PROBABILISTIC REASONING

Outline

- Representing Knowledge in an uncertain domain
- Semantics of Bayesian networks
- Efficient representation of conditional distributions
- Exact/Approximate inference in Bayesian networks
- Other approaches to uncertain reasoning

Based on the textbook by Stuart Russell & Peter Norvig:

Artificial Intelligence, A Modern Approach (2nd Edition)

Chapter 14; excluding Section 14.6

1. REPRESENTING KNOWLEDGE IN AN UNCERTAIN DOMAIN

- Conditional independence relations provide means to simplify probabilistic representations of the world.
- A **Bayesian network** is a data structure representing the dependencies among variables $X_1,\ldots,X_n$ of a given domain.
- As a result, a compact specification of the full joint probability distribution $\mathbf{P}(X_1,\ldots,X_n)$ is obtained.
- Bayesian networks are also called *belief networks*, *probabilistic networks*, *causal networks*, or *knowledge maps*.

Bayesian Networks: Syntax

Definition. A belief network is a *directed acyclic graph* (DAG) $G = \langle\{X_1,\ldots,X_n\},E\rangle$ where

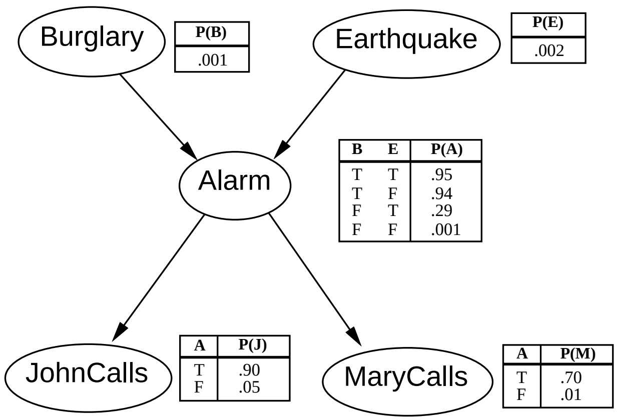
1. *nodes* $X_1,\ldots,X_n$ are discrete/continuous random variables,
2. the set of *arrows* (or links)
$$E \subseteq \{X_1,\ldots,X_n\}^2 = \{\langle X_i,X_j\rangle \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n\},$$
3. an arrow $\langle X,Y\rangle \in E$ of G represents a *direct influence* relationship between the variables X and Y, and
4. each node X is assigned a completely specified probability distribution $\mathbf{P}(X|\mathrm{Parents}(X))$ where
$$\mathrm{Parents}(X) = \{Y \mid \langle Y,X\rangle \in E\}.$$

Example. Consider a network based on five Boolean random variables:

1. *Burglary* = "a burglar enters our home".
2. *Earthquake* = "an earthquake occurs".
3. *Alarm* = "our burglar alarm goes off".
   The alarm is fairly reliable at detecting a burglary, but may occasionally respond to minor earthquakes.
4. *JohnCalls* = "Our neighbor John calls and reports an alarm."
   He always calls when he hears the alarm, but sometimes confuses telephone ringing with the alarm.
5. *MaryCalls* = "Our neighbor Mary calls and reports an alarm ".
   She likes loud music and sometimes misses the alarm altogether.

Shorthands B, E, A, J, and M are also introduced for these variables.

- The relationships of the variables are given as a Bayesian network.
- The probability distributions $\mathbf{P}(X \mid \text{Parents}(X))$ associated with variables X are given as *conditional probability tables* (CPTs).



2. SEMANTICS OF BAYESIAN NETWORKS

- A Bayesian network for the random variables $X_1, \ldots, X_n$ is a representation of the joint probability distribution $\mathbf{P}(X_1, \ldots, X_n)$.
- As before, a shorthand x_i is used for the atomic event $X_i = x_i$.
- Arrows encode conditional independence relations and therefore the probabilities of atomic events are determined by

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

where $\text{Parents}(x_i)$ refers to the assignments of $Y \in \text{Parents}(X_i)$.

Example. Let us compute the probability of $j \wedge m \wedge a \wedge \neg b \wedge \neg e$:

$$\begin{aligned} & P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) \\ = \; & P(j|a)P(m|a)P(a|\neg b \wedge \neg e)P(\neg b)P(\neg e) \\ = \; & 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 = 0.00063 \,. \end{aligned}$$

Conditional Independence Revisited

Definition. Let $P(\psi) > 0$. Sentences ϕ_1 and ϕ_2 are *conditionally independent given* ψ $\iff$ $P(\phi_1 \wedge \phi_2 \mid \psi) = P(\phi_1 \mid \psi)P(\phi_2 \mid \psi)$.

