Note Title 4/22/2007

Algorithms: MaxFlow / Min-Cut & Belief Propagation.

(page 1)

These are two popular inference algorithms. They are often compared for benchmark problems like binocular stereo vision.

The lecture gives a brief introduction.

Formulate a vision problem as an optimization problem

Minimize an energy function

$$E(x_1, \dots, x_n) = \sum_{i,j} a_{ij} x_i x_j + \sum_i a_i x_i + c.$$

$x_i \in \{0, 1\}$.

Example: Kumar & Herbert

"building detection" $x_i = 1$, pixel i building
$x_i = 0$, pixel i background.

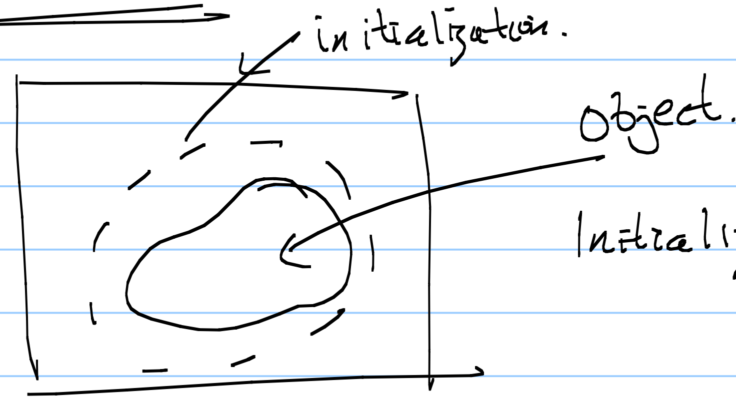
This is a discriminative random field

Input image $\{I_i : i = 1 \text{ to } n\}$,

$a_i = \phi(I)_i$ local evidence at pixel i.

$a_{ij} = \phi(I)_{ij}$

Grab Cut: Blake et al.



Initialize by hand.

Use the initialization to give the energy function.

Let $x_i = 1$, denote inside object.
$x_i = 0$, denote outside object.

Let ϕ be a filter, e.g. the colour.

Define distribution $p(\phi(I)_i \mid x_i = 1)$
$p(\phi(I)_i \mid x_i = 0)$

e.g. histogram distribution
Gaussian distribution

Learn the distributions from data inside the initialization and outside the initialization..

This is incorrect, there will be some contamination. But it may be a good enough approximation.

(page 3) Define the unary terms to be

$$\sum_i x_i \log \frac{p(\phi(I_i)|x_i=1)}{p(\phi(I_i)|x_i=0)}.$$

For the binary terms,

penalty

$$\lambda \sum_i \sum_{j \in N(i)} \{1-\delta_{x_i,x_j}\} F(I_i - I_j)$$

Pay a penalty if neighbouring points have different labels.

Make this penalty smaller if the intensity of the two pixels is very different.

For example: $F(I_i - I_j) = e^{-\gamma |I_i - I_j|}$

Full energy:

$$\sum_i x_i \log \frac{p(\phi(I_i)|x_i=1)}{p(\phi(I_i)|x_i=0)}$$

$$\lambda \sum_i \sum_{j \in N(i)} \{1-\delta_{x_i,x_j}\} F(I_i - I_j)$$

The energy functions in Corso's talk are also of this form - except no. of models > 2.

(page 4) **Back to Max-Flow / Min-Cut.**

Re-express $E(x_1, \ldots, x_n)$ as

$$E(x_1, \ldots x_n) = -\sum_{i,j} a_{ij} x_i (1-x_j) + \sum_{i,j} a_{ij} x_i + \sum_i a_i x_i + c$$

$$= \sum_{i,j} a_{ij}' x_i (1-x_j) + \sum_i a_i' x_i + c'$$

$$= \sum_{i,j} a_{ij}' x_i (1-x_j) + \sum_{i : a_i' > 0} a_i' x_i (1-x_S)$$

$$+ \sum_{i : a_i' < 0} |a_i'| (1-x_i) x_T + c''.$$

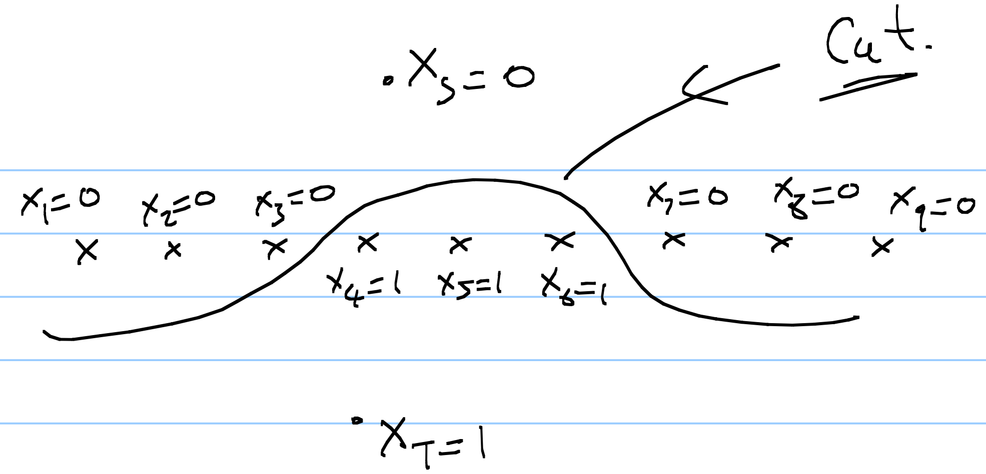
where $x_S = 0, x_T = 1$ are fixed nodes
S - source, T - sink.

