Probabilistic reasoning over time -

Hidden Markov Models

Artificial intelligence (EDAP01)
Lecture 09
2020-02-17
Elin A. Topp

Material based on course book, chapter 15

Recap / Continuation

- Uncertainty & probability (chapter 13)
  - Uncertainty
  - Probability
  - Syntax and Semantics
  - Inference
  - Independence and Bayes' Rule
- Bayesian Networks (chapter 14.1-3)
  - Syntax
  - Semantics
  - Efficient representation

Conditional independence

$\mathbb{P}(\textit{Leg-size, Person, Curved})$ has $2^3 - 1 = 7$ independent entries (must sum up to 1)

But: If there is a person, the probability for "Curved" does not depend on whether the pattern has leg-size (this dependency is now "implicit" in some sense):

(1) $\mathbb{P}(\textit{Curved} \mid \textit{leg-size, person}) = \mathbb{P}(\textit{Curved} \mid \textit{person})$

The same holds when there is no person:

(2) $\mathbb{P}(\textit{Curved} \mid \textit{leg-size}, \neg\textit{person}) = \mathbb{P}(\textit{Curved} \mid \neg\textit{person})$

Curved is *conditionally independent* of *Leg-size* given *Person*:

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

Writing out the full joint distribution using chain rule:

$$\begin{aligned}
&\mathbb{P}(\textit{Leg-size, Curved, Person}) \\
&= \mathbb{P}(\textit{Leg-size} \mid \textit{Curved, Person})\, \mathbb{P}(\textit{Curved, Person}) \\
&= \mathbb{P}(\textit{Leg-size} \mid \textit{Curved, Person})\, \mathbb{P}(\textit{Curved} \mid \textit{Person})\, \mathbb{P}(\textit{Person}) \\
&= \mathbb{P}(\textit{Leg-size} \mid \textit{Person})\, \mathbb{P}(\textit{Curved} \mid \textit{Person})\, \mathbb{P}(\textit{Person})
\end{aligned}$$

gives thus $2 + 2 + 1 = 5$ independent entries

Question: What does that mean in the world of our coloured and partially skewed dice?

Conditional independence (2)

In most cases, the use of conditional independence reduces the size of the representation of the joint distribution from exponential in *n* to linear in *n*.

Hence:

Conditional independence is our most basic and robust form of knowledge about uncertain environments

Summary

Probability is a way to formalise and represent uncertain knowledge

The *joint probability distribution* specifies probability over every *atomic event* (which is a combination of the relevant values of the random variables)

Queries can be answered by *summing* over atomic events (marginal probability)

Bayes' rule can be applied to compute posterior probabilities so that *diagnostic* probabilities can be assessed from *causal* ones

For *nontrivial* domains, we must find a way to *reduce* the size of joint distributions

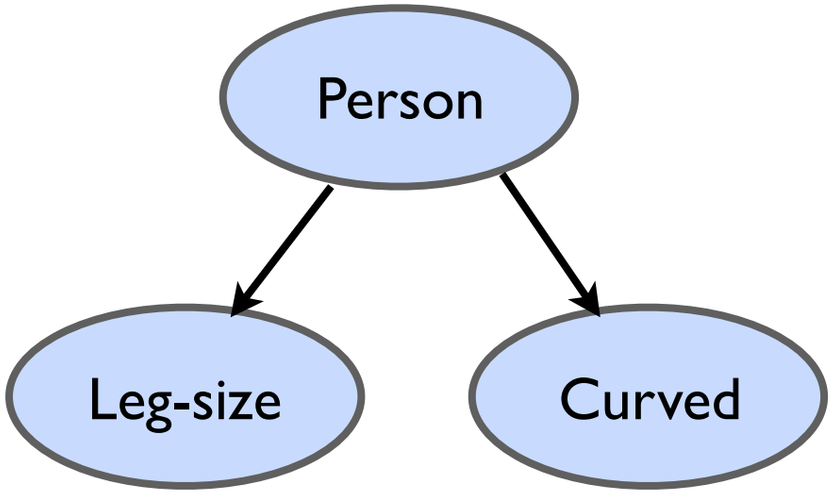
Independence and *conditional independence* provide the tools

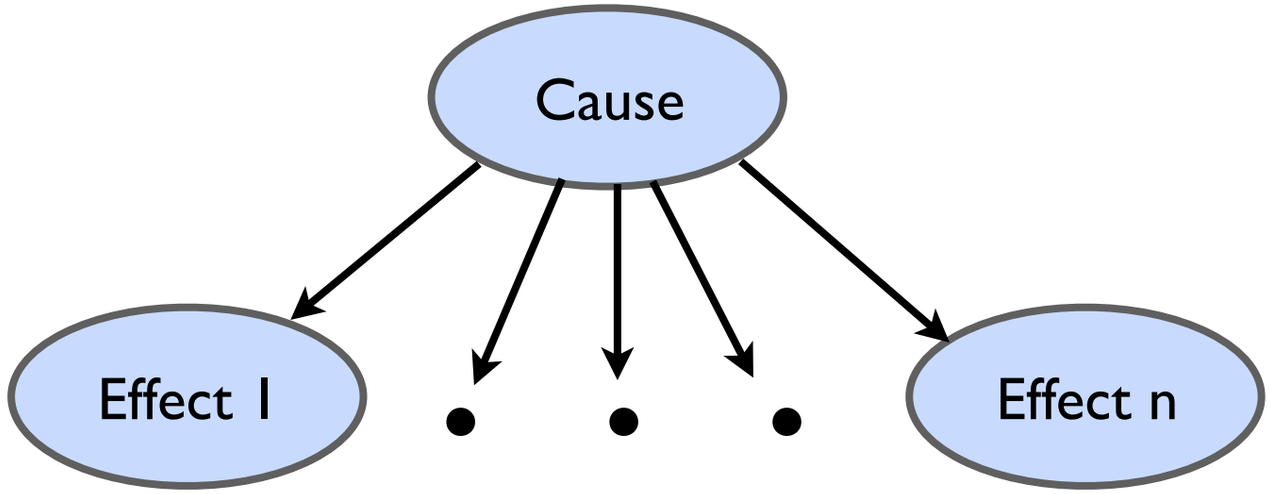
Bayes' and product rule & conditional independence

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

An example of a *naive Bayes* model:

$$\mathbb{P}(\mathit{Cause}, \mathit{Effect}_1, \ldots, \mathit{Effect}_n) = \mathbb{P}(\mathit{Cause}) \prod_i \mathbb{P}(\mathit{Effect}_i \mid \mathit{Cause})$$





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

Question: How does this "network" look for the dice example?

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

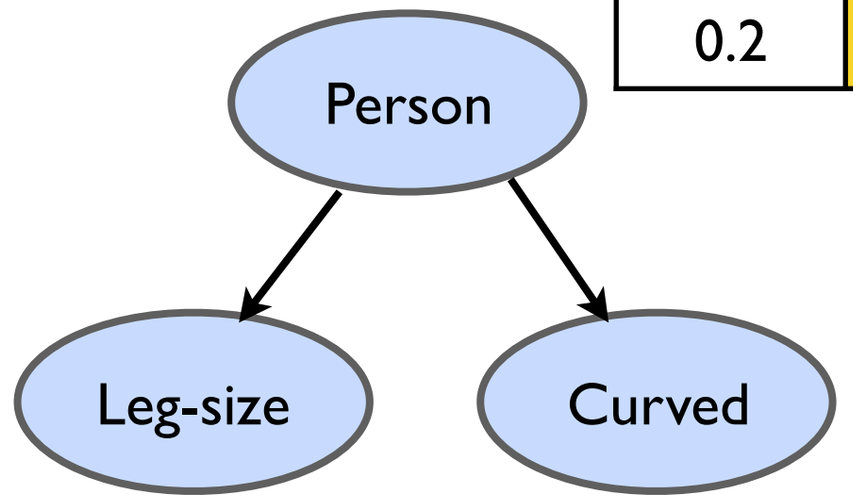
Example

Topology of network encodes conditional independence assertions:

