Graph Cuts

Reminder:
Markov Random Fields (MRF)

1D: HMM (Hidden Markov Models)
Markov chain

\[
\begin{align*}
\sum_{t} V_t(x_t) + \sum_{t} D_t(x_t, x_{t+1})
\end{align*}
\]

- solve with dynamic programming

2D

- image = pixel grid + neighborhood in 2 classes
- local prior
  - For each pixel \( p \)
    - (based on its color, the texture around... local descriptor)

\[
\begin{align*}
\sum_{p} V_p(x_p) + \sum_{p \neq q} D_{pq}(x_p, x_q)
\end{align*}
\]

- MRF

\[
\begin{align*}
\Rightarrow \text{general case: NP-hard}
\end{align*}
\]

Max Flow

- Each edge \( = \text{pipe} \)
- maximum capacity
  - (unit: L/s)
- Maximum flow
  - (but can pour from "source" to "sink"?)

First Algorithm: Ford-Fulkerson

Flow (current)

Residual graph

Maximum Capacity

Path through edges not full yet

Also: Iteratively
- Find a path connecting source and sink in the residual graph

Back
Problem:

- saturate this path
  (i.e. add flow along this path until not possible to add more)

\[ \text{Residual graph:} \quad \{ \text{edges not saturated}, \]
\[ \quad \text{and reverse of edges with zero flow}\}

**Core of graph:**

- find a path connecting the source & the sink
- saturate it

**Cornel**

- centrality
- find a path connecting the source & the sink
- saturate it

**Issue:**

- complexity (in worst-case)

- on-saturate an already-saturated edge many many times
- it's not looking at "optimal paths"

**Variation:**

\[ \text{Dinic (Dimic)} \]
\[ \text{Edmonds-Karp} \]

**Idea:**

- don't pick a long path
- pick the shortest path

\[ \Rightarrow \text{better worst-case complexity} \]
\[ O(\text{trees}^3) \]

**Moving other algorithms**

- Push-Relabel

- in practice: Dijkstra/Kalantari heuristic for 2D image segmentation problem

**Link between maximum flow & minimum cut**

- **U:** all nodes reachable from the source
- **residual graph:**
  - from the source in the residual graph

\[ U \text{ : all nodes reachable from the source} \]

- **partition of the graph:**
  - \( U \) & \( \overline{U} \)
  - Load of the graph

\[ \text{Total flow} = \text{capacity of all edges from source} \]
\[ \text{maximum flow} = \text{value of cuts} \]

1) All edges going out of \( U \) are saturated
2) Flow on edges going to \( U \) are 0

**Because these edges are saturated**

- Flow on edges going to \( U \) are 0
- Flow on edges going to \( \overline{U} \) are 0

**Because these edges are saturated**
### Theorem

**Best cut**

- The cost of the best cut is given by the maximum flow solution.
- The cost of the best cut is equal to the maximum flow.

### Image Segmentation as a Min-cut Problem

- **Image grid:** Each pixel is represented by a node.
- **Local preference for each pixel:**
  \[ V_p(x_p) \]
- **Interaction between neighbors:**
  \[ D_{ij}(x_p, x_q) \]

#### Conditions

\[ V_p(x_p) = \begin{cases} +1 & \text{if } x_p = 0 \\ -2 & \text{otherwise} \end{cases} \]

#### Interaction between pixels -> clique with a,b

- **Positivity of all capacities:** Required by max-flow
  \[ a, b > 0 \]

#### Sub-modularity:

- **M:**
  \[ M_{ij} = \begin{cases} +10 & \text{if } x_p = 0, x_q = 1 \\ -1 & \text{otherwise} \end{cases} \]

- **M is submodular:**
  \[ M_{ij} + M_{kl} \leq M_{ijkl} \]

- It can be more to be off-diagonal
  → choose different cliques for neighboring pixels

- Active construction → interaction terms are submodular.

### Any MRF with submodular interaction matrix can be solved exactly by a graph cut

While: if not submodular: NP-hard in the general case.
IV. Multi-level energies

- proposed labeling: \( x \) (\( x_p \))
- criterion to optimize:
  \[ \sum_p v_p(x_p) + \sum_{pq} \delta_{pq} (x_p, x_q) \]

\[ \Rightarrow \text{con't apply the technique done} \]

1) \( x \)-expansion:

- sequence of binary problems
  - iteratively:
    - solve one class \( c \) (\( c(x) \))
    - binary problem:
      - given the current solution:
        - for each pixel: either keep the current choice or move to the other class \( c \)
  - the binary problem obtained needs to subdivide
    \[ \Rightarrow \text{iteration matrix} \]
    \[ \Rightarrow \text{matrix} \]

  \[ \text{Supplementary info: } \delta_{pq}(x, y) = 0 \text{ if } y \]

  \[ \Rightarrow \delta_{pq} \text{ is a distance between labels} \]
  \[ - \delta_{pq}(a, b) \geq 0 \text{ if } a, b \]
  \[ - \delta_{pq}(a, b) + \delta_{pq}(b, c) \geq \delta_{pq}(a, c) \]

2) \( a \)-\( b \) swap

- similar idea: sequence of binary problems
  - iteratively:
    - pick 2 classes \( c, \beta \)
    - binary problem: allow pixels with one of these 2 labels to move to the other one
  - conditions: binary classes to be submodular

\[ \Rightarrow 2 \times 2 \text{ submatrix} \]
\[ \Rightarrow \text{corners in the kernel} \]
\[ \Rightarrow \text{constraint: weaker than for } x \text{-expansion} \]

Example of iteration matrix: Pitts model
\[ \delta_{pq}(a, b) = \delta_{a=b} = \begin{cases} 0 & \text{if } a=b \\ 2 & \text{otherwise} \end{cases} \]

\[ \Rightarrow \text{distances between labels} \]
How to set the problem

Image of a cow on grass

Interactive Segmentation by clicking mouse dues

Infinite links

Tsukada's construction

Cost of cutting line

Ty of receding sequence segmentation

Infinite links between any pixel p at time k and the same pixel at time k-1

A/B: needs pixel-wise registration of all images

Application: image → 5 levels of gray: 0, 0.3, 0.5, 1, 2, 3

Statistics on pixel colors

Histogram of background pixels in a set of segmented images

Intersection between pixels

Estimate the probability of a cut there

\[ E[\|\text{color}(p) - \text{color}(q)\|^2] \]

Constraint that foreground can only shrink with time

Official worst-case complexity: cubic

Uses heuristics by BLK → fast

\[ O(N^2) \] (frame)

\[ O(N) \] (only cut)

Looks like a certain Renaissance