Probability Propagation and Related Algorithms

Lecture 4
Saint Flour Summerschool, July 8, 2006

Steffen L. Lauritzen, University of Oxford

Overview of lectures

1. Conditional independence and Markov properties
2. More on Markov properties
3. Graph decompositions and junction trees
4. Probability propagation and related algorithms
5. Log-linear and Gaussian graphical models
6. Conjugate prior families for graphical models
7. Hyper Markov laws
8. Structure learning and Bayes factors
9. More on structure learning.

Markov properties for undirected graphs

(P) *pairwise Markov:* $\alpha \not\sim \beta \implies \alpha \perp\!\!\!\perp \beta \,|\, V \setminus \{\alpha, \beta\}$;

(L) *local Markov:* $\alpha \perp\!\!\!\perp V \setminus \mathrm{cl}(\alpha) \,|\, \mathrm{bd}(\alpha)$;

(G) *global Markov:* $A \perp_{\mathcal{G}} B \,|\, S \implies A \perp\!\!\!\perp B \,|\, S$;

(F) *Factorization:* $f(x) = \prod_{a \in \mathcal{A}} \psi_a(x)$, $\mathcal{A}$ being complete subsets of V.

It then holds that

$$\text{(F)} \implies \text{(G)} \implies \text{(L)} \implies \text{(P)}.$$

If $f(x) > 0$ even

$$\text{(F)} \iff \text{(G)} \iff \text{(L)} \iff \text{(P)}.$$

Markov properties for directed acyclic graphs

(O) *ordered Markov:* $\alpha \perp\!\!\!\perp \{\mathrm{pr}(\alpha) \setminus \mathrm{pa}(\alpha)\} \mid \mathrm{pa}(\alpha)$;

(L) *local Markov:* $\alpha \perp\!\!\!\perp \{\mathrm{nd}(\alpha) \setminus \mathrm{pa}(\alpha)\} \mid \mathrm{pa}(\alpha)$;

(G) *global Markov:* $A \perp_{\mathcal{D}} B \mid S \implies A \perp\!\!\!\perp B \mid S$.

(F) *Factorization:* $f(x) = \prod_{v \in V} f(x_v \mid x_{\mathrm{pa}(v)})$.

It then *always* holds that

$$\text{(F)} \iff \text{(G)} \iff \text{(L)} \iff \text{(O)}.$$

Relation between different graphs

P directed Markov w.r.t. $\mathcal{D}$ implies P factorizes w.r.t. $\mathcal{D}^m$.

$\mathcal{D}$ is *perfect* if skeleton $\mathcal{G} = \sigma(\mathcal{D}) = \mathcal{D}^m$, implying that *directed and undirected separation properties are identical,* i.e. $A \perp_{\mathcal{G}} B \mid S \iff A \perp_{\mathcal{D}} B \mid S$.

$\mathcal{G} = \sigma(\mathcal{D})$ for some DAG $\mathcal{D}$ if and only if $\mathcal{G}$ is chordal.

Two DAGs $\mathcal{D}$ and $\mathcal{D}'$ are *Markov equivalent,* i.e. $A \perp_{\mathcal{D}} B \mid S \iff A \perp_{\mathcal{D}'} B \mid S$, *if and only if $\sigma(\mathcal{D}) = \sigma(\mathcal{D}')$ and $\mathcal{D}$ and $\mathcal{D}'$ have same unmarried parents.*

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 both of the following holds:

(i) $A \perp_{\mathcal{G}} B \mid S$;

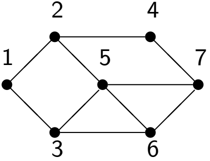
(ii) S is complete.

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 $\mathcal{G}_{B \cup S}$.

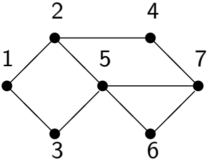
A graph is *prime* if no proper decomposition exists.

