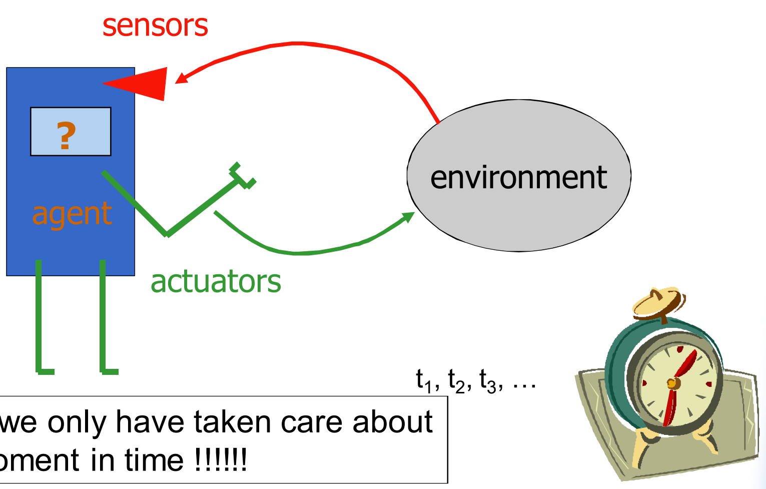
Intelligent Autonomous Agents and Cognitive Robotics

Topic 6: Probabilistic Reasoning over Time (Dynamic Bayesian Networks)

Ralf Möller, Rainer Marrone
Hamburg University of Technology

1

Temporal Probabilistic Agent



So far we only have taken care about one moment in time !!!!!!

2

Time and Uncertainty

- The world changes over time, we need to track and predict it
- Examples:
  diabetes management, localization, speech recognition, …
- Basic idea: copy state and evidence variables for each time step
- **X**$_t$ – set of unobservable state variables at time t
  - e.g., BloodSugar$_t$, StomachContents$_t$, …
- **E**$_t$ – set of evidence variables at time t
  - e.g., MeasuredBloodSugar$_t$, PulseRate$_t$, FoodEaten$_t$,…
- Assumes discrete time steps

3

Dynamic Bayesian Networks

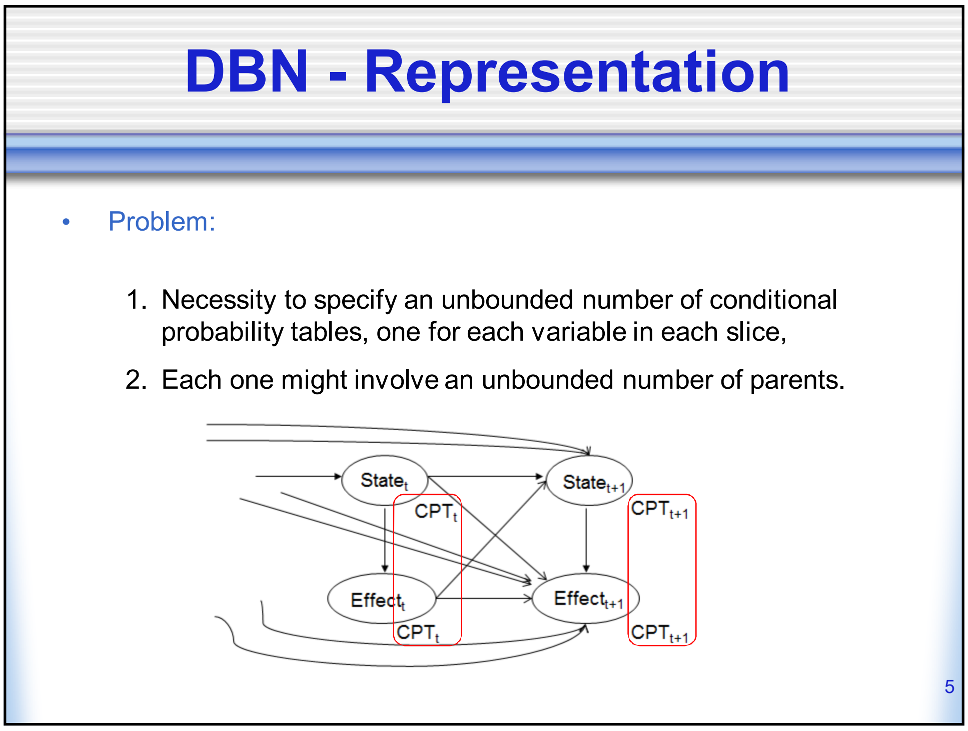
- How can we model *dynamic* situations with a Bayesian network?
- Example: *Is it raining today?*

$$X_t = \{R_t\}$$
$$E_t = \{U_t\}$$

➡ next step: specify dependencies among the variables.

The term "dynamic" means we are modeling a dynamic system, not that the network structure changes over time.

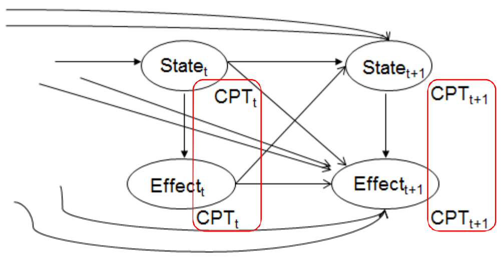
4



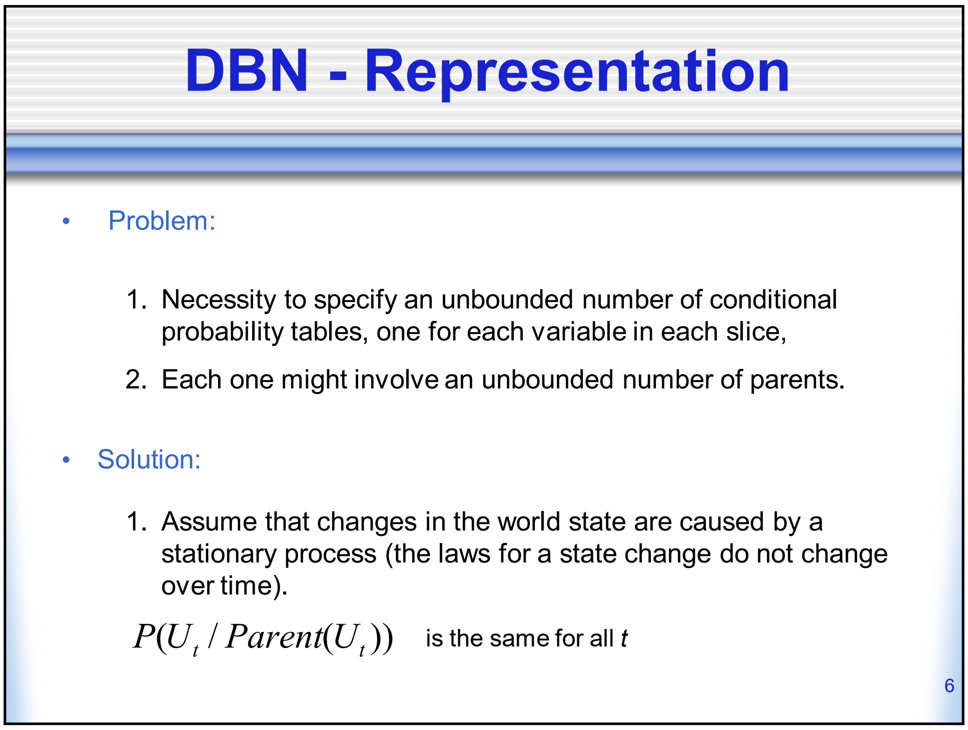
DBN - Representation

- Problem:

  1. Necessity to specify an unbounded number of conditional probability tables, one for each variable in each slice,
  2. Each one might involve an unbounded number of parents.



5



DBN - Representation

- Problem:

  1. Necessity to specify an unbounded number of conditional probability tables, one for each variable in each slice,
  2. Each one might involve an unbounded number of parents.

- Solution:

  1. Assume that changes in the world state are caused by a stationary process (the laws for a state change do not change over time).

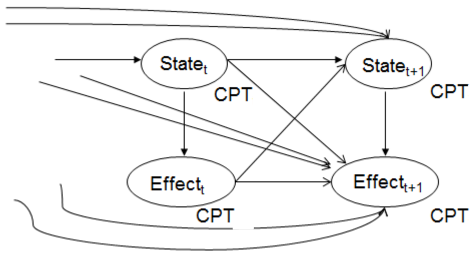
$P(U_t / Parent(U_t))$ is the same for all *t*

6

DBN - Representation

- Problem:

1. Necessity to specify an unbounded number of conditional probability tables, one for each variable in each slice →solved
2. Each one might involve an unbounded number of parents.



DBN - Representation

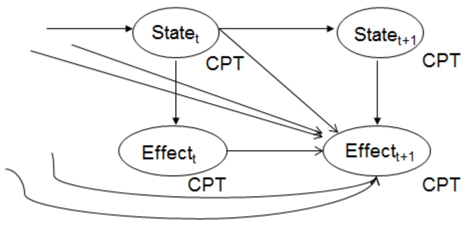
- Solution cont.:

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

   Using the first-order Markov process:

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

Transition Model



DBN - Representation

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

     Using the first-order Markov process:

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

9

DBN - Representation

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

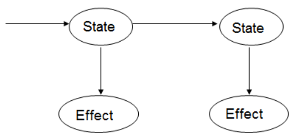
     Using the first-order Markov process:

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

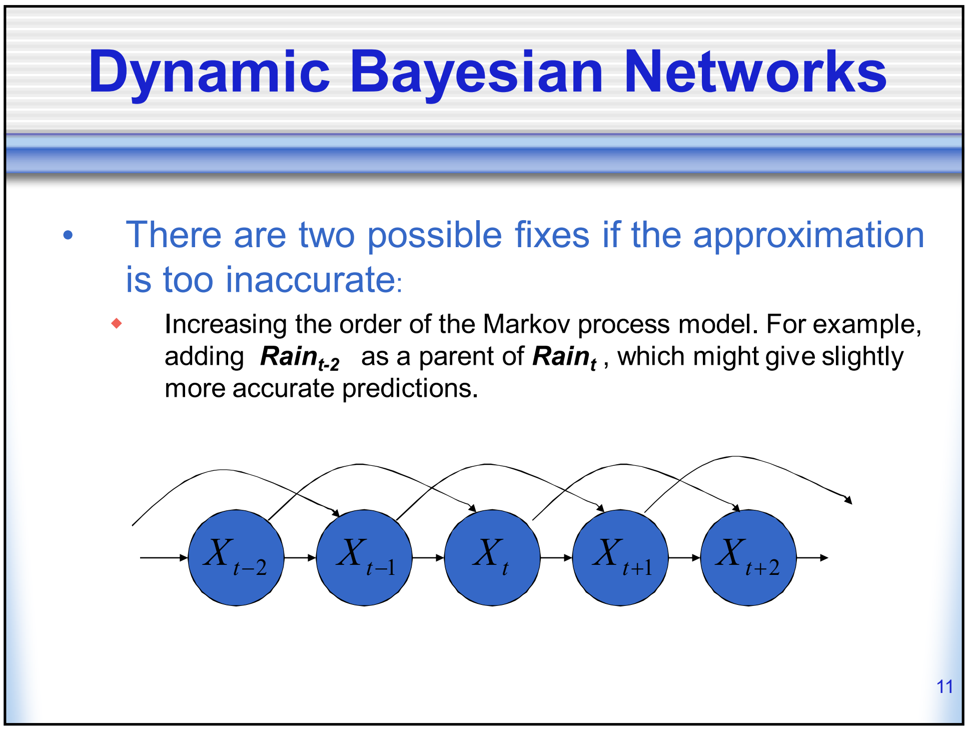
In addition to restricting the parents of the state variable X_t, we must restrict the parents of the evidence variable E_t

Sensor Model

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

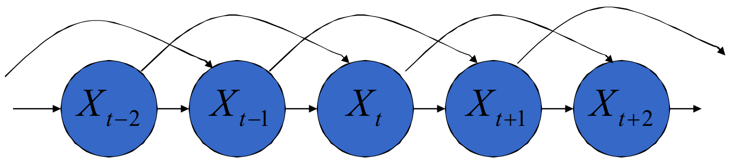


10

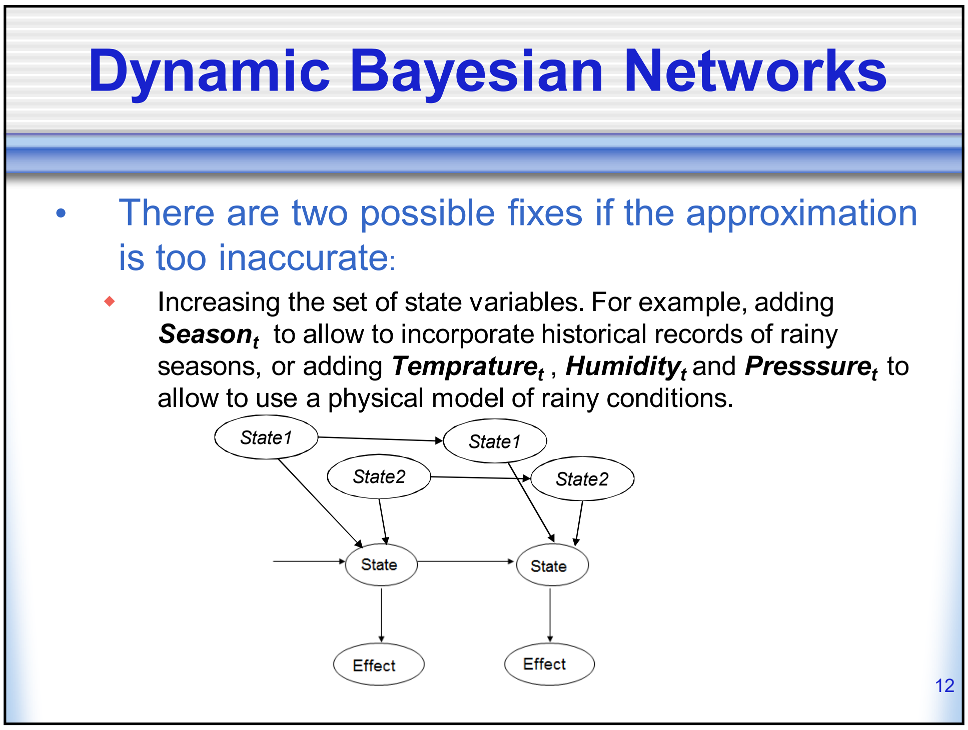


Dynamic Bayesian Networks

- There are two possible fixes if the approximation is too inaccurate:
  - Increasing the order of the Markov process model. For example, adding ***Rain***$_{t-2}$ as a parent of ***Rain***$_t$, which might give slightly more accurate predictions.

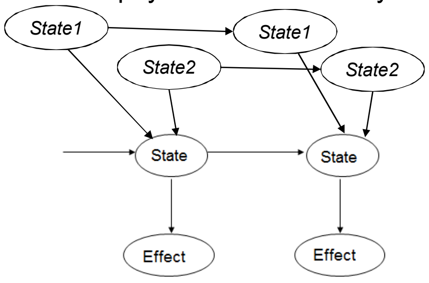


11



Dynamic Bayesian Networks

- There are two possible fixes if the approximation is too inaccurate:
  - Increasing the set of state variables. For example, adding ***Season***$_t$ to allow to incorporate historical records of rainy seasons, or adding ***Temprature***$_t$, ***Humidity***$_t$ and ***Presssure***$_t$ to allow to use a physical model of rainy conditions.



