporal Probabilistic Agent

- Previous and current states (PDBs, Bayesian networks)
  - From visible probabilistic data to derived probabilistic data
- Forecasting (temporal PDBs, dynamic Bayesian networks)
  - From visible and current estimated environment state to estimation about the next environment state



$t_1, t_2, t_3, \ldots$

Time and Uncertainty

- The world changes, we need to track and predict it
  - Examples: diabetes management, traffic monitoring
- Basic idea: copy state and evidence variables of Bayesian network for each time step (snapshots)
- $\mathbf{X}_t$: set of unobservable state variables at time t
  - e.g., BloodSugar_t, StomachContents_t
- $\mathbf{E}_t$: set of evidence variables at time t
  - e.g., $\text{MeasuredBloodSugar}_t$, PulseRate_t, FoodEaten_t
  - Observation at time t is $\mathbf{E}_t = \mathbf{e}_t$ for some set of values e_t
- Assumes discrete time steps
  - Notation: $\mathbf{X}_{a:b}$ denotes the set of variables from $\mathbf{X}_a$ to $\mathbf{X}_b$
- Dynamic Bayesian Network (DBN)

DBN - Representation

- Problem:

  1. Process behavior might change (CPTs of evidence variables change)
  2. Evidence and state variables might depend on all previous states (unbounded number of Parents in CPT)

- Solution:

  1. Assume that changes in the world state are caused by a stationary process (unchanging process over time).

$$P(U_t / Parent(U_t))$$ is the same for all *t*

"Dynamic" means we are modeling a dynamic system, not that the structure of a Bayesian network changes over time.

DBN - Representation

- Solution cont.:

2. Use **Markov assumption** - The current state depends on only in a finite history of previous states.

   Usually: First-order Markov process (only previous state matters)

$$P(X_t / X_{0:t-1}) = P(X_t / X_{t-1})$$

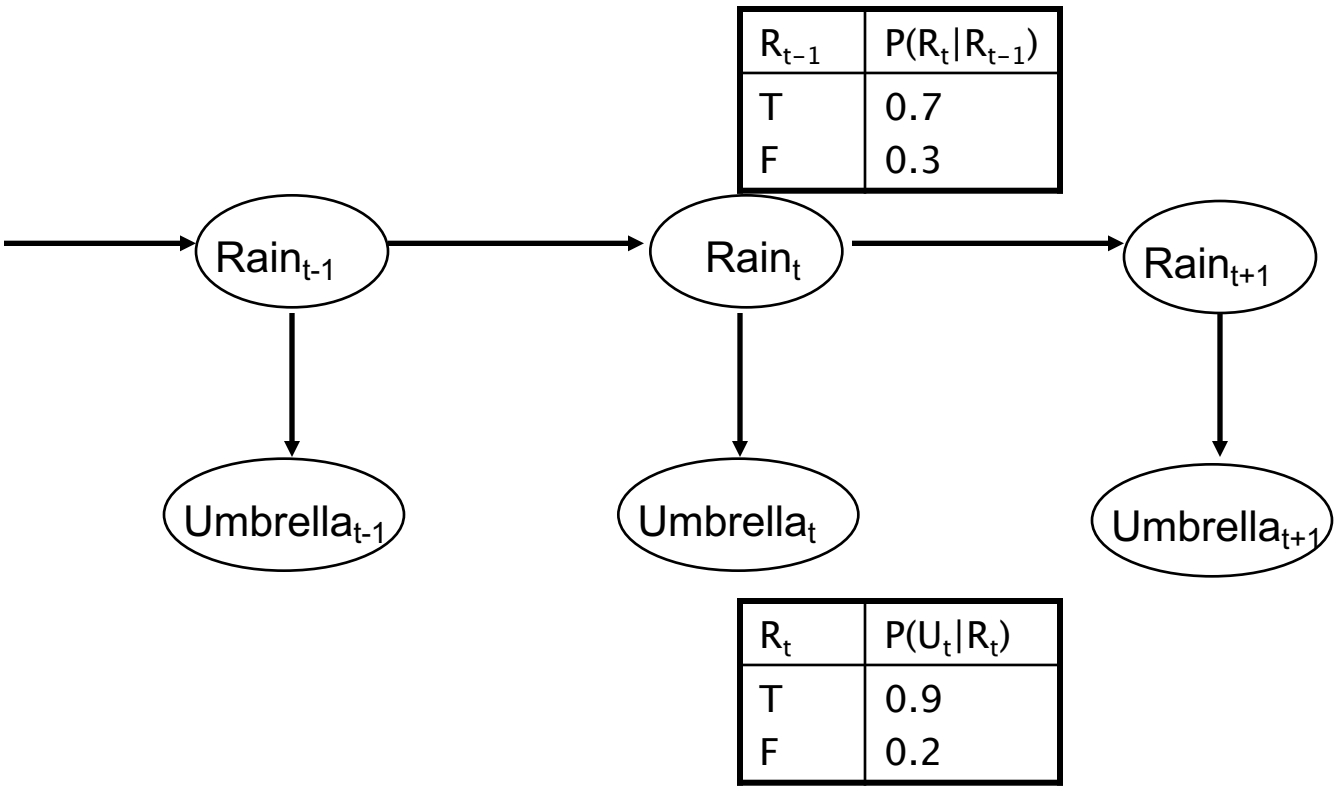
Transition Model

In addition to restricting the parents of the state variable X_t, we must restrict the parents of the evidence variable E_t

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

Sensor Model

Example


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

Complete Joint Distribution

- Given:
  - Transition model: $P(X_t | X_{t-1})$
  - Sensor model: $P(E_t | X_t)$
  - Prior probability: $P(X_0)$

- Then we can specify complete joint distribution:

$$P(X_0, X_1, \ldots, X_t, E_1, \ldots, E_t) = P(X_0) \prod_{i=1}^{t} P(X_i | X_{i-1}) P(E_i | X_i)$$

Inference Tasks

- **Filtering:** What is the probability that it is raining today, given all the umbrella observations up through today?
- **Prediction:** What is the probability that it will rain the day after tomorrow, given all the umbrella observations up through today?
- **Smoothing:** What is the probability that it rained yesterday, given all the umbrella observations through today?
- **Most likely explanation / most probable explanation:** if the umbrella appeared the first three days but not on the fourth, what is the most likely weather sequence to produce these umbrella sightings?

DBN – Basic Inference

- Filtering or Monitoring:

  Compute the belief state - the posterior distribution over the *current* state, given all evidence to date.

$$P(X_{t+1} / e_{1:t+1})$$

DBN – Basic Inference

- Filtering cont.

Given the results of filtering up to time *t*, one can easily compute the result for *t+1* from the new evidence e_{t+1}

$$P(X_{t+1} / e_{1:t+1}) = f(e_{t+1}, P(X_t / e_{1:t}))$$ (for some function *f*)

$$= P(X_{t+1} / e_{1:t}, e_{t+1})$$ (dividing up the evidence)

$$= \alpha P(e_{t+1} / X_{t+1}, e_{1:t}) P(X_{t+1} / e_{1:t})$$ (using Bayes' Theorem)

$$= \alpha P(e_{t+1} / X_{t+1}) P(X_{t+1} / e_{1:t})$$ (by the Markov property of evidence)

α is a normalizing constant used to make probabilities sum up to 1.

DBN – Basic Inference

- Filtering cont.

The second term $P(X_{t+1} / e_{1:t})$ represents a one-step prediction of the next step, and the first term $P(e_{t+1} / X_{t+1})$ updates this with the new evidence.

