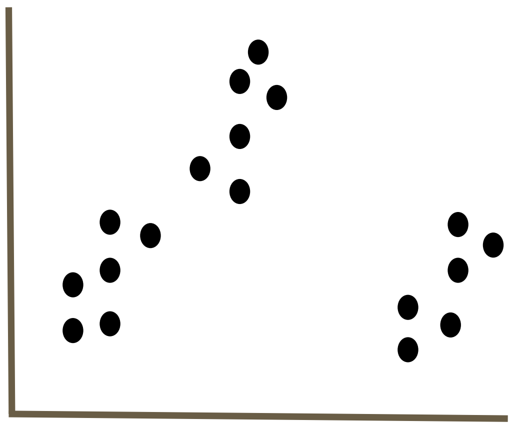
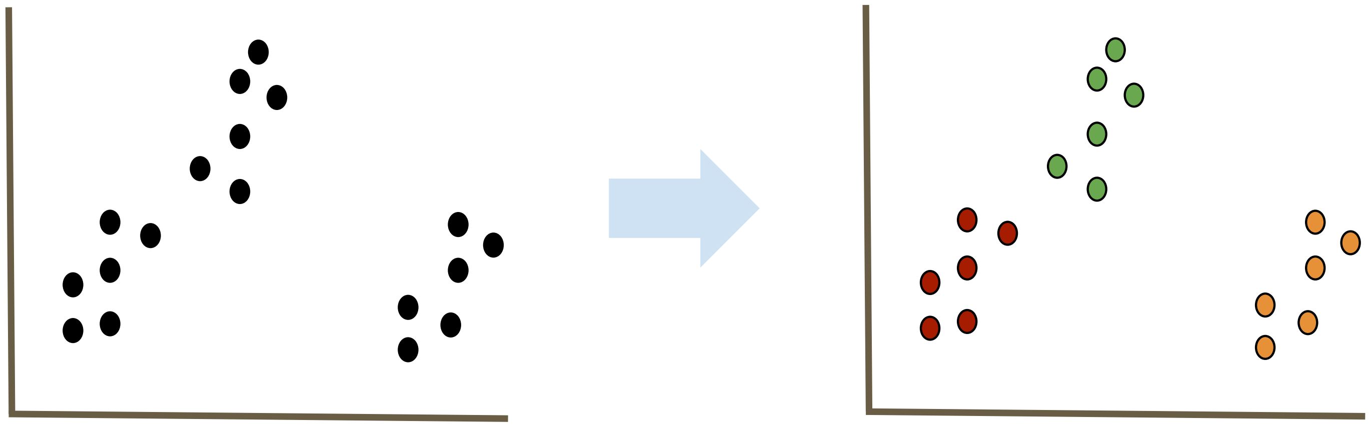
# Clustering - Kmeans

Boston University CS 506 - Lance Galletti

# What is a Clustering



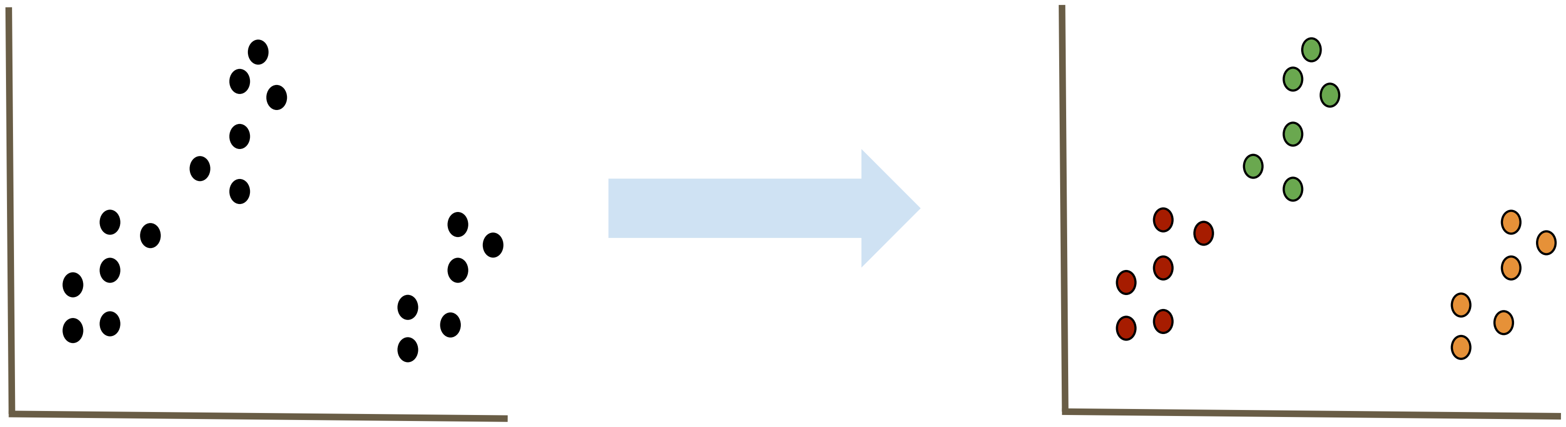
# What is a Clustering



# What is a Clustering

A clustering is a grouping / assignment of objects (data points) such that objects in the same group / cluster are:

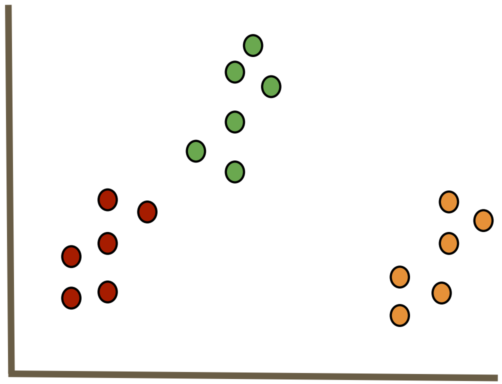
- similar to one another
- dissimilar to objects in other groups

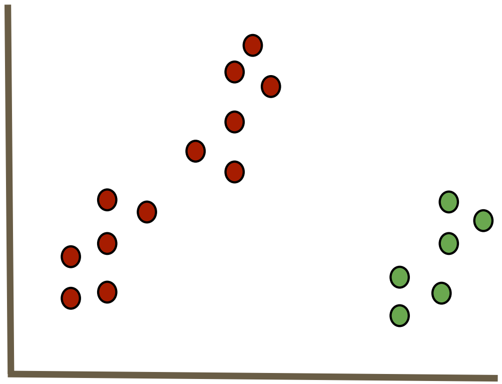


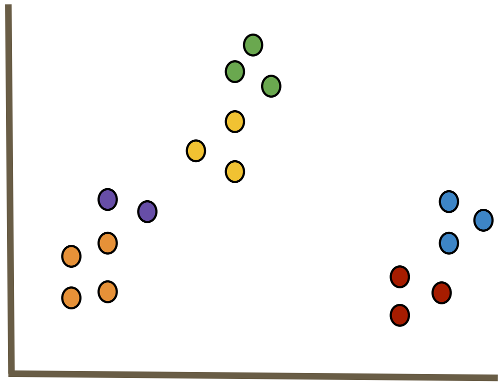
# Applications

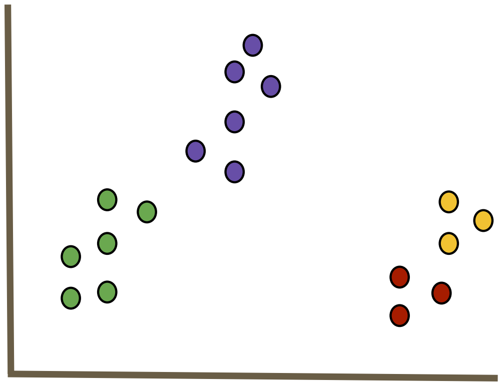
- Outlier detection / anomaly detection
  - Data Cleaning / Processing
  - Credit card fraud, spam filter etc.
- Feature Extraction
- Filling Gaps in your data
  - Using the same marketing strategy for similar people
  - Infer probable values for gaps in the data (similar users could have similar hobbies, likes / dislikes etc.)

# Clusters can be Ambiguous









# Types of Clusterings

**Partitional**
Each object belongs to exactly one cluster

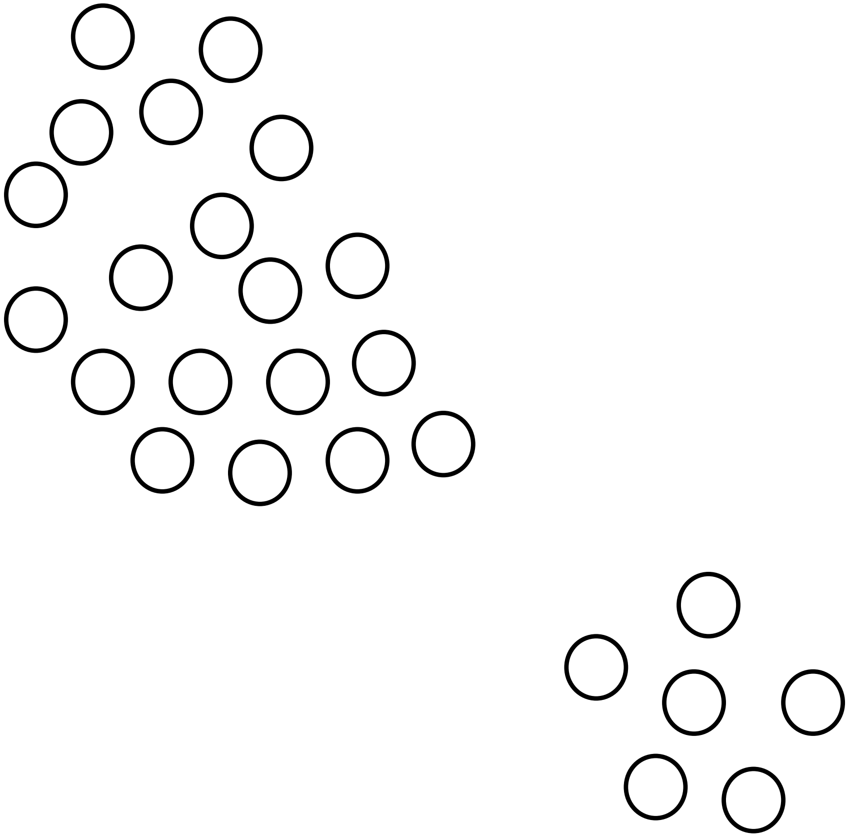
**Hierarchical**
A set of nested clusters organized in a tree

**Density-Based**
Defined based on the local density of points

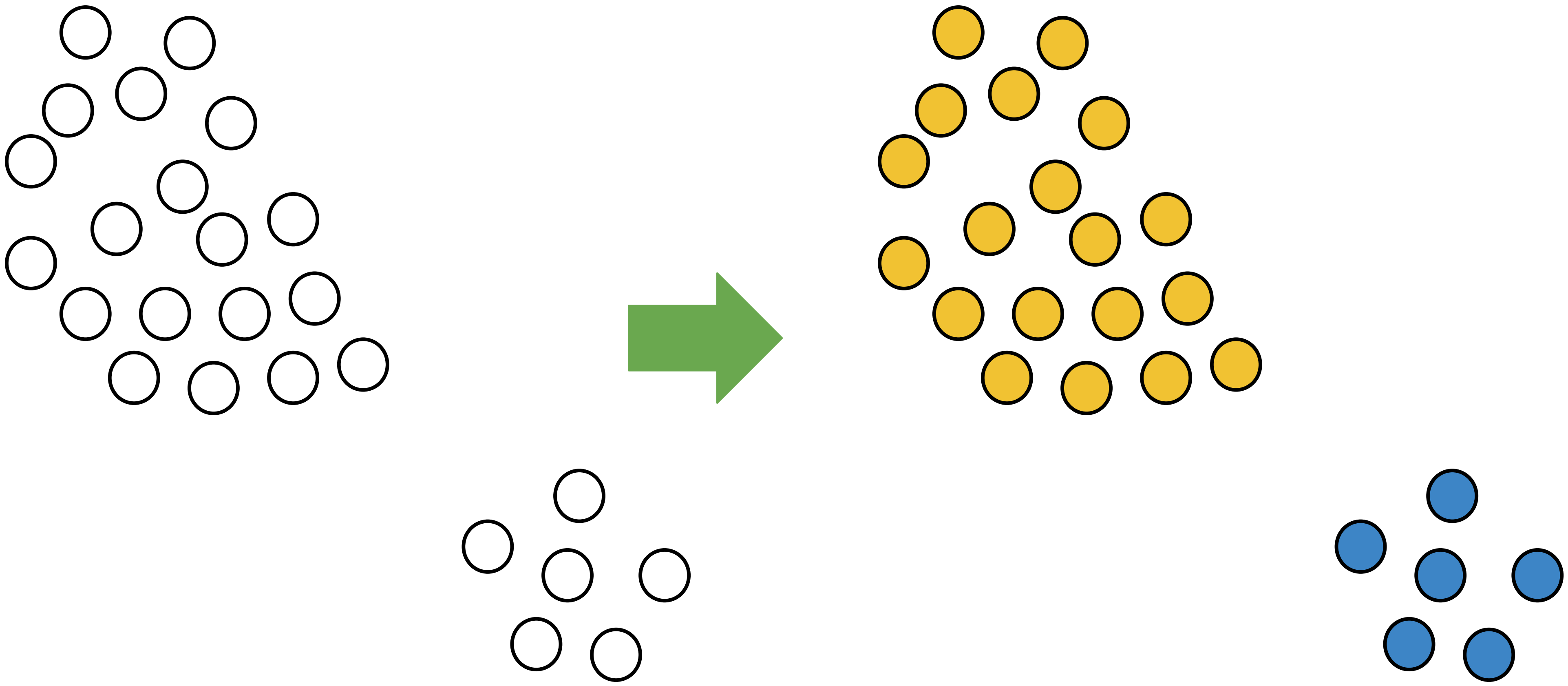
**Soft Clustering**
Each point is assigned to every cluster with a certain probability

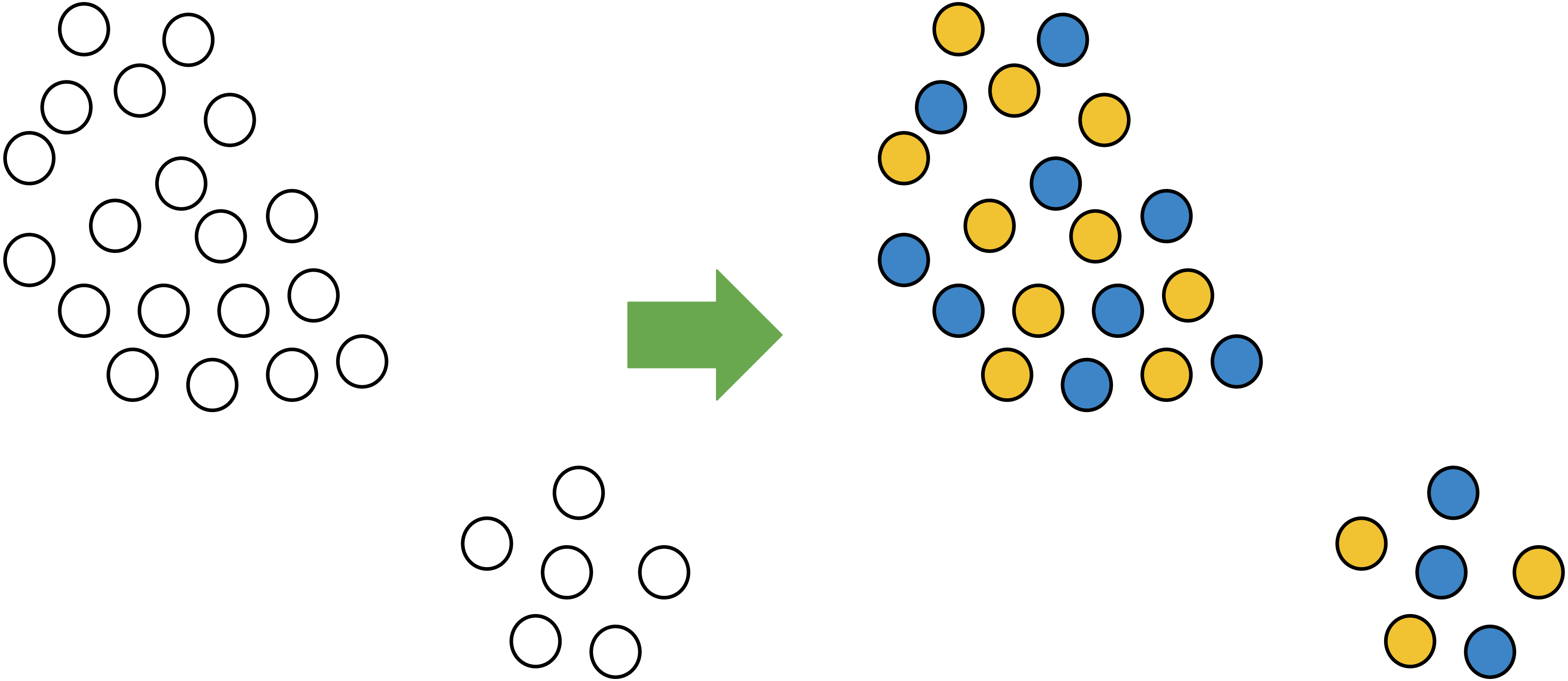
# Partitional Clustering

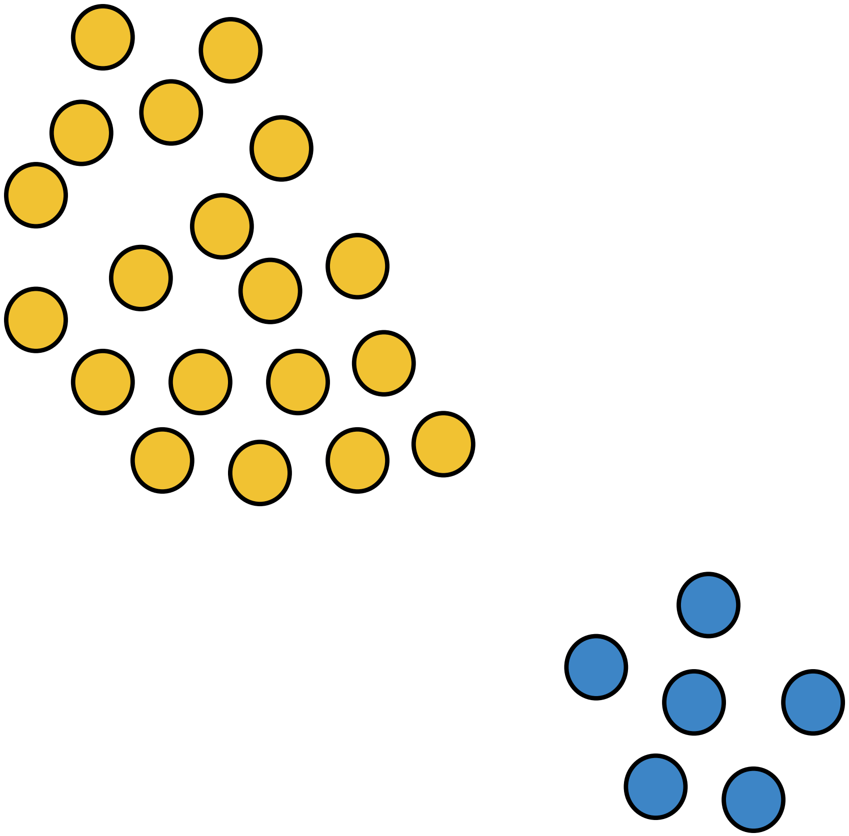
# Partitional Clustering

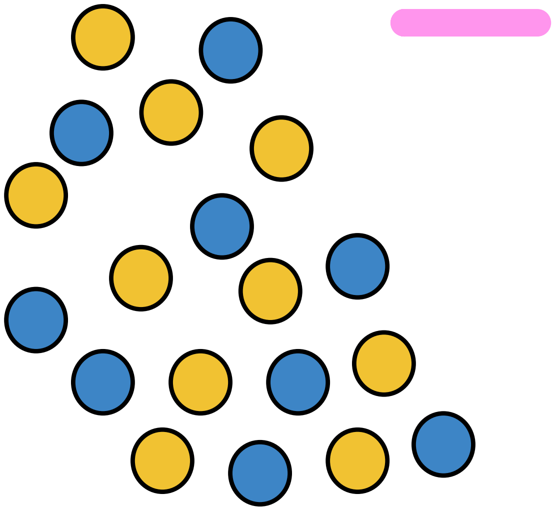


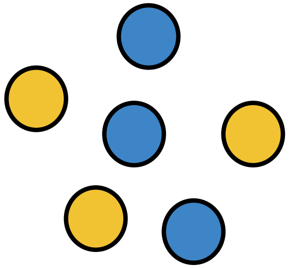
**Goal**: partition dataset into k partitions



