THINKFUL

Medoid-based Algorithms

DATA SCIENCE

Agenda

- Overview of k-medoids clustering
- Step-by-step walkthrough of PAM algorithm
- Advantages and disadvantages
- Scaling to large datasets (CLARA, CLARANS)

Warm-Up

Up until now, we have considered clustering on quantitative data, which naturally lends itself to being framed in the context of centroids, distances, and means. Consider the following questions:

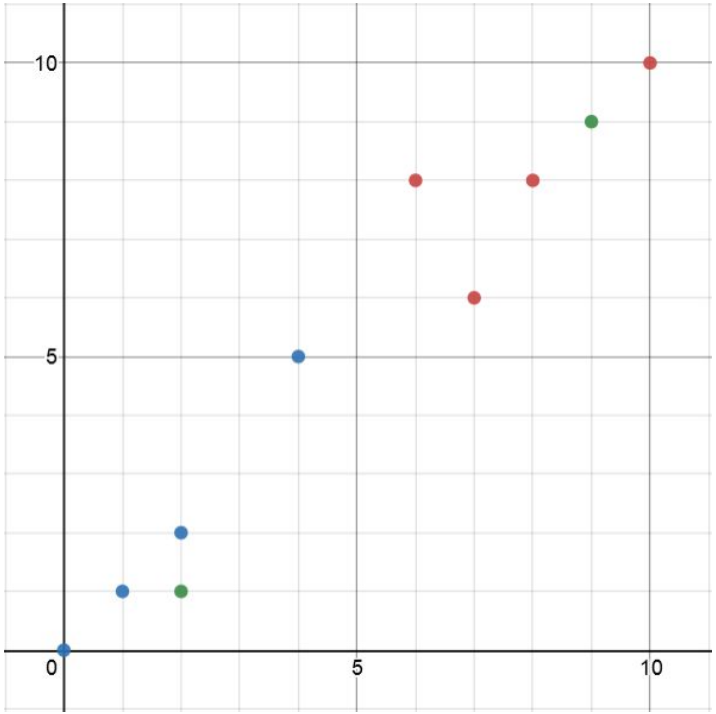
- Why do distances and means fail to extend naturally to nominal categorical data?
- Is there a measure of central tendency other than a mean that would be better suited for categorical data?
- Is there any potential advantage to using actual data points as “characteristic values for clusters” instead of centroids?

Overview

K-medoids is an alternative to k-means that works with categorical and mixed data

- Instead of being restricted to Euclidean distance (more generally, distance metrics that are compatible with statistical measures of central tendency like mean/median/mode), we can use any dissimilarity measure
- Additionally, the clusters are characterized by actual data points instead of centroids, which will usually not be one of the original data points
- Primary goal: minimize the within-cluster sum-of-dissimilarity from observations to corresponding medoids

Overview



UP NEXT

PAM Algorithm Walkthrough

Algorithm Summary

The simplest (naive) implementation of k-medoids, called PAM (Partitioning Around Medoids), shares key similarities with k-means: an iterative approach that alternates between two steps (after choosing an initial set of k medoids):

- **Assignment:** Assign each point to a cluster/medoid. For a specific point, this assignment is done by calculating the dissimilarity of said point with each of the current medoids and picking the medoid with minimum dissimilarity

