MRF's, CRF's and Refining Localization

D.A. Forsyth

Program and Context

- CRF’s and MRF’s are important in semantic segmentation
- Work an interesting simple problem to set up
  - Have a box on an object, but we’d like tighter boundaries
    - What to do?
  - Early (and very good) techniques
    - Grab Cut
    - Obj Cut
  - Both use MRF/CRF models and inference
    - cover that quickly

Markov random field - formal

Definition

Given an undirected graph $G = (V, E)$, a set of random variables $X = (X_v)_{v \in V}$ indexed by V form a Markov random field with respect to G if they satisfy the local Markov properties:

Pairwise Markov property: Any two non-adjacent variables are conditionally independent given all other variables:

$$X_u \perp\!\!\!\perp X_v \mid X_{V \setminus \{u,v\}}$$

Local Markov property: A variable is conditionally independent of all other variables given its neighbors:

$$X_v \perp\!\!\!\perp X_{V \setminus \mathrm{N}[v]} \mid X_{\mathrm{N}(v)}$$

where $\mathrm{N}(v)$ is the set of neighbors of v, and $\mathrm{N}[v] = v \cup \mathrm{N}(v)$ is the closed neighbourhood of v.

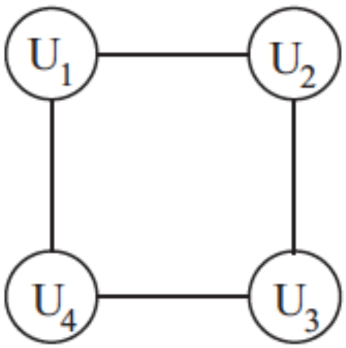
Global Markov property: Any two subsets of variables are conditionally independent given a separating subset:

$$X_A \perp\!\!\!\perp X_B \mid X_S$$

where every path from a node in A to a node in B passes through S.

The Global Markov property is stronger than the Local Markov property, which in turn is stronger than the Pairwise one. [3] However, the above three Markov properties are equivalent for a positive probability.[4]

MRF - First case for us



- The graph is a 2D grid
- Each random variable is a binary random variable
  - eg inside object, outside object
- In this case

$$p(x) \propto \exp\left[\frac{1}{2}\sum_i \sum_j \mathrm{goodness}(x_i, x_j)\right]$$

Look at Ch15 of AML for some examples, BUT that uses different inference procedures and has 1, -1 labels. I'm using Greig; Porteous; Seheult notation (see web page for paper)

Notice

- If the goodness of a pair is high, p is higher
- Because these are binary, we can simplify
- We want:
  - better for neighbors to agree than disagree
  - the goodness for both 0 is the same as for both 1
- Can then simplify

$$p(x) \propto \exp\left[\frac{1}{2}\sum_i \sum_j \mathrm{goodness}(x_i, x_j)\right]$$

- To get

$$p(x) \propto \exp\left[\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}\{x_i x_j + (1-x_i)(1-x_j)\}\right]$$

Important

- We want:
  - better for neighbors to agree than disagree
  - the goodness for both 0 is the same as for both 1
- This means

$$p(x) \propto \exp\left[\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}\{x_i x_j + (1 - x_i)(1 - x_j)\}\right]$$

This is >=0 for i neq j

First model

- At each pixel, there is an unknown binary label
  - 0=out, 1=in
- These binary labels form an MRF
  - where it is cheaper to agree than to disagree
- At each pixel, there are measurements
  - conditioned on the label
  - details to follow
- Q: how do we get the MAP set of labels?

Model

- At each pixel we have observations y
  - yields likelihood

$$l(y|x) = \prod_{i=1}^{n} f(y_i|x_i) = \prod_{i=1}^{n} f(y_i|1)^{x_i} f(y_i|0)^{1-x_i}$$

  - what is f? (later)
  - write $\lambda_i = \ln\{f(y_i|1)/f(y_i|0)\}$

- Then

$$\log p(x|y) = \sum_{i=1}^{n} \lambda_i x_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} \{x_i x_j + (1 - x_i)(1 - x_j)\}$$

