Logic, probability and statistical relational learning

Vera Koponen

Uppsala, 11–12 December 2023

Notation and terminology

I use the same notation and terminology as in the "Notes on logic and probability" which are available on Canvas.

References to definitions, theorems etc also refer to the same notes.

Why formal/symbolic logic in AI?

1. A higher level artificial intelligence should be able to express knowledge and reason about it.
2. This can be achieved by using some kind of language. Formal languages have been constructed partly for the sake of turning the "art of making conclusions", and related problems, into a mathematically precise activity, and partly to facilitate the automation of reasoning.
3. The formal languages of propositional logic and first-order logic are the cornerstones of the formal approaches to knowledge representation and deduction, and the multitude of other formal logical languages that exist are modifications or extensions of these.

Statistical relational learning (SRL) $\approx$ logic + probability

Our world consists of *objects* and these objects may have certain *properties* and different objects may be *related* in various ways.

The same is true for many types of data sets (databases).

With formal (logical) languages one can make general statements such as

"there exists z such that z is a parent of x and of y",
or formally $\exists z(\mathit{Parent}(z, x) \wedge \mathit{Parent}(z, y))$
expressing that x and y are (possibly half) siblings.

Such a statement makes sense on any data set, or *domain* of objects as I will say, in which the basic/atomic properties/relations are present, in the above example the relation $\mathit{Parent}(x, y)$.

Statistical relational learning $\approx$ logic + probability

This is relevant if we want to *extrapolate* conclusions induced from one domain to another domain,

which is important if we want to use a model of inference, e.g. a probabilistic graphical model, learned from one domain to make inferences about another domain.

Although the issue of extrapolation across domains will be a key issue we often begin a new topic, for the sake of simplicity, by considering *propositional logic* which makes *no* reference to a domain of objects.

Formulas as programs (propositional logic)

Let p, q, r be propositional variables. A disjunction of literals, for example $p \vee \neg q \vee r$, is called a *clause*. A conjunction of clauses is called a *conjunctive normal form (CNF)*.

A clause can be seen as a *rule* with which one can draw conclusions given some evidence.

Example. $p \vee \neg q \vee r \equiv (\neg p \wedge q) \rightarrow r$,
so $p \vee \neg q \vee r$ can be viewed as the rule "if p is false and q is true, then r is true".

A *set of clauses* can thus be viewed as a *set of rules*.

We can also view a CNF as a set of rules since, because a CNF expresses that a finite set of clauses hold.

By a *logic program* we mean either a set of clauses, or a CNF.

Learning a logic program from a set of examples

Let F, M, H be propositional variables and let $\mathbf{E} = \{(0,1,1),(1,0,1),(0,0,0)\}$ be a set of truth assignments to F, M, H (in this order).

Problem: Find clauses (rules) which are satisfied by all truth assignments in $\mathbf{E}$.

Solution:
1. Choose a truth assignment which does not belong to $\mathbf{E}$, for example $(1,1,1)$. Form a clause $\neg F \vee \neg M \vee \neg H$ which is *satisfied by all truth assignments except* $(1,1,1)$. Set $Q := \{\neg F \vee \neg M \vee \neg H\}$. Since also the simpler clause $\neg F \vee \neg M$ is satisfied by all truth assignments in $\mathbf{E}$ we can simplify Q to $Q := \{\neg F \vee \neg M\}$ (if we wish).

Now we have found one clause/rule, but we can continue.

Recall: $\mathbf{E} = \{(0,1,1),(1,0,1),(0,0,0)\}$.

2. Choose a truth assignment other than $(1,1,1)$ which is does not belong to $\mathbf{E}$, for example $(0,1,0)$. Form a clause $F \vee \neg M \vee H$ which is satisfied by all truth assignments except for $(0,1,0)$. Set $Q := Q \cup \{F \vee \neg M \vee H\} = \{\neg F \vee \neg M,\ F \vee \neg M \vee H\}$. Since the simpler clause $\neg M \vee H$ is also satisfied by all truth assignments in $\mathbf{E}$ we can simplify Q to $Q := \{\neg F \vee \neg M,\ \neg M \vee H\}$.

3. Choose a truth assignment other than those chosen previously which does not belong to $\mathbf{E}$, for example $(1,1,0)$ and form a clause $\neg F \vee \neg M \vee H$ which is satisfied by all truth assignments except for $(1,1,0)$. After simplifications we get
$Q := \{\neg F \vee \neg M,\ \neg M \vee H,\ \neg F \vee H\}$.

We could continue and find 2 more rules since there are two more truth assignments which do not belong to $\mathbf{E}$.

In fact, by the above procedure we can find a set of clauses Q which is satisfied by all truth assignments in $\mathbf{E}$ and not by any other truth assignment.

Whether this is desirable or not depends on the situation. If the logic program Q is not satisfied by any truth assignment outside of $\mathbf{E}$, then it is only useful for drawing conclusions in the context of $\mathbf{E}$.

See Proposition 3.2.3 and Corollary 3.2.4.

Learning from examples and nonexamples

Let p, q, r be propositional variables, let
$\mathbf{E} = \{(1,0,1),(1,0,0),(0,1,1)\}$ and
$\mathbf{N} = \{(1,1,0),(0,1,0)\}$
and observe that these two sets are disjoint.

Problem: Find a set of clauses Q such that *every* clause in Q is satisfied by *every* truth assignment in $\mathbf{E}$, and *every* clause in Q is falsified by *at least one* truth assignment in $\mathbf{N}$.

Solution: We can proceed as in the previous example with the modification that in each step we choose a truth assignment from $\mathbf{N}$. Thus the procedure will stop after at most $|\mathbf{N}|$. If we continue for $|\mathbf{N}|$ steps we get rules that hold for all truth assignments that do *not* belong to $\mathbf{N}$.

See Proposition 3.2.6 and the algorithms in Section 3.3.

What is the point of nonexamples?

If we only have a set of examples $\mathbf{E}$ and we want to find a set Q of clauses satisfied by all $\vec{a} \in \mathbf{E}$, then, according to the procedure described earlier, we have (if $\mathbf{E}$ is a lot smaller than $\{0,1\}^n$ which may well be the case in real situations) a very large number of truth assignments outside of $\mathbf{E}$. Which ones should we choose?

