

# CSE/STAT 416

## k-means Clustering

Hunter Schafer  
Paul G. Allen School of Computer Science & Engineering  
University of Washington

May 15, 2023

? Questions? Raise hand or [sli.do](https://sli.do) #cs416  
🎵 Listening to: Peach Pit



# Clustering Overview

# Recap

For the past 7 weeks, we have covered different **supervised learning** algorithms

Now, we're going to explore **unsupervised learning** methods where don't have labels / outputs in your datasets anymore.

Note that several of the concepts you learnt for supervised learning, such as cross-validation, overfitting, bias-variance tradeoff, accuracy, error, etc. no longer apply in unsupervised learning!



# Unsupervised Learning

A type of machine learning that detects underlying patterns in **unlabeled** data.

Examples of unsupervised learning tasks:

- Cluster similar articles together.

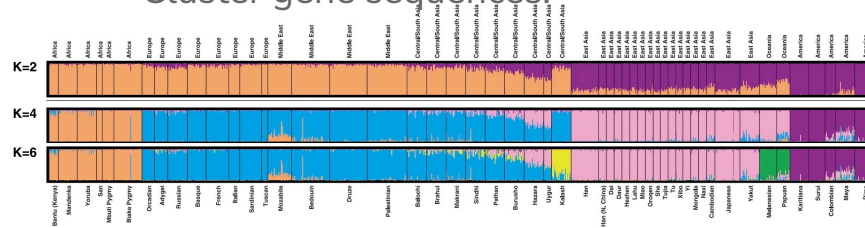
## Coupled indoor navigation for people who are blind

[A Nanavati](#), [XZ Tan](#), [A Steinfeld](#) - Companion of the 2018 ACM/IEEE ..., 2018 - dl.acm.org

This paper presents our design of an autonomous navigation system for a mobile robot that guides people who are blind and low vision in indoor settings. It begins by presenting user ...

☆ Save 📄 Cite Cited by 11 **Related articles**

- Cluster gene sequences.



- Recommend items, searches, movies, etc.

- unsupervised
- unsupervised learning
- unsupervised recommender system
- unsupervised learning recommendation system
- unsupervised learning example
- unsupervised machine learning
- unsupervised
- unsupervised learning algorithms
- Unsupervised Sitcom
- unsupervised clustering
- unsupervised vs supervised learning

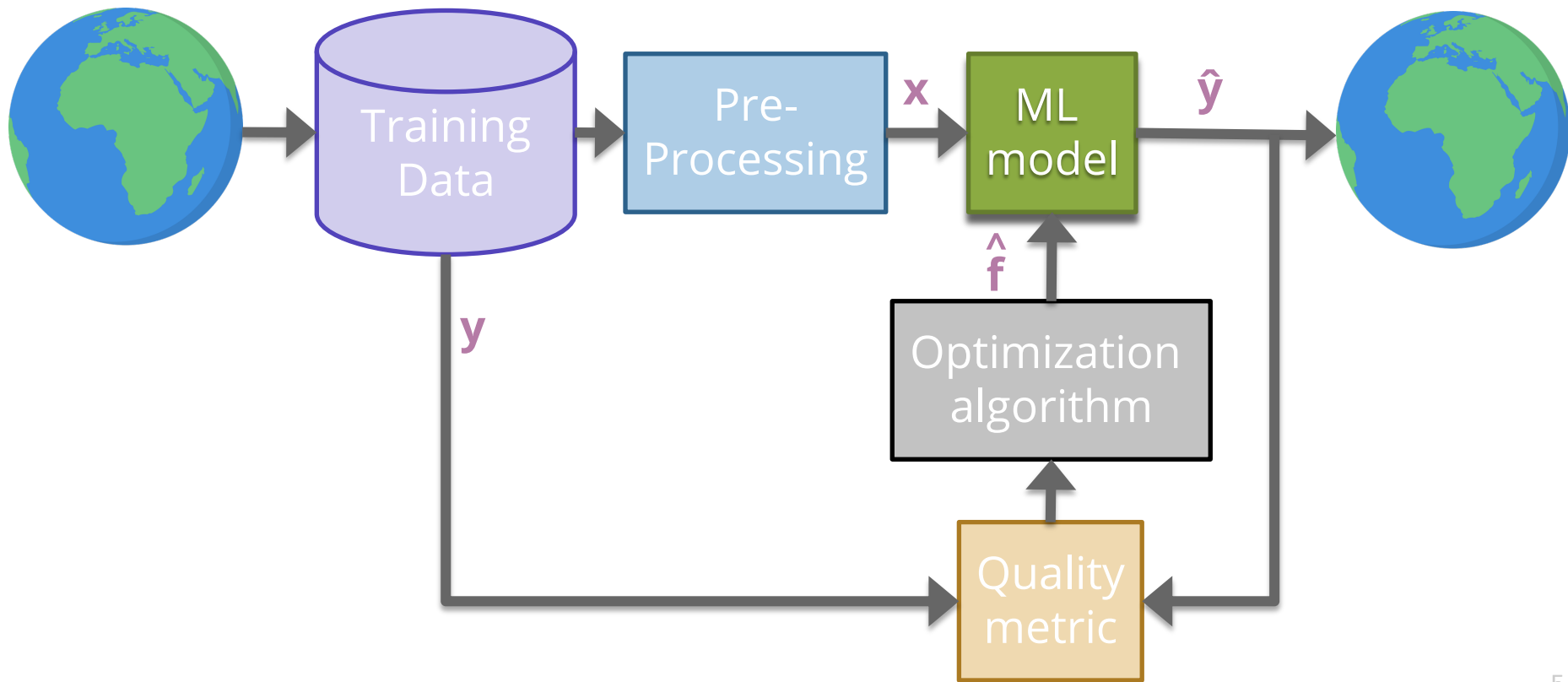
Products related to this item

Sponsored @

Page 1 of 55

< >





# Clustering



## SPORTS

## WORLD NEWS

Note that we're not talking about learning user preferences (yet – come back next week 😊).

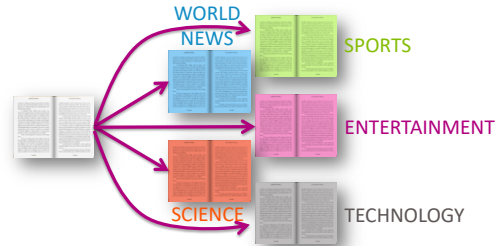
Our case study is **document retrieval**. Given that someone read a particular article, what similar articles would you recommend (without personalization)?

# Labeled Data

What if the labels are known? Given labeled training data.



Can do multi-class classification methods to predict label.



However, not all articles fit cleanly into one label.

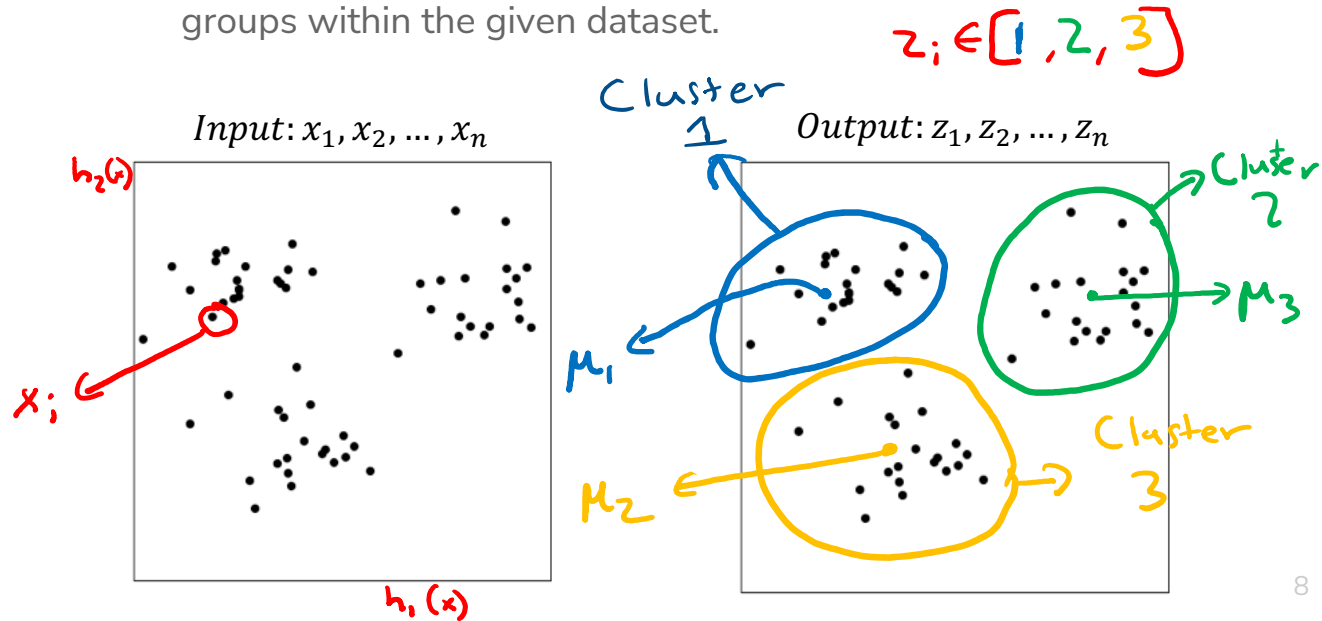
Further, oftentimes real-world data doesn't have labels.