Proposition. If $P(\psi) > 0$, $P(\phi_1 \wedge \psi) > 0$, and $P(\phi_2 \wedge \psi) > 0$, then ϕ_1 and ϕ_2 are conditionally independent given ψ $\iff$ $P(\phi_1 \mid \phi_2 \wedge \psi) = P(\phi_1 \mid \psi)$ and $P(\phi_2 \mid \phi_1 \wedge \psi) = P(\phi_2 \mid \psi)$ hold.

Proof. For the former equation, we note that

$$\begin{aligned} & P(\phi_1 \wedge \phi_2 \mid \psi) = P(\phi_1 \mid \psi)P(\phi_2 \mid \psi) \\ \iff \; & \tfrac{P(\phi_1 \wedge \phi_2 \wedge \psi)}{P(\psi)} = \tfrac{P(\phi_1 \wedge \psi)}{P(\psi)} \cdot \tfrac{P(\phi_2 \wedge \psi)}{P(\psi)} \\ \iff \; & P(\phi_1 \wedge \phi_2 \wedge \psi)P(\psi) = P(\phi_1 \wedge \psi)P(\phi_2 \wedge \psi) \\ \iff \; & P(\phi_1 \mid \phi_2 \wedge \psi) = \tfrac{P(\phi_1 \wedge \phi_2 \wedge \psi)}{P(\phi_2 \wedge \psi)} = \tfrac{P(\phi_1 \wedge \psi)}{P(\psi)} = P(\phi_1 \mid \psi). \end{aligned}$$

A Method for Constructing Bayesian Networks

- In a Bayesian network $G = \langle \{X_1, \ldots, X_n\}, E \rangle$, a node $X_j \neq X_i$ is a *predecessor* of X_i $\iff$ there are nodes $Y_1, \ldots, Y_m$ such that $Y_1 = X_j$, $Y_m = X_i$, and $\forall j \in \{1, \ldots, m-1\}$: $\langle Y_j, Y_{j+1} \rangle \in E$.
- Because G is a DAG, we may assume that the nodes $X_1, \ldots, X_n$ are ordered so that the predecessors of X_i are among $X_1, \ldots, X_{i-1}$. Thus also $\text{Parents}(X_i) \subseteq \{X_1, \ldots, X_{i-1}\}$.
- By the definition of conditional probability, we have that

$$\begin{aligned} & P(x_1, \ldots, x_n) = \\ & P(x_n \mid x_{n-1}, \ldots, x_1)P(x_{n-1}, \ldots, x_1) = \\ & P(x_n \mid x_{n-1}, \ldots, x_1)P(x_{n-1} \mid x_{n-2}, \ldots, x_1) \cdots P(x_2 \mid x_1)P(x_1) = \\ & \textstyle\prod_{i=1}^{n} P(x_i \mid x_{i-1}, \ldots, x_1). \end{aligned}$$

- A Bayesian network is a correct representation if each variable X is conditionally independent of its predecessors Y given $\text{Parents}(X)$.
- Under the assumptions on conditional independence and node ordering, it can be established that

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

- The choice of $\text{Parents}(X)$ for a random variable X affects how far conditional independence assumptions can be applied.
- $\text{Parents}(X)$ should contain all variables that directly influence X.

Example. Only *Alarm* directly influences *MaryCalls*. Given *Alarm*, *MaryCalls* is conditionally independent of the other variables:

$$\mathbf{P}(\mathit{MaryCalls} \mid \mathit{JohnCalls}, \mathit{Alarm}, \mathit{Earthquake}, \mathit{Burglary})$$
$$= \mathbf{P}(\mathit{MaryCalls} \mid \mathit{Alarm}).$$

On Compactness and Node Ordering

- A Bayesian network can be a compact representation of the joint probability distribution (*locally structured* or *sparse* system).
- If each Boolean variable directly influences at most k other, then only $n2^k$ probabilities have to be specified (instead of 2^n).