The energy is now expressed in terms which are only non-zero if the corresponding nodes take different values

This defines a min-cut problem.

Split $\{x_i\}$ into two sets $X_0 = \{i : x_i = 0\}$
$X_1 = \{i : x_i = 1\}$

(page 5)



The only penalties are paid across the cut.

General Result → combinatorial optimization literature

There are algorithms which can find the global minimum (best cut) provided $a_{ij} \geq 0$, or $a_{ij} \leq 0$ (original formulation), for all i, j.

Note: this is different from the standard convexity criteria used to ensure that an energy function of continuous variables has a global minimum.

Convexity corresponds to the condition

$$\frac{\partial^2 E}{\partial x_i \partial x_j} \geq 0 \text{ as a matrix.}$$

In other words a_{ij} is a +ve definite matrix. This is impossible if $a_{ij} \leq 0, \forall i,j$

Hence the condition for min-cut is inconsistent with convexity.

(page 6) There are further results which ensure that min-cut algorithms can find the optimal solution for more general energy functions (e.g. Kolmogorov & Zabih, Freedman).

There is also an extension to cases where the $\{x_i\}$ can take several labels.

$$x_i \in \{1, 2, .., M\}$$

An α-expansion move (Kolmogorov & Zabih).
This is an operation that increases the no. of pixels with label α (and keep fixed, or decreases, all other labels).

<u>Algorithm</u>: Find the best α-expansion-move, If this has lower energy that current energy, then make the move.
Repeat for all α.
Stop when there is no possible α-expansion.

It can be proven that this procedure will convergence to a minimum within a multiplicative factor of the global minimum.

(page 7) Relationship of min-cut to the max-flow problem.

nodes. ↙ ↙ edges

Max Flow Graph $G=(V,E)$
Edge $(u,v)\in E$ have non-negative capacity $c(u,v)\geq 0$
Source s and sink T.

A Flow $f: V\times R\to R$ is required to obey the following constraints:

Capacity, $\forall u,v\in V$, require $f(u,v)\leq c(u,v)$
Skew-symmetry, $\forall u,v\in V$, require $f(u,v)=-f(v,u)$.
flow conservation $\forall u\in V-\{s,t\}$
require $\sum_{v\in V} f(u,v)=0$.

Total value of the flow f is $|f|=\sum_{v\in V} f(s,v)$.

A cut (S,T) is a partition on nodes V into sets S & $T=V-S$, with $s\in S$, $t\in T$. ?

The net flow across the cut (S,T) is $f(S,T)$
The capacity of the cut is $c(S,T)$.

(page 8) **Max-Flow & Min-Cut Theorem.**

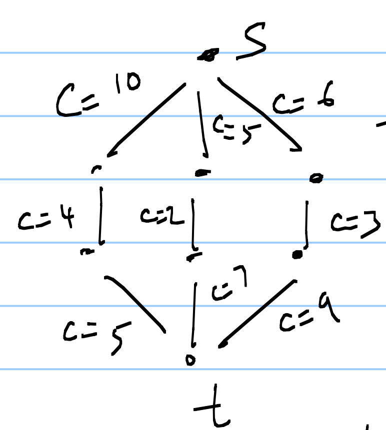
If f is a flow in the network (V,E) then f is a maximum flow if, and only if, $|f| = c(S,T)$ for some cut.

Note: the value of any flow is bounded above by the capacity of any cut.

Intuitively → the capacity is the restrictions in size of pipes that the water can flow down. The size of the pipes determines the maximum amount of flow.

1st level

2nd level



Typically a more complicated set of connections.

← capacity to 1st level is 21

capacity from 1st level to 2nd is 9.

capacity from 2nd to t is 21.

So maximum capacity is determined between levels 1 & 2. This is the position of the cut.

Max-Flow Algorithms

The relationship between Min-Cut and Max-Flow motivates the Ford-Fulkerson strategy:

Start with $f(u,v)=0, \ \forall u,v \in V$
(i.e. initial flow is zero)

At each iteration, increase the flow by finding an "augmenting path"
— a path from source to sink that has the capacity to take more flow.

The running time of the Ford-Fulkerson algorithm depends on how the augmenting path is chosen
— e.g. breadth first search.

