Artificial Intelligence

B.Tech(AIDS)_V Semester

Academic Year: 2024-2025

Dr D. L. Sreenivasa Reddy

Email: dlsrinivasareddy_it@cbit.ac.in

Syllabus

UNIT-IV

Probabilistic Reasoning: Representing knowledge in an Uncertain Domain, The semantics of Bayesian networks, efficient representation of conditional distribution, Inference in Bayesian Networks, Inference in Temporal Models

Hidden Markov models. Markov Decision Process: MDP formulation, utility theory, utility functions, value iteration, policy iteration and partially observable MDPs.

QUANTIFYING UNCERTAINTY

- **Uncertainty.**
- **Belief state**—a representation of the set of all possible world states(ω) that it might be in.
- **Degree of belief**
- **A logical agent Vs A probabilistic agent**
- **Utility Theory:**
- **Decision theory = probability theory + utility theory** .
- The fundamental idea of decision theory is that an agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of the action. This is called the principle of maximum expected utility (MEU)

- Basic Probability Notation:
  - Ω= Sample Space:The possible worlds are mutually exclusive
  - ω= elements of the space, Basic Axioms of Probability is

$$0 \leq P(\omega) \leq 1 \text{ for every } \omega \text{ and } \sum_{\omega \in \Omega} P(\omega) = 1 \quad \text{For any proposition } \phi,\ P(\phi) = \sum_{\omega \in \phi} P(\omega).$$

  - Ex: Dice, P(Total=11) = P((5,6))+P((6,5)) = 1/36+ 1/36 = 1/18.
- Unconditional or prior Unconditional probability probabilities
- In AI, the sets are always described by **propositions(Favourable event)** in a formal language.
- Conditional or posterior probabilities.
- Ex: Rolling dice, Conditional probability doubles given that the first die is a 5.
- P(Doubles|Die Posterior probability 1 =5)

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)},$$

which holds whenever $P(b) > 0$. For example,

$$P(doubles \mid Die_1 = 5) = \frac{P(doubles \wedge Die_1 = 5)}{P(Die_1 = 5)}.$$

- Product rule: $P(a \wedge b) = P(a \mid b)P(b)$
- P(Doubles|Die Posterior probability 1 =5)
- Random variables
- Range
- Ex:The range of Total for two dice is the set {2,...,12} and the range of Die1 is {1,...,6}
- ∑x P(X =x)

$$\mathbf{P}(Weather) = \langle 0.6, 0.1, 0.29, 0.01 \rangle$$

$$P(Weather = sun) = 0.6$$
$$P(Weather = rain) = 0.1$$
$$P(Weather = cloud) = 0.29$$
$$P(Weather = snow) = 0.01\,,$$

Joint Probability Distribution

Full joint probability distribution:

Bird	Flier	Young	Probability
T	T	T	0.0
T	T	F	0.2
T	F	T	0.04
T	F	F	0.01
F	T	T	0.01
F	T	F	0.01
F	F	T	0.23
F	F	F	0.5

- P(Bird=T) = P(B) = 0.0 + 0.2 + 0.04 + 0.01 = 0.25
- P(Bird=T, Flier=F) = P(B, ~F) = P(B, ~F, Y) + F(B, ~F, ~Y) = 0.04 + 0.01 = 0.05

- Predefined ordering <sun,rain,cloud,snow>on the range of Weather
- Probability distribution for the random variable Weather
- Categorical distribution.
- The P notation Categorical distribution is also used for conditional distributions: P(X |Y) gives the values of P(X =xi |Y =yj) for each possible i, j pair
- Probability density function:
- Joint probability distribution of Weather and Cavity

$$\mathbf{P}(Weather, Cavity) = \mathbf{P}(Weather \mid Cavity)\mathbf{P}(Cavity)\,,$$

instead of as these $4 \times 2 = 8$ equations (using abbreviations W and C):

$$P(W = sun \wedge C = true) = P(W = sun|C = true)\,P(C = true)$$
$$P(W = rain \wedge C = true) = P(W = rain|C = true)\,P(C = true)$$
$$P(W = cloud \wedge C = true) = P(W = cloud|C = true)\,P(C = true)$$
$$P(W = snow \wedge C = true) = P(W = snow|C = true)\,P(C = true)$$
$$P(W = sun \wedge C = false) = P(W = sun|C = false)\,P(C = false)$$
$$P(W = rain \wedge C = false) = P(W = rain|C = false)\,P(C = false)$$
$$P(W = cloud \wedge C = false) = P(W = cloud|C = false)\,P(C = false)$$
$$P(W = snow \wedge C = false) = P(W = snow|C = false)\,P(C = false)\,.$$

- Example:
  - Consider the medical domain consisting of three Boolean variables: PickledLiver, Jaundice, Bloodshot, where the first indicates if a given patient has the "disease" PickledLiver, and the second and third describe symptoms of the patient.
  - Assume that Jaundice and Bloodshot are independent.

  - Based on no other information it knows that the prior probability P(PickledLiver) = 2^{-17}.
  - This represents the doctor's initial belief in this diagnosis.
  - After examination, Doctor determines that the patient has jaundice.
  - Doctor knows that P(Jaundice) = 2^{-10} and P(Jaundice | PickledLiver) = 2^{-3}
  - Computing the new updated probability in the patient having PickledLiver as:

$$P(\text{PickledLiver} \mid \text{Jaundice}) = P(P)P(J|P)/P(J)$$
$$= (2^{-17} * 2^{-3})/2^{-10}$$
$$= 2^{-10}$$

So, based on this new evidence, the doctor increases her belief in this diagnosis from 2-17 to 2-10 .

- Example 2: Patient's eyes are bloodshot, add this new piece of evidence and update the probability of PickledLiver given Jaundice and Bloodshot.

Say, $P(\text{Bloodshot}) = 2^{-6}$ and $P(\text{Bloodshot} \mid \text{PickledLiver}) = 2^{-1}$. Then, she computes the new conditional probability:

$$P(\text{PickledLiver} \mid \text{Jaundice}, \text{Bloodshot}) = (P(P)P(J|P)P(B|P))/(P(J)P(B))$$
$$= 2^{-10} * [2^{-1} / 2^{-6}]$$
$$= 2^{-5}$$

So, after taking both symptoms into account, the doctor's belief that the patient has a PickledLiver is 2^{-5}.

Representing Knowledge in an Uncertain Domain
Probability distribution

Probability of all possible worlds can be described using a table called a **full joint probability distribution** – the elements are indexed by values of random variables.

- Ex: A full joint probability distribution for toothache

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

  - P(toothache=true) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2
  - P(toothache=false) = 0.072+ 0.008 + 0.144 + 0.576 = 0.8
  - Can be described the table in a short way as: P(Toothache) = ⟨0.2, 0.8⟩

Example of probabilistic inference

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	**.108**	**.012**	**.072**	**.008**
¬ *cavity*	**.016**	**.064**	**.144**	**.576**

$$P(\phi) = \Sigma_{\omega:\omega|=\phi} P(\omega)$$
$$\mathbf{P(Y)} = \Sigma_{z \in \mathbf{Z}} \mathbf{P(Y}, z)$$

P(toothache) (= P(Toothache=true))
= 0.108 + 0.012 + 0.016 + 0.064 = 0.2

P(cavity ∨ toothache)
= 0.108+0.012+0.072+0.008+0.016+0.064 = 0.28

P(¬cavity|toothache)
= P(¬cavity ∧ toothache) / P(toothache)
= (0.016 + 0.064) / (0.108 + 0.012 + 0.016 + 0.064)
= 0.4

Normalization

	toothache		¬ toothache	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

$P(\neg cavity \mid toothache)$
$= P(\neg cavity \wedge toothache) \,/\, P(toothache)$
$= (0.016 + 0.064) \,/\, (0.108 + 0.012 + 0.016 + 0.064) = 0.4$

$P(cavity \mid toothache)$
$= P(cavity \wedge toothache) \,/\, P(toothache)$
$= (0.108 + 0.012) \,/\, (0.108 + 0.012 + 0.016 + 0.064) = 0.6$

Notice that denominators are identical in both formulas!

We even do not need to know the exact value of denominator:
$P(\neg cavity \mid toothache) + P(cavity \mid toothache) = 1$

We can use a **normalization constant** α instead, computed such that the evaluated distribution adds up to 1.

$\mathbf{P}(Cavity \mid toothache) = \alpha\, \mathbf{P}(Cavity, toothache)$
$= \alpha\, [\mathbf{P}(Cavity, toothache, catch) + \mathbf{P}(Cavity, toothache, \neg catch)]$
$= \alpha\, [\langle 0.108, 0.016 \rangle + \langle 0.012, 0.064 \rangle]$
$= \alpha\, [\langle 0.12, 0.08 \rangle] = [\langle 0.6, 0.4 \rangle]$

$P(\neg a) = 1 - P(a)$

inclusion-exclusion principle

$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$

chain rule

$\mathbf{P}(A,B,C,D)$

$= \mathbf{P}(A|B,C,D)\ \mathbf{P}(B,C,D)$

$= \mathbf{P}(A|B,C,D)\ \mathbf{P}(B|C,D)\ \mathbf{P}(C,D)$

$= \mathbf{P}(A|B,C,D)\ \mathbf{P}(B|C,D)\ \mathbf{P}(C|D)\ \mathbf{P}(D)$

Inference using full joint distributions

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	.108	.012	.072	.008
¬ *cavity*	.016	.064	.144	.576

How to answer questions?

Knowledge base is represented using full joint distribution.

To compute posterior probability of a query proposition given observed evidence, we add up probabilities of possible worlds in which the proposition is true (**marginalization** or **summing out**).

$P(\phi) = \Sigma_{\omega:\omega|=\phi} P(\omega)$

$\mathbf{P(Y)} = \Sigma_{z \in \mathbf{z}} \mathbf{P(Y}, z)$

Let us go back to **diagnostic problems**.

Usually we are looking for disease (the source of problems) based on symptoms (observations).

- we are interested in the **diagnostic direction** expressed as conditional probability P(disease|symptoms)

However, from past experience we often have other information:

- the probability of disease P(disease)
- the probability of symptoms P(symptoms)
- the **causal relation** expressed as conditional probability P(symptoms|disease)

How can this information be exploited to get the probability of the diagnostic direction?

Recall the product rule

$P(a \wedge b) = P(a|b)\ P(b) = P(b|a)\ P(a)$

We can deduce a so called **Bayes' rule** (law or theorem):

$P(a|b) = P(b|a)\ P(a) / P(b)$

in general:

$$\mathbf{P}(Y|X) = \mathbf{P}(X|Y)\ \mathbf{P}(Y) / \mathbf{P}(X) = \alpha\ \mathbf{P}(X|Y)\ \mathbf{P}(Y)$$

It looks like two steps backward as now we need to know $\mathbf{P}(X|Y)$, $\mathbf{P}(Y)$, $\mathbf{P}(X)$.

But these are the values that we frequently have.

$P(\text{cause}|\text{effect}) = P(\text{effect}|\text{cause})\ P(\text{cause}) / P(\text{effect})$

- $P(\text{effect}|\text{cause})$ describes the **causal direction**
- $P(\text{cause}|\text{effect})$ describes the **diagnostic relation**

Medical diagnosis

- from past cases we know P(symptoms|disease), P(disease), P(symptoms)
- for a new patient we know symptoms and looking for diagnosis P(disease|symptoms)

Example:

- meningitis causes a stiff neck 70% of the time
- the prior probability of meningitis is 1/50 000
- the prior probability of stiff neck is 1%

What is the probability that a patient having a stiff neck has meningitis?

$P(m|s) = P(s|m).P(m) / P(s) = 0.7 * 1/50000 / 0.01 = 0.0014$

Why the conditional probability for the diagnostic direction is not stored directly?

- diagnostic knowledge is often more fragile than causal knowledge
- for example, if there is a sudden epidemic of meningitis, the unconditional probability of meningitis P(m) will go up so P(m|s) should also go up while the causal relation P(s|m) is unaffected by the epidemic, as it reflects how meningitis works

What if there are more observations?

We can exploit conditional independence as follows

$$\mathbf{P}(Toothache, Catch, Cavity)$$
$$= \mathbf{P}(Toothache|Cavity)\ \mathbf{P}(Catch|Cavity)\ \mathbf{P}(Cavity)$$

If all the effects are conditionally independent given the cause variable, we get:

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(Effect_i|Cause)$$

Such a probability distribution is called a **naive Bayes model** (it is often used even in cases where the "effect" variables are not actually conditionally independent given the value of the cause variable).

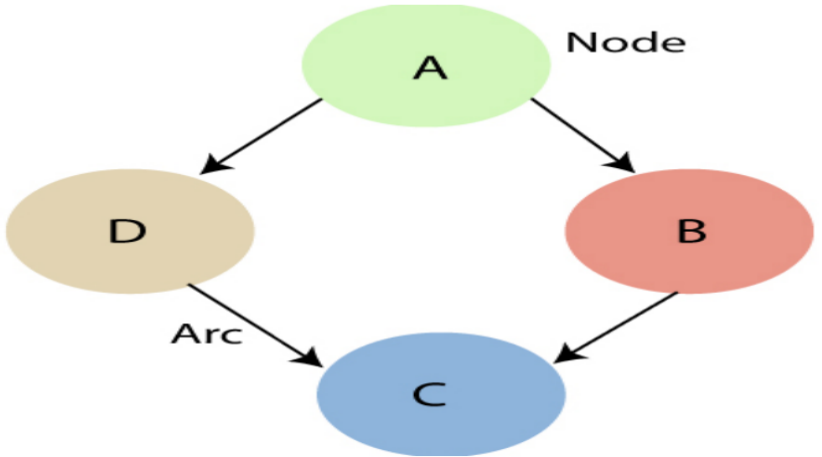
Bayesian Network/Belief network/ probabilistic network/causal network/knowledge map/

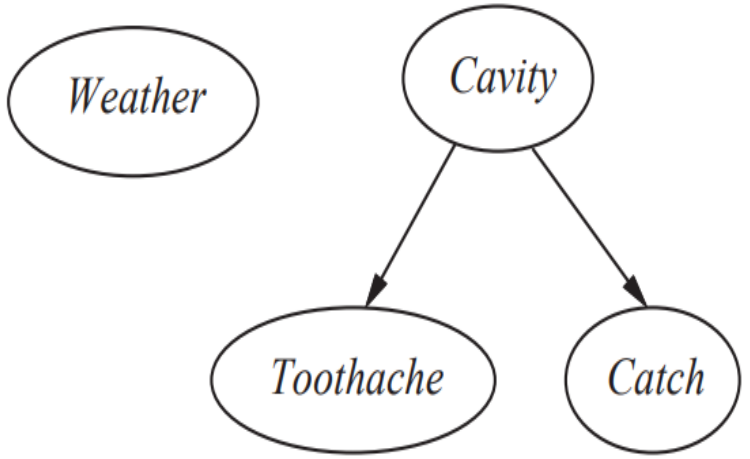
- Bayesian Network is a **data structure** to represent the dependencies among variables
- Extension of Bayesian Network is called decision network or influence diagram.
- A Bayesian network is a directed graph(DAG) in which each node is annotated with quantitative probability information(CPT)
  - 1. Each node corresponds to a random variable, which may be discrete or continuous.
  - 2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y , X is said to be a parent of Y.
  - 3. Each node Xi has a conditional probability distribution **P(Xi | Parents(Xi))**that quantifies the effect of the parents on the node.

Probabilistic Reasoning-Bayesian Network

- Directed Acyclic Graph
- Table of conditional probabilities.
- The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an Influence diagram.
- Weather is independent of the other variables
- Toothache and Catch are conditionally independent, given Cavity.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:





SEMANTICS OF BAYESIAN NETWORKS

Representing the full joint distribution:

P(X1 = x1 ∧ . . . ∧ Xn = xn)= P(x1, . . . , xn)

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

The Bayesian network represents the full joint probability distribution.

$$P(x_1,\ldots,x_n) = \Pi_i\ P(x_i \mid parents(X_i))$$

Tables **P**(X | Parents(X)) correspond to conditional probability defined by the underlying full joint probability distribution.

Because the full joint probability distribution can be used answer any query (in its domain) we can calculate the same answer using the Bayesian network (via marginalization).

How to build a Bayesian network?

We already have a clue:

$$P(x_1,\ldots,x_n) = \Pi_i\, P(x_i \mid \text{parents}(X_i))$$

Let us decompose $P(x_1,\ldots,x_n)$ using the chain rule

$$P(x_1,\ldots,x_n) = \Pi_i\, P(x_i \mid x_{i-1},\ldots,x_1).$$

Then we will get

$$\mathbf{P}(X_i \mid \text{Parents}(X_i)) = \mathbf{P}(X_i \mid X_{i-1},\ldots,X_1)$$

under the condition $\text{Parents}(X_i) \subseteq \{X_{i-1},\ldots,X_1\}$, which is satisfied by numbering the nodes in a way that is consistent with the partial order implicit in the graph structure.

