

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:

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

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

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

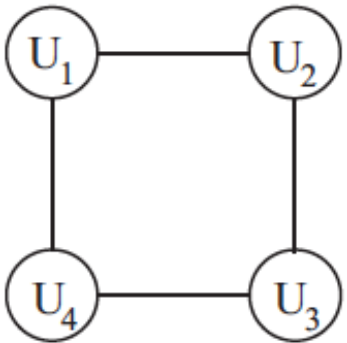
$$X_A \perp\!\!\!\perp X_B \mid X_S$$

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

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

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

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

- To get

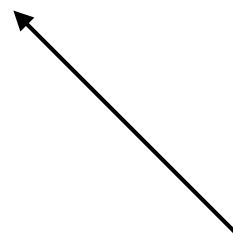
$$p(x) \propto \exp \left[\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \beta_{ij} \{x_i x_j + (1 - x_i)(1 - x_j)\} \right]$$

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

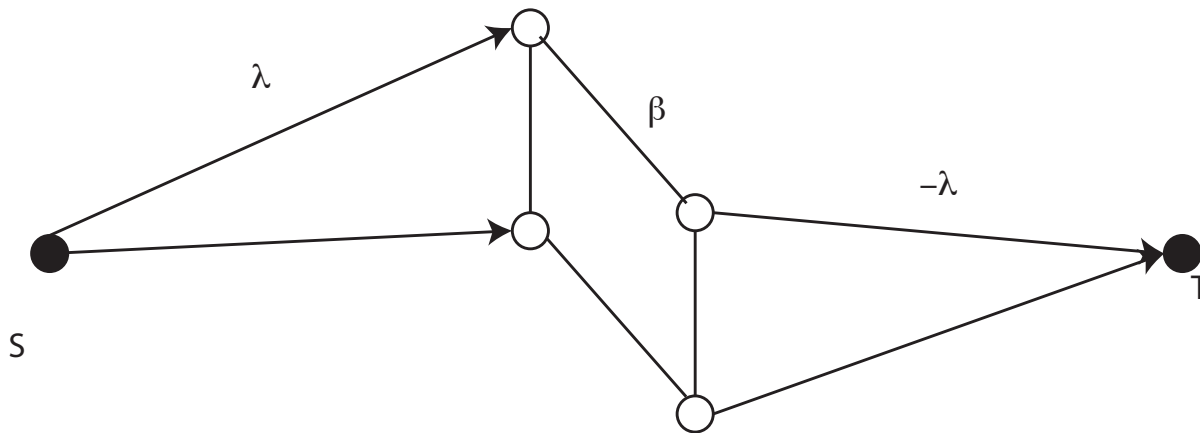
- Maximise

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

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

Graph cut (quick but clean)

