Segmentation
Overview of today’s lecture

- Finish temporal models.
- Normalized cuts.
- Boundaries.
- Clustering for segmentation.
Most of these slides were adapted from:

- Srinivasa Narasimhan (16-385, Spring 2015).
- James Hays (Brown University).
Image segmentation by pairwise similarities

- Image = \{ pixels \}
- Segmentation = partition of image into segments
- Similarity between pixels i and j
  \[ S_{ij} = S_{ji} \geq 0 \]

- Objective: “similar pixels, with large value of \( S_{ij} \), should be in the same segment, dissimilar pixels should be in different segments”
Relational Graphs

- \( G = (V, E, S) \)
  - \( V \): each node denotes a pixel
  - \( E \): each edge denotes a pixel-pixel relationship
  - \( S \): each edge weight measures pairwise similarity

- Segmentation = node partitioning
  - break \( V \) into disjoint sets \( V_1, V_2 \)
Weighted graph partitioning

Pixels $i \in I$ = vertices of graph $G$
Edges $ij = \text{pixel pairs with } S_{ij} > 0$

Similarity matrix $S = [S_{ij}]$
$\deg_i = \sum_{j \in G} S_{ij}$ degree of $I$

$\deg A = \sum_{i \in A} \deg_i$ degree of $A \subseteq G$

$\text{Assoc}(A,B) = \sum_{i \in A} \sum_{j \in B} S_{ij}$
Cuts in a Graph

• (edge) cut = set of edges whose removal makes a graph disconnected

• weight of a cut: \( \text{cut}(A, B) = \sum_{i \in A} \sum_{j \in B} S_{ij} = \text{Assoc}(A, B) \)

• the normalized cut

\[
\text{NCut}(A, B) = \text{cut}(A, B) \left( \frac{1}{\deg A} + \frac{1}{\deg B} \right)
\]

• Normalized Cut criteria: minimum \( \text{cut}(A, \bar{A}) \)

\[
S_{ij} = \frac{1}{d(x_i, x_j)}
\]
Grouping with Spectral Graph Partitioning

SGP: data structure = a weighted graph, weights describing data affinity

\[
\min \, Ncut(A, B) = \frac{cut(A, B)}{\deg(A)} + \frac{cut(A, B)}{\deg(B)}
\]

\[
cut(A, B) = \sum_{i \in A} \sum_{j \in B} S(i, j)
\]

\[
\deg(A) = \sum_{i \in A} \sum_{j \in G} S(i, j)
\]

Segmentation is to find a node partitioning of a relational graph, with minimum total cut-off affinity.

Discriminative models are used to evaluate the weights between nodes.

The solution sought is the cuts of the minimum energy.

NP-Hard!
Matrix representation of the graph problem:

\[
M = \begin{bmatrix}
0 & 2 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 2 & 2 & 0 & 0 & 0 & 0.1 & 0 \\
5 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 2 & 0 & 0.1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.1 & 0 & 1 & 0 & 0 & 3 \\
0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 4 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 7 \\
0 & 0.1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 3 & 4 & 7 & 1 & 0
\end{bmatrix}
\]

affinity matrix
Eigenvector approach to segmentation

Represent a connected component (or cluster $C$)
By a weight vector $\mathbf{w}$ such that (indicator vector):

$$w_i = \begin{cases} 
1 & \text{if } i \in C \\
0 & \text{if } i \notin C
\end{cases}$$

$\mathbf{w}^t \mathbf{M} \mathbf{w}$ is the association of $C$ because:

$$\mathbf{w}^t \mathbf{M} \mathbf{w} = \sum_{i,j \in C} m_{ij}$$

If $C$ is a good cluster, then the average association between features In $C$ should be large. Therefore, we want:

$\mathbf{w}^t \mathbf{M} \mathbf{w}$ is large

Suggests algorithm:
- Build matrix $\mathbf{M}$
- Find $\mathbf{w}$ such that $\mathbf{w}^t \mathbf{M} \mathbf{w}$ is maximum.

