Grouping
What is grouping?
K-means

Input: set of data points, k

1. Randomly pick k points as means

2. For i in [0, maxiters]:
   1. Assign each point to nearest center
   2. Re-estimate each center as mean of points assigned to it
K-means - the math

Input: set of data points $X$, $k$

1. Randomly pick $k$ points as means $\mu_i, i = 1, \ldots, k$

2. For iteration in $[0, \text{maxiters}]$:
   1. Assign each point to nearest center
      \[ y_i = \arg \min_j ||x_i - \mu_j||^2 \]
   2. Re-estimate each center as mean of points assigned to it
      \[ \mu_j = \frac{\sum_{i:y_i=j} x_i}{\sum_{i:y_i=j} 1} \]
K-means - the math

• An objective function that must be minimized:

$$\min_{\mu, y} \sum_{i} \| x_i - \mu y_i \|^2$$

• Every iteration of k-means takes a downward step:
  • Fixes $\mu$ and sets $y$ to minimize objective
  • Fixes $y$ and sets $\mu$ to minimize objective
K-means on image pixels
K-means on image pixels

Picture courtesy David Forsyth

One of the clusters from k-means
K-means on image pixels

• **What is wrong?**
• **Pixel position**
  • Nearby pixels are likely to belong to the same object
  • Far-away pixels are likely to belong to different objects
• **How do we incorporate pixel position?**
  • Instead of representing each pixel as \((r,g,b)\)
  • Represent each pixel as \((r,g,b,x,y)\)
K-means on image pixels
The issues with k-means

- Captures pixel similarity but
  - Doesn’t capture continuity
  - Captures proximity only weakly
  - Can merge far away objects together
- Requires knowledge of k!
Oversegmentation and superpixels

• We don’t know k. What is a safe choice?
• Idea: Use large k
  • Can potentially break big objects, but will hopefully not merge unrelated objects
  • Later processing can decide which groups to merge
  • Called *superpixels*
Regions ↔ Boundaries
Does Canny always work?
The aperture problem
The aperture problem
“Globalisation”
Images as graphs

• Each pixel is node
• Edge between “similar pixels”
  • *Proximity*: nearby pixels are more similar
  • *Similarity*: pixels with similar color are more similar
• Weight of edge = similarity
Segmentation is graph partitioning
Segmentation is graph partitioning

- Every partition “cuts” some edges
- Idea: minimize total weight of edges cut!
Criterion: Min-cut?

- Min-cut carves out small isolated parts of the graph
- In image segmentation: individual pixels
Normalized cuts

- “Cut” = total weight of cut edges
- Small cut means the groups don’t “like” each other
- But need to normalize w.r.t how much they like themselves
- “Volume” of a subgraph = total weight of edges within the subgraph
Normalized cut

\[ \frac{\text{cut}(A, \bar{A})}{\text{vol}(A)} + \frac{\text{cut}(A, \bar{A})}{\text{vol}(\bar{A})} \]
Min-cut vs normalized cut

• Both rely on interpreting images as graphs
• By itself, min-cut gives small isolated pixels
  • But can work if we add other constraints
• min-cut can be solved in polynomial time
  • Dual of max-flow
• N-cut is NP-hard
  • But approximations exist!
Random walk
Random walk
Random walk
Random walk

- Given that ghosts inhabit set A, how likely are they to stay in A?
Random walk

- Given that ghosts inhabit set A, how likely are they to stay in A?
Random walk

- Given that ghosts inhabit set A, how likely are they to stay in A?
Random walk

- Given that ghosts inhabit set A, how likely are they to stay in A?
Random walk

• Key idea: Partition should be such that ghost should be likely to stay in one partition
• Normalized cut criterion is the same as this
• But how do we find this partition?
Graphs and matrices

- $w(i,j) = \text{weight between } i \text{ and } j$ \textit{(Affinity matrix)}
- $d(i) = \text{degree of } i = \sum_j w(i, j)$
- $D = \text{diagonal matrix with } d(i) \text{ on diagonal}$
Graphs and matrices
Graphs and matrices

$$E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}}$$

$$E = D^{-1}W$$
Graphs and matrices

• How do we represent a clustering?
• A label for N nodes
  • 1 if part of cluster A, 0 otherwise
• An N-dimensional vector!

\[
v_1 = \begin{bmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Graphs and matrices

• How do we represent a clustering?
• A label for N nodes
  • 1 if part of cluster A, 0 otherwise
• An N-dimensional vector!

