

Lecture 14 - Clustering



DSC 40A, Spring 2023

Announcements

- ▶ Midterm 1 is Friday during lecture.
- ▶ No groupwork tonight. Instead, TA and tutors will host a mock exam and review session **today from 7-9pm in FAH 1301**.
 - ▶ Note the room change (same building).
 - ▶ You'll take the midterm from Winter 2022, when I last taught this class.
 - ▶ **Bring paper.** Formula sheet and exam questions provided.

Midterm 1 is Friday during lecture

- ▶ [Formula sheet](#) will be provided for you. No other notes.
- ▶ No calculators. This implies no crazy calculations.
- ▶ Assigned seats will be posted on Campuswire.
- ▶ We will not answer questions during the exam. State your assumptions if anything is unclear.
- ▶ The exam will include long-answer homework-style questions, as well as short-answer questions such as multiple choice or filling in a numerical answer.
- ▶ The exam covers Homeworks 1 through 4, which includes Monday's lecture, but not clustering.

Midterm study strategy

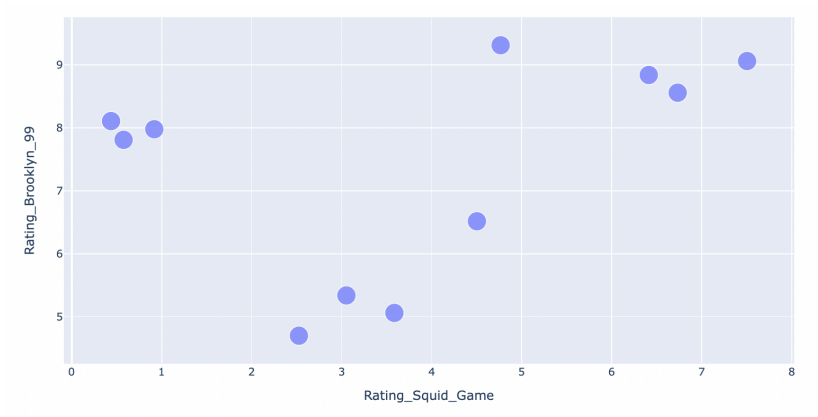
- ▶ Review the written solutions to previous homeworks and groupworks.
- ▶ Identify which concepts are still iffy. Re-watch podcasts, post on Campuswire, come to office hours, use resources [on course website](#).
- ▶ Work through past exams [on course website](#).
- ▶ Study in groups.
- ▶ Summarize key facts and formulas.

Agenda

- ▶ The clustering problem.
- ▶ k-Means Clustering algorithm.
- ▶ Why does k-Means work?
- ▶ Practical considerations.

The clustering problem

Question: how might we “cluster” these points into groups?



Problem statement: clustering

Goal: Given a list of n data points, stored as vectors in \mathbb{R}^d , $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$, and a positive integer k , **place the data points into k groups of nearby points.**

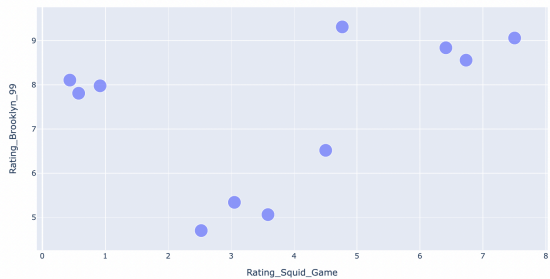
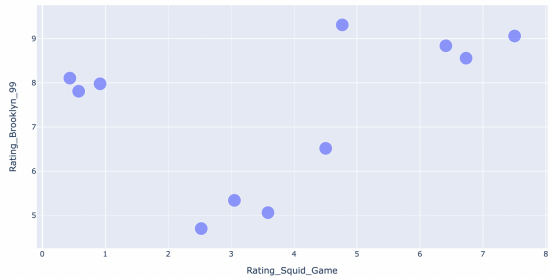
- ▶ These groups are called “clusters”.
- ▶ Think about groups as **colors**.
 - ▶ i.e., the goal of clustering is to assign each point a color, such that points of the same color are close to one another.
- ▶ Note, unlike with regression, there is no “right answer” that we are trying to predict — there is no y !
 - ▶ Clustering is an **unsupervised** method.

