Discrete Inference and Learning
Lecture 1

MVA
2017 – 2018

http://thoth.inrialpes.fr/~alahari/disinflearn

Slides based on material from Stephen Gould, Pushmeet Kohli, Nikos Komodakis, M. Pawan Kumar, Carsten Rother
Optimization problems

• Can be written as

\[
\min_{x} f(x) \quad \text{(optimize an objective function)} \\
\text{s.t. } x \in \mathcal{C} \quad \text{(subject to some constraints)}
\]

\textbf{feasible set}, containing all } x \text{ satisfying the constraints

discrete variables
Optimization problems

• Can be written as

\[
\min_x f(x) \quad \text{(optimize an objective function)}
\]

s.t. \( x \in \mathcal{C} \) \quad \text{(subject to some constraints)}

• Two main problems in this context
  – Optimize the objective \( \textbf{\textit{(inference)}} \)
  – Learn the parameters of \( f \) \( \textbf{\textit{(learning)}} \)
Optimization problems

• Several applications, e.g., computer vision

Low-level vision problems

Interactive figure-ground segmentation [Boykov and Jolly, 2001; Boykov and Funka-Lea, 2006]
Surface context [Hoiem et al., 2005]
Semantic labeling [He et al., 2004; Shotton et al., 2006; Gould et al., 2009]

Stereo matching [Kolmogorov and Zabih, 2001; Scharstein and Szeliski, 2002]
Image denoising [Felzenszwalb and Huttenlocher 2004]
Optimization problems

• Several applications, e.g., computer vision

High-level vision problems

Object detection [Felzenszwalb et al., 2008]

Pose estimation [Akhter and Black, 2015; Ramakrishna et al., 2012]

Scene understanding [Fouhey et al., 2014; Ladicky et al., 2010; Xiao et al., 2013; Yao et al., 2012]
Optimization problems

- Several applications, e.g., medical imaging
Optimization problems

• Inherent in all these problems are graphical models

Pixel labeling

Object detection
Pose estimation

Scene understanding
Conditional Markov Random Fields

- Also known as: Markov networks, undirected graphical models, MRFs
- Note: Not making a distinction between CRFs and MRFs
- $\mathbf{X} \in \mathcal{X}$: observed random variables
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$: output random variables
- $\mathbf{Y}_c$ are subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \psi_c(\mathbf{Y}_c; \mathbf{X})$$

Partition function $= \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \psi_c(\mathbf{Y}_c; \mathbf{X})$ Exponential number of configurations!
Maximum a posteriori (MAP) inference

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y \mid x) \]

\[ = \arg \max_{y \in \mathcal{Y}} \frac{1}{Z(x)} \prod_c \psi_c(y_c; x) \]

\[ = \arg \max_{y \in \mathcal{Y}} \log \left( \frac{1}{Z(x)} \prod_c \psi_c(y_c; x) \right) \]

\[ = \arg \max_{y \in \mathcal{Y}} \sum_c \log \psi_c(y_c; x) - \log Z(x) \]

\[ = \arg \max_{y \in \mathcal{Y}} \sum_c \log \psi_c(y_c; x) - E(Y; X) \]
Maximum a posteriori (MAP) inference

\[ y^* = \arg\max_{y \in \mathcal{Y}} P(y \mid x) = \arg\max_{y \in \mathcal{Y}} \sum_c \log \psi_c (Y_c; X) \]

\[ = \arg\min_{y \in \mathcal{Y}} E(y; x) \]

MAP inference \(\Leftrightarrow\) Energy minimization

The energy function is \( E(Y; X) = \sum_c \psi_c (Y_c; X) \)

where \( \psi_c (\cdot) = -\log \psi_c (\cdot) \)
Clique potentials

• Defines a mapping from an assignment of random variables to a real number

\[ \psi_c : \mathcal{V}_c \times \mathcal{X} \rightarrow \mathbb{R} \]

• Encodes a preference for assignments to the random variables (lower is better)

• Parameterized as \[ \psi_c(y_c; x) = w_c^T \phi_c(y_c; x) \]
Clique potentials

- **Arity**

\[ E(y; x) = \sum_c \psi_c(y_c; x) \]

\[ = \sum_{i \in V} \psi^U_i(y_i; x) + \sum_{ij \in E} \psi^P_{ij}(y_i, y_j; x) + \sum_{c \in C} \psi^H_c(y_c; x). \]

