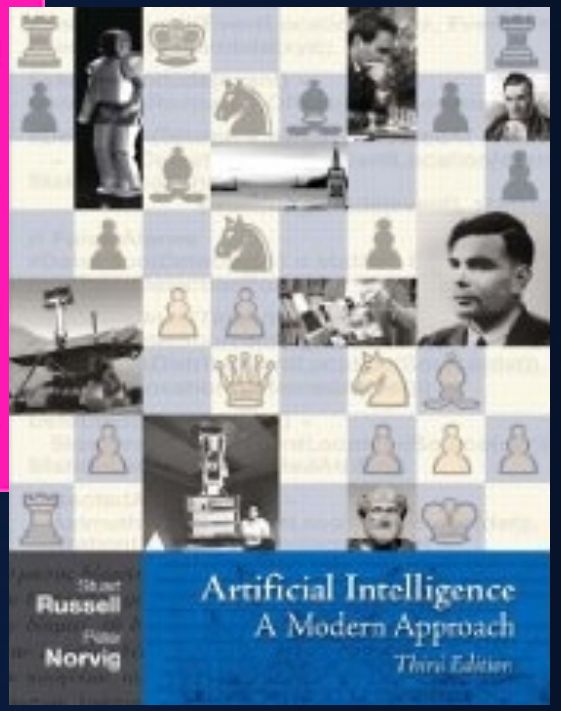


AI Fundamentals: uncertain reasoning

Maria Simi

Probabilistic Reasoning over time AIMA chapter 15

LESSON 3: TIME AND UNCERTAINTY - INFERENCE IN TEMPORAL MODELS

Reasoning in a changing world

So far we have been doing uncertain reasoning in a static world.

A changing world is modeled using a variable for each aspect of the world state *at each point in time*.

For reasoning in an evolving world one needs:

- A **belief state:** the states of the world that are possible
- A **transition model:** to predict how the world will evolve
- A **sensor model:** to update the belief state from perceptions

The transition and sensor model themselves may be uncertain:

- the *transition model* gives the probability distribution of the variables at time *t*, given the state of the world at past times;
- the *sensor model* describes the probability of each percept at time *t*, given the current state of the world.

Examples

1. **Treating a diabetic patient.** The task is to assess the current state of the patient, including the actual blood sugar level and insulin level from other observable parameters such as recent insulin doses, food intake, blood sugar measurements, and other physical signs.
2. **The umbrella example.** *You are the security guard stationed at a secret underground installation. You want to know whether it's raining today, but your only access to the outside world occurs each morning when you see the director coming in with, or without, an umbrella.*

In both cases, and in many real world scenarios, the dynamic aspects of the problem are essential for a correct prediction.

States and observations

We view the world as a series of snapshots, or **time slices**, each of which contains a set of random variables, some observable and some not.

$\mathbf{X}_t$: will denote the set of state unobservable (hidden) variables at time t

$\mathbf{E}_t = \boldsymbol{e}_t$ will denote the set of observations (**evidence variables**) at time t; $\boldsymbol{e}_t$ the values

We will assume that the state sequence starts at t=0.

We will use the notation $a{:}b$ to denote the sequence of integers from a to b (inclusive), and the notation $\mathbf{X}_{a:b}$ to denote the set of variables from $\mathbf{X}_a$ to $\mathbf{X}_b$.

Our umbrella world is represented by a sequence of evidence variables $\mathbf{E}_t = \{U_t\}$ (whether the umbrella appears) and state unobservable variables $\mathbf{X}_t = \{R_t\}$ (it's raining):

state variables: R_0, R_1, R_2, …

evidence variables: U_1, U_2, …

Transition model

The transition model specifies **how the world evolves**, i.e. the probability distribution over the latest state variables, given the previous values starting from time 0:

$\mathbf{P}(\mathbf{X}_t \,|\mathbf{X}_{0:t-1})$

The sequence of states can become very large, unbounded as t increases.