<u>See</u> "Introduction to Algorithms"
Cormen, Leiserson, Rivest.

The Belief Propagation Algorithm.

BP is an alternative algorithm that we can use to perform inference on this type of task.

Recall relation between energy minimization and probabilistic inference.

Define: $P(x_1 \ldots x_n) = \frac{1}{Z[T]} e^{-E(x_1 \ldots x_n)/T}$

This can be re-expressed in the form:

$$P(\underline{x}) = \frac{1}{Z} \prod_{i,j} \psi_{ij}(x_i, x_j) \prod_i \psi_i(x_i)$$

(Because $E(\underline{x})$ has unary and binary terms only).

BP algorithm uses messages $m_{ij}(x_j)$
- message from node i to node j, when node j is in state x_j.

Message Update algorithm:

$$m_{ij}(x_j : t+1) = \sum_{x_i} \psi_{ij}(x_i, x_j) \psi_i(x_i) \prod_{k \neq j} m_{ki}(x_i : t)$$

Sum-product (Pearl)
max-product (Gallager) - replace $\sum_{x_i}$ by $\max_{x_i}$

(page 11) Run the message passing algorithm

This gives estimates of the unary and binary marginals.

$$p(x_i) = \sum_{\{x_j : j \neq i\}} P(\underline{x}) \quad \text{unary marginal.}$$

$$P(x_i, x_j) = \sum_{\{x_k : k \neq i, k \neq j\}} P(\underline{x}) \quad \text{binary marginal.}$$

The estimates given by BP are

$$b_i(x_i; t) \propto \psi_i(x_i) \prod_k m_{kj}(x_j : t)$$

$$b_{kj}(x_k, x_j; t) \propto \psi_k(x_k) \psi_j(x_j) \psi_{kj}(x_k, x_j) \prod_{r \neq j} m_{rk}(x_k; t) \prod_{l \neq k} m_{lj}(x_j : t)$$

If BP converges, it converges to a fixed point of the Bethe Free Energy:

$$F_b = \sum_{i,j} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{\psi_{ij}(x_i, x_j) \psi_i(x_i) \psi_j(x_j)}$$

$$- \sum_i (n_i - 1) \sum_{x_i} b_i(x_i) \log \frac{b_i(x_i)}{\psi_i(x_i)}.$$

where n_i is the number of nodes j connected to i.

(page 12) There a number of alternative algorithms, Wainwright et al, Teh & Welling, Yuille.
<u>Current</u> interesting work by Darwiche & Choi.

Wainwright's observation

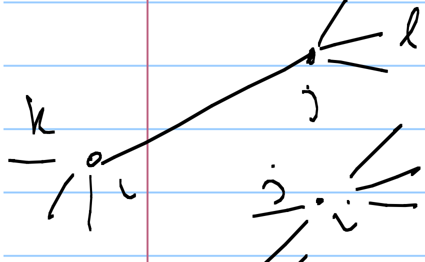
$$P(\underline{x}) = \prod_{i,j} \frac{\psi_{ij}(x_i,x_j)\,\psi_i(x_i)\,\psi_j(x_j)}{\prod_i \{\psi_i(x_i)\}^{n_i-1}}$$

$$= \prod_{ij} \frac{b_{ij}(x_i,x_j;t)}{\prod_i \{b_i(x_i:t)\}^{n_i-1}}.$$

<u>Hence</u> BP re-parameterizes the distribution from ψ's to b's.

Local estimates of probability.

$$B(x_i,x_j,x_{N(i,j)}:t) = \frac{1}{Z_{ij}}\, b_{ij}(x_i,x_j:t) \prod_{k\in N(i)/j} \frac{b_{ik}(x_i,x_k;t)}{b_i(x_i:t)} \prod_{l\in N(j)/i} \frac{b_{lj}(x_j,x_l)}{b_j(x_j:t)}$$



$$B(x_i,\underline{x}_{N(i)}:t) = \frac{1}{Z_i}\, b_i(x_i:t) \prod_{j\in N(i)} \frac{b_{ij}(x_i,x_j:t)}{b_i(x_i:t)}$$

Truncation the distribution — this is exact on a tree
No approximation — convergence guaranteed.

BP is repeated marginalization.

$$b_i(x_i : t+1) = \sum_{\underline{x}_{N(i)}} B(x_i, \underline{x}_{N(i)} : t)$$

$$b_{ij}(x_i, x_j : t+1) = \sum_{\underline{x}_{N(ij)}} B(x_i, x_j ; \underline{x}_{N(ij)} : t)$$

The fewer the number of closed loops, then the better the approximation.

BP can converge to a fixed point of Bethe or not converge to anything.

other algorithms, e.g. Yuille, will converge to a local minimum of the Bethe Free Energy (but require hidden loops that must converge).

It is also unclear whether minimization the Bethe Free energy is a good thing.