

Artificial Intelligence

Probabilistic Temporal Reasoning (AIMA 14.1-14.2)

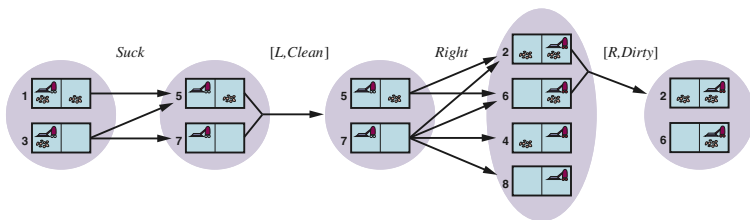
Christopher Simpkins

Kennesaw State University



Probabilistic Temporal Reasoning

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**.

$$b' = Update(Predict(b, a), o).$$

- ▶ 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 their measurements can change rapidly over time, depending on recent food intake, insulin doses, metabolic activity, the time of day, and so on.
- ▶ To assess the current state from the history of evidence and 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 Δ between slides assumed to be the same for every interval.
- ▶ Particular applications use particular choices of Δ , e.g., $\frac{1}{30}$ of a second.
- ▶ Choice of Δ 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.

- ▶ Δ 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 :

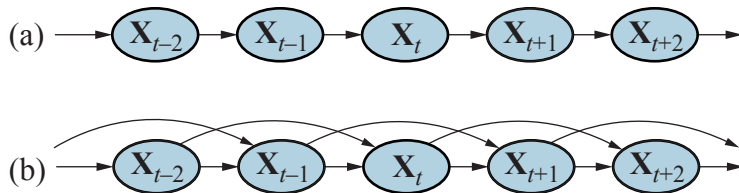
- ▶ State sequence starts at time $t = 0$. (In umbrella world, $\mathbf{X} = \{R_0, R_1, R_2, \dots\}$.)
- ▶ Evidence starts arriving at time $t = 1$. (In umbrella world, $\mathbf{E} = \{U_1, U_2, \dots\}$.)
- ▶ $\mathbf{X}_{a:b}$ denotes set of variables from \mathbf{X}_a to \mathbf{X}_b , inclusive.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

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

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.

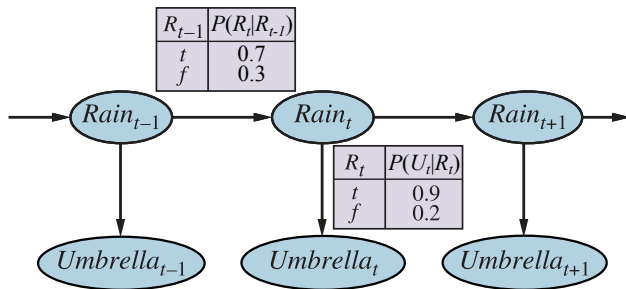
Example: Umbrella world

- ▶ $P(R_t | R_{t-1})$ 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.

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



- ▶ Transition model is $P(\text{Rain}_t \mid \text{Rain}_{t-1})$.
- ▶ Sensor model is $P(\text{Umbrella}_t \mid \text{Rain}_t)$.
- ▶ 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 Distribution Over All Variables in a Temporal Model

For the initial state of the system at time 0 we specify a prior $P(\mathbf{x}_0)$. Now we can use Equation 13.2

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i)) \quad (13.2)$$

applied to the temporal variables in the dynamic version:

$$P(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \underbrace{P(\mathbf{X}_0)}_{\text{Initial state model}} \prod_{i=1}^t \underbrace{P(\mathbf{X}_i | \mathbf{X}_{i-1})}_{\text{Transition model}} \underbrace{P(\mathbf{E}_i | \mathbf{X}_i)}_{\text{Sensor model}} \quad (14.3)$$

Standard Bayes nets can only represent a finite set of variables. Dynamic Bayes nets 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:
 - ▶ $P(r_t | r_{t-1}, r_{t-2}) \ll P(r_t | r_{t-1}, \neg r_{t-2})$.
2. Add additional state variables. E.g., add $Season_t$ for historical records, or $Temperature_t$, $Humidity_t$, and $Pressure_t$ to use a physical model of rainy conditions.

