Approximate inference in graphical models.

- Forward and Backward KL divergence
- Variational Inference
- Mean Field: Naive and Structured
- Marginal Polytope
- Local Polytope
- Relaxation methods
- Loopy BP
- LP relaxations for MAP inference

Figures from D. Sontag, Murphy’s book
Approximate marginal inference

- Given the joint $p(x_1, \cdots, x_n)$ represented as a graphical model, we want to perform **marginal inference**, e.g., $p(x_i|e)$
- We showed in last lecture that doing this exactly is NP-hard
- We also covered variable elimination (VE), which can solve these type of queries for any graphical model, but ···
- Almost all approximate inference algorithms in practice are
  - Variational algorithms (e.g., mean-field, loopy belief propagation)
  - Sampling methods (e.g., Gibbs sampling, MCMC)
Variational Methods

- **Goal**: Approximate a difficult distribution \( p(x|e) \) with a new distribution \( q(x) \)
  - \( p(x|e) \) and \( q(x) \) should be "close"
  - Computation on \( q(x) \) should be easy

- How should we measure distance between distributions?
- The **Kullback-Leibler divergence** (KL-divergence) between two distributions \( p \) and \( q \) is defined as
  \[
  D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}
  \]

- It measures the expected number of extra bits (nats) required to describe samples from \( p(x) \) using a code based on \( q \) instead of \( p \)
- \( D(p||q) \geq 0 \) for all \( p, q \), with equality if and only if \( p = q \)
- The KL-divergence is asymmetric
Suppose $p$ is the true distribution

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

This is difficult to optimize because the expectations w.r.t. $p$ are typically intractable.

We can reverse the KL

$$D(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}$$

Typically the expectation w.r.t. $q$ will be tractable, but ···

··· computing $p(x)$ is still hard, due to the partition function

What can we do?
Variational Inference

Let’s look at the unnormalized distribution

\[
J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}
\]

\[
= \sum_x q(x) \log \frac{q(x)}{Z \cdot p(x)}
\]

\[
= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z
\]

\[
= KL(q \| p) - \log Z
\]

Since \( Z \) is constant, by minimizing \( J(q) \), we will force \( q \) to become close to \( p \).

The KL is always non-negative, so we see that \( J(q) \) is an upper bound on the negative log likelihood (NLL)

\[
J(q) = KL(q \| p) - \log Z \geq - \log Z = - \log p(D)
\]
(1). We can alternatively write

\[ J(q) = \mathbb{E}_q[\log q(x)] + \mathbb{E}_q[-\log \hat{p}(x)] = -\mathbb{H}(q) + \mathbb{E}_q[E(x)] \]

which is the expected energy minus the entropy.

In physics, \( J(q) \) is called the \textbf{variational free energy} or \textbf{Helmholtz free energy}

(2). Another alternative:

\[ J(q) = \mathbb{E}_q[\log q(x) - \log p(x)p(D)] \]
\[ = \mathbb{E}_q[\log q(x) - \log p(x) - \log p(D)] \]
\[ = \mathbb{E}_q[-\log p(D)] + KL(q||p) \]

This is the expected NLL plus a penalty term that measures how far apart the two distributions are.
Before we do something let’s inspect again

\[
KL(p \| q) = \sum_x p(x) \log \frac{p(x)}{q(x)}
\]

What is the difference between the solution to

\[
\arg \min_q KL(p \| q)
\]

and

\[
\arg \min_q KL(q \| p)
\]

They differ only when \( q \) is minimized over a restricted set of probability distribution \( Q = \{q_1, \cdots \} \), and \( p \neq q \). Why?
Forward or Reverse KL

- Minimizing $KL(p||q)$ or $KL(q||p)$ will give different results

- I projection, or Information projection

\[
KL(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}
\]

This is infinite if $p(x) = 0$ and $q(x) > 0$. Thus we must ensure that if $p(x) = 0$ then $q(x) = 0$

- Thus the reverse KL is zero forcing and $q$ will under-estimate the support of $p$

- M projection or moment projection

\[
KL(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}
\]