12

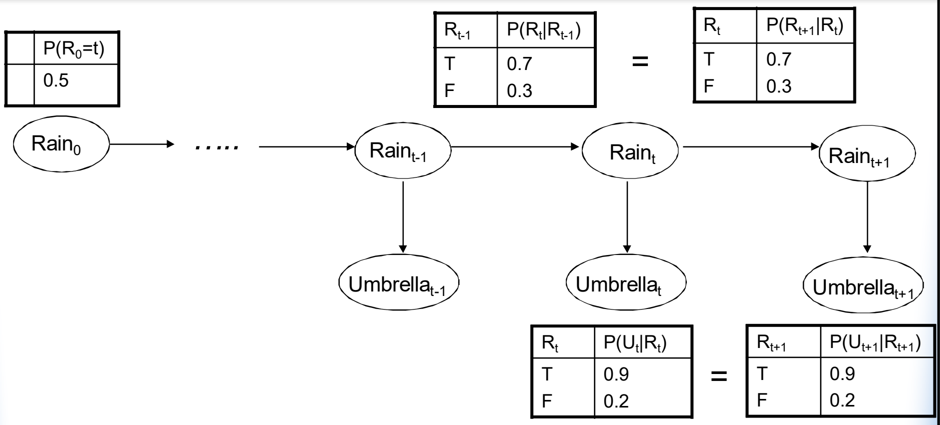
Complete Joint Distribution

- 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:

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

13

Simple Example



	$P(R_0=t)$
	0.5

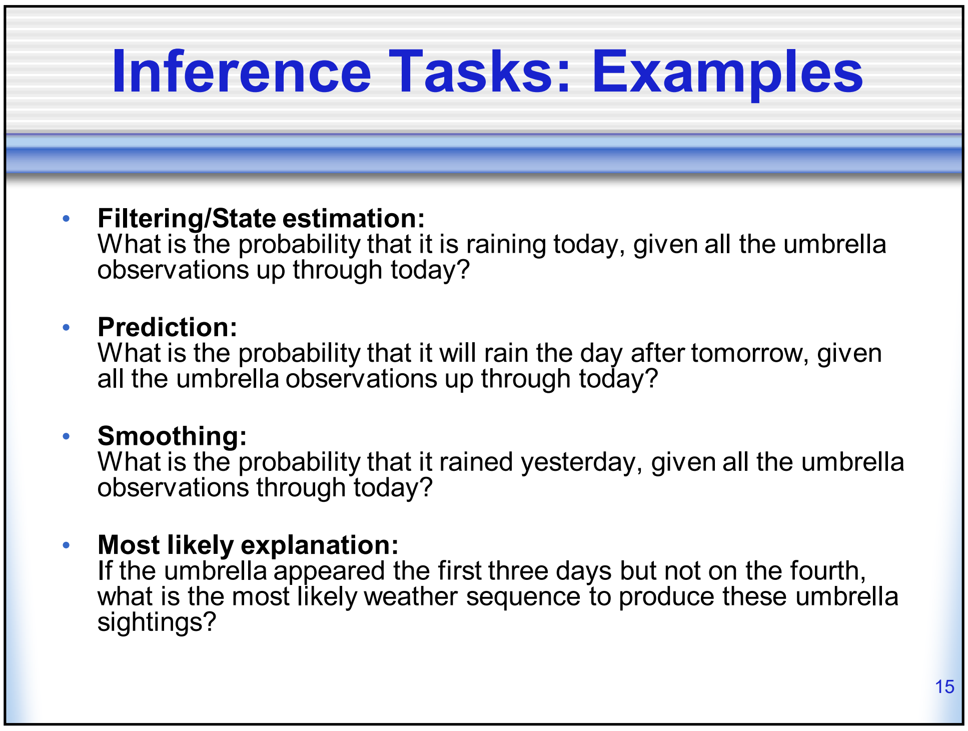
R_{t-1}	$P(R_t\|R_{t-1})$
T	0.7
F	0.3

R_t	$P(R_{t+1}\|R_t)$
T	0.7
F	0.3

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

R_{t+1}	$P(U_{t+1}\|R_{t+1})$
T	0.9
F	0.2

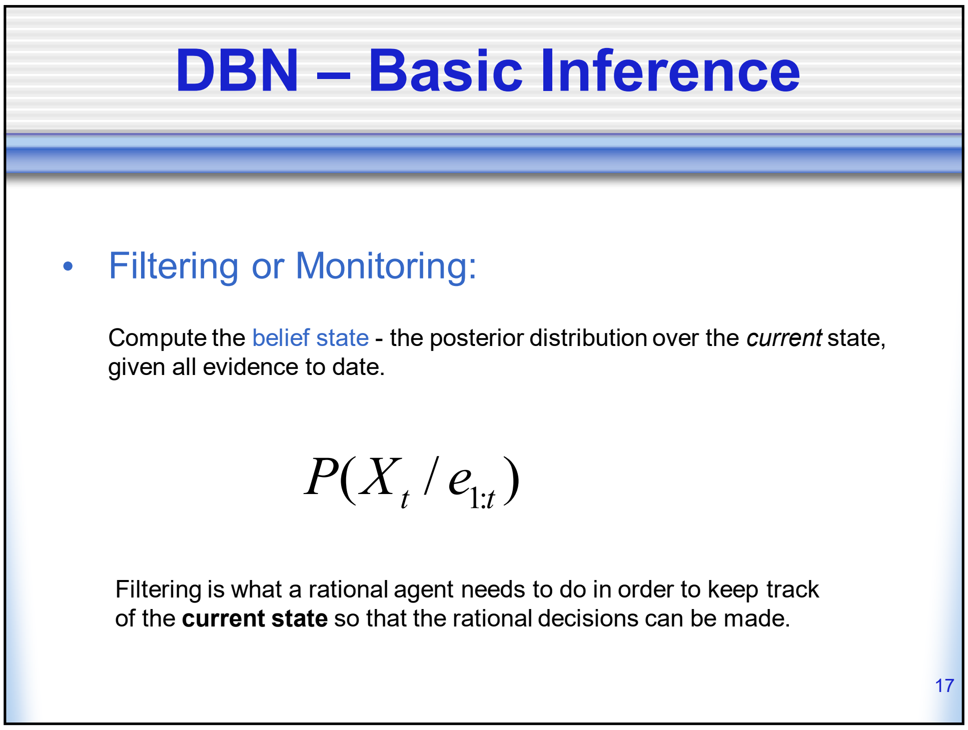
14



Inference Tasks: Examples

- **Filtering/State estimation:**
  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:**
  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?

15



DBN – Basic Inference

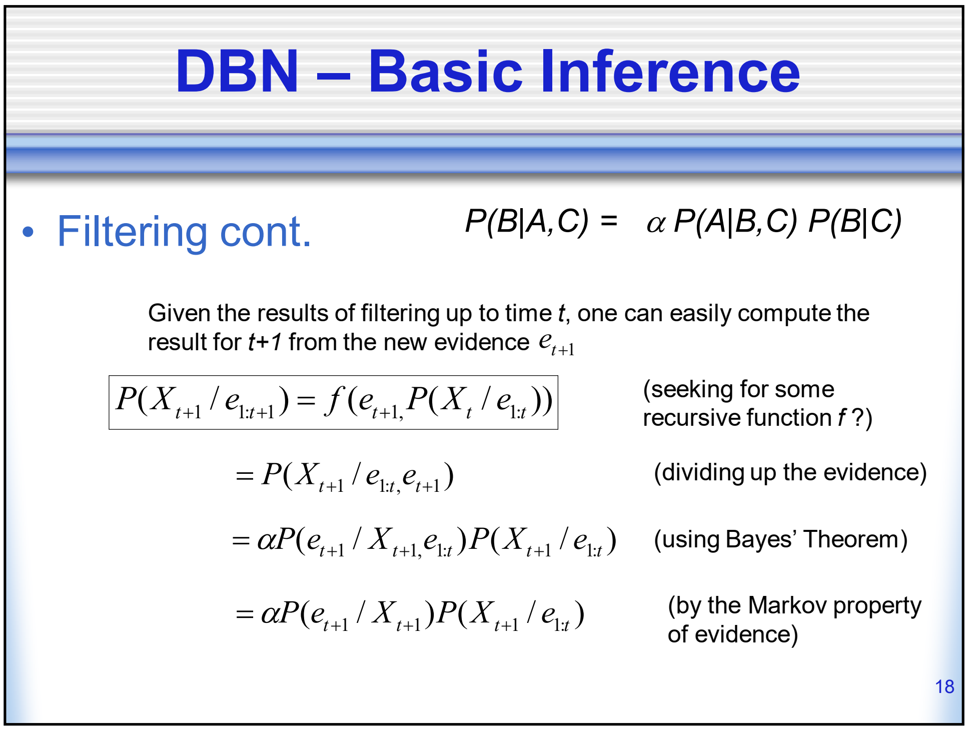
- Filtering or Monitoring:

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

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

Filtering is what a rational agent needs to do in order to keep track of the **current state** so that the rational decisions can be made.

17



DBN – Basic Inference

- Filtering cont.

P(B|A,C) = α P(A|B,C) P(B|C)

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}

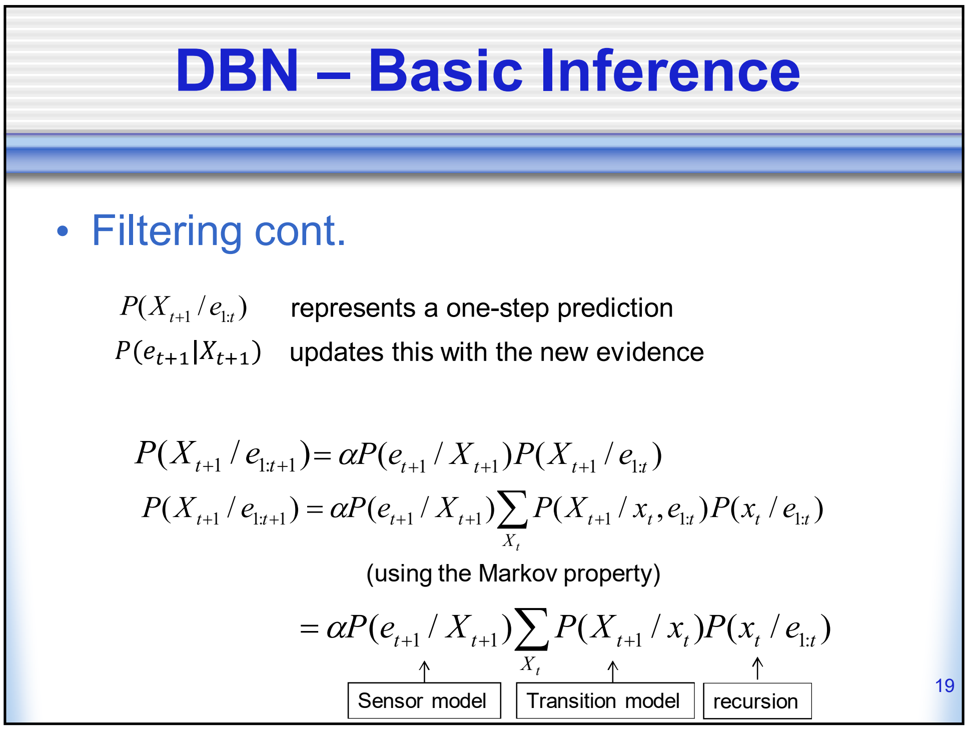
$$\boxed{P(X_{t+1}/e_{1:t+1}) = f(e_{t+1,}P(X_t/e_{1:t}))}$$ (seeking for some recursive function *f* ?)

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

18



DBN – Basic Inference

- Filtering cont.

$P(X_{t+1}/e_{1:t})$ represents a one-step prediction

$P(e_{t+1}|X_{t+1})$ updates this with the new evidence

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

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

(using the Markov property)

$$= \alpha \underbrace{P(e_{t+1}/X_{t+1})}_{\text{Sensor model}}\sum_{X_t} \underbrace{P(X_{t+1}/x_t)}_{\text{Transition model}}\underbrace{P(x_t/e_{1:t})}_{\text{recursion}}$$

19

DBN – Basic Inference

For two steps in the Umbrella example: $$= \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_t) P(x_t / e_{1:t})$$

- 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

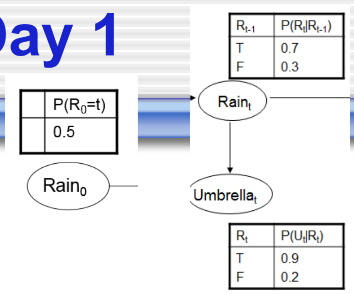
$$P(R_2 / u_1) = \sum_{r_1} P(R_2 / r_1) P(r_1 / u_1)$$

and updating it with the evidence for t=2 gives

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

20

Example: Day 1



evidence prediction

$$P(R_1 | u_1) = P(u_1 / R_1) \sum_{r_0} P(R_1 / r_0) P(r_0)$$

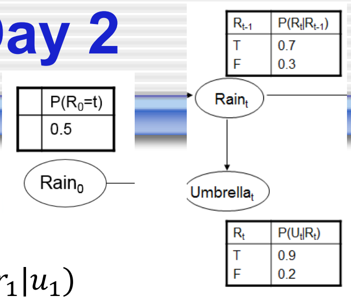
$P(R_0)$=<0.5,0.5>

$P(R_1) = \sum_{r_0} P(R_1|r_0)\, P(r_0)$=<0.7,0.3>0.5+<0.3,0.7>0.5=<0.5,0.5>

$$P(R_1|u_1) = \alpha P(u_1|R_1)P(R_1) = \alpha < 0.9,0.2 >< 0.5,0.5 >$$
$$= \alpha < 0.45,01 > \approx < 0.818,0.182 >$$

21

Example: Day 2



evidence prediction

$$P(R_2|u_1,u_2) = \alpha P(u_2|R_2) \sum_{r_1} P(R_2|r_1)P(r_1|u_1)$$

$$P(R_1|u_1) \approx <0.818, 0.182>$$

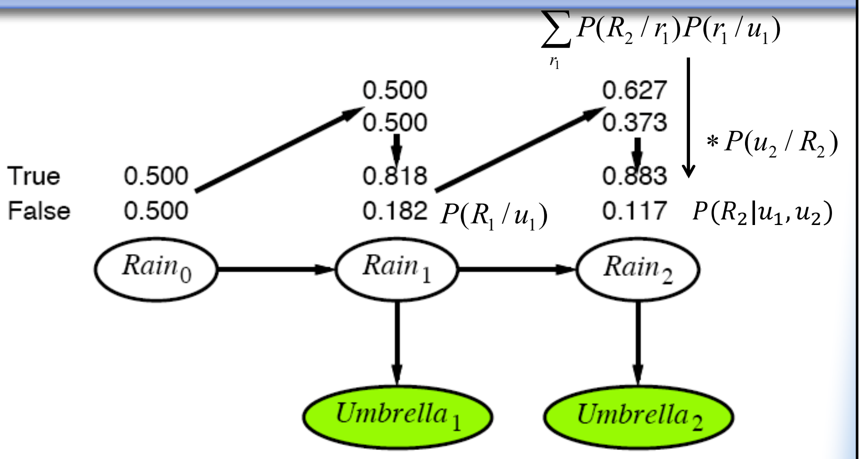
$P(R_2|u_1) = \sum_{r_1} P(R_2|r_1)\, P(r_1|u_1)$=

<0.7,0.3>0.818+<0.3,0.7>0.182 ≈ <0.627,0.373>

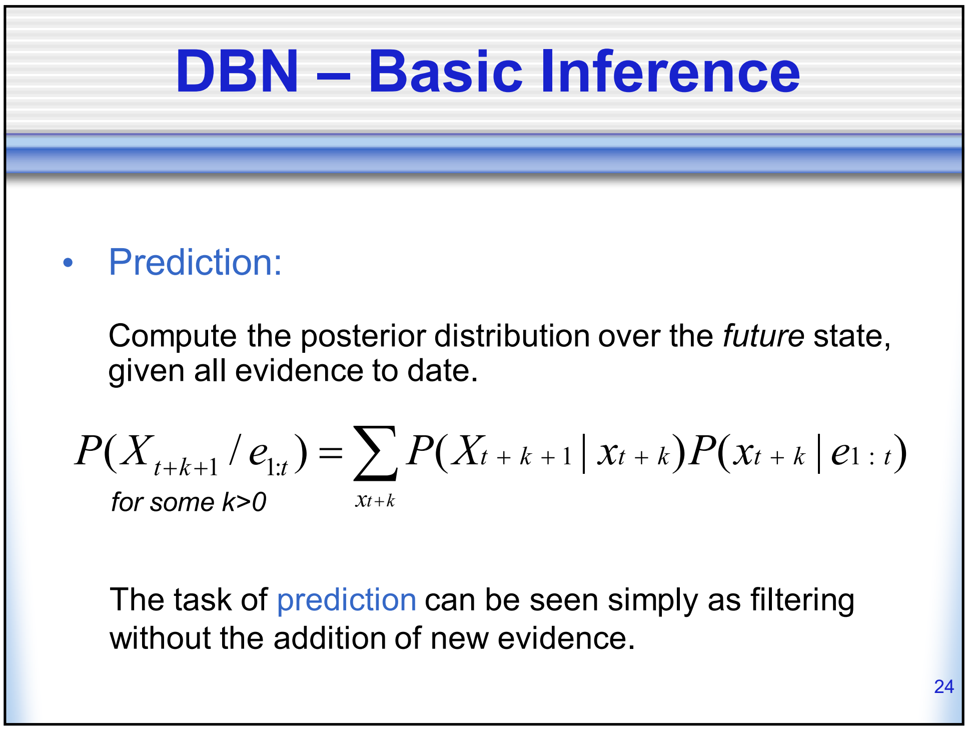
$$P(R_2|u_1,u_2) = \alpha P(u_2|R_2)P(R_2|u_1) = \alpha <0.9,0.2><0.627,0.373>$$
$$= \alpha <0.565,0075> \approx <0.883,0.117>$$

