

CSE/STAT 416

k-means Clustering

Tanmay Shah
Paul G. Allen School of Computer Science & Engineering
University of Washington

July 29, 2024

? Questions? Raise hand

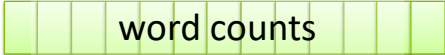


TF-IDF = Term Frequency \times Inverse Document Frequency \rightarrow

TF-IDF

Goal: Emphasize important words

Appear frequently in the document (common locally)

Term frequency =  word counts
"the" appears in Doc1 200 times, $TF("the", doc1) = 200 / total$
Appears rarely in the corpus (rare globally)

Inverse doc freq. = 
$$\log \frac{\# \text{ docs}}{1 + \# \text{ docs using word}}$$

"the" appears in all 1000 docs

$$IDF("the") = \log \left(\frac{1000}{1+1000} \right)$$
$$\approx \log(1) = 0$$



Do a pair-wise multiplication to compute the TF-IDF for each word

Words that appear in every document will have a small IDF making the TF-IDF small

$$\text{num features} \rightarrow D = W$$

\uparrow num unique words

Poll Everywhere

Think 

1.5 min

What is the $TF - IDF("rain", Doc_1)$ with the following documents (assume standard pre-processing)

Doc 1: It is going to rain today.

Doc 2: Today I am not going outside.

Doc 3: I am going to watch the season premiere.

$$TF-IDF("rain", Doc1) = TF("rain", Doc1) \cdot IDF("rain")$$

$$TF("rain", Doc1) = \frac{1}{6}$$

$$IDF("rain") = \log\left(\frac{3}{1+1}\right)$$

$$TF-IDF("rain", Doc1) = \frac{1}{6} \cdot \log(1.5) = 0.07$$

Some Issues

1. How do we find a good line that divides the data in half?
2. Potential Errors: Points close together might be split up into separate bins
3. Large computation cost: Only dividing the points in half doesn't speed things up that much...



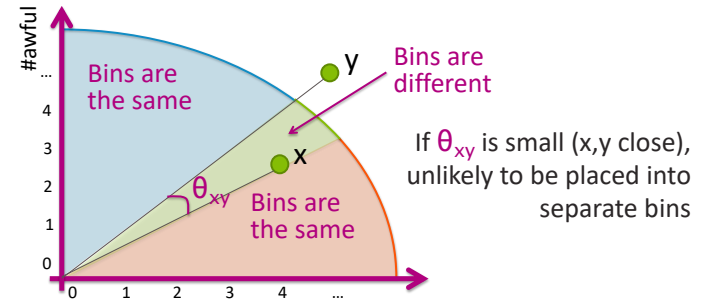
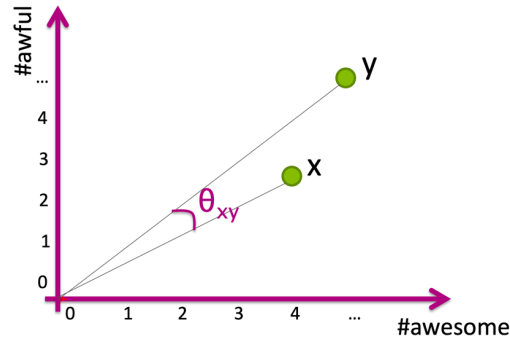
1. How to choose line?

Wild Idea: Choose the line randomly!

Choose a slope randomly between 0 and 90 degrees

How bad can randomly picking it be?

If two points have a small cosine distance, it is unlikely that we will split them into different bins!



If θ_{xy} is small (x,y close), unlikely to be placed into separate bins

Some Issues

1. How do we find a good line that divides the data in half?
2. Potential Errors: Points close together might be split up into separate bins
3. Large computation cost: Only dividing the points in half doesn't speed things up that much...





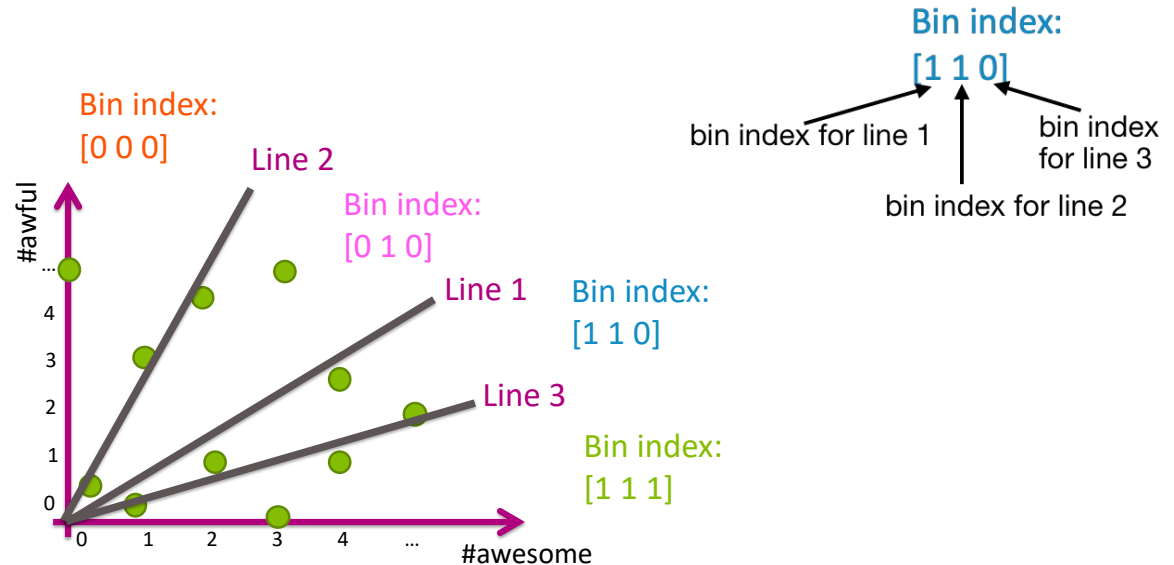
Brain Break



More Bins

Can reduce search cost by adding more lines, increasing the number of bins.

For example, if we use 3 lines, we can make more bins!



LSH with Many Bins

Create a table of all data points and calculate their bin index based on some chosen lines. Store points in hash table indexed by all bin indexes

2D Data	Sign (Score ₁)	Bin 1 index	Sign (Score ₂)	Bin 2 index	Sign (Score ₃)	Bin 3 index
$x_1 = [0, 5]$	-1	0	-1	0	-1	0
$x_2 = [1, 3]$	-1	0	-1	0	-1	0
$x_3 = [3, 0]$	1	1	1	1	1	1
...

Bin	[0 0 0] = 0	[0 0 1] = 1	[0 1 0] = 2	[0 1 1] = 3	[1 0 0] = 4	[1 0 1] = 5	[1 1 0] = 6	[1 1 1] = 7
Data indices:	{1,2}	--	{4,8,11}	--	--	--	{7,9,10}	{3,5,6}

When searching for a point x_q :

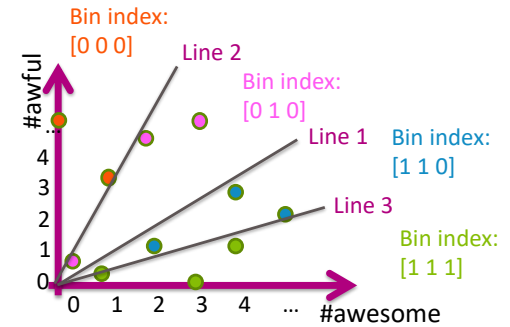
Find its bin index based on the lines

Only search over the points in that bin

LSH Example

Imagine my query point was (2, 2)

This has bin index [0 1 0]



Bin	[0 0 0] = 0	[0 0 1] = 1	[0 1 0] = 2	[0 1 1] = 3	[1 0 0] = 4	[1 0 1] = 5	[1 1 0] = 6	[1 1 1] = 7
Data indices:	{1,2}	--	{4,8,11}	--	--	--	{7,9,10}	{3,5,6}

By using multiple bins, we have reduced the search time!

However, it's more likely that we separate points from their true nearest neighbors since we do more splits 😞

Often with approximate methods, there is a tradeoff between speed and accuracy.

Improve Quality

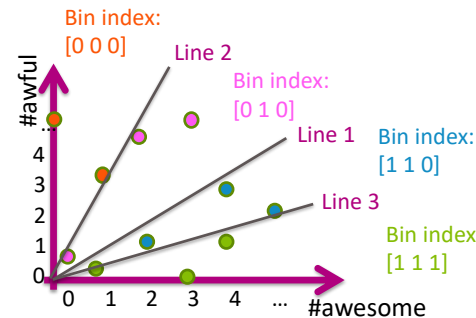
The nice thing about LSH is we can actually tune this tradeoff by looking at nearby bins. If we spend longer searching, we are likely to find a better answer.

What does "nearby" mean for bins?

Bin	$[0\ 0\ 0]$ = 0	$[0\ 0\ 1]$ = 1	$[0\ 1\ 0]$ = 2	$[0\ 1\ 1]$ = 3	$[1\ 0\ 0]$ = 4	$[1\ 0\ 1]$ = 5	$[1\ 1\ 0]$ = 6	$[1\ 1\ 1]$ = 7
Data indices:	{1,2}	--	{4,8,11}	--	--	--	{7,9,10}	{3,5,6}



Next closest bins
(flip 1 bit)



In practice, set some time "budget" and keep searching nearby bins until budget runs out

Locality Sensitive Hashing (LSH)

Pre-Processing Algorithm

Draw h lines randomly

For each data point, compute $Score(x_i)$ for each line

Translate the scores into binary indices

Use binary indices as a key to store the point in a hash table

Querying LSH

For query point x_q compute $Score(x_q)$ for each of the h lines

Translate scores into binary indices. Lookup all data points that have the same key.