How do we define a group?

- ▶ One solution: pick k cluster centers, i.e. **centroids**:

$$\vec{\mu}_1, \vec{\mu}_2, \dots, \vec{\mu}_k \text{ in } \mathbb{R}^d$$

- ▶ These k centroids define the k groups.
- ▶ Each data point “belongs” to the group corresponding to the nearest centroid.
- ▶ This reduces our problem from being “find the best group for each data point” to being “find the best locations for the centroids”.



How do we pick the centroids?

- ▶ Let's come up with an **cost function**, C , which describes how good a set of centroids is.
 - ▶ Cost functions are a generalization of empirical risk functions.
- ▶ One possible cost function:

$C(\mu_1, \mu_2, \dots, \mu_k)$ = total squared distance of each data point \vec{x}_i to its closest centroid μ_j

- ▶ This C has a special name, **inertia**.
- ▶ Lower values of C lead to “better” clusterings.
 - ▶ **Goal:** Find the centroids $\mu_1, \mu_2, \dots, \mu_k$ that minimize C .

Discussion Question

Suppose we have n data points, $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$, each of which are in \mathbb{R}^d .

Suppose we want to cluster our dataset into k clusters. How many ways can we assign points to clusters?

- a) $d \cdot k$
- b) d^k
- c) n^k
- d) k^n
- e) $n \cdot k \cdot d$

How do we minimize inertia?

- ▶ **Problem:** there are exponentially many possible clusterings. It would take too long to try them all.
- ▶ **Another Problem:** we can't use calculus or algebra to minimize C , since to calculate C we need to know which points are in which clusters.
- ▶ We need another solution.

k-Means Clustering

k-Means Clustering, i.e. Lloyd's Algorithm

Here's an algorithm that attempts to minimize inertia:

1. Pick a value of k and randomly initialize k centroids.
2. Keep the centroids fixed, and update the groups.
 - ▶ Assign each point to the nearest centroid.
3. Keep the groups fixed, and update the centroids.
 - ▶ Move each centroid to the center of its group.
4. Repeat steps 2 and 3 until the centroids stop changing.

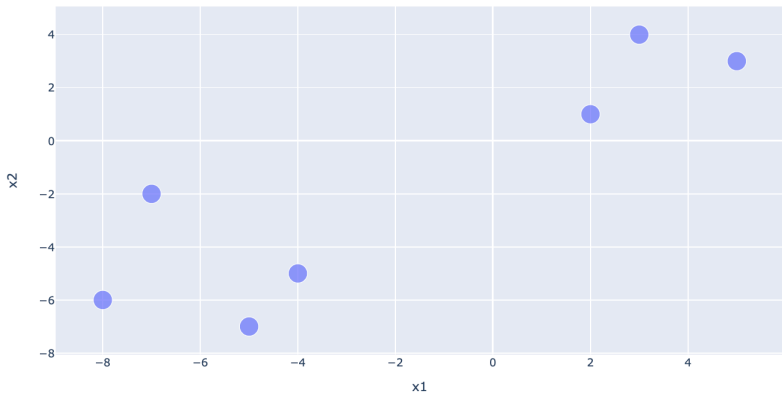
Example

See the following site for an interactive visualization of k-Means Clustering: <https://tinyurl.com/40akmeans>

An example by hand

Suppose we choose the initial centroids $\mu_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\mu_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.

Where will the centroids move to after one iteration of k-Means Clustering?



Demo

Let's see k-Means Clustering in action. [Follow along here.](#)

Why does k-Means work?

What is the goal of k-Means Clustering?

- ▶ Recall, our goal is to find the centroids $\mu_1, \mu_2, \dots, \mu_k$ that minimize inertia:

$C(\mu_1, \mu_2, \dots, \mu_k)$ = total squared distance of each data point \vec{x}_i to its closest centroid μ_j

- ▶ Let's argue that each step of the k-Means Clustering algorithm reduces inertia.
 - ▶ After enough iterations, inertia will be small enough.