P(W=sunny)	P(W=rainy)	P(W=cloudy)	P(W=snow)
0.72	0.1	0.08	0.1

Weather

P(Per)	P(¬Per)
0.2	0.8



Per	P(L|Per)	P(¬L|Per)
T	0.6	0.4
F	0.1	0.9

Per	P(C|Per)	P(¬C|Per)
T	0.9	0.1
F	0.2	0.8

Weather is (unconditionally, absolutely) independent of the other variables

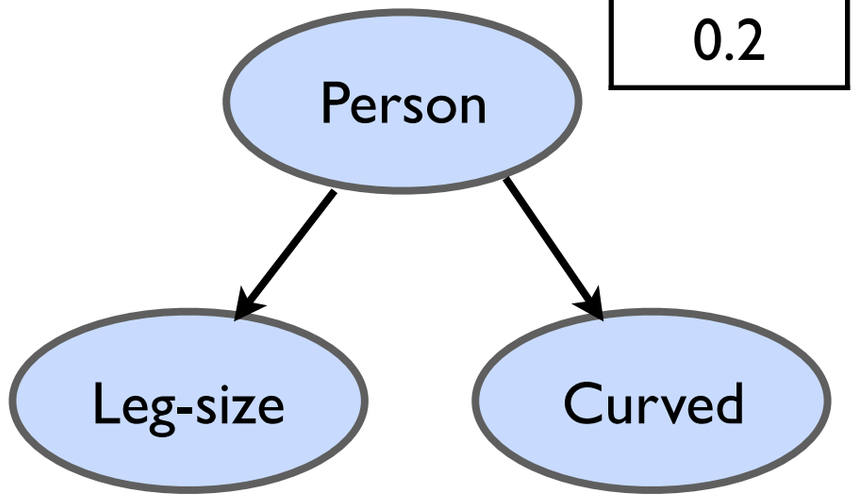
Leg-size and *Curved* are conditionally independent given *Person*

Example

Topology of network encodes conditional independence assertions:

P(W=sunny)	P(W=rainy)	P(W=cloudy)
0.72	0.1	0.08

P(Per)
0.2



Per	P(T|Per)
T	0.6
F	0.1

Per	P(C|Per)
T	0.9
F	0.2

Weather is (unconditionally, absolutely) independent of the other variables

Leg-size and *Curved* are conditionally independent given *Person*

We can skip the dependent columns in the tables to reduce complexity!

Example 2

I am at work, my neighbour John calls to say my alarm is ringing, but neighbour Mary does not call.

Sometimes the alarm is set off by minor earthquakes.

Is there a burglar?

Variables: *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

Network topology reflects "causal" knowledge:

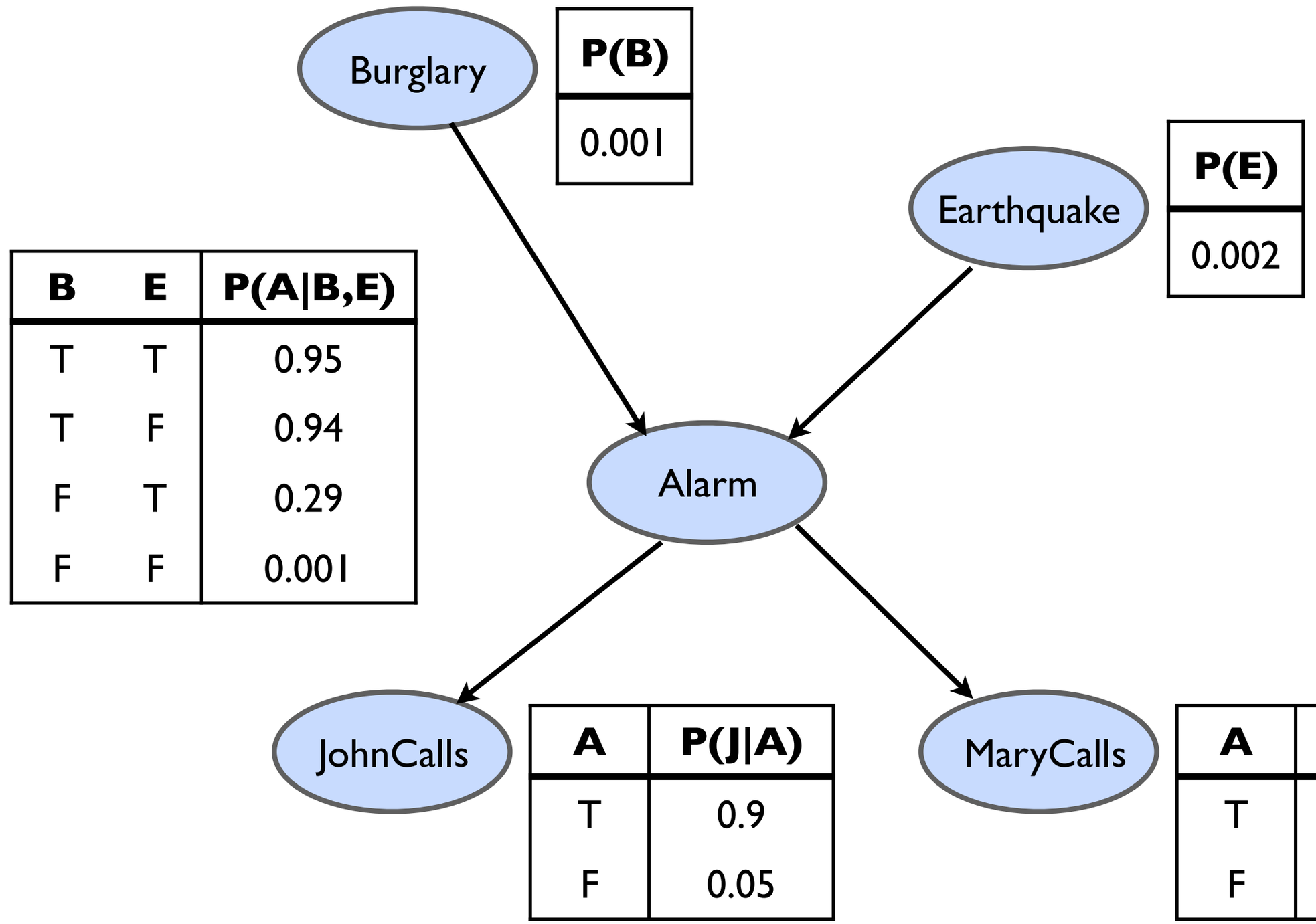
A burglar can set the alarm off

An earthquake can set the alarm off

The alarm can cause John to call

The alarm can cause Mary to call

Example 2 (2)



