Alma Mater Studiorum · Università di Bologna

Second Cycle Degree
Artificial Intelligence

Fundamentals of Artificial Intelligence and Knowledge Representation
Module 3

Student:
Matteo Canghiari

Academic Year 2025/2026

Chapter 1

Introduction to uncertainty and probabilistic reasoning

1.1 Basic probability notation

Every agent based on **decision theory** needs a formal language to use and represent probabilistic informations. Typically AI needs a more suited and consistent approach than the traditional probability theory. This section includes all the necessary definitions and examples to understand the subsequent arguments in depth.

Definition

The set of all possible worlds is called the **sample space**, denoted Ω. Any subset $A \subseteq \Omega$ is an **event**. Any element $\omega \in \Omega$ is called **sample point**.

Definition

A **probability space** is a sample space with an assignment $P(\omega)$ for every $\omega \in \Omega$ where:

- $0 \leq P(\omega) \leq 1$
- $\sum P(w) = 1$ for every $\omega \in \Omega$

Definition

A **random variable** is a function from sample points to some range, e.g., the reals or Booleans.

e.g. *Odd(1) = true*

Definition

$\boldsymbol{P}$ induces a **probability distribution** for any random variable $\boldsymbol{X}$:

$$P(X = x_i) = \sum_{\omega:X(\omega)=x_i} P(\omega)$$

A **probability distribution** gives values for all possible assignment.

Definition

Prior or **unconditional probabilities** of propositions correspond to belief prior to arrival of any new evidence.

e.g. *P(Cavity = True) = 0.1*

Definition

The **Joint Probability Distribution** for a set of random variables gives the probability of every sample point on those random variables.

e.g. $\boldsymbol{P}$*(Weather, Cavity) = a* 2×4 *matrix of values:*

Weather =	*sunny*	*rain*	*cloudy*	*snow*
Cavity = True	0.144	0.02	0.016	0.02
Cavity = False	0.576	0.08	0.064	0.08

Table 1.1: Probability distribution of the Weather random variable

Every question about a certain domain can be answered by the joint distribution because every event is a sum of sample points.

Definition

A function $p : R \to R$ is a **probability density function** (**pdf**) for X if it is a nonnegative integrable function s.t.

$$\int_{Val(X)} p(x)dx = 1$$

Definition

Conditional or **posterior probabilities** $P(X|Evidence)$ represent a more informed distribution in the light of new **evidence**.

e.g. $P(cavity|toothache) = 0.8$

It does not mean *"if I have toothache then there is* 80% *of chance that there is also a cavity"*, instead the evidence mean *"given toothache evidence is all I know"*.

The typically definition of conditional or posterior probability is:

$$P(a|b) = \frac{P(a \wedge b)}{P(b)} \text{ if } P(b) \neq 0$$

Otherwise, numerator can be written by the **product rule**:

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

The product rule at the same time is applied to whole distributions, not only for single values as done previously.

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

1.2 Inference using full joint distribution

This paragraph describes a new method to retrieve informations from data, named **probabilistic inference**. It allows the computation of conditional probabilities for query propositions by given evidence. Starting from an example is defined the **full joint distribution** as the knowledge base from which answers to all questions.

Example

e.g. (*Toothache, Cavity, Catch*) is just a domain consisting of three Boolean variables. *Catch* condition occurs when the dentist's steel probe catches in the tooth. Based on the domain, the **full joint distribution** seems like this:

	toothache		$\neg$*toothache*	
	catch	$\neg$*catch*	*catch*	$\neg$*catch*
cavity	0.108	0.012	0.072	0.008
$\neg$*cavity*	0.016	0.064	0.144	0.576

Table 1.2: Full joint distribution of Toothache, Cavity and Catch

The equation

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

gives a direct way to calculate probabilities of any assertions, summing up all the possible worlds that satisfy the original proposition.

e.g. $P(toothache) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$

e.g. $P(cavity \vee toothache) = 0.108+0.012+0.072+0.008+0.016+0.064 = 0.28$
It's also possible compute conditional probabilities:
e.g. $P(\neg cavity|toothache) = \frac{P(\neg cavity \wedge toothache)}{P(toothache)} = \frac{0.016+0.064}{0.2} = 0.4$
Notice that in this calculation the term $P(toothache)$ remains constant, no matter which value of *Cavity* is computed. In fact, it can be viewed as a **normalization constant** (α) for the whole distribution $\mathbf{P}(Cavity|toothache)$, ensuring that the positive and negative case sum up to one, as the second probability axiom requires.

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

Definition

The first probability calculated $P(toothache)$ is called **marginalization**, or more simply **summing out**, because it sums up the probabilities for each possible value of the other variables.

Definition

The second one $P(\neg cavity|toothache)$ is named **conditioning**, a variant of marginalization that involves conditional probabilities instead of joint probabilities.

Definition

From the example, it's possible to extract a general inference procedure. Let $\mathbf{Y}$ be the query variables. Let $\mathbf{E}$ be the list of evidence variables, let $\mathbf{e}$ be the list of observed values for them, and let $\mathbf{H}$ be the unobserved variables. The **probability query** $\mathbf{P}(Y|\mathbf{e})$ defines the posterior joint distribution of a set of **query variables** $\mathbf{Y}$ given specific values $\mathbf{e}$ for some **evidence variables** $\mathbf{E}$:

$$\mathbf{P}(Y|e) = \alpha\mathbf{P}(Y, E = e) = \alpha \sum_h \mathbf{P}(Y, E = e, H = h)$$

The full joint distribution can answer probabilistic queries for discrete variables, but only for small domains. It does not scale well: for a domain described by n Boolean variables, it requires an input table of size $O(2^n)$ and takes $O(2^n)$ time to process a question. The full joint distribution in tabular form is just not a practical tool for building reasoning systems.

Chapter 2

Bayesian network representation

2.1 Independence

If we expand the full joint distribution defined in Figure 1.2 by adding a new random variable, *Weather*, it becomes $\mathbf{P}(Weather, Toothache, Cavity, Catch)$, which has $2 \times 2 \times 2 \times 4 = 32$ entries. But, what is the relationship between these four random variables? For instance, are the $P(cloudy, toothache, cavity, catch)$ and $P(toothache, cavity, catch)$ related? This last question can be expressed in probabilistic terms as:

$$P(cloudy, toothache, catch, cavity) = \\ P(cloudy|tootchache, cavity, catch)P(toothache, cavity, catch)$$

At the same time, we can imagine that *Toothache, Cavity, Catch* should be independent from *Weather*. Therefore, the following assertion seems reasonable:

$$P(cloudy|toothache, catch, cavity) = P(cloudy)$$

From this, we can deduce:

$$P(cloudy, toothache, catch, cavity) = P(cloudy)P(toothache, catch, cavity)$$

Or generally:

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

Thus, the initial 32 entries table can be divided from one 8-entries table and one 4-entries table. The property used in the previously equation is called **independence**.

First of all are introduced some basic definitions and examples to understand the

effectiveness of independence.

Definition

A and B are **independent**, denoted $\mathbf{P} \models (A \perp B)$, if and only if $\mathbf{P}(A|B) = \mathbf{P}(A)$ or $\mathbf{P}(B|A) = \mathbf{P}(B)$ or $\mathbf{P}(A|B) = \mathbf{P}(A)\mathbf{P}(B)$

When they are available, independence assertions can help in reducing the size of the domain representation and the complexity of the inference problem. Unfortunately, clean separation of entire sets of variables by independence are quite rare. Moreover, even the independence subset can be quite large, for instance, dentistry might involve dozens of diseases and symptoms, all of which are associated. To handle such problems, we need more specific methods than the general concept of independence, one of them is named **conditional independence**. Let see an example of conditional independence.

Example

i.e. given $\mathbf{P}(Toothache, Cavity, Catch)$ has $2^3 - 1 = 7$ independent entries[a]. If I have a cavity, the probability that the probe catches in it does not depend on whether I have toothache:

$$P(catch|toothache, cavity) = P(catch|cavity)$$

The same independence hold if I haven't got a cavity:

$$P(catch|toothache, \neg cavity) = P(catch|\neg cavity)$$

Catch is **conditional indipendent** of Toothache given Cavity[b].

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

$$\mathbf{P} \models (Toothache \perp Catch|Cavity)$$

Using the chain rule, the full joint distribution becomes:

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

$2 + 2 + 1 = 5$ indipendent numbers, we have less entries than before.