Constructing Bayesian networks (algorithm)

Nodes:
determine the set of random variables that are required to model the domain and order them

- any order will work, but the resulting networks will be different
- a recommended order is such that causes precede effects

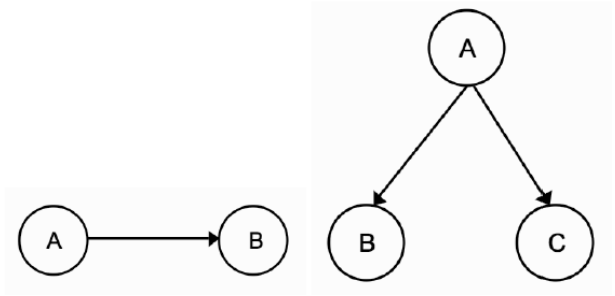
Arcs:
choose variables X_i in a given order from 1 to n

- in the set $\{X_1,\ldots,X_{i-1}\}$ choose a minimal set of parents for X_i, such that $\mathbf{P}(X_i \mid \text{Parents}(X_i)) = \mathbf{P}(X_i \mid X_{i-1},\ldots,X_1)$ holds
- for each parent insert a link from the parent to X_i
- write down the conditional probability table $\mathbf{P}(X_i \mid \text{Parents}(X_i))$

Some properties:

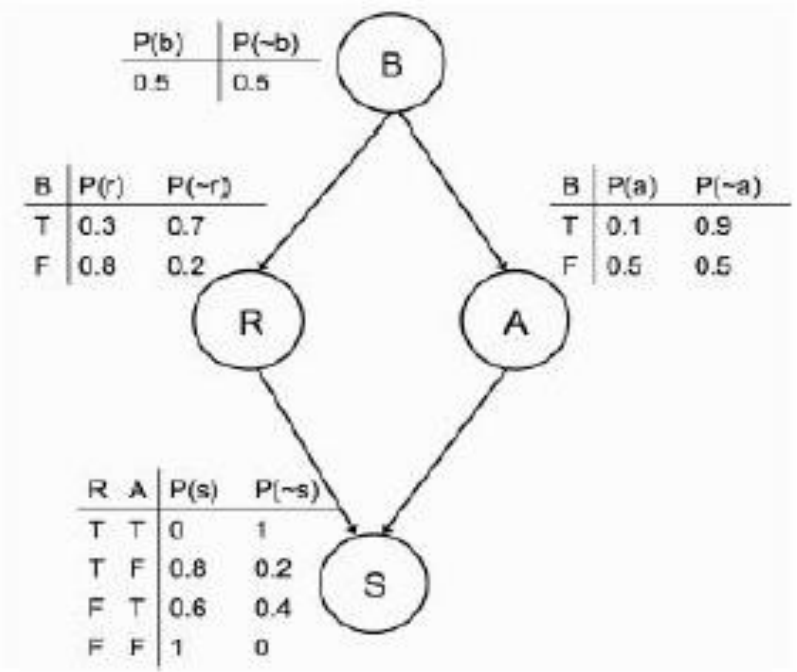
- the construction method guarantees that the network is acyclic
- the network does not contain no redundant probability values an so it is always consistent (satisfies the axioms of probability)

Graphs



DAG

Parent=Immediate Predecessor
Child=Descendent
Each variable is conditionally independent of its non descendants in the graph, given its parents

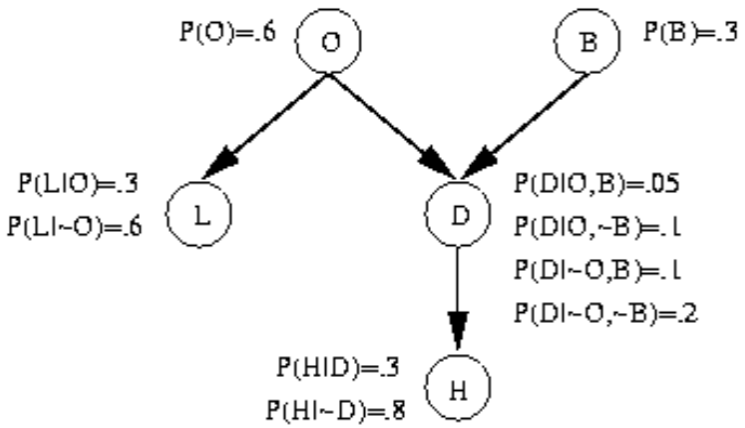


Bayesian Networks

P(X1=x1, ..., Xn=xn) = P(xi | Parents(Xi)) * ... * P(xn | Parents(Xn))

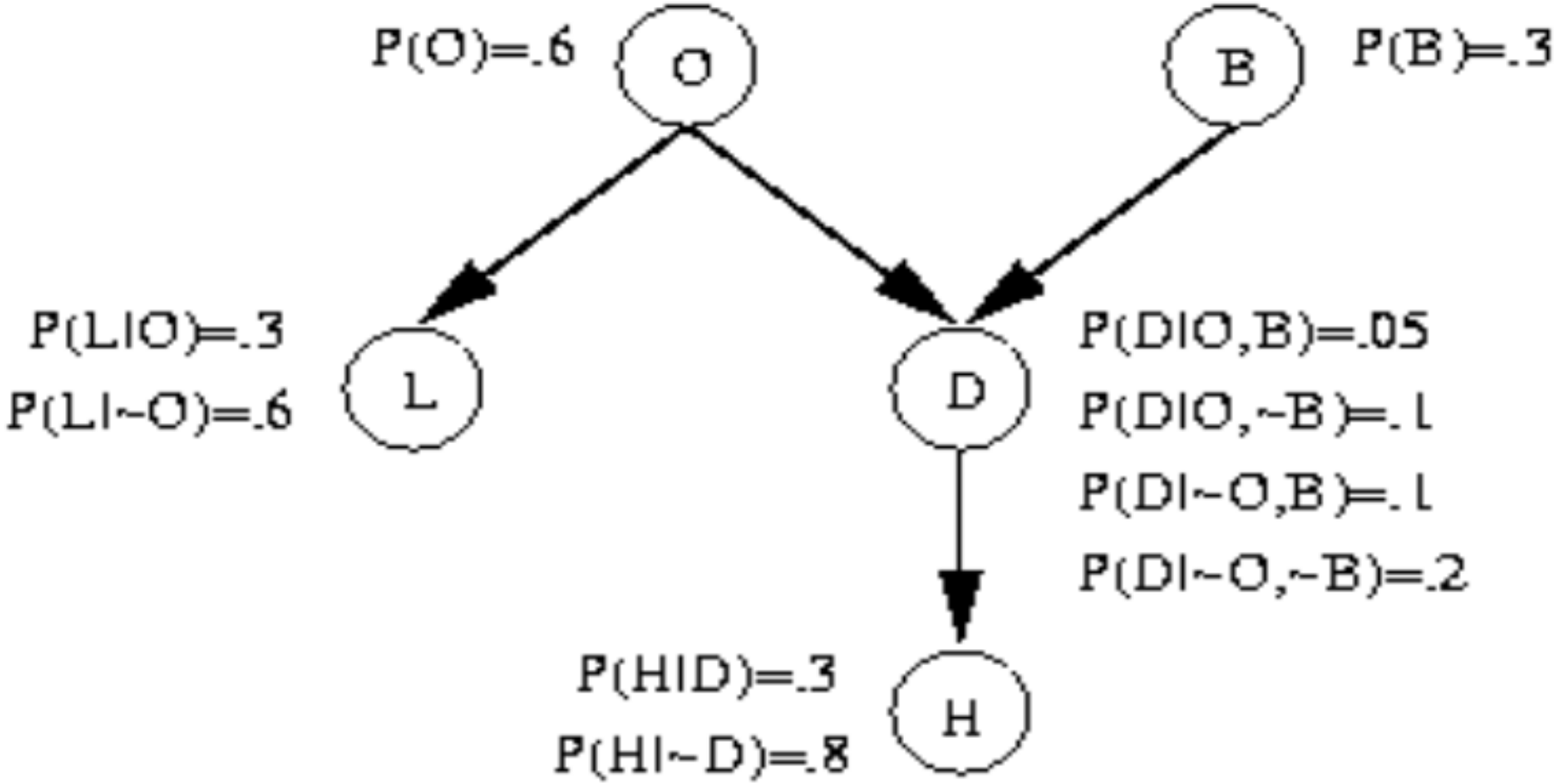
Given this information, define the following five Boolean random variables:

O: Everyone is Out of the house
L: The Light is on
D: The Dog is outside
B: The dog has Bowel troubles
H: I can Hear the dog barking



From this information, the following direct causal influences seem a

1. H is only directly influenced by D. Hence H is conditionally independent of L, O and B given D.
2. D is only directly influenced by O and B. Hence D is conditionally independent of L given O and B.
3. L is only directly influenced by O. Hence L is conditionally independent of D, H and B given O.
4. O and B are independent.


P(O)=.6
O
B
P(B)=.3
P(L|O)=.3
P(L|~O)=.6
L
D
P(D|O,B)=.05
P(D|O,~B)=.1
P(D|~O,B)=.1
P(D|~O,~B)=.2
P(H|D)=.3
P(H|~D)=.8
H

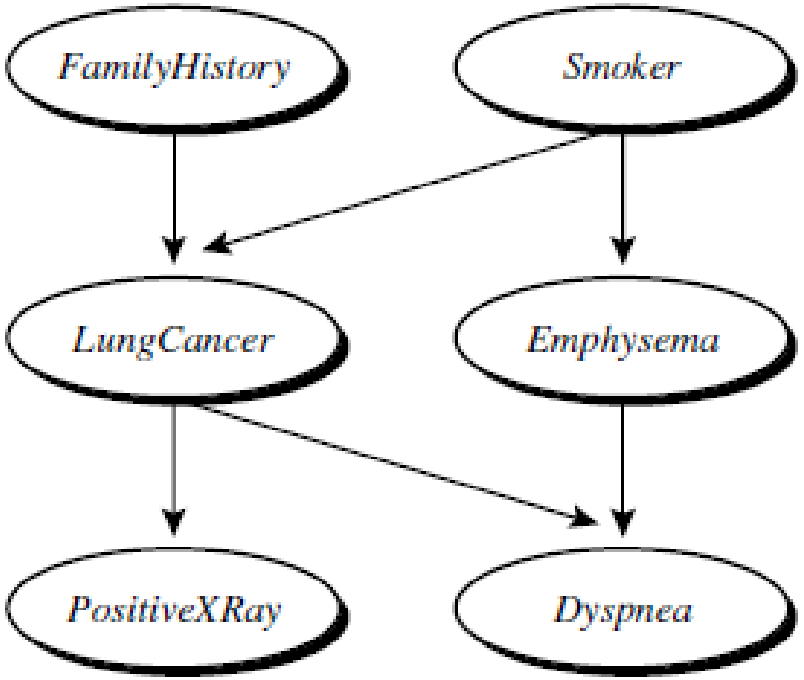
Computing Joint Probabilities from a Bayesian Net

Goal: Compute P(B,~O,D,~L,H)

P(B,~O,D,~L,H) = P(H,~L,D,~O,B)
= P(H | ~L,D,~O,B) * P(~L,D,~O,B) by Product Rule
= P(H|D) * P(~L,D,~O,B) by Conditional Independence of H and L,O, and B given D
= P(H|D) P(~L | D,~O,B) P(D,~O,B) by Product Rule
= P(H|D) P(~L|~O) P(D,~O,B) by Conditional Independence of L and D, and L and B, given O
= P(H|D) P(~L|~O) P(D | ~O,B) P(~O,B) by Product Rule
= P(H|D) P(~L|~O) P(D|~O,B) P(~O | B) P(B) by Product Rule
= P(H|D) P(~L|~O) P(D|~O,B) P(~O) P(B) by Independence of O and B
= (.3)(1 - .6)(.1)(1 - .6)(.3)
= 0.00144

where all of the numeric values are available directly in the Bayesian Net (since P(~A|B) = 1 - P(A|B)).

(a)



(b)

	FH, S	*FH, ¬S*	*¬FH, S*	*¬FH, ¬S*
LC	0.8	0.5	0.7	0.1
¬LC	0.2	0.5	0.3	0.9

$P(LungCancer = yes \mid FamilyHistory = yes, Smoker = yes) = 0.8$

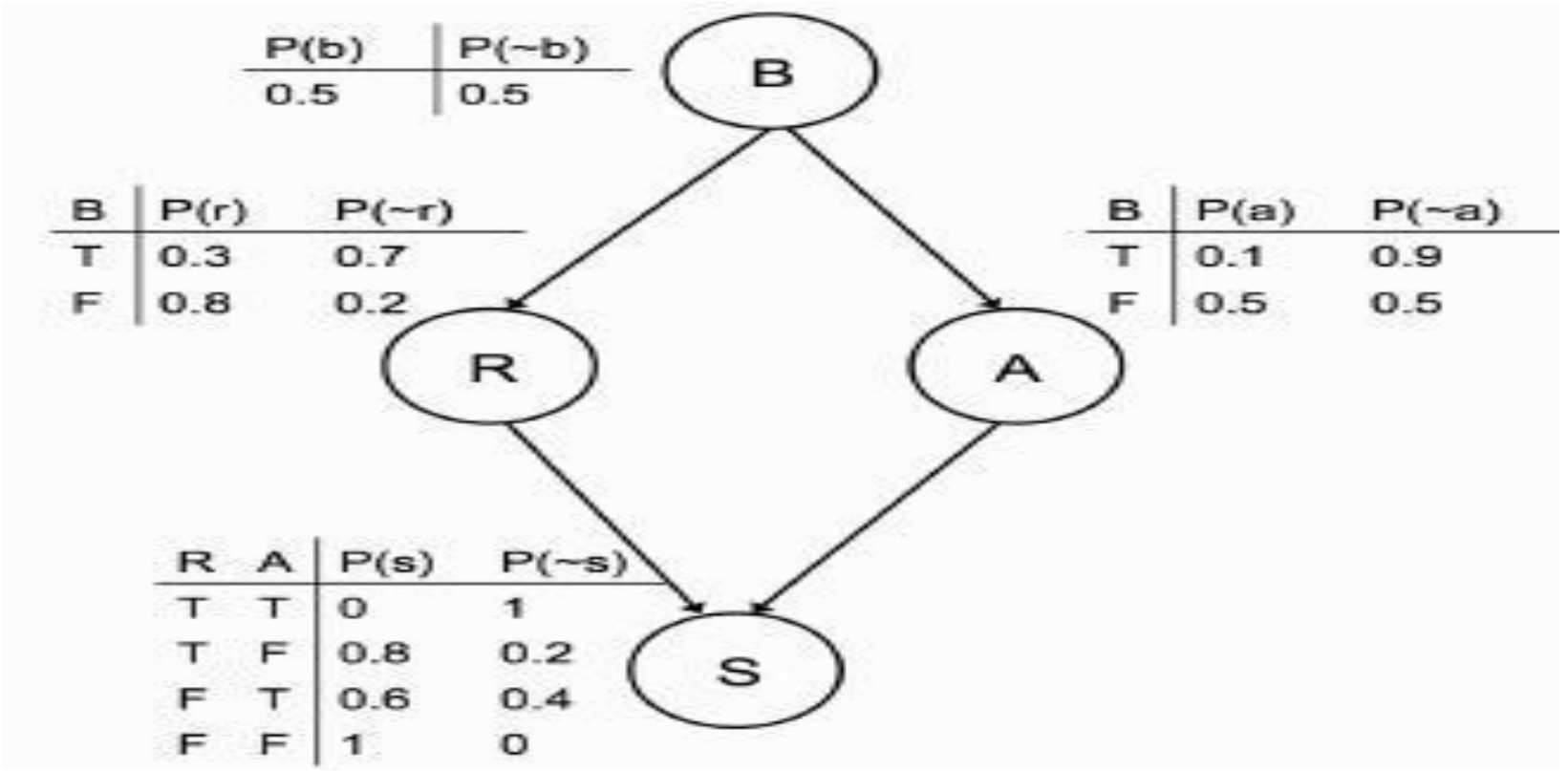
$P(LungCancer = no \mid FamilyHistory = no, Smoker = no) = 0.9$

joint probability distribution

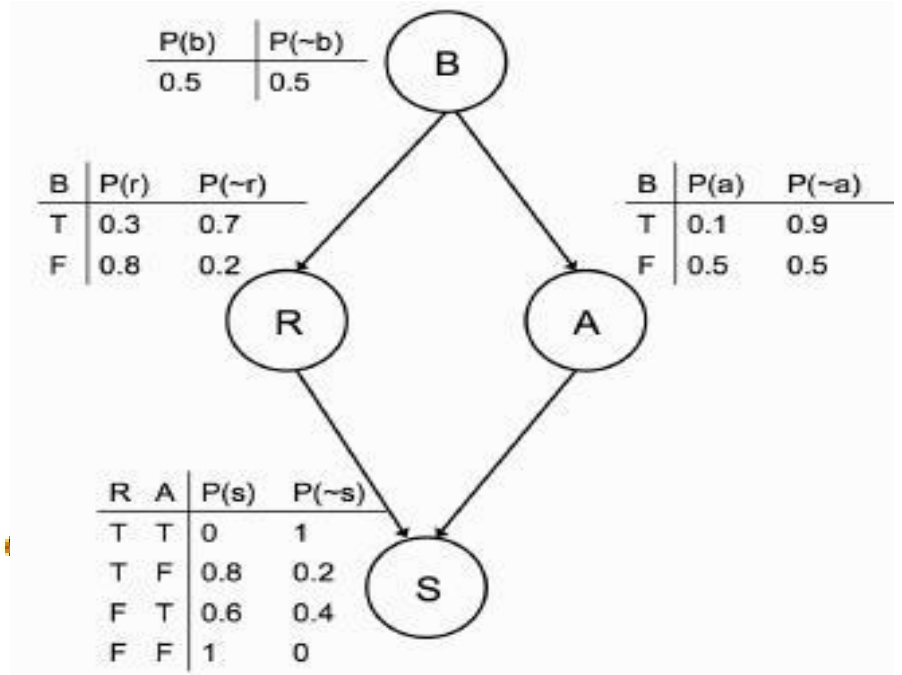
$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Parents(Y_i)),$$