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

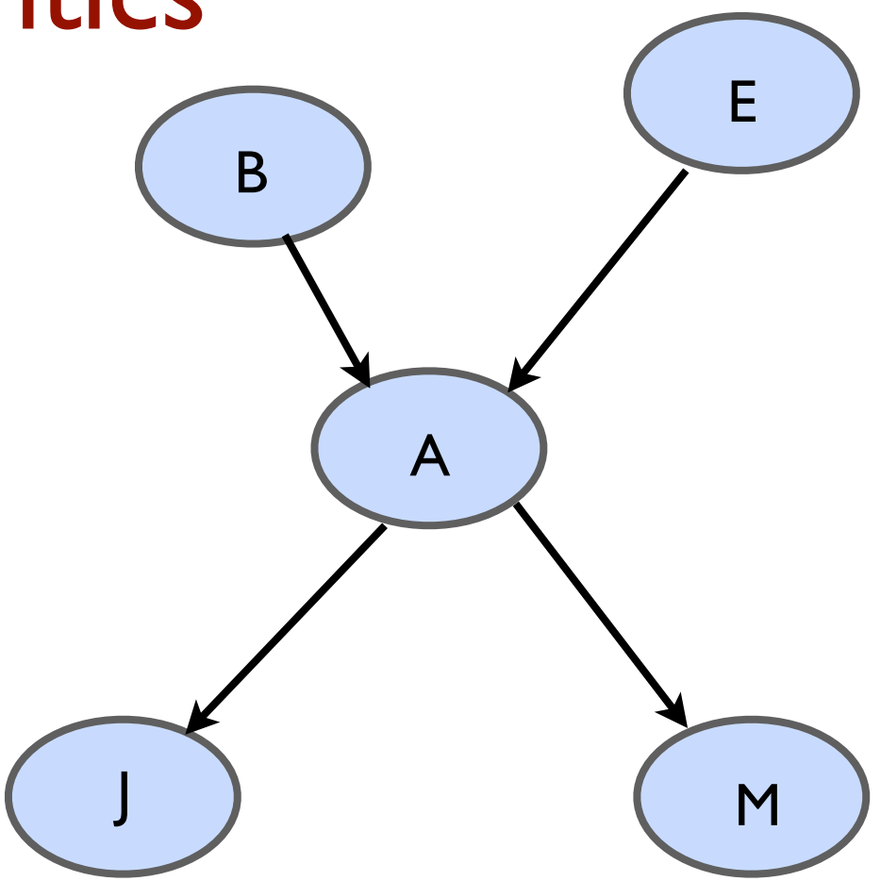
Global semantics

Global semantics defines the full joint distribution as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i))$$

E.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

=



Global semantics

Global semantics defines the full joint distribution as the product of the local conditional distributions:

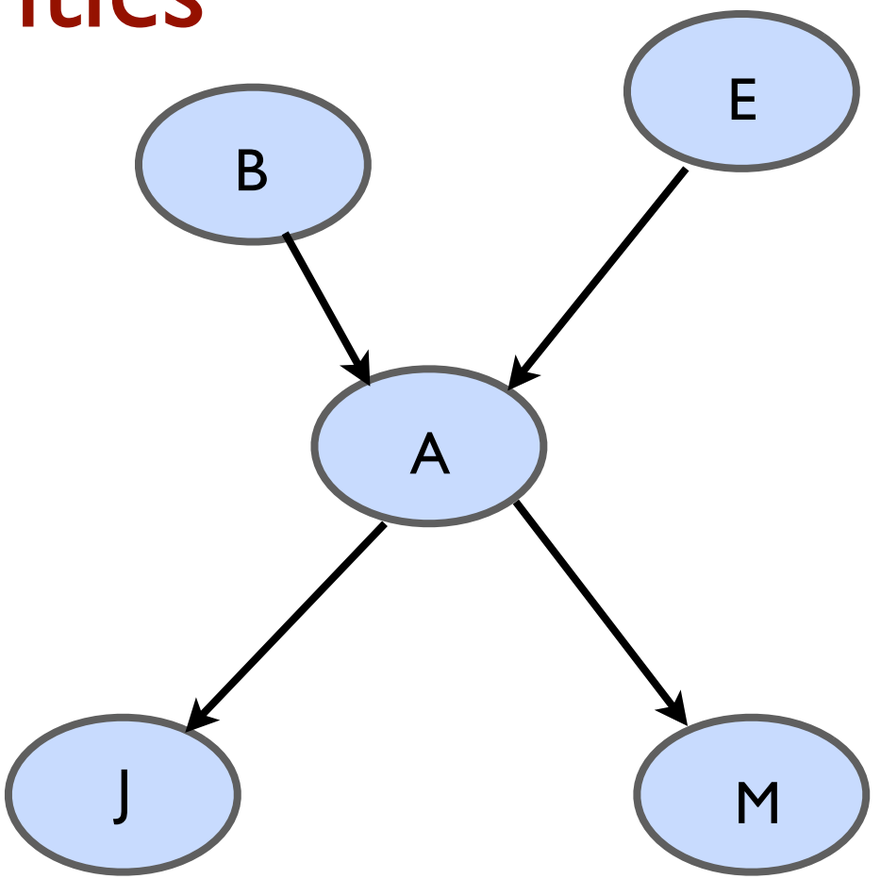
$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i))$$

E.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$$= P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$$

$$= 0.9 * 0.7 * 0.001 * 0.999 * 0.998$$

$$\approx 0.000628$$



Constructing Bayesian networks

We need a method such that a series of locally testable assertions of conditional independence guarantees the required global semantics.

1. Choose an ordering of variables $X_1,..., X_n$

2. For $i = 1$ to n

    add X_i to the network

    select parents from $X_1,..., X_{i-1}$ such that

$$\mathbb{P}(X_i \mid Parents(X_i)) = \mathbb{P}(X_i \mid X_1,..., X_{i-1})$$

This choice of parents guarantees the global semantics:

$$\mathbb{P}(X_1,..., X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid X_1,..., X_{i-1}) \qquad \text{(chain rule)}$$

$$= \prod_{i=1}^{n} \mathbb{P}(X_i \mid Parents(X_i)) \qquad \text{(by construction)}$$

Constructing Bayesian networks

We need a method such that a series of locally testable assertions of conditional independence guarantees the required global semantics.

1. Choose an ordering of variables $X_1,..., X_n$

2. For $i = 1$ to n

    add X_i to the network

    select parents from $X_1,..., X_{i-1}$ such that

    $$\mathbb{P}(X_i \mid Parents(X_i)) = \mathbb{P}(X_i \mid X_1,..., X_{i-1})$$

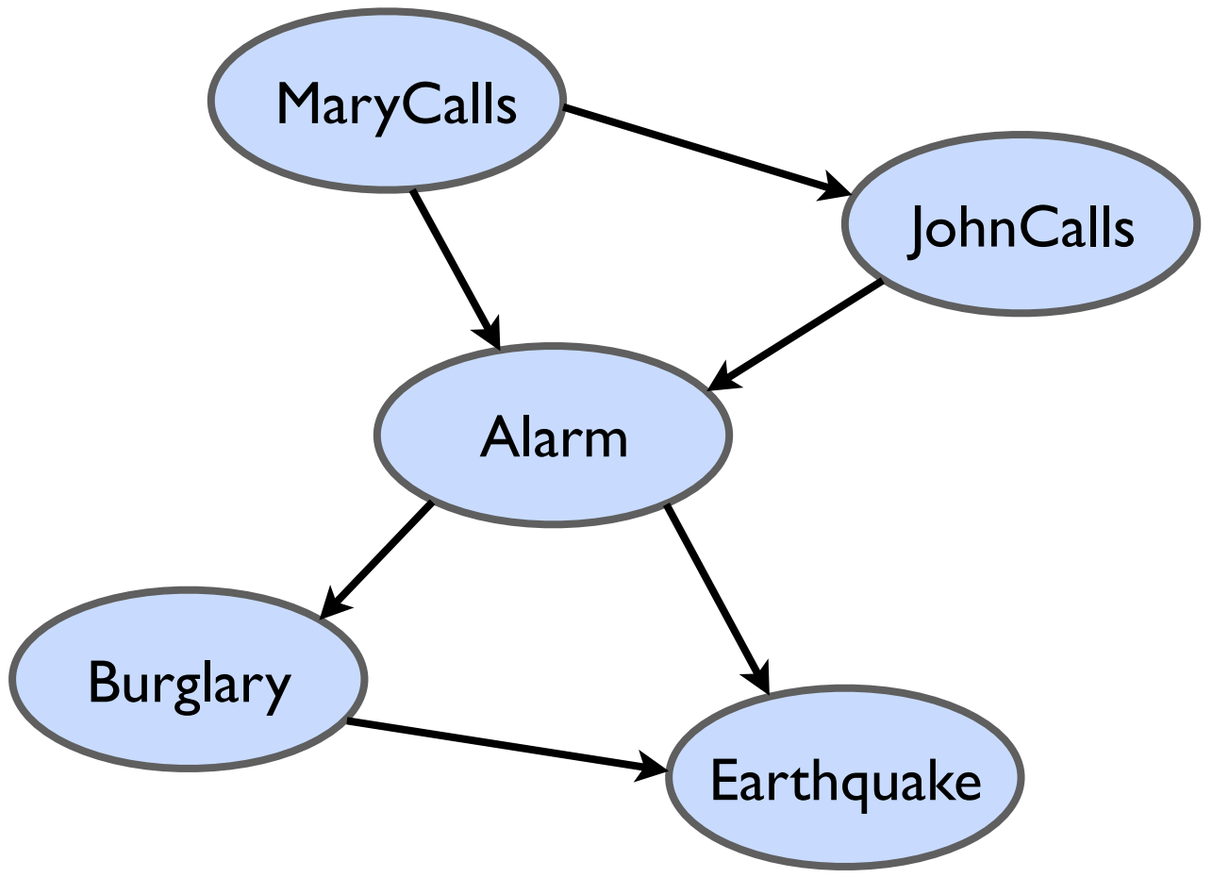
This choice of parents guarantees the global semantics:

$$\mathbb{P}(X_1,..., X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid X_1,..., X_{i-1}) \quad \text{(chain rule)}$$

$$= \prod_{i=1}^{n} \mathbb{P}(X_i \mid Parents(X_i)) \quad \text{(by construction)}$$

Question: What does that mean in the world of our coloured and partially skewed dice?

Construction example



Deciding conditional independence is hard in noncausal directions

(Causal models and conditional independence seem hardwired for humans!)

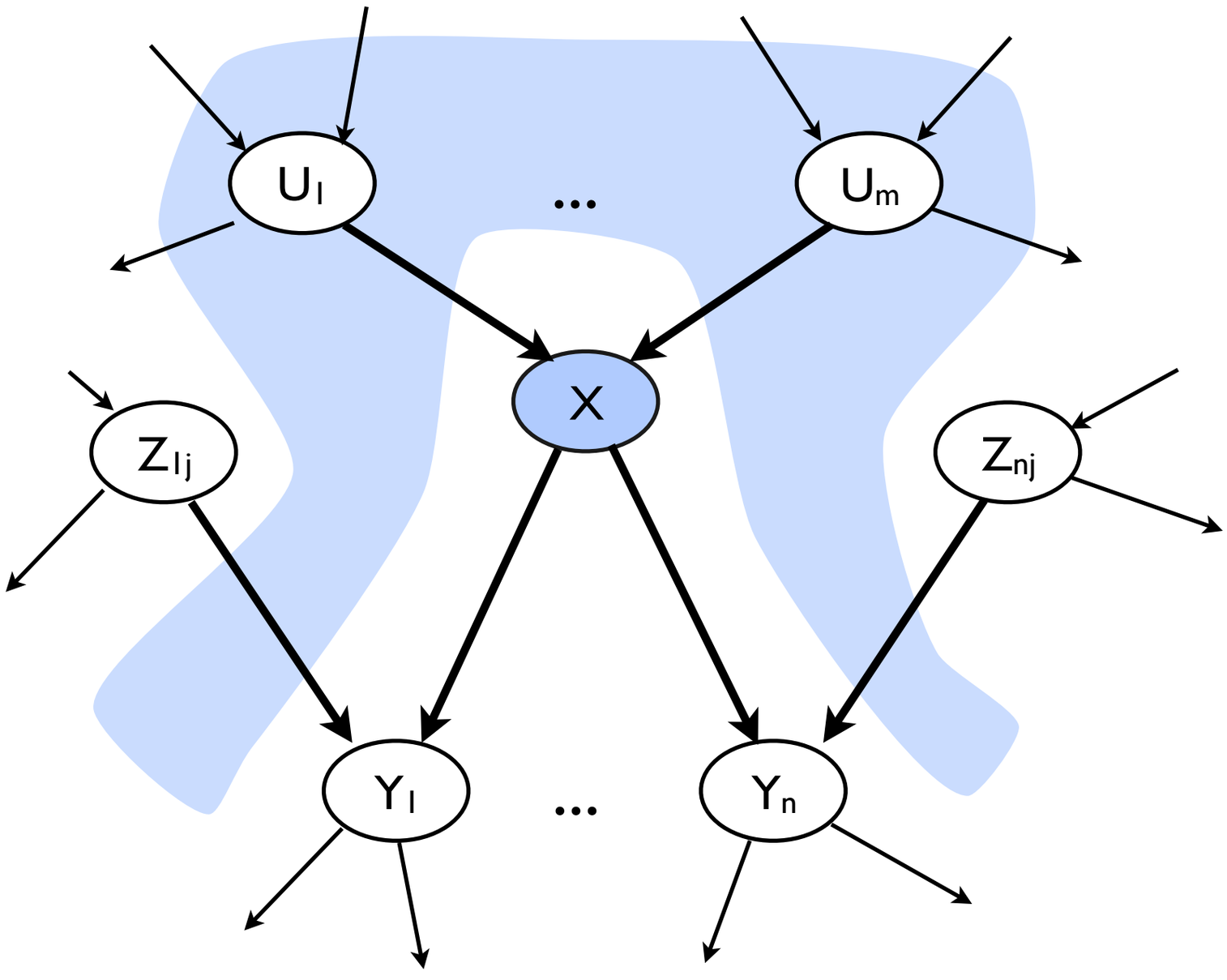
Assessing conditional probabilities is hard in noncausal directions

Network is less compact: *1 + 2 + 4 +2 +4 = 13* numbers

Hence: Choose preferably an order corresponding to the cause → effect "chain"

Local semantics

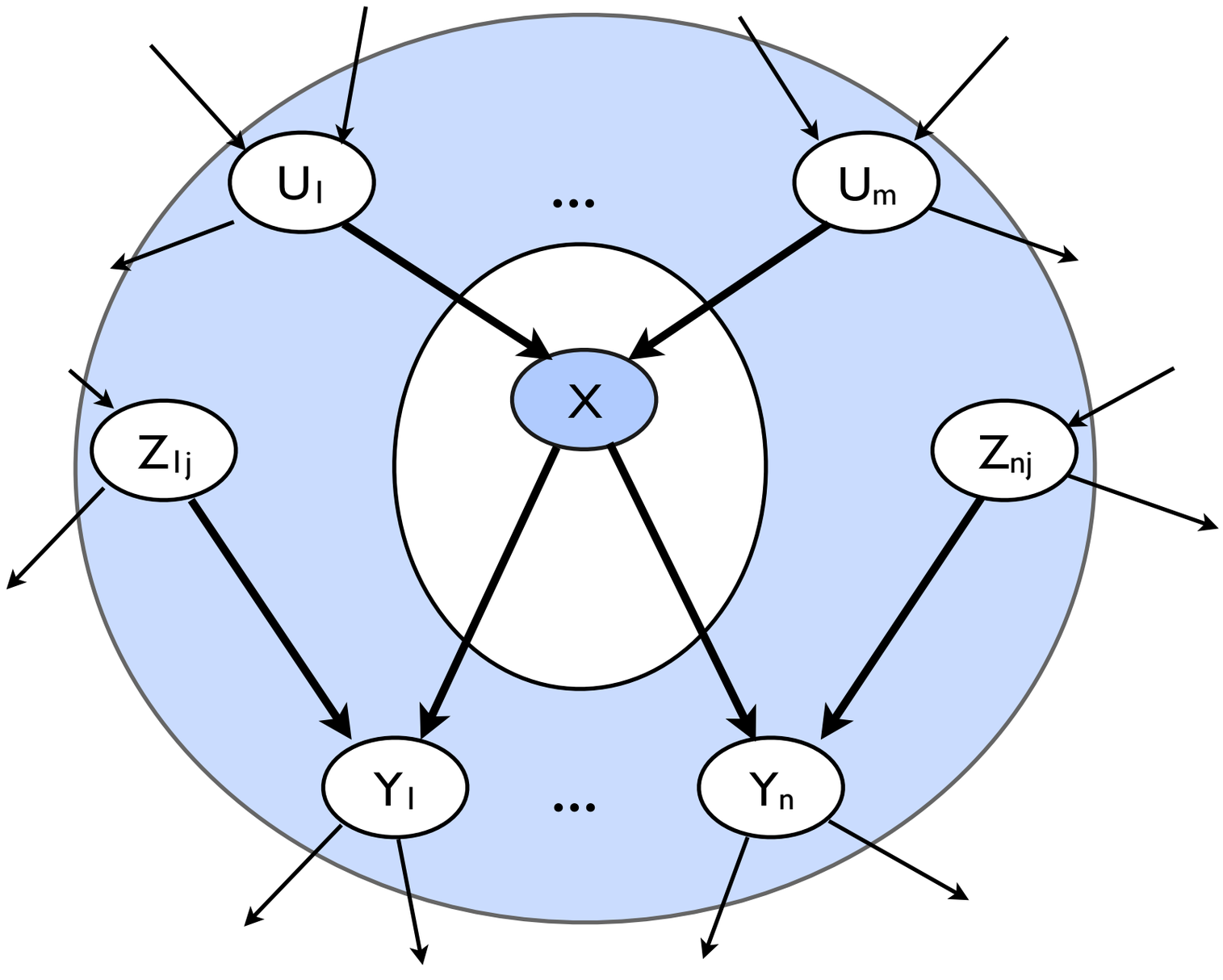
Local semantics: each node is conditionally independent of its non-descendants given its parents