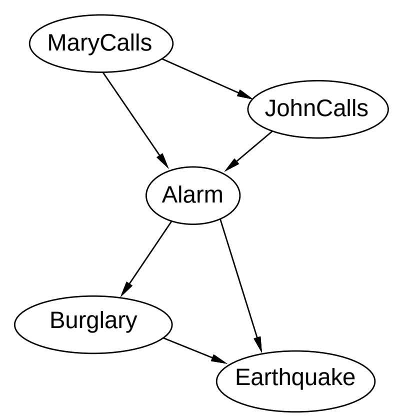
  Example. When $n = 30$ and $k = 5$, we would have to specify $n2^k = 960$ and $2^n = 1073741824$ probabilities, respectively.
- A clear **trade-off:** number of arrows (accuracy of probabilities) *versus* cost of specifying extra information (extending CPTs).
- Choosing a good node ordering is a non-trivial task.
- Heuristics: the *root causes* of the domain should be added first, then the variables influenced by them, and so forth.

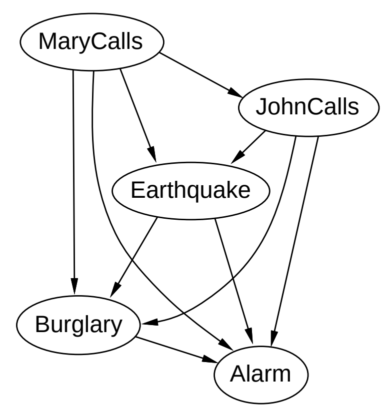
Example. Let us reconstruct the Bayesian network for the alarm domain using a different node ordering:

$$\mathit{MaryCalls}, \mathit{JohnCalls}, \mathit{Alarm}, \mathit{Burglary}, \mathit{Earthquake}$$

1. As the first node, *MaryCalls* gets no parents.
2. When *JohnCalls* is added, *MaryCalls* becomes a parent of *JohnCalls*, as $\mathbf{P}(\mathit{JohnCalls} \mid \mathit{MaryCalls}) \neq \mathbf{P}(\mathit{JohnCalls})$.
3. Similarly, *Alarm* depends on both *MaryCalls* and *JohnCalls*.
4. Since $\mathbf{P}(\mathit{Burglary} \mid \mathit{Alarm}, \mathit{JohnCalls}, \mathit{MaryCalls}) = \mathbf{P}(\mathit{Burglary} \mid \mathit{Alarm})$, the only parent of *Burglary* is *Alarm*.
5. Nodes *Burglary* and *Alarm* become parents of *Earthquake*, as $\mathbf{P}(\mathit{Earthquake} \mid \mathit{Burglary}, \mathit{Alarm}, \mathit{JohnCalls}, \mathit{MaryCalls}) = \mathbf{P}(\mathit{Earthquake} \mid \mathit{Burglary}, \mathit{Alarm})$.

- The resulting Bayesian network is given below on the left:



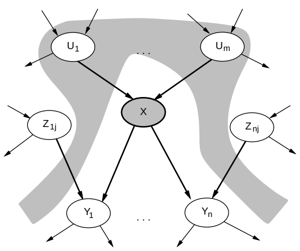


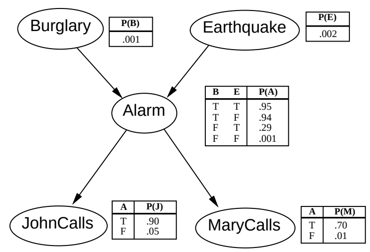
- The one on the right is obtained with another ordering and it as complex (31 probabilities) as the full joint distribution!

Conditional Independence Relations

Two (equivalent) topolocial criteria can be utilized:

1. A node X is conditionally independent of its **non-descendants** (e.g., Z_{ij}s below), given its parents (i.e., U_is below).

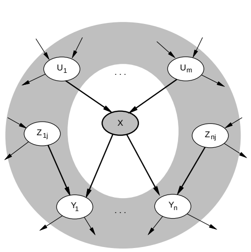


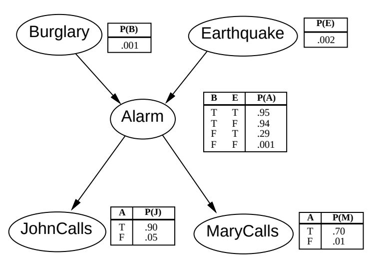


Example. In the burglary example, one may conclude that:

JohnCalls is independent of *Burglary* and *Earthquake* given *Alarm*.