- The sample graphical model. ‘B’ denotes a node stating whether the exam was boring, ‘R’ whether or not you revised, ‘A’ whether or not you attended lectures, and

‘S’ whether or not you will be scared before the exam.



$$
\begin{aligned}
P(s) &= \sum_{b,r,a} P(b,r,a,s) \\
&= \sum_{b,r,a} P(b) \times P(r|b) \times P(a|b) \times P(s|r,a) \\
&= \sum_{b} P(b) \times \sum_{r,a} P(r|b) \times P(a|b) \times P(s|r,
\end{aligned}
$$



$$
\begin{aligned}
P(s) &= 0.3 \times 0.1 \times 0 + 0.3 \times 0.9 \times 0.8 + 0.7 \times 0.1 \times 0.6 + 0.7 \times 0.9 \times 1 \\
&= 0.328.
\end{aligned}
$$

Constructing Bayesian networks (notes)

A Bayesian networks can often be far **more compact** than the full joint probability distribution (provided that the network is sparse).

- random variables are often influenced by a few other variables
- assume that each random variable is directly influenced by at most k other variables (and we have n such variables); then the space of representation is
  - $n.2^k$ for Bayesian network
  - 2^n for full joint distribution
- We can also **ignore some slight dependencies**, which makes the network smaller in exchange for less accuracy
  - for example, we assumed that call from Mary and John is driven by the alarm sound only but not for example by earthquake
- naturally we will get a compact Bayesian network only if we **choose the node ordering right**

An example (burglary detection)

We have a **burglar alarm** installed at home.
It is fairly reliable at detecting a **burglary**,
but occasionally responds to minor **earthquakes**.

Our neighbors Mary and John promised to **call us** when they hear the alarm.

- John nearly always calls when he hears alarm, but sometimes confuses the telephone ringing with the alarm
- Mary likes loud music and often misses the alarm altogether

We would like to **estimate the probability of a burglary given the evidence of who has or has not called**.

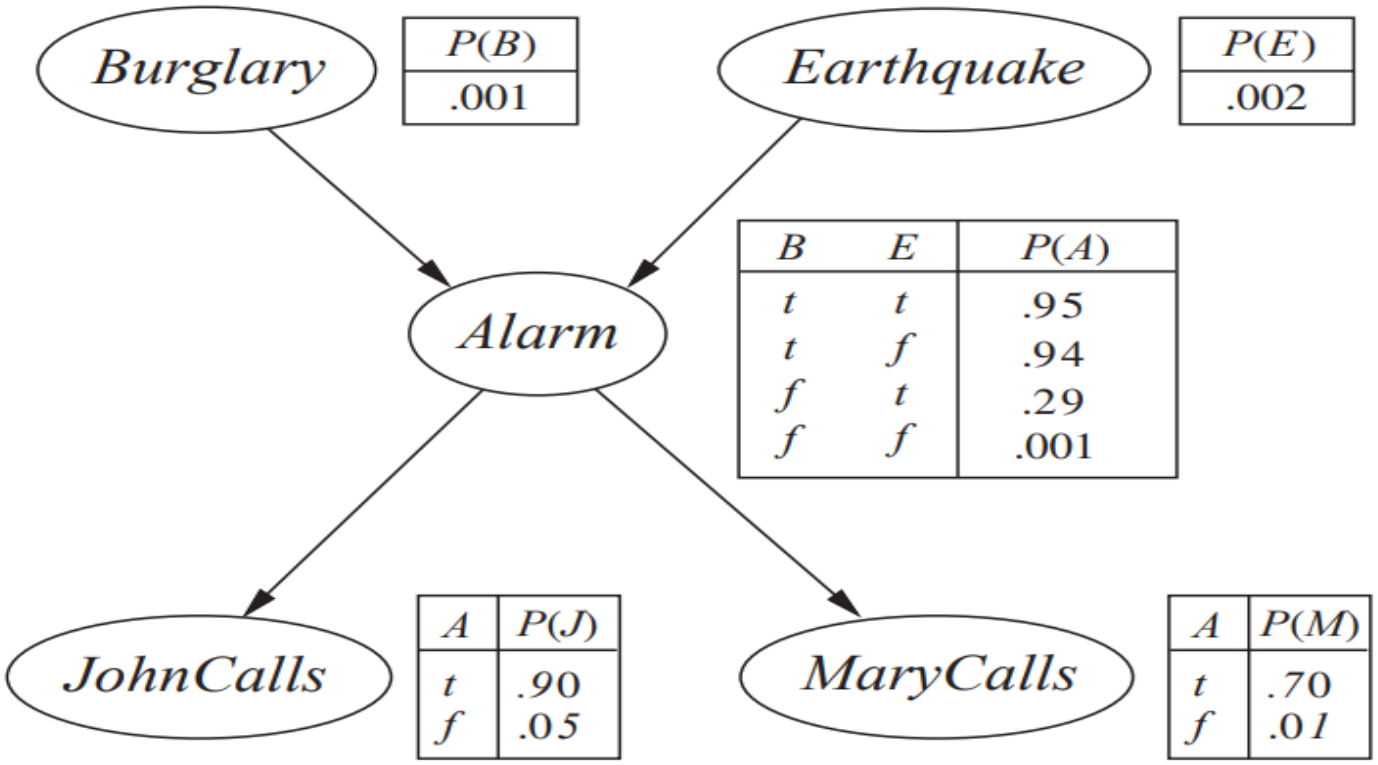
Other assumptions:

- neighbors do not perceive burglary directly and they do not notice minor earthquakes
- neighbors do not confer (they are independent)

Example-1:

- A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for Burglary, Earthquake, Alarm, JohnCalls, and MaryCalls , respectively.

- A table for a Boolean variable with k Boolean parents contains 2k independently specifiable probabilities



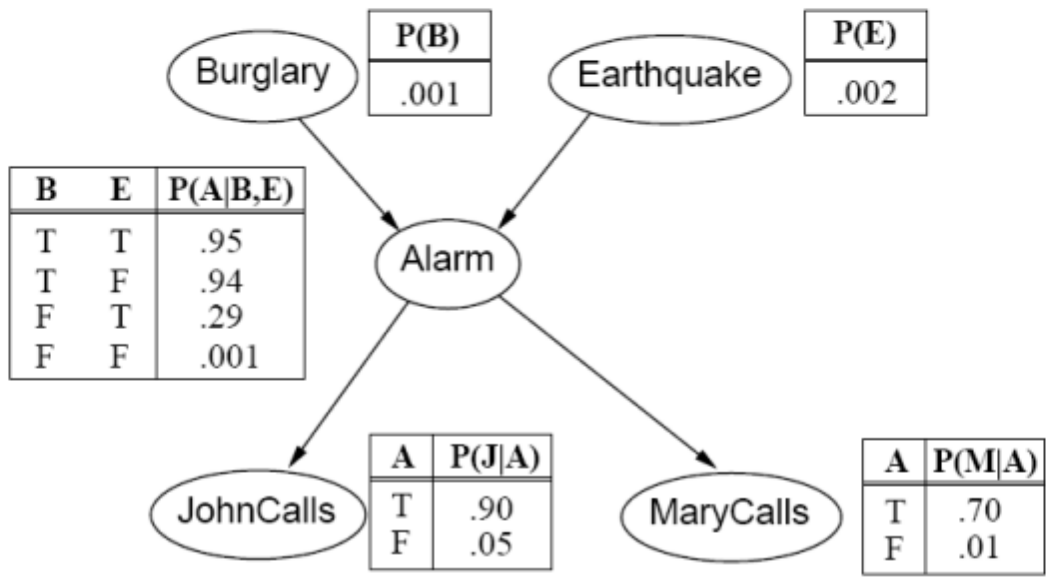
Bayesian network for burglary detection

Random Boolean variables represent possible events.

- some events (the telephone ringing, passing helicopter, alarm failure, …) are ignored

Conditional probability tables (CPTs) describe the conditional probability distributions

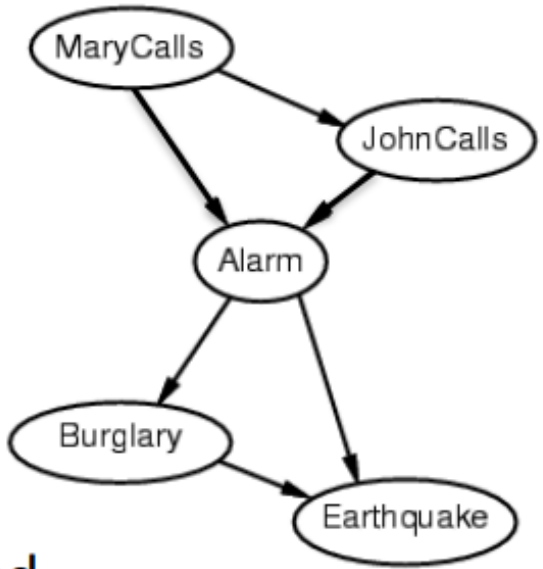
- recall that we can keep probability only for values true



Constructing Bayesian networks (an example)

Let us use the following order of random variables: MarryCalls, JohnCalls, Alarm, Burglary, Earthquake

- MarryCalls has no parents
- if Marry calls then the alarm is probably active which would make it more likely that John calls
- alarm is probably active if Marry or John calls
- if we know the alarm state then the calls from Marry and John do not influence whether the burglary happened



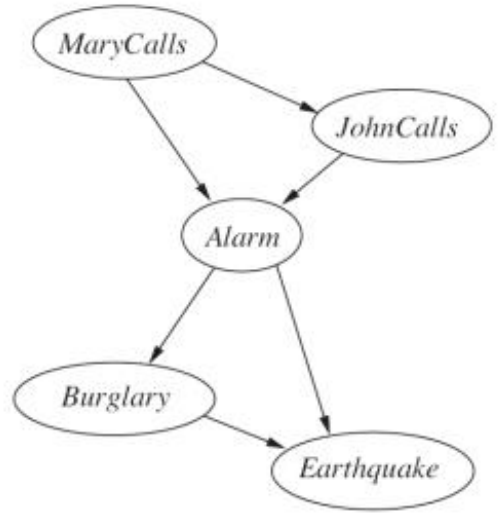
P(Burglary | Alarm, JohnCalls, MarryCalls) = P(Burglary | Alarm)

- the alarm is an earthquake detector of sorts, but if there was a burglary then it explains the alarm and the probability of an earthquake is only slightly above normal

Other orderings of nodes

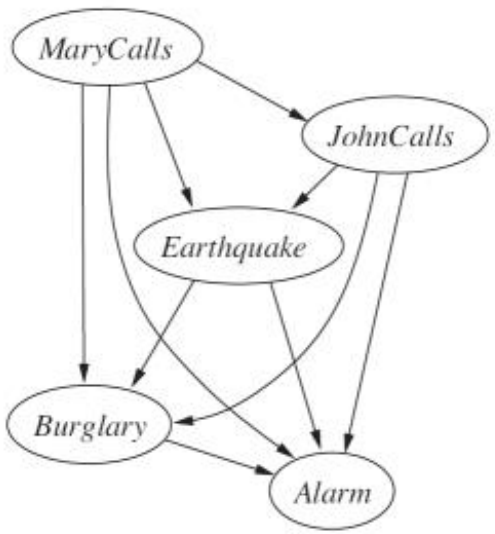
Only two more arcs (in comparison with the previous network) but the problem is how to fill in the CPTs.

- The same problem as using either causal or diagnostic direction.
- It is better to follow the causal direction (causes before effects).
  - leads to smaller networks and easier -to-fill CPTs



When using a wrong ordering we may get big networks, where nothing is saved in comparison to full joint probability distribution.

- MaryCalls, JohnCalls, Earthquake, Burglary, Alarm



Conditional independence in Bayesian networks

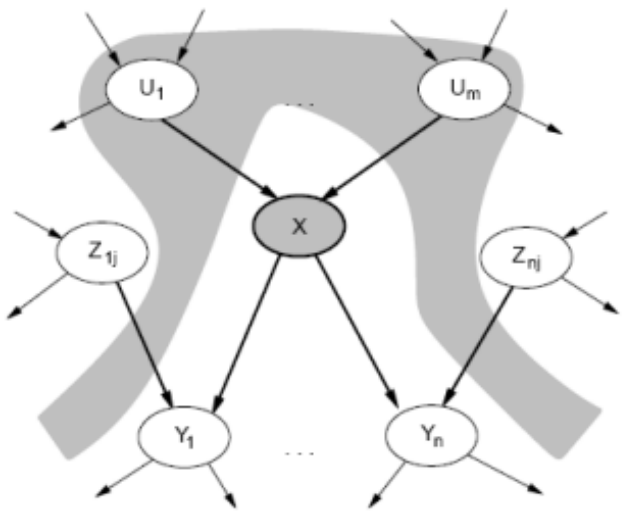
So far we looked at Bayesian networks in terms of the representation of the full joint distribution.

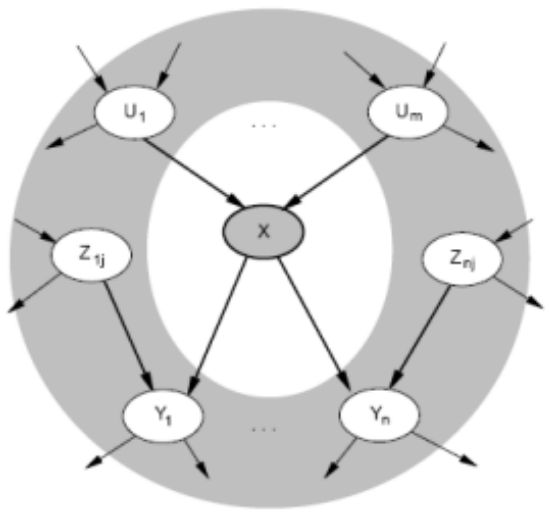
– useful to derive a method for constructing networks

Let us explore topological semantics of networks

a node is conditionally independent of its non-descendants given its parents

a node is conditionally independent of all other nodes given its parents, children, and children's parents (**Markov blanket**)





Inference by enumeration

X=Query Variable
E=Evidence variable
Y=hidden variable

We introduced the Bayesian networks to **do inference** – to deduce posterior probability of some variable(s) **X** from the query given the values **e** of observed variables (evidence), while having the other variables **Y** hidden.

$$\mathbf{P}(X|\mathbf{e}) = \alpha\, \mathbf{P}(X,\mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X,\mathbf{e},\mathbf{y})$$

the distribution $\mathbf{P}(X,\mathbf{e},\mathbf{y})$ can be computed as follows

$$P(x_1,\ldots,x_n) = \prod_i P(x_i \mid \text{parents}(X_i))$$

We can do some arithmetic tricks by moving some terms $P(x_i \mid \text{parents}(X_i))$ outside the summation.