From a purely theoretical point of view, the choice of $\vec{a} \in \{0,1\}^n \setminus \mathbf{E}$ from which we construct the next clause does not matter. But from a practical point of view it may the of interest to choose $\vec{a} \in \{0,1\}^n \setminus \mathbf{E}$ which are unlikely to represent a situation within the context of interest (in the particular application).

A set of *nonexamples* $\mathbf{N} \subseteq \{0,1\}^n \setminus \mathbf{E}$ can have the role of describing situations which are definately not within the context of interest (for the particular application) and therefore we can primarily choose truth assignments from $\mathbf{N}$ to construct clauses (or "rules") valid for all truth assignments in $\mathbf{E}$.

First-order logic programs

Analogously as in propositional logic we can consider a first-order clause

$$\theta_1(\vec{x}) \vee \ldots \vee \theta_n(\vec{x})$$

(so each $\theta_i(\vec{x})$ is a literal) to be a "rule".

However, in first-order logic we interpret formulas as being true/false in *structures*, not with respect to truth assignments to propositional variables.
Let $\mathcal{A}$ be a first-order structure.
We view the above clause as a valid rule in $\mathcal{A}$ if

$$\mathcal{A} \models \forall \vec{x}\big(\theta_1(\vec{x}) \vee \ldots \vee \theta_n(\vec{x})\big)$$

which, for every choice of $1 \leq k < n$, is the case if and only if

$$\mathcal{A} \models \forall \vec{x}\big(\big(\neg\theta_1(\vec{x}) \wedge \ldots, \wedge\neg\theta_k(\vec{x})\big) \rightarrow \big(\theta_{k+1}(\vec{x}) \vee \ldots \vee \theta_n(\vec{x})\big)\big).$$

Concept learning

A special (but interesting) case is when we want to learn rules of the form

$$(\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x})) \rightarrow \psi(\vec{x})$$

where $\varphi_i(\vec{x})$, $i = 1, \ldots, n$ are literals and $\psi(\vec{x})$ is an atomic formula.

This is a variant of the problem of *concept learning*.

Let $\mathcal{A}$ be a structure and $\psi(\vec{x})$ an atomic formula.
To learn the concept "$\psi(\vec{x})$", given the evidence represented/coded by $\mathcal{A}$ means to find literals $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x})$ such that the relation symbol occuring in $\varphi_i(\vec{x})$ does not occur in $\psi(\vec{x})$ and

$$\mathcal{A} \models \forall \vec{x}\big((\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x})) \rightarrow \psi(\vec{x})\big).$$

Learning the concept of daughter

Let F, P, D, G be relation symbols. We interpret $F(x)$, $P(x, y)$, $D(x, y)$ and $G(x, y)$ as expressing that "x is female", "x is a parent of y", "y is a daughter of x", and "x is a grandmother of y", respectively.

Let a, b, e, g, h, m, n, t denote some individuals and suppose that we have some data regarding their kinships, as encoded by the structure $\mathcal{A}$ where

- $A = \{a, b, e, g, h, m, n, t\}$ is the domain of $\mathcal{A}$,
- $F^{\mathcal{A}} = \{m, h, e, n\}$ (i.e. m, n, e, and n are females),
- $P^{\mathcal{A}} = \{(g, m), (h, m), (h, t), (t, e), (n, e), (a, t), (e, b)\}$ (i.e. g is a parent of m, h is a parent of m, and so on),
- $D^{\mathcal{A}} = \{(g, m), (h, m), (t, e), (n, e)\}$, and
- $G^{\mathcal{A}} = \{(h, e), (n, b)\}$.

Choose a literal, for example $P(x, y)$, and test if

$$\mathcal{A} \models \forall x \forall y \big(P(x, y) \rightarrow D(x, y)\big).$$

The above is false, since for example $\mathcal{A} \models P(\hat{h}, \hat{t}) \wedge \neg D(\hat{h}, \hat{t})$ (that is, h is a parent of t but t is not a daughter of h).

Choose a literal, for example $P(x, y)$, and test if

$$\mathcal{A} \models \forall x \forall y \big(P(x, y) \rightarrow D(x, y)\big).$$

The above is false, since for example $\mathcal{A} \models P(\hat{h}, \hat{t}) \wedge \neg D(\hat{h}, \hat{t})$ (that is, h is a parent of t but t is not a daughter of h).

Hence we need to strenghten the assumptions. So choose for example the literal $F(x)$ and test if

$$\mathcal{A} \models \forall x \forall y \big((P(x, y) \wedge F(x)) \rightarrow D(x, y)\big).$$

This does not hold either because $\mathcal{A} \models P(\hat{e}, \hat{b}) \wedge F(\hat{e}) \wedge \neg D(\hat{e}, \hat{b})$.

Choose a literal, for example $P(x, y)$, and test if

$$\mathcal{A} \models \forall x \forall y \big(P(x, y) \rightarrow D(x, y)\big).$$

The above is false, since for example $\mathcal{A} \models P(\hat{h}, \hat{t}) \wedge \neg D(\hat{h}, \hat{t})$ (that is, h is a parent of t but t is not a daughter of h).

Hence we need to strenghten the assumptions. So choose for example the literal $F(x)$ and test if

$$\mathcal{A} \models \forall x \forall y \big((P(x, y) \wedge F(x)) \rightarrow D(x, y)\big).$$

This does not hold either because $\mathcal{A} \models P(\hat{e}, \hat{b}) \wedge F(\hat{e}) \wedge \neg D(\hat{e}, \hat{b})$. Now we add the literal $F(y)$ and test if

$$\mathcal{A} \models \forall x \forall y \big((P(x, y) \wedge F(x) \wedge F(y)) \rightarrow D(x, y)\big).$$

One can verify that the above is true.

Choose a literal, for example $P(x, y)$, and test if

$$\mathcal{A} \models \forall x \forall y \big(P(x, y) \rightarrow D(x, y)\big).$$

The above is false, since for example $\mathcal{A} \models P(\hat{h}, \hat{t}) \wedge \neg D(\hat{h}, \hat{t})$ (that is, h is a parent of t but t is not a daughter of h).