Markov assumptions: the transition model specifies the probability distribution over the latest state variables, given a **finite fixed number** of previous states:

$\mathbf{P}(\mathbf{X}_t \,|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \,|\mathbf{X}_{t-1})$ *first order* Markov process/chain

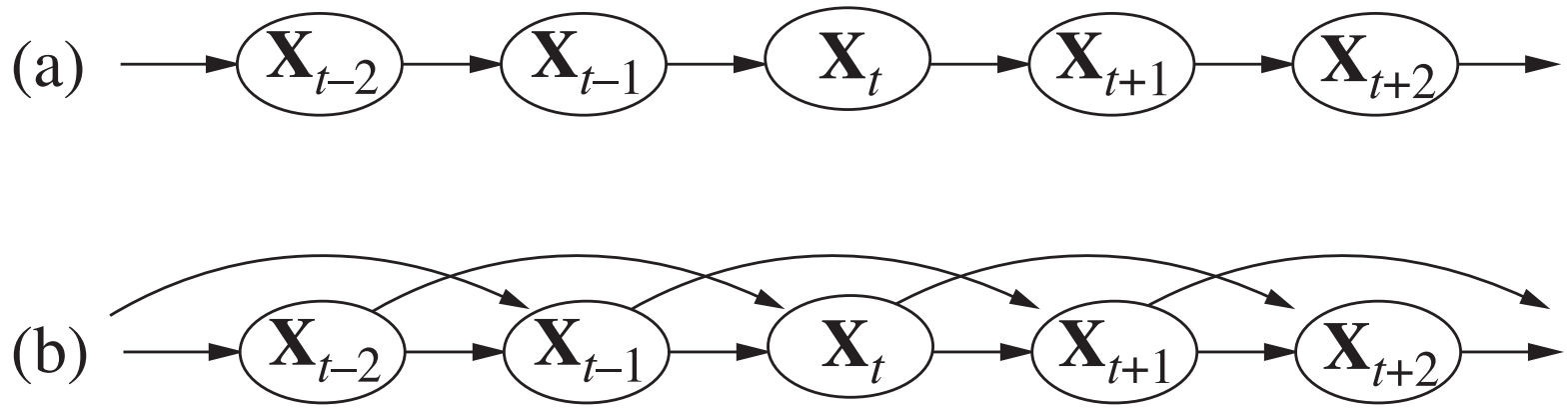
$\mathbf{P}(\mathbf{X}_t \,|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \,|\mathbf{X}_{t-1}, \mathbf{X}_{t-2})$ *second order* Markov process/chain

Additionally, we assume a **stationary process**, i.e. change is governed by laws that do not themselves change over time:

$\mathbf{P}(\mathbf{X}_t \,|\mathbf{X}_{t-1})$ is the same for all t $\mathbf{P}(r_t \,|R_{t-1}) = <0.7, 0.3>$

Transition model - graphically

Bayes neworks under different Markov [independence] assumptions



a. *first order* Markov process/chain
b. *second order* Markov process/chain

Sensor model

The **sensor/observation model** under Markov sensor assumption, postulates that evidence only depends on the current state

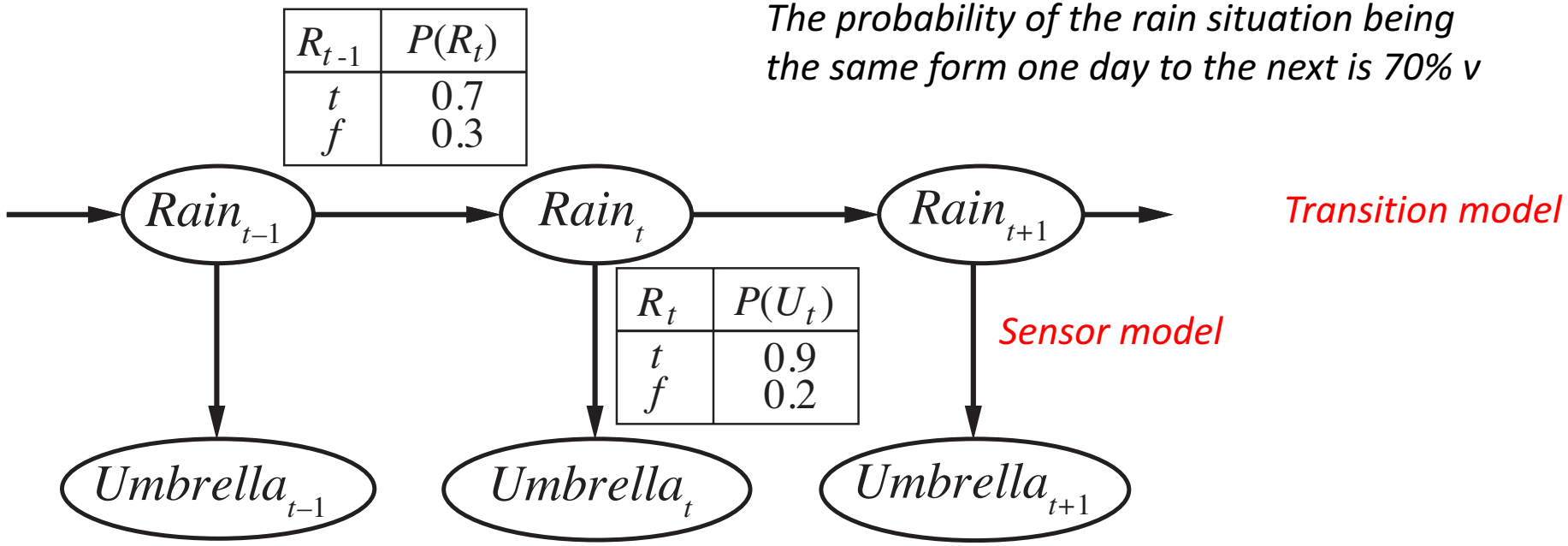
$$\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$$