- This is infinite if $q(x) = 0$ and $p(x) > 0$. This is zero avoiding, and the forward KL over-estimates the support of $p$
$q^* = \arg\min_{q \in Q} KL(p \| q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$

$p(x)$ is a 2D Gaussian and $Q$ is the set of all Gaussian distributions with diagonal covariance matrices.
KL divergence - 1 projection

\[ q^* = \arg \min_{q \in Q} KL(q \| p) = \sum_x q(x) \log \frac{q(x)}{p(x)} \]

\( p(x) \) is a 2D Gaussian and \( Q \) is the set of all Gaussian distributions with diagonal covariance matrices.
In this example, both the M-projection and I-projection find an approximate $q(x)$ that has the correct mean (i.e., $\mathbb{E}_p(z) = \mathbb{E}_q(x)$).

What if $p(x)$ is multimodal?
M projection (Mixture of Gaussians)

\[ q^* = \underset{q \in Q}{\arg \min} KL(p || q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \]

\( p(x) \) is a mixture of two 2D Gaussians and \( Q \) is the set of all 2D Gaussian distributions (with arbitrary covariance matrices).

M-projection yields a distribution \( q(x) \) with the correct mean and covariance.
I projection (Mixture of Gaussians)

\[ q^* = \arg \min_{q \in Q} KL(q \| p) = \sum_x q(x) \log \frac{q(x)}{p(x)} \]

The I-projection does not necessarily yield the correct moments

\( p = \text{Blue, } q^* = \text{Red (two local minima!)} \)
One of the most popular variational inference algorithms [Opper & Saad 01]

Assume that the posterior fully factorizes

$$q(x) = \prod_i q_i(x_i)$$

Our goal is to

$$\min_{q_1, \ldots, q_D} KL(q \| p)$$

where we optimize over the parameters of each marginal distribution \(q_i\)

Minimize the upper bound \(J(q) \geq -\log p(D)\) or alternatively we want to maximize the lower bound

$$L(q) = -J(q) = \sum_x q(x) \log \frac{\tilde{p}(x)}{q(x)} \leq \log p(D)$$

We can do the maximization one node at a time, in an iterative fashion
Mean Field Updates

- Focus on $q_j$ (holding all other terms constant)

$$L(q_j) = \sum_x \prod_i q_i(x) \left[ \log \tilde{p}(x) - \sum_k \log q_k(x_k) \right]$$

$$= \sum_{x_j} \sum_{x_{-j}} q_j(x_j) \prod_{i \neq j} q_i(x_i) \left[ \log \tilde{p}(x) - \sum_k \log q_k(x_k) \right]$$

$$= \sum_{x_j} q_j(x_j) \sum_{x_{-j}} \prod_{i \neq j} q_i(x_i) \log \tilde{p}(x) -$$

$$\sum_{x_j} q_j(x_j) \sum_{x_{-j}} \prod_{i \neq j} q_i(x_i) \left[ \sum_{k \neq j} \log q_k(x_k) + \log q_j(x_j) \right]$$

$$= \sum_{x_j} q_j(x_j) \log f_j(x_j) - \sum_{x_j} q_j(x_j) \log q_j(x_j) + \text{const}$$

where

$$\log f_j(x_j) = \sum_{x_{-j}} \prod_{i \neq j} q_i(x_i) \log \tilde{p}(x) = \mathbb{E}_{-q_j} [\log \tilde{p}(x)]$$

- So we average out all the variables except $x_j$, and can rewrite $L(q_j)$ as

$$L(q_j) = -KL(q_j \parallel f_j)$$
Suppose that we have an arbitrary graphical model

\[ p(x; \theta) = \frac{1}{Z(\theta)} \prod_{c \in C} \phi_c(x_c) = \exp \left( \sum_{c \in C} \theta_c(x_c) - \ln Z(\theta) \right) \]