Hence we need to strenghten the assumptions. So choose for example the literal $F(x)$ and test if

$$\mathcal{A} \models \forall x \forall y \big((P(x, y) \wedge F(x)) \rightarrow D(x, y)\big).$$

This does not hold either because $\mathcal{A} \models P(\hat{e}, \hat{b}) \wedge F(\hat{e}) \wedge \neg D(\hat{e}, \hat{b})$. Now we add the literal $F(y)$ and test if

$$\mathcal{A} \models \forall x \forall y \big((P(x, y) \wedge F(x) \wedge F(y)) \rightarrow D(x, y)\big).$$

One can verify that the above is true.

But are the assumptions unnecessarily strong? One can check that

$$\mathcal{A} \models \forall x \forall y \big((P(x, y) \wedge F(y)) \rightarrow D(x, y)\big)$$

which is better if we want the conditions to the left of '$\rightarrow$' to explain as "accurately as possible" when $D(x, y)$ holds. In fact the converse implication "$\leftarrow$" also holds.

Learning the concept of grandmother

It is impossible to find n and literals $\varphi_1(x, y), \ldots, \varphi_n(x, y)$ such that

$$\mathcal{A} \models \forall x \forall y \big(\big(\varphi_1(x, y) \wedge \ldots \wedge \varphi_n(x, y) \big) \rightarrow G(x, y) \big)$$

and $\mathcal{A} \models \exists x \exists y \big(\varphi_1(x, y) \wedge \ldots \wedge \varphi_n(x, y) \big)$. (Why the last condition?)

This does not mean that it is impossible to learn the concept "x is a grandmother of y".
But we may need to consider more variables that x and y.

Learning the concept of grandmother

It is impossible to find n and literals $\varphi_1(x,y),\ldots,\varphi_n(x,y)$ such that

$$\mathcal{A} \models \forall x \forall y \big(\big(\varphi_1(x,y) \wedge \ldots \wedge \varphi_n(x,y) \big) \rightarrow G(x,y) \big)$$

and $\mathcal{A} \models \exists x \exists y \big(\varphi_1(x,y) \wedge \ldots \wedge \varphi_n(x,y) \big)$. (Why the last condition?)

This does not mean that it is impossible to learn the concept "x is a grandmother of y".
But we may need to consider more variables that x and y.

By reasoning similarly as in the previous example we can conclude that

$$\mathcal{A} \models \forall x \forall y \forall z \big(\big(P(x,z) \wedge P(z,y) \wedge F(x) \big) \rightarrow G(x,y) \big)$$

In fact we even have

$$\mathcal{A} \models \forall x \forall y \forall z \big(\big(P(x,z) \wedge P(z,y) \wedge F(x) \big) \leftrightarrow G(x,y) \big),$$

which is the usual definition of grandmother.

General problem of finding rules

Let L be first-order language and let $\mathbf{E}$ be a set of L-structures (the examples).

Problem: Find literals $\varphi_1(\vec{x}), \ldots, \varphi_n(\vec{x})$, $\psi_1(\vec{x}), \ldots, \psi_m(\vec{x})$ such that

$$\mathcal{A} \models \forall \vec{x}\big((\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x})) \rightarrow (\psi_1(\vec{x}) \vee \ldots \vee \psi_m(\vec{x}))\big)$$

holds for every $\mathcal{A} \in \mathbf{E}$ and such that

$$\mathcal{A} \models \exists \vec{x}(\varphi_1(\vec{x}) \wedge \ldots \wedge \varphi_n(\vec{x}))$$

for at least one $\mathcal{A} \in \mathbf{E}$.

In Section 3.4.2 a systematic search procedure for finding such literals is described.

For more about concept learning see Section 3.4.3.

Probabilities on possible worlds

What do we mean by saying that "the probability that an event **E** holds is α?"

The approach usually taken in SRL is roughly this:

The "world" can be in different states. In some way we assign each state of the "world", or each "possible world", a probability.

Now an event **E** is just a set of possible worlds and its probability is

$$\mathbb{P}(\mathbf{E}) = \sum_{W \in \mathbf{E}} \mathbb{P}(W).$$

We will consider two ways of making this idea precise, one way related to *propositional logic*, and one way related to *first-order logic*.

Possible worlds in propositional logic

Let $p_1, \ldots, p_n$ be propositional variables.

A truth assignment to $p_1, \ldots, p_n$ is represented by a vector/tuple/sequence $\vec{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$.

In this context a *possible world* is a truth assignment $\bar{a} \in \{0,1\}^n$ and an *event* $\mathbf{E}$ is a subset of $\{0,1\}^n$ (i.e. $\mathbf{E} \subseteq \{0,1\}^n$).

To be able to talk about "the probability of $\mathbf{E}$" we need to have a *probability distribution on* $\{0,1\}^n$.

Suppose that $\mathbb{P}$ is a probability distribution on $\{0,1\}^n$. The *probability of an event* $\mathbf{E} \subseteq \{0,1\}^n$ is defined to be

$$\mathbb{P}(\mathbf{E}) = \sum_{\bar{a} \in \mathbf{E}} \mathbb{P}(\bar{a}).$$

Suppose that $\mathbf{E}_1, \mathbf{E}_2 \subseteq \{0,1\}^n$. The *probability of* $\mathbf{E}_2$ *conditioned on* $\mathbf{E}_1$ is defined to be

$$\mathbb{P}(\mathbf{E}_2 \mid \mathbf{E}_1) = \frac{\mathbb{P}(\mathbf{E}_1 \cap \mathbf{E}_2)}{\mathbb{P}(\mathbf{E}_1)}.$$

Probabilities of (events defined by) formulas

As above let $\mathbb{P}$ be a probability distribution on $\{0,1\}^n$.
For all $\varphi, \psi \in \mathbf{Prop}(p_1, \ldots, p_n)$ we define

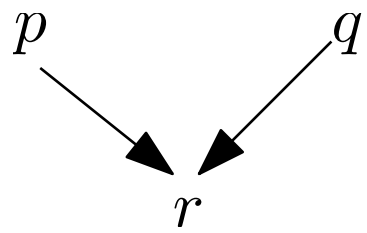
$$\mathbb{P}(\varphi) = \mathbb{P}(\{\bar{a} \in \{0,1\}^n : \bar{a} \models \varphi\})$$

and

$$\mathbb{P}(\varphi \mid \psi) = \frac{\mathbb{P}(\varphi \wedge \psi)}{\mathbb{P}(\psi)} = \frac{\mathbb{P}(\{\bar{a} \in \{0,1\}^n : \bar{a} \models \varphi \wedge \psi\})}{\mathbb{P}(\{\bar{a} \in \{0,1\}^n : \bar{a} \models \psi\})}.$$

Example

Let p, q, r be propositional variables.
Consider the following Bayesian network:



$w(p) = 3/4$	$w(r \mid p \wedge q) = 3/4$
$w(q) = 1/3$	$w(r \mid \neg p \wedge q) = 1/2$
	$w(r \mid p \wedge \neg q) = 1/2$
	$w(r \mid \neg p \wedge \neg q) = 1/4$

The Bayesian network defines, in a natural way, a probability distribution $\mathbb{P}$ on $\{0,1\}^3$.