Putting all together, the **complete join distribution** over all the variables, for any t:

$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \mathbf{P}(\mathbf{X}_0) \prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})\, \mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$$

$\mathbf{P}(\mathbf{X}_0)$	the initial state model over set of variables **X**
$\mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{t-1})$	the transition model under *Markov assumption*
$\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$	the sensor model under *Markov sensor assumption*

The Bayesian net for the *Umbrella* example



The transition model is $P(R_t \mid R_{t-1})$

The sensor model is $P(U_t \mid R_t)$

Hidden Markov Models

Hidden Markov Models (HMM) are temporal probabilistic models, under the Markov assumptions, in which the state of the process is described by a **single discrete random variable**.

The Umbrella example is an HMM, since states are described by a single state variable.

Releasing the assumptions:

Assuming that 'Rain' only depends on rain the previous day may be a too strong assumption. There are two ways to improve the accuracy of the approximation:.

1. Increasing the order of the Markov process model, for example use a second order assumption.
2. Increasing the set of state variables: we could add *Season, Temperature, Humidity, Pressure* … as state variables. This may imply adding new sensors.

Inference in temporal models

FILTERING, PREDICTION, SMOOTHING, MOST LIKELY EXPLANATION

Inference in temporal models

Basic inference tasks based on the temporal model:

- **Filtering**, or **state estimation**: compute the belief state (*posterior probability distribution*) given evidence from previous and current states. Related is the likelihood of the evidence sequence.
- **Prediction:** computing the posterior distribution over a **future** state, given all evidence to date.
- **Smoothing:** computing the posterior distribution over a **past** state, given all evidence up to the present. Looking back with the knowledge of today provides a more accurate estimate.
- **Most likely explanation/sequence**: given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations. Ex. In speech the most likely sequence of words, given a series of sounds.
- **Learning**: The transition and sensor models, can be learned from observations.

Filtering

A filtering algorithm maintains a current state estimate and updates it, rather than going back over the entire history of percepts for each update

The filtering function f takes into account the new evidence and the state estimation computed for the previous time.

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

This process is called **recursive estimation** and can be seen as made of two parts:

1. **Prediction**: the current state distribution is projected forward from t to $t+1$: $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$
2. **Filtering**: then it is updated using the new evidence e_{t+1}: $\boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1})$

Computing filtering

It can be computed as follows:

$$\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1}) \quad \textit{Dividing up the evidence}$$

$$= \alpha\, \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}\boldsymbol{e}_{1:t})\, \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}) \quad \textit{Bayes's rule to } \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1})$$

$$= \alpha\, \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1})\, \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}) \quad \textit{Markov's sensor assumption}$$

$$= \alpha\, \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t, \boldsymbol{e}_{1:t})\ \boldsymbol{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t}) \quad \textit{Summing different values of } \boldsymbol{X}_t$$

$$= \alpha\, \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)\ \boldsymbol{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t}) \quad \textit{Markov's sensor assumption}$$

Finally: $\boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \alpha\, \boldsymbol{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)\ \boldsymbol{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})$ (*Filtering*)

----update--- *----------predict----------*

We can think at $\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t})$ as a *message* $f_{1:t+1}$ that is propagated forward in the sequence. This makes evident the recursive structure:

$f_0 = \boldsymbol{P}(\boldsymbol{X}_0)$

$f_{1:t+1} = \alpha\, \mathit{Forward}(f_{1:t}, \boldsymbol{e}_{t+1})$ where *Forward* implements the *filtering* update

Computing filtering for umbrellas

Day 0: $\boldsymbol{P}(R_0) = \langle 0.5, 0.5 \rangle$ f_0 no observation, a 50% chance of rain

Day 1:

Predict: $P(\boldsymbol{R}_1) = \sum_{r_0} \boldsymbol{P}(\boldsymbol{R}_1 | r_0)\, \boldsymbol{P}(r_0) = \boldsymbol{P}(R_1|\mathrm{r}_0) \times P(\mathrm{r}_0) + \boldsymbol{P}(r_1|\neg \mathrm{r}_0) \times P(\neg \mathrm{r}_0) =$

$= \langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$

Update, observing $U_1 = \text{true}$

$\boldsymbol{P}(\boldsymbol{R}_1|u_1) = \alpha\, \boldsymbol{P}(u_1 |\boldsymbol{R}_1)\, P(\boldsymbol{R}_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle = \alpha \langle 0.45, 0.1 \rangle \approx \langle 0.818, 0.182 \rangle$

Day 2: $f_{1:1}$

Predict: $P(R_2) = \sum_{r_1} P(R_2 \mid r_1)\, \boldsymbol{P}(r_1) = \langle 0.7, 0.3 \rangle \times 0.818 + \langle 0.3, 0.7 \rangle \times 0.182 = \langle 0.627, 0.373 \rangle$

Update, observing $U_2 = \text{true}$

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

The probability of rain increases from day 1 to day 2 because rain persists.

Predictions

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

The filtering process already incorporates a **one-step** prediction.

In general at time $t+k+1$:

$$\boldsymbol{P}(\boldsymbol{X}_{t+k+1}|\boldsymbol{e}_{1:t}) = \sum_{X_{t+k}} \boldsymbol{P}(\boldsymbol{X}_{t+k+1}|x_{t+k})\ \boldsymbol{P}(x_{t+k}|\boldsymbol{e}_{1:t})$$

This computation involves only the transition model and not the sensor model.

We can show that the predicted distribution for rain converges to a fixed point $\langle 0.5, 0.5 \rangle$, after which it remains constant for all time (the **stationary distribution** of the Markov process).

The ***mixing time*** is the time to reach the stationary distribution.

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

Likelihood of evidence sequence

We can use a forward recursion to compute the likelihood of the evidence sequence, $P(\boldsymbol{e}_{1:t})$.

Useful if we want to compare two models producing the same evidence sequence.

For this recursion, we can derive a recursive equation similar to filtering and use a likelihood message

$$\ell_{1:t}(X_t) = P(X_t, e_{1:t})$$

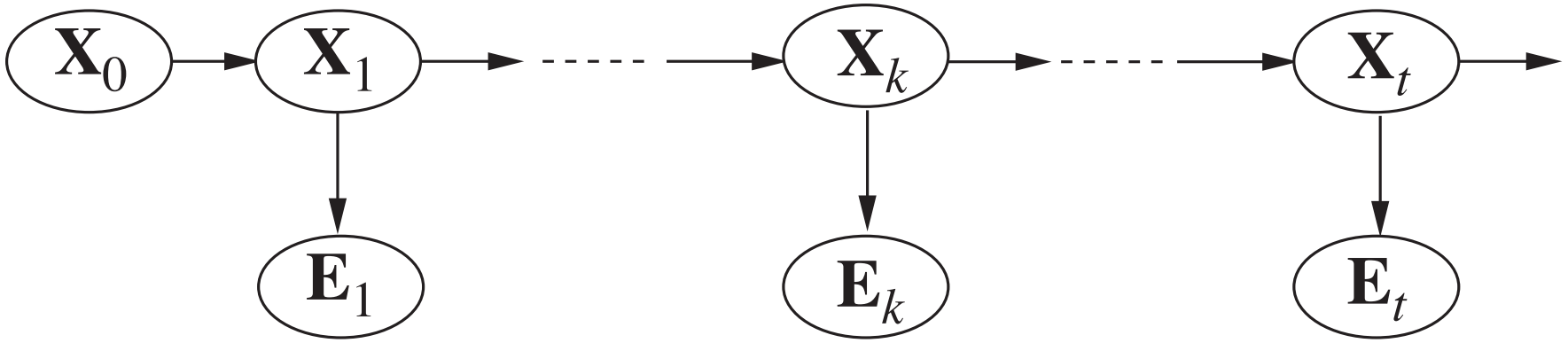
Once we have $\ell_{1:t}(X_t)$ we can compute the likelihood of the evidence sequence by summing out on the values of X_t

$$L_{1:t} = P(\boldsymbol{e}_{1:t}) = \sum_{X_t} \ell_{1:t}(x_t)$$

Smoothing

Smoothing is the process of computing the posterior distribution of the state at **some past time** k given a complete sequence of observations up to the present t