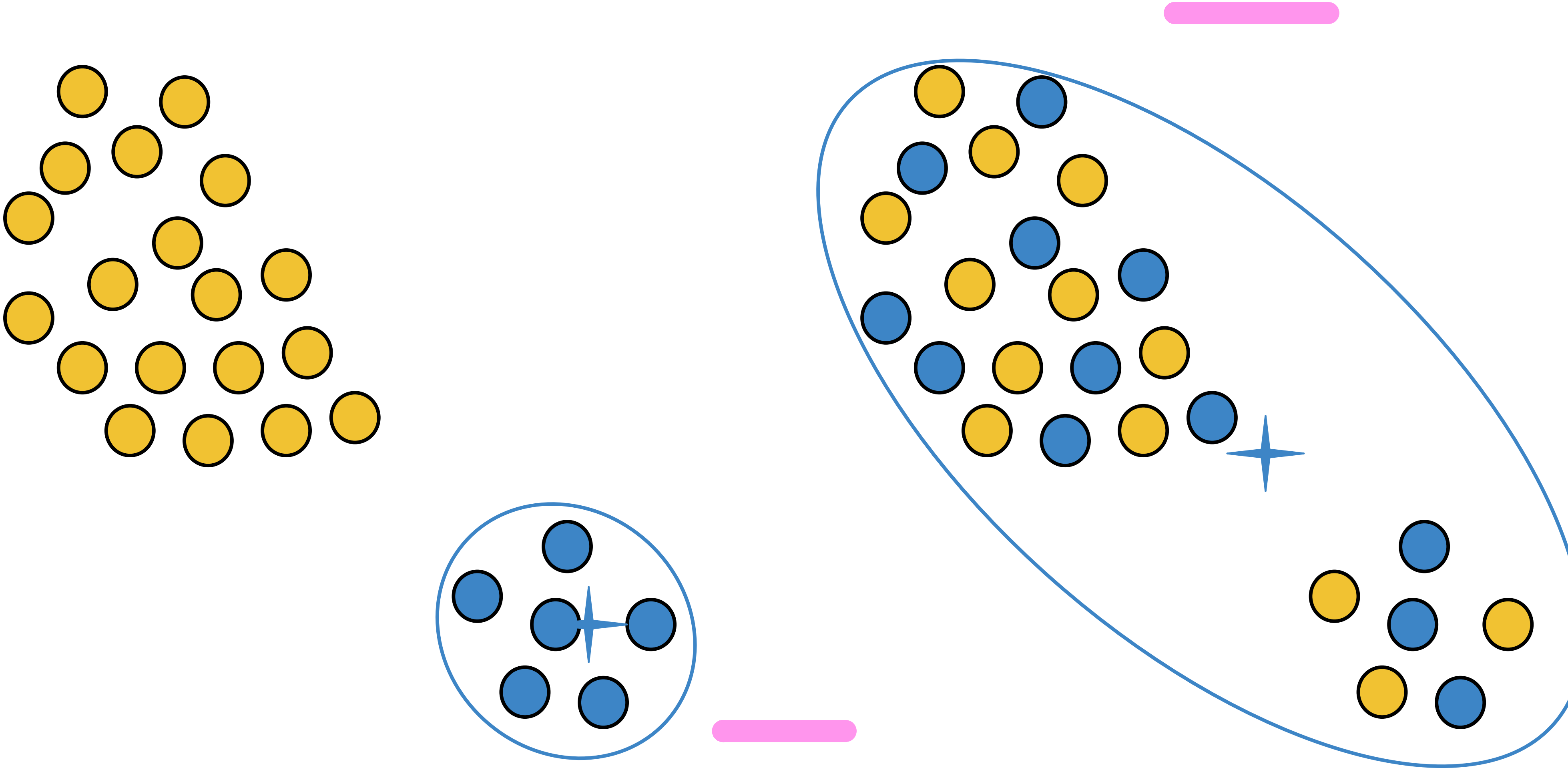


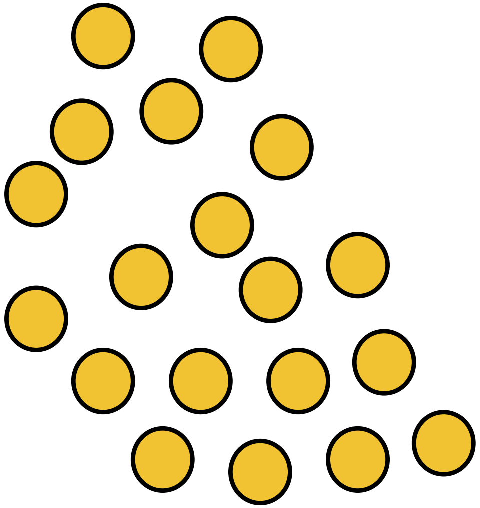




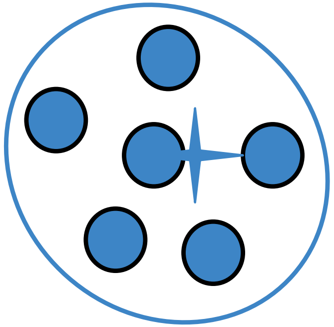


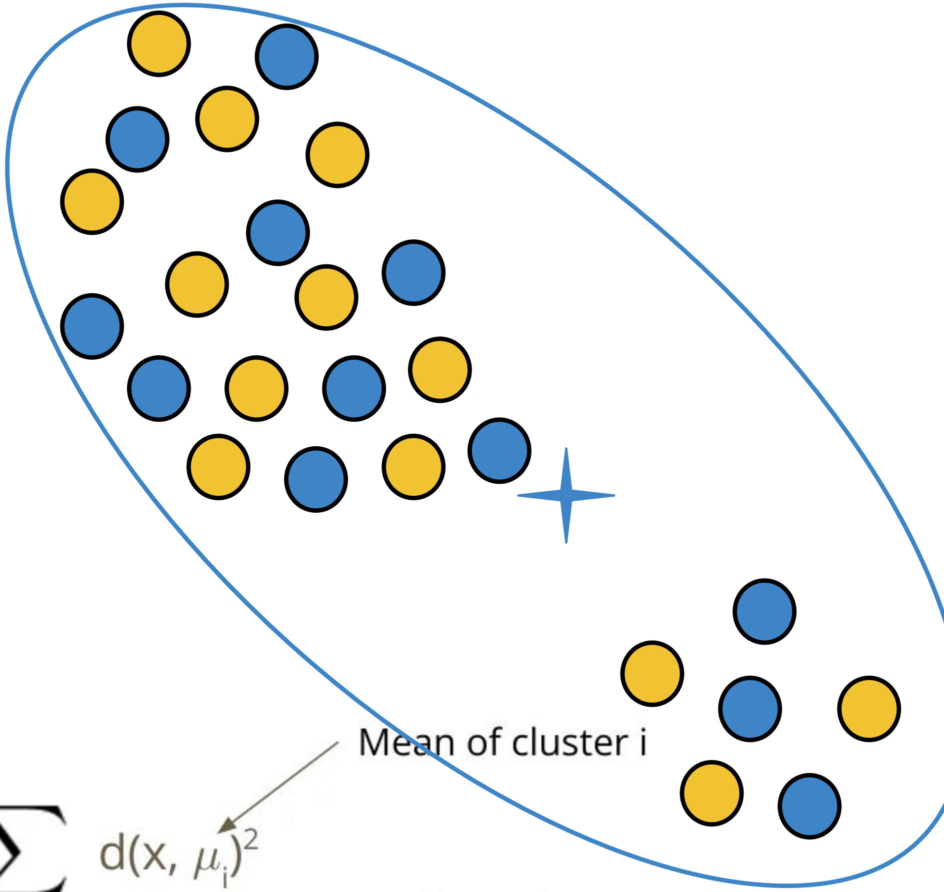
variance for the blue cluster on the left is smaller than the variance on the right





we want to make clusters with the smallest variance (smallest centering around the mean)





Mean of cluster i

$$\frac{1}{|C_i|}\sum_{x\in C_i} d(x, \mu_i)^2$$

Cluster i

# Cost Function

$$\sum_{i}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

the sum over k clusters
- for every clusters
  - take the sum of the squared distances among the mean (minimize the squared distance from the mean)

for every point in cluster i

- Way to evaluate and compare solutions
- Hope: can find some algorithm that find solutions that make the cost small

# K-means

Given $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ our dataset, **d** the euclidean distance, and **k**

Find **k** centers $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$ that minimize the **cost function**:

$$\sum_{i}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

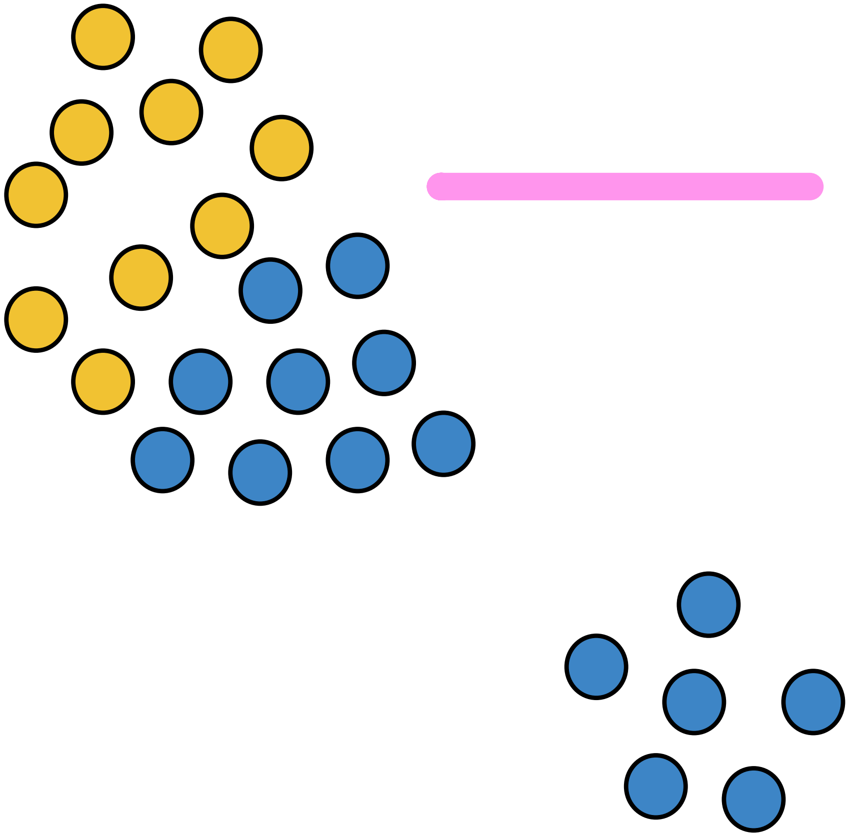
k= 1 ---> you have one cluster

When **k=1** and **k=n** this is easy. Why?

k= 2 ---> every point is its own cluster

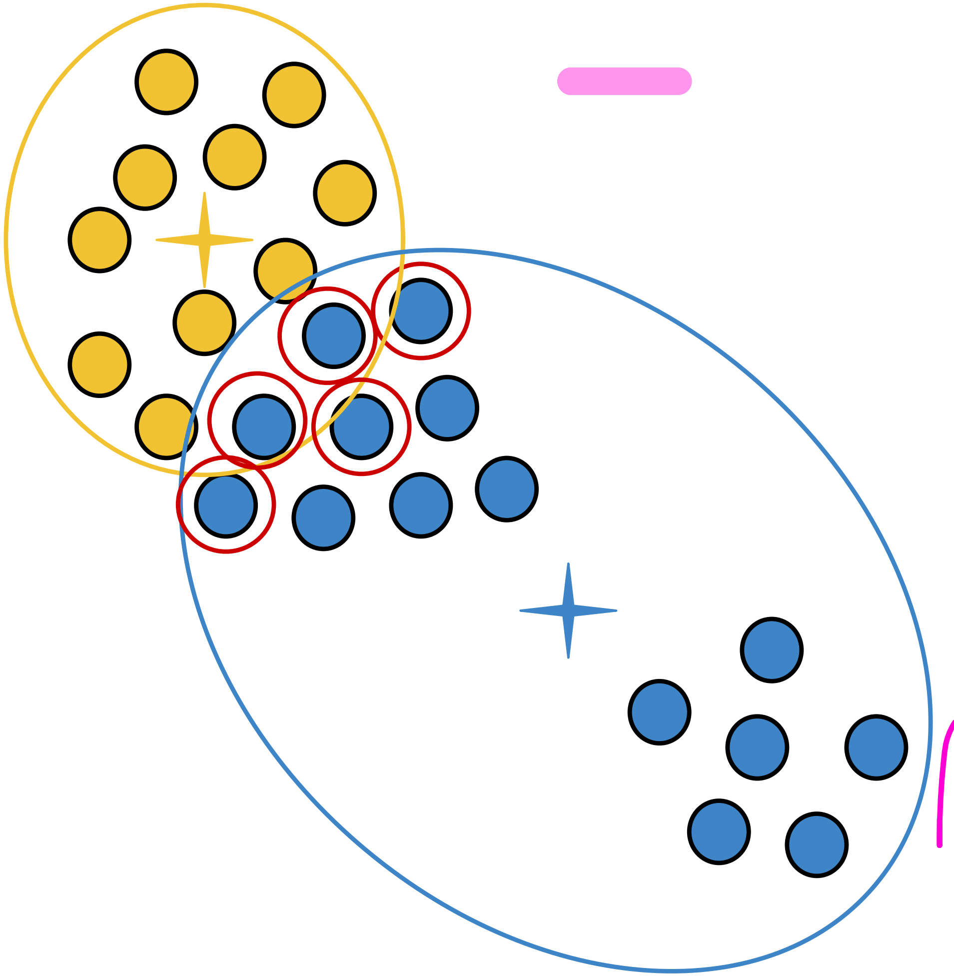
When $\mathbf{x}_i$ lives in more than 2 dimensions, this is a very difficult (**NP-hard**) problem talked a little around here how the cluster would even start off as this in the frist place