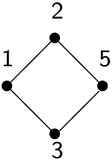
Examples

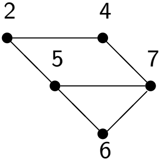


The graph to the left is prime

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

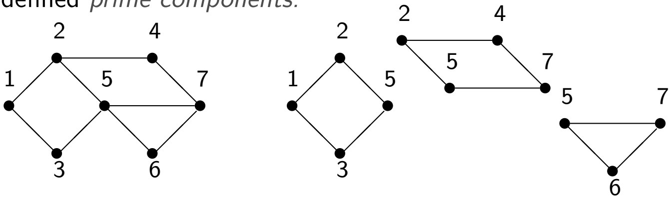






Decomposability

Any graph can be recursively decomposed into its uniquely defined *prime components:*



A graph is *decomposable* (or rather fully decomposable) if it is complete or admits a proper decomposition into *decomposable* subgraphs.

Definition is recursive. Alternatively this means that *all prime components are cliques.*

Decomposition of Markov properties

Let (A, B, S) be a decomposition of $\mathcal{G}$. Then *P factorizes w.r.t. $\mathcal{G}$ if and only if both of the following hold:*

(i) $P_{A\cup S}$ and $P_{B\cup S}$ factorize w.r.t. $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$;

(ii) $f(x)f_S(x_S) = f_{A\cup S}(x_{A\cup S})f_{B\cup S}(x_{B\cup S})$.

Recursive decomposition of a *decomposable graph* yields:

$$f(x)\prod_{S\in\mathcal{S}} f_S(x_S)^{\nu(S)} = \prod_{C\in\mathcal{C}} f_C(x_C).$$

Here $\mathcal{S}$ is the set of *complete separators* occurring in the decomposition process and $\nu(S)$ the number of times a given S appears.

More generally if $\mathcal{Q}$ denotes the prime components of $\mathcal{G}$:

$$f(x) \prod_{S \in \mathcal{S}} f_S(x_S)^{\nu(S)} = \prod_{Q \in \mathcal{Q}} f_Q(x_Q).$$

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.*

Trees are chordal graphs and thus decomposable.

Algorithms associated with chordality

Maximum Cardinality Search (MCS) Tarjan and Yannakakis (1984) *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 $V = \{1, \ldots, |V|\}$, let

$$S_\lambda = \mathrm{bd}(\lambda) \cap \{1, \ldots, \lambda - 1\}$$

and $\pi_\lambda = |S_\lambda|$. Call λ a *ladder vertex* if $\lambda = |V|$ or if $\pi_{\lambda+1} < \pi_\lambda + 1$ and let Λ be the set of ladder vertices.

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

The numbers $\nu(S)$ in the decomposition formula are $\nu(S) = |\{\lambda \in \Lambda : S_\lambda = S\}|$.

Junction tree

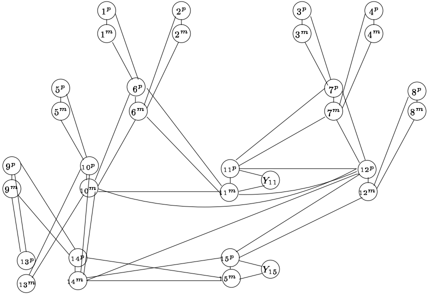
Let $\mathcal{A}$ be a collection of finite subsets of a set V. A *junction tree* $\mathcal{T}$ of sets in $\mathcal{A}$ is an undirected tree with $\mathcal{A}$ as a vertex set, satisfying the *junction tree property:*

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.*

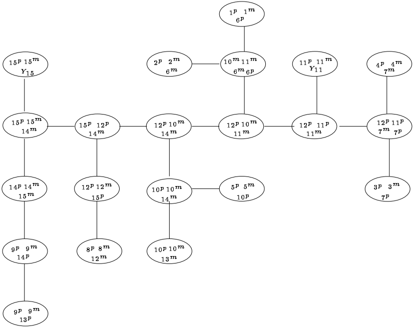
The junction tree can be *constructed directly from the MCS ordering* $C_\lambda, \lambda \in \Lambda$.

A chordal graph



This graph is chordal, but it might not be that easy to see. . . Maximum Cardinality Search is handy!

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.

Junction trees of prime components

In general, the *prime components* of any undirected graph can be arranged in a junction tree in a similar way, using an algorithm of Tarjan (1985), see also Leimer (1993).

