Artificial Intelligence 2

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Today program

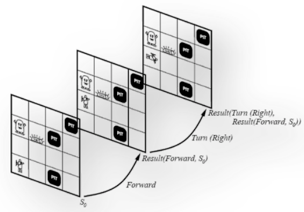
Agent in **partially observable environment** maintains a belief state from the percepts observed and a **sensor model** and using a **transition model** the agent can predict how the world might evolve in the next time step.

- a **belief state** represents which states of the world are currently possible (by explicit enumeration of states or by logical formulas)
- the probability theory allows to quantify the degree of belief in elements of the believe state
- we can also describe probability of state transitions

Probabilistic reasoning over time

- representation of state transitions
- basic inference tasks
- inference algorithms for temporal models
- specific kinds of models (hidden Markov models, dynamic Bayesian networks)

Time and uncertainty



In **situation calculus**, we view the world as a series of snapshots (**time slices**).
A similar approach can be applied in probabilistic reasoning.

Each time slice (**state**) is described as a set of random variables:

- **hidden** (not observable) random **variables $\mathbf{X}_t$**
- **observable** random **variables $\mathbf{E}_t$** (with observed values $\mathbf{e}_t$)

t is an identification of the time slice (we assume **discrete time** with uniform time steps)

Notation:

- $X_{a:b}$ denotes a set of variables from X_a to X_b

A model example (umbrella world)

You are the security guard stationed at a secret underground installation and you want to know whether it is **raining today**:

- hidden random variable $\mathbf{R}_t$

But your only access to the outside world occurs each morning when you see the the director coming in **with, or without, an umbrella**.

- observable random variable $\mathbf{U}_t$

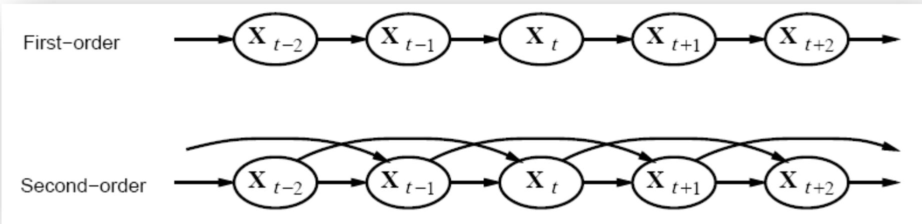
Transition model

The **transition model** specifies the probability distribution over the latest state variables given the previous values.

This is given by $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1})$.

Problem #1: the set $\mathbf{X}_{0:t-1}$ is unbounded in size as t increases

- we can make a **Markov assumption** – the current state depends only on a finite fixed number of previous states; processes satisfying this assumption are called **Markov processes** or **Markov chains**
- **first-order Markov chain** – the current state depends only on the previous state $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$



Problem #2: there are infinitely many possible values of t

- We assume that changes in the world state are caused by a **stationary process** (a process of change is governed by laws that do not themselves change very time)
- the conditional probability tables $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ are identical for all t

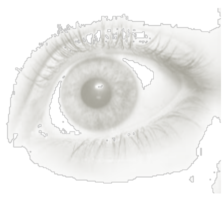
Sensor model

A **sensor (observation) model** describes how the evidence (observed) variables $\mathbf{E}_t$ depend on other variables.

They could depend on previous variables as wells as the current state variables.

We make a **sensor Markov assumption** – the evidence variables depend only on the hidden state variables $\mathbf{X}_t$ from the same time.

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

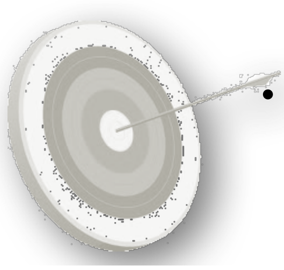


Improving model accuracy

The first-order Markov assumption says that the state variables contain all the information needed to characterize the probability distribution for the next time slice.

What if this assumption is only approximate?

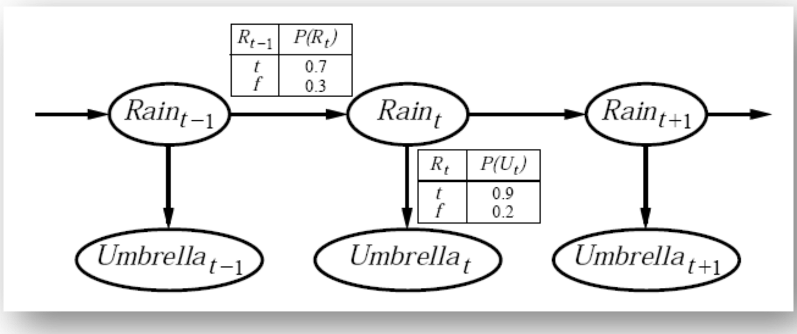
- **increase the order** of the Markov process model
- **increase the set** of state variables
  - For example we could add $Season_t$ to incorporate historical records or we could add $Temperature_t$, $Humidity_t$, $Pressure_t$ to use a physical model of rainy conditions.
  - The first solution (increasing the order) can always be reformulated as an increase in set of state variables.