22

Example



23



DBN – Basic Inference

- Prediction:

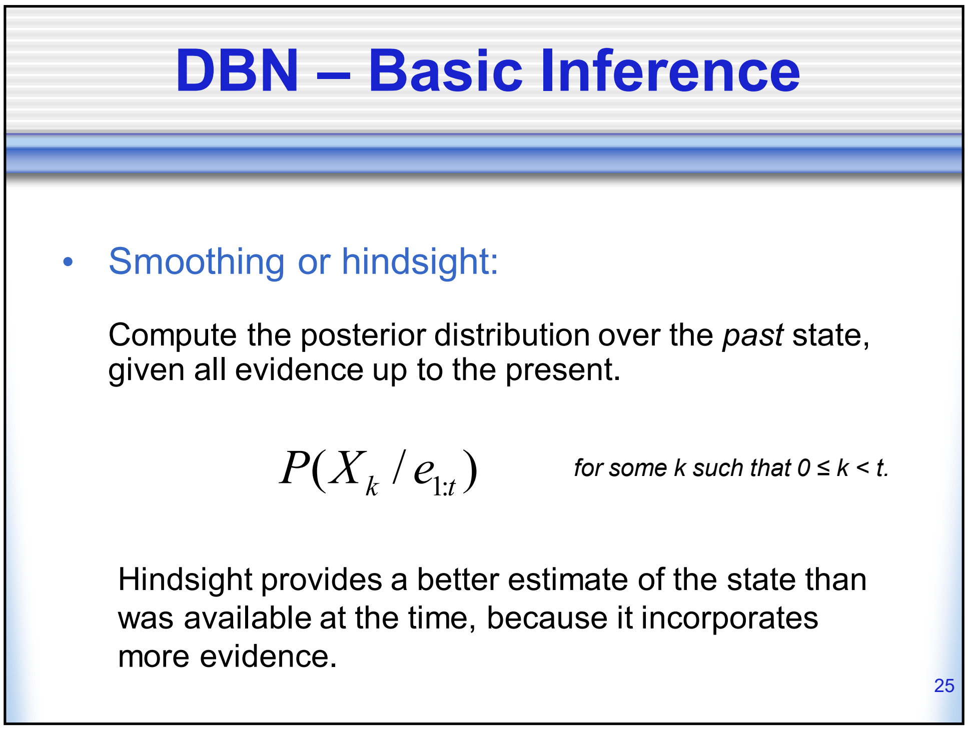
  Compute the posterior distribution over the *future* state, given all evidence to date.

$$P(X_{t+k+1} / e_{1:t}) = \sum_{x_{t+k}} P(X_{t+k+1} \mid x_{t+k}) P(x_{t+k} \mid e_{1:t})$$

for some k>0

The task of prediction can be seen simply as filtering without the addition of new evidence.

24



DBN – Basic Inference

- Smoothing or hindsight:

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

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

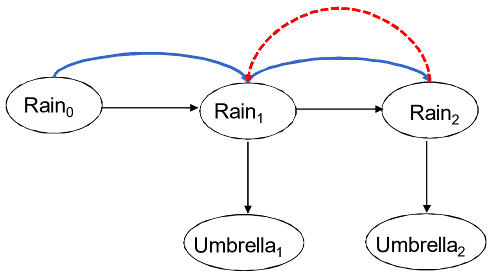
for some k such that 0 ≤ k < t.

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

25

Smoothing

- Can I use future information to increase the accuracy of filtering for past states?



Umbrella$_1$=t Umbrella$_2$=t

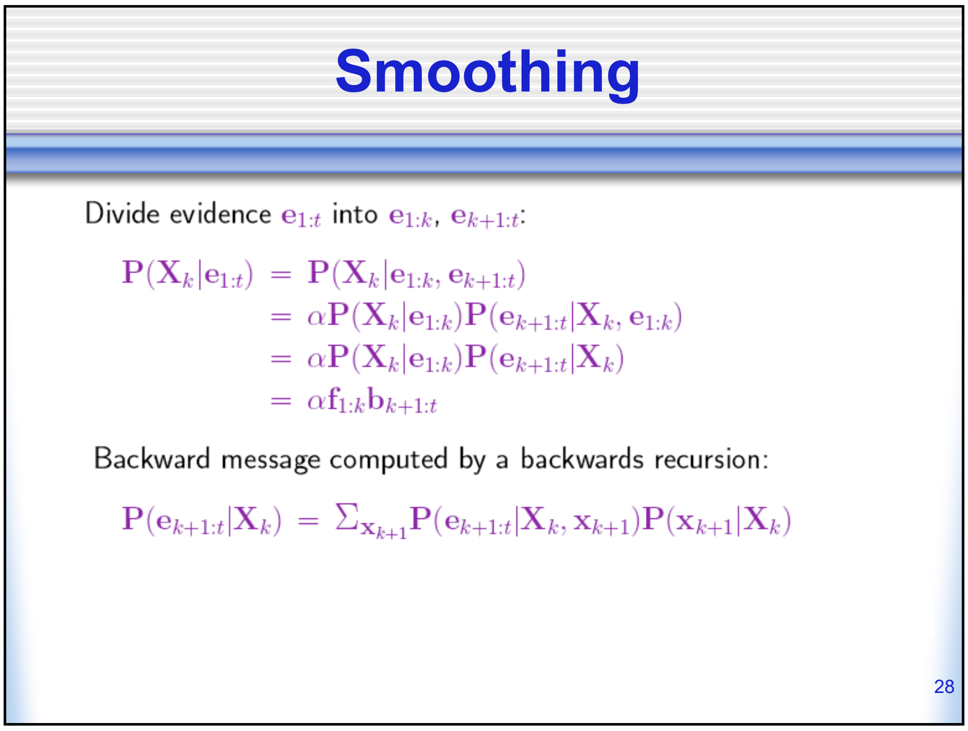
26

Smoothing

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

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

27



Smoothing

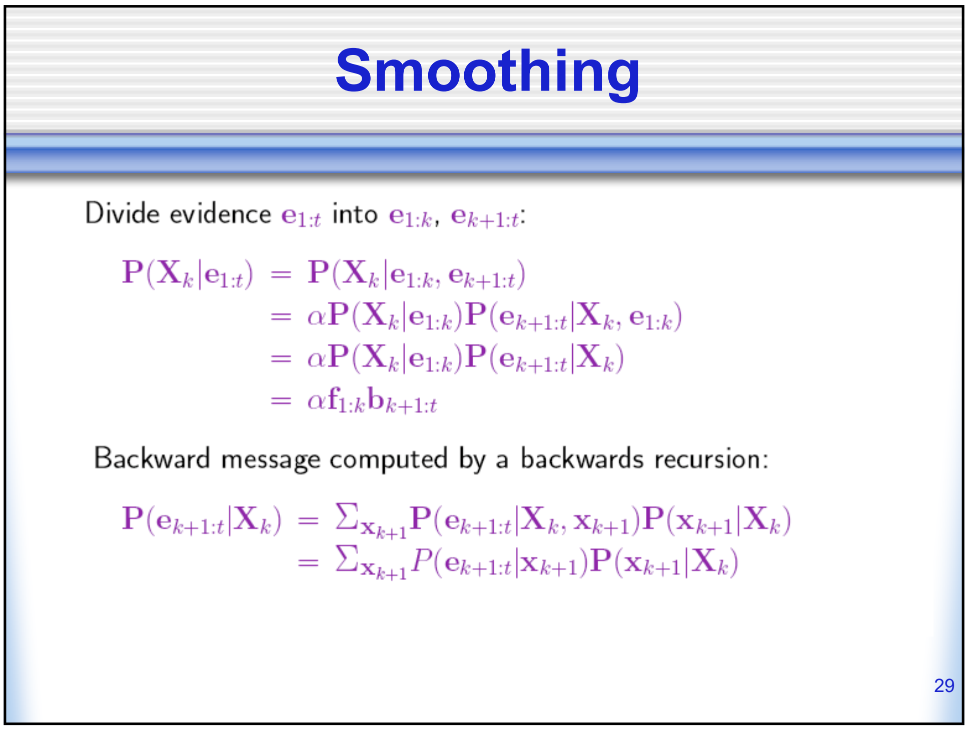
Divide evidence $\mathbf{e}_{1:t}$ into $\mathbf{e}_{1:k}$, $\mathbf{e}_{k+1:t}$:

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

Backward message computed by a backwards recursion:

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

28



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:

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

29

Smoothing

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

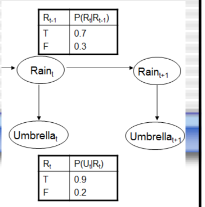
$$\begin{aligned}
\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{e}_{1:k}) \\
&= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)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

Backward message computed by a backwards recursion:

$$\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \Sigma_{\mathbf{x}_{k+1}}\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \Sigma_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \Sigma_{\mathbf{x}_{k+1}}\underbrace{P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})}_{\boxed{\text{Sensor model}}}\underbrace{P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})}_{\boxed{\text{recursion}}}\underbrace{\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)}_{\boxed{\text{Transition model}}}
\end{aligned}$$

30

Example



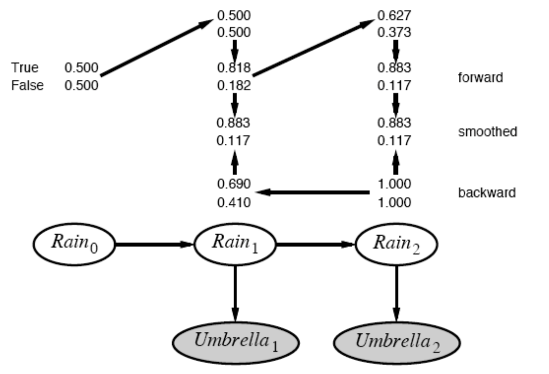
R_{t-1}	$P(R_t\|R_{t-1})$
T	0.7
F	0.3

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

- Smoothed estimate for rain at k=1, given u_1, u_2.
  $\mathbf{P}(R_1|u_1,u_2) = \alpha\ \mathbf{P}(R_1|u_1)\mathbf{P}(u_2|R_1)$
- The first term is taken from the forward example
  <0.818, 0.182>
- $\mathbf{P}(u_2|R_1) = \Sigma_{r2}\ P(u_2|r_2)P(|r_2)\ \mathbf{P}(r_2|R_1)$
  = (0.9 x 1 x <0.7,0.3>)+(0.2 x 1 x <0.3,0.7>)
  = <0.69, 0.41>
- $\mathbf{P}(R_1|u_1,u_2) = \alpha$ <0.818, 0.182> x <0.69. 0.41>
  $\approx$ <0.883, 0.117>
- If we do it for each time slice $O(t^2)$!!!

31

Example contd.



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

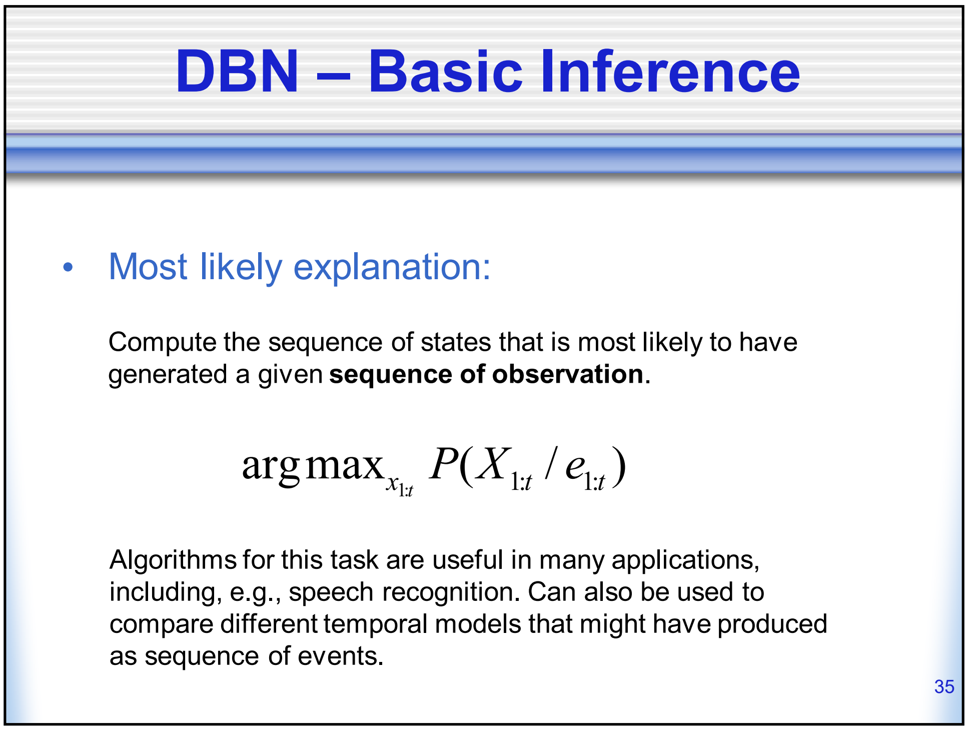
32

Forward-Backward Algorithm

```
function FORWARD-BACKWARD(ev, prior) returns a vector of probability distributions
  inputs: ev, a vector of evidence values for steps $1, \ldots, t$
          prior, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$
  local variables: fv, a vector of forward messages for steps $0, \ldots, t$
                   b, a representation of the backward message, initially all 1s
                   sv, a vector of smoothed estimates for steps $1, \ldots, t$

  fv[0] ← prior
  for $i = 1$ to $t$ do
      fv[i] ← FORWARD(fv[i − 1], ev[i])
  for $i = t$ downto 1 do
      sv[i] ← NORMALIZE(fv[i] × b)
      b ← BACKWARD(b, ev[i])
  return sv
```

33



DBN – Basic Inference

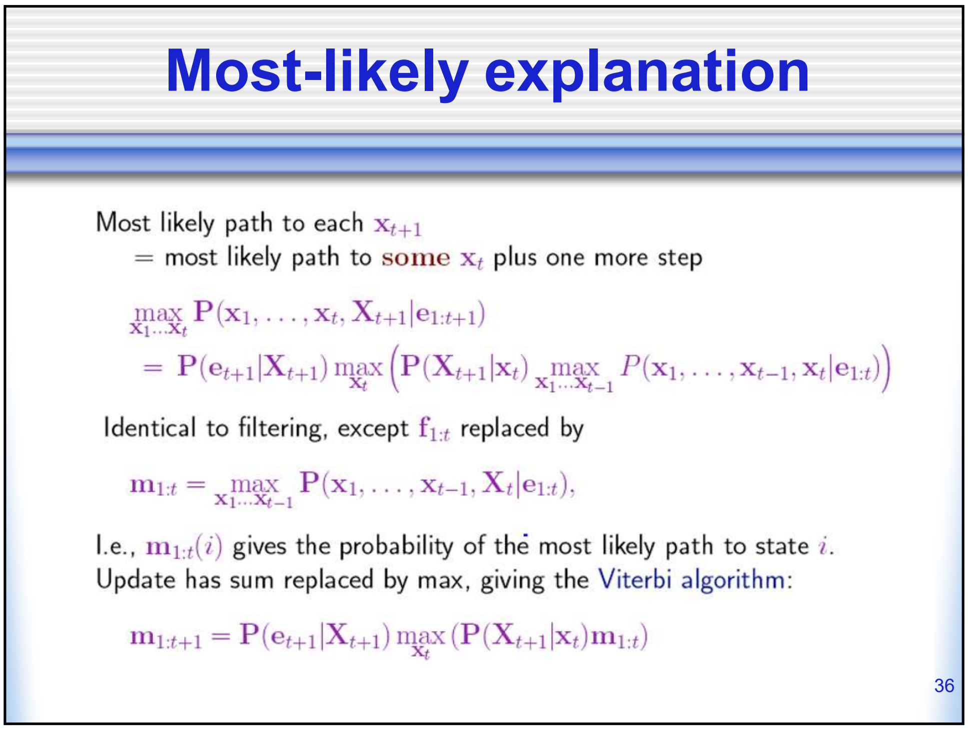
- Most likely explanation:

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

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

Algorithms for this task are useful in many applications, including, e.g., speech recognition. Can also be used to compare different temporal models that might have produced as sequence of events.

