DYNAMIC PROGRAMMING VERSUS GRAPH CUT ALGORITHMS
FOR FITTING NON-PARAMETRIC MODELS TO IMAGE DATA

Chris Glasbey

Biomathematics and Statistics Scotland
QUESTION 1: WHAT IS IN THESE FIELDS?

log-transformed SAR

Airborne Synthetic Aperture Radar (SAR), East Anglia, UK
QUESTION 2: HOW FAT IS THIS SHEEP?

Ultrasound image of sheep’s back: Identify layer of fat
Compare data ($y$) with boundary templates ($\mu$)
Compare data ($y$) with boundary templates ($\mu$), to produce:

$$f_i(y, \beta_i) = \sum_{k=-K}^{K} \left( y_{\beta_i+k,i} - \mu_{k,i} \right)^2$$
Fit non-parametric model to find upper (lower) boundary:

$$\hat{\beta} = \arg\min_{\beta \in S} \left\{ \sum f_i(y, \beta_i) + \lambda \sum_{|i-j|=1} (\beta_i - \beta_j)^2 \right\}$$

Boundary is row $\beta_i$ in column $i$, $S = \{1, 2, \ldots, n\}$
Toy example:

One of approx. 700 possible connected paths

Both dynamic programming (DP) and graph cut (GC) algorithms can find path from left to right with globally minimum cost
PLAN

1. Dynamic programming (DP)
2. Graph cut algorithm (GC)
3. Application to SAR
4. Extension of GC
5. Summary
Minimum cost paths to column 2
Minimum cost paths to column 3
Minimum cost paths to all columns
Minimum cost in final column
Trace path back to first column
Minimum cost path
Result: Automatic and hand-drawn boundaries:

(Glasbey and Young, 2002, *Applied Statistics*)
What if image is 3D?

1D line

$\beta$: 1D array of 2D vectors

DP: YES

2D surface

$\beta$: 2D array of scalars

DP: NO
2. GRAPH CUT ALGORITHM

GC algorithm can also find minimum cost path, by formulating a network flow problem

There is efficient algorithm for finding max flow: source $\rightarrow$ sink

$\text{max flow} \equiv \text{min cut to disconnect source from sink}$

We construct a network for which min cut is the optimal path
First formulation of network
Max flow

Sink \((\sum \text{flow} = 6)\)

Source

Capacity

Flow

Max flow
source set of nodes & sink set of nodes

sink ($\sum$ flow $= 6$)

capacity

flow

$\infty$ $\infty$ $\infty$ $\infty$ $\infty$ $\infty$ $\infty$ $\infty$ $\infty$ $\infty$ $\infty$ $\infty$

[Diagram of a network flow problem with capacities and flows indicated on the edges and nodes.]
max flow \equiv min cut

\begin{align*}
\text{sink} \quad (\sum \text{flow} = 6) \\
\text{max flow} \equiv \text{min cut}
\end{align*}
max flow $\equiv$ min cut

sink ($\sum$ flow = 6)

min cut = 6

capacity

source

flow

max flow $\equiv$ min cut

$\Sigma$ flow = 6
For path smoothness we also need diagonal links.
Cut no longer min: ∞ cut source → sink!
Ford-Fulkerson alg: find path with spare capacity
Maximise flow along path

sink \ (\Sigma \ flow = 8)
Repeat until no paths have spare capacity
max flow \equiv \min \text{cut}

\[
\begin{align*}
\text{sink} & \quad (\sum \text{flow} = 10) \\
\text{source} & \\
\end{align*}
\]
For a general smoothness penalty $g$:

$$\hat{\beta} = \arg \min_{\beta \in S} \left\{ \sum_{i} f_i(y, \beta_i) + \sum_{|i-j|=1} g(|\beta_i - \beta_j|) \right\}$$

where $S = \{1, 2, \ldots, n\}$ and image size is $n \times P$

We construct a network with $(n + 1) \times P$ nodes, and edge capacities:

$$C\{(\text{source} \rightarrow (1, i))\} = \infty$$
$$C\{(n + 1, i) \rightarrow \text{sink}\} = \infty$$
$$C\{(\beta, i) \rightarrow (\beta + 1, i)\} = f_i(y, \beta)$$
$$C\{(\beta, i) \leftrightarrow (\beta, j) \mid |i-j| = 1\} = g(1)$$
$$C\{(\beta, i) \rightarrow (\beta - k, k) \mid |i-j| = 1\} = \{g(k+1) - 2g(k) + g(k-1)\} \quad \text{for } k \geq 1$$