We get, for example,
$\mathbb{P}(p) = 3/4$, $\mathbb{P}(\neg q) = 1 - 1/3 = 2/3$,
$\mathbb{P}(p \wedge \neg q) = 3/4 \cdot 2/3$, and
$\mathbb{P}(p \wedge \neg q \wedge r) = 1/2 \cdot 3/4 \cdot 2/3 = 1/4$.

Possible worlds in first-order logic

In first-order logic a "possible world" is a first-order structure, i.e. a domain (a set of objects) with relations and/or functions on it.

The relations and functions that are of interest in a given context will be "named" by relation and function symbols, respective, which form the first-order language used in the given context.

Once a first-order language L is fixed a *possible world* is an L-structure.

Let $\mathbf{W}$ be a set of L-structures (possible worlds) and suppose that $\mathbb{P}$ is a probability measure on $\mathbf{W}$.

By an *event* we simply mean a measurable subset $\mathbf{E} \subseteq \mathbf{W}$ of possible worlds and its probability is $\mathbb{P}(\mathbf{E})$.

If $\mathbf{W}$ is finite, which is the case in practical situations, then every subset of $\mathbf{W}$ is measurable so we have $\mathbb{P}(\mathbf{E}) = \sum_{\mathcal{A} \in \mathbf{W}} \mathbb{P}(\mathcal{A})$.

Events defined by L-sentences

If φ and ψ are L-sentences then we let

$$\mathbb{P}(\varphi) = \mathbb{P}\big(\{\mathcal{A} \in \mathbf{W} : \mathcal{A} \models \varphi\}\big)$$

and

$$\mathbb{P}(\varphi \mid \psi) = \frac{\mathbb{P}(\varphi \wedge \psi)}{\mathbb{P}(\psi)}.$$

Fixing a domain

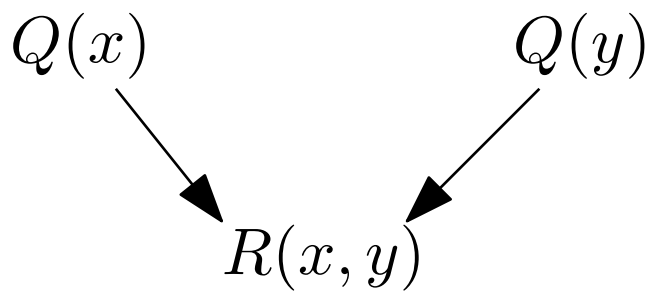
If different structures in $\mathbf{W}$ have different domains then it is harder to define a probability measure on $\mathbf{W}$.

Therefore we will let all structures in $\mathbf{W}$ have the same domain, say D, and *we use the notation* $\mathbf{W}_D$ *for the set of all L-structures with domain D.*

Defining a probability distribution on $\mathbf{W}_D$: an example

The question remains of how, concretely, to define a probability distribution on $\mathbf{W}_D$ for a *finite* domain $D = \{1, \ldots, n\}$ (where for simplicity we identify the objects with numbers).

Let $L = \{Q, R\}$ where Q and R are relation symbols of arity 1 and 2, respectively. Suppose that we have somehow "learned" the following Bayesian network (BN):



$w(Q(x)) = w(Q(y)) = 1/4$	$w(R(x,y)\|Q(x) \wedge Q(y)) = 1/2$ $w(R(x,y)\|Q(x) \wedge \neg Q(y)) = 1/4$ $w(R(x,y)\|\neg Q(x) \wedge Q(y)) = 1/4$ $w(R(x,y)\|\neg Q(x) \wedge \neg Q(y)) = 1/8$

We interpret the BN as saying:

- For every $i \in D$, the probability that $\mathcal{A} \models Q(i)$ for a random $\mathcal{A} \in \mathbf{W}_D$ is $1/4$.
- For all $i, j \in D$, the probability that, for a random $\mathcal{A} \in \mathbf{W}_D$, $\mathcal{A} \models R(i,j)$ is $1/2$ *given that we know that* $\mathcal{A} \models Q(i) \wedge Q(j)$.
- For all $i, j \in D$, the probability that, for a random $\mathcal{A} \in \mathbf{W}_D$, $\mathcal{A} \models R(i,j)$ is $1/4$ *given that we know that* $\mathcal{A} \models Q(i) \wedge \neg Q(j)$.
- and similarly for the remaining cases.

It follows that, for all distinct $i, j \in D$, the probability that $\mathcal{A} \models Q(i) \wedge Q(j) \wedge R(i,j)$ for a random $\mathcal{A} \in \mathbf{W}_D$, is $1/4 \cdot 1/4 \cdot 1/2 = 1/32$.

Following the above idea we define a probability distribution $\mathbb{P}$ on $\mathbf{W}_D$ as follows.

Let $\mathcal{A} \in \mathbf{W}_D$ and let

- $q =$ the number of $i \in D$ such that $\mathcal{A} \models Q(\hat{i})$,
- $r_1 =$ the number of pairs $(i,j) \in D^2$ such that $\mathcal{A} \models R(\hat{i},\hat{j}) \wedge Q(\hat{i}) \wedge Q(\hat{j})$,
- $r_2 =$ the number of pairs $(i,j) \in D^2$ such that $\mathcal{A} \models R(\hat{i},\hat{j}) \wedge Q(\hat{i}) \wedge \neg Q(\hat{j})$,
- $r_3 =$ the number of pairs $(i,j) \in D^2$ such that $\mathcal{A} \models R(\hat{i},\hat{j}) \wedge \neg Q(\hat{i}) \wedge Q(\hat{j})$ and
- $r_4 =$ the number of pairs $(i,j) \in D^2$ such that $\mathcal{A} \models R(\hat{i},\hat{j}) \wedge \neg Q(\hat{i}) \wedge \neg Q(\hat{j})$.