We can compute the KL

\[
KL(q || p) = \sum_x q(x) \ln \frac{q(x)}{p(x)}
\]

\[
= -\sum_x q(x) \ln p(x) - \sum_x q(x) \ln \frac{1}{q(x)}
\]

\[
= -\sum_x q(x) \left( \sum_{c \in C} \theta_c(x_c) - \ln Z(\theta) \right) - H(q(x))
\]

\[
= -\sum_{c \in C} \sum_x q(x) \theta_c(x_c) + \sum_x q(x) \ln Z(\theta) - H(q(x))
\]

\[
= -\sum_{c \in C} \mathbb{E}_q[\theta_c(x_c)] + \ln Z(\theta) - H(q(x))
\]

The partition function can be considered as constant when minimizing over \( q \)
Mean Field for Variational Inference

\[
\max_{q \in Q} \sum_{c \in C} \sum_{x_c} q(x_c) \theta_c(x_c) + H(q(x))
\]

- Although this function is concave and thus in theory should be easy to optimize, we need some compact way of representing \( q(x) \)

- **Mean field**: assume a factored representation of the joint distribution

\[
q(x) = \prod_{i \in V} q_i(x_i)
\]

This is called "naive” mean field
Suppose that $Q$ consists of all fully factorized distributions, then we can simplify

$$\max_{q \in Q} \sum_{c \in C} \sum_{x_c} q(x_c) \theta_c(x_c) + H(q(x))$$

since $q(x_c) = \prod_{i \in C} q_i(x_i)$

The joint entropy decomposes as a sum of local entropies

$$H(q) = - \sum_x q(x) \ln q(x)$$

$$= - \sum_x q(x) \ln \prod_{i \in V} q_i(x_i) = - \sum_x q(x) \sum_{i \in V} \ln q_i(x_i)$$

$$= - \sum_{i \in V} \sum_x q(x) \ln q_i(x_i)$$

$$= - \sum_{i \in V} \sum_{x_i} q_i(x_i) \ln q_i(x_i) \sum_{x_{-i}} q(x_{-i}|x_i) = \sum_{i \in V} H(q_i)$$
Suppose that $Q$ consists of all fully factorized distributions, then we can simplify

$$\max_{q \in Q} \sum_{c \in C} \sum_{x_c} q(x_c) \theta_c(x_c) + H(q(x))$$

since $q(x_c) = \prod_{i \in C} q_i(x_i)$

The joint entropy decomposes as a sum of local ones $H(q) = \sum_{i \in V} H(q_i)$

Putting these together, we obtain

$$\max_q \sum_{c \in C} \sum_{x_c} \theta_c(x_c) \prod_{i \in c} q_i(x_i) + \sum_{i \in V} H(q_i)$$

subject to the constraints

$$q_i(x_i) \geq 0 \quad \forall i \in V, x_i$$

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i \in V$$
Naive Mean Field for Pairwise MRFs

- For pairwise MRFs we have
  \[
  \max_q \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j) q_i(x_i) q_j(x_j) - \sum_{i \in V} \sum_{x_i} q_i(x_i) \ln q_i(x_i) \tag{1}
  \]

- This is a non-concave optimization problem, with many local maxima!

- We can do block coordinate ascent
  1. For each \( i \in V \)
     - Fully maximize Eq. (1) wrt \( \{q_i(x_i), \forall x_i\} \)
  2. Repeat until convergence

- Constructing the Lagrangian, taking the derivatives and setting to zero yields the update

  \[
  q_i(x_i) \leftarrow \frac{1}{Z_i} \exp \left\{ \theta_i(x_i) + \sum_{j \in N(i)} \sum_{x_j} q_j(x_j) \theta_{ij}(x_i, x_j) \right\}
  \]

- See Mean field example for the Ising Model, Murphy 21.3.2
Structured mean-field approximations

- Rather than assuming a fully-factored distribution for $q$, we can use a structured approximation, such as a spanning tree.
- For example, for a factorial HMM, a good approximation may be a product of chain-structured models (see Murphy 21.4.1).
Mean field inference approximates posterior as product of marginal distributions

Allows use of different forms for each variable: useful when inferring statistical parameters of models, or regression weights

An alternative approximate inference algorithm is **loopy belief propagation**

Same algorithm shown to do exact inference in trees last class

In loopy graphs, BP not guaranteed to give correct results, may not converge, but often works well in practice
Algorithm 22.1: Loopy belief propagation for a pairwise MRF

1. Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
2. Initialize messages $m_{s \rightarrow t}(x_t) = 1$ for all edges $s \rightarrow t$;
3. Initialize beliefs $\text{bel}_s(x_s) = 1$ for all nodes $s$;
4. repeat
5. Send message on each edge
   $$m_{s \rightarrow t}(x_t) = \sum_{x_s} \left( \psi_s(x_s) \psi_{st}(x_s, x_t) \prod_{u \in \text{nbr}_s \setminus t} m_{u \rightarrow s}(x_s) \right);$$
6. Update belief of each node $\text{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \text{nbr}_s} m_{t \rightarrow s}(x_s)$;
7. until beliefs don’t change significantly;
8. Return marginal beliefs $\text{bel}_s(x_s)$;
Loopy BP for Factor Graph

\[
m_{i \rightarrow f}(x_i) = \prod_{h \in M(i) \setminus f} m_{h \rightarrow i}(x_i)
\]

\[
m_{f \rightarrow i}(x_i) = \sum_{x_c \setminus x_i} f(x_c) \prod_{j \in N(f) \setminus i} m_{j \rightarrow f}(x_j)
\]

\[
\mu_i(x_i) \propto \prod_{f \in M(i)} m_{f \rightarrow i}(x_i)
\]
Convergence of LBP

- Can we predict when will converge?
  - Unroll messages across time in a computation tree: \( T \) iterations of LBP is exact computation in tree of height \( T + 1 \)
  - If leaves’ effect on root diminishes over time will converge

- Can we make it more likely to converge?
  - Damp the messages to avoid oscillations
  - Can we speed up convergence?

- Change from synchronous to asynchronous updates
  - Update sets of nodes at a time, e.g., spanning trees (tree reparameterization)
More theoretical analysis of LBP from variational point of view: (Wainwright & Jordan, 2008)

- Dense tome
- Simplify by considering pairwise UGMs, discrete variables
Variational Inference for Graphical Models

- Suppose that we have an arbitrary graphical model
  \[ p(x; \theta) = \frac{1}{Z(\theta)} \prod_{c \in C} \phi_c(x_c) = \exp \left( \sum_{c \in C} \theta_c(x_c) - \ln Z(\theta) \right) \]

- We can compute the KL
  \[ KL(q \| p) = \sum_x q(x) \ln \frac{q(x)}{p(x)} \]
  \[ = - \sum_x q(x) \ln p(x) - \sum_x q(x) \ln \frac{1}{q(x)} \]
  \[ = - \sum_x q(x) \left( \sum_{c \in C} \theta_c(x_c) - \ln Z(\theta) \right) - H(q(x)) \]
  \[ = - \sum_{c \in C} \sum_x q(x) \theta_c(x_c) + \sum_x q(x) \ln Z(\theta) - H(q(x)) \]
  \[ = - \sum_{c \in C} \mathbb{E}_q[\theta_c(x_c)] + \ln Z(\theta) - H(q(x)) \]

- The partition function is a constant when minimizing over \( q \)
The log-partition Function

- Since $KL(q||p) \geq 0$ we have
  \[ -\sum_{c \in C} \mathbb{E}_q[\theta_c(x_c)] + \ln Z(\theta) - H(q(x)) \geq 0 \]

  which implies
  \[ \ln Z(\theta) \geq \sum_{c \in C} \mathbb{E}_q[\theta_c(x_c)] + H(q(x)) \]

- Thus, any approximating distribution $q(x)$ gives a lower bound on the log-partition function

- Recall that $KL(p||q) = 0$ if and only if $p = q$. Thus, if we optimize over all distributions we have
  \[ \ln Z(\theta) = \max_q \sum_{c \in C} \mathbb{E}_q[\theta_c(x_c)] + H(q(x)) \]

