Some figures and materials courtesy David Barber, Bayesian Reasoning and Machine Learning http://www.cs.ucl.ac.uk/staff/d.barber/brml/
The Marginal Inference Problem

General Problem Formulation

- We are given some factor graph with \( N \) unobserved discrete variables:

\[
p(x) = \frac{1}{Z} \prod_{f \in F} \phi_f(x_f) \quad Z = \sum_{x} \prod_{f \in F} \phi_f(x_f)
\]

- We would like to compute the marginal distributions of all \( N \) variables:

\[
p(x_i) \propto \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_N} \left[ \prod_{f \in F} \phi_f(x_f) \right] \propto \sum_{x \setminus x_i} \left[ \prod_{f \in F} \phi_f(x_f) \right]
\]

Issues Impacting this Computational Problem

- **Irrelevant**: Was original graph directed or undirected?
- **Irrelevant**: Are we computing marginals of prior distribution, or of posterior given observed data?
- **Simplification**: Factor graph is a tree (no cycles)
- **Simplification**: Factor graph has only pairwise factors
Sum-Product Belief Propagation for Tree-Structured Factor Graphs
What is the marginal distribution of $x_1$?

\[
p_1(x_1) \propto \sum_{x_2} \sum_{x_3} \sum_{x_4} \phi_{12}(x_1, x_2) \phi_{23}(x_2, x_3) \phi_{24}(x_2, x_4)
\]

Distributive Law

Messages from Leaves

Message Update

\[
m_{21}(x_1) = \sum_{x_2} \phi_{12}(x_1, x_2) m_{32}(x_2) m_{42}(x_2)
\]
A Tree-Structured Factor Graph

What is the marginal distribution of $x_1$?

$p_1(x_1) \propto \sum_{x_2} \sum_{x_3} \sum_{x_4} \phi_f(x_1, x_2) \phi_g(x_2, x_3, x_4)$

$p_1(x_1) \propto \sum_{x_2} \phi_f(x_1, x_2) \left[ \sum_{x_3, x_4} \phi_g(x_2, x_3, x_4) \right]$}

$m_{g2}(x_2)$

$m_{21}(x_1) = \sum_{x_2} \phi_{12}(x_1, x_2) m_{g2}(x_2)$

$p(x) \propto \phi_f(x_1, x_2) \phi_g(x_2, x_3, x_4)$

Distributive Law

Message from Factor

Message Update
Sum-Product Belief Propagation

Set of neighbors of node $s$: $\Gamma(s) = \{ f \in \mathcal{F} \mid s \in f \}$

$K_s$ $\longrightarrow$ number of discrete states for random variable $x_s$

$p_s(x_s)$ $\longrightarrow$ marginal distribution of the $K_s$ discrete states of random variable $x_s$

$\bar{m}_{sf}(x_s)$ $\longrightarrow$ message from variable $s$ to factor $f$, vector of $K_s$ non-negative numbers

$m_{fs}(x_s)$ $\longrightarrow$ message from factor $f$ to variable $s$, vector of $K_s$ non-negative numbers
**Sum-Product Belief Propagation**

Set of *neighbors* of node $s$: \[ \Gamma(s) = \{ f \in \mathcal{F} \mid s \in f \} \]

Special Case: Factor of degree one

\[ m_{fs}(x_s) = \phi_f(x_f), \quad f = \{s\} \]

Special Case: Variable of degree one

\[ \bar{m}_{sf}(x_s) = 1, \quad \Gamma(s) = \{f\} \]

\[ p_s(x_s) \propto m_{fs}(x_s) \]

\[ m_{fs}(x_s) = \prod_{f \in \Gamma(s)} m_{fs}(x_s) \]

\[ p_s(x_s) \propto \prod_{f \in \Gamma(s)} m_{fs}(x_s) \]

\[ \bar{m}_{sf}(x_s) = \prod_{g \in \Gamma(s) \setminus f} m_{gs}(x_s) \propto \frac{p_s(x_s)}{m_{fs}(x_s)} \]

\[ m_{fs}(x_s) = \sum_{x_f \setminus s} \phi_f(x_f) \prod_{t \in f \setminus s} \bar{m}_{tf}(x_t) \]
Sum-Product Belief Propagation