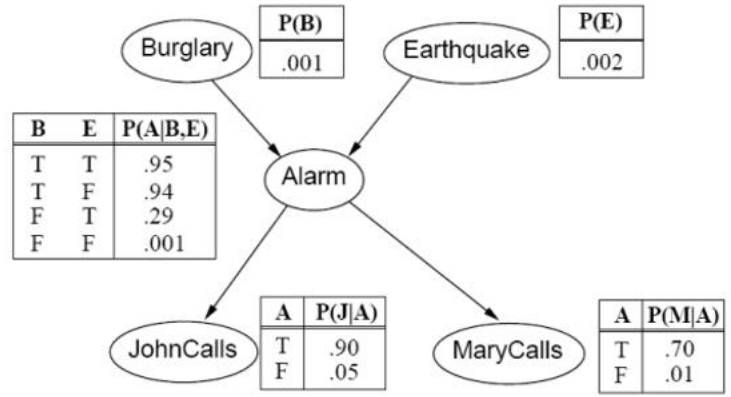
Inference by enumeration (example)

Assume a query about the probability of burglary when both Marry and John calls

$$P(b \mid j,m)$$

$$= \alpha \Sigma_e \Sigma_a P(b) P(e) P(a|b,e) P(j|a) P(m|a)$$

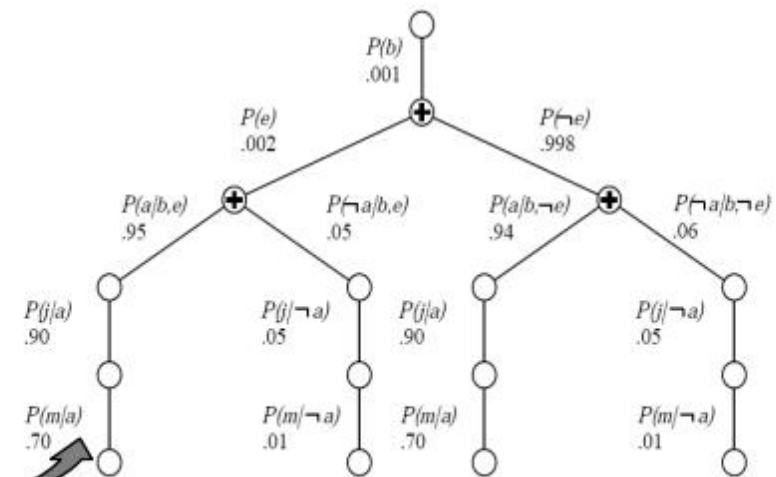
$$= \alpha P(b) \Sigma_e P(e) \Sigma_a P(a|b,e) P(j|a) P(m|a)$$



The structure of computation can be describes using a tree structure.

- it is very similar to solving CSPs and SAT

Notice that some parts are repeated!



Inference by enumeration

```
function ENUMERATION-ASK(X, e, bn) returns a distribution over X
  inputs: X,  the query variable
          e, observed values for variables E
          bn, a Bayesian network with variables {X} ∪ E ∪ Y

  Q(X) ← a distribution over X, initially empty
  for each value x_i of X do
      extend e with value x_i for X
      Q(x_i) ← ENUMERATE-ALL(VARS[bn], e)
  return NORMALIZE(Q(X))
```

```
function ENUMERATE-ALL(vars, e) returns a real number
  if EMPTY?(vars) then return 1.0
  Y ← FIRST(vars)
  if Y has value y in e
      then return P(y | Pa(Y)) × ENUMERATE-ALL(REST(vars), e)
      else return Σ_y P(y | Pa(Y)) × ENUMERATE-ALL(REST(vars), e_y)
          where e_y is e extended with Y = y
```

Enumeration repeats the same parts of the computation.

We can remember the result and reuse it later.

$$\mathbf{P}(B \mid j,m)$$
$$= \alpha\, \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a \mid B,e) P(j \mid a) P(m \mid a)$$
$$= \alpha\, \mathbf{f}_1(B) \sum_e \mathbf{f}_2(E) \sum_a \mathbf{f}_3(A,B,E)\, \mathbf{f}_4(A)\, \mathbf{f}_5(A)$$

Factors $\mathbf{f}_i$ are matrices (tables) corresponding to CPTs.

Evaluation will be done **from right to left**.

- **the product of factors** corresponds to the pointwise product (it is not a multiplication of matrices)
- **summing out a variable** is done by adding up the sub-matrices formed by fixing the variable to each of its values in turn

Operations on factors

The pointwise **product** of two factors yields a new factor whose variables are the union of the variables from the original factors.

$f(X_1,\ldots,X_j,Y_1,\ldots,Y_k,Z_1,\ldots Z_l) = f(X_1,\ldots,X_j,Y_1,\ldots,Y_k) \cdot f(Y_1,\ldots,Y_k,Z_1,\ldots Z_l)$

A	B	$f_1(A,B)$
T	T	0.3
T	F	0.7
F	T	0.9
F	F	0.1

*

B	C	$f_2(B,C)$
T	T	0.2
T	F	0.8
F	T	0.6
F	F	0.4

=

A	B	C	$f_3(A,B,C)$
T	T	T	0.06 = 0.3*0.2
T	T	F	0.24 = 0.3*0.8
T	F	T	0.42 = 0.7*0.6
T	F	F	0.28 = 0.7*0.4
F	T	T	0.18 = 0.9*0.2
F	T	F	0.72 = 0.9*0.8
F	F	T	0.06 = 0.1*0.6
F	F	F	0.04 = 0.1*0.4

Then we **sum out** a variable to eliminate it: $\Sigma_a f(A,B,C) = f(B,C)$

A	B	C	$f_3(A=T,B,C)$
T	T	T	0.06
T	T	F	0.24
T	F	T	0.42
T	F	F	0.28

+

A	B	C	$f_3(A=F,B,C)$
F	T	T	0.18
F	T	F	0.72
F	F	T	0.06
F	F	F	0.04

=

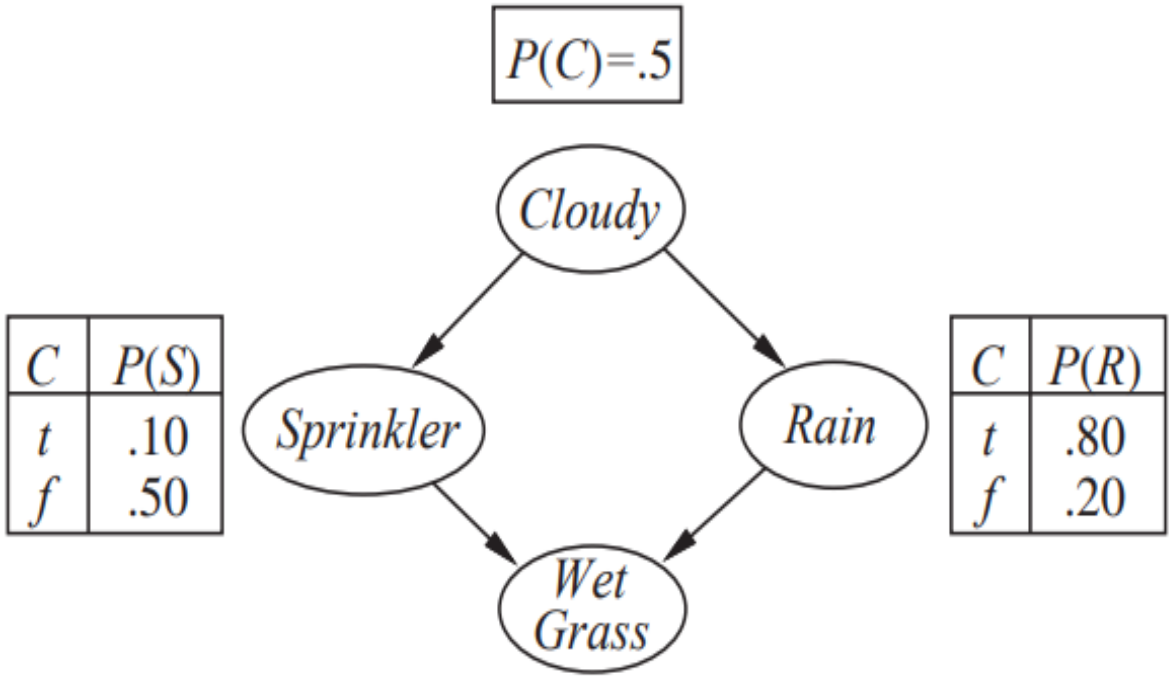
B	C	$f_4(B,C)$
T	T	0.24
T	F	0.96
F	T	0.48
F	F	0.31

Variable elimination (algorithm)

```
function ELIMINATION-ASK(X, e, bn) returns a distribution over X
  inputs: X, the query variable
          e, evidence specified as an event
          bn, a belief network specifying joint distribution P(X_1, ..., X_n)
  factors ← []; vars ← REVERSE(VARS[bn])
  for each var in vars do
      factors ← [MAKE-FACTOR(var, e)|factors]
      if var is a hidden variable then factors ← SUM-OUT(var, factors)
  return NORMALIZE(POINTWISE-PRODUCT(factors))
```

- The algorithm works for any ordering of variables.
- The complexity is given by the size of the largest factor constructed during the operation of the algorithm.
- Eliminate whichever variable minimizes the size of the next factor to be constructed (heuristic).

Variable Elimination-Clustering

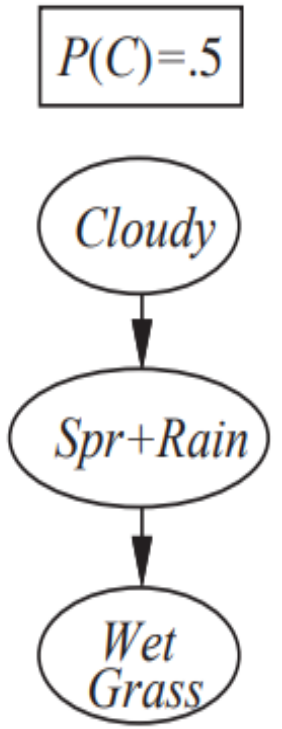


C	P(S)
t	.10
f	.50

C	P(R)
t	.80
f	.20

S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00

(a)

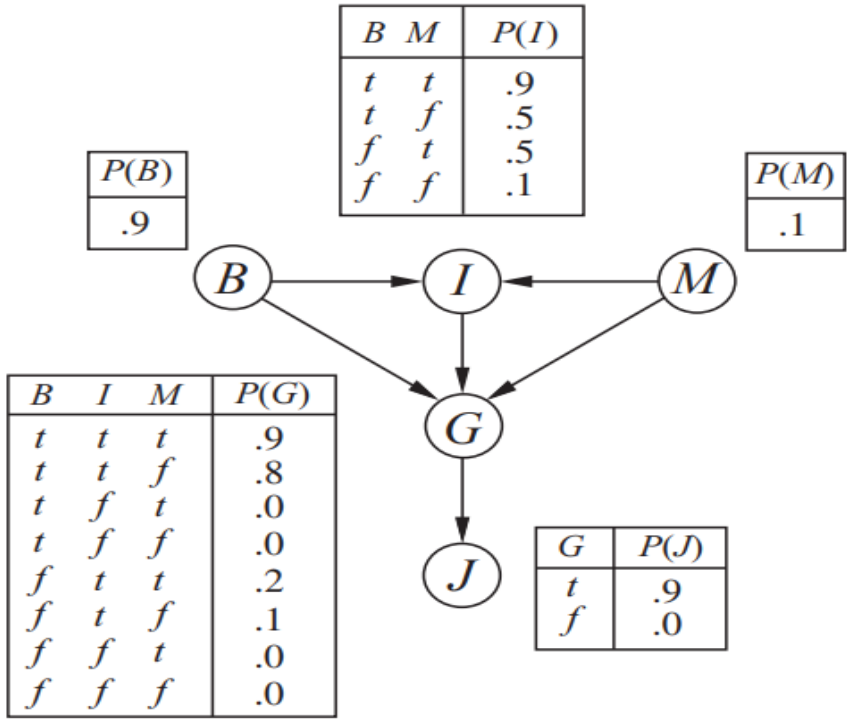


S+R	P(W)
t t	.99
t f	.90
f t	.90
f f	.00

C	P(S+R=x)			
	t t	t f	f t	f f
t	.08	.02	.72	.18
f	.10	.40	.10	.40

(b)

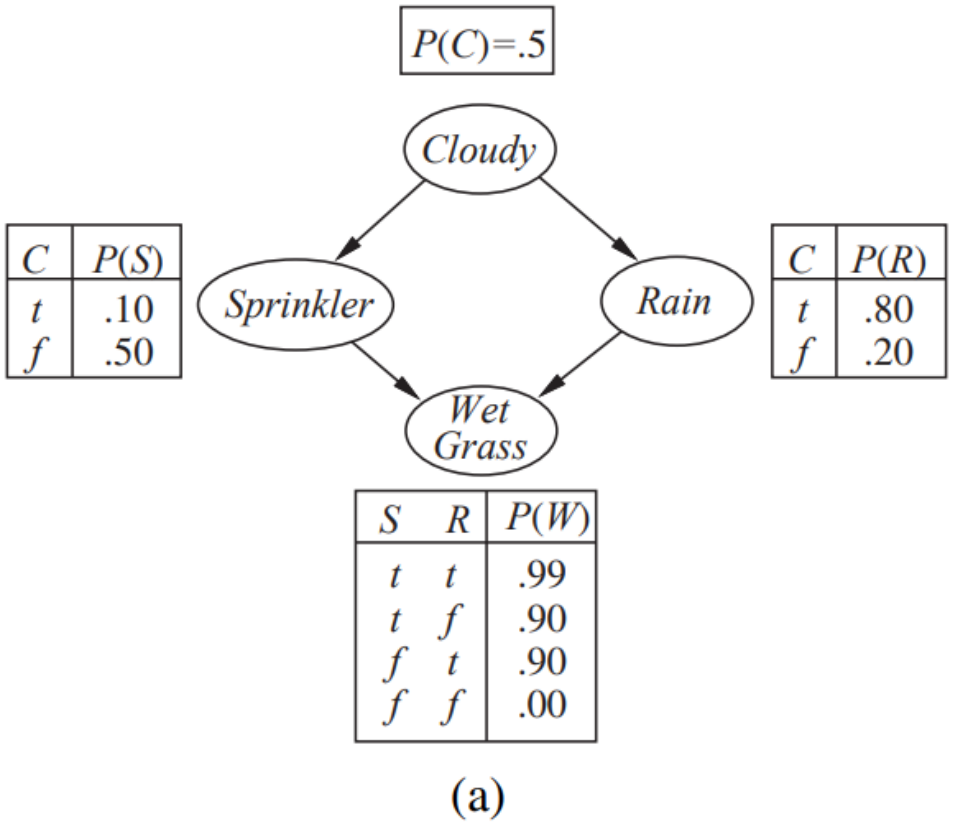
Assignment-2



Where
A=simple Bayes net with Boolean variables
B = BrokeElectionLaw,
I = Indicted,
M = PoliticallyMotivatedProsecutor ,
G = FoundGuilty,
J = Jailed.

Calculate the value of $P(b, i, m, \neg g, j)$

Calculate the probability that someone goes to jail given that they broke the law, have been indicted, and face a politically motivated prosecutor.



(a)

Find

1.P(w|C)

2. P(W,C,~S,R)

3. P(S,R,C)

4. P(W)

Probabilistic Reasoning over Time

A CHANGING WORLD IS MODELED USING A VARIABLE FOR EACH ASPECT OF THE WORLD STATE at each point in state.

THE TRANSITION MODEL DESCRIBES 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

Time and Uncertainty

Inference in Temporal Models

Hidden Markov Models

Kalman Filters.

Time and Uncertainty

- Static Worlds: RANDOM VARIABLE HAS A **SINGLE FIXED** VALUE

In situation calculus, we view the world as a series of snapshots (time slices).

A similar approach can be applied in probabilistic reasoning.

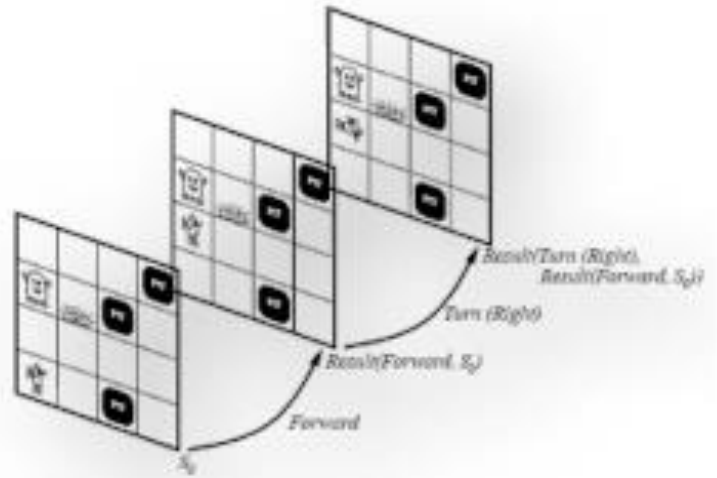
Each time slice (state) is described as a set of random variables:

X_t =Hidden (not observable) random variables

E_t=Observable random variables (with observed values **et)**

t = An identification of the time slice (we assume **discrete time** with uniform time steps)

Notation: **Xa : b** denotes a set of variables from Xa to Xb



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 **Rt**

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

– observable random variable **Ut**

The transition model

Specifies the probability distribution over the latest state variables given the previous values.