35



Most-likely explanation

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

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

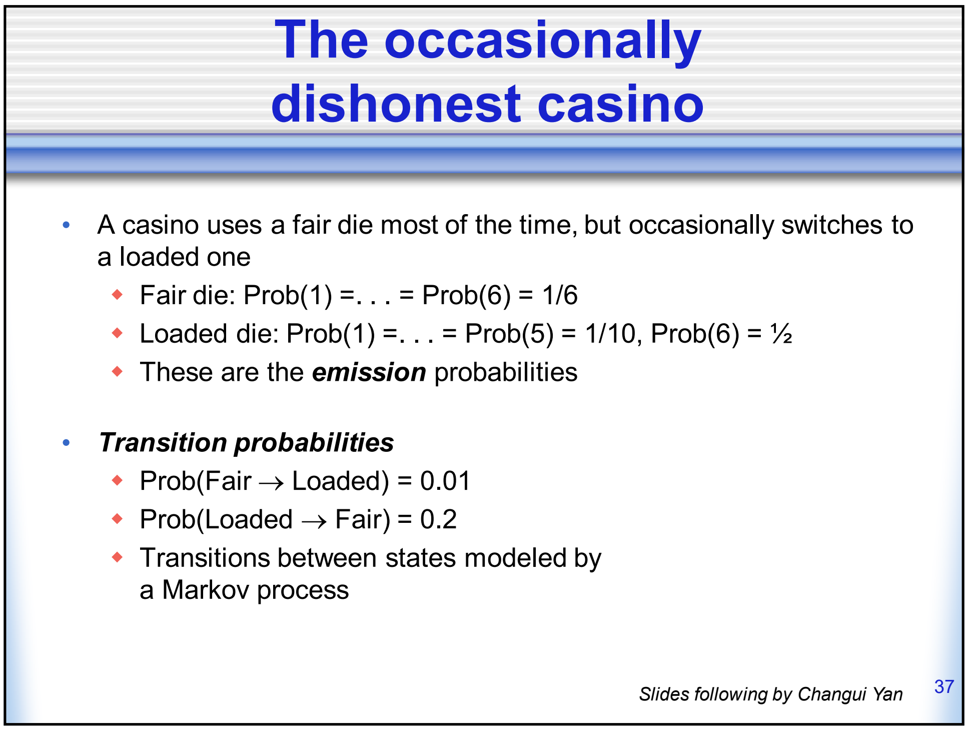
Identical to filtering, except $\mathbf{f}_{1:t}$ replaced by

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

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:

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

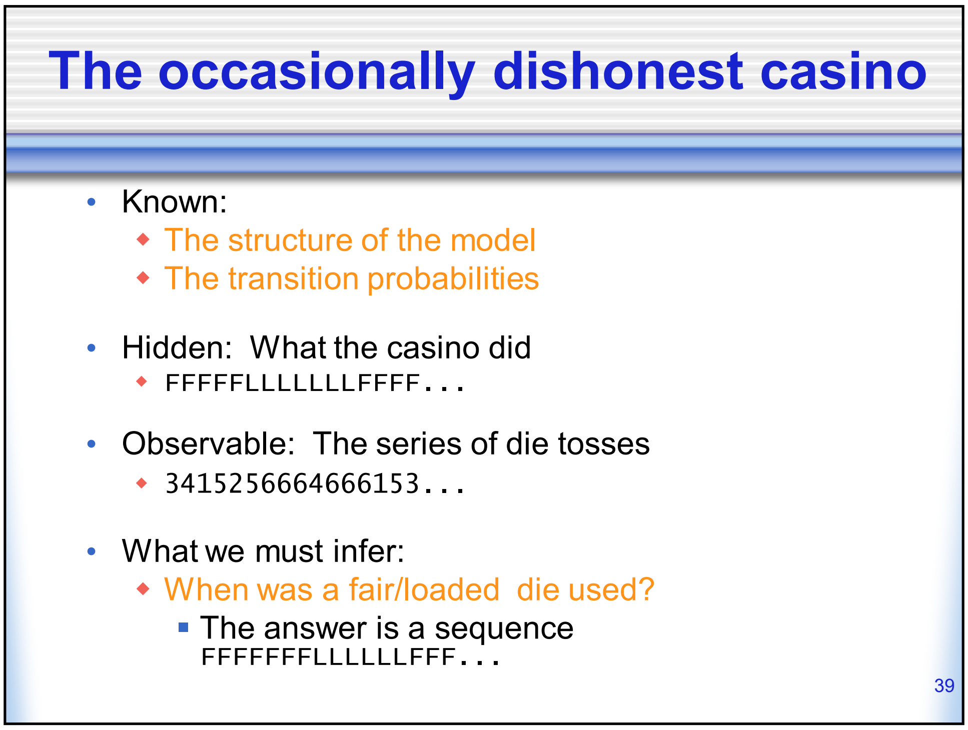
36



The occasionally dishonest casino

- A casino uses a fair die most of the time, but occasionally switches to a loaded one
  - Fair die: Prob(1) =. . . = Prob(6) = 1/6
  - Loaded die: Prob(1) =. . . = Prob(5) = 1/10, Prob(6) = ½
  - These are the ***emission*** probabilities
- ***Transition probabilities***
  - Prob(Fair → Loaded) = 0.01
  - Prob(Loaded → Fair) = 0.2
  - Transitions between states modeled by a Markov process

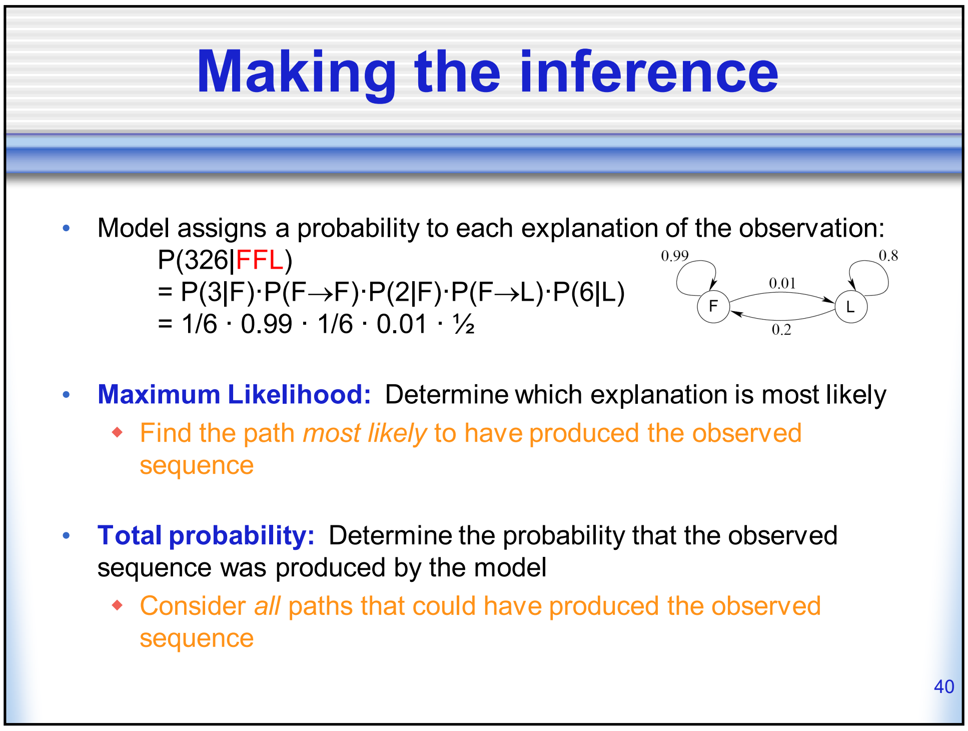
Slides following by Changui Yan 37



The occasionally dishonest casino

- Known:
  - The structure of the model
  - The transition probabilities
- Hidden: What the casino did
  - `FFFFFLLLLLLLFFFF...`
- Observable: The series of die tosses
  - `3415256664666153...`
- What we must infer:
  - When was a fair/loaded die used?
    - The answer is a sequence `FFFFFFFLLLLLLFFF...`

39



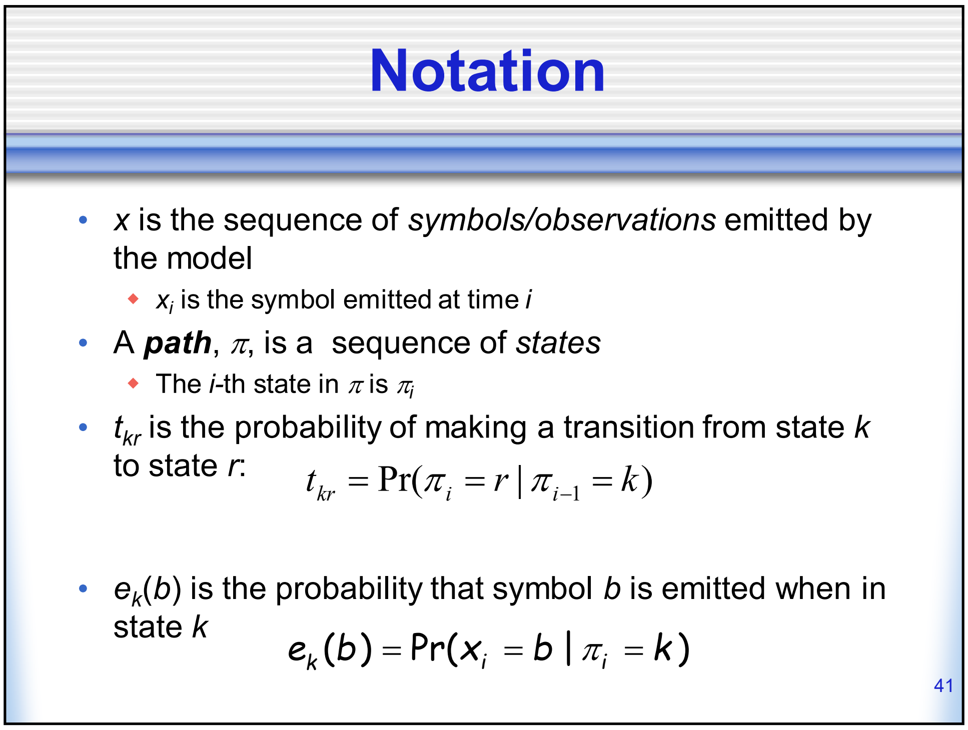
Making the inference

- Model assigns a probability to each explanation of the observation:
  P(326|FFL)
  = P(3|F)·P(F→F)·P(2|F)·P(F→L)·P(6|L)
  = 1/6 · 0.99 · 1/6 · 0.01 · ½

$$\begin{tikzcd} F \arrow[loop above, "0.99"] \arrow[r, bend left, "0.01"] & L \arrow[loop above, "0.8"] \arrow[l, bend left, "0.2"] \end{tikzcd}$$

- **Maximum Likelihood:** Determine which explanation is most likely
  - Find the path *most likely* to have produced the observed sequence
- **Total probability:** Determine the probability that the observed sequence was produced by the model
  - Consider *all* paths that could have produced the observed sequence

40

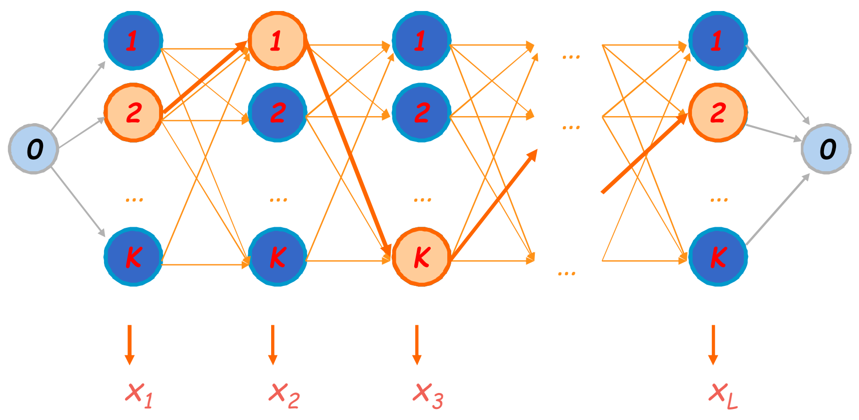


Notation

- *x* is the sequence of *symbols/observations* emitted by the model
  - x_i is the symbol emitted at time *i*
- A ***path***, π, is a sequence of *states*
  - The *i*-th state in π is π_i
- t_{kr} is the probability of making a transition from state *k* to state *r*: $$t_{kr} = \Pr(\pi_i = r \mid \pi_{i-1} = k)$$
- $e_k(b)$ is the probability that symbol *b* is emitted when in state *k* $$e_k(b) = \Pr(x_i = b \mid \pi_i = k)$$

41

A “parse” of a sequence



$$\Pr(x,\pi)=t_{0\pi_1}\prod_{i=1}^{L}e_{\pi_i}(x_i)\cdot t_{\pi_i\pi_{i+1}}$$

42

The occasionally dishonest casino

$x=\langle x_1,x_2,x_3\rangle=\langle 6,2,6\rangle$

$\pi^{(1)}=FFF$

$$\Pr(x,\pi^{(1)})=t_{0F}e_F(6)t_{FF}e_F(2)t_{FF}e_F(6)$$
$$=0.5\times\frac{1}{6}\times0.99\times\frac{1}{6}\times0.99\times\frac{1}{6}$$
$$\approx 0.00227$$

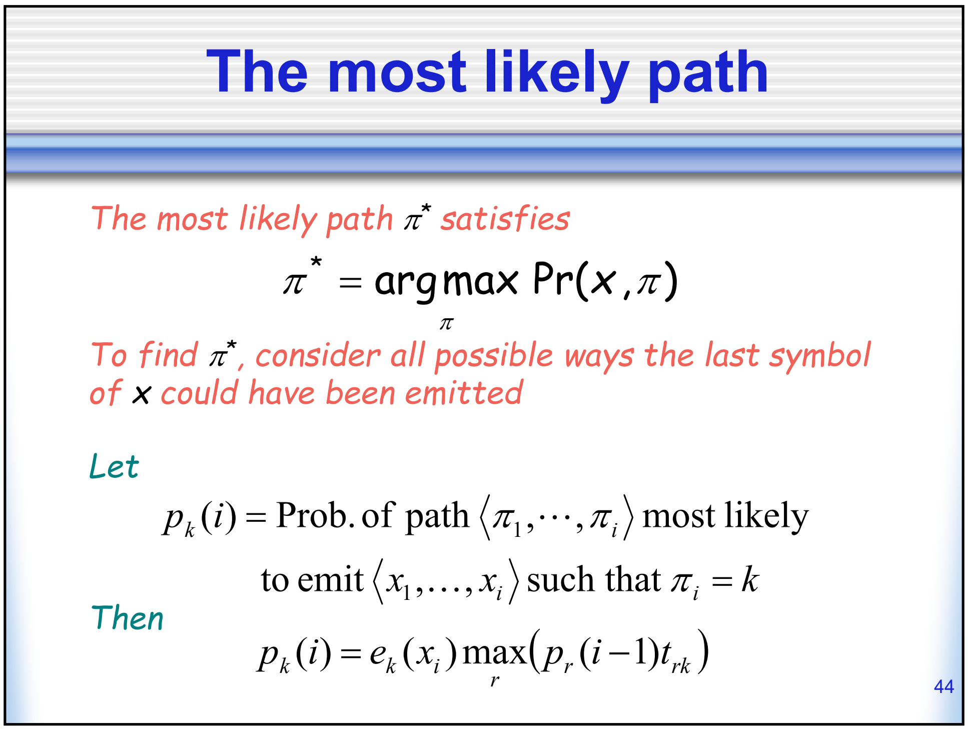
$\pi^{(2)}=LLL$

$$\Pr(x,\pi^{(2)})=t_{0L}e_L(6)t_{LL}e_L(2)t_{LL}e_L(6)$$
$$=0.5\times0.5\times0.8\times0.1\times0.8\times0.5$$
$$=\boxed{0.008}$$