**Unary**  
**Pairwise**  
**Higher-order**
Clique potentials

- Arity

4-connected, $\mathcal{N}_4$

8-connected, $\mathcal{N}_8$
Reason 1: Texture modelling

Training images

Test image

Test image (60% Noise)

Result MRF
4-connected (neighbours)

Result MRF
4-connected

Result MRF
9-connected
(7 attractive; 2 repulsive)
Reason2: Discretization artefacts

[Boykov et al. ‘03; ‘05]
Graphical representation

• Example

\[ E(y) = \psi(y_1, y_2) + \psi(y_2, y_3) + \psi(y_3, y_4) + \psi(y_4, y_1) \]
Graphical representation

- Example

\[ E(y) = \sum_{i,j} \psi(y_i, y_j) \]
Graphical representation

• Example

\[ E(\mathbf{y}) = \psi(y_1, y_2, y_3, y_4) \]
A Computer Vision Application

Binary Image Segmentation

How?

Cost function Models our knowledge about natural images

Optimize cost function to obtain the segmentation
A Computer Vision Application

Binary Image Segmentation

Object - white, Background - green/grey

Each vertex corresponds to a pixel

Edges define a 4-neighbourhood grid graph

Assign a label to each vertex from $L = \{\text{obj}, \text{bkg}\}$
Graph $G = (V,E)$

Cost of a labelling $f : V \rightarrow L$

Object - white, Background - green/grey

Cost of label ‘obj’ low Cost of label ‘bkg’ high

A Computer Vision Application

Binary Image Segmentation
A Computer Vision Application

Binary Image Segmentation

Graph $G = (V, E)$

Cost of a labelling $f : V \rightarrow L$

Object - white, Background - green/grey

Per Vertex Cost

Cost of label ‘obj’ high Cost of label ‘bkg’ low

UNARY COST
Graph \( G = (V,E) \)

Cost of a labelling \( f : V \to L \)

- Cost of same label low
- Cost of different labels high

Object - white, Background - green/grey

A Computer Vision Application

Binary Image Segmentation

Per Edge Cost
A Computer Vision Application

Binary Image Segmentation

Object - white, Background - green/grey

Cost of a labelling \( f : V \rightarrow L \)

Cost of same label high

Cost of different labels low

Graph \( G = (V,E) \)

Per Edge Cost

PAIRWISE COST
A Computer Vision Application

Binary Image Segmentation

Object - white, Background - green/grey

Problem: Find the labelling with minimum cost $f^*$
A Computer Vision Application

Binary Image Segmentation

Graph $G = (V,E)$

Problem: Find the labelling with minimum cost $f^*$
Another Computer Vision Application

Stereo Correspondence

How?

Minimizing a cost function
Another Computer Vision Application

Stereo Correspondence

Graph $G = (V,E)$

- Vertex corresponds to a pixel
- Edges define grid graph

$L = \{\text{disparities}\}$
Another Computer Vision Application

Stereo Correspondence

Cost of labelling $f$:

Unary cost + Pairwise Cost

Find minimum cost $f^*$
The General Problem

Graph $G = (V, E)$

Discrete label set $L = \{1, 2, \ldots, h\}$

Assign a label to each vertex $f: V \rightarrow L$

Cost of a labelling $Q(f)$

 Unary Cost    Pairwise Cost

Find $f^* = \arg \min Q(f)$
Overview

• Basics: problem formulation
  – Energy Function
  – MAP Estimation
  – Computing min-marginals
  – Reparameterization

• Solutions
  – Belief Propagation and related methods [Lecture 1]
  – Graph cuts [Lecture 2]
Random Variables $V = \{V_a, V_b, \ldots\}$

Labels $L = \{l_0, l_1, \ldots\}$  

Data $D$

Labelling $f: \{a, b, \ldots\} \rightarrow \{0, 1, \ldots\}$

Energy Function
Energy Function

\[ Q(f) = \sum_a \theta_{a;f(a)} \]