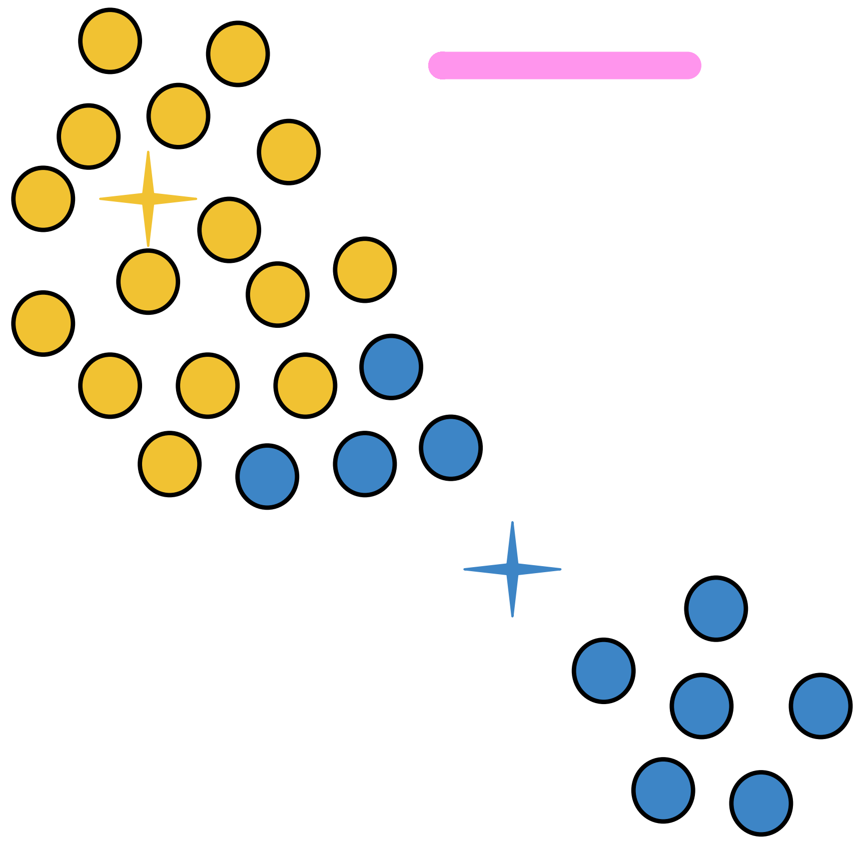
- says that you could plot the centers randomly and then the algorithm would shift our centers correctly



this is a bad because if we take the mean of the clusters (the stars) --->

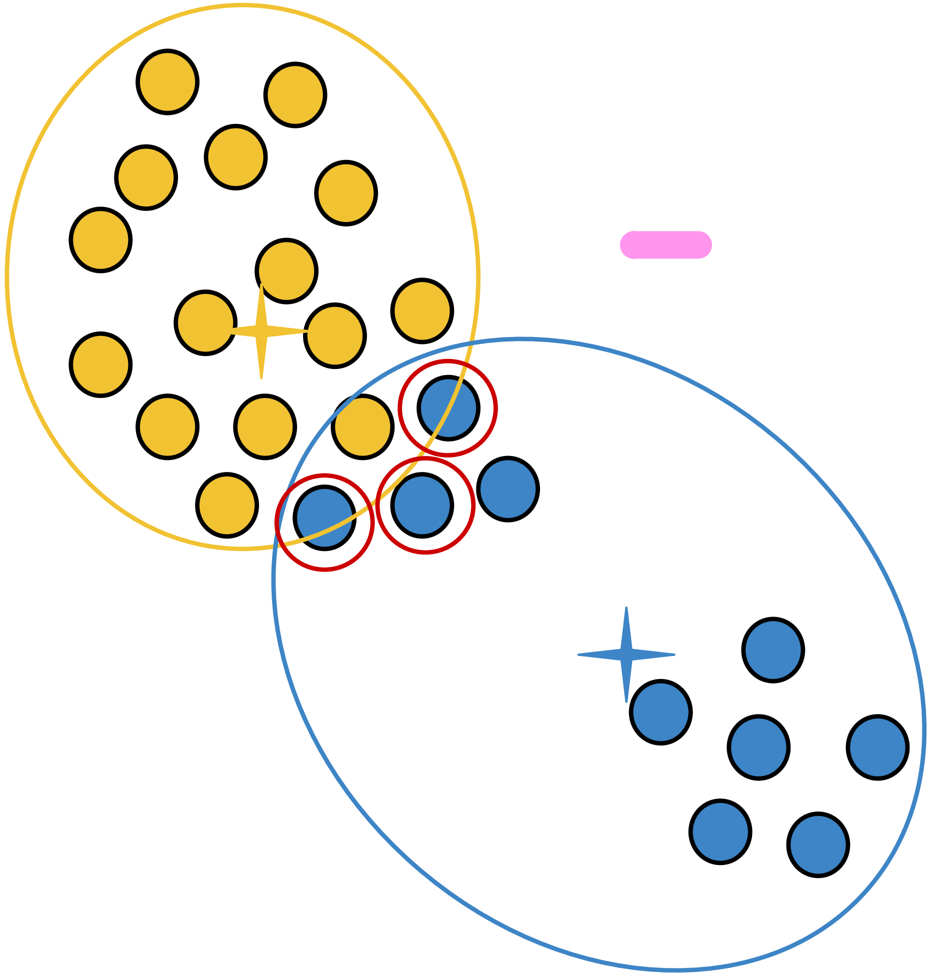
- we see that the red points are closer to the yellow mean than the blue mean

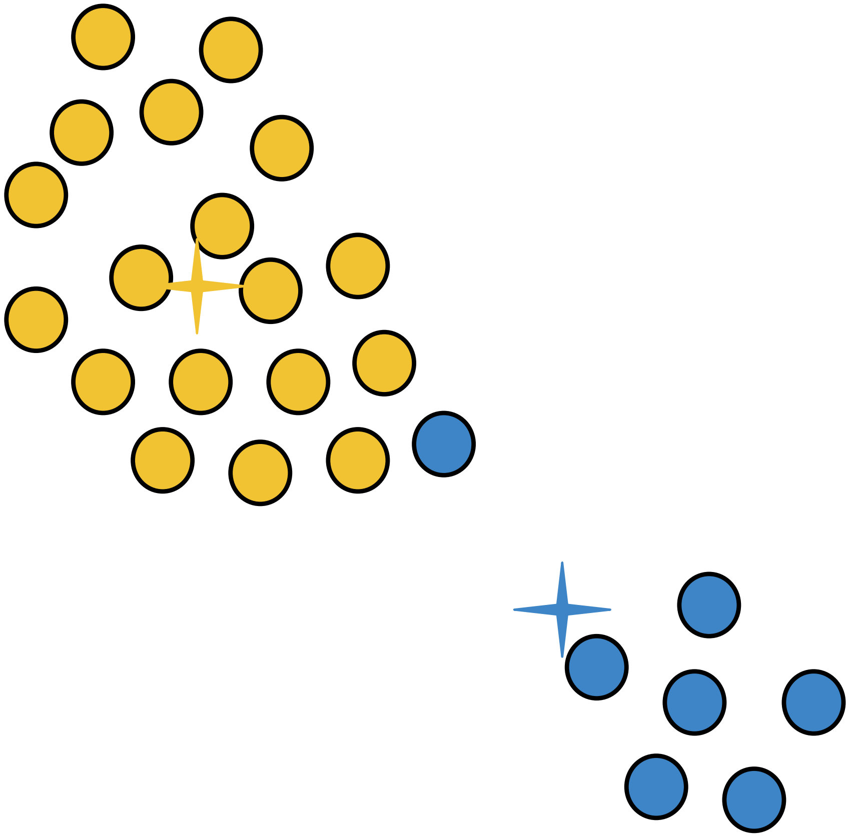


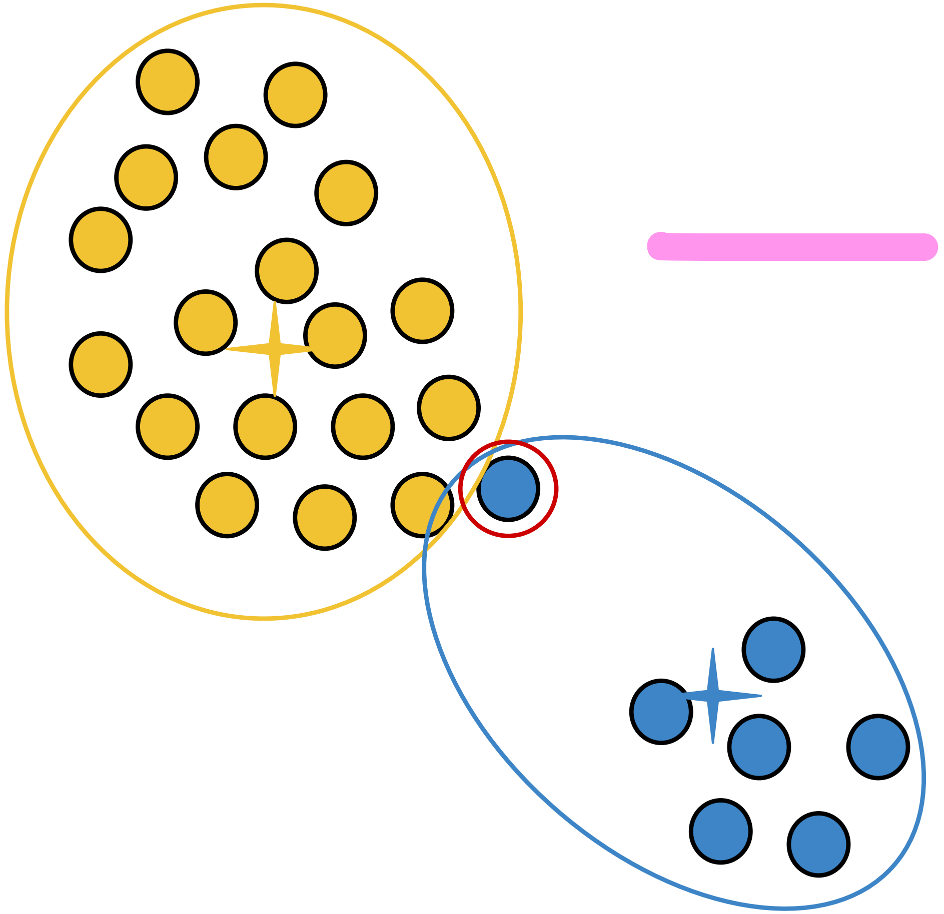


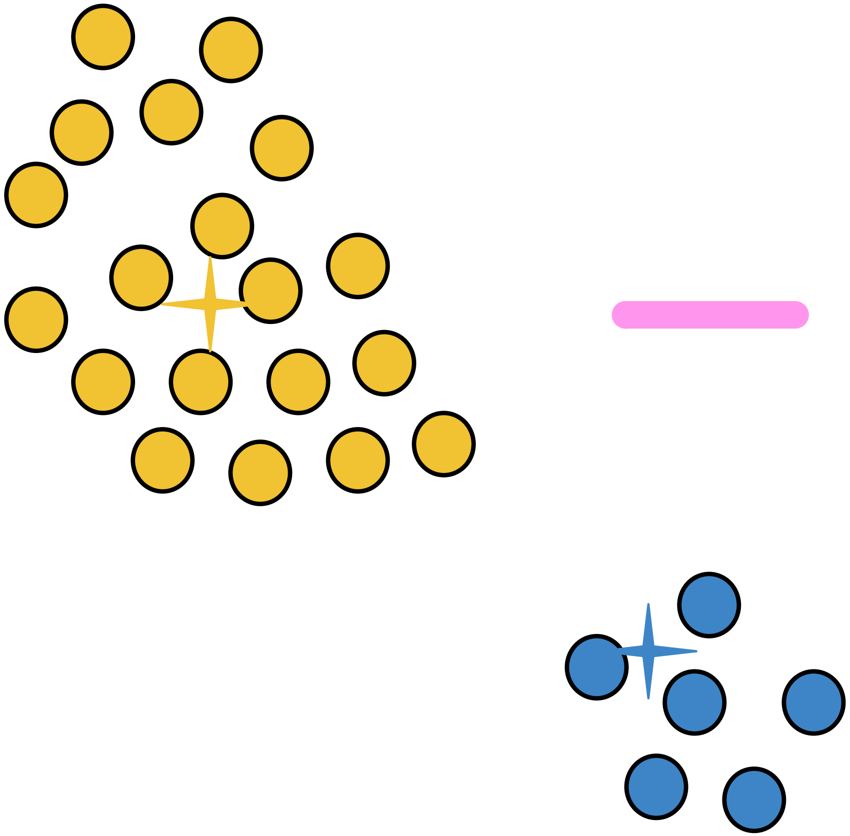
Saying that if we convert a couple of the reds to the yellow --->

- it won't increase the variance of the yellow points that much because those points are already close to the yellow mean
- in contrast --> the blue points ---> the farther ones from the mean penalize the cost function heavily (because its a SQUARED DIFFERENCE FROM THE MEAN)
    - so, moving over the edges hevaily helps reduce the variance





