- Probability methods we've covered
- Graph partitioning method
- Mumford-Shah functional approach
- Normalized cuts

1 Probability methods we've covered

Both maximum likelihood and MRF start with local measurements and then solve the image segmentation problem globally.

But the maximum likelihood technique is flawed because images are not made of constant-brightness patches. Also, maximum likelihood is not guaranteed to find the globally best solution; it may return a locally optimum segmentation.

A Bayesian approach using Markov random fields gives good solutions, but the configuration space is huge: $\#pixels \cdot \#brightness\ levels$. Searching for an exact answer in this space is impractical. We are still exploring methods to approximate the solution.

Besides working with brightnesses, we can also segment images by looking at their texture components. Using texture information in any of the algorithms drives up the complexity, though. Incorporating motion information also adds to the complexity (see lecture 8).

2 Graph partitioning method

2.1 Energy functions and probability distributions

We can treat image segmentation as a graph partitioning problem. This appears to be a different setup than the probabilistic approaches, but in fact we can represent any graph partition as a probability distribution.

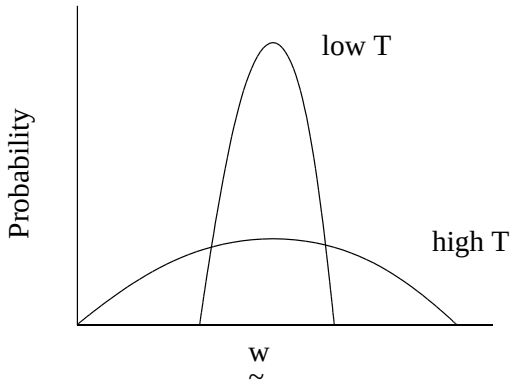

Figure 1: Effects on probability distribution by changing temperature

Our solutions will be minimizations of an energy function E. Gibbs distribution gives the probability of a particular E as $\frac{1}{Z}e^{-E}$. Different configurations $\underset{\sim}{\omega}$ have probabilities $\frac{1}{Z}e^{-E(\underset{\sim}{\omega})}$ where Z is used to scale the sum of all the probabilities to 1.

Our problem is to find the $\underset{\sim}{\omega}$ with the MAP (maximum *a postereori*) estimate. This $\underset{\sim}{\omega}$ will have the smallest $E(\underset{\sim}{\omega})$.

2.2 Temperature

We can skew the differences between energy functions with a temperature factor: $\frac{1}{Z}e^{\frac{-E(\underset{\sim}{\omega})}{T}}$. A large T value makes small differences less significant. As the temperature cools (T goes to 0), small differences in energies are magnified. The Geman and Geman paper describes this more (see pp. 721 and 730).

3 Mumford-Shah functional approach

The Mumford-Shah approach specifies a function to minimize in order to find the best noise-free image and the locations of edges between segments.

Blake-Zisserman provides an algorithm for the minimization.

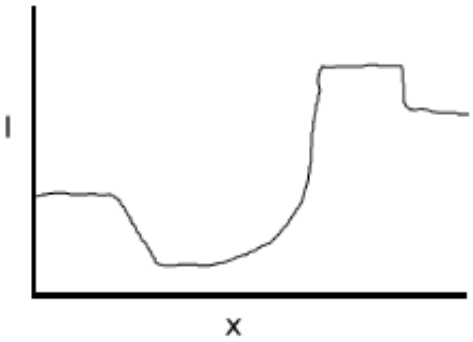

Figure 2: Image I

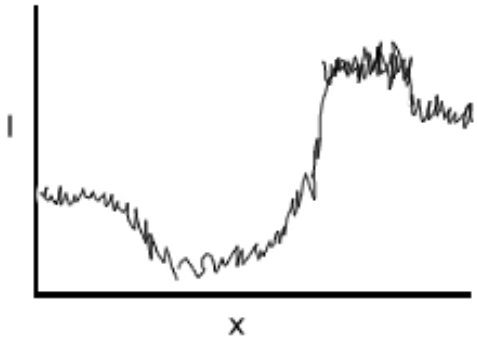

Figure 3: Image $\tilde{I}$

3.1 1-D setup

Image I has 3 edges in it.

$\tilde{I}$ shall be a noisy version of I.

We could run one of the edge-detection algorithms on I, but we would like to solve the problem globally.

3.2 Minimization problem

Our goal is to simultaneously find I and the set of edge locations $\{s\}$ by minimizing the following:

$$\lambda_1 \int (I - \tilde{I})^2 + \lambda_2 \int_{D \setminus \{s\}} |\nabla I|^2 + \lambda_3 \#\{s\}$$

The first term gives weight to solution images that are similar to the noisy image. Without this, a field of only one brightness and no edges, for example, could be returned as an optimal solution.