Example: Battery Drainage in a Mobile Robot

Two state variables: velocity and position. Use Newton's laws of motion to calculate new positions. Add probabilistic error (e.g., Gaussian noise) to account for uncertainty in velocity due to terrain, wind, etc.

Problems:

- ▶ Battery level affects velocity as it drains.
- ▶ Battery level depends on power used in all previous movements, violating the Markov assumption.

Solution: Add a state variable for battery level. Track level in one of two ways:

1. Decrease level at each time step in response to movement executed in previous step.
2. Better: add a new sensor for battery level.

Inference in Temporal Models

Given the general structure of a probabilistic temporal model, we can perform basic inference tasks:

- ▶ **Filtering**, a.k.a., **state estimation** is the task of computing the **belief state** $P(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ – the posterior distribution over the most recent state given all the evidence to date.
- ▶ **Prediction** is the task of computing the posterior distribution over the future state, given all evidence to date: $P(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ for some $k > 0$.
- ▶ **Smoothing** is the task of computing the posterior distribution over a past state, given all evidence up to the present: $P(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ for some k such that $0 \leq k < t$.
- ▶ **Most likely explanation**: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations: $\operatorname{argmax}_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$.

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

Filtering, a.k.a., **state estimation** is the task of computing the **belief state** $P(\mathbf{X}_t | e_{1:t})$ – the posterior distribution over the most recent state given all the evidence to date.

- ▶ Umbrella example: compute probability of rain today given all umbrella observations so far.
- ▶ Rational agent estimates its current state to enable rational decisions.
- ▶ Nearly identical calculation provides likelihood of evidence sequence $P(e_{1:t})$
- ▶ The term “filtering” comes from signal processing, which sees the problem of state estimation as “filtering out the noise” in a signal to estimate its underlying properties.

Need for efficiency. A useful filtering algorithm needs to maintain a current state estimate and update it, rather than going back over the entire history of percepts for each update.

- ▶ Otherwise, the cost of each update increases as time goes by.

In other words, given the result of filtering up to time t , the agent needs to compute the result for $t + 1$ from the new evidence e_{t+1} . For some function f :

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

Recursive State Estimation

We can view the calculation as being composed of two parts: first, the current state distribution is projected forward from t to $t + 1$; then it is updated using the new evidence e_{t+1} . This two-part process emerges quite simply when the formula is rearranged:

$$\begin{aligned} P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, e_{t+1}) && \text{(Divide the evidence)} \\ &= \alpha P(e_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) && \text{(Bayes rule, given } \mathbf{e}_{1:t}) \\ &= \underbrace{\alpha P(e_{t+1} \mid \mathbf{X}_{t+1})}_{\text{update}} \underbrace{P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})}_{\text{prediction}} && \text{(Sensor Markov assumption)} \end{aligned}$$

Now plug in an expression for one-step prediction $P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ conditioned on the current state \mathbf{X}_t to obtain the central result in probabilistic temporal reasoning:

$$\begin{aligned} P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \alpha P(e_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \\ &= \underbrace{\alpha P(e_{t+1} \mid \mathbf{X}_{t+1})}_{\text{sensor model}} \sum_{\mathbf{x}_t} \underbrace{P(\mathbf{X}_{t+1} \mid \mathbf{x}_t)}_{\text{transition model}} \underbrace{P(\mathbf{x}_t \mid \mathbf{e}_{1:t})}_{\text{recursion}} \end{aligned} \quad (14.5)$$

The last step applies the Markov assumption in the transition model. All the terms come either from the model or from the previous state estimate. Hence, we have the desired recursive formulation.