$$+K$$

To obtain MAP estimate

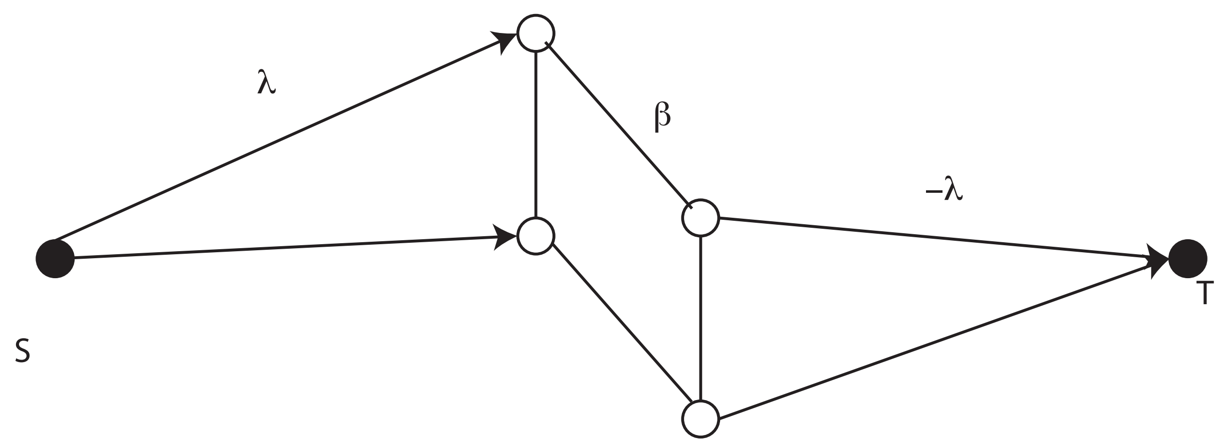
- Maximise

$$\sum_{i=1}^{n} \lambda_i x_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} \{x_i x_j + (1 - x_i)(1 - x_j)\}$$

- But how?
  - blank search won't do it (why?)
- In this special case, graph cut works

Graph cut (quick but clean)

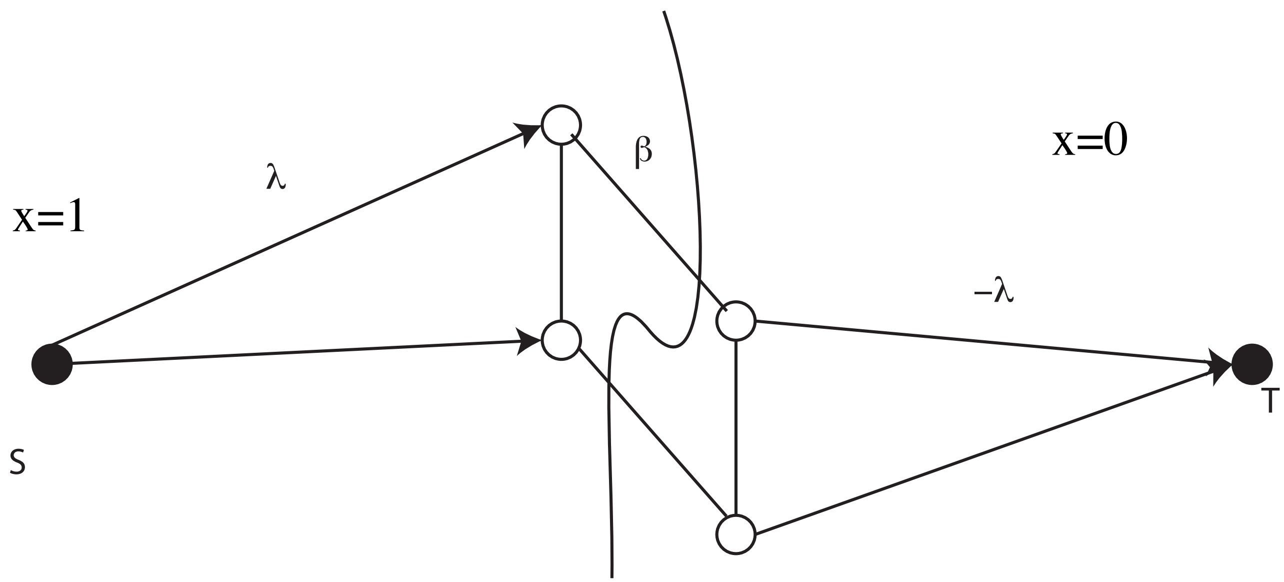
Consider a capacitated network comprising $n + 2$ vertices, being a source s, a sink t and the n pixels. There is a directed edge (s, i) from s to pixel i with capacity $c_{si} = \lambda_i$, if $\lambda_i > 0$; otherwise, there is a directed edge (i, t) from i to t with capacity $c_{it} = -\lambda_i$. There is an undirected edge (i, j) between two internal vertices (pixels) i and j with capacity $c_{ij} = \beta_{ij}$ if the corresponding pixels are neighbours.



