GraphCut-based Optimisation for Computer Vision

Ľúbor Ladický
Overview

• Motivation
• Min-Cut / Max-Flow (Graph Cut) Algorithm
• Markov and Conditional Random Fields
• Random Field Optimisation using Graph Cuts
  • Submodular vs. Non-Submodular Problems
  • Pairwise vs. Higher Order Problems
  • 2-Label vs. Multi-Label Problems
• Recent Advances in Random Field Optimisation
• Conclusions
Overview

• Motivation
• Min-Cut / Max-Flow (Graph Cut) Algorithm
• Markov and Conditional Random Fields
• Random Field Optimisation using Graph Cuts
  • Submodular vs. Non-Submodular Problems
  • Pairwise vs. Higher Order Problems
  • 2-Label vs. Multi-Label Problems
• Recent Advances in Random Field Optimisation
• Conclusions
Image Labelling Problems

Assign a label to each image pixel

Geometry Estimation

Image Denoising

Object Segmentation

Depth Estimation
• Labellings highly structured
Image Labelling Problems

- Labellings highly structured
- Labels highly correlated with very complex dependencies

- Neighbouring pixels tend to take the same label
- Low number of connected components
- Classes present may be seen in one image
- Geometric / Location consistency
- Planarity in depth estimation
- … many others (task dependent)
Image Labelling Problems

- Labelling highly structured
- Labels highly correlated with very complex dependencies
- Independent label estimation too hard
• Labelling highly structured
• Labels highly correlated with very complex dependencies
• Independent label estimation too hard
• Whole labelling should be formulated as one optimisation problem
Image Labelling Problems

- Labelling highly structured
- Labels highly correlated with very complex dependencies
- Independent label estimation too hard
- Whole labelling should be formulated as one optimisation problem
- Number of pixels up to millions
  - Hard to train complex dependencies
  - Optimisation problem is hard to infer
Image Labelling Problems

- Labelling highly structured
- Labels highly correlated with very complex dependencies
- Independent label estimation too hard
- Whole labelling should be formulated as one optimisation problem
- Number of pixels up to millions
  - Hard to train complex dependencies
  - Optimisation problem is hard to infer

Vision is hard !
• You can either
  • Change the subject from the Computer Vision to the History of Renaissance Art

Vision is hard!
• You can
  either
  • Change the subject from the Computer Vision to the History of Renaissance Art
  or
  • Learn everything about Random Fields and Graph Cuts

Vision is hard!
Foreground / Background Estimation

Rother et al. SIGGRAPH04
Foreground / Background Estimation

\[ E(x) = \sum_{i \in V} \psi_i(x_i) + \sum_{i \in V, j \in N_i} \psi_{ij}(x_i, x_j) \]

Data term

\[ \psi_i(x_i = 0) = -\log(p(x_i \notin FG)) \]
\[ \psi_i(x_i = 1) = -\log(p(x_i \in FG)) \]

Estimated using FG / BG colour models

Smoothness term

\[ \psi_{ij}(x_i, x_j) = K_{ij} \delta(x_i \neq x_j) \]

where \[ K_{ij} = \lambda_1 + \lambda_2 \exp(-\beta(I_i - I_j)^2) \]

Intensity dependent smoothness
\[ E(x) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j) \]

Data term \hspace{1cm} Smoothness term

\[ x^* = \arg \min_{x \in \mathcal{L}} E(x) \]
Foreground / Background Estimation

\[ E(x) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j) \]

Data term \hspace{1cm} Smoothness term

\[ x^* = \arg \min_{x \in \mathcal{L}} E(x) \]

How to solve this optimisation problem?
Forefront / Background Estimation

\[ E(x) = \sum_{i \in V} \psi_i(x_i) + \sum_{i \in V, j \in N_i} \psi_{ij}(x_i, x_j) \]

Data term \hspace{2cm} Smoothness term

\[ x^* = \arg \min_{x \in \mathcal{L}} E(x) \]

How to solve this optimisation problem?

