Machine Learning - Lecture 14

MRF Applications & Graph Cuts

28.06.2012

Bastian Leibe
RWTH Aachen
http://www.mmp.rwth-aachen.de
leibe@umic.rwth-aachen.de
Announcements

• Exam dates
  - We’ve sent around a Doodle poll with several choices
  - July 26/27
  - August 02/03
  - August 22-24
  - Please enter your preference on the Doodle poll
    - First choice in green
    - All other possible dates in yellow
  - If none of the dates fits you, send us an email.
Course Outline

• Fundamentals (2 weeks)
  - Bayes Decision Theory
  - Probability Density Estimation

• Discriminative Approaches (5 weeks)
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees

• Generative Models (4 weeks)
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference
Topics of This Lecture

• Recap: Exact inference
  - Sum-Product algorithm
  - Max-Sum algorithm
  - Junction Tree algorithm

• Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials

• Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
Recap: Factor Graphs

- Joint probability
  - Can be expressed as product of factors: \( p(x) = \frac{1}{Z} \prod_s f_s(x_s) \)
  - Factor graphs make this explicit through separate factor nodes.

- Converting a directed polytree
  - Conversion to undirected tree creates loops due to moralization!
  - Conversion to a factor graph again results in a tree!

Image source: C. Bishop, 2006
Recap: Sum-Product Algorithm

- **Objectives**
  - Efficient, exact inference algorithm for finding marginals.

- **Procedure:**
  - Pick an arbitrary node as root.
  - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  - Compute the product of received messages at each node for which the marginal is required, and normalize if necessary.

\[
p(x) \propto \prod_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x)
\]

- **Computational effort**
  - Total number of messages = \(2 \cdot \text{number of graph edges}\).
Recap: Sum-Product Algorithm

- Two kinds of messages
  - Message from factor node to variable nodes:
    - **Sum** of factor contributions
    \[ \mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} F_s(x, X_s) \]
    \[ = \sum_{X_s} f_s(x_s) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m) \]
  - Message from variable node to factor node:
    - **Product** of incoming messages
    \[ \mu_{x_m \rightarrow f_s}(x_m) \equiv \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m) \]

⇒ Simple propagation scheme.
Recap: Sum-Product from Leaves to Root

Message definitions:

\[ \mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} f_s(x_s) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m) \]

\[ \mu_{x_m \rightarrow f_s}(x_m) \equiv \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m) \]

\[ \mu_{x \rightarrow f}(x) = 1 \]

\[ \mu_{f \rightarrow x}(x) = f(x) \]

Image source: C. Bishop, 2006
Recap: Sum-Product from Root to Leaves

Message definitions:

\[
\mu_{f_s \rightarrow x}(x) = \sum_{X_s} f_s(x_s) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m)
\]

\[
\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)
\]

\[
\mu_{x \rightarrow f}(x) = 1 \quad \mu_{f \rightarrow x}(x) = f(x)
\]
Recap: Max-Sum Algorithm

• **Objective:** an efficient algorithm for finding
  - Value $x^{\text{max}}$ that maximises $p(x)$;
  - Value of $p(x^{\text{max}})$.
  $\Rightarrow$ Application of dynamic programming in graphical models.

• **Key ideas**
  - We are interested in the maximum value of the joint distribution
    $$ p(x^{\text{max}}) = \max_x p(x) $$
    $\Rightarrow$ Maximize the product $p(x)$.
  - For numerical reasons, use the logarithm.
    $$ \ln \left( \max_x p(x) \right) = \max_x \ln p(x). $$
    $\Rightarrow$ Maximize the sum (of log-probabilities).
Recap: Max-Sum Algorithm

• Initialization (leaf nodes)

\[ \mu_{x \rightarrow f}(x) = 0 \quad \mu_{f \rightarrow x}(x) = \ln f(x) \]