Forward Message Propagation

$$P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = \alpha \underbrace{P(\mathbf{e}_{t+1} | \mathbf{X}_{t+1})}_{\text{sensor model}} \sum_{\mathbf{x}_t} \underbrace{P(\mathbf{X}_{t+1} | \mathbf{x}_t)}_{\text{transition model}} \underbrace{P(\mathbf{x}_t | \mathbf{e}_{1:t})}_{\text{recursion}} \quad (14.5)$$

We can think of the filtered estimate $P(\mathbf{X}_t | \mathbf{e}_{1:t})$ as a “message” $\mathbf{f}_{1:t}$ that is propagated forward along the sequence, modified by each transition and updated by each new observation. The process is given by

$$\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$$

where

- ▶ FORWARD implements the update in Equation 14.5 and
- ▶ the process begins with $\mathbf{f}_{1:0} = P(\mathbf{X}_0)$.

When all the state variables are discrete, the time for each update is constant (i.e., independent of t), and the space required is also constant. (The constants depend, of course, on the size of the state space and the specific type of the temporal model in question.)

- ▶ *The time and space requirements for updating must be constant if a finite agent is to keep track of the current state distribution indefinitely.*

Example: Filtering in the Umbrella World

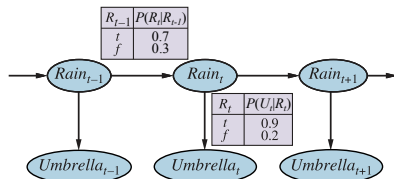
Compute $P(R_2 | u_{1:2})$:

- ▶ Day 0: no observations, only prior beliefs: $P(R_0) = \langle 0.5, 0.5 \rangle$
- ▶ Day 1: umbrella appears, $U_1 = true$.
Prediction from $t = 0 : 1$:

$$\begin{aligned} P(R_1) &= \sum_{r_0} P(R_1 | r_0)P(r_0) \\ &= \langle 0.7, 0.3 \rangle \cdot 0.5 + \langle 0.3, 0.7 \rangle \cdot 0.5 \\ &= \langle 0.5, 0.5 \rangle \end{aligned}$$

Then update step incorporates evidence for $t = 1$ and normalizes:

$$\begin{aligned} P(R_1 | u_1) &= \alpha P(u_1 | R_1)P(R_1) \\ &= \alpha \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle \\ &= \alpha \langle 0.45, 0.1 \rangle \\ &\approx \langle 0.818, 0.182 \rangle \end{aligned}$$



- ▶ Day 2: umbrella appears, $U_2 = true$. Prediction from $t = 1 : 2$:

$$\begin{aligned} P(R_2 | u_1) &= \sum_{r_1} P(R_2 | r_1)P(r_1 | u_1) \\ &= \langle 0.7, 0.3 \rangle \cdot 0.818 + \langle 0.3, 0.7 \rangle \cdot 0.182 \\ &\approx \langle 0.627, 0.373 \rangle \end{aligned}$$

Then update step incorporates evidence for $t = 2$:

$$\begin{aligned} P(R_2 | u_1, u_2) &= \alpha P(u_2 | R_2)P(R_2 | u_1) \\ &= \alpha \langle 0.9, 0.2 \rangle \langle 0.627, 0.373 \rangle \\ &= \alpha \langle 0.565, 0.075 \rangle \\ &\approx \langle 0.883, 0.117 \rangle \end{aligned}$$

Inference in Temporal Models

Given the general structure of a probabilistic temporal model, we can perform basic inference tasks:

- ▶ **Filtering**, a.k.a., **state estimation** is the task of computing the **belief state** $P(\mathbf{X}_t | \mathbf{e}_{1:t})$ – the posterior distribution over the most recent state given all the evidence to date.
- ▶ **Prediction** is the task of computing the posterior distribution over the future state, given all evidence to date: $P(\mathbf{X}_{t+k} | \mathbf{e}_{1:t})$ for some $k > 0$.
- ▶ **Smoothing** is the task of computing the posterior distribution over a past state, given all evidence up to the present: $P(\mathbf{X}_k | \mathbf{e}_{1:t})$ for some k such that $0 \leq k < t$.
- ▶ **Most likely explanation**: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations: $\operatorname{argmax}_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} | \mathbf{e}_{1:t})$.

