Each step maximizes log-likelihood over $y_k$ alone. Coordinate ascent.

\[ L(D) = \frac{1}{2} \sum_{i} \frac{m(x_i)}{\psi^{(i)}(x_i)} \log \psi^{(i)}(x_i) - \frac{N}{2} \sum_{i} \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} \left( \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} - 1 \right) \]

\[ = \frac{m(x_i)}{\psi^{(i)}(x_i)} - \frac{N}{2} \sum_{i} \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} \left( \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} - 1 \right) \]

\[ = \frac{m(x_i)}{\psi^{(i)}(x_i)} - \frac{N}{2} \sum_{i} \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} \left( \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} - 1 \right) \]

\[ = \frac{m(x_i)}{\psi^{(i)}(x_i)} - \frac{N}{2} \sum_{i} \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} \left( \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} - 1 \right) \]

\[ \text{Example:} \quad \psi^{(i)}(x_i) = \frac{m(x_i)}{\psi^{(i)}(x_i)} - \frac{N}{2} \sum_{i} \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} \left( \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} - 1 \right) \]

\[ = \frac{m(x_i)}{\psi^{(i)}(x_i)} - \frac{N}{2} \sum_{i} \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} \left( \frac{\psi^{(i)}(x_i)}{\psi^{(i)}(x_i)} - 1 \right) \]

Summary: have efficient algos for ADDC, phy(x) & Max Likelihood.

Junction Tree Algorithm: general inference tool for graphical mod.

- Workhorse, inference spf is main too.

- Node elimination algo must be repeated for each query.
- Redundant work gets discarded & repeated.

- Instead look closely at cliques we are eliminating connect the m & their separators.

- Tree: unique path between vertices, root node.
from node elimination

- junction tree property: each node appears connected (in a path) to its other instantiations through the tree & all subtrees.

- more generally, undirected graphs not on maximal cliques:

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

- want to generally compute \( p(X_c | X_e) \) or \( p(X_e | X_c) \)

- efficiently infer marginals & cond.

- for a single clique

  \[ p(X_c, X_e) \] then set \( p(X_e) = \sum_{X_c} p(X_c, X_e) \)

  \[ p(X_e | X_c) = \frac{p(X_c, X_e)}{p(X_c)} \]

**Example**: \( A \rightarrow B \rightarrow C \)

\[ p(x) = p(x_A) p(x_B | x_A) p(x_C | x_B) \]

- easily can get \( p(A, B) \) by \( p(x_A) p(x_B | x_A) \)

- but how to get \( p(B, C) \)?

First \( p(x) = \prod_{c} p(x_k | x_{\sim c}) = \frac{1}{Z} \prod_{c} \psi_c(x_c) \)

- moralization
Junction Trees Continued: want local marginals everywhere
\[ \Pi \Psi(X_i) = \prod_{S \in \text{cliques}} \prod_{\delta \in \text{separators}} \Psi_S(X_i) \]
Can write
\[ \Psi_{AB} = p(X_A) p(X_B | X_A) = p(X_A, X_B) \]
\[ \Psi_{BC} = p(X_C | X_B) \neq p(X_B, X_C) \]

Would like all cliques
More flexible notation needed:

\[ p(X) = \frac{1}{Z} \prod_{S \in \text{cliques}} \prod_{\delta \in \text{separators}} \Phi_\delta(X_i) \]

2 = 1 in practice
Put this to a normal set

Separators are function of intersection of neighbouring cliques
Supercut, if all \( \Psi_S(X_i) = 1 \) get previous.
But doesn't span anything new. Since any separator is subset of clique

![Graph and notation]

TRUNCULATE

... directed graph

\[ p(X) = \frac{p(A, B) p(B, C)}{p(B)} \]
\[ \Psi_{AB} = \prod_{\delta} \phi_\delta \]

But division by zero can occur, but numerator will also be 0 so treat \( \frac{0}{0} = 0 \)

Since
\[ p(B) = \prod_{\delta} p(B | \delta) \]
How to get the product over marginals form from \( p(x; k_{Na}) \)?

**Junction Tree Algorithm**

**Message Passing between cliques**

To get marginals with \( p(x) \) staying consistent:

\[
v_A = 0 \quad s = G \quad w \neq x
\]

**Mini-example**

Marginal:

\[
\phi^*_{S'} = \frac{\psi_S}{\sum_{w \in S} \psi_w} \quad \phi^*_{S''} = \frac{\psi_{S''}}{\sum_{w \in S} \psi_w}
\]

\[
p(S) = \frac{\psi_S}{\sum_{w \in S} \psi_w} \quad i.e. \quad p(B) = \frac{\psi_B}{\sum_{w \in B} \psi_w}
\]

If \( \psi \) is unknown:

\[
\phi^*_{S'} = \frac{\phi^*_{S'} \psi_{S'}}{\phi^*_{S'} \psi_{S'} + \phi^*_{S''} \psi_{S''}} \quad \text{where} \quad \text{p(x) unchanged}
\]

From \( s \) to \( U \):

\[
\psi_{S'} = \phi_{S'} \psi_{S'} \quad \text{AND} \quad \psi_{S''} = \phi_{S''} \psi_{S''}
\]

Can pool these out now.