$\boldsymbol{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:t})$ for $0 \leq k < t$.



Computing smoothing

$$\boldsymbol{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:t}) = \boldsymbol{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:k}, \boldsymbol{e}_{k+1:t}) \qquad \textit{Splitting the evidence}$$

$$= \alpha\, \boldsymbol{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:k})\, \mathbf{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k, \boldsymbol{e}_{1:k}) \qquad \textit{Bayes's rule}$$

$$= \alpha\, \boldsymbol{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:k})\, \mathbf{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k) \qquad \textit{Conditional independence}$$

$$= \alpha\, f_{1:k} \times b_{k+1:t} \qquad \times \text{ is pointwise multiplication}$$

We define another *message* $b_{k+1:t} = \mathbf{P}(\boldsymbol{e}_{k+1:t}|\, \boldsymbol{X}_k)$ that can be computed by a recursive process that runs **backward** from t.

The forward message $f_{1:k}$ is defined as before for filtering

The terms in $f_{1:k}$ and $b_{k+1:t}$ can be implemented by two recursive calls, one running forward from 1 to k and using the *filtering equation*, the other running backward from t to $k+1$

Computing smoothing going backwards

$$\boldsymbol{P}(\boldsymbol{e}_{k+1:t}|\,\boldsymbol{X}_k) = \sum_{k+1} \boldsymbol{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_{1:k}, \boldsymbol{x}_{k+1})\, \boldsymbol{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k) \qquad \textit{conditioning on } \boldsymbol{X}_{k+1}$$

$$= \sum_{k+1} \boldsymbol{P}(\boldsymbol{e}_{k+1:t}|\,x_{k+1})\, \boldsymbol{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k) \qquad \textit{the evidence only depends on } \boldsymbol{x}_{k+1}$$

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

$$= \sum_{k+1} P(\boldsymbol{e}_{k+1}|\,x_{k+1})\, P(\boldsymbol{e}_{k+2:t}|\,x_{k+1})\, \boldsymbol{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k) \qquad \textit{conditional independence}$$

$b_{k+1:t} = P(\boldsymbol{e}_{k+2:t}|\,x_{k+1})$ is defined by the previous equation and implemented by *Backword*.

The backward phase starts with $\mathbf{P}(\boldsymbol{e}_{t+1:t}|\,\boldsymbol{X}_t) = \mathbf{P}(\,|\,\boldsymbol{X}_t)\,\mathbf{1}$ **(a vector of 1's)**, because $(t{+}1{:}t)$ is an empty sequence with probability 1 of observing it.

$$b_{t+1:t} = \mathbf{P}(\,|\,\boldsymbol{X}_t)\,\mathbf{1}$$

$$b_{k+1:t} = \textit{Backward}(b_{k+2:t}, \boldsymbol{e}_{k+1})$$

where *Backward* implements the update defined before

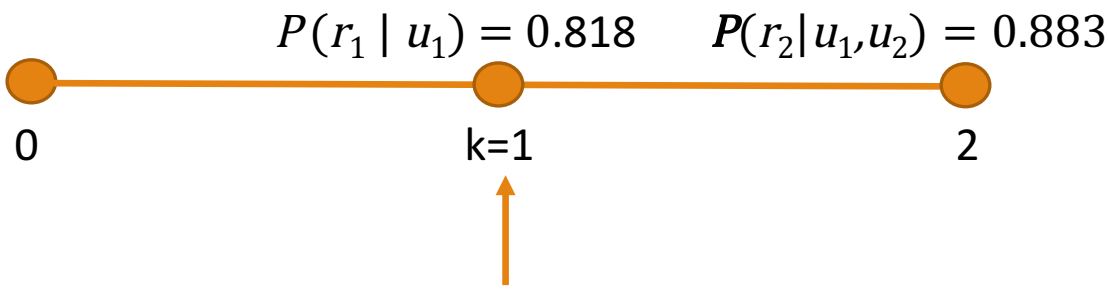
Computing the smoothed estimate for *rain*