Prediction

Prediction is the task of computing the posterior distribution over the future state, given all evidence to date: $P(\mathbf{X}_{t+k} | \mathbf{e}_{1:t})$ for some $k > 0$.

- ▶ Umbrella example: compute probability of rain three days from now, given all the observations to date.
- ▶ Prediction is useful for evaluating possible courses of action based on their expected outcomes.

Prediction can be seen simply as filtering without the addition of new evidence. Filtering already includes a one-step prediction. So we can take Equation 14.5:

$$P(\mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = \alpha \underbrace{P(e_{t+1} | \mathbf{X}_{t+1})}_{\text{sensor model}} \sum_{\mathbf{x}_t} \underbrace{P(\mathbf{X}_{t+1} | \mathbf{x}_t)}_{\text{transition model}} \underbrace{P(\mathbf{x}_t | \mathbf{e}_{1:t})}_{\text{recursion}} \quad (14.5)$$

remove the sensor model and extend the prediction to $t + k + 1$:

$$P(\mathbf{X}_{t+k+1} | \mathbf{e}_{1:t}) = \sum_{\mathbf{x}_{t+k}} \underbrace{P(\mathbf{X}_{t+k+1} | \mathbf{x}_{t+k})}_{\text{transition model}} \underbrace{P(\mathbf{x}_{t+k} | \mathbf{e}_{1:t})}_{\text{recursion}} \quad (14.6)$$

Making predictions is hard, especially about the future.

As we try to predict further and further into the future, the predicted distribution for rain converges to a fixed point $\langle 0.5, 0.5 \rangle$, after which it remains constant for all time.

- ▶ This is the **stationary distribution** of the Markov process defined by the transition model.
- ▶ The **mixing time** is the time it takes to reach the fixed point.
- ▶ Prediction usually only effective for $k \ll$ mixing time.

The more the uncertainty in the transition model, the shorter will be the mixing time and the more the future is obscured.

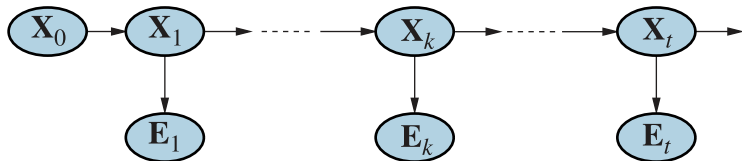
Inference in Temporal Models

Given the general structure of a probabilistic temporal model, we can perform basic inference tasks:

- ▶ **Filtering**, a.k.a., **state estimation** is the task of computing the **belief state** $P(\mathbf{X}_t | \mathbf{e}_{1:t})$ – the posterior distribution over the most recent state given all the evidence to date.
- ▶ **Prediction** is the task of computing the posterior distribution over the future state, given all evidence to date: $P(\mathbf{X}_{t+k} | \mathbf{e}_{1:t})$ for some $k > 0$.
- ▶ **Smoothing** is the task of computing the posterior distribution over a past state, given all evidence up to the present: $P(\mathbf{X}_k | \mathbf{e}_{1:t})$ for some k such that $0 \leq k < t$.
- ▶ **Most likely explanation**: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations: $\operatorname{argmax}_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} | \mathbf{e}_{1:t})$.

Smoothing

Smoothing is the task of computing the posterior distribution over a past state, given all evidence up to the present: $P(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ for some k such that $0 \leq k < t$.



- ▶ Umbrella example: compute the probability that it rained last Wednesday, given all the observations of the umbrella carrier made up to today.
- ▶ Smoothing provides a better estimate of the state at time k than was available at that time, because it incorporates more evidence.
 - ▶ When tracking a moving object with inaccurate position observations, smoothing gives a smoother estimated trajectory than filtering – hence the name

Recursive Message Passing for Smoothing

Split the computation into two parts – the evidence up to k and the evidence from $k + 1$ to t :

$$\begin{aligned} P(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= P(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\ &= \alpha P(\mathbf{X}_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) && \text{(using Bayes' Rule, given } \mathbf{e}_{1:k}) \\ &= \alpha P(\mathbf{X}_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) && \text{(using conditional independence)} \\ &= \alpha \mathbf{f}_{1:k} \odot \mathbf{b}_{k+1:t} && (14.8) \end{aligned}$$

\odot is elementwise multiplication, a.k.a. Hadamard product.