- Transform into min-cut / max-flow problem
- Solve it using min-cut / max-flow algorithm
Overview

• Motivation
• Min-Cut / Max-Flow (Graph Cut) Algorithm
• Markov and Conditional Random Fields
• Random Field Optimisation using Graph Cuts
  • Submodular vs. Non-Submodular Problems
  • Pairwise vs. Higher Order Problems
  • 2-Label vs. Multi-Label Problems
• Applications
• Conclusions
Max-Flow Problem

Task:
Maximize the flow from the sink to the source such that
1) The flow it conserved for each node
2) The flow for each pipe does not exceed the capacity
Max-Flow Problem

\[
\text{max} \sum_{i \in V} f_{si}
\]

s.t.
\[
0 \leq f_{ij} \leq c_{ij}, \quad \forall (i, j) \in E
\]
\[
\sum_{j \in N(i)} f_{ji} - f_{ij} = 0, \quad \forall i \in V \setminus \{s, t\}
\]
Max-Flow Problem

\[
\text{flow from node } i \text{ to node } j
\]

\[
\text{flow from the source}
\]

\[
\text{capacity}
\]

\[
\text{set of edges}
\]

\[
\text{set of nodes}
\]

\[
\begin{align*}
\text{max} & \quad \sum_{i \in V} f_{si} \\
\text{s.t.} & \quad 0 \leq f_{ij} \leq c_{ij}, \\
& \quad \sum_{j \in N(i)} f_{ji} - f_{ij} = 0, \\
& \quad \forall i \in V \setminus \{s, t\}
\end{align*}
\]
Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
  flow += maximum capacity in the path
  Build the residual graph ("subtract" the flow)
  Find the path in the residual graph
End
Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

\[ \text{flow } += \text{ maximum capacity in the path} \]

Build the residual graph ("subtract" the flow)

Find the path in the residual graph

End
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
  flow += maximum capacity in the path
Build the residual graph (“subtract” the flow)
Find the path in the residual graph
End

flow = 3
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

Build the residual graph ("subtract" the flow)

Find the path in the residual graph

End

\[ r_{ij} = c_{ij} - f_{ij} + f_{ji} \]

flow = 3
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

- Find the path from source to sink
- While (path exists)
  - flow += maximum capacity in the path
  - Build the residual graph ("subtract" the flow)
  - Find the path in the residual graph
- End

flow = 3
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

1. Find the path from source to sink
2. While (path exists)
   a. flow += maximum capacity in the path
   b. Build the residual graph ("subtract" the flow)
   c. Find the path in the residual graph
3. End

flow = 3
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
    flow += maximum capacity in the path
    Build the residual graph (“subtract” the flow)
    Find the path in the residual graph
End

flow = 6
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
    flow += maximum capacity in the path
    Build the residual graph ("subtract" the flow)
    Find the path in the residual graph
End

flow = 6
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
    flow += maximum capacity in the path
    Build the residual graph ("subtract" the flow)
    Find the path in the residual graph
End

flow = 6
Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
  flow += maximum capacity in the path
  Build the residual graph ("subtract" the flow)
  Find the path in the residual graph
End

flow = 6
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
  flow += maximum capacity in the path
Build the residual graph ("subtract" the flow)
Find the path in the residual graph
End

flow = 11
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

Build the residual graph ("subtract" the flow)

Find the path in the residual graph

End

flow = 11
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
    flow += maximum capacity in the path
    Build the residual graph ("subtract" the flow)
    Find the path in the residual graph
End

flow = 11
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
    flow += maximum capacity in the path
    Build the residual graph ("subtract" the flow)
    Find the path in the residual graph
End

flow = 11
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

flow = 13
Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

  flow += maximum capacity in the path

Build the residual graph ("subtract" the flow)

Find the path in the residual graph

End

flow = 13
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

\[
\text{flow } += \text{ maximum capacity in the path}
\]

Build the residual graph ("subtract" the flow)

Find the path in the residual graph

End

flow = 13
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
    flow += maximum capacity in the path
    Build the residual graph (“subtract” the flow)
    Find the path in the residual graph
End

flow = 13
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
  flow += maximum capacity in the path
Build the residual graph ("subtract" the flow)
Find the path in the residual graph
End

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

Build the residual graph ("subtract" the flow)

Find the path in the residual graph

End

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
  flow += maximum capacity in the path
Build the residual graph (“subtract” the flow)
Find the path in the residual graph
End

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
    flow += maximum capacity in the path
    Build the residual graph ("subtract" the flow)
    Find the path in the residual graph
End

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Find the path from source to sink
While (path exists)
  flow += maximum capacity in the path
Build the residual graph ("subtract" the flow)
Find the path in the residual graph
End

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal?

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal?

1. Let S be the set of reachable nodes

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal?

