RE-CAP: MRFS & BELIEF PROPAGATION

- Last time we talked about MRFs, learning despite the intractable partition function.
- Then we talked about MRFs, and specifically pair-wise MRFs (although we said you could convert generic clique MRFs to a pair-wise factor graph).
- We talked about Belief Propagation & Loopy BP as a way of doing inference (for both max-marginal and MAP estimation).
TRW-* Algorithms: Represent a graph with cycles as a convex combination of overlapping trees (each tree is formed by dropping some edges).

Do belief propagation on each tree, using input from other trees and appropriate weighting.

**Gaussian Belief Propagation**

We've considered belief propagation for discrete-valued variables because messages are finite dimensional vectors. What if the variables are continuous-valued? The messages would be functions of continuous variables. But if all our unary and pairwise potentials were uni-variate and bi-variate Gaussians, so would their products and marginals with updated means and co-variances.

\[
m_{i\rightarrow j}(x) = \int \frac{\Psi_i(x)\psi_j(x, x_j) \prod_{k:(i,k)\in E \land k\neq j} m_{k\rightarrow i}(x_k)}{\int \Psi_i(x) \prod_{k:(i,k)\in E \land k\neq j} m_{k\rightarrow i}(x_k)} \, dx_j
\]

- But if all our unary and pairwise potentials were uni-variate and bi-variate Gaussians, so would their products and marginals with updated means and co-variances.
- Pass messages that contain these means and co-variances.

**Particle Belief Propagation**

The other option is to create a finite sample-set for each continuous variable, for all.

Define \( \Psi_i(x) \) as a continuous function of \( x_i \), but by averaging over the samples of \( x_i \).

Define \( m_{i\rightarrow j}(x) \) as a continuous function of \( x_j \), but by averaging over the samples of \( x_i \).

\[
m_{i\rightarrow j}(x) = \frac{1}{S} \sum_{x'_j} \left[ \Psi_i(x'_j) \psi_j(x'_j, x_j) \prod_{k:(j,k)\in E \land k\neq i} m_{k\rightarrow i}(x'_k) \right]
\]

- Periodically re-sample all continuous variables based on the current estimate of their marginal distribution.

**MRFS: PAIRWISE TERMS**

For MRFS, the pairwise terms are often made in the form factorized by a label dependent factor

\[
P(x_i \in V) = \frac{1}{Z} \prod_{x \in V} \Psi_i(x) \prod_{(i,j)\in E} \psi_{ij}(x_i, x_j)
\]

\[
\Psi_i(x_i) = -\log E_{ij,x_i}, \quad \psi_{ij}(x_i, x_j) = -\log E_{ij}(x_i, x_j)
\]

- It's common to make the pairwise term factor into a term for labels scaled by a location dependent factor

\[
E_{ij}(x_i, x_j) = \mu_{ij} \times V(x_i, x_j)
\]

- Examples of label compatibility \( V \):

\[
V(x_i, x_j) = \delta[x_i \neq x_j]
\]

Called Pott's Model. Penalty if \( x_i \) not equal to \( x_j \).

Sometimes, you might have different compatibilities. For example, for the case of disparity:

\[
V(x_i, x_j) = \min(2, |x_i - x_j|)
\]

Or a general matrix for segmentation class labels (car and road has less penalty than car and water).
MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(\{x_i \in V\}) = \frac{1}{Z} \prod_{x_i \in V} \Psi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \]

\[ \Psi_i(x_i) = -\log E_{\omega x_i}, \quad \psi_{ij}(x_i, x_j) = -\log E_{\eta}(x_i, x_j) \]

- Its common to make the pairwise term factor into a term for labels scaled by a location dependent factor
  \[ E_{\eta}(x_i, x_j) = \mu_{ij} \times V(x_i, x_j) \]

The per-edge weight \( \mu_{ij} \) can also be based on different things:

- \( \mu_{ij} = 1 \), but graph has only edges in a neighborhood: \( E = \bigcup \{ (i, j) : j \in \mathcal{N}_i \} \)
  E.g., for each pixel \( i \), add an edge to all neighbors in a 5x5 neighborhood around that pixel \( i \).