2. A node X is conditionally independent of all other nodes in the network, given its **Markov blanket** $mb(X)$, i.e., its parents, children, and children's parents.





Example. *Burglary* is independent of *JohnCalls* and *MaryCalls* given *Alarm* and *Earthquake*.

There is yet another criterion called **d-separation**, but unlike the first edition of the textbook it is not covered by the second.

3. EFFICIENT REPRESENTATION OF CONDITIONAL DISTRIBUTIONS

- Specifying conditional probability tables means often a lot of work.
- To ease this process, some **canonical distributions** such as *deterministic* and *noisy* logical relationships have been proposed.
- When using a canonical distribution it is often enough to supply certain parameters rather than a complete CPT.
- There are also canonical continuous distributions such as *Gaussian distributions* and *probit/logit distributions*.

Deterministic Nodes

- In the deterministic case, there is no uncertainty and the value of X is obtained as a (logical) function from those of Parents(X).
- Deterministic nodes can also encode other fixed numerical functions depending on the variables involved.

Example. Define $NorthAmerican \leftrightarrow Canadian \vee US \vee Mexican$. This corresponds to specifying a CPT as follows:

Canadian	*US*	*Mexican*	*NorthAmerican*
F	F	F	0.0
T	F	F	1.0
⋮	⋮	⋮	⋮

Noisy Logical Relationships

- Noisy logical relationships add some uncertainty to the scenario.
- A **noisy OR** relationship comprises the following principles:
  1. Each cause has an independent chance of causing the effect.
  2. All possible causes are listed.
  3. Whatever inhibits some cause from causing an effect is independent of whatever inhibits other causes from causing the effect. Inhibitors are summarized as **noise parameters**.
- A noisy OR relationship in which a variable depends on k parents can be described using k parameters.

  In contrast to this, 2^k entries are needed if a full CPT is specified.

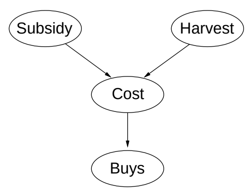
Example. Let us consider a medical domain including the variables *Fever* (a symptom), *Cold*, *Flu*, and *Malaria* (diseases). Using noise parameters $P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6$,
$P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2$, and
$P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$, we get the following CPT:

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	*F*	*F*	0.0	1.0
F	*F*	*T*	0.9	0.1
F	*T*	*F*	0.8	0.2
F	*T*	*T*	0.98	$0.02 = 0.2 \times 0.1$
T	*F*	*F*	0.4	0.6
T	*F*	*T*	0.94	$0.06 = 0.6 \times 0.1$
T	*T*	*F*	0.88	$0.12 = 0.6 \times 0.2$
T	*T*	*T*	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

Bayesian networks with Continuous Variables

- Many real-world problems involve continuous quantities/variables.
- Continuous variables can be **discretized** but as a side-effect the resulting CPTs can become very large.
- Another possibility is to use standard probability density functions over the domains of continuous variables.
- A **hybrid Bayesian network** involves both discrete and continuous variables.

Example. Consider a system two Boolean random variables *Subsidy* and *Buys*, and two continuous random variables *Harvest* and *Cost*.



For *Cost*, we need to specify $\mathbf{P}(Cost \mid Harvest, Subsidy)$.

- The discrete parent is handled by explicitly enumerating both $\mathbf{P}(Cost \mid Harvest, subsidy)$ and $\mathbf{P}(Cost \mid Harvest, \neg subsidy)$.
- The parameters of the cost distribution (e.g. **linear Gaussian** distribution) are given as a function of the variable *Harvest*.
- The distribution $\mathbf{P}(Buys \mid Cost)$ can be determined by a soft threshold function, e.g., based on a **probit** distribution.

4. EXACT INFERENCE IN BAYESIAN NETWORKS

- An agent gets values for evidence variables from its percepts and asks about the possible values of other variables so that it can decide what action to take (recall the decision theoretic design).
- The basic task of a probabilistic reasoning system is to compute $\mathbf{P}(X \mid E_1 = e_1, \ldots, E_m = e_m)$ given a **query variable** X and exact values $e_1, \ldots, e_m$ of some **evidence variables** $E_1, \ldots, E_m$.
- The remaining variables $Y_1, \ldots, Y_n$ act as **hidden variables**.