The “forward” message, $\mathbf{f}_{1:k}$ can be computed by filtering forward using Equation 14.5.

$$P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \alpha \underbrace{P(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})}_{\text{sensor model}} \sum_{\mathbf{X}_t} \underbrace{P(\mathbf{X}_{t+1} \mid \mathbf{x}_t)}_{\text{transition model}} \underbrace{P(\mathbf{x}_t \mid \mathbf{e}_{1:t})}_{\text{recursion}} \quad (14.5)$$

Backward Message for Smoothing

The backward message, $\mathbf{b}_{k+1:t}$ can be computed by a recursive process running backward from t :

$$\begin{aligned} P(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} \mid \mathbf{X}_k) && \text{(conditioning on } \mathbf{X}_{k+1}\text{)} \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} \mid \mathbf{X}_k) && \text{(by conditional independence)} \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \\ &= \sum_{\mathbf{x}_{k+1}} \underbrace{P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})}_{\text{sensor model}} \underbrace{P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})}_{\text{recursion}} \underbrace{P(\mathbf{x}_{k+1} \mid \mathbf{X}_k)}_{\text{transition model}} \end{aligned} \quad (14.9)$$

The last step follows by the conditional independence of \mathbf{e}_{k+1} and $\mathbf{e}_{k+2:t}$, given \mathbf{x}_{k+1} .

Equation 14.9 in message form is

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

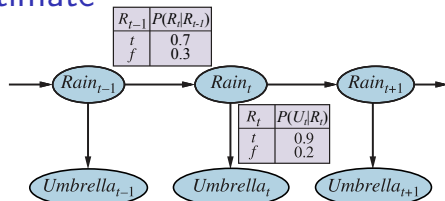
$\mathbf{e}_{t+1:t}$ is an empty sequence, so we initialize the backward phase with $\mathbf{b}_{t+1:t} = P(\mathbf{e}_{t+1:t} \mid \mathbf{X}_t) = \mathbf{1}$, where $\mathbf{1}$ is a vector of 1s.

Example: Smoothed Umbrella World State Estimate

Let's compute the smoothed estimate for the probability of rain at time $k = 1$, given the umbrella observations on days 1 and 2.

$$P(\mathbf{X}_k | \mathbf{e}_{1:t}) = \alpha P(\mathbf{X}_k | \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} | \mathbf{X}_k) \quad (14.8)$$

$$P(R_1 | u_1, u_2) = \alpha P(R_1 | u_1) P(u_2 | R_1) \quad (14.10)$$



From filtering for Day 1 we already know $P(R_1 | u_1) = \alpha P(u_1 | R_1) P(R_1) = \langle 0.818, 0.182 \rangle$.

So then we need to compute $P(u_2 | R_1)$ by applying the backward recursion in Equation 14.9:

$$P(\mathbf{e}_{k+1:t} | \mathbf{X}_k) = \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} | \mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t} | \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} | \mathbf{X}_k) \quad (14.9)$$

$$\begin{aligned} P(u_2 | R_1) &= \sum_{r_2} P(u_2 | r_2) P(r_2 | R_1) \\ &= (0.9 \cdot 1 \cdot \langle 0.7, 0.3 \rangle) + (0.2 \cdot 1 \cdot \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle \end{aligned}$$

Then plug both terms into Equation 14.10:

$$\begin{aligned} P(R_1 | u_1, u_2) &= \alpha P(R_1 | u_1) P(u_2 | R_1) \\ &= \alpha \langle 0.818, 0.182 \rangle \odot \langle 0.69, 0.41 \rangle \approx \langle 0.883, 0.117 \rangle \end{aligned} \quad (14.10)$$

Knowing it rained on Day 2 makes it more likely that it rained on Day 1.