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

[a] Why 7 independent entries and not 8 as before? Simply, if we know 7 of them the 8th is automatically determined, must be the last value remaining.

[b] This introduces the meaning of the flow of influence.

The Section 1.1 defined the **product rule**. It can be written in two forms:

$$P(a \wedge b) = P(a|b)P(b) \text{ and } P(a \wedge b) = P(b|a)P(a)$$

Combining the right-hand side of each equation and dividing by $P(a)$, we get the **Bayes' rule**.

Bayes' theorem

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}$$

or in distribution form:

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

This turns out to be very useful for assessing **diagnostic** probability from **causal** probability.

On the surface, Bayes' rule does not seem very useful. It allows to compute the single term $P(b|a)$ in terms of three items: $P(a|b)$, $P(b)$ and $P(a)$. But the Bayes' rule is useful in practice because there are many cases where we have probabilities for these three items and need to compute the fourth. Often, we perceive as evidence the **effect** of some unknown **cause** and we would like to solve for that cause. In that case, the Bayes' rules becomes:

$$P(cause|effect) = \frac{P(effect|cause)P(cause)}{P(effect)}$$

The conditional probability $P(effect|cause)$ defines the relationship in the **causal** direction, while $P(cause|effect)$ describes the **diagnostic** direction. Let see an example.

Example

Say 1 individual in 50.000 suffers from meningitis, 1% from a stiff neck, and 70% of the times meningitis causes a stiff neck. *What is the probability that an individual with a stiff neck has meningitis?*

$P(s|m) = 0.7$
$P(m) = 1/50.000$
$P(s) = 0.01$
$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.7 \times (1/50.000)}{0.01} = 0.0014$

We have seen that the Bayes' rule seems useful for answering probabilistic queries conditionated on one piece of evidence. But, what happens when we have two or more pieces of evidence? For instance, what a dentist conclude if her steel probe cathes in the tooth of a patient?

Example

i.e. If we know the full joint distribution 1.2, we can define the answer as:

$\mathbf{P}(Cavity|toothache \wedge catch) = \alpha\langle 0.108, 0.016\rangle = \langle 0.871, 0.129\rangle$

However, this approach does not scale up to larger number of variables. We can try using the Bayes' rule to reformulate the problem:

$\mathbf{P}(Cavity|toothache \wedge catch) = \alpha\mathbf{P}(toothache \wedge catch|Cavity)\mathbf{P}(Cavity)$

For this reformulation, we must know the conditional probabilities of the conjuction for each value of Cavity. That might be simple for just two variables, but again it does not scale up. Thus, we need to find some assertions about the domain that will enable us to simplify the expressions.

The notion of **independence** provides a clue. It would be nice if Toothache and Catch were independent, but they aren't: if the probe catches in the tooth, then it is likely that the tooth has a cavity and that cavity causes the tootchache. By this last assertion, we can allude that these variables are independent, given the presence or the abscence of a cavity. Each effects is directly caused by the cavity, but neither has a direct effect on the other. Mathematically, this property is written as follows:

$$\mathbf{P}(toothache \wedge catch|Cavity) = \mathbf{P}(toothache|Cavity)\mathbf{P}(catch|Cavity)$$

This equation introduces the meaning of **conditional independence**: *toothache* is conditionally independent from *catch* given *Cavity*. Now the information requirements are the same as for inference, using each piece of evidence separately: the prior probability $\mathbf{P}(Cavity)$ for the query variable and the conditional probability for each effect, given its cause.

$$\mathbf{P}(toothache \wedge catch|Cavity) =$$
$$\alpha\mathbf{P}(toothache|Cavity)\mathbf{P}(catch|Cavity)\mathbf{P}(Cavity)$$

Definition

The **conditional independence** of two variables X and Y, given a third variable Z, is:

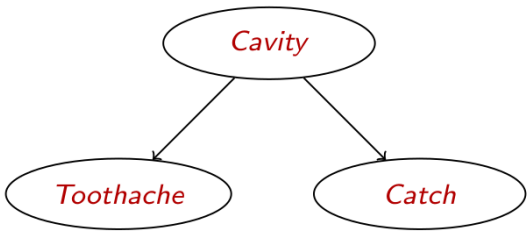
$$\mathbf{P}(X, Y|Z) = \mathbf{P}(X|Z)\mathbf{P}(Y|Z)$$

In this way, the original table is decomposed into three small tables. The table 1.2 has seven independent entries. The smaller tables contain five indipendent numbers, 2 for the conditional probability distributions and 1 for the prior distribution $\mathbf{P}(Cavity)$. Right now the size of the representation grows as $O(n)$ instead of $O(2^n)$, it grows by a **linear** pace not anymore by a **exponential** pace. Finally, we can say

that conditional independence and absolute independence can allow probabilistic systems to scale up.

Example

i.e. Conceptually, Cavity **separates** Toothache and Catch because it is a direct cause of both of them.



Definition

The full joint distribution can be written as:

$\mathbf{P}(Cause, Effect_1, Effect_2, ..., Effect_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(Effect_i|Cause)$

This probability distribution is called **Naive Bayes**[a].

[a]The naive Bayes model is the most common way to solve labeling tasks, such as classification. The total number of parameters grows **linearly**.

2.2 Bayesian network representation

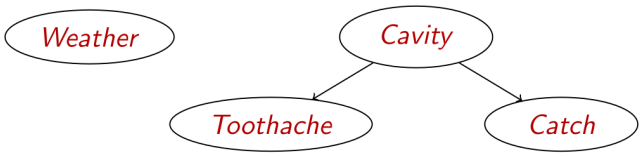
The previous paragraph noted the importance of absolute and conditional independence relationships in simplifying probabilistic representation. This section introduces a systematic way to represent such relationships in the form of **Bayesian networks**. We define the syntax and semantics of these networks and show how they can be used to capture uncertain knowledge.

A Bayesian network is a simple graphical notation for conditional independence assertions and hence for a compact specification of full joint distribution. The Bayesian network's syntax is composed by:

1. Each node corresponds to a random variable.
2. A set of directed links or arrows connects pairs of nodes.
3. Each node X_i has a conditional probability distribution $\mathbf{P}(X_i|Parents(X_i))$, that quantifies the effect of the parents on the node.

Example

i.e. Topology of network encodes conditional independence assertions:



- Weather is independent of the other variables[a].
- Toothache and Catch are conditionally independent given Cavity[b].

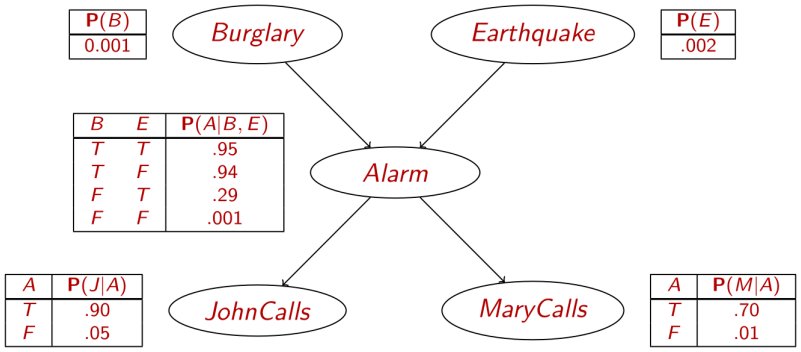
[a]Formally, the absolute or conditional independence is indicated by the absence of a link between nodes.

[b]The intuitive meaning of an arrow is typically that X has a direct influnce on Y, which suggests that causes should be parents of effects.

Example

i.e. I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

The random variables are: *Burglar, Earthquake, Alarm, MaryCalls, JohnCalls* [abc].



[a]The network topology reflects **causal** knowledge, from the causes nodes we define the effects nodes.

[b]For each node the conditional distribution are shown as a **conditional probability table**, or simply CPT.

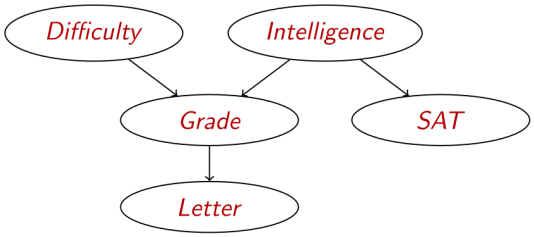
[c]Let's take a look at the tables. In this network we are talking about joint distribution, not full joint distribution. Simply, the full joint distribution about boolean random variables can be computed by $1 - P(a)$.

