Probabilistic reasoning over time -

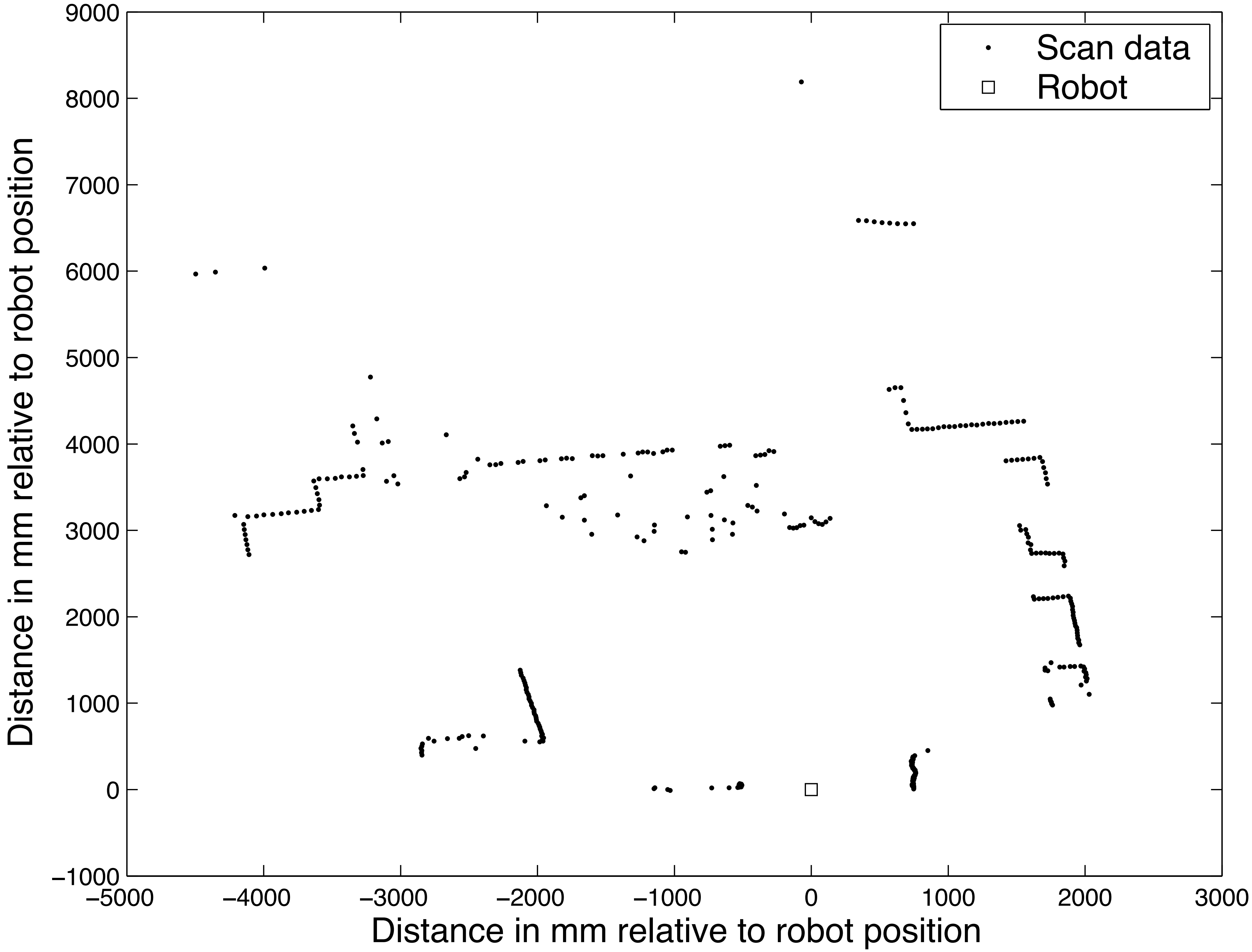
Hidden Markov Models

(recap BNs)

Applied artificial intelligence (EDAF70)
Lecture 09
2018-02-14
Elin A. Topp

Material based on course book, chapter 15

A robot's view of the world...



Prior probability

Prior or *unconditional probabilities* of propositions

e.g., $P(\mathit{Person} = \mathit{true}) = 0.2$ and

$P(\mathit{Weather} = \mathit{sunny}) = 0.72$ (e.g., known from statistics)

correspond to belief *prior to the arrival of any (new) evidence*

Probability distribution gives values for all possible assignments (normalised):

$\mathbb{P}(\mathit{Weather}) = \langle 0.72, 0.1, 0.08, 0.1 \rangle$

Joint probability distribution for a set of (independent) random variables gives the probability of every atomic event on those random variables (i.e., every sample point):

$\mathbb{P}(\mathit{Weather}, \mathit{Person})$ = a 4×2 matrix of values:

Weather / Person	sunny	rain	cloudy	snow
true	0,144	0,02	0,016	0,02
false	0,576	0,08	0,064	0,08

Inference

Probabilistic inference:

Computation of posterior probabilities given observed evidence

starting out with the full joint distribution as "knowledge base":