# Unlabeled Data

In many real world contexts, there aren't clearly defined labels so we won't be able to do classification

We will need to come up with methods that uncover structure from the (unlabeled) input data  $X$ .

**Clustering** is an automatic process of trying to find related groups within the given dataset.



# Define Clusters

In their simplest form, a **cluster** is defined by

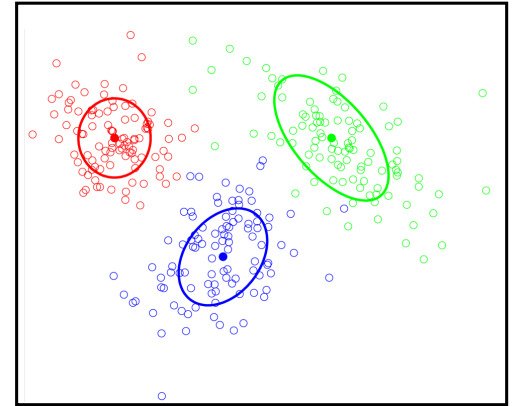
- The location of its center (**centroid**)
- Shape and size of its **spread**

**Clustering** is the process of finding these clusters and **assigning** each example to a particular cluster.

- $x_i$  gets assigned  $z_i \in [1, 2, \dots, k]$
- Usually based on closest centroid

Will define some kind of objective function for a clustering that determines how good the assignments are

- Based on distance of assigned examples to each cluster.
- Close distance reflects strong similarity between datapoints.

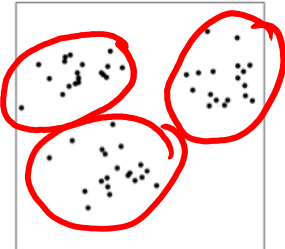
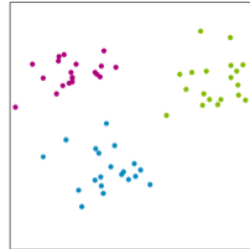


# When Might This Work?

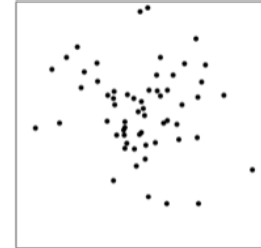
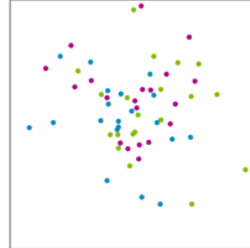
Clustering is easy when distance captures the clusters.

Ground Truth (not visible)

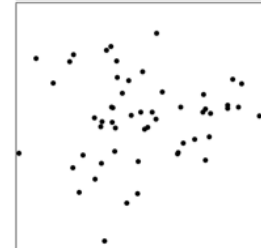
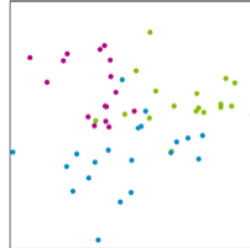
Given Data



Easy?



Impossible?



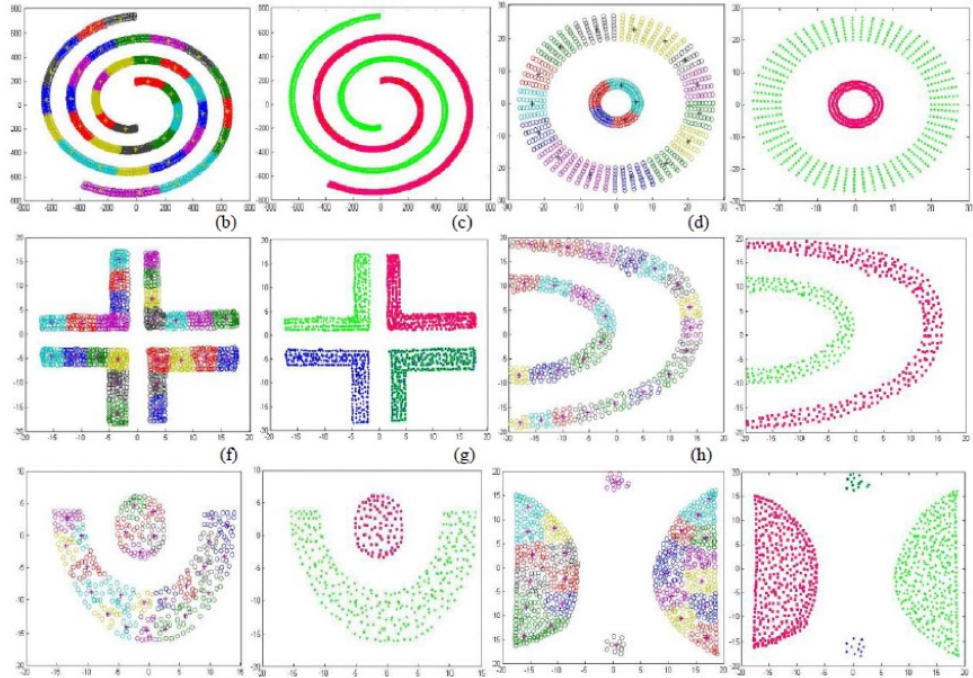
Maybe?



# Not Always Easy

There are many clusters that are harder to learn with this setup

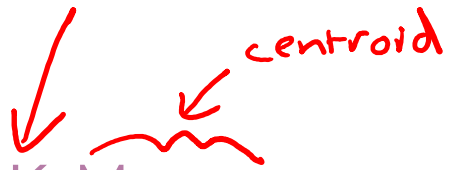
Distance does not determine clusters







Hyperparameter




K-Means  
Clustering

# K-Means Clustering Algorithm

We define the criterion of assigning point to a cluster based on **its distance**.

Shorter distance => Better Clustering

*Hyper parameter  $k$*



## Algorithm

Given a dataset of  $n$  datapoints and a particular choice of  $k$

Step 0: Initialize cluster centroids randomly

Repeat until convergence:

Step 1: Assign each example to its closest cluster centroid

Step 2: Update the centroids to be the average of all the points assigned to that cluster

