CSE/STAT 416

Section 7, 5/16

May 16, 2019

Agenda

- Lecture Recap: Pros and Cons of K-means
- Intro to Spectral Clustering
- Spectral Clustering vs. K-means demo
- Time for questions/review

Recall from Class...

The k-means algorithm

- Start with k randomly initialized centers, μ_j.
- Repeat until the centers stop moving:
  - Fix centers and assign each point to the closest center (update each datapoints's z_i value
  - Fix the z_i and update the centers μ_j (set μ_j as the centroid of all points with $z_i = j$.

At every step, the objective

$$\sum_{j=1}^{k} \sum_{i:z_i=j} ||\mu_j - x_i||_2^2$$

gets smaller (clusters get more homogeneous in terms of distance)

Cons of K-means

- **Problem:** Sensitive to initial conditions

Cons of K-means

- **Problem:** Sensitive to initial conditions
    - **Solution:** Try many initial conditions and compare performance, pick initial conditions intelligently (k-means++)

Cons of K-means

- **Problem:** Sensitive to initial conditions
  - **Solution:** Try many initial conditions and compare performance, pick initial conditions intelligently (k-means++)
- **Problem:** Must select K in advance

Cons of K-means

- **Problem:** Sensitive to initial conditions
  - **Solution:** Try many initial conditions and compare performance, pick initial conditions intelligently (k-means++)
- **Problem:** Must select K in advance
  - **Solution:** Can try many ks and compare cluster heterogeneity
  - Be careful of overfitting: clusters always get more homogeneous as K gets larger
  - Should penalize large k.

Cons of K-means

- **Problem:** Sensitive to initial conditions
  - **Solution:** Try many initial conditions and compare performance, pick initial conditions intelligently (k-means++)
- **Problem:** Must select K in advance
  - **Solution:** Can try many ks and compare cluster heterogeneity
  - Be careful of overfitting: clusters always get more homogeneous as K gets larger
  - Should penalize large k.
- **Problem:** Assumes linear cluster boundaries, and assumes minimizing within-node distance is best objective.

Cons of K-means

- **Problem:** Sensitive to initial conditions
  - **Solution:** Try many initial conditions and compare performance, pick initial conditions intelligently (k-means++)
- **Problem:** Must select K in advance
  - **Solution:** Can try many ks and compare cluster heterogeneity
  - Be careful of overfitting: clusters always get more homogeneous as K gets larger
  - Should penalize large k.
- **Problem:** Assumes linear cluster boundaries, and assumes minimizing within-node distance is best objective.
  - **Solution:** Use a new algorithm, or at least a new representation of the data

Kmeans interactive demo

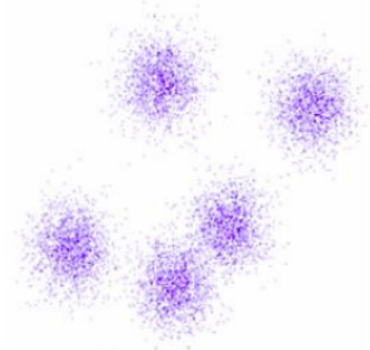
- https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Motivation for Spectral Clustering

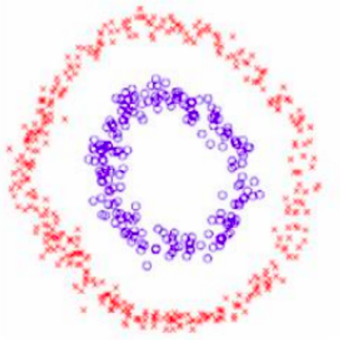
- $\sum_{j=1}^{k} \sum_{i:z_i=j} ||\mu_j - x_i||_2^2$ is not necessarily the best objective function

Motivation for Spectral Clustering

- $\sum_{j=1}^{k} \sum_{i:z_i=j} ||\mu_j - x_i||_2^2$ is not necessarily the best objective function
- Kmeans prioritizes compactness; what if we want to prioritize connectivity?



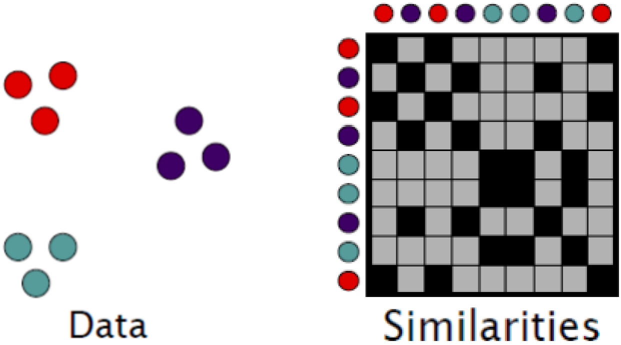
Compactness



Connectivity

Input for Spectral Clustering

First, turn the data into a similarity matrix, or affinity matrix.