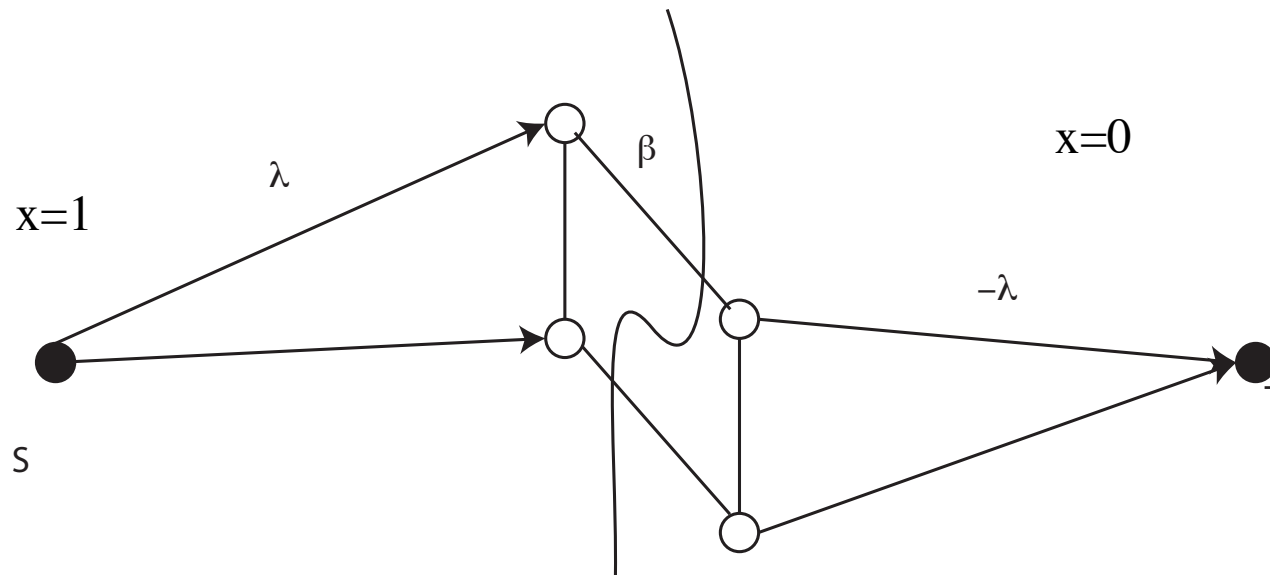
Consider a capacitated network comprising $n + 2$ vertices, being a source s , a sink t and the n pixels. There is a directed edge (s, i) from s to pixel i with capacity $c_{si} = \lambda_i$, if $\lambda_i > 0$; otherwise, there is a directed edge (i, t) from i to t with capacity $c_{it} = -\lambda_i$. 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 $x = (x_1, \dots, x_n)$ 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

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



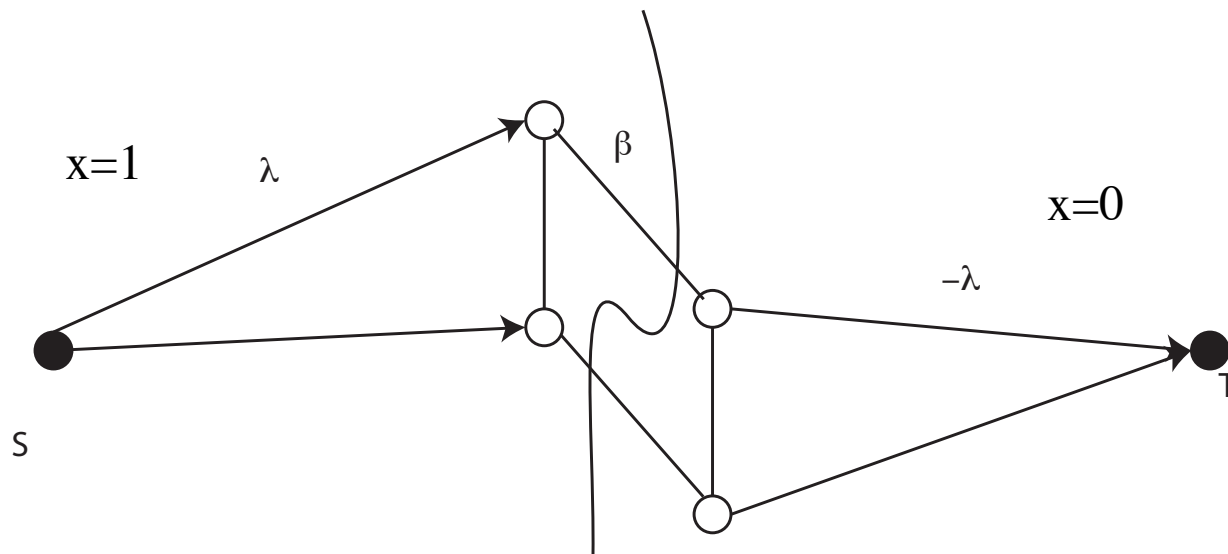
Graph cut (quick but clean)

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 $\mathbf{z} = (z_1, \dots, z_n, \dots, z_N)$ of grey values, indexed by the (single) index n . The segmentation of the image is expressed as an array of “opacity” values $\underline{\alpha} = (\alpha_1, \dots, \alpha_N)$ 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:

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

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:

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

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:

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

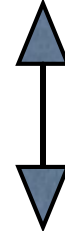
The smoothness term can be written as

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

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 $\text{dis}(\cdot)$ is the Euclidean distance of neighbouring pixels. This energy