Inference by enumeration

	leg-size		¬ leg-size	
	curved	¬ curved	curved	¬ curved
person	0,108	0,012	0,072	0,008
¬ person	0,016	0,064	0,144	0,576

For any proposition Φ, sum the atomic events where it is true:

$$P(\Phi) = \sum_{\omega:\omega \models \Phi} P(\omega)$$

P(person ∨ leg-size) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28

P(leg-size) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2

Can also compute posterior probabilities:

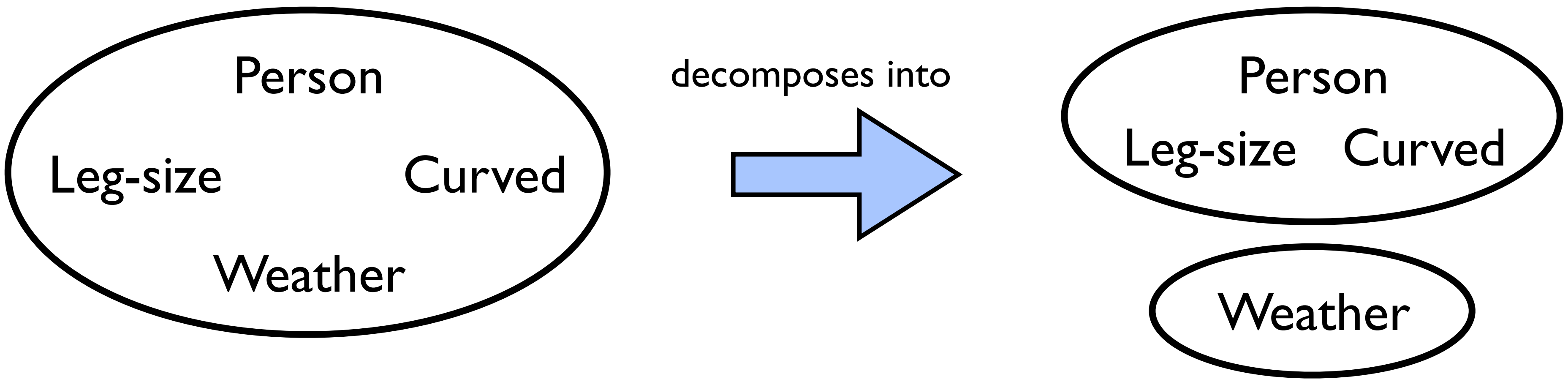
$$P(\neg person \mid leg\text{-}size) = \frac{P(\neg person \wedge leg\text{-}size)}{P(leg\text{-}size)}$$

$$= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4$$

Independence

A and *B* are *independent* iff

$$\mathbf{P}(A \mid B) = \mathbf{P}(A) \quad \text{or} \quad \mathbf{P}(B \mid A) = \mathbf{P}(B) \quad \text{or} \quad \mathbf{P}(A, B) = \mathbf{P}(A)\,\mathbf{P}(B)$$



$$\mathbb{P}(\textit{Leg-size}, \textit{Curved}, \textit{Person}, \textit{Weather}) = \mathbb{P}(\textit{Leg-size}, \textit{Curved}, \textit{Person})\,\mathbb{P}(\textit{Weather})$$

32 entries reduced to 8 + 4 (Weather is not Boolean!).
This absolute (*unconditional*) independence is powerful but rare!

Some fields (like robotics and computer vision, or, as used in the book, dentistry) have still a lot, maybe hundreds, of variables, none of them being independent.

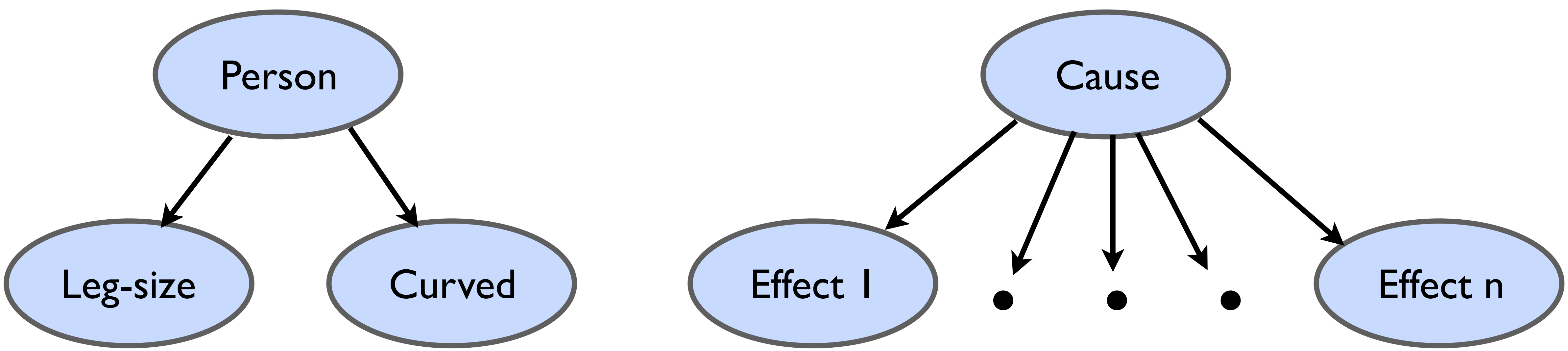
What can be done to overcome this mess...?