• Recursion

  ➢ Messages

\[ \mu_{f \rightarrow x}(x) = \max_{x_1, \ldots, x_M} \left[ \ln f(x, x_1, \ldots, x_M) + \sum_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f}(x_m) \right] \]

\[ \mu_{x \rightarrow f}(x) = \sum_{l \in \text{ne}(x) \setminus f} \mu_{f_l \rightarrow x}(x) \]

  ➢ For each node, keep a record of which values of the variables gave rise to the maximum state:

\[ \phi(x) = \arg\max_{x_1, \ldots, x_M} \left[ \ln f(x, x_1, \ldots, x_M) + \sum_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f}(x_m) \right] \]
Recap: Max-Sum Algorithm

- Termination (root node)
  - Score of maximal configuration
    \[ p_{\text{max}} = \max_x \left[ \sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right] \]
  - Value of root node variable giving rise to that maximum
    \[ x_{\text{max}} = \arg \max_x \left[ \sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right] \]
  - Back-track to get the remaining variable values
    \[ x_{n-1}^{\text{max}} = \phi(x_n^{\text{max}}) \]

Slide adapted from Chris Bishop
Junction Tree Algorithm

• Motivation
  - Exact inference on general graphs.
  - Works by turning the initial graph into a junction tree and then running a sum-product-like algorithm.
  - Intractable on graphs with large cliques.

• Main steps
  1. If starting from directed graph, first convert it to an undirected graph by moralization.
  2. Introduce additional links by triangulation in order to reduce the size of cycles.
  3. Find cliques of the moralized, triangulated graph.
  4. Construct a new graph from the maximal cliques.
  5. Remove minimal links to break cycles and get a junction tree.
     ⇒ Apply regular message passing to perform inference.
Junction Tree Algorithm

- Starting from an undirected graph...
1. Convert to an undirected graph through moralization.
   - Marry the parents of each node.
   - Remove edge directions.
Junction Tree Algorithm

2. Triangulate
   - Such that there is no loop of length > 3 without a chord.
   - This is necessary so that the final junction tree satisfies the “running intersection” property (explained later).
Junction Tree Algorithm

3. Find cliques of the moralized, triangulated graph.
4. Construct a new junction graph from maximal cliques.

- Create a node from each clique.
- Each link carries a list of all variables in the intersection.
  - Drawn in a “separator” box.
5. **Remove links to break cycles** $\Rightarrow$ **junction tree**.
   - For each cycle, remove the link(s) with the minimal number of shared nodes until all cycles are broken.
   - Result is a maximal spanning tree, the **junction tree**.
Junction Tree - Properties

- **Running intersection property**
  - “If a variable appears in more than one clique, it also appears in all intermediate cliques in the tree”.
  - This ensures that neighboring cliques have consistent probability distributions.
  - Local consistency $\rightarrow$ global consistency
Junction Tree: Example 1

- **Algorithm**
  1. Moralization
  2. Triangulation (not necessary here)
Junction Tree: Example 1

- **Algorithm**
  1. Moralization
  2. Triangulation (not necessary here)
  3. Find cliques
  4. Construct junction graph

(b) Moral graph

(c) Junction graph

Image source: J. Pearl, 1988
Junction Tree: Example 1

- **Algorithm**
  1. Moralization
  2. Triangulation (not necessary here)
  3. Find cliques
  4. Construct junction graph
  5. Break links to get junction tree

(c) Junction graph

(d) Junction tree

Image source: J. Pearl, 1988
Junction Tree: Example 2

- Without triangulation step
  - The final graph will contain cycles that we cannot break without losing the running intersection property!

Image source: J. Pearl, 1988
Junction Tree: Example 2

- When applying the triangulation
  - Only small cycles remain that are easy to break.
  - Running intersection property is maintained.

Image source: J. Pearl, 1988
Junction Tree Algorithm

- **Good news**
  - The junction tree algorithm is efficient in the sense that for a given graph there does not exist a computationally cheaper approach.

