Unit 7: Representation and reasoning with uncertain knowledge

F.J. Martín Mateos, J.L. Ruiz Reina,
A. Riscos Núñez

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

Limitations of rule-based knowledge

$$\begin{array}{l} \textsc{If}\ \textit{Symptom} = \textit{toothache} \\ \textsc{Then}\ \textit{Diagnosis} = \textit{caries} \end{array}$$

- Is this **correct** knowledge?
- Maybe a more exhaustive knowledge would be better:

$$\begin{array}{ll} \textsc{If} & \textit{Symptom} = \textit{toothache} \\ \textsc{Then} & \textit{Diagnosis} = \textit{caries}\ \textsc{Or} \\ & \textit{Diagnosis} = \textit{sinusitis}\ \textsc{Or} \\ & \textit{Diagnosis} = \textit{wisdom tooth}\ \textsc{Or} \\ \ldots & \end{array}$$

Limitations of rule-based knowledge

- Why can't we rely on precise and exact knowledge?
- Possible causes:
  - Too hard to be exhaustive
  - Theoretical ignorance: lack of complete information on the field
  - Practical ignorance: even if we know the rules, we cannot apply them (unknown facts)

Alternative: Degrees of Belief

- We *believe*, based on our *observations*, that a patient who suffers toothache has a cavity (caries) with 80% probability.
- In probability theory, this belief is expressed as follows: $P(Caries = true | Toothache = true) = 0.8$
- Degrees of **belief** $\neq$ degrees of **truth**
- Probability is subject to **change**, when observing new *evidence*
- The *probability theory* will be the framework to represent uncertain knowledge

Random variables

- A random variable captures knowledge of a “part” of the world that we might be uncertain about
  - Example: random variable *Caries* describes the fact that a patient might have caries (or not)
  - Our description of the world will be given by a set of random variables
- A random variable can take a value from its *domain*
  - Example: possible values for *Caries* are *true* and *false*
- Notation: random variables capitalized and their values in lower case

Random variables

Types of random variables

Boolean (notation: *caries* and $\neg$*caries* are equivalent to *Caries* = *true* and *Caries* = *false*, respectively)

Discrete (including Boolean)

Continuous

- In what follows, we focus on discrete variables

Propositions

- We can express propositions in terms of random variables and connectives from propositional logic

Examples

- $caries \wedge \neg toothache$
- $Caries = true \vee Weather = cloudy$
- $Roll1 = 5 \wedge (Roll2 = 4 \vee Roll2 = 5 \vee Roll2 = 6)$

- **Probabilities** assigned to propositions will represent our degree of **belief** on them

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

Intuitive idea

- Given a proposition *a*, its unconditional (or *prior*) probability, denoted by $P(a)$, quantifies the degree of belief that *a* holds, in the absence of any other information or evidence
  - Example: $P(caries) = 0,1, P(caries, \neg toothache) = 0,05$
  - Notation: $P(caries, \neg toothache)$ is equivalent to $P(caries \wedge \neg toothache)$
  - Frequence approach: number of positive cases (when *a* holds) divided by total number of cases

Probability: axiomatic definition

- A probability function is a function defined over the set of propositions (wrt a given set of random variables), verifying the following properties:
  - $0 \leq P(a) \leq 1$, for every proposition a
  - $P(\mathit{true}) = 1$ and $P(\mathit{false}) = 0$ where *true* and *false* represent any tautological or insatisfiable proposition, respectively
  - $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$, for any pair of propositions a and b
- Probability calculus is built on this three axioms. For example:
  - $P(\neg a) = 1 - P(a)$
  - $P(a \vee b) = P(a) + P(b)$, if a and b are disjoint.
  - $\sum_{i=1}^{n} P(D = d_i) = 1$, where D is a r.v. and $d_i, i = 1, \ldots, n$ are its possible values

Probability distributions

- The probability distribution of a random variable indicates the probability for this variable to take each of its possible values
  - Example: if *Weather* is a r.v. with values *rain*, *sunny* , *cloudy* and *snow*, its probability distribution could be:

$$\begin{aligned} P(Weather = sunny) &= 0,7 \\ P(Weather = rain) &= 0,2 \\ P(Weather = cloudy) &= 0,08 \\ P(Weather = snow) &= 0,02 \end{aligned}$$

- Notation: we will use **P** (in bold face), to express in a compact way a probability distribution (for a fixed order on the values)
  - Example:

$$\mathbf{P}(Weather) = \langle 0,7; 0,2; 0,08; 0,02 \rangle$$

Joint distribution

- *Joint* probability distribution: probability of each one of the possible combinations of values for $n \geq 2$ random variables
  - Notation: $\mathbf{P}(X, Y)$ expresses in a compact way a table with those probabilities
  - Example: $\mathbf{P}(Weather, Caries)$ denotes a 4×2 size table

Atomic events

- Given a set of variables describing our "world", an *atomic event* is a particular type of proposition:
  - Conjunction of elementary propositions, showing a precise value for all of the variables
- Example: if we only have *Caries* and *Toothache* as random variables in our world description, then the possible atomic

- $caries \wedge toothache$
- $caries \wedge \neg toothache$
- $\neg caries \wedge toothache$
- $\neg caries \wedge \neg toothache$

Characteristics of atomic events

- Mutually exclusive
- Exhaustive (some atomic event *must* occur)
- An atomic event determines which propositions are *true* and which are *false*
- Any proposition is equivalent to the disjunction of a set of atomic events.

  for example, *caries* is equivalent to

  $(caries \wedge toothache) \vee (caries \wedge \neg toothache)$
- For any proposition a,

$$P(a) = \sum_{e_i \in \mathbf{e}(a)} P(e_i)$$

where $\mathbf{e}(a)$ is the set of atomic events whose disjunction is equivalent to a

Full joint distribution

- *Full joint* distribution (FJD):
  probability of each one of the atomic events
  - A FJD is a complete specification (in probabilistic terms) of the described domain

Example of a FJD:

	toothache	*toothache*	¬*toothache*	¬*toothache*
	cavity	¬*cavity*	*cavity*	¬*cavity*
caries	0.108	0.012	0.072	0.008
¬*caries*	0.016	0.064	0.144	0.576

Probability calculus using FJD

- Relies on the formula $P(a) = \sum_{e_i \in \mathbf{e}(a)} P(e_i)$

Examples:

- $P(caries \vee toothache) =$
  $P(caries, toothache, cavity) + P(caries, toothache, \neg cavity) +$
  $P(caries, \neg toothache, cavity) + P(caries, \neg toothache, \neg cavity) +$
  $P(\neg caries, toothache, cavity) + P(\neg caries, toothache, \neg cavity) =$
  $0,108 + 0,012 + 0,072 + 0,008 + 0,016 + 0,064 = 0,28$
- $P(caries) =$
  $P(caries, toothache, cavity) + P(caries, toothache, \neg cavity) +$
  $P(caries, \neg toothache, cavity) + P(caries, \neg toothache, \neg cavity) =$
  $0,108 + 0,012 + 0,072 + 0,008 = 0,2$

Probability calculus using FJD

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

- Notation:
  - $\mathbf{Y}$ is a vector of r.v.s, representing any possible combination of values
  - $\mathbf{z}$ represents precise combinations of values for the set $\mathbf{Z}$ of r.v.s (remaining unassigned variables)
  - There is a value $\mathbf{P}(\mathbf{Y}, \mathbf{z})$ in the sum for each $\mathbf{z}$, and each of them appears in the FJD
- Problems:
  - We very rarely know the probabilities of all atomic events
  - Exponential size of the FJD
- Conditional probabilities are a better option for expressing our knowledge of the domain

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

Conditional probability

- *Conditional* (or *a posteriori*) probability of a given b (a and b propositions):
  - Denoted as $P(a|b)$, it quantifies the degree of belief that a holds, given that *all that we know* is that b holds.
  - Example: $P(caries|toothache) = 0.8$ means that once we know that a patient has toothache (*and we only know that*), our degree of belief about the patient having caries is 0.8.