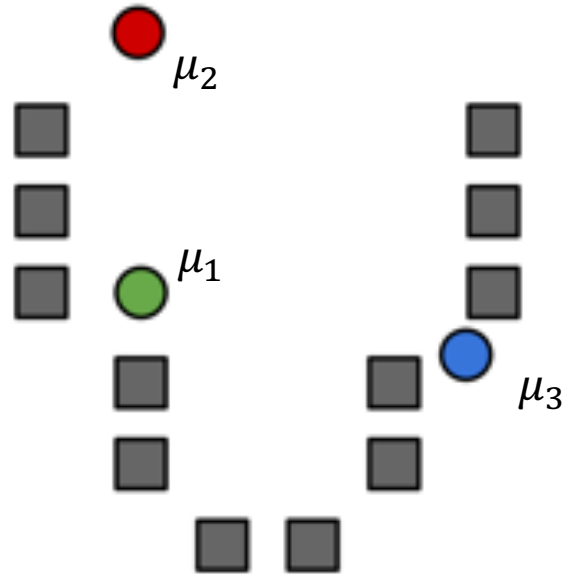


# Step 0

Start by choosing the initial cluster centroids

A common default choice is to choose centroids  $\mu_1, \dots, \mu_k$  randomly

Will see later that there are smarter ways of initializing

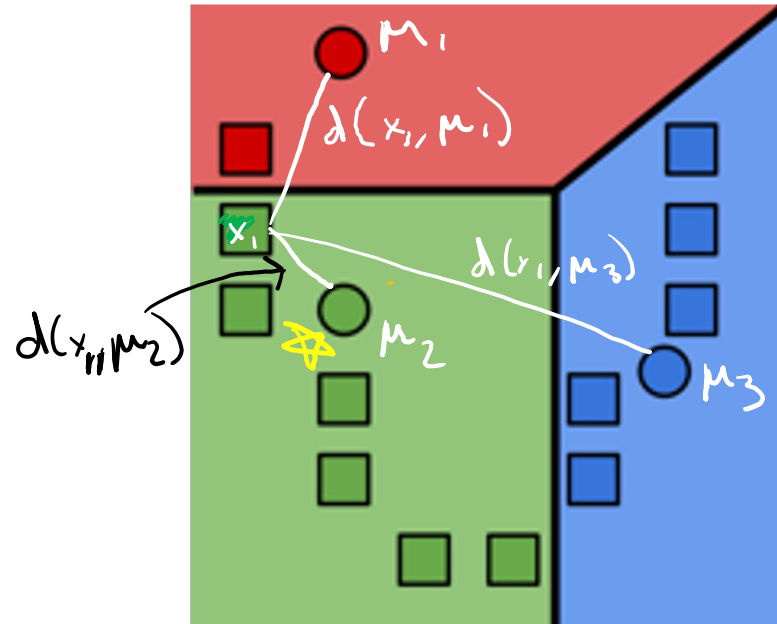


# Voronoi Tesselation

Assign each example to its closest cluster centroid

For  $i = 1$  to  $n$

$$z_i \leftarrow \operatorname{argmin}_{j \in [k]} \|\mu_j - x_i\|_2^2$$

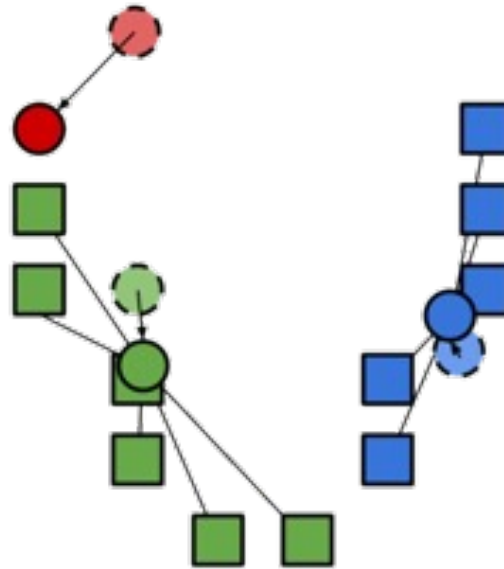


## Step 2

Update the centroids to be the mean of points assigned to that cluster.

$$\mu_j = \frac{\sum_{i=1}^n \mathbf{1}\{z_i = j\} x_i}{\sum_{i=1}^n \mathbf{1}\{z_i = j\}}$$

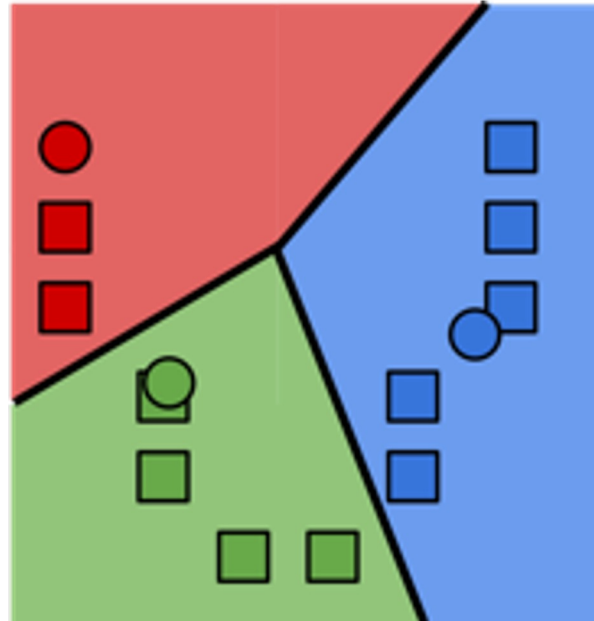
Computes center of mass for cluster!



= sum of datapoints assigned to cluster j  
= number of datapoints assigned to cluster j

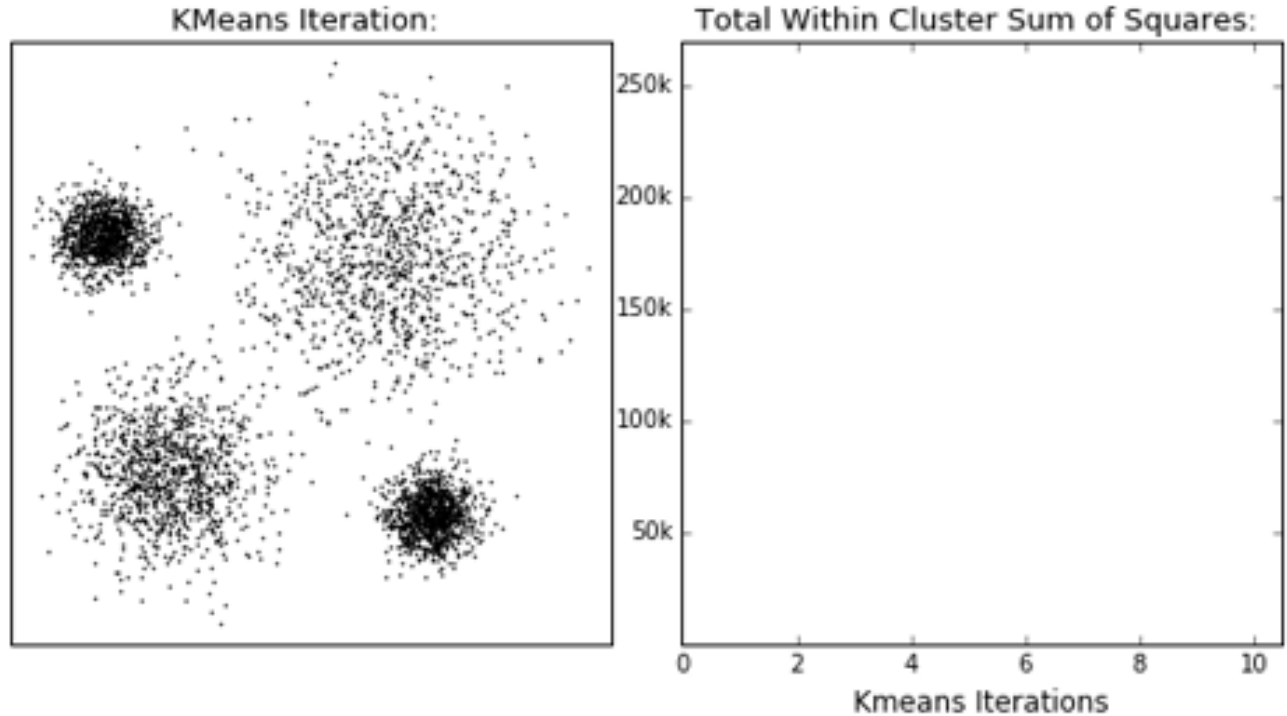
Repeat  
until  
convergence

Repeat Steps 1 and 2 until convergence



# Stopping Conditions

- Cluster assignments haven't changed
- Centroids haven't changed
- Some number of max iterations have been passed



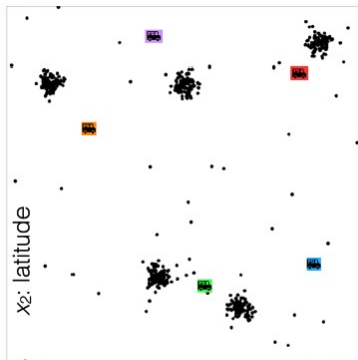
# slido

Think 

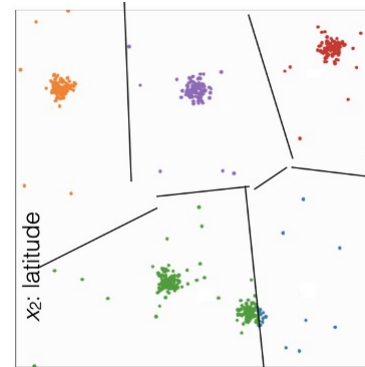
1 min

slido #cs416

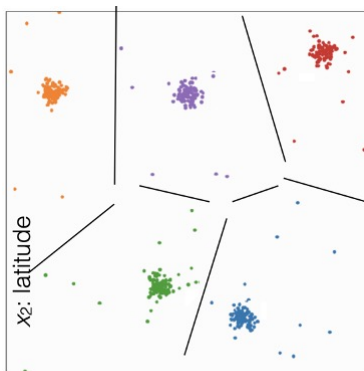
■ What cluster assignment would result from these centroids?



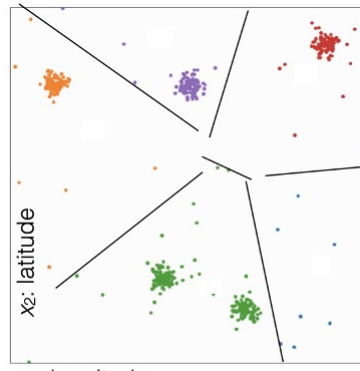
$x_1$ : longitude  
 $x_2$ : latitude  
Centroids



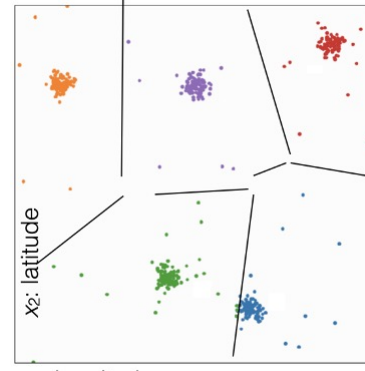
$x_1$ : longitude  
 $x_2$ : latitude  
(D)



$x_1$ : longitude  
 $x_2$ : latitude  
(A)



$x_1$ : longitude  
 $x_2$ : latitude  
(B)



$x_1$ : longitude  
 $x_2$ : latitude  
(C)



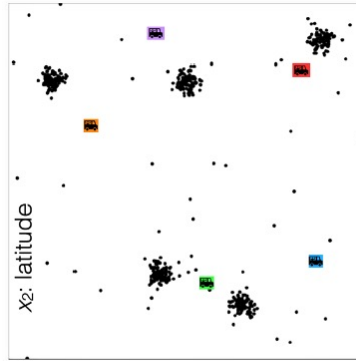
# slido

Group 

1 min

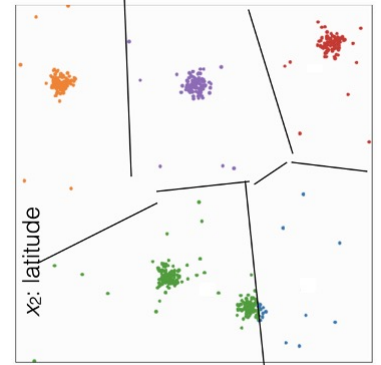
slido #cs416

What cluster assignment would result from these centroids?