We begin the discussion with a simple toy example, the *student network*.

Example

i.e. A student's grade depends on intelligence and on the difficulty of the course. SAT scores are correlated with intelligence. A professor writes recommendation letters by only looking at grades.

In this case, our probability space is composed by five relevant random variables: *Difficulty (D)*, *Intelligence (I)*, *SAT score (S)*, *Grade (G)* and *Letter (L)*.



Consider a particular student, George, that he would like to reason using the student network. We might ask how likely George is to get a strong recommendation from his professor in Analysis. Knowing nothing else about George and his grade, this probability is around the 50 percent.

We now find out that George is not so intelligent. The probability that he gets a strong letter from the professor goes down to 39. We now further discover that Analysis is an easy class. The probability that George receive a strong letter is now about 51 percent.

Queries and answers such as this, where we predict the behaviors from causes to effects, are called **causal reasoning** or **prediction**[a].

Now, assume a recruiter, trying to hire George based on our previous model. By a prior probability, the recruiter believes that George is 30 percent likely to be intelligent. He obtains George's grade record for the class Analysis and sees that George received a low score in that class. His probability that George has high intelligence goes down, about 7.9 percent. We note that the probability that the class is a difficult one also goes up, from 40 percent to 62.9 percent. Now, consider that the recruiter lost George's trascript of records, and has only the recommendation letter from George's professor in Analysis, which is weak. The probability that George has high intelligence still goes down, but only to 14 percent. Note that if the recruiter has both the grade and the letter, we have the same probability as if he had only the grade.

Queries such as this, where we reason from effects to causes, are named **evidential reasoning** or **explanation**[b].

Finally, George submits his high SAT score to the recruiter. The probability

that George has high intelligence goes up dramatically, from 7.9 percent to 57.8 percent. Intuitively, the reason that the SAT score outweights the poor grade is that a student with low intelligence are unlikely to get good scores on their SAT, whereas students with high intelligence can still get C grades in hard class. Indeed, we see that the probability that Analysis is a difficult one goes up from 62.9 percent to 76 percent.

This last pattern is a interesting one. The information about SAT score told us other informations about the student's intelligence, which, in conjuction with the student's grade, gave us some clues about the difficulty of the course. Let examine this pattern in probabilistic terms.

We are saying

$P_s(i^1|g^3) = 0.079$

On the other hand, if we consider that Analysis is a hard class, we have

$P_s(i^1|d^1, g^3) = 0.11$

Here, we are partially explaining why George has got a poor grade. By the way, taking a more tricky example, for instance George got a middle grade in Analysis, we have that

$P_s(i^1|g^2) = 0.175$

Also if Analysis is a hard class, we get

$P_s(i^1|d^1, g^2) = 0.34$

In effect, we have justified the poor grade via the difficulty of the class. Explaining away is an instance of a general pattern called **intercasual reasoning**, where different causes of the same effect, so they are parents of the effect node, can interact.

[a] It reflects the causal direction, from the parent nodes we are just defining which is their influence on their children.

[b] In this case, we are not reasoning from top to bottom, but instead from bottom to top.

Compactness

A Bayesian network can often be more *compact* than the full joint distribution. The **compactness** of a Bayesian network is a property that makes easy to handle domain with many random variables.

At the same time, a Bayesian network grows *linearly*, instead of an *exponential* growth by full joint distribution. Assuming a domain composed by **n** Boolean vari-

ables, where each of them is associated with a CPT *(Conditional Probability Table)*[1]. If each variable has no more then **k** parents, the complete network requires $O(n \times 2^k)$ numbers, as we already know to complete the full joint distribution are necessary $O(2^n)$.

Example

i.e. Comparison of parameters required from the previously example between Bayesian network and full joint distribution.

For the burglar network:

$1 + 1 + 4 + 2 + 2 = 10$ numbers required by the Bayesian network

$2^5 - 1 = 47$ numbers required by the full joint distribution

Global semantics

A Bayesian network is a directed acyclic graph with some numeric parameters attached to each node. One way to define what the network means is to define the way in which it rapresents a full joint distribution.

Definition

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

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

Example

i.e.
$P(j, m, a, \neg b, \neg e)$
$= P(j|a)P(m|a)P(a|\neg b \wedge \neg e)P(\neg b)P(\neg e)$
$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 = 0.000628$

Flow of influence

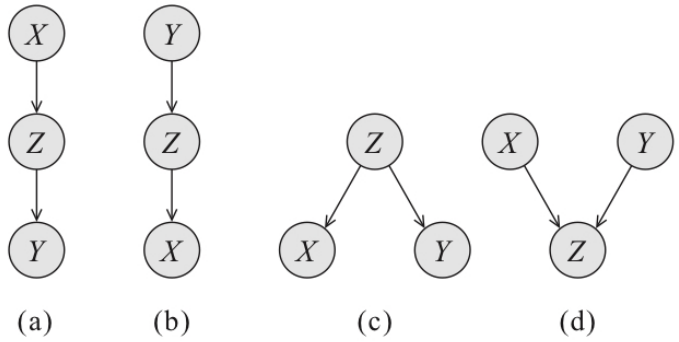
Until now, we used the intuition that edges represent direct dependence. For instance, we said that the letter recommendation from the professor depends only on the student's grade; this state was encoded by the fact that there is an exit edge from G that arrive to L. This intuition is **not** always true.

[1] A CPT for a Boolean variable $\mathbf{X_i}$ with **k** parents, has $\mathbf{2^k}$ rows for the combinations of parents values and usually is defined only the True case; the negative case, however $X_i = False$, can be simply computed by the third probability axiom $\mathbf{1 - p}$.

The aim of this section is to understand when we can guarantee independence between random variables. First of all, we begin with a simple case analysis: we try to understand when a variable **X** can influence **Y** given **Z**.

Direct connection. This is the simple case, when X and Y are directly connected via an edge. If X and Y are directly connected, we can always get examples where they influence each other, regardless of **Z**.

Indirect connection. Now we are considering the more complicated case when X and Y are not directly connected, but there is a trail between them. There are four cases where X and Y are connected via Z.



The first two correspond to causal chains, the third to a common cause, and the last one to a common effect.

(a) Causal trail. We have a chain like $X \to Z \to Y$. X cannot influence Y via Z if Z is observed.

(b) Effect trail. Now we have the same trail but in the opposite direction, so $Y \to Z \to X$. Another time, Y cannot influence X via Z if Z is observed.

(c) Common cause. This type of trail defines the same grade of independence as before. If Z is observed then neither X or Y can influence each other.

(d) Common effect. In the previously cases, we see a common pattern: if Z is observed then neither X or Y can influence each other. By the way, this kind of trail, $X \to Z \leftarrow Y$, has a new behavior. If Z is not observed influence cannot flow along the trail. So if Z is observed X and Y are independent. In the student example we analyzed this case, which we called *intercausal reasoning*; we showed that the probability that student has high intelligence goes down when we observe that his grade is a poor score, but then goes up when we observe that the class is a hard one. Let us consider a variant of the same case. Assume that we do not observe the student's grade, but we observed that he received an awful recommendation letter.

Intuitively, the same phenomenon happens. The weak letter told us that he received a low grade, and it is sufficient to correlate Intelligence and Difficulty.

Definition

If influence can flow from X to Y via Z, the trail $X \longleftrightarrow Z \longleftrightarrow Y$ is **active**.

If we consider a longer trail $X_1 \rightarrow \cdots \rightarrow X_n$, the first variable X_1 can influence the last variable X_n, if influence can flow through every single node of the trail. This will be true if and only if every two-edge trail $X_{i-1} \longleftrightarrow X_i \longleftrightarrow X_{i+1}$ along the trail allows influence to flow.

Definition

Let **Z** be a subset of observed variables. The trail $X_{i-1} \longleftrightarrow X_i \longleftrightarrow X_{i+1}$ is **active given** Z if

- $\forall X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, X_i or one of its descendants are in Z[a]
- no other node along the trail is in Z

[a]The trail reported is the common effect case.

However, inside the Bayesian network literacy there is another important notion, named **d-separation**.

Definition

Two sets of nodes **X**, **Y** are d-separated given Z if there is no active trail between any $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$ given **Z**.