Conditional probability

- Mathematical definition of conditional probability:

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

- Alternative: $P(a \wedge b) = P(a|b)P(b)$ (*product rule*)
  - Or, analogously, $P(a \wedge b) = P(b|a)P(a)$
- Notation $\mathbf{P}(X|Y)$ to denote a table of all conditional probabilities of any value of X given any value of Y
  - Product rule in compact form: $\mathbf{P}(X,Y) = \mathbf{P}(X|Y)\mathbf{P}(Y)$
  - Note: the above formula *does not express* a matrix product, but the existing relation between the corresponding entries of the tables

Conditional probability *vs* logical implication

- Conditional probabilities formalize the fact that the degrees of belief are updated when we know additional evidences in our uncertain world
- Conditional probabilities *are not* logical implications with uncertainty
  - $P(a|b) = 0.8$ is not the same as saying "whenever b holds, then $P(a) = 0.8$"
  - Since $P(a|b)$ reflects that b is *the only* known evidence

Probabilistic inference

- By *probabilistic inference* we mean computing the probability of a given proposition, *conditioned* by the observation of some evidences
  - That is, computing probabilities of the form $P(a|b)$ where a is the *query* proposition and b is the *observed* proposition (or *evidence*)
  - The *knowledge base* will be given by a FJD (represented in an *efficient* way, from some conditional probabilities, as we will see)
- The main goal in this unit is to study probabilistic inference algorithms

Probabilistic inference from a FJD

- In principle, we can compute *conditional probabilities* using the FJD
- For example: probability of having *caries*, if we have observed *toothache*

$$P(caries|toothache) = \frac{P(caries \wedge toothache)}{P(toothache)} =$$

$$= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6$$

Normalization

- We could avoid to explicitly compute $P(toothache)$
- $\frac{1}{P(toothache)}$ can be seen as a constant *normalizing* the distribution $\mathbf{P}(Caries, toothache)$ ensuring that it adds up to 1:

$$\mathbf{P}(Caries|toothache) = \alpha\mathbf{P}(Caries, toothache) =$$

$$= \alpha[\mathbf{P}(Caries, toothache, cavity) + \mathbf{P}(Caries, toothache, \neg cavity)] =$$

$$= \alpha[\langle 0.108; 0.016\rangle + \langle 0.012; 0.064\rangle] = \alpha\langle 0.12; 0.08\rangle = \langle 0.6; 0.4\rangle$$

- That is, we first compute $P(caries, toothache)$ and $P(\neg caries, toothache)$ and after that, we multiply by the constant α ensuring that they sum 1; in this way, we have $P(caries|toothache)$ and $P(\neg caries|toothache)$

Probabilistic inference from a FJD

- In general, given a random variable X and a set of observed variables $\mathbf{E}$, with oberved values (or *evidences*) $\mathbf{e}$), we have:

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

  - It is an equality between "vectors" (one component for each of the possible values of X)
  - There is an addend for each combination $\mathbf{y}$ of values of non-observed variables $\mathbf{Y}$
  - For each value of x of X, each addend $\mathbf{P}(x,\mathbf{e},\mathbf{y})$ is an entry of the FJD
  - α is a normalization constant, ensuring that the probability distribution adds up to 1

Probabilistic inference

- If we have a FJD, the previous formula gives us a method to do probabilistic inferences
- From a practical pint of way, there is an important drawback with this approach: exponentiality
  - With n variables, time complexity is $O(2^n)$
  - In a real problem we may have hundreds or thousands of variables
- However, we will see that the possible *independence relationships* between the random variables that describe our world, may help a lot in reducing the size of the domain representation

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

Probabilistic independence

- Intuitively, two random variables are independent if knowing the value taken by one of them does not update (neither increasing nor decreasing it) our degree of belief about the values that the other variable may take.
  - Assuming indepndence between two variables is usually based on previous knowledge about the modeled domain
- Formally, two random variables X and Y are *independent* if $\mathbf{P}(X|Y) = \mathbf{P}(X)$ (equivalently, $\mathbf{P}(Y|X) = \mathbf{P}(Y)$ or $\mathbf{P}(X,Y) = \mathbf{P}(X) \cdot \mathbf{P}(Y)$)
  - In general, two propositions a and b are independent if $P(a|b) = P(a)$
- Asumming independence between variables in our representation makes the representation more manageable
  - Reduce exponentiality (*factorizing* the problem)

Conditional independence

- Nevertheless, in our example *Toothache* and *Cavity are not independent*
  - Both depend on *Caries*
- But they are independent *once we know* the value of *Caries*
  - That is: $\mathbf{P}(Toothache|Cavity, Caries) = \mathbf{P}(Toothache|Caries)$ or equivalently $\mathbf{P}(Cavity|Toothache, Caries) = \mathbf{P}(Cavity|Caries)$
  - Also equivalent: $\mathbf{P}(Cavity, Toothache|Caries) = \mathbf{P}(Cavity|Caries)\mathbf{P}(Toothache|Caries)$

Conditional independence

- Intuitively: X is conditionally independent of Y given a set of variables $\mathcal{Z}$, if our degree of belief on X taking a given value, knowing the values taken by the variables in $\mathcal{Z}$, would not be updated (neither increasing nor decreasing it), if in addition we knew the value of Y
    - In the example: if we already know if someone has *Caries* or not, then knowing also if it has *Toothache* or not will not modify our degree of belief of having a *Cavity* or not.
- Formally, two r.v. X and Y are independent given a set of r.v. $\mathcal{Z}$ if $\mathbf{P}(X, Y|\mathcal{Z}) = \mathbf{P}(X|\mathcal{Z})\mathbf{P}(Y|\mathcal{Z})$
    - Or equivalently $\mathbf{P}(X|Y, \mathcal{Z}) = \mathbf{P}(X|\mathcal{Z})$ ó $\mathbf{P}(Y|X, \mathcal{Z}) = \mathbf{P}(Y|\mathcal{Z})$

Conditional independence

- Conditional independece between some variables is key for efficently storing the FJD. For example:

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

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

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

  - Instead of a table with seven independent entries, we only need 5 independent entries (in three tables)

Conditional independence

- If *Caries* had n independent effects (given *Caries*), the size of the representation of the FJD grows $O(n)$ instead of $O(2^n)$
- In general, one *Cause* with n independent effects E_i (given *Cause*), we have

$$\mathbf{P}(Cause, E_1, \dots, E_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(E_i | Cause)$$

- These strong indepedence relationships do not hold always, but sometimes it is worth assuming them.
- In general, the *cause-effect* relationships and the *conditional independences* may hold at several levels between the r.v. of a domain. Knowing these relationships is essential for an efficient repesentation of the domain, as we will see.

Outline

1 Uncertainty and knowledge. Random variables

2 Unconditional probability

3 Conditional probability: probabilistic inference

4 Independence. Conditional independence

5 Bayesian networks

6 Constructing bayesian networks

7 Deducing conditional independences

8 Exact inference on bayesian networks

9 Approx. inference on bayesian networks

10 Learning a bayesian classifier: naive Bayes

The alarm (Pearl, 1990):

- We have a security alarm installed in a house
- The alarm is fairly good at detecting a burglary (but it is not perfect)
- Also it may go off due to minor earthquakes
- We have two neighbours, John and Mary, who promised to call the police if they hear the alarm
  - But John or Mary, or both, could miss the alarm and not call the police (if they have loud music at their homes, for example)
  - It may happen also, although it is unlikely, that they call the police but the alarm did not go off (for example, if they confuse it with the telephone ringing)

Modeling the alarm example

- Uncertainty in this situation makes advisable a probabilistic model
- Random variables (all boolean):
  - *Robbery*: a burglar is in my home
  - *Earthquake*: a minor earthquake has occurred
  - *Alarm*: the alarm has gone off
  - *John*: John calls the police
  - *Mary*: Mary calls the police
- Independence relations:
  - *Robbery* and *Earthquake* are independent
  - Given *Alarm*, *John* is conditionally independent from the rest of variables. The same for *Mary*.
    - That is, knowing if the alarm has gone off or not, our degree of belief about the fact that John (or Mary) calls the police, would not be updated if in addition we knew the value taken by any other variable.

Factoring the FJD in the alarm example

- Apply the product rule several times in a given order, and taking into account the conditional independence relationships, we have:

$$\mathbf{P}(John, Mary, Alarm, Robbery, Earthquake) =$$

$$= \mathbf{P}(John|Mary, Alarm, Robbery, Earthquake) \cdot \mathbf{P}(Mary|Alarm, Robbery, Earthquake)$$

$$\cdot \mathbf{P}(Alarm|Robbery, Earthquake) \cdot \mathbf{P}(Robbery|Earthquake) \cdot \mathbf{P}(Earthquake) =$$

$$= \mathbf{P}(John|Alarm) \cdot \mathbf{P}(Mary|Alarm) \cdot \mathbf{P}(Alarm|Robbery, Earthquake) \cdot$$

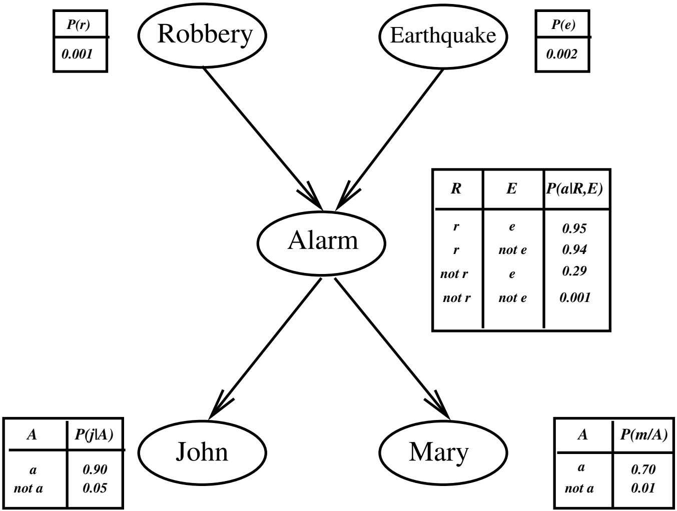
$$\cdot \mathbf{P}(Robbery) \cdot \mathbf{P}(Earthquake)$$

- Observations:
  - Instead of storing the FJD, we only to store five smaller tables
  - This smaller tables are *conditional* probability distributions, describing cause-effect relationships, which usually are more natural for the expert

Bayesian networks

- In general, conditional independence relationships allows to dramatically simplify a FJD, making them of practical use
- *Bayesian networks* (or *belief networks*) are a graphical and compact way to represent uncertain knowledge, based on this idea