Problem: $\mathbf{w}$ is a binary vector
Replace binary vector with continuous weight vector. Interpretation: 
\( w_i \) large if \( i \) belongs to \( C \).

Problem becomes:
• Find \( w \) such that \( w^t M w \) is maximum
• Construct the corresponding component \( C \) by:
  \( i \) belongs to \( C \) if \( w_i \) is large.

Problem with scale:
The relative values of the \( w_i \)'s are important, the total magnitude of \( w \) is not.
Normalization:
\[
\max_w \frac{w^t M w}{w^t w}
\]
Replace binary vector with continuous weight vector. Interpretation:

\[ w_i \text{ large if } i \text{ belongs to } C. \]

Problem becomes:

- Find \( w \) such that \( w^t M w \) is maximum
- Construct the corresponding component \( C \) by:
  \[ i \text{ belongs to } C \text{ if } w_i \text{ is large.} \]

Problem with scale:
The relative values of the \( w_i \)'s are important, the total magnitude of \( w \) is not.

Normalization:

\[
\begin{align*}
\text{Max} & \quad \frac{w^t M w}{w^t w}
\end{align*}
\]

**Rayleigh’s ratio theorem:**

Given a symmetric matrix \( M \), the maximum of the ratio

\[
\frac{w^t M w}{w^t w}
\]

is obtained for the eigenvector \( w_{\text{max}} \) corresponding to the largest eigenvalue \( \lambda_{\text{max}} \) of \( M \).
Brightness Image Segmentation
Brightness Image Segmentation
Results on color segmentation
Segmentation from boundaries
Intuition:
• Duality between regions and boundaries
• Maybe “easier” to estimate boundaries first
• Then use the boundaries to generate segmentations at different levels of granularity


Complete package: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html
Finding boundaries

- $P_b =$ Probability of boundary
- $\chi^2$ difference between histograms at different orientations $\rightarrow$ Classifier $\rightarrow$ Probability of boundary
Combining multiple cues
gPb (global Pb)

- Idea: We could use the Pb contours to generate an affinity matrix and then use Ncuts
- j and i have lower affinity because they cross higher Pb values
**gPb (global Pb)**

Not good: generate segmentation from the eigenvectors

Good: Combine the gradients of the eigenvectors!!
Final step: Convert closed boundary (UCM = Ultrametric Contour Map)
Different thresholds on contours yield segmentations at different levels of granularity
Guaranteed to produce a hierarchical segmentation
Intuition:
- Duality between regions and boundaries
- Maybe “easier” to estimate boundaries first
- Then use the boundaries to generate segmentations at different levels of granularity

Complete package:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html
Clustering: group together similar points and represent them with a single token

Key Challenges:
1) What makes two points/images/patches similar?
2) How do we compute an overall grouping from pairwise similarities?
Why do we cluster?

- **Summarizing data**
  - Look at large amounts of data
  - Patch-based compression or denoising
  - Represent a large continuous vector with the cluster number

- **Counting**
  - Histograms of texture, color, SIFT vectors

- **Segmentation**
  - Separate the image into different regions

- **Prediction**
  - Images in the same cluster may have the same labels
How do we cluster?

• K-means
  – Iteratively re-assign points to the nearest cluster center

• Agglomerative clustering
  – Start with each point as its own cluster and iteratively merge the closest clusters

• Mean-shift clustering
  – Estimate modes of pdf
K-means clustering
1. Select initial centroids at random
1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.
1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.

3. Compute each centroid as the mean of the objects assigned to it (go to 2)
1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.

3. Compute each centroid as the mean of the objects assigned to it (go to 2)
1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.

