Local Computation

Steffen Lauritzen, University of Oxford

Graphical Models, Lecture 13, Michaelmas Term 2010

November 22, 2010

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 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*.

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

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

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

After incorporation of observations *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

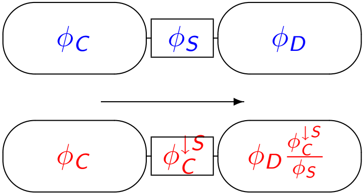
Consonance For subsets A and B: $\phi^{\downarrow(A\cap B)} = \left(\phi^{\downarrow B}\right)^{\downarrow A}$

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:

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

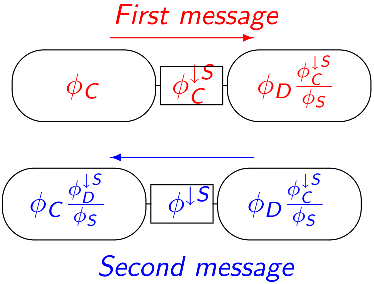
After the message has been sent, D *contains the D-marginal of* $\phi_C \phi_D / \phi_S$.

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 $p(x_v \mid x_E^*)$ can readily be computed from any ϕ_S with $v \in S$.

The correctness of the algorithm is easily established by induction:

We have on the previous overheads shown correctness for a junction tree with only two cliques.

Now consider a leaf clique L of the juction tree and let $V* = \cup_{C:C \in \mathcal{C} \setminus \{L\}} C$.

We can then think of L and V^* forming a junction tree of two cliques with separator $S^* = L \cap C^*$ where C^* is the neighbour of L in the junction tree.

After a message has been sent from L to V^* in the CollInfo phase, ϕ_{V^*} is equal to the V^*-marginal of κ.

By induction, when all messages have been sent except the one from the neighbour clique C^* to L, all cliques other than L contain the relevant marginal of κ, and

$$\phi_{V^*} = \frac{\prod_{C:C\in\mathcal{C}\setminus\{L\}} \phi_C}{\prod_{S:S\in\mathcal{S}\setminus\{S^*\}} \phi_S}.$$

Now let, V^* send its message back to L. To do this, it needs to calculate $\phi_{V^*}^{\downarrow S^*}$. But since $S^* \subseteq C^*$, and $\phi_{C^*} = \phi_{V^*}^{\downarrow C^*}$ we have

$$\phi_{V^*}^{\downarrow S^*} = \phi_{C^*}^{\downarrow S^*}$$

and sending a message from V^* to L is thus equivalent to sending a message from C^* to L. Thus, after this message has been sent, $\phi_L = \kappa^{\downarrow L}$ as desired.

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.

Local computation algorithms have been developed with a variety of purposes. For example:

- Kalman filter and smoother
- Solving sparse linear equations;
- Decoding digital signals;
- Estimation in hidden Markov models;
- Peeling in pedigrees;
- Belief function evaluation;
- Probability propagation.

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.

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 $a \subseteq C^*$:

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

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.

Note that if $a = C$ and $\phi_a = n^{\downarrow a}(x_a)$, then $T_a \phi_a = \phi_a$. Hence the explicit formula for the MLE.