Expert Systems and Local Computation

Steffen Lauritzen, University of Oxford

Graduate Lectures Hilary Term 2011

February 2, 2011

An *expert system* attempts to crystallise and codify knowledge of experts into a tool, usable by non-specialist.

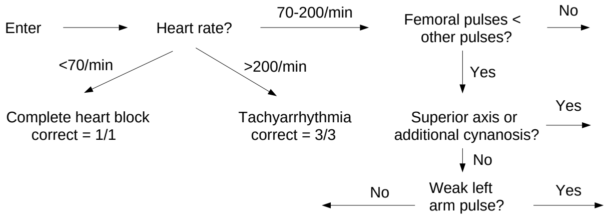
The *knowledge base* encodes the knowledge of the domain.

The *inference engine* consists of algorithms for processing knowledge base and specific information to obtain conclusions.

Classical expert systems *make model of expert.*

Probabilistic expert systems *model the domain* and use Bayesian reasoning.

Classification trees



Not necessarily computerized. Can be constructed using e.g. CART.

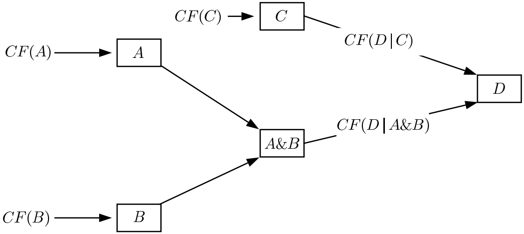
Production systems

Uses *rules:* IF (A_1 & A_2 & ... & A_k) THEN B; for example

- IF the animal has hair THEN it is a mammal.
- IF the animal gives milk THEN it is a mammal.
- IF the animal has feathers THEN it is a bird.
- IF the animal flies AND it lays eggs THEN it is a bird.

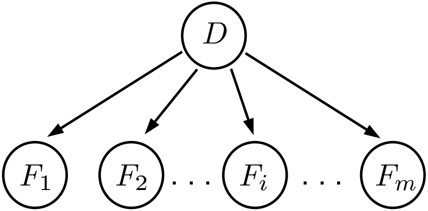
Inference "chaining" (forwards and backwards)

Certainty factors



Production rules with "certainty factor". Need calculus to combine certainty factors.

Naive Bayes

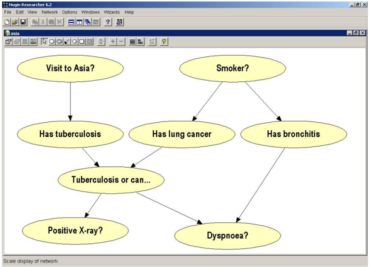


Disease probabilities D used. F_i are findings and $P(F_i \mid D)$ are specified.

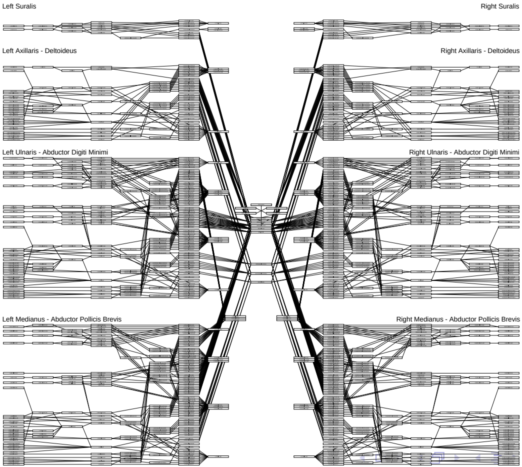
$P(D \mid F_1, \ldots, F_m)$ is calculated by Bayes' formula.

Directed graphical model, to be used for reasoning.

“Bayesian” because it reasons “reversely”, from symptoms to causes, in contrast to feedforward neural networks which were common when BNs were introduced.



MUNIN


Left Suralis
Right Suralis
Left Axillaris - Deltoideus
Right Axillaris - Deltoideus
Left Ulnaris - Abductor Digiti Minimi
Right Ulnaris - Abductor Digiti Minimi
Left Medianus - Abductor Pollicis Brevis
Right Medianus - Abductor Pollicis Brevis

Formal definition

A *Bayesian network* represents the *knowledge base* as a directed graphical model:

Formal definition

A *Bayesian network* represents the *knowledge base* as a directed graphical model:

- A Directed Acyclic Graph $\mathcal{D} = (V, E)$;

Formal definition

A *Bayesian network* represents the *knowledge base* as a directed graphical model:

- A Directed Acyclic Graph $\mathcal{D} = (V, E)$;
- Nodes V represent (random) variables $X_v, v \in V$;

Formal definition

A *Bayesian network* represents the *knowledge base* as a directed graphical model:

- A Directed Acyclic Graph $\mathcal{D} = (V, E)$;
- Nodes V represent (random) variables $X_v, v \in V$;
- Specify for all $v \in V$: $p(x_v \,|\, x_{\mathrm{pa}(v)})$;

Formal definition

A *Bayesian network* represents the *knowledge base* as a directed graphical model:

- A Directed Acyclic Graph $\mathcal{D} = (V, E)$;
- Nodes V represent (random) variables $X_v, v \in V$;
- Specify for all $v \in V$: $p(x_v \mid x_{\mathrm{pa}(v)})$;
- Joint distribution is then $p(x) = \prod_{v \in V} p(x_v \mid x_{\mathrm{pa}(v)})$.

Formal definition

A *Bayesian network* represents the *knowledge base* as a directed graphical model:

- A Directed Acyclic Graph $\mathcal{D} = (V, E)$;
- Nodes V represent (random) variables $X_v, v \in V$;
- Specify for all $v \in V$: $p(x_v \mid x_{\mathrm{pa}(v)})$;
- Joint distribution is then $p(x) = \prod_{v \in V} p(x_v \mid x_{\mathrm{pa}(v)})$.
- *Inference engine* uses *probability propagation* to calculate $p(x_v \mid x_E^*)$ for $E \subseteq V$ since $p(x_E^*) = \sum_{y: y_E = x_E^*} p(y)$ has *too many terms*.

The general problem

Factorizing density on $\mathcal{X} = \times_{v \in V} \mathcal{X}_v$ with V and $\mathcal{X}_v$ finite:

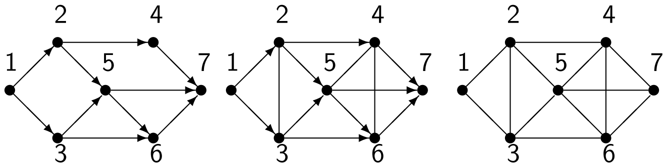
$$p(x) = \prod_{C \in \mathcal{C}} \phi_C(x).$$

The *potentials* $\phi_C(x)$ depend on $x_C = (x_v, v \in C)$ only.
Basic task to calculate *marginal* probability

$$p(x_E^*) = \sum_{y_{V \setminus E}} p(x_E^*, y_{V \setminus E})$$

for $E \subseteq V$ and fixed x_E^*, *but sum has too many terms.*
A second purpose is to get the *prediction* $p(x_v \mid x_E^*) = p(x_v, x_E^*)/p(x_E^*)$ for $v \in V$.

The *moral graph* $\mathcal{D}^m$ of a DAG $\mathcal{D}$ is obtained by adding undirected edges between unmarried parents and subsequently dropping directions, as in the example below:



Undirected factorizations

If P factorizes w.r.t. $\mathcal{D}$, it factorizes w.r.t. the moralised graph $\mathcal{D}^m$.

This is seen directly from the factorization:

$$f(x) = \prod_{v \in V} f(x_v \mid x_{\mathrm{pa}(v)}) = \prod_{v \in V} \psi_{\{v\} \cup \mathrm{pa}(v)}(x),$$

since $\{v\} \cup \mathrm{pa}(v)$ are all complete in $\mathcal{D}^m$.

Hence if *P satisfies any of the directed Markov properties w.r.t. $\mathcal{D}$, it satisfies all Markov properties for $\mathcal{D}^m$.*

Graph decomposition

Consider an undirected graph $\mathcal{G} = (V, E)$. A partitioning of V into a triple (A, B, S) of subsets of V forms a *decomposition* of $\mathcal{G}$ if

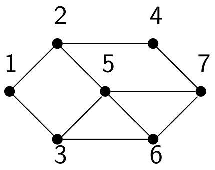
$$A \perp_{\mathcal{G}} B \,|\, S \text{ and } S \text{ is complete.}$$

The decomposition is *proper* if $A \neq \emptyset$ and $B \neq \emptyset$.

The *components* of $\mathcal{G}$ are the induced subgraphs $\mathcal{G}_{A \cup S}$ and $\mathcal{G}_{B \cup S}$.

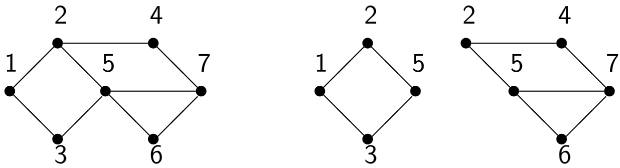
A graph is *prime* if no proper decomposition exists.

Examples



The graph to the left is prime

Decomposition with $A = \{1, 3\}$, $B = \{4, 6, 7\}$ and $S = \{2, 5\}$



Junction tree

Let $\mathcal{A}$ be a collection of finite subsets of a set V. A *junction tree* $\mathcal{T}$ of sets in $\mathcal{A}$ is an undirected tree with $\mathcal{A}$ as a vertex set, satisfying the *junction tree property:*

> *If $A, B \in \mathcal{A}$ and C is on the unique path in $\mathcal{T}$ between A and B it holds that $A \cap B \subset C$.*

If the sets in $\mathcal{A}$ are pairwise incomparable, *they can be arranged in a junction tree if and only if $\mathcal{A} = \mathcal{C}$ where $\mathcal{C}$ are the cliques of a chordal graph.*

Chordal graphs and junction trees

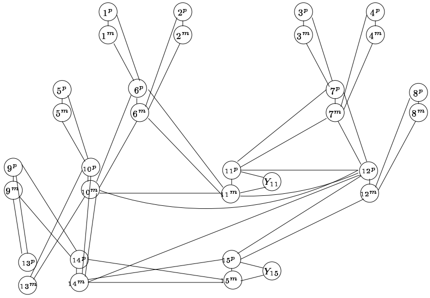
The following are equivalent for any undirected graph $\mathcal{G}$.

(i) *$\mathcal{G}$ is chordal* ie all cycles of length ≥ 4 have chords;

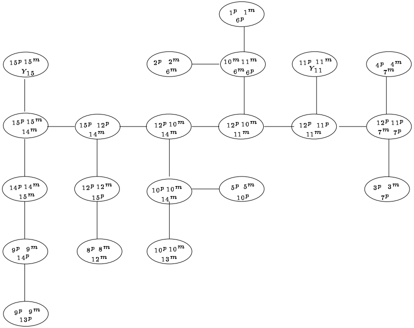
(ii) *All prime components of $\mathcal{G}$ are cliques;*

(iii) *Cliques of $\mathcal{G}$ can be arranged in a junction tree.*

A chordal graph



Junction tree



Cliques of graph arranged into a tree with $C_1 \cap C_2 \subseteq D$ for all cliques D on path between C_1 and C_2.

The computational structure is set up in several steps:

1. *Moralisation:* Constructing $\mathcal{D}^m$, exploiting that if P factorizes over $\mathcal{D}$, it factorizes over $\mathcal{D}^m$.
2. *Triangulation:* Adding edges to find chordal graph $\tilde{\mathcal{G}}$ with $\mathcal{G} \subseteq \tilde{\mathcal{G}}$. This step is non-trivial (NP-complete) to optimize;
3. *Constructing junction tree:* The cliques of $\tilde{\mathcal{G}}$ are found and arranged in a junction tree.
4. *Initialization:* Assigning potential functions ϕ_C to cliques.

The complete process above is known as *compilation*.

Computation is then performed by *message passing* after observations have been incorporated.

Initialization

1. For every vertex $v \in V$ we find a clique $C(v)$ in the triangulated graph $\tilde{\mathcal{G}}$ which contains $\mathrm{pa}(v)$. Such a clique exists because $v \cup \mathrm{pa}(v)$ are complete in $\mathcal{D}^m$ by construction, and hence in $\tilde{\mathcal{G}}$;

Initialization

1. For every vertex $v \in V$ we find a clique $C(v)$ in the triangulated graph $\tilde{\mathcal{G}}$ which contains $\mathrm{pa}(v)$. Such a clique exists because $v \cup \mathrm{pa}(v)$ are complete in $\mathcal{D}^m$ by construction, and hence in $\tilde{\mathcal{G}}$;
2. Define potential functions ϕ_C for all cliques C in $\tilde{\mathcal{G}}$ as

$$\phi_C(x) = \prod_{v: C(v)=C} p(x_v \mid x_{\mathrm{pa}(v)})$$

where the product over an empty index set is set to 1, i.e. $\phi_C \equiv 1$ if no vertex is assigned to C.

Initialization

1. For every vertex $v \in V$ we find a clique $C(v)$ in the triangulated graph $\tilde{\mathcal{G}}$ which contains $\mathrm{pa}(v)$. Such a clique exists because $v \cup \mathrm{pa}(v)$ are complete in $\mathcal{D}^m$ by construction, and hence in $\tilde{\mathcal{G}}$;
2. Define potential functions ϕ_C for all cliques C in $\tilde{\mathcal{G}}$ as

$$\phi_C(x) = \prod_{v:C(v)=C} p(x_v \mid x_{\mathrm{pa}(v)})$$

where the product over an empty index set is set to 1, i.e. $\phi_C \equiv 1$ if no vertex is assigned to C.
3. It now holds that

$$p(x) = \prod_{C \in \mathcal{C}} \phi_C(x).$$

Next we perform the following steps

1. *Incorporating observations:* If $X_E = x^*_E$ is observed, we modify potentials as

$$\phi_C(x_C) \leftarrow \phi_C(x) \prod_{e \in E \cap C} \delta(x^*_e, x_e),$$

with $\delta(u, v) = 1$ if $u = v$ and else $\delta(u, v) = 0$. Then:

$$p(x \mid X_E = x^*_E) = \frac{\prod_{C \in \mathcal{C}} \phi_C(x_C)}{p(x^*_E)}.$$

2. Marginals $p(x^*_E)$ and $p(x_C \mid x^*_E)$ are then calculated by a local *message passing* algorithm.

Separators

Between any two cliques C and D which are neighbours in the junction tree their intersection $S = C \cap D$ is called a *separator*.

We assign potentials to separators, initially $\phi_S \equiv 1$ for all $S \in \mathcal{S}$, where $\mathcal{S}$ is the set of separators.

Finally let

$$\kappa(x) = \frac{\prod_{C \in \mathcal{C}} \phi_C(x_C)}{\prod_{S \in \mathcal{S}} \phi_S(x_S)}, \tag{1}$$

and *now it holds that* $p(x \mid x_E^*) = \kappa(x)/p(x_E^*)$.

The expression (1) will be *invariant* under the message passing.

Marginalization

The *A-marginal* of a potential ϕ_B for $A \subseteq V$ is

$$\phi_B^{\downarrow A}(x) = \phi_B^{\downarrow A}(x_A) = \sum_{y_{A \cap B}: y_{A \cap B} = x_{A \cap B}} \phi_B(y)$$

Since ϕ_B depends on x through x_B only it is true that if $B \subseteq V$ is 'small', marginal can be computed easily.

Note that the marginal $\phi^{\downarrow A}$ depends on x_A only.

Marginalization satisfies

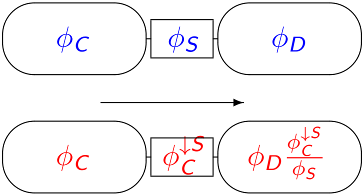
Consonance For subsets A and B: $\phi^{\downarrow(A\cap B)} = \left(\phi^{\downarrow B}\right)^{\downarrow A}$

Distributivity If ϕ_C depends on x_C only and $C \subseteq B$:
$(\phi\phi_C)^{\downarrow B} = \left(\phi^{\downarrow B}\right)\phi_C$.

Essentially the distributivity ensures that we can move factors in a sum outside of the summation sign.

Messages

When C *sends message* to D, the following happens:



Before *After*

Computation is *local*, involving only variables within cliques.

The expression

$$\kappa(x) = \frac{\prod_{C\in\mathcal{C}} \phi_C(x_C)}{\prod_{S\in\mathcal{S}} \phi_S(x_S)}$$

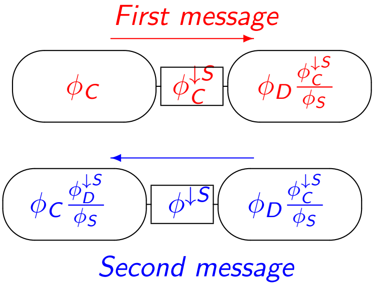
is *invariant under the message passing* since $\phi_C\phi_D/\phi_S$ is:

$$\frac{\phi_C\,\phi_D \frac{\phi_C^{\downarrow S}}{\phi_S}}{\phi_C^{\downarrow S}} = \frac{\phi_C\phi_D}{\phi_S}.$$

After the message has been sent, D *contains the D-marginal of $\phi_C\phi_D/\phi_S$.*

Second message

If D *returns message* to C, the following happens:



Now all sets contain the relevant marginal of $\phi = \phi_C\phi_D/\phi_S$:
The separator contains

$$\phi^{\downarrow S} = \left(\frac{\phi_C\phi_D}{\phi_S}\right)^{\downarrow S} = (\phi^{\downarrow D})^{\downarrow S} = \left(\phi_D\frac{\phi_C^{\downarrow S}}{\phi_S}\right)^{\downarrow S} = \frac{\phi_C^{\downarrow S}\phi_D^{\downarrow S}}{\phi_S}.$$

C contains

$$\phi_C\frac{\phi^{\downarrow S}}{\phi_C^{\downarrow S}} = \frac{\phi_C}{\phi_S}\phi_D^{\downarrow S} = \phi^{\downarrow C}$$

Further messages between C and D are neutral! Nothing will change if a message is repeated.

Two phases:

- COLLINFO: messages are sent from leaves towards arbitrarily chosen root R.

  After COLLINFO*, the root potential satisfies* $\phi_R(x_R) = \kappa^{\downarrow R}(x_R) = p(x_R, x^*_E)$.

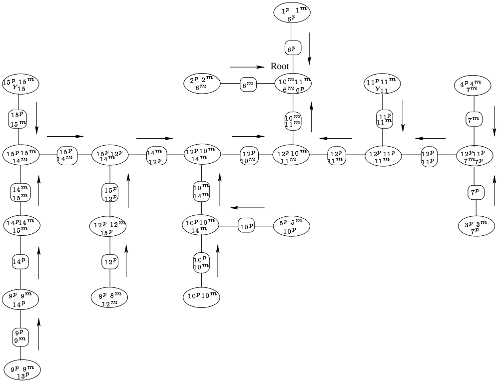
Two phases:

- COLLINFO: messages are sent from leaves towards arbitrarily chosen root R.
  After COLLINFO*, the root potential satisfies* $\phi_R(x_R) = \kappa^{\downarrow R}(x_R) = p(x_R, x_E^*)$.
- DISTINFO: messages are sent from root R towards leaves.
  After COLLINFO *and subsequent* DISTINFO*, it holds for all* $B \in \mathcal{C} \cup \mathcal{S}$ *that* $\phi_B(x_B) == \kappa^{\downarrow B}(x_B) = p(x_B, x_E^*)$.

Two phases:

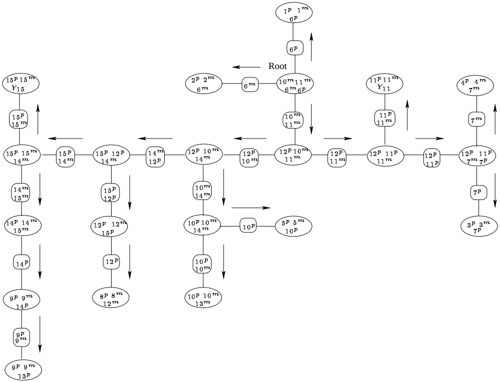
- COLLINFO: messages are sent from leaves towards arbitrarily chosen root R.
  After COLLINFO*, the root potential satisfies* $\phi_R(x_R) = \kappa^{\downarrow R}(x_R) = p(x_R, x_E^*)$.
- DISTINFO: messages are sent from root R towards leaves.
  After COLLINFO *and subsequent* DISTINFO*, it holds for all* $B \in \mathcal{C} \cup \mathcal{S}$ *that* $\phi_B(x_B) == \kappa^{\downarrow B}(x_B) = p(x_B, x_E^*)$.
- Hence $p(x_E^*) = \sum_{x_S} \phi_S(x_S)$ for any $S \in \mathcal{S}$ and $p(x_v \mid x_E^*)$ can readily be computed from any ϕ_S with $v \in S$.

CollInfo



Messages are sent from leaves towards root.

DistInfo



After CollInfo, messages are sent from root towards leaves.

Alternative scheduling of messages

Local control:

Allow clique to send message if and only if it has already received message from all other neighbours. Such messages are *live.*

Using this protocol, there will be one clique who first receives messages from all its neighbours. This is effectively the root R in CollInfo and DistInfo.

Additional messages never do any harm (ignoring efficiency issues) as κ is invariant under message passing.

Exactly two live messages along every branch is needed.

Replace sum-marginal with *A–maxmarginal:*

$$\phi_B^{\downarrow A}(x) = \max_{y_B : y_A = x_A} \phi_B(y)$$

Satisfies *consonance:* $\phi^{\downarrow(A\cap B)} = \left(\phi^{\downarrow B}\right)^{\downarrow A}$ and *distributivity:* $(\phi\phi_C)^{\downarrow B} = \left(\phi^{\downarrow B}\right)\phi_C$, if ϕ_C depends on x_C only and $C \subseteq B$.

COLLINFO *yields maximal value of density f.*

DISTINFO *yields configuration with maximum probability.*

Viterbi decoding for HMMs is special case.
Since (1) remains invariant, *one can switch freely between max- and sum-propagation.*

After CollInfo, the root potential is $\phi_R(x) \propto p(x_R \mid x_E)$
Modify DistInfo as follows:

1. Pick random configuration $\check{x}_R$ from ϕ_R.
2. Send message to neighbours C as $\check{x}_{R \cap C} = \check{x}_S$ where $S = C \cap R$ is the separator.
3. Continue by picking $\check{x}_C$ according to $\phi_C(x_{C \setminus S}, \check{x}_S)$ and send message further away from root.

When the sampling stops at leaves of junction tree, a configuration $\check{x}$ has been generated from $p(x \mid x^*_E)$.

The scaling operation on p:

$$(T_a p)(x) \leftarrow p(x)\frac{n^{\downarrow a}(x_a)}{np^{\downarrow a}(x_a)}, \quad x \in \mathcal{X}$$

is potentially very complex, as it cycles through all $x \in \mathcal{X}$, which is huge if V is large.

If we exploit a factorization of p w.r.t. a junction tree $\mathcal{T}$ for a decomposable $\mathcal{C} \supseteq \mathcal{A}$

$$p(x) = \frac{\prod_{C \in \mathcal{C}} \phi_C(x_C)}{\prod_{S \in \mathcal{S}} \phi_S(x_S)},$$

we can avoid scaling p and only scale the corresponding factor ϕ_{C^*} with $a \subseteq C^*$.

Scaling the factor ϕ_{C^*} involves

$$(T_a\phi_{C^*})(x_{C^*}) \leftarrow \phi_{C^*}(x_{C^*})\frac{n^{\downarrow a}(x_a)}{np^{\downarrow a}(x_a)}, \quad x_{C^*} \in \mathcal{X}_{C^*}$$

where $p^{\downarrow a}$ is calculated by probability propagation.

The scaling can now be made by changing the ϕ's:

$$\phi_B \leftarrow \phi_B \text{ for } B \neq C^*, \quad \phi_{C^*} \leftarrow T_a\phi_{C^*}.$$

This can reduce the complexity considerably.