Then we define

$$\begin{aligned}\mathbb{P}(\mathcal{A}) =& (1/4)^q (3/4)^{n-q} (1/2)^{r_1} (1/2)^{q^2-r_1} (1/4)^{r_2} (3/4)^{q(n-q)-r_2} \\ & (1/4)^{r_3} (3/4)^{(n-q)q-r_3} (1/8)^{r_4} (7/8)^{(n-q)^2-r_4}.\end{aligned}$$

Observe that $\mathbb{P}(\mathcal{A})$ only depends on the size of D, the (conditional) probabilities of the BN and the numbers q, r_1, r_2, r_3, r_4.

Let $\varepsilon > 0$ be small and suppose that n (the size of D) is large enough.
By the law of large numbers, with probability at least $1 - \varepsilon$, the following hold for a random $\mathcal{A} \in \mathbf{W}_D$:

- The number of $i \in D$ such that $\mathcal{A} \models Q(i)$ is between $(1/4 - \varepsilon)n$ and $(1/4 + \varepsilon)n$.
- The number of pairs $(i, j) \in D^2$ such that $\mathcal{A} \models Q(i) \wedge Q(j) \wedge R(i, j)$ is between $(1/4 \cdot 1/2 - \varepsilon)n$ and $(1/4 \cdot 1/2 + \varepsilon)n$.
- and similarly for other boolean combinations of $Q(i), Q(j)$ and $R(i, j)$.

So with this probabilistic model, the proportions of objects (or tuples of objects) which satisfy atomic relations "stabilize" as the domain grows. This has nice consequences with respect to predicting probabilities on large domains.

Learning a probabilistic graphical model and the probabilities on domains approach

The approach of considering *probabilities on possible worlds* is useful when making *(probabilistic) inferences*, that is, predicting probabilities of events.

But when making inferences we assume that we already have a probability distribution on the possible worlds.

Now we turn to the problem of constructing a *probabilistic graphical model* which can be used to determine a probability distribution on $\mathbf{W}_D$ for any finite domain D.

When adressing this problem we use the *probabilities on domains approach*.

Probabilities on domains

Let L be a first-order language and let $\mathcal{M}$ be an L-structure with domain M.

Suppose that $\mu : M \to \mathbb{R}$ is a probability distribution and (for $n > 1$) let $\mu^n : M^n \to \mathbb{R}$ be the probability distribution on M^n defined by $\mu^n(a_1, \ldots, a_n) = \mu(a_1) \cdot \ldots \cdot \mu(a_n)$.

Let $\varphi(x_1, \ldots, x_n)$ denote an L-formula such that all free variables of it belong to the sequence $x_1, \ldots, x_n$.

We define *the probability that* $\varphi(x_1, \ldots, x_n)$ *is satisfied by an* n*-tuple from* M, denoted $\mathbb{P}_\mu\big(\varphi(x_1, \ldots, x_n)\big)$, by

$$\mathbb{P}_\mu^{\mathcal{M}}\big(\varphi(x_1, \ldots, x_n)\big) = \mu^n\big(\{(a_1, \ldots, a_n) \in M^n : \mathcal{M} \models \varphi(a_1, \ldots, a_n)\}\big).$$

Formulas as random variables

Let $\varphi(x_1, \ldots, x_n)$ denote an L-formula all of which free variables appear in $x_1, \ldots, x_n$.

We can view $\varphi(x_1, \ldots, x_n)$ as a binary random variable, $\varphi : M^n \to \mathbb{R}$, as follows:

For every $(a_1, \ldots, a_n) \in M^n$, let

$$\varphi(a_1, \ldots, a_n) = \begin{cases} 1 & \text{if } \mathcal{M} \models \varphi(\hat{a}_1, \ldots, \hat{a}_n) \\ 0 & \text{otherwise.} \end{cases}$$

Note that, as a random variable, $\varphi(x_1, \ldots, x_n)$ depends on $\mathcal{M}$, but as a formula it does not depend on $\mathcal{M}$.

Learning a Bayesian network from a structure

We can think of the L-structure $\mathcal{M}$ as representing a data set (e.g. relational database) from which we want to learn a BN.

Let $\varphi_1(x_1, \ldots, x_n), \ldots, \varphi_m(x_1, \ldots, x_n)$ be L-formulas such that all free variables in each φ_i belong to the sequence $x_1, \ldots, x_n$.
Each $\varphi_i(x_1, \ldots, x_n)$ determines a random variable, also denoted φ_i.

Algorithm for finding the arrows/arcs of the BN:

Set $k := 1$ and as long as $k < m$ do:

> For every $i = 1, \ldots, k$, add an arrow from φ_i to φ_{k+1}.
> Set $I := \{1, \ldots, k\}$.
> For $i = 1, \ldots, k$ do:
>
> > If φ_{k+1} and φ_i are conditionally independent over $(\varphi_j : j \in I \setminus \{i\})$, then remove i from I (i.e. set $I := I \setminus \{i\}$) and remove the arrow from φ_i to φ_{k+1}. Otherwise do nothing.
>
> Set $k := k + 1$.

Finding the (conditional) probabilities associated to the vertices

Next, for every vertex φ_k with parents $\varphi_{i_1}, \ldots, \varphi_{i_s}$, say, one computes the conditional probability

$$\mu^n\big(\varphi_k = 1 \mid \varphi_{i_1} = b_1, \ldots, \varphi_{i_s} = b_s\big)$$

for every choice of $b_1, \ldots, b_s \in \{0, 1\}$.

If φ_k has no parent then one just computes $\mu^n(\varphi_k)$.

Remarks:

- If $\mathcal{M}$ represents e.g. a relational database and one uses the mathematical definition of independence then it may be unlikely that any φ_k is (conditionally) independent from some other $\varphi_{j_1}, \ldots, \varphi_{j_s}$. Therefore it may be more practically useful to consider "approximate independence" where an error marginal is allowed.

- When constructing a *probabilistic graphical model* from a learning set of data one wants to avoid *overfittning* the model to the particular dataset used. So, for example, weak dependencies in the structure $\mathcal{M}$ may be ignored in the BN that is constructed.
- The DAG constructed by the algorithm depends on the order of the formulas/random variables $\varphi_1, \ldots, \varphi_n$. Hence one can try different orders and see which one gives the simplest DAG.