Notice

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



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

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

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

An alternative strategy

- Variational inference
 - High level:
 - come up with simpler model that is “most like” intractable model
 - extract information from that
- Currently:
 - chose x_i (each 0 or 1) to maximize expression below

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

New setup

- H - hidden variables, 1 or -1
 - used to write x
- X - observations

$$\log p(H|X) = \sum_i \overset{\text{known}}{\downarrow} w_i H_i + \sum_{ij} \overset{\text{known}}{\downarrow} \theta_{ij} H_i H_j - \log Z$$

↑
Unknowable,
but not depending
on H

- We want to maximize this by choice of H
 - notice the -1, 1 trick

15.2 VARIATIONAL INFERENCE

We could just ignore intractable models, and stick to tractable models. This isn't a good idea, because intractable models are often quite natural. The discrete Markov random field model of an image is a fairly natural model. Image labels *should* depend on pixel values, and on neighboring labels. It is better to try and deal with the intractable model. One really successful strategy for doing so is to choose a tractable parametric family of probability models $Q(H; \theta)$, then adjust θ to find parameter values $\hat{\theta}$ that represent a distribution that is “close” in the right sense to $P(H|X)$. One then extracts information from $Q(H; \hat{\theta})$. This process is known as **variational inference**. What is remarkable is that (a) it is possible to find a $Q(H; \hat{\theta})$ without too much fuss and (b) information extracted from this distribution is often accurate and useful.

15.2.1 The KL Divergence

Assume we have two probability distributions $P(X)$ and $Q(X)$. A measure of their similarity is the **KL-divergence** (or sometimes **Kullback-Leibler divergence**) written

$$\mathbb{D}(P \parallel Q) = \int P(X) \log \frac{P(X)}{Q(X)} dX$$

(you've clearly got to be careful about zeros in P and Q here). This likely strikes you as an odd measure of similarity, because it isn't symmetric. It is not the case that $\mathbb{D}(P \parallel Q)$ is the same as $\mathbb{D}(Q \parallel P)$, which means you have to watch your P 's and Q 's. Furthermore, some work will demonstrate that it does not satisfy the triangle inequality, so KL divergence lacks two of the three important properties of a metric.

KL divergence has some nice properties, however. First, we have

$$\mathbb{D}(P \parallel Q) \geq 0$$

with equality only if P and Q are equal almost everywhere (i.e. except on a set of measure zero).

Remember this: *The KL divergence measures the similarity of two probability distributions. It is always non-negative, and is only zero if the two distributions are the same. However, it is not symmetric.*

KL divergence and Maximum likelihood

Second, there is a suggestive relationship between KL divergence and maximum likelihood. Assume that X_i are IID samples from some *unknown* $P(X)$, and we wish to fit a parametric model $Q(X|\theta)$ to these samples. This is the usual situation we deal with when we fit a model. Now write $H(P)$ for the entropy of $P(X)$, defined by

$$H(P) = - \int P(X) \log P(X) dx = -\mathbb{E}_P[\log P].$$

The distribution P is unknown, and so is its entropy, but it is a constant. Now we can write

$$\mathbb{D}(P \parallel Q) = \mathbb{E}_P[\log P] - \mathbb{E}_P[\log Q]$$

KL divergence and Maximum Likelihood

Then

$$\begin{aligned}\mathcal{L}(\theta) = \sum_i \log Q(X_i|\theta) &\approx \int P(X) \log Q(X|\theta) dX &= \mathbb{E}_{P(X)}[\log Q(X|\theta)] \\ & &= -H(P) - \mathbb{D}(P \parallel Q)(\theta).\end{aligned}$$

Equivalently, we can write

$$\mathcal{L}(\theta) + \mathbb{D}(P \parallel Q)(\theta) = -H(P).$$

Recall P doesn't change (though it's unknown), so $H(P)$ is also constant (though unknown). This means that when $\mathcal{L}(\theta)$ goes up, $\mathbb{D}(P \parallel Q)(\theta)$ must go down. When $\mathcal{L}(\theta)$ is at a maximum, $\mathbb{D}(P \parallel Q)(\theta)$ must be at a minimum. All this means that, when you choose θ to maximize the likelihood of some dataset given θ for a parametric family of models, you are choosing the model in that family with smallest KL divergence from the (unknown) $P(X)$.