Do exact nearest neighbor search just on this bin.

If there is more time in the computation budget, go look at nearby bins until this budget runs out.

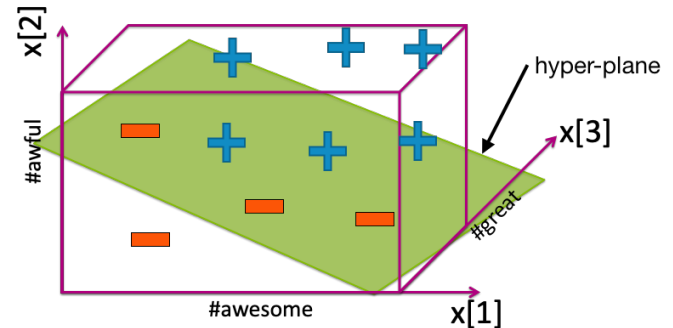
Higher Dimensions

Pick random hyper-plane to separate points for points in higher dimensions.

Unclear how to pick h for LSH and you can't do cross-validation here (why?)

Generally people use $h \approx \log(d)$

$$\text{Score}(x) = w_1 \# \text{awesome} + w_2 \# \text{awful} + w_3 \# \text{great}$$



Recap

Theme: Use local methods for classification and regression and speed up nearest neighbor search with approximation methods.

Ideas:

1-NN Regression and Classification

k-NN Regression and Classification

Weighted k-NN vs Kernel Regression

Locality Sensitive Hashing

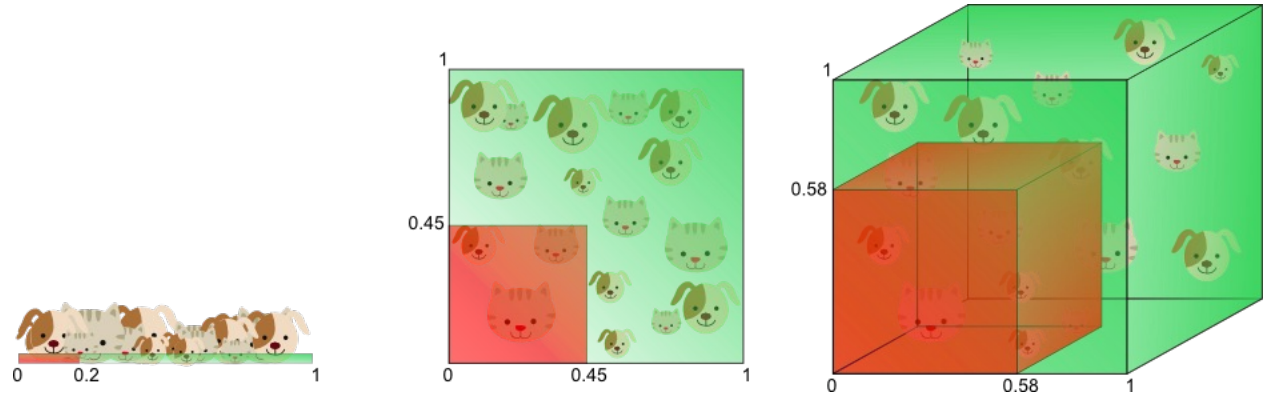


Curse of Dimensionality

High Dimensions

Methods like k-NN and k-means that rely on computing distances start to struggle in high dimensions.

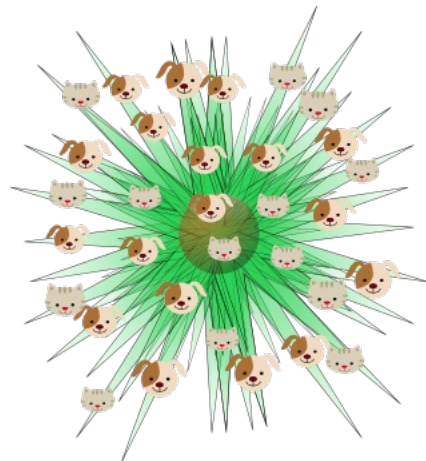
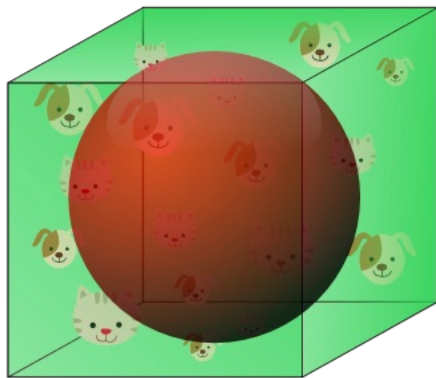
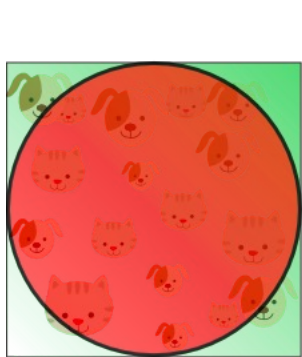
As the number of dimensions grow, the data gets sparser!



Need more data to make sure you cover all the space in high dim.

Even Weirder

It's believable with more dimensions the data becomes more sparse, but what's even weirder is the sparsity is not uniform!



As D increases, the “mass” of the space goes towards the corners.

Most of the points aren't in the center.

Your nearest neighbors start looking like your farthest neighbors!



Practicalities

Have you pay attention to the number of dimensions

Very tricky if $n < D$

Can run into some strange results if D is very large

Later, we will talk about ways of trying to do dimensionality reduction in order to reduce the number of dimensions here.



HW5 – ML Practice with Kaggle

Kaggle is a website that hosts ML competitions. Very popular among people trying to learn more about ML! Let's you practice a lot of real-world ML skills on challenging problems.

About halfway through the course, want to give you a chance to apply what you've learned in the real-world! HW5 will be hosted as an internal Kaggle competition.

TAs in section tomorrow will walk through how to do submissions, but all instructions are on the Kaggle website (linked from HW spec).

NOT graded as a competition. The leaderboard is just for fun, and we will provide a small amount of EC to the top 3 teams.



Submission and Grading

You will need to submit three things:

On Kaggle, submit your predictions for the test set

On Gradescope, submit your Jupyter Notebook that trains/evaluates your models and answers some questions we asked you to answer.

On Gradescope, submit concept questions.

Part of your grade will be from the accuracy on the private test set. Will count 95% accuracy on the private test set as full credit for the performance part of the assignment.

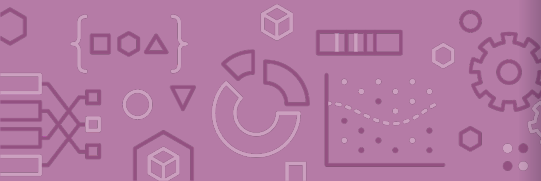


Groups

You will be able to work on the Kaggle portion with a team of up to 3 people. You can work alone if you choose, but we recommend working with a team since that's a great learning experience!

If you work as a team, you can make a team submission on Kaggle and submit your code/report answers on Gradescope together.

Concept portion should be done individually.



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.

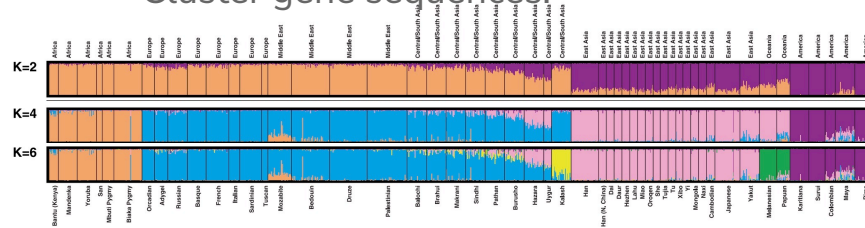
Coupled indoor navigation for people who are blind

[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.