Examples. Recalling the alarm example, the problem is to calculate distributions such as $\mathbf{P}(Burglary \mid JohnCalls, MaryCalls)$ and $\mathbf{P}(Alarm \mid JohnCalls, Earthquake)$?

Inference by Enumeration

- We introduce shorthands $\mathbf{E}$ and $\mathbf{Y}$ for $E_1, \ldots, E_m$ and $Y_1, \ldots, Y_n$, respectively, and similarly $\mathbf{e}$ and $\mathbf{y}$ for their values.
- A query $\mathbf{P}(X \mid \mathbf{e})$ can be answered by exhaustive enumeration:

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

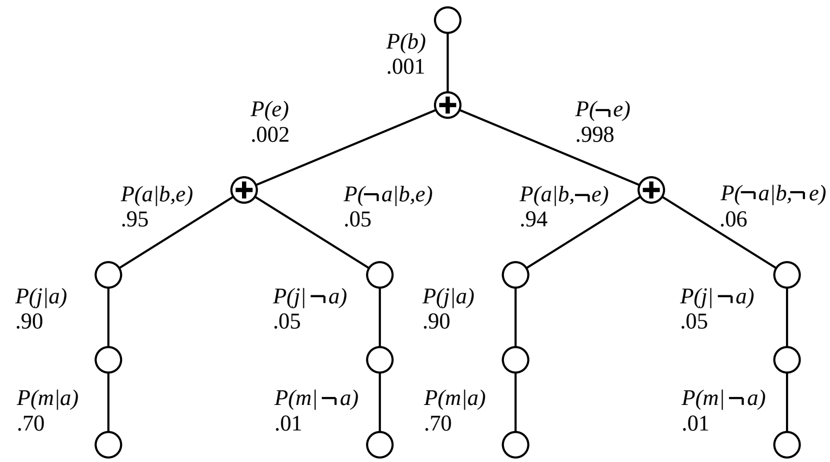
  where α is a normalizing constant.
- If a Bayesian network is used, this leads to the computation of sums of products of conditional probabilities from the network.
- The time complexity for a network of n variables is of order 2^n.

Example. Consider the query $\mathbf{P}(B \mid j, m)$ in the burglary example.

For this query, E and A are hidden variables and enumeration amounts to computing the following distribution (in a depth first fashion):

$$\begin{aligned}
\mathbf{P}(B \mid j, m) &= \alpha \mathbf{P}(B, j, m) \\
&= \alpha \textstyle\sum_e \sum_a \mathbf{P}(B, e, a, j, m) \\
&= \alpha \textstyle\sum_e \sum_a \mathbf{P}(B) P(e) \mathbf{P}(a \mid B, e) P(j \mid a) P(m \mid a) \\
&= \alpha \mathbf{P}(B) \textstyle\sum_e P(e) \sum_a \mathbf{P}(a \mid B, e) P(j \mid a) P(m \mid a) \\
&= \alpha \langle 0.00059224, 0.0014919 \rangle \\
&\approx \langle 0.284, 0.716 \rangle
\end{aligned}$$

The details of computing $P(b \mid j, m)$ are analyzed next.



☞ Certain subexpressions are computed repeatedly.

Variable Elimination Algorithm

- The enumeration algorithm can be improved substantially by doing calculations in a bottom-up fashion using **factors** which are matrices of probabilities.
- The **pointwise product** of two factors $\mathbf{f}_1(\mathbf{X},\mathbf{Y})$ and $\mathbf{f}_2(\mathbf{Y},\mathbf{Z})$ is defined by $(\mathbf{f}_1 \times \mathbf{f}_2)(\mathbf{X},\mathbf{Y},\mathbf{Z}) = \mathbf{f}_1(\mathbf{X},\mathbf{Y})\mathbf{f}_2(\mathbf{Y},\mathbf{Z})$.
- A variable X can be **summed out** from a product of factors $\mathbf{f}_i(X,\mathbf{Y})$ by computing $\sum_x(\mathbf{f}_1(x,\mathbf{Y}) \times \ldots \times \mathbf{f}_n(x,\mathbf{Y}))$.
- Multiplication takes place only when summing out variables.
- Every variable that is not an ancestor of a query variable or evidence variable is irrelevant to the query and thus removable.