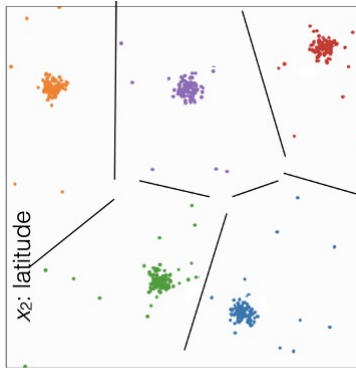
x<sub>1</sub>: longitude

Centroids



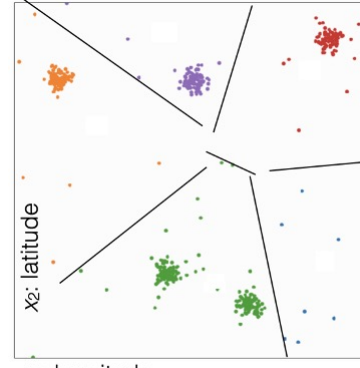
x<sub>1</sub>: longitude

(D)



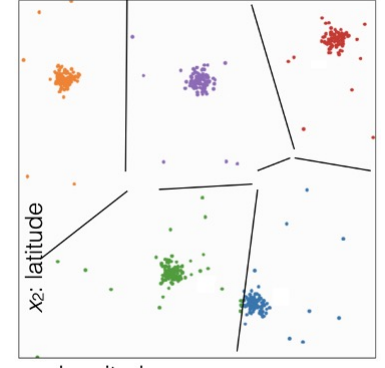
x<sub>1</sub>: longitude

(A)



x<sub>1</sub>: longitude

(B)



x<sub>1</sub>: longitude

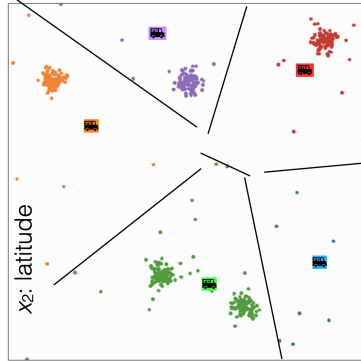
(C)

# slido

Think 

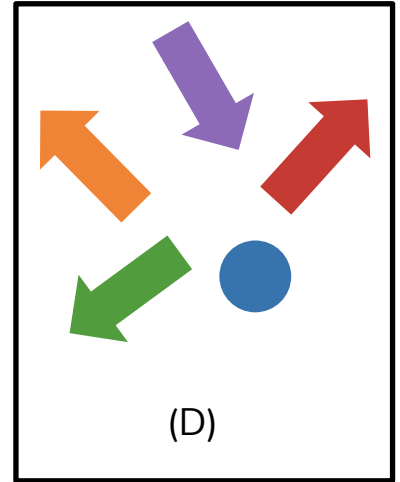
1 min

In what direction would each of the centroids (roughly) move?

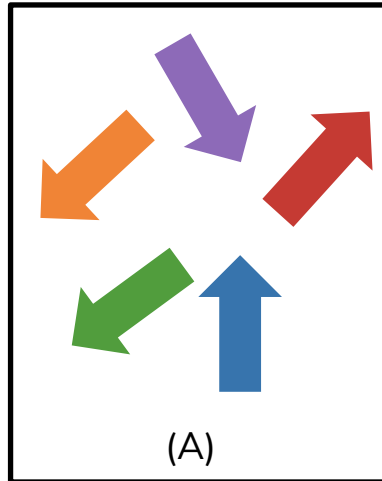


x<sub>1</sub>: longitude

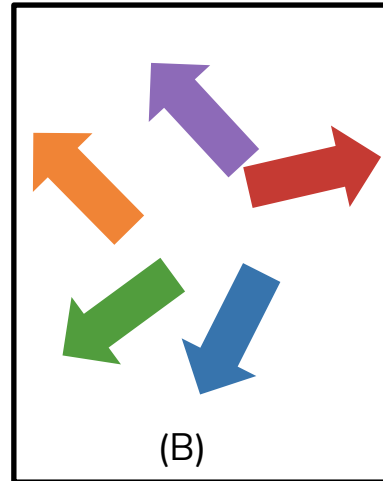
Cluster Assignments



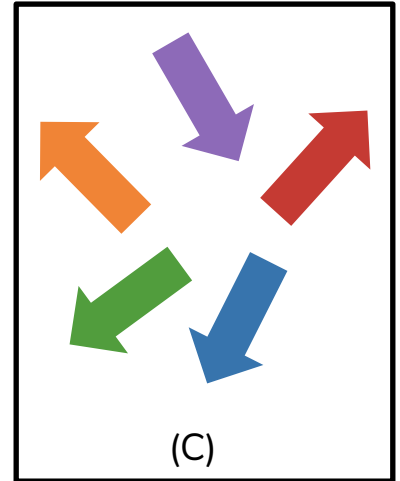
(D)



(A)



(B)



(C)

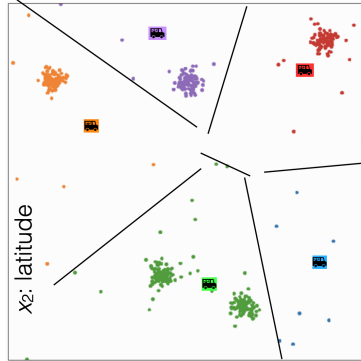
slido #cs416

# slido

Group 

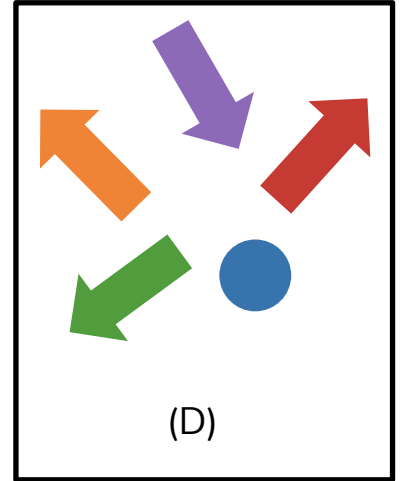
2 min

In what direction would each of the centroids (roughly) move?

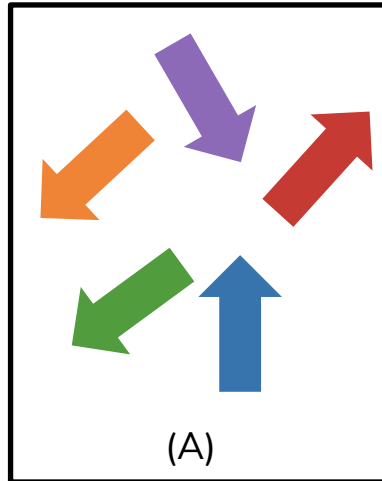


x<sub>1</sub>: longitude

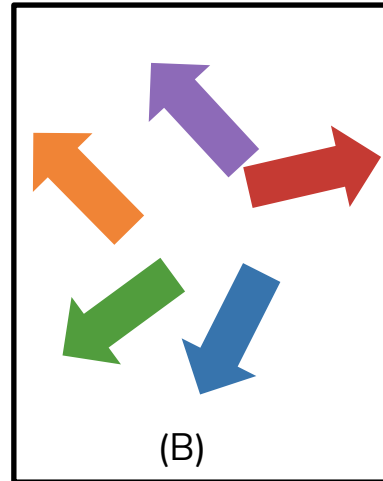
Cluster Assignments



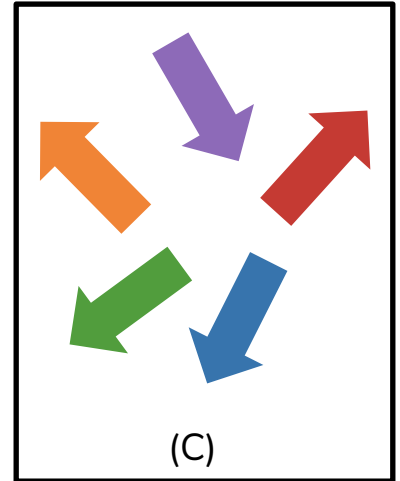
(D)



