Lecture 16 – Clustering (continued) and Introduction to Probability

DSC 40A, Fall 2022 @ UC San Diego

Dr. Truong Son Hy, with help from **many others**

Announcements

- Look at the readings linked on the course website!
- Groupwork Relsease Day: Thursday afternoon
  Groupwork Submission Day: Monday midnight
  Homework Release Day: Friday after lecture
  Homework Submission Day: Friday before lecture
- See `dsc40a.com/calendar` for the Office Hours schedule.
- We will grade the midterm soon.

Agenda

- Review of k-Means clustering algorithm.
- Why does k-Means work?
- Practical considerations.
- Introduction to Probability

Summary: K-Means clustering

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

- Clusters are defined by **centroids**, $\mu_1, \mu_2, ..., \mu_k$. Each data point "belongs" to the group corresponding to the nearest centroid.

- We want to find the centroids that minimize **inertia**:

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

- k-Means Clustering is an algorithm that attempts to minimize inertia.

Summary: Lloyd's Algorithm

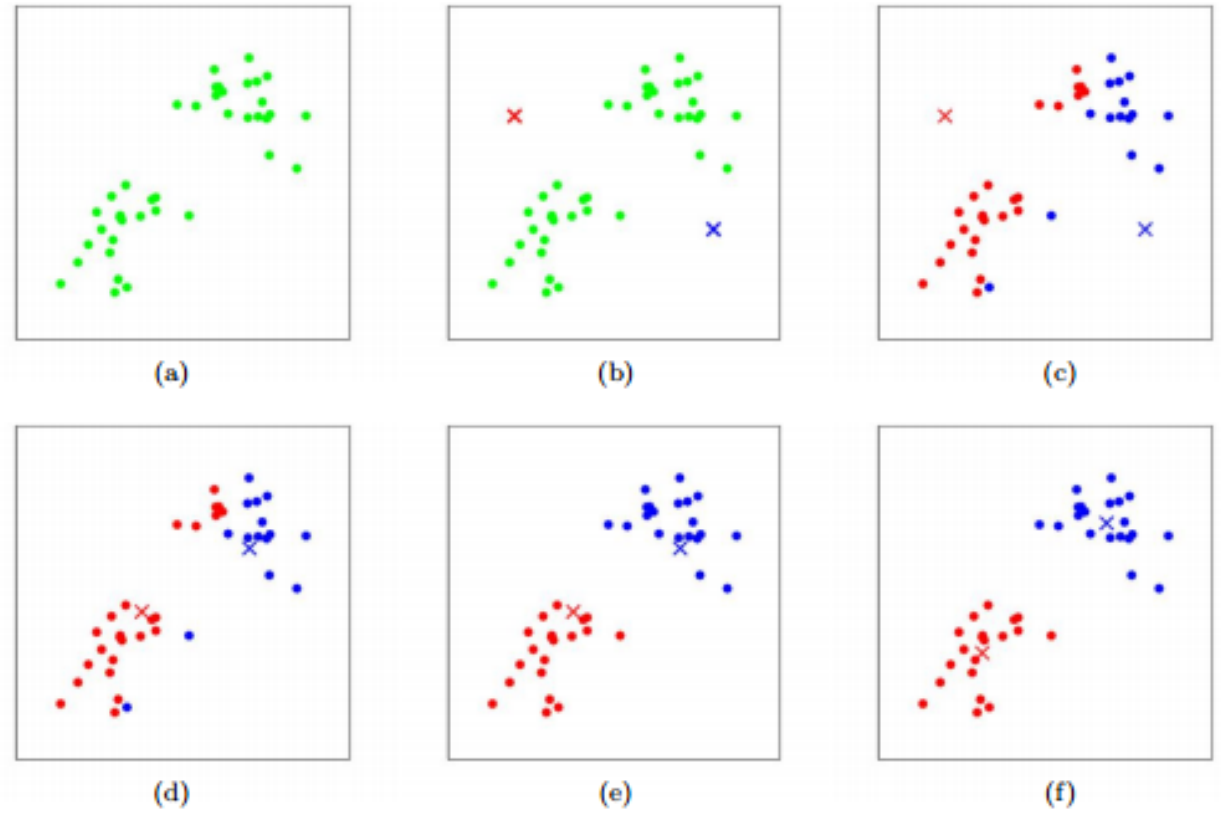
1. Pick a value of k and randomly initialize k centroids.

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 by averaging their coordinates.