Now we obtain the one-step prediction for the next step by conditioning on the current state X_t:

$$P(X_{t+1} / e_{1:t+1}) = \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_t, e_{1:t}) P(x_t / e_{1:t})$$

$$= \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / \boldsymbol{x_t}) P(x_t / e_{1:t})$$

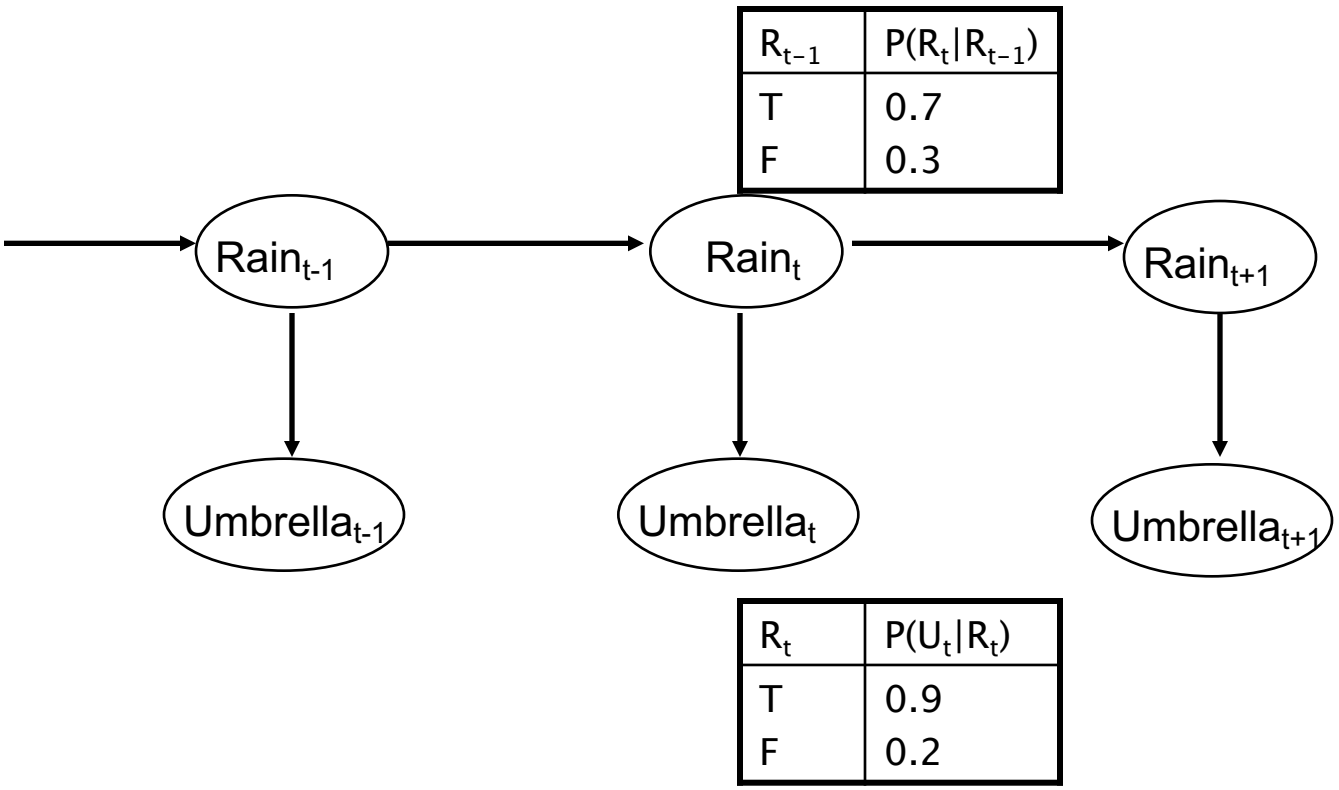
(using the Markov property)

Anmeldung zur Klausur wird nach der Vorlesung freigeschaltet

Lübeck

titut für Informationssysteme

Example


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

Filtering: Forward Messages

$$P(X_{t+1} / e_{1:t+1}) = f(e_{t+1}, P(X_t / e_{1:t}))$$

$$= \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_t) P(x_t / e_{1:t})$$

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

Time and space **constant** (independent of t)

DBN – Basic Inference

$$P(X_{t+1} / e_{1:t+1}) = \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_t) P(x_t / e_{1:t})$$

Illustration for two steps in the Umbrella example:

- On day 1, the umbrella appears so U1=true. The prediction from t=0 to t=1 is

$$P(R_1) = \sum_{r_0} P(R_1 / r_0) P(r_0)$$

and updating it with the evidence for t=1 gives

$$P(R_1 / u_1) = \alpha P(u_1 / R_1) P(R_1)$$

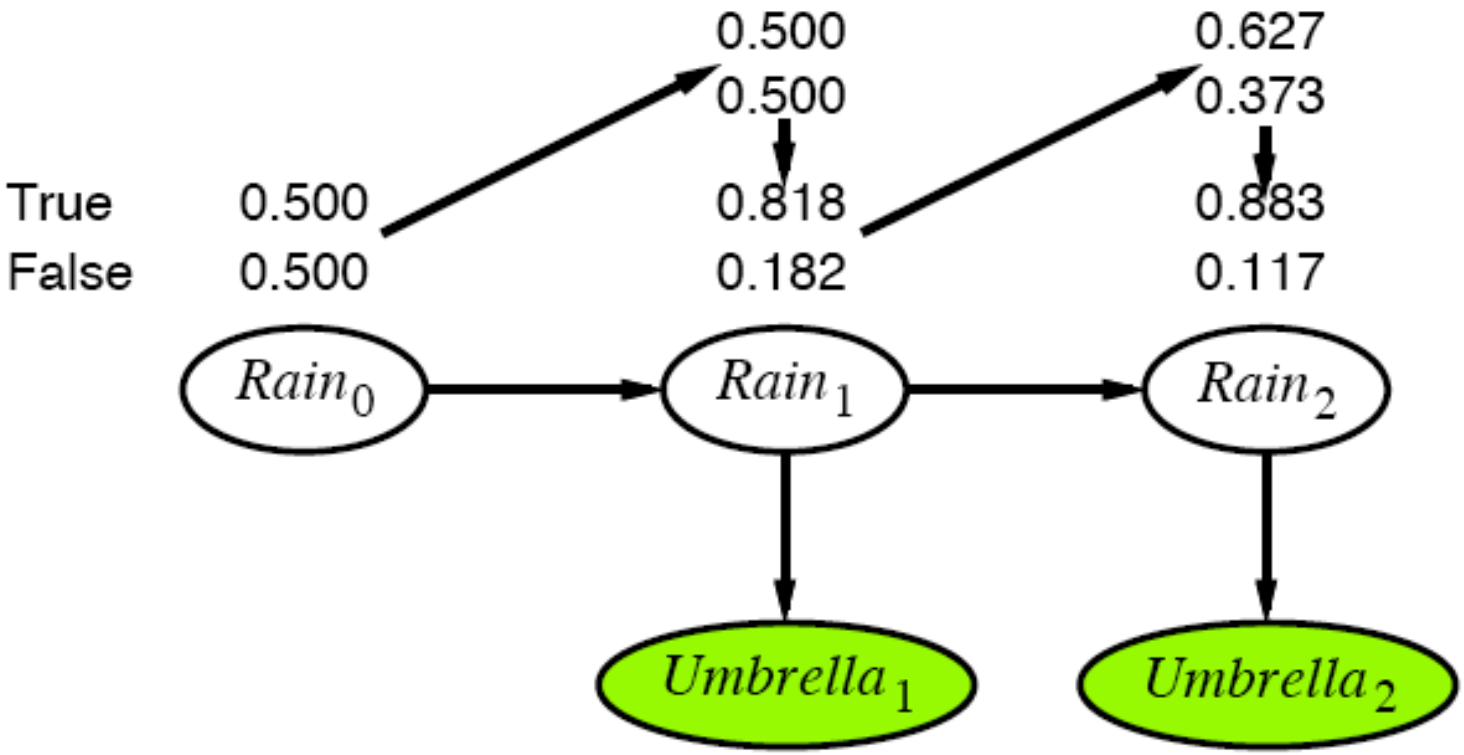
- On day 2, the umbrella appears so U2=true. The prediction from t=1 to t=2 is

$$P(R_2 / u_1) = \sum_{r_1} P(R_2 / r_1) P(r_1 / u_1)$$

and updating it with the evidence for t=2 gives

$$P(R_2 / u_1, u_2) = \alpha P(u_2 / R_2) P(R_2 / u_1)$$

Example cntd.