This is given by $\mathbf{P(X_t \mid X_{0:t-1})}$

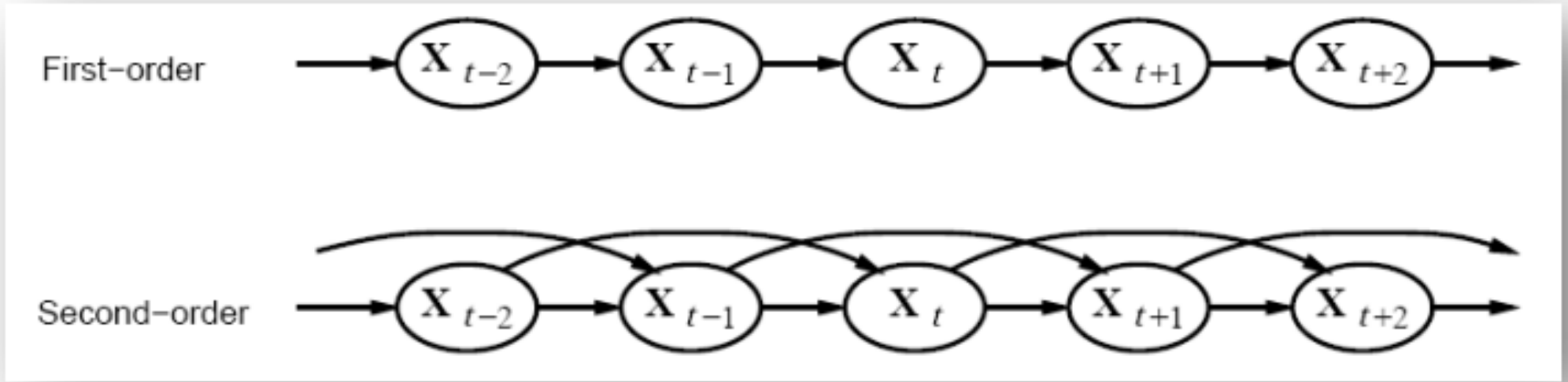
When X0:t-1 is unbounded as **t** increases, there are infinitely many possible values of **t** there is a Problem

This can be solved by Markov chains, Markov processes.

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 (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.

Sensor Markov assumption – the evidence variables depend only on the hidden state variables $\mathbf{X_t}$ from the same time.

$$P(E_t \mid X_{0:t}, E_{1:t-1}) = P(E_t \mid X_t)$$

How to Improve 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.

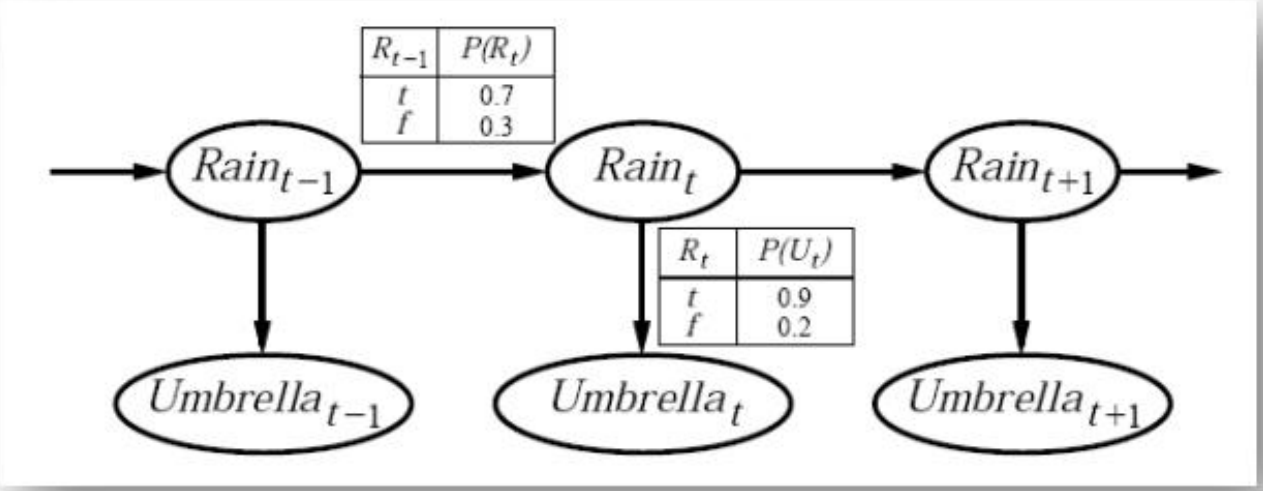
– increase the set of state variables.

- For example we could add $\mathbf{Season_t}$ to incorporate historical records or we could add $\mathbf{Temperature_t}$, $\mathbf{Humidity_t}$, $\mathbf{Pressure_t}$ 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(State Estimation):** the task of computing the posterior distribution over the most recent state, given all evidence to date $P(X_t \mid e_{1:t})$
- **Prediction:** the task of computing the posterior distribution over the future state, given all evidence to date $P(X_{t+k} \mid e_{1:t})$ for $k>0$
- **Smoothing:** the task of computing posterior distribution over a past state, given all evidence up to the present $P(X_k \mid e_{1:t})$ for $k: 0 \leq k < t$
- **Most likely explanation:** the task to find the sequence of states that is most likely generated a given sequence of observations.

  $$\textbf{argmax}_{x_{1:t}} \mathbf{P(x_{1:t} \mid e_{1:t})}$$

- Learning: EM Algorithm(Modelling)

Formulas

- Markov Transition Model for First order:

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

- Markov Transition Model for Second order:

$$\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$$

- Sensor Markov Assumption Model:

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

- The joint distribution for any t

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

  - Where Initial State: $\mathbf{P}(\mathbf{X}_0)$
  - Transition model: $\mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})$
  - Sensor Model: $\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$

Filtering

The task of computing the posterior distribution over the *most recent state,* given all evidence to date – $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$.

A useful filtering algorithm needs to maintain a current state estimate and update it, rather than going back over (**recursive estimation**):

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

How to define the function f?

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

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) && \text{(dividing up the evidence)} \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) && \text{(using Bayes' rule)} \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) && \text{(by the sensor Markov assumption)}
\end{aligned}$$

Bayes rule

sensor Markov assumption

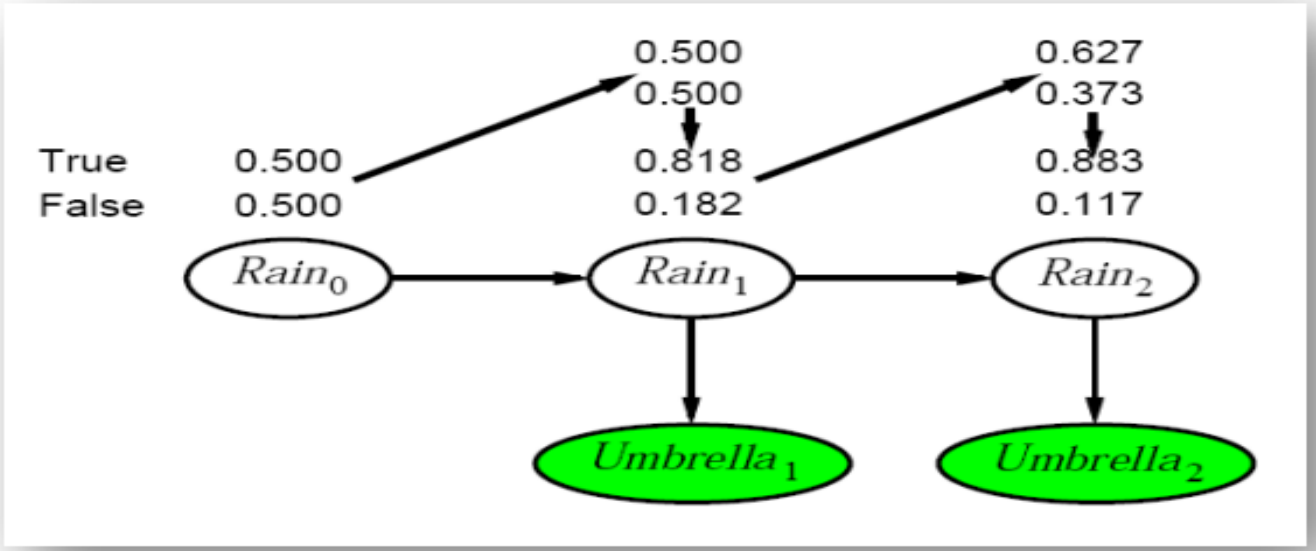
conditioning
$\mathbf{P}(\mathbf{Y}) = \Sigma_z\, \mathbf{P}(\mathbf{Y} \mid \mathbf{z})\, \mathbf{P}(\mathbf{z})$

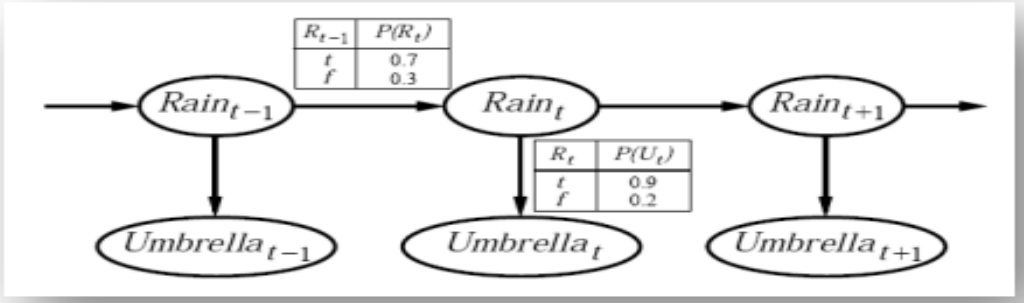
A message $\mathbf{f}_{1:t}$ is propagated forward over the sequence:

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

Filtering Example

$$\mathbf{P}(\mathbf{R}_{t+1} \mid \mathbf{u}_{1:t+1}) = \alpha\, \mathbf{P}(\mathbf{u}_{t+1} \mid \mathbf{R}_{t+1})\, \mathbf{P}(\mathbf{R}_{t+1} \mid \mathbf{u}_{1:t}) = \alpha\, \mathbf{P}(\mathbf{u}_{t+1} \mid \mathbf{R}_{t+1}) \sum_{\mathbf{r}_t} \mathbf{P}(\mathbf{R}_{t+1} \mid \mathbf{r}_t)\, \mathrm{P}(\mathbf{r}_t \mid \mathbf{u}_{1:t})$$





$\mathbf{P}(\mathbf{R}_0) = \langle 0.5, 0.5 \rangle$

$\mathbf{P}(\mathbf{R}_1)$
$= \sum_{\mathbf{r}_0} \mathbf{P}(\mathbf{R}_1 \mid \mathbf{r}_0)\, \mathrm{P}(\mathbf{r}_0)$
$= \langle 0.5, 0.5 \rangle$

$\mathbf{P}(\mathbf{R}_1 \mid \mathbf{u}_1)$
$= \alpha\, \mathbf{P}(\mathbf{u}_1 \mid \mathbf{R}_1)\, \mathbf{P}(\mathbf{R}_1)$
$= \alpha \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle$
$\approx \langle 0.818, 0.182 \rangle$

$\mathbf{P}(\mathbf{R}_2 \mid \mathbf{u}_1)$
$= \sum_{\mathbf{r}_1} \mathbf{P}(\mathbf{R}_2 \mid \mathbf{r}_1)\, \mathrm{P}(\mathbf{r}_1 \mid \mathbf{u}_1)$
$= \langle 0.7, 0.3 \rangle \times 0.818$
$+ \langle 0.3, 0.7 \rangle \times 0.182$
$\approx \langle 0.627, 0.372 \rangle$

$\mathbf{P}(\mathbf{R}_2 \mid \mathbf{u}_1, \mathbf{u}_2)$
$= \alpha\, \mathbf{P}(\mathbf{u}_2 \mid \mathbf{R}_2)\, \mathbf{P}(\mathbf{R}_2 \mid \mathbf{u}_1)$
$= \alpha \langle 0.9, 0.2 \rangle \langle 0.627, 0.372 \rangle$
$= \langle 0.883, 0.117 \rangle$

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:

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

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

Prediction

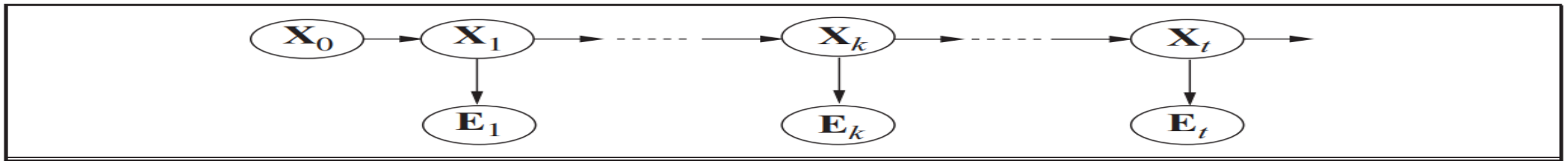
- We can use a Forward Recursion to Compute the **Likelihood** of the evidence sequence (e1:t).

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

- It is identical to that for filtering

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

Smoothing



Smoothing computes $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$, the posterior distribution of the state at some past time given a complete sequence of observations from 1 to t.

Split the computation into two parts, the evidence up to k and the evidence from k + 1 to t

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

where "$\times$" represents pointwise multiplication of vectors

$\mathbf{b}_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$, analogous to the forward message $\mathbf{f}_{1:k}$.

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

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

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

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

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

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: $0 \leq k < t$.

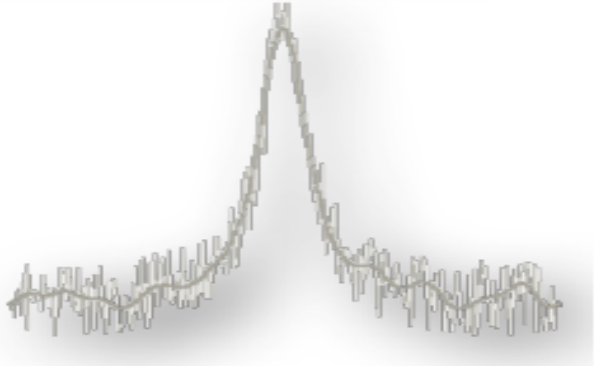
We again exploit a recursive message-passing approach, now in two parts.

