Inference Problems

- Compute the likelihood of observed data
- Compute the marginal distribution \( p(x_A) \) over a particular subset of nodes \( A \subset V \)
- Compute the conditional distribution \( p(x_A | x_B) \) for disjoint subsets \( A \) and \( B \)
- Compute a mode of the density \( \hat{x} = \arg \max_{x \in \mathcal{X}^m} p(x) \)

- Methods we have
  - Brute force
  - Elimination
  - Message Passing
    - (Forward-backward, Max-product /BP, Junction Tree)

  Individual computations independent  Sharing intermediate terms

© Eric Xing @ CMU, 2005-2015
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

Belief: Must be 14 of us

Slides adapted from Matt Gormley (2016)

adapted from MacKay (2003) textbook
Sum-Product Belief Propagation

Belief:
Must be 14 of us

Each soldier receives reports from all branches of tree

Factor Message

matrix-vector product (for a binary factor)

\[
\mu_{\alpha \rightarrow i}(x_i) = \sum \psi_{\alpha}(x_{\alpha}) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \rightarrow \alpha}(x_{\alpha}[i])
\]

\[\begin{array}{c|c|c}
\text{P} & \text{a} & \text{n} \\
\hline
\text{d} & 0.8 & 0.16 \\
\hline
\text{n} & 8 & 0.2 \\
\end{array} \]

\[\begin{array}{c|c|c}
\text{P} & \text{a} & \text{n} \\
\hline
\text{d} & 0.1 & 0.8 \\
\hline
\text{n} & 1 & 0 \\
\end{array} \]

\[\begin{array}{c|c|c}
\text{P} & \text{v} & \text{n} \\
\hline
\text{d} & 1 & 1 \\
\hline
\text{n} & 8 & 0.2 \\
\end{array} \]
Junction Tree Revisited

- General Algorithm on Graphs with Cycles
  - Steps:
    - Triangularization
    - Construct JT

=> Message Passing on Clique Trees

\[ \widetilde{\phi}_{S}(x_{S}) \leftarrow \sum_{x_{B} \setminus S} \phi_{B}(x_{B}) \]

\[ \phi_{C}(x_{C}) \leftarrow \frac{\widetilde{\phi}_{S}(x_{S})}{\phi_{S}(x_{S})} \phi_{C}(x_{C}) \]
An Ising model on 2-D image

- Nodes encode hidden information (patch-identity).
- They receive local information from the image (brightness, color).
- Information is propagated through the graph over its edges.
- Edges encode ‘compatibility’ between nodes.
Why Approximate Inference?

\[ p(X) = \frac{1}{Z} \exp\left\{ \sum_{i<j} \theta_{ij} x_i x_j + \sum_i \theta_{i0} x_i \right\} \]
For an MxM grid the new factor has $2^M$ entries.

In general, for high treewidth graphs like this, we turn to approximate inference (which we’ll cover soon!)
Approaches to inference

- **Exact inference algorithms**
  - The elimination algorithm
  - Message-passing algorithm (sum-product, belief propagation)
  - The junction tree algorithms

- **Approximate inference techniques**
  - Variational algorithms
    - Loopy belief propagation
    - Mean field approximation
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods
Recap of Belief Propagation
Recap: Belief Propagation

\[ M_{k \rightarrow i}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j) \psi_i(x_i) \prod_k M_{k \rightarrow i}(x_i) \]

\[ b_i(x_i) \propto \psi(x_i) \prod_k M_{k \rightarrow i}(x_i) \]
Beliefs and messages in FG

\[ m_{i \rightarrow a}(x_i) = \prod_{c \in N(i) \setminus a} m_{c \rightarrow i}(x_i) \]

\[ b_i(x_i) \propto \prod_{a \in N(i)} m_{a \rightarrow i}(x_i) \]

“beliefs”

“messages”

\[ b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} m_{i \rightarrow a}(x_i) \]

\[ m_{a \rightarrow i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a) \setminus i} m_{j \rightarrow a}(x_j) \]
(Acyclic) Belief Propagation

In a factor graph with no cycles:
1. Pick any node to serve as the root.
2. Send messages from the leaves to the root.
3. Send messages from the root to the leaves.
A node computes an outgoing message along an edge only after it has received incoming messages along all its other edges.