Why does k-Means work? (Step 1)

Step 1: Pick a value of k and randomly initialize k centroids.

- ▶ After initializing our k centroids, we have an initial value of inertia. We are going to argue that this only decreases.

Why does k-Means work? (Step 2)

Step 2: Keep the centroids fixed, and update the groups by assigning each point to the nearest centroid.

- ▶ Assuming the centroids are fixed, for each \vec{x}_i we have a choice — which group should it be a part of?
- ▶ Whichever group we choose, inertia will be calculated using the squared distance between \vec{x}_i and that group's centroid.
- ▶ Thus, to minimize inertia, we assign each \vec{x}_i to the group corresponding to the closest centroid.

Note that this analysis holds every time we're at Step 2, not just the first time.

Why does k-Means work? (Step 3)

Step 3: Keep the groups fixed, and update the centroids by moving each centroid to the center of its group (by averaging coordinates).

- ▶ Before we justify why this is optimal, let's revisit inertia.

Aside: separating inertia

- ▶ Inertia:

$C(\mu_1, \mu_2, \dots, \mu_k)$ = total squared distance of each data point \vec{x}_i to its closest centroid μ_j

- ▶ Note that an equivalent way to write inertia is

$C(\mu_1, \mu_2, \dots, \mu_k) = C(\mu_1) + C(\mu_2) + \dots + C(\mu_k)$ where
 $C(\mu_j)$ = total squared distance of each data point \vec{x}_i in group j to centroid μ_j

- ▶ What's the point?

Why does k-Means work? (Step 3)

$C(\mu_1, \mu_2, \dots, \mu_k) = C(\mu_1) + C(\mu_2) + \dots + C(\mu_k)$ where

$C(\mu_j)$ = total squared distance of each data point \vec{x}_i
in group j to centroid μ_j

Step 3: Keep the groups fixed, and update the centroids by moving each centroid to the center of its group (by averaging coordinates).

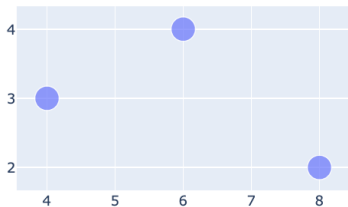
- ▶ Let's argue why this minimizes $C(\mu_j)$, for each group j .

Why does k-Means work? (Step 3)

$C(\mu_j)$ = total squared distance of each data point \vec{x}_i
in group j to centroid μ_j

Suppose group j contains the points $(4, 3)$, $(6, 4)$, and $(8, 2)$.

Where should we put $\mu_j = \begin{bmatrix} a \\ b \end{bmatrix}$ to minimize $C(\mu_j)$?



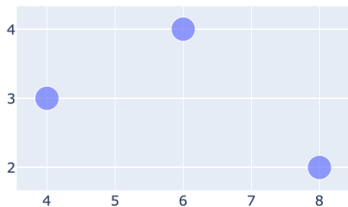
Why does k-Means work? (Step 3)

Why does k-Means work? (Step 3)

$C(\mu_j)$ = total squared distance of each data point \vec{x}_i
in group j to centroid μ_j

Suppose group j contains the points $(4, 3)$, $(6, 4)$, and $(8, 2)$.

Where should we put $\mu_j = \begin{bmatrix} a \\ b \end{bmatrix}$ to minimize $C(\mu_j)$?



Cost and empirical risk

- ▶ On the previous slide, we saw a function of the form

$$\begin{aligned}C(\mu_j) = C(a, b) &= (4 - a)^2 + (3 - b)^2 \\ &+ (6 - a)^2 + (4 - b)^2 \\ &+ (8 - a)^2 + (2 - b)^2\end{aligned}$$

- ▶ $C(a, b)$ can be thought of as the sum of two separate functions, $f(a)$ and $g(b)$.
 - ▶ $f(a) = (4 - a)^2 + (6 - a)^2 + (8 - a)^2$ computes the total squared distance of each x_1 coordinate to a .
 - ▶ From earlier in the course, we know that $a^* = \frac{4+6+8}{3} = 6$ minimizes $f(a)$.