- This casts exact inference as a variational optimization problem
Rewriting Objective in terms of Moments

\[
\ln Z(\theta) = \max_q \sum_{c \in C} \mathbb{E}_q[\theta_c(x_c)] + H(q(x)) \\
= \max_q \sum_{c \in C} \sum_{x} q(x)\theta_c(x_c) + H(q(x)) \\
= \max_q \sum_{c \in C} \sum_{x_c} q(x_c)\theta_c(x_c) + H(q(x))
\]

- Assume that \( p(x) \) is in the exponential family, and let \( f(x) \) be its sufficient statistic vector
- Define \( \mu_q = \mathbb{E}[f(x)] \) to be the marginals of \( q(x) \)
- We can re-write the objective as

\[
\ln Z(\theta) = \max_{\mu \in M} \max_{q: \mathbb{E}_q[f(x)] = \mu} \sum_{c \in C} \sum_{x_c} \mu_c(x_c)\theta_c(x_c) + H(q(x))
\]

where \( M \) is the **marginal polytope**, having all valid marginal vectors
Rewriting Objective in terms of Moments

- We next push the max inside

\[
\ln Z(\theta) = \max_{\mu \in M} \sum_c \sum_{x_c} \theta_c(x_c) \mu_c(x_c) + H(\mu)
\]

\[
H(\mu) = \max_{q : \mathbb{E}_q[f(x)] = \mu} H(q)
\]

- For discrete random variables, the marginal polytope \( M \) is the set of all mean parameters for the given model that can be generated from a valid prob. distribution

\[
M = \left\{ \mu \in \mathbb{R}^d \mid \exists p \text{ s.t. } \mu = \sum_{x \in \mathcal{X}^m} p(x)f(x) \text{ for some } p(x) \geq 0, \sum_{x \in \mathcal{X}^m} p(x) = 1 \right\}
\]

\[= \text{conv} \{f(x), x \in \mathcal{X}^m\}\]

with \( \text{conv} \) the convex hull (it has exponential number of facets)

- For a discrete-variable MRF, the sufficient statistic vector \( f(x) \) is simply the concatenation of indicator functions for each clique of variables that appear together in a potential function

- For example, if we have a pairwise MRF on binary variables with \( m = |V| \) variables and \( |E| \) edges, \( d = 2m + 4|E| \)
Marginal Polytope for Discrete MRFs

\[ \vec{\mu}' = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \]

\[ \vec{\mu} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \]

\[ \frac{1}{2} (\vec{\mu}' + \vec{\mu}) \]

Valid marginal probabilities

Marginal polytope

(Wainwright & Jordan, '03)

Assignment for \( X_1 \)
Assignment for \( X_2 \)
Assignment for \( X_3 \)
Edge assignment for \( X_1X_3 \)
Edge assignment for \( X_1X_2 \)
Edge assignment for \( X_2X_3 \)

\( X_1 = 1 \)
\( X_2 = 1 \)
\( X_3 = 0 \)

\( X_1 = 0 \)
\( X_2 = 1 \)
\( X_3 = 0 \)
\[
\ln Z(\theta) = \max_{\mu \in M} \sum_{c \in C} \sum_{x_c} \theta_c(x_c) \mu_c(x_c) + H(\mu)
\]

We still haven’t achieved anything, because:

- The marginal polytope \(M\) is complex to describe (in general, exponentially many vertices and facets)
- \(H(\mu)\) is very difficult to compute or optimize over

We now make two approximations:

- We replace \(M\) with a relaxation of the marginal polytope, e.g. the local consistency constraints \(M_L\)
- We replace \(H(\mu)\) with a function \(\tilde{H}(\mu)\) which approximates \(H(\mu)\)
Local Consistency Constraints

- For every "cluster" of variables to choose a local assignment
  \[ \mu_i(x_i) \in \{0, 1\} \quad \forall i \in V, x_i \]
  \[ \sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V \]
  \[ \mu_{ij}(x_i, x_j) \in \{0, 1\} \quad \forall i, j \in E, x_i, x_j \]
  \[ \sum_{x_i, x_j} \mu_{ij}(x_i, x_j) = 1 \quad \forall i, j \in E \]

- Enforce that these local assignments are globally consistent
  \[ \mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \]
  \[ \mu_j(x_j) = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j \]

- The local consistency polytope, \( M_L \) is defined by these constraints
- The \( \mu_i \) and \( \mu_{ij} \) are called pseudo-marginals
polytope for a tree-structured MRF, and the pseudomarginals are the marginals. marginal polytope, i.e., $M \subseteq M_L$
Mean-field vs relaxation

$$\max_{q \in Q} \sum_{c \in C} \sum_{x_c} q(x_c) \theta_c(x_c) + H(q(x))$$

- Relaxation algorithms work directly with pseudo-marginals which may not be consistent with any joint distribution.
- Mean-field algorithms assume a factored representation of the joint distribution.