It's possible to summarize few steps to follow to determine if X and Y are independent given Z:

1. Mark all nodes in Z or having descendants in Z.
2. Traverse the graph from X to Y, stopping if we get to a **blocked** node[2].
3. If we can't reach Y, then X and Y are independent.

Another aspects about independence are introduces by the meaning of **local semantics** and **Markov blanket**.

[2]A node is blocked if that node is the middle of an unmarked v-structure *(common effect case)*, or belongs to Z *(cannot be both)*.

Definition

Local semantics define that each node is conditionally independent of its **non-descendants** given its parents[a].

[a] Here, we are considering the parents of the first node, not the parents of non-descendants.

Definition

Each node is conditionlly independent of all the others nodes given its **Markov blanket**: so its parents, children and children's parents.

Chapter 3

Constructing Bayesian networks

The notion of global semantics[1] describes what a Bayesian network means. The next step is to explain how to construct a Bayesian network in such a way that the resulting joint distribution is a good representation of a given domain.

Another clue about the global semantics is that the Bayesian network is a correct representation of the domain only if each node is conditionally independent of its predecessors in the node ordering, given its parents. We can satisfy this condition with this methodology:

1. First determine the set of variables that are required to model the domain. Order them, $X_1, \ldots, X_n$. Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects.
2. For each random variable defined do these actions:
   - Choose, from $X_1, \ldots, X_{i-1}$ a minimal set of parents for X_i, such the global semantics equation is satisfied.
   - For each parent insert a link from it to X_i.
   - Define the conditional probability table.

Intuitively, the parents subset of node X_i should contain all the nodes in the set $X_1, \ldots, X_n$ that directly influence X_i. This choice of parents guarantees the global semantics:

$$\mathbf{P}(X_1, \ldots, X_n)$$
$$= \prod_{i=1}^{n} \mathbf{P}(X_i | X_1, \ldots, X_i - 1)$$

[1] It defines the full joint distribution as the product of local conditional probability distributions.

$$= \prod_{i=1}^{n} \mathbf{P}(X_i | Parents(X_i))$$

Example

i.e. Let's construct a Bayesian network that help us win a prize.

We're guests on a TV game show. We stand in front of three closed doors. A prize hides behind one of them. We choose the door on the left. At this point, the host, who knows where prize is, opens the middle door, to reveal it is empty. We are offered to modify our choice. Should we?

The necessary nodes to model the domain are: *Guest*, *Host* and *Prize*.

We can say that *Guest* has a direct influence on *Host*, but at the same time it doesn't happens between $Guest \rightarrow Prize$, because he don't know where is located the amount of money. So he has not an influence on *Prize*, cannot be insert a link between them. However, the *Prize* has a direct influence on *Host*. The final result it's a *v-structure*, where *Host* is the same child either for *Guest* and *Prize*.

In probabilistic terms, it could be described as:

$P(P) = P(P|G) \implies P \perp G$
$P(H) \neq P(H|G)$
$P(H|G) \neq P(H|G, P)$

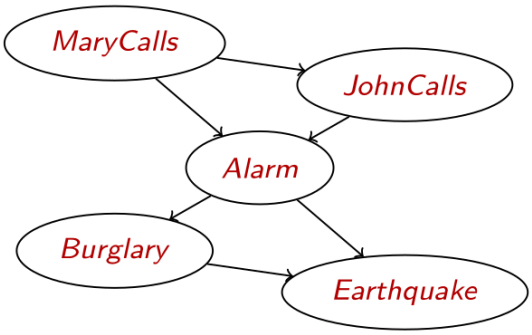
We can get a quite compact network only if we choose the node ordering well. What happens if we choose a strange order? Consider the Burglary example again.

Example

Suppose we decide to order the nodes as follows:

$$MaryCalls, JohnCalls, Alarm, Burglary, Earthquake$$

We then get a more complicated network shown as below.



The process goes as follows:

1. *MaryCalls*: first node, no parents, it is like a *prior probability*.
2. *JohnCalls*: if Mary calls, probably the alarm has gone off, which would make it more likely John calls. *JohnCalls* needs *MaryCalls* as parent.
3. *Alarm*: clearly, if both call, it is more likely that the alarm has gone off. *Alarm* needs *MaryCalls* and *JohnCalls* as parents nodes.
4. *Burglary*: if we know the alarm state, then the call from Mary or John doesn't tell us anything about the Burglary:
   $\mathbf{P}(Burglary|Alarm, JohnCalls, MaryCalls) = \mathbf{P}(Burglary|Alarm)$
   Hence we need just *Alarm* as parent.
5. *Earthquake*: if the alarm is on, it is more likely that there has been an earthquake. But if we know that there has been a Burglary, then that explain the alarm, so the probability of an earthquake would decrease. Hence, we need both *Alarm* and *Burglary* as parent.

The resulting network has two more links than the original one and requires three more probabilities to be specified. What's worse, some of the links represent strange relationships that require difficult probability judgments.

If we try to build a Bayesian network by the **diagnostic direction**, from effects to causes, we end up having to specify additional dependencies between independent causes. If we use the **causal direction**, we end up having to specify fewer numbers, and the network seems to be more easier to handle.

3.1 Causal networks

Causal networks is the last notion that tell us something about the Bayesian networks, in particular it defines how to design networks that belong to an uncertain domains.

In the previous paragraph we saw that any ordering of nodes permits a consistent construction of the Bayesian network, but at the same time if the ordering respects the causality knowledge, the network seems to be easier to handle.

Definition

Causal networks are a restricted class of Bayesian networks that forbids all but not causally compatible orderings.

Let's clarify this concept with an example.

Example

i.e. Assume this simple network, composed by two nodes.

(a) (b)

A and B represent the same domain and they are equally good distributions. Each of them produce the same full joint distribution, whatever the ordering of nodes chosen. But these representation are not equivalent. Intuitively, the main difference is that the first graph follows **casuality**, instead the second one describes the diagnostic direction.

When defining causal networks we have to consider a particular question:

which responds to which?

It substitutes the initial question that we used to assume: *is there any flow of influence?* We can define a recurrent method to answer the first question, as follows:

1. Draw an arrow from **Fire** to **Smoke** if nature assigns a value to **Smoke** on the basis of what nature learns about **Fire**.

2. Do not draw an arrow from **Smoke** to **Fire** if you judge that nature assigns a **Fire** a truth value based on variables other than **Smoke**.

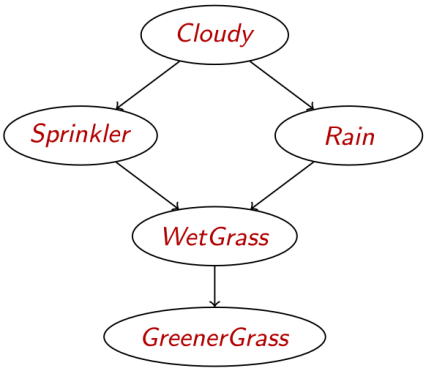
3. For each variable X_i that can take values $x_i = f_i(OtherVariables)$, draw $X_j \to X_i$ iff X_j is one of the arguments of f_i

$x_i = f_i(\dots)$ is called **structural equation**

The last point might be a misunderstanding, it's simply defining which variables would be linked to X_i.

Example

i.e. Lawn network.

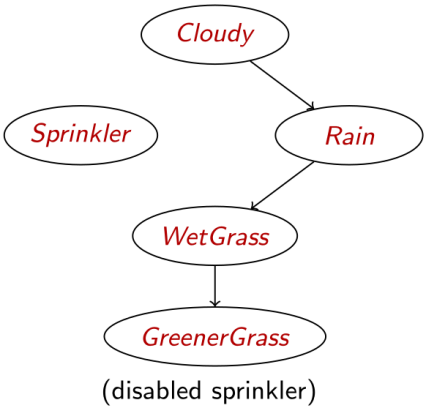


As written before, the data structure represents a Lawn network. It's possible to describe the same network by **structural equations**, as:

$C = f_C(U_C)$
$R = f_R(C, U_R)$
$S = f_S(C, U_S)$
$W = f_W(S, R, U_W)$
$G = f_G(W, U_G)$

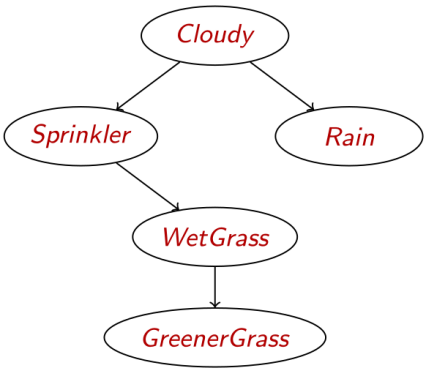
It's important to understand what structural equation semantic means. Generally U_* stands for uncertainty, there are no nodes that are not dominated by **uncertainty**. Upper case letters, as C, R, S, W, G, define the nodes. Finally, $f_*(\dots)$ functions symbolise which random variables assign sample points to their child node; this is translated via a direct connection between nodes.

