Image segmentation

Prof. Noah Snavely
CS1114
http://www.cs.cornell.edu/cs1114

Cornell University
Computer Science

Image segmentation

- Given an image, can we find and segment all the objects?

Why image segmentation?

- Makes the image much easier to analyze
  - Large number of pixels → small number of segments

- Find structures we care about (e.g., lightsticks, bones)

- Compression

Image segmentation

- A (much) more difficult version of thresholding to find the red lightstick
  - We don't know
    - what colors the objects are
    - how many objects there are
    - whether each object is even a constant color

Image segmentation

- To start with, we'll look at a very simple version of segmentation

- **Color quantization**
  - Take an image with (possibly) many colors, convert it to an image with a small number of colors

Demo

How do we quantize the colors?

- Option 1: Could choose a fixed *palette* (red, green, blue, purple, white, …)

16 colors

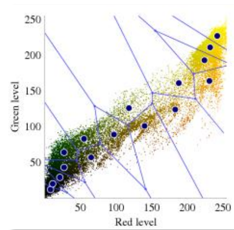
- Option 2: Could optimize the palette for a given image

16 colors

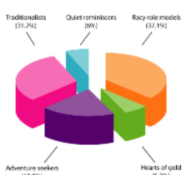
How do we compute the optimal palette?

- This is an example of a *clustering* problem

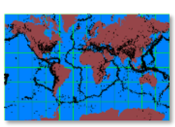
1. Find groups of pixels (clusters) that have similar color (i.e., similar RGB values)
2. Assign all the pixels in each cluster the same color



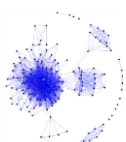
Applications of clustering



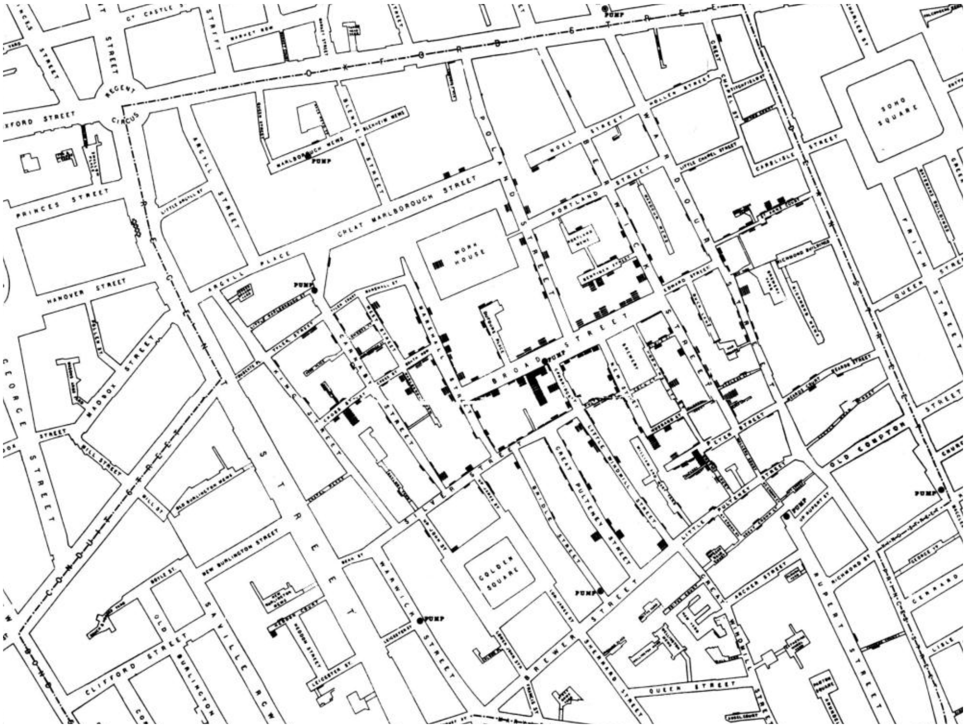
- Economics or politics
  - Finding similar-minded or similar behaving groups of people (market segmentation)
  - Find stocks that behave similarly



- Spatial clustering
  - Earthquake centers cluster along faults



- Social network analysis
  - Recognize communities of similar people



Clustering

- The distance between two items is the distance between their vectors
- We'll also assume for now that we know the number of clusters

Example

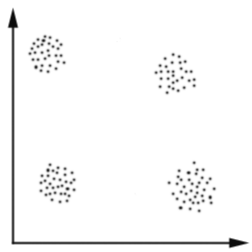


Figure from Johan Everts

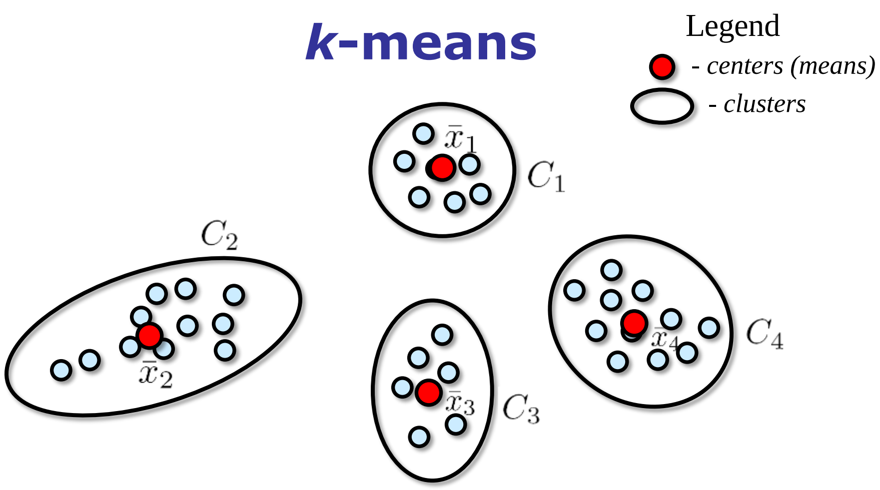
Clustering algorithms

- There are many different approaches
- How is a cluster is represented?
  - Is a cluster represented by a data point, or by a point in the middle of the cluster?
- What algorithm do we use?
  - An interesting class of methods uses graph partitioning
  - Edge weights are distances

