Recap: 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.

Recap: Junction Tree Example

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

Recap: MRF Structure for Images

- **Basic structure**
  - *Noisy observations* "True" image content

- **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".
Recap: How to Set the Potentials?

- **Unary potentials**
  - E.g. color model, modeled with a Mixture of Gaussians
  \[
  \phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) \mathcal{N}(y_i; \mu_k, \Sigma_k)
  \]
  - Learn color distributions for each label

```
\( \phi(x_p = 1, y_p) \)
```
```
\( \phi(x_p = 0, y_p) \)
```

```
\( \phi(x_i; y_i; \mu_\phi) = \log \sum_k \theta_\phi(x_i; k) \mathcal{N}(y_i; \mu_k; \Sigma_k) \)
```

- **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
    \[ g_{ij}(y) = e^{-\|y_i - y_j\|^2/2\beta} \]
    - Discourages label changes except in places where there is also a large change in the observations.

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 (~1 MPixel/sec).

Topics of This Lecture

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

Example: Image Segmentation

\[ E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i (1 - x_j) \]
\( N = \text{number of pixels} \)

- Image \((D)\)
- \(X_i, X_j\)
- Unary Cost \((c_i)\)

Example: Image Segmentation

\[ E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i (1 - x_j) \]
\( N = \text{number of pixels} \)

- Dark (negative) Bright (positive)
Example: Image Segmentation

\[ E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i (1 - x_j) \]

Discontinuity Cost \( c_{ij} \)

\[ x^* = \arg \min E(x) \]

Global Minimum \( (x^*) \)

How to minimize \( E(x) \)?

Graph Cuts - Basic Idea

- Construct a graph such that:
  1. Any st-cut corresponds to an assignment of \( x \)
  2. The cost of the cut is equal to the energy of \( x : E(x) \)

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)

Simple Example of Energy

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

Unary potentials \( D_p(L_p) \)

Pairwise potentials \( w_{pq} \delta(L_p \neq L_q) \)

\( L_p \in \{s,t\} \)

(binary object segmentation)

Adding Regional Properties

Regional bias example

Suppose \( I \) and \( I' \) are given “expected” intensities of object and background

\[ D_p(s) = \exp \left( -\frac{|I_p - I'|^2}{2\sigma^2} \right) \]

\[ D_p(t) = \exp \left( -\frac{|I_p - I'|^2}{2\sigma^2} \right) \]

NOTE: hard constrains are not required, in general.
**Adding Regional Properties**

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

**EM-style optimization**

**Topics of This Lecture**
- Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Graph construction
  - Extension to non-binary case
  - Applications

**How Does it Work? The s-t-Mincut Problem**

**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 an st-cut?
  - Sum of cost of all edges going from source to sink.

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

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

Augmenting Path and Push-Relabel

<table>
<thead>
<tr>
<th>Year</th>
<th>Author(s)</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>Dantzig</td>
<td>O(nm)</td>
</tr>
<tr>
<td>1962</td>
<td>Ford &amp; Fulkerson</td>
<td>O(nm)</td>
</tr>
<tr>
<td>1970</td>
<td>Dinits</td>
<td>O(n^3)</td>
</tr>
<tr>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>O(nm log n)</td>
</tr>
<tr>
<td>1973</td>
<td>Dinits</td>
<td>O(n^3)</td>
</tr>
<tr>
<td>1974</td>
<td>Karp</td>
<td>O(nm log n)</td>
</tr>
<tr>
<td>1976</td>
<td>Dinits</td>
<td>O(nm log n)</td>
</tr>
<tr>
<td>1980</td>
<td>Gold &amp; Tarjan</td>
<td>O(nm log n)</td>
</tr>
<tr>
<td>1982</td>
<td>Tarjan &amp; Tarjan</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1990</td>
<td>Hao &amp; Orlin</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1991</td>
<td>Hao et al.</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1991</td>
<td>Hao &amp; Orlin</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1992</td>
<td>Cheriyan &amp; Hegde</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1993</td>
<td>Cheriyan et al.</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1994</td>
<td>Ahn</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1995</td>
<td>Ahn</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1995</td>
<td>Ahn</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1997</td>
<td>Goldberg &amp; Ruan</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>1997</td>
<td>Goldberg &amp; Ruan</td>
<td>O(n^2 log n)</td>
</tr>
</tbody>
</table>

Algorithms assume non-negative edge weights
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

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

Algorithm assume non-negative capacity

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

Algorithm assume non-negative capacity

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

Algorithm assume non-negative capacity

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

Algorithm 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

When Can s-t Graph Cuts Be Applied?