3. Compute each centroid as the mean of the objects assigned to it (go to 2)

Repeat previous 2 steps until no change
K-means Clustering

Given k:

1. Select initial centroids at random.

2. Assign each object to the cluster with the nearest centroid.

3. Compute each centroid as the mean of the objects assigned to it.

4. Repeat previous 2 steps until no change.
K-means: design choices

• Initialization
  – Randomly select K points as initial cluster center
  – Or greedily choose K points to minimize residual

• Distance measures
  – Traditionally Euclidean, could be others

• Optimization
  – Will converge to a *local minimum*
  – May want to perform multiple restarts
K-means clustering using intensity or color
How to choose the number of clusters?

- Minimum Description Length (MDL) principal for model comparison

- Minimize Schwarz Criterion
  - also called Bayes Information Criteria (BIC)

\[
\text{Distortion} + \lambda \, (\#\text{parameters}) \log R
\]

\[
= \text{Distortion} + \lambda m k \log R
\]

- \(m = \#\text{dimensions}\)
- \(k = \#\text{Centers}\)
- \(R = \#\text{Records}\)
K-Means pros and cons

- **Pros**
  - Finds cluster centers that minimize conditional variance (good representation of data)
  - Simple and fast*
  - Easy to implement

- **Cons**
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - All clusters have the same parameters (e.g., distance measure is non-adaptive)
  - *Can be slow: each iteration is $O(KNd)$ for $N$ $d$-dimensional points

- **Usage**
  - Rarely used for pixel segmentation
Agglomerative clustering

1. Say “Every point is its own cluster”
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Agglomerative clustering

How to define cluster similarity?
- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids

How many clusters?
- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges
Conclusions: Agglomerative Clustering

Good
• Simple to implement, widespread application
• Clusters have adaptive shapes
• Provides a hierarchy of clusters

Bad
• May have imbalanced clusters
• Still have to choose number of clusters or threshold
• Need to use an “ultrametric” to get a meaningful hierarchy
Mean shift segmentation


• Versatile technique for clustering-based segmentation
Mean shift algorithm

- Try to find *modes* of this non-parametric density
Mean Shift Algorithm

A ‘mode seeking’ algorithm
Fukunaga & Hostetler (1975)
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Find the region of highest density
Mean Shift Algorithm
A ‘mode seeking’ algorithm
Fukunaga & Hostetler (1975)

Pick a point
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Draw a window
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the (weighted) mean

[Diagram with data points and a circle highlighting a cluster]
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**
Mean Shift Algorithm

A ‘mode seeking’ algorithm
Fukunaga & Hostetler (1975)

Shift the window
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the mean
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

**Shift** the window
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the mean
Mean Shift Algorithm
A ‘mode seeking’ algorithm
Fukunaga & Hostetler (1975)

**Shift** the window
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the mean
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window
Mean Shift Algorithm

A ‘mode seeking’ algorithm
Fukunaga & Hostetler (1975)

Compute the mean
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

- Shift the window

Mean Shift Algorithm
A ‘mode seeking’ algorithm
Fukunaga & Hostetler (1975)

Compute the mean
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the mean
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

To understand the theory behind this we need to understand…

Shift the window
Attraction basin

- **Attraction basin**: the region for which all trajectories lead to the same mode
- **Cluster**: all data points in the attraction basin of a mode
Attraction basin
Mean shift clustering

• The mean shift algorithm seeks *modes* of the given set of points

1. Choose kernel and bandwidth

2. For each point:
   a) Center a window on that point
   b) Compute the mean of the data in the search window
   c) Center the search window at the new mean location
   d) Repeat (b,c) until convergence

3. Assign points that lead to nearby modes to the same cluster
Segmentation by Mean Shift

- Compute features for each pixel (color, gradients, texture, etc)
- Set kernel size for features $K_f$ and position $K_s$
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that are within width of $K_f$ and $K_s$
Mean shift segmentation results

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html
Mean-shift: other issues

• Speedups
  – Binned estimation
  – Fast search of neighbors
  – Update each window in each iteration (faster convergence)

• Other tricks
  – Use kNN to determine window sizes adaptively

• Lots of theoretical support
References

Basic reading:
• Szeliski, Sections 5.2, 5.3, 5.4, 5.5.