Markov blanket

Each node is conditionally independent of all others given its

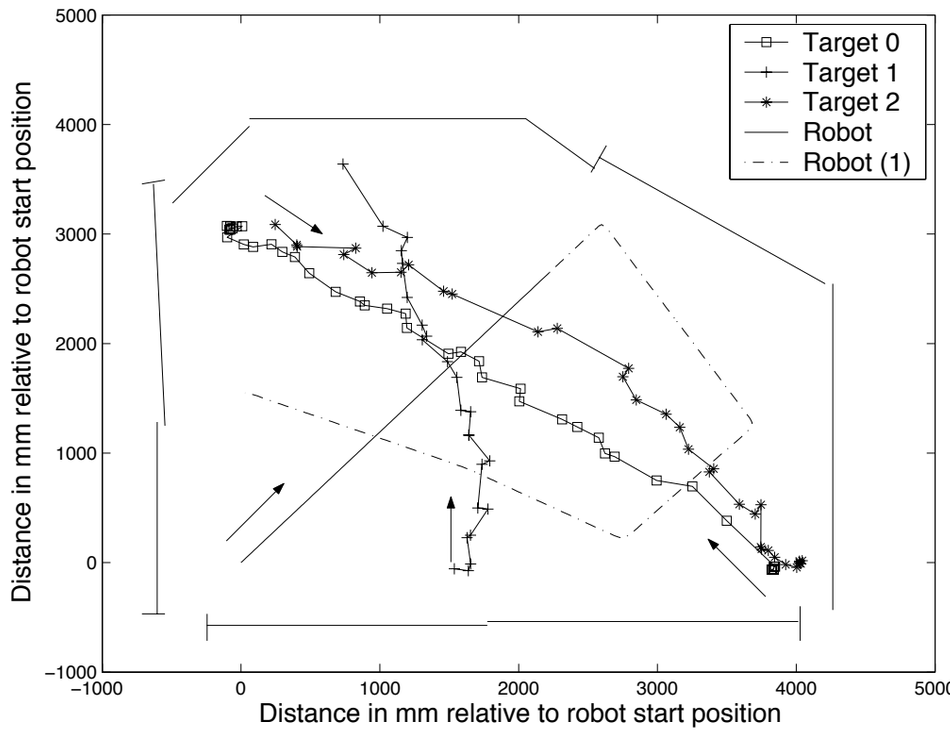
Markov blanket: parents + children + children's parents

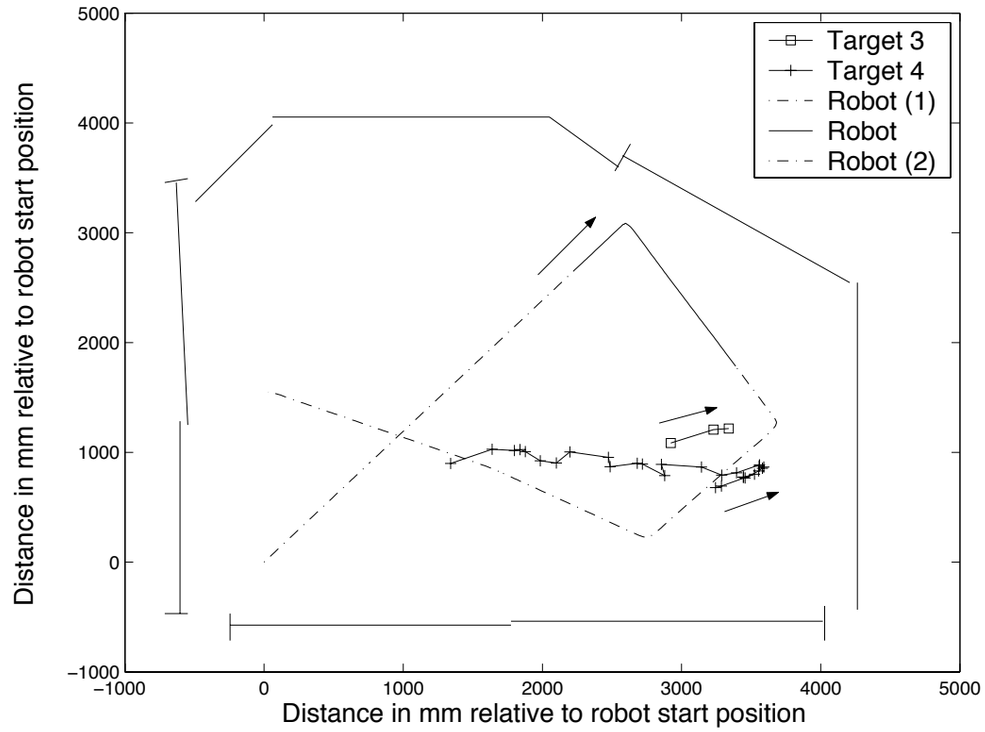


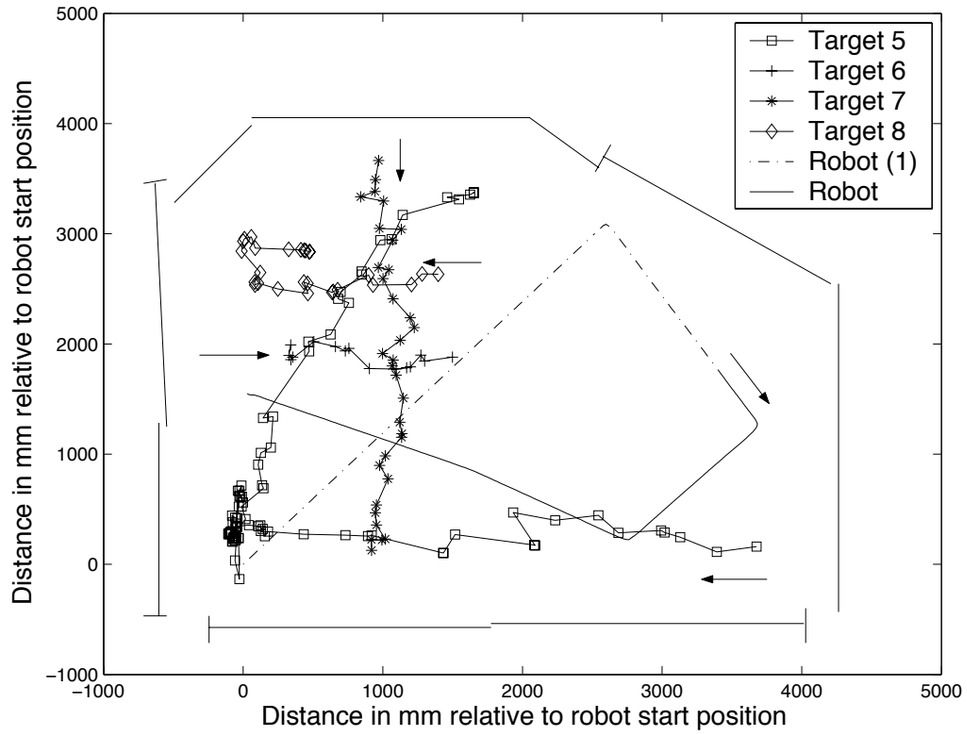
Outline

- Reasoning over time
  - Transition model and observation model
  - The Markov assumption
  - Stationary process
  - Forward Filtering
- Hidden Markov Models
- (Smoothing / Forward-Backward algorithm) (not in the lecture)