4. Repeat steps 2 and 3 until the centroids stop changing.

Summary: K-Means visualization



Why does k-Means work?

What is the goal of k-Means Clustering?

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

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

- 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)

Let's look at each step one at a time.

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 re-visit inertia.

Aside: separating inertia

- Inertia:

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

- Note that an equivalent way to write inertia is

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

- What's the point?

Why does k-Means work? (Step 3)

$$C(\mu_1, \mu_2, ..., \mu_k) = C(\mu_1) + C(\mu_2) + ... + C(\mu_k) \text{ where}$$

$C(\mu_j)$ = 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).

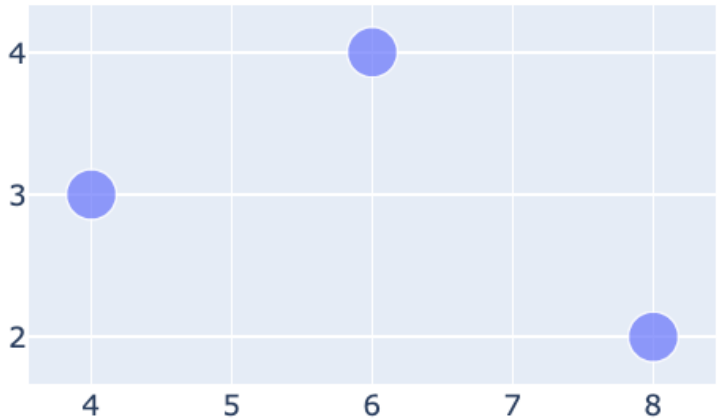
- Let's argue why this minimizes $C(\mu_j)$, for each group j.

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 $C(\mu_j)$?



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.

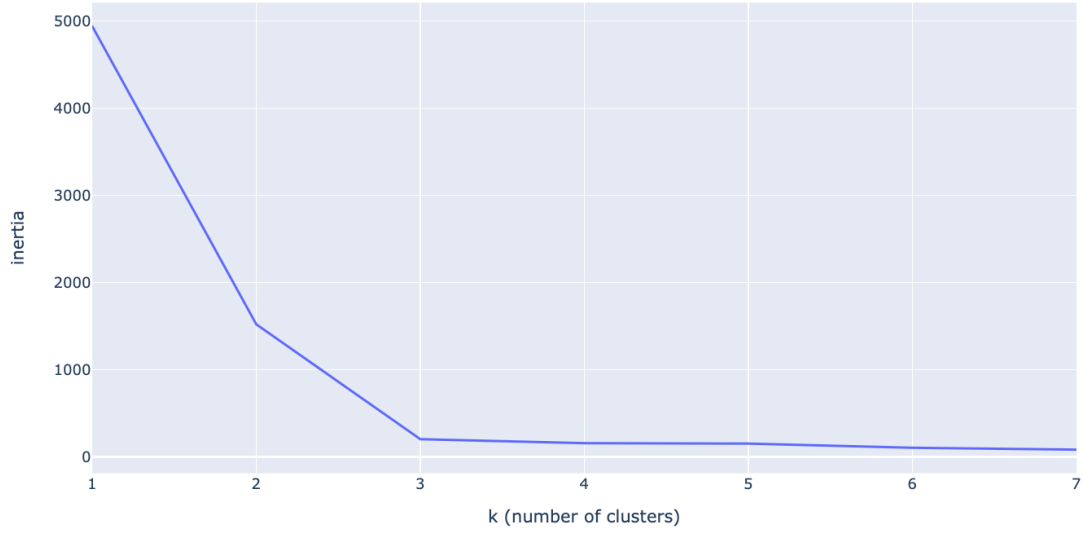
  - **k-Means++**: choose one initial centroid at random, and choose the remaining initial centroids by maximizing distance from all other centroids.