All $C \geq 0$ provided $g$ is convex
In toy example  \( g(z) = \begin{cases} 0 & \text{if } |z| \leq 1 \\ \infty & \text{otherwise} \end{cases} \)

So edge capacities are:

\[
\begin{align*}
C\{(\text{source} \rightarrow (1, i))\} &= \infty \\
C\{(n + 1, i) \rightarrow \text{sink}\} &= \infty \\
C\{((\beta, i) \rightarrow (\beta + 1, i))\} &= f_i(y, \beta) \\
C\{((\beta, i) \leftrightarrow (\beta, j) \mid |i - j| = 1)\} &= 0 \\
C\{((\beta, i) \rightarrow (\beta - 1, j) \mid |i - j| = 1)\} &= \infty \\
C\{((\beta, i) \rightarrow (\beta - k, j) \mid |i - j| = 1)\} &= 0 \quad \text{for } k \geq 2
\end{align*}
\]
For ultrasound application $g(z) = \lambda z^2$

So edge capacities are:

$$
C\{\text{source} \rightarrow (1, i)\} = \infty
$$

$$
C\{(n + 1, i) \rightarrow \text{sink}\} = \infty
$$

$$
C\{(\beta, i) \rightarrow (\beta + 1, i)\} = f_i(y, \beta)
$$

$$
C\{(\beta, i) \leftrightarrow (\beta, j) \mid |i - j| = 1\} = \lambda
$$

$$
C\{(\beta, i) \rightarrow (\beta - k, j) \mid |i - j| = 1\} = 2\lambda \quad \text{for } k \geq 1
$$

Image $340 \times 400$ in size $\Rightarrow$ network has 136K nodes and 46M edges!

Result is same optimal path as DP
What if image is 3D?

1D line

$\beta$: 1D array of 2D vectors

DP: YES
GC: NO

2D surface

$\beta$: 2D array of scalars

DP: NO
GC: YES
3. APPLICATION TO SAR

Fit non-parametric model for SAR image restoration:

$$\hat{\beta} = \arg \min_{\beta \in S} \left\{ \sum_{i} f_{i}(y, \beta_{i}) + \lambda \sum_{\|i-j\|=1} |\beta_{i} - \beta_{j}| \right\}$$  \hspace{1cm} \text{where} \hspace{1cm} f_{i}(y, \beta_{i}) = (y_{i} - \beta_{i})^{2}$$

Equivalent to finding 2D surface in 3D: DP: NO, GC: YES
However, DP can be used as a 'building block' in an iterative algorithm to find a local optimum for

\[
\hat{\beta} = \arg \min_{\beta \in S} \left\{ \sum_{i} f_i(y, \beta_i) + \lambda \sum_{\|i-j\|=1} |\beta_i - \beta_j| \right\}
\]

Iterated DP (Leung et al, 2004, *BMVC conference*)

0) Initialise $\beta$

1) For each column in turn, use DP to estimate $\beta$ in that column, given current values of all other $\beta$’s

2) Apply (1) to rows

3) Repeat (1) and (2) until convergence
Image size $P = 250 \times 250$, $S = \{65, 75, \ldots, 165\}$, $\lambda = 50$

Starting from $\beta = y$, iterated DP converged in 26 iterations, taking 1.2sec CPU to find local minimum of $388P$
For SAR application, unlike DP, GC can be used unmodified

As the smoothness penalty is $g(z) = \lambda |z|

Edge capacities are:

\[
\begin{align*}
C\{\text{source} \rightarrow (1, i)\} &= \infty \\
C\{(n+1, i) \rightarrow \text{sink}\} &= \infty \\
C\{(\beta, i) \rightarrow (\beta + 1, i)\} &= f_i(y, \beta) \\
C\{(\beta, i) \leftrightarrow (\beta, j) \mid \|i - j\| = 1\} &= \lambda \\
C\{(\beta, i) \rightarrow (\beta - k, j) \mid \|i - j\| = 1\} &= 0 \quad \text{for } k \geq 1
\end{align*}
\]