A Bayesian network view

The transition and sensor models can be described using a **Bayesian network**.

In addition to $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ and $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$ we need to say how everything gets started $\mathbf{P}(\mathbf{X}_0)$.

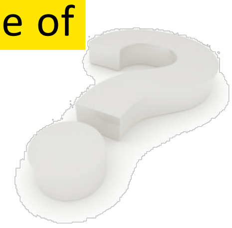


We have a specification of the complete joint distribution:

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

Basic inference tasks

- **Filtering**: the task of computing the posterior distribution over the *most recent state*, given all evidence to date
  $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$
- **Prediction**: the task of computing the posterior distribution over the *future state*, given all evidence to date
  $\mathbf{P}(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ for $k>0$
- **Smoothing**: the task of computing posterior distribution over a *past state*, given all evidence up to the present
  $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ for $k: 0 \le k < t$
- **Most likely explanation**: the task to find the sequence of states that is most likely generated a given sequence of observations
  $\text{argmax}_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$



Filtering

The task of computing the posterior distribution over the *most recent state,* given all evidence to date – $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$.
A useful filtering algorithm needs to maintain a current state estimate and update it, rather than going back over (**recursive estimation**):

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

How to define the function f?

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

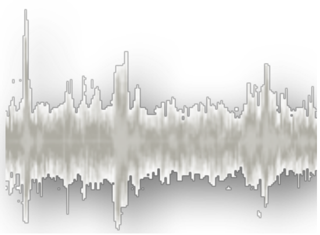
Bayes rule

sensor Markov assumption

conditioning
$\mathbf{P}(\mathbf{Y}) = \Sigma_z \mathbf{P}(\mathbf{Y}|\mathbf{z})\, P(\mathbf{z})$

A message $\mathbf{f}_{1:t}$ is propagated forward over the sequence:

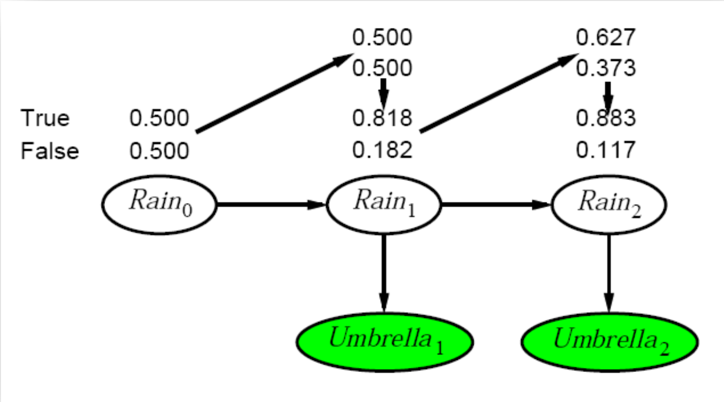
$\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t}) = \mathbf{f}_{1:t}$
$\mathbf{f}_{1:t+1} = \alpha\, \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$
$\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0)$

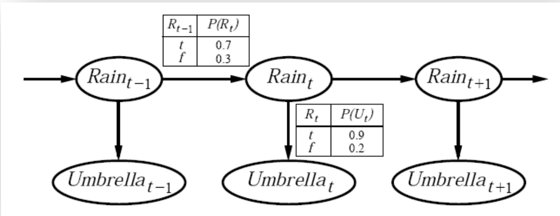


Filtering (example)

$\mathbf{P}(R_{t+1} | \mathbf{u}_{1:t+1})$

$= \alpha\, \mathbf{P}(\mathbf{u}_{t+1} | R_{t+1})\, \mathbf{P}(R_{t+1} | \mathbf{u}_{1:t}) = \alpha\, \mathbf{P}(\mathbf{u}_{t+1} | R_{t+1})\, \Sigma_{r_t}\, \mathbf{P}(R_{t+1} | r_t)\, P(r_t | \mathbf{u}_{1:t})$