(A)

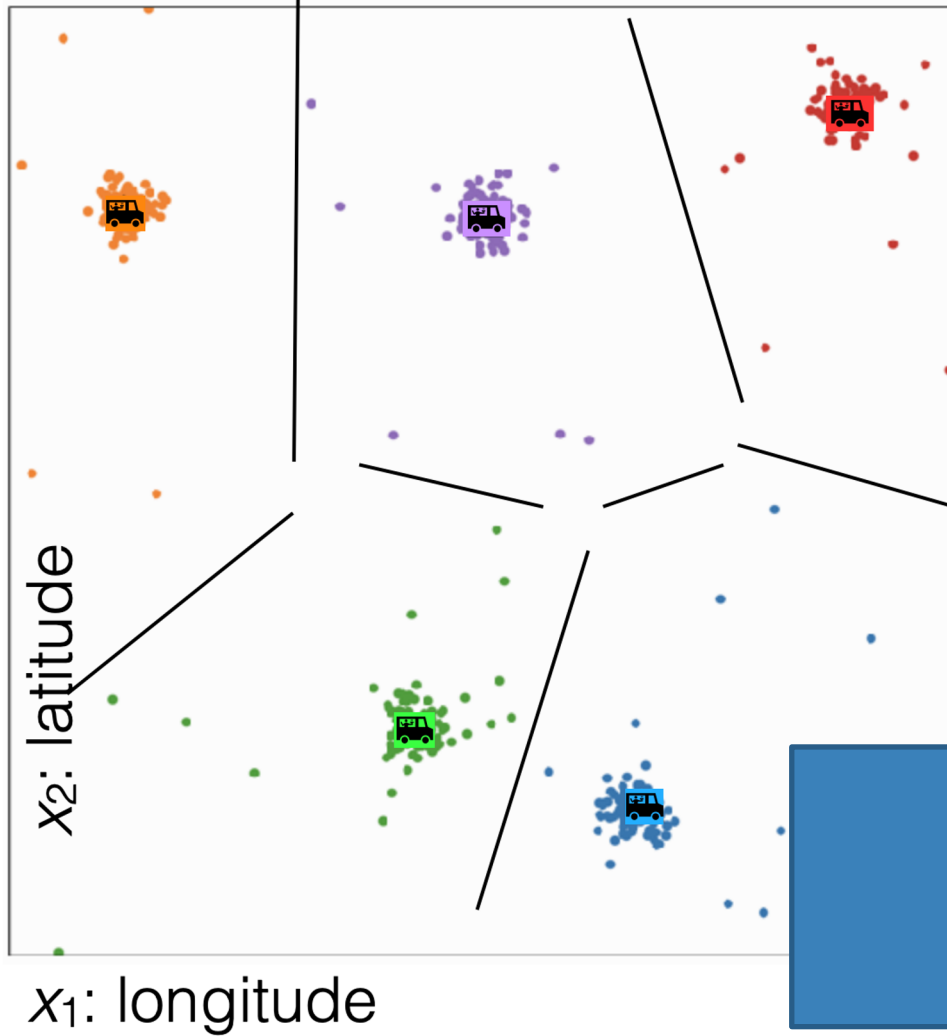


(B)



(C)

slido #cs416

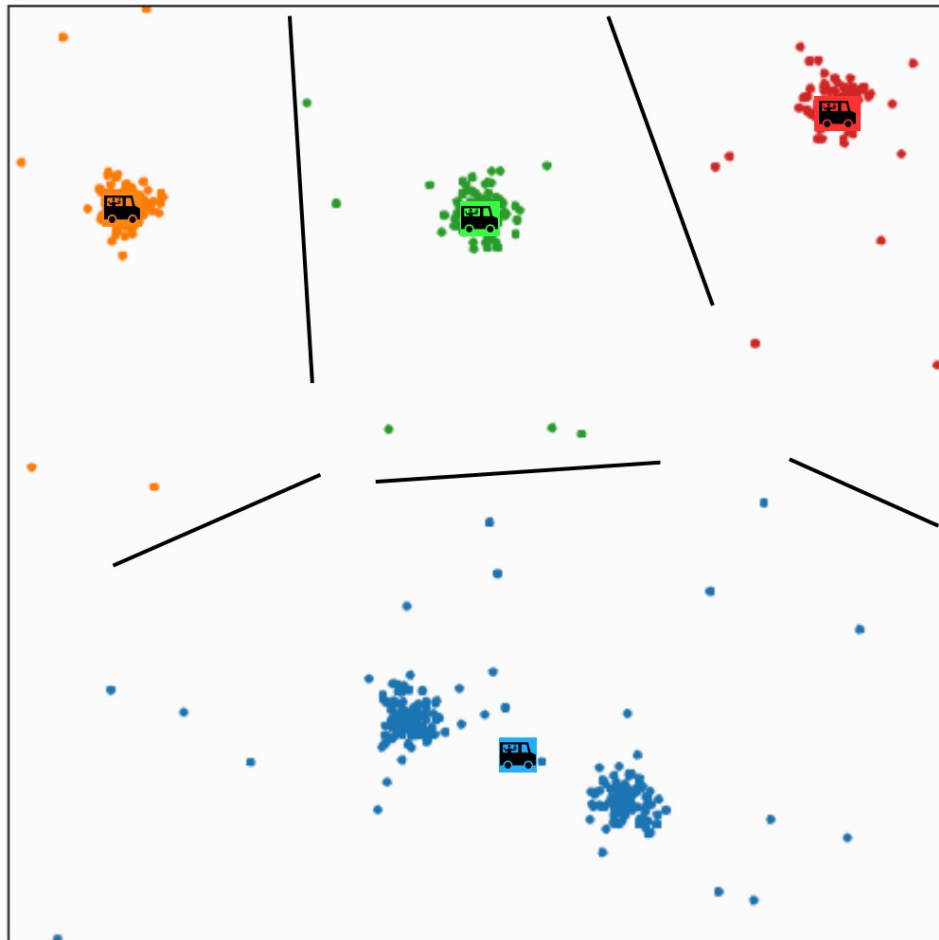
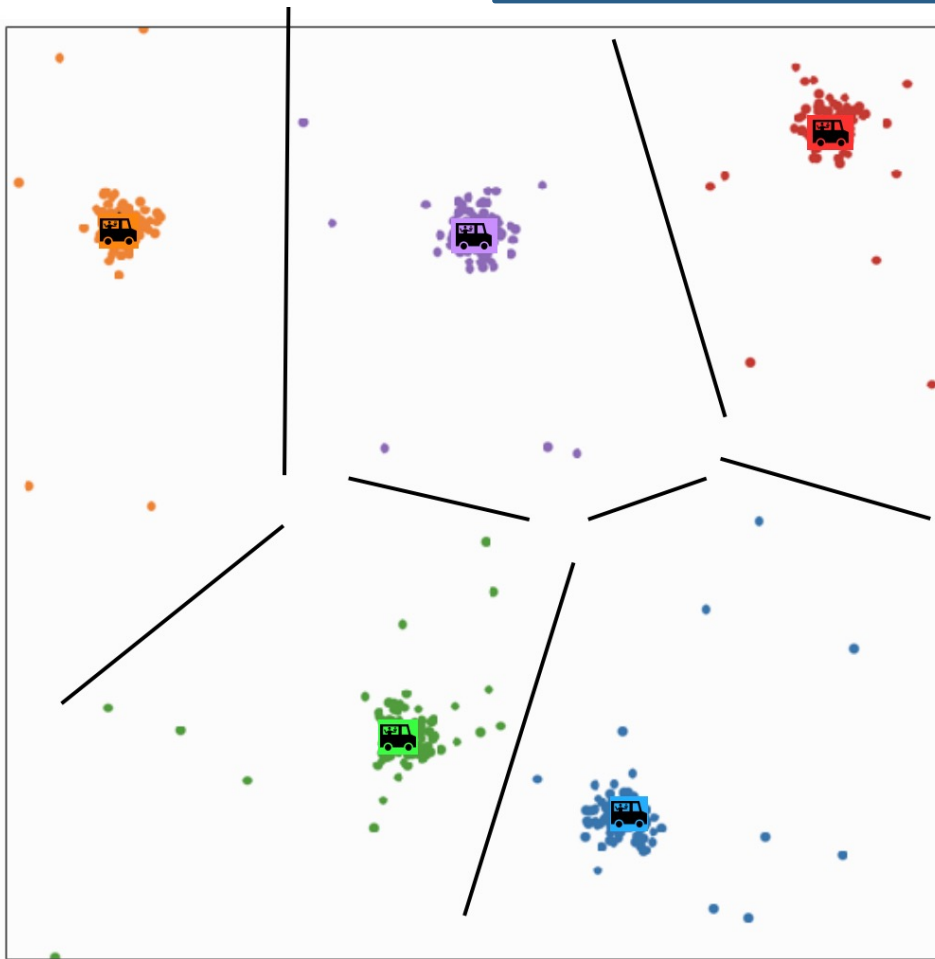