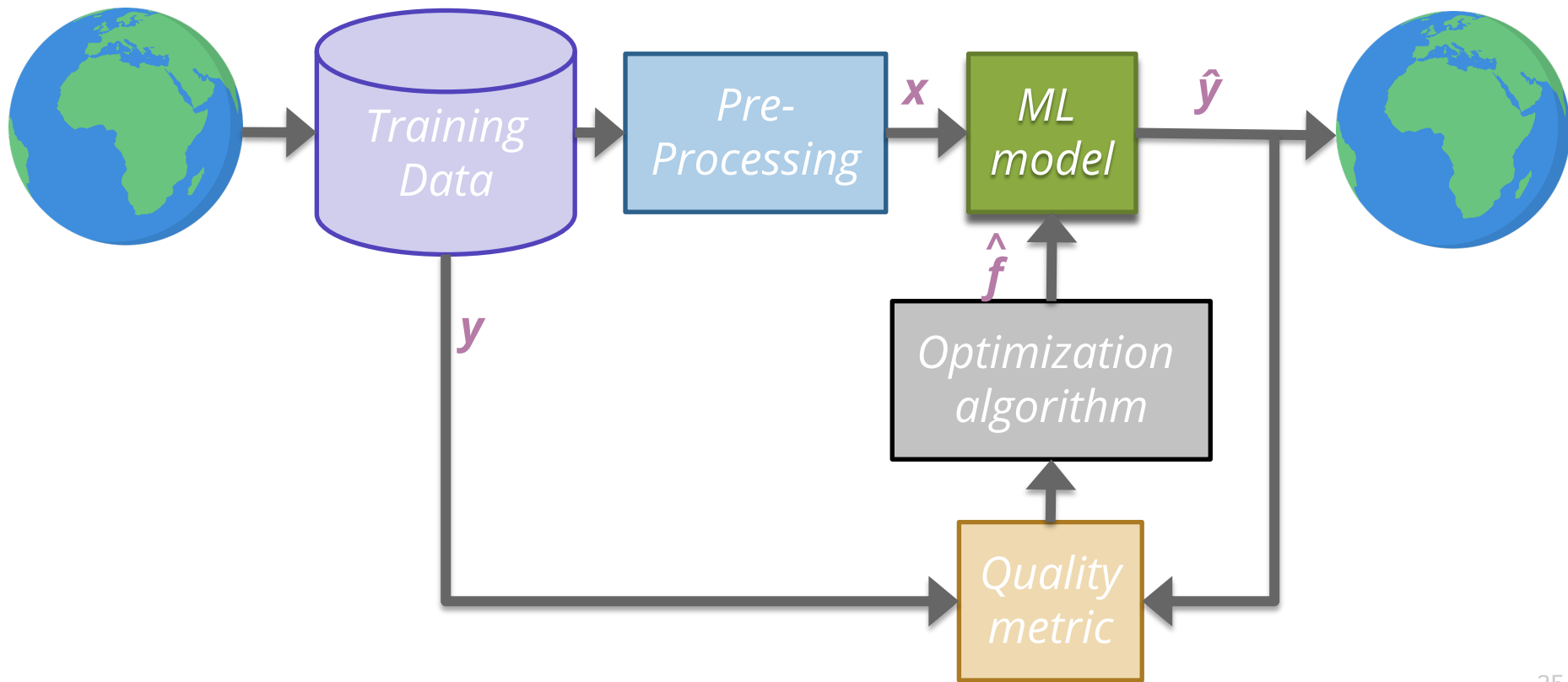
- unsupervised
- unsupervised learning
- unsupervised recommender system
- unsupervised learning recommendation system
- unsupervised learning example
- unsupervised machine learning
- unsupervised
- unsupervised learning algorithms
- Unsupervised Sitcom
- unsupervised clustering
- unsupervised vs supervised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

Products related to this item

Sponsored @

Page 1 of 55

24



Clustering



SPORTS

WORLD NEWS

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

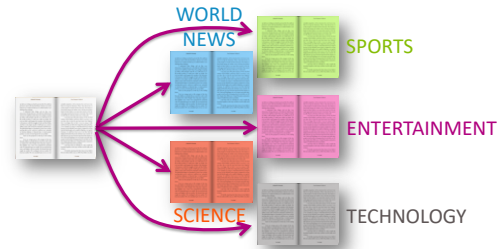
Our case study is **document retrieval**. Given that someone read a particular article, what similar articles would you recommend (without personalization)?

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.

Input: x_1, x_2, \dots, x_n



Output: z_1, z_2, \dots, z_n



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.

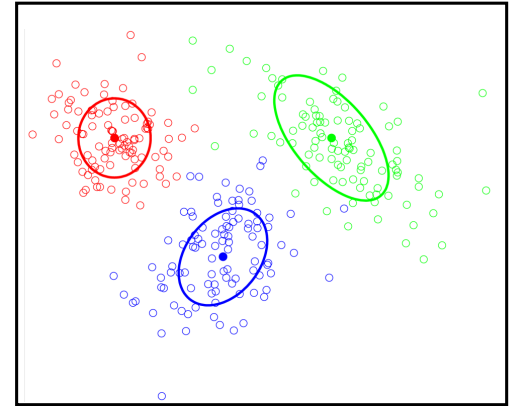
x_i gets assigned $z_i \in [1, 2, \dots, k]$

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.

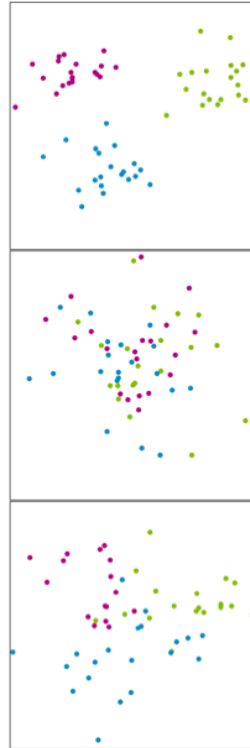
Close distance reflects strong similarity between datapoints.



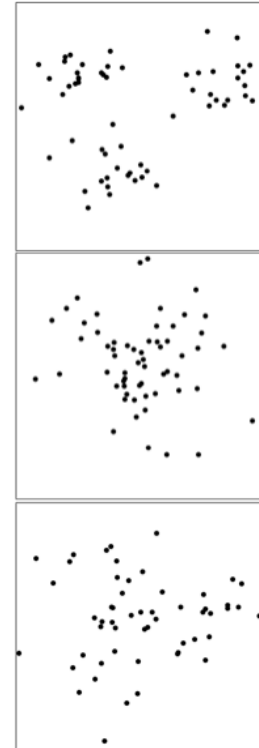
When Might This Work?

Clustering is easy when distance captures the clusters.

Ground Truth (not visible)



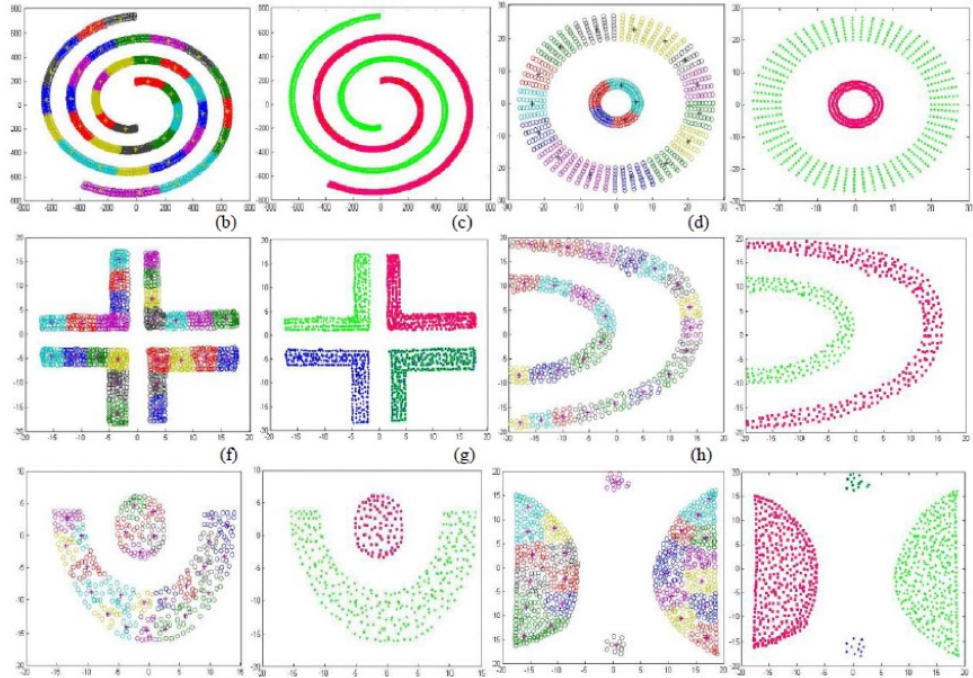
Given Data



Not Always Easy

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

Distance does not determine clusters



slido

Group 

2 min

Think of 1-2 problems that you might want to use clustering for.

For each problem, describe:

- Why unsupervised learning is the right approach.
- What the input features are for the clustering algorithm.
- What clusters you hypothesize would emerge.