Slides adapted from Matt Gormley (2016)
(Acyclic) Belief Propagation

In a factor graph with no cycles:
1. Pick any node to serve as the root.
2. Send messages from the leaves to the root.
3. Send messages from the root to the leaves.
A node computes an outgoing message along an edge only after it has received incoming messages along all its other edges.

Slides adapted from Matt Gormley (2016)
What if there is a loop?
What if the graph is loopy?
Belief Propagation on loopy graphs

- **BP Message-update Rules**

\[
M_{i \rightarrow j}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j) \psi_i(x_i) \prod_k M_{k \rightarrow i}(x_i)
\]

- **May not converge or converge to a wrong solution**

\[
b_i(x_i) \propto \psi_i(x_i) \prod_k M_k(x_k)
\]
A fixed point iteration procedure that tries to minimize $F_{\text{bethe}}$

Start with random initialization of messages and beliefs

While not converged do

$$b_i(x_i) \propto \prod_{a \in N(i)} m_{a \rightarrow i}(x_i) \quad b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} m_{i \rightarrow a}(x_i)$$

$$m^\text{new}_{i \rightarrow a}(x_i) = \prod_{c \in N(i) \setminus a} m_{c \rightarrow i}(x_i) \quad m^\text{new}_{a \rightarrow i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a) \setminus i} m_{j \rightarrow a}(x_j)$$

At convergence, stationarity properties are guaranteed

However, not guaranteed to converge!
Loopy Belief Propagation

- If BP is used on graphs with loops, messages *may* circulate indefinitely

- But let’s **run it anyway** and hope for the best … 😊

- **How to stop it?**
  - Stop after fixed # of iterations
  - Stop when no significant change in beliefs
  - If solution is not oscillatory but converges, it usually is a good approximation

---

Loopy-belief Propagation for Approximate Inference: An Empirical Study
Kevin Murphy, Yair Weiss, and Michael Jordan.
*UAI ’99 (Uncertainty in AI).*
So what is going on?

- Is it a dirty hack that you bet your luck?
How to measure how close we are to the correct answer?
Approximate Inference

- Let us call the actual distribution $P$

$$P(X) = \frac{1}{Z} \prod_{f_a \in F} f_a(X_a)$$

- We wish to find a distribution $Q$ such that $Q$ is a “good” approximation to $P$

- Recall the definition of KL-divergence

$$KL(Q_1 \parallel Q_2) = \sum_X Q_1(X) \log \left( \frac{Q_1(X)}{Q_2(X)} \right)$$

- $KL(Q_1 \parallel Q_2) \geq 0$
- $KL(Q_1 \parallel Q_2) = 0$ iff $Q_1 = Q_2$
- We can therefore use KL as a scoring function to decide a good $Q$
- But, $KL(Q_1 \parallel Q_2) \neq KL(Q_2 \parallel Q_1)$
Which KL?

- Computing $KL(P\|Q)$ requires inference!
- But $KL(Q\|P)$ can be computed without performing inference on $P$

\[
KL(Q \| P) = \sum_X Q(X) \log \left( \frac{Q(X)}{P(X)} \right)
\]

\[
= \sum_X Q(X) \log Q(X) - \sum_X Q(X) \log P(X)
\]

\[
= -H_Q(X) - E_Q \log P(X)
\]

- Using $P(X) = \frac{1}{Z} \prod_{f_a \in F} f_a(X_a)$

\[
KL(Q \| P) = -H_Q(X) - E_Q \log \left( \frac{1}{Z} \prod_{f_a \in F} f_a(X_a) \right)
\]

\[
= -H_Q(X) - \log \frac{1}{Z} - \sum_{f_a \in F} E_Q \log f_a(X_a)
\]
Optimization function

\[ KL(Q \parallel P) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a) + \log Z \]

- We will call \( F(P, Q) \) the “Free energy” *
- \( F(P, P) =? \)
- \( F(P, Q) \geq F(P, P) \)

*Gibbs Free Energy
The Energy Functional

- Let us look at the functional

\[ F(P, Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a) \]

- \( \sum_{f_a \in F} E_Q \log f_a(X_a) \) can be computed if we have marginals over each \( f_a \)

- \( H_Q = -\sum_X Q(X) \log Q(X) \) is harder! Requires summation over all possible values

- Computing \( F \), is therefore hard in general.

- Our goals is to:

\[ Q^* = \arg \max_{Q \in \mathcal{Q}} F(P, Q) \]

Can we approximate it?