See more details, comments and discussions in Section 5.1.

Example of constructing a BN

Suppose that a database contains information about the items among I_1, I_2, I_3 that customers called $A, B, C, \ldots, L$ have bought.

customer	buys I_1	buys I_2	buys I_3
A	yes	no	yes
B	no	yes	no
C	no	no	yes
D	yes	yes	yes
E	no	no	no
F	yes	no	yes
G	yes	yes	yes
H	no	no	no
I	no	yes	yes
J	yes	yes	yes
K	no	no	yes
L	yes	yes	yes

The above information can be represented by an L-structure $\mathcal{M}$ as follows, where $L = \{I_1, I_2, I_3\}$ and I_1, I_2, I_3 are unary relation symbols.

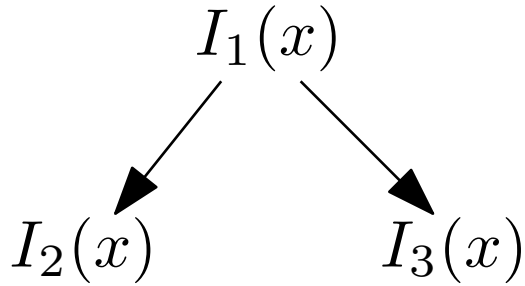
The domain of $\mathcal{M}$ is $M = \{A, B, C, \ldots, L\}$ and

$$I_1^{\mathcal{M}} = \{A, D, F, G, J, L\},$$
$$I_2^{\mathcal{M}} = \{B, D, G, I, J, L\},$$
$$I_3^{\mathcal{M}} = \{A, C, D, F, G, I, K, L\}.$$

Let $\mu : M \to \mathbb{R}$ be the uniform probability measure (also called *counting measure*).
Each formula $I_1(x), I_2(x), I_3(x)$ can also be seen as a binary random variable.

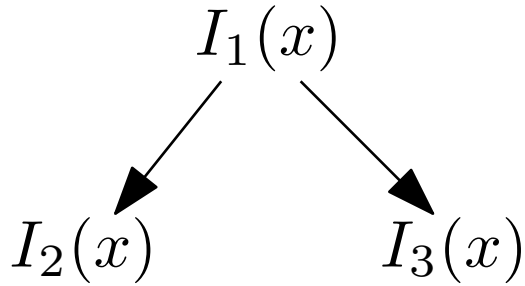
For example, we have
$\mu(I_1 = 1) = \mu(I_1(x)) = 1/2$, $\mu(I_2 = 1 \mid I_1 = 1) = 2/3$, and
$\mu(I_2 = 1 \mid I_1 = 0) = 1/3$.

If we use the algorithm given above with the order I_1, I_2, I_3 we get the following BN.



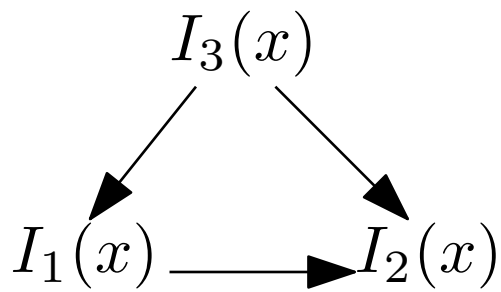
$\mu(I_1(x)) = 1/2$	$\mu(I_2(x) \mid I_1(x)) = 2/3$ $\mu(I_2(x) \mid \neg I_1(x)) = 1/3$	$\mu(I_3(x) \mid I_1(x)) = 1$ $\mu(I_3(x) \mid \neg I_1(x)) = 1/2$

If we use the algorithm given above with the order I_1, I_2, I_3 we get the following BN.



$\mu(I_1(x)) = 1/2$	$\mu(I_2(x) \mid I_1(x)) = 2/3$ $\mu(I_2(x) \mid \neg I_1(x)) = 1/3$	$\mu(I_3(x) \mid I_1(x)) = 1$ $\mu(I_3(x) \mid \neg I_1(x)) = 1/2$

If we instead use the order I_3, I_1, I_2 we get the following DAG:



Using the model for inference/prediction, via the probabilities on possible worlds approach

Let us consider the first BN constructed above (using the order I_1, I_2, I_3.
It was learned from a data set (or structure) by using the "probabilities on domains approach" on a particular domain of customers $\{A, B, C, \ldots, L\}$.

Now we want to use it to predict the behaviour of customers in general. To do this we want, for any set N of customers, to assign a probability to every possible scenario regarding the customers in N (and we call such a scenario a possible world).

Suppose $N = \{1, \ldots, n\}$, for simplicity, and let $\mathbf{W}$ be the set of all L-structures with domain N, where $L = \{I_1, I_2, I_3\}$.

We interpret the (first) BN as saying that, for every $i \in N$, the formula in the left column below holds with the probability given in the right column independently of what is the case for $j \neq i$.

$I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot 1 = 1/3$
$\neg I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} = 1/12$
$I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot 1 = 1/6$
$I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot 0 = 0$
$\neg I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{2} = 1/6$
$I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot 0 = 0$
$\neg I_1(\hat{i}) \wedge I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} = 1/12$
$\neg I_1(\hat{i}) \wedge \neg I_2(\hat{i}) \wedge \neg I_3(\hat{i})$	$\frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{2} = 1/6$

Let $\mathcal{N} \in \mathbf{W}$ and, for $l = 1, \ldots, 8$, let k_l be the number of $i \in N$ such that the formula on the l^{th} row in the table above holds in $\mathcal{N}$. Then define

$$\mathbb{P}(\mathcal{N}) = (1/3)^{k_1}(1/12)^{k_2}(1/6)^{k_3}0^{k_4}(1/6)^{k_5}0^{k_6}(1/12)^{k_7}(1/6)^{k_8}$$

and recall that $0^0 = 1$ and $0^c = 0$ if $c > 0$.

Using the above probability distribution on $\mathbf{W}$ we can, for example, infer that

$$\mathbb{P}\Big(\forall x\big((I_1(x) \wedge I_2(x)) \rightarrow I_3(x)\big)\Big) = 1$$

and

$$\mathbb{P}\Big(\forall x\big(I_3(x) \rightarrow (I_1(x) \vee I_2(x))\big)\Big) = (5/6)^n.$$

See the details in the argument at the end of Section 5.2.1.