- Same, but \( \mu_{ij} = \exp \left( -\frac{||p_i - p_j||^4}{2\sigma^4} \right) \). Here \( p_i \) is the \( [x, y] \) location of pixel \( i \) and \( \sigma^2 \) is some chosen paramter.

- Same, but \( \mu_{ij} = \exp \left( -\frac{||p_i - p_j||^4}{2\sigma^2} - \frac{||I_i - I_j||^4}{2\epsilon^4} \right) \). Add a term based on pixel intensities \( I_j \) in a reference image. Kind of like a bilateral filter, but used to define compatibility weights.

MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(\{x_i \in V\}) = \frac{1}{Z} \prod_{x_i \in V} \Psi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \]

- Kind of like message passing as we'll see.
- Motivated as approximating the joint probability as a product of marginals.
  \[ P(\{x_i \in V\}) = \prod_i Q_i(x_i) \]

And then compute \( x_i = \arg\max Q_i(x_i) \) for each node/pixel independently.

MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(\{x_i \in V\}) = \frac{1}{Z} \prod_{x_i \in V} \Psi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \]

\[ P(\{x_i \in V\}) = \prod_i Q_i(x_i) \]

- Compute \( Q_i \) iteratively with a loopy BP-like algorithm.
- Update \( Q_i \) in each step based messages from all neighbors
  - In message from \( i \rightarrow j \), this will include \( i \)'s belief based on message from \( j \rightarrow i \).
MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(x_i \in V) = \frac{1}{Z} \prod_{x_i \in V} \prod_{(x_i, x_j) \in E} \psi_{ij}(x_i, x_j) \]

Each \( x_i \in \mathcal{L} \).

- At iteration 0, set \( Q_0^i(x_i) = \frac{1}{Z_i} \Psi_i(x_i) \), with \( Z_i = \sum_{x_i} Q_0^i(x_i) = 1 \).
- At each iteration \( t \), compute for all \( i \):
  \[ \Psi_i(x_i) \prod_{j \in \mathcal{E}_i} \exp \left( \sum_{x_j \in \mathcal{L}_j} Q_t^j(x_j) \log \psi_{ij}(x_i, x_j) \right) \]

Think of \( \sum_{x_j \in \mathcal{L}_j} Q_t^j(x_j) \log \psi_{ij}(x_i, x_j) = E_{Q_t^i(x_i)} \log \psi_{ij}(x_i, x_j) \)

Kind of like \( m_{j \rightarrow i} \), except in log-domain and \( Q_t^j(x_j) \) includes belief from \( i \) to \( j \) in the previous iteration.

Repeat iteratively till convergence.

MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(x_i \in V) = \frac{1}{Z} \prod_{x_i \in V} \prod_{(x_i, x_j) \in E} \psi_{ij}(x_i, x_j) \]

Each \( x_i \in \mathcal{L} \).

- At iteration 0, set \( Q_0^i(x_i) = \frac{1}{Z_i} \Psi_i(x_i) \), with \( Z_i = \sum_{x_i} Q_0^i(x_i) = 1 \).
- At each iteration \( t \), compute for all \( i \):
  \[ \Psi_i(x_i) \prod_{j \in \mathcal{E}_i} \exp \left( \sum_{x_j \in \mathcal{L}_j} Q_t^j(x_j) \log \psi_{ij}(x_i, x_j) \right) \]

- Normalize to get \( Q_t^{i+1}(x_i) = \frac{1}{Z_t^{i+1}} \Psi_i(x_i) \prod_{j \in \mathcal{E}_i} \exp \left( \sum_{x_j \in \mathcal{L}_j} Q_t^j(x_j) \log \psi_{ij}(x_i, x_j) \right) \)

Let’s consider the case when \( E_{ij}(x_i, x_j) = \mu_j V(x_i, x_j) \) and all pixels have the same set of possible labels.