Furthermore, structural equations are capable to describe the same domain even if are made local changes in the environment. We assume that the sprinkler has been turned off. The resulting representations via Bayesian network and structural equations change as:



$C = f_C(U_C)$
$R = f_R(C, U_R)$
$S = f_S(U_S)$
$W = f_W(R, U_W)$
$G = f_G(W, U_G)$

Again, if we cover the lawn such as Rain has not any effect, the representations are like:



(lawn covered by a tent)

$C = f_C(U_C)$
$R = f_R(C, U_R)$
$S = f_S(C, U_S)$
$W = f_W(S, U_W)$
$G = f_G(W, U_G)$

What happens if we set off Sprinkler? In probabilistic terms, it might be described as follows:

$$P(GreenerGrass|Sprinkler = True) = ?^a$$

This tricky question has two means:

- If Sprinkler is on, then the probability of WetGrass and GreenerGrass might increase.
- Sprinkler set off tell us that it's not raining outside.

These assertions gives a clue: we are expressing the same things as we might do with *prior probabilities* and *conditional probabilities*. However, the same changes are solved via Bayes nets semantics:

$$P(c, r, w, g|do(S = true))^b$$

The right hand side of the argument is named **do-operator**. It allows to represent interventions and predict their observable consequences, without changing the domain.

[a] Here, we are observing that Sprinkler is on.
[b] Instead, we are turning Sprinkler on to see what happens.

Bayesian networks and structural equations express the domain exactly in the same way, but nevertheless there is a huge difference. If structural equations can afford interventions and domain still be the same, Bayesian network does not! In Bayesian networks we are deducing behavioral aspects observing some evidences, differently in structural equations we are deducing behavioral aspects doing some operations.

3.2 Representing conditional distributions

Even if the number of parents k is smallish, fill any CPT requires $O(2^k)$ numbers and it grows exponentially with number of parents. Instead, if we assume a continuous-valued parent or child, CPT becomes **infinite**[2]. Usually, a solution to solve these types of problem consists of **canonical distributions**.

The simplest case is provided by **deterministic nodes**:

$$X = f(Parents(X))$$

A deterministic node has its values specified exactly by the values of its parents, with no uncertainty.

Example

i.e. Logical relationship, so there is no stochasticity.

The relationship between the parent nodes $(Canadian, US, Mexico)$ and the child node $(NorthAmerican)$ is simply that the child is the disjunction, or the logical OR, of the parents.

[2] One of the nicest thing of Bayesian network is the capability to combine together different types of variables, sush as descrete, continuos, deterministic, not-deterministic and so on.

Canadian	US	Mexico	NorthAmerican
F	F	F	F
T	F	F	T
F	T	F	T
F	F	T	T
T	T	F	T
T	F	T	T
F	T	T	T
T	T	T	T

i.e. Numerical relationship.

If the parent nodes are a lake's inflows ($Rivers, Runoff, Precipitation$) and the outflows ($Rivers, Evaporation, Seepage$) and the child node is the change in the water level of the lake, then the value of the child is the sum of the inflow parents minus the sum of the outflow parents.

$$\frac{Level}{t} = inflow + precipitation - outflow - evaporation$$

Another interesting example is the **Noisy-OR distribution**, which is very useful representation. Now, we will describe an example to understand the intuition of Noisy-OR distribution given a medical domain.

Example

i.e. Medical representation.

Cold	Flu	Malaria	P(Fever)	P(¬Fever)
F	F	F	0.0	...

Imagine that you are looking to the symptom, which is the ($Fever$), and the symptom can be caused by different deseases, such as ($Cold, Flu, Malaria$). However, the network is defined by three different causes parent of the same effect.

As we can suppose, it is very common in medical domain to diagnose the cause given the symptom, ($Fever$). Furthermore, as written before, the size of this representation is exponential, it requires 2^3 parameters, and it encreases with the number of the parents.

We can make some simplified assumptions, that let us to reduce the size of the representation from exponential to linear with the number of parents. For instance, $Malaria$ in the 90% of the cases will cause $Fever$, in the last 10% fails to cause $Fever$. This is called the **failure probability**.

The second row of the table shows what we are talking about: we got $Malaria$ but not the other symptoms and the probability of not having ($Fever$) is 0.1.

The main idea about failure probabilities is that each of them is independent from other failure probabilities. So, if we got ($Flu, Malaria$) but not ($Cold$) then the failure probability will be $0.1 \times 0.2 = 0.2$, mixed failure probabilities are solved by the product of each single failure probability.

Generally, Noisy-OR distributions by failure probabilities let to reduce the size of representations from exponential to linear, the initial table has the same number of parameters such as the number of parents present. The failure probabilities let us to solve every question about the domain analyzed. Despite Noisy-OR distribution is a powerful tool, this works only if all the causes are already listed there and all the variables are Boolean.

Definition

Noisy-OR distributions model multiple causes for the same effect if and only if:

- Parents $U_1, \ldots, U_k$ include all causes[a].
- Each failure probability is independent from other failure probabilities.

$$P(X|U_1 \ldots U_k, \neg U_{j+1} \ldots \neg U_k) = 1 - \prod_{i=1}^{j} q_i$$

Where q_i stands for **failure probability** of **i cause**.

[a] If missing, we can add the **leak node**.

Hybrid networks

Many real-world problems involve continuos quantities, such as money, temperature, height and mass; in fact, much of statistics deals with random variables whose domains are continuos. By definition, continuos variables have an infinite number of possible values, so it is impossible to specify conditional probabilities for each value. There are many different ways to handle continuos variables:

- **Discretization**: dividing up the possible values into a set of intervals. Discretization is sometimes an adequate solution, but often results in large errors and the CPTs could be quite huge.
- **Finitely parameterized canonical families**: the most common way is to define standard families of probability density functions that are specified by a finite number of parameters.

A network with both discrete and continuos variables is called **hybrid Bayesian**

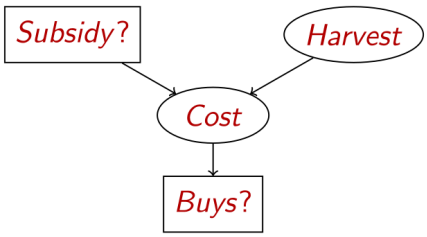
network. To design a hybrid network, we have to specify two new kinds of distributions:

- the conditional distribution for a continuos variable given discrete or continuos parents;
- the conditional distribution for a discrete variable given continuos parents.

Let's set a new example for better understanding.

Example

i.e. Harvest hybrid network.



Consider the network shown previously, in which a customer buys some fruit depending on its cost, which depends in turn on the size of the harvest and from the government's subsidy. The variable $(Cost)$ is continuos and has continuos and descrete parents $(Harvest, Subsidy)$, the variable $(Buys)$ is descrete and it has a continuos parent.

One standard way to represent the first type of relationship is the **Linear Gaussian Model**, where the distribution of $Cost$ is the Gaussian distribution with fixed **variance** and variable **mean**. The mean depends on the value of the **continuos parent** $(Harvest)$, and it can be represented as a linear function of $(Harvest)$. Moreover, the parameters (a, b) of the mean change based on presence or absence of $(Subsidy)$.

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

Before moving on, we focus on some aspects of the equation:

- $(a_* h + b_*)$ defines the mean of the distribution, if h changes also the mean will change[a].
- (σ_*) is the standard deviation, which is fixed, never change.
- Presence or abscence of $(Subsidy)$ determines the choice of (a, b) parameters, but it hasn't any influence on $(Harvest)$ behavior, again, if it encreased the cost could decrease and vice versa.

Now we turn to the distributions for descrete variables with continuos parents. We now consider the *Buys* node. It seems reasonable to assume that the customer will buy if the cost is low and will not buy if it is high and the probability of buying varies smoothly in some intermediate region. In other words, the conditional distribution is like a soft threshold function.