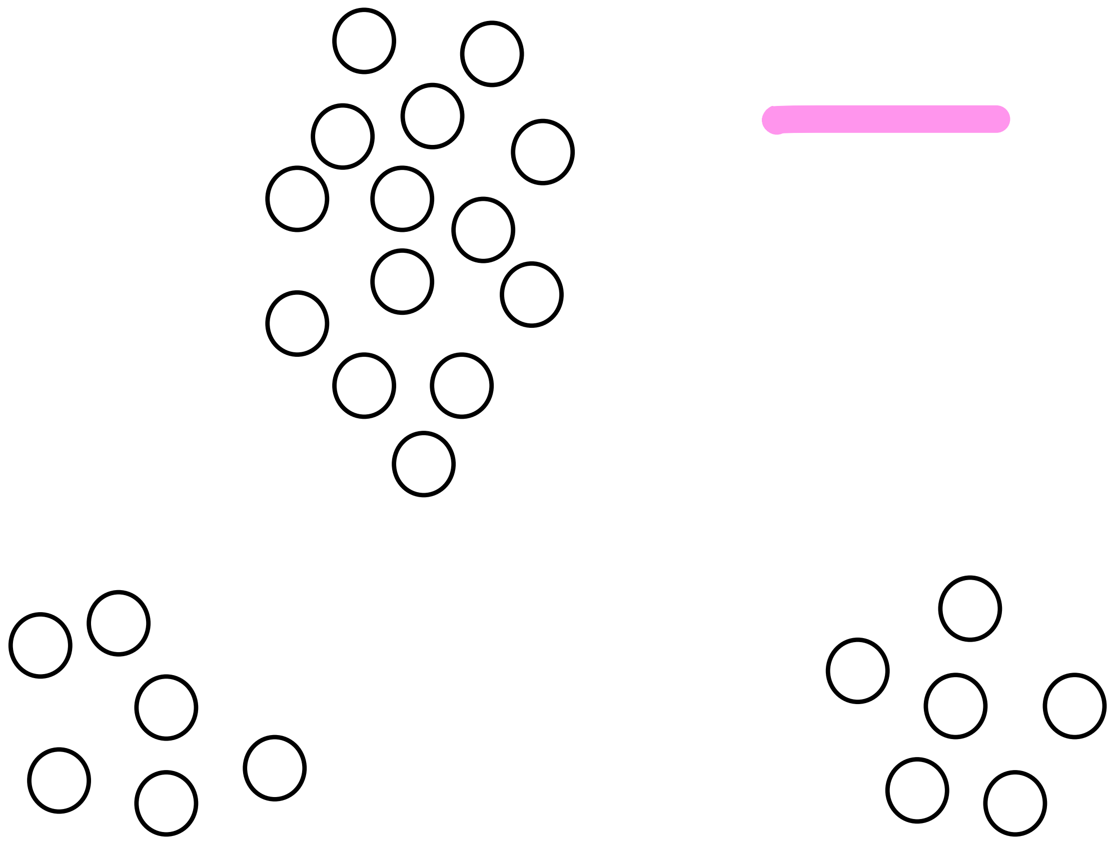


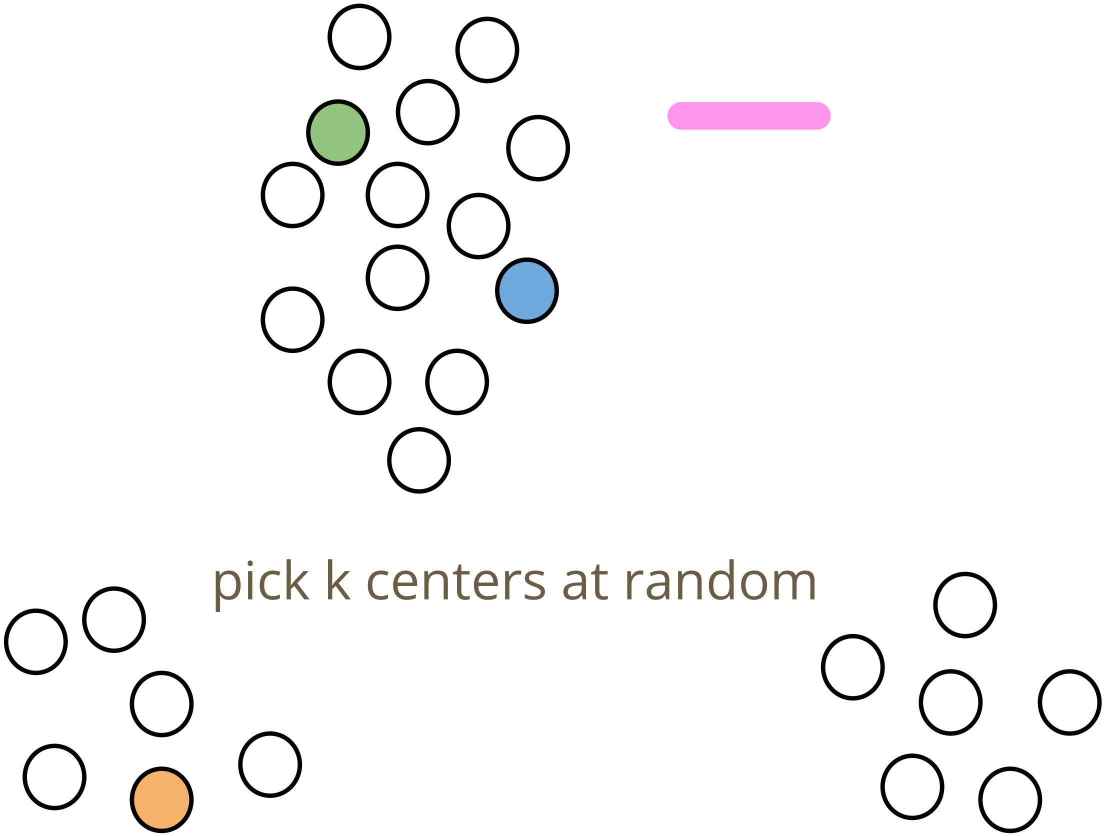


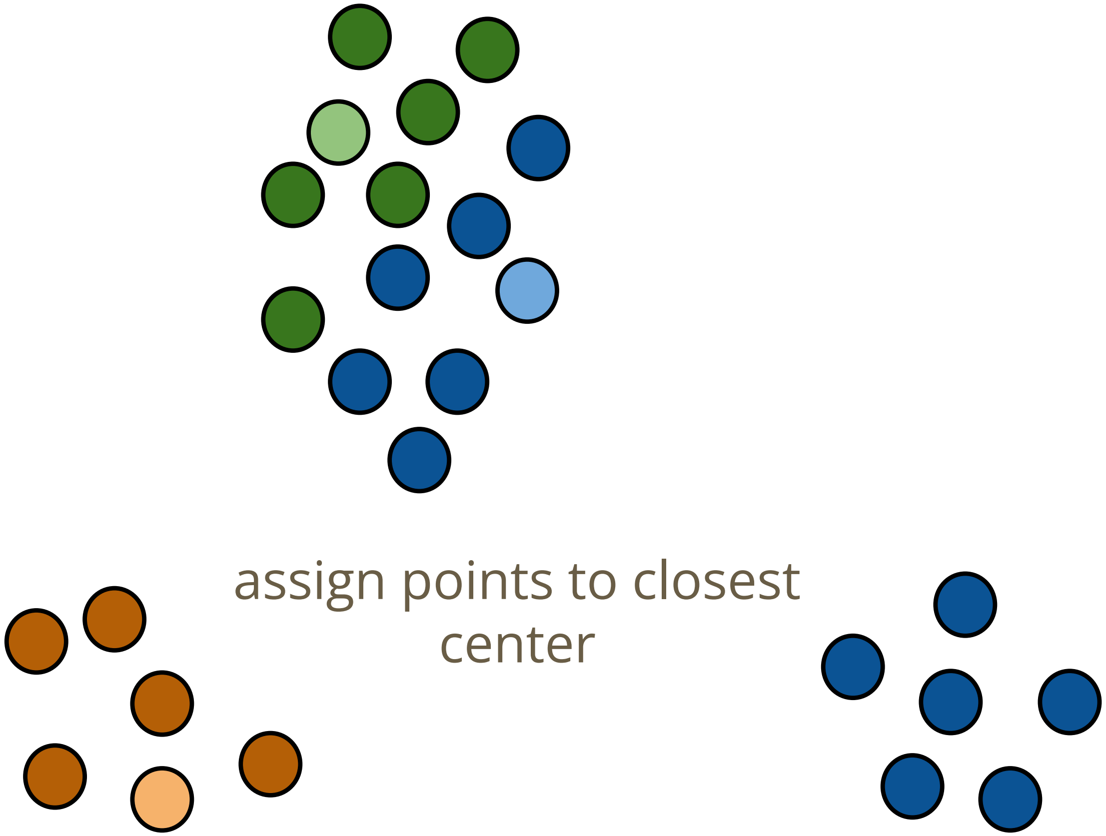
# K-means - Lloyd's Algorithm

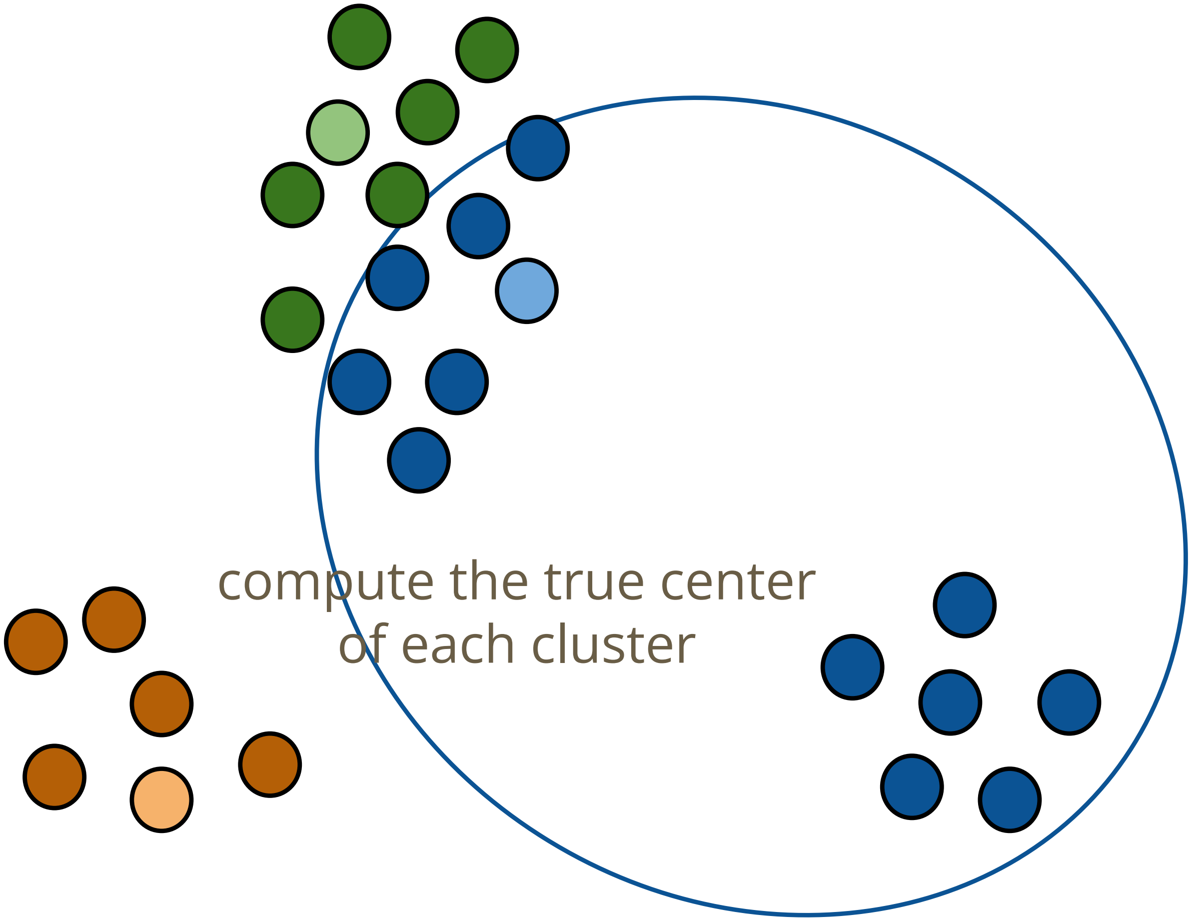
1. Randomly pick **k** centers $\{\mu_1, \ldots, \mu_k\}$
2. Assign each point in the dataset to its closest center
3. Compute the new centers as the means of each cluster
4. Repeat 2 & 3 until convergence