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

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

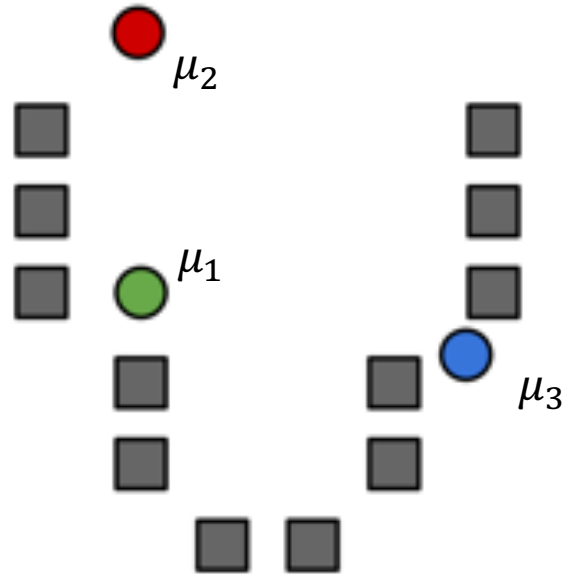


Step 0

Start by choosing the initial cluster centroids

A common default choice is to choose centroids μ_1, \dots, μ_k randomly

Will see later that there are smarter ways of initializing

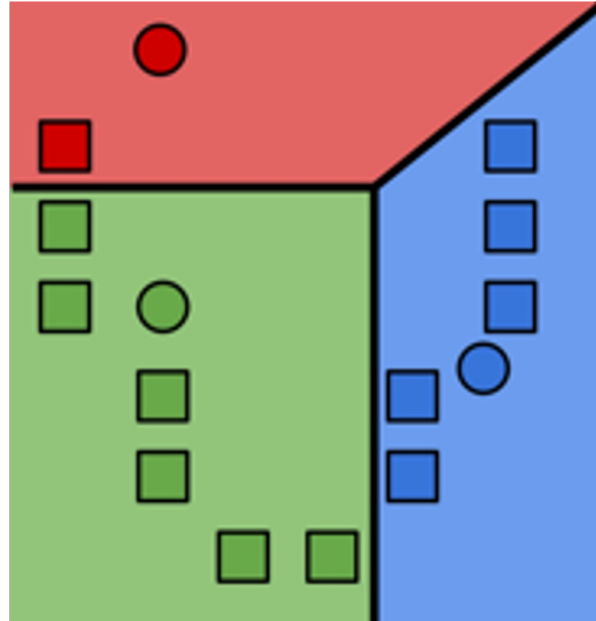


Step 1

Assign each example to its closest cluster centroid

For $i = 1$ to n

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

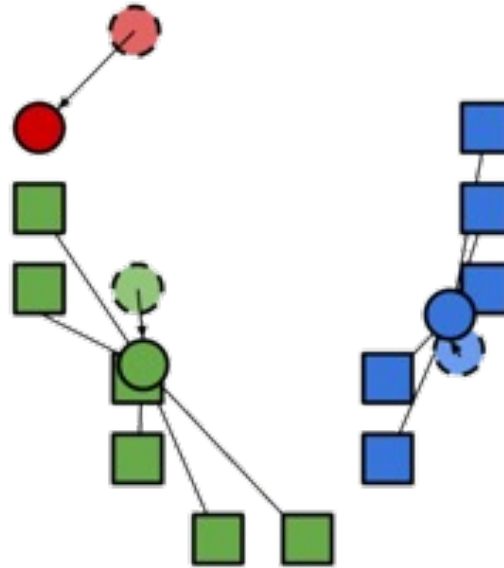


Step 2

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

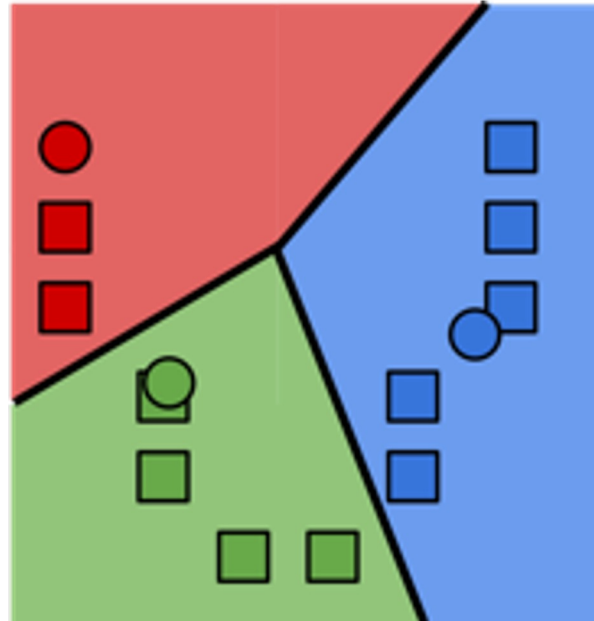
$$\mu_j = \frac{\sum_{i=1}^n \mathbf{1}\{z_i = j\} x_i}{\sum_{i=1}^n \mathbf{1}\{z_i = j\}}$$

Computes center of mass for cluster!



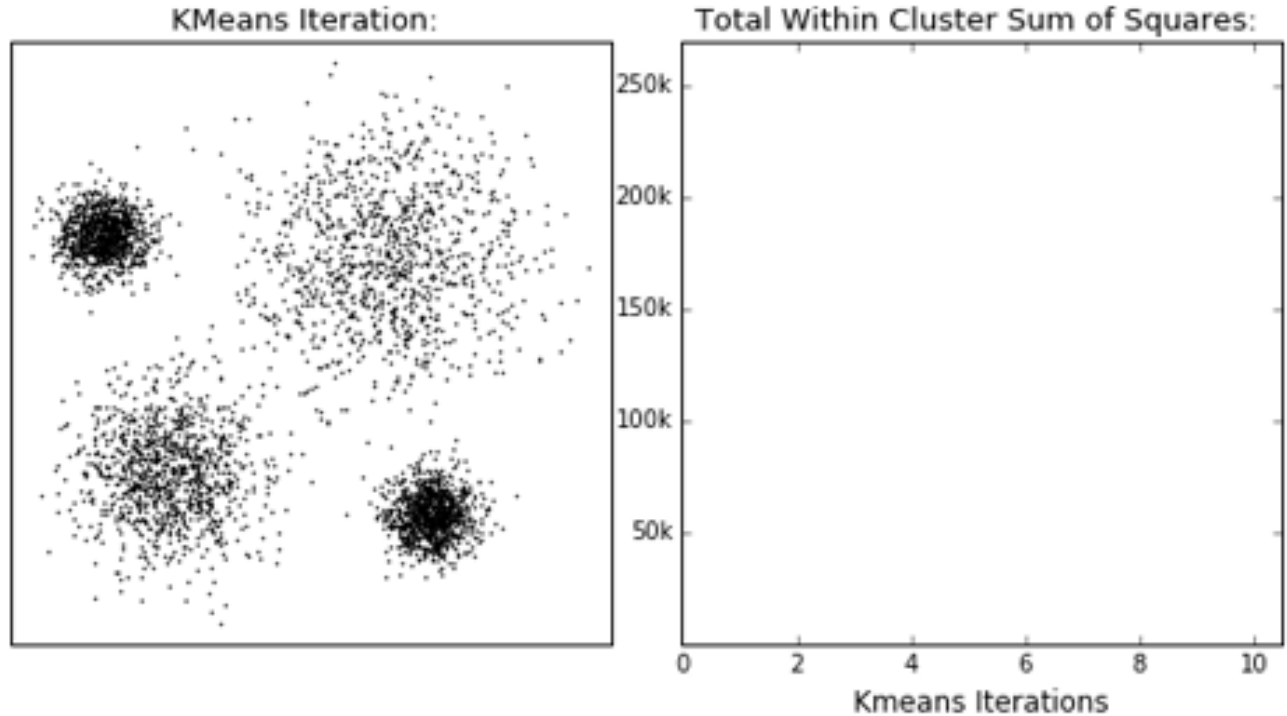
Repeat
until
convergence

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



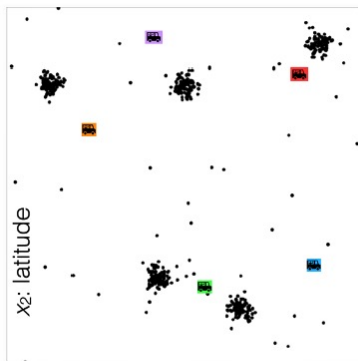
slido

Think 

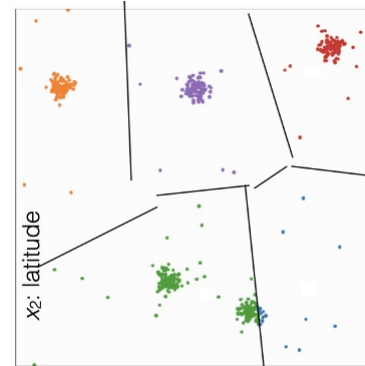
1 min

slido #cs416

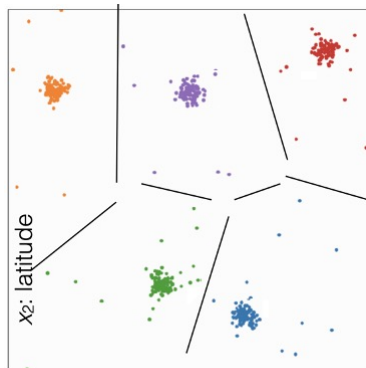
What cluster assignment would result from these centroids?



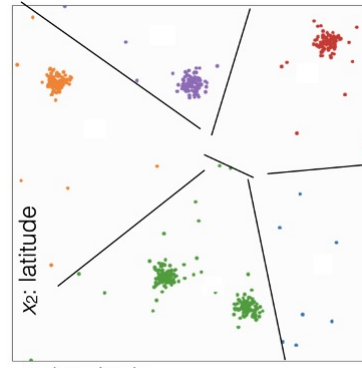
x_1 : longitude
Centroids



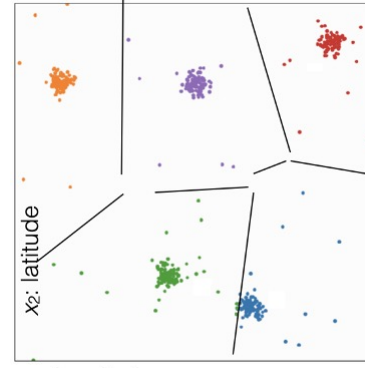
x_1 : longitude
(D)



x_1 : longitude
(A)



x_1 : longitude
(B)



x_1 : longitude
(C)

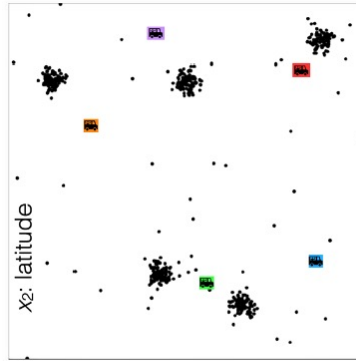
slido

Group 

1 min

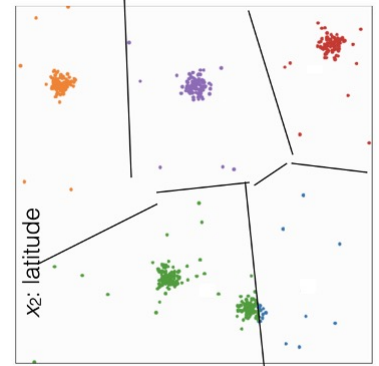
slido #cs416

What cluster assignment would result from these centroids?