Full clique marginals \( V \& W \) agree with marginal on \( S \).

**Mini-example 2**

\[
\phi^*_{S'} = \frac{\psi_{S'}}{\sum_{w \in S} \psi_w} \quad \phi^*_{S''} = \frac{\psi_{S''}}{\sum_{w \in S} \psi_w}
\]

with evidence \( x = 1 \), Init as above:

\[
\phi^*_{S'} = \frac{\psi_{S'}}{\psi_S} \quad \phi^*_{S''} = \frac{\psi_{S''}}{\psi_S}
\]

and \( \psi_{S'} = \psi_{S''} = \psi_S \), as before. End.

**Message Passing** Makes marginals with proper interactions.

On big junction tree, can keep iterating messages but inefficient.
- Send message only after hearing from all neighbours.

JTA: - No need to iterate mindlessly.

- Initialize: pick a root
  - $\Phi = 1$ for all cliques
  - $\Psi_c(x_0) = p(x; I(x_i)) \forall i$
    
    $p(x) = \prod_i p(x; I(x_i)) = \prod_c \Psi_c(x_0)$

    $= \prod_c \Psi_c(x_0) \quad \forall x_0$

    $\prod_c \Psi_c(x_0) \Leftarrow \text{done} = 1$

- Update cliques & separators on a tree structure recursively

```
Collect Evidence (node)
for each child of node
  $\mathcal{I}$ update (node, collect Evidence (child))
return (node);
```

```
Distribute Evidence (node)
for each child of node
  $\mathcal{I}$ update (child, node)
  distribute evidence (child)
```

collect

```
root
```

```
root
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```

```
node
```
TRIANGULATION

To get a valid Junction Tree → triangulate

Undiv. Node Elimination Alog → Triangulates Graph

Triangulation (like Moralization) adds links → more complex graph, fewer independencies

such that: No 4 or more node cycle within graph where some nodes do not l

i.e. a 4-cycle

3-cycle

2-cycle

1-cycle

example:

triangle:

optimal triangulation (adds fewest links) is NP

but any triangulation (can be bad (heuristic)) is P

want to keep a clique size small

also need resulting clique graph to be junction tree

a) tree (no loops)

b) satisfy junction tree property: cliques between path a Pb & Pb contain AB nodes
- When junction tree algo completes (induction proof)
  \[ \mathcal{C} = p(x_c, \overline{x_e}) \]
  \[ \Phi_S = p(x_S, \overline{x_E}) \]  
  - all cliques are marginals.

- Hugin (Junction Tree Algorithm) \(\rightarrow\) Jensen's book
  
  - Moralize (poly complex)
  
  - Introduce evidence \(x_e \rightarrow \overline{x_e}\) (poly)
  
  - Triangulate (poly or NP for optimal)

- Construct Junction Tree (poly, Kruskal)

- Propagate Probabilities (poly in # of cliques, expo in clique size)
  \[ \mathcal{A}^* = \sum_S \Psi_S \]
  \[ \Psi_W^* = \frac{\Phi_C^* \Psi_W}{\Phi_S} \]

- Without Proof: Triangulated Graph \(\iff\) decomposable

- With Proof: Decomposable \(\iff\) junction tree
  
  - largest cardinality of separator nodes
  
  \[ \text{max weight spanning tree} \]

- Not a junction tree
  \[ \text{ separators } = (B \cap C \cap D) \]
  \[ |\text{ separators }| = |D| + |(CD)| \]
  \[ = 3 \]

- Algo to find junction tree: max card. of separators \(\rightarrow\) Kruskal
- Kruskal: Max Span Tree: 1) Begin with no edges between cliques
   2) Add edge with max separator card.
   3) Make sure we don’t make a loop
   4) Repeat 2 until graph connected
   (all cliques are connected via a path)

- Example: Example: B, C, A

- Can compute fast marginals
- Can also compute fast argmax

\[ p(X_1, X_2) = \text{argmax} \left\{ \text{argmax} \left\{ \text{argmax} \left\{ \cdots \right\} \right\} \right\} \]

\[ p(X_1, X_2) = \text{argmax} \left\{ \text{argmax} \left\{ \text{argmax} \left\{ \cdots \right\} \right\} \right\} \]

\[ p(X_1, X_2) = \text{argmax} \left\{ \text{argmax} \left\{ \text{argmax} \left\{ \cdots \right\} \right\} \right\} \]

- i.e. what is most likely state of patient if he has

- Fever & headache?

\[ p(X_1, X_2) = \text{argmax} \left\{ \text{argmax} \left\{ \text{argmax} \left\{ \cdots \right\} \right\} \right\} \]

\[ p(X_1, X_2) = \text{argmax} \left\{ \text{argmax} \left\{ \text{argmax} \left\{ \cdots \right\} \right\} \right\} \]

\[ p(X_1, X_2) = \text{argmax} \left\{ \text{argmax} \left\{ \text{argmax} \left\{ \cdots \right\} \right\} \right\} \]