Graph cut (quick but clean)

For any binary image $x = (x_1, \ldots, x_n)$ let $B = \{s\} \cup \{i: x_i = 1\}$ and $W = \{i: x_i = 0\} \cup \{t\}$ define a two-set partition of the network vertices and put

$$C(x) = \sum_{k \in B} \sum_{l \in W} c_{kl}\,.$$



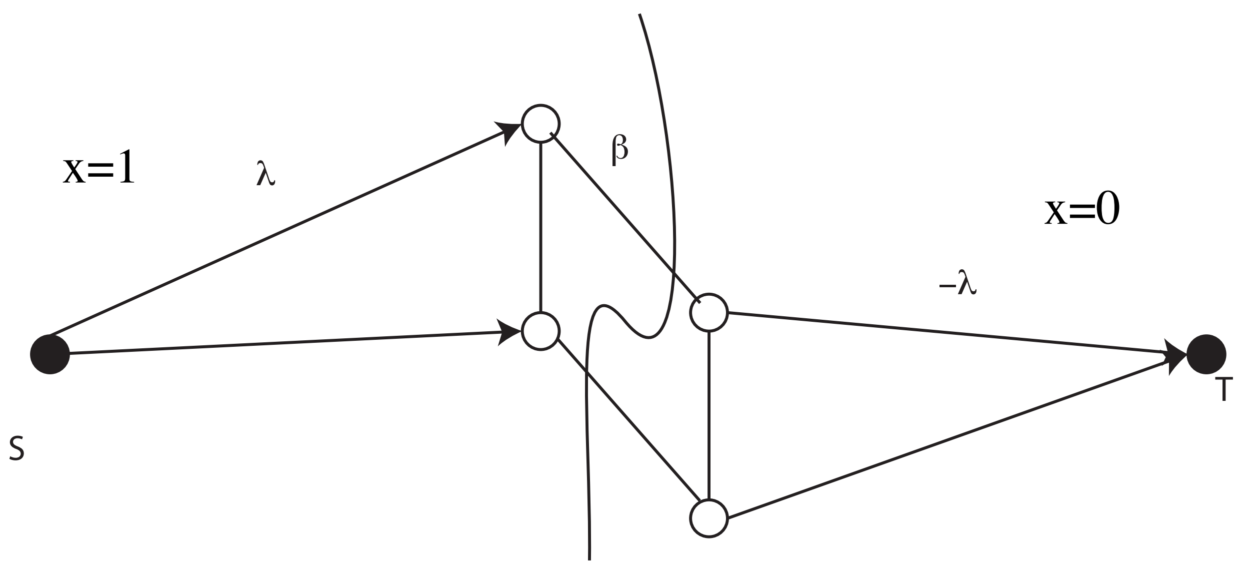
Graph cut (quick but clean)

The set of edges with a vertex in B and a vertex in W is called a *cut* and $C(x)$ is called the *capacity* of the cut.

It is readily seen that $C(x)$ may be written

$$C(x) = \sum_{i=1}^{n} x_i \max(0, -\lambda_i) + \sum_{i=1}^{n} (1 - x_i) \max(0, \lambda_i) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} (x_i - x_j)^2$$

which differs from $-L(x|y)$ by a term which does not depend on x;