- s-t graph cuts can only globally minimize binary energies that are submodular.
- Submodularity is the discrete equivalent to convexity.

Applications: Maxflow in Computer Vision

- Specialized algorithms for vision problems
  - Grid graphs
  - Low connectivity (m ~ O(n))
- Dual search tree augmenting path algorithm
  - Finds approximate shortest augmenting paths efficiently.
  - High worst-case time complexity.
  - Empirically outperforms other algorithms on vision problems.
  - Efficient code available on the web

Topics of This Lecture

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

Example: Graph Construction

\[ E(s, t) \]

\[ E(a_1, a_2) \]

Source (i)

\[ \alpha_1 \]

\[ \alpha_2 \]

Sink (j)
Example: Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5a_1 + 9a_2 + 4a_2 \]

Example: Graph Construction

\[ E'(a_1, a_2) = 2a_1 + 5a_1 + 9a_2 + 4a_2 + a_1 \bar{a}_2 + 2\bar{a}_1 a_2 \]
Example: Graph Construction
\[ E(a_1, a_2) = 2a_1 + 5a_1 + 9a_2 + 4a_2 + a_1 a_2 + 2a_1 a_2 \]

**How Does the Code Look Like?**

```c
// is the label of pixel p (0 or 1)
G.label_p = g->nodeID(p);
// Add a node to the graph */
G.nodeID(p) = G->add_node(G);
// Set cost of terminal edges */
set_weights(G.nodeID(p), fgCost(p), bgCost(p));

for all adjacent pixels p,q
    add_weights(G.nodeID(p), G.nodeID(q), cost);
end

G.compute_maxflow();
label_p = G->is_connected_to_source(G.nodeID(p));
// is the label of pixel p (0 or 1)
```

**Cost of cut = 11**

\[ a_1 = 1 \quad a_2 = 1 \quad E(1,1) = 11 \]

**Example: Graph Construction**

\[ E(a_1, a_2) = 2a_1 + 5a_1 + 9a_2 + 4a_2 + a_1 a_2 + 2a_1 a_2 \]

**How Does the Code Look Like?**

```c
// is the label of pixel p (0 or 1)
G.label_p = g->nodeID(p);
// Add a node to the graph */
G.nodeID(p) = G->add_node(G);
// Set cost of terminal edges */
set_weights(G.nodeID(p), fgCost(p), bgCost(p));

for all adjacent pixels p,q
    add_weights(G.nodeID(p), G.nodeID(q), cost);
end

G.compute_maxflow();
label_p = G->is_connected_to_source(G.nodeID(p));
// is the label of pixel p (0 or 1)
```

**Cost of cut = 7**

\[ a_1 = 1 \quad a_2 = 0 \quad E(1,0) = 7 \]
Topics of This Lecture

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

Dealing with Non-Binary Cases

- Limitation to binary energies is often a nuisance.
  - 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.

$\alpha$-Expansion Move

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

$\alpha$-Expansion Algorithm

1. Start with any initial solution
2. For each label “$\alpha$” in any (e.g. random) order:
   1. Compute optimal $\alpha$-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

- 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
Topics of This Lecture

• 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.

GrabCut: Data Model

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

Iterated Graph Cuts

• Obtained from interactive user input

Applications: Interactive 3D Segmentation

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

Image source: Carsten Rother
References and Further Reading

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

• Try the GraphCut implementation at
  [http://pub.ist.ac.at/~vnk/software.html](http://pub.ist.ac.at/~vnk/software.html)