\[ Q_t^{i+1}(x_i) = \exp \left( -E_{i}(x_i) - \sum_{j \in \mathcal{E}_i} \sum_{x_j \in \mathcal{L}_j} Q_t^j(x_j) \mu_j V(x_i, x_j) \right) \]

\[ Q_t^{i+1}(x_i) = \exp \left( -E_{i}(x_i) - \sum_{j \in \mathcal{E}_i} \sum_{x_j \in \mathcal{L}_j} Q_t^j(x_j) \mu_j V(x_i, x_j) \right) \]
MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(x_i \in V) = \frac{1}{Z} \exp \left( - \sum_{x \in V} E_i(x_i) - \sum_{\{x,y\} \in E} E_{ij}(x_i, x_j) \right) \]

\[ E_{ij}(x_i, x_j) = \mu_{ij} V(x_i, x_j) \]

\[ Q^{t+1}(x_i) = \exp \left( -E_i(x_i) - \sum_{I \notin E} V(x_i, I') \sum_{J \in \{I\} \in E} Q^{t}(I') \mu_y \right) \]

- Let's say our graph is over pixels in an image, and there are \( C \) classes in \( I \). Then,
  - \( E_{ij} \) is a \( H \times W \times C \) image. Represent as \( U[n] \in \mathbb{R}^C \) for unary.
  - At each iteration, \( Q^t[n], Q'[n] \) are also \( H \times W \times C \) images.
  - \( V \) is a \( C \times C \) matrix

Let's consider the case where \( \mu_y = k(-n_j - n_i) \).

Think of it as a kernel with support based on size of neighborhood \( \mathcal{N}_i \).

E.g., \( k[n] = \exp(-n^2/(2\sigma^2)) \)

MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(x_i \in V) = \frac{1}{Z} \exp \left( - \sum_{x \in V} E_i(x_i) - \sum_{\{x,y\} \in E} E_{ij}(x_i, x_j) \right) \]

\[ E_{ij}(x_i, x_j) = \mu_{ij} V(x_i, x_j) \]

\[ Q^{t+1}(x_i) = \exp \left( -E_i(x_i) - \sum_{I \notin E} V(x_i, I') \sum_{J \in \{I\} \in E} Q^{t}(I') \mu_y \right) \]

- The summation \( \sum_{\{I\} \in E} Q^{t}(I') \mu_y \) is a convolution on the previous marginal distribution "image"!
  - Convolving all channels with the same kernel \( k \).
  - The \( \times V^T \) implies doing a matrix-multiply to the \( C \)-dimensional vector at each pixel location \( (Q' \ast k)[n] \).
  - Get \( Q^{t+1}[n] \) from \( Q^{t+1}[n] \) by doing a per-pixel normalization.

```python
def bilateral(Q, n, sigma):
    # Compute bilateral filter

def normalize(Q, I, sigma):
    # Normalize Q to sum to 1

def conv2(Q, k):
    # Convolution implementation

Q[0, 0] = exp(-Q[0, 0])
```

MRF INFERENCE: MEAN FIELD ALGORITHM

\[ P(x_i \in V) = \frac{1}{Z} \exp \left( - \sum_{x \in V} E_i(x_i) - \sum_{\{x,y\} \in E} E_{ij}(x_i, x_j) \right) \]

\[ E_{ij}(x_i, x_j) = \mu_{ij} V(x_i, x_j) \]

\[ Q^{t+1}(x_i) = \exp \left( -E_i(x_i) - \sum_{I \notin E} V(x_i, I') \sum_{J \in \{I\} \in E} Q^{t}(I') \mu_y \right) \]

**Spatial** \( \mu_y \)

\[ \mu_y = k(-n_j - n_i), \quad Q^{t+1}[n] = \exp\left(-U[n] - (Q' \ast k)[n] \times V^T\right) \]

**Intensity-based** \( \mu_y \)

\[ \mu_y = \exp\left(-\frac{||n_i - n_j||^2}{2\sigma^2} - \frac{||I[n_i] - I[n_j][n]^2}{2\sigma^2} \right) \]

Replace the convolution with Bilateral filtering!

\[ Q^{t+1}[n] = \exp\left(-U[n] - BFilt(Q'[n]; I[n], \sigma_j, \sigma_I) \times V^T\right) \]
MRF INFERENCE: MEAN FIELD ALGORITHM

In the fully connected MRF case with "bilateral" weights, there are efficient data-structures you can use to do the bilateral filtering.