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

Previously…
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.

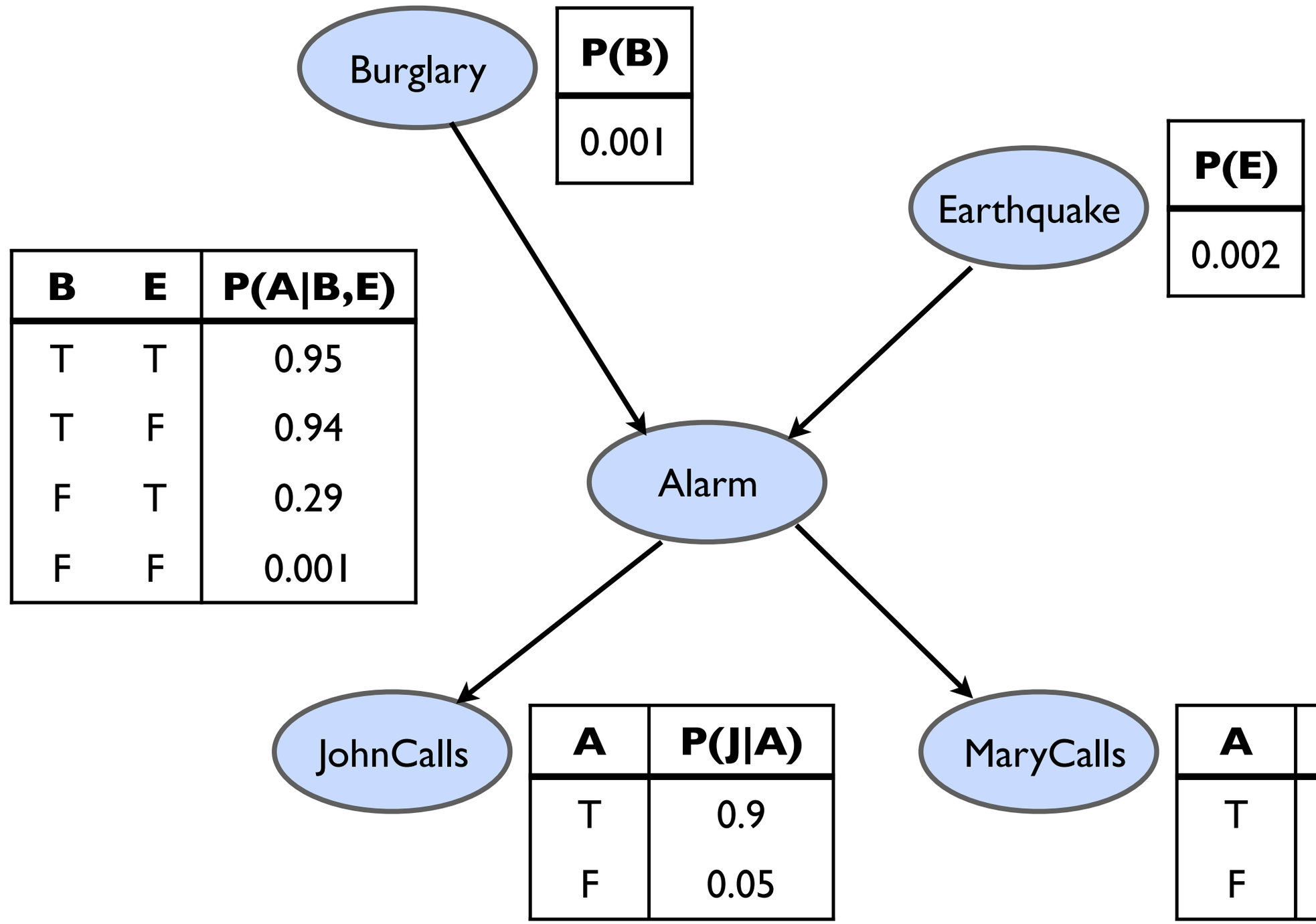
We assume to start out in $\boldsymbol{X}_0$, but evidence will only arrive after the first state transition is made: $\boldsymbol{E}_1$ is then the first piece of evidence to be plugged into the chain.

The "general" transition model would then specify

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

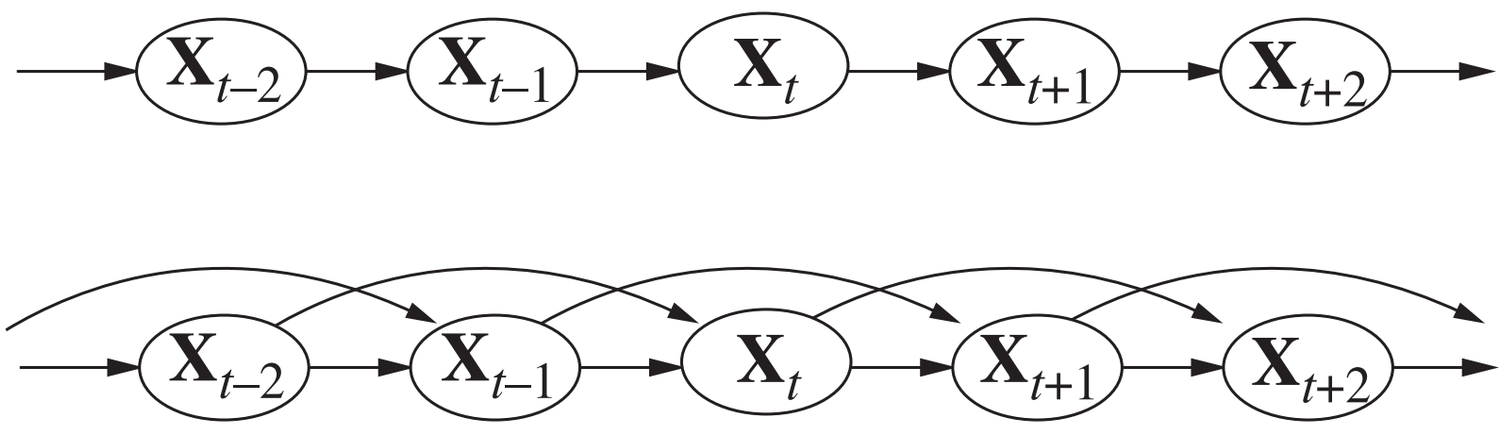
... this would mean we need full joint distributions over all time steps... or not?