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

$$\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) = \Sigma_{\mathbf{x}_{k+1}}\, \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{conditioning}$$
$$= \Sigma_{\mathbf{x}_{k+1}}\, \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{conditional independence}$$
$$= \Sigma_{\mathbf{x}_{k+1}}\, \mathbf{P}(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)$$
$$= \Sigma_{\mathbf{x}_{k+1}}\, \mathbf{P}(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{conditional independence}$$

Using the backward message-passing notation:

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



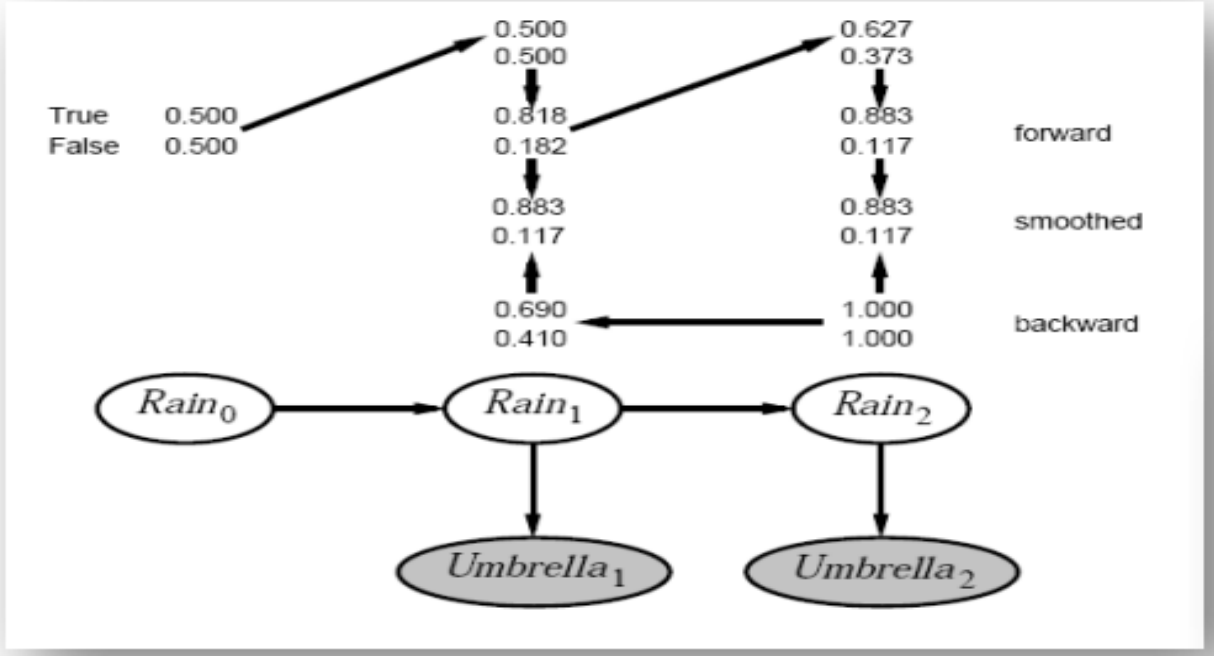
Example

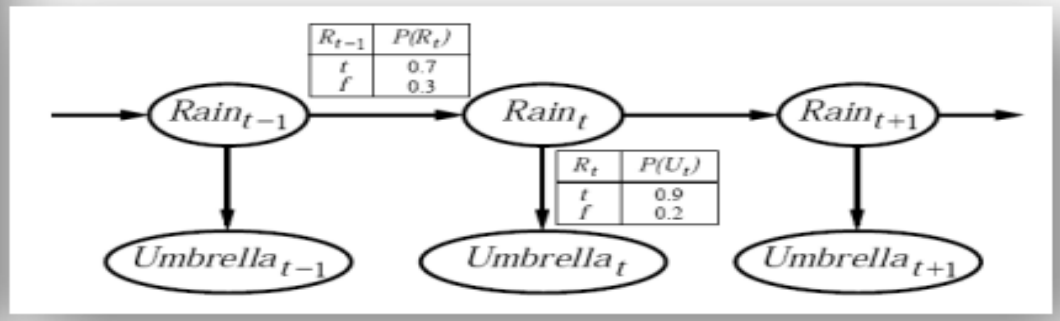
$$\mathbf{P}(R_k \mid u_{1:t+1}) = \alpha\, \mathbf{P}(R_k \mid u_{1:k})\ \mathbf{P}(u_{k+1:t} \mid R_k)$$

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

$$\mathbf{P}(\mid R_2) = \mathbf{1}$$

$$\mathbf{P}(u_2 \mid R_1) = \Sigma_{r_2}\ P(u_2 \mid r_2)\ P(\mid r_2)\ \mathbf{P}(r_2 \mid R_1)$$
$$= 0.9 \times 1 \times \langle 0.7, 0.3 \rangle + 0.2 \times 1 \times \langle 0.3, 0.7 \rangle = \langle 0.69, 0.41 \rangle$$



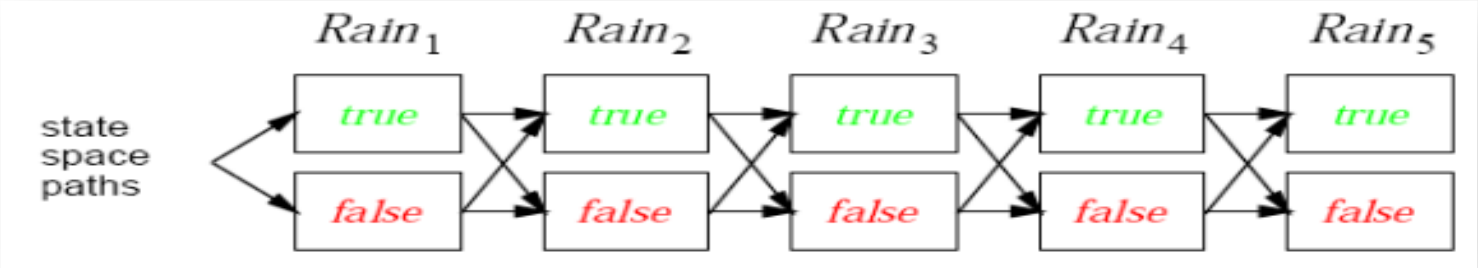


Most likely explanation/sequence

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**.

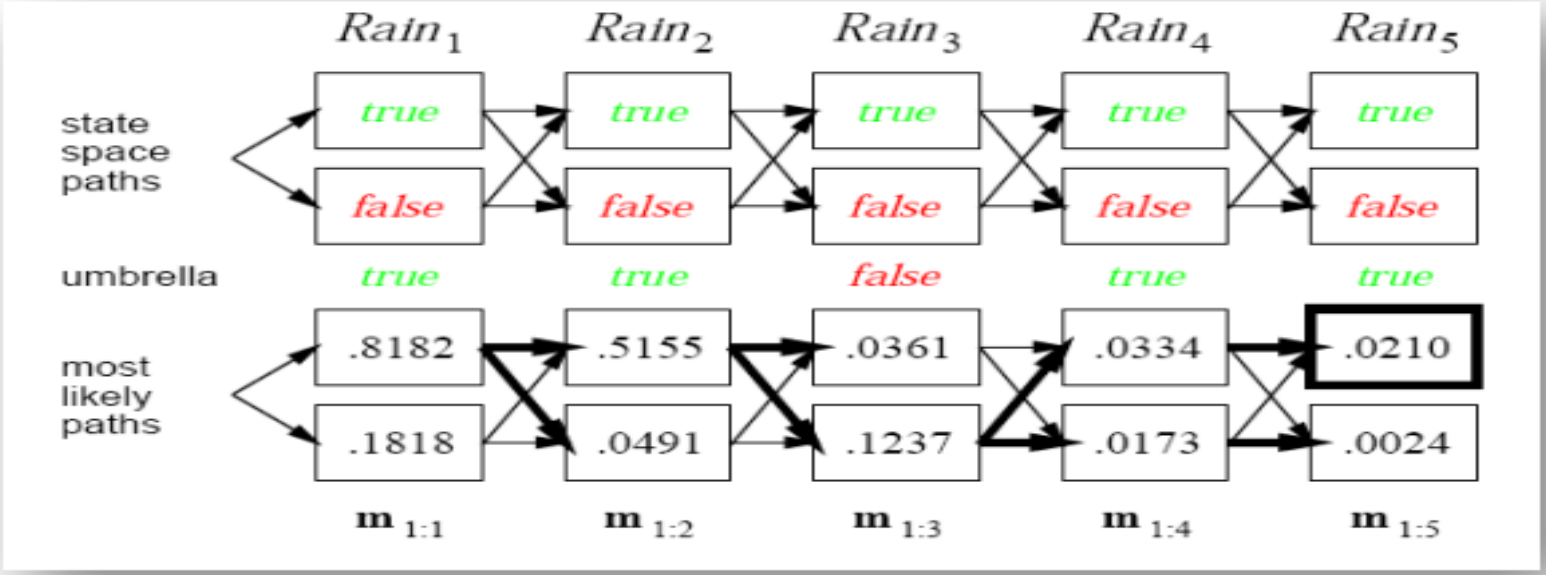
Viterbi algorithm

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.

$$\max_{x_1,\ldots,x_t} \mathbf{P}(x_1,\ldots,x_t,X_{t+1}|e_{1:t+1})$$
$$= \alpha\, \mathbf{P}(e_{t+1}|X_{t+1}) \max_{x_t} (\mathbf{P}(X_{t+1}|x_t) \max_{x_1,\ldots,x_{t-1}} \mathbf{P}(x_1,\ldots,x_t|e_{1:t}))$$

Again, we use an approach of forward message passing:

$$m_{1:t} = \max_{x_1,\ldots,x_{t-1}} \mathbf{P}(x_1,\ldots,x_{t-1},X_t|e_{1:t})$$
$$m_{1:t+1} = \mathbf{P}(e_{t+1}|X_{t+1}) \max_{x_t} (\mathbf{P}(X_{t+1}|x_t)\, m_{1:t})$$



(a) Possible state sequences for $Rain_t$ can be viewed as paths through a graph

(b) Operation of the Viterbi algorithm

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** $\mathbf{P}(X_t \mid X_{t-1})$ becomes an S×S matrix **T**, where:

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

for the umbrella world $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$

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:

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

$$\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}; \qquad \mathbf{O}_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$$

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:

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

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

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

The backward message propagation (from smoothing)

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

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

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, $\mathbf{P}(\mathbf{X}_0)$
  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:**

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

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 – $\mathbf{P}(\mathbf{X}_{t-d} \mid \mathbf{e}_{1:t})$. This is called **fixed-lag smoothing**.

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

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

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

An incremental approach:

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

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

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

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

Smoothing with a fixed time lag

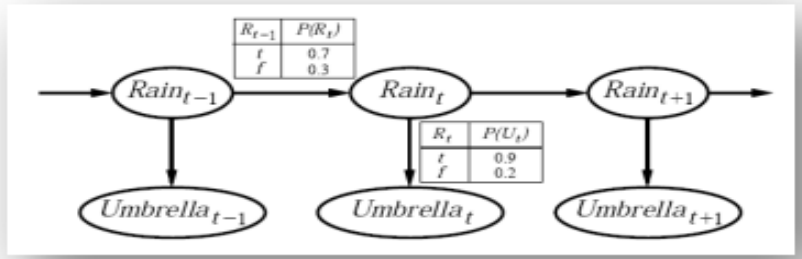
function FIXED-LAG-SMOOTHING(e_t, hmm, **d**) **returns** a distribution over $\mathbf{X}_{t-d}$
  inputs: e_t, the current evidence for time step t
    hmm, a hidden Markov model with S x S transition matrix T
    d, the length of the lag for smoothing
  static: t, the current time, initially 1
    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 $\leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{B}\mathbf{T}\mathbf{O}_t$
  else B $\leftarrow \mathbf{B}\mathbf{T}\mathbf{O}_t$
  $t \leftarrow t+1$
  if $t >$ **d then return** NORMALIZE(**f** x **B1**) **else return** null

Summary of last lecture

We know how to do **probabilistic reasoning over time**

- transition model $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$, sensor mode $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$
- Markov assumptions



Basic **inference tasks**:

- filtering: $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$
- prediction: $\mathbf{P}(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ pro $k>0$
- smoothing: $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ pro $k: 0 \leq k < t$
- most likely explanation: $\text{argmax}_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$

Hidden Markov Models (HMM)

- a special case with a single state variable and a single sensor variable
- inference tasks solved by means of matrix operations

Localization (an example of HMM)

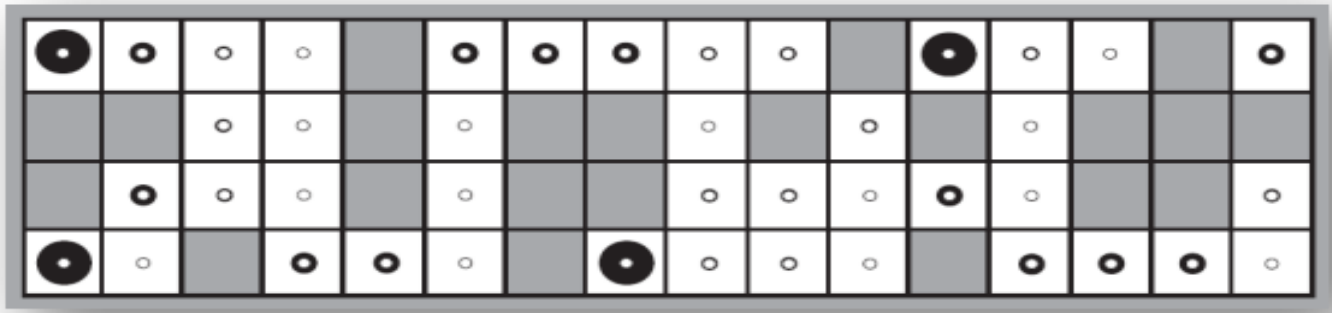
Assume a robot that moves randomly in a grid world, has a map of the world and (noisy) sensors reporting obstacles laying immediately to the north, south, east, and west. The robot needs to find its location.

A possible model:

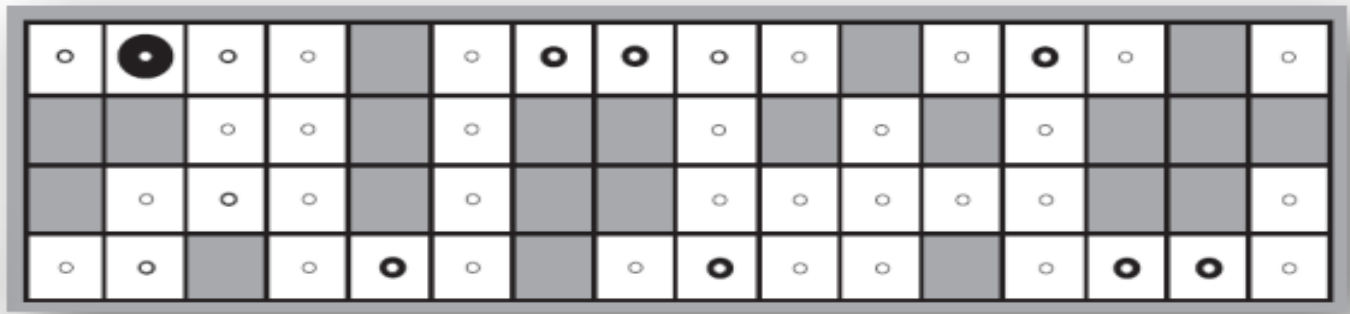
- **random variables** X_t describe robot's location at times t
  - possible values are 1,..,n for n locations
  - Nb(i) – a set of neighboring locations for location i
- **transition tables**
  - $P(X_{t+1}=j|X_t=i) = 1/ |Nb(i)|$, if $j \in Nb(i)$, otherwise 0
- **sensor variables** E_t describe observations (evidence) at times t (four sensor for four directions NSEW)
  - values indicate detection of obstacle at a given direction NSEW (16 values for all directions)
  - assume that sensor's error rate is ε
- **sensor tables**
  - $P(E_t=e_t|X_t=i) = (1-\varepsilon)^{4-d_{it}} \varepsilon^{d_{it}}$
    where d_{it} is the number of deviations of observation e_t from the true values for square i

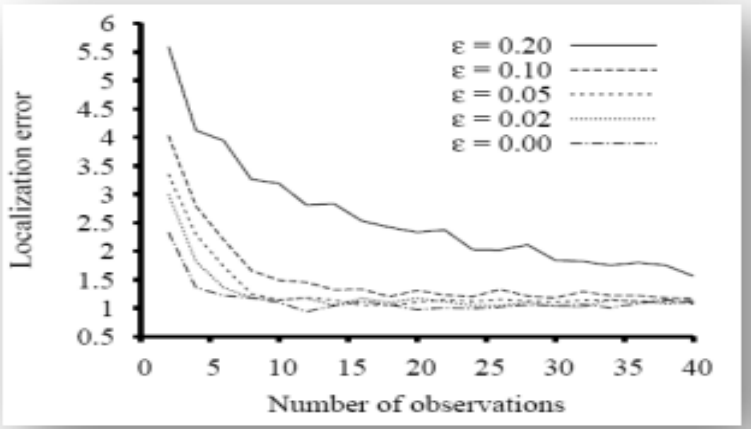
Localization (a practical example)

$\mathbf{P}(X_1 \mid E_1 = \text{NSW})$



$\mathbf{P}(X_2 \mid E_1 = \text{NSW}, E_2 = \text{NS})$





The localization error defined as the Manhattan distance from the true location

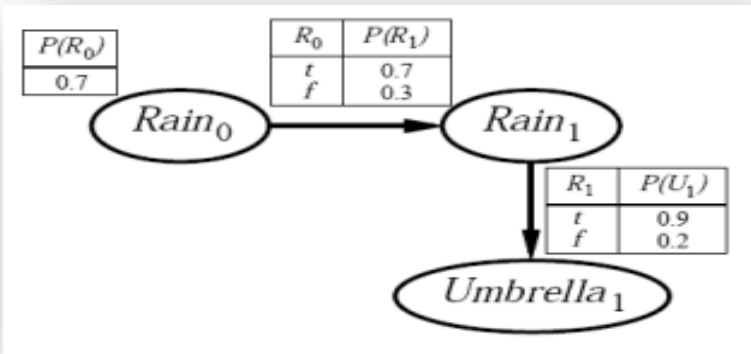
Dynamic Bayesian networks

Dynamic Bayesian network (DBN) is a Bayesian network that represents a temporal probability model.

the variables and links exactly replicated from slice to slice

It is enough to describe one slice.