Set of neighbors of node $s$: $\Gamma(s) = \{ f \in \mathcal{F} \mid s \in f \}$

$p_s(x_s) \propto \prod_{f \in \Gamma(s)} m_{fs}(x_s)$

$\bar{m}_{sf}(x_s) = \prod_{g \in \Gamma(s) \setminus f} m_{gs}(x_s) \propto \frac{p_s(x_s)}{m_{fs}(x_s)}$

$m_{fs}(x_s) = \sum_{x_f \setminus s} \phi_f(x_f) \prod_{t \in f \setminus s} \bar{m}_{tf}(x_t)$

Special Case: Factor of degree two

Recover pairwise MRF sum-product BP updates as a special case.
CHAPTER 2. NONPARAMETRIC AND GRAPHICAL MODELS

Figure 2.14. For a tree–structured graph, each node $i$ partitions the graph into $|\Gamma(i)|$ disjoint subtrees.

Conditioned on $x_i$, the variables $x_j \setminus i$ in these subtrees are independent. The BP algorithm exploits this structure to recursively decompose the computation of $p(x_i | y)$ into a series of simpler, local calculations.

From the Hammersley–Clifford Theorem, Markov properties are expressed through the algebraic structure of the pairwise MRF's factorization into clique potentials. As illustrated in Fig. 2.15, tree–structured graphs allow multi–dimensional integrals (or summations) to be decomposed into a series of simpler, one–dimensional integrals. As in dynamic programming [24, 90, 303], the overall integral can then be computed via a recursion involving messages sent between neighboring nodes. This decomposition is an instance of the same distributive law underlying a variety of other algorithms [4, 50, 255], including the fast Fourier transform. Critically, because messages are shared among similar decompositions associated with different nodes, BP efficiently and simultaneously computes the desired marginals for all nodes in the graph.

$\overline{j \setminus i} \triangleq \{j\} \cup \{k \in V \mid \text{no path from } k \to j \text{ intersects } i\}$
Sum-Product for “Nearly” Trees

- Sum-product algorithm computes exact marginal distributions for any factor graph which is tree-structured (no cycles)
- This includes some undirected graphs with cycles
Early work on belief propagation (Pearl, 1980’s) focused on directed graphical models, was complicated by directionality of edges and multiple parents (polytrees).

Factor graph framework makes this a simple special case.
Junction Tree Propagation for Factor Graphs with Cycles
Undirected Graphical Models

- $X_s$ → Random variable associated with node $s$.
- $\mathcal{V}$ → Set of $N$ nodes or vertices, $\{1, 2, \ldots, N\}$
- $\mathcal{E}$ → Set of undirected edges $(s,t)$ linking pairs of nodes.
- $\mathcal{C}$ → Set of cliques (fully connected subsets) of nodes.

Undirected Markov Random Field:

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(x_c)$$

$$Z = \sum_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c)$$

- $\phi_c(x_c) \geq 0$ → Arbitrary non-negative potential function.
- $Z > 0$ → Normalization constant, or partition function. Defined as sum over all possible joint states.
An Undirected View of Graph Elimination

Elimination Order: (6,5,4,3,2,1)

- For the node being eliminated, *marginalize* over values of associate variable
- Compute a *new potential function* involving all other variables which are neighbors of the just-marginalized variables, and are thus related by some potential
- If necessary, add edges to create a *clique* for the newly created potential
An Undirected View of Graph Elimination

Elimination Order: (6, 5, 4, 3, 2, 1)
An Undirected View of Graph Elimination

Elimination Order: (6, 5, 4, 3, 2, 1)
The **clique tree** contains the cliques (fully connected subsets) which are generated as elimination executes.

The **separator sets** contain the variables which are shared among each linked pair of cliques.
# Marginal Inference Algorithms

## One Marginal
- **Tree**
  - Elimination applied to leaves of tree

## All Marginals
- **Graph**
  - Elimination algorithm
  - Junction tree algorithm:
    - Belief propagation on a junction tree (special clique tree)

**Belief Propagation or Sum-Product Algorithm**
Clique-Based Inference Algorithms

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c) \]

\[ z_c = \{x_s \mid s \in c\}, \, c \in C \]

\[ p(z) \propto \prod_{c \in C} \phi_c(z_c) \]

- For each clique \( c \), define a variable \( z_c \) which enumerates joint configurations of dependent variables
- Does this define an equivalent joint distribution?