1. Let \( S \) be the set of reachable nodes
2. The flow equal to the sum of capacities from \( S \) to \( V - S \)

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal?

1. Let S be the set of reachable nodes
2. The flow equal to the sum of capacities from S to V – S
3. The flow is upper bounded by the sum of capacities from any set A to V – A

flow = 15
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal?

1. Let $S$ be the set of reachable nodes
2. The flow equal to the sum of capacities from $S$ to $V - S$
3. The flow is upper bounded by the sum of capacities from any set $A$ to $V - A$
4. The solution must be globally optimal
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Order does matter
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Order does matter
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Order does matter
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Order does matter
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Order does matter
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Order does matter

- Standard algorithm not polynomial
- Breath first leads to $O(VE^2)$
  - Path found in $O(E)$
  - At least one edge gets saturated
  - The saturated edge distance to the source has to increase and is at most $V$ leading to $O(VE)$

(Edmonds & Karp, Dinic)
Max-Flow Problem

Ford & Fulkerson algorithm (1956)

Order does matter

- Standard algorithm not polynomial
- Breath first leads to $O(VE^2)$
  (Edmonds & Karp)
- Various methods use different algorithm
to find the path and vary in complexity
Task:

Minimize the cost of the cut

1) Each node is either assigned to the source $S$ or sink $T$
2) The cost of the edge $(i, j)$ is taken if $(i \in S)$ and $(j \in T)$
Min-Cut Problem

\[
\min_{S,T} \sum_{i \in S, j \in T} c_{ij}
\]
Min-Cut Problem

edge costs

source set
sink set

\[
\min_{S,T} \sum_{i \in S, j \in T} c_{ij}
\]
Min-Cut Problem

\[ \text{edge costs} \]

\[ \text{source set} \]

\[ \text{sink set} \]

\[ \min_{S,T} \sum_{i \in S, j \in T} C_{ij} \]

\[ \text{cost} = 18 \]
Min-Cut Problem

Source

Sink

cost = 25

edge costs

source set

sink set
Min-Cut Problem

Source set

Sink set

Edge costs

\[
\min_{S,T} \sum_{i \in S, j \in T} C_{ij}
\]

Cost = 23
Min-Cut Problem

\[
\min_x \sum_{(s,i) \in E} c_{si} x_i + \sum_{(i,t) \in E} c_{it} (1 - x_i) \\
+ \sum_{(i,j) \in E, i,j \notin \{s,t\}} c_{ij} (1 - x_i)x_j
\]

\[x_i = 0 \Rightarrow x_i \in S \quad x_i = 1 \Rightarrow x_i \in T\]
Min-Cut Problem

\[
\begin{align*}
\min_x & \quad \sum_{(s,i) \in E} c_{si} x_i + \sum_{(i,t) \in E} c_{it} (1 - x_i) \\
& \quad + \sum_{(i,j) \in E, i,j \notin \{s,t\}} c_{ij} (1 - x_i) x_j
\end{align*}
\]

\[x_i = 0 \Rightarrow x_i \in S\]
\[x_i = 1 \Rightarrow x_i \in T\]
Min-Cut Problem

\[ \min_{x} \sum_{(s,i) \in E} c_{si}x_i + \sum_{(i,t) \in E} c_{it}(1 - x_i) + \sum_{(i,j) \in E, i,j \notin \{s,t\}} c_{ij}(1 - x_i)x_j \]

\[ x_i = 0 \Rightarrow x_i \in S \quad x_i = 1 \Rightarrow x_i \in T \]

Dual to max-flow problem
Min-Cut Problem

\[ C(x) = 5x_1 + 9x_2 + 4x_3 + 3x_3(1-x_1) + 2x_1(1-x_3) + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) + 1x_5(1-x_1) + 6x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) + 8(1-x_5) + 5(1-x_6) \]
Min-Cut Problem

\[ C(x) = 5x_1 + 9x_2 + 4x_3 + 3x_3(1-x_1) + 2x_1(1-x_3) \]
\[ \quad + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \]
\[ \quad + 1x_5(1-x_1) + 6x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \]
\[ \quad + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) \]
\[ \quad + 8(1-x_5) + 5(1-x_6) \]
Min-Cut Problem

\[ C(x) = 2x_1 + 9x_2 + 4x_3 + 2x_1(1-x_3) \]

\[ + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \]

\[ + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \]

\[ + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) \]