$q(x) = \prod_{i \in V} q_i(x_i)$
Naive Mean-Field

- Using the same notation naive mean-field is:

\[
(*) \max_\mu \sum_{c \in C} \sum_{x_c} \mu_c(x_c) \theta_c(x_c) + \sum_{i \in V} H(\mu_i) \quad \text{subject to}
\]

\[
\mu_i(x_i) \geq 0, \quad \forall i \in V, x_i
\]

\[
\sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V
\]

\[
\mu_c(x_c) = \prod_{i \in c} \mu_i(x_i)
\]

- Corresponds to optimizing over an inner bound on the marginal polytope:

- We obtain a lower bound on the partition function, i.e., \((*) \leq \ln Z(\theta)\)
MAP Inference

- Recall the MAP inference task

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

we assume any evidence has been subsumed into the potentials

- As the partition function is a constant we can alternatively

\[ \arg \max_x \prod_{c \in C} \phi_c(x_c) \]

This is the **max product** inference task

- Since the log is monotonic, let \( \theta_c(x_c) = \log \phi_c(x_c) \)

\[ \arg \max_x \sum_{c \in C} \theta_c(x_c) \]

This is called the **max-sum**
Application: protein side-chain placement

- Find "minimum energy" configuration of amino acid side-chains along fixed carbon backbone:

  ![Diagram of protein side-chains and rotamers](attachment:image.png)

  - Side-chain (corresponding to 1 amino acid)
  - Protein backbone

- Orientations of the side-chains are represented by discretized angles called rotamers

- Rotamer choices for nearby amino acids are energetically coupled (attractive and repulsive forces)

(Yanover, Meltzer, Weiss '06)
Application: Dependency parser

- Given a sentence, predict the dependency tree that relates the words

- Arc from head word of each phrase to words that modify it

- May be non-projective: each word and its descendants may not be a contiguous subsequence

- $m$ words $\Rightarrow m(m-1)$ binary arc selection variables $x_{ij} \in \{0, 1\}

- We represent the problem as

$$\max_x \theta_T(x) + \sum_{ij} \theta_{ij}(x_{ij}) + \sum_i \theta_{i|}(x_{i|})$$

with $x_{i|} = \{x_{ij}\}_{j \neq i}$ (all outgoing edges)
Application: Semantic Segmentation

- Use Potts to encode that neighboring pixels are likely to have the same discrete label and hence belong to the same segment

\[
p(x, \theta) = \max_x \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{i,j}(x_i, x_j)
\]
MAP as an integer linear program (ILP)

- MAP as a discrete optimization problem is
  \[ x^* = \arg \max_x \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j) \]

- To turn this into an integer linear program (ILP) we introduce indicator variables
  1. \( \mu_i(x_i) \), one for each \( i \in V \) and state \( x_i \)
  2. \( \mu_{ij}(x_i, x_j) \), one for each edge \( ij \in E \) and pair of states \( x_i, x_j \)

- The objective function is then
  \[ \max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i) \mu_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j) \]

- What is the dimension of \( \mu \), if binary variables?
- Are these two problems equivalent?
Constraints

\[
\max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i) \mu_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j)
\]

- For every "cluster" of variables to choose a local assignment
  \[\mu_i(x_i) \in \{0, 1\} \quad \forall i \in V, x_i\]
  \[\sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V\]
  \[\mu_{ij}(x_i, x_j) \in \{0, 1\} \quad \forall i, j \in E, x_i, x_j\]
  \[\sum_{x_i, x_j} \mu_{ij}(x_i, x_j) = 1 \quad \forall i, j \in E\]

