Uncertainty: Probability as Degree of Belief

At the start of the course, I presented a uniform approach to knowledge representation and reasoning using probability. The world:

\[ V = \{ V_1, V_2, \ldots, V_n \} \]

Query:

\[ Q = \{ Q_1, Q_2, \ldots, Q_q \} \]

Observed:

\[ o = \{ o_1, o_2, \ldots, o_m \} \]

Latent variables:

\[ L = \{ L_1, L_2, \ldots, L_l \} \]

The world is represented by RVs \( V = \{ V_1, V_2, \ldots, V_n \} \). These are partitioned:

1. Query variables \( Q = \{ Q_1, Q_2, \ldots, Q_q \} \). We want to compute a distribution over these.
2. Observed variables \( O = \{ o_1, o_2, \ldots, o_m \} \). We know the values of these.
3. Latent variables \( L = \{ L_1, L_2, \ldots, L_l \} \). Everything else.

To compute a conditional distribution from a knowledge base \( \Pr(V) \) we have to sum over the latent variables:

\[
\Pr(Q|o_1, o_2, \ldots, o_m) = \sum_L \Pr(Q, L|o_1, o_2, \ldots, o_m) = \frac{1}{Z} \sum_L \Pr(Q, L|o_1, o_2, \ldots, o_m) \\
\]

Bayes’ theorem tells us how to update an inference when new information is available.

For example, if we now receive a new observation \( O' = o' \) then

\[
\Pr(Q|o', o_1, o_2, \ldots, o_m) = \frac{1}{Z} \Pr(o'|Q, o_1, o_2, \ldots, o_m) \Pr(Q|o_1, o_2, \ldots, o_m) \\n\]

After \( O' \) observed

Before \( O' \) observed

Knowledge base
General knowledge representation and inference: the BIG PICTURE

Simple eh?

HAH!! No chance...

Even if all your RVs are just Boolean:

• For $n$ RVs knowing the knowledge base $\Pr(V)$ means storing $2^n$ numbers.
• So it looks as though storage is $O(2^n)$.
• You need to establish $2^n$ numbers to work with.
• Look at the summations. If there are $n$ latent variables then it appears that time complexity is also $O(2^n)$.
• In reality we might well have $n > 1000$, and of course it’s even worse if variables are non-Boolean.

And it really is this hard. The problem in general is #P-complete.

Even getting an approximate solution is provably intractable.

---

Bayesian Networks

Having seen that in principle, if not in practice, the full joint distribution alone can be used to perform any inference of interest, we now examine a practical technique.

• We introduce the Bayesian Network (BN) as a compact representation of the full joint distribution.
• We examine the way in which a BN can be constructed.
• We examine the semantics of BNs.
• We look briefly at how inference can be performed.
• We briefly introduce the Markov random field (MRF) as an alternative means of representing a distribution.

---

Conditional probability—a brief aside...

A brief aside on the dangers of interpreting implication versus conditional probability:

• $\Pr(X = x|Y = y) = 0.1$ does not mean that if $Y = y$ is then $\Pr(X = x) = 0.1$.
• $\Pr(X)$ is a prior probability. It applies when you haven’t seen the value of $Y$.
• The notation $\Pr(X|Y = y)$ is for use when $y$ is the entire evidence.
• $\Pr(X|Y = y \land Z = z)$ might be very different.

Conditional probability is not analogous to logical implication.

---

Implication and conditional probability

In general, it is difficult to relate implication to conditional probability.

\[
\Pr(A \rightarrow B) = \Pr(\neg A \lor B)
\]
\[
\Pr(A|B) = \frac{\Pr(A,B)}{\Pr(B)}
\]

Imagine that fish are very rare, and most fish can swim.

With implication,

\[
\Pr(\text{fish} \rightarrow \neg \text{swim}) = \Pr(\neg \text{fish} \lor \neg \text{swim}) = \text{LARGE!}
\]

With conditional probability,

\[
\Pr(\neg \text{swim}|\text{fish}) = \frac{\Pr(\neg \text{swim} \land \text{fish})}{\Pr(\text{fish})} = \text{SMALL!}
\]
Bayesian networks: exploiting independence

One of the key reasons for the introduction of Bayesian networks is to let us exploit independence.