Algorithm Summary

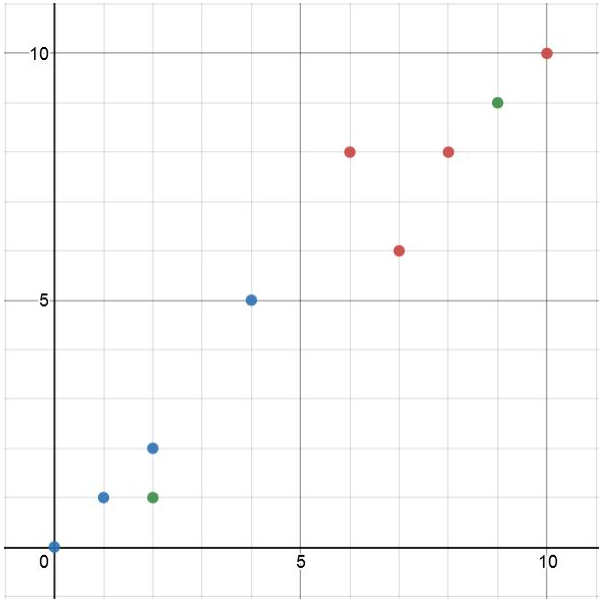
- **Update:** Recalculate the medoid for each cluster by swapping each medoid with each other non-medoid point and finding the swap that results in the lowest total dissimilarity across all points in the cluster
- This process is repeated until convergence (until the assignments and medoids no longer change)
- Like Lloyd's algorithm, this approach is an approximation/heuristic
  - Results can be sensitive to initial choice of medoids; running several times is recommended

K-Medoids: Initial State

x_1	x_2	Initial medoid	Initial cost
0	0	(2,1)	3
1	1	(2,1)	1
2	1	(2,1)	0
2	2	(2,1)	1
4	5	(2,1)	6
6	8	(9,9)	4
7	6	(9,9)	5
8	8	(9,9)	2
9	9	(9,9)	0
10	10	(9,9)	2

Total cost: 24

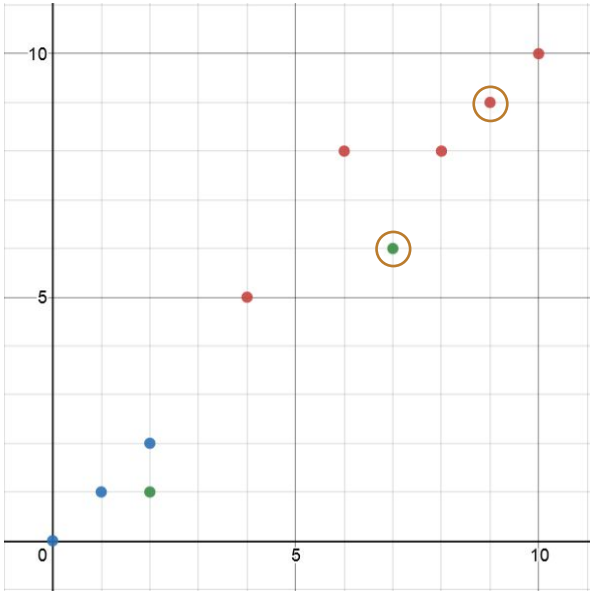
Dissimilarity measure/cost:
Manhattan distance



K-Medoids: Swap 1

x_1	x_2	Initial medoid	Initial cost	Assigned medoid after swap	Cost after swap
0	0	(2,1)	3	(2,1)	3
1	1	(2,1)	1	(2,1)	1
2	1	(2,1)	0	(2,1)	0
2	2	(2,1)	1	(2,1)	1
4	5	(2,1)	6	(7,6)	4
6	8	(9,9)	4	(7,6)	3
7	6	(9,9)	5	(7,6)	0
8	8	(9,9)	2	(7,6)	3
9	9	(9,9)	0	(7,6)	5
10	10	(9,9)	2	(7,6)	7
			24		27

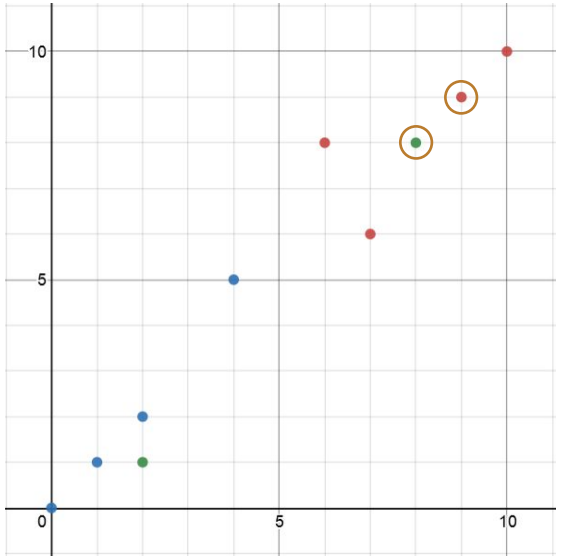
Current swap under consideration:
(9,9) and (7,6)