\[ + 5(1-x_5) + 5(1-x_6) \]

\[ + 3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 2x_1 + 9x_2 + 4x_3 + 2x_1(1-x_3) + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 3x_6(1-x_5) + 6(1-x_4) + 5(1-x_5) + 5(1-x_6) + 3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5) = 3 + 3x_1(1-x_3) + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 3 + 2x_1 + 9x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) + 5(1-x_5) + 5(1-x_6) + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 3 + 2x_1 + 9x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) + 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4) + 5(1-x_5) + 5(1-x_6) + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 3 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) + 2x_5(1-x_4) + 6(1-x_4) + 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5) + 3x_2 + 3x_6(1-x_2) + 3x_5(1-x_6) + 3(1-x_5) \]

= 

\[ 3 + 3x_2(1-x_6) + 3x_6(1-x_5) \]
Min-Cut Problem

\[ C(x) = 6 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) + 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) + 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 6 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3) \]
\[ + 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1) \]
\[ + 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6) \]
\[ + 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \]
\[ + 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 11 + 2x_1 + 1x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_4(1-x_1) + 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6) + 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) + 2(1-x_5) + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 11 + 2x_1 + 1x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_4(1-x_1) + 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6) + 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) + 2(1-x_5) + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 13 + 1x_2 + 4x_3 + 5x_1(1-x_3) \]
\[ + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \]
\[ + 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6) \]
\[ + 2x_4(1-x_5) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \]
\[ + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 13 + 1x_2 + 2x_3 + 5x_1(1-x_3) \]
\[ + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \]
\[ + 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6) \]
\[ + 2x_4(1-x_5) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \]
\[ + 3x_3(1-x_5) \]
Min-Cut Problem

\[ C(x) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3) \]
\[ + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \]
\[ + 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5) \]
\[ + 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \]
Min-Cut Problem

\[ C(x) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3) \]
\[ + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) \]
\[ + 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5) \]
\[ + 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \]
Min-Cut Problem

\[ C(x) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) + 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5) + 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \]
Min-Cut Problem

Min cut

\[ C(x) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3) + 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4) + 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5) + 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5) \]
Overview

- Motivation
- Min-Cut / Max-Flow (Graph Cut) Algorithm
- Markov and Conditional Random Fields
- Random Field Optimisation using Graph Cuts
  - Submodular vs. Non-Submodular Problems
  - Pairwise vs. Higher Order Problems
  - 2-Label vs. Multi-Label Problems
- Recent Advances in Random Field Optimisation
- Conclusions
• Markov / Conditional Random fields model probabilistic dependencies of the set of random variables
Markov and Conditional RF

- Markov / Conditional Random fields model conditional dependencies between random variables
- Each variable is conditionally independent of all other variables given its neighbours
Markov and Conditional RF

- Markov / Conditional Random fields model conditional dependencies between random variables
- Each variable is conditionally independent of all other variables given its neighbours
- Posterior probability of the labelling $x$ given data $D$ is:

$$\Pr(x|D) = \frac{1}{Z} \exp\left(- \sum_{c \in C} \psi_c(x_c)\right)$$

partition function  
cliques  
potential functions
Markov and Conditional RF

• Markov / Conditional Random fields model conditional dependencies between random variables

• Each variable is conditionally independent of all other variables given its neighbours

• Posterior probability of the labelling $x$ given data $D$ is:

$$\Pr(x|D) = \frac{1}{Z} \exp(-\sum_{c \in C} \psi_c(x_c))$$

  - partition function
  - cliques
  - potential functions

• Energy of the labelling is defined as:

$$E(x) = - \log \Pr(x|D) - \log Z = \sum_{c \in C} \psi_c(x_c)$$
The most probable (Max a Posteriori (MAP)) labelling is defined as:

\[ x^* = \arg \max_{x \in L} \Pr(x|D) = \arg \min_{x \in L} E(x) \]
• The most probable (Max a Posteriori (MAP)) labelling is defined as:

\[ x^* = \arg \max_{x \in \mathcal{L}} \Pr(x|D) = \arg \min_{x \in \mathcal{L}} E(x) \]

• The only distinction (MRF vs. CRF) is that in the CRF the conditional dependencies between variables depend also on the data
• The most probable (Max a Posteriori (MAP)) labelling is defined as:

$$x^* = \arg \max_{x \in L} \Pr(x|D) = \arg \min_{x \in L} E(x)$$

• The only distinction (MRF vs. CRF) is that in the CRF the conditional dependencies between variables depend also on the data.