Entry $(i, j) = 1$ tells us how similar datapoint i is to datapoint j. Matrix should be symmetric and non-negative.

[0] Figure credit: `www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture21_2.pdf`

Examples of similarity measures

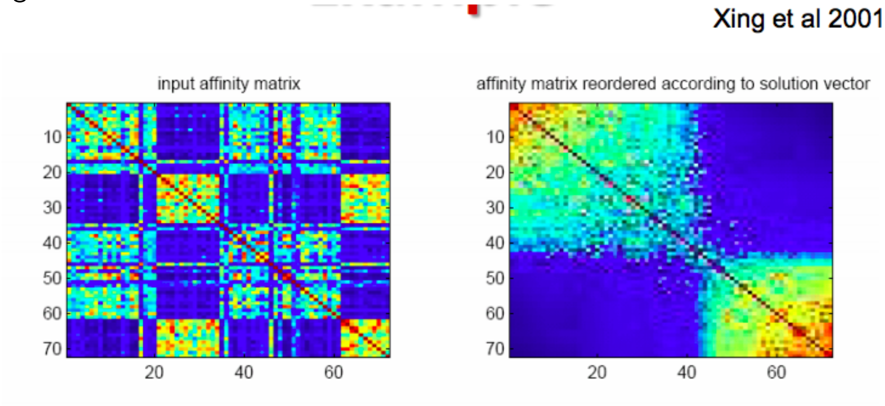
- If the data are represented as vectors:
  - Entry $(i, j) = 1$ if datapoint j is one of the k nearest neighbors of datapoint i.
  - Entry $(i, j) = e^{-||x_i - x_j||^2/(2\sigma^2)}$ (Gaussian Kernel Function)
  - Cosine similarity
  - $1/$(distance) for any measure of distance

Examples of similarity measures

- If the data are represented as vectors:
  - Entry $(i, j) = 1$ if datapoint j is one of the k nearest neighbors of datapoint i.
  - Entry $(i, j) = e^{-||x_i - x_j||^2/(2\sigma^2)}$ (Gaussian Kernel Function)
  - Cosine similarity
  - $1/(\text{distance})$ for any measure of distance
- A cool thing about a similarity matrix is that you can define one even if your data are not vectors.
  - Context specific notions of similarity
  - Co-authorship, friendship, etc.

Spectral Clustering Algorithm

Main idea: rearrange the rows and columns of the matrix to get a block diagonal form.



Maximize total similarity within the blocks, minimize total similarity outside of the blocks.

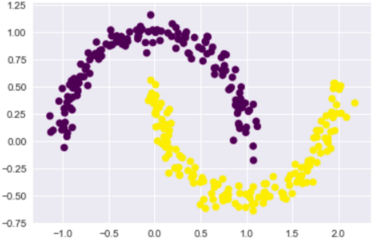
How does it work?

Algorithm:

- Encode data as similarity matrix
- Compute eigenvalues of similarity matrix, use the first few to obtain low-dimensional representation of similarity matrix
  - (the low dimensional representation that keeps as much information about similarity as possible)
- Apply K-means (or a similar algorithm) in this low dimensional space to get clustering

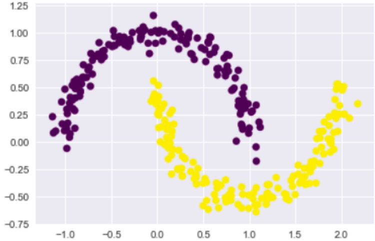
Discussion

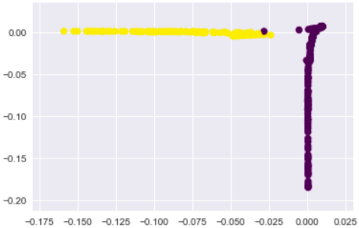
If we end up applying K-means anyways, how does it draw non-linear decision boundaries?



Discussion

If we end up applying K-means anyways, how does it draw non-linear decision boundaries?





The boundaries it draws *are* linear, but they are linear in a transformed space

KMeans and Spectral Clustering Demo

Demo notebook

Pros and Cons of Spectral Clustering

- Pros:
  - Can handle arbitrary cluster shapes
  - Mathematically elegant, and run time is reasonable for medium-sized problems
  - Allows for interesting, context-specific definitions of similarity
- Cons:
  - Still need to pick K in advance (either know it or test several different Ks)
  - Results highly dependent on what similarity metric is chosen