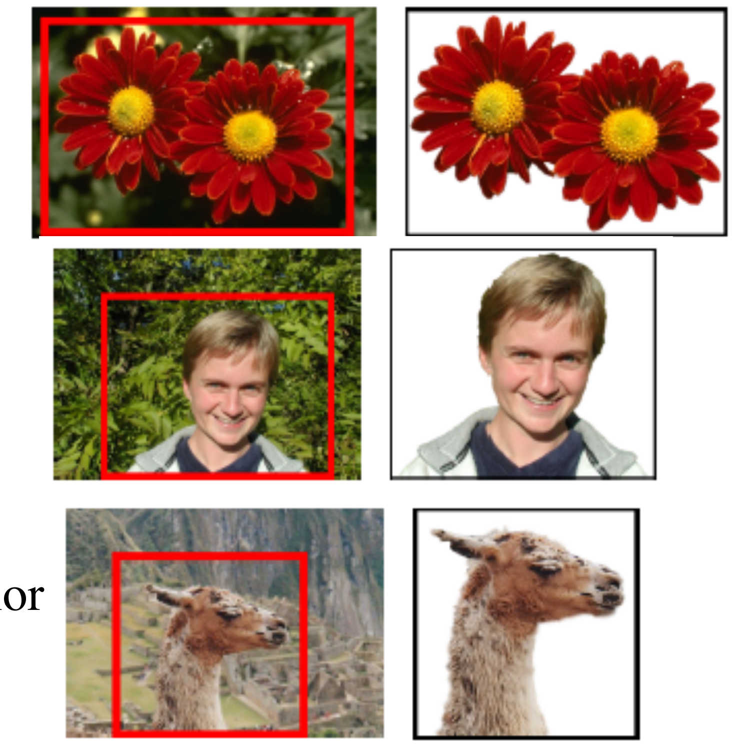
Graph cut, II

- SO
  - set up the graph as described, and do a min-cut
  - this is polynomial
- Ifs, ands, buts
  - this only works in the case it is cheaper to agree than to disagree
    - more general case, it's max cut which isn't funny at all
  - this only works for the binary case
    - but approximations for some multilabel cases are very good
- More details
  - there are *many* min-cut algorithms with different complexities
    - adapted to different types of problem
  - significant literature on best min-cut algorithm for vision applications
    - we'll ignore - search github

Grab Cut

- Originally for matting
  - extracting an object from an image
- Process
  - user places box
    - grabcut segments intended object
  - user perhaps iterates with strokes, etc.

- For us:
  - segments using graph cuts
    - clever iterative model of interior/exterior
  - extremely simple shape prior on object



Simplest case: grey level image

Their paper [Boykov and Jolly 2001] addresses the segmentation of a monochrome image, given an initial trimap T. The image is an array $\mathbf{z} = (z_1, \ldots, z_n, \ldots, z_N)$ of grey values, indexed by the (single) index n. The segmentation of the image is expressed as an array of "opacity" values $\underline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ at each pixel. Generally $0 \leq \alpha_n \leq 1$, but for hard segmentation $\alpha_n \in \{0, 1\}$, with 0 for background and 1 for foreground. The parameters $\underline{\theta}$ describe image foreground and background grey-level distributions, and consist of histograms of grey values:

$$\underline{\theta} = \{h(z; \alpha), \alpha = 0, 1\}, \tag{1}$$

one for background and one for foreground. The histograms are assembled directly from labelled pixels from the respective trimap regions T_B, T_F. (Histograms are normalised to sum to 1 over the grey-level range: $\int_z h(z; \alpha) = 1$.)

Grey level image, II

An energy function $\mathbf{E}$ is defined so that its minimum should correspond to a good segmentation, in the sense that it is guided both by the observed foreground and background grey-level histograms and that the opacity is "coherent", reflecting a tendency to solidity of objects. This is captured by a "Gibbs" energy of the form:

$$\mathbf{E}(\underline{\alpha},\underline{\theta},\mathbf{z}) = U(\underline{\alpha},\underline{\theta},\mathbf{z}) + V(\underline{\alpha},\mathbf{z}) \,. \tag{2}$$

The data term U evaluates the fit of the opacity distribution $\underline{\alpha}$ to the data $\mathbf{z}$, given the histogram model $\underline{\theta}$, and is defined to be:

$$U(\underline{\alpha},\underline{\theta},\mathbf{z}) = \sum_{n} -\log h(z_n;\alpha_n) \,. \tag{3}$$

The smoothness term can be written as

$$V(\underline{\alpha},\mathbf{z}) = \gamma \sum_{(m,n)\in\mathbf{C}} dis(m,n)^{-1} \, [\alpha_n \neq \alpha_m] \exp -\beta (z_m - z_n)^2, \tag{4}$$

where $[\phi]$ denotes the indicator function taking values $0,1$ for a predicate ϕ, $\mathbf{C}$ is the set of pairs of neighboring pixels, and where $dis(\cdot)$ is the Euclidean distance of neighbouring pixels. This energy

Notice

$$\sum_{i=1}^{n} \lambda_i x_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} \{x_i x_j + (1 - x_i)(1 - x_j)\}$$

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in\mathbf{C}} dis(m,n)^{-1} \, [\alpha_n \neq \alpha_m] \exp -\beta (z_m - z_n)^2,$$

$$U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = \sum_n -\log h(z_n; \alpha_n) \, .$$

They're minimizing, and GPS are maximizing; this means they use a cost (not goodness) for disagreeing (not agreeing)

Improving this

- Where does trimap come from?
  - start with
    - inside: a bunch of pixels in “deep interior” of box
    - outside: a bunch of pixels outside box
- Histograms for color images are clumsy
  - too big
- Initial trimap is messy
  - reestimate using segmentation

Replace histograms

- Use mixture of normals
  - have some interior, some exterior pixels
  - build mixture of normal model for each case
    - AML ch 9 if you've forgotten
    - now you can compute $p(y| 1)$, etc. from this

Re-estimation

- Use initial trimap to make GMM
- Segment with graph cut
  - Now you have a trimap
- Re-estimate GMMs, and iterate