Then *every pair of neighbours* (C, D) *in the junction tree represents a decomposition of* $\mathcal{G}$ into $\mathcal{G}_{\tilde{C}}$ and $\mathcal{G}_{\tilde{D}}$, where $\tilde{C}$ is the set of vertices in cliques connected to C but separated from D in the junction tree, and similarly with $\tilde{D}$.

Tarjan's algorithm is based on a slightly more sophisticated algorithm (Rose $et\ al$. 1976) known as *Lexicographic Search* (LEX) which runs in $O(|V|^2)$ time.

Markov properties of junction tree

Let $Q \in \mathcal{Q}$ be the prime components of a graph $\mathcal{G}$, arranged in a junction tree $\mathcal{T}$.

Using that any graph decomposition also yields a decomposition of the Markov properties now gives that

The distribution of $X = (X_v, v \in V)$ *factorizes w.r.t.* $\mathcal{G}$ *if and only if* $X_Q, Q \in \mathcal{Q}$ *factorizes w.r.t.* $\mathcal{T}$ *and each of* X_Q *factorizes w.r.t.* $\mathcal{G}_Q$*.*

In particular, *if* $\mathcal{G}$ *is decomposable,* $X = (X_v, v \in V)$ *factorizes w.r.t.* $\mathcal{G}$ *if and only if* $X_C, C \in \mathcal{C}$ *factorizes w.r.t.* $\mathcal{T}$*,* i.e. the Markov property has essentially been transferred to that of a tree of cliques.

Local computation

Local computation algorithms similar to probability propagation have been developed independently in a number of areas with a variety of purposes. For example:

- Kalman filter and smoother (Thiele 1880; Kalman and Bucy 1961);
- Solving sparse linear equations (Parter 1961);
- Decoding digital signals (Viterbi 1967; Bahl *et al*. 1974);
- Estimation in hidden Markov models (Baum 1972);
- Peeling in pedigrees (Elston and Stewart 1971; Cannings *et al*. 1976);

- Belief function evaluation (Kong 1986; Shenoy and Shafer 1986);
- Probability propagation (Pearl 1986; Lauritzen and Spiegelhalter 1988; Jensen *et al.* 1990);
- Abstract framework (Shenoy and Shafer 1990; Lauritzen and Jensen 1997).

Also dynamic programming, linear programming, optimizing decisions, calculating Nash equilibria in cooperative games, and many others. *List is far from exhaustive!*

All algorithms are using, explicitly or implicitly, a *graph decomposition* and *a junction tree* or similar to make the computations.

An abstract perspective

V is *large* finite set and $\mathcal{C}$ collection of *small* subsets of V.

$\phi_C, C \in \mathcal{C}$ are *valuations* with *domain* C.

Combination: $\phi_A \otimes \phi_B$ has domain $A \cup B$.

$\otimes$ is assumed *commutative* and *associative*.

For $A \subset V$ $\phi^{\downarrow A}$ denotes the A*-marginal* of ϕ. $\phi^{\downarrow A}$ has domain A.

Assume *consonance:* $\phi^{\downarrow(A\cap B)} = \left(\phi^{\downarrow B}\right)^{\downarrow A}$

and *distributivity:* $(\phi \otimes \phi_C)^{\downarrow B} = \left(\phi^{\downarrow B}\right) \otimes \phi_C$, if $C \subseteq B$.

Computational challenge

Calculate marginals $\psi_A = \phi^{\downarrow A}$ of *joint valuation*

$$\phi = \otimes_{C \in \mathcal{C}} \phi_C$$

with domain $V = \cup_{C \in \mathcal{C}} C$.

Direct computation of $\phi^{\downarrow A}$ is impossible if V is large.

Challenge: calculate $\phi^{\downarrow A}$ using only *local* operations, i.e. operating on factors ψ_B with domain $B \subseteq C$ for some $C \in \mathcal{C}$.

Typically also a *second purpose* of calculation.

