Machine Learning - Lecture 15

Efficient MRF Inference with Graph Cuts

23.01.2014

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

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

Image source: J. Pearl, 1988
Recap: MRF Structure for Images

- **Basic structure**

- **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
    \[
    \phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k)p(k|x_i)\mathcal{N}(y_i; \bar{y}_k, \Sigma_k)
    \]
  
  ⇒ Learn color distributions for each label

\[
\phi(x_p = 1, y_p)
\]
\[
\phi(x_p = 0, y_p)
\]
Recap: How to Set the Potentials?

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

B. Leibe
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).
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

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

\( E: \{0,1\}^N \rightarrow \mathbb{R} \)

\[ 0 \rightarrow \text{fg} \]

\[ 1 \rightarrow \text{bg} \]

\( N = \text{number of pixels} \)
Example: Image Segmentation

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

\( E: \{0,1\}^N \rightarrow \mathbb{R} \)

0 → fg
1 → bg

\( N = \text{number of pixels} \)

Unary Cost \( (c_i) \)

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

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

- \( E: \{0,1\}^N \rightarrow \mathbb{R} \)
- 0 \( \rightarrow \) fg
- 1 \( \rightarrow \) bg

\( N = \) number of pixels

Discontinuity Cost \((c_{ij})\)
Example: Image Segmentation

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

Global Minimum \((x^*)\)

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

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

\(E: \{0,1\}^N \rightarrow \mathbb{R}\)

0 → fg
1 → bg

\(N = \text{number of pixels}\)
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 $x$: $E(x)$

$$E(x)$$

Slide credit: Pushmeet Kohli
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

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

- **Unary potentials**
  - \( D_p(L_p) \)
  - \( L_p \in \{s, t\} \)
  - (binary object segmentation)

- **Pairwise potentials**
  - \( \sum_{pq \in N} w_{pq} \cdot \delta(L_p \neq L_q) \)
  - \( n \)-links

**Diagram**
- \( D_p(t) \)
- \( D_p(s) \)
- \( t \)-links
- \( s \)-links
- A cut
Adding Regional Properties

Regional bias example

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

\[
D_p(s) \propto \exp \left( -\| I_p - I^s \|_2^2 / 2\sigma^2 \right)
\]

\[
D_p(t) \propto \exp \left( -\| I_p - I^t \|_2^2 / 2\sigma^2 \right)
\]

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

“expected” intensities of object and background \( I^s \) and \( I^t \) can be re-estimated.

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

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

EM-style optimization

Slide credit: Yuri Boykov

[Boykov & Jolly, ICCV’01]
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
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

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 an 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 an st-cut?
Sum of cost of all edges going from \(S\) to \(T\)

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

\[ 2 + 1 + 4 = 7 \]
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>$O(n^2mU)$</td>
</tr>
<tr>
<td>1955</td>
<td>Ford &amp; Fulkerson</td>
<td>$O(m^2U)$</td>
</tr>
<tr>
<td>1970</td>
<td>Dinitz</td>
<td>$O(n^2m)$</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>$O(n^2m^{1/2})$</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>$O(nm + n^2 \log^2 n)$</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>$O(n^{3/2} \log(n^2/m) \log U)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(n^{2/3}m \log(n^2/m) \log U)$</td>
</tr>
</tbody>
</table>

### Algorithms

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

Algorithms assume non-negative edge weights.
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 = 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

Flow = 0 + 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

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 = 2

Source

v_1

v_2

Sink

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

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

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

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

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

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

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
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
    \[\text{http://pub.ist.ac.at/~vnk/software.html}\]

Slide credit: Pushmeet Kohli
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]

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

Submodularity ("convexity")

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  \[ \Rightarrow \text{Solution will be globally optimal.} \]
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

\[ E(a_1, a_2) \]
Example: Graph Construction

\[ E(a_1, a_2) = 2a_1 \]
Example: Graph Construction

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

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

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

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + a_1\bar{a}_2 + 2\bar{a}_1a_2 \]
Example: Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\overline{a}_1 + 9a_2 + 4\overline{a}_2 + a_1\overline{a}_2 + 2\overline{a}_1a_2 \]
Example: Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + a_1\bar{a}_2 + 2\bar{a}_1a_2 \]

Cost of cut = 11

\[ a_1 = 1, a_2 = 1 \]

\[ E(1,1) = 11 \]
Example: Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + a_1\bar{a}_2 + 2\bar{a}_1a_2 \]
How Does the Code Look Like?

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)
How Does the Code Look Like?

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)
How Does the Code Look Like?

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)
How Does the Code Look Like?

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)

b\_gCost(a_1)

\_cost(p,q)

b\_gCost(a_2)

\_fgCost(a_1)

\_fgCost(a_2)

\_a_1 = bg \_a_2 = fg
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.
**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
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.

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

Global optimum of the energy
Iterated Graph Cuts

Result

Color model
(Mixture of Gaussians)

Energy after each iteration

1 2 3 4

Foreground
Background

R
G

B. Leibe

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://pub.ist.ac.at/~vnk/software.html](http://pub.ist.ac.at/~vnk/software.html)