Bayesian network in the alarm example (J. Pearl)



P(r)
0.001

P(e)
0.002

R	E	P(a|R,E)
r	e	0.95
r	not e	0.94
not r	e	0.29
not r	not e	0.001

A	P(j|A)
a	0.90
not a	0.05

A	P(m/A)
a	0.70
not a	0.01

Bayesian networks

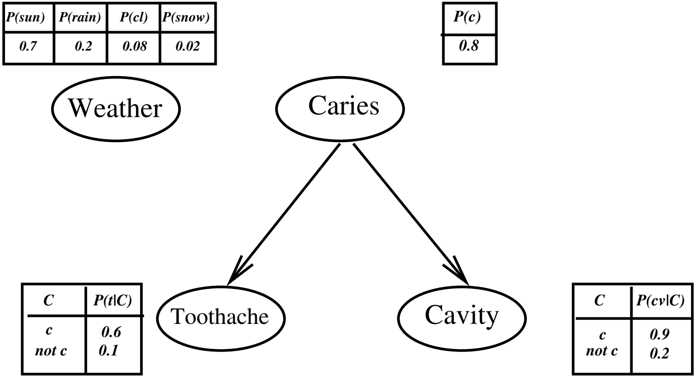
A bayesian network is *directed acyclic graph* with the following components:

- A set of *nodes*, one for each random variable of the "world" represented
- A set of *directed arcs* connecting nodes
  - If there is an arc connecting X with Y, we say that X is a *parent* of Y ($parents(X)$ denotes the set of parents variable of X)
  - From the concept of parent variable, we define also the concepts of *ancestors* and *descendants* of a random variable.
- Each node X_i has an associated *conditional probability distribution (CPT)* $\mathbf{P}(X_i|parents(X_i))$
  - If X_i is boolean, we usually ommit the probability of the *false* value.

Bayesian networks: intuition

- Intuitively, if we know the values taken by the parents of a variable, then our degree of belief about the variable taking a given value, would not be updated if in addition we knew the value taken by any other non-descendant variable
- Formally, any variable X is conditionally independent of its non-descendants, given $\mathit{parents}(X)$.
- The domain expert task is to decide which conditional independence relationships hold (or, in other words, the *topology of the network*).

The caries example as a bayesian network(Russell y Norvig)



P(sun)	P(rain)	P(cl)	P(snow)
0.7	0.2	0.08	0.02

P(c)
0.8

C	P(t|C)
c	0.6
not c	0.1

C	P(cv|C)
c	0.9
not c	0.2

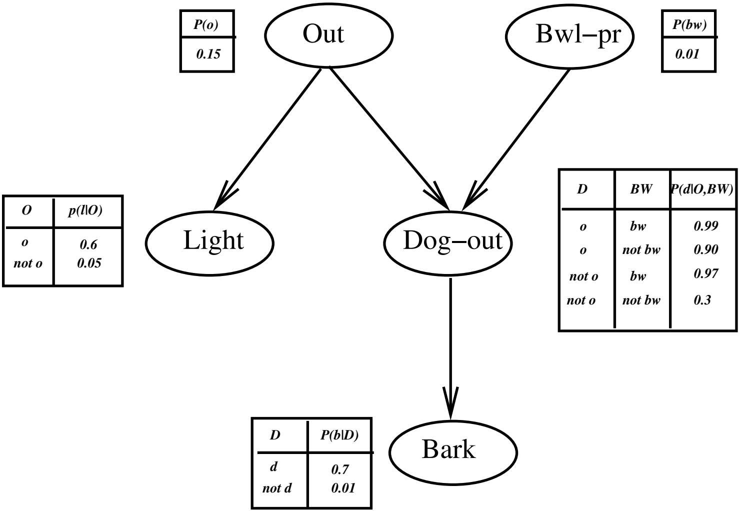
Observation on the caries example

- The topolgy of the previous network expresses that :
  - *Caries* has a *direct influence* on *Toothache* and *Cavity*
  - *Toothache* and *Cavity* are conditionally independent given *Caries*
  - *Weather* is independent of the rest of variables
- Since the vaiables are boolean we do not need to explicitly give the probabilities of $\neg caries$, $\neg toothache$, ...

Example: family out (Charniak, 1991):

- Suppose I want to model a situation that would allow me to know when my family is at home, based on the folowing information:
  - If my wife is not at home, usually (but not always) turns on the light of the entrance; but also she turns the light on in some other circumstances
  - If there is nobody at home, the dog is out
  - If the dog has bowel problems, usually he is also out
  - If the dog is out, I can hear him barking
  - It may be the case that I hear barking, but it is not my dog
- Random (boolean) variables in this problem:
  - *Out* (nobody at home), *Light* (entrance light turned on), *Dogout* (my dog is out), *Bwlpr* (my dog has bowel problems) y *Bark* (I hear a dog barking)

Bayesian network in the “family out” problem



P(o)
0.15

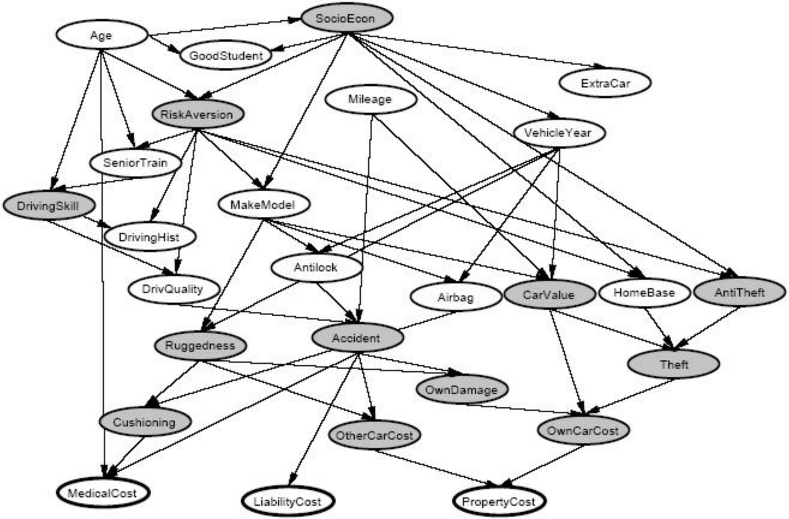
P(bw)
0.01

O	p(l|O)
o	0.6
not o	0.05

D	BW	P(d|O,BW)
o	bw	0.99
o	not bw	0.90
not o	bw	0.97
not o	not bw	0.3

D	P(b|D)
d	0.7
not d	0.01

Car insurance (Binder et al.)



Bayesian networks represent FJD

- Consider a bayesian network with n random variables
  - In a given ordering: $X_1, \ldots, X_n$
- Assume that:
  - $parents(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$ (this condition is satisfied by any order consistent with the arcs in the graph).
  - $\mathbf{P}(X_i|X_{i-1}, \ldots, X_1) = \mathbf{P}(X_i|parents(X_i))$ (that is, any variable is conditionally independent of its non-descendants, given its parents).
- This conditions formally state our intuition when we rpresent our "world" by the bayesian network
  - For example, the alarm network establishes that $\mathbf{P}(Mary|John, Alarm, Earthquake, Robbery) = \mathbf{P}(Mary|Alarm)$

Bayesian networks represent FJD

- Under the previous conditions, and repeatedly applying the product rule:

$$\mathbf{P}(X_1,\dots,X_n) = \mathbf{P}(X_n|X_{n-1}\dots,X_1)\mathbf{P}(X_{n-1}\dots,X_1) = \dots$$

$$\dots = \prod_{i=1}^{n}\mathbf{P}(X_i|X_{i-1},\dots,X_1) = \prod_{i=1}^{n}\mathbf{P}(X_i|padres(X_i))$$

- A bayesian network *represents the FJD* given by $\mathbf{P}(X_1,\dots,X_n) = \prod_{i=1}^{n}\mathbf{P}(X_i|padres(X_i))$
  - An example: the probability that the alarm has gone off, both John and Mary call the police, but nothing has happened is:

$$P(j,m,a,\neg r,\neg e) = P(j|a)P(m|a)P(a|\neg r,\neg e)P(\neg r)P(\neg e) =$$

$$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 = 0.00062$$

Compact representations

- *Locally structured* domains:
  - Existing independence relationships betweem the variables of a domain make the bayesian networks a compact and efficient representation for the FJD, much more than a table with all the probabilities of the atomic events
- In addition, for a domain expert is usually more natural to give conditional probabilities, instead of the joint probabilities of the FJD.