Learning when data is missing

Now suppose that some data is missing, as in the following table which is otherwise like the table in the previous example.

customer	buys I_1	buys I_2	buys I_3
A	yes	no	yes
B	no	yes	no
C		no	yes
D	yes	yes	yes
E	no	no	no
F	yes	no	yes
G	yes	yes	
H	no	no	no
I	no	yes	yes
J		yes	yes
K	no	no	yes
L	yes	yes	yes

Note that we do not have full information about which customers bought I_1.

We may still want to estimate the probability that a random customer buys I_1, given the information in the table.

This can be done by computing the *maximum likelyhood estimate*.

The *Expectation Maximisation algorithm*, or *EM algorithm*, can be used, which in this context works as follows.

Let X_A, X_B, ..., X_L be random variables defined as follows for every $\Lambda \in \{A, B, \ldots, L\}$:

$$X_\Lambda = \begin{cases} 1 \text{ if } \Lambda \text{ bought } I_1, \\ 0 \text{ if } \Lambda \text{ did not buy } I_1. \end{cases}$$

We denote "the probability that $X_\Lambda = 1$" by "$\mathbb{P}(X_\Lambda = 1)$".

- If we know that a customer Λ bought I_1 we let $\mathbb{P}(X_\Lambda = 1) = 1$.
- If we know that a customer Λ did not buy I_1 we let $\mathbb{P}(X_\Lambda = 1) = 0$.
- If we do not know if a customer Λ bought I_1 we let $\mathbb{P}(X_\Lambda = 1) = p_0$, where p_0 is an initial "guess".

Let $X = X_A + X_B + \ldots + X_L$, so X is the number of customers who bought I_1.
Next we compute the expected value of X (called the *expected count*):

$$\begin{aligned}\mathbb{E}(X) &= \mathbb{E}(X_A) + \mathbb{E}(X_B) + \ldots + \mathbb{E}(X_L) \\ &= \mathbb{P}(X_A = 1) + \mathbb{P}(X_B = 1) + \ldots + \mathbb{P}(X_L = 1) = 5 + 2p_0.\end{aligned}$$

Since there are 12 customers in total, the next estimate of the probability that a random customer bought I_1 is now set to

$$p_1 = \frac{\mathbb{E}(X)}{12} = \frac{5 + 2p_0}{12}.$$

Now we repeat the procedure with the estimate p_1 (in place of p_0) which gives the new estimate $p_2 = \frac{5+2p_1}{12}$.

More generally, once we have an estimate p_n, the above procedure gives the new estimate

$$p_{n+1} = \frac{5 + 2p_n}{12}.$$

If $0 < p_0 < 1$ then p_n converges as $n \to \infty$ and $lim_{n\to\infty} p_n = 1/2$.

Thus the maximum likelihood estimate of the probability that a customer buys I_1 is $1/2$.

It turns out that there is a quicker way to compute the same estimate, as justified by the following result:

Proposition 5.3.1 in the Notes:

Let (S, μ) be a probability space where S is finite and μ is the *counting measure*, i.e. $\mu(s) = 1/|S|$ for all $s \in S$.

Let $X : S \to \{0, 1\}$ be a random variable and suppose that $S = S_1 \cup S_0 \cup S_u$, $X(s) = 1$ for all $s \in S_1$, $X(s) = 0$ for all $s \in S_0$ and suppose that we have no information about the value $X(s)$ if $s \in S_u$.

If $S_u \neq S$ (so that we know the value $X(s)$ for at least one $s \in S$) then the EM algorithm converges and the estimate of $\mu(\{s \in S : X(s) = 1\})$ to which the EM algorithm converges is equal to $\frac{|S_1|}{|S_1 \cup S_0|}$.

So to estimate $\mathbb{P}(I_1(x))$ we could just have counted the entries with 'yes' in the column for I_3 and divided it with the number of entries with either 'yes' or 'no', which also gives $1/2$.

Now let's estimate the probability $\mathbb{P}(I_3(x) \mid I_1(x))$.
As

$$\mathbb{P}(I_3(x) \mid I_1(x)) = \frac{\mathbb{P}(I_3(x) \wedge I_1(x))}{\mathbb{P}(I_1(x))}$$

we need to estimate $\mathbb{P}(I_3(x) \wedge I_1(x))$.

By the proposition above we just count the number of customers who bought both I_1 and I_3 and the number of customers for which we can answer the following question with certainty:
Did the customer buy both I_1 and I_3?

We get the numbers 4 and 9, so the estimate of $\mathbb{P}(I_3(x) \wedge I_1(x))$ is $4/9$.

It follows that the estimate of $\mathbb{P}(I_3(x) \mid I_1(x))$ is

$$\frac{4/9}{1/2} = 8/9.$$

More general BN's

The BN's considered so far has had the following property:

Every conditional probability associated to a random variable has dependend only on a *boolean combination of values of its parents* in the DAG.

However, we may want to allow conditional probabilies (associated to a BN) to depend on the parents of a vertex/random variable in other, more complex, ways.

For example, logical quantifiers, or aggregation functions, can be used.

Example using first-order quantification

Let's think of $C(x, y)$ as meaning "x and y are (somehow) connected", and $S(x)$ as meaning "x is sick". Let $\varphi(x)$ denote the following formula:

$$\exists y_1 \exists y_2 \exists y_3 \big(y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3 \wedge C(x, y_1) \wedge C(x, y_2) \wedge C(x, y_3)\big).$$

Let us consider a BN with an arrow from $C(x, y)$ to $S(x)$ and with the following conditional probabilities associated to the vertices $C(x, y)$ and $S(x)$:

$$\boxed{\begin{array}{l} \mu(C(x, y)) = 1/100 \\ \mu(S(x) \mid \varphi(x)) = 1/2 \\ \mu(S(x) \mid \neg\varphi(x)) = 1/4 \end{array}}$$

Note that if D is a domain and $a \in D$ to know if $\varphi(a)$ holds we may have to check, *for all* $b \in D$, if $C(a, b)$ holds, so $S(a)$ depends on all $C(a, d_1), \ldots, C(a, d_n)$, if $D = \{d_1, \ldots, d_n\}$.

Determining the probability of an event

Let D be a domain with size n for some large n. The BN defines a probabity distribution on the set $\mathbf{W}_D$ of possible worlds with domain D as described in Chapter 6 of my notes. But here I will argue in an informal way.