DBN – Basic Inference

$$P(X_{t+1} / e_{1:t+1}) = \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_t) P(x_t / e_{1:t})$$

- Prediction:

  Compute the posterior distribution over the *future* state, given all evidence to date.

$$P(X_{t+k} / e_{1:t})$$

for some k>0

The task of prediction can be seen simply as filtering without the addition of new evidence
(e.g., assume uniform distribution for evidence values).

DBN – Basic Inference

- Smoothing or hindsight inference:

  Compute the posterior distribution over the *past* state, given all evidence up to the present.

$$P(X_k / e_{1:t})$$

for some k such that 0 ≤ k < t.

Hindsight provides a better estimate of the state than was available at the time, because it incorporates more evidence from the “future”.

Smoothing

Divide evidence $\mathbf{e}_{1:t}$ into $\mathbf{e}_{1:k}$, $\mathbf{e}_{k+1:t}$:

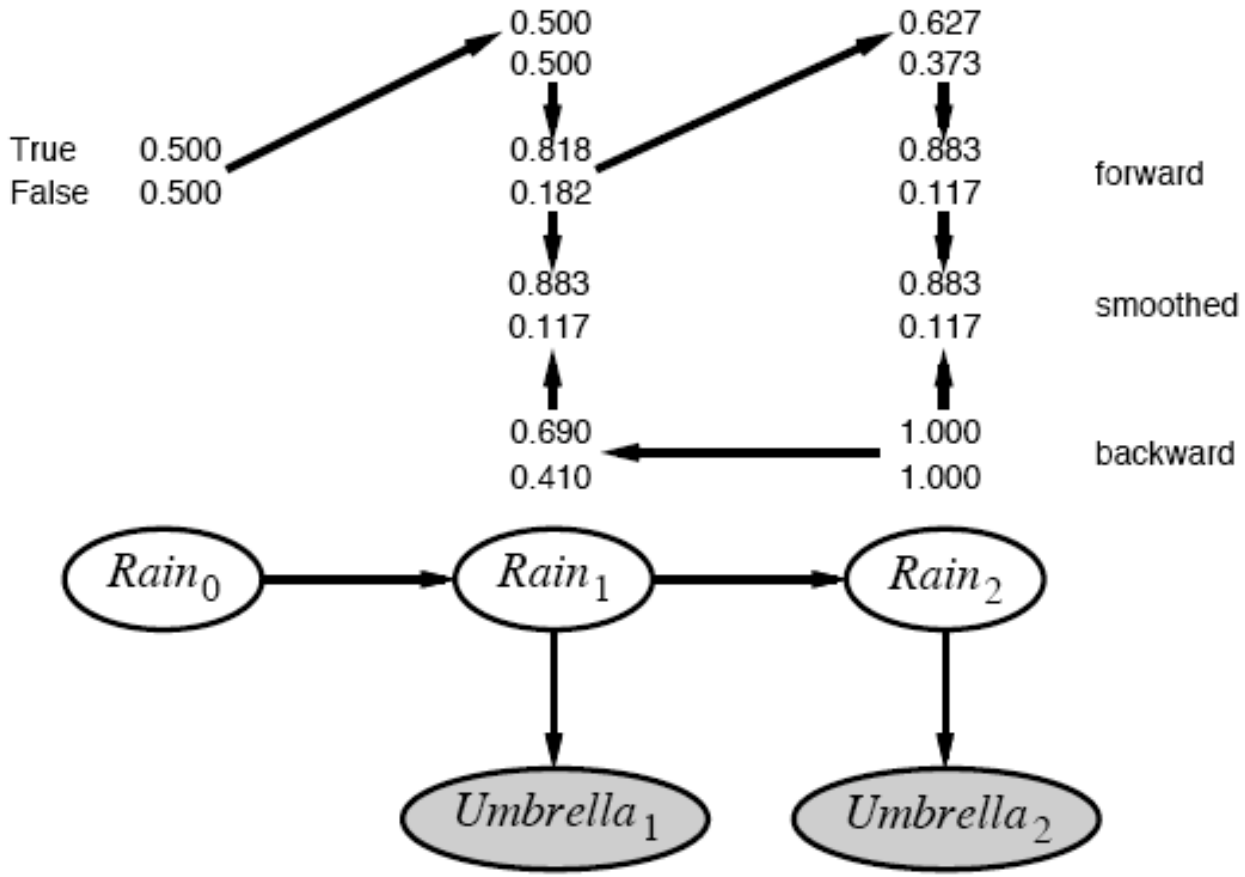
$$\begin{aligned}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k},\mathbf{e}_{k+1:t}) \\ &= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k,\mathbf{e}_{1:k}) \\ &= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)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

Backward message computed by a backwards recursion:

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

Forward–backward algorithm: cache forward messages along the way
Time linear in t (polytree inference), space $O(t|\mathbf{f}|)$

Example contd.



DBN – Basic Inference

- Most likely explanation:

  Compute the sequence of states that is most likely to have generated a given sequence of observation.

$$\arg\max_{x_{1:t}} P(X_{1:t} \mid e_{1:t})$$

Algorithms for this task are useful in many applications, including, e.g., speech recognition.

Most-likely explanation

Most likely sequence $\neq$ sequence of most likely states!!!!

Most likely path to each $\mathbf{x}_{t+1}$
= most likely path to **some** $\mathbf{x}_t$ plus one more step

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

$$P(X_{t+1} / e_{1:t+1}) = f(e_{t+1,} P(X_t / e_{1:t} \quad))$$

$$= \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_{\ t}) P(x_t / e_{1:t})$$

Most-likely explanation

Most likely sequence $\neq$ sequence of most likely states!!!!

Most likely path to each $\mathbf{x}_{t+1}$
= most likely path to **some** $\mathbf{x}_t$ plus one more step

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

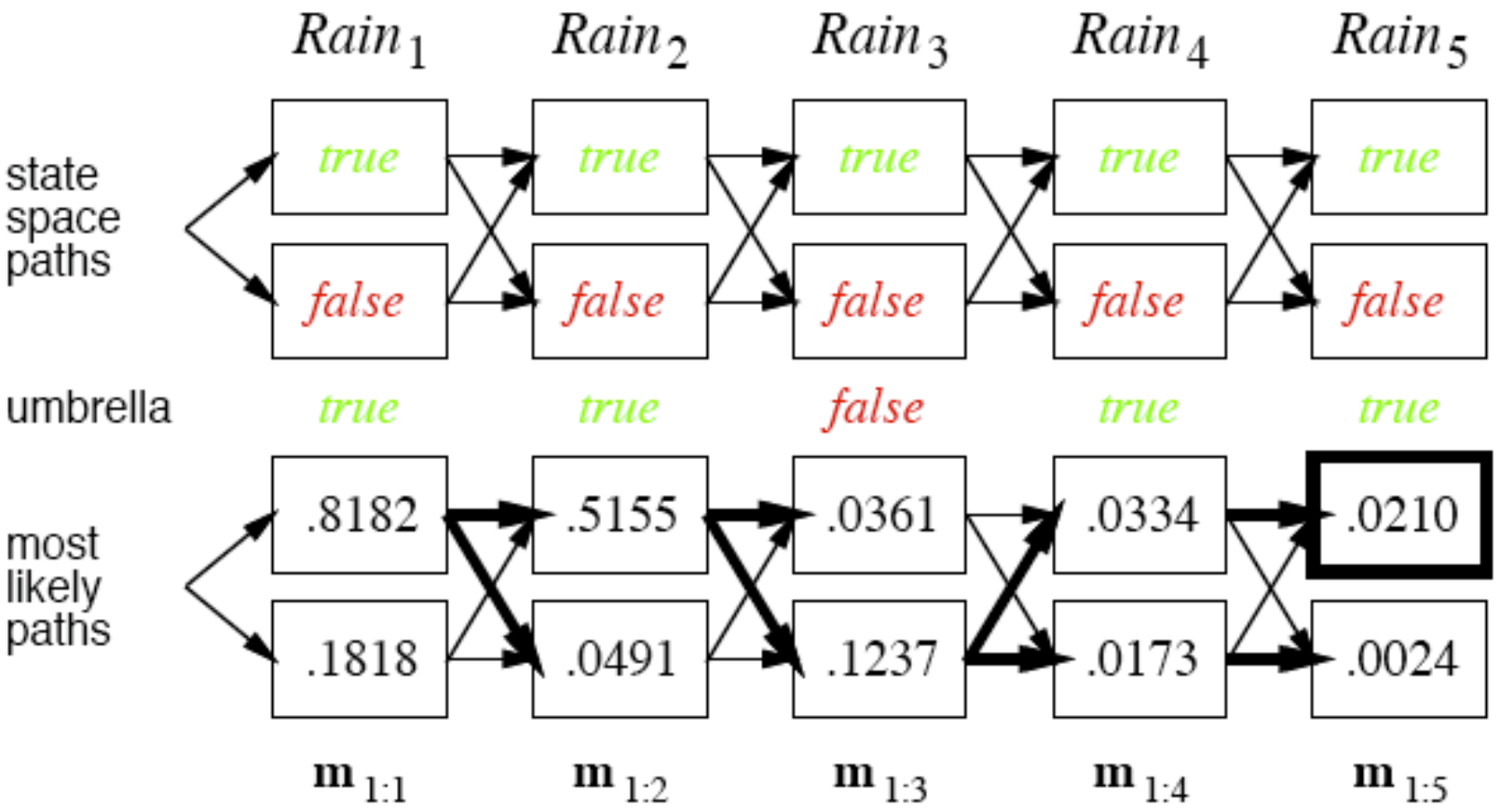
Identical to filtering, except $\mathbf{f}_{1:t}$ replaced by