Unary Potential

Easy to minimize

Neighbourhood
Energy Function

\[ E : (a,b) \in E \text{ iff } V_a \text{ and } V_b \text{ are neighbours} \]

\[ E = \{ (a,b), (b,c), (c,d) \} \]
Energy Function

Label $l_1$

Label $l_0$

$V_a$ $V_b$ $V_c$ $V_d$

$D_a$ $D_b$ $D_c$ $D_d$

Pairwise Potential

$Q(f) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$
Energy Function

\[ Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)} \]
Overview

• Basics: problem formulation
  – Energy Function
  – MAP Estimation
  – Computing min-marginals
  – Reparameterization

• Solutions
  – Belief Propagation and related methods [Lecture 1]
  – Graph cuts [Lecture 2]
\[ Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)} \]

\[ 2 + 1 + 2 + 1 + 3 + 1 + 3 = 13 \]
MAP Estimation

\[ Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)} \]

\[ 5 + 1 + 4 + 0 + 6 + 4 + 7 = 27 \]
MAP Estimation

\[ Q(f; \theta) = \sum_a \theta_a; f(a) + \sum_{(a,b)} \theta_{ab}; f(a)f(b) \]

\[ f^* = \arg \min Q(f; \theta) \]

Equivalent to maximizing the associated probability
**MAP Estimation**

16 possible labellings

\[
f^* = \{1, 0, 0, 1\}
\]
\[
q^* = 13
\]

<table>
<thead>
<tr>
<th>f(a)</th>
<th>f(b)</th>
<th>f(c)</th>
<th>f(d)</th>
<th>Q(f; \theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f(a)</th>
<th>f(b)</th>
<th>f(c)</th>
<th>f(d)</th>
<th>Q(f; \theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>
Computational Complexity

Segmentation

$2^{|V|}$

$|V| = \text{number of pixels} \approx 153600$

Can we do better than brute-force?

MAP Estimation is NP-hard!!
MAP Inference / Energy Minimization

- Computing the assignment minimizing the energy in NP-hard in general
  \[
  \text{argmin}_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}; \mathbf{x}) = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} | \mathbf{x})
  \]

- Exact inference is possible in some cases, e.g.,
  - Low treewidth graphs \(\rightarrow\) message-passing
  - Submodular potentials \(\rightarrow\) graph cuts

- Efficient approximate inference algorithms exist
  - Message passing on general graphs
  - Move-making algorithms
  - Relaxation algorithms
Overview

• Basics: problem formulation
  – Energy Function
  – MAP Estimation
  – Computing min-marginals
  – Reparameterization

• Solutions
  – Belief Propagation and related methods [Lecture 1]
  – Graph cuts [Lecture 2]
Min-Marginals

\[ f^* = \text{arg min } Q(f; \theta) \text{ such that } f(a) = i \]

Min-marginal \( q_{a;i} \)

Not a marginal (no summation)
Min-Marginals

16 possible labellings

\[ q_{a;0} = 15 \]

<table>
<thead>
<tr>
<th>f(a)</th>
<th>f(b)</th>
<th>f(c)</th>
<th>f(d)</th>
<th>Q(f; \theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f(a)</th>
<th>f(b)</th>
<th>f(c)</th>
<th>f(d)</th>
<th>Q(f; \theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>
Min-Marginals

16 possible labellings

\[ q_{a;1} = 13 \]

<table>
<thead>
<tr>
<th>f(a)</th>
<th>f(b)</th>
<th>f(c)</th>
<th>f(d)</th>
<th>Q(f; \theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f(a)</th>
<th>f(b)</th>
<th>f(c)</th>
<th>f(d)</th>
<th>Q(f; \theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>
Min-Marginals and MAP

- Minimum min-marginal of any variable = energy of MAP labelling

\[ \min_i q_{a;i} \]

\[ \min_i \left( \min_f Q(f; \theta) \enspace \text{such that } f(a) = i \right) \]

\( V_a \) has to take one label

\[ \min_f Q(f; \theta) \]
Summary

Energy Function

$$Q(f; \theta) = \sum_a \theta_a;f(a) + \sum_{(a,b)} \theta_{ab};f(a)f(b)$$

MAP Estimation

$$f^* = \arg\min Q(f; \theta)$$

Min-marginals

$$q_{a;i} = \min Q(f; \theta) \quad \text{s.t.} \quad f(a) = i$$
Overview

• Basics: problem formulation
  – Energy Function
  – MAP Estimation
  – Computing min-marginals
  – Reparameterization