- **Bad news**
  - This may still be too costly.
  - Effort determined by number of variables in the largest clique.
  - Grows exponentially with this number (for discrete variables).
  - \( \Rightarrow \) Algorithm becomes impractical if the graph contains large cliques!
Loopy Belief Propagation

- Alternative algorithm for loopy graphs
  - Sum-Product on general graphs.
  - Strategy: simply ignore the problem.
  - Initial unit messages passed across all links, after which messages are passed around until convergence
    - Convergence is not guaranteed!
    - Typically break off after fixed number of iterations.
  - Approximate but tractable for large graphs.
  - Sometime works well, sometimes not at all.
Topics of This Lecture

- Recap: Exact inference
  - Sum-Product algorithm
  - Max-Sum algorithm
  - Junction Tree algorithm

- Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials

- Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
Markov Random Fields (MRFs)

• What we’ve learned so far...
  ➢ We know they are undirected graphical models.
  ➢ Their joint probability factorizes into clique potentials,
    \[ p(x) = \frac{1}{Z} \prod_C \psi_C(x_C) \]
    which are conveniently expressed as energy functions.
    \[ \psi_C(x_C) = \exp\{-E(x_C)\} \]
  ➢ We know how to perform inference for them.
    - Sum/Max-Product BP for exact inference in tree-shaped MRFs.
    - Loopy BP for approximate inference in arbitrary MRFs.
    - Junction Tree algorithm for converting arbitrary MRFs into trees.

• But what are they actually good for?
  ➢ And how do we apply them in practice?
Markov Random Fields

- Allow rich probabilistic models.
  - But built in a local, modular way.
  - Learn local effects, get global effects out.
- Very powerful when applied to regular structures.
  - Such as images...

Observed evidence

Hidden “true states”

Neighborhood relations
Applications of MRFs

• Movie “No Way Out” (1987)
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising

Results by [Roth & Black, CVPR’05]
Applications of MRFs

• Many applications for low-level vision tasks
  ➢ Image denoising

Observation process

"True" image content

Noisy observations

“Smoothness constraints”

Results by [Roth & Black, CVPR’05]
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting

Since 1699, when French explorers landed at the great bend of the Mississippi River and celebrated the first Mardi Gras in North America, New Orleans has brewed a fascinating melange of cultures. It was French, then Spanish, then French again, then sold to the United States. Through all these years, and even into the 1900s, others arrived from everywhere: Acadians (Cajuns), Africans, indige-
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration

Results by [Roth & Black, CVPR’05]
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Image upscaling

*Convert a low-res image into a high-res image!*

Image source: [Freeman et al., CG&A'03]
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution

Image source: [Freeman et al., CG&A’03]
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow
  - Stereo depth estimation

Stereo image pair

Disparity map
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow
  - Stereo depth estimation

- MRFs have become a standard tool for such tasks.
  - Let’s look at how they are applied in detail...
MRF Structure for Images

- Basic structure

  ![MRF Structure Diagram]

  - Two components
    - Observation model
      - How likely is it that node $x_i$ has label $L_i$ given observation $y_i$?
      - This relationship is usually learned from training data.
    - Neighborhood relations
      - Simplest case: 4-neighborhood
      - Serve as smoothing terms.
      - Discourage neighboring pixels to have different labels.
      - This can either be learned or be set to fixed “penalties”.

  Noisy observations

  “True” image content
MRF Nodes as Pixels

Original image

Degraded image

Reconstruction from MRF modeling pixel neighborhood statistics

These neighborhood statistics can be learned from training data!

Slide adapted from William Freeman
Simple Binary Image Denoising Model

- **MRF Structure**

  ![Diagram of MRF structure with nodes and edges]

  **Observation process**

  **“Smoothness constraints”**

  **Prior**

  **Smoothness**

  **Observation**

  Example: simple energy function

  \[
  E(x, y) = h \sum_i x_i + \beta \sum_{\{i,j\}} \delta(x_i \neq x_j) - \eta \sum_i x_i y_i
  \]

  - Smoothness term: fixed penalty \( \beta \) if neighboring labels disagree.
  - Observation term: fixed penalty \( \eta \) if label and observation disagree.