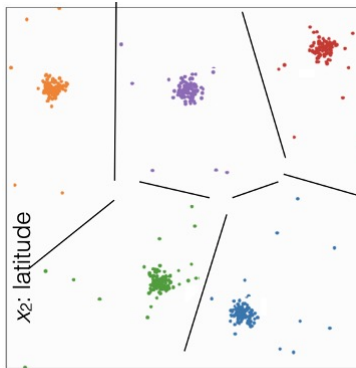
x_1 : longitude

Centroids



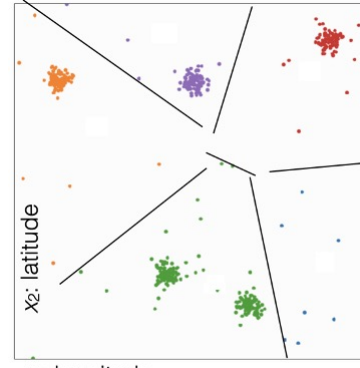
x_1 : longitude

(D)



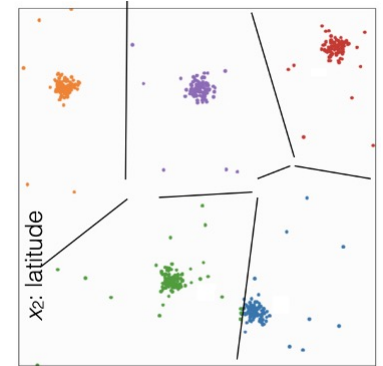
x_1 : longitude

(A)



x_1 : longitude

(B)



x_1 : longitude

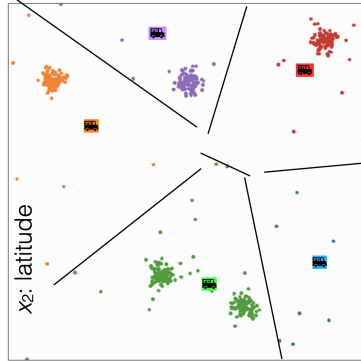
(C)

slido

Think 

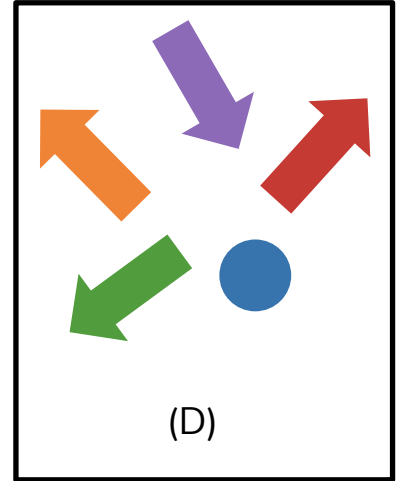
1 min

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

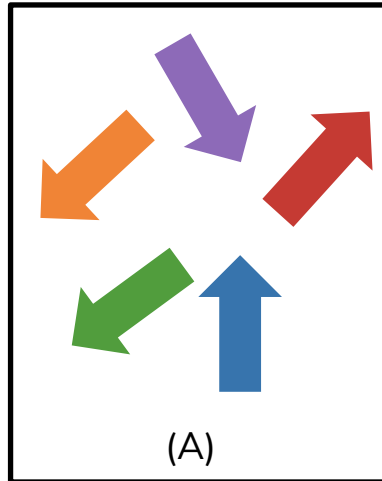


x₁: longitude

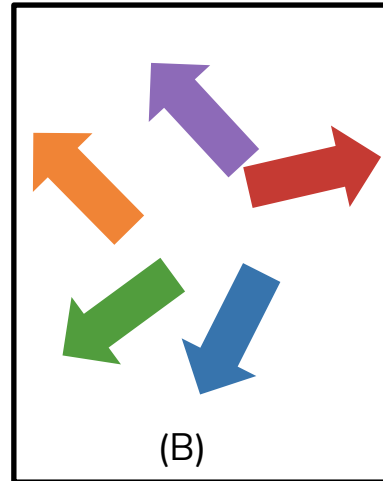
Cluster Assignments



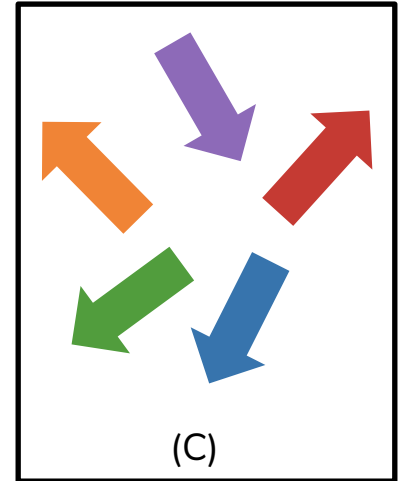
(D)



(A)



(B)



(C)

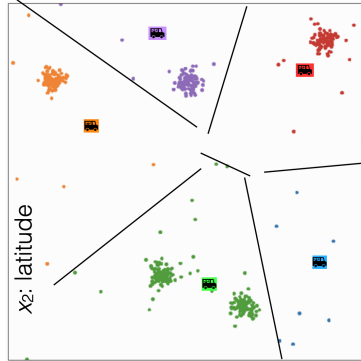
slido #cs416

slido

Group 

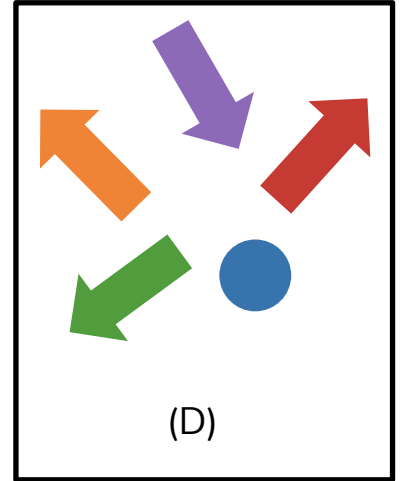
2 min

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

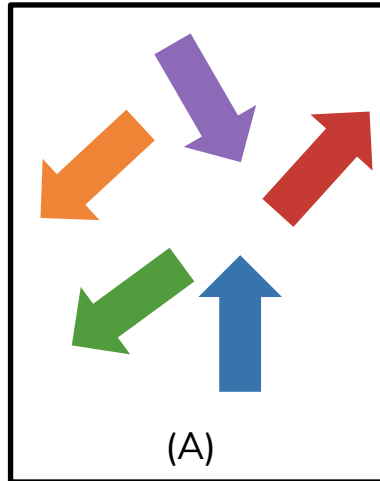


x₁: longitude

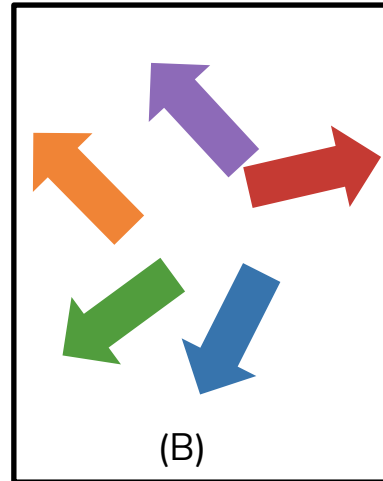
Cluster Assignments



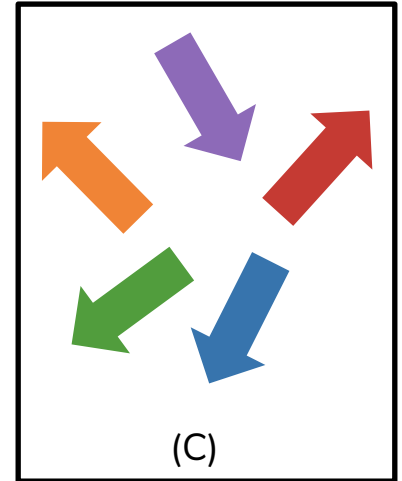
(D)



(A)

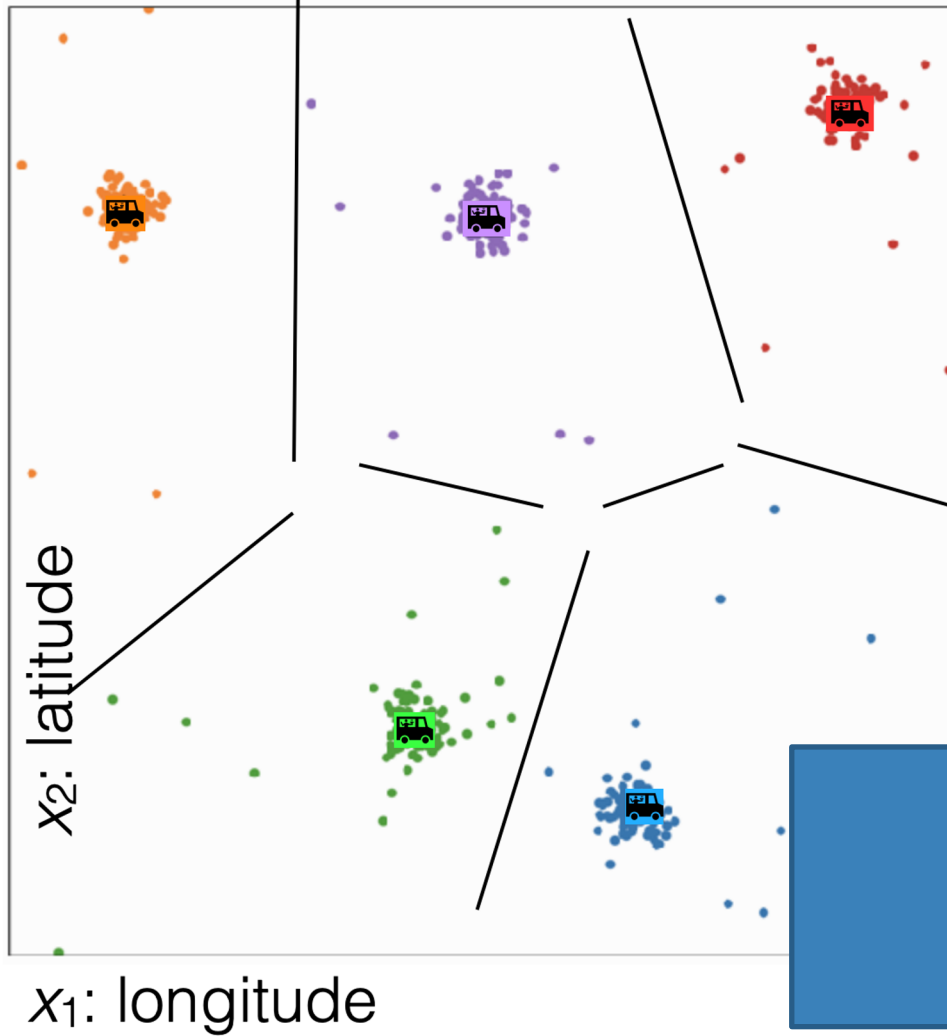


(B)