⇒ network has 750K nodes, 4.6M edges
Iterated DP took 1.2sec CPU to find local minimum of $388P$

GC took 1.1sec CPU with one implementation (21min with another) to find global minimum of $374P$
4. EXTENSION OF GC

If the smoothness penalty $g$ is not convex, such as the indicator function:

$$g(z) = \begin{cases} 
0 & \text{if } z = 0 \\
\lambda & \text{otherwise}
\end{cases}$$

GC cannot be used to find the global optimum, because

$$C\{(\beta, i) \rightarrow (\beta - 1, j) \mid \|i - j\| = 1\} = \{g(2) - 2g(1) + g(0)\} = -\lambda$$

However GC can still sometimes find local optima by iterative search.
GC swap algorithm (Boykov et al, 2001, *IEEE PAMI*)

0) Initialise $\beta$

1) For every pair of values in $S$, say $(\alpha, \gamma)$

use GC to optimise swapping any $\beta_i = \alpha \rightarrow \gamma$ and any $\beta_i = \gamma \rightarrow \alpha$

Using a network which only includes nodes for which $\beta_i = \alpha, \gamma$

with edge capacities:

$$C\{\text{source} \rightarrow i\} = f_i(y, \alpha) + \sum_{\{j: \|i-j\|=1, \beta_j \neq \alpha, \gamma\}} g(\|\alpha - \beta_j\|)$$

$$C\{i \rightarrow \text{sink}\} = f_i(y, \gamma) + \sum_{\{j: \|i-j\|=1, \beta_j \neq \alpha, \gamma\}} g(\|\gamma - \beta_j\|)$$

$$C\{i \leftrightarrow j \mid \|i-j\|=1\} = g(\|\alpha - \gamma\|)$$

2) Repeat (1) until no further improvement
Initial path

Network for $\beta : \ 2 \leftrightarrow 3$
Initial path

Network for $\beta$: $2 \leftrightarrow 3$
Initial path

Network for $\beta$: $2 \leftrightarrow 3$
Initial path

Network for $\beta: \ 2 \leftrightarrow 3$
Improved path

Network for $\beta : 2 \leftrightarrow 3$
For SAR application with non-convex $g$:

$$\hat{\beta} = \arg \min_{\beta \in S} \left\{ \sum_i (y_i - \beta_i)^2 + \sum_{\|i-j\|=1} g(\|\beta_i - \beta_j\|) \right\}$$

where $g(z) = \begin{cases} 0 & \text{if } z = 0 \\ 2000 & \text{otherwise} \end{cases}$

The GC swap algorithm converged in 5-9 iterations from $\beta = y$ or 1000 random starts, to minimised cost between 430-435$P$

In comparison, starting from $\beta = y$, Iterated DP converged to 515$P$

and best found using simulated annealing type methods was 438$P$
5. SUMMARY: DP v GC

In many cases, image segmentation, warping and restoration can be formulated as non-parametric modelling:

\[
\hat{\beta} = \arg \min_{\beta \in S} \left\{ \sum_{i} f_i(y, \beta_i) + \sum_{\|i-j\|=1} g(\|\beta_i - \beta_j\|) \right\}
\]

\(\beta\): an \(I\)-dimensional array of \(B\)-dimensional vectors

Global optimum can be found by:

- Dynamic programming if \(I = 1\)
- Graph cut algorithm if \(B = 1\) and \(g\) convex

Otherwise, local optima may be found by:

- Iterated DP
- GC swap algorithm
However, GC is a much more complicated algorithm than DP

- Which makes it less flexible to adapt to new problems
- Or embed in local optimisers when conditions for global optimality are not met
- GC cannot handle MRF priors with higher-order neighbourhoods
- Different versions of the GC algorithm run at very different speeds

For further details, see Glasbey *(IWSM, 2010)*
http://www.bioss.sari.ac.uk/staff/chris.html