yes -->
always
converges

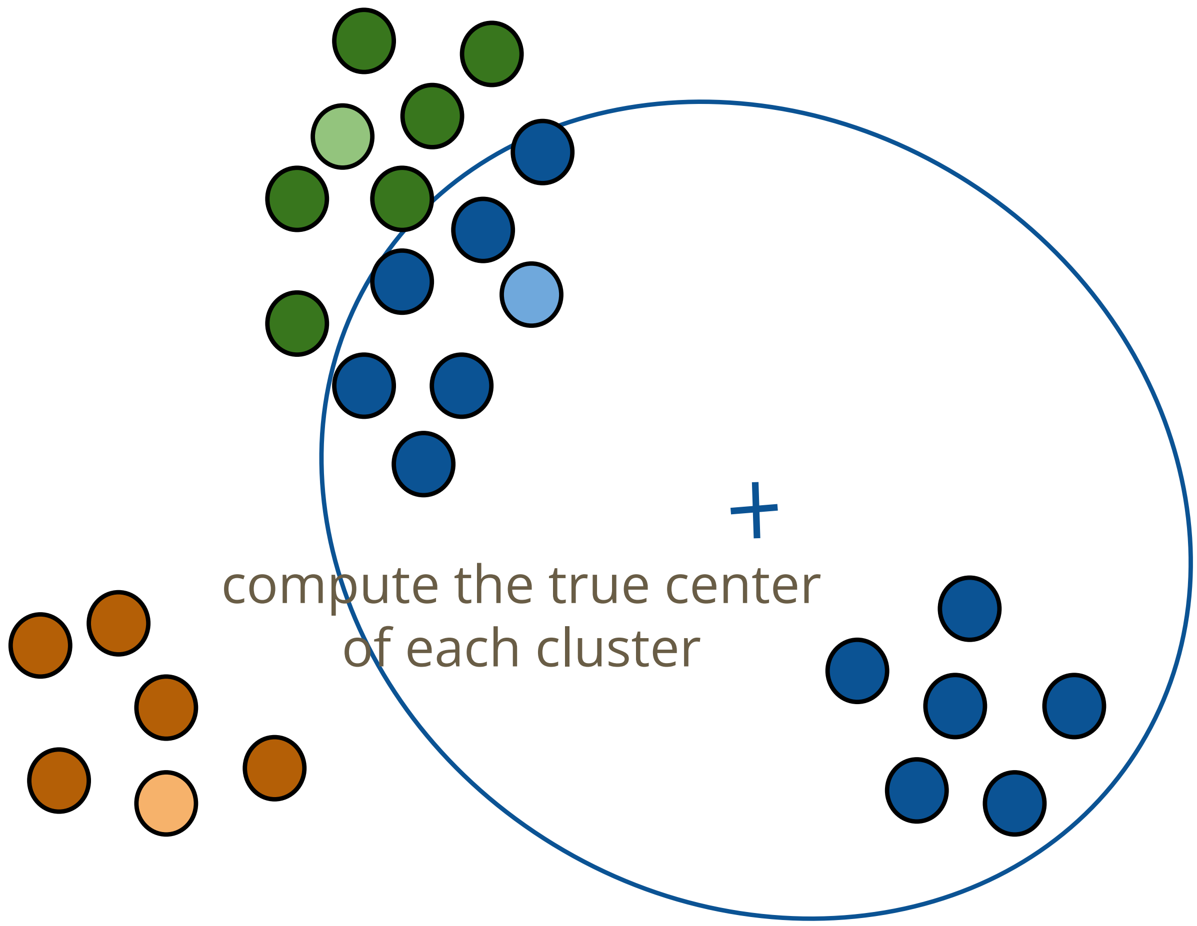


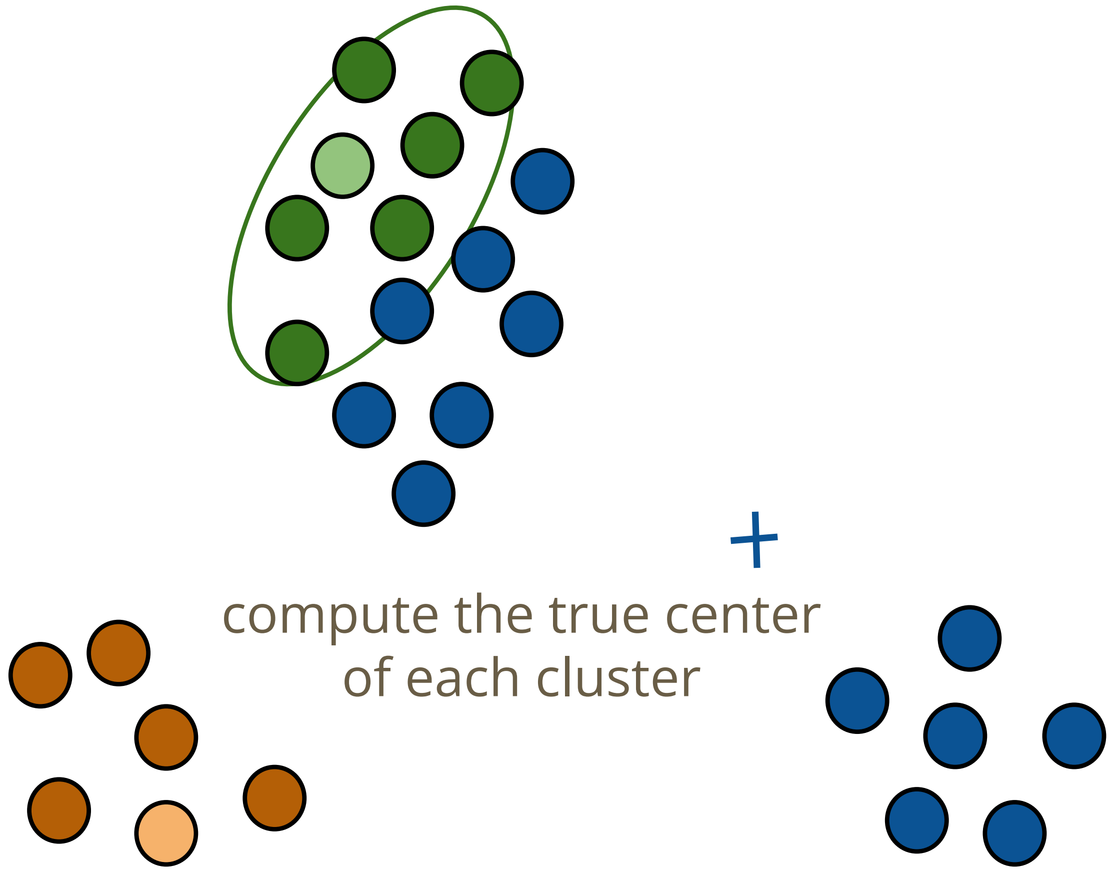

pick k centers at random

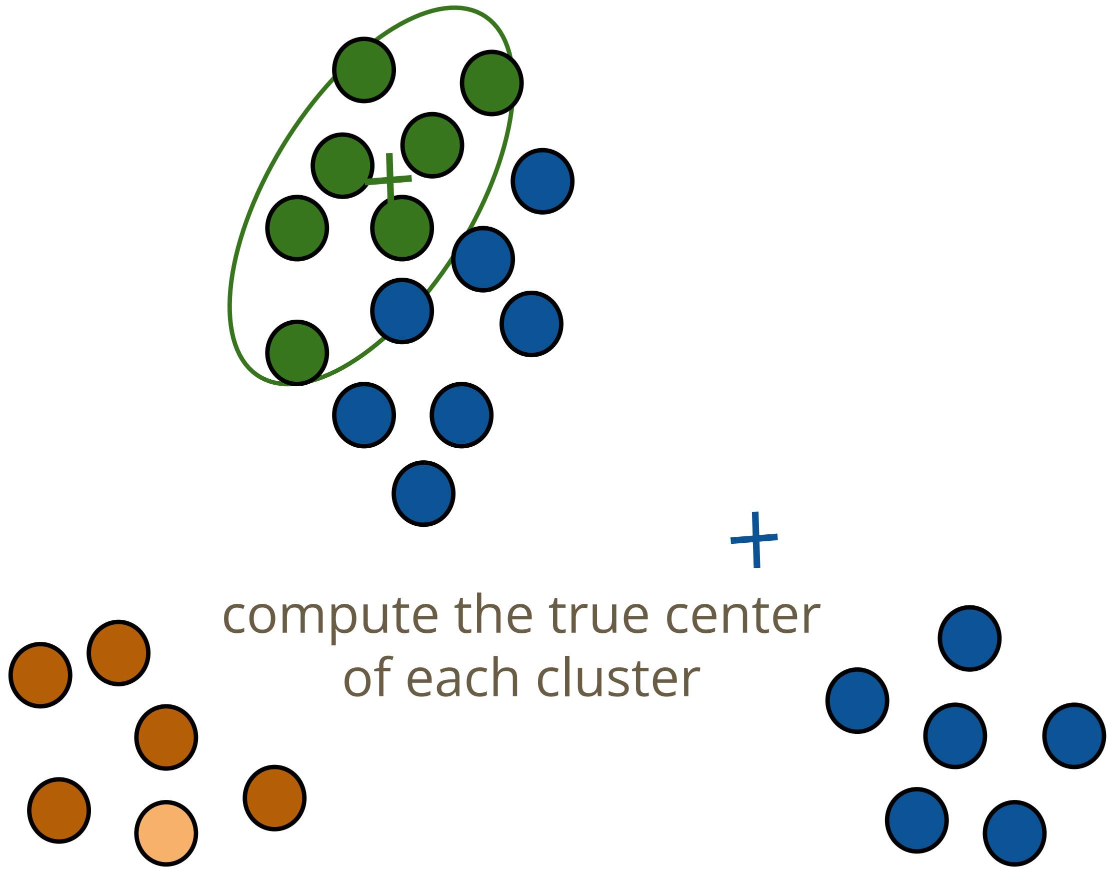

assign points to closest center

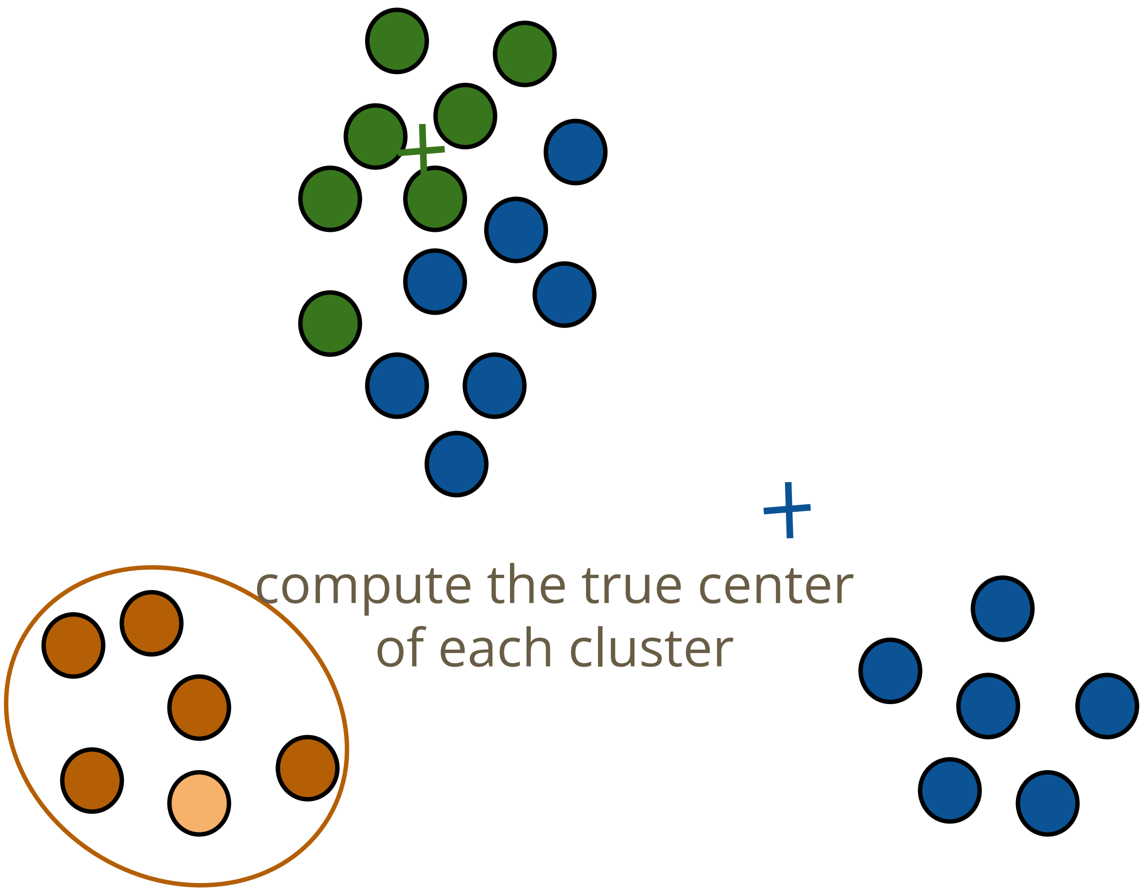


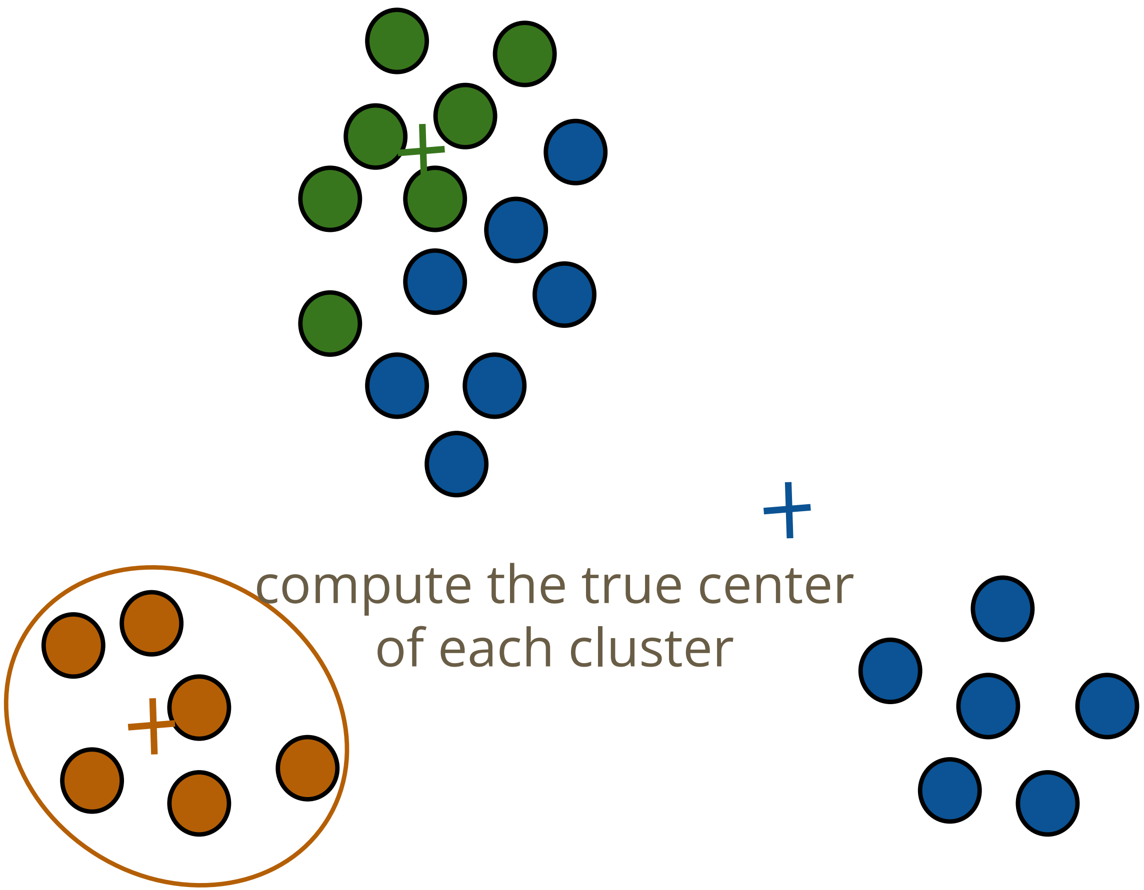
compute the true center
of each cluster

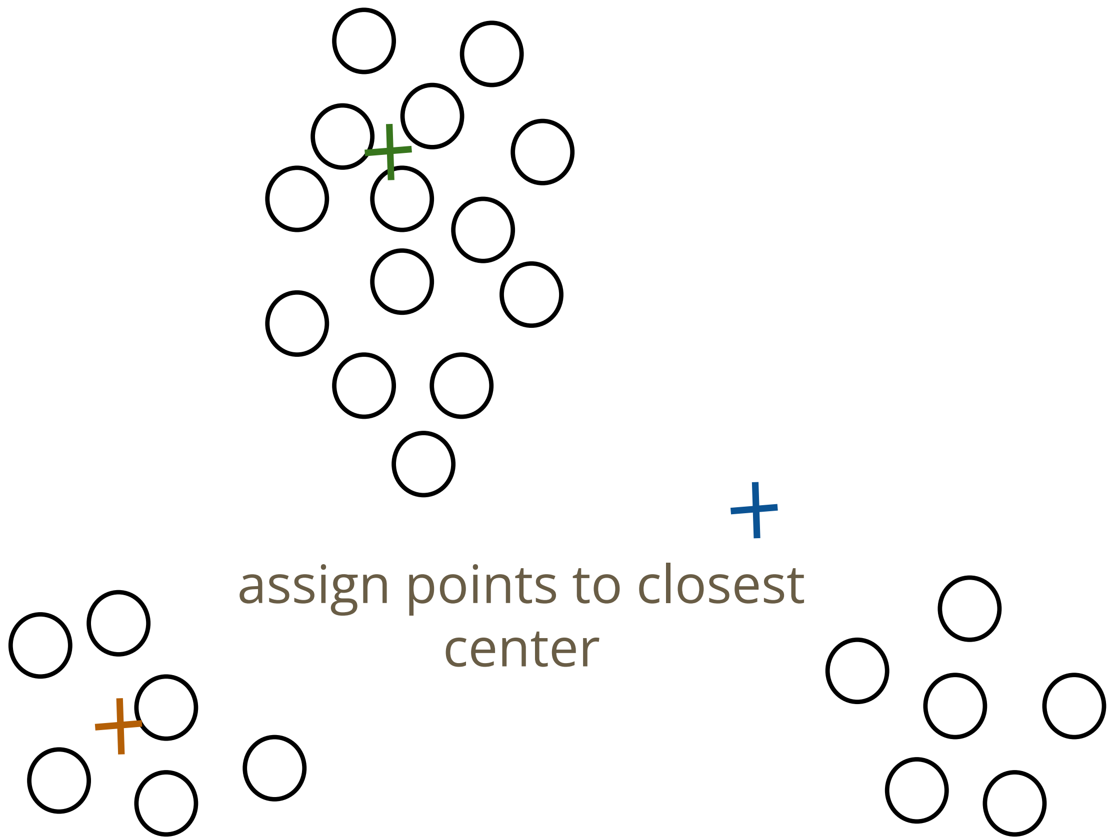

compute the true center
of each cluster

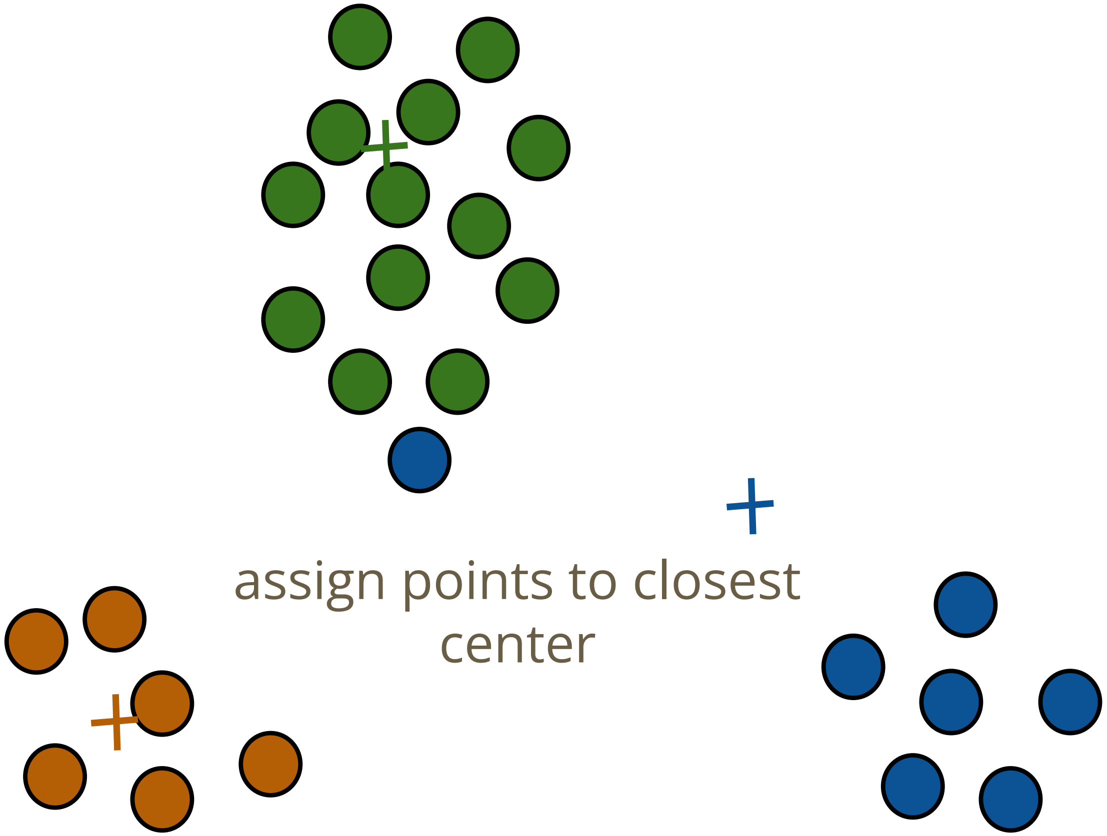

compute the true center
of each cluster

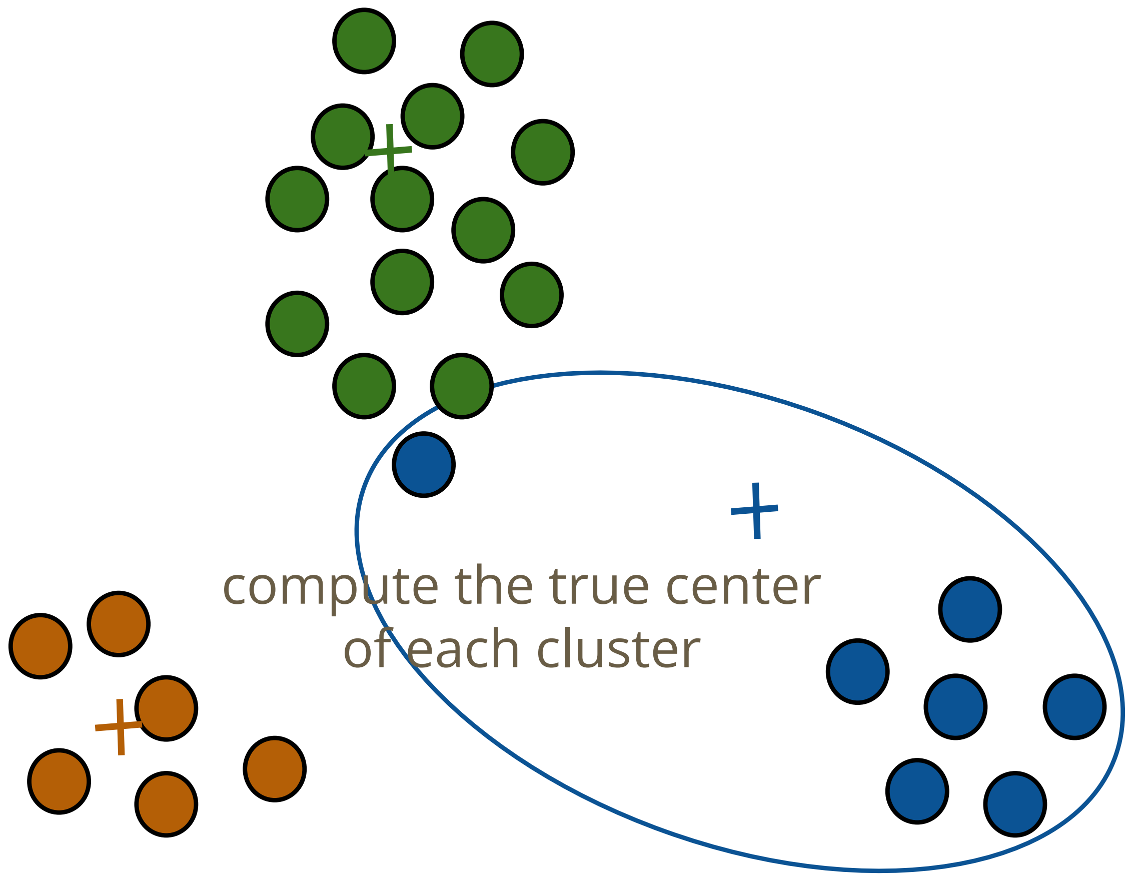

compute the true center
of each cluster