A probability perspective

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* (likelihood)

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

for $E \subseteq V$ and fixed x_E^*, *but sum has too many terms.*

A second purpose is to get the *prediction*
$p(x_v \,|\, x_E^*) = p(x_v, x_E^*)/p(x_E^*)$ for $v \in V$.

Sparse linear equations

- Valuations ϕ_C are *equation systems* involving variables with labels C;
- $\phi_A \otimes \phi_B$ *concatenates* equation systems;
- $\phi_B^{\downarrow A}$ *eliminates* variables in $B \setminus A$;
- Marginal $\phi^{\downarrow A}$ of joint valuation *reduces* the system of equation to a smaller one;
- Second computation finds a *solution* of the equation system by substitution.

Constraint satisfaction

- ϕ_C represent *constraints* involving variables in C;
- $\phi_A \otimes \phi_B$ represents *jointly feasible* configurations;
- $\phi_B^{\downarrow A}$ finds *implied constraints*;
- Marginal $\phi^{\downarrow A}$ finds *extendible* configurations;
- Second computation *identifies* jointly feasible configurations.

If represented by indicator functions, $\otimes$ is ordinary product and $\phi^{\downarrow E}(x_E^*) = \oplus_{y_{V \setminus E}} \phi(x_E^*, y_{V \setminus E})$, where $1 \oplus 1 = 1 \oplus 0 = 0 \oplus 1 = 1$ and $0 \oplus 0 = 0$.

Computational structure

Algorithms all (implicitly or explicitly) arrange the collection of sets $\mathcal{C}$ in a *junction tree* $\mathcal{T}$.

Hence, this works *only if $\mathcal{C}$ are cliques of chordal graph $\mathcal{G}$.*

If this is not so from the outset, a *triangulation* is used to construct chordal graph $\mathcal{G}'$ with $E \subseteq E'$.

Clearly, in a probabilistic perspective, *if P factorizes w.r.t. $\mathcal{G}$ it factorizes w.r.t. $\mathcal{G}'$*.

Henceforth *we assume this has been done* and $\mathcal{G}$ is chordal.

Computations are executed by *message passing*.

Setting up the structure

In many applications P is initially factorizing over a *directed acyclic graph* $\mathcal{D}$. The computational structure is then set up in several steps:

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

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:

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

2. Marginals $p(x_E^*)$ and $p(x_C \,|\, x_E^*)$ 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 intersection $S = C \cap D$. In fact, S are the *minimal separators* appearing in the 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.

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.

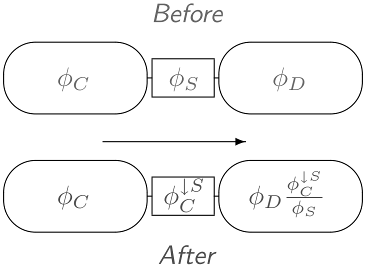
Marginalization satisfies

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

Messages

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



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:

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

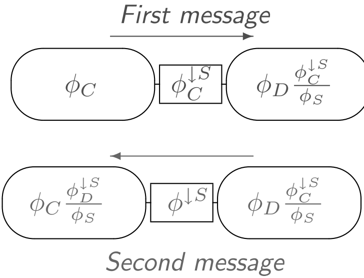
After the message has been sent, D *contains the* D*-marginal of* $\phi_C \phi_D / \phi_S$.

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.

Message passing

Two phases:

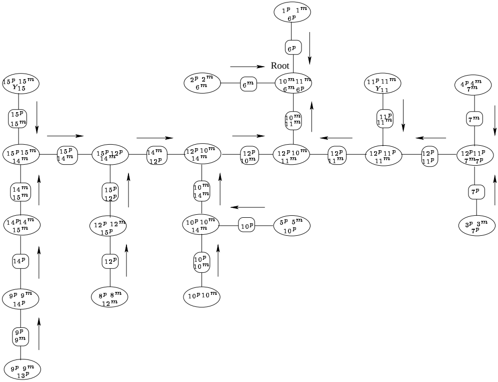
- $\textsc{CollInfo}$: messages are sent from leaves towards arbitrarily chosen root R.

  After $\textsc{CollInfo}$*, the root potential satisfies* $\phi_R(x_R) = p(x_R, x_E^*)$.