Compact representations

- With n variables, if every variable is directly influenced by k variables at most, then a bayesian network would need $n \cdot 2^k$ numbers, instead of 2^n numbers neededby the FJD.
  - For example, for $n = 30$ and $k = 5$, we would need 960 numbers, instead of 2^{30} (billions)
- Sometimes is worth to assume conditional independence (even if it is not complete), to be able to manage the representation of the world

Bayesian networks

In the following, we will deal with the following issues:

- Constructing the network
    - Obtaining a good (approximately) and compact model of our world
- Deduce conditional indpendences from the network, beyond the independences assumed to construct the network (graphical criterion of *d-separation*)
- Probabilistic inferences (computing probabilities of the form $\mathbf{P}(X|\mathbf{e})$
    - Exact infernce: *enumeration* and *variable elimination* algorithms
    - Approximate inference: *rejection sampling* and *likelihood weighting* algorithms
- Lerning a simple bayesian network from examples and using it as a classifier: *naive Bayes* earning algorithm

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

An algorithm to construct a bayesian network

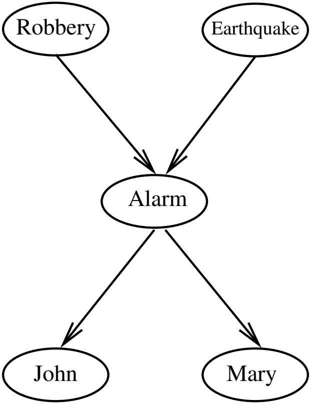
- We are given set of random **VARIABLES** representing a kowledge domain (with uncertainty)

Method for constructing a bayesian network

```
FUNCTION CONSTRUCT_NETWORK(VARIABLES)
1. Let ($X_1$,...,$X_n$) an ordering of VARIABLES
2. Let NET be a bayesian network, initially ``empty''
3. FOR $i = 1, \ldots, n$:
   3.1 Add a new node labaled $X_i$ to NET
   3.2 Let parents($X_i$) a minimal subset of {$X\_i - 1''$,...,$X_1$}
       such that $X_i$ and each element of {$X\_i - 1''$,...,$X_1$}
       are conditionally independent, given parents($X_i$)
   3.3 Add to NET a directed arc between any element in
       parents($X_i$) and $X_i$
   3.4 Assign to node $X_i$ the conditional probaility table
       $P$($X_i$|parents($X_i$))
4. Return RED
```

Example: construction of the alarm network

- Using the order *Robbery*, *Earthquake*, *Alarm*, *John*, *Mary*, and applying the previous algorithm, we obtain:

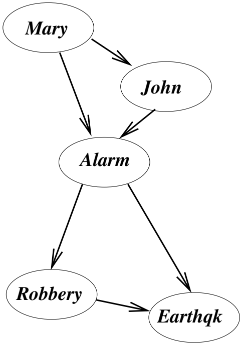


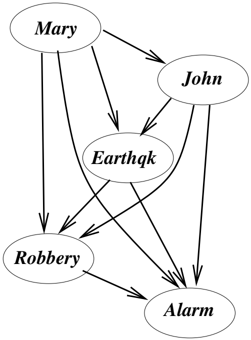
Construction of bayesian networks

- Problem: choose the right ordering
  - In general, we should start with the “root causes”, following with those directly influenced by them, and so on..., until “final effects”, variables not influencing any other variable
  - This will give us a *causal model*, with the conditional probablities in the tables expressing “causes”, rather than “diagnosis’.
  - This is usually preferred by the experts

Construction of bayesian networks

- A bad ordering may lead us to less efficient representations
- Example: network in the left (*Mary*, *John*, *Alarm*, *Robbery* and *Earthquake*) and network in the rigth (*Mary*, *John*, *Earthquake*, *Robbery* and *Alarm*)





Outline

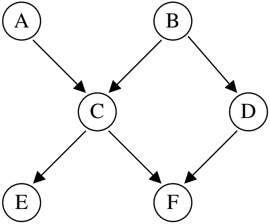
1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

Conditional independence in bayesian networks

- Once the network has been constructed (assuming a number of conditional independence relationships), we can *deduce* additional (conditional) independence relationships
  - *d-separation* provides a graphical criterion to deduce these new coditional independences
- Given a bayesian network, we say that a set of random variables $\mathcal{Z}$ *d-separates* two random variables $\mathbf{X}$ and $\mathbf{Y}$ if every undirected path in the network, connecting $\mathbf{X}$ and $\mathbf{Y}$ holds some of the following conditions:
  - The path contains $\mathbf{Z} \in \mathcal{Z}$ with arcs in the same direction $(\ldots \longleftarrow \mathbf{Z} \longleftarrow \ldots, \ldots \longrightarrow \mathbf{Z} \longrightarrow \ldots)$ or a with a *peak* $(\ldots \longleftarrow \mathbf{Z} \longrightarrow \ldots)$
  - The path contains $\mathbf{Z} \notin \mathcal{Z}$ with a *valley* $(\ldots \longrightarrow \mathbf{Z} \longleftarrow \ldots)$ and any descendant of $\mathbf{Z}$ is in $\mathcal{Z}$
- Theorem: In a bayesian network, if $\mathcal{Z}$ d-sparates X and Y, then X and Y are conditionally independent given $\mathcal{Z}$.

Conditional independence in bayesian networks

- Examples



- A is (conditionally) independent of B given $\emptyset$
- E is conditionally independent of F given $\{C\}$
- $\{F\}$ does not d-separate A and D
- A is not conditionally independent of D given $\{C\}$
- $\{F\}$ does not d-separate E and D.
- D is conditionally independent of E given $\{C\}$

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

Probabilistic inference on bayesian networks

- Probabilistic inference:
  - Compute the *a posteriori* probability for a *query variable*, given the observed values of a set of *evidence variables*
  - For example, the probability of a robbery, knowing that both John and Mary called the police
  - That is, compute $\mathbf{P}(Robbery|john, mary)$

Probabilistic inference on a bayesian network

- Notation:
  - X denotes the query variable
  - **E** denotes a set of *evidence variables* $E_1, E_2, \ldots, E_n$ and **e** denotes a concrete value for each of these variables (an observation)
  - **Y** denotes the rest of variables and **y** is a concrete value for each of these variables

Inferencia by enumeration

- Recall the formula:

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

- This formula is our starting point:
    - Since a bayesian network is representation of a FJD, we are able to compute any a posteriori probability from the information we have in the bayesian network
    - Essentially, we will have a sum of products of entries of the probability tables of the network

An example of probabilistic inference

- The alarm example (for the sake of simplification, we use initials):

$$\mathbf{P}(R|j,m) = \langle P(r|j,m); P(\neg r|j,m)\rangle =$$

$$= \alpha\langle \sum_e \sum_a P(r,e,a,j,m); \sum_e \sum_a P(\neg r,e,a,j,m)\rangle =$$

$$= \alpha\langle \sum_e \sum_a P(r)P(e)P(a|r,e)P(j|a)P(m|a);$$
$$\sum_e \sum_a P(\neg r)P(e)P(a|\neg r,e)P(j|a)P(m|a)\rangle$$

An example of probabilistic inference

- In this example we need to compute a sum with 2 × 4 addends, each of them being a product of five numbers taken form the probability tables of the network
  - In the worst case, with n boolean variables, this computation is $O(n2^n)$
- A first improvement is to take out common factors, with those probabilities only involving variables not appearing in the summation:

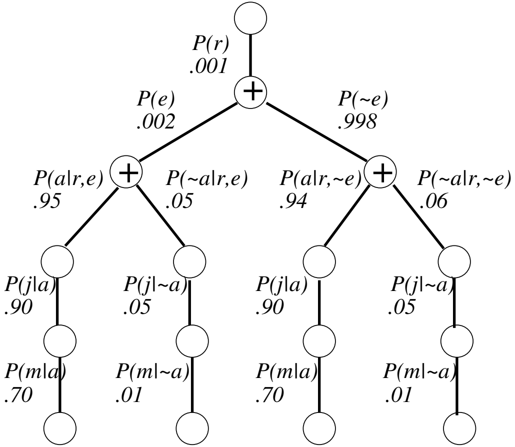
$$\mathbf{P}(R|j,m) = \alpha\langle P(r)\sum_e P(e)\sum_a P(a|r,e)P(j|a)P(m|a);$$

$$P(\neg r)\sum_e P(e)\sum_a P(a|\neg r,e)P(j|a)P(m|a)\rangle =$$

$$= \alpha\langle 0.00059224; 0.0014919\rangle = \langle 0.284; 0.716\rangle$$

Inference by enumeration

- The operations carried out in the previous formula can be represented by the following tree:



Algorithm: inference by enumeration

- Input: the query variable X, a set of observed values $\mathbf{e}$ for the evidence variables and a bayesian network
- Output: $\mathbf{P}(X|\mathbf{e})$

Inference by enumeration

```
FUNCTION INFERENCE_ENUMERATION(X,e,NET)
1. Let $Q(X)$ be a probability distribution over
   the values of $X$, initially empty
2. FOR each value $x_i$ of $X$:
   2.1 Extend $e$ with value $x_i$ for $X$
   2.2 Let $Q(x_i)$ be equal to
       ENUM_AUX(VARIABLES(NET),$e$,NET)
3. Return NORMALIZE($Q(X)$)
```

Algorithm of inference by enumeration

Inference by enumeration

```
FUNCION ENUM_AUX(VARS,e,NET)
1. If VARS is empty, return 1
2. Else,
   2.1 Let $Y$ equal to FIRST(VARS)
   2.2 If $Y$ has value $y$ in $e$,
       2.2.1 return P($y$|parents($Y$,$e$))·ENUM_AUX(RESTO(VARS),$e$)
       2.2.2 Else, return
            SUM($y$,P($y$|parents($Y$,$e$))·ENUM_AUX(REST(VARS),$e_y$))
           (where: parents($Y$,$e$) is the set of values in $e$ taken by
           the parents of $Y$ in NET, and $e_y$ extends $e$ with value $y$
           for $Y$)
```

Algorithm of inference by enumeration

- Remark:
  - For the algorithm to work ,**`VARIABLES(NET)`** must return the variables in an order consistent with the implicit order in the graph of the network, from top to bottom.
- Depth-first:
  - The algorithm generates the previous operations tree, from top to bottom, in a depth-first manner
  - Therefore, it has a linear space cost
- But it may carry out repeated operations
  - In the exampl,e, $P(j|a)P(m|a)$ and $P(j|\neg a)P(m|\neg a)$ are computed twice
  - When there are many variables, these redundant calculations are unacceptable.