These fully-connected MRFs are used in the successful DeepLab image segmentation method from Google!

MRF INFERENCE: GRAPH CUTS

Let’s consider the binary case where each $x_i \in \{\alpha, \beta\}$.

We’re going to convert this into a min-cut problem on a weighted graph $G$.

Without loss of generality, set $E_\alpha(x_i, x_j) = e_{ij} \delta(x_i \neq x_j)$ such that $e_{ij}$ is positive.

Can do this because there are only two labels: remaining to the unary terms.

$E_\alpha(x_i, x_j) = C(x_i = \alpha) + C(x_i = \beta) + e_{ij} \delta(x_i \neq x_j) + C$

Express $E_\alpha(x_i, x_j) = t^\alpha \delta(\alpha) + t^\beta \delta(\beta)$

Make sure these terms are positive (by adding a constant if necessary).
MRF INFERENCE: GRAPH CUTS

Treat these as weights on the edge.

Now solve the min-cut between $\alpha$ and $\beta$ in this graph: Delete edges to separate $\alpha$ and $\beta$ so that sum of weights on deleted edges is minimized.

From the cut graph, assign variables based on whether they remain connected to $\alpha$ or $\beta$.

This corresponds to minimizing the original MAP energy.

$$\{p, q, \ldots, r\} = \arg \min \sum_i \delta(x_i \neq \alpha) + \sum_i \delta(x_i \neq \beta) + \sum_{(i,j)} e_{ij} \delta(x_i \neq x_j)$$

- Treat these as weights on the edge.
- Now solve the min-cut between $\alpha$ and $\beta$ in this graph: Delete edges to separate $\alpha$ and $\beta$ so that sum of weights on deleted edges is minimized.
- From the cut graph, assign variables based on whether they remain connected to $\alpha$ or $\beta$.
- This corresponds to minimizing the original MAP energy.

MRF INFERENCE: GRAPH CUTS

Generalization to Multi-labels

- NP Hard, but approximate iterative algorithms that calls the binary graph-cut solver multiple times.
- At each iteration consider two kinds of moves.
  - $\alpha, \beta$ swap
    - For all pairs of labels $\alpha, \beta$
      - Consider the subset of nodes which have labels $\alpha$ or $\beta$.
      - Solve a binary cut to figure out whether to swap labels for some pixels in this sub-set.
      - See if this decreases the original energy. If so, keep, otherwise stick with original.
  - $\alpha$ expansion
    - For all labels $\alpha$
      - Solve a binary cut to figure out whether to set some pixels that are not $\alpha$ to $\alpha$.
      - See if this decreases the original energy. If so, keep, otherwise stick with original.
    - Requires conditions on $E_{ij}(x_i, x_j)$:
      $$E_{ij}(p, q) = 0 \Leftrightarrow p = q; E_{ij}(p, q) = E_{ij}(q, p) \geq 0; E_{ij}(p, q) \leq E_{ij}(p, r) + E_{ij}(q, r)$$
    - Satisfies all three: metric. Satisfies only the last two: semi-metric.

MRF INFERENCE: GRAPH CUTS

Generalization to Multi-labels

- $\alpha, \beta$ swap move.

Given a current labeling, and a candidate pair of $\alpha$ and $\beta$ to swap:

- Consider the set of variables $V_\alpha$ and $V_\beta$ that currently have these labels.
- Consider the union of these two sets, and set up a binary segmentation problem on these variables.
- Edge weights remain the same. But change the unary entries of the node as
  $$E_{i}^{\alpha}(x_i = \alpha) = E_{i}(x_i = \alpha) + \sum_{(i,j) \in E, j \in \bar{V}_\alpha \cup V_\beta} E_{ij}(\alpha, x_j)$$
  $$E_{i}^{\beta}(x_i = \beta) = E_{i}(x_i = \beta) + \sum_{(i,j) \in E, j \in \bar{V}_\alpha \cup V_\beta} E_{ij}(\beta, x_j)$$
  where $x_j$ is the current value of the variable (which is not $\alpha, \beta$)

- Solve a graph-cut to re-label variables in $V_\alpha$, $V_\beta$ to either $\alpha$ or $\beta$. 