

# Artificial Intelligence

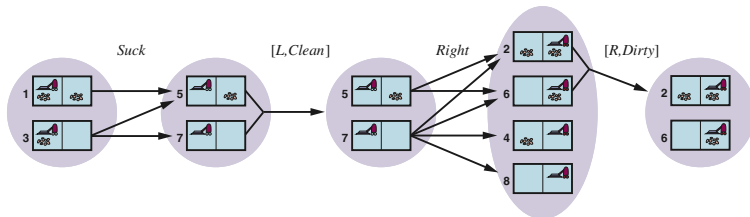
## Probabilistic Temporal Reasoning

Christopher Simpkins

Kennesaw State University

# Probabilistic Temporal Reasoning (AIMA 14.1-14.3)

Recall belief state maintenance from the Kindergarten vacuum world (square not being actively cleaned can become dirty):



- ▶ Most real-world environments partially observable. Belief state maintenance is core task.
- ▶ Also known as **monitoring**, **filtering**, and **state estimation**.

$$b' = \text{Update}(\text{Predict}(b, a), o).$$

- ▶ Equation above is called a recursive state estimator because it computes the new belief state from the previous one rather than by examining the entire percept sequence.

Previous methods allowed us to represent belief states as sets of possible worlds, but could not compute or represent how likely each state were. In this lesson we use the tools of probability theory to quantify our degree of belief in elements of the belief state.

# Elements of Stochastic Temporal Models

A changing world is modeled using

- ▶ a variable for each aspect of the world state at each point in time,
- ▶ a transition model describing the probability distribution of the variables at time  $t$ , given the state of the world at past times, and
- ▶ a sensor model (a.k.a. observation model) describing the probability of each percept at time  $t$ , given the current state of the world.

# Time and Uncertainty

Static worlds: each random variable has a single fixed value.

Example: car repair. Car stays broken during diagnosis.

Dynamic worlds: each random variable has a value for each time step which depends on the values of other random variables in previous time steps.

Example: Diabetes.

- ▶ Task is to assess current state of the patient – blood sugar and insulin levels – in order to choose patient's food intake and insulin dose.
- ▶ Blood sugar levels and measurements thereof can change rapidly over time, depending on recent food intake and insulin doses, metabolic activity, the time of day, and so on.
- ▶ To assess the current state from the history of evidence and to predict the outcomes of treatment actions, we must model these changes.

Other examples:

- ▶ Robot location tracking
- ▶ Tracking national economic activity
- ▶ Interpreting written or spoken sequences of words.

# Discrete-Time Models

In **Discrete-time** models:

- ▶ World is a series of snapshots or **time slices**.
- ▶ Time slices numbered  $0, 1, 2, \dots$
- ▶ Time interval  $\Delta$  between slides assumed to be the same for every interval.
- ▶ Particular applications use particular choices of  $\Delta$ , e.g.,  $\frac{1}{30}$  of a second.
- ▶ Choice of  $\Delta$  should reflect rates of change.
  - ▶ E.g., blood glucose can change significantly in 10 minutes, so  $\Delta = 1$  minute might be appropriate.

Discrete vs continuous time:

- ▶ Continuous time systems can be modeled by stochastic differential equations (SDEs).
- ▶ Discrete-time models presented here are approximations to SDEs.

# States and Observations

Each time slice contains a set of random variables:

- ▶  $\mathbf{X}_t$  is a set of unobservable state variables at time  $t$ .
- ▶  $\mathbf{E}_t$  is a set of observable evidence variables at time  $t$ .

Observation at time  $t$  is  $\mathbf{E}_t = e_t$  for some set of values  $e_t$ .

**Example: Security guard in underground facility.** You are stuck inside with no access to outside world. Each morning you observe a person walking in with or without an umbrella.

- ▶  $\Delta$  is one day, each  $t$  is a day.
- ▶  $\mathbf{E}_t$  contains a single evidence variable,  $Umbrella_t$  or  $U_t$  for short.
- ▶  $\mathbf{X}_t$  contains a single state variable,  $Rain_t$  or  $R_t$  for short.

**Example: diabetes.**

- ▶  $\mathbf{E}_t = \{MeasuredBloodSugar_t, PulseRate_t\}$
- ▶  $\mathbf{X}_t = \{BloodSugar_t, StomacheContents_t\}$

A few assumptions about  $t$ :

- ▶ State sequence starts at time  $t = 0$ . (In umbrella world,  $\mathbf{X} = \{R_0, R_1, R_2, \dots\}$ .)
- ▶ Evidence starts arriving at time  $t = 1$ . (In umbrella world,  $\mathbf{E} = \{U_1, U_2, \dots\}$ .)
- ▶  $\mathbf{X}_{a:b}$  denotes set of variables from  $\mathbf{X}_a$  to  $\mathbf{X}_b$ , inclusive.