  \( x_i, y_i \in \{-1, 1\} \)

  \( x_i, y_i \) are the true and noisy labels, respectively.
MRF Nodes as Patches

More general relationships expressed by potential functions \( \Phi \) and \( \Psi \).

Slide credit: William Freeman
Network Joint Probability

- Interpretation of the factorized joint probability

\[ P(x, y) = \prod_{i} \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j) \]

- Scene
- Image
- Image-scene compatibility function
- Local observations
- Scene-scene compatibility function
- Neighboring scene nodes

Slide credit: William Freeman
Energy Formulation

• Energy function

\[ E(x, y) = \sum_{i} \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j) \]

Single-node potentials \( \phi \) and Pairwise potentials \( \psi \)

• Single-node (unary) potentials \( \phi \)
  - Encode local information about the given pixel/patch.
  - How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?

• Pairwise potentials \( \psi \)
  - Encode neighborhood information.
  - How different is a pixel/patch’s label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)
How to Set the Potentials? Some Examples

- **Unary potentials**
  - E.g. color model, modeled with a Mixture of Gaussians
    \[
    \phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k)p(k|x_i)N(y_i; \bar{y}_k, \Sigma_k)
    \]
  
  \[\Rightarrow \text{Learn color distributions for each label}\]
How to Set the Potentials? Some Examples

- **Pairwise potentials**
  - Potts Model
    \[ \psi(x_i, x_j; \theta_\psi) = \theta_\psi \delta(x_i \neq x_j) \]
    - Simplest discontinuity preserving model.
    - Discontinuities between any pair of labels are penalized equally.
    - Useful when labels are unordered or number of labels is small.
  - Extension: “contrast sensitive Potts model”
    \[ \psi(x_i, x_j, g_{ij}(y); \theta_\psi) = \theta_\psi g_{ij}(y) \delta(x_i \neq x_j) \]
    where
    \[ g_{ij}(y) = e^{-\beta \|y_i - y_j\|^2} \]
    \[ \beta = 2 \cdot \text{avg} \left( \|y_i - y_j\|^2 \right) \]
    - Discourages label changes except in places where there is also a large change in the observations.
Example: MRF for Image Segmentation

- **Extended MRF structure**

  Pairwise potential
  \[ \phi(D| x_i, x_j) \]

  Unary potential
  \[ \phi(D| x_i) \]

- **Energy formulation**

  \[
  E(x) = \sum_{i \in S} \left( \phi(D| x_i) + \sum_{j \in N_i} \left( \phi(D| x_i, x_j) + \psi(x_i, x_j) \right) \right) + \text{const}
  \]

  - Unary likelihood
  - Contrast Term
  - Uniform Prior (Potts Model)

Slide credit: Phil Torr
Example: MRF for Image Segmentation

- MRF structure

Unary potential
\[ \phi(D | x_i) \]

Pairwise potential
\[ \phi(D | x_i, x_j) \]

- Pixels
- Labels
- Prior Potts model

Data (D)
Unary likelihood
Pair-wise Terms
MAP Solution

Slide credit: Phil Torr
Energy Minimization

• Goal:
  - Infer the optimal labeling of the MRF.

• Many inference algorithms are available, e.g.
  - Simulated annealing
  - Iterated conditional modes (ICM)
  - Belief propagation
  - Graph cuts
  - Variational methods
  - Monte Carlo sampling

• Recently, Graph Cuts have become a popular tool
  - Only suitable for a certain class of energy functions.
  - But the solution can be obtained very fast for typical vision problems (~1MPixel/sec).

What you saw in the movie.
Too simple.
Last lecture
Today!
For more complex problems
Topics of This Lecture

• Recap: Exact inference
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm

• Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials

• Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
Graph Cuts for Binary Problems

- Idea: convert MRF into source-sink graph

Minimum cost cut can be computed in polynomial time (max-flow/min-cut algorithms)

\[ w_{pq} = \exp \left\{ - \frac{\Delta I_{pq}}{2\sigma^2} \right\} \]

Slide credit: Yuri Boykov

[Boykov & Jolly, ICCV’01]
Simple Example of Energy

\[ E(L) = \sum_p D_p(L_p) + \sum_{pq \in N} w_{pq} \cdot \delta(L_p \neq L_q) \]

- **Unary potentials**: \( D_p(L_p) \)
- **Pairwise potentials**: \( w_{pq} \cdot \delta(L_p \neq L_q) \)

- \( \delta \) is the Kronecker delta function.
- \( t \)-links correspond to unary potentials, \( n \)-links to pairwise potentials.