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

I.e., $\mathbf{m}_{1:t}(i)$ gives the probability of the most likely path to state i.
Update has sum replaced by max, giving the Viterbi algorithm:

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

Rain/Umbrella Example



DBN – Special Cases

- Hidden Markov Model (HMMs):

  Temporal probabilistic model in which the state of the process
  is described by a single discrete random variable. (The simplest kind of DBN)

- Kalman Filter Models (KFMs):

  Estimate the state of a physical system from noisy observations over time.
  Also known as linear dynamical systems (LDSs).

DBN – Basic Inference

- Filtering

$$P(X_{t+1} / e_{1:t+1}) = \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_t) P(x_t / e_{1:t})$$

- Smoothing

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

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

- Most likely sequence

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

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

Hidden Markov Models

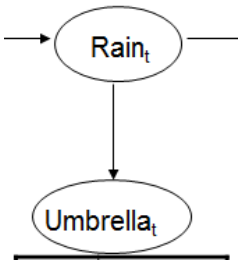
$\mathbf{X}_t$ is a single, discrete variable (usually $\mathbf{E}_t$ is too)
Domain of X_t is $\{1, \ldots, S\}$

Transition matrix $\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i)$, e.g., $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$ *new state* / *old state*

Sensor matrix $\mathbf{O}_t$ for each time step, diagonal elements $P(e_t | X_t = i)$

e.g., with $U_1 = true$, $\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$ $U_3 = false$ $\mathbf{O}_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$

R_{t-1}	$P(R_t \mid R_{t-1})$
T	0.7
F	0.3



R_t	$P(U_t \mid R_t)$
T	0.9
F	0.2

Forward and backward messages as column vectors:

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

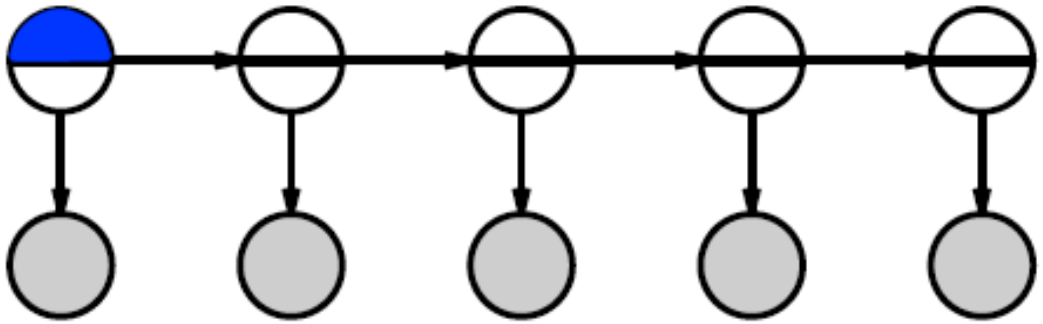
Forward-backward algorithm needs time $O(S^2 t)$ and space $O(St)$

Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

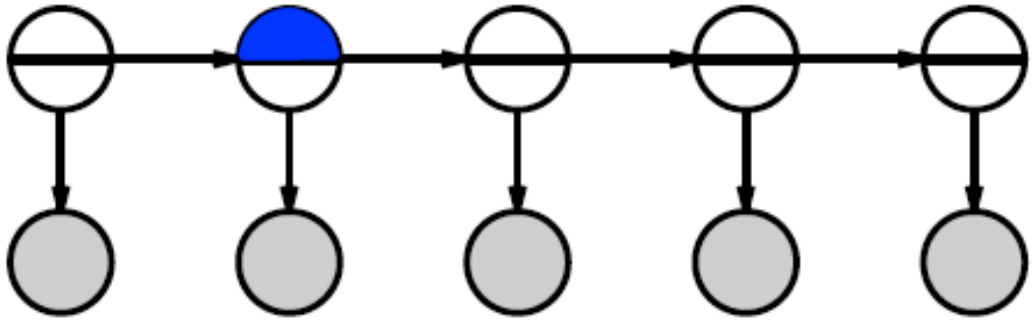


Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

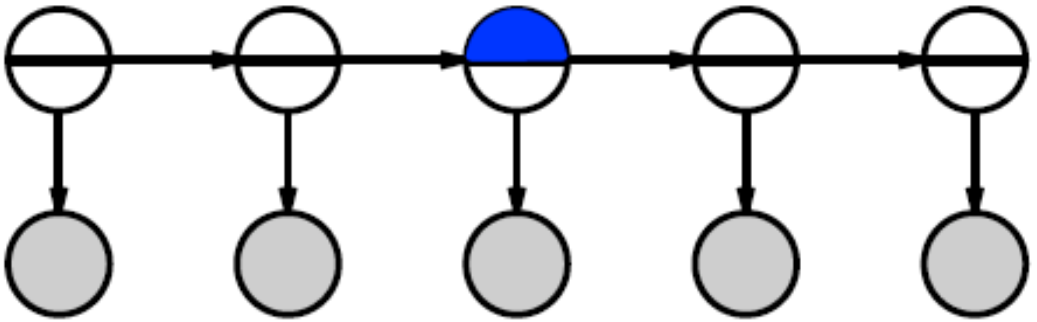


Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

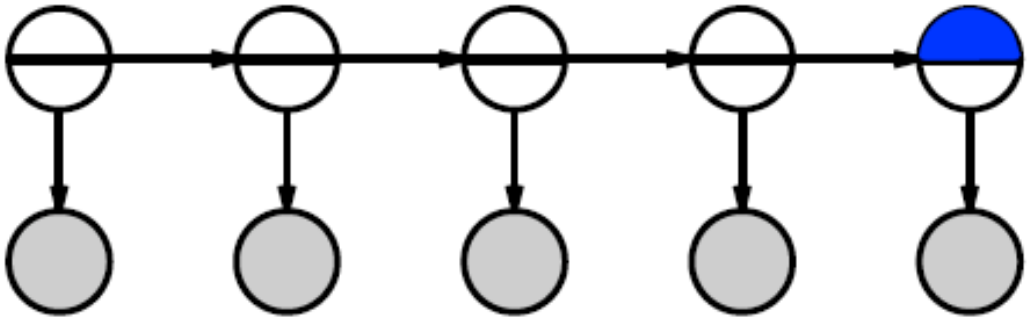


Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

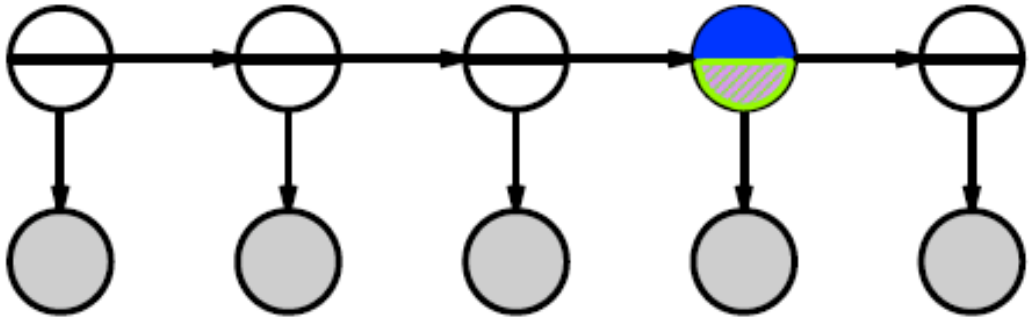


Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

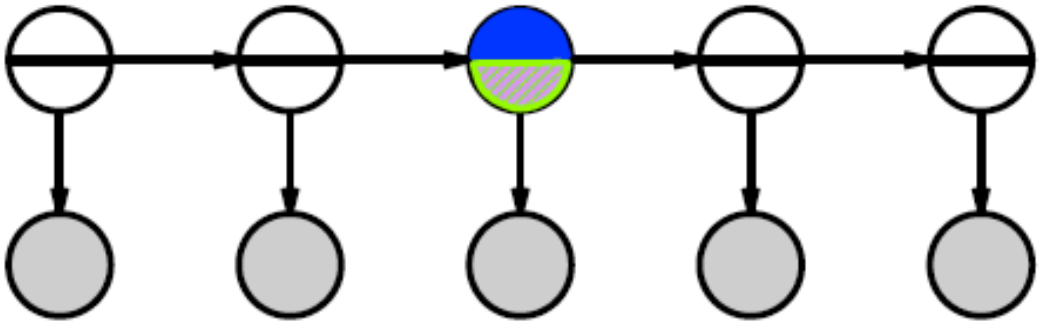


Country Dance Algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

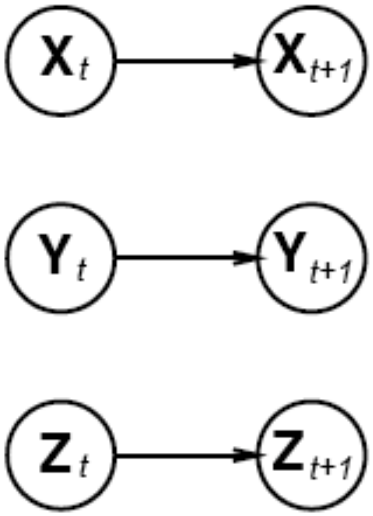
$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha' (\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

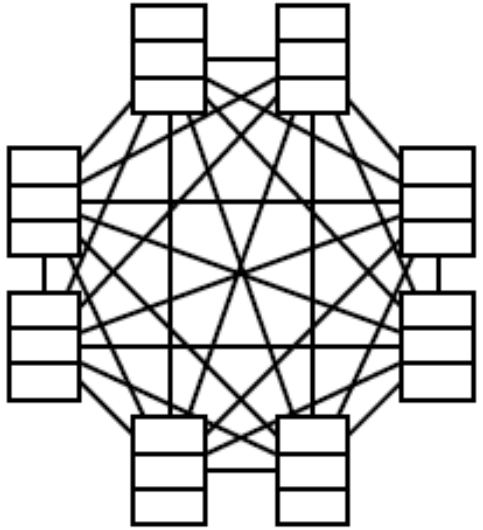
Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$



DBNs vs. HMMs

Every HMM is a single-variable DBN; every discrete DBN is an HMM





Consider the **transition model**

Sparse dependencies $\Rightarrow$ exponentially fewer parameters;

e.g., 20 state variables, three parents each

DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$

DBN – Special Cases

- Hidden Markov Model (HMMs):

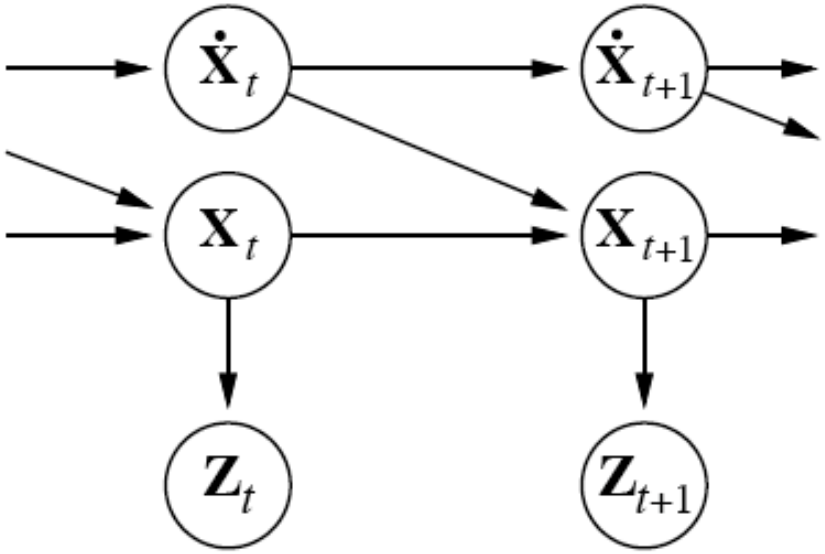
  Temporal probabilistic model in which the state of the process
  is described by a single discrete random variable. (The simplest kind of DBN)

- Kalman Filter Models (KFMs):

  Estimate the state of a physical system from noisy observations over time.
  Also known as linear dynamical systems (LDSs).

Kalman Filters

Modelling systems described by a set of continuous variables,
e.g., tracking a bird flying—$\mathbf{X}_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$.
Airplanes, robots, ecosystems, economies, chemical plants, planets, ...



Gaussian prior, linear Gaussian transition model and sensor model

Updating Gaussian Distributions

Prediction step: if $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ is Gaussian, then prediction

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

is Gaussian. If $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$ is Gaussian, then the updated distribution