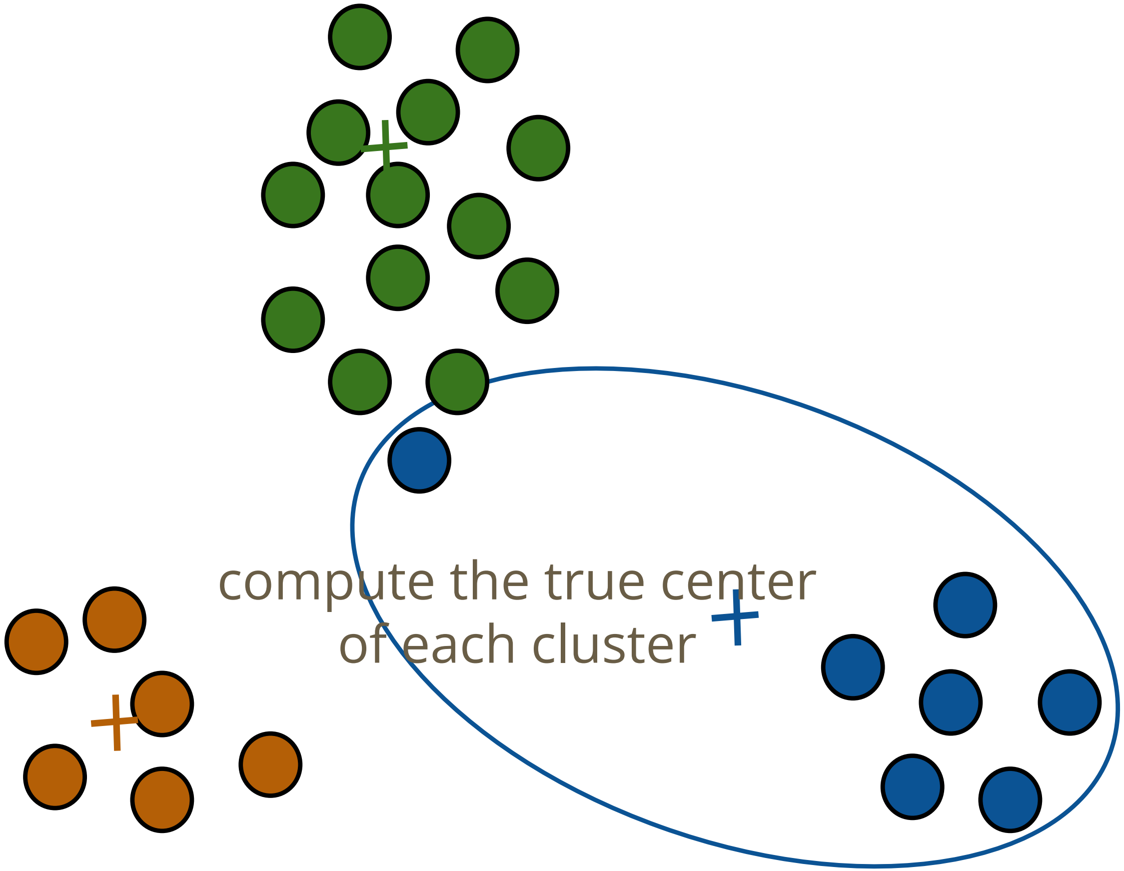

compute the true center
of each cluster

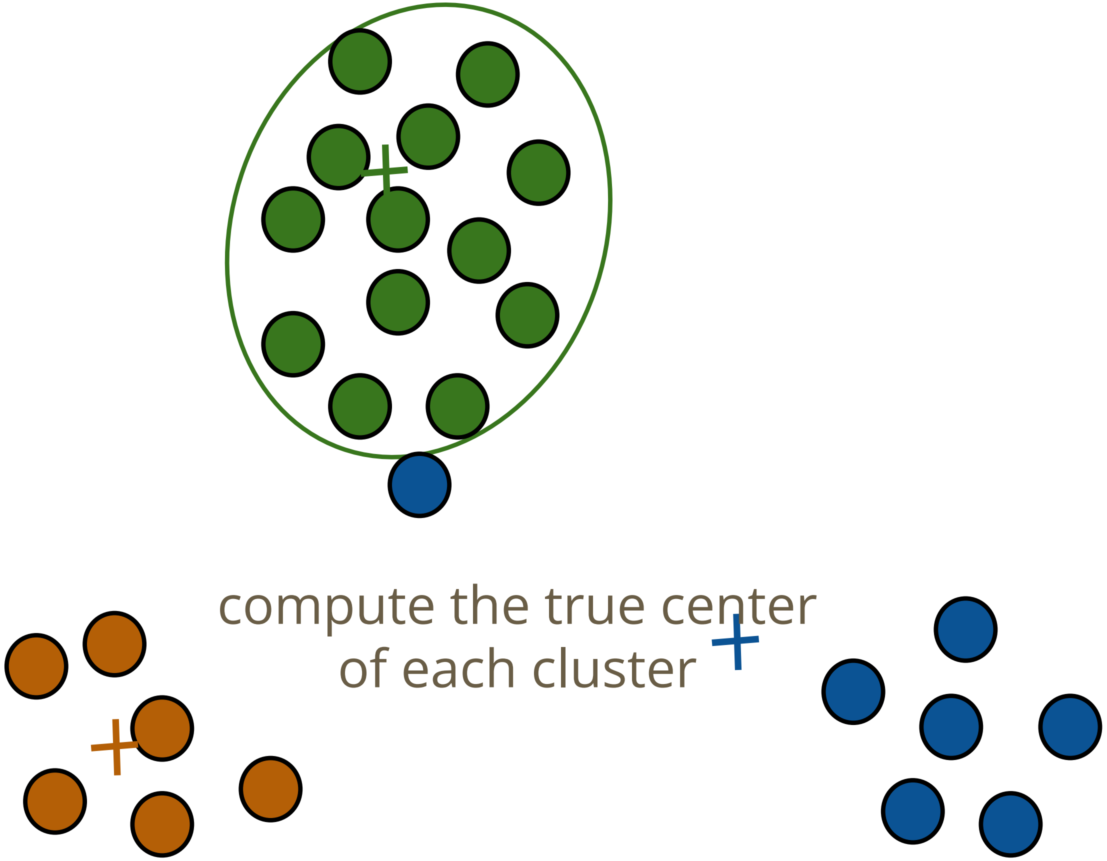

compute the true center
of each cluster

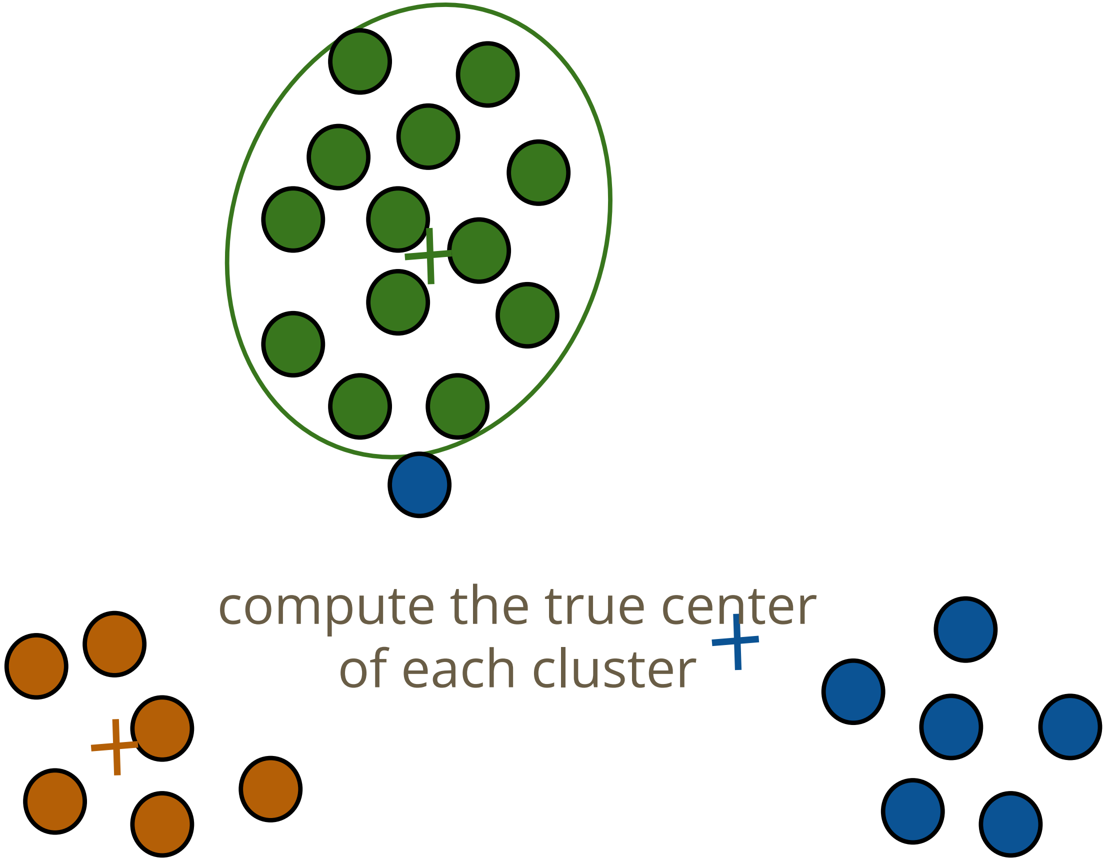

assign points to closest center

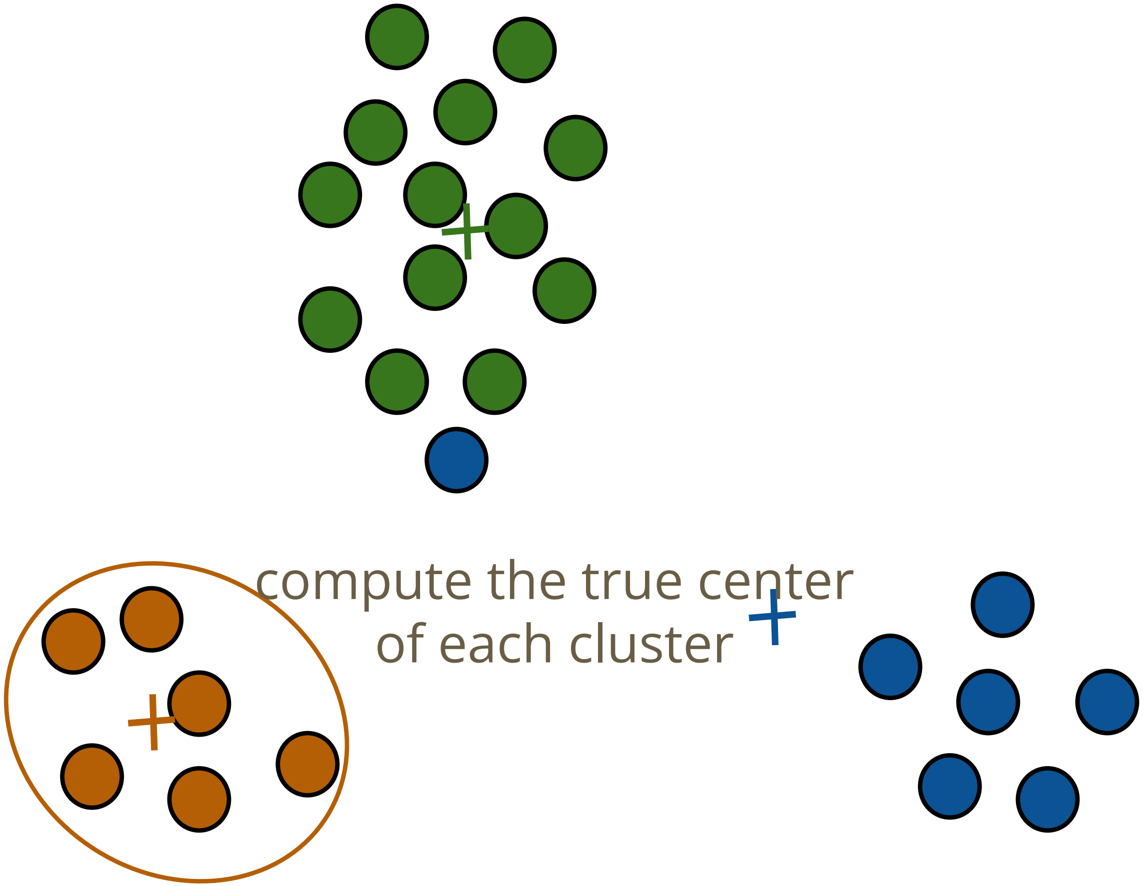

assign points to closest center

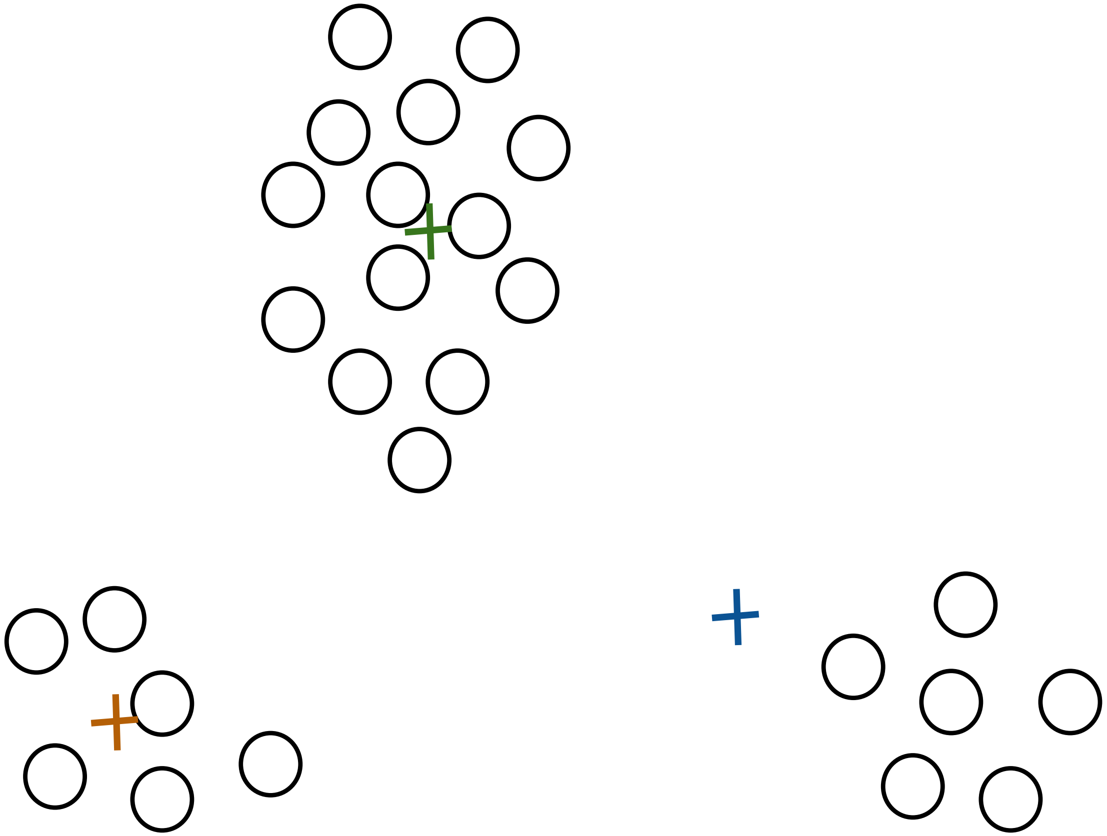

compute the true center
of each cluster

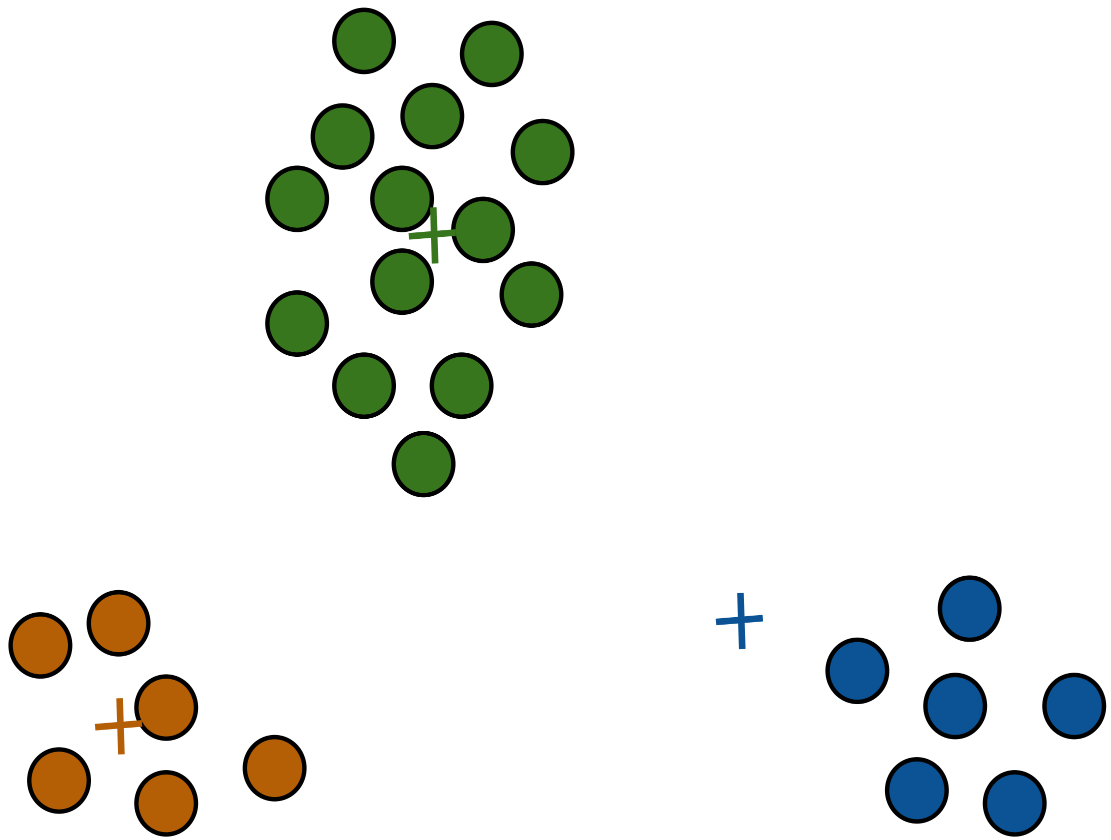

compute the true center
of each cluster

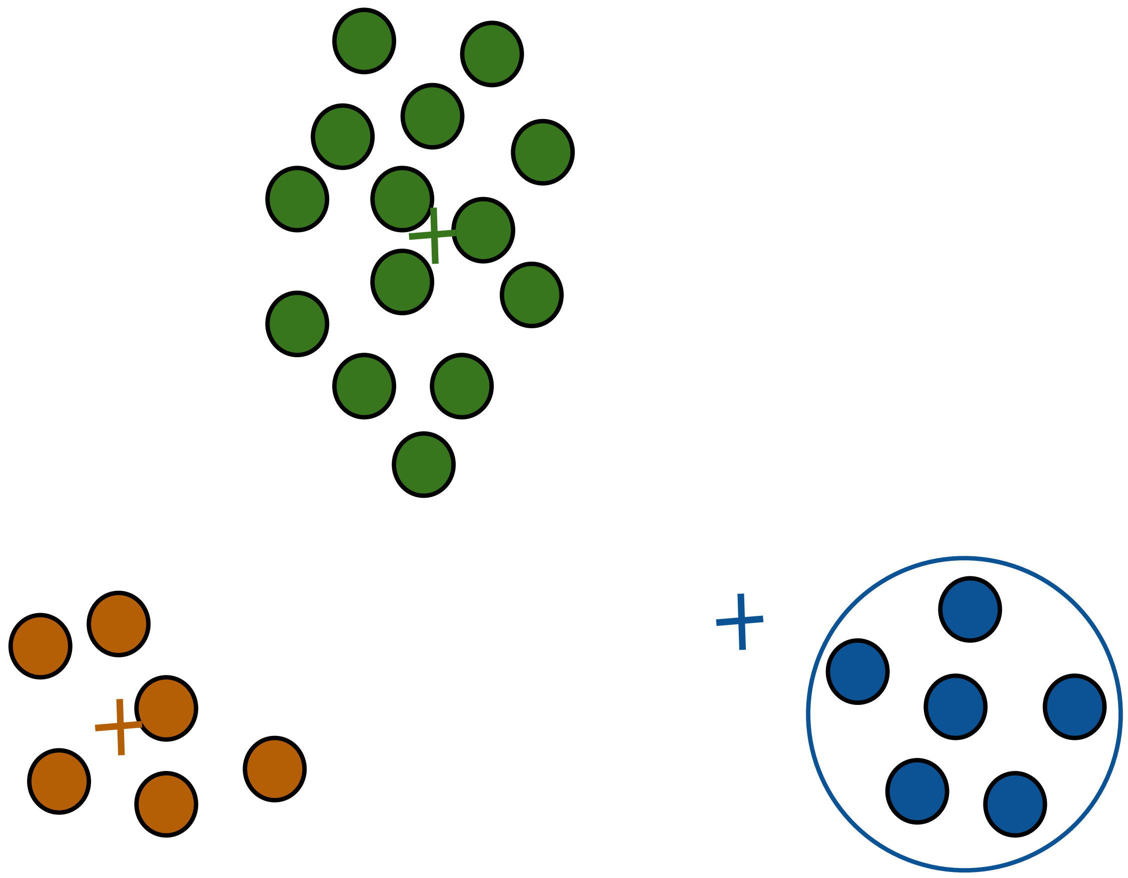

compute the true center
of each cluster

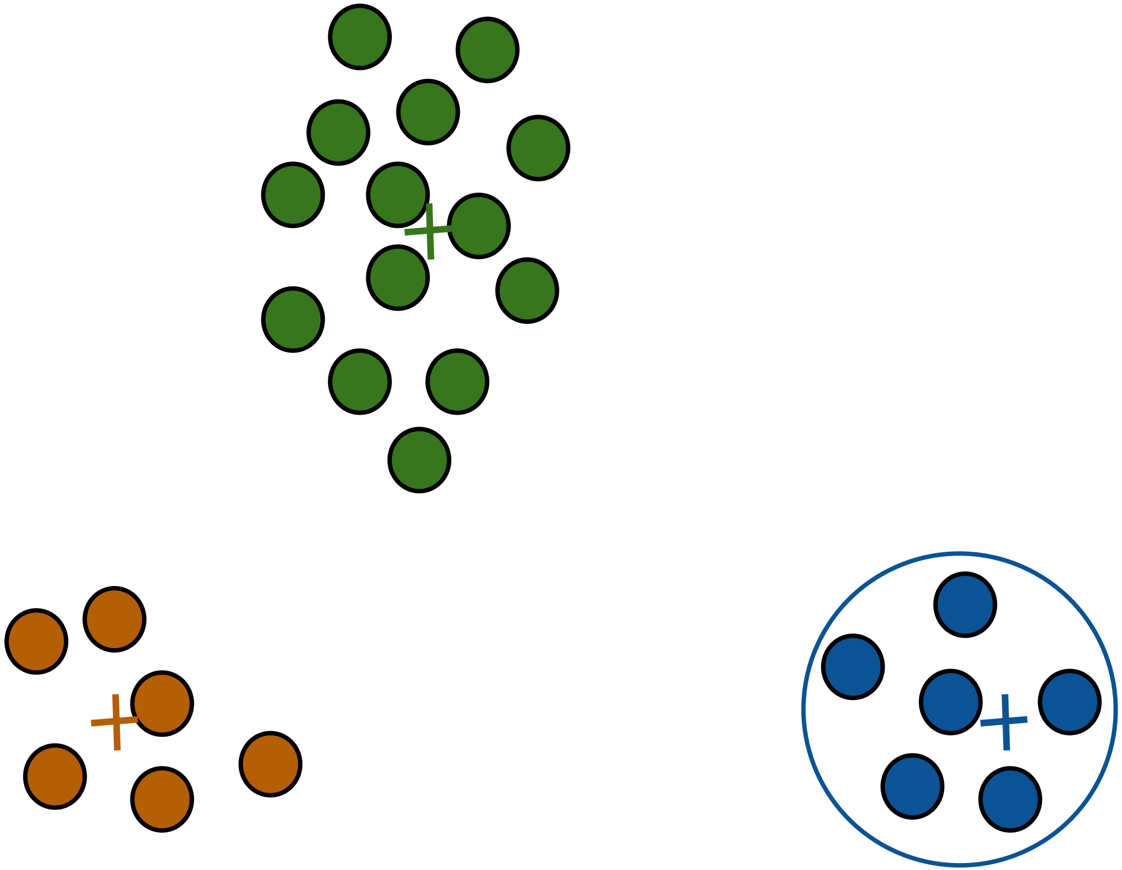