### Graph Representation

- \( D_p(t) \) and \( D_p(s) \) are nodes.
- A cut partitions the graph.

### Energy Function

- \( w_{pq} = \exp \left\{ -\frac{\Delta I_{pq}}{2\sigma^2} \right\} \)

### Parameters

- \( \Delta I_{pq} \) measures the difference in intensity between \( p \) and \( q \).
- \( \sigma \) controls the steepness of the exponential function.

### Object Segmentation

- \( L_p \in \{ s, t \} \)
- \( s \) and \( t \) represent different objects.

**Slide credit:** Yuri Boykov
Adding Regional Properties

Regional bias example

Suppose \( I^s \) and \( I^t \) are given “expected” intensities of object and background

\[
D_p(s) \propto \exp \left( -\| I_p - I^s \|^2 / 2\sigma^2 \right)
\]

\[
D_p(t) \propto \exp \left( -\| I_p - I^t \|^2 / 2\sigma^2 \right)
\]

NOTE: hard constrains are not required, in general.

[Boykov & Jolly, ICCV’01]

Slide credit: Yuri Boykov
Adding Regional Properties

“expected” intensities of object and background $I^s$ and $I^t$ can be re-estimated

EM-style optimization

$D_p(s) \propto \exp \left( -\| I_p - I^s \|^2 / 2\sigma^2 \right)$

$D_p(t) \propto \exp \left( -\| I_p - I^t \|^2 / 2\sigma^2 \right)$
Adding Regional Properties

- More generally, unary potentials can be based on any intensity/color models of object and background.

\[ D_p(L_p) = - \log p(I_p|L_p) \]

Object and background color distributions

Slide credit: Yuri Boykov

[Boykov & Jolly, ICCV’01]
Topics of This Lecture

• Recap: Exact inference
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm

• Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials

• Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
How Does it Work? The s-t-Mincut Problem

Graph (V, E, C)

Vertices V = \{v_1, v_2 \ldots v_n\}
Edges E = \{(v_1, v_2) \ldots\}
Costs C = \{c_{(1,2)} \ldots\}
The s-t-Mincut Problem

What is an st-cut?
An st-cut (S,T) divides the nodes between source and sink.

What is the cost of a st-cut?
Sum of cost of all edges going from S to T

5 + 2 + 9 = 16
The s-t-Mincut Problem

What is an st-cut?
An st-cut \((S, T)\) divides the nodes between source and sink.

What is the cost of a st-cut?
Sum of cost of all edges going from \(S\) to \(T\)

What is the st-mincut?
st-cut with the minimum cost

2 + 1 + 4 = 7

Slide credit: Pushmeet Kohli
How to Compute the s-t-Mincut?

Solve the dual maximum flow problem

Compute the maximum flow between Source and Sink

Constraints
Edges: Flow < Capacity
Nodes: Flow in = Flow out

Min-cut/Max-flow Theorem
In every network, the maximum flow equals the cost of the st-mincut