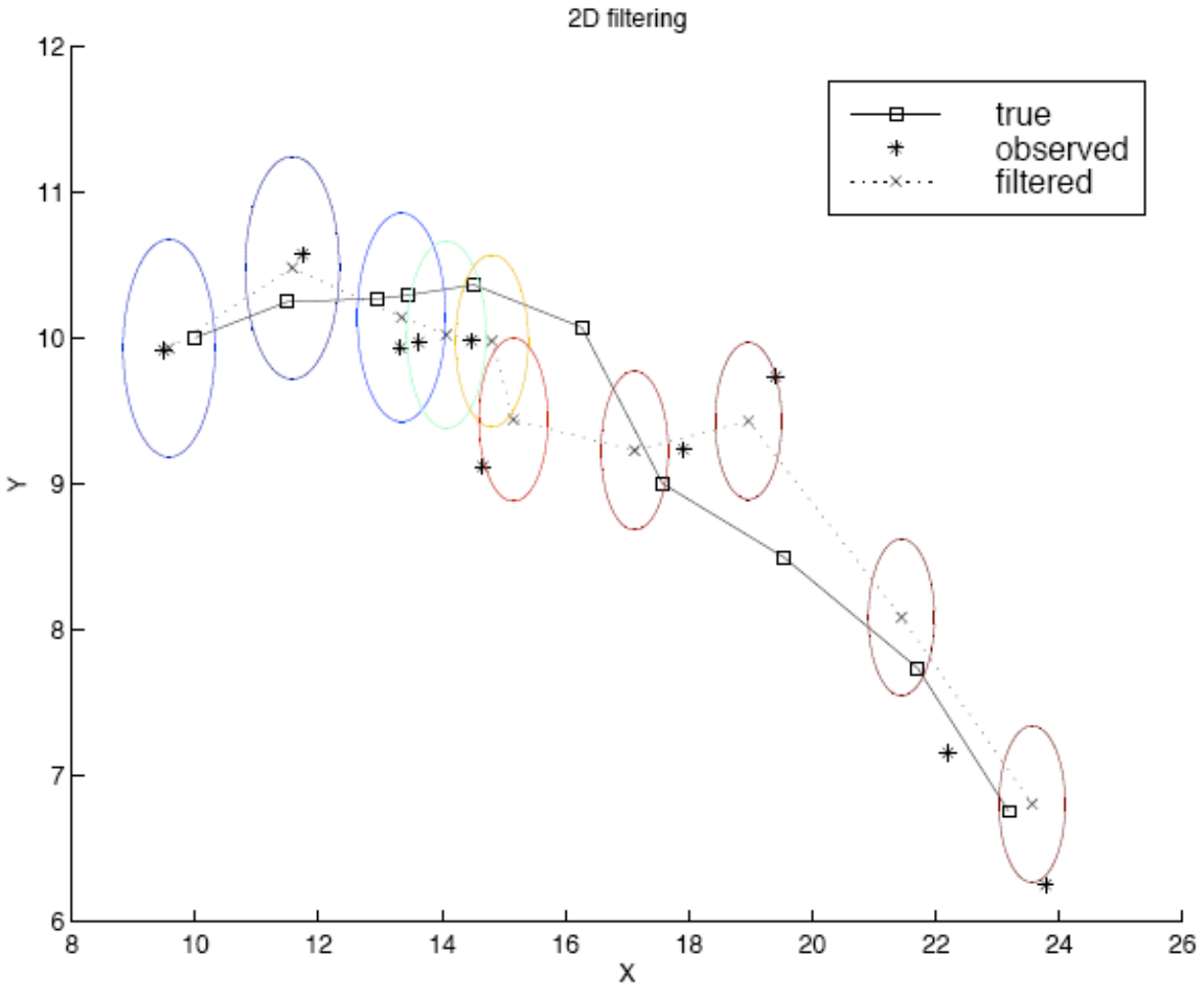
$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

is Gaussian

Hence $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ is multivariate Gaussian $N(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ for all t

General (nonlinear, non-Gaussian) process: description of posterior grows **unboundedly** as $t \to \infty$

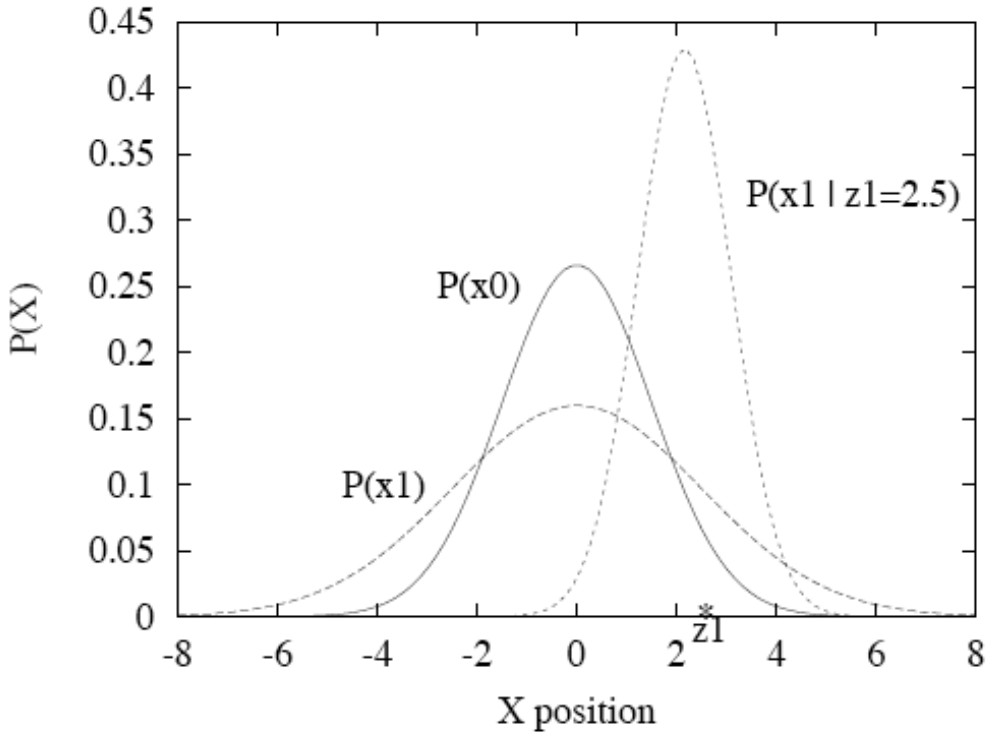
2-D Tracking: Filtering



Simple 1-D Example

Gaussian random walk on X–axis, s.d. σ_x, sensor s.d. σ_z

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



z1: first observation

s.d.= standard deviation

General Kalman Update

Transition and sensor models:

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

$\mathbf{F}$ is the matrix for the transition; $\boldsymbol{\Sigma}_x$ the transition noise covariance
$\mathbf{H}$ is the matrix for the sensors; $\boldsymbol{\Sigma}_z$ the sensor noise covariance

Filter computes the following update:

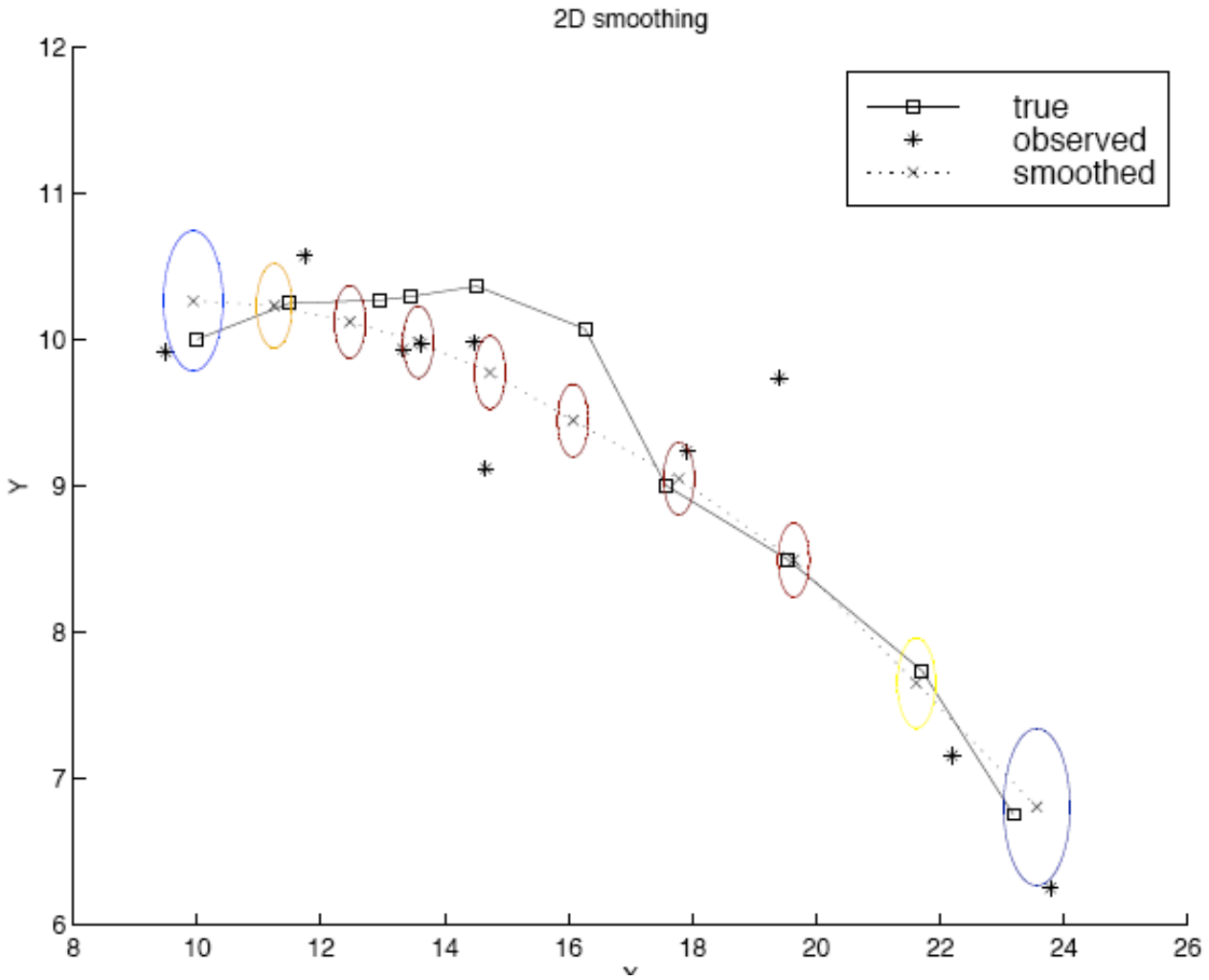
Details left for your studies

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

where $\mathbf{K}_{t+1} = (\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top + \boldsymbol{\Sigma}_z)^{-1}$
is the Kalman gain matrix

