Grouping
Regions ↔ Boundaries
Grouping by clustering
Grouping by clustering

• Idea: embed pixels into high-dimensional space (e.g. 3-dimensions)
• Each pixel is a point
• Group together points
K-means

• Assumption: each group is a Gaussian with different means and same standard deviation

\[ P(x_i | \mu_j) \propto e^{-\frac{1}{2\sigma^2} \| x_i - \mu_j \|^2} \]

• Suppose we know all \( \mu_j \). Which group should a point \( x_i \) belong to?
  • The \( j \) with highest \( P(x_i | \mu_j) \)
  • = The \( j \) with smallest \( \| x_i - \mu_j \|^2 \)
K-means

• Assumption: each group = a Gaussian with different means and same standard deviation

• If means are known, then trivial assignment to groups. How?

• Assign data point to nearest mean!
K-means

• Problem: means are not known
• What if we know a set of points from each cluster?
  • $x_{k_1}, x_{k_2}, \ldots, x_{k_n}$ belong to cluster $k$
• What should be $\mu_k$?

$$\mu_k = \frac{(x_{k_1} + x_{k_2} + \ldots + x_{k_n})}{n}$$
K-means

• Problem: means are not known
• If assignment of points to clusters is known, then finding means is easy
• How? Compute the mean of every cluster!
K-means

• Given means, can assign points to clusters
• Given assignments, can compute means
• Idea: iterate!
K-means

• Step-1: randomly pick $k$ centers
K-means

- Step 2: Assign each point to nearest center
K-means

• Step 3: re-estimate centers
K-means

• Step 4: Repeat
K-means

- Step 4: Repeat
K-means

- Step 4: Repeat
K-means

- Ground-truth vs k-means
K-means - another example
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

Iteration 1

Iteration 5

Final: Iteration 17
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+position
The issues with k-means

• Captures pixel similarity but
  • Doesn’t capture continuity of contours
  • Captures near/far relationships only weakly
  • Can merge far away objects together

• Requires knowledge of k!

• Can it deal with texture?
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