compute the true center
of each cluster

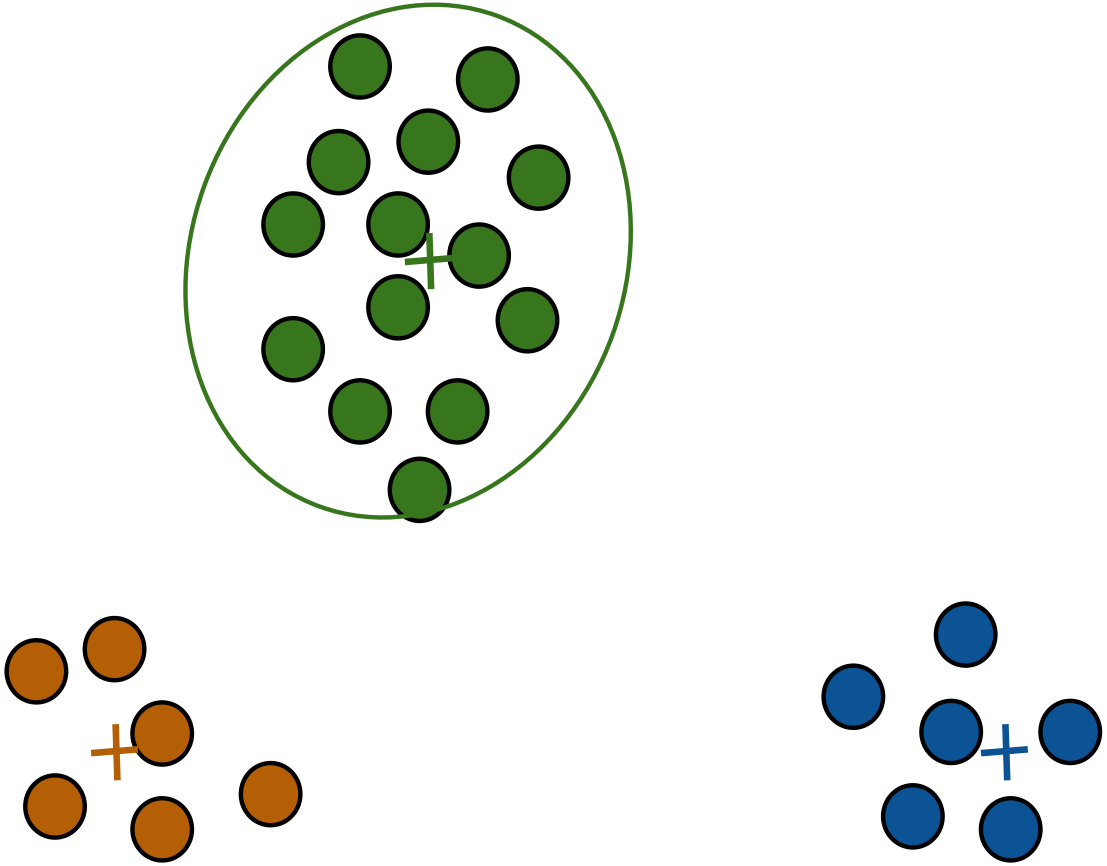

compute the true center
of each cluster

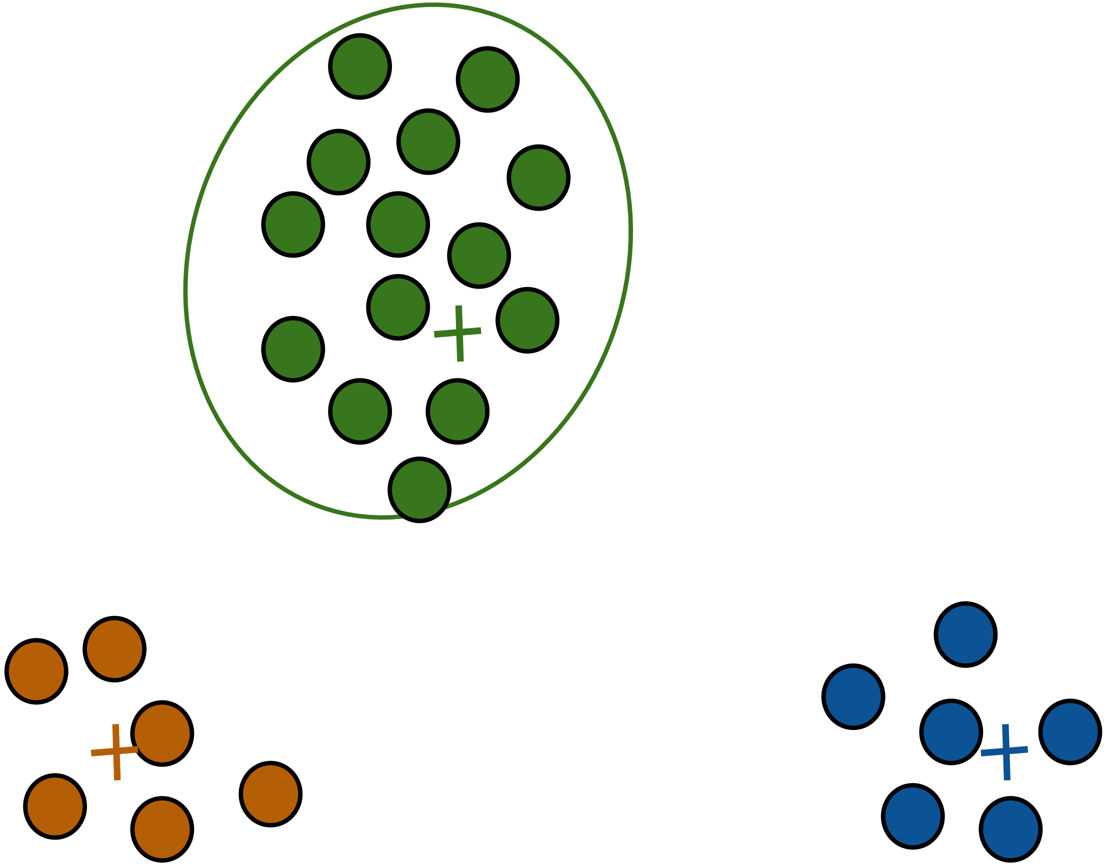


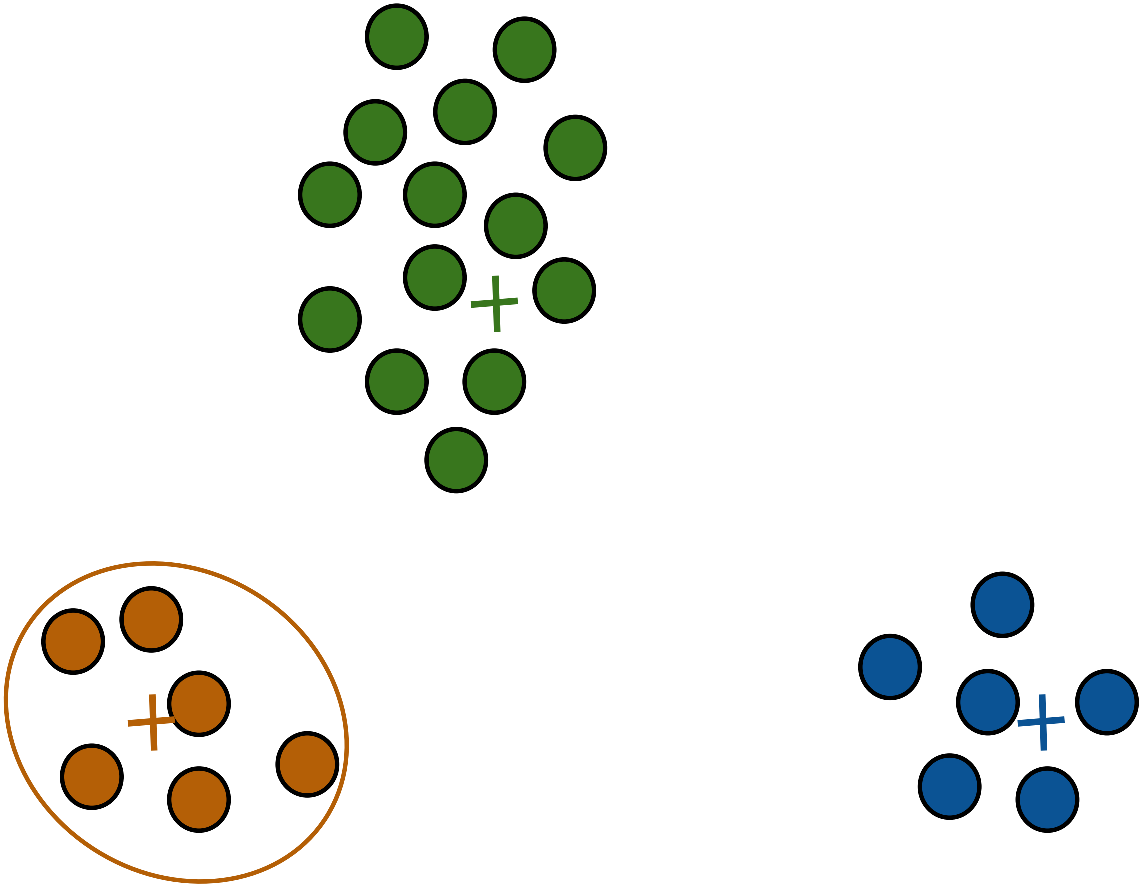


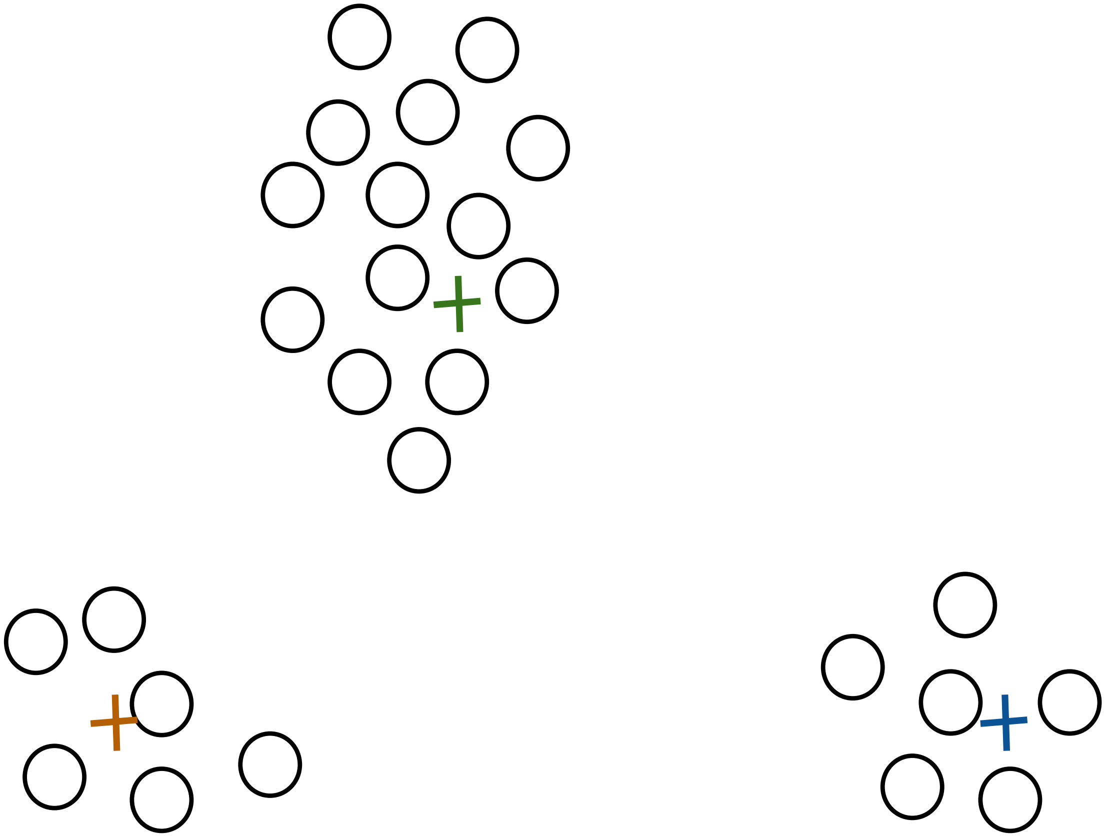


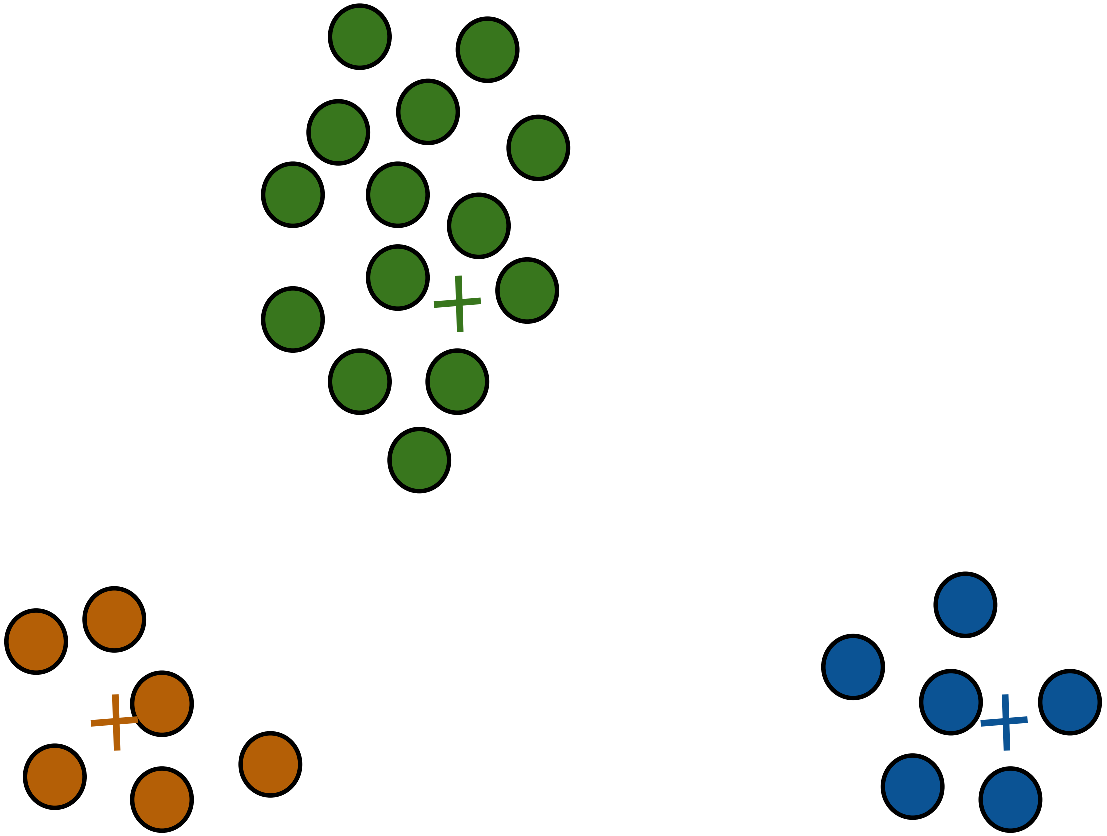


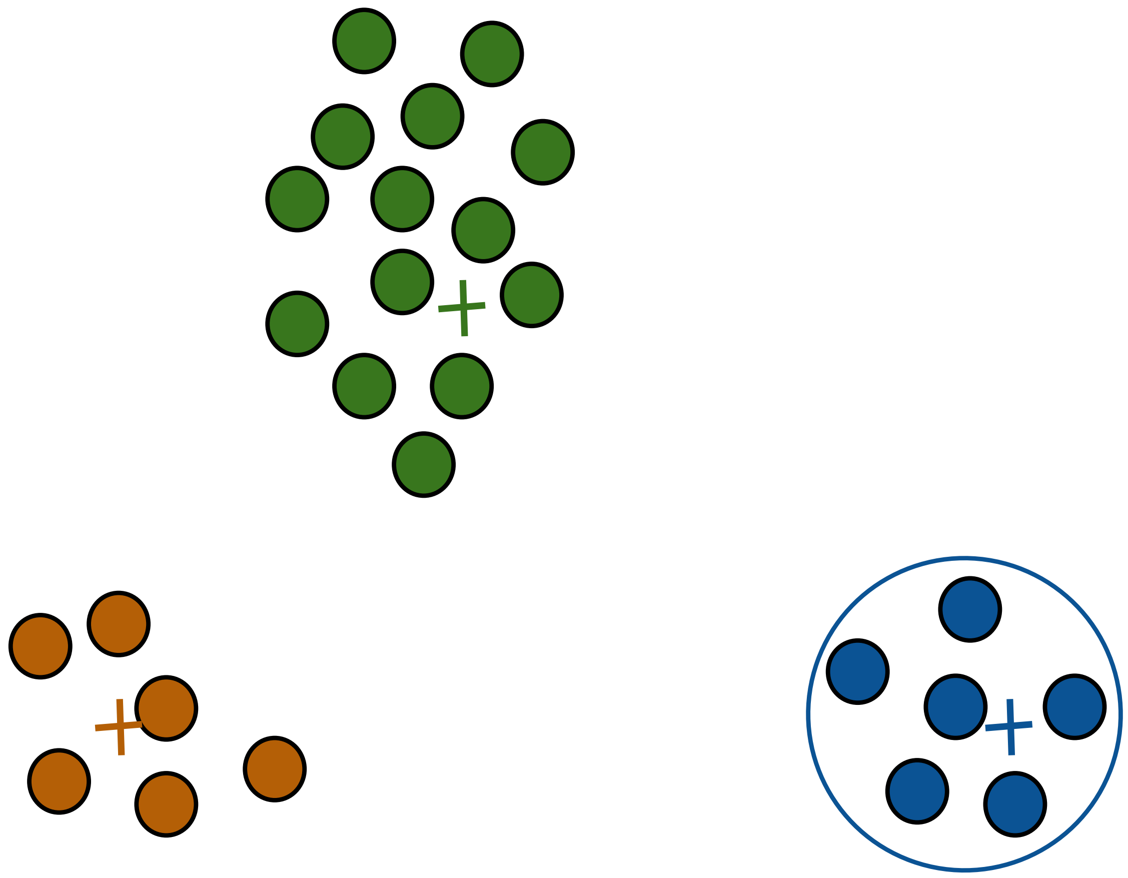


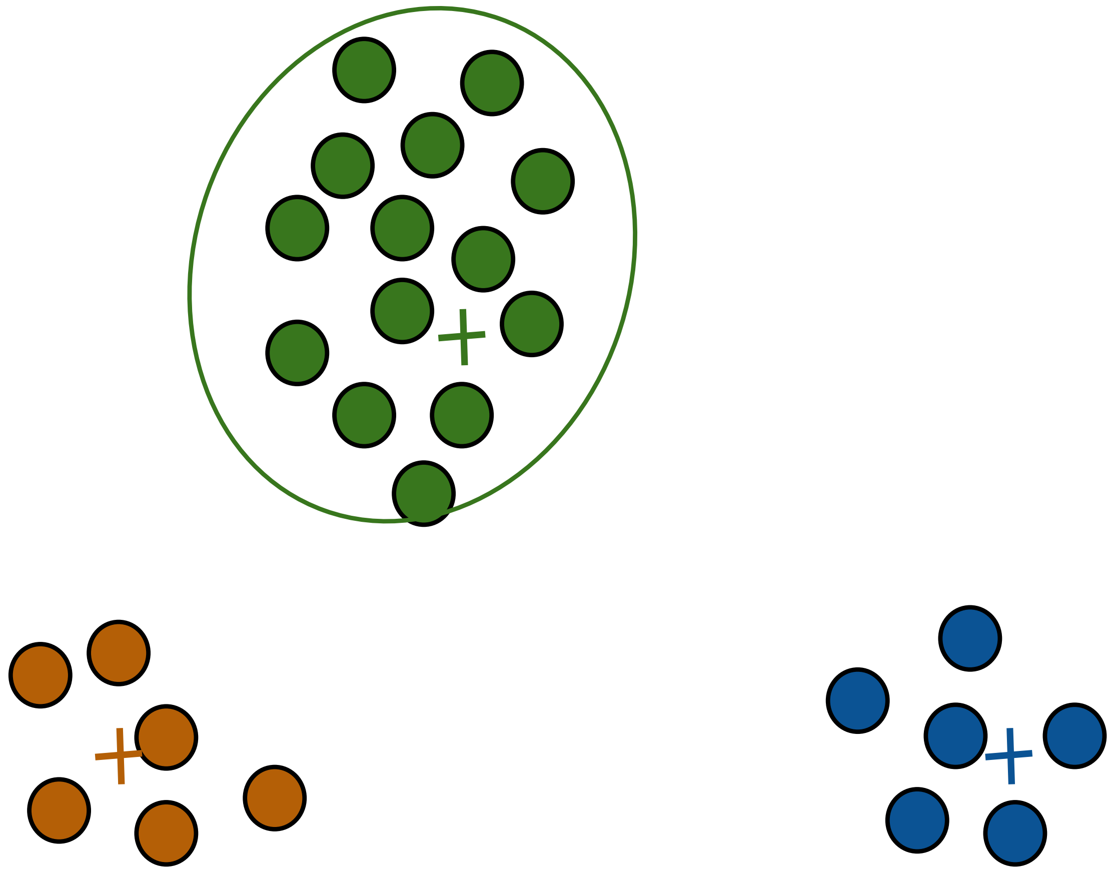


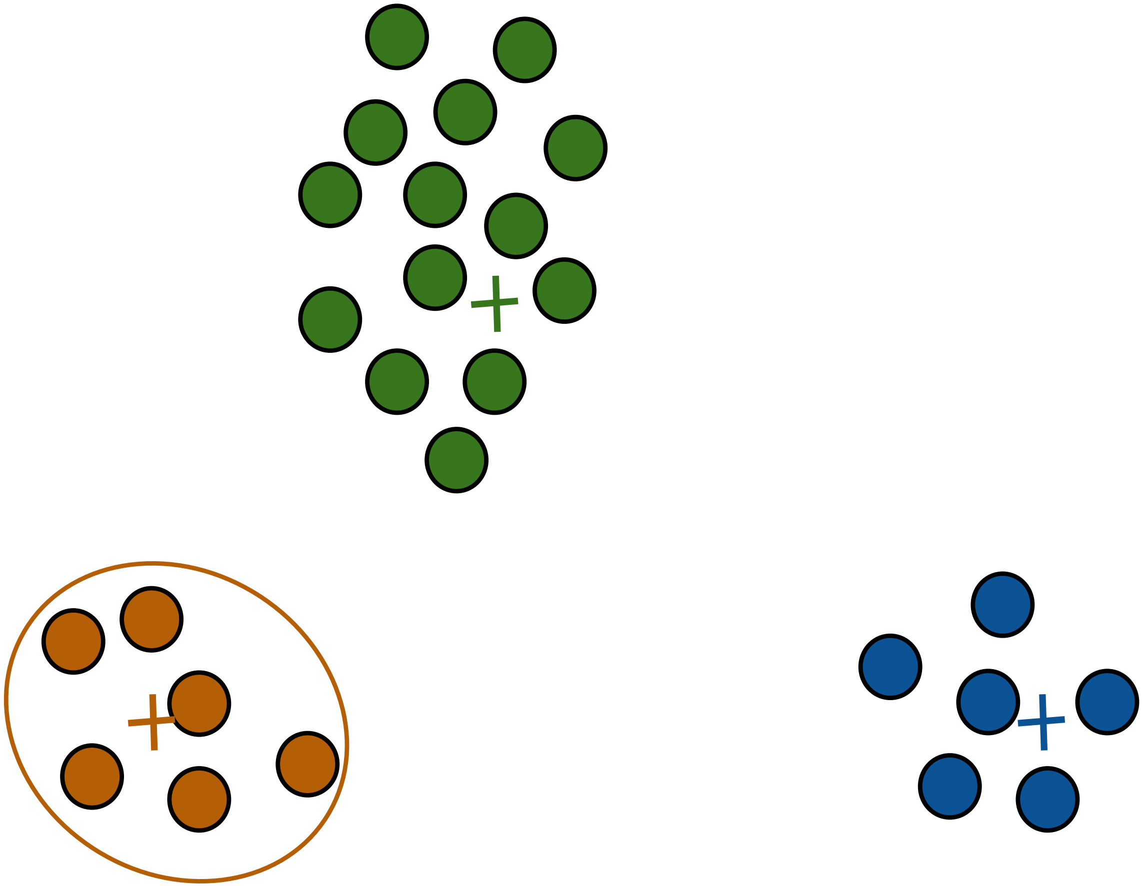


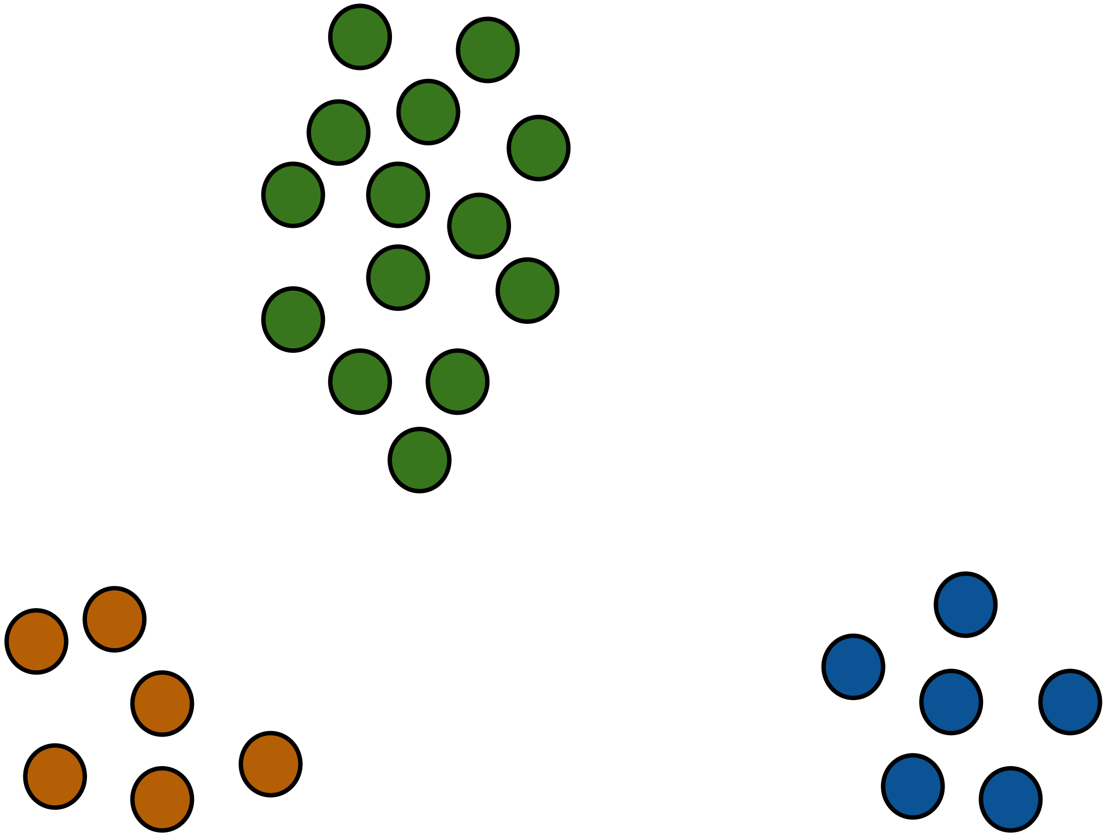










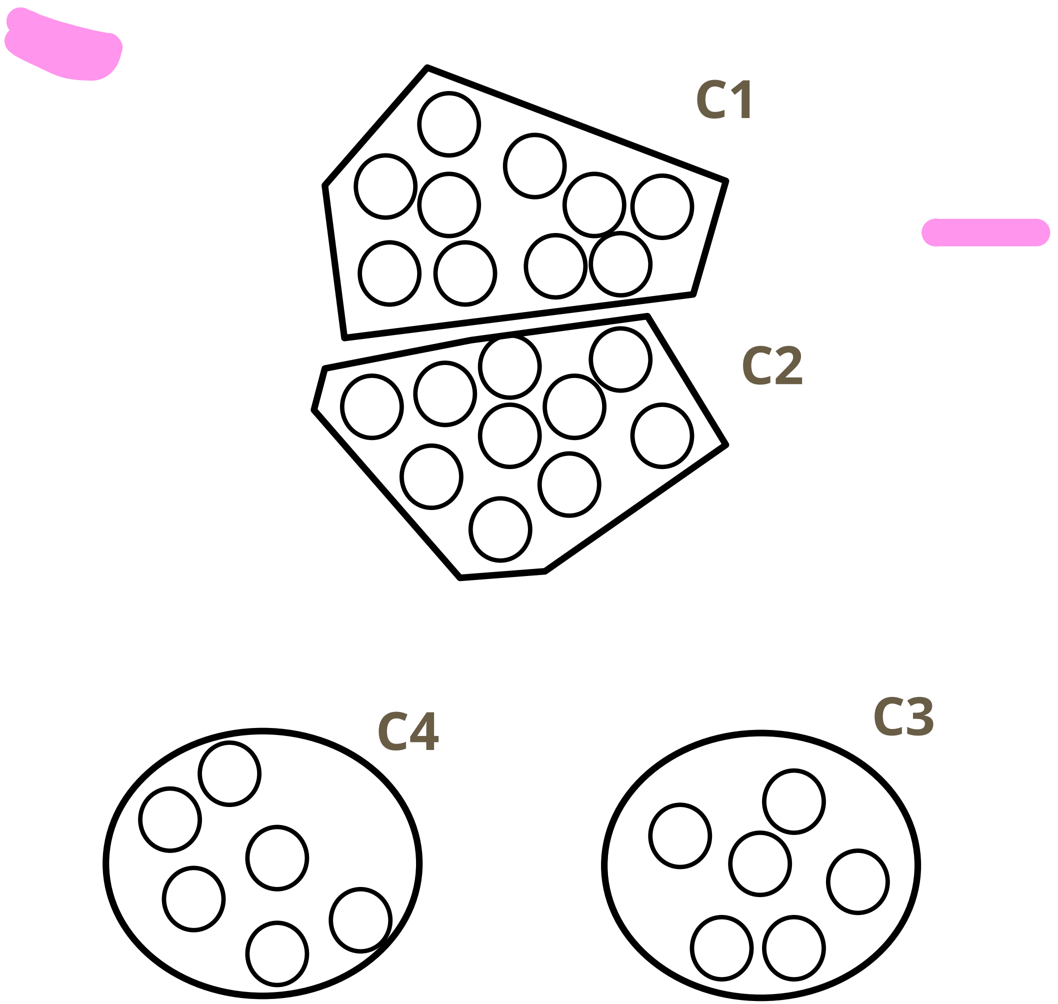


# Questions

going over live
coding example
around here