<table>
<thead>
<tr>
<th></th>
<th>$v_1$</th>
<th>$v_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Graphs and matrices

• How do we represent a clustering?
• A label for N nodes
  • 1 if part of cluster A, 0 otherwise
• An N-dimensional vector!

<table>
<thead>
<tr>
<th></th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Graphs and matrices

\[ E = D^{-1}W \]
Graphs and matrices

\[ E = D^{-1}W \]

\[ E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}} \]
Graphs and matrices

\[ E = D^{-1}W \]

\[ E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}} \]
Graphs and matrices

\[ E = D^{-1}W \]

\[ E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}} \]

<table>
<thead>
<tr>
<th></th>
<th>( v_3 )</th>
<th>( Ev_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Graphs and matrices

E = D⁻¹W
Graphs and matrices

\[ \mathbf{E} = \mathbf{D}^{-1} \mathbf{W} \]

\[ E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}} \]
Graphs and matrices

\[
D^{-1} W y \simeq y
\]

Define z so that \( y = D^{-\frac{1}{2}} z \)

\[
D^{-1} W D^{-\frac{1}{2}} z \simeq D^{-\frac{1}{2}} z
\]

\[
\Rightarrow D^{-\frac{1}{2}} W D^{-\frac{1}{2}} z \simeq z
\]

\[
\Rightarrow (I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}) z \simeq 0
\]
Graphs and matrices

\[ (I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}})z \approx 0 \]

\[ \Rightarrow \mathcal{L}z \approx 0 \]

\[ \mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \]

is called the

Normalized Graph Laplacian
Graphs and matrices

\[ \mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \]

• We want \( \mathcal{L}z \approx 0 \)
• Trivial solution: all nodes of graph in one cluster, nothing in the other
• To avoid trivial solution, look for the eigenvector with the second smallest eigenvalue

\[ \mathcal{L}z = \lambda z \]
\[ \lambda_1 < \lambda_2 < \ldots < \lambda_N \]
• Find \( z \) s.t. \( \mathcal{L}z = \lambda_2 z \)
Normalized cuts

• Approximate solution to normalized cuts
• Construct matrix $W$ and $D$
• Construct normalized graph laplacian
  \[ \mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \]
• Look for the second smallest eigenvector
  \[ \mathcal{L}z = \lambda_2 z \]
• Compute \( y = D^{-\frac{1}{2}} z \)
• **Threshold \( y \) to get clusters**
  • Ideally, sweep threshold to get lowest $N$-cut value
More than 2 clusters

- Given graph, use N-cuts to get 2 clusters
- Each cluster is a graph
  - Re-run N-cuts on each graph
Normalized cuts

• NP Hard

• But approximation using *eigenvector of normalized graph laplacian*
  • Smallest eigenvector: trivial solution
  • *Second smallest eigenvector: good partition*
  • *Other eigenvectors: other partitions*

• An instance of “Spectral clustering”
  • Spectrum = set of eigenvalues
  • Spectral clustering = clustering using eigenvectors of (various versions of) graph laplacian
Images as graphs

- Each pixel is a node
- What is the edge weight between two nodes / pixels?
  - $F(i)$: intensity / color of pixel $i$
  - $X(i)$: position of pixel $i$

$$
\omega_{ij} = e^{\frac{-\|F(i)-F(j)\|_2^2}{\sigma_I}} \cdot \left\{ \begin{array}{ll}
\frac{-\|X(i)-X(j)\|_2^2}{\sigma_X} & \text{if } \|X(i)-X(j)\|_2 < r \\
0 & \text{otherwise,}
\end{array} \right.
$$
Computational complexity

• A 100 x 100 image has 10K pixels
• A graph with 10K pixels has a 10K x 10K affinity matrix
• Eigenvalue computation of an N x N matrix is $O(N^3)$
• Very very expensive!
Eigenvectors of images

• The eigenvector has as many components as pixels in the image
Eigenvectors of images

- The eigenvector has as many components as pixels in the image
Another example

2nd eigenvector  3rd eigenvector  4th eigenvector
Recursive N-cuts

First partition

2nd eigenvector

2nd eigenvector of 1st subgraph

recursive partition
N-Cuts resources

- [https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf](https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf)
Images as graphs

- Enhancement: edge between far away pixel, weight $= 1 –$ magnitude of *intervening contour*
Eigenvectors of images