A static network



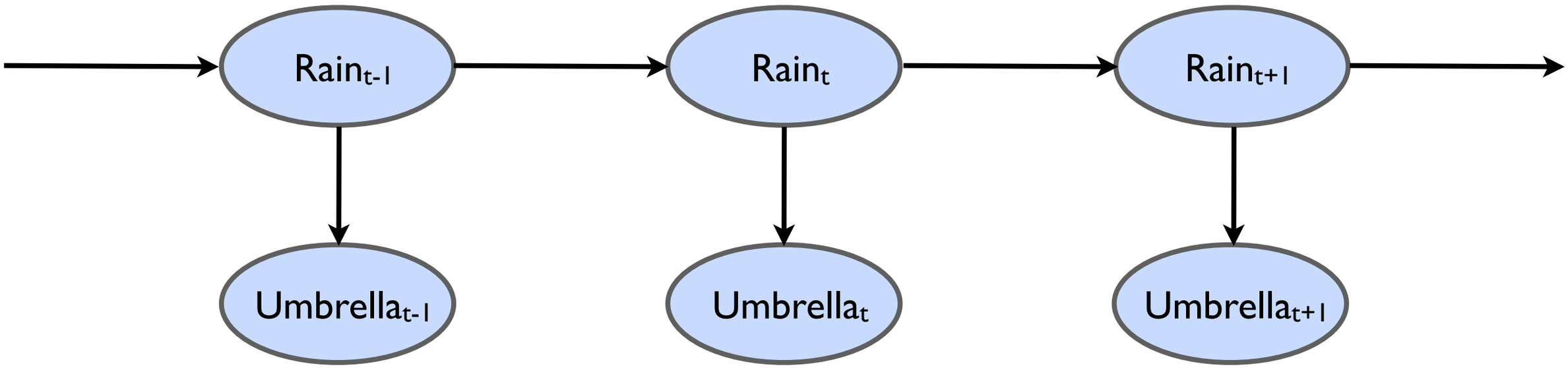
The Markov assumption

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

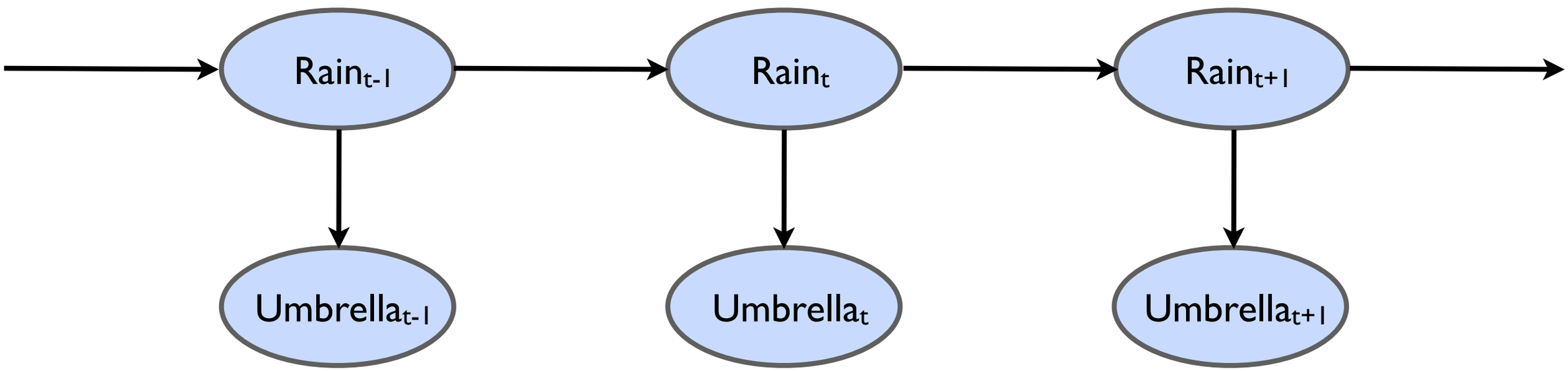


R_t	$P(U_t \mid R_t)$
T	0.9
F	0.2

A first-order Markov chain as Bayesian network

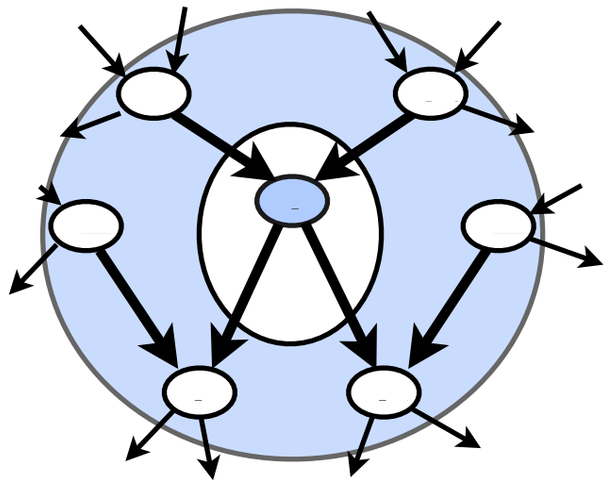
R_{t-1}	$P(R_t \mid R_{t-1})$
T	0.7
F	0.3

"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).

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:
  $\mathbb{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$
- **Predicting**: Computing the posterior over a *future* state, using evidence up to this point: $\mathbb{P}(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ 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: $\mathbb{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ 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)

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

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

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

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

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

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

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

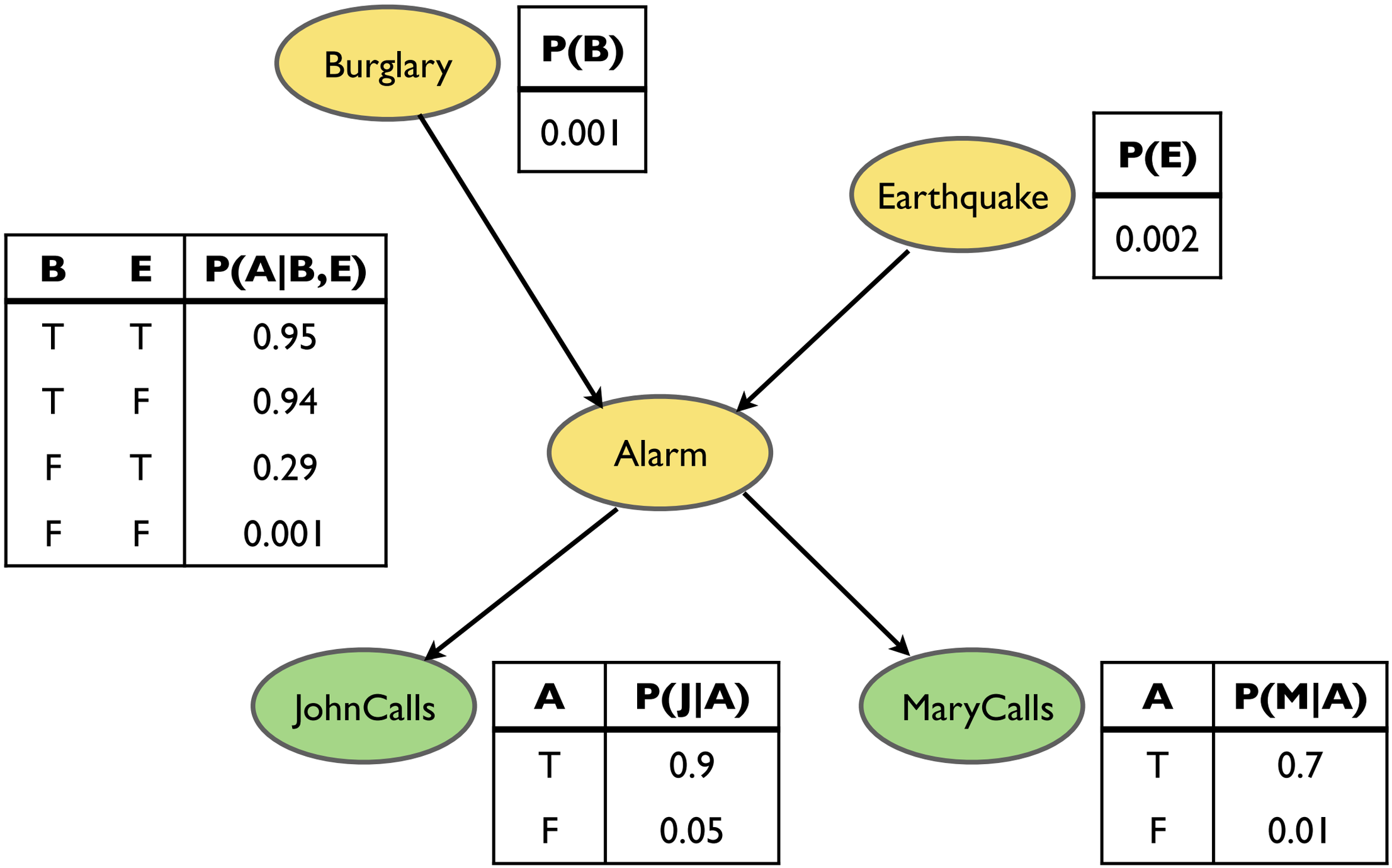
$\mathbf{f}_{1:t+1} = \alpha\ FORWARD(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$ through "forward step function")