• Typically we define an energy first and then pretend there is an underlying probabilistic distribution there, but there isn’t really (Psssssst, don’t tell anyone)
Overview

• Motivation
• Min-Cut / Max-Flow (Graph Cut) Algorithm
• Markov and Conditional Random Fields
• Random Field Optimisation using Graph Cuts
  • Submodular vs. Non-Submodular Problems
  • Pairwise vs. Higher Order Problems
  • 2-Label vs. Multi-Label Problems
• Recent Advances in Random Field Optimisation
• Conclusions
Pairwise CRF models

Standard CRF Energy

\[ E(x) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in N_i} \psi_{ij}(x_i, x_j) \]

Data term  Smoothness term
The energy / potential is called submodular (only) if for every pair of variables:

\[ E(0, 0, \bar{x}_{ij}) + E(1, 1, \bar{x}_{ij}) \leq E(0, 1, \bar{x}_{ij}) + E(1, 0, \bar{x}_{ij}) \]
Graph Cut based Inference

For 2-label problems $x \in \{0, 1\}$:

$$E(x) = \sum_{i \in V} \psi_i(x_i) + \sum_{i \in V, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j)$$

$$= \sum_{i \in V} (g_i^1 x_i + g_i^0 (1 - x_i)) + \sum_{i \in V, j \in \mathcal{N}_i} (g_{ij}^{00} (1 - x_i)(1 - x_j)$$

$$+ g_{ij}^{01} (1 - x_i)x_j + g_{ij}^{10} x_i(1 - x_j) + g_{ij}^{11} x_i x_j)$$
For 2-label problems $x \in \{0, 1\}$:

$$E(x) = \sum_{i \in V} \psi_i(x_i) + \sum_{i \in V, j \in N_i} \psi_{ij}(x_i, x_j)$$

$$= \sum_{i \in V} (g_i^1 x_i + g_i^0 (1 - x_i)) + \sum_{i \in V, j \in N_i} (g_{ij}^{00} (1 - x_i)(1 - x_j)$$

$$+ g_{ij}^{01} (1 - x_i)x_j + g_{ij}^{10} x_i (1 - x_j) + g_{ij}^{11} x_i x_j)$$

Pairwise potential can be transformed into:

$$\psi_{ij}(x_i, x_j) = K_{ij} + g'_i x_i + g'_j x_j + c_{ij} (1 - x_i)x_j + c_{ij} x_i (1 - x_j)$$

where

$$K_{ij} = g_{ij}^{00}$$

$$g'_i = \frac{g_{ij}^{10} + g_{ij}^{11} - g_{ij}^{01} - g_{ij}^{00}}{2}$$

$$g'_j = \frac{g_{ij}^{01} + g_{ij}^{11} - g_{ij}^{10} - g_{ij}^{00}}{2}$$

$$c_{ij} = \frac{g_{ij}^{01} + g_{ij}^{10} - g_{ij}^{00} - g_{ij}^{11}}{2}$$
Graph Cut based Inference

For 2-label problems \( x \in \{0, 1\} : \\

\[
E(x) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j) \\
= \sum_{i \in \mathcal{V}} (g_i^1 x_i + g_i^0 (1 - x_i)) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} (g_{ij}^{00} (1 - x_i)(1 - x_j) \\
+ g_{ij}^{01} (1 - x_i)x_j + g_{ij}^{10} x_i(1 - x_j) + g_{ij}^{11} x_i x_j)
\]

Pairwise potential can be transformed into :

\[
\psi_{ij}(x_i, x_j) = K_{ij} + g'_i x_i + g'_j x_j + c_{ij} (1 - x_i) x_j + c_{ij} x_i (1 - x_j)
\]

where

\[
K_{ij} = g_{ij}^{00} \\
g'_i = \frac{g_{ij}^{10} + g_{ij}^{11} - g_{ij}^{01} - g_{ij}^{00}}{2} \\
g'_j = \frac{g_{ij}^{01} + g_{ij}^{11} - g_{ij}^{10} - g_{ij}^{00}}{2} \\
c_{ij} = \frac{g_{ij}^{01} + g_{ij}^{10} - g_{ij}^{00} - g_{ij}^{11}}{2} \geq 0
\]
Graph Cut based Inference