No more changes

$k=5$

Different  $k$  gives different results

$k=4$



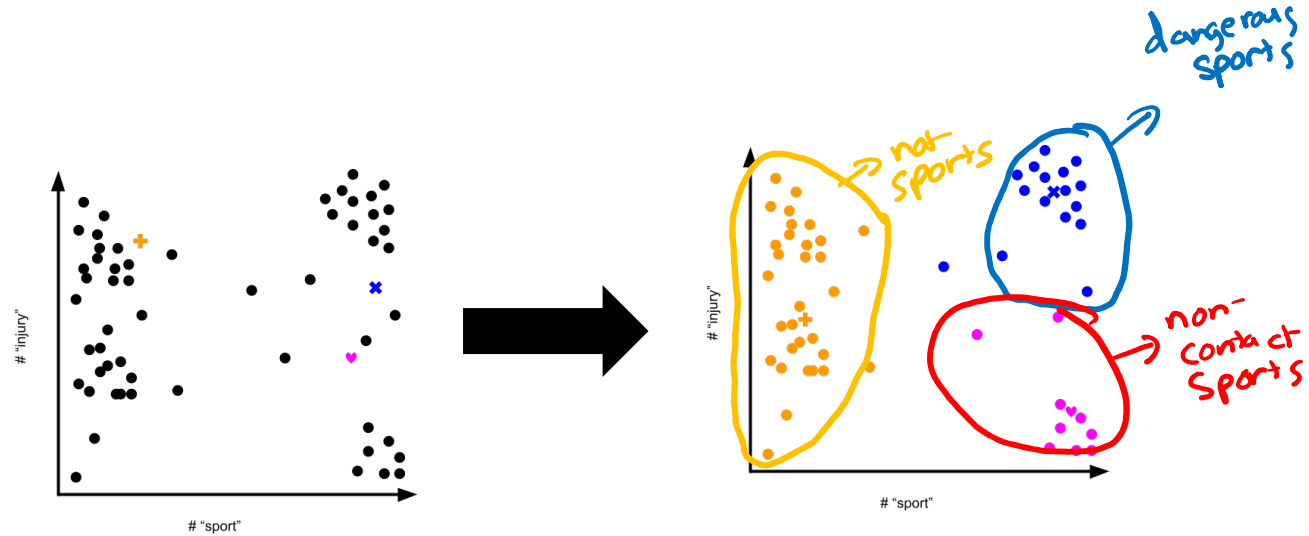
# slido

Group 

1.5 min

You are clustering news articles using the features “# sport” and “# injury.” How would you interpret these clusters?

“This is a cluster of ...[some characterization]... articles.”





## Brain Break



# Effect of Initialization



# slido

Group 

1.5 min

slido #cs416

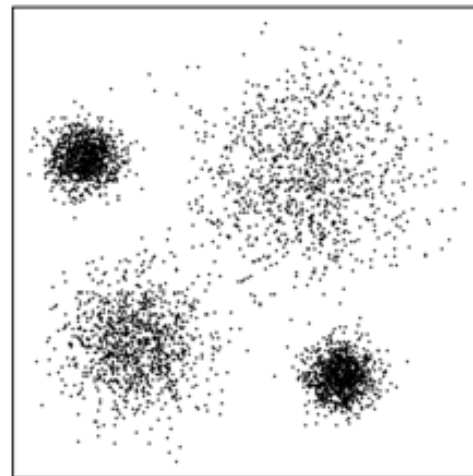
What convergence guarantees do you think we will have with k-means?

Converges to the global optimum

Converges to a local optima

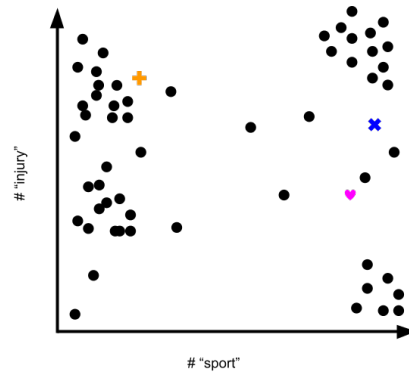
None

KMeans Iteration:

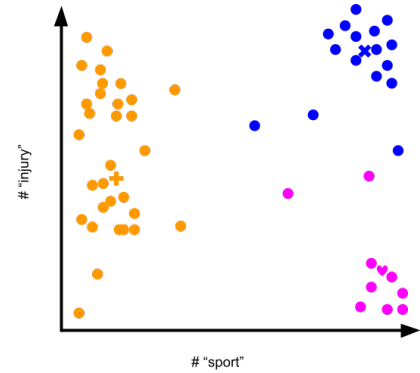


# Effects of Initialization

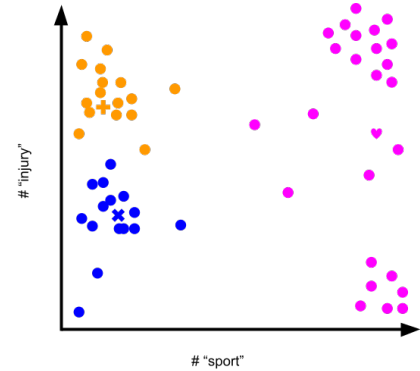
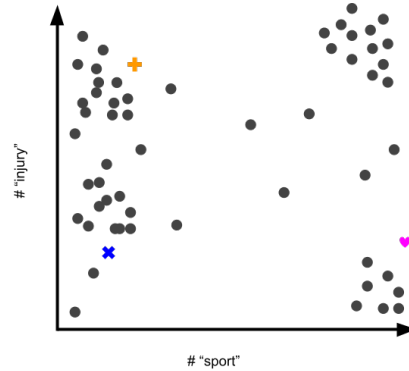
Results **heavily depend** on initial centroids



Initialization



Final Clusters

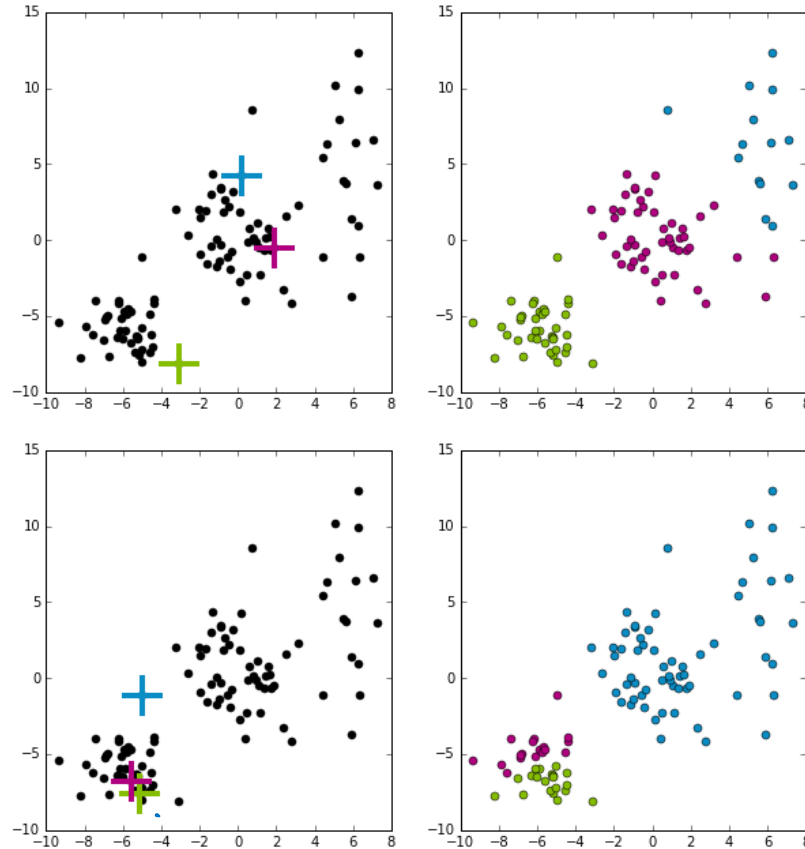


# Effect of initialization

What does it mean for something to converge to a local optima?

Some initialization can be bad and affect the quality of clustering

Initialization will greatly impact results!



# Smart Initializing w/ k-means++

Making sure the initialized centroids are “good” is critical to finding quality local optima. Our purely random approach was wasteful since it’s very possible that initial centroids start close together.