Let's compute the smoothed estimate at time k=1 for the probability of rain. Given the observations at time 1 and 2.

$$\boldsymbol{P}(R_1 \mid u_1, \underline{u}_2) = \alpha\, \boldsymbol{P}(R_1 \mid u_1)\, \boldsymbol{P}(u_2 \mid R_1)$$

$$\boldsymbol{P}(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$$

already computed in the filtering stage.



$$\boldsymbol{P}(u_2 \mid R_1) = \sum_{r_2} P(u_2 \mid r_2)\, P(\mid r_2)\, \boldsymbol{P}(\boldsymbol{r}_2 \mid R_1)$$

$$= P(u_2 \mid r_2)\, P(\mid r_2)\, \boldsymbol{P}(\boldsymbol{r}_2 \mid R_1) + P(u_2 \mid \neg r_2)\, P(\mid \neg r_2)\, \boldsymbol{P}(\neg \boldsymbol{r}_2 \mid R_1)$$

$$= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle$$

$$\boldsymbol{P}(R_1 \mid u_1, \underline{u}_2) = \alpha\, \langle 0.818, 0.182 \rangle \times \langle 0.69, 0.41 \rangle \approx \langle 0.883, 0.117 \rangle$$

The **smoothed estimate** for rain on day 1 is higher than the *filtered estimate*.

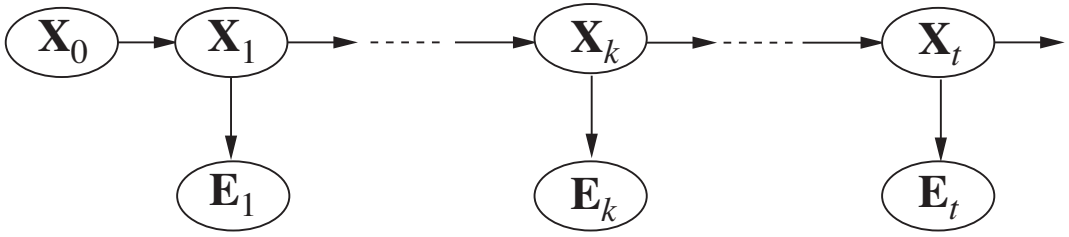
The complexity for smoothing at a specific time k with respect to evidence $e_{1:t}$ is O(t). For smoothing a sequence O(t^2).

Improving efficiency

The algorithm, called **forward–backward**, uses dynamic programming to reduce the complexity over the whole sequence to O(*t*).

The trick is to record the results computed during the forward filtering phase, over the whole sequence, and reuse them during the backward phase.

Note that this is consistent with the fact that the Bayesian network structure is a **polytree**.



The *forward–backward* algorithm is the basis for many applications that deal with sequences of noisy observations.

Improvements are required to deal with (1) *space complexity* for long squences and (2) *online computations*, where new observations continuously arrive (**fixed-lag smoothing**)

Forward–backward algorithm for smoothing

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$[0] \leftarrow$ *prior*
  for $i = 1$ **to** t **do**
    fv$[i] \leftarrow$ FORWARD(**fv**$[i-1]$, **ev**$[i]$)
  for $i = t$ **downto** 1 **do**
    sv$[i] \leftarrow$ NORMALIZE(**fv**$[i] \times$ **b**)
    b $\leftarrow$ BACKWARD(**b**, **ev**$[i]$)
  return sv