For 2-label problems $x \in \{0, 1\}$:

$$E(x) = \sum_{i \in V} \psi_i(x_i) + \sum_{i \in V, j \in N_i} \psi_{ij}(x_i, x_j)$$

$$= \sum_{i \in V} \left( g^1_i x_i + g^0_i (1 - x_i) \right) + \sum_{i \in V, j \in N_i} \left( g_{ij}^{00} (1 - x_i)(1 - x_j) + g_{ij}^{01} (1 - x_i)x_j + g_{ij}^{10} x_i(1 - x_j) + g_{ij}^{11} x_i x_j \right)$$

Pairwise potential can be transformed into:

$$\psi_{ij}(x_i, x_j) = K_{ij} + g'_i x_i + g'_j x_j + c_{ij} (1 - x_i)x_j + c_{ij} x_i(1 - x_j)$$

where

$$K_{ij} = g_{ij}^{00}$$

$$g'_i = \frac{g_{ij}^{10} + g_{ij}^{11} - g_{ij}^{01} - g_{ij}^{00}}{2}$$

$$c_{ij} = \frac{g_{ij}^{01} + g_{ij}^{10} - g_{ij}^{00} - g_{ij}^{11}}{2} \geq 0$$

$$g'_j = \frac{g_{ij}^{01} + g_{ij}^{11} - g_{ij}^{10} - g_{ij}^{00}}{2}$$
Graph Cut based Inference

After summing up:

\[ E(x) = K + \sum_{i \in V} c_i x_i + \sum_{i \in V, j \in N_i} c_{ij} (1 - x_i) x_j + c_{ij} x_i (1 - x_j) \]
After summing up:

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

Let:

\[ c_{it} = c_i \text{ and } c_{si} = 0 \text{ if } c_i \geq 0, \]

\[ c_{it} = 0 \text{ and } c_{si} = -c_i \text{ otherwise} \]
Graph Cut based Inference

After summing up:

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

Let:

- \( c_{it} = c_i \) and \( c_{si} = 0 \) if \( c_i \geq 0 \).
- \( c_{it} = 0 \) and \( c_{si} = -c_i \) otherwise.

Then:

- \( c_i x_i = c_{it} x_i \)
- \( c_i x_i = -c_{si} + c_{si}(1 - x_i) \)
Graph Cut based Inference

After summing up:

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

Let:

- \( c_{it} = c_i \) and \( c_{si} = 0 \) if \( c_i \geq 0 \),
- \( c_{it} = 0 \) and \( c_{si} = -c_i \) otherwise

Then:

\[ c_i x_i = c_{it} x_i \]
\[ c_i x_i = -c_{si} + c_{si}(1 - x_i) \]

Equivalent st-mincut problem is:

\[ x^* = \arg \min_x \sum_{i \in V} c_{si} x_i + c_{it}(1 - x_i) + \sum_{i \in V, j \in N_i} c_{ij}(1 - x_i)x_j + c_{ij} x_i (1 - x_j) \]
\[ E(x) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j) \]

**Data term**

\[
\psi_i(x_i = 0) = -\log(p(x_i \notin FG))
\]

\[
\psi_i(x_i = 1) = -\log(p(x_i \in FG))
\]

Estimated using FG / BG colour models

**Smoothness term**

\[
\psi_{ij}(x_i, x_j) = K_{ij} \delta(x_i \neq x_j)
\]

where

\[
K_{ij} = \lambda_1 + \lambda_2 \exp(-\beta(I_i - I_j)^2)
\]

Intensity dependent smoothness
Foreground / Background Estimation

\[ \psi_i(x_i = 1) \]

\[ \psi_i(x_i = 0) \]

Rother et al. SIGGRAPH04
Extendable to (some) Multi-label CRFs

- each state of multi-label variable encoded using multiple binary variables
- the cost of every possible cut must be the same as the associated energy
- the solution obtained by inverting the encoding
Dense Stereo Estimation

- For each pixel assigns a disparity label: $z$
- Disparities from the discrete set $\{0, 1, .., D\}$
Dense Stereo Estimation

Data term

$$\psi_i(z_i) = d(F^L(x_i, y_i), F^R(x_i - z_i, y_i))$$

Left image feature

Shifted right image feature

Left Camera Image

Right Camera Image
Dense Stereo Estimation

Data term

\[ \psi_i(z_i) = d(F^L(x_i, y_i), F^R(x_i - z_i, y_i)) \]