Suppose that we want to understand how likely it is that the following statement is true: *At least one fifth of the population (in D) is sick*.

A brute force way of determining whether it is true is the following:

Consider each one of the roughly 2^{n^2} structures (possible worlds) in $\mathbf{W}_D$ and for each such structure check whether at least one fifth of the population is sick and if so compute the probability of the structure, and then sum the probabilities of all the structures in which at least one fifth of the population is sick.

Of course this procedure is not computationally efficient for large domains as it requires exponential time in the size n of the domain.

Next we illustrate that we can answer the question by a procedure which is independent of the domain size

Determining the probability of an event

By the law of large numbers and the Chernoff bound, it holds that, for every $\varepsilon > 0$ and every $a \in D$ and sufficiently large n, with probability at least $1 - e^{-cn}$, the proportion of $b \in D$ such that $C(a, b)$ holds is in $(0.01 - \varepsilon, 0.01 + \varepsilon)$, where $c > 0$ is a constant that depends only on ε.

As there are (only) n choices of a, there is $C > 0$ such that, with probability $1 - e^{-Cn}$, for *all* $a \in D$ the proportion of $b \in D$ such that $C(a, b)$ holds is in $(0.01 - \varepsilon, 0.01 + \varepsilon)$. It follows that (if n is large enough) then, with probability at least $1 - e^{-Cn}$, $\varphi(a)$ holds for all $a \in D$.

Next, note that if $\varphi(a)$ holds for all $a \in D$ then the probability that $S(a)$ holds is $1/2$ for all $a \in D$, independently of what the case is for other $a' \in D$.

Hence, with probability tending to 1 as $n \to \infty$, approximately one half (and thus at least one fifth) of the population is sick.

Another example

Still consider a BN with an arrow from $C(x, y)$ to $S(x)$ but now suppose that the conditional probabilities associated to the vertices are as follows:

$$\boxed{\begin{array}{l}\mu(C(x,y)) = 1/100 \\ \mu(S(x)) = \text{the proportion of } y \text{ (in the domain)} \\ \text{such that } C(x,y) \text{ holds.}\end{array}}$$

By the argument already given, for every $\varepsilon > 0$ there is $C > 0$ such that, with probability $1 - e^{-Cn}$,

$(*)$ for *all* $a \in D$ the proportion of $b \in D$ such that $C(a, b)$ holds is in $(0.01 - \varepsilon, 0.01 + \varepsilon)$ if n large enough.

Conditioned on $(*)$ the probability that $S(a)$ holds is in $(0.01 - \varepsilon, 0.01 + \varepsilon)$ for all $a \in D$. So given the same condition the proportion of $a \in D$ such that $S(a)$ holds is close to 0.01.

It follows that with probability tending to 1 as $n \to \infty$, the proportion of the population which is sick is close to $1/100$.

Recall the statement:
At least one fifth of the population is sick.

In the first example above the probability that this statement is true tends to 1 as the domain size $n \to \infty$.
In the second example above the probability that this statement is true tends to 0 as the domain size $n \to \infty$.

We need not always have convergence to 0 or 1, or convergence at all. It depends on the probabilistic graphical model used and the statement in question.

But it turns out that for several types of BN's, if an event can be expressed by first-order logic or certain other logics (of interest in AI), then its probability converges as the domain size tends to infinity.

This can be used to estimate probabilities in a computationally efficient way.

Convergence and random sampling

Let $\mathbb{G}$ be a graphical model, D_n a domain of size n, and let $\mathbb{P}_n$ the probability distribution on $\mathbf{W}_n$ (the set of possible worlds with domain D_n).

Let φ be a logical formula which, for every n, defines an event $\mathbf{E}_n := \{\mathcal{A} \in \mathbf{W}_n : \mathcal{A} \models \varphi\}$. *Suppose that we know, by some theoretical result, that* $\lim_{n\to\infty} \mathbb{P}_n(\mathbf{E}_n)$ *exists.* Let α be the limit (which we may not know).

Let some "error marginal" $\varepsilon > 0$ be given.
By the law of large numbers there is an integer k such that, for any large enough n, if we take k random samples $\mathcal{A}_1, \ldots, \mathcal{A}_k \in \mathbf{W}_n$, then, with probability at least $1 - \varepsilon$, the proportion of the samples which belong to $\mathbf{E}_n$ is in $[\alpha - \varepsilon, \alpha + \varepsilon]$.

To know if $\mathcal{A}_i \in \mathbf{E}_n$ we need to know if $\mathcal{A}_i \models \varphi$, but this can be done in time bounded by $\mathcal{O}(n^p)$, where p is the length of φ.
Moreover, the estimate thus obtained is, with probability at least $1 - \varepsilon$, valid for every $n' \geq n$.

"Semantic analysis of the formula and the BN"

Suppose that φ is a logical formula, $\mathbb{G}$ a graphical model, $\mathbf{E}_n := \{\mathcal{A} \in \mathbf{W}_n : \mathcal{A} \models \varphi\}$, and that $\mathbb{P}_n$ is the probability distribution on $\mathbf{W}_n$ which is determined by $\mathbb{G}$.

In the two concrete examples that we saw above we estimated $\lim_{n\to\infty} \mathbb{P}_n(\mathbf{E}_n)$ (without using this notation) without making any random sampling .

Instead we analyzed the meaning of the BN and the meaning of the statement/formula that defined the event under consideration. By that procedure we estimated the asymptotic probability of the event in a way that was independent of the size of the domain, i.e. the time needed is constant with respect to domain size.

This is relevant since, typically, it is the domain size which is the large parameter (e.g. in the order of 10^5 or more) while the length of the logical formula that defines an event is much smaller (say ≤ 1000 or smaller).

Asymptotic elimination of quantifiers and aggregation functions

The informally explained arguments on the previous slide can be made precise for various types of logics and BNs. The results obtained are often called "almost sure (or asymptotic) elimination of quantifiers (or aggregation functions)", and they often have a convergence result for probabilities of events that can be expressed by the logic under consideration.

See for example Section 6 in my notes for results about first-order logic and "lifted Bayesian first-order networks".

Results also exist for "conditional probability logic (CPL)" and lifted Bayesian networks using CPL-formulas,

as well as for "probability logic with aggregation functions (PLA)" and "PLA-networks".

More discussion in this direction (with references) is found in Section 6.3 of my notes.