Bayes' Rule and conditional independence

$\mathbb{P}(\,Person \mid leg\text{-}size \wedge curved)$
$= \alpha\;\; \mathbb{P}(\,leg\text{-}size \wedge curved \mid Person)\; \mathbb{P}(\,Person)$
$= \alpha\;\; \mathbb{P}(\,leg\text{-}size \mid Person)\; \mathbb{P}(\,curved \mid Person)\; \mathbb{P}(\,Person)$

An example of a *naive Bayes* model:

$$\mathbb{P}(\,Cause, Effect_1, \ldots, Effect_n) = \;\; \mathbb{P}(\,Cause) \prod_i \mathbb{P}(\,Effect_i \mid Cause)$$



The total number of parameters is *linear* in *n*

Bayesian networks

A simple, graphical notation for *conditional independence assertions* and hence for compact specification of full joint distributions

Syntax:

a set of nodes, one per random variable

a directed, acyclic graph (link ≈ "directly influences")

a conditional distribution for each node given its parents:

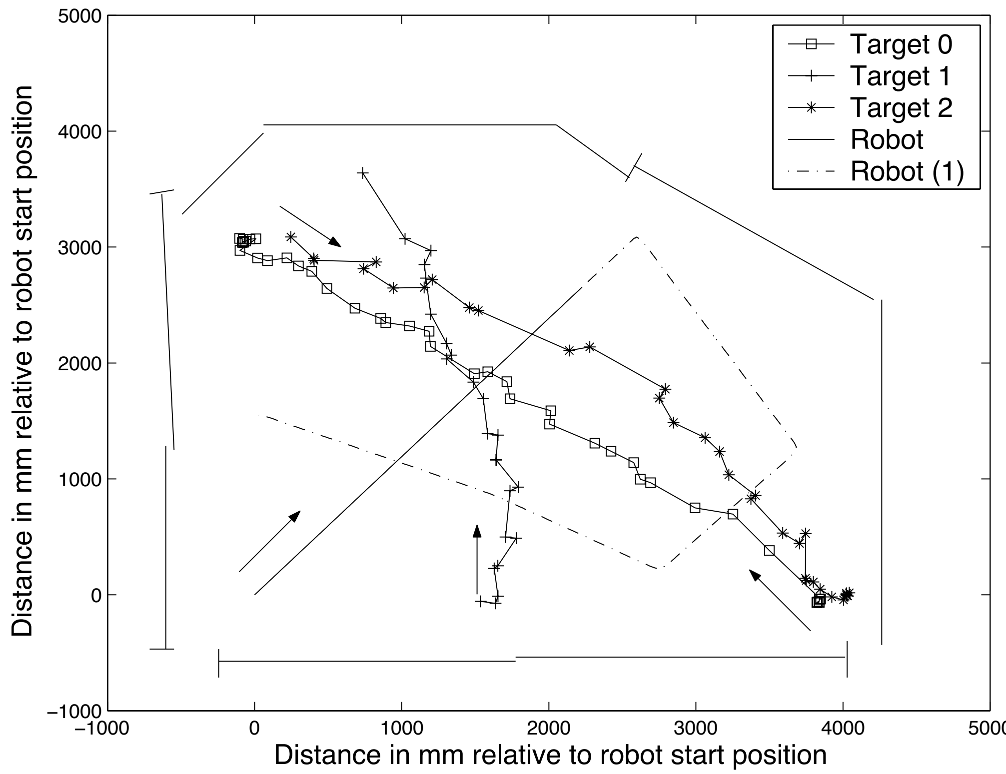
$\mathbf{P}(X_i \mid Parents(X_i))$

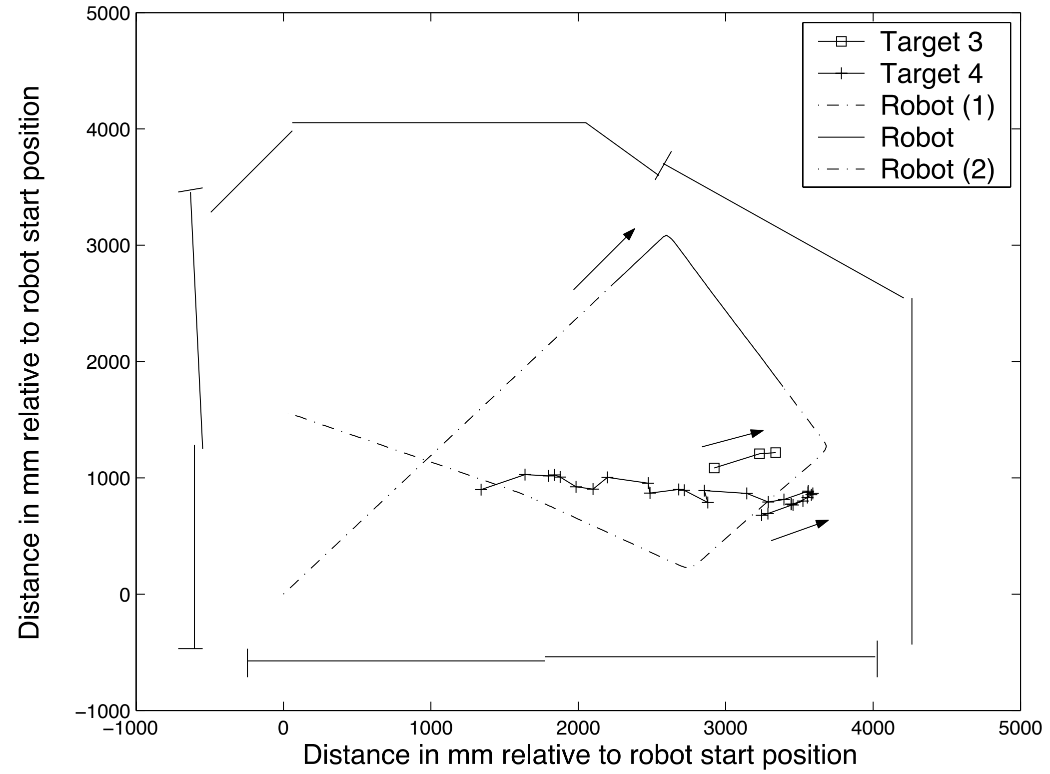
In the simplest case, conditional distribution represented as a

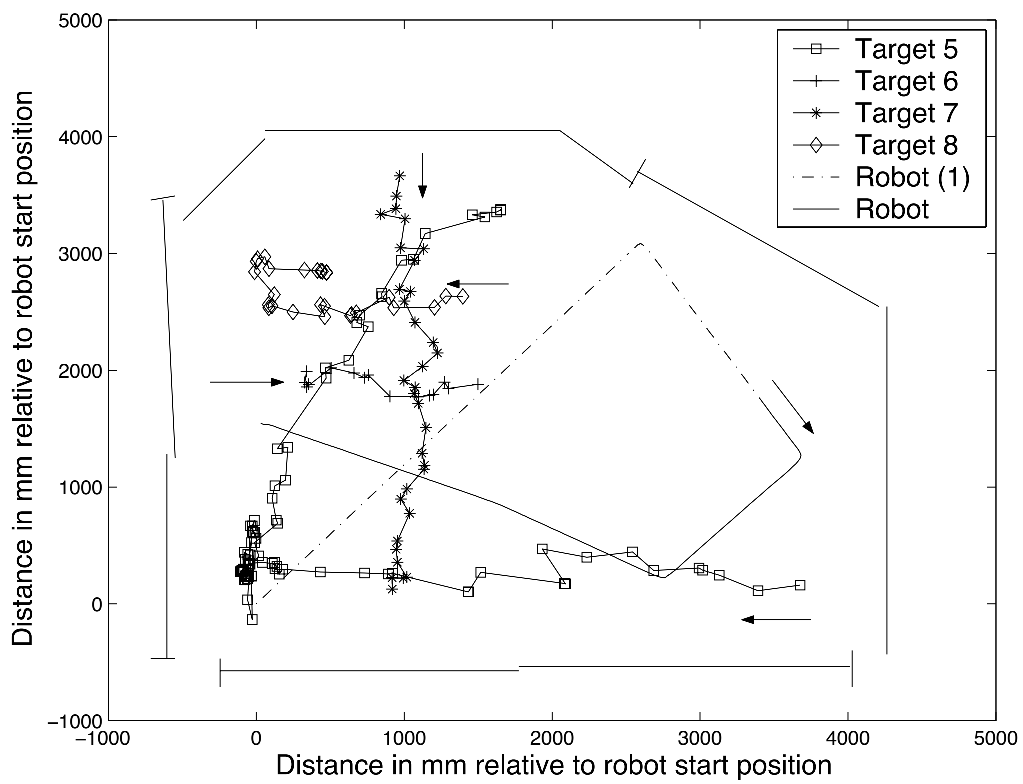
conditional probability table (CPT)

giving the distribution over X_i for each combination of parent values

Tracking and associating... while moving ...







Probabilistic reasoning over time

... means to keep track of the current state of

- a process (temperature controller, other controllers)
- an agent with respect to the world (localisation of a robot in some "world")

in order to make predictions or to simply understand what might have caused this current state.

This involves both a **transition model** (how the state is assumed to change) and a **sensor model** (how observations / percepts are related to the world state).

Previously:

the focus was on what was possible to happen (e.g., search), now it is on what is likely / unlikely to happen

the focus was on static worlds (Bayesian networks), now we look at dynamic processes where everything, both state AND observations, depend on time.

Three classes of approaches

Hidden Markov models

Probabilistic filters (Kalman or Particle filters, Gaussian Mixture Models)

Dynamic Bayesian networks (cover actually the other two as special cases)

But first, some basics ...