15.2.2 The Variational Free Energy

We have a $P(H|X)$ that is hard to work with (usually because we can't evaluate $P(X)$) and we want to obtain a $Q(H)$ that is “close to” $P(H|X)$. A good choice of “close to” is to require that

$$\mathbb{D}(Q(H) \parallel P(H|X))$$

is small. Expand the expression for KL divergence, to get

$$\begin{aligned}\mathbb{D}(Q(H) \parallel P(H|X)) &= \mathbb{E}_Q[\log Q] - \mathbb{E}_Q[\log P(H|X)] \\ &= \mathbb{E}_Q[\log Q] - \mathbb{E}_Q[\log P(H, X)] + \mathbb{E}_Q[\log P(X)] \\ &= \mathbb{E}_Q[\log Q] - \mathbb{E}_Q[\log P(H, X)] + \log P(X)\end{aligned}$$

which at first glance may look unpromising, because we can't evaluate $P(X)$. But $\log P(X)$ is fixed (although unknown). Now rearrange to get

$$\begin{aligned}\log P(X) &= \mathbb{D}(Q(H) \parallel P(H|X)) - (\mathbb{E}_Q[\log Q] - \mathbb{E}_Q[\log P(H, X)]) \\ &= \mathbb{D}(Q(H) \parallel P(H|X)) - \mathbb{E}_Q.\end{aligned}$$

Here

$$\mathbb{E}_Q = (\mathbb{E}_Q[\log Q] - \mathbb{E}_Q[\log P(H, X)])$$

is referred to as the **variational free energy**. We can't evaluate $\mathbb{D}(Q(H) \parallel P(H|X))$. But, because $\log P(X)$ is fixed, when \mathbb{E}_Q goes down, $\mathbb{D}(Q(H) \parallel P(H|X))$ must go down too. Furthermore, a minimum of \mathbb{E}_Q will correspond to a minimum of $\mathbb{D}(Q(H) \parallel P(H|X))$. And we can evaluate \mathbb{E}_Q .

Variational Inference

We now have a strategy for building approximate $Q(H)$. We choose a family of approximating distributions. From that family, we obtain the $Q(H)$ that minimises E_Q (which will take some work). The result is the $Q(H)$ in the family that minimizes $\mathbb{D}(Q(H) \| P(H|X))$. We use that $Q(H)$ as our approximation to $P(H|X)$, and extract whatever information we want from $Q(H)$.

- Questions:
 - what $Q(H)$?
 - hard, case by case basis; essentially, so that calculations go through
 - How to minimize?
 - straightforward (long, dull) calculation
 - (AML Ch15 for easiest example)

Variational Inference

We want to construct a $Q(H)$ that approximates the posterior for a Boltzmann machine. We will choose $Q(H)$ to have one factor for each hidden variable, so $Q(H) = q_1(H_1)q_2(H_2)\dots q_N(H_N)$. We will then assume that all but one of the terms in Q are known, and adjust the remaining term. We will sweep through the terms doing this until nothing changes.

The i 'th factor in Q is a probability distribution over the two possible values of H_i , which are 1 and -1 . There is only one possible choice of distribution. Each q_i has one parameter $\pi_i = P(\{H_i = 1\})$. We have

$$q_i(H_i) = (\pi_i)^{\frac{(1+H_i)}{2}} (1 - \pi_i)^{\frac{(1-H_i)}{2}} .$$

Notice the trick; the power each term is raised to is either 1 or 0, and I have used this trick as a switch to turn on or off each term, depending on whether H_i is 1 or -1 . So $q_i(1) = \pi_i$ and $q_i(-1) = (1 - \pi_i)$. This is a standard, and quite useful, trick. We wish to minimize the variational free energy, which is

$$\mathbb{E}_Q = (\mathbb{E}_Q[\log Q] - \mathbb{E}_Q[\log P(H, X)]).$$