Practical considerations

Initialization

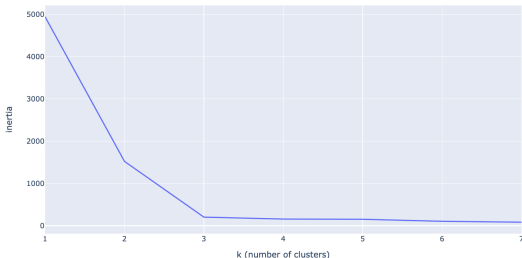
- ▶ Depending on our initial centroids, k-Means may “converge” to a clustering that doesn’t actually have the lowest possible inertia.
 - ▶ In other words, like gradient descent, k-Means can get caught in a **local minimum**.
- ▶ Some solutions:
 - ▶ Run k-Means several times, each with different randomly chosen initial centroids. Keep track of the inertia of the final result in each attempt. Choose the attempt with the lowest inertia.
 - ▶ **k-Means++**: choose one initial centroid at random, and place other centroids far from all other centroids.

Choosing k

- ▶ Note that as k increases, inertia decreases.
 - ▶ Intuitively, as we add more centroids, the distance between each point and its closest centroid will drop.
- ▶ But the goal of clustering is to put data points into groups, and having a large number of groups may not be meaningful.
- ▶ This suggests a tradeoff between k and inertia.

The “elbow” method

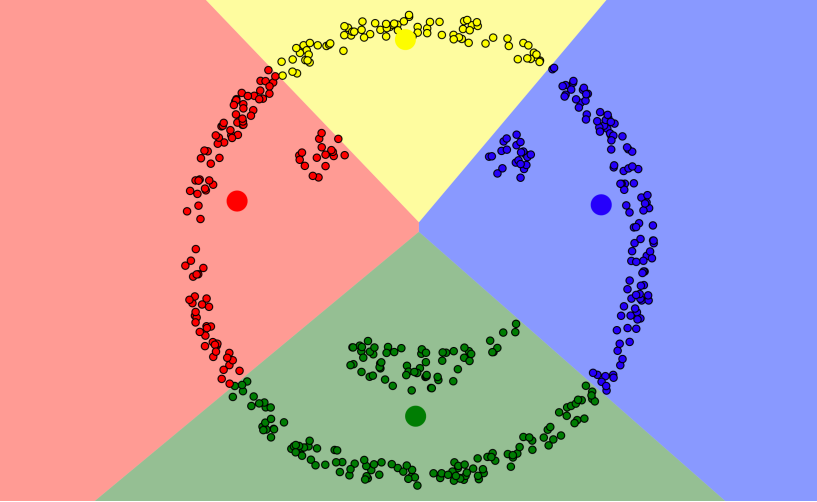
- ▶ Strategy: run k-Means Clustering for many choices of k (e.g. $k = 1, 2, 3, \dots, 8$).
- ▶ Compute the value of inertia for each resulting set of centroids.
- ▶ Plot a graph of inertia vs k .
- ▶ Choose the value of k that appears at an “elbow”.



See the notebook for a demo.

Low inertia isn't everything!

- ▶ Even if k-Means works as intended and finds the choice of centroids that minimize inertia, the resulting clustering may not look “right” to us humans.
 - ▶ Recall, inertia measures the total squared distance to centroids.
 - ▶ This metric doesn't always match our intuition.
- ▶ Let's look at some examples at <https://tinyurl.com/40akmeans>.
 - ▶ Go to “I'll Choose” and “Smiley Face”. Good luck!



Other clustering techniques

- ▶ k-Means Clustering is just one way to cluster data.
- ▶ There are many others, each of which work differently and produce different kinds of results.
- ▶ Another common technique: **agglomerative clustering**.
 - ▶ High level: start out with each point being in its own cluster. Repeatedly combine clusters until only k are left.
- ▶ Check out [this chart](#).

Summary

- ▶ k-Means Clustering attempts to minimize inertia.
 - ▶ We showed that it minimizes inertia at each step, but it's possible that it converges to a local minimum.
 - ▶ Different initial centroids can lead to different clusterings.
- ▶ To choose k , the number of clusters, we can use the elbow method.
- ▶ Next time: switching gears to probability and combinatorics.