Avoiding redundant calculations

- Idea:
  - Carry out all the operations corresponding to a summation *only once*, for all the possible values of the variable of that summation
  - Instead of multiplying *numbers*, multiply *probability tables*
  - We will call those tables as *factors*

Avoiding redundant calculus

- For example, the operations

$$\mathbf{P}(R|j,m) = \alpha \mathbf{P}(R) \sum_e P(E) \sum_a \mathbf{P}(A|R,E) P(j|A) P(m|A)$$

  can be seen as operations between five tables or *factors*

- This operations between factors are of two types: *pointwise product* and *summing out*
- It is the *variable elimination* algorithm
- Let us see in more detail how this algorithm works with the query $\mathbf{P}(R|j,m)$

Variable elimination algorithm: an example

- First, let us show the initial factors in this query:
  - Each variable has a factor corresponding with its probability table in the network
  - In these tables, the values for the evidence variables are fixed:

Variable elimination algorithm: an example

- The factor corresponding to M is obtained from the table $\mathbf{P}(M|A)$.
  - Since M is an evidence variable and its value is fixed to *true*, the corresponding factor, denoted as $\mathbf{f_M}(A)$, is $\mathbf{P}(m|A)$ (a table with entries $P(m|a)$ and $P(m|\neg a)$):

A	$\mathbf{f_M}(A) = \mathbf{P}(m|A)$
a	0.70
$\neg a$	0.01

- The factor corresponding to J is obtained from the table $\mathbf{P}(J|A)$
  - Since J is an evidence variable and its value is fixed to *true*, the corresponding factor, denoted as $\mathbf{f_J}(A)$, is $\mathbf{P}(j|A)$ (a table with entries $P(j|a)$ and $P(j|\neg a)$):

A	$\mathbf{f_J}(A) = \mathbf{P}(j|A)$
a	0.90
$\neg a$	0.05

Variable elimination algorithm: an example

- The factor corresponding to variable A, denoted $\mathbf{f_A}(A, R, E)$ is obtained from $\mathbf{P}(A|R, E)$
  - None of these variables is of evidence, and therefore their values are not fixed
  - It is a table with $2 \times 2 \times 2$ entries, oen for each combination of values of A, R and E
  - In this case, it is exactly the corresponding table of the network

A	R	E	$\mathbf{f_A}(A, R, E) = \mathbf{P}(A\|R, E)$
a	r	e	0.95
a	r	$\neg e$	0.94
a	$\neg r$	e	0.29
a	$\neg r$	$\neg e$	0.001
$\neg a$	r	e	0.05
$\neg a$	r	$\neg e$	0.06
$\neg a$	$\neg r$	e	0.71
$\neg a$	$\neg r$	$\neg e$	0.999

Variable elimination algorithm: an example

- The factor corresponding to E, denoted as $\mathbf{f_E}(E)$, is the table $\mathbf{P}(E)$

E	$\mathbf{f_E}(E) = \mathbf{P}(E)$
e	0.002
$\neg e$	0.998

- The factor corresponding to R, denoted as $\mathbf{f_R}(R)$, is the table $\mathbf{P}(R)$

R	$\mathbf{f_R}(R) = \mathbf{P}(R)$
r	0.001
$\neg r$	0.999

Variable elimination algorithm: an example

- Once we have the initial factors we “eliminate” the variables that are not query or evidence variables
  - Intuitively the “elimination” of a variable is the analogue of carrying out the corresponding summation, but a the level of tables.
  - When we eliminate all those variables, then we will have only factors depending on the query variable; finally, we multiply them and normalize the result.
- Note: the elimination order does not affect the final result, but it may have an impact on the efficiency

Variable elimination algorithm: an example

- For example, we eliminate first the variable A
  - This corresponds to carry out the summation $\sum_a \mathbf{P}(A|R,E)P(j|A)P(m|A)$, that is, $\sum_a \mathbf{f_A}(A,R,T)\mathbf{f_J}(A)\mathbf{f_M}(A)$
  - The product of $\mathbf{f_M}$, $\mathbf{f_J}$ and $\mathbf{f_A}$, denoted $\mathbf{f}_{\times\mathbf{A}}(A,R,E)$ is a table with an entry for each combination of values of A, R and E, obtained multiplying the corresponding entries in those factors:
  - That is, for each value v_1 of A, v_2 of R and v_3 of E we have $\mathbf{f}_{\times\mathbf{A}}(v_1,v_2,v_3) = \mathbf{f_M}(v_1)\mathbf{f_J}(v_1)\mathbf{f_A}(v_1,v_2,v_3)$. For example: $\mathbf{f}_{\times\mathbf{A}}(true,false,true) = \mathbf{f_M}(true)\mathbf{f_J}(true)\mathbf{f_A}(true,false,true) = 0.70 \times 0.90 \times 0.29 = 0.1827$

Variable elimination algorithm: an example

- Now, we have to *sum out* the values of A in $\mathbf{f}_{\times\mathbf{A}}$ (that is, to carry out the summation $\sum_a$)
  - Thus, we obtain a table $\mathbf{f}_{\overline{\mathbf{A}}}(R, E)$ making $\mathbf{f}_{\overline{\mathbf{A}}}(v_1, v_2) = \sum_a \mathbf{f}_{\times\mathbf{A}}(a, v_1, v_2)$ for each value v_1 of R and v_2 of E, ranging a over the possible values of A
  - We have *eliminated* the variable A
  - Once we have summed out the variable A, we store $\mathbf{f}_{\overline{\mathbf{A}}}$ and we discard $\mathbf{f}_{\mathbf{M}}, \mathbf{f}_{\mathbf{J}}$ and $\mathbf{f}_{\mathbf{A}}$, since we will no longer need them

R	E	a	$\neg a$	$\mathbf{f}_{\overline{\mathbf{A}}}(R, T)$
r	e	$0.70 \times 0.90 \times 0.95 = 0.5985$	$0.01 \times 0.05 \times 0.05 = 0.00003$	0.59853
r	$\neg e$	$0.70 \times 0.90 \times 0.94 = 0.5922$	$0.01 \times 0.05 \times 0.06 = 0.00003$	0.59223
$\neg r$	e	$0.70 \times 0.90 \times 0.29 = 0.1827$	$0.01 \times 0.05 \times 0.71 = 0.00036$	0.18306
$\neg r$	$\neg e$	$0.70 \times 0.90 \times 0.001 = 0.00063$	$0.01 \times 0.05 \times 0.999 = 0.0005$	0.00113

Variable elimination algorithm: an example

- Elimination of E:
  - We denote as $\mathbf{f}_{\overline{\mathbf{E}}}(R)$ the factor resulting after multiplying $\mathbf{f}_{\overline{\mathbf{A}}}(R, E)$ and $\mathbf{f}_{\mathbf{E}}(E)$ and summing out E

R	e	$\neg e$	$\mathbf{f}_{\overline{\mathbf{T}}}(R)$
r	$0.59853 \times 0.002 = 0.001197$	$0.59223 \times 0.998 = 0.591046$	0.59224
$\neg r$	$0.18306 \times 0.002 = 0.000366$	$0.00113 \times 0.998 = 0.001128$	0.00149

  - We now discard $\mathbf{f}_{\overline{\mathbf{A}}}$ and $\mathbf{f}_{\mathbf{E}}$

Variable elimination algorithm: an example

- The last variable is R:
  - Since it is the query variable, we only multiply the factors where it appears:
  - Multiply $\mathbf{f}_{\overline{\mathbf{T}}}(R)$ and $\mathbf{f}_{\mathbf{R}}(R)$), obtaining $\mathbf{f}_{\times\mathbf{R}}(R)$

R	$\mathbf{f}_{\overline{\mathbf{T}}}(R) \times \mathbf{f}_{\mathbf{R}}(R)$
r	$0.59224 \times 0.001 = 0.00059$
$\neg r$	$0.00149 \times 0.999 = 0.00149$

  - Finally, we normalize the table: $\mathbf{P}(R|j,m) = \langle 0.28417; 0.71583\rangle$
- This table finally computed is just $\mathbf{P}(R|j,m)$. That is, $P(r|j,m) = 0.28417$ and $P(\neg r|j,m) = 0.71583$

Variable elimination algorithm: some comments

- Initially, we have a set of factors (one factor for each network variable)
- For each variable that is not a query or evidence variable, we replace all the factors mentioning that variable, for the factor that results after eliminating the variable.
  - Eliminate a variable: first, carry out a pointwise product of all the factors mentioning that variable, and then sum out that variable
- Finally, for the query variable, just carry out a pointwise product of the remaining factors, and then normalize
- The resulting normalized factor is the desired probability distribution of the query variable given the evidences