The second term minimizes the brightness gradient between all pixels in the domain, D, except where there are edges given in $\{s\}$.

The third term adds a penalty to bigger numbers of edges. This prevents cases such as all pixel boudaries being listed as edges from being minima.

The λ_i must be adjusted for the type of image. For example, if the source image is known to be noisy, λ_1 should be set to a small value because we know that the real image I is especially different from $\tilde{I}$.

3.3 2-D

The 2-D version of the minimization problem looks the same, except the integrals are double integrals. Also, instead of counting the edges in the third term, we use the sum of the lengths of the edge curves. It is noted that this representation still does not account for coherence of edges.

4 Normalized cuts

Normalized cuts is a counterpart to the Mumford-Shah function that uses graph theory to find groups of related pixels in an image.

The following part of the lecture is presented in the 1997 CVPR paper by Shi and Malik.

4.1 Graph partitioning problem

Take the graph $G = (V, E)$. The nodes V shall be pixels, and the edges E shall have weights ω_{ij} that describe the relatedness of pixels i and j in terms of the desired image segmentation. If we were doing a segmentation based on

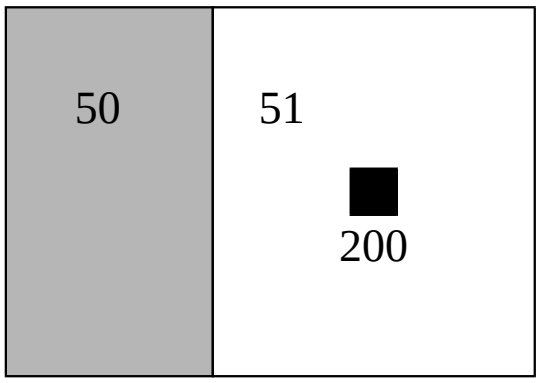

Figure 4: Effects on probability distribution by changing temperature

brightness, ω would be large for edges between pixels of similar brightness and small otherwise. An example weight function for this case would be $\omega_{ij} = e^{\frac{-(I_i - I_j)^2}{\sigma^2}}$. $\omega = 1$ for $I_i = I_j$ and falls off for differing I_i and I_j.

To get a segmentation from a graph, we need to produce a number of sets of nodes that list the pixels in each segment. The sum of the edge weights that span different sets of nodes is called a cut of the graph. Clearly, the weights of edges that connect between groups will be small, and edge weights within a group will be large.

Cutting the graph is the counterpart to the Mumford-Shah technique. Here, we want to globally minimize the graph cut, which can be easier than minimizing the Mumford-Shah equation.

$$cut(A, B) = \sum_{u \in A, v \in B} W(u, v), with A \cap B = \emptyset$$

$$assoc(A, A') = \sum_{u \in A, v \in A'} W(u, v), where\ A\ and\ A'\ not\ necessarily\ disjoint$$

4.2 Minimum cut (not the right answer)

Unfortunately, minimum cut (which can be solved in polynomial time) will not give the right answer.

In this image, the outlier has tiny connections to its neighbors, so minimum cut says we should cut that one pixel out. Really, we want the edge

between the brightness-50 and brightness-51 areas. The 200 pixel should be ignored.

outlier figure

4.3 Normalized cut

Normalized cut uses ratios of edge weight totals instead of the raw totals. It favors regions with strongly connected interiors.

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

For more than two regions, normalized cut extends like this:

$$Ncut(A,B,C\ldots) = \frac{cut(A,\bar{A})}{assoc(A,V)} + \frac{cut(B,\bar{B})}{assoc(B,V)} + \frac{cut(C,\bar{C})}{assoc(C,V)} + \ldots$$

And, as described at the beginning of this lecture, e^{-Ncut} can be interpreted as a probability distribution for the particular cut.

4.4 Minimizing the normalized cut

Minimizing the normalized cut for a graph is NP-hard. But we can use an approximation from spectral graph theory.

Set up the matrices W and D where $W(i,j) = \omega_{ij}$ and $D(i,i) = \sum_j W(i,j)$ (D is a diagonal matrix of the row sums of W). Though we skip most of the math here:

$$Ncut(A,B) = \frac{y^{\mathrm{T}}(D-W)y}{y^{\mathrm{T}}Dy} \; with \; y_i \in \{1,-b\}, y^{\mathrm{T}} \neq 0$$

This is a generalized eigenvalue problem of the form $Ax = \lambda Bx$. Specifically, $(D-W)y = \lambda Dy$.

When we solve, the answer elements may not be exactly 1 and $-b$. That's the approximation.

Note that W and D are huge. For a 100x100 pixel image, W is 10,000 on a side, so W and D are each 100,000,000 elements. However, many elements are zero, so we can use a special techinque for sparse matrices by Lanczos.