There are two ways to make soft threshold, as:

- **Probit distribution**, uses the integral of the standard normal distribution to compute a threshold.
- **Sigmoid distribution**, uses the *logistic function* to produce a soft threshold[b].

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

[a]The star ($*$) symbols stand for the enumeration of ($Subsidy$).

[b]The sigmoid or logit distribution is widely used in neural networks, more than the probit one.

Chapter 4

Exact Inference

The basic task for any probabilistic system is to compute the conditional probability distribution for a set of **query variables**, given some **evidences**. By the way, we will consider only one query variable at the time, generally named **simple query**, that looks like:

$$\mathbf{P}(X_i|E = e)$$

where

- **P** is the probability distribution.
- X_i defines the query variable, or more simply what we are looking for.
- $E = e$ denotes the set of evidence variables, and e is a particular observed event that belongs to one of the random variables.

We have already seen how to compute posterior probabilities, such as in the burglary network. From the same network, we might ask which is the probability that a burglary occured given $JohnCalls$ and $MaryCalls$ true:

$$\mathbf{P}(Burglary|JohnCalls = True, MaryCalls = True) = \langle 0.284, 0.716 \rangle$$

In this section we discuss a smart way for computing this conditional probability without constructing its explict representation.

4.1 Inference by enumeration

In the previous sections we explained that any conditional probability can be computed by summing terms from the full joint distribution. Specifically, a query like

$\mathbf{P}(X|e)$ can be solve using:

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

where

- α simbolyze the normalization, deduced by the definition of product rule.
- X the query variable.
- e set of evidences, here we consider events that belongs to evidence variables.
- y all the hidden values included inside the full joint distribution.

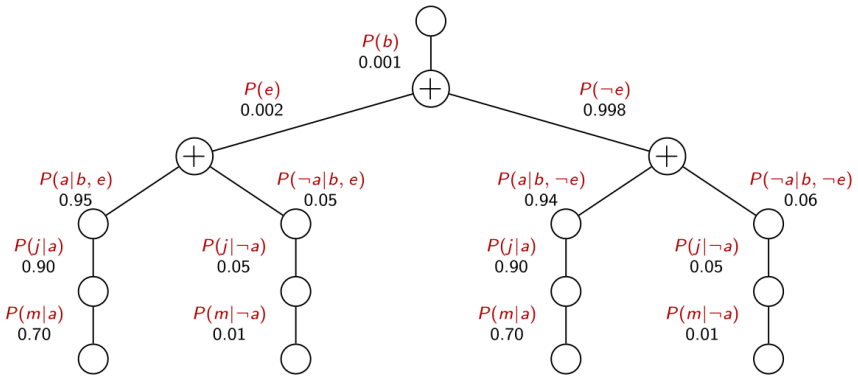
Now, we have discovered that a Bayesian network describes a domain in the same way of a full joint distribution does. More exactly, by Bayesian network we have demonstrated that the previously equation can be written as products of conditional probabilities from network.

Considering the query $\mathbf{P}(Burglary|JohnCalls = True, MaryCalls = True)$, the hidden values are $Earthquake$ and $Alarm$. From the definition of global semantics, the expressions can be notated as follows:

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

Compared to the previous expression, we added four new terms. In the worst case, the complexity of the algorithm for a network with n Boolean variables is $O(n2^n)$. An improvement can be obtained moving the prior probability $P(b)$ outside the summations over a and e, and the prior probability $P(e)$ outside the summation a. Hence, we have:

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



Despite these changes, the complexity of the expression is exponential, $O(2^n)$ - better than $O(n2^n)$, but still quite huge. The main reason complexity is still exponential

is attributable to the presence of repeated computation, or better, the products $P(j|a)P(m|a)$ and $P(j|\neg a)P(m|\neg a)$ are computed twice, once for each value of e.

4.2 Inference by variable elimination

One way to decrease the time complexity is the elimination of repeated calculations. The main idea is simple: do the computation once and save the result for later. There are several versions of this approach; we define the **variable elimination** algorithm.

Variable elimination works by evaluating expressions in **right-to-left** order. Intermediate results are stored, and summations over each variable are done only for those portions of the expression that depend on the variable. By this algorithm the equation becomes:

$$\begin{aligned}
&\mathbf{P}(B|j,m) \\
&= \alpha P(B) \textstyle\sum_e P(e) \sum_a P(a|B,e) P(j,a) P(m,a) \\
&= \alpha f_B(B) \textstyle\sum_e f_E(e) \sum_a f_A(b,e) f_J(a) f_M(a)
\end{aligned}$$

Each part of the expression was annotated with the name of the corresponding **factor**; a factor is a matrix containing values of its argument variables. For instance, the factor $f_J(a)$ correspond to $P(j|a)$, and it's a two-element vector containing the same values of the CPT.

$$f_J(a) = \begin{bmatrix} P(j|a) \\ P(j|\neg a) \end{bmatrix} = \begin{bmatrix} 0.90 \\ 0.05 \end{bmatrix}$$

Even if factors are matrix like representation, the product between two or more of them is not the ordinary matrix multiplication but instead the **pointwise product** operation. The process of evaluation is a process of **summing out** variables, composed by two main steps:

- Move any constant factors outside the summations, as happens for the prior probabilities.
- Add up submatrices in pointwise product of remaining factors, creating new factors.

Let's see an example for better understanding.

Example

i.e. *JohnCalls*, *MaryCalls* and *Alarm* CPTs.

A	$\mathbf{P}(J\|A)$
T	.90
F	.05

A	$\mathbf{P}(M\|A)$
T	.70
F	.01

B	E	$\mathbf{P}(A\|B,E)$
T	T	.95
T	F	.94
F	T	.29
F	F	.001

We just said that each CPTs can be written as a factor - a matrix like representation contaning the same values of the conditional probability.

We are going to sum out A from the product of f_A, f_J and f_M. Before summing out A, we have to solve the pointwise product of the same factors.

$$
\begin{aligned}
&\mathbf{P}(B|j,m)\\
&= \alpha\mathbf{P}(B|j,m)\\
&= \alpha P(b)\sum_e P(e)\sum_a P(e)P(a|e,b)P(j|a)P(m|a)\\
&= \alpha f_B(B)\sum_e f_E(e)\sum_a f_A(b,e)f_J(a)f_M(a)\\
&= \alpha f_B(B)\sum_e f_E(e)\sum_a f_{AJM}(b,e,a)\\
&= \alpha f_B(B)\sum_e f_E(e) f_{\overline{A}JM}(b,e)
\end{aligned}
$$

$$\mathbf{P}(B|j,m) = \alpha f_B(B)\sum_e f_E(e) f_{\overline{A}JM}(b,e)$$

Examing this sequence, we see that the two computational operations are performed: initially we define for each conditional probability its corresponding factor, then we compute the pointwise product of f_A, f_J and f_M (*as shown in the fourth passage*) and finally we sum out A from the new factor ($f_{\overline{A}JM}(b,e)$) created.

In the end we repeat the same process summing out E.

$$
\begin{aligned}
&\mathbf{P}(B|j,m)\\
&= \alpha f_B(B)\sum_e f_E(e) f_{\overline{A}JM}(b,e)\\
&= \alpha f_B(B)\sum_e f_{\overline{A}JME}(b,e)\\
&= \alpha f_B(B) f_{\overline{AE}JM}(b)
\end{aligned}
$$

Before starting the computation, it would be a good idea to remove all irrelevant random variables. According to this last observation, there are some theorems that can help us achieve our goal.

Theorem

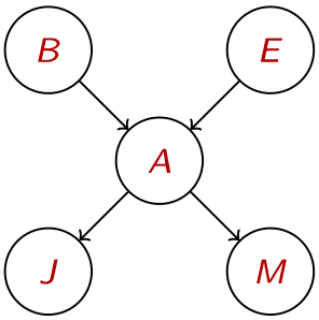
Y is **irrelevant** unless $Y \in Ancestors(X \cup \mathbf{E})$

Theorem

Y is **irrelevant** if *d-separated* from $\{X\}$ by **E**

Example

i.e. Application of previous theorems.



