Local computation in junction trees

Aarhus University, Fall 2003, Lecture 4

Steffen L. Lauritzen, Aalborg University

Bayesian network

Recall that a *Bayesian network* consists of

- Directed Acyclic Graph $\mathcal{D} = (V, E)$ with finite node set V
- V represent (random) variables $X_v, v \in V$
- Specify conditional distributions of (graph) children given (graph) parents: $p(x_v \mid x_{\mathrm{pa}(v)})$
- Joint distribution is then $p(x) = \prod_{v \in V} p(x_v \mid x_{\mathrm{pa}(v)})$

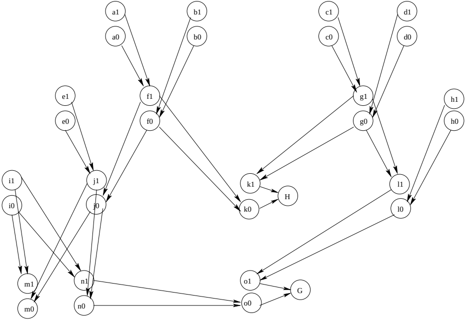
Need efficient algorithm to calculate $p(x_v \mid x_E^*)$ for $E \subseteq V$ since $p(x_E^*) = \sum_{y : y_E = x_E^*} p(y)$ has *too many terms*.

Computational structure

The computational structure is set up in four steps:

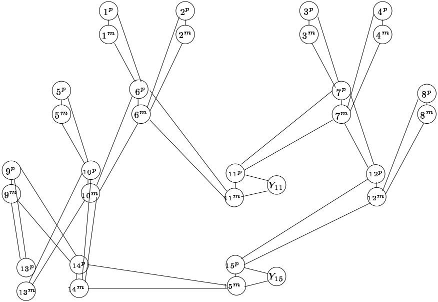
1. Forming *moral graph* (dependence graph): add links between parents and drop directions. Known as *moralization*
2. Adding edges to make graph *triangulated,* i.e. to ensure all cycles of length ≥ 4 have chords. *triangulation*
3. Arranging maximal cliques in *junction tree,* i.e. a tree with $C_1 \cap C_2 \subseteq D$ for all cliques D on path between C_1 and C_2.
4. Assigning suitable *potential functions* to cliques.

Bayesian allele network



An individual is represented with two alleles. Directed links indicate inheritance. Phenotypic information is available for two individuals.

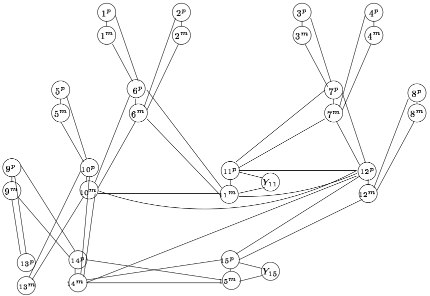
Moral graph



Dependence graph reflects factorisation of joint density into terms of form

$$\phi(x_{\{v\}\cup\mathrm{pa}(v)}) = p(x_v \mid x_{\mathrm{pa}(v)}).$$

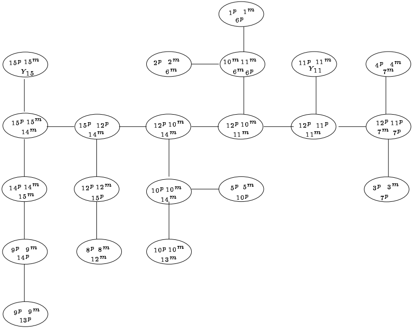
Triangulation



Links are added to make graph *triangulated*, i.e. so that all cycles of length ≥ 4 have chords.

Optimisation of this step is NP-complete but good practical algorithms exist.

Junction tree



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

Can be done if and only if graph is triangulated.

Assigning potentials to cliques

Procedure involves the following steps:

1. For each node $v \in V$, find clique $C(v)$ so that $\{v\} \cup \mathrm{pa}(v) \subseteq C(v)$.
2. For each clique C, define potential $\phi_C(x_C)$ as

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

It now holds that

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

Basic computation

This involves 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 we introduce their *separator* $S = C \cap D$.

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

We also 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 $A \subseteq B$ is

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

If ϕ_B depends on x through x_B only and $B \subseteq V$ is 'small', marginal can be computed easily.

Marginalization satisfies

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

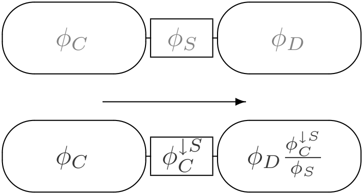
Distributivity If ϕ_C depends on x_C only and $C \subseteq B$:

$$(\phi \phi_C)^{\downarrow B} = \left(\phi^{\downarrow B}\right) \phi_C.$$

Messages

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

Before



After

Computation is *local*, involving only variables within cliques. *κ in (1) is invariant* since $\phi_C \phi_D / \phi_S$ is.

Message passing

Two phases:

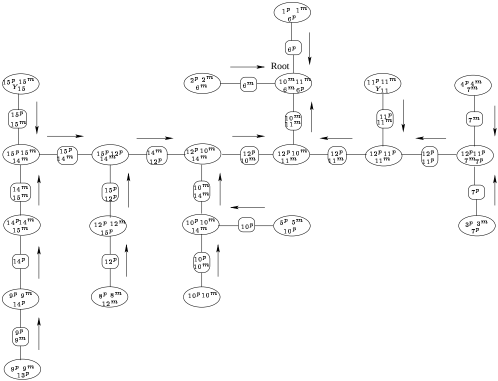
- COLLINFO: messages are sent from leaves towards arbitrarily chosen root R.

  After COLLINFO*, the root potential satisfies* $\phi_R(x_R) = p(x_R, x_E^*)$.

- 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* $\phi_B(x_B) = p(x_B, x_E^*)$.

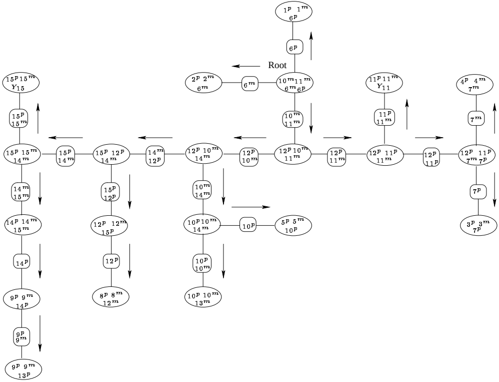
Hence $p(x_E^*) = \sum_{x_S} \phi_S(x_S)$ for any $S \in \mathcal{S}$ and $p(x_v \mid x_E^*)$ can easily be computed from any ϕ_S with $v \in S$.

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).

Exactly two live messages along every branch is needed.

Extensions

- Maximization (dynamic programming)
- Random sampling from conditionals
- Sparse linear equations
- Linear programming
- Constraint satisfaction
- Optimizing decisions
- Nash equilibria in cooperative games

Kalman filter and smoother, Viterbi decoding, Baum-Welch etc. are special cases.

Maximization

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.

Since (1) remains invariant, one can switch freely between max- and sum-propagation

Random propagation

After CollInfo, the root potential is $\phi_R(x) \propto p(x_R \,|\, x_E)$

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 $p(x \,|\, x_E^*)$.

Sparse linear equations

Substitute ϕ_C with equation systems over variables in C.

Substitute multiplication with combination of equations,

Substitute marginalisation to $A \subseteq C$ with solving for variables in C.

Obvious modification to *linear programming:*

ϕ_C are optimization problems subject to convex constraints.

Marginalisation derives constraints implied on subset of variables, etc.

In abstract sense, a very fundamental algorithm in mathematics!