Smoothing all the sequence

Forward filtering phase

Backward phase reusing result from the filtering

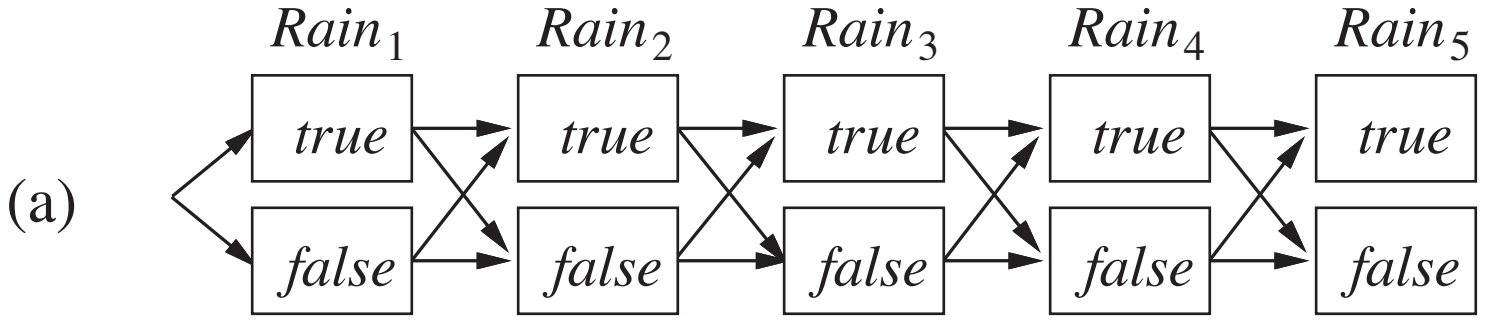
Finding the most likely sequence

Suppose we observe [*true, true, false, true, true*] for the *Umbrella* variable.

What is the most likely sequence for *Rain*?

There are 2^5 possible *Rain* sequences, we want to discover which is the one maximizing the likelihood, **in linear time**.

Each sequence is a path through a graph whose nodes are the possible states at each time step.



Likelihood of a path: product of the transition probabilities along the path and the probabilities of the given observations at each state

Computing the most likely sequence

We can write a recursive equation:

$$\max_{x1\ldots xt} \boldsymbol{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, X_{t+1} \mid \boldsymbol{e}_{1:t+1}) =$$

$$= \alpha\, \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \max_{xt} [\boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) \max_{x1\ldots xt-1} \boldsymbol{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})]$$

The forward message in this case is:

$$\mathbf{m}_{1:t} = \max_{x1\ldots xt-1} \boldsymbol{P}(X_1, \ldots, X_{t-1}, \boldsymbol{X}_t \mid \boldsymbol{e}_{1:t})$$

the probabilities of the most likely path to each state X_t

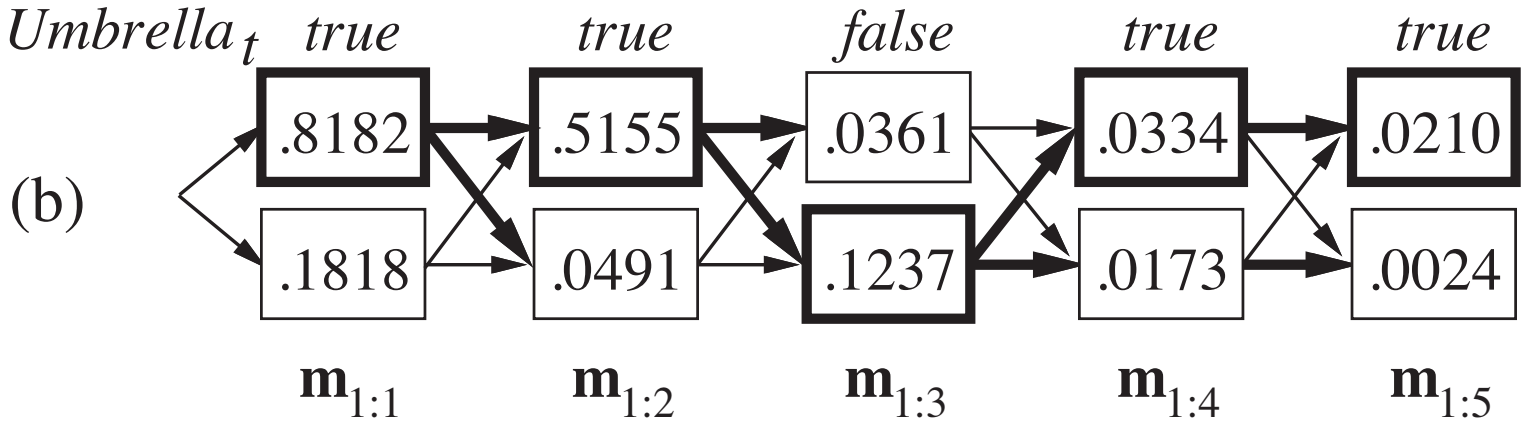
The most likely sequence overall can be computed in one pass.

For each state, the best state that leads to it is recorded (marked as black arrows in the example) so that the optimal sequence is identified by following black arrows backwards from the best final state.