Left image feature \quad Shifted right image feature

Smoothness term

\[ \psi_{ij}(z_i, z_j) = K |z_i - z_j| \]
Dense Stereo Estimation

Encoding

\[ z_i = 0 \iff \{ x_i^0 = 0, x_i^1 = 1, x_i^2 = 1, x_i^3 = 1, \ldots, x_i^D = 1 \} \]
\[ z_i = 1 \iff \{ x_i^0 = 0, x_i^1 = 0, x_i^2 = 1, x_i^3 = 1, \ldots, x_i^D = 1 \} \]
\[ z_i = 2 \iff \{ x_i^0 = 0, x_i^1 = 0, x_i^2 = 0, x_i^3 = 1, \ldots, x_i^D = 1 \} \]
\[ \vdots \]
\[ z_i = D \iff \{ x_i^0 = 0, x_i^1 = 0, x_i^2 = 0, x_i^3 = 0, \ldots, x_i^D = 0 \} \]
The cost of every cut should be equal to the corresponding energy under the encoding.

\[
\text{cost} = \psi_i(0) + \psi_k(D)
\]
The cost of every cut should be equal to the corresponding energy under the encoding.
Dense Stereo Estimation

Unary Potential

The cost of every cut should be equal to the corresponding energy under the encoding

\[ \text{cost} = \infty \]
Dense Stereo Estimation

Pairwise Potential

\[
\text{cost} = \psi_i(1) + \psi_k(0) + K
\]

The cost of every cut should be equal to the corresponding energy under the encoding.
Dense Stereo Estimation

Pairwise Potential

The cost of every cut should be equal to the corresponding energy under the encoding

cost = \psi_i(0) + \psi_k(D) + D K
Dense Stereo Estimation

Extendable to any convex cost

\[ \psi_{ij}(z_i, z_j) = f(z_i - z_j) \]

Convex function

\[ c_{x_i^j x_k^m} = c_{x_k^m x_i^j} = \frac{f(l_i - l_k + 1) - 2f(l_i - l_k) + f(l_i - l_k - 1)}{2} \]
Graph Cut based Inference

Move making algorithms

- Original problem decomposed into a series of subproblems solvable with graph cut
- In each subproblem we find the optimal move from the current solution in a restricted search space
Example: [Boykov et al. 01]

**αβ-swap**
- each variable taking label $\alpha$ or $\beta$ can change its label to $\alpha$ or $\beta$
- all $\alpha\beta$-moves are iteratively performed till convergence

$$T_{\alpha\beta}(x_i, t_i) = \begin{cases} 
\alpha & \text{if } x_i \in \{\alpha, \beta\} \text{ and } t_i = 0 \\
\beta & \text{if } x_i \in \{\alpha, \beta\} \text{ and } t_i = 1
\end{cases}$$

**α-expansion**
- each variable either keeps its old label or changes to $\alpha$
- all $\alpha$-moves are iteratively performed till convergence

Transformation function:

$$T_{\alpha}(x_i, t_i) = \begin{cases} 
\alpha & \text{if } t_i = 0 \\
x_i & \text{if } t_i = 1
\end{cases}$$
Graph Cut based Inference

Sufficient conditions for move making algorithms:
(all possible moves are submodular)

\[ \alpha \beta \text{-swap: } \text{semi-metricity} \]

\[ \forall l_a, l_b \in \mathcal{L}, \]
\[ \psi^p(l_a, l_a) = 0 \]
\[ \psi^p(l_a, l_b) = \psi^p(l_b, l_a) \geq 0 \]

Proof:

\[ \psi^p(l_a, l_b) + \psi^p(l_b, l_a) - \psi^p(l_a, l_a) - \psi^p(l_b, l_b) = 2\psi^p(l_a, l_b) \geq 0 \]
Graph Cut based Inference

Sufficient conditions for move making algorithms:
(all possible moves are submodular)

\[ \alpha\text{-expansion: } \text{metricity} \]

\[ \forall l_a, l_b, l_c \in \mathcal{L} \]
\[ \psi^p(l_a, l_a) = 0 \]
\[ \psi^p(l_a, l_b) = \psi^p(l_b, l_a) \geq 0 \]
\[ \psi^p(l_a, l_b) + \psi^p(l_b, l_c) \geq \psi^p(l_a, l_c) \]