$\pi^{(3)}=LFL$

$$\Pr(x,\pi^{(3)})=t_{0L}e_L(6)t_{LF}e_F(2)t_{FL}e_L(6)$$
$$=0.5\times0.5\times0.2\times\frac{1}{6}\times0.01\times0.5$$
$$\approx 0.0000417$$

43



The most likely path

The most likely path π^* satisfies

$$\pi^* = \arg\max_{\pi} \Pr(x,\pi)$$

To find π^*, consider all possible ways the last symbol of x could have been emitted

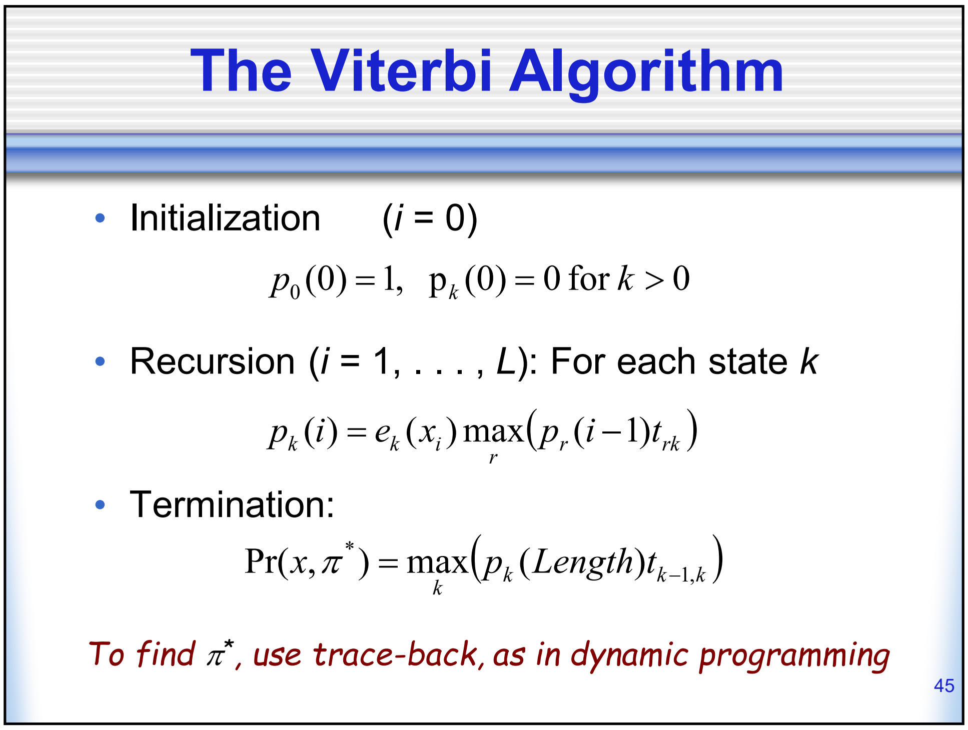
Let

$$p_k(i) = \text{Prob. of path } \langle \pi_1,\cdots,\pi_i \rangle \text{ most likely to emit } \langle x_1,\ldots,x_i \rangle \text{ such that } \pi_i = k$$

Then

$$p_k(i) = e_k(x_i)\max_r\left(p_r(i-1)t_{rk}\right)$$

44



The Viterbi Algorithm

- Initialization (*i* = 0)

$$p_0(0) = 1, \quad \mathrm{p}_k(0) = 0 \text{ for } k > 0$$

- Recursion (*i* = 1, . . . , *L*): For each state *k*

$$p_k(i) = e_k(x_i)\max_r\left(p_r(i-1)t_{rk}\right)$$

- Termination:

$$\Pr(x,\pi^*) = \max_k\left(p_k(Length)t_{k-1,k}\right)$$

To find π^*, use trace-back, as in dynamic programming

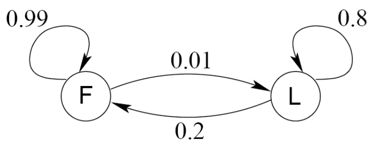
45

Viterbi: Example

x

			6	2	6
π	F	0	(1/6)×(1/2) = 1/12	(1/6)×max{(1/12)×0.99, (1/4)×0.2} = 0.01375	(1/6)×max{0.01375×0.99, 0.02×0.2} = 0.00226875
	L	0	(1/2)×(1/2) = 1/4	(1/10)×max{(1/12)×0.01, (1/4)×0.8} = 0.02	(1/2)×max{0.01375×0.01, 0.02×0.8} = 0.08