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

Outline

- Reasoning over time
  - Transition model and observation model
  - The Markov assumption
  - Stationary process
  - Forward Filtering
- Hidden Markov Models
- (Smoothing / Forward-Backward algorithm) (not in the lecture)

Observable and "hidden" variables



“HMM”
Hidden Markov models

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

Main idea:

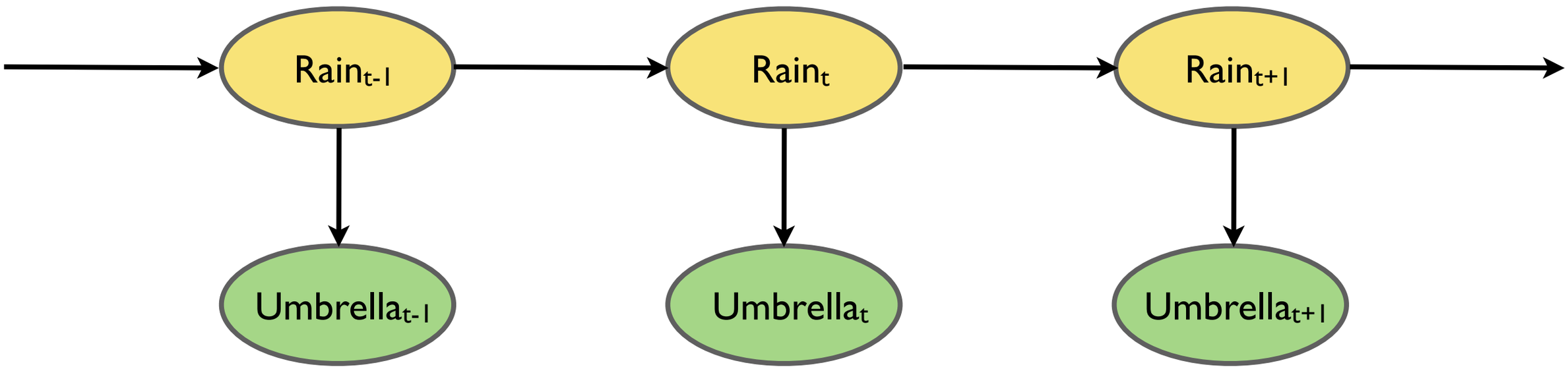
The state is represented by a *single discrete random variable*, taking on values that represent the (all) possible states of the world.

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.

The state is then assumed not to be observable directly, but some observations of “sensor readings” can be made.

An HMM as Bayesian network

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



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

"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 $S \times S$ matrix $\boldsymbol{T}$:

$$\boldsymbol{T}_{ij} = P(X_t = j \mid X_{t-1} = i)$$ 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 $S \times S$ diagonal matrix $\boldsymbol{O}$ in time step t with

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

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

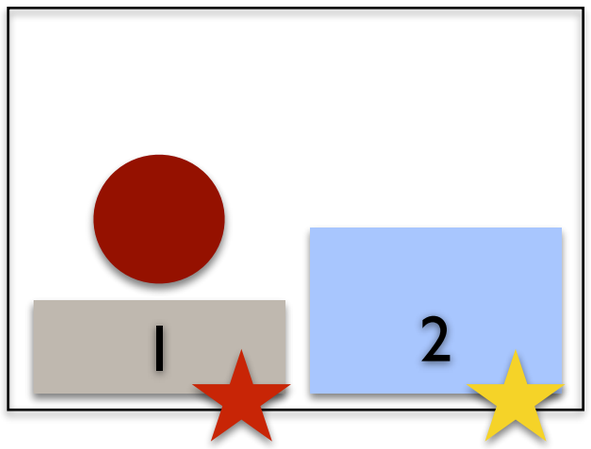
Forward-filtering equations as matrix-vector operations

Forward-equation

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

becomes $\boldsymbol{f}_{1:t+1} = \alpha \; \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$ (Matrix-matrix and matrix-vector scalar multiplication!)

Example matrix setup for a two-state world and three sensor readings



Ball behaviour:

P(step t: ball in 1 | step t-1: ball in 1) = 0.7

P(step t: ball in 2 | step t-1: ball in 1) = 0.3

P(step t: ball in 1 | step t-1: ball in 2) = 0.4

P(step t: ball in 2 | step t-1: ball in 2) = 0.6

Sensor correct:
"red" in state 1, "yellow" in state 2

P(sensor correct) = 0.8

P(sensor incorrect) = 0.15

P(sensor fails) = 0.05

$$T = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} \quad O_r = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 0.15 \end{pmatrix} \quad O_y = \begin{pmatrix} 0.15 & 0.0 \\ 0.0 & 0.8 \end{pmatrix} \quad O_f = \begin{pmatrix} 0.05 & 0.0 \\ 0.0 & 0.05 \end{pmatrix}$$

forward filtering with $\boldsymbol{f}_{1:0} = \mathbb{P}(\boldsymbol{X}_0)$ becomes then: $\boldsymbol{f}_{1:t+1} = \alpha \ \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$

Assignment 3

Set up a localisation problem as an HMM, do Forward Filtering to track an agent (robot) in a room (grid-world) without any landmarks

- Models (transition and observation) are given
- Implementation of "robot" and "sensor" simulation must be done
- Implementation of filtering algorithm including evaluation must be done

- Submit implementation by **February 26**
- Do peer review (organised through Canvas)
- Improve implementation
- Read article
- Write report
- Submit implementation AND report by **March 3**

Summary

Inference in temporal models

- Filtering and prediction (FORWARD)

Hidden Markov Models

- Simplified matrix representation for Forward-backward calculations
- the states causing the observable (uncertain) evidence are themselves HIDDEN, i.e. unobservable

Outline (additional material)

- Reasoning over time
  - Transition model and observation model
  - The Markov assumption
  - Stationary process
  - Forward Filtering
- Hidden Markov Models
- (Smoothing / Forward-Backward algorithm) (not in the lecture)

Prediction - filtering without the update

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

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

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

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

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

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

with x indicating componentwise (pointwise, cf course book, page 574) multiplication

Smoothing: calculating backward message

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

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

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

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

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

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

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

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

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

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

$\mathbb{P}(\boldsymbol{X}_k \mid \boldsymbol{e}_{1:t}) = fb(\boldsymbol{e}_{1:k}, \mathbb{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k))$ *with* $0 \leq k < t$

understand the past from the recent past

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

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)

Forward-backward equations as matrix-vector operations

Forward-equation (recap)

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

becomes $\boldsymbol{f}_{1:t+1} = \alpha \; \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$ (Matrix-matrix and matrix-vector scalar multiplication!)

Backward-equation (recap)

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

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

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

Cf. https://en.wikipedia.org/wiki/Forward–backward_algorithm
for an illustration of the book-example in the "umbrella world"

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-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-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-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 $\boldsymbol{b}_{t+1:t}$

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