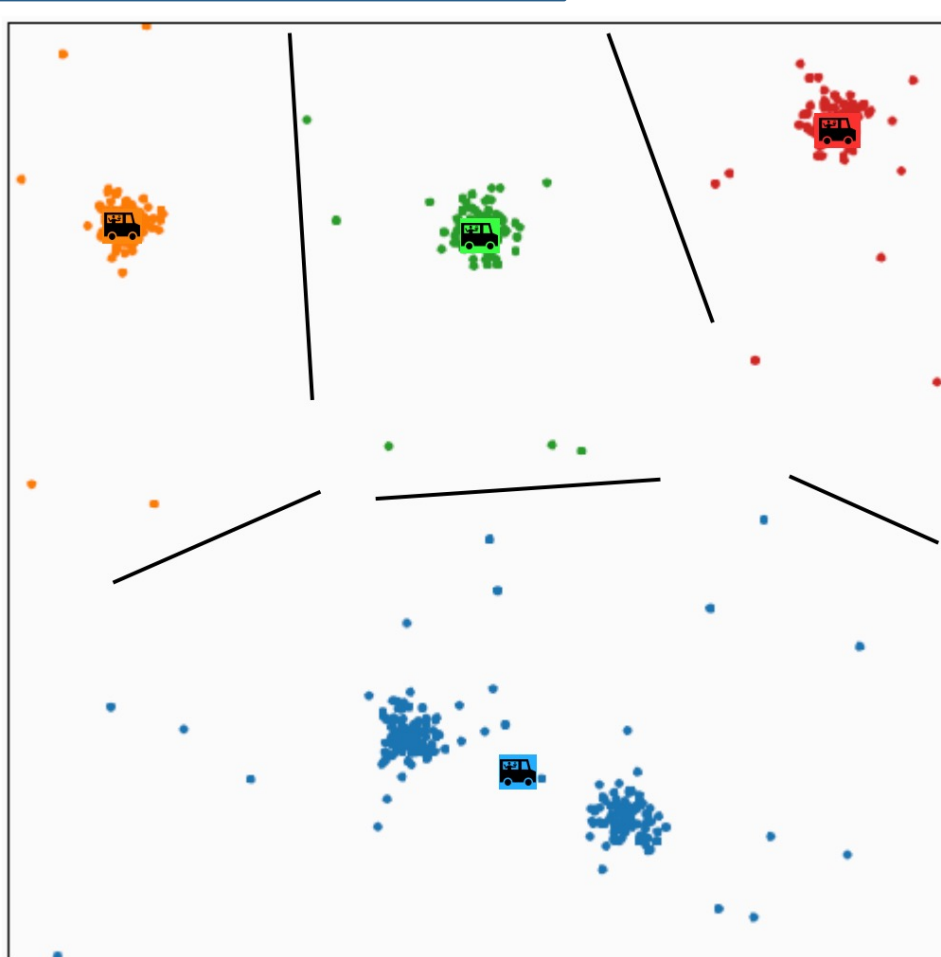
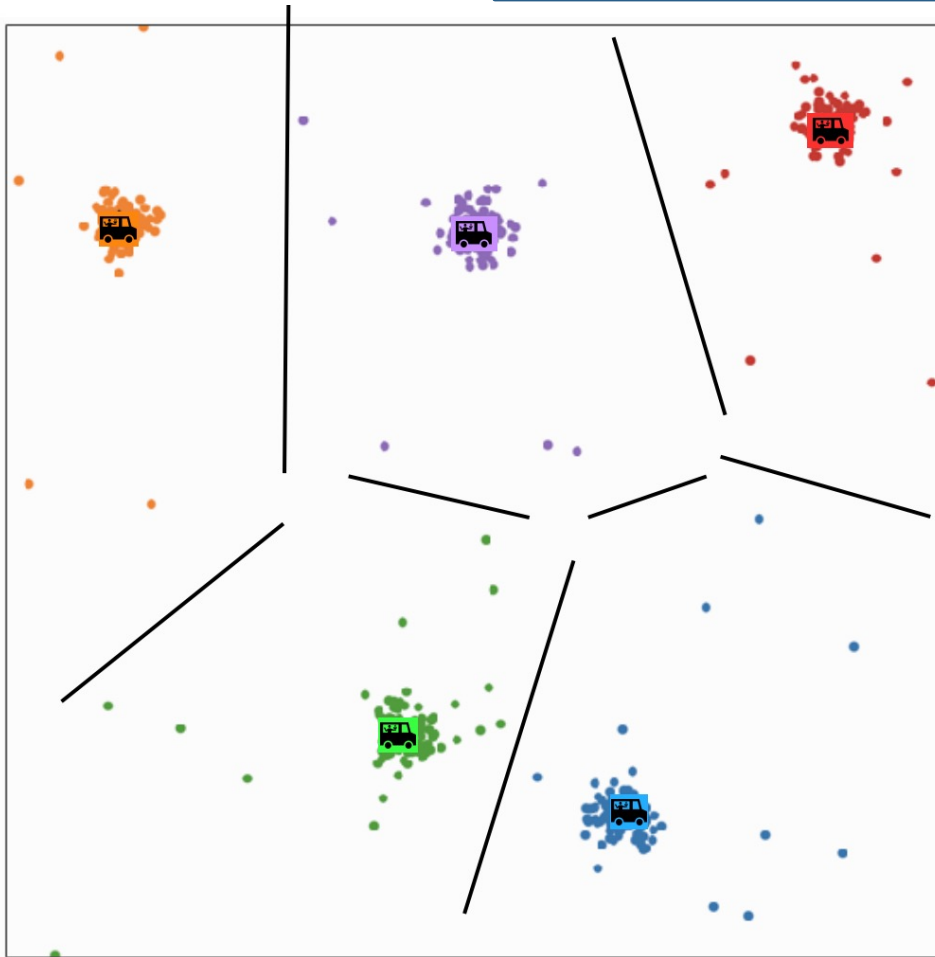
(C)

slido #cs416



No more changes

Different k gives different results



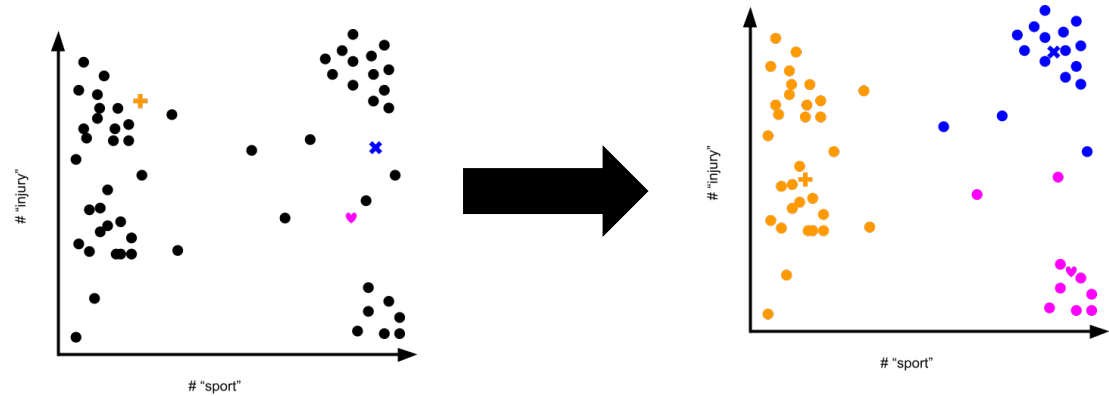
slido

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

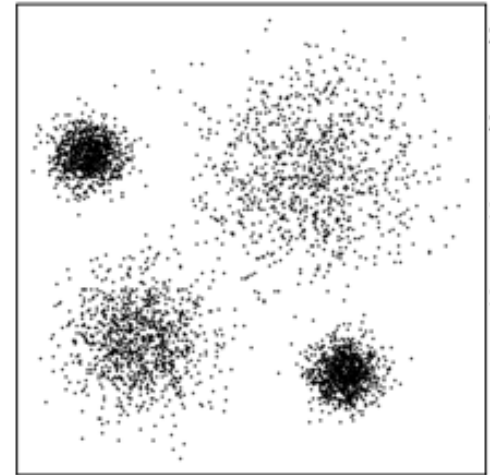
What convergence guarantees do you think we will have with k-means?