"if one more step is possible, you should say no"

if it has converged
--> say yes
---> "is this a possible final state for lloyds"




1

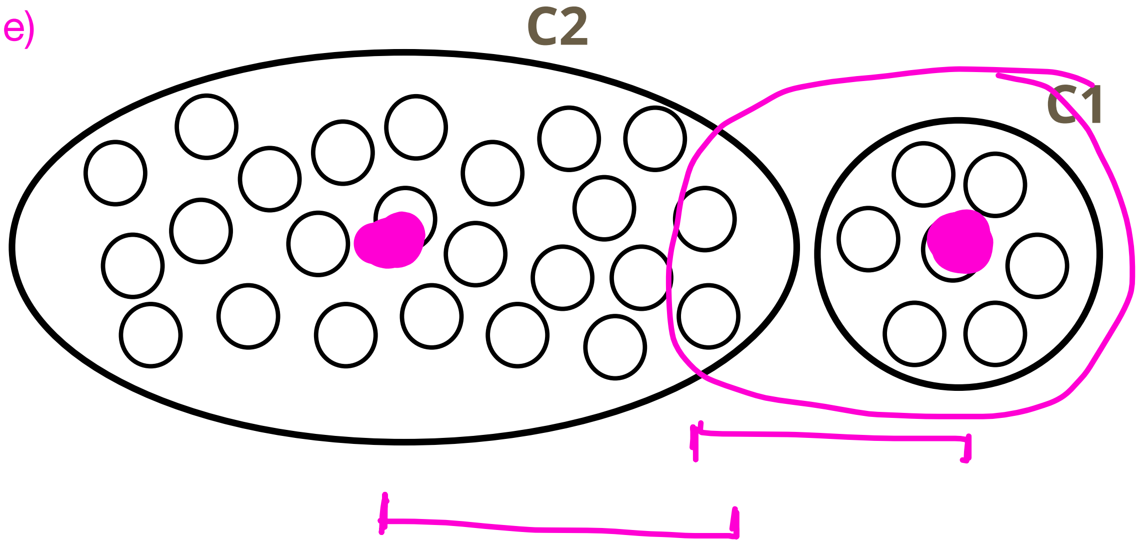
yes

think about it like this:

- the images are a snapshot of the lloyds algorithm **right after the points are assigned a center**
- so, what we need to do is compute the **new center** (the pink dot) and determine -----> if there are points that are closer to the new centers that are in a different cluster, then answer no (i.e. one more step is possible)

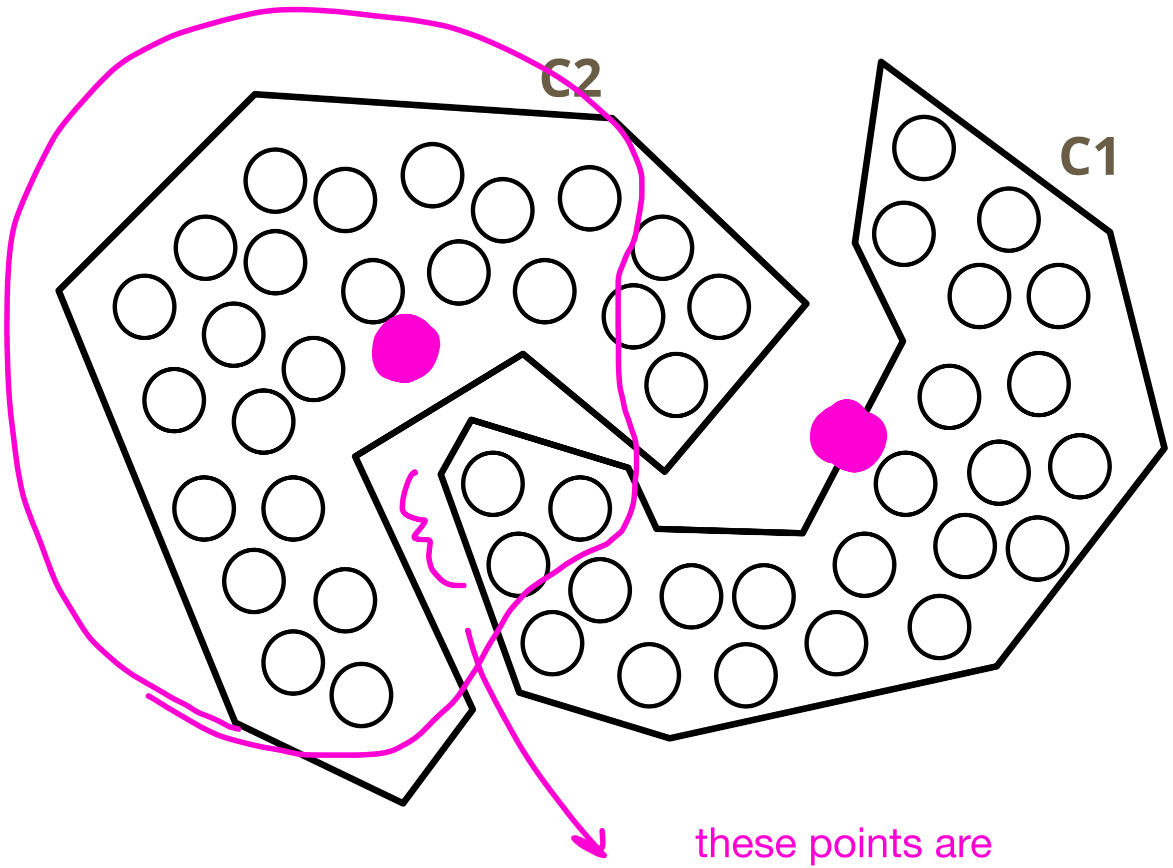
2

no



3