Can we suggest an easy family?
Work out a simple case
Do you remember this from lecture 6?

\[ p(a, b, c, d) = \frac{\phi(a, b, c)\phi(b, c, d)}{Z} = \frac{p(a, b, c)p(b, c, d)}{p(c, b)} \]
A Tree example

\[ F(P, Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a) \]

\[ p(X_1, \ldots, X_8) = \frac{1}{Z} \phi_{43}(X_4, X_3) \phi_{32}(X_3, X_2) \phi_{21}(X_2, X_1) \phi_{15}(X_1, X_5) \phi_{56}(X_5, X_6) \cdots \]

\[ H(X_1, \cdots, X_8) = - \sum_{a \in \text{num}} \mathbb{E}[\log p(X_a)] + \sum_{i \in \text{den}} \mathbb{E}[\log p(x_i)] \]

\[ F(X_1, \cdots, X_8) = - \sum_{a \in \text{num}} \mathbb{E}[\log \frac{p(X_a)}{f(X_a)}] + \sum_{i \in \text{den}} \mathbb{E}[\log \frac{p(X_i)}{f(X_i)}] \]
For a general tree

\[ F(P, Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a) \]

- The probability can be written as:
  \[ b(x) = \prod_a b_a(x_a) \prod_i b_i(x_i)^{1-d_i} \]

\[ H_{\text{tree}} = -\sum_a \sum_{x_a} b_a(x_a) \ln b_a(x_a) + \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i) \]

\[ F_{\text{Tree}} = \sum_a \sum_{x_a} b_a(x_a) \ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \ln b_i(x_i) \]

Degree of the node

- involves summation over edges and vertices and is therefore easy to compute

\[ = F_{12} + F_{23} + \ldots + F_{67} + F_{78} - F_1 - F_5 - F_2 - F_6 - F_3 - F_7 \]
Let’s extend it to a general graph
Bethe Approximation to Gibbs Free Energy

- For a general graph, choose $\hat{F}(P,Q) = F_{\text{Bethe}}$

$$H_{\text{Bethe}} = -\sum_a \sum_{x_a} b_a(x_a) \ln b_a(x_a) + \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i)$$

$$F_{\text{Bethe}} = \sum_a \sum_{x_a} b_a(x_a) \ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \ln b_i(x_i) = -\langle f_a(x_a) \rangle - H_{\text{Bethe}}$$

- Called “Bethe approximation” after the physicist Hans Bethe

$$F_{\text{Bethe}} = F_{12} + F_{23} + \ldots + F_{67} + F_{78} - F_1 - F_5 - 2F_2 - 2F_6 \ldots - F_8$$

- Equal to the exact Gibbs free energy when the factor graph is a tree
- In general, $H_{\text{Bethe}}$ is not the same as the $H$ of a tree
Bethe Approximation

● Pros:
  ● Easy to compute, since entropy term involves sum over pairwise and single variables

● Cons:
  ● \( \hat{F}(P, Q) = F_{\text{bethe}} \) may or may not be well connected to \( F(P, Q) \)
  ● It could, in general, be greater, equal or less than \( F(P, Q) \)

● Optimize each \( b(x_a) \)'s.
  ● For discrete belief, constrained opt. with \textit{Lagrangian} multiplier
  ● For continuous belief, not yet a general formula
  ● Not always converge
Bethe Free Energy for Factor Graph of Discrete RVs

\[
F_{\text{Bethe}} = \sum_a \sum x_a b_a(x_a) \ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum x_i b_i(x_i) \ln b_i(x_i)
\]

\[
H_{\text{Bethe}} = - \sum_a \sum x_a b_a(x_a) \ln b_a(x_a) + \sum_i (d_i - 1) \sum x_i b_i(x_i) \ln b_i(x_i)
\]

\[
F_{\text{Bethe}} = - \sum_a \langle f_a(x_a) \rangle - H_{\text{Bethe}}
\]

How about optimizing this:

\[
\min_{b_a(x_a), b_i(x_i)} F_{\text{Bethe}}
\]

Subject to: 
Minimizing the Bethe Free Energy

\[ L = F_{\text{Bethe}} + \sum_i \gamma_i \{1 - \sum_{x_i} b_i(x_i)\} \]

\[ + \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{ai}(x_i) \left\{ b_i(x_i) - \sum_{X_a \setminus x_i} b_a(X_a) \right\} \]