Converges to the global optimum

Converges to a local optima

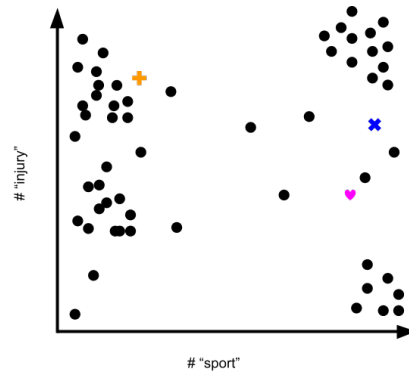
None

KMeans Iteration:

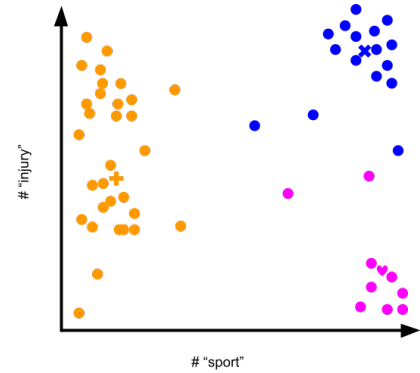


Effects of Initialization

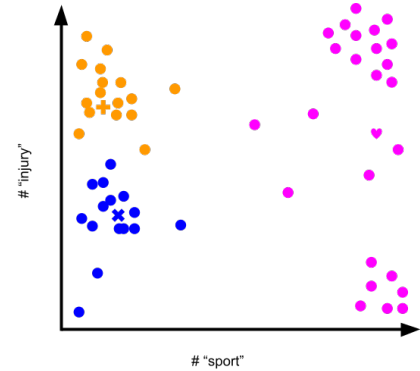
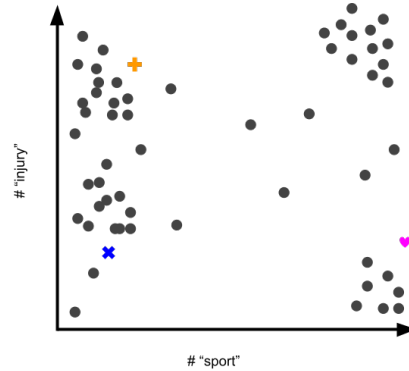
Results **heavily depend** on initial centroids



Initialization



Final Clusters

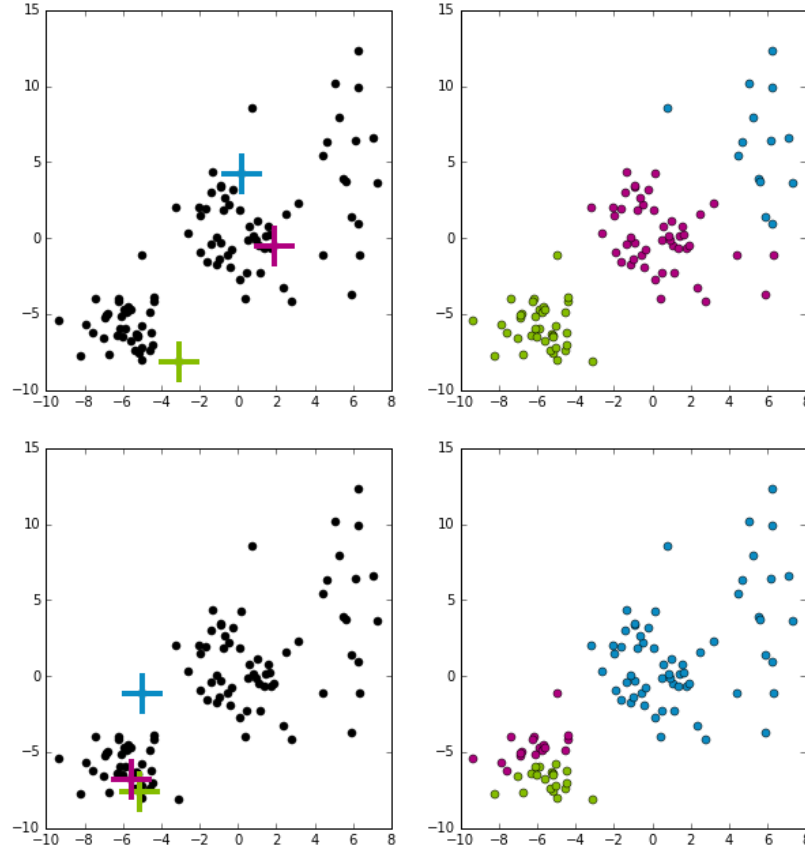


Effect of initialization

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

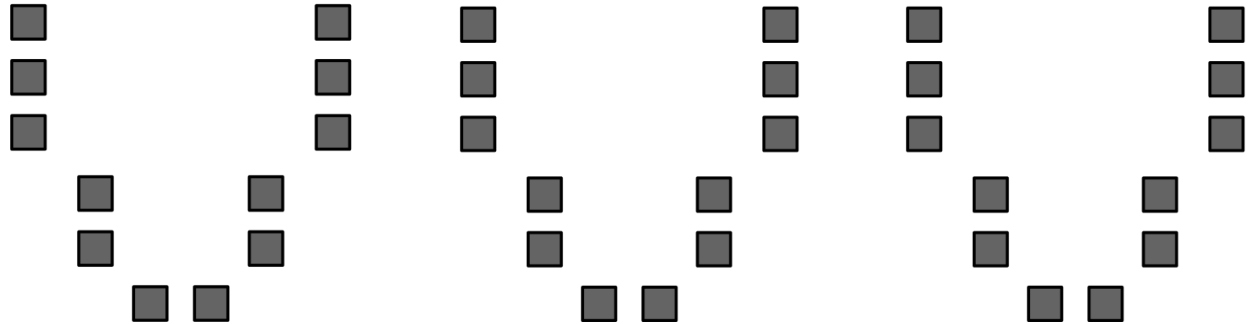
1. Choose first cluster μ_1 from the data uniformly at random
2. For each datapoint x_i , compute the distance between x_i and the closest centroid from the current set of centroids (starting with just μ_1). Denote that distance $d(x_i)$.
3. Choose a new centroid from the remaining data points, where 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

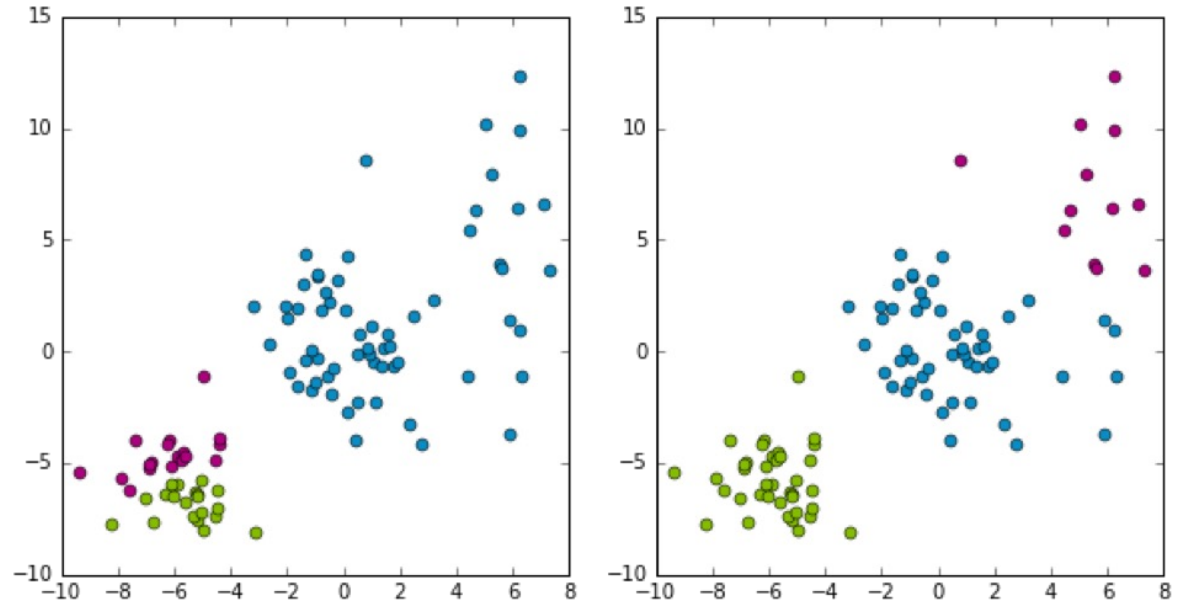
Computationally more expensive at beginning when compared to simple random initialization



Assessing Performance

Which Cluster?

Which clustering would I prefer?

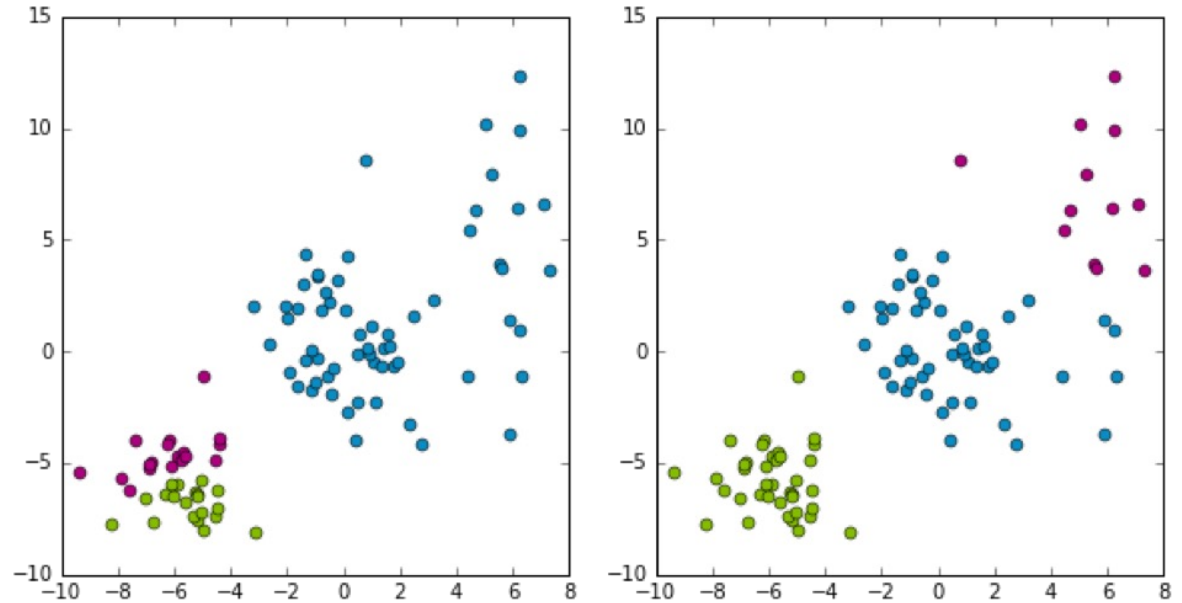


Don't know, there is no “right answer” in clustering 🙋 .