$$p_k(i) = e_k(x_i)\max_r\left(p_r(i-1)t_{rk}\right)$$



Viterbi gets it right more often than not

```
Rolls    315116246446644245321131631164152133625144543631656626566666
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL
Rolls    651166453132651245636664631636663162326455235266666625151631
Die      LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi  LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF
Rolls    222555441666566563564324364131513465146353411126414626253356
Die      FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Rolls    366163666466232534413661661163252562462255265252266435353336
Die      LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi  LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Rolls    233121625364414432335163243633665562466662632666612355245242
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

Dynamic Bayesian Networks

- In addition to the discussed tasks, methods are needed for *learning* the transition and sensor models from observation.
- Learning can be done by inference, where inference provides an estimate of what transitions actually occurred and of what states generated the sensor readings. These estimates can be used to update the models.
- The updated models provide new estimates, and the process iterates to convergence.

48

DBN – Special Cases

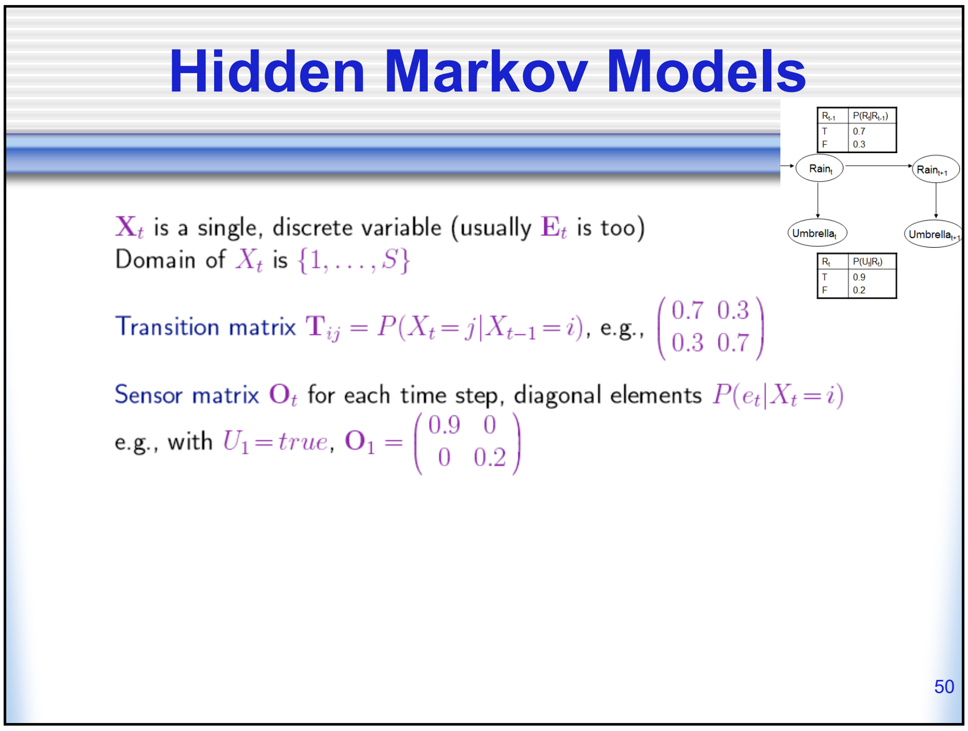
- Hidden Markov Model (HMMs):

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

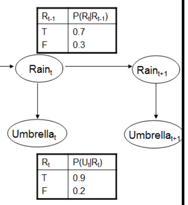
- Kalman Filter Models (KFMs):

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

49



Hidden Markov Models



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

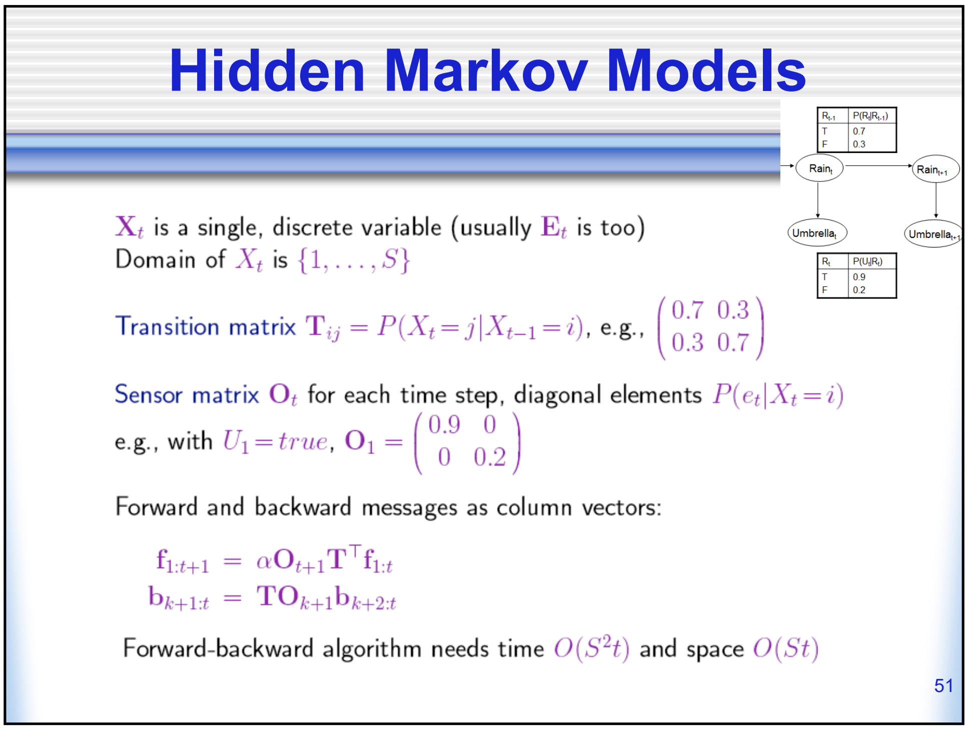
R_t	$P(U_t \mid R_t)$
T	0.9
F	0.2

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

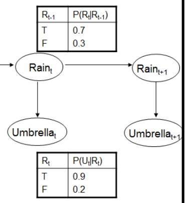
Transition matrix $\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i)$, e.g., $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$

Sensor matrix $\mathbf{O}_t$ for each time step, diagonal elements $P(e_t | X_t = i)$
e.g., with $U_1 = true$, $\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$

50



Hidden Markov Models



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

$\mathbf{X}_t$ 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., $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$

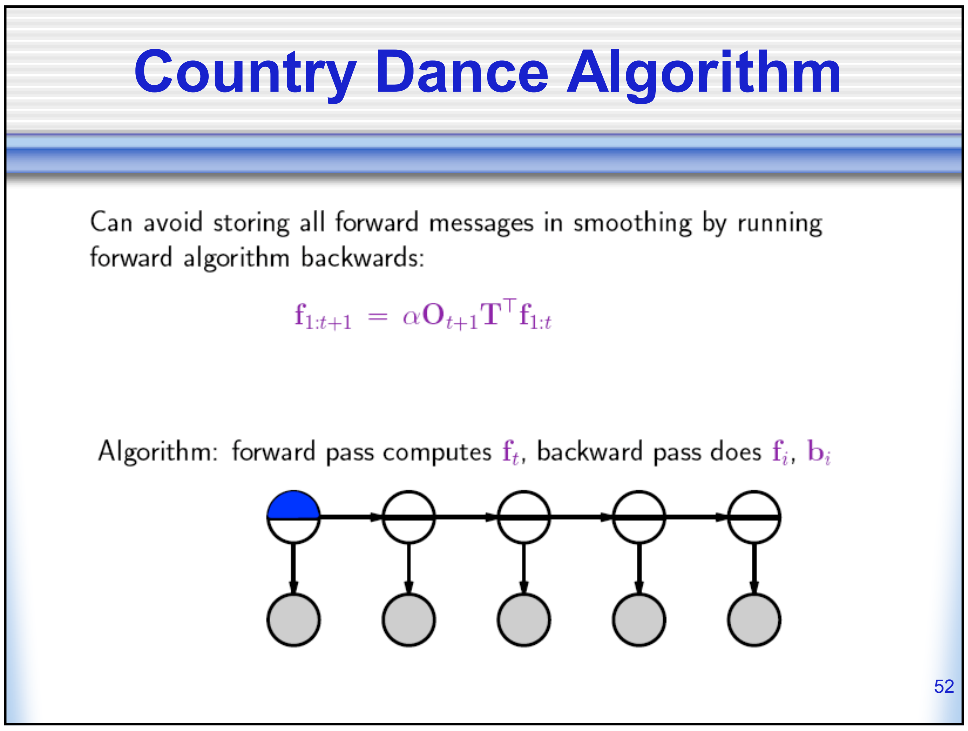
Sensor matrix $\mathbf{O}_t$ for each time step, diagonal elements $P(e_t | X_t = i)$
e.g., with $U_1 = true$, $\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$

Forward and backward messages as column vectors:

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

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

51

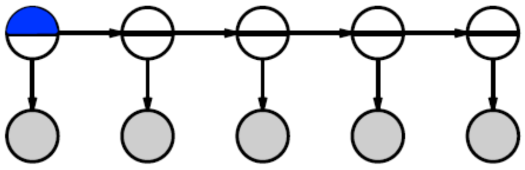


Country Dance Algorithm

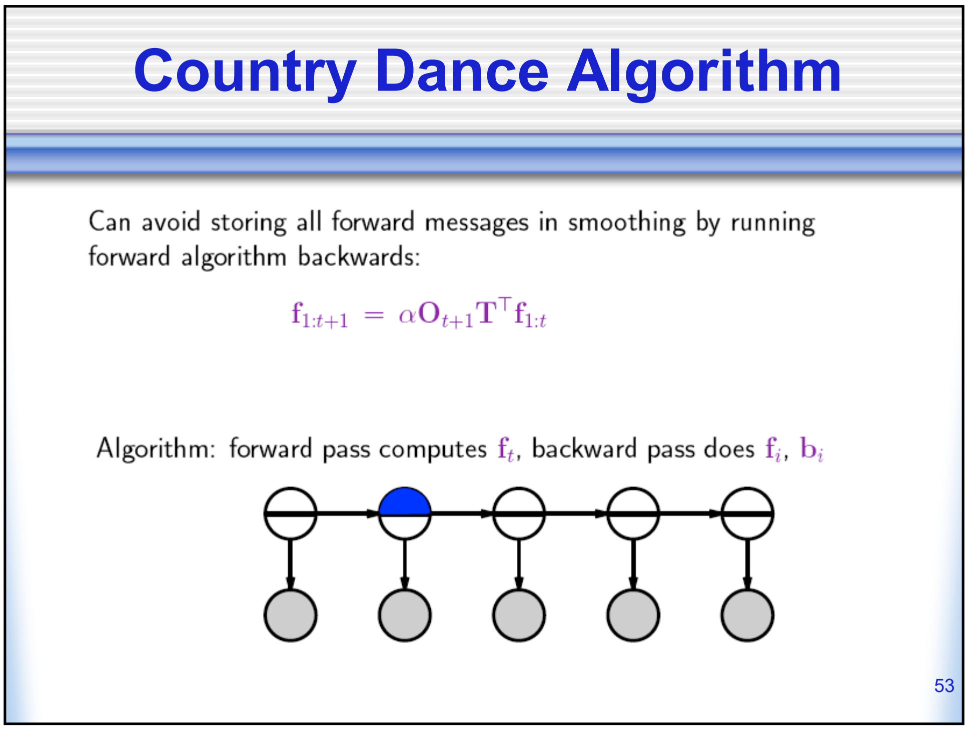
Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

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



52

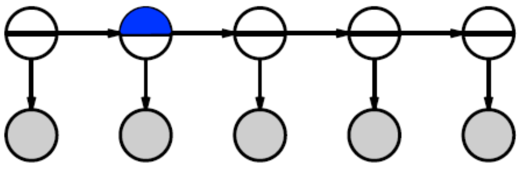


Country Dance Algorithm

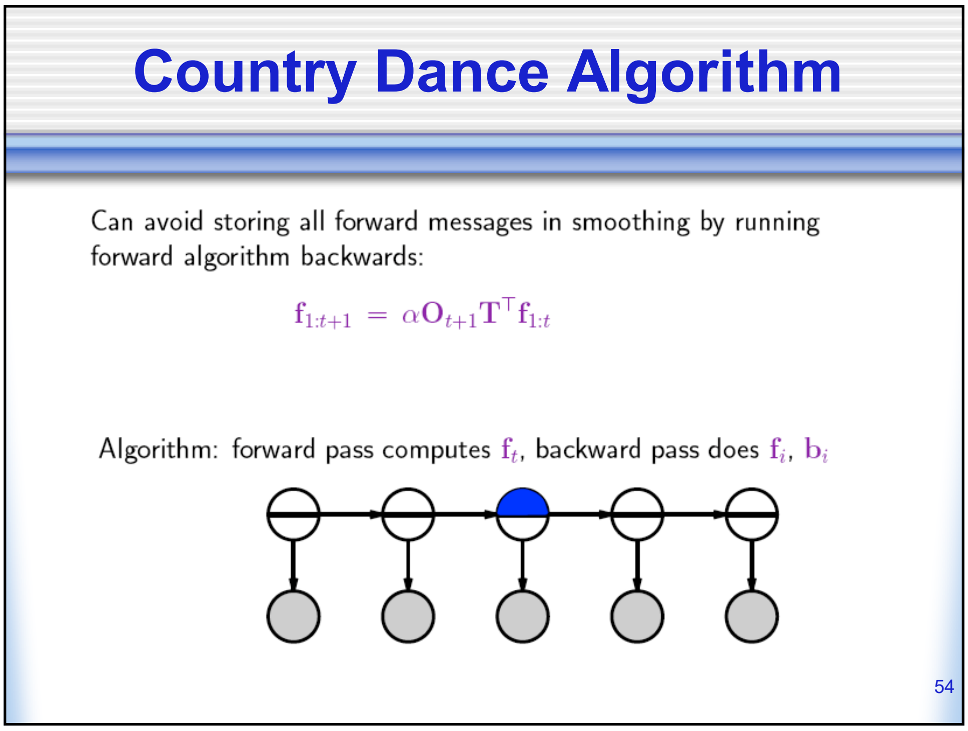
Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

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



53

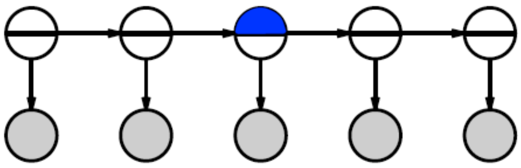


Country Dance Algorithm

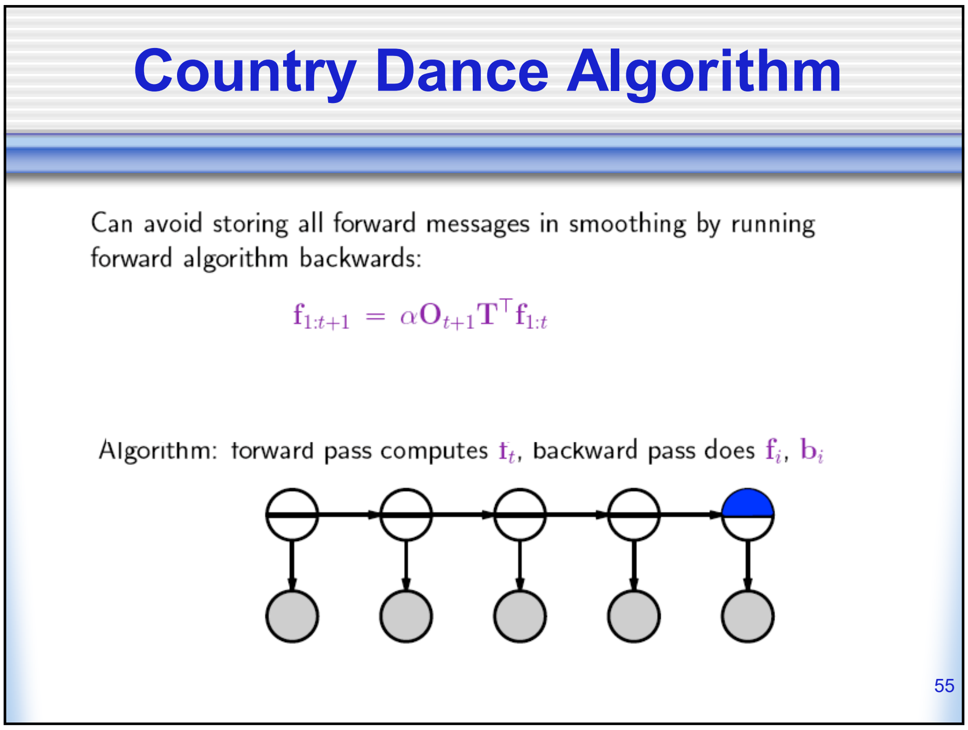
Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

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



54

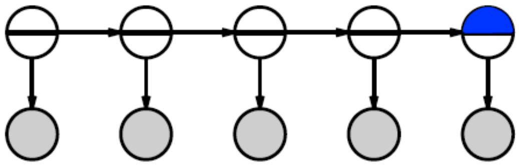


Country Dance Algorithm

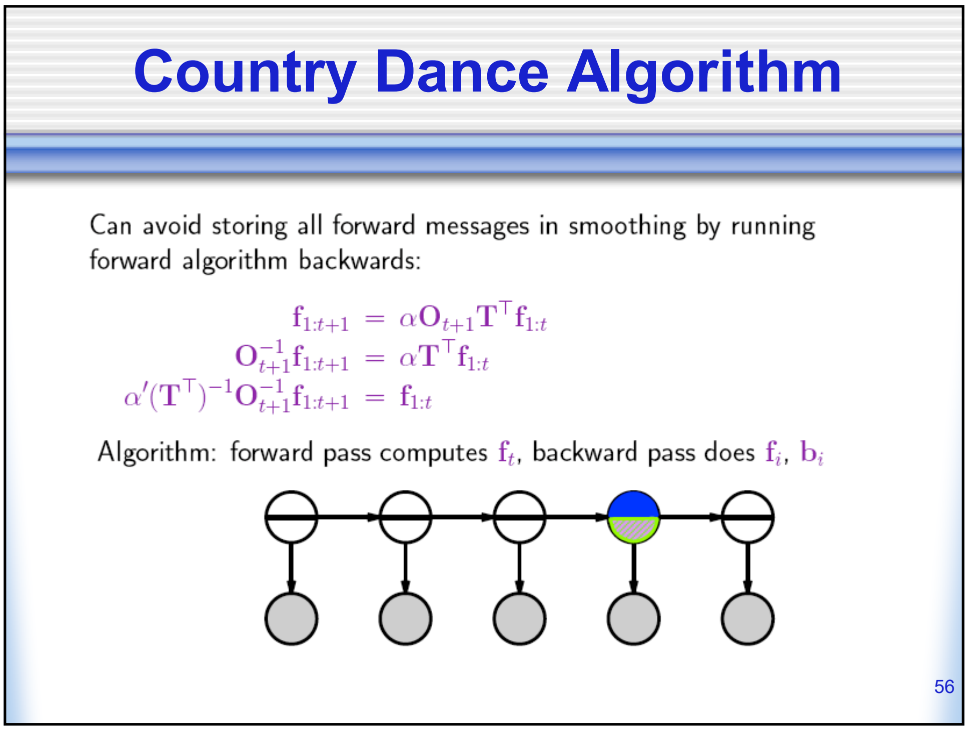
Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

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



55

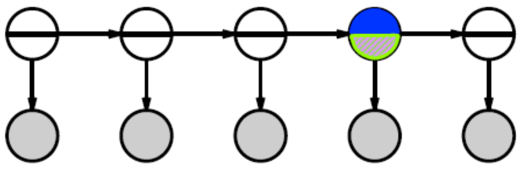


Country Dance Algorithm

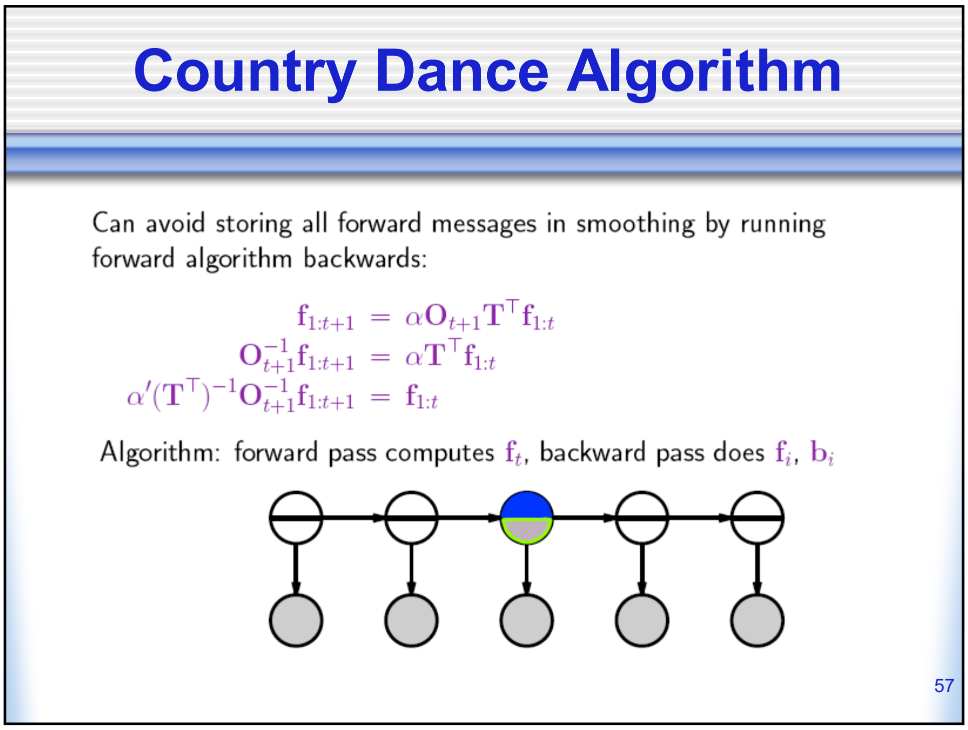
Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

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

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



56

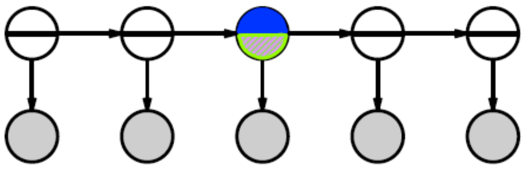


Country Dance Algorithm

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 $\mathbf{f}_i$, $\mathbf{b}_i$

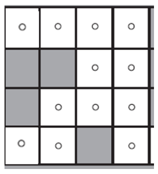


57

Applications

- Speech recognition
- Robot localization
- …

58



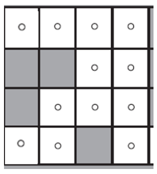
One non-deterministic operation MOVE.

$$P(X_{t+1} = j \mid X_t = i) = \mathbf{T}_{ij} = (1/N(i) \text{ if } j \in \text{NEIGHBORS}(i) \text{ else } 0)$$

E_t has 16 possible values, each a four-bit sequence giving the presence or absence of an obstacle: NSWE.
ε is the error rate. All four bits right $(1-\varepsilon)^4$. All wrong ε^4.