$\mathbf{P}(R_0) = \langle 0.5, 0.5\rangle$

$\mathbf{P}(R_1)$

$= \Sigma_{r_0}\, \mathbf{P}(R_1 | r_0)\, P(r_0)$

$= \langle 0.5, 0.5\rangle$

$\mathbf{P}(R_1 | \mathbf{u}_1)$

$= \alpha\, \mathbf{P}(\mathbf{u}_1 | R_1)\, \mathbf{P}(R_1)$

$= \alpha\, \langle 0.9, 0.2\rangle \langle 0.5, 0.5\rangle$

$\approx \langle 0.818, 0.182\rangle$

$\mathbf{P}(R_2 | \mathbf{u}_1)$

$= \Sigma_{r_1}\, \mathbf{P}(R_2 | r_1)\, P(r_1 | \mathbf{u}_1)$

$= \langle 0.7, 0.3\rangle \times 0.818$

$+ \langle 0.3, 0.7\rangle \times 0.182$

$\approx \langle 0.627, 0.372\rangle$

$\mathbf{P}(R_2 | \mathbf{u}_1, \mathbf{u}_2)$

$= \alpha\, \mathbf{P}(\mathbf{u}_2 | R_2)\, \mathbf{P}(R_2 | \mathbf{u}_1)$

$= \alpha\, \langle 0.9, 0.2\rangle \langle 0.627, 0.372\rangle$

$= \langle 0.883, 0.117\rangle$

Prediction

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

We can see this task as filtering without the addition of new evidence:

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

After some time (**mixing time**) the predicted distribution converges to the **stationary distribution** of the Markov process and remains constant.

Smoothing

The task of computing posterior distribution over a *past state*, given all evidence up to the present – $\mathbf{P}(\mathbf{X}_k | \mathbf{e}_{1:t})$ for k: $0 \le k < t$.

We again exploit a recursive message-passing approach, now in two parts.

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

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

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

$= \alpha\, \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}$

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

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

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

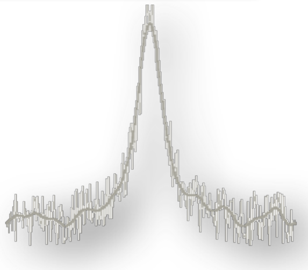
$= \Sigma_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} | \mathbf{x}_{k+1})\, P(\mathbf{e}_{k+2:t} | \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} | \mathbf{X}_k)$ — **conditional independence**

Using the backward message-passing notation:

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

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

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



Smoothing (example)

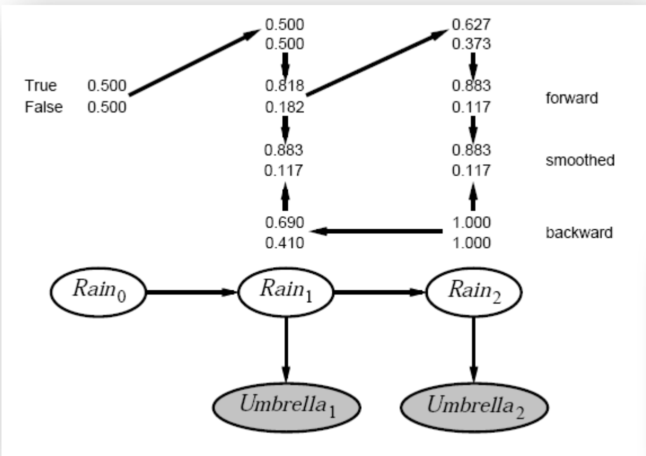
$\mathbf{P}(R_k | u_{1:t+1}) = \alpha\, \mathbf{P}(R_k | u_{1:k})\, \mathbf{P}(u_{k+1:t} | R_k)$

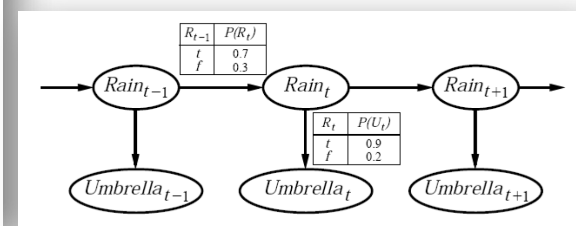
$P(u_{k+1:t} | R_k) = \Sigma_{r_{k+1}} P(u_{k+1} | r_{k+1})\, P(u_{k+2:t} | r_{k+1})\, \mathbf{P}(r_{k+1} | R_k)$