$\boldsymbol{\Sigma}_t$ and $\mathbf{K}_t$ are independent of observation sequence, so compute offline

2-D Tracking: Smoothing

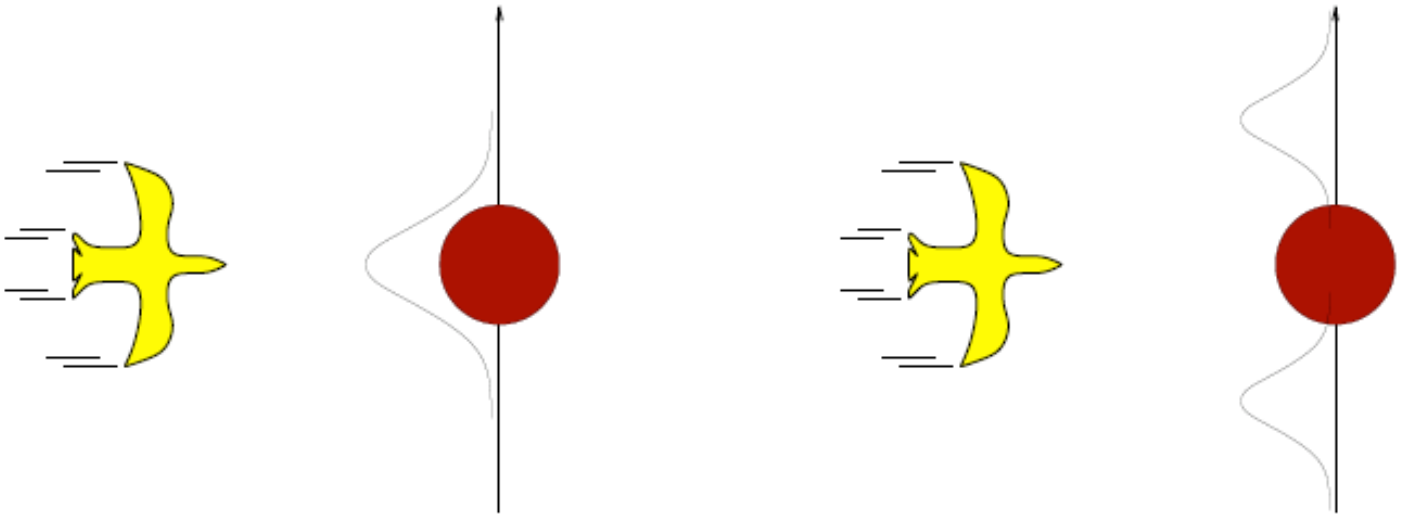


Where it breaks

Cannot be applied if the transition model is nonlinear

Extended Kalman Filter models transition as locally linear around $\mathbf{x}_t = \boldsymbol{\mu}_t$
Fails if systems is locally unsmooth

Standard solution: switching Kalman filter



Keeping track of many objects: Identity uncertainty

Non-Standard-Datenbanken

Dynamische Bayessche Netze

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Learning Dynamic Bayesian Networks

- *Learning* requires the full smoothing inference, rather than filtering, because it provides better estimates of the state of the process.

- Learning the parameters of a BN is done using Expectation – Maximization (EM) Algorithms.
  - Iterative optimization method to estimate some unknown parameters.

State View: Hidden Markov models

Set of states: $\{s_1, s_2, \ldots, s_N\}$

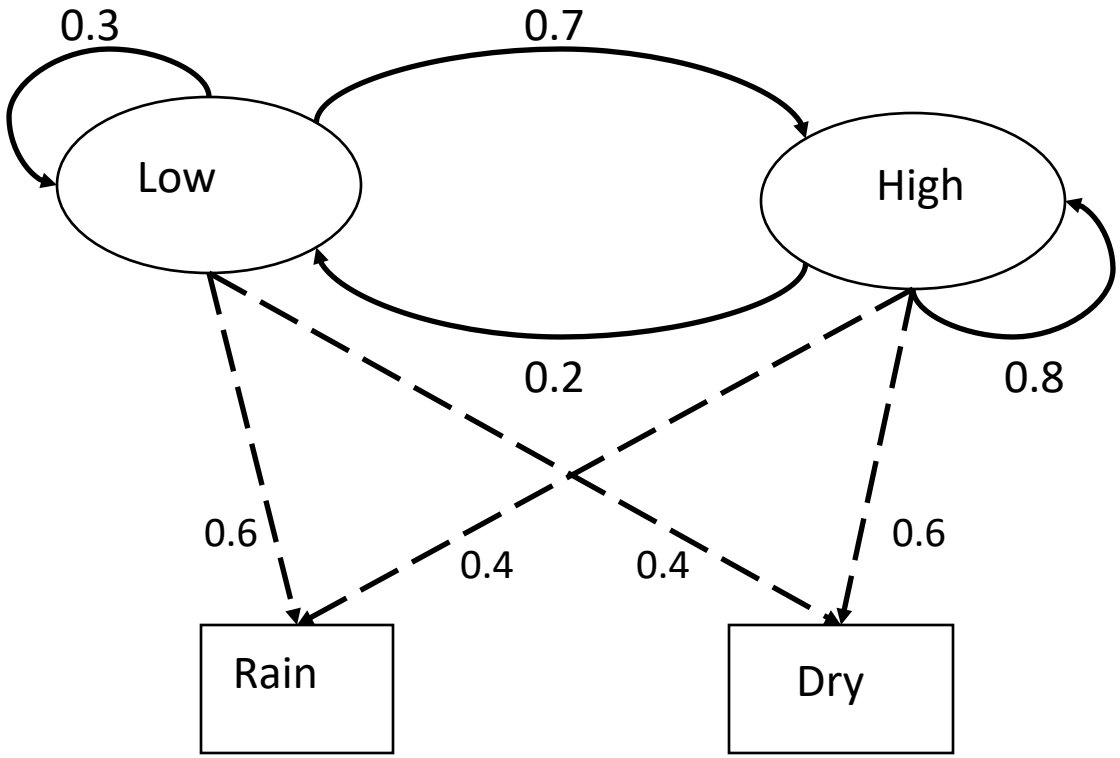
Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$
Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

States are not visible, but each state generates one of M different observations

$$\{v_1, v_2, \ldots, v_M\}$$

Example of Hidden Markov Model



State View: Hidden Markov models

To define a hidden Markov model, the following probabilities have to be specified:

- Matrix of transition probabilities $A=(a_{ij})$, $a_{ij}= P(s_j \mid s_i)$,
- Matrix of observation probabilities $B=(b_i (v_m))$, $b_i(v_m) = P(v_m \mid s_i)$ and a
- Vector of initial probabilities $\pi=(\pi_i)$, $\pi_i = P(s_i)$.

Model is represented by (A, B, π).

State View: Learning problem (1)

Given some training observation sequences $O=o_1 o_2 \ldots o_t$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters (A, B, π) that best fit training data, i.e., that is maximizes $P(O \mid A, B, \pi)$.