**Idea:** Try to select a set of points farther away from each other.

**k-means++** does a slightly smarter random initialization

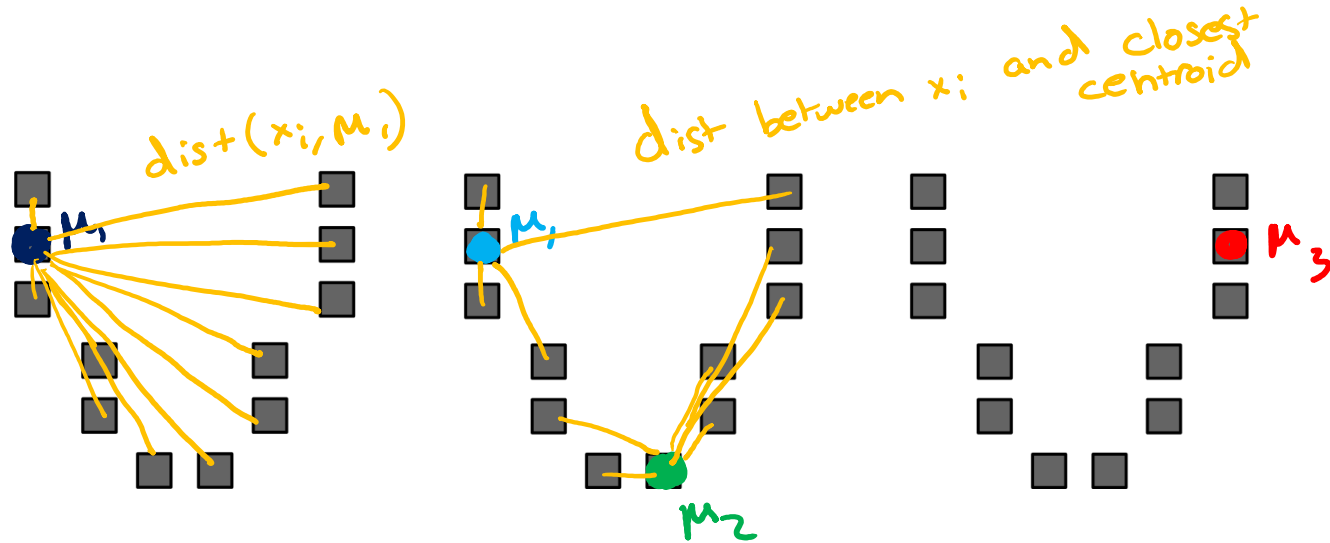
1. Choose first cluster  $\mu_1$  from the data uniformly at random
2. For each datapoint  $x_i$ , compute the distance between  $x_i$  and the closest centroid from the current set of centroids (starting with just  $\mu_1$ ). Denote that distance  $d(x_i)$ .
3. Choose a new centroid from the remaining data points, where the probability of  $x_i$  being chosen is proportional to  $d(x_i)^2$ .
4. Repeat 2 and 3 until we have selected  $k$  centroids.

# k-means++ Example

Start by picking a point at random

Then pick points proportional to their distances to their centroids

This tries to maximize the spread of the centroids!



# k-means++

## Pros / Cons

### Pros

Improves quality of local minima

Faster convergence to local minima

### Cons

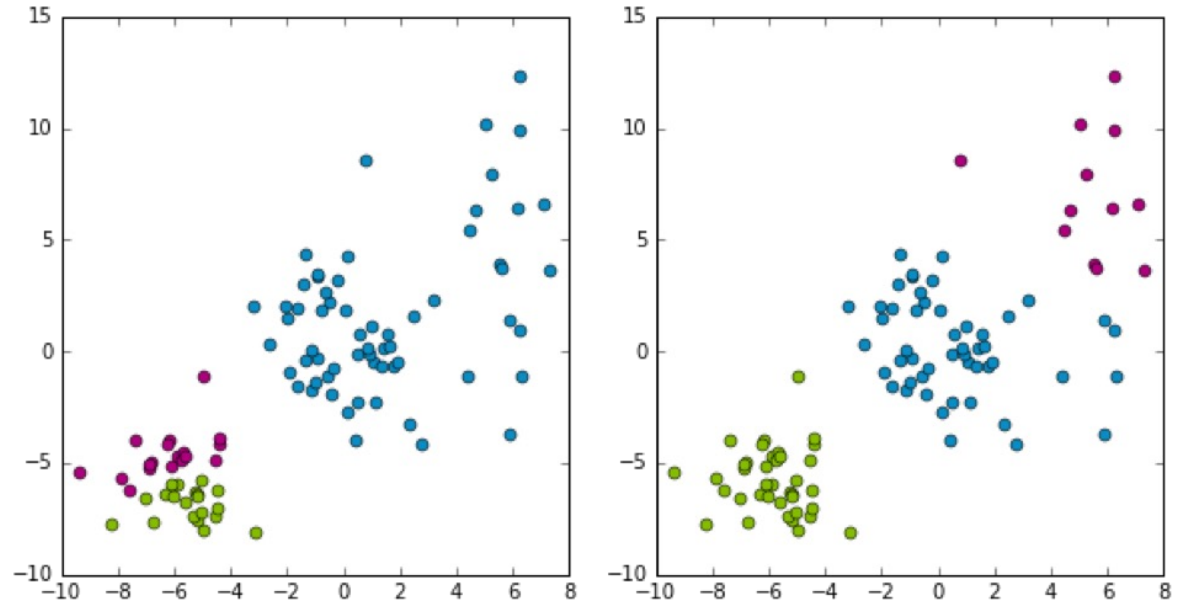
Computationally more expensive at beginning when compared to simple random initialization



# Assessing Performance

# Which Cluster?

Which clustering would I prefer?



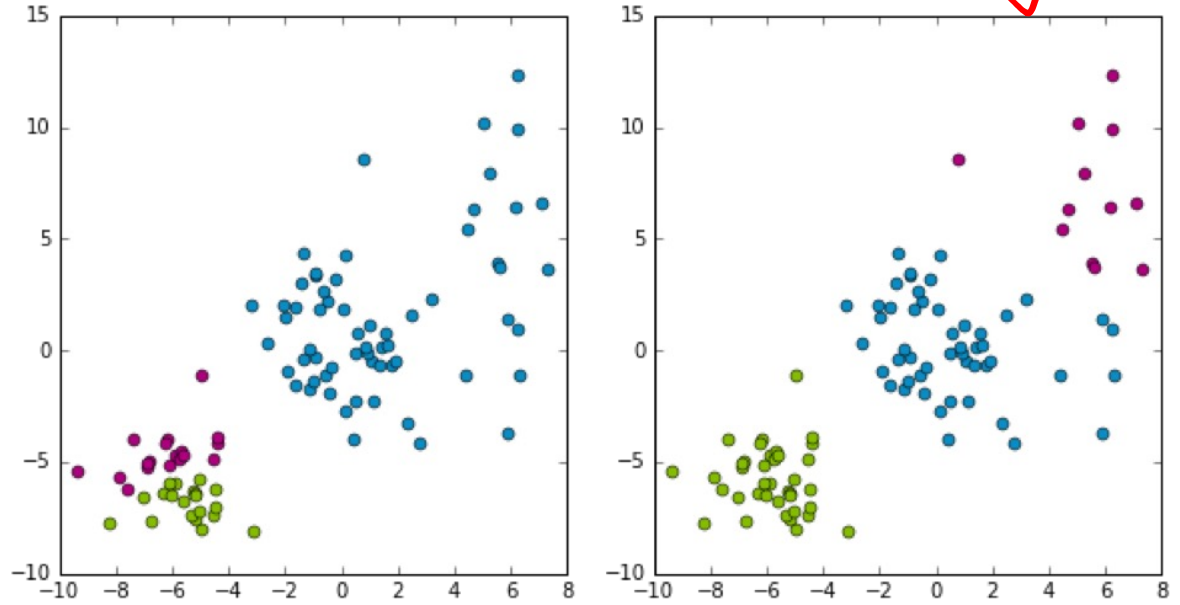
Don't know, there is no “right answer” in clustering 🙄.

Depends on the practitioner's domain-specific knowledge and interpretation of results!



# Which Cluster?

Which clustering does k-means prefer?



k-means is trying to optimize the **heterogeneity** objective

$$\operatorname{argmin}_{z, \mu} \sum_{j=1}^k \sum_{i=1}^n \mathbf{1}\{z_i = j\} \left\| \mu_j - x_i \right\|_2^2$$

sum over clusters  $\rightarrow$   $\sum_{j=1}^k$

$\rightarrow$   $\sum_{i=1}^n$

$\rightarrow$   $\mathbf{1}\{z_i = j\}$

$\rightarrow$   $\left\| \mu_j - x_i \right\|_2^2$

total distance between points in that cluster and the centroid

# Coordinate Descent

k-means is trying to minimize the heterogeneity objective

$$\underset{Z, \mu}{\operatorname{argmin}} \sum_{j=1}^k \sum_{i=1}^n \mathbf{1}\{z_i = j\} \|\mu_j - x_i\|_2^2$$

Step 0: Initialize cluster centers

Repeat until convergence:

*Fix  $\mu$ , minimize  $Z$*

Step 1: Assign each example to its closest cluster centroid

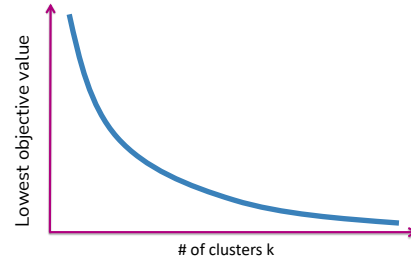
Step 2: Update the centroids to be the mean of all the points assigned to that cluster

*Fix  $Z$ , minimize  $\mu$*

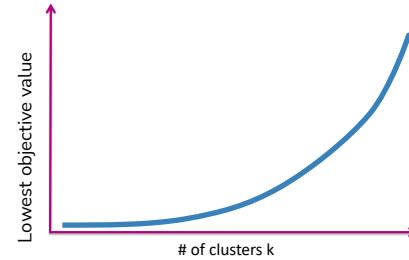
**Coordinate Descent** alternates how it updates parameters to find minima. On each of iteration of Step 1 and Step 2, heterogeneity decreases or stays the same.