Reasoning over time

With

$\boldsymbol{X}_t$ the current state description at time *t*

$\boldsymbol{E}_t$ the evidence obtained at time *t*

we can describe a *state transition model* and a *sensor model* that we can use to model a time step sequence - a chain of states and sensor readings according to discrete time steps - so that we can understand the ongoing process.

We assume to start out in $\boldsymbol{X}_0$, but evidence will only arrive after the first state transition is made: $\boldsymbol{E}_1$ is then the first piece of evidence to be plugged into the chain.

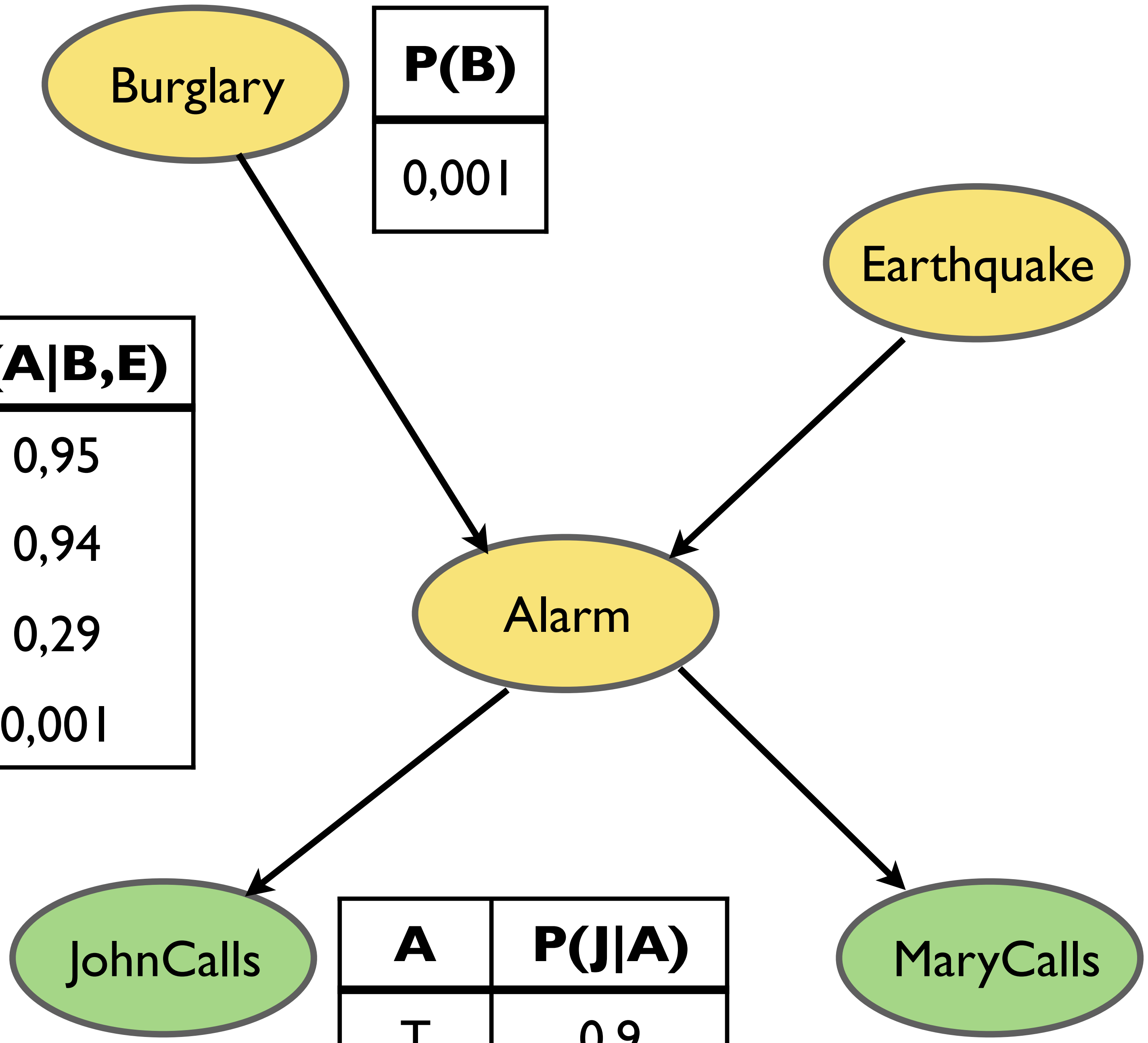
The "general" transition model would then specify

$$\mathbb{P}(\boldsymbol{X}_t \mid \boldsymbol{X}_{0:t-1})$$

... this would mean we need full joint distributions over all time steps... or not?