Proof:

\[ \psi^p(l_a, l_b) + \psi^p(l_c, l_a) - \psi^p(l_a, l_a) - \psi^p(l_b, l_c) \]
\[ = \psi^p(l_a, l_b) + \psi^p(l_c, l_a) - \psi^p(l_b, l_c) \geq 0 \]
Object-class Segmentation

\[ E(x) = \sum_{i \in V} \psi_i(x_i) + \sum_{i \in V, j \in N_i} \psi_{ij}(x_i, x_j) \]

**Data term**

**Smoothness term**

**Data term**

Discriminatively trained classifier

**Smoothness term**

\[ \psi_{ij}(x_i, x_j) = K_{ij} \delta(x_i \neq x_j) \]

where \[ K_{ij} = \lambda_1 + \lambda_2 \exp(-\beta(I_i - I_j)^2) \]
Object-class Segmentation

Original Image

Initial solution

grass
Object-class Segmentation

Original Image | Initial solution | Building expansion

- Grass
- Building
Object-class Segmentation

Original Image

Initial solution

Building expansion

Sky expansion
Object-class Segmentation

Original Image

Initial solution

Building expansion

Sky expansion

Tree expansion
Object-class Segmentation

Original Image

Initial solution

Building expansion

Sky expansion

Tree expansion

Final Solution
Range-(swap) moves [Veksler 07]
- Each variable in the convex range can change its label to any other label in the convex range
- All range-moves are iteratively performed till convergence

Range-expansion [Kumar, Veksler & Torr 1]
- Expansion version of range swap moves
- Each variable can change its label to any label in a convex range or keep its old label
Dense Stereo Reconstruction

Data term

Same as before

Smoothness term

\[ \psi_{ij}(z_i, z_j) = \min(K|z_i - z_j|, T) \]
Dense Stereo Reconstruction

Data term

Same as before

Smoothness term

\[ \psi_{ij}(z_i, z_j) = \min\left(K|z_i - z_j|, T\right) \]

Convex part  Truncation
Graph Cut based Inference

Original Image

Initial Solution
Graph Cut based Inference

Original Image

Initial Solution

After 1st expansion
Graph Cut based Inference

Original Image

Initial Solution

After 1st expansion

After 2nd expansion
Graph Cut based Inference

Original Image

Initial Solution

After 1\textsuperscript{st} expansion

After 2\textsuperscript{nd} expansion

After 3\textsuperscript{rd} expansion
Graph Cut based Inference

Original Image  →  Initial Solution  →  After 1\textsuperscript{st} expansion  

After 2\textsuperscript{nd} expansion  →  After 3\textsuperscript{rd} expansion  →  Final solution
Graph Cut based Inference

- Extendible to certain classes of binary higher order energies
- Higher order terms have to be transformable to the pairwise submodular energy with binary auxiliary variables $z_C$

\[
\psi_c(x_c) = \min_{z_c} \psi^P_c(x_c, z_c)
\]

Higher order term  Pairwise term

Example: \[
\psi(x_1, x_2, x_3) = -x_1x_2x_3 = \min z (2 - x_1 - x_2 - x_3)
\]

Diagram:

- Initial solution
- Propose move
- Transf. to pairwise subm.
- Graph Cut
- Update solution
- Solution
If the energy is not submodular / graph-representable
  – Overestimation by a submodular energy
  – Quadratic Pseudo-Boolean Optimisation (QPBO)
Overestimation by a submodular energy (convex / concave)

- We find an energy $E'(t)$ s.t.
  - it is tight in the current solution $E'(t_0) = E(t_0)$
  - Overestimates $E(t)$ for all $t$ $E'(t) \geq E(t)$
- We replace $E(t)$ by $E'(t)$
- The moves are not optimal, but guaranteed to converge
- The tighter over-estimation, the better
Graph Cut based Inference

Quadratic Pseudo-Boolean Optimisation

- Each (binary) variable is encoded using two binary variables $x_i$ and $\bar{x}_i$ s.t. $\bar{x}_i = 1 - x_i$

- For negative costs ($K < 0$)

$$Kx_i(1-x_j) = K(1- \bar{x}_i)(1-x_j) = K - Kx_j - K\bar{x}_i(1-x_j) \text{ (submodular)}$$

- Solved by dropping the constraint $x_i = 1 - \bar{x}_i$

- If $x_i = 1 - \bar{x}_i$ the solution for $x_i$ is optimal
Thank you

To be continued..