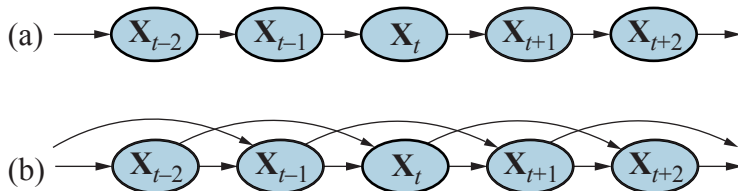
- ▶ Note: inclusive, unlike Python where `u[1:3]` includes only `u[1]` and `u[2]`

# Transition Models

The transition model specifies the probability distribution over the latest state variables, given previous values:  $Pr(\mathbf{X}_t \mid \mathbf{X}_{0:t-1})$

- ▶ Problem:  $\mathbf{X}_{0:t-1}$  is unbounded – size increases as  $t$  increases.
- ▶ Solution: **Markov assumption** – current state depends on a finite fixed number of previous states.
  - ▶ First studied by Andrei Markov (1856-1922), now called **Markov processes** or **Markov chains**.

In these Bayes nets:



- ▶ (a) First-order Markov process:  $Pr(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = Pr(\mathbf{X}_t \mid \mathbf{X}_{t-1})$
- ▶ (b) Second-order Markov process:  $Pr(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = Pr(\mathbf{X}_t \mid \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$

# Time-Homogeneous Processes

Problem:

- ▶ Infinitely many values of  $t$ .
- ▶ Could be different distributions for each variable at each time step.

Solution: assume **time homogeneity**, i.e., state changes caused by laws that don't change over time.

Example: Umbrella world

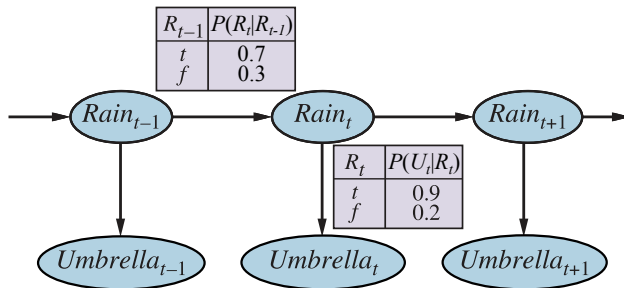
- ▶  $Pr(R_t | R_{t-1})$  is the same for all  $t$
- ▶ Only need one conditional probability table.



## Sensor Models

**Sensor Markov assumption:** state alone is sufficient to generate current sensor values.

$$Pr(\mathbf{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{1:t-1}) = Pr(\mathbf{E}_t \mid \mathbf{X}_t) \quad (14.2)$$



- ▶ Transition model is  $Pr(Rain_t \mid Rain_{t-1})$ .
- ▶ Sensor model is  $Pr(Umbrella_t \mid Rain_t)$ .
- ▶ Arrows go from actual state to sensor values: states *cause* sensor values.
  - ▶ Inference goes in other direction: given sensor values, what are the state values.

# Full Joint Over All Variables in a Temporal Model

For the initial state of the system at time 0 we specify a prior  $Pr(\mathbf{x}_0)$ . Now we can use Equation 13.2 ( $Pr(x_1, \dots, x_n) = \prod_{i=1}^n Pr(x_i | \text{parents}(X_i))$ ):

$$Pr(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \underbrace{Pr(\mathbf{X}_0)}_{\text{Initial state model}} \prod_{i=1}^t \underbrace{Pr(\mathbf{X}_i | \mathbf{X}_{i-1})}_{\text{Transition model}} \underbrace{Pr(\mathbf{E}_i | \mathbf{X}_i)}_{\text{Sensor model}}$$

Standard Bayes nets can only represent a finite set of variables. Dynamic bayesian networks overcome this limitation by:

- ▶ defining infinite sets by integer indices, and
- ▶ using implicit universal quantification to define sensor and transition models for every time step.

# Markov Model Considerations

Sometimes Markov assumption is valid, sometimes it's only an approximation. Two ways to improve the approximation:

1. Increase the order of the Markov process model. E.g., In Palo Alto, CA, rarely rains more than two days in a row. A 2nd-order Markov model could express this fact:

$$Pr(r_t \mid r_{t-1}, r_{t-2}) \ll Pr(r_t \mid r_{t-1}, \neg r_{t-2}).$$

2. Increase the set of state variables.

# Inference in Temporal Models

- ▶ **Filtering**, a.k.a., **state estimation** is
- ▶ **Prediction:**
- ▶ **Smoothing:**
- ▶ **Most likely explanation:**

# Learning Temporal Models

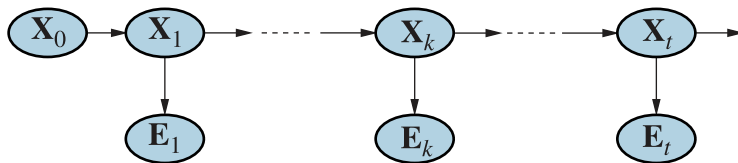
Unknown transition and sensor models can be learned from observations.