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

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

Coordinate Descent

k-means is trying to minimize the heterogeneity objective

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

Step 0: Initialize cluster centers

Repeat until convergence:

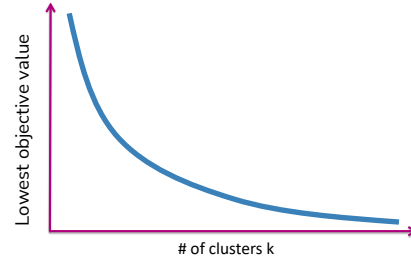
Step 1: Assign each example to its closest cluster centroid

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

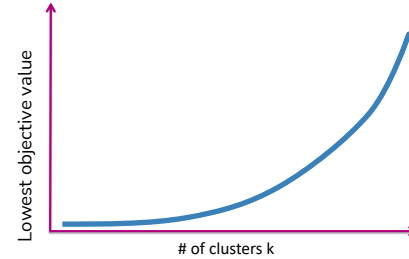
Coordinate Descent alternates how it updates parameters to find minima. On each of iteration of Step 1 and Step 2, heterogeneity decreases or stays the same.

=> Will converge in finite time

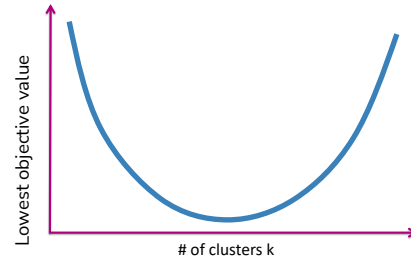
Consider training k-means to convergence for different values of k . Which of the following graphs shows how the heterogeneity objective will change based on the value of k ?



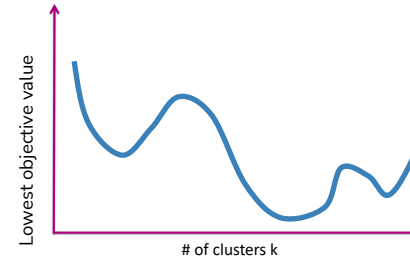
A



B

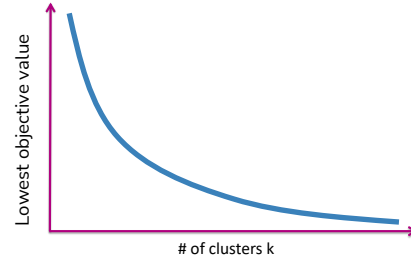


C

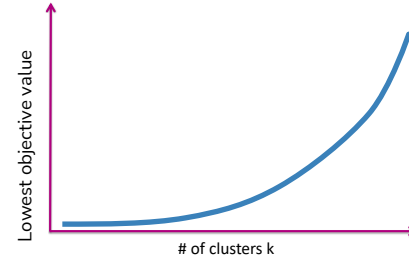


D

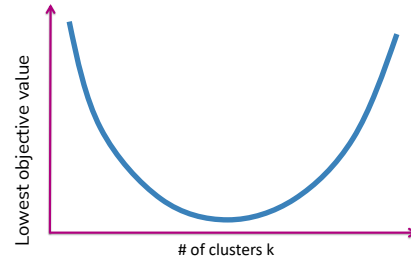
Consider training k-means to convergence for different values of k . Which of the following graphs shows how the heterogeneity objective will change based on the value of k ?



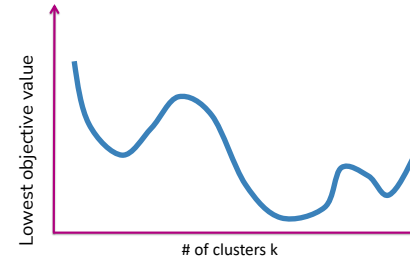
A



B



C

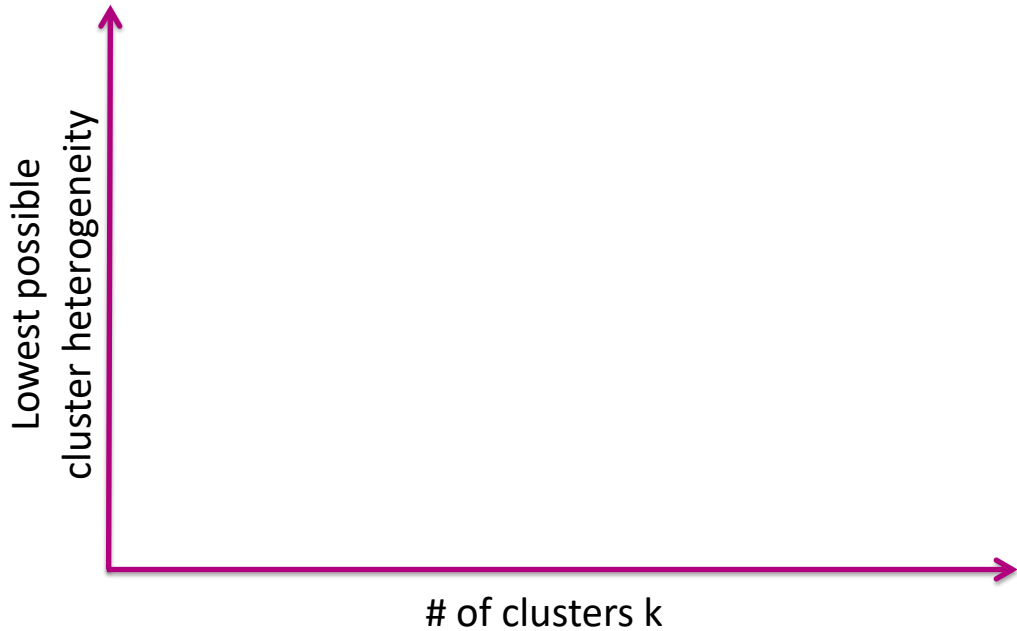


D

How to Choose k?

No right answer! Depends on your application.

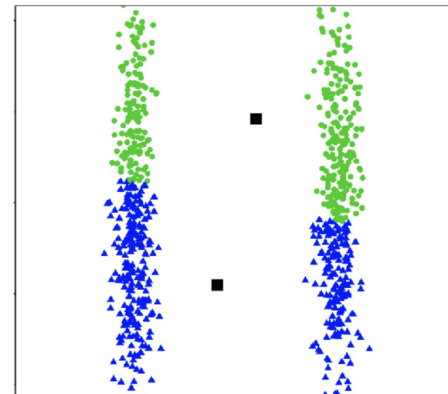
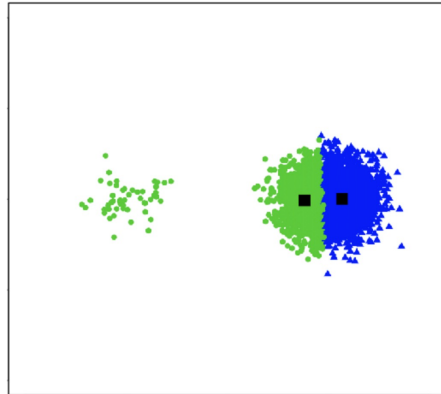
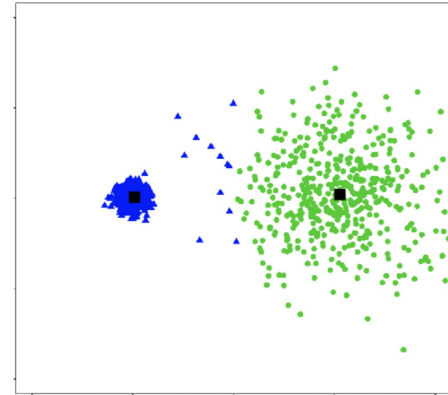
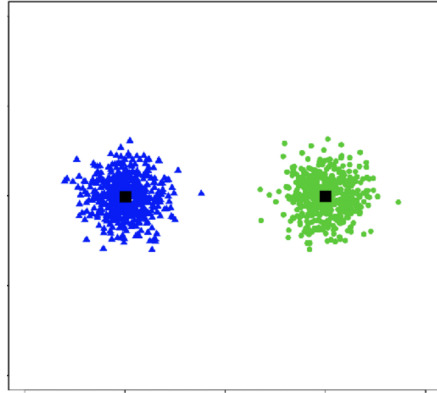
General, look for the “elbow” in the graph



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.

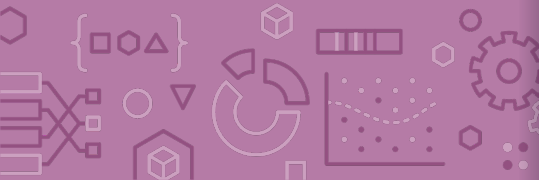
In clustering, the “labels” don't have meaning. To give meaning to the labels, human inputs is required

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++