Let's consider the following simple query:

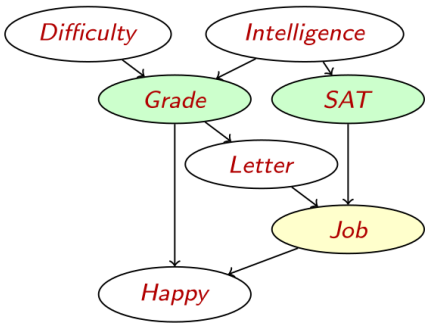
$$P(JohnCalls|Burglary = true)$$

So, which is the probability of $JohnCalls$ given $Burglary$ true? The solution of this kind of query can be written as products of conditional probabilities, such as:

$$P(JohnCalls|Burglary = true)$$
$$= \alpha P(b) \sum_e P(e) \sum_a P(a)P(a|b,e)P(j|a)P(m|a)$$

As you can assume, we have already moved out probabilities ($P(b)$, $P(e)$) that do not belong in the summations. By the way, also another observation we can do: node $MaryCalls$ does not belong to $JohnCalls$'s Ancestors. Therefore, $MaryCalls$ conditional probability should not be considered inside the equation; Ancestors' theorem help us to simplify the final expression. In probabilistic terms these assertions are defined as:

$X = JohnCalls$, $\mathbf{E} = \{Burglary\}$
$Ancestors(\{X\} \cup \mathbf{E}) = \{Alarm, Earthquake\}$ so $MaryCalls$ is **irrelevant**.



Now we switch to the student network, the new simple query is:

$$P(Job|Grade, SAT) =?$$

The second theorem uses the notion of *d-separation*. It could be convenient to pick up on its main concept:

> The nodes X and Y are *d-separated* given Z if there is no *active trail* between X and Y given Z. Z cannot be part of the active trail.

For $P(Job|Grade, SAT)$, not only $Happy$ is irrelevant, it does not belong to $Ancestors(\{Job\} \cup Grade, SAT)$, but also $Difficulty$ and $Intelligence$ are *d-separated* from Job, given $Grade$ and SAT. Accordingly, $Happy$, $Difficulty$ and $Intelligence$ should not be considered in the simple query.

4.3 Complexity of exact inference

The complexity of exact inference depends strictly on the structure of the network. The burglary network belongs to the family of networks in which there is at most one path between any couple of nodes. These are called **polytrees**, and they have a particularly nice property: the complexity of exact inference in polytrees is linear in the size of the network.

Theorem

The complexity of variable elimination is $O(d^k n)$ if the network is a **polytrees** one, in other words any couple of nodes are connected by at most one path.

If the network is not a polytrees one, variable elimination can have exponential complexity in the worst case; generally named as **NP-hard** problem.

Chapter 5

Approximate Inference

Until now, we have used **exact inference** to compute any kind of probability. However, exact inference seems to lose accuracy and efficiency when the Bayesian network analyzed is too huge, especially if it is composed by several continuos variables. So, exact inference is not always the best choice, we could lose some crucial informations.

Therefore it's essential to consider **approximate inference** methods. This section describes some randomized sampling algorithms, that provide approximate answers to our probability query.

The main idea about sampling algorithms can be summarize in three steps, as follows:

- Draw **N** samples from a sampling distribution S.
- Compute an approximate posterior probability P.
- Show this converges to the true probability P.

Moving on, the next paragraphs will describe four sampling algorithms: **Sampling from an empty network**, **Rejection sampling**, **Likelihood weighting** and **Markov chain Monte Carlo**. Each of them guarantee the generation of samples from a well-known probability distribution.

5.1 Sampling from an empty network

This type of algorithm is the simpliest and works only for those Bayesian networks that have no given evidences. The idea is to sample each random variable that

composed the network, following its topological order. Note that the probability distribution from which the value is sampled is conditionated on the values already assigned to the variable's parents.

The algorithm, shown below, generates samples from the prior joint distribution specified by the network. Let's consider a new example for better understanding.

Example

i.e. Sampling by sprinkler network.

From the same network, used to understand the first basic concepts about Bayesian networks, we assume the same topological order, such as:

$$[Cloudy, Sprinkler, Rain, WetGrass]$$

By the sampling algorithm, we have to associate a sample value with each node:

1. $\mathbf{P}(Cloudy)$.
   From the initial CPT, the boolean variable has the same probability to assume $True$ or $False$, $\langle 0.5, 0.5\rangle$. In this case, the algorithm gives us 0.1, so $Cloudy = True$.
2. $\mathbf{P}(Sprinkler|Cloudy = True)$.
   Again, we know that $P(Sprinkler = True|Cloudy = True) = 0.1$ and $P(Sprinkler = False|Cloudy = True) = 0.9$. The method return us 0.2, so $Sprinkler = False$.
3. $\mathbf{P}(Rain|Cloudy = True)$.
   Analyzing the CPT, we have $P(Rain = True|Cloudy = True) = 0.8$ and $P(Rain = False|Cloudy = True) = 0.2$. The sample value is 0.7, the boolean value associated is $True$.
4. $\mathbf{P}(WetGrass|Sprinkler = False, Rain = True)$.
   Another time, we use the conditional probability table to see what happens if are given these evidences. Therefore, $P(WetGrass = True|Sprinkler = False, Rain = True) = 0.9$ and $P(WetGrass = False|Sprinkler = False, Rain = True) = 0.1$, the final value is $True$.

In this case, *Prior-Sample* return us $\langle t, f, t, t\rangle$

From the previous example, we can compute which is the probability that *Prior-Sample* generates a particular event. First, let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by the algorithm, then we have:

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

so, the probability that a specific event is generated is equal to the product of each

conditional probability given the parents of X_i. Accordingly to the definition of *global semantics*, we have:

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

This last observation tell us exactly that the algorithm generates samples from the prior joint distribution!

Generally, the answers are computed by counting the actual samples generated. Suppose there are N total samples, and let $N_{PS}(x_1, \ldots, x_n)$ be the number of times the specific event occurs in the set of samples. Then we have:

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

We expect $\frac{N_{PS}}{N}$ to converge to its expected value according to the sampling probability.

For instance, consider the previously event: *[t,f,t,t]*. The sampling probability for this event is:

$$S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324.$$

Hence, we expect 32.4% of the samples to be of this event. That is, estimates derived from *Prior-Sample* are **consistent**, briefly:

$$P(x_1 \ldots x_n) \approx \frac{N_{PS}(x_1,\ldots,x_m)}{N}$$

5.2 Rejection sampling

Rejection sampling is a general method for computing conditional probabilities. The algorithm, shown below, is very simple, it generates samples from the prior distribution specified by the network, and it discards all those samples that do not match with the given evidences. Finally, the estimated probability $\hat{\mathbf{P}}(X|\mathbf{e})$ is obtained by counting how often $X = x$ occurs in the remaining set of samples.

Example

i.e. Continuing with our example from **Sampling method for empty network** one.

Let us assume the probability distribution of

$$\mathbf{P}(Rain|Sprinkler = True) = ?,$$

using 100 samples. Of the 100 generated, suppose that 73 have $Sprinkler = False$ and are discarded, while 27 have $Sprinkler = True$; of the 27, 8 have

$Rain = True$ and the last 19 have $Rain = False$. Hence,

$$\mathbf{P}(Rain|Sprinkler = True) = \alpha(\langle 8, 19\rangle) = \langle 0.296, 0.704\rangle.$$

The true answer is $\langle 0.3, 0.7\rangle$. As more samples are collected, the estimate will converge to the true probability.

Before moving on, let $\hat{\mathbf{P}}(X|\mathbf{e})$ be the estimated probability that the algorithm returns. From the definition of the algorithm, we have:

$$\hat{\mathbf{P}}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e}) = \frac{\mathbf{N}_{PS}(X,\mathbf{e})}{N_{PS}(\mathbf{e})}.$$

From the previous equation[1], this becomes:

$$\hat{\mathbf{P}}(X|\mathbf{e}) \approx \frac{\mathbf{P}(X,\mathbf{e})}{P(\mathbf{e})} = \mathbf{P}(X|\mathbf{e}).$$

Rejection sampling produces **consistent** estimate of the true probability[2].

However, rejection sampling is affected by a major problem. Previously, we said that the estimated probability will converge to the true probability as the number of samples collected grows. But, as the number of evidence variables grows the fraction of samples ($N_{PS}(\mathbf{e})$) drops exponentially! Therefore, rejection sampling is not a good choice when the Bayesian network seems to be too complex.

5.3 Likelihood weighting

Likelihood weighting avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence **e**. We begin by describing how the algorithm is designed, then we show a simple example to pratically understand how it works.