Variable elimination algorithm: some comments

- Pointwise product of factors:
  - If $\mathbf{f_1}(\mathbf{X}, \mathbf{Y})$ and $\mathbf{f_2}(\mathbf{Y}, \mathbf{Z})$ are two factors whose variables in common are $\mathbf{Y}$, we define its product $\mathbf{f}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ as the factor whose entries are $f(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f_1(\mathbf{x}, \mathbf{y}) f_2(\mathbf{y}, \mathbf{z})$
  - It is similar to a *join* operation in databases, multiplying the corresponding values
- Sum out:
  - If $\mathbf{f}_{\times Y}(Y, \mathbf{Z})$ is a factor (obtained as a pointwise product of the factors mentioning Y), summing out is to obtain $\mathbf{f}_{\overline{Y}}(\mathbf{Z})$ whose entries are $\mathbf{f}_{\overline{Y}}(\mathbf{z}) = \sum_y \mathbf{f}_{\times Y}(y, \mathbf{Z})$
  - Summing out is similar to *aggregate* in databases

A previous step: irrelevant variables

- Before applying the variable elimination algorithm, we can discard some variables that are irrelevant for the particular query
- Example:
  - If in the example, the query is $\mathbf{P}(J|r)$, then we have to compute $\alpha\mathbf{P}(R)\sum_e P(E)\sum_a P(A|R,E)\mathbf{P}(J|A)\sum_m P(M|A)$
  - But $\sum_m P(M|a) = 1$; so the variable M is irrelevant for that query
  - Note: any "leaf" in the network, that is not a query or evidence variable can be safely ignored for that query
- And in general, it can be shown that any variable that is not an ancestor of a query variable or evidence variable is irrelevant to that query, and then it can be removed for that query.

The variable elimination algorithm

- Input: the query variable X, a set of observed values $\mathbf{e}$ for the evidence variables, and a bayesian network
- Output: $\mathbf{P}(X|\mathbf{e})$

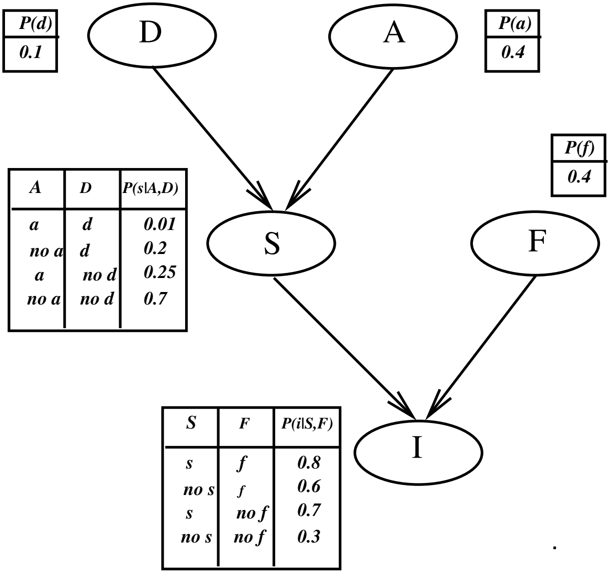
Algoritmo de eliminación de variables

```
FUNCION VARIABLE_ELIMINATION(X,e,RED)
1. Let NET' be the network resulting after removing from RED
   all the irrelevant variables for the query
2. Let FACTORS be the set of factors corresponding
   to each variable of NET'
3. Let VARS_ORD be the set of variables of NET',
   excluding evidence variables and ordered by a given
   elimination order
5. For each VAR in VARS_ORD, DO:
   5.1 If VAR is a query variable, eliminate from FACTORS all
       the factors mentioning that variable, and include
       their pointwise product
   5.2 Else, eliminate from FACTORS all the factors mentioning
       that variable, and include the factor that results
       after making their pointwise product and
       summing out VAR
6. Return the NORMALIZATION of the only factor remaining in
   FACTORS
```

Another example (Béjar)

- Consider the following variables:
  - D: regular sporting activities
  - A: healthy nutrition
  - S: high blood preasure
  - F: smoker
  - I: to have a heart attack
- The causal relationships and the associated probabilistic knowledge are modeled in the following bayesian network

Example: bayesian network



P(d)
0.1

P(a)
0.4

P(f)
0.4

A	D	P(s|A,D)
a	d	0.01
no a	d	0.2
a	no d	0.25
no a	no d	0.7

S	F	P(i|S,F)
s	f	0.8
no s	f	0.6
s	no f	0.7
no s	no f	0.3

Example of probabilistic inference

- Let us compute the probability of a person being smoker, given that he has had a heart attack and that he does not practice sports on a regular basis: $\mathbf{P}(F|i,\neg d)$
- Directly applying the formula:
  - $\mathbf{P}(F|i,\neg d) = \alpha\mathbf{P}(F,i,\neg d) = \alpha\sum_{S,A}\mathbf{P}(F,i,\neg d,A,S)$
  - We factorize according to the network: $\mathbf{P}(F|i,\neg d) = \alpha\sum_{S,A}P(\neg d)P(A)P(S|\neg d,A)\mathbf{P}(F)\mathbf{P}(i|S,F)$
  - We take out common factors: $\mathbf{P}(F|i,\neg d) = \alpha P(\neg d)\mathbf{P}(F)\sum_{A}P(A)\sum_{S}P(S|\neg d,A)\mathbf{P}(i|S,F)$

Example of probabilistic inference

- Let us calculate:
  - For $F = \mathit{true}$:

$$P(f|i,\neg d) = \alpha \cdot P(\neg d) \cdot P(f) \cdot$$

$$\cdot [P(a) \cdot (P(s|\neg d, a) \cdot P(i|s,f) + P(\neg s|\neg d, a) \cdot P(i|\neg s, f)) +$$

$$+ P(\neg a) \cdot (P(s|\neg d, \neg a) \cdot P(i|s,f) + P(\neg s|\neg d, \neg a) \cdot P(i|\neg s, f))] =$$

$$= \alpha \cdot 0.9 \cdot 0.4 \cdot [0.4 \cdot (0.25 \cdot 0.8 + 0.75 \cdot 0.6) + 0.6 \cdot (0.7 \cdot 0.8 + 0.3 \cdot 0.6)] =$$

$$= \alpha \cdot 0.253$$

  - Similarly, for $F = \mathit{false}$, $P(\neg f|i,\neg d) = \alpha \cdot 0,274$
  - Normalizing, $\mathbf{P}(F|i,\neg d) = \langle 0.48; 0.52 \rangle$

Applying elimination variables (initial factors)

- Factors corresponding to I:

S	F	$\mathbf{f_I}(S,F) = \mathbf{P}(i \mid S,F)$
s	f	0.8
s	$\neg f$	0.7
$\neg s$	f	0.6
$\neg s$	$\neg f$	0.3

- Factor corresponding to F:

F	$\mathbf{f_F}(F) = \mathbf{P}(F)$
f	0.4
$\neg f$	0.6

Applying elimination variables (initial factors)

- Factor corresponding to S:

S	A	$\mathbf{f_S}(S, A) = \mathbf{P}(S \mid \neg d, A)$
s	a	0.25
s	$\neg a$	0.7
$\neg s$	a	0.75
$\neg s$	$\neg a$	0.3

- Factor corresponding to A:

A	$\mathbf{f_A}(A) = \mathbf{P}(A)$
a	0.4
$\neg a$	0.6

- Factor corresponding to D: $\mathbf{f_D}()$ (it does not depend on D, since its value is fixed to $\neg d$, therefore is a table with only one entry): 0.9
  - As we will see, factors with only one entry are irrelevant for the final result

Applying elimination variables (initial factors)

- We will use the following elimination ordering: S, A, F
- Maybe other ordering could be better for the efficiency of this query computation
- We could apply some heuristics to choose a good elimination order (see later)

Eliminating S

- First we multiply $\mathbf{f}_{\mathbf{I}}(S, F)$ and $\mathbf{f}_{\mathbf{S}}(S, A)$ obtaining $\mathbf{f}_{\times\mathbf{S}}(S, A, F)$ and in that factor we sum out the variable S, obtaining $\mathbf{f}_{\overline{\mathbf{S}}}(A, F)$:

A	F	s	$\neg s$	$\mathbf{f}_{\overline{\mathbf{S}}}(A, F)$
a	f	0.25×0.80	0.75×0.60	0.65
a	$\neg f$	0.25×0.70	0.75×0.30	0.40
$\neg a$	f	0.70×0.80	0.30×0.60	0.74
$\neg a$	$\neg f$	0.70×0.70	0.30×0.30	0.58

- We remove $\mathbf{f}_{\mathbf{I}}(S, F)$ and $\mathbf{f}_{\mathbf{S}}(S, A)$ from the list of factors and include $\mathbf{f}_{\overline{\mathbf{S}}}(A, F)$

Eliminating A

- First we multiply $\mathbf{f}_{\mathbf{A}}(A)$ and $\mathbf{f}_{\overline{\mathbf{S}}}(A, F)$ obtaining $\mathbf{f}_{\times\mathbf{A}}(A, F)$, and in that factor we sum out the variable A, obtaining $\mathbf{f}_{\overline{\mathbf{A}}}(F)$

