Machine Learning - Lecture 17

Efficient MRF Inference with Graph Cuts

07.07.2015

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de

Recap: MRF Structure for Images

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

- 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
  - \( \phi(x_i, y_i; \theta_k) = -\theta_k \log \sum_k p(k \mid x_i, \mathcal{N}(y_i \mid \bar{y}_i, \Sigma_k)) \)
  - \( \Rightarrow \) Learn color distributions for each label

- Pairwise potentials
  - Potts Model
    - \( \psi(x_i, x_j; \theta_p) = \theta_p \delta(x_i \neq x_j) \)
    - 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"
      - \( \psi(x_i, x_j, y_i, y_j; \theta_{ps}) = \theta_p \theta_{ps} \delta(x_i \neq x_j) \)
      - where,
      - \( \delta(x_i \neq x_j) = e^{-|\beta|(|y_i - y_j|)} \)
      - Discourages label changes except in places where there is also a large change in the observations.

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
  - Applications

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
Construct a graph such that:

1. Any st-cut corresponds to an assignment of $x$: $E(x)$
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)
More generally, unary potentials can be based on any intensity/color models of object and background. Suppose $\mu_1$ and $\mu_2$ are given “expected” intensities of object and background.

\[ p(I_1 | s) \propto \exp \left( - \frac{||I_1 - \mu_1||^2}{2\sigma_1^2} \right) \]

\[ p(I_2 | t) \propto \exp \left( - \frac{||I_2 - \mu_2||^2}{2\sigma_2^2} \right) \]

\[ p(I_1 | t) \propto \exp \left( - \frac{||I_1 - \mu_2||^2}{2\sigma_2^2} \right) \]

\[ p(I_2 | s) \propto \exp \left( - \frac{||I_2 - \mu_1||^2}{2\sigma_1^2} \right) \]

NOTE: hard constrains are not required, in general.

“expected” intensities of object and background $\mu_1$ and $\mu_2$ can be re-estimated.

EM-style optimization

Object and background color distributions

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

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 st-Mincut Problem

Graph (V, E, C)

Vertices $V = \{v_1, v_2, \ldots, v_n\}$

Edges $E = \{(v_i, v_j), \ldots\}$

Costs $C = \{c_{ij}, c_{ij}, \ldots\}$
The st-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

Slide credit: Pushmeet Kohli

B. Leibe

The st-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 \( T \)

2 + 1 + 4 = 7

Slide credit: Pushmeet Kohli

B. Leibe

How to Compute the st-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

B. Leibe

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>1956</td>
<td>Edsger W. Dijkstra</td>
<td>( O(V^2E) )</td>
</tr>
<tr>
<td>1957</td>
<td>Ford &amp; Fulkerson</td>
<td>( O(V^2E) )</td>
</tr>
<tr>
<td>1970</td>
<td>Dinitz</td>
<td>( O(V^2E(\log V)) )</td>
</tr>
<tr>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>( O(V^2E\log V) )</td>
</tr>
<tr>
<td>1973</td>
<td>Tarjan</td>
<td>( O(V^2E) )</td>
</tr>
<tr>
<td>1974</td>
<td>Hopcroft &amp; Tarjan</td>
<td>( O(V^2E(\log V)) )</td>
</tr>
<tr>
<td>1977</td>
<td>Ahuja et al.</td>
<td>( O(V^2E\log V) )</td>
</tr>
<tr>
<td>1985</td>
<td>Ahuja &amp; Orlin</td>
<td>( O(V^2E\log V) )</td>
</tr>
<tr>
<td>1985</td>
<td>Chartrand &amp; Neuman</td>
<td>( O(V^3) )</td>
</tr>
<tr>
<td>1988</td>
<td>Goldin &amp; Lawler</td>
<td>( O(V^3\log^2 V) )</td>
</tr>
<tr>
<td>1990</td>
<td>Ahuja et al.</td>
<td>( O(V^3\log^2 V) )</td>
</tr>
<tr>
<td>1992</td>
<td>Ahuja &amp; Orlin</td>
<td>( O(V^3\log^2 V) )</td>
</tr>
<tr>
<td>1994</td>
<td>Goldberg &amp; Tarjan</td>
<td>( O(V^3) )</td>
</tr>
<tr>
<td>1997</td>
<td>Goldberg &amp; Rabin</td>
<td>( O(V^3\log^2 V) )</td>
</tr>
</tbody>
</table>

Slide credit: Andrew Goldberg

B. Leibe

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

B. Leibe

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

B. Leibe
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**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source -&gt; v1</td>
<td>2/0</td>
</tr>
<tr>
<td>v1 -&gt; v2</td>
<td>0/9</td>
</tr>
<tr>
<td>v2 -&gt; v3</td>
<td>0/2</td>
</tr>
<tr>
<td>v3 -&gt; Sink</td>
<td>2/3</td>
</tr>
</tbody>
</table>