Slide credit: Pushmeet Kohli
## History of Maxflow Algorithms

### Augmenting Path and Push-Relabel

<table>
<thead>
<tr>
<th>Year</th>
<th>Discoverer(s)</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>Dantzig</td>
<td>$O(n^2 mU)$</td>
</tr>
<tr>
<td>1955</td>
<td>Ford &amp; Fulkerson</td>
<td>$O(m^2 U)$</td>
</tr>
<tr>
<td>1970</td>
<td>Dinitz</td>
<td>$O(n^2 m)$</td>
</tr>
<tr>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>$O(m^2 \log U)$</td>
</tr>
<tr>
<td>1973</td>
<td>Dinitz</td>
<td>$O(nm \log U)$</td>
</tr>
<tr>
<td>1974</td>
<td>Karzanov</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>1977</td>
<td>Cherkassky</td>
<td>$O(n^2 m^{1/2})$</td>
</tr>
<tr>
<td>1980</td>
<td>Galil &amp; Naamad</td>
<td>$O(nm \log^2 n)$</td>
</tr>
<tr>
<td>1983</td>
<td>Sleator &amp; Tarjan</td>
<td>$O(nm \log n)$</td>
</tr>
<tr>
<td>1986</td>
<td>Goldberg &amp; Tarjan</td>
<td>$O(nm \log(n^2/m))$</td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja &amp; Orlin</td>
<td>$O(nm + n^2 \log U)$</td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja et al.</td>
<td>$O(nm \log(n \sqrt{\log U/m}))$</td>
</tr>
<tr>
<td>1989</td>
<td>Cheriyan &amp; Hagerup</td>
<td>$E(nm + n^2 \log^2 n)$</td>
</tr>
<tr>
<td>1990</td>
<td>Cheriyan et al.</td>
<td>$O(n^3/\log n)$</td>
</tr>
<tr>
<td>1990</td>
<td>Alon</td>
<td>$O(nm + n^{8/3} \log n)$</td>
</tr>
<tr>
<td>1992</td>
<td>King et al.</td>
<td>$O(nm + n^{2+\epsilon})$</td>
</tr>
<tr>
<td>1993</td>
<td>Phillips &amp; Westbrook</td>
<td>$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$</td>
</tr>
<tr>
<td>1994</td>
<td>King et al.</td>
<td>$O(nm \log_{m/(n \log n)} n)$</td>
</tr>
<tr>
<td>1997</td>
<td>Goldberg &amp; Rao</td>
<td>$O(m^{3/2} \log(n^2/m) \log U)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(n^{2/3} m \log(n^2/m) \log U)$</td>
</tr>
</tbody>
</table>

$n$: #nodes  
$m$: #edges  
$U$: maximum edge weight

Algorithms assume non-negative edge weights

Slide credit: Andrew Goldberg
Maxflow Algorithms

Flow = 0

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Flow = 0

Source

Sink

v1

v2
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Flow = 0 + 2

Source

v₁
2 - 2
9
1
2
4
5 - 2
v₂
Sink

Algorithms assume non-negative capacity
Maxflow Algorithms

Flow = 2

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 2

Source

\[ \begin{array}{c}
0 & 9 \\
1 & v_1 & v_2 \\
3 & 2 & 4 \\
& Sink
\end{array} \]

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 2

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 2 + 4

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 6

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 6 + 1

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity
**Maxflow Algorithms**

![Diagram of a network flow problem]

- Flow = 7

**Augmenting Path Based Algorithms**

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

**Algorithms assume non-negative capacity**

Slide credit: Pushmeet Kohli

B. Leibe
Applications: Maxflow in Computer Vision

- Specialized algorithms for vision problems
  - Grid graphs
  - Low connectivity \((m \sim O(n))\)

- Dual search tree augmenting path algorithm
  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently.
  - High worst-case time complexity.
  - Empirically outperforms other algorithms on vision problems.
  - Efficient code available on the web
  [http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html](http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html)
When Can s-t Graph Cuts Be Applied?

$$E(L) = \sum_p E_p(L_p) + \sum_{pq \in N} E(L_p, L_q)$$

- **s-t graph cuts** can only globally minimize **binary energies** that are **submodular**. [Boros & Hummer, 2002, Kolmogorov & Zabih, 2004]

\[
E(L) \text{ can be minimized by } s-t \text{ graph cuts } \iff E(s, s) + E(t, t) \leq E(s, t) + E(t, s)
\]

**Submodularity** is the discrete equivalent to convexity.

- Implies that every local energy minimum is a global minimum.
  $$\Rightarrow$$ Solution will be globally optimal.
Recap: Simple Binary Image Denoising Model