K-Medoids: Swap 2

x_1	x_2	Initial medoid	Initial cost	Assigned medoid after swap	Cost after swap
0	0	(2,1)	3	(2,1)	3
1	1	(2,1)	1	(2,1)	1
2	1	(2,1)	0	(2,1)	0
2	2	(2,1)	1	(2,1)	1
4	5	(2,1)	6	(2,1)	6
6	8	(9,9)	4	(8,8)	2
7	6	(9,9)	5	(8,8)	3
8	8	(9,9)	2	(8,8)	0
9	9	(9,9)	0	(8,8)	2
10	10	(9,9)	2	(8,8)	4
			24		22

Current swap under consideration:
(9,9) and (8,8)



K-Medoids: Swap 3, 4, etc.

x_1	x_2	Initial medoid	Initial cost	Assigned medoid after swap	Cost after swap
0	0	(2,1)	3	(2,1)	3
1	1	(2,1)	1	(2,1)	1
2	1	(2,1)	0	(2,1)	0
2	2	(2,1)	1	(2,1)	1
4	5	(2,1)	6	(2,1)	6
6	8	(9,9)	4	(8,8)	2
7	6	(9,9)	5	(8,8)	3
8	8	(9,9)	2	(8,8)	0
9	9	(9,9)	0	(8,8)	2
10	10	(9,9)	2	(8,8)	4
			24		27

Repeat this process for all possible swaps; new medoids at the end of the iteration are based on configuration with lowest total cost

- With 2 medoids and 8 non-medoid points, we would repeat the process on the previous slide 2 * 8 or 16 times per iteration and take the lowest of the 16 calculated total costs
- More generally, for k medoids and n data points, we would need to evaluate k(n-k) swaps per iteration
- If none of the 16 costs are lower than the cost we started with, the algorithm terminates

Advantages

K-Medoids has several advantages over k-means:

- Flexibility
  - Works with categorical and mixed data types
  - Works with arbitrary dissimilarity measure
  - Only requires a similarity matrix, not access to the data itself
- Robustness to outliers
  - In k-means, the centroids can be anywhere in the feature space
  - In contrast, medoids must be observations from the dataset, which limits the extent to which an outlier can “drag” the representative points away from the true center

Disadvantages

However, k-medoids also has drawbacks of its own

- Poor scalability
  - Naïve implementations scale quadratically with the size of the data, $O(n^2)$
  - Much slower than k-means, which scales linearly, O(n)
  - Arbitrary dissimilarity measures are not constrained by triangle inequality, so previously discussed k-means optimizations cannot be applied
- User must specify k in advance
  - Also a drawback of k-means
  - No “one true approach” to choosing k; most approaches are a combination of heuristics and domain knowledge

Summary

- K-medoids clustering is a popular medoid-based unsupervised learning algorithm
- Much like k-means, the simplest implementation of k-medoids, PAM, alternates between an assignment step (assigning observations to medoids) and an update step (recalculating centroids based on newly assigned observations)
- PAM has several advantages over k-means, but scales very poorly with large data sets; CLARA and CLARANS are two algorithms which rely on sampling to reduce computation time

EXERCISE

1. Jupyter notebook
2. Data set: Student life survey data
3. As we did previously, isolate the questions related to stress and calculate the dissimilarity matrix
4. Run K-Medoids with random initial medoids
5. Re-run K-Medoids with more thoughtfully chosen initial medoids

Thank You