**PROBLEM:** We have defined multiple copies of the variables in the true model, but not enforced any relationships among them
Clique-Based Inference Algorithms

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c) \]

\[ z_c = \{ x_s \mid s \in c \}, \ c \in C \]

\[ p(z) \propto \prod_{c \in C} \phi_c(z_c) \prod_{d \neq c} \phi_{cd}(z_c, z_d) \]

- For each clique \( c \), define a variable \( z_c \) which enumerates joint configurations of dependent variables.
- Add potentials enforcing consistency between all pairs of clique variables which share one of the original variables:

\[ \psi_{cd}(z_c, z_d) = \begin{cases} 
1 & z_c = z_d \text{ for all } x_s, s \in c \cap d \\
0 & \text{otherwise}
\end{cases} \]

**PROBLEM:** The graph may have a large number of pairwise consistency constraints, and inference will be difficult.
Clique-Based Inference Algorithms

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c) \]

\[ z_c = \{ x_s | s \in c \}, \ c \in C \]

\[ p(z) \propto \prod_{c \in C} \phi_c(z_c) \prod_{(c,d) \in E(C)} \phi_{cd}(z_c, z_d) \]

- For each clique \( c \), define a variable \( z_c \) which enumerates joint configurations of dependent variables
- Add potentials enforcing consistency between some subset of pairs of cliques, taking advantage of transitivity of equality:

\[ x_a = x_b, x_b = x_c \rightarrow x_a = x_c \]

**Question:** How many edges are needed for global consistency? When can we build a tree-structured clique graph?
• This clique tree has the *junction tree* property: the clique nodes containing any variable from the original model form a *connected* subtree.

• We can exactly represent the distribution *ignoring redundant constraints*.

• Note that not all clique trees are junction trees:
The Junction Tree Property: A clique tree is a *junction tree* if for every pair of cliques *C* and *D*, all cliques on the (unique) path between *C* and *D* contain any shared variables.

- Given a set of cliques, how can we efficiently find a clique tree with the junction tree (running intersection) property?
- How can we be sure that at least one junction tree exists?
- Strategy: Augment the graph with additional edges
  - Cliques of original graph are always subsets of cliques of the augmented graph, so original distribution still factorizes appropriately
  - As cliques grow, will eventually be able to construct a junction tree

**Question:** Which undirected graphs have junction trees?
The Junction Tree Property: A clique tree is a junction tree if for every pair of cliques C and D, all cliques on the (unique) path between C and D contain any shared variables.

- A chord is an edge connecting two non-adjacent nodes in some cycle
- A cycle is chordless if it contains no chords
- A graph is triangulated if it contains no chordless cycles

**Theorem:** The maximal cliques of a graph have a corresponding junction tree if and only if that undirected graph is triangulated

**Lemma:** For a non-complete triangulated graph with at least 3 nodes, there is a decomposition of the nodes into disjoint sets A, B, S such that S separates A from B, and S is complete.

- Key induction argument in constructing junction tree from triangulation
- Implies existence of elimination ordering which introduces no new edges
Constructing a Junction Tree

**Theorem:** A clique tree is a junction tree if and only if it is a maximal spanning tree of the weighted clique intersection graph

- **Graph:** Fully connected with nodes corresponding to maximal cliques
- **Edge weights:** Cardinality of separator set (intersection) of cliques
- **Computational complexity:** Quadratic in number of maximal cliques

**Junction Tree Algorithms** for General-Purpose Inference

1. Triangulate the target undirected graphical model
   - Any elimination ordering generates a valid triangulation
   - Optimal triangulation is NP-hard (in multiple ways)
2. Arrange triangulated cliques into a junction tree
3. Execute variant of sum-product algorithm on junction tree
Consider a junction tree linking a set of cliques, with pairwise equality constraints among intersections:

\[ m_{ts}(x_s) \propto \sum_{x_t} \psi_{st}(x_s, x_t) \psi_{t}(x_t) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t) \]

Messages are functions of the separating sets (variables shared among cliques):

\[ \mu_{ji}(x_{S_{ji}}) \propto \sum_{x_{R_j}} \psi_{C_j}(x_{C_j}) \prod_{k \neq j} \mu_{kj}(x_{S_{kj}}) \]

\[ R_j = C_j \setminus S_{ij} \]