F	a	$\neg a$	$\mathbf{f}_{\overline{\mathbf{A}}}(F)$
f	$0.4.65$	0.6×0.74	0.704
$\neg f$	0.4×0.4	$0.6 \times o.58$	0.508

- We remove $\mathbf{f}_{\mathbf{A}}(A)$ and $\mathbf{f}_{\overline{\mathbf{S}}}(A, F)$ from the list of factors, and we include $\mathbf{f}_{\overline{\mathbf{A}}}(F)$

Last step

- Last step: multiply the remaining factors and normalize
  - Product

F	$\mathbf{f_D}() \times \mathbf{f_{\overline{A}}}(F) \times \mathbf{f_F}(F)$
f	$0.9 \times 0.704 \times 0.4 = 0.253$
$\neg f$	$0.9 \times 0.508 \times 0.6 = 0.274$

  - Normalizing: $\mathbf{P}(F|i, \neg d) = \langle 0.48; 0.52 \rangle$ (note that the factor $\mathbf{f_D}$ turns out to be irrelevant, since it does not affect to the normalization step)
- Therefore, the probability of a person being smoker, given that he has had a heart attack and that he does not practice sports on a regular basis is 0.48

Complexity of elimination variables

- The algorithmic complexity (both in time and space) is dominated by the size of the biggest factor obtained during the elimination process
- And that is influenced by the elimination ordering used
  - We could use heuristic criteria to chose the next variable to eliminate in each step
  - With the purpose of controlling the size of the factors
- If the network is *singly connected* (*polytree*), it can be shown that the complexity (both in time and space) is *linear* in the size of the network (the number of total entries in its tables)
  - A network is singly connected if there is at most one undirected path between any two nodes in the networtk
  - For example, the network of the alarm example is singly connected.

Complexity of exact inference

- But in general, the algorithm has exponentian complexity (both in time and space) in the worst case
  - This is not surprising: in particular bayesian inference includes propositional inference and thus the SAT problem
- When exact inference inference cannot be done in a reasonable time, it is essential to use approximate inference mnethods
- Randomized algorithms, based on sampling methods that simlulate the network probability distribution

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. Learning a bayesian classifier: naive Bayes

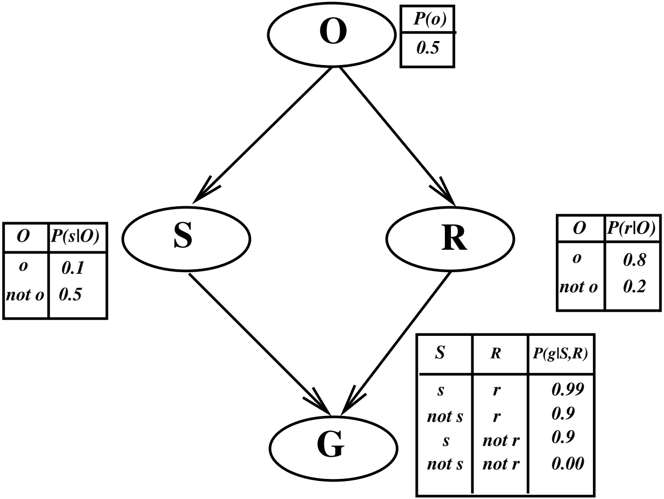
Samplin

- *Sampling* with respect a given probability distribution means to generate events in such a way that the probability of generating an event is the probability that can be computed from the distribution.
- The simplest sampling: let A be a boolean r.v. such that $\mathbf{P}(A) = \langle \theta, 1 - \theta \rangle$
  - For that, it is enough to have a method to generate random numbers uniformly distributed on the interval $[0, 1]$
  - If we generate $x < \theta$, we return a; otherwise $\neg a$
  - In the limit, the number of samples with value a divided by the total number of samples is θ
- It is quite natural to generalize this idea to design a sampling method with respect the JFD of a bayesian network

Example of bayesian network (Russell & Norvig)

- Consider the following random variables:
  - O: overcast, cloudy sky
  - S: the sprinklers have worked
  - R: it rained
  - G: the grass is wet
- Causal relationships and the associated probabilistic knowledge are reflected in the following network
  - Note: this network is not singly connected

Otro ejemplo: red bayesiana



P(o)
0.5

O	P(s|O)
o	0.1
not o	0.5

O	P(r|O)
o	0.8
not o	0.2

S	R	P(g|S,R)
s	r	0.99
not s	r	0.9
s	not r	0.9
not s	not r	0.00

Example of prior sampling

- Let us see how we can generate a complete atomic event fron the JFD represented by the previous network
  - Sample from $\mathbf{P}(O) = \langle 0.5; 0.5 \rangle$ (applying the method described previously), we obtain o
  - Sampling from $\mathbf{P}(S|o) = \langle 0.1; 0.9 \rangle$; we obtaing $\neg s$
  - Sampling from $\mathbf{P}(R|o) = \langle 0.8; 0.2 \rangle$; we obtain r
  - Sampling from $\mathbf{P}(G|\neg s, r) = \langle 0.9; 0.1 \rangle$; we obtain g
- The atomic event generated has been $\langle o, \neg s, r, g \rangle$
- The probability of this event being generated is $0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324$, since each sampling is done independently

Prior sampling algorithm

- Assume that NET is bayesian network with variables $X_1, \ldots, X_n$ ordered in a consistent way, with respect to the implicit graph ordering

Prior sampling algorithm

```
FUNCTION PRIOR-SAMPLING(NET)
1. FOR $i = 1, \ldots, n$, DO: let $x_i$ the result of a sampling from
   the distribution $P(X_i|padres(X_i))$
2. Return $(x_1, \ldots, x_n)$
```

Properties of the prior sampling algorithm

- Let $S_{MP}(x_1, \ldots, x_n)$ be the probability of generating the event $(x_1, \ldots, x_n)$ using **`PRIOR-SAMPLING`**
- It is easy to deduce (*just* reasoning on the algorithm) that $S_{MP}(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i))$
- Therefore, if we repeat prior sampling N times, and denote $N_{MP}(x_1, \ldots, x_n)$ the number of times that the event $(x_1, \ldots, x_n)$ is returned, then

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

  - Where $\approx$ means that in the limit (as N approaches to ∞) that equality is exact (according to the law of large numbers)
  - That is called a *consistent estimate*

Rejection sampling

- The previous property is the basis for the approximate inference algorithms that we will see
  - Recall: probabilistic inference means to compute $\mathbf{P}(X|\mathbf{e})$ where $\mathbf{e}$ denotes the values observed for some (evidence) variables and X is the query variable
- Rejection sampling: generate atomic events using prior sampling and compute proportions of the values of X, between those events in which $\mathbf{e}$ "holds"
- For example, to estimate $\mathbf{P}(R|s)$, generate 100 samples; if we have 27 samples in which the value of S is s, and in 8 of these samples we have R with value r, and 19 with value $\neg r$, then the estimate is:

$$\mathbf{P}(R|s) \approx \mathit{Normalize}(\langle 8; 19 \rangle) = \langle 0.296; 0.704 \rangle$$

- The exact probability is $\langle 0.3; 0.7 \rangle$

Rejection sampling

- Input: a query variable X, a set of observed values $\mathbf{e}$ for the evidence variables, a bayesian network NET (with n variables) and a number N of total samples to generate

Rejection sampling algorithm

```
FUNCION REJECTION-SAMPLING(X,e,NET,N)
1. Let $N[X]$ be a vector with one component for each possible
   value of the query variable $X$, initially with zeros
2. FOR $k = 1, \ldots, N$:
   2.1 Let $(y1, \ldots, yn)$ equal to PRIOR-SAMPLING(NET)
   2.2 If $y = (y1, \ldots, yn)$ is consistent with $e$ then make
       $N[x]$ equal to $N[x] + 1$, where $x$ is the value of $X$
       in $y$
3. Return NORMALIZE($N[X]$)
```

- From the properties of **`PRIOR-SAMPLING`** we can deduce that this algorithm returns a consistent estimate of $\mathbf{P}(X|\mathbf{e})$
- Problem: too many samples are rejected (especially if the number of variables is high)

Likelihood weighting

- It is possible to design an algorithm generating only events that are consistente with the evidence **e**
- The value for the evidence variable are fixed, and only the values for the rest of variables are sampled
- But not all the generated events count the same: those in which the evidence (in conjuntion with the rest of values sampled) is less likely, shoudld be given less "weight"
- Each generated event is weighted by the product of the probability of each value in **e** (conditiones to its parents)

Example of likelihood weighting

- Assume that we want to compute $\mathbf{P}(R|s,g)$. Let us show how we can generate an event, with its corresponding weight w (initially, $w = 1.0$):
  - Sampling from $\mathbf{P}(O) = \langle 0.5; 0.5 \rangle$; we get o
  - Since S is an evidence variable (with value s), we make w equal to $w \times P(s|o)$ (that is, $w = 0.1$)
  - Sampling from $\mathbf{P}(R|o) = \langle 0.8; 0.2 \rangle$; we get r
  - G is an evidence variable (with value g); therefore, we make w equal to $w \times P(g|s,r)$ (that is, $w = 0.099$)
- The complete weighted event would be $\langle o, s, r, g \rangle$ with a weight equal 0.099
- This event counts for $R = true$, but weighted 0.099, instead of 1 (intuitively, this reflects that it is less likely that the sprinklers work on a rainy day)