d_{it} is the number of bits that are different between the true values for square *i* and the actual reading e_t, then the probability that a robot in square *i* would receive a sensor reading e_t is:

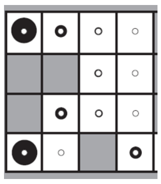
$$P(E_t = e_t \mid X_t = i) = \mathbf{O}_{t_{ii}} = (1-\epsilon)^{4-d_{it}} \epsilon^{d_{it}}$$

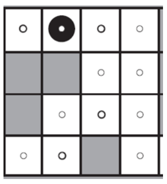


Cell numbers: start in top row, left to right

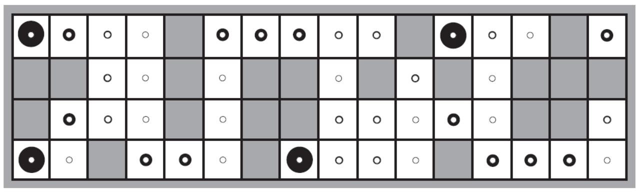
Matrix for **NSW**

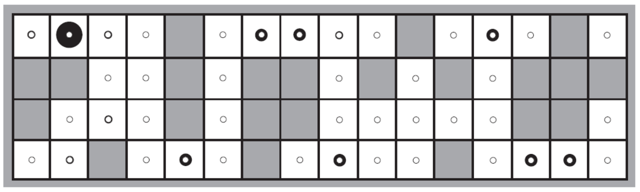
	1	2	3	…	12
1	$(1-\varepsilon)^4$				
2		$(1-\varepsilon)^3\,\varepsilon$			
3			$(1-\varepsilon)^2\,\varepsilon^2$		
…				….	
12					$(1-\varepsilon)^2\,\varepsilon^2$


E_1=NSW


E_2=NS

Example AIMA


(a) Posterior distribution over robot location after E_1 = NSW


(b) Posterior distribution over robot location after E_1 = NSW, E_2 = NS

Performance

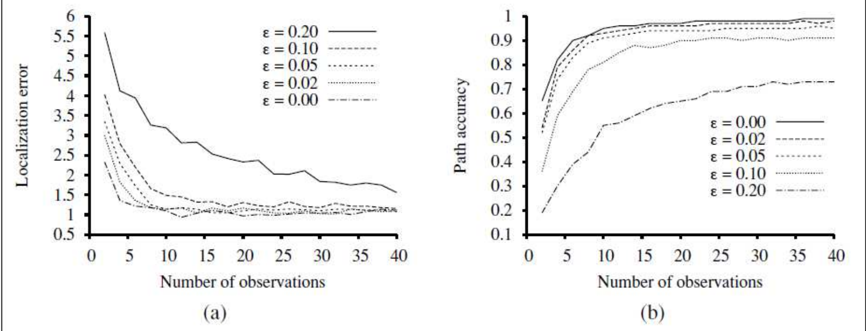


Figure 15.8 Performance of HMM localization as a function of the length of the observation sequence for various different values of the sensor error probability ϵ; data averaged over 400 runs. (a) The localization error, defined as the Manhattan distance from the true location. (b) The Viterbi path accuracy, defined as the fraction of correct states on the Viterbi path.

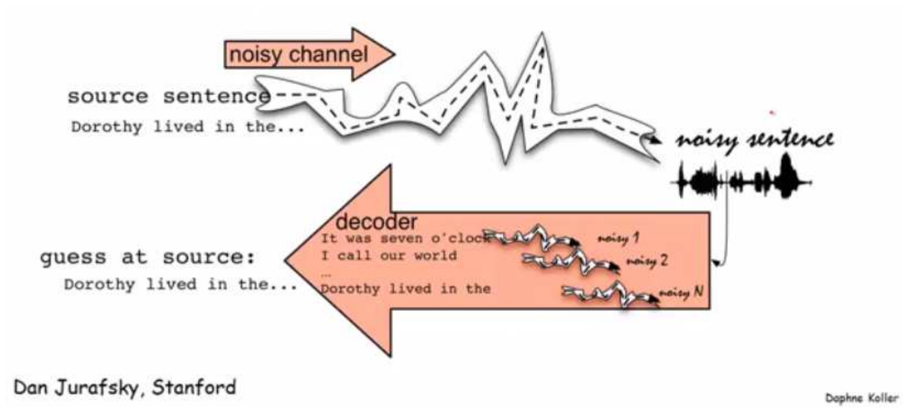
Last time

- Filtering
- Prediction
- Smoothing
- Viterbi for *most likely path/state sequence* for given observation
- HMM
  - Only one state variable
  - Efficient computation because of matrix operations

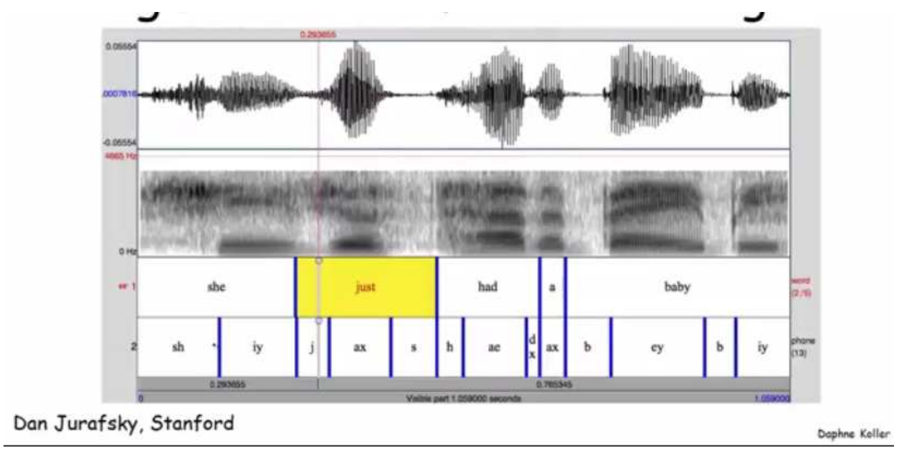
$$\begin{aligned}
\mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1}\mathbf{T}^{\top}\mathbf{f}_{1:t} \\
\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^{\top}\mathbf{f}_{1:t} \\
\alpha'(\mathbf{T}^{\top})^{-1}\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t}
\end{aligned}$$

63

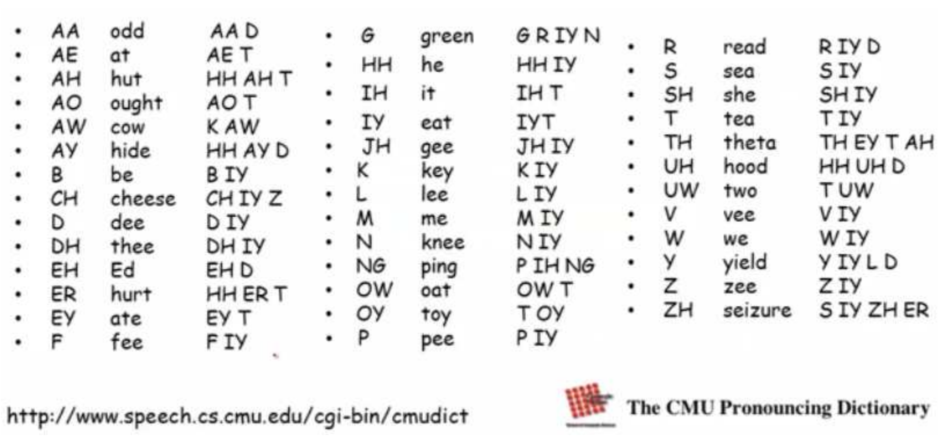
Speech recognition



Segmentation of Acoustic signals



Phonetic alphabet

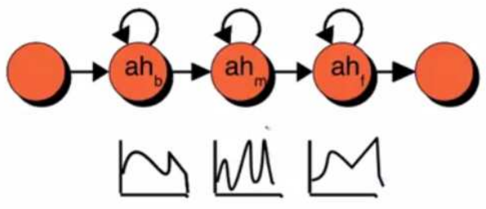


AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

http://www.speech.cs.cmu.edu/cgi-bin/cmudict

The CMU Pronouncing Dictionary

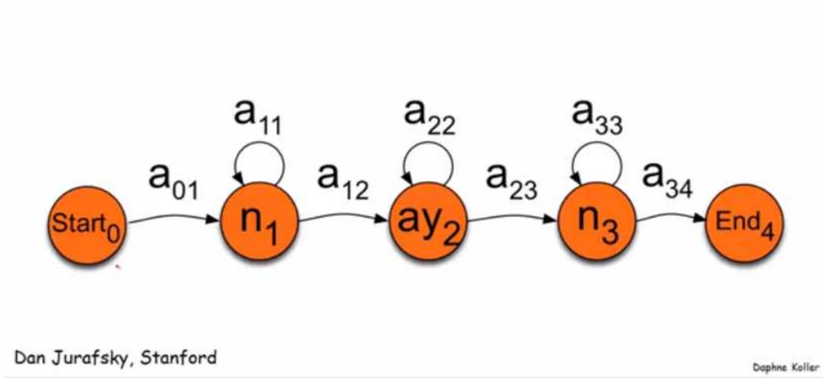
HMM ah



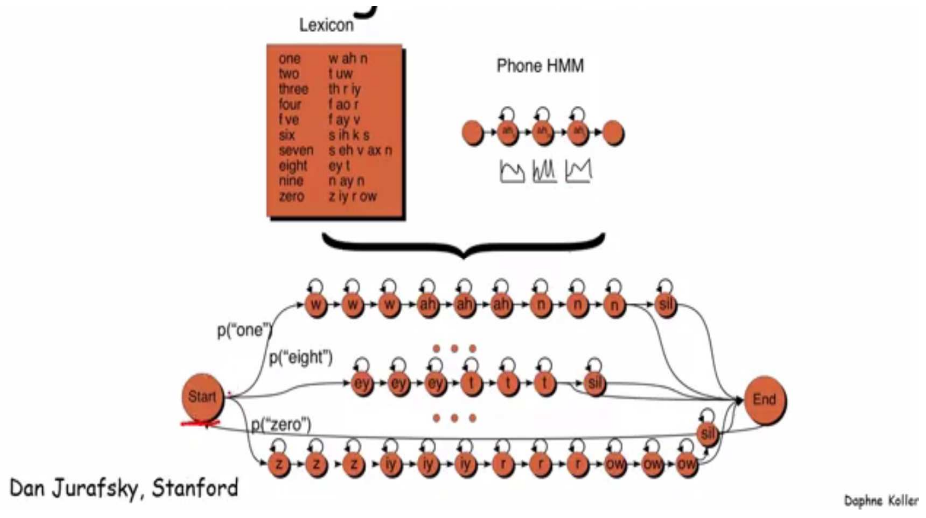
Dan Jurafsky, Stanford

Daphne Koller

Word HMM: nine



Recognition HMM

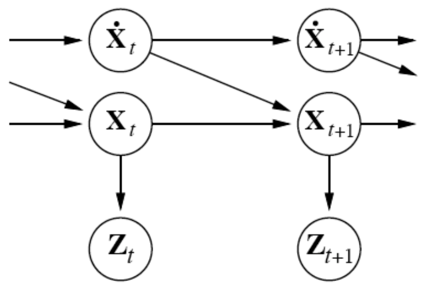


Lexicon

one	w ah n
two	t uw
three	th r iy
four	f ao r
f ve	f ay v
six	s ih k s
seven	s eh v ax n
eight	ey t
nine	n ay n
zero	z iy r ow

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

70

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

71

Simple 1-D Example

Prior $P(x_0) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}$.

Transition model $P(x_{t+1}|x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)}$

Sensor model $P(z_t|x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(z_t-x_t)^2}{\sigma_z^2}\right)}$

Prediction

$$\begin{aligned} P(x_1) &= \int_{-\infty}^{\infty} P(x_1|x_0)P(x_0)\,dx_0 = \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(x_1-x_0)^2}{\sigma_x^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}\,dx_0 \\ &= \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{\sigma_0^2(x_1-x_0)^2+\sigma_x^2(x_0-\mu_0)^2}{\sigma_0^2\sigma_x^2}\right)}\,dx_0\,. \\ &= \alpha\, e^{-\frac{1}{2}\left(\frac{(x_1-\mu_0)^2}{\sigma_0^2+\sigma_x^2}\right)} \end{aligned}$$

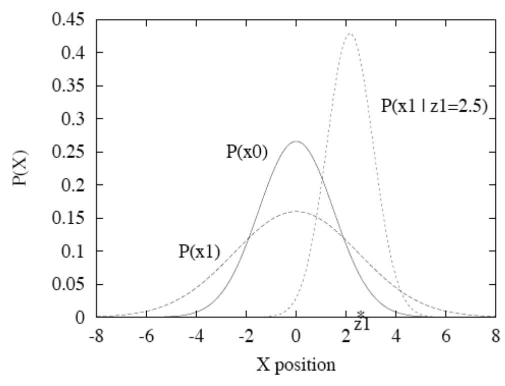
(by using **completing the square**. *Not discussed here)*

72

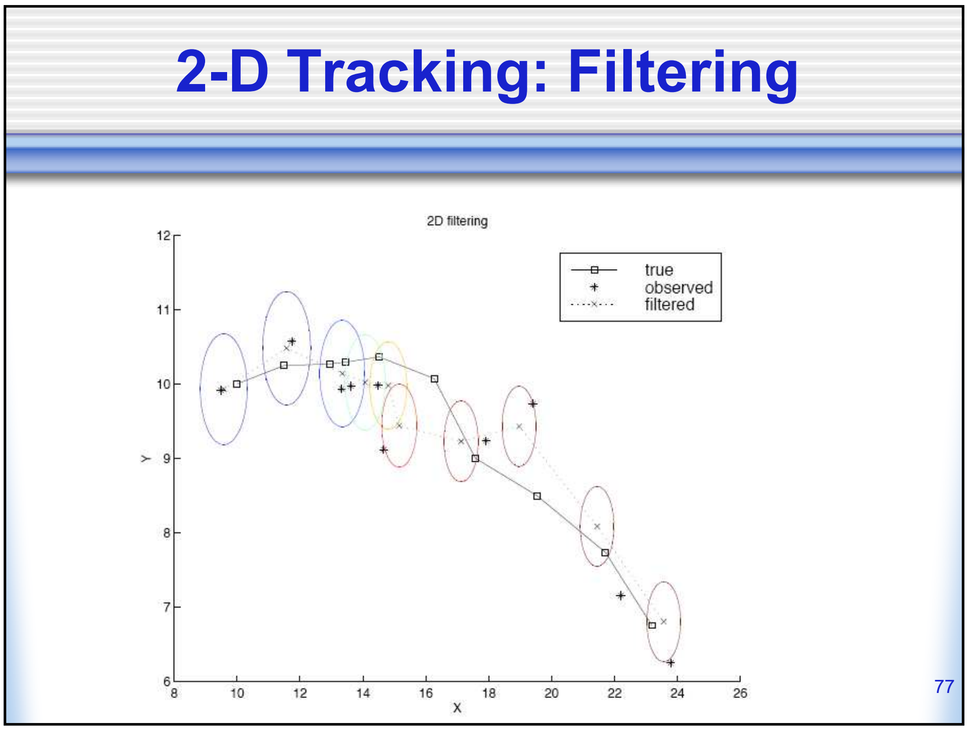
Simple 1-D Example

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

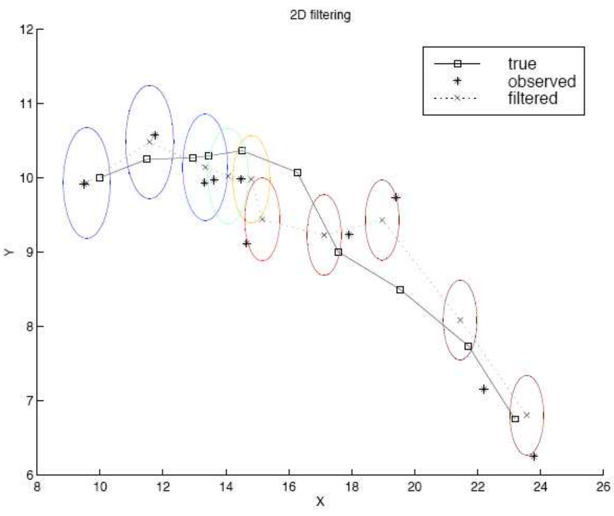
$$\mu_{t+1} = \frac{(\sigma_t^2+\sigma_x^2)z_{t+1}+\sigma_z^2\mu_t}{\sigma_t^2+\sigma_x^2+\sigma_z^2} \qquad \sigma_{t+1}^2 = \frac{(\sigma_t^2+\sigma_x^2)\sigma_z^2}{\sigma_t^2+\sigma_x^2+\sigma_z^2}$$



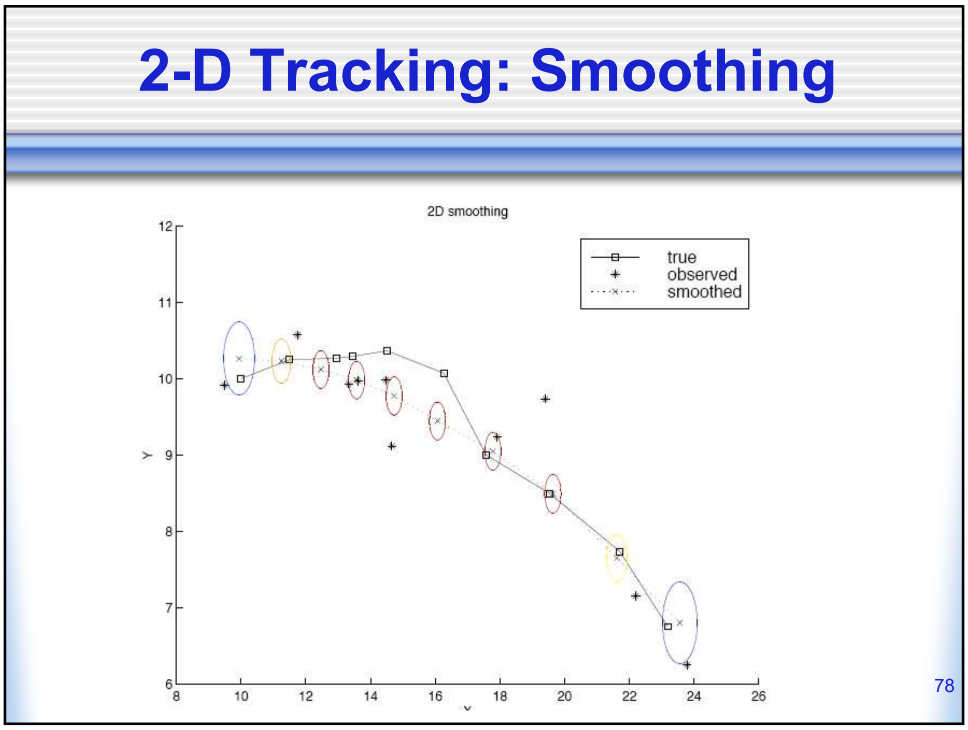
75



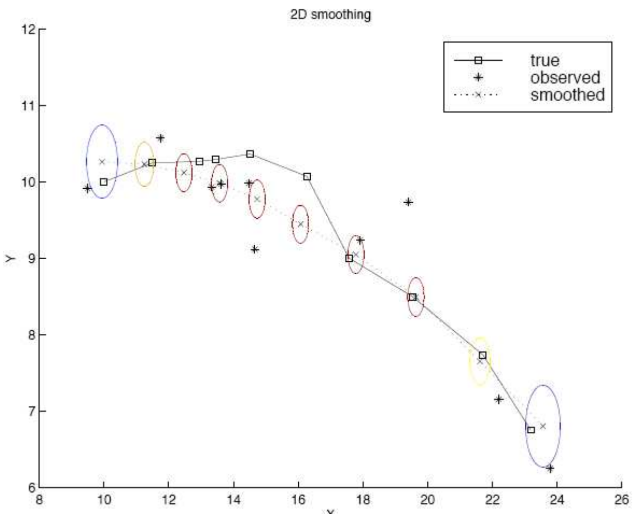
2-D Tracking: Filtering



77



2-D Tracking: Smoothing



78

Where it breaks

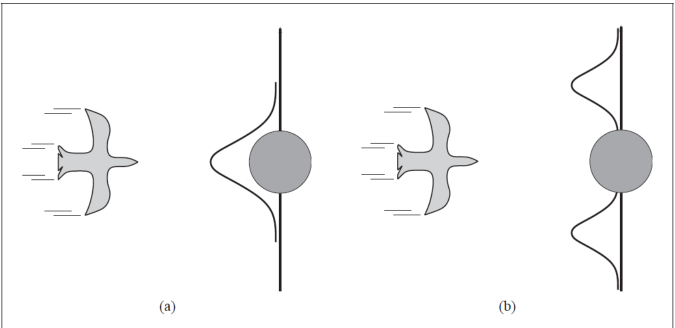


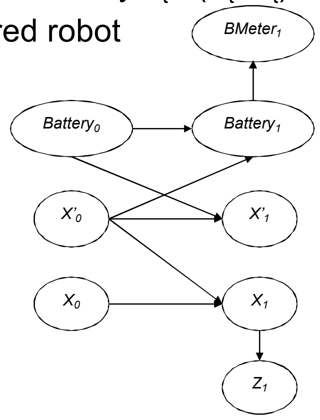
Figure 15.12 FILES: figures/kalman-bird1.eps (Tue Nov 3 16:23:06 2009) figures/kalman-bird2.eps (Tue Nov 3 16:23:06 2009). A bird flying toward a tree (top views). (a) A Kalman filter will predict the location of the bird using a single Gaussian centered on the obstacle. (b) A more realistic model allows for the bird's evasive action, predicting that it will fly to one side or the other.

One solution → switching kalman filters

79

Creating DBNs with failures

- $\mathbf{X'}_t$ =(X_t, Y_t) for velocity $\mathbf{X}_t$ =(X_t, Y_t) for position
- Battery powered robot