$\mathbf{P}(\,| R_2) = \mathbf{1}$

$\mathbf{P}(u_2 | R_1) = \Sigma_{r_2} P(u_2 | r_2)\, P(\,| r_2)\, \mathbf{P}(r_2 | R_1)$

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



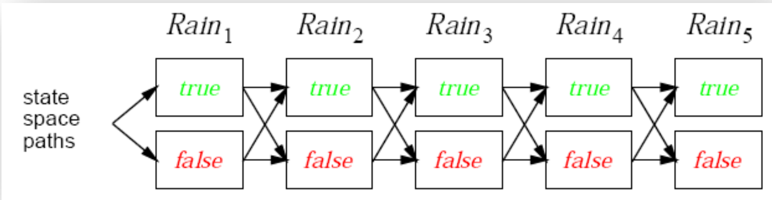


Most likely explanation/sequence

The task to find the sequence of states that is most likely generated a given sequence of observations $\text{argmax}_{x_{1:t}}\ P(x_{1:t} \mid \mathbf{e}_{1:t})$.

This is different from smoothing for each past state and taking the sequence of most probable states!

We can see each sequence as a **path through a graph** whose nodes are possible states at each time step:



Because of the Markov property the most likely path to a given state consists of the most likely path to some previous state followed by a transition to that state.

This can be described using a **recursive formula**.

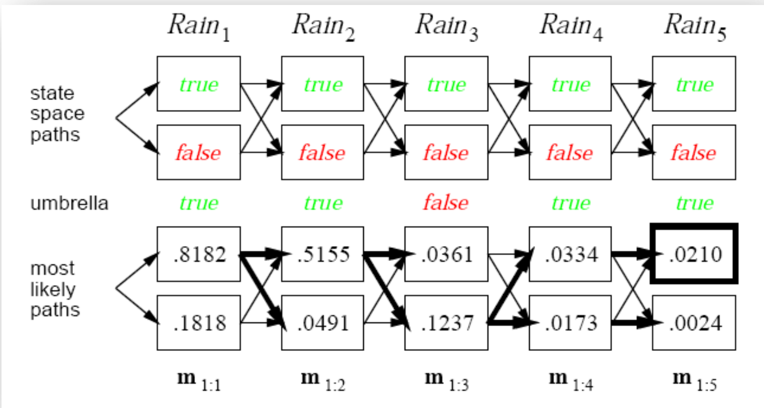
Viterbi algorithm

The most likely path to a given state consists of the most likely path to some previous state followed by a transition to that state.

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

Again, we use an approach of forward message passing:

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



Hidden Markov models

Assume that the state of process is described by a single discrete random variable X_t (there is also a single evidence variable E_t).
This is called a **hidden Markov model** (HMM).

This restricted model allows for a simple and elegant **matrix implementation** of all the basic algorithms.
Assume that variable X_t takes values from the set {1,…S}, where S is the number of possible states.
The **transition model P**$(X_t \mid X_{t-1})$ becomes an S×S matrix **T**, where:

$$\mathbf{T}_{(i,j)} = P(X_t = j \mid X_{t-1}=i)$$

We also put the **sensor model** in matrix form. Now we know the value of the evidence variable e_t so we describe $P(E_t = e_t \mid X_t=i)$, using a diagonal matrix $\mathbf{O}_t$, where:

$$\mathbf{O}_{t\,(i,i)} = P(E_t = e_t \mid X_t=i)$$

Matrix formulation of algorithms

The forward message propagation (from filtering)

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

$\mathbf{f}_{1:t+1} = \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\, \Sigma_{\mathbf{x}_t}\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)\, P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$

can be reformulated using matrix operations (message $\mathbf{f}_{1:t}$ is modelled as a one-column matrix) as follows:

$\mathbf{T}_{(i,j)} = P(X_t = j \mid X_{t-1}=i)$

$\mathbf{O}_{t\,(i,i)} = P(E_t = e_t \mid X_t=i)$

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

The backward message propagation (from smoothing)

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

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

can be reformulated using matrix operations (message $\mathbf{b}_{k:t}$ is modelled as a one-column matrix) as follows:

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

Full smoothing

What if we need to **smooth the whole sequence of states**?

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

The time complexity of smoothing with respect to evidence $\mathbf{e}_{1:t}$ is $O(t)$

One obvious method to smooth the whole sequence is to run the smoothing algorithm for each time step – this results in time complexity $O(t^2)$.

A better approach uses **dynamic programming** (reuse already computed information) reducing the time complexity to $O(t)$.