Example. The computation of the previous distribution

$$\mathbf{P}(B \mid j,m) = \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a \mid B,e) P(j \mid a) P(m \mid a)$$

takes place bottom-up using factors as follows:

1. $\mathbf{f}_M(A) = \langle P(m \mid a), P(m \mid \neg a) \rangle$;
2. $\mathbf{f}_J(A) = \langle P(j \mid a), P(j \mid \neg a) \rangle$ is defined analogously;
3. $\mathbf{f}_A(A,B,E) = \mathbf{P}(A|B,E)$ is three-dimensional;
4. the variable A is summed out from the product of these three:
$$\mathbf{f}_{J,M}(B,E) = \sum_a (\mathbf{f}_A(a,B,E) \times \mathbf{f}_J(a) \times \mathbf{f}_M(a));$$
5. E is summed out similarly and $\mathbf{P}(B \mid j,m) = \alpha \mathbf{f}_B(B) \times \mathbf{f}_{J,M}(B)$.

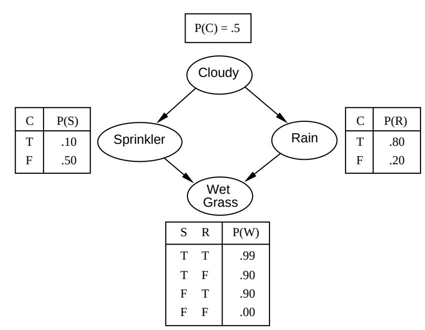
The Complexity of Exact Inference

- A **polytree** is a **singly connected** graph: there is at most one undirected path between any two nodes.
- If a belief network forms a polytree, the probability distribution $\mathbf{P}(X \mid \mathbf{e})$ can be computed very efficiently (in **linear time**).
- For **multiply connected** networks, in which at least two variables are connected by several paths, variable elimination can have exponential time and space complexity in the worst case.
- In general, exact inference in Bayesian networks is NP-hard (even #P-hard) as it includes propositional inference as a special case.

Clustering Algorithms

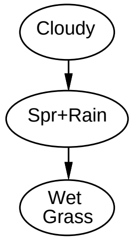
- Multiply connected Bayesian networks can be transformed into polytrees by combining some nodes into cluster nodes.

Example. Consider clustering the nodes *Sprinkler* and *Rain* in the following multiply connected network:



- The following polytree network is obtained:

P(C) = .5



C	P(S+R=x) TT	TF	FT	FF
T	.08	.02	.72	.18
F	.40	.10	.40	.10

S+R	P(W)
T T	.99
T F	.90
F T	.90
F F	.00

- Linear time algorithms can be used for query answering, but the size of the network grows exponentially in the worst case.
- Typically, there are several ways to compose cluster nodes and it is non-trivial to choose the best way to perform clustering.

5. APPROXIMATE INFERENCE IN BAYESIAN NETWORKS

- Randomized sampling algorithms provide approximate answers whose accuracy depends on the number of samples generated.
- Here sampling is applied to the computation of posterior probabilities given a prior distribution (a Bayesian network).
- There are several approximation methods including
  - Direct sampling
  - Rejection sampling
  - Likelihood weighting
  - Markov chain Monte Carlo algorithm

Direct Sampling Methods

- In direct sampling, the world described by a Bayesian network (without evidence) is simulated stochastically.
- Atomic events are randomly generated in topological order by selecting definite values for random variables.
- The value for a random variable X is chosen according to the conditional probability table associated with X.
- **Prior sampling** produces the event $x_1,\ldots,x_n$ with probability

$$S_{PS}(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid \mathrm{Parents}(X_i)) = P(x_1,\ldots,x_n).$$

- The posterior distribution $\mathbf{P}(X \mid \mathbf{e}) = \frac{\mathbf{P}(X,\mathbf{e})}{\mathbf{P}(\mathbf{e})}$ is estimated by counting the frequencies with which events occur.
- The number of samples N affects accuracy:

$$\lim_{N\to\infty} \frac{N_{PS}(x_1,\ldots,x_n)}{N} = S_{PS}(x_1,\ldots,x_n) = P(x_1,\ldots,x_n).$$

- Direct sampling is not very useful if the event $\mathbf{e}$ occurs very rarely.