This algorithm is the famous **Viterbi algorithm**, named after A. Viterbi [1967]

0,05454

Viterbi algorithm on the Umbrella example



$\mathbf{m}_{1:1} = \alpha\, P(u_1 \mid R_1)\, P(R_1) = \langle \mathbf{0.8182, 0.1818} \rangle$

$\mathbf{m}_{1:2} = \alpha\, P(u_2 \mid R_2) \times \max_{r1}[P(R_2 \mid r_1) \times \mathbf{m}_{1:1}] =$ *we need to consider all possible states at time 1*

$= \alpha \langle \mathbf{0.9}, \mathbf{0.2} \rangle \times \langle \max_{true}[\langle 0.7, 0.3 \rangle \times \langle 0.8182, 0.1818 \rangle], \max_{false}(\langle 0.3, 0.7 \rangle \times \langle 0.8182, 0.1818 \rangle] \rangle =$

$= \alpha \langle 0.9, 0.2 \rangle \times \langle \max_{true}(0.5155, 0.0545), \max_{false}(0.2454, 0.1273) \rangle =$

$= \alpha \langle \mathbf{0.9}, \mathbf{0.2} \rangle \times \langle 0.5155, 0.2454 \rangle = \alpha \langle 0.5155, 0.491 \rangle$

$\mathbf{m}_{1:3} = \alpha\, P(\neg u_3 \mid R_3) \times \max_{r2}[P(R_3 \mid r_2) \times \mathbf{m}_{1:2}] = \langle 0.361, 0.1237 \rangle$

$\mathbf{m}_{1:4} = \alpha\, P(u_4 \mid R_4) \times \max_{r3}[P(R_3 \mid r_2) \times \mathbf{m}_{1:3}] = \langle 0.334, 0.173 \rangle$

$\mathbf{m}_{1:5} = \alpha\, P(u_5 \mid R_5) \times \max_{r4}[P(R_4 \mid r_3) \times \mathbf{m}_{1:4}] = \langle 0.2010, 0.0024 \rangle$

Applications of the Viterbi algorithm

- In telecommunications Viterbi decoders are used for decoding a bitstream encoded using a technique called **convolutional code** or **trellis code**. (the original application by Viterbi).
- In natural language processing:
  - Different kinds of “sequence tagger”: PoS taggers, NE taggers …
  - Dependency parsing
  - Speech recognition (speech-to-text), speech synthesis, speech diarization (recognizing sequences from the same speaker), keyword spotting …
- ….

Hidden Markov Model: reformulation

S: number of states for the [unique] hidden variable X_t

X_t values are denoted by integers 1, . . . , S

Transition model:

$P(X_t \mid X_{t-1})$ is the S x S matrix **T** with $\mathbf{T}_{ij} = P(X_t = j \mid X_{t-1} = i)$

$\mathbf{T}_{ij}$ is the probability of a transition from state i to state j.

Sensor model:

Is an S x S diagonal matrix O_t whose i^{th} diagonal entry is $P(e_t \mid X_t = i)$ and whose other entries are 0.

The **forward equation** becomes:

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

and the **backward equation** becomes

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

Example:

$$\mathbf{T} = \boldsymbol{P}(X_t \mid X_{t-1}) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$$

$$O_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$$

$$O_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$$

Conclusions

- ✓ A dynamic world can be represented using a set of random variables to represent the state at each point in time. A special case of Bayesian network.
- ✓ A temporal probability model can be thought of as containing a **transition model** describing the state evolution and a **sensor model** describing the observation process.
- ✓ Assuming the **Markov property** and **stationary processes** greatly simplifies the representation and the computation.
- ✓ The principal inference tasks in temporal models are: **filtering**, **prediction**, and **computing the most likely explanation**. Simple, recursive algorithms exist *linear* in the length of the sequence.
- ✓ Algorithms can be further simplified assuming **Hidden Markov Models**, i.e. states are represented by a single discrete variable.
- ✓ **Hidden Markov models**, **Kalman filters**, and **dynamic Bayesian networks** are left out.

Your turn

- ✓ Study and implement the Forward–Backward algorithm for smoothing
- ✓ Study and implement the Viterbi algorithm for computing the most likely sequence.
- ✓ Describe a problem that can be tackled with Viterbi.

References

- ✓ Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach* (3rd edition). Pearson Education 2010 [Cap 14 – Probabilistic reasoning]