State View: Learning problem (2)

If training data has information about sequence of hidden states, then use maximum likelihood estimation of parameters:

$$a_{ij} = P(s_j \mid s_i) = \frac{\text{Number of transitions from state } S_i \text{ to state } S_j}{\text{Number of transitions out of state } S_i}$$

$$b_i(v_m) = P(v_m \mid s_i) = \frac{\text{Number of times observation } V_m \text{ occurs in state } S_i}{\text{Number of times in state } S_i}$$

Otherwise: Use iterative expectation-maximization algorithm to find local maximum of $P(O \mid A, B, \pi)$: Baum-Welch Algorithm

Alternative: Viterbi Path Counting Algorithm

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss, A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains, Ann. Math. Statist. Volume 41, Number 1, 164-171, **1970**

Leonard E. Baum and Ted Petrie, Statistical Inference for Probabilistic Functions of Finite State Markov Chains, Ann. Math. Statist. Volume 37, Number 6, 1554-1563, **1966**.

Baum-Welch algorithm

General idea:

$$a_{ij} = P(s_j \mid s_i) = \frac{\text{Expected number of transitions from state } S_i \text{ to state } S_j}{\text{Expected number of transitions out of state } S_i}$$

$$b_i(v_m) = P(v_m \mid s_i) = \frac{\text{Expected number of times observation } V_m \text{ occurs in state } S_i}{\text{Expected number of times in state } S_i}$$

$\pi_i = P(s_i) =$ Expected frequency in state S_i at time $k=1$.

Baum-Welch algorithm: Expectation step(1)

Define variable $\xi_k(i,j)$ as the probability of being in state S_i at time k and in state S_j at time $k+1$, given the observation sequence $O_1 O_2 \ldots O_T$ with $k < t$

$$\xi_k(i,j) = P(X_k = s_i , X_{k+1} = s_j \mid o_1 o_2 \ldots o_t)$$

$$\xi_k(i,j) = \frac{P(X_k = s_i , X_{k+1} = s_j , o_1 o_2 \ldots o_t)}{P(o_1 o_2 \ldots o_t)} =$$

$$\frac{P(X_k = s_i , o_1 o_2 \ldots o_k)\ a_{ij}\ b_j(o_{k+1})\ P(o_{k+2} \ldots o_t \mid X_{k+1} = s_j)}{P(o_1 o_2 \ldots o_t)} =$$

$$\alpha\ \text{forward}_k(i)\ a_{ij}\ b_j(o_{k+1})\ \text{backward}_{k+1}(j)$$

Baum-Welch algorithm: Expectation step(2)

Define variable $\gamma_k(i)$ as the probability of being in state $X_k=S_i$ at time k, given the observation sequence $O_1 O_2 \ldots O_t$.

$$\gamma_k(i) = P(X_k = s_i \mid O_1 O_2 \ldots O_t)$$

$$\gamma_k(i) = \frac{P(X_k = s_i , O_1 O_2 \ldots O_t)}{P(O_1 O_2 \ldots O_t)} =$$

$$\alpha \; \text{forward}_k(i) \; \text{backward}_k(i)$$

Baum-Welch algorithm: Expectation step(3)

We calculated $\xi_k(i,j) = P(X_k = s_i , X_{k+1} = s_j \mid o_1 o_2 \ldots o_t)$

and $\gamma_k(i) = P(X_k = s_i \mid o_1 o_2 \ldots o_t)$

Expected number of transitions from state S_i to state S_j =

$= \sum_k \xi_k(i,j)$

Expected number of transitions out of state S_i = $\sum_k \gamma_k(i)$

Expected number of times observation V_m occurs in state S_i =

$= \sum_k \gamma_k(i)$, k is such that $O_k = V_m$

Expected frequency in state S_i at time k=1 : $\gamma_1(i)$.

Baum-Welch algorithm: Maximization step

$$a_{ij} = \frac{\text{Expected number of transitions from state } S_j \text{ to state } S_i}{\text{Expected number of transitions out of state } S_j} = \frac{\sum_k \xi_k(i,j)}{\sum_k \gamma_k(i)}$$

$$b_i(v_m) = \frac{\text{Expected number of times observation } v_m \text{ occurs in state } S_i}{\text{Expected number of times in state } S_i} = \frac{\sum_k \xi_k(i,j)}{\sum_{k,o_k = v_m} \gamma_k(i)}$$

$$\pi_i = (\text{Expected frequency in state } S_i \text{ at time } k=1) = \gamma_1(i).$$

Learning (1)

- The techniques for learning DBN are mostly straightforward extensions of the techniques for learning BNs
- Parameter learning
  - The transition model $P(X_t \mid X_{t-1})$ / The observation model $P(Y_t \mid X_t)$
  - Offline learning
    - Parameters must be tied across time-slices
    - The initial state of the dynamic system can be learned independently of the transition matrix
  - Online learning
    - Add the parameters to the state space and then do online inference (filtering)
  - The usual criterion is maximum-likelihood(ML)

- The goal of parameter learning is to compute
  - $\theta^*_{ML} = \text{argmax}_\theta P(Y \mid \theta) = \text{argmax}_\theta \log P(Y \mid \theta)$
  - $\theta^*_{MAP} = \text{argmax}_\theta \log P(Y \mid \theta) + \log P(\theta)$
  - Two standard approaches: gradient ascent and EM(Expectation Maximization)

Learning (2)

- Structure learning
  - Intra-slice connectivity: Structural EM
  - Inter-slice connectivity:
    For each node in slice t, we must choose its parents from slice t-1
  - Given structure is unrolled to a certain extent,
    the inter-slice connectivity is identical for all pairs of slices:
    - Constraints on Structural EM

Summary

Temporal models use state and sensor variables replicated over time

Markov assumptions and stationarity assumption, so we need
- transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$
- sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$

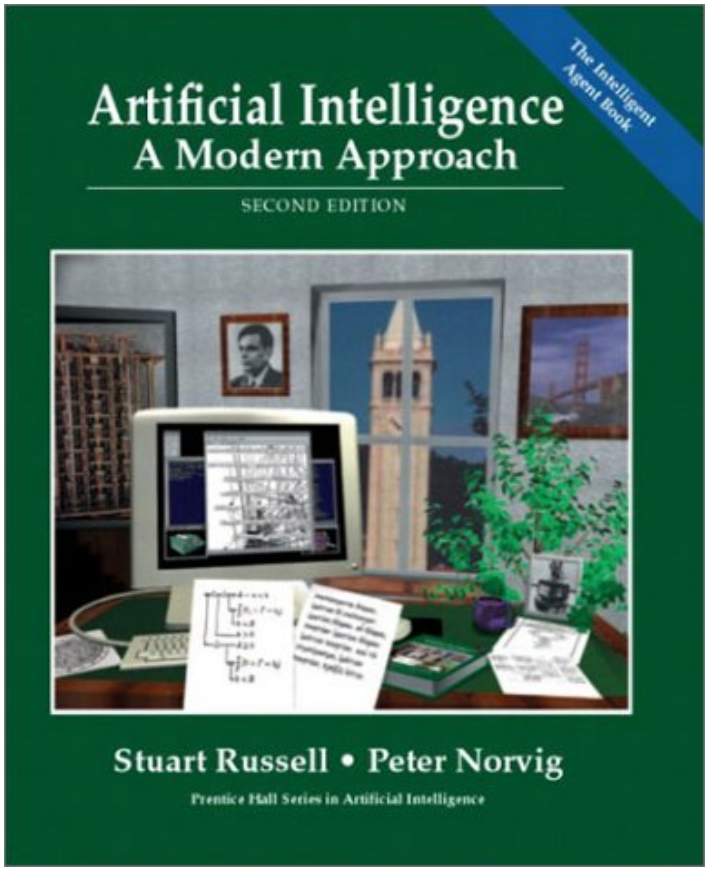
Tasks are filtering, prediction, smoothing, most likely sequence;
all done recursively with constant cost per time step

Hidden Markov models have a single discrete state variable; used for speech recognition

Kalman filters allow n state variables, linear Gaussian, $O(n^3)$ update

Dynamic Bayes nets subsume HMMs, Kalman filters; exact update intractable

Literature



http://aima.cs.berkeley.edu