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

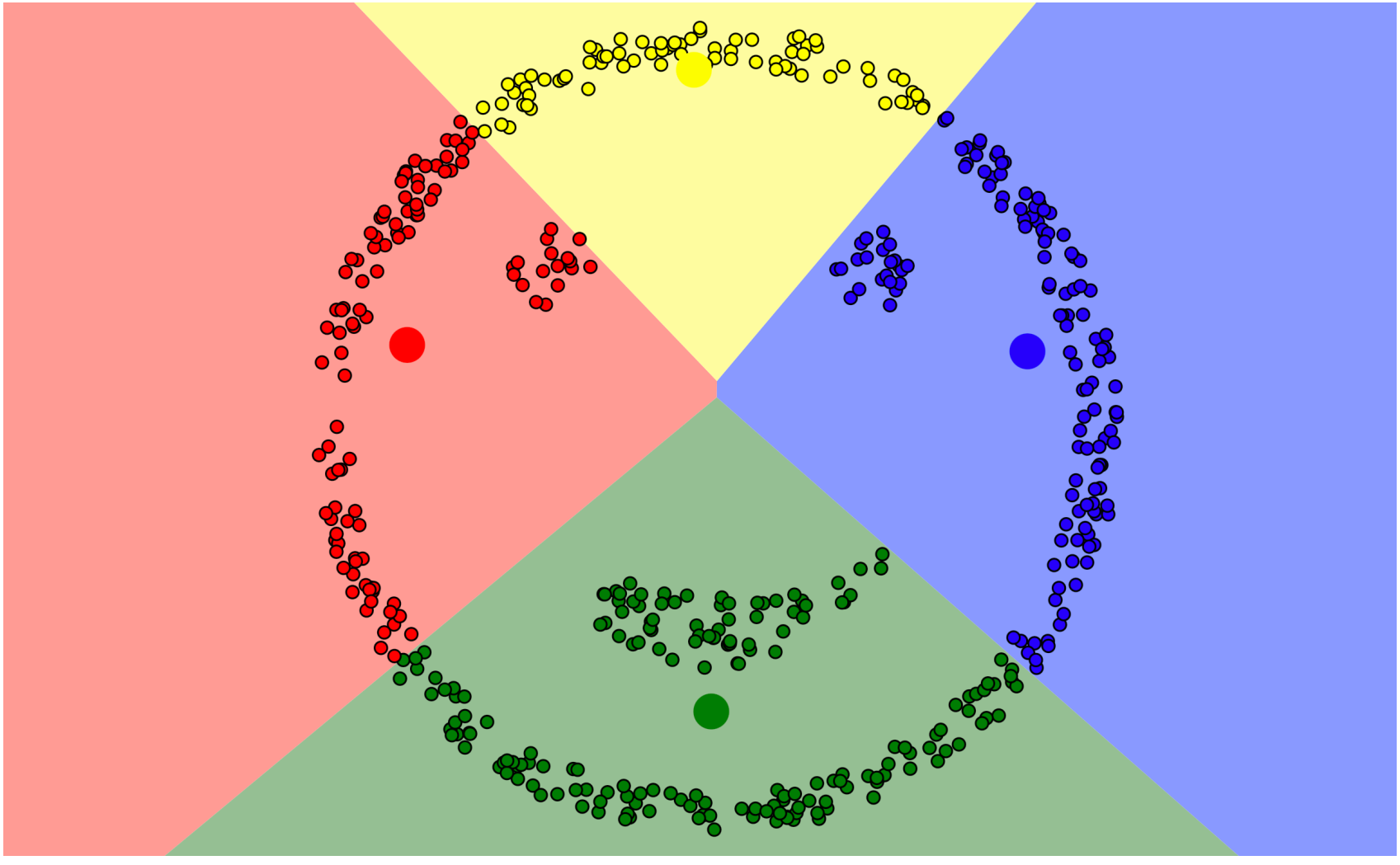
- Strategy: run k-Means Clustering for many choices of k (e.g. $k = 1, 2, 3, ..., 8$).
- Compute the value of inertia for each resulting set of centroids.
- Plot a graph of inertia vs k.
- Choose the value of k that appears at an “elbow”.



See the notebook for a demo.

Low inertia isn't everything!