- prior distribution $\mathbf{P(X_0)}$
- transition model $\mathbf{P(X_1 \mid X_0)}$
- sensor model $\mathbf{P(E_1 \mid X_1)}$



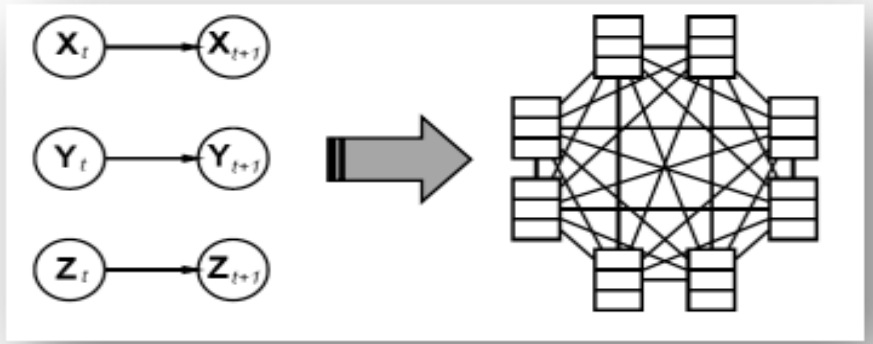
Each variable has parents either at the same slice or in the previous slice (Markov assumption).

DBN vs HMM

A hidden Markov model is a special case of a dynamic Bayesian network.

Similarly, a dynamic Bayesian network can be encoded as a hidden Markov model

one random variable in HMM whose values are n-tuples of values of state variables in DBN



What is the difference?

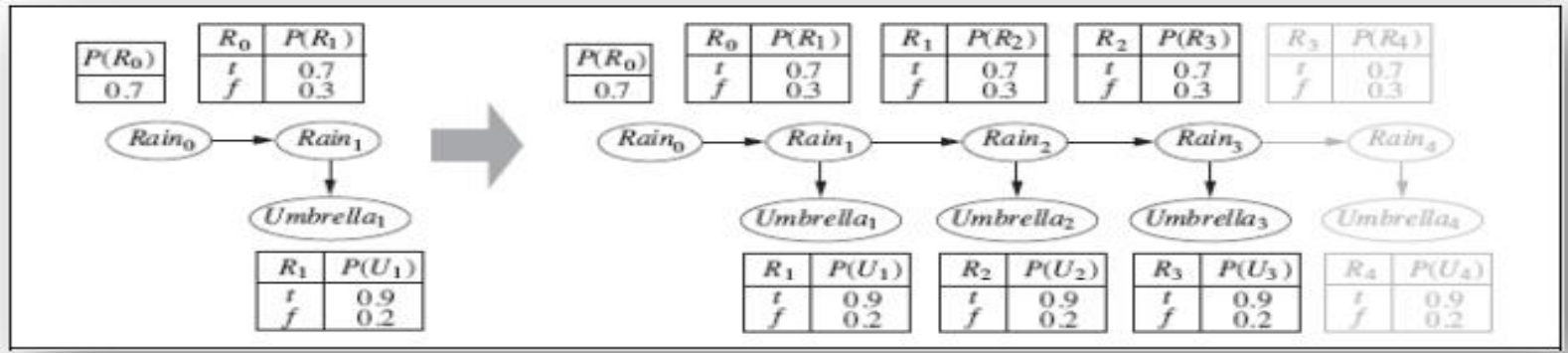
The relationship between DBN and HMM is roughly analogous to the relationship between ordinary Bayesian networks and full tabulated joint distribution.

- DBN with 20 Boolean state variables, each of which has three parents
  - the transition model has $20 \times 2^3 = 160$ probabilities
- Hidden Markov model has one random variable with 2^{20} values
  - the transition model has $2^{20} \times 2^{20} \approx 10^{12}$ probabilities
  - HMM requires much more space and inference is much more expensive

Exact inference in DBNs

Dynamic Bayesian networks are Bayesian networks and we already have algorithms for inference in Bayesian networks.

We can construct the full Bayesian network representation of a DBN by replicating slices to accommodate the observations (**unrolling**).



A naive application of unrolling would not be particularly efficient as the inference time will increase with new observations.

We can use an incremental approach that keeps in memory only two slices (via summing out the variables from previous slices).

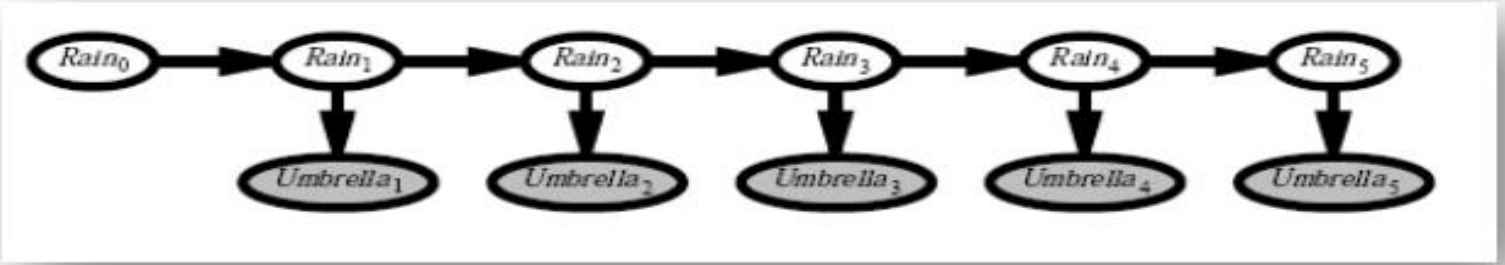
- This is similar to **variable elimination** in Bayesian networks.
- The bad news are that "constant" space to represent the **largest factor will be exponential** in the number of state variables.
  $O(d^{n+k})$, where n is the number of variables, d is the domain size, k is the maximum number of parents of any state variable

Approximate inference in DBNs

We can try approximate inference methods (**likelihood weighting** is most easily adapted to the DBN context).

We sample non-evidence nodes of the network in topological order, weighting each sample by the likelihood in accords to the observed evidence variables.

- each sample must go through a full network
- we can simply run all N samples together through the network (N samples are similar to the forward message, the samples themselves are approximate representation of the current state distribution)



There is a **problem**!

Samples are generated completely independently of the evidence!

- This behaviors decreases accuracy of the method as the weight of samples is usually very small and we need much larger number of samples
  - We need to **increase the number of samples exponentially** with t.
  - If a constant number of samples is used them the method accuracy is goes down significantly.

Particle filtering

We need to focus the set of samples on the high-probability regions of the sample space.

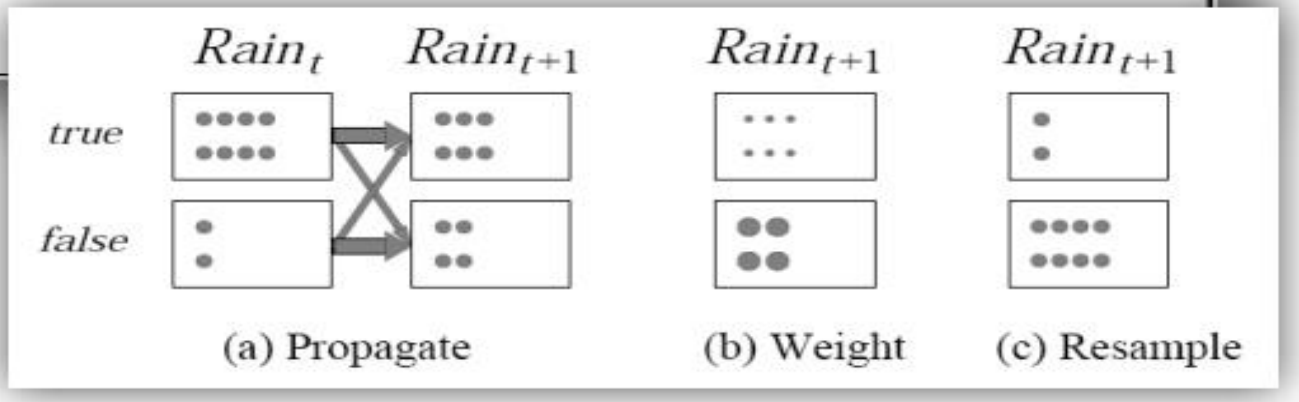
Particle filtering is doing exactly that by resampling the population of samples.

- a population of N samples is created by **sampling from the prior distribution** $\mathbf{P}(\mathbf{X}_0)$
- each sample is **propagated forward** by sampling the next state value $\mathbf{x}_{t+1}$ given the current value $\mathbf{x}_t$ for the sample, based on the transition model $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$
- each sample is **weighted** by the likelihood it assigns to the new evidence, $\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1})$
- the population is **resampled** to generate a new population of N samples
  - each sample is selected from the current population such that the probability of selection is proportional to its weight (the new samples are unweighted)

Particle filtering algorithm

function PARTICLE-FILTERING(e,N, dbn) returns a set of samples for the next time step
  inputs: e, the new incoming evidence
    N, the number of samples to be maintained
    dbn, a DBN with prior $\mathbf{P}(\mathbf{X}_0)$, transition model $\mathbf{P}(\mathbf{X}_1|\mathbf{X}_0)$, and sensor model $\mathbf{P}(\mathbf{E}_1|\mathbf{X}_1)$
  static: S, a vector of samples of size N, initially generated from $\mathbf{P}(\mathbf{X}_0)$
  local variables: W, a vector of weights of size N

  for i = 1 to N **do**
    $S[i] \leftarrow$ sample from $\mathbf{P}(\mathbf{X}_1|\mathbf{X}_0 = S[\mathrm{i}])$
    $W[i] \leftarrow \mathbf{P}(\mathbf{e}|\mathbf{X}_1 = S[i])$
  $S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S, W)
  return S



Particle filtering (soundness)

Assume that samples at time t are consistent with probability distribution

$$N(\mathbf{x}_t \mid \mathbf{e}_{1:t}) / N = P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$

After **propagation** to time t+1, the number of samples is

$$N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t}) = \Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) N(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$

The total **weight** of the samples in state $\mathbf{x}_{t+1}$ after evidence is

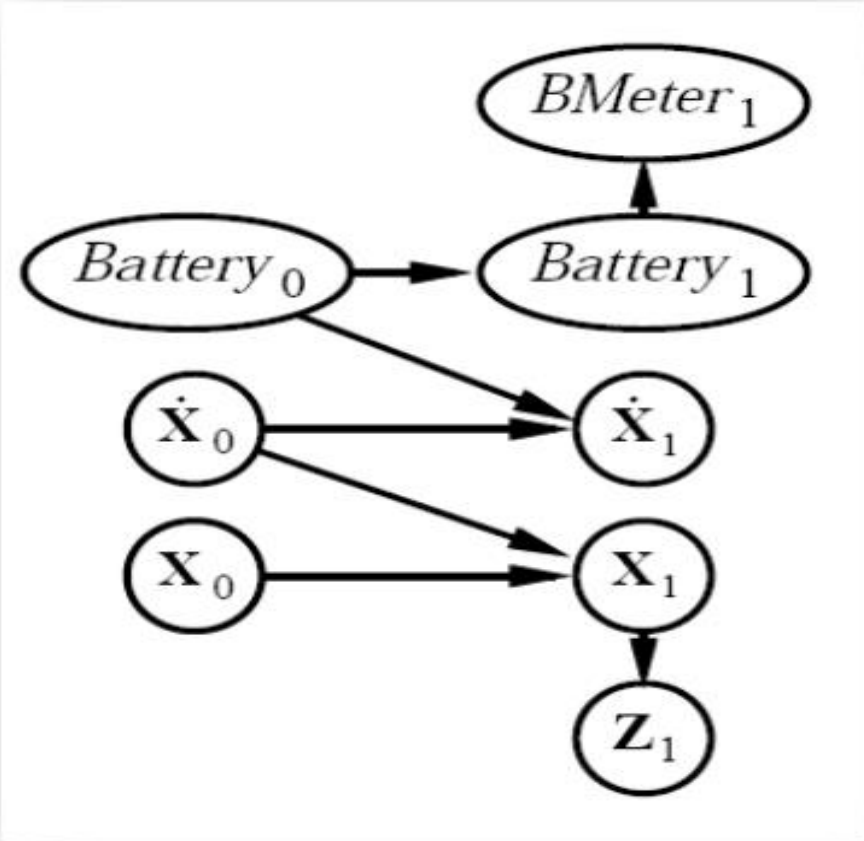
$$W(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1}) = P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t})$$

After **resampling** we will get

$$N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1}) / N = \alpha W(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1})$$
$$= \alpha P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) N(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t})$$
$$= \alpha P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) \Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) N(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$
$$= \alpha' P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1}) \Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$
$$= P(\mathbf{x}_{t+1} \mid \mathbf{e}_{1:t+1})$$

Complex example (DBN)

Let us consider a battery-powered mobile robot and model its behavior using a DBN.



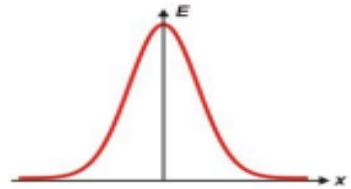
- we need state variables modelling **position** of the robot $\mathbf{X}_t = (X_t, Y_t)$, its **velocity** $\mathbf{X}_t = (X_t, Y_t)$, and actual **battery level**
  - the position at the next time step depends on the current position and velocity
  - the velocity at the next steps depends on the current velocity and the state of battery
- we assume some method of **measuring position** (a fixed camera or onboard GPS) $\mathbf{Z}_t$
- similarly we assume a sensor **measuring battery level** $\mathbf{BMeter}_t$

Sensor failures

A fully accurate sensor uses a sensor model with probabilities 1.0 "along the diagonal" and probabilities 0.0 elsewhere.

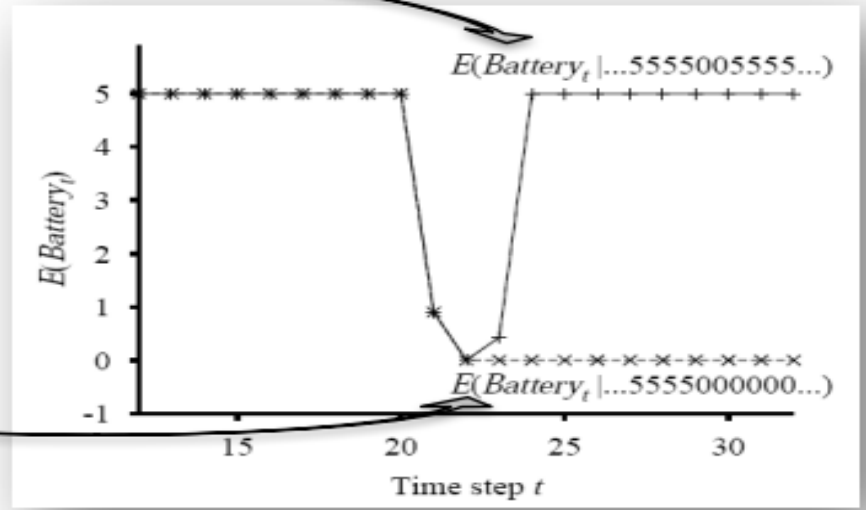
In reality, noise always creeps into measurements. A Gaussian distribution with small variance might be used (for continuous measurements)

– **Gaussian error model**



Real sensors fail. When a sensor fails, it simply sends nonsense (but does not say that it that way). Then the Gaussian error model does not work as we need.

- after two wrong measurements we are almost certain that the battery is empty
- if the failure disappears then the battery level quickly returns to 5, as if by magic (**transient failures**)
  - in the meantime, the system may do some wrong decisions (shut down)
- if the wrong measures are still coming then the model is certain that the batters is empty (though a chance of such a sudden change is small) (**persistent failure**)



Sensor transient failure model

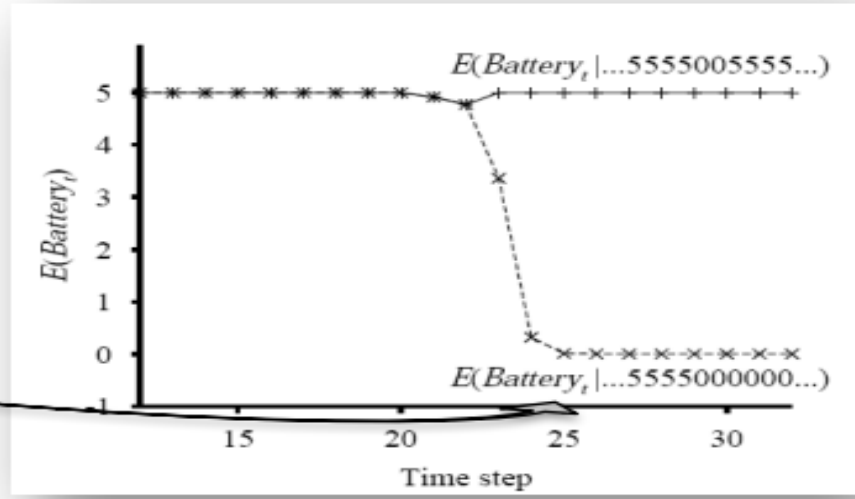
The simplest kind of failure model allows a certain probability that the sensor will return some completely incorrect value, regardless of the true state of the world.