- $\textsc{DistInfo}$: messages are sent from root R towards leaves. *After* $\textsc{CollInfo}$ *and subsequent* $\textsc{DistInfo}$*, it holds for all* $B \in \mathcal{C} \cup \mathcal{S}$ *that* $\phi_B(x_B) = p(x_B, x_E^*)$.

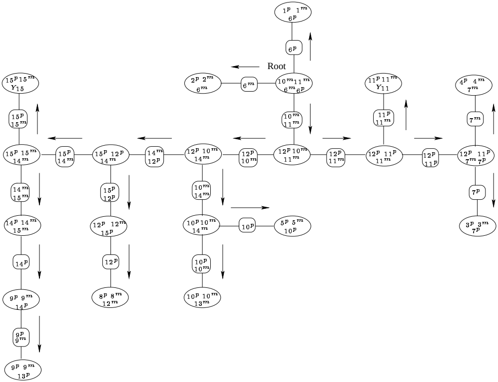
Hence $p(x_E^*) = \sum_{x_S} \phi_S(x_S)$ for any $S \in \mathcal{S}$ and $p(x_v \,|\, x_E^*)$ can readily be computed from any ϕ_S with $v \in S$.

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 $\text{C}\small{\text{OLL}}\text{I}\small{\text{NFO}}$ and $\text{D}\small{\text{IST}}\text{I}\small{\text{NFO}}$.

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.

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

Random propagation

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

Modify $\textsc{DistInfo}$ *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* $p(x \,|\, x_E^*)$*.*

References

Bahl, L., Cocke, J., Jelinek, F., and Raviv, J. (1974). Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, **20**, 284–7.

Baum, L. E. (1972). An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, **3**, 1–8.

Cannings, C., Thompson, E. A., and Skolnick, M. H. (1976). Recursive derivation of likelihoods on pedigrees of arbitrary complexity. *Advances in Applied Probability*, **8**, 622–5.

Elston, R. C. and Stewart, J. (1971). A general model for the genetic analysis of pedigree data. *Human Heredity*, **21**, 523–42.

Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, **4**, 269–82.

Kalman, R. E. and Bucy, R. (1961). New results in linear filtering and prediction. *Journal of Basic Engineering*, **83 D**, 95–108.

Kong, A. (1986). *Multivariate belief functions and graphical models*. Ph.D. Thesis, Department of Statistics, Harvard University, Massachusetts.

Lauritzen, S. L. and Jensen, F. V. (1997). Local computation with valuations from a commutative semigroup.

Annals of Mathematics and Artificial Intelligence, **21**, 51–69.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, Series B*, **50**, 157–224.

Leimer, H.-G. (1993). Optimal decomposition by clique separators. *Discrete Mathematics*, **113**, 99–123.

Parter, S. (1961). The use of linear graphs in Gauss elimination. *SIAM Review*, **3**, 119–30.

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, **29**, 241–88.

Rose, D. J., Tarjan, R. E., and Lueker, G. S. (1976). Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, **5**, 266–83.

Shenoy, P. P. and Shafer, G. (1986). Propagating belief functions using local propagation. *IEEE Expert*, **1**, 43–52.

Shenoy, P. P. and Shafer, G. (1990). Axioms for probability and belief–function propagation. In *Uncertainty in artificial intelligence 4*, (ed. R. D. Shachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer), pp. 169–98. North-Holland, Amsterdam, The Netherlands.

Tarjan, R. E. (1985). Decomposition by clique separators. *Discrete Mathematics*, **55**, 221–32.

Tarjan, R. E. and Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, **13**, 566–79.

Thiele, T. N. (1880). Om Anvendelse af mindste Kvadraters Methode i nogle Tilfælde, hvor en Komplikation af visse Slags uensartede tilfældige Fejlkilder giver Fejlene en 'systematisk' Karakter. *Vidensk. Selsk. Skr. 5. Rk., naturvid. og mat. Afd.*, **12**, 381–408. French version: *Sur la Compensation de quelques Erreurs quasi-systématiques par la Méthode des moindres Carrés.* Reitzel, København, 1880.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, **13**, 260–9.