- ▶ As with static Bayesian networks, dynamic Bayes net learning can be done as a by-product of inference.
- ▶ Inference provides an estimate of transitions that actually occurred and the states that generated the sensor readings, and these estimates can be used to learn the models.
- ▶ Learning via iterative update algorithm, expectation–maximization or EM, or Bayesian updating of the model parameters given the evidence.

We'll return to these ideas in our lesson on **statistical learning**.

# Filtering and Prediction

# Smoothing



# Forward-Backward Smoothing Algorithm

**function** FORWARD-BACKWARD(**ev**, *prior*) **returns** a vector of probability distributions

**inputs:** **ev**, a vector of evidence values for steps  $1, \dots, t$

*prior*, the prior distribution on the initial state,  $\mathbf{P}(\mathbf{X}_0)$

**local variables:** **fv**, a vector of forward messages for steps  $0, \dots, t$

**b**, a representation of the backward message, initially all 1s

**sv**, a vector of smoothed estimates for steps  $1, \dots, t$

**fv**[0]  $\leftarrow$  *prior*

**for**  $i = 1$  **to**  $t$  **do**

**fv**[ $i$ ]  $\leftarrow$  FORWARD(**fv**[ $i - 1$ ], **ev**[ $i$ ])

**for**  $i = t$  **down to**  $1$  **do**

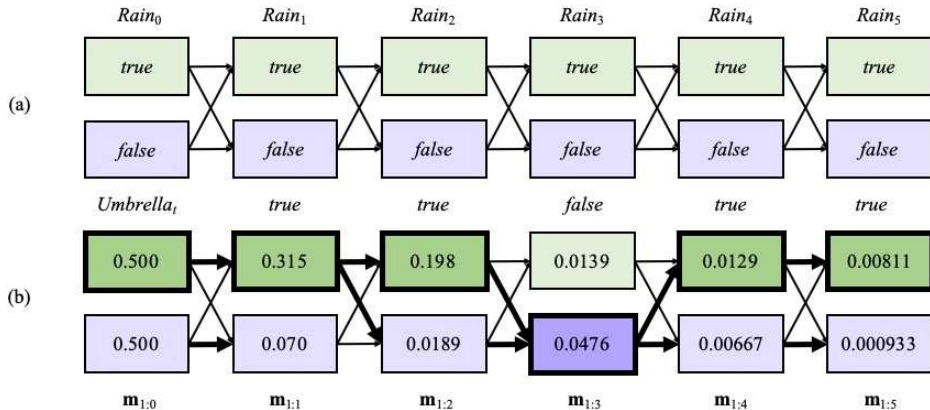
**sv**[ $i$ ]  $\leftarrow$  NORMALIZE(**fv**[ $i$ ]  $\times$  **b**)

**b**  $\leftarrow$  BACKWARD(**b**, **ev**[ $i$ ])

**return** **sv**



# Finding the Most Likely Sequence

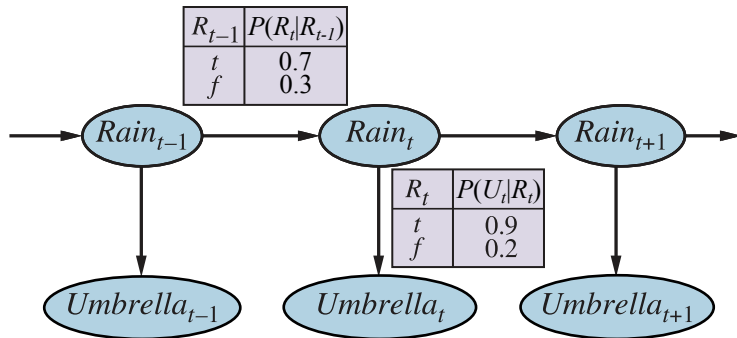


# Hidden Markov Models (HMMs)

An HMM is a temporal probabilistic model in which the state of the process is described by a single, discrete random variable.