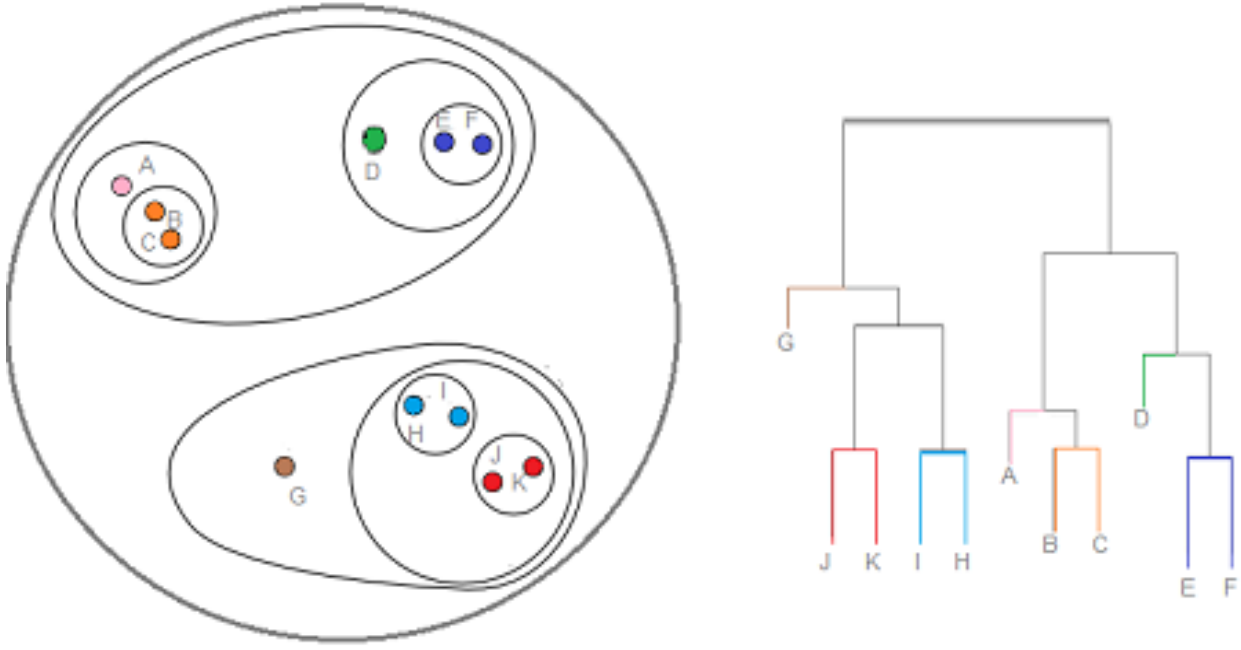
- Even if k-Means works as intended and finds the choice of centroids that minimize inertia, the resulting clustering may not look "right" to us humans.
  - Recall, inertia measures the total squared distance to centroids.
  - This metric doesn't always match our intuition.
- Let's look at some examples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



Other clustering techniques

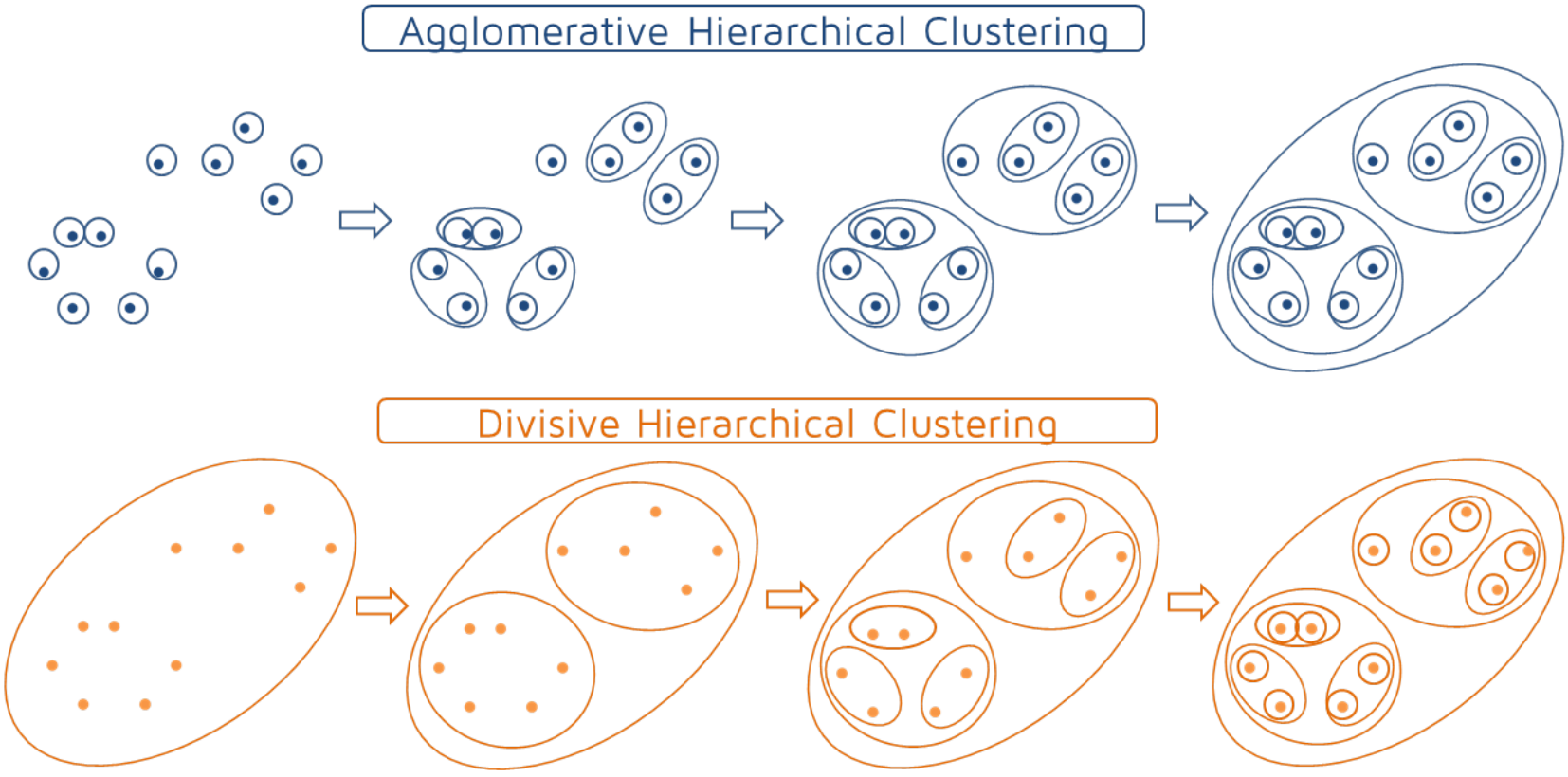
- k-Means Clustering is just one way to cluster data.
- There are many others, each of which work differently and produce different kinds of results.
- Another common technique: **agglomerative (hierarchical) clustering**.
  - High level: start out with each point being in its own cluster. Repeatedly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