Algorithm: likelihood weighting

- Input: a query variable X, a set **e** of observed values for the evidence variables, a bayesian network NET (with n variables) and a numbre N para la variables de evidencia, una RED bayesiana (con n variables) and a number N of total samples to generate

Likelihood weighting algorithm

```
FUNCTION LIKELIHOOD-WEIGHTING(X,e,NET,N)
1. Let $W[X]$ be a vector with one component for each possible
   value of the query variable $X$, initially with zeros
2. FOR $k = 1, \dots, N$:
   2.1 Let $[(y1, \dots, yn), w]$ be a weighted sample returned
       by WEIGHTED-SAMPLE(NET,e)
   2.2 Make $W[x]$ equal to $W[x] + w$, wher $x$ is the value taken
       by the query variable $X$ in $y$
3. Return NORMALIZE($W[X]$)
```

Algorithm: likelihood weighting

Obtaining weighted samples

```
FUNCTION WEIGHTED-SAMPLE(NET,e)
1. Hacer $w = 1.0$
2. FOR $i = 1, \ldots, n$:
   2.1 If the variable $X_i$ has value $x_i$ in $e$, then
       $w = w \times p(X_i = x_i | padres(X_i))$
   2.2 Else, let $x_i$ be a value of $X_i$ returned sampling
       from $P(X_i | padres(X_i))$
3. Return $[(x_1, \ldots, x_n), w]$
```

More on approximate inference algorithms

- It can be shown that the algorithm **`LIKELIHOOD-WEIGHTING`** returns a consistent estimate of $\mathbf{P}(X|\mathbf{e})$
- Note that as the number of evidences increase, the algorithm suffer a degradation in performance (since most of the samples will have an infinitesimal likelihood)
- There are many others approximate inference algorithms for bayesian networks, more sophisticated than the two algorithms we have seen here

Outline

1. Uncertainty and knowledge. Random variables
2. Unconditional probability
3. Conditional probability: probabilistic inference
4. Independence. Conditional independence
5. Bayesian networks
6. Constructing bayesian networks
7. Deducing conditional independences
8. Exact inference on bayesian networks
9. Approx. inference on bayesian networks
10. **Learning a bayesian classifier: naive Bayes**

Learning a probabilistic classifier

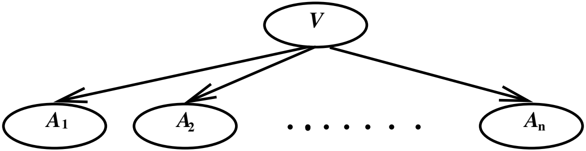
- In this final section, we return to the field of machine learning
    - In particular, we will give an algorithm to learn a classifier, called *naive Bayes*
- As usual, we have a training set of classified data and we want to learn a model to classify new data
    - In previous units, we have seen how to learn from data decision trees, rules, neural networks,. . .
    - Now we will see how to learn a simple bayesian network, and we will use that network to classify new examples
- Probabilistic classifier: the learned model allows to predict, for a new example, *the probabilities* of belonging to each class

Naive Bayes classifier

- Suppose we have n attributes (or *features*) $A_1, \ldots, A_n$ and a finite set V of possible classes
- An example is a tuple of concrete values for each attribute
- We have a training set D with examples, each example with its class.
- We want to learn to classify new examples $\langle a_1, \ldots, a_n \rangle$
  - We will learn a probabilistic model
  - Once learned, we will use it to compute the probability of each class, given the example

Naive Bayes classifier

- In the probabilistic model both the attributes and the class will be considered random variables
- The *naive* assumption:
  - Assume that the attributes are (mutually) conditionally independent given the class
- This means that we assume that the model can be represented by the following bayesian network:



- This assumption is very strong and usually unrealistic:
  - Despite that, naive Bayes classifiers work quite well

Naive Bayes classifier

- The class predicted for a new example $\langle a_1, \dots, a_n \rangle$, denoted v_{NB}, will be the most likely class, given the attribute values:

$$v_{NB} = \underset{v_j \in V}{argmax}\, P(v_j | a_1, \dots, a_n)$$

- Applying the conditional independences assumed in the bayesian network, we have:

$$P(V|a_1, \dots, a_n) = \alpha P(V, a_1, \dots, a_n) = \alpha P(V) \prod_i P(a_i|V)$$

- If we are interested only in knowing the most likely class, then we can ignore the normalization constant and therefore:

$$v_{NB} = \underset{v_j \in V}{argmax}\, P(v_j) \prod_i P(a_i|v_j)$$

Probabilities estimates in *Naive Bayes*

- This means that to predict the class of new examples, it will be enough to learn the probabilities $P(v_j)$ (*a priori*) and $P(a_i|v_j)$ (*conditioned*)
  - These are precisely the probability tables of the network
  - We will use the training set to learn them
- The probabilities of the bayesian network are *estimated* by computing frecuencies in the training set:

$$P(v_j) = \frac{n(V = v_j)}{N} \qquad P(a_i|v_j) = \frac{n(A_i = a_i, V = v_j)}{n(V = v_j)}$$

where N is the total number of examples in the trainig set, $n(V = v_j)$ is the number of examples classified as v_j, and $n(A_i = a_i, V = v_j)$ is the number of examples classified as v_j whose value of attribute A_i is a_i.

Naive Bayes classifier: an example

- Let us see a naive Bayes classifier applied to the "Play tennis" example. Training set:

Ej.	Outlook	Temperature	Humidity	Wind	PlayTennis
D_1	Sunny	High	High	Weak	-
D_2	Sunny	High	High	Strong	-
D_3	Overcast	High	High	Weak	+
D_4	Rainy	Mild	High	Weak	+
D_5	Rainy	Low	Normal	Weak	+
D_6	Rainy	Low	Normal	Strong	-
D_7	Overcast	Low	Normal	Strong	+
D_8	Sunny	Mild	High	Weak	-
D_9	Sunny	Low	Normal	Weak	+
D_{10}	Rainy	Mild	Normal	Weak	+
D_{11}	Sunny	Mild	Normal	Strong	+
D_{12}	Overcast	Mild	High	Strong	+
D_{13}	Overcast	High	Normal	Weak	+
D_{14}	Rainy	Mild	High	Strong	-

Naive Bayes classifier: an example

- Assume we want to predict if a sunny day, with mild temperature, high humidity, and strong wind is good for playing tennis
- According to the Naive Bayes classifier:

$$v_{NB} = \underset{v_j \in \{+,-\}}{argmax}\ P(v_j)P(sunny|v_j)P(mild|v_j)P(high|v_j)P(strong|v_j)$$

- Thus, we need to estimate all those probabilities, and we do it computing frequencies of the corresponding combinations in tne table:
    - $p(+) = 9/14$, $p(-) = 5/14$, $p(sunny|+) = 2/9$, $p(sunny|-) = 3/5$, $p(mild|+) = 4/9$, $p(mild|-) = 2/5$, $p(high|+) = 3/9$, $p(high|-) = 4/5$, $p(strong|+) = 3/9$ y $p(strong|-) = 3/5$

Naive Bayes classifier: an example

- Therefore, the a posteriori probabilities (without normalizing) are:
  - $P(+)P(sunny|+)P(mild|+)P(high|+)P(strong|+) = 0.0070$
  - $P(-)P(sunny|-)P(mild|-)P(high|-)P(strong|-) = 0.0411$
- The Naive Bayes classifier returns the class $-$ (the one with the higher probability)

Technical details about probability estimates: log-probabilities

- There is a risk that some of the probability estimates are too low
- If we do not have many exaples in the training set, some of the probability estimates may be 0
- These may have two undesirable consequences:
  - Inaccurate estimates
  - May affect the classification, since the estimates are multiplied; if one of them is 0, then the a posteriori probability is 0
- A first approach to deal with this issue: use the logarithm of the probabilities
- Products are transformed into sums:

$$v_{NB} = \underset{v_j \in V}{argmax}[log(P(v_j)) + \sum_i log(P(a_i|v_j))]$$

Technical details about probability estimates: smoothing

- Problem: very low probabilities or even zero, due to the lack of some combinations in the training set
- Solution: use *Laplace smoothing* to estimate the conditional probabilities:

$$P(a_i|v_j) = \frac{n(A_i = a_i, V = v_j) + k}{n(V = v_j) + k|A_i|}$$

  where k is a constant and $|A_i|$ is the number of possible values for that attribute $|A_i|$.
- Intuitivelly: we are estimating as if we had k additional examples with class v_j for each possible value of the attribute A_i
  - For example (with $k = 1$):

$$p(mild|-) = (2+1)/(5+3) = 3/8$$

- Usually $k = 1$, but other values are possible
  - Choosing k: experiment with some value candidates, and choose according to its performance on a *validation set*

Bibliography

- Russell, S. and Norvig, P. *Artificial Intelligence (A Modern Approach), 3rd edition* (Prentice–Hall, 2010)
  - Ch. 13: “Quantifying Uncertainty”
  - Ch. 14: “Probabilistic Reasoning”
  - Sect. 20.2: “Learning with Complete Data”