X

Observable and "hidden" variables



P(B)
0,001

P(E)
0,002

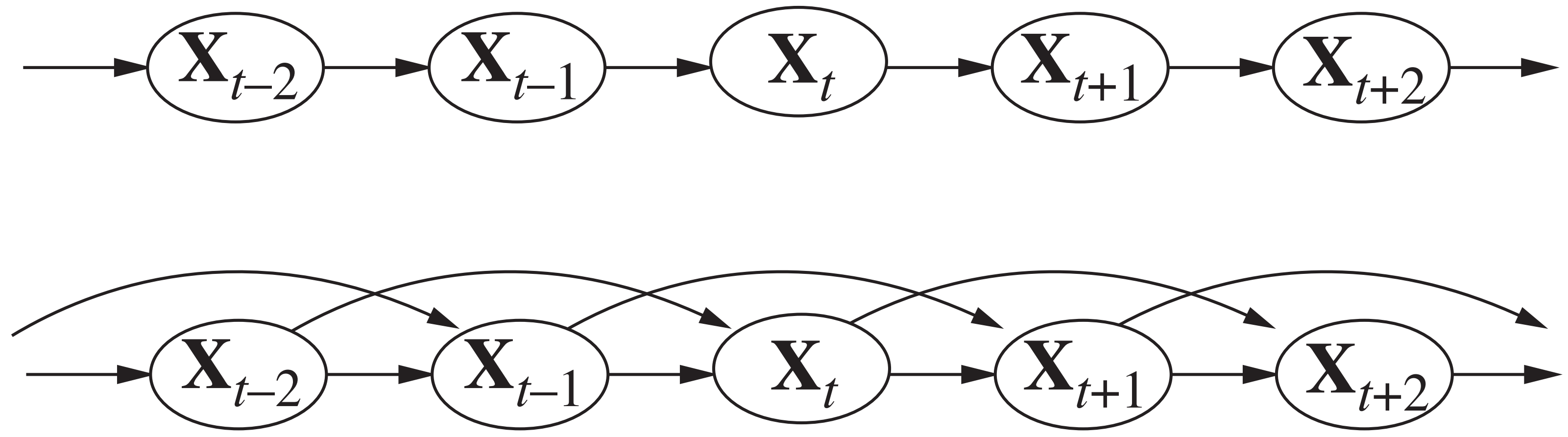
B	**E**	**P(A|B,E)**
T	T	0,95
T	F	0,94
F	T	0,29
F	F	0,001

A	**P(J|A)**
T	0,9
F	0,05

A	**P(M|A)**
T	0,7
F	0,01

The Markov assumption

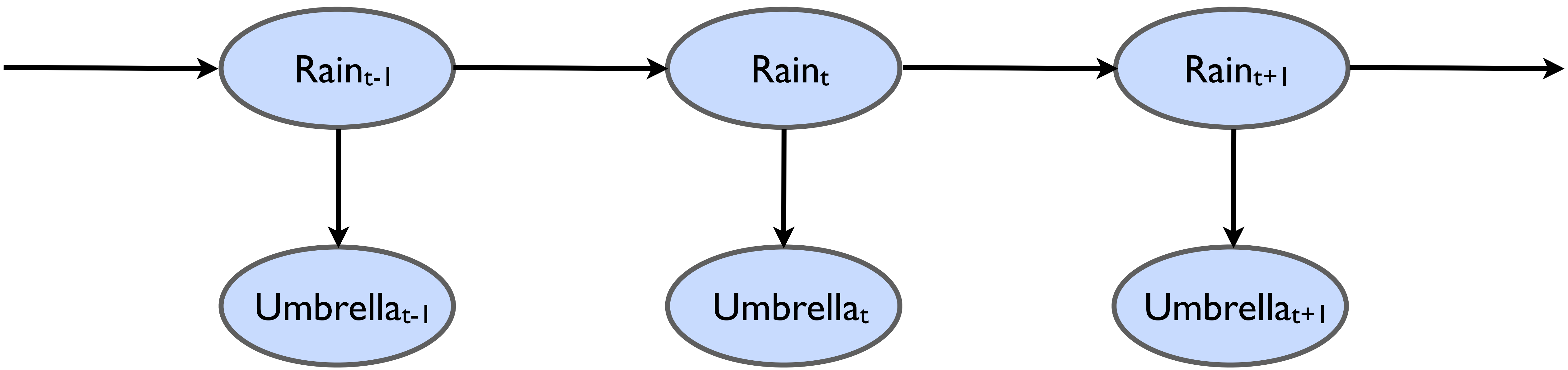
A process is *Markov* (i.e., complies with the Markov assumption), when any given state $\boldsymbol{X}_t$ depends only on a *finite and fixed number of previous states.*



A first-order Markov chain as Bayesian network

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

"cause" / state



"effect" / evidence

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

Inference for any t

With

$\mathbb{P}(\boldsymbol{X}_0)$ the prior probability distribution in $t=0$ (i.e., the *initial state model*),

$\mathbb{P}(\boldsymbol{X}_i \mid \boldsymbol{X}_{i-1})$ the state transition model and

$\mathbb{P}(\boldsymbol{E}_i \mid \boldsymbol{X}_i)$ the sensor model

we have the complete joint distribution for all variables for any t.

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

An issue with the Markov assumption

First-order Markov chain:

State variables (at t) contain ALL information needed for t+1.

Sometimes, that is too strong an assumption (or too weak in some sense).