Complexity of Smoothing

The forward and backward recursions each take constant time per step. So:

- ▶ The time complexity of smoothing with respect to evidence $e_{1:t}$ is $O(t)$ – the complexity for smoothing at a particular time step k .
- ▶ If we want to smooth the whole sequence, we could run the whole smoothing process once for each time step – a time complexity of $O(t^2)$.

However, we can apply dynamic programming to reduce the complexity to $O(t)$.

- ▶ Recall that we reused the results of the forward-filtering phase in our smoothing calculations.
- ▶ The key to the linear-time algorithm is to record the results of forward filtering over the whole sequence.
- ▶ Then we run the backward recursion from t down to 1, computing the smoothed estimate at each step k from the computed backward message $\mathbf{b}_{k+1:t}$ and the stored forward message $\mathbf{f}_{1:k}$.

This is the essence of the forward–backward smoothing algorithm ...

- ▶ BTW, you'll see this algorithmic structure again when you learn about deep neural networks.

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**

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

▶ $P(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \alpha P(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$ (14.5) and

▶ the process begins with $\mathbf{f}_{1:0} = P(\mathbf{X}_0)$.

BACKWARD($\mathbf{b}_{k+2:t}, \mathbf{e}_{k+1}$) = $P(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) =$

▶ $\sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} \mid \mathbf{X}_k)$ (14.9) and

▶ initialize backward phase with $\mathbf{b}_{t+1:t} = P(\mathbf{e}_{t+1:t} \mid \mathbf{X}_t) = \mathbf{1}$, where $\mathbf{1}$ = vector of 1s.

Inference in Temporal Models

Given the general structure of a probabilistic temporal model, we can perform basic inference tasks:

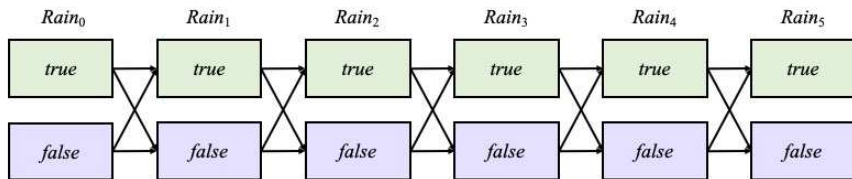
- ▶ **Filtering**, a.k.a., **state estimation** is the task of computing the **belief state** $P(\mathbf{X}_t | e_{1:t})$ – the posterior distribution over the most recent state given all the evidence to date.
 - ▶ **Prediction** is the task of computing the posterior distribution over the future state, given all evidence to date: $P(\mathbf{X}_{t+k} | e_{1:t})$ for some $k > 0$.
 - ▶ **Smoothing** is the task of computing the posterior distribution over a past state, given all evidence up to the present: $P(\mathbf{X}_k | e_{1:t})$ for some k such that $0 \leq k < t$.
- ▶ **Most likely explanation**: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations: $\operatorname{argmax}_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} | e_{1:t})$.

Finding the Most Likely Sequence

Most likely explanation: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations: $\operatorname{argmax}_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$.

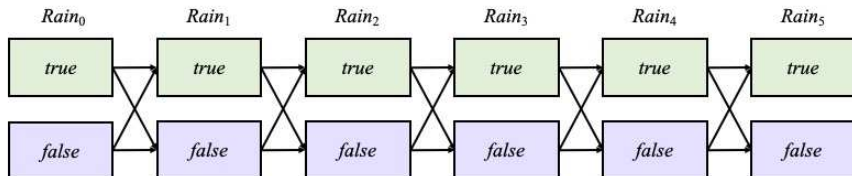
- ▶ Umbrella example: if umbrella appears on each of the first three days and is absent on the fourth, then the most likely explanation is that it rained on the first three days and did not rain on the fourth. On the other hand, perhaps the director forgot an umbrella on a rainy day, or took an umbrella on a sunny day out of caution.
- ▶ MLE algorithms are useful for speech recognition – where the aim is to find the most likely sequence of words, given a series of sounds, reconstruction of bit strings transmitted over a noisy channel, and many others.