Example. Let us produce one sample for the lawn watering domain:

$\mathbf{P}(\mathit{Cloudy}) = \langle 0.5, 0.5\rangle \implies$ return *true*

$\mathbf{P}(\mathit{Sprinkler} \mid \mathit{cloudy}) = \langle 0.1, 0.9\rangle \implies$ return *false*

$\mathbf{P}(\mathit{Rain} \mid \mathit{cloudy}) = \langle 0.8, 0.2\rangle \implies$ return *true*

$\mathbf{P}(\mathit{WetGrass} \mid \neg\mathit{sprinkler}, \mathit{rain}) = \langle 0.9, 0.1\rangle \implies$ return *true*

Example. E.g., $\mathbf{P}(\mathit{WetGrass} \mid \mathit{sprinkler} \wedge \mathit{rain})$ converges slowly.

Rejection Sampling in Bayesian Networks

- In its simplest form, rejection sampling can be used to compute conditional probabilities such as $P(X \mid \mathbf{e})$.
- Samples are generated from the prior distribution, but samples which do not match the evidence are rejected.
- The estimated distribution $\hat{\mathbf{P}}(X \mid \mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e}) = \frac{\mathbf{N}_{PS}(X,\mathbf{e})}{N_{PS}(\mathbf{e})}$.
- With sufficiently many samples $\hat{\mathbf{P}}(X \mid \mathbf{e}) \approx \frac{\mathbf{P}(X,\mathbf{e})}{P(\mathbf{e})} = \mathbf{P}(X \mid \mathbf{e})$.
- Rejection sampling tends to reject too many samples.

Example. Suppose that out of 100 samples, 73 are rejected as $Sprinkler = false$. Out of the remaining 27 samples, 8 satisfy $Rain = true$. Thus $\mathbf{P}(Rain \mid sprinkler) \approx \alpha\langle 8, 19\rangle = \langle 0.296, 0.704\rangle$.

Likelihood Weighting

- Likelihood weighting is similar to rejection sampling, but the values of evidence variables $\mathbf{E}$ are kept fixed while sampling others.
- The CPTs of the Bayesian network are consulted to to see how likely the event $\mathbf{e}$ is.
- In this way, the conditional probability $P(\mathbf{e} \mid x, \mathbf{y})$ is interpreted as a likelihood weight for that particular run.
- An estimate of $P(X = x \mid \mathbf{e})$ is obtained as a weighted proportion of runs with $X = x$ among the runs accumulated so far.
- Likelihood weighting converges faster than rejection sampling.
- Getting accurate probabilities for unlikely events is still a problem.

Example. Let us estimate $\mathbf{P}(Rain \mid sprinkler, wetgrass)$ by likelihood weighting. Initially, the weight w is set to 1.0.

The values of variables are chosen randomly as follows:

1. $\mathbf{P}(Cloudy) = \langle 0.5, 0.5\rangle \implies$ *cloudy* is randomly chosen.
2. *Sprinkler* is an evidence variable that has been set to *true*: w is revised to $w \times P(sprinkler \mid cloudy) = 0.1$.
3. $\mathbf{P}(Rain \mid cloudy) = \langle 0.8, 0.2\rangle \implies$ *rain* is randomly chosen.
4. *WetGrass* is an evidence variable with value *true*: w is revised to $w \times P(wetgrass \mid sprinkler, rain) = 0.099$.

☞ We have completed a run saying that $Rain = true$ given *sprinkler* and *wetgrass* with a likelihood weight 0.099.

Why Likelihood Weighting Works

- The algorithm samples each non-evidence variable in $\mathbf{Z} = \{X\} \cup \mathbf{Y}$ given the values of its parents:
$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i \mid \mathrm{Parents}(Z_i)).$$
- The weight for a given sample is $w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i \mid \mathrm{Parents}(E_i))$.
- The *weighted* probability $S_{WS}(\mathbf{z}, \mathbf{e})w(\mathbf{z}, \mathbf{e}) = P(\mathbf{y}, \mathbf{e})$.
- Likelihood weighting estimates are shown consistent as follows:

$$\begin{aligned}
\hat{P}(x \mid \mathbf{e}) &= \alpha \textstyle\sum_{\mathbf{y}} N_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) \\
&\approx \alpha' \textstyle\sum_{\mathbf{y}} S_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) \quad \text{(for large } N\text{)} \\
&= \alpha' \textstyle\sum_{\mathbf{y}} P(x, \mathbf{y}, \mathbf{e}) \\
&= \alpha' P(x, \mathbf{e}) \;=\; P(x \mid \mathbf{e}).
\end{aligned}$$