$P(BMeter_t=0 \mid Battery_t=5) = 0.03$

Let us call this the **transient failure model**.

The model brings some "inertia" that helps to overcome temporary blips in the meter reading.

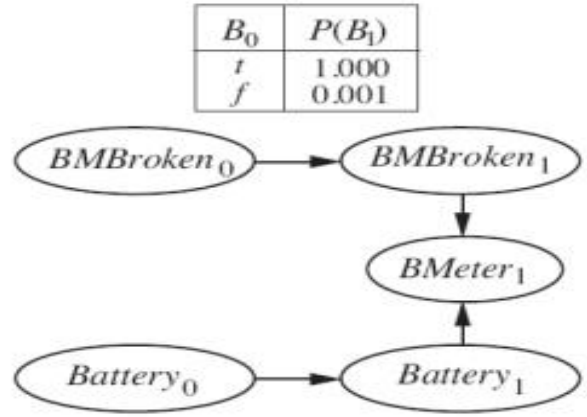
However, after more wrong readings the robot gradually coming to believe that the its battery is empty while in fact it may by that the meter has failed!

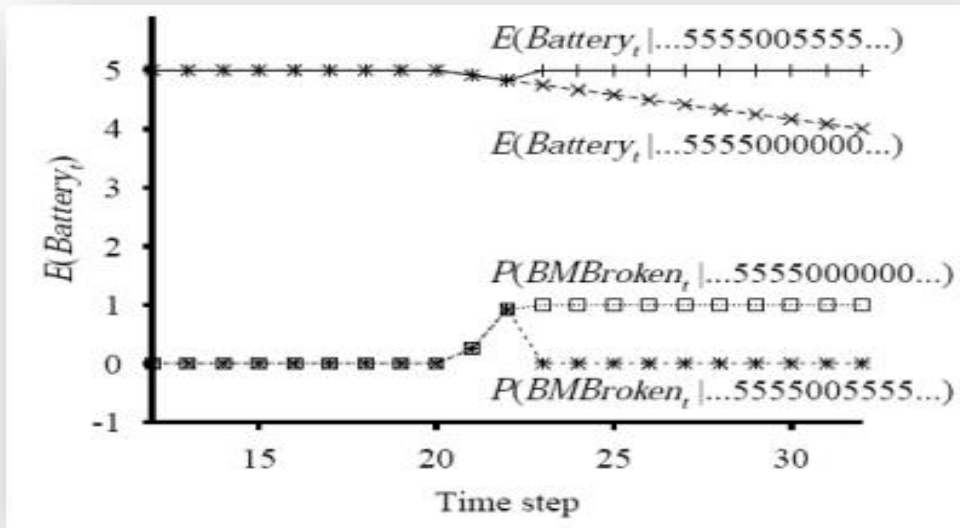


Sensor persistent failure model

To model persistent failures we use additional state variable, **BMBroken**, that describes the status of the battery meter.

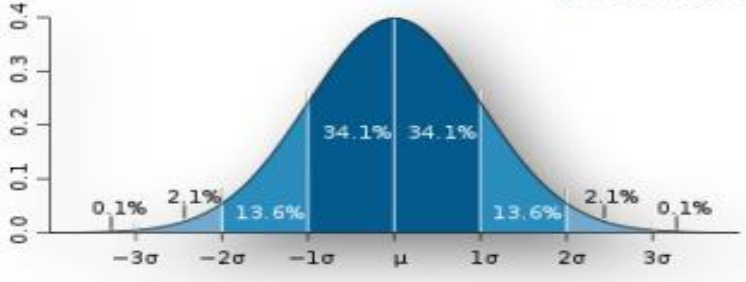
- The **persistence arc** has a CPT that gives a small probability of failure in any given time step (0.001), but specifies that the sensor stays broken once it breaks.
- When the sensor is OK, the sensor model is identical to the transient failure model.
- Once the sensor is known to be broken, the robot can only assume that its battery discharges at the "normal" rate.





So far we assumed discrete random variables so the probability distribution can be captured by tables.

- **How can we handle continuous variables?**
  - **discretization**
    - dividing up the possible values into a fixed set of intervals
    - often results in a considerably loss of accuracy and very large CPTs
  - we can also use **standard families of probability density functions** that are specified by a finite number of parameters
    - Gaussian (or normal) distribution
      - described by the mean μ and the variance σ^2 as parameters



$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)},$$

Continuous variables (conditional probability)

How are the conditional probability tables specified in hybrid Bayesian networks?

- dependence of **continuous variable on the continuous variable** can be described using linear Gaussian distribution
  - » the mean values varies linearly with the value of the parent
  - » standard deviation is fixed

$$P(Cost = c | Harvest = h, Subsidy? = true) = N(a_t h + b_t, \sigma_t)(c) = \frac{1}{\sigma_t \sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right)$$

- dependence of **continuous variable on the discrete variable**
  - » for each value of the discrete variable we specify parameters of standard distribution
- Dependence of **discrete variables on the continuous variable**
  - » "soft" threshold function

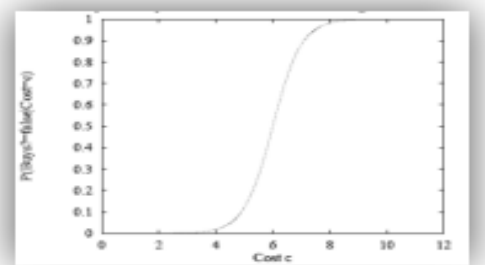
probit (probability unit) distribution

$$\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$$
$$P(Buys? = true \mid Cost = c) = \Phi((-c + \mu)/\sigma)$$

logit (logistic function) distribution

$$P(Buys? = true \mid Cost = c) = \frac{1}{1 + exp(-2\frac{-c+\mu}{\sigma})}$$

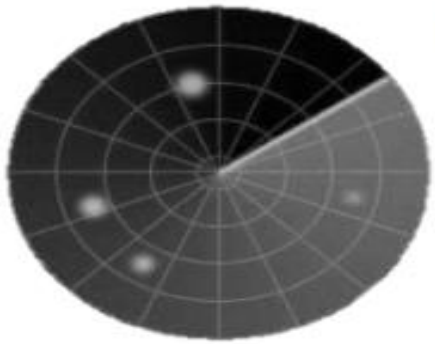
  - » probit is often a better fit to real situations, (the underlying decision process has a hard threshold, but the precise location of th threshold is subject to random Gaussian noise)
  - » logit has much longer "tails", sometimes easier to deal with mathematically

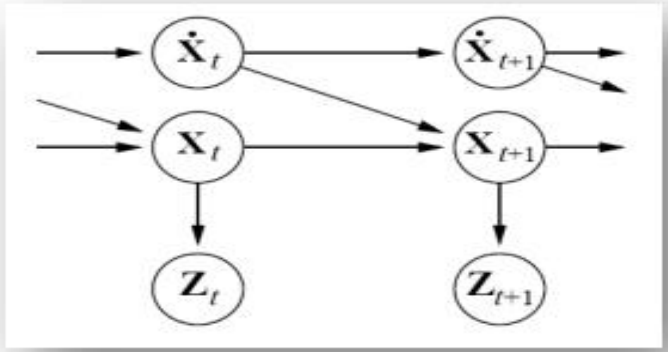


Assume a problem of detecting actual position of an aircraft based on observations on radar screen. We will model the problem using a dynamic Bayesian network.

- random state variables describe actual **location** and **speed** of the aircraft
  - the next location depends on the previous location and speed and can be modelled using linear Gaussian distribution

    $P(X_{t+\Delta}=x_{t+\Delta} \mid X_t=x_t , X_t=x_t) = N(x_t+x_t\Delta,\sigma^2)\ (x_{t+\Delta})$
- we observe location Z_t
  - again, we can use Gaussian distribution in the sensor model





Kalman filters (properties)

Gaussian distribution has some nice properties when solving the tasks of filtering, prediction, and smoothing.

- If the distribution $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ is Gaussian and the transition model $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$ is linear Gaussian then $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ is also a Gaussian distribution.

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

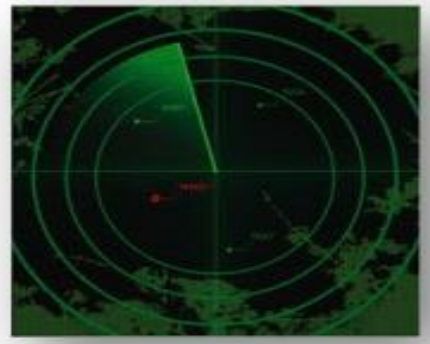
- If $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ is Gaussian and the sensor model $\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})$ is linear Gaussian then $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1})$ is also a Gaussian distribution.

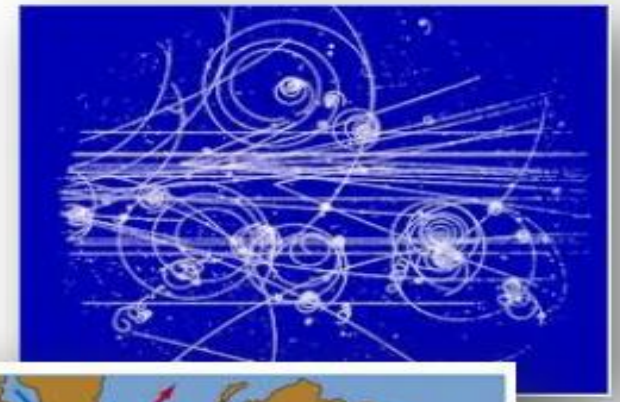
$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

- To solve tasks we can use the message passing technique.

Kalman filters (applications)

- Classical application of the Kalman filter is **tracking movements** of aircrafts and missiles using radars.
- Kalman filters are used to **reconstruct trajectories** of particles in physics and monitoring ocean currents.
- The range of applications is much larger than tracking of motion: **any system characterized by continuous state variables and noisy measurements** will do (pulp mills, chemical plants, nuclear reactors, plant ecosystems, and national economies).





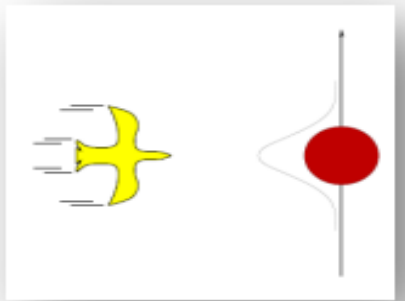
Kalman filters (non-linearities)

Kalman filters assumes linear Gaussian transition and sensor models.

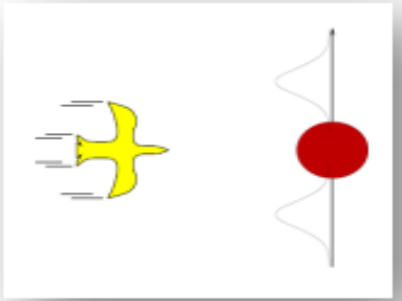
What if the model is non-linear?

Example: assume a bird heading at high speed straight for a tree trunk

Kalman filter



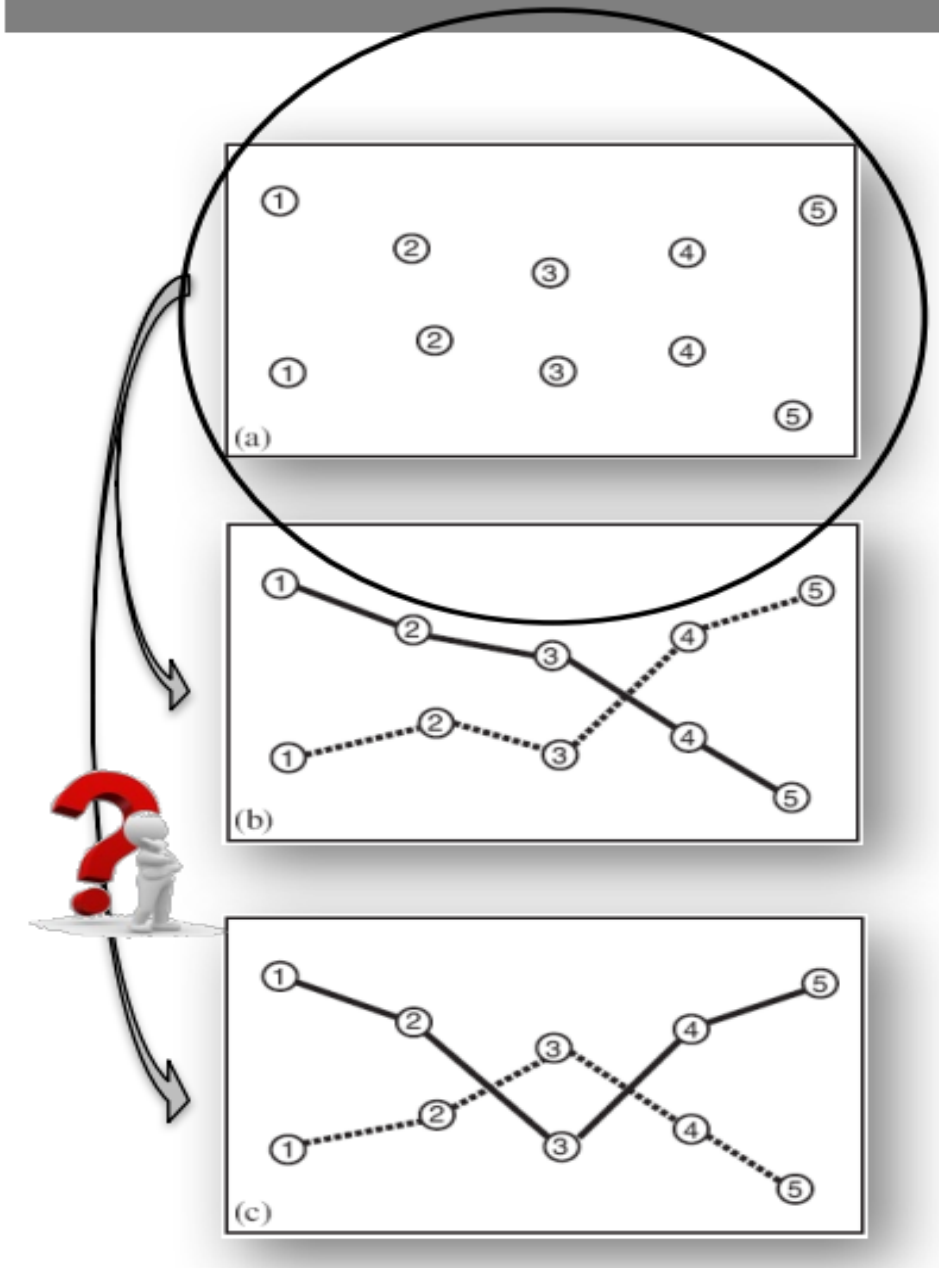
a more realistic model



A standard solution: **switching Kalman filter**

- **multiple Kalman filters** run in parallel, each using a different model of the system (one for straight flight, one for sharp left turns , and one for sharp right turns)
- a **weighted sum of predictions** is used, where the weight depends om how well each filter fits the current data
- this is a special case of DBN obtained by **adding a discrete "maneuver" state variable** to the network

Keeping track of many objects



We have considered state estimation problems involving a single object.

What do happen if **two or more objects** generate the observations?

Additional problem:

- Uncertainty about which object generated which observation!

Keeping track of many objects (approaches)

Data association problem:
the problem of associating observation data with the objects that generated them

Exact reasoning means summing out the variables over all possible assignments of objects to observations

- for a single time slice and n objects it means n! mappings
- for T time slices we have $(n!)^T$ mappings

Many different **approximate methods** are used.

- choose a single "best" assignment at each time step
  - **nearest-neighbor filter** (chooses the closest pairing of predicted position and observation)
  - a better approach is to choose the assignment that **maximizes the joint probability** of the current observations given the predicted positions (the Hungarian algorithm)
- Any method that commits to as single best assignment at each time step fails miserably under more difficult conditions (the prediction on the next step may be significantly wrong)
  - **particle filtering** (maintains a large collection of possible current assignments)
  - **MCMC** (explores the space of possible current assignments)

Keeping track of many objects (in reality)

Real applications of data association are typically much more complicated.

- **false alarm (clutter)**
  (observations not caused by real objects)
- **detection failures**
  (no observation is reported for a real object)
- **new and disappearing objects**
  (new objects arrive and old ones disappear)