- Enforce that these local assignments are globally consistent
  \[\mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i\]
  \[\mu_j(x_j) = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j\]
MAP as an integer linear program (ILP)

\[
\max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i) \mu_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j)
\]

subject to:

\[
\mu_i(x_i) \in \{0, 1\} \quad \forall i \in V, x_i
\]

\[
\mu_{ij}(x_i, x_j) \in \{0, 1\} \quad \forall i, j \in E, x_i, x_j
\]

\[
\sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V
\]

\[
\mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i
\]

\[
\mu_j(x_j) = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\]

- Many extremely good off-the-shelf solvers, such as CPLEX and Gurobi
- But it might be too slow...
Linear Programming Relaxation for MAP

\[ MAP(\theta) = \max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i) \mu_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j) \]

subject to:

\[ \mu_i(x_i) \in \{0, 1\} \quad \forall i \in V, x_i \]
\[ \mu_{ij}(x_i, x_j) \in \{0, 1\} \quad \forall i, j \in E, x_i, x_j \]
\[ \sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V \]
\[ \mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \]
\[ \mu_j(x_j) = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j \]

- Relax integrality constraints, allowing the variables to be between 0 and 1

\[ \mu_i(x_i) \in [0, 1] \quad \forall i \in V, x_i \quad \mu_{ij}(x_i, x_j) \in [0, 1] \quad \forall ij \in E, x_i, x_j \]
Linear Programing Relaxation for MAP

\[
LP(\theta) = \max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i) \mu_i(x_i) + \sum_{ij} (x_i, x_j) \mu_{ij}(x_i, x_j)
\]

\[
\begin{align*}
\mu_i(x_i) & \in [0, 1] \quad \forall i \in V, x_i \\
\mu_{ij}(x_i, x_j) & \in [0, 1] \quad \forall i, j \in E, x_i, x_j \\
\sum_{x_i} \mu_i(x_i) & = 1 \quad \forall i \in V \\
\mu_i(x_i) & = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \\
\mu_j(x_j) & = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\end{align*}
\]

- Linear programs can be solved relatively efficient via Simplex method, interior point, ellipsoid algorithm
- Since the LP relaxation maximizes over a larger set of solutions, its value can only be higher

\[
MAP(\theta) \leq LP(\theta)
\]

- LP relaxation is tight for tree-structured MRFs
- Faster algorithms by deriving the dual (dual variables represent messages)
- Zero limit temperature of the variational inference for Marginals
Introducing Lagrange multipliers and solving we get (see Murphy 22.3.5.4)

\[ M_{i \rightarrow j}(x_i) \propto \max_{x_j} \left[ \exp\{\theta_{ij}(x_i, x_j) + \theta_j(x_j)\} \prod_{u \in N(j) \setminus i} M_{u \rightarrow j}(x_j) \right] \]

Thus we pass messages for a fixed number of iterations, or until the messages do not change too much.

We decode the local scoring functions by

\[ \mu_s(x_s) \propto \exp(\theta_s(x_s)) \prod_{t \in N(s)} M_{t \rightarrow s}(x_s) \]

We then compute the maximal value of \( \mu_s(x_s) \)

What if two solutions that have the same score?
Stereo Estimation

- Tsukuba images from Middlebury stereo database

![Left](image1.png) ![Right](image2.png)

- MRF for each pixel, with states the disparity
- Our unary is the matching term

$$\theta_i(d_i) = |L(x + d_i, y) - R(x, y)|$$

where pixel $p_i = (x, y)$

- The pairwise factor $\theta_{ij}$ between neighboring pixels favor smoothness
Stereo Estimation

- If we only use the unary terms. How would you do inference in this case?

- If full graphical model

left, right, up, down sweeps

[Credit: Coughlan BP Tutorial]
Subsequent iterations:

2 3 4 5

... 20

Note:
Little change after first few iterations.
Model can be improved to give better results
-- this is just a simple example to illustrate BP.

[Credit: Coughlan BP Tutorial]