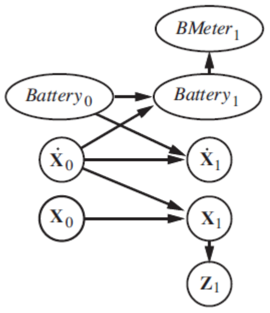


81

Failure of sensors

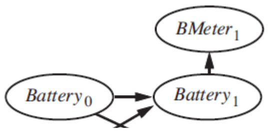
- Sensor measurements are noisy
- Real sensors can fail
- May use a Gaussian error model for *discrete variables*

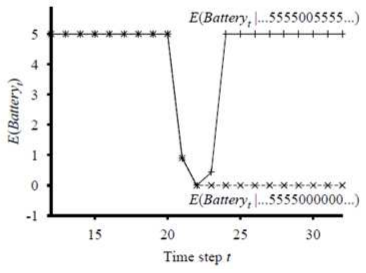
- Transient failure
- Persistent failure



82

Transient failure model





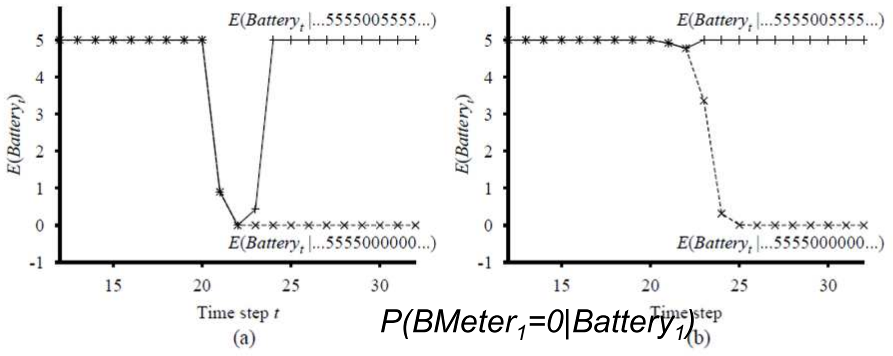
P(BMeter₁=0|Battery₁) P(Battery₁)

$P(Battery_1|BMeter_1=0) = \alpha <0.99, 0.006, 0.004><0.05, 0.05, 0.9>$

=<0,92178771 , 0,01117318 , 0,06703911 >

83

Transient failure model



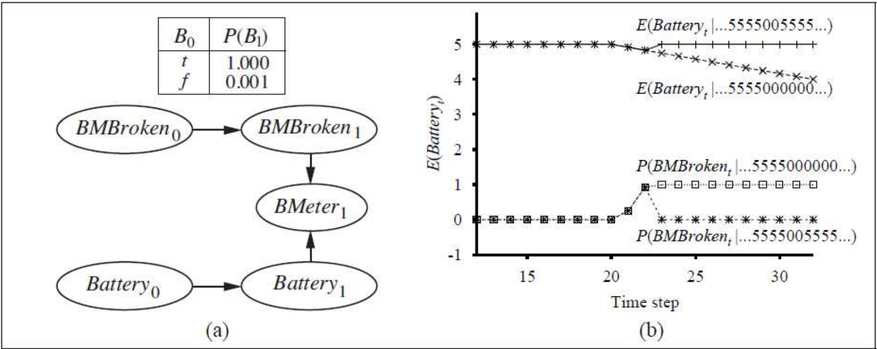
$P(BMeter_1=0|Battery_1)$

$P(Battery_1|BMeter_1=0) = \alpha <0.8, 0.1, 0.1><0.05, 0.05, 0.9>$

$\approx <0.44, 0.06, 0.5>$

84

Persistent failure model

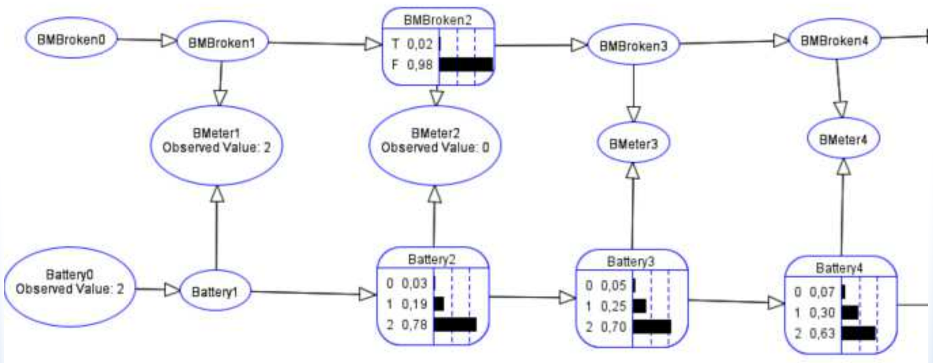


B_0 | P(B_1)
t | 1.000
f | 0.001
BMBroken_0
BMBroken_1
BMeter_1
Battery_0
Battery_1
(a)
E(Battery_t |...5555005555...)
E(Battery_t |...5555000000...)
P(BMBroken_t |...5555000000...)
P(BMBroken_t |...5555005555...)
E(Battery_t)
Time step
(b)

Figure 15.15 FILES: figures/battery-persistence.eps (Tue Nov 3 16:22:26 2009). (a) A DBN fragment showing the sensor status variable required for modeling persistent failure of the battery sensor. (b) Upper curves: trajectories of the expected value of $Battery_t$ for the "transient failure" and "permanent failure" observations sequences. Lower curves: probability trajectories for $BMBroken$ given the two observation sequences.

85

Example

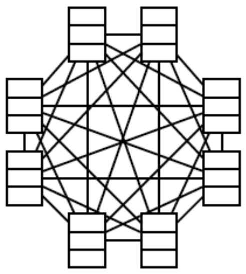


86

DBNs vs. HMMs

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

$$\begin{tikzcd}
X_t \arrow[r] & X_{t+1} \\
Y_t \arrow[r] & Y_{t+1} \\
Z_t \arrow[r] & Z_{t+1}
\end{tikzcd}$$



Consider the transition model

Sparse dependencies $\Rightarrow$ exponentially fewer parameters;

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

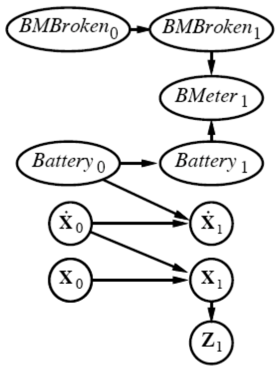
DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$

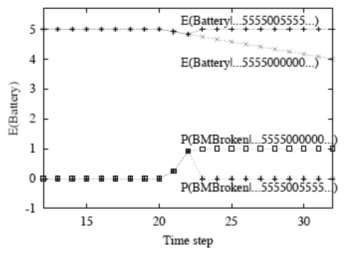
87

DBNs vs. Kalman Filters

Every Kalman filter model is a DBN, but few DBNs are KFs;
real world requires non-Gaussian posteriors

E.g., where are bin Laden and my keys? What's the battery charge?

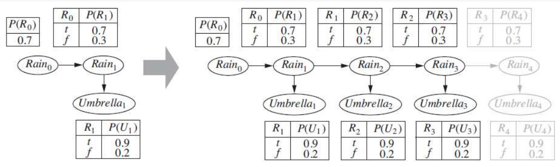




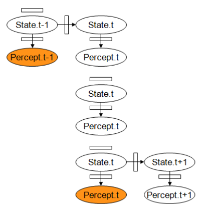
88

Exact Inference in DBNs

Naive:



Rollup filtering:



$O(d^{n+k})$ largest factor

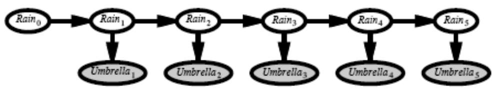
20 state variables (4 values)
mean 4^{20+1} =4.398.046.511.104

d = possible values for variables
n = number of states
k = number of parents

90

Approximate inference: Likelihood Weighting

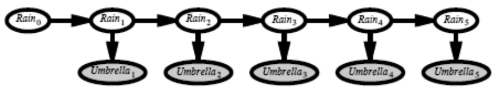
Set of weighted samples approximates the belief state



91

Likelihood Weighting

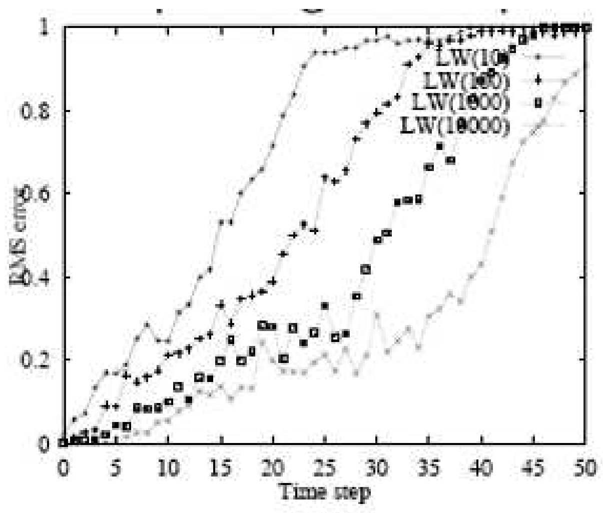
Set of weighted samples approximates the belief state



LW samples pay no attention to the evidence!
$\Rightarrow$ fraction "agreeing" falls exponentially with t
$\Rightarrow$ number of samples required grows exponentially with t

92

Likelihood Weighting



93

Solution

- Instead of running one example at a time run N.
  - The N samples also represent an approximate representation of the current state distribution.
- Instead of using initial examples throw low weighted ones away.
  - Must add new examples else lose to much.

94

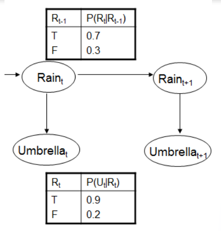
Idea: Particle filtering

A population of N initial-state samples is created sampling from $P(X_0)$

1. Based on the transition matrix propagate examples forward. $P(X_{t+1}|x_t)$
2. Each sample is weighted by the likelihood it assigns to the new evidence $P(e_{t+1}|x_{t+1})$.
3. **Resample examples based on it‘s weight.**

95

Particle Filtering

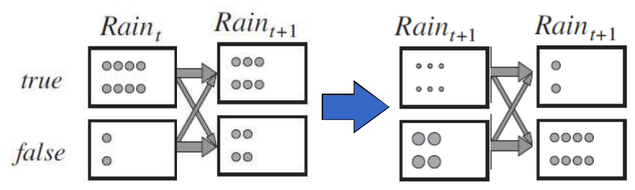


R_{t-1}	$P(R_t|R_{t-1})$
T	0.7
F	0.3

R_t	$P(U_t|R_t)$
T	0.9
F	0.2

Our current particles,10

Propagate forward

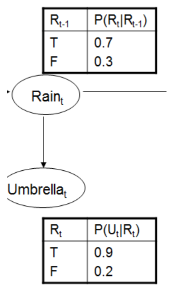


No umbrella is observed at t+1

resampling

96

Example



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

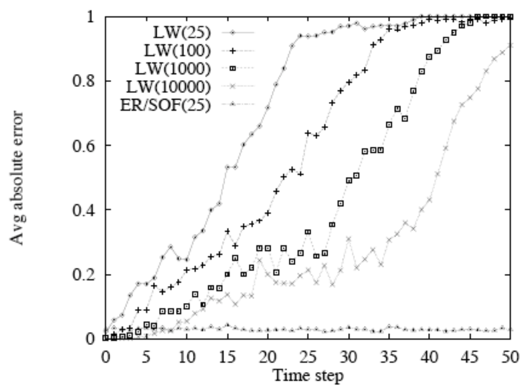
$N(r_{t+1}|e) = \Sigma_{x_t} P(x_{t+1}|x_t)\ N(x_t|e)$
For rain = 0.7*8+0.3*2= 6.2 => 6
For not rain = 0.3 *8 + 0.7*2= 3.8 => 4

Suppose no umbrella for t+1
total weight(rain particles) = 0.1 * 6= 0.6
total weight(not rain) = 0.8 * 4= 3.2
Normalized =<0.17, 0.83>

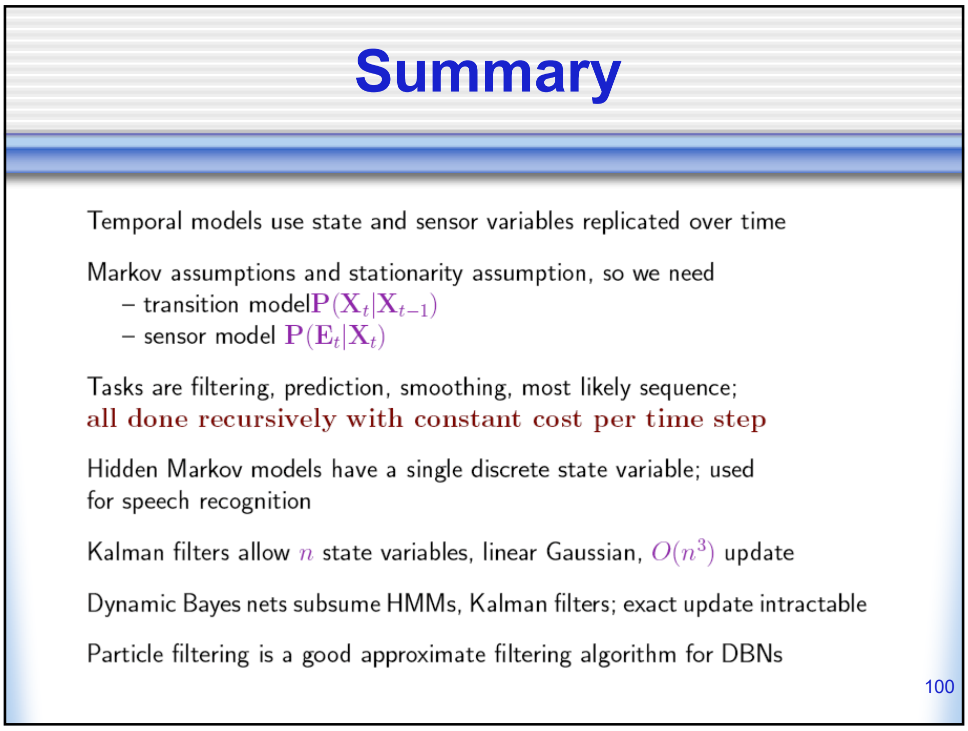
97

Particle Filtering: Performance

Approximation error of particle filtering remains bounded over time, at least empirically—theoretical analysis is difficult



99



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

Particle filtering is a good approximate filtering algorithm for DBNs

100