• Solutions
  – Belief Propagation and related methods [Lecture 1]
  – Graph cuts [Lecture 2]
Reparameterization

Add a constant to all $\theta_{a;i}$
Subtract that constant from all $\theta_{b;k}$

$$Q(f; \theta') = Q(f; \theta)$$
Reparameterization

Add a constant to one $\theta_{b;k}$

Subtract that constant from $\theta_{ab;ik}$ for all ‘i’

$$Q(f; \theta') = Q(f; \theta)$$
Reparameterization

$\theta'$ is a reparameterization of $\theta$, iff

$$Q(f; \theta') = Q(f; \theta), \text{ for all } f \quad \theta' \equiv \theta$$

Equivalently

$$\theta'_{a;i} = \theta_{a;i} + M_{ba;i}$$

$$\theta'_{b;k} = \theta_{b;k} + M_{ab;k}$$

$$\theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k} - M_{ba;i}$$

Kolmogorov, PAMI, 2006
Recap

MAP Estimation

\[ f^* = \arg \min Q(f; \theta) \]

\[ Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)} \]

Min-marginals

\[ q_{a;i} = \min Q(f; \theta) \text{ s.t. } f(a) = i \]

Reparameterization

\[ Q(f; \theta') = Q(f; \theta), \text{ for all } f \]

\[ \theta' \equiv \theta \]
Overview

• Basics: problem formulation
  – Energy Function
  – MAP Estimation
  – Computing min-marginals
  – Reparameterization

• Solutions
  – Belief Propagation and related methods [Lecture 1]
  – Graph cuts [Lecture 2]
Belief Propagation

• Remember, some MAP problems are easy

• Belief Propagation gives exact MAP for chains

• Exact MAP for trees

• Clever Reparameterization
Two Variables

Add a constant to one $\theta_{b;k}$

Subtract that constant from $\theta_{ab;ik}$ for all ‘i’

Choose the right constant $\theta'_{b;k} = q_{b;k}$
Choose the right constant $\theta'_{b;k} = q_{b;k}$
Choose the *right* constant

\[ \theta'_{b;k} = q_{b;k} \]
Choose the right constant \( \theta'_{b;k} = q_{b;k} \)

Potentials along the red path add up to 0

\[ f(a) = 1 \]
Choose the right constant \( \theta'_{b;k} = q_{b;k} \).
Two Variables

\[ f(a) = 1 \]

\[ \theta'_b;0 = q_b;0 \]

\[ \theta'_b;1 = q_b;1 \]

Minimum of min-marginals = MAP estimate

Choose the \textit{right} constant \[ \theta'_b;k = q_b;k \]
Choose the right constant $\theta'_{b;k} = q_{b;k}$.

Two Variables

\[ f(a) = 1 \]

\[ \theta'_{b;0} = q_{b;0} \]

\[ f^*(b) = 0 \quad f^*(a) = 1 \]
Choose the right constant $\theta'_{b;k} = q_{b;k}$
Recap

We only need to know two sets of equations

General form of Reparameterization

\[
\begin{align*}
\theta'_{a;i} &= \theta_{a;i} + M_{ba;i} \\
\theta'_{b;k} &= \theta_{b;k} + M_{ab;k} \\
\theta'_{ab;ik} &= \theta_{ab;ik} - M_{ab;k} - M_{ba;i}
\end{align*}
\]

Reparameterization of (a,b) in Belief Propagation

\[
\begin{align*}
M_{ab;k} &= \min_i \left\{ \theta_{a;i} + \theta_{ab;ik} \right\} \\
M_{ba;i} &= 0
\end{align*}
\]
Reparameterize the edge \((a,b)\) as before
Reparameterize the edge \((a,b)\) as before

Potentials along the red path add up to 0
Reparameterize the edge (b,c) as before

Potentials along the red path add up to 0
Generalizes to any length chain
Three Variables

\[ f(a) = 1 \quad f(b) = 1 \]

\[ f^*(c) = 0 \quad f^*(b) = 0 \quad f^*(a) = 1 \]

Only Dynamic Programming
Why Dynamic Programming?