- **MRF Structure**

  \[ E(x, y) = h \sum_i x_i + \beta \sum_{\{i,j\}} \delta(x_i \neq x_j) - \eta \sum_i x_i y_i \]

  - **Prior**
  - **Smoothness** ("Potts model")
  - **Observation**

- **Example: simple energy function**

  - Smoothness term: fixed penalty $\beta$ if neighboring labels disagree.
  - Observation term: fixed penalty $\eta$ if label and observation disagree.
Converting an MRF into an s-t Graph

- **Conversion:**

- **Energy:**

\[
E(x, y) = h \sum_i x_i + \beta \sum_{\{i,j\}} \delta(x_i \neq x_j) - \eta \sum_i x_i y_i
\]

- **Unary potentials** are straightforward to set.
  - How?
  - Just insert \(x_i = 1\) and \(x_i = -1\) into the unary terms above...
Converting an MRF into an s-t Graph

• Conversion:

• Energy:

  $$E(x, y) = h \sum_{i} x_i + \beta \sum_{\{i,j\}} \delta(x_i \neq x_j) - \eta \sum_{i} x_i y_i$$

  - **Unary potentials** are straightforward to set. How?
  - **Pairwise potentials** are more tricky, since we don’t know $x_i$!
    - Trick: the pairwise energy only has an influence if $x_i \neq x_j$.
    - (Only!) in this case, the cut will go through the edge $\{x_i, x_j\}$.
Topics of This Lecture

- Recap: Exact inference
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm

- Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials

- Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
Dealing with Non-Binary Cases

- Limitation to binary energies is often a nuisance.
  \[ \Rightarrow \text{E.g. binary segmentation only...} \]
- We would like to solve also multi-label problems.
  - The bad news: Problem is NP-hard with 3 or more labels!
- There exist some approximation algorithms which extend graph cuts to the multi-label case:
  - \( \alpha \)-Expansion
  - \( \alpha \beta \)-Swap
- They are no longer guaranteed to return the globally optimal result.
  - But \( \alpha \)-Expansion has a guaranteed approximation quality (2-approx) and converges in a few iterations.
**α-Expansion Move**

- **Basic idea:**
  - Break multi-way cut computation into a sequence of binary s-t cuts.

Slide credit: Yuri Boykov
**α-Expansion Algorithm**

1. Start with any initial solution
2. For each label “α” in any (e.g. random) order:
   1. Compute optimal α-expansion move (s-t graph cuts).
   2. Decline the move if there is no energy decrease.
3. Stop when no expansion move would decrease energy.
Example: Stereo Vision

Original pair of “stereo” images

Depth map

ground truth

Slide credit: Yuri Boykov
$\alpha$-Expansion Moves

- In each $\alpha$-expansion a given label “$\alpha$” grabs space from other labels.

For each move, we choose the expansion that gives the largest decrease in the energy: $\Rightarrow$ binary optimization problem.

Slide credit: Yuri Boykov
Topics of This Lecture

• Recap: Exact inference
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm

• Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials

• Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications
GraphCut Applications: “GrabCut”

- Interactive Image Segmentation [Boykov & Jolly, ICCV’01]
  - Rough region cues sufficient
  - Segmentation boundary can be extracted from edges

- Procedure
  - User marks foreground and background regions with a brush.
  - This is used to create an initial segmentation which can then be corrected by additional brush strokes.

User segmentation cues
GrabCut: Data Model

- Obtained from interactive user input
  - User marks foreground and background regions with a brush
  - Alternatively, user can specify a bounding box

Slide credit: Carsten Rother

Global optimum of the energy
Iterated Graph Cuts

Result

Color model
(Mixture of Gaussians)

Energy after each iteration

Slide credit: Carsten Rother
GrabCut: Example Results

- This is included in the newest version of MS Office!

Image source: Carsten Rother
Applications: Interactive 3D Segmentation
References and Further Reading

• A gentle introduction to Graph Cuts can be found in the following paper:

• Try the GraphCut implementation at [http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html](http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html)