Agglomerative hierarchical clustering



- Agglomerative clustering is a “bottom-up” method for creating hierarchical clusters.
- A dendrogram is a diagram representing a tree. For example, the right figure is a dendrogram representing the hierarchical clustering.

Agglomerative vs. Divisive

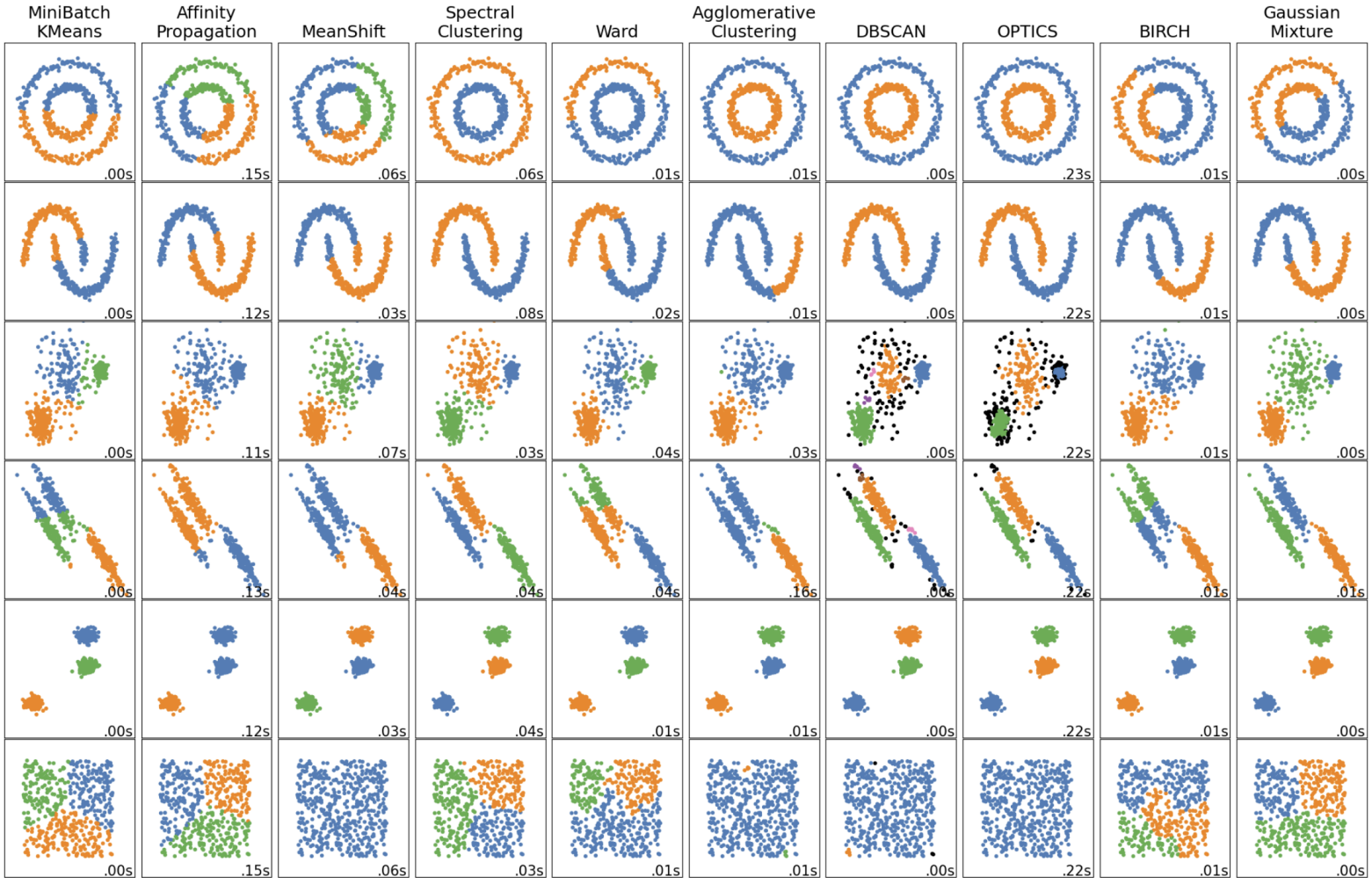


Summary of clustering methods

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes, outlier removal, transductive	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points
Bisecting K-Means	number of clusters	Very large `n_samples`, medium `n_clusters`	General-purpose, even cluster size, flat geometry, no empty clusters, inductive, hierarchical	Distances between points

https://scikit-learn.org/stable/modules/clustering.html

Summary of clustering methods



A comparison of the clustering algorithms in scikit-learn

https://scikit-learn.org/stable/modules/clustering.html

Introduction to Probability

Why do we need probability in Machine Learning?

- Probability is one of the foundations of Machine Learning.
- Learning algorithms will make decisions using probability. Classification models must predict a probability of class membership.
- Algorithms are designed using probability (e.g. Naive Bayes or Gaussian Mixture Models, etc.).
- The data we collect/observe can be understood as samples from some probability distribution.

What is probability?

Informally, a probability distribution $p : X \to \mathbb{R}$ over some domain X is a function such that $\sum_{x \in X} p(x) = 1$ and $p(x) \geq 0$ for all $x \in X$.

What is probability?

Informally, a probability distribution $p : X \to \mathbb{R}$ over some domain X is a function such that $\sum_{x \in X} p(x) = 1$ and $p(x) \geq 0$ for all $x \in X$.