Hence, increase either the order (second-order Markov chain)

or

add information into the state variable(s) (***R*** could include also *Season*, *Humidity*, *Pressure*, *Location*, instead of only "*Rain*")

Note: It is possible to express an increase in order by increasing the number of state variables, keeping the order fixed - for the umbrella world you could use

R = *<RainYesterday, RainToday>*

When things get too complex, rather add another sensor (e.g., observe coats).

X

Inference in temporal models - what can we use all this for?

- **Filtering**: Finding the **belief state**, or doing **state estimation**, i.e., computing the posterior distribution over the *most recent state*, using evidence up to this point:
  $\mathbb{P}(\boldsymbol{X}_t \mid \mathbf{e}_{1:t})$
- **Predicting**: Computing the posterior over a *future* state, using evidence up to this point: $\mathbb{P}(\boldsymbol{X}_{t+k} \mid \mathbf{e}_{1:t})$ for some $k>0$ (can be used to evaluate course of action based on predicted outcome)
- **Smoothing**: Computing the posterior over a past state, i.e., understand the past, given information up to this point: $\mathbb{P}(\boldsymbol{X}_k \mid \mathbf{e}_{1:t})$ for some k with $0 \leq k < t$
- **Explaining**: Find the best explanation for a series of observations, i.e., computing $argmax_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$ - can be efficiently handled by **Viterbi** algorithm
- **Learning**: If sensor and / or transition model are not known, they can be learned from observations (by-product of inference in Bayesian network - both static or dynamic). Inference gives estimates, estimates are used to update the model, updated models provide new estimates (by inference). Iterate until converging - again, this is an instance of the EM-algorithm.

Filtering: Prediction & update (FORWARD-step)

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

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

$= \alpha\ \mathbb{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \mathbb{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ (Bayes' Rule)

$= \alpha\ \mathbb{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\ \mathbb{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ (1. update under Markov assumption (sensor model), 2. one-step prediction)

$= \alpha\ \mathbb{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbb{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t})\ P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$ (sum over atomic events for $\mathbf{X}$)

$= \alpha\ \mathbb{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbb{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)\ P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$ (Markov assumption)

$\mathbb{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ ("forward message", propagated recursively

$\mathbf{f}_{1:t+1} = \alpha\ FORWARD(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$ through "forward step function")

$\mathbf{f}_{1:0} = \mathbb{P}(\mathbf{X}_0)$

Prediction - filtering without the update

$$\mathbb{P}(\boldsymbol{X}_{t+k+1} \mid \mathbf{e}_{1:t}) = \sum_{\boldsymbol{x}_{t+k}} \mathbb{P}(\boldsymbol{X}_{t+k+1} \mid \boldsymbol{x}_t)\, P(\boldsymbol{x}_{t+k} \mid \mathbf{e}_{1:t})$$ (k-step prediction)

For large k the prediction gets quite blurry and will eventually converge into a *stationary distribution* at the *mixing point*, i.e., the point in time when this convergence is reached - in some sense this is when "everything is possible".

Smoothing: “explaining” backward

$\mathbb{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) = fb(\mathbf{X}_k, \mathbf{e}_{1:k}, \mathbb{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k))$ *with* $0 \leq k < t$ (understand the past from the recent past)

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

$= \alpha\ \mathbb{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\ \mathbb{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k})$ (Bayes’ Rule)

$= \alpha\ \mathbb{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\ \mathbb{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$ (Markov assumption)

$= \alpha\ \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}$ (forward-message × backward-message)

Smoothing: calculating backward message

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

$= \sum_{\mathbf{x}_{k+1}} \mathbb{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1})\, \mathbb{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)$ (conditioning on $\mathbf{X}_{k+1}$, i.e., looking "backward")

$= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1})\, \mathbb{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)$ (cond. indep. - Markov assumption)

$= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\, \mathbb{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)$ (decompose)

$= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})\, P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\, \mathbb{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)$ (1. sensor, 2. backward msg, 3. transition model)

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

$\mathbb{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$ ("backward message", propagated recursively)

$\mathbf{b}_{k+1:t} = BACKWARD(\mathbf{b}_{k+2:t}, \mathbf{e}_{k+1})$ (through "backward step function")

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

Smoothing "in a nutshell": Forward-Backward-algorithm

$\mathbb{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) = fb(\mathbf{e}_{1:k}, \mathbb{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k))$ *with* $0 \leq k < t$ — understand the past from the recent past

$= \alpha \; \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}$ — by first filtering (forward) until step k, then explaining backward from t to $k+1$

Obviously, it is a good idea to store the filtering (forward) results for later smoothing

Drawback of the algorithm: not really suitable for online use (t is growing, ...)

Consequently, try with fixed-lag-smoothing (keeping a fixed-length window, BUT: "simple" Forward-Backward does not really do it efficiently - here we need HMMs)

“HMM”
Hidden Markov models

A specific class of models (sensor and transition) to be plugged into the previously discussed algorithms - which makes the algorithms more specific as well!

Main idea:

The state is represented by a *single discrete random variable*, taking on values that represent the (all) possible states of the world.

Complex states, e.g., the location and the heading of a robot in a grid world can be merged into one variable; the possible values are then all possible tuples of the values for each original “single” variable.