- Set derivative to zero
Constrained Minimization of the Bethe Free Energy

\[
L = F_{\text{Bethe}} + \sum_i \gamma_i \left\{ \sum_{x_i} b_i(x_i) - 1 \right\} \\
+ \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{ai}(x_i) \left\{ \sum_{X_a \setminus x_i} b_a(X_a) - b_i(x_i) \right\}
\]

\[
\frac{\partial L}{\partial b_i(x_i)} = 0 \quad \Leftrightarrow \quad b_i(x_i) \propto \exp \left( \frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x_i) \right)
\]

\[
\frac{\partial L}{\partial b_a(X_a)} = 0 \quad \Leftrightarrow \quad b_a(X_a) \propto \exp \left( -E_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i) \right)
\]
Bethe = BP on FG

- We had:
  \[ b_i(x_i) \propto \exp \left( \frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x_i) \right) \]
  \[ b_a(X_a) \propto \exp \left( -\log f_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i) \right) \]
- Identify \( \lambda_{ai}(x_i) = \log(m_{i\rightarrow a}(x_i)) = \log \prod_{b \in N(i) \neq a} m_{b\rightarrow i}(x_i) \)
- to obtain BP equations:

\[ b_i(x_i) \propto f_i(x_i) \prod_{a \in N(i)} m_{a\rightarrow i}(x_i) \]

“beliefs”

\[ b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} \prod_{c \in N(i) \setminus a} m_{c\rightarrow i}(x_i) \]

The “belief” is the BP approximation of the marginal probability.
Using $b_{a \rightarrow i}(x_i) = \sum_{X_a \setminus x_i} b_a(X_a)$, we get

$$m_{a \rightarrow i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a) \setminus i} m_{b \rightarrow j}(x_j) \prod_{b \in N(j) \setminus a}$$

(A sum product algorithm)
Summary so far

\[ P(X) = \frac{1}{Z} \prod_{f_a \in F} f_a(X_a) \]

\[ F(P, Q) = -H_Q(X) - \sum_{f_a \in F} E_Q \log f_a(X_a) \]

\[ \hat{F}(P, Q) = \sum_a \sum_{x_a} b_a(x_a) \log \frac{f_a(x_a)}{b_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \log b_i(x_i) \]

\[ b_a(X_a) \propto \exp \left( -\log f_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x_i) \right) \]

\[ b_i(x_i) \propto \exp \left( \frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x_i) \right) \]
The Theory Behind LBP

- For a distribution $p(X|\theta)$ associated with a complex graph, computing the marginal (or conditional) probability of arbitrary random variable(s) is intractable.

- Variational methods
  - formulating probabilistic inference as an optimization problem:
    $$q^* = \arg\min_{q \in S} \left\{ F_{\text{Bethe}}(p, q) \right\}$$
    $$F_{\text{Bethe}} = \sum_a \sum_{x_a} b_a(x_a) \ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \ln b_i(x_i) = -\langle f_a(x_a) \rangle - H_{\text{bethe}}$$

Optimizing the marginal in the Bethe energy is a way to make $q$ tractable!
But we do not optimize \( q(X) \) explicitly, focus on the set of beliefs

\[ b = \{ b_{i,j} = \tau(x_i, x_j), \quad b_i = \tau(x_i) \} \]

Relax the optimization problem

- approximate objective: \( H_q \approx F(b) \)
- relaxed feasible set: \( M \rightarrow M_o \quad (M_o \supseteq M) \)

\[ b^* = \arg \min_{b \in M_o} \left\{ \langle E \rangle_b + F(b) \right\} \]

The loopy BP algorithm:

- a fixed point iteration procedure that tries to solve \( b^* \)
The Theory Behind LBP

- But we do not optimize $q(X)$ explicitly, focus on the set of beliefs

  - *e.g.*, $b = \{b_{i,j} = \tau(x_i, x_j), \ b_i = \tau(x_i)\}$

- Relax the optimization problem

  - approximate objective: $H_{Net} = H(b_{i,j}, b_i)$
  - relaxed feasible set: $\mathcal{M}_o = \{ \tau \geq 0 | \sum_{x_i} \tau(x_i) = 1, \sum_{x_i} \tau(x_i, x_j) = \tau(x_j) \}$

  $$b^* = \arg\min_{b \in \mathcal{M}_o} \left\{ \langle E \rangle_b + F(b) \right\}$$

- The loopy BP algorithm:

  - a fixed point iteration procedure that tries to solve $b^*$