3 variables $\equiv 2$ variables + book-keeping
n variables $\equiv (n-1)$ variables + book-keeping

Start from left, go to right

Reparameterize current edge (a,b)

$$M_{ab;k} = \min_i \{ \theta_{a;i} + \theta_{ab;ik} \}$$

$$\theta'_{b;k} = \theta_{b;k} + M_{ab;k} \quad \theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k}$$

Repeat
Why Dynamic Programming?

Start from left, go to right

Reparameterize current edge (a,b)

\[ M_{ab;k} = \min_i \{ \theta_{a;i} + \theta_{ab;ik} \} \]

\[ \theta'_{b;k} = \theta_{b;k} + M_{ab;k} \]

\[ \theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k} \]

Repeat

Why stop at dynamic programming?
Reparameterize the edge \((c,b)\) as before

\[ \theta'_{b;i} = q_{b;i} \]
Reparameterize the edge \((b,a)\) as before

\[
\theta'_{a;i} = q_{a;i}
\]
Three Variables

All min-marginals are computed
Reparameterize the edge \((1,2)\)
Reparameterize the edge (1,2)
Reparameterize the edge (2,3)
Chains

Reparameterize the edge \((n-1,n)\)

Min-marginals \(e_n(i)\) for all labels
Belief Propagation on Chains

Start from left, go to right

Reparameterize current edge \((a,b)\)

\[
M_{ab;k} = \min_i \{ \theta_{a;i} + \theta_{ab;ik} \}
\]

\[
\theta'_{b;k} = \theta_{b;k} + M_{ab;k} \quad \theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k}
\]

Repeat till the end of the chain

Start from right, go to left

Repeat till the end of the chain
Belief Propagation on Chains

• Generalizes to chains of any length
• A way of computing reparam constants

• Forward Pass - Start to End
  • MAP estimate
  • Min-marginals of final variable

• Backward Pass - End to start
  • All other min-marginals
Computational Complexity

Number of reparameterization constants = (n-1)h

Complexity for each constant = O(h)

Total complexity = O(nh^2)

Better than brute-force O(h^n)
Reparameterize the edge (4,2)
Reparameterize the edge \((3, 1)\)

Min-marginals \(e_1(i)\) for all labels
Start from leaves and move towards root

Pick the minimum of min-marginals

Backtrack to find the best labeling $x$
Computational Complexity

Number of reparameterization constants = (n-1)h

Complexity for each constant = \(O(h)\)

Total complexity = \(O(nh^2)\)

Better than brute-force \(O(h^n)\)
Belief Propagation on Cycles

Where do we start? Arbitrarily
Reparameterize (a,b)
Belief Propagation on Cycles

Potentials along the red path add up to 0
Belief Propagation on Cycles

Potentials along the red path add up to 0
Belief Propagation on Cycles

Potentials along the red path add up to 0
Belief Propagation on Cycles

Potentials along the red path add up to 0
Belief Propagation on Cycles

\[ \theta'_{a;0} - \theta_{a;0} = q_{a;0} \]
\[ \theta'_{a;1} - \theta_{a;1} = q_{a;1} \]

Potentials along the red path add up to 0
Belief Propagation on Cycles

\[ \theta'_{a;1} - \theta_{a;1} \]
\[ \theta'_{a;0} - \theta_{a;0} = q_{a;0} \]
\[ \theta'_{a;1} - \theta_{a;1} = q_{a;1} \]

Pick minimum min-marginal. Follow red path.
Belief Propagation on Cycles

Potentials along the red path add up to 0
Belief Propagation on Cycles

Potentials along the red path add up to 0
Belief Propagation on Cycles

Potentials along the red path add up to 0
Belief Propagation on Cycles

\[ \theta'_{a;1} - \theta_{a;1} = q_{a;1} \leq \theta'_{a;0} - \theta_{a;0} = q_{a;0} \]

Potentials along the red path add up to 0
Belief Propagation on Cycles

\[ \theta'_{a;1} - \theta_{a;1} = q_{a;1} \leq \theta'_{a;0} - \theta_{a;0} = q_{a;0} \]

Problem Solved
Belief Propagation on Cycles

\[ \theta_{a;1}' - \theta_{a;1} = q_{a;1} \geq \theta_{a;0}' - \theta_{a;0} = q_{a;0} \]