One approach: *k*-means

- Suppose we are given *n* points, and want to find *k* clusters
- We will find *k* cluster centers (or means), and assign each of the *n* points to the nearest cluster center $\bar{x}_j$
  - A *cluster* is a subset of the *n* points, called C_j
  - We'll call each cluster center a *mean*

k-means



- How do we define the best *k* means?

k-means

- Idea: find the centers that minimize the sum of squared distances to the points
- Objective:

Given input points $x_1, x_2, x_3, \ldots, x_n$, find the clusters $C_1, C_2, \ldots C_k$ and the cluster centers $\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_k$ that minimize

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} |x_i - \bar{x}_j|^2$$

Optimizing *k*-means

Given input points $x_1, x_2, x_3, \ldots, x_n$, find the clusters $C_1, C_2, \ldots C_k$ and the cluster centers $\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_k$ that minimize

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} |x_i - \bar{x}_j|^2$$

- This is called an *objective function*
- Goal is to find the clusters and means that minimize this objective function
- How do we do this?

Optimizing k-means

Given input points $x_1, x_2, x_3, \ldots, x_n$, find the clusters $C_1, C_2, \ldots C_k$ and the cluster centers $\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_k$ that minimize

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} |x_i - \bar{x}_j|^2$$

- Brute-force approach:
  1. Try every possible clustering
     - (The best mean for a cluster is just the average of the cluster elements)
  2. Check the value of the objective for this clustering
  3. Pick the clustering that gives the minimum value

Optimizing k-means

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} |x_i - \bar{x}_j|^2$$

- Brute-force approach:
  1. Try every possible clustering
     - (The best mean for a cluster is just the average of the cluster elements)
  2. Check the value of the objective for this clustering
  3. Pick the clustering that gives the minimum value

- How much work is this?
  - (How many possible clusterings are there?)

Optimizing *k*-means

Given input points $x_1, x_2, x_3, \ldots, x_n$, find the clusters $C_1, C_2, \ldots C_k$ and the cluster centers $\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_k$ that minimize

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} |x_i - \bar{x}_j|^2$$

- The bad news: it is practically impossible to find the *global minimum* of this objective function
  - no one has ever come up with an algorithm that is faster than exponential time (and probably never will)
- There are many problems like this (called *NP-hard*)

Optimizing *k*-means

- It's possible to prove that this is a hard problem (you'll learn how in future CS courses – it involves reductions)
- What now?
- We shouldn't give up… it might still be possible to get a "pretty good" solution

Greedy algorithms

- Many CS problems can be solved by repeatedly doing whatever seems best at the moment
  - I.e., without needing a long-term plan
- These are called greedy algorithms
- Example: sorting by swapping out-of-order pairs (e.g., bubble sort)

Making change

- For US currency (quarters, dimes, nickels, pennies) we can make change with a greedy algorithm:
  1. While remaining change is > 0
  2. Give the highest denomination coin whose value is >= remaining change

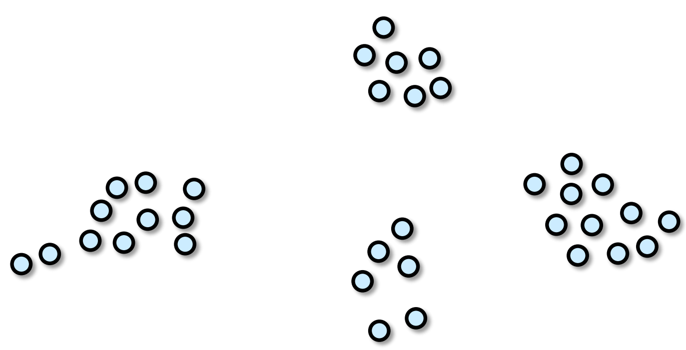
41 cents: 

- What if our denominations are 50, 49, and 1?
  - How should we give back 98 cents in change?
  - Greedy algorithms don't always work…
  - (This problem requires more advanced techniques)

A greedy method for *k*-means

- Pick a random point to start with, this is your first cluster center
- Find the farthest point from the cluster center, this is a new cluster center
- Find the farthest point from any cluster center and add it
- Repeat until we have *k* centers

A greedy method for *k*-means



A greedy method for *k*-means

- Unfortunately, this doesn't work that well
- The answer we get could be **much** worse than the optimum

The *k*-centers problem

- Let's look at a related problem: *k*-centers
- Find *k* cluster centers that minimize the *maximum* distance between any point and its nearest center
  - We want the worst point in the worst cluster to still be good (i.e., close to its center)
  - Concrete example: place *k* hospitals in a city so as to minimize the maximum distance from a hospital to a house

An amazing property

- This algorithm gives you a solution that is no worse than twice the optimum
- Such results are sometimes difficult to achieve, and the subject of much research
  - Mostly in CS6810, a bit in CS4820
  - You can't find the optimum, yet you can prove something about it!
- Sometimes related problems (e.g. k-means vs. k-centers) have very different guarantees

Next time

- More on clustering