# HMM Matrix Formulation

$$T_{ij} = \Pr(X_t = j \mid X_{t-1} = i)$$



$$T_{ij} = \Pr(X_t \mid X_{t-1}) = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}$$

# Fixed Lag Smoothing Algorithm

**function** FIXED-LAG-SMOOTHING( $e_t, hmm, d$ ) **returns** a distribution over  $\mathbf{X}_{t-d}$

**inputs:**  $e_t$ , the current evidence for time step  $t$

$hmm$ , a hidden Markov model with  $S \times S$  transition matrix  $\mathbf{T}$

$d$ , the length of the lag for smoothing

**persistent:**  $t$ , the current time, initially 1

$\mathbf{f}$ , the forward message  $\mathbf{P}(X_t | e_{1:t})$ , initially  $hmm.PRIOR$

$\mathbf{B}$ , the  $d$ -step backward transformation matrix, initially the identity matrix

$e_{t-d:t}$ , double-ended list of evidence from  $t-d$  to  $t$ , initially empty

**local variables:**  $\mathbf{O}_{t-d}, \mathbf{O}_t$ , diagonal matrices containing the sensor model information

add  $e_t$  to the end of  $e_{t-d:t}$

$\mathbf{O}_t \leftarrow$  diagonal matrix containing  $\mathbf{P}(e_t | X_t)$

**if**  $t > d$  **then**

$\mathbf{f} \leftarrow \text{FORWARD}(\mathbf{f}, e_{t-d})$

remove  $e_{t-d-1}$  from the beginning of  $e_{t-d:t}$

$\mathbf{O}_{t-d} \leftarrow$  diagonal matrix containing  $\mathbf{P}(e_{t-d} | X_{t-d})$

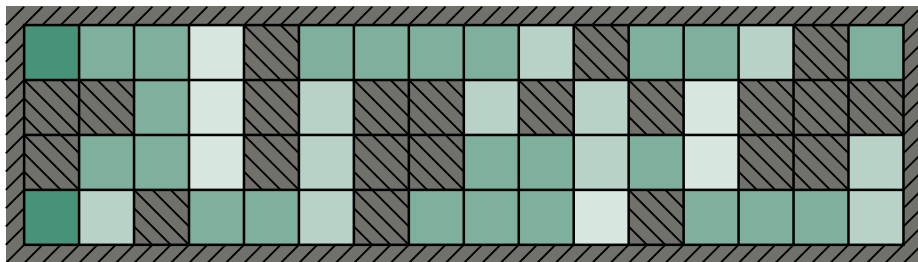
$\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1} \mathbf{T}^{-1} \mathbf{B} \mathbf{O}_t$

**else**  $\mathbf{B} \leftarrow \mathbf{B} \mathbf{O}_t$

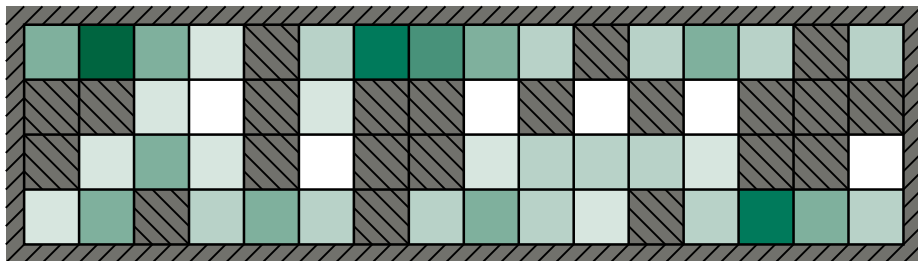
$t \leftarrow t + 1$

**if**  $t > d + 1$  **then return** NORMALIZE( $\mathbf{f} \times \mathbf{B} \mathbf{1}$ ) **else return** null

## Localization with HMMs

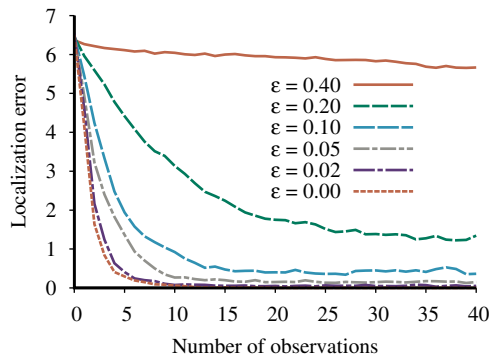


(a) Posterior distribution over robot location after  $E_1 = 1011$

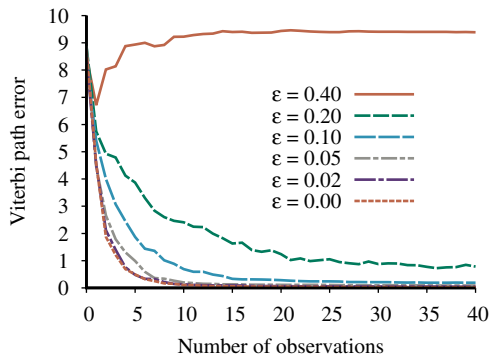


(b) Posterior distribution over robot location after  $E_1 = 1011$ ,  $E_2 = 1010$

# HMM Performance



(a)



(b)