=> Will converge in finite time

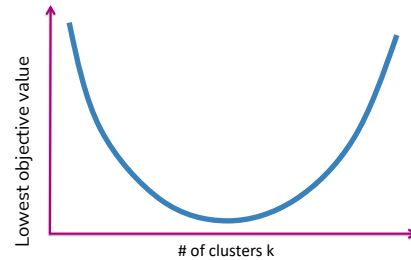
Consider training k-means to convergence for different values of  $k$ . Which of the following graphs shows how the heterogeneity objective will change based on the value of  $k$ ?



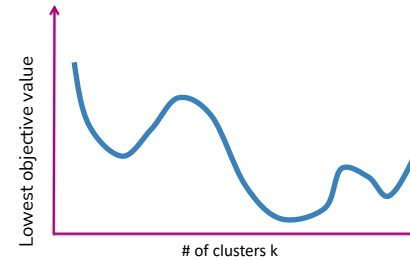
A



B



C



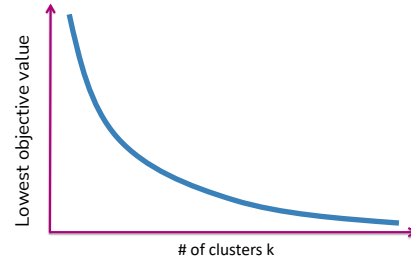
D

# slido

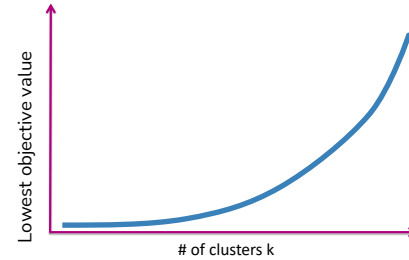
Group 

2 mins

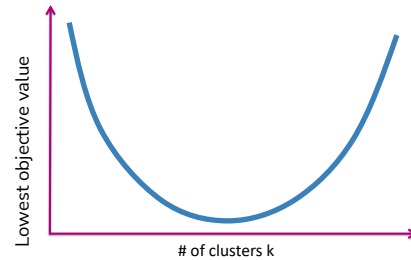
Consider training k-means to convergence for different values of  $k$ . Which of the following graphs shows how the heterogeneity objective will change based on the value of  $k$ ?



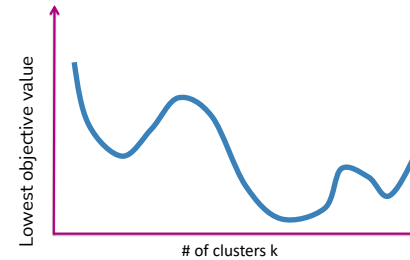
A



B



C

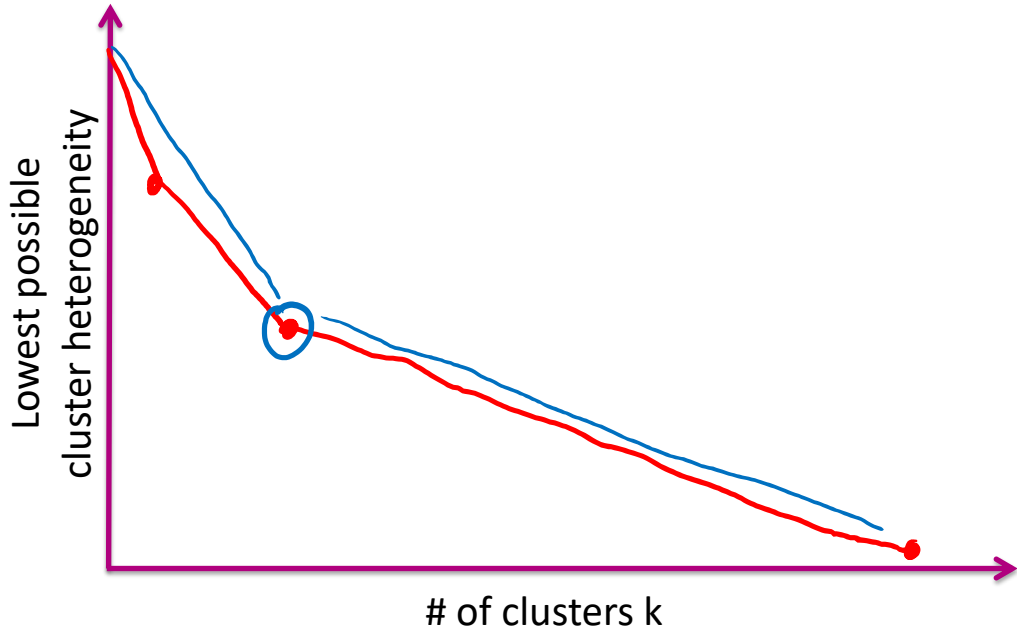


D

# How to Choose k?

No right answer! Depends on your application.

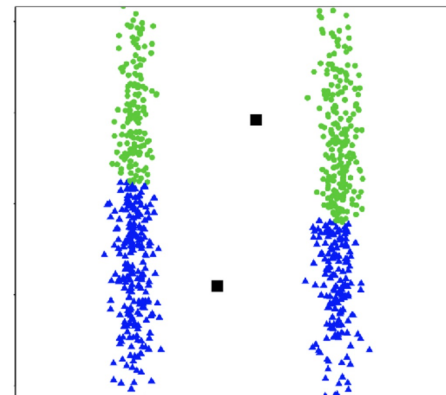
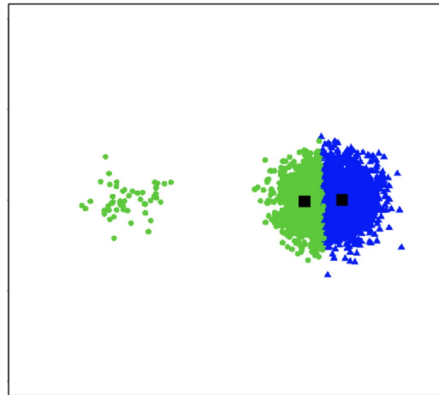
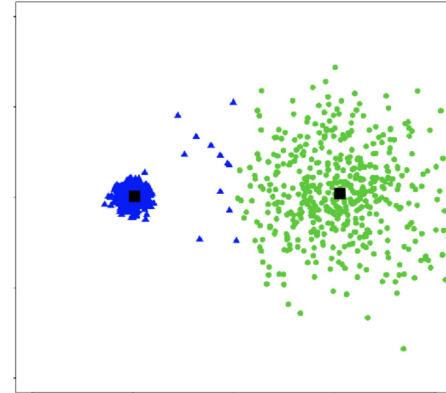
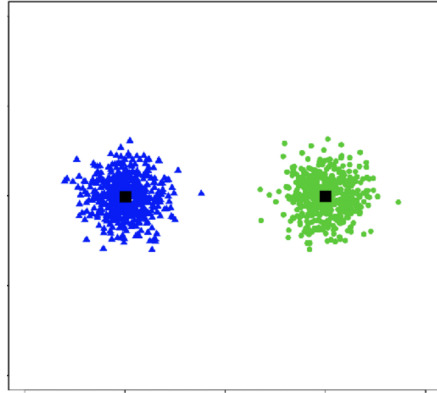
General, look for the “elbow” in the graph



Note: You will usually have to run k-means multiple times for each k

# Cluster shape

- k-means works well for well-separated **hyper-spherical** clusters of the same size



# Clustering VS Classification

Clustering looks like we assigned labels (by coloring or numbering different groups) but we didn't use any **labeled** data.

In clustering, the “labels” don't have meaning. To give meaning to the labels, human inputs is required

Classification learns from minimizing the error between a prediction and an actual **label**.

Clustering learns by minimizing the distance between points in a cluster.

Classification quality metrics (accuracy / loss) do not apply to clustering (since there is no label).

You can't use validation set / cross-validation to choose the best choice of k for clustering.



# Recap

Differences between classification and clustering

What types of clusters can be formed by k-means

K-means algorithm

Convergence of k-means

How to choose k

Better initialization using k-means++