We can represent the possible states in a system with a **state trellis**:



Most likely sequence estimation is finding the most likely path through this graph, where the likelihood of any path is the product of the transition probabilities along the path and the probabilities of the given observations at each state.

Most Likely Paths Through A State Trellis



Say we want to estimate the most likely path that reaches $Rain_5 = true$.

- ▶ By the Markov property, the most likely path to $Rain_5 = true$ consists of a path to a state at $t = 4$ followed by a transition to $Rain_5 = true$.
- ▶ The state at $t = 4$ that becomes part of the path to $Rain_5 = true$ is whichever state maximizes the likelihood of that path.

So there is a recursive relationship between most likely paths to each state x_{t+1} and most likely paths to each state x_t . We can use this property to construct a recursive algorithm for computing the most likely path given the evidence.

Viterbi Algorithm

We will use a recursively computed message $\mathbf{m}_{1:t}$, like the forward message $\mathbf{f}_{1:t}$ in the filtering algorithm:

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

By applying a similar derivation to the one we used for the filtering equation (14.5), we get:

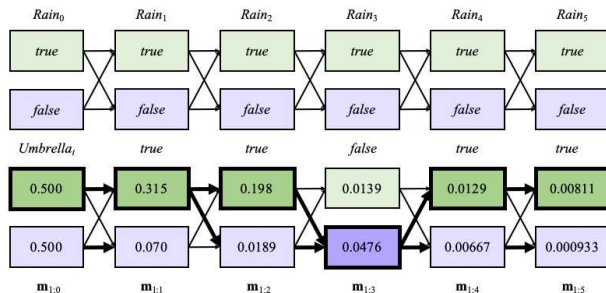
$$\mathbf{m}_{1:t+1} = P(e_{t+1} | \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} P(\mathbf{X}_{t+1} | \mathbf{x}_t) \max_{\mathbf{x}_{1:t-1}} P(\mathbf{x}_{1:t-1}, \mathbf{x}_t, \mathbf{e}_{1:t}) \quad (14.11)$$

- ▶ The final term $P(\mathbf{x}_{1:t-1}, \mathbf{x}_t, \mathbf{e}_{1:t})$ is exactly the entry for the particular state \mathbf{x}_t in the message vector $\mathbf{m}_{1:t}$.
- ▶ Equation 14.11 is essentially identical to the filtering equation (14.5) except that the summation over \mathbf{x}_t in Equation (14.5) is replaced by the maximization over \mathbf{x}_t in Equation (14.11).
- ▶ there is no normalization constant α in Equation (14.11).

Thus, the algorithm for computing the most likely sequence is similar to filtering: it starts at time 0 with the prior $\mathbf{m}_{1:0} = P(\mathbf{X}_0)$ and then runs forward along the sequence, computing the \mathbf{m} message at each time step using Equation (14.11).

Operation of Viterbi Algorithm in Umbrella World

Operation of the Viterbi algorithm for the umbrella observation sequence $[true, true, false, true, true]$, where the evidence starts at time 1.



- ▶ For each t , we show the values of the message $m_{1:t}$, which gives the probability of the best sequence reaching each state at time t .
- ▶ For each state, the bold arrow leading into it indicates its best predecessor as measured by the product of the preceding sequence probability and the transition probability.
- ▶ Following the bold arrows back from the most likely state in $m_{1:5}$ gives the most likely sequence, shown by the bold outlines and darker shading.

Practical computational considerations

For longer sequences, likely to get floating point underflow. Two solutions:

- ▶ Normalilze m at each step. Doesn't affect correctness because $max(cx, cy) = c \cdot max(x, y)$.
- ▶ Use log probabilities with addition in place of multiplication. Correctness is unaffected because the log function is monotonic, so $max(\log x, \log y) = \log max(x, y)$

Fun homework: [Markov Chains Recognition](#)