"HMM"
State transition and sensor model

We get the following notation:

X_t the state at time t, taking on values 1 ... S, with S the number of possible states / values.

E_t the observation at time t

The ***transition*** model $P(X_t \mid X_{t-1})$ is then expressed as S x S matrix $\boldsymbol{T}$:

$$\boldsymbol{T}_{ij} = P(X_t = j \mid X_{t-1} = i) \text{ in time step } t$$

The ***sensor*** model for the corresponding observations depending on the current state, i.e., $P(e_t \mid X_t = i)$ is then expressed as S x S diagonal matrix $\mathbf{O}$ in time step t with

$$\mathbf{O}_{\mathbf{e\_t}ij} = P(e_t \mid X_t = i) \quad \text{for } i = j \qquad \text{and}$$

$$\mathbf{O}_{\mathbf{e\_t}ij} = 0 \qquad \text{for } i \neq j$$

Forward-backward equations as matrix-vector operations

Forward-equation (recap)

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

becomes $\boldsymbol{f}_{1:t+1} = \alpha\;\; \boldsymbol{O}_{t+1}\, \boldsymbol{T}^T \boldsymbol{f}_{1:t}$

Backward-equation (recap)

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

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

Forward-Backward-equation is then still $\alpha\;\; \boldsymbol{f}_{1:k}\;\times\; \boldsymbol{b}_{k+1:t}$

Smoothing in constant space

Idea

propagate both $\boldsymbol{f}$ and $\boldsymbol{b}$ *in the same direction*, hence avoiding to store the $\boldsymbol{f}_{1:k}$ for a shifting / growing time slice $k{:}t$

Propagate the forward-message $\boldsymbol{f}$ "backward" with

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

Start with computing $\boldsymbol{f}_{t:t}$ in a standard forward-run, forgetting all the intermediate messages, then compute both $\boldsymbol{f}$ and $\boldsymbol{b}$ simultaneously "backward" to do smoothing for each step this should be done for (NOTE: works obviously only if $\boldsymbol{T}^T$ and O can be inverted, i.e., every sensor reading must be possible in every state, though it can be very unlikely)

Fixed-lag smoothing (online)

Idea

if we can do smoothing with constant space requirements, we can also find an efficient recursive algorithm for online smoothing (a shifting "window"), independent of the length d of the investigated time slice $t\text{-}d$ (with t growing).

We need to compute

$\alpha\ \boldsymbol{f}_{1:t-d} \times \boldsymbol{b}_{t-d+1:t}$ for time slice $t\text{-}d$. In $t+1$, when a new observation arrives, we need

$\alpha\ \boldsymbol{f}_{1:t-d+1} \times \boldsymbol{b}_{t-d+1:t+1}$ for time slice $t\text{-}d+1$.

We can get $\boldsymbol{f}_{1:t-d+1}$ from $\boldsymbol{f}_{1:t-d}$, applying standard filtering.

For the backward message, some more inspection has to be done ($\boldsymbol{b}_{t-d+1:t+1}$ depends on the new evidence in $t+1$) but there is a way by looking at how $\boldsymbol{b}_{t-d+1:t}$ relates to $\boldsymbol{b}_{t+1:t}$

X

Fixed-lag smoothing (online)

Backward recursion:

apply the recursive equation for $\boldsymbol{b}_{t-d+1:t}$ $\boldsymbol{d}$ times:

$$\boldsymbol{b}_{t-d+1:t} = \left(\prod_{i=t-d+1}^{t} \boldsymbol{TO}_i\right)\boldsymbol{b}_{t+1:t} = \boldsymbol{B}_{t-d+1:t}\ \boldsymbol{1}$$

Then, after the next observation, this will be:

$$\boldsymbol{b}_{t-d+2:t+1} = \left(\prod_{i=t-d+2}^{t+1} \boldsymbol{TO}_i\right)\boldsymbol{b}_{t+2:t+1} = \boldsymbol{B}_{t-d+2:t+1}\ \boldsymbol{1}$$

Do some matrix "division" and get an incremental update for $\boldsymbol{B}$ (and ultimately $\boldsymbol{b}_{t-d+2:t+1}$):

$$\boldsymbol{B}_{t-d+2:t+1} = \boldsymbol{O}^{-1}_{t-d+1}\ \boldsymbol{T}^{-1}\boldsymbol{B}_{t-d+1:t}\ \boldsymbol{TO}_{t+1}$$

The full algorithm for fixed-lag smoothing

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**
  $\mathbf{f} \leftarrow$ 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 $\mathbf{P}(e_{t-d}|X_{t-d})$
  $\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{B}\mathbf{T}\mathbf{O}_t$
else $\mathbf{B} \leftarrow \mathbf{B}\mathbf{T}\mathbf{O}_t$
$t \leftarrow t+1$
if $t > d$ **then return** NORMALIZE($\mathbf{f} \times \mathbf{B1}$) **else return** null

Summary

Inference in temporal models

- Filtering and prediction (FORWARD)

- Smoothing (FORWARD-BACKWARD)

Hidden Markov Models

- Simplified matrix representation for Forward-backward calculations