Problem Not Solved
Belief Propagation on Cycles

Reparameterize $(a,b)$ again
Belief Propagation on Cycles

Reparameterize \((a, b)\) again

But doesn’t this overcount some potentials?
Belief Propagation on Cycles

Reparameterize \((a,b)\) again

Yes. But we will do it anyway
Belief Propagation on Cycles

Keep reparameterizing edges in some order
Hope for convergence and a good solution
Belief Propagation

- Generalizes to any arbitrary random field

- Complexity per iteration \( O(|E||L|^2) \)

- Memory required \( O(|E||L|) \)
Computational Issues of BP

Complexity per iteration \( O(|E||L|^2) \)

Special Pairwise Potentials \( \theta_{ab;ik} = w_{ab} d(|i-k|) \)

- Potts
- Truncated Linear
- Truncated Quadratic

Felzenszwalb & Huttenlocher, 2004
Computational Issues of BP

Memory requirements $O(|E||L|)$

Half of original BP Kolmogorov, 2006

Some approximations exist

Yu, Lin, Super and Tan, 2007
Lasserre, Kannan and Winn, 2007

But memory still remains an issue
Computational Issues of BP

Order of reparameterization

Randomly

In some fixed order

The one that results in maximum change

Residual Belief Propagation

Elidan et al., 2006
Summary of BP

Exact for chains

Exact for trees

Approximate MAP for general cases

Not even convergence guaranteed

So can we do something better?
Other alternatives

- Integer linear programming and relaxation

- TRW, Dual decomposition methods

- Extensively studied
  - Schlesinger, 1976
  - Koster et al., 1998, Chekuri et al., ’01, Archer et al., ’04
  - Wainwright et al., 2001, Kolmogorov, 2006
  - Globerson and Jaakkola, 2007, Komodakis et al., 2007
  - Kumar et al., 2007, Sontag et al., 2008, Werner, 2008
  - Batra et al., 2011, Werner, 2011, Zivny et al., 2014
Where do we stand?

Chain/Tree, 2-label: Use BP

Chain/Tree, multi-label: Use BP

Grid graph: Use TRW, dual decomposition, relaxation
Note on Dynamic Programming
Dynamic Programming (DP)

- DP ≈ “careful brute force”
- DP ≈ recursion + memoization + guessing
- Divide the problem into subproblems that are connected to the original problem
- Graph of subproblems has to be acyclic (DAG)
- Time = #subproblems · time/subproblem
5 easy steps of DP

1. Define subproblems
2. Guess part of solution
3. Relate subproblems (recursion)
4. Recurse + memoize
   OR build DP table bottom-up
   - check subprobs be acyclic / topological order
5. Solve original problem

Analysis:

- #subproblems
- #choices
- time/subproblem
- time
- extra time
## 5 easy steps of DP

<table>
<thead>
<tr>
<th>Step</th>
<th><strong>Fibonacci</strong></th>
<th><strong>Shortest paths</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Subproblems</td>
<td>$F_k, 1 \leq k \leq n$</td>
<td>$\delta_k(s, v), \ v \in V, \ 0 \leq k \leq V$</td>
</tr>
<tr>
<td>#subproblems</td>
<td>$n$</td>
<td>$V^2$</td>
</tr>
<tr>
<td>2. Guessing</td>
<td>$F_{n-1}, F_{n-2}$</td>
<td>edges coming into $v$</td>
</tr>
<tr>
<td>#choices</td>
<td>1</td>
<td>$\text{indegree}(v)$</td>
</tr>
<tr>
<td>3. Recurrence</td>
<td>$F_n = F_{n-1} + F_{n-2}$</td>
<td>$\delta_k(s, v) = \min{ \delta_{k-1}(s, u) + w(u, v) \mid (u, v) \in E }$</td>
</tr>
<tr>
<td>time/subproblem</td>
<td>$O(1)$</td>
<td>$O(\text{indegree}(v))$</td>
</tr>
<tr>
<td>4. Topological order</td>
<td>for $k = 1, ..., n$</td>
<td>for $k = 0, 1, ..., V - 1$ for $v \in V$</td>
</tr>
<tr>
<td>total time</td>
<td>$O(n)$</td>
<td>$O(V E)$</td>
</tr>
<tr>
<td>5. Original problem</td>
<td>$F_n$</td>
<td>$\delta_{V-1}(s, v)$</td>
</tr>
<tr>
<td>extra time</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>