Example:

- We toss a fair coin, what is the probability of getting head and tail?

What is probability?

Informally, a probability distribution $p : X \to \mathbb{R}$ over some domain X is a function such that $\sum_{x \in X} p(x) = 1$ and $p(x) \geq 0$ for all $x \in X$.

Example:

- We toss a fair coin, what is the probability of getting head and tail? $p(\text{head}) = p(\text{tail}) = 0.5$

- We toss a fair dice, what is the probability of getting 1, 2, 3, 4, 5, and 6, respectively?

What is probability?

Informally, a probability distribution $p : X \to \mathbb{R}$ over some domain X is a function such that $\sum_{x \in X} p(x) = 1$ and $p(x) \geq 0$ for all $x \in X$.

Example:

- We toss a fair coin, what is the probability of getting head and tail? $p(\text{head}) = p(\text{tail}) = 0.5$
- We toss a fair dice, what is the probability of getting 1, 2, 3, 4, 5, and 6, respectively? $p(i) = \frac{1}{6}, \forall i \in \{1, .., 6\}$
- If we toss the fair coin and the fair dice infinite number of times, what does the frequencies look like?

What is probability?

Informally, a probability distribution $p : X \to \mathbb{R}$ over some domain X is a function such that $\sum_{x \in X} p(x) = 1$ and $p(x) \geq 0$ for all $x \in X$.

Example:

- We toss a fair coin, what is the probability of getting head and tail? $p(\text{head}) = p(\text{tail}) = 0.5$
- We toss a fair dice, what is the probability of getting 1, 2, 3, 4, 5, and 6, respectively? $p(i) = \frac{1}{6}, \forall i \in \{1, .., 6\}$
- If we toss the fair coin and the fair dice infinite number of times, what does the frequencies look like? Tends to the uniform distribution.