The initial pay-off is that this makes it easier to represent $\Pr(V)$.
A further pay-off is that it introduces structure that can lead to more efficient inference.

Here is a very simple example.

If I toss a coin and roll a die, the full joint distribution of outcomes requires $2 \times 6 = 12$ numbers to be specified.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>$T$</td>
<td>0.015</td>
<td>0.033</td>
<td>0.022</td>
<td>0.067</td>
</tr>
</tbody>
</table>

Here $\Pr(\text{Coin} = H) = 0.3$ and the die has probability $i/21$ for the $i$th outcome.

Exploiting independence

A slightly more complex example:

<table>
<thead>
<tr>
<th></th>
<th>CP</th>
<th>¬CP</th>
<th>SB</th>
<th>¬SB</th>
<th>SB</th>
<th>¬SB</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬HD</td>
<td>0.024</td>
<td>0.006</td>
<td>0.016</td>
<td>0.008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HD</td>
<td>0.0019</td>
<td>0.0076</td>
<td>0.1881</td>
<td>0.7524</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- HD = Heart disease
- CP = Chest pain
- SB = Shortness of breath

Similarly, say instead of just considering HD, SB and CP we also consider the outcome of the Oxford versus Cambridge tiddlywinks competition TC:

TC = {Oxford, Cambridge, Draw}.

Exploiting independence

BUT: if we assume the outcomes are independent then

$\Pr(\text{Coin, Dice}) = \Pr(\text{Coin}) \Pr(\text{Dice})$

Where $\Pr(\text{Coin})$ has two numbers and $\Pr(\text{Dice})$ has six.

So instead of 12 numbers we only need 8.

Now

$\Pr(\text{HD, SB, CP, TC}) = \Pr(\text{TC}|\text{HD, SB, CP}) \Pr(\text{HD, SB, CP})$.

Assuming that the patient is not an extraordinarily keen fan of tiddlywinks, their cardiac health has nothing to do with the outcome, so

$\Pr(\text{TC}|\text{HD, SB, CP}) = \Pr(\text{TC})$

and $2 \times 2 \times 2 \times 3 = 24$ numbers has been reduced to $3 + 8 = 11$. 
Conditional independence

However although in this case we might not be able to exploit independence directly we can say that

$$\Pr(CP, SB|HD) = \Pr(CP|HD) \Pr(SB|HD)$$

which simplifies matters.

Conditional independence: $A \perp B|C$

- $A$ is conditionally independent of $B$ given $C$, written $A \perp B|C$, if
  $$\Pr(A, B|C) = \Pr(A|C) \Pr(B|C).$$
- If we know that $C$ is the case then $A$ and $B$ are independent.
- Equivalently $\Pr(A|B, C) = \Pr(A|C)$, (Prove this!)

Although $CP$ and $SB$ are not independent, they do not directly influence one another in a patient known to have heart disease.

This is much nicer!

$$\Pr(HD|CP, SB) \propto \Pr(CP|HD) \Pr(SB|HD) \Pr(HD)$$

Bayesian networks

After a regrettable incident involving an inflatable gorilla, a famous College has decided to install an alarm for the detection of roof climbers.

- The alarm is very good at detecting climbers.
- Unfortunately, it is also sometimes triggered when one of the extremely fat geese that lives in the College lands on the roof.
- One porter’s lodge is near the alarm, and inhabited by a chap with excellent hearing and a pathological hatred of roof climbers: he always reports an alarm. His hearing is so good that he sometimes thinks he hears an alarm, even when there isn’t one.
- Another porter’s lodge is a good distance away and inhabited by an old chap with dodgy hearing who likes to listen to his collection of DEATH METAL with the sound turned up.

Bayesian networks

Also called probabilistic/belief/causal networks or knowledge maps.

- Each node is a random variable (RV).
- Each node $N_i$ has a distribution $\Pr(N_i|\text{parents}(N_i))$
- A Bayesian network is a directed acyclic graph.
- Roughly speaking, an arrow from $N$ to $M$ means $N$ directly affects $M$. 

Also called probabilistic/belief/causal networks or knowledge maps.

- Each node is a random variable (RV).
- Each node $N_i$ has a distribution $\Pr(N_i|\text{parents}(N_i))$
- A Bayesian network is a directed acyclic graph.
- Roughly speaking, an arrow from $N$ to $M$ means $N$ directly affects $M$. 

Bayesian networks

14

Also called probabilistic/belief/causal networks or knowledge maps.

- Each node is a random variable (RV).
- Each node $N_i$ has a distribution $\Pr(N_i|\text{parents}(N_i))$
- A Bayesian network is a directed acyclic graph.
- Roughly speaking, an arrow from $N$ to $M$ means $N$ directly affects $M$. 

Bayesian networks

15

Also called probabilistic/belief/causal networks or knowledge maps.

- Each node is a random variable (RV).
- Each node $N_i$ has a distribution $\Pr(N_i|\text{parents}(N_i))$
- A Bayesian network is a directed acyclic graph.
- Roughly speaking, an arrow from $N$ to $M$ means $N$ directly affects $M$. 

Bayesian networks

16
Bayesian networks

Note that:

- In the present example all RVs are discrete (in fact Boolean) and so in all cases $\Pr(N_i|\text{parents}(N_i))$ can be represented as a table of numbers.
- Climber and Goose have only prior probabilities.
- All RVs here are Boolean, so a node with $p$ parents requires $2^p$ numbers.

A BN with $n$ nodes represents the full joint probability distribution for those nodes as

$$
\Pr(N_1,...,N_n) = \Pr(N_n|N_{n-1},...,N_1) \Pr(N_{n-1}|N_{n-2},...,N_1) \cdots \Pr(N_1).$$

Repeating this gives

$$
\Pr(N_1,...,N_n) = \Pr(N_n|N_{n-1},...,N_1) \prod_{i=1}^{n} \Pr(N_i|N_{i-1},...,N_1).$$

Now compare equations. We see that BNs make the assumption

$$
\Pr(N_i|N_{i-1},...,N_1) = \Pr(N_i|\text{parents}(N_i))
$$

for each node, assuming that $\text{parents}(N_i) \subseteq \{N_{i-1},...,N_1\}$.

Each $N_i$ is conditionally independent of its predecessors given its parents.

Semantics

- When constructing a BN we want to make sure the preceding property holds.
- This means we need to take care over ordering.
- In general causes should directly precede effects.

Here, $\text{parents}(N_i)$ contains all preceding nodes having a direct influence on $N_i$.

But it's not quite that straightforward: what if we want to talk about nodes other than predecessors and parents?

For example, it is possible to show:

Any node $A$ is conditionally independent of the $N_i$—its non-descendants—given the $P_i$—its parents.
Semantics

It is also possible to show:

Any node $A$ is conditionally independent of all other nodes given the Markov blanket $M_i$—that is, its parents, its children and its children’s parents.

Semantics: what’s REALLY going on here?

There is a general method for inferring exactly what conditional independences are implied by a Bayesian network.

Let $X$, $Y$ and $Z$ be disjoint subsets of the RVs.

Consider a path $p$ consisting of directed (in any orientation) edges from some $x \in X$ to some $y \in Y$. For example

The path $p$ is said to be blocked by $Z$ if one of three conditions holds…

Semantics: what’s REALLY going on here?

Path $p$ is blocked with respect to $Z$ if:

1. $p$ contains a node $z \in Z$ that is tail-to-tail:

2. $p$ contains a node $z \in Z$ that is head-to-tail:

3. $p$ contains a node $N$ that is head-to-head, $N \notin Z$, and none of $N$’s descendents is in $Z$:

Finally:

1. $X$ and $Y$ are d-separated by $Z$ if all paths $p$ from some $x \in X$ to some $y \in Y$ are blocked.
2. If $X$ and $Y$ are d-separated by $Z$ then $X \perp Y | Z$. 
More complex nodes

How do we represent \( \Pr(N_i|\text{parents}(N_i)) \) when nodes can denote general discrete and/or continuous RVs?

- BNs containing both kinds of RV are called hybrid BNs.
- Naive discretisation of continuous RVs tends to result in both a reduction in accuracy and large tables.
- \( O(2^p) \) might still be large enough to be unwieldy.
- We can instead attempt to use standard and well-understood distributions, such as the Gaussian.
- This will typically require only a small number of parameters to be specified.

Example:

- A continuous RV with one continuous and one discrete parent.

The probit distribution

\[
\Phi(x) = \int_{-\infty}^{x} N(y)dy
\]

Pr(GRC = true|size) with \( t = 100 \) and different values of \( s \)

\[
\Phi(t - \text{size}/s)
\]

Example: a continuous RV with one continuous and one discrete parent.

\[
\Pr(\text{Speed of car}|\text{Throttle position}, \text{Tuned engine})
\]

where SC and TP are continuous and TE is Boolean.

- For a specific setting of \( ET = \text{true} \) it might be the case that SC increases with TP, but that some uncertainty is involved

\[
\Pr(\text{SC}|\text{TP}, ET) = N(g_{et}\text{TP} + c_{et}, \sigma^2_{et}).
\]

- For an un-tuned engine we might have a similar relationship with a different behaviour

\[
\Pr(\text{SC}|\text{TP}, \neg ET) = N(g_{\neg et}\text{TP} + c_{\neg et}, \sigma^2_{\neg et}).
\]

There is a set of parameters \( \{g, c, \sigma\} \) for each possible value of the discrete RV.

Example: a discrete RV with a continuous parent

\[
\Pr(\text{Go roofclimbing}|\text{Size of fine})
\]

We could for example use the probit distribution

\[
\Phi(t - \text{size}/s) = \Phi\left(\frac{t - \text{size}}{s}\right)
\]

where

\[
\Phi(x) = \int_{-\infty}^{x} N(y)dy
\]

and \( N \) is the Gaussian density with zero mean and variance 1.

\[
\Phi\left(\frac{t - \text{size}}{s}\right)
\]

Pr(GRC = true|size) with \( t = 100 \) and different values of \( s \)
Basic inference

We saw earlier that the full joint distribution can be used to perform all inference tasks:

$$\Pr(Q|o_1, o_2, \ldots, o_m) = \frac{1}{Z} \sum_L \Pr(Q, L, o_1, o_2, \ldots, o_m)$$

where

- $Q$ is the query.
- $o_1, o_2, \ldots, o_m$ are the observations.
- $L$ are the latent variables.
- $1/Z$ normalises the distribution.
- The query, observations and latent variables are a partition of the set $V = \{V_1, V_2, \ldots, V_n\}$ of all variables.

Performing exact inference

$\Pr(Q, L, o_1, o_2, \ldots, o_m)$ has a particular form expressing conditional independences:

$$\Pr(C|l_1, l_2) \propto \sum_A \sum_G \Pr(C) \Pr(G) \Pr(A|C, G) \Pr(l_1|A) \Pr(l_2|A).$$

Consider the computation of the query $\Pr(C|l_1, l_2)$

We have

$$\Pr(C|l_1, l_2) \propto \sum_A \sum_G \Pr(C) \Pr(G) \Pr(A|C, G) \Pr(l_1|A) \Pr(l_2|A).$$

Here there are 5 multiplications for each set of values that appears for summation, and there are 4 such values.

In general this gives time complexity $O(n2^n)$ for $n$ Boolean RVs.

The naive implementation of this approach yields the Enumerate-Joint-Ask algorithm, which unfortunately requires $O(2^n)$ time and space for $n$ Boolean RVs.

The enumeration_ask algorithm improves matters to $O(2^n)$ time and $O(n)$ space by performing the computation depth-first.

However matters can be improved further by avoiding duplication of computations.
Performing exact inference

Looking more closely we see that

\[ \Pr (C|l_1,l_2) \propto \sum_A \sum_G \Pr (C) \Pr (G) \Pr (A|C,G) \Pr (l_1|A) \Pr (l_2|A) \]

This is a result of introducing assumptions about conditional independence.

There is some freedom in terms of how we factorize the expression. This is a result of introducing assumptions about conditional independence.

Performing exact inference: variable elimination

Similarly for \( \Pr (A|C,G) \), which is dependent on \( A, C \) and \( G \)

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>( F_A(A,C,G) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>( \top )</td>
<td>( \top )</td>
<td>0.98</td>
</tr>
<tr>
<td>( \top )</td>
<td>( \top )</td>
<td>( \bot )</td>
<td>0.96</td>
</tr>
<tr>
<td>( \top )</td>
<td>( \bot )</td>
<td>( \top )</td>
<td>0.2</td>
</tr>
<tr>
<td>( \top )</td>
<td>( \bot )</td>
<td>( \bot )</td>
<td>0.08</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \top )</td>
<td>( \top )</td>
<td>0.02</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \top )</td>
<td>( \bot )</td>
<td>0.04</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot )</td>
<td>( \top )</td>
<td>0.8</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot )</td>
<td>( \bot )</td>
<td>0.92</td>
</tr>
</tbody>
</table>

Can we write \( \Pr (A|C,G) \Pr (l_1|A) \Pr (l_2|A) \) as

\( F_A(A,C,G)F_{L1}(A)F_{L2}(A) \)

in a reasonable way?

Performing exact inference: variable elimination

Taking the second possibility:

\[ \Pr (C) \sum_G \Pr (G) \sum_A \Pr (A|C,G) \Pr (l_1|A) \Pr (l_2|A) \]

where \( C, G, A, L_1, L_2 \) denote the relevant factors.

The basic idea is to evaluate this from right to left (or in terms of the tree, bottom up) storing results as we progress and re-using them when necessary.

\[ \Pr (l_1|A) \] depends on the value of \( A \). We store it as a table \( F_{L1}(A) \). Similarly for \( \Pr (l_2|A) \).

\[ F_{L1}(A) = \begin{pmatrix} 0.99 \\ 0.08 \end{pmatrix} \]

\[ F_{L2}(A) = \begin{pmatrix} 0.6 \\ 0.001 \end{pmatrix} \]

as \( \Pr (l_1|a) = 0.99, \Pr (l_1|\neg a) = 0.08 \) and so on.

Performing exact inference: variable elimination

Yes, provided multiplication of factors is defined correctly. Looking at

\[ \Pr (C) \sum_G \Pr (G) \sum_A \Pr (A|C,G) \Pr (l_1|A) \Pr (l_2|A) \]

note that:

1. The values of the product

\[ \Pr (A|C,G) \Pr (l_1|A) \Pr (l_2|A) \]

in the summation over \( A \) depend on the values of \( C \) and \( G \) external to it, and the values of \( A \).

2. So

\[ F_A(A,C,G)F_{L1}(A)F_{L2}(A) \]

should be a table collecting values where correspondences between RVs are maintained.

This leads to a definition for multiplication of factors best given by example.
Performing exact inference: variable elimination

\[ F(A, B)F(B, C) = F(A, B, C) \]

where

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F(A, B)</th>
<th>B</th>
<th>C</th>
<th>F(B, C)</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F(A, B, C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>0.3</td>
<td>T</td>
<td>T</td>
<td>0.1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>0.3 \times 0.1</td>
</tr>
<tr>
<td>T</td>
<td>⊥</td>
<td>0.9</td>
<td>⊥</td>
<td>T</td>
<td>0.8</td>
<td>T</td>
<td>⊥</td>
<td>T</td>
<td>0.3 \times 0.8</td>
</tr>
<tr>
<td>⊥</td>
<td>T</td>
<td>0.4</td>
<td>⊥</td>
<td>T</td>
<td>0.8</td>
<td>⊥</td>
<td>⊥</td>
<td>T</td>
<td>0.9 \times 0.8</td>
</tr>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>0.1</td>
<td>⊥</td>
<td>⊥</td>
<td>0.3</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>0.9 \times 0.3</td>
</tr>
</tbody>
</table>

This process gives us

\[ F_A(A, C, G)F_L(A)F_L2(A) = \]

<table>
<thead>
<tr>
<th>C</th>
<th>G</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>⊤</td>
<td>⊤</td>
<td>0.98 \times 0.99 \times 0.6</td>
</tr>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>0.96 \times 0.99 \times 0.6</td>
</tr>
<tr>
<td>⊥</td>
<td>⊤</td>
<td>0.2 \times 0.99 \times 0.6</td>
</tr>
<tr>
<td>⊤</td>
<td>⊥</td>
<td>0.08 \times 0.99 \times 0.6</td>
</tr>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>0.02 \times 0.08 \times 0.001</td>
</tr>
<tr>
<td>⊥</td>
<td>⊤</td>
<td>0.04 \times 0.08 \times 0.001</td>
</tr>
<tr>
<td>⊤</td>
<td>⊥</td>
<td>0.8 \times 0.08 \times 0.001</td>
</tr>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>0.92 \times 0.08 \times 0.001</td>
</tr>
</tbody>
</table>

Performing exact inference: variable elimination

How about

\[ F_{\mathcal{A},L1,L2}(C, G) = \sum_A F_A(A, C, G)F_{L1}(A)F_{L2}(A) \]

To denote the fact that \( A \) has been summed out we place a bar over it in the notation.

\[ \sum_A F_A(A, C, G)F_{L1}(A)F_{L2}(A) = F_A(a, C, G)F_{L1}(a)F_{L2}(a) + F_A(\neg a, C, G)F_{L1}(\neg a)F_{L2}(\neg a) \]

where

<table>
<thead>
<tr>
<th>C</th>
<th>G</th>
<th>F_A(a, C, G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊤</td>
<td>⊤</td>
<td>0.98</td>
</tr>
<tr>
<td>⊥</td>
<td>⊤</td>
<td>0.96</td>
</tr>
<tr>
<td>⊤</td>
<td>⊥</td>
<td>0.2</td>
</tr>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>0.08</td>
</tr>
</tbody>
</table>

and similarly for \( F_A(\neg a, C, G), F_{L1}(\neg a) \) and \( F_{L2}(\neg a) \).
Performing exact inference: variable elimination

Now, say for example we have ¬c, g. Then doing the calculation explicitly would give

$$\sum_{A} \Pr (A|\neg c, g) \Pr (l|A) \Pr (l2|A) = \Pr (a|\neg c, g) \Pr (l|a) \Pr (l2|a) + \Pr (a|\neg c, g) \Pr (l|a) \Pr (l2|a)$$

$$= (0.2 \times 0.99 \times 0.6) + (0.8 \times 0.08 \times 0.001)$$

which matches!

Continuing in this manner form

$$F_{G,A,L1,L2}(C,G) = F_{G}(G)F_{A,L1,L2}(C,G),$$

sum out G to obtain

$$F_{G,A,L1,L2}(C) = \sum_{G} F_{G}(G)F_{A,L1,L2}(C,G),$$

form

$$F_{C,A,L1,L2} = F_{C}(C)F_{G,A,L1,L2}(C)$$

and normalise.

---

Performing exact inference: variable elimination

What’s the computational complexity now?

- For Bayesian networks with suitable structure we can perform inference in linear time and space.
- However in the worst case it is still \#P-hard.

Consequently, we may need to resort to approximate inference.

---

Approximate inference for Bayesian networks

Markov chain Monte Carlo (MCMC) methods also provide a method for performing approximate inference in Bayesian networks.

Say a system can be in a state S and moves from state to state in discrete time steps according to a probabilistic transition

$$\Pr (S \rightarrow S').$$

Let $$\pi_t(S)$$ be the probability distribution for the state after t steps, so

$$\pi_{t+1}(S') = \sum_{s} \Pr (s \rightarrow S') \pi_t(s).$$

If at some point we obtain $$\pi_{t+1}(s) = \pi_t(s)$$ for all s then we have reached a stationary distribution $$\pi$$. In this case

$$\forall s' \pi(s') = \sum_{s} \Pr (s \rightarrow s') \pi(s).$$

There is exactly one stationary distribution for a given $$\Pr (S \rightarrow S')$$ provided the latter obeys some simple conditions.

---

Approximate inference for Bayesian networks

The condition of detailed balance

$$\forall s, s' \pi(s) \Pr (s \rightarrow s') = \pi(s') \Pr (s' \rightarrow s)$$

is sufficient to provide a $$\pi$$ that is a stationary distribution. To see this simply sum:

$$\sum_{s} \pi(s) \Pr (s \rightarrow s') = \sum_{s} \pi(s') \Pr (s' \rightarrow s)$$

$$= \pi(s') \sum_{s} \Pr (s' \rightarrow s)$$

$$= \pi(s').$$

If all this is looking a little familiar, it’s because we now have another excellent application for the material in Mathematical Methods for Computer Science.

That course used the alternative term local balance.
Approximate inference for Bayesian networks

Recalling once again the basic equation for performing probabilistic inference

\[ \Pr(Q|o_1, o_2, \ldots, o_m) \propto \sum_L \Pr(Q, L, o_1, o_2, \ldots, o_m) \]

where

- \( Q \) is the query.
- \( o_1, o_2, \ldots, o_m \) are the observations.
- \( L \) are the latent variables.
- \( 1/Z \) normalises the distribution.

The query, observations and latent variables are a partition of the set \( V = \{V_1, V_2, \ldots, V_n\} \) of all variables.

We are going to consider obtaining samples from the distribution 
\[ \Pr(Q, L|o_1, o_2, \ldots, o_m). \]

Approximate inference for Bayesian networks

To see that \( \Pr(Q, L|o) \) is the stationary distribution we just demonstrate detailed balance:

\[ \pi(s) \Pr(s \rightarrow s') = \Pr(s|o) \Pr(s'|s, o) \]
\[ = \Pr(s, s'|o, o) \Pr(s'|s, o) \]
\[ = \Pr(s', s, o) \Pr(s|s', o) \Pr(s'|o) \]
\[ = \Pr(s|s, o) \Pr(s'|s', o) \]
\[ = \Pr(s' \rightarrow s) \pi(s'). \]

As a further simplification we can exploit conditional independence.

For example, sampling from \( \Pr(S_i|s, o) \) may be equivalent to sampling \( S_i \) conditional on some smaller set.

Approximate inference for Bayesian networks

The observations are fixed. Let the state of our system be a specific set of values for a query variable and the latent variables

\[ S = (S_1, S_2, \ldots, S_{n+1}) = (Q, L_1, L_2, \ldots, L_i) \]

and define \( \bar{S} \) to be the state vector with \( S_i \) removed

\[ \bar{S}_i = (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_{n+1}). \]

To move from \( s \) to \( s' \) we replace one of its elements, say \( s_i \), with a new value \( s'_i \) sampled according to

\[ s'_i \sim \Pr(S_i|\bar{S}, o_1, \ldots, o_m) \]

This has detailed balance, and has \( \Pr(Q, L|o_1, \ldots, o_m) \) as its stationary distribution.

It is known as Gibbs sampling.

Approximate inference for Bayesian networks

So:

- We successively sample the query variable and the unobserved variables, conditional on the remaining variables.
- This gives us a sequence \( s_1, s_2, \ldots \) sampled according to \( \Pr(Q, L|o) \).

Finally, note that as

\[ \Pr(Q|o) = \sum_1^n \Pr(Q, L|o) \]

we can just ignore the values obtained for the unobserved variables. This gives us \( q_1, q_2, \ldots \) with

\[ q_i \sim \Pr(Q|o). \]
Approximate inference for Bayesian networks

To see that the final step works, consider what happens when we estimate the expected value of some function of \( Q \).

\[
E[f(Q) \mid o] = \sum_q f(q) \Pr(q \mid l) = \sum_q \sum_l f(q) \Pr(q \mid l) \Pr(l \mid o)
\]

so sampling using \( \Pr(q, l \mid o) \) and ignoring the values for \( l \) obtained works exactly as required.

---

Markov random fields

Markov random fields (MRFs) (sometimes called undirected graphical models or Markov networks) provide an alternative approach to representing a probability distribution while expressing conditional independence assumptions.

We now have:

1. An undirected graph \( G = (N, E) \).
2. \( G \) has a node \( N_i \) for each RV.
3. For each maximal clique \( c \) in \( G \) there is a clique potential \( \phi_c(N_c) > 0 \) where \( N_c \) is the set of nodes in \( c \).
4. The probability distribution expressed by \( G \) is

\[
\Pr(N) \propto \prod_c \phi_c(N_c).
\]

---

Markov random fields—conditional independence

The test for conditional independence is now simple: if \( X, Y \) and \( Z \) are disjoint subsets of the RVs then:

1. Remove the nodes in \( Z \) and any attached edges from the graph.
2. If there are no paths from any variable in \( X \) to any variable in \( Y \) then \( X \perp Y \mid Z \).

Final things to note:

1. MRFs have their own algorithms for inference.
2. They are an alternative to BNs for representing a probability distribution.
3. There are trade-offs that might make a BN or MRF more or less favourable.
4. For example: potentials offer flexibility because they don’t have to represent conditional distributions…
5. … BUT you have to normalize the distribution you’re representing.