- **forward-backward algorithm**
- the practical drawback of this approach is that its space complexity can be too high – it is $O(|\mathbf{f}|t)$.

Forward-backward algorithm

```
function FORWARD-BACKWARD(ev, prior) returns a vector of probability distributions
  inputs: ev, a vector of evidence values for steps 1,...,t
          prior, the prior distribution on the initial state, P(X_0)
  local variables: fv, a vector of forward messages for steps 0,...,t
                   b, a representation of the backward message, initially all 1s
                   sv, a vector of smoothed estimates for steps 1,...,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
```

Full smoothing efficiently

Can be smoothing the whole sequence of states done with smaller memory consumption while keeping the time complexity O(t)?

Ideas:

- For message-passing in one direction we need constant space independent of t.
- Can the message $\mathbf{f}_{1:t}$ be obtained from the message $\mathbf{f}_{1:t+1}$?
- Then we can pass the forward message in the reverse (backward) direction together with the backward message.

Let us exploit **matrix operations:**

$$\mathbf{f}_{1:t+1} = \alpha\, \mathbf{O}_{t+1}\, \mathbf{T}^{T}\, \mathbf{f}_{1:t} \quad \rightarrow \quad \mathbf{f}_{1:t} = \alpha'(\mathbf{T}^{T})^{-1}\, (\mathbf{O}_{t+1})^{-1}\, \mathbf{f}_{1:t+1}$$

Algorithm:

- first, run the forward-message propagation to get $\mathbf{f}_{1:t}$
- then during the backward stage compute both $\mathbf{f}_{1:k}$ and $\mathbf{b}_{k+1:t}$

Smoothing with a fixed time lag

Assume smoothing in an on-line setting where smoothed estimates must be computed for a fixed number d of back time steps – $\mathbf{P}(\mathbf{X}_{t-d}|\mathbf{e}_{1:t})$. This is called **fixed-lag smoothing**.

In the ideal case, we want incremental computation in a constant time per update.

we have $\mathbf{P}(\mathbf{X}_{t-d}|\mathbf{e}_{1:t}) = \alpha\, \mathbf{f}_{1:t-d} \times \mathbf{b}_{t-d+1:t}$

we need $\mathbf{P}(\mathbf{X}_{t-d+1}|\mathbf{e}_{1:t+1}) = \alpha\, \mathbf{f}_{1:t-d+1} \times \mathbf{b}_{t-d+2:t+1}$

An incremental approach:

- we can use $\mathbf{f}_{1:t-d+1} = \alpha\, \mathbf{O}_{t-d+2}\, \mathbf{T}^{T}\, \mathbf{f}_{1:t-d}$
- we need incremental computation of $\mathbf{b}_{t-d+2:t+1}$ from $\mathbf{b}_{t-d+1:t}$

$\mathbf{b}_{t-d+1:t} = \mathbf{T}\, \mathbf{O}_{t-d+1}\, \mathbf{b}_{t-d+2:t} = (\Pi_{i=t-d+1,\ldots,t}\, \mathbf{T}\, \mathbf{O}_i)\, \mathbf{b}_{t+1:t} = \mathbf{B}_{t-d+1:t}\, \mathbf{1}$

$\mathbf{b}_{t-d+2:t+1} = (\Pi_{i=t-d+2,\ldots,t+1}\, \mathbf{T}\, \mathbf{O}_i)\, \mathbf{b}_{t+2:t+1} = \mathbf{B}_{t-d+2:t+1}\, \mathbf{1}$

$$\mathbf{B}_{t-d+2:t+1} = (\mathbf{O}_{t-d+1})^{-1}\, \mathbf{T}^{-1}\, \mathbf{B}_{t-d+1:t}\, \mathbf{T}\, \mathbf{O}_{t+1}$$

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 x S transition matrix T
    d, the length of the lag for smoothing
  static: t, the current time, initially 1
    f, a probability distribution, the forward message $\mathbf{P}(X_t|e_{1:t})$, initially PRIOR[*hmm*]
    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**
    $\mathbf{f} \leftarrow$ FORWARD($\mathbf{f}, e_t$)
    remove e_{t-d-1} from the beginning of $e_{t-d:t}$
    $\mathbf{O}_{t-d} \leftarrow$ diagonal matrix containing $\mathbf{P}(e_{t-d}|X_{t-d})$
    B $\leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{BTO}_t$
  else B $\leftarrow \mathbf{BTO}_t$
  $t \leftarrow t+1$
  if $t >$ **d then return** NORMALIZE(**f** x **B1**) **else return** null

Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz