Probability Propagation

Steffen Lauritzen, University of Oxford

Graphical Models, Lecture 12, Michaelmas Term 2010

November 19, 2010

Characterizing chordal graphs

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

(i) *$\mathcal{G}$ is chordal;*

(ii) *$\mathcal{G}$ is decomposable;*

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

(iv) *$\mathcal{G}$ admits a perfect numbering;*

(v) *Every minimal (α, β)-separator are complete;*

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

Algorithms associated with chordality

Maximum Cardinality Search (MCS) *identifies whether a graph is chordal or not.*

If a graph $\mathcal{G}$ is chordal, MCS *yields a perfect numbering* of the vertices. In addition it *finds the cliques* of $\mathcal{G}$:

From an MCS numbering $V = \{1, \dots, |V|\}$, let

$$B_\lambda = \mathrm{bd}(\lambda) \cap \{1, \dots, \lambda - 1\}$$

and $\pi_\lambda = |B_\lambda|$. A *ladder vertex* is either $\lambda = |V|$ or one with $\pi_{\lambda+1} < \pi_\lambda + 1$. Let Λ be the set of ladder vertices.

The cliques are $C_\lambda = \{\lambda\} \cup B_\lambda, \lambda \in \Lambda$.

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.*

The junction tree can be *constructed directly from the MCS ordering $C_\lambda, \lambda \in \Lambda$*.

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 $v \in V$.

Computational structure

Algorithms all arrange the collection of sets $\mathcal{C}$ in a junction tree $\mathcal{T}$. Hence, they works *only if $\mathcal{C}$ are cliques of chordal graph $\mathcal{G}$.*

If the initial model is based on a DAG $\mathcal{D}$, the first step is to form the *moral graph* $\mathcal{G} = \mathcal{D}^m$, exploiting that if P factorizes w.r.t. $\mathcal{D}$, it also factorizes w.r.t. $\mathcal{D}^m$.

If $\mathcal{G}$ is not chordal from the outset, *triangulation* is used to construct chordal graph $\mathcal{G}'$ with $E \subseteq E'$. Again, *if P factorizes w.r.t. $\mathcal{G}$ it factorizes w.r.t. $\mathcal{G}'$*.This step is non-trivial and it is NP-complete to optimize.

When this has been done, the computations are executed by *message passing*.

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 $\mathcal{D}^m$.

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;

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:* Using MCS, the cliques of $\tilde{\mathcal{G}}$ are found and arranged in a junction tree.

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:* Using MCS, the cliques of $\tilde{\mathcal{G}}$ are found and arranged in a junction tree.
4. *Initialization:* Assigning potential functions ϕ_C to cliques.

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:* Using MCS, 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*.

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

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

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

Overview

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 their intersection $S = C \cap D$ is called a *separator*. In fact, *the sets S are the minimal separators appearing in any decomposition sequence.*

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.

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

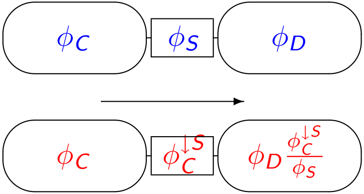
Consonance For subsets A and B: $\phi^{\downarrow(A\cap B)} = \left(\phi^{\downarrow B}\right)^{\downarrow A}$

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:

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

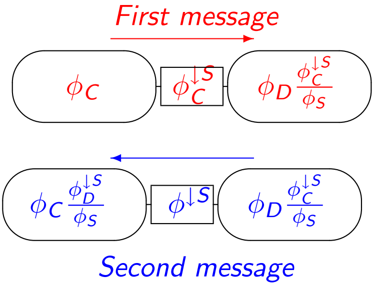
After the message has been sent, D *contains the D-marginal of* $\phi_C \phi_D / \phi_S$.

To see this, calculate

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

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

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

since, as before

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

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* $\phi_R(x_R) = \kappa^{\downarrow R}(x_R) = p(x_R, x_E^*)$.

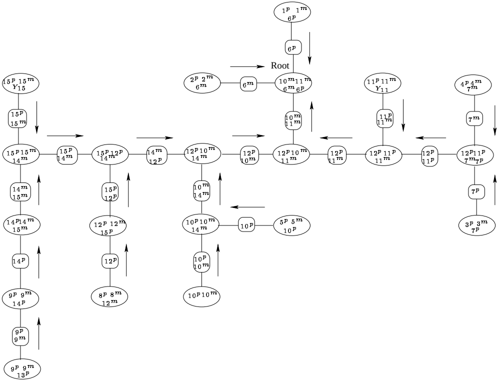
Two phases:

- 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:

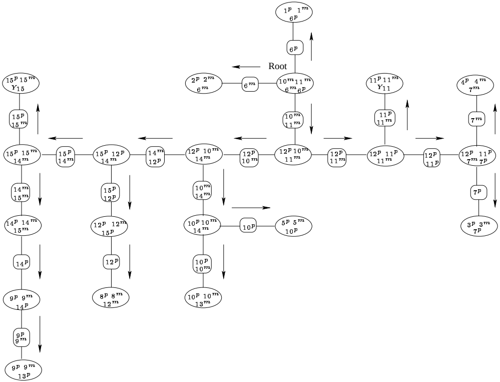
- COLLINFO: messages are sent from leaves towards arbitrarily chosen root R.
  After COLLINFO, *the root potential satisfies* $\phi_R(x_R) = \kappa^{\downarrow R}(x_R) = p(x_R, x_E^*)$.
- DISTINFO: messages are sent from root R towards leaves.
  After COLLINFO *and subsequent* DISTINFO, *it holds for 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 $p(x_v \,|\, x_E^*)$ 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.