The algorithm, shown below, fixes the values for the evidence variables **E** and samples only the non-evidence variables. This guarantees that each event generated is consistent with the evidence, avoiding the rejection of all them that cannot be associated with it. However, not all the events are equal. In fact, before start counting how many times an evidence variable occurs in the set of samples, each event is **weighted** by the **likelihood** that the event accords to the evidence. Generally it is computed by the product of the conditional probabilities for each evidence variable, given its parents.

[1] $P(x_1 \dots x_n) \approx \frac{N_{PS}(x_1,\dots,x_m)}{N}$.

[2] Note that we are not comparing the estimated distribution with the prior joint distribution, but instead with the posterior distribution. Here, we are talking about conditional probabilities, not anymore of samples assigned to each random variable that composed the network.

Example

i.e. Sprinkler network.

The query variable, following the topological order, is:

$$\mathbf{P}(Rain|Cloudy = True, WetGrass = True) = ?$$

The process goes as follows: first of all, the weight is set to 1 and then an event is generated:

1. *Cloudy* is an evidence variable with value *true*. Therefore, the *weight* value becomes:
$$w \leftarrow w \times P(Cloudy = True) = 1.0 \times 0.5 = 0.5$$
2. *Sprinkler* is not an evidence variable, we must sample it. Looking to the CPT, we have that $\mathbf{P}(Sprinkler|Cloudy = True) = \langle 0.1, 0.9 \rangle$. Suppose the algorithm returns *true*.
3. *Rain* is not an evidence variable, we have to sample it from $\mathbf{P}(Rain|Cloudy = True) = \langle 0.8, 0.2 \rangle$. Suppose the algorithm returns *false*.
4. $WetGrass$ is an evidence variable with value *true*. Therefore, we set:
$$w \leftarrow w \times P(WetGrass = True|Sprinkler = True, Rain = False)$$
$$= 0.5 \times 0.90 = 0.45$$

Finally, the *likelihood weighting* algorithm returns the event $[t, f, f, t]$ with weight 0.45.

From the sprinkler example, we derive the general form of *likelihood weighting*. Remember that the evidence variables $\mathbf{E}$ are fixed with values $\mathbf{e}$, and all the non-evidence variables, including the query one X, are named $\mathbf{Z}$.

Likelihood weighting algorithm samples each non-evidence variable $\mathbf{Z}$ given its parents, as follows:

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i|parents(Z_i))^3.$$

Unlike the first prior distributions $P(\mathbf{z})^4$, the distribution S_{WS} pays some attention to the evidence: the value sampled for each non-evidence variable Z_i will be influenced by evidence variables included in Z_i's ancestors. For instance, when sampling *Sprinkler* the algorithm pays attention to the evidence $Cloudy = True$, since that *Cloudy* is its parent node. By the way, when sampling *Sprinkler* and *Rain* the algorithm ignores the evidence in the child node $WetGrass = True$; this means will be generated many samples with $Sprinkler = False$ and $Rain = False$ despite

[3] $Parents(Z_i)$ can include both non-evidence variables and evidence variables.

[4] $\mathbf{P}(Cloudy)$ is a prior distribution, no evidence influences its behavior.

they are against the $WetGrass$ evidence[5].

But, which is the role of w in the likelihood weighting algorithm? The weight w makes up for the difference between the true probability and the estimated one. Without w term, the estimated probability would be flawed. Moving on, the weight for a given sample $(\mathbf{z}, \mathbf{e})$ is:

$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i | parents(E_i))$$

Finally, mixing the two equations we get **consistent** probabilities from likelihood sampling:

$$\begin{aligned} & S_{WS}(\mathbf{z}, \mathbf{e}) w(\mathbf{z}, \mathbf{e}) \\ & = \prod_{i=1}^{l} P(z_i | parents(Z_i)) \prod_{i=1}^{m} P(e_i | parents(E_i)) \\ & \approx P(\mathbf{z}, \mathbf{e}) \end{aligned}$$

Despite likelihood weighting is more efficient than rejection sampling, also this algorithm suffers the growth of evidence variables. As more evidence variables are given the algorithm's performance decrease, only few samples can reach the total weight.

5.4 Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo (*MCMC*) algorithms work quite differently from rejection sampling or likelihood weighting. Instead of creating each time a new sample, MCMC algorithms generate each sample by making a random change to the preceding sample. It can be helpful to think of an MCMC algorithm as a set of worlds[6], where we are in a *current state* specifying a value for every variable and generate a *new state* by making random changes to the current state.

Every sample defined in that way is meaningful, but how to do design them? To answer this query, we need to change something seen so far, and in this case we have to redefine the way for drawing samples.

Until now, every method studied suffers from one main problem: estimated distributions never converge to true distributions in the long run, expecially when many samples are generated. This last observation can be an useful hint for understanding the **Gibbs sampling**.

[5]This is one of the main problem of **likelihood weighting** algorithm, it does care about the evidences included only in the ancerstors sets.

[6]All the several worlds are not *atomic event* but only *states*.

Given the image shown below, the main idea of Gibbs sampling consists of drawing all the states which represent every single change for any non-evidence variable. As likelihood sampling, evidence variables are fixed.

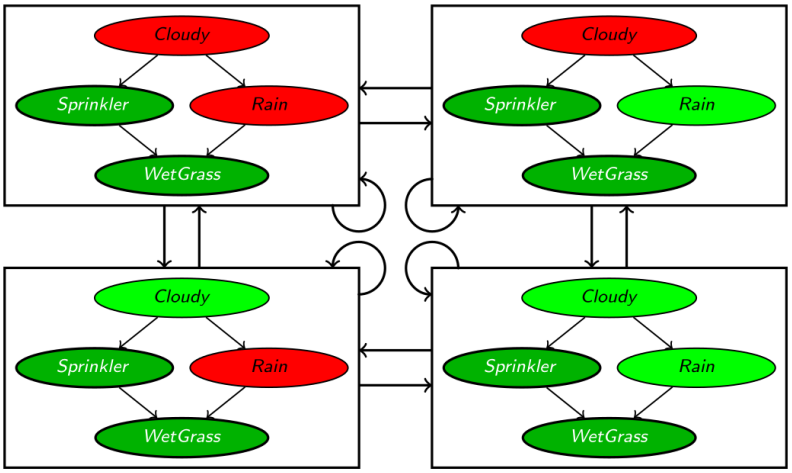
Moving on, we suppose that our current state is:

$$\langle Cloudy = True, Sprinkler = True, Rain = False, WetGrass = True \rangle.$$

To make a new sample, the algorithm focuses on one non-evidence variable at a time, in this case *Rain*, and it generates a new related value. The next state will be the same if *Rain* is again *false*, or it would be

$$\langle Cloudy = True, Sprinkler = True, Rain = True, WetGrass = True \rangle$$

if the sampled value is *true*.



The sampling for the non-evidence variable X_i is done by looking its *Markov blanket*[7]. Therefore, Gibbs method samples only the Markov blanket associated, without any time recompute all the random variables that composed the network. Probabilities given the Markov blanket are calculated as follows:

$$P(x_i|mb(X_i)) = \alpha P(x_i|parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j|parents(Z_j)).$$

Before contuining, let's consider the terms that made up the above equation:

- $P(x_i|parents(X_i))$ defines the probability of the same variable x_i. In this case, all parents of x_i are included inside $parents(X_i)$.

[7]Given a random variable X_i its **Markov blanket**, X_i is conditionally independent of all other nodes in the network.

- $\prod_{Z_j \in Children(X_i)} P(z_j|parents(Z_j))$ computes each child's probability of x_i. Included in $parents(Z_j)$ are all the other parents of x_i's children.

Example

i.e. Example of Markov blanket sampling of the sprinkler network.

Given the Markov blanket of $Rain$

$$\langle Cloudy, Sprinkler, WetGrass \rangle$$

the result of the $P(Rain|mb(Rain))$ is equal to:

$$P(Rain|mb(Rain)) = \alpha P(Rain|Cloudy)P(WetGrass|Sprinkler, Rain).$$

Although computing the probabilities given Markov blanket may seem quite time consuming, this algorithm is more efficient than likelihood weighting or rejection sampling. All Markov blanket probability distributions are always the same, and focusing only on Markov blanket allows us to avoid computing the whole graph. Therefore, every new sample already has the pre-calculated distributions because we change the *current state* sampling just one random variable, leaving the others fixed.