Inference by Markov Chain Simulation

- A **Markov chain Monte Carlo** (MCMC) algorithm generates the next state by sampling a value for a nonevidence variable X_i conditioned by the current values of the variables in $mb(X_i)$.
- The simulation starts from a random state $\mathbf{x}$ for $\mathbf{X} = \{X\} \cup \mathbf{Z}$.
- Each round of the simulation consists of the following steps:
  1. Increase the count $\mathbf{N}[x]$ by one for the current value x of X.
  2. Sample the value of each X_i in $\mathbf{X}$ using $\mathbf{P}(X_i \mid mb(X_i))$.
- The estimate for the distribution $\mathbf{P}(X \mid \mathbf{e})$ is obtained by normalizing the counts in $\mathbf{N}[X]$.

Why MCMC Works

- The sampling process settles into a "dynamic equilibrium" in which the long-run fraction of time spent in each state is exactly proportional to its posterior probability.
- A **Markov chain** is defined by **transition probabilities** $q(\mathbf{x} \rightarrow \mathbf{x}')$ from a state $\mathbf{x}$ to a state $\mathbf{x}'$.
- Let $\pi_t(\mathbf{x})$ denote the probability of a state $\mathbf{x}$ after t steps.
- For the next step, we have $\pi_{t+1}(\mathbf{x}') = \sum_{\mathbf{x}} \pi_t(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}')$.

Definition. The chain has reached its **stationary distribution** π if $\pi_{t+1} = \pi_t$, i.e., π is defined by $\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}')$ for all $\mathbf{x}'$.

Detailed Balance

- One interpretatation of the equation $\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}')$ is that the expected *outflow* from each state (population) is equal to the expected *inflow* from all the states.
- Assuming the equality of flows in both directions leads to the property of **detailed balance**: for all $\mathbf{x}$ and $\mathbf{x}'$:

$$\pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') = \pi(\mathbf{x}') q(\mathbf{x}' \rightarrow \mathbf{x}).$$

- Stationarity is implied by detailed balance:

$$\begin{aligned} \textstyle\sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') &= \textstyle\sum_{\mathbf{x}} \pi(\mathbf{x}') q(\mathbf{x}' \rightarrow \mathbf{x}) \\ &= \pi(\mathbf{x}') \textstyle\sum_{\mathbf{x}} q(\mathbf{x}' \rightarrow \mathbf{x}) \\ &= \pi(\mathbf{x}'). \end{aligned}$$

Gibbs sampler

- Let X_i be the variable to be sampled and $\overline{X_i}$ all the hidden variables other than X_i.
- The **Gibbs sampler** is based on transition probabilites

$$q(\mathbf{x} \rightarrow \mathbf{x}') = q((x_i, \overline{\mathbf{x}_i}) \rightarrow (x_i', \overline{\mathbf{x}_i})) = P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) = P(x_i' \mid mb(X_i)).$$

- Gibbs sampler is in detailed balance with the true posterior:

$$\begin{aligned} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') &= P(\mathbf{x} \mid \mathbf{e}) P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) \\ &= P(x_i, \overline{\mathbf{x}_i}, \mathbf{e}) P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) \\ &= P(x_i \mid \overline{\mathbf{x}_i}, \mathbf{e}) P(\overline{\mathbf{x}_i} \mid \mathbf{e}) P(x_i' \mid \overline{\mathbf{x}_i}, \mathbf{e}) \\ &= P(x_i', \overline{\mathbf{x}_i}, \mathbf{e}) P(x_i \mid \overline{\mathbf{x}_i}, \mathbf{e}) \\ &= \pi(\mathbf{x}') q(\mathbf{x}' \rightarrow \mathbf{x}). \end{aligned}$$

SUMMARY

- **Conditional independence** information can be used for structuring and simplifying knowledge about an uncertain domain.
- **Bayesian networks** provide a natural way to represent conditional independence information.
- A Bayesian network is a complete (and often also very compact) representation of the joint probability distribution.
- Efficient algorithms exist for Bayesian networks that are topologically *polytrees*, but reasoning with Bayesian networks is NP-hard in general.
- Probabilities can be estimated by **sampling methods**.

QUESTIONS

- Build a Bayesian network for the soccer domain.
  1. Choose appropriate variables for the description of the domain.
  2. Choose an ordering for the variables.
  3. Construct the actual belief network by
     (i) analyzing dependencies among variables and
     (ii) defining CPTs for each variable.
- Make both causal and diagnostic inferences using the network.