---

**Flow = 2**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source -&gt; v1</td>
<td>2/0</td>
</tr>
<tr>
<td>v1 -&gt; v2</td>
<td>0/2</td>
</tr>
<tr>
<td>v2 -&gt; v3</td>
<td>2/3</td>
</tr>
<tr>
<td>v3 -&gt; Sink</td>
<td>0/4</td>
</tr>
</tbody>
</table>

---

**Flow = 2 + 4**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source -&gt; v1</td>
<td>4/0</td>
</tr>
<tr>
<td>v1 -&gt; v2</td>
<td>4/5</td>
</tr>
<tr>
<td>v2 -&gt; v3</td>
<td>0/2</td>
</tr>
<tr>
<td>v3 -&gt; Sink</td>
<td>4/0</td>
</tr>
</tbody>
</table>

---

**Flow = 6**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source -&gt; v1</td>
<td>5/4</td>
</tr>
<tr>
<td>v1 -&gt; v2</td>
<td>2/0</td>
</tr>
<tr>
<td>v2 -&gt; v3</td>
<td>3/2</td>
</tr>
<tr>
<td>v3 -&gt; Sink</td>
<td>5/4</td>
</tr>
</tbody>
</table>

---

**Flow = 7**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source -&gt; v1</td>
<td>1/0</td>
</tr>
<tr>
<td>v1 -&gt; v2</td>
<td>0/2</td>
</tr>
<tr>
<td>v2 -&gt; v3</td>
<td>4/0</td>
</tr>
<tr>
<td>v3 -&gt; Sink</td>
<td>3/2</td>
</tr>
</tbody>
</table>

---

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.

\[ E(L) = \sum_{\text{t-links}} E_t(L_p) + \sum_{\text{n-links}} E(L_p, L_q), \quad L_p \in \{s, t\} \]

- Submodularity is the discrete equivalent to convexity. \( \Rightarrow \) Solution will be globally optimal.

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

\[ E(a_i, a_j) = 2a_i \]

\[ E(a_i, a_j) = 2a_i + 5(1-a_i) \]

\[ E(a_i, a_j) = 2a_i + 5(1-a_i) + 9a_j + 4(1-a_j) \]

Example: Graph Construction

\[ E(a_i, a_j) = 2a_i \]

\[ E(a_i, a_j) = 2a_i + 5(1-a_i) \]

\[ E(a_i, a_j) = 2a_i + 5(1-a_i) + 9a_j + 4(1-a_j) \]
Example: Graph Construction

\[ E(a_i, a_j) = 2a_i + 5(1-a_i) + 9a_j + 4(1-a_j) + (1-a_i)a_j + 2(1-a_j)a_i \]

Slide credit: Pushmeet Kohli

Example: Graph Construction

\[ E(a_i, a_j) = 2a_i + 5(1-a_i) + 9a_j + 4(1-a_j) + (1-a_i)a_j + 2(1-a_j)a_i \]

Slide credit: Pushmeet Kohli

Example: Graph Construction

\[ E(a_i, a_j) = 2a_i + 5(1-a_i) + 9a_j + 4(1-a_j) + (1-a_i)a_j + 2(1-a_j)a_i \]

Slide credit: Pushmeet Kohli

How Does the Code Look Like?

```c
// Graph *g;
For all pixels p
    /* Add a node to the graph */
    nodeID(p) = g->add_node();
    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));
end
For all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q), cost);
end
g->compute_maxflow();
label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```

Slide credit: Pushmeet Kohli
Graph \( g \):

For all pixels \( p \):

1. Add a node to the graph */
   nodeID\(p\) = \( g \rightarrow \text{add_node}() \);
2. Set cost of terminal edges */
   set_weights(nodeID\(p\), fgCost\(p\), bgCost\(p\));

end

for all adjacent pixels \( p,q \):

add_weights(nodeID\(p\), nodeID\(q\), cost);

end

\( g \rightarrow \text{compute_maxflow}(); \)

label\(p\) = \( g \rightarrow \text{is_connected_to_source}(\text{nodeID}\(p\)) \);

// is the label of pixel \( p \) (0 or 1)

**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 and converges in a few iterations.

**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
**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

**α-Expansion Moves**

- In each α-expansion a given label “α” grabs space from other labels

For each move, we choose the expansion that gives the largest decrease in the energy: 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

For each move, we choose the expansion that gives the largest decrease in the energy: Binary optimization problem
Iterated Graph Cuts

Result

Energy after each iteration

Color model (Mixture of Gaussians)

GrabCut: Example Results

References and Further Reading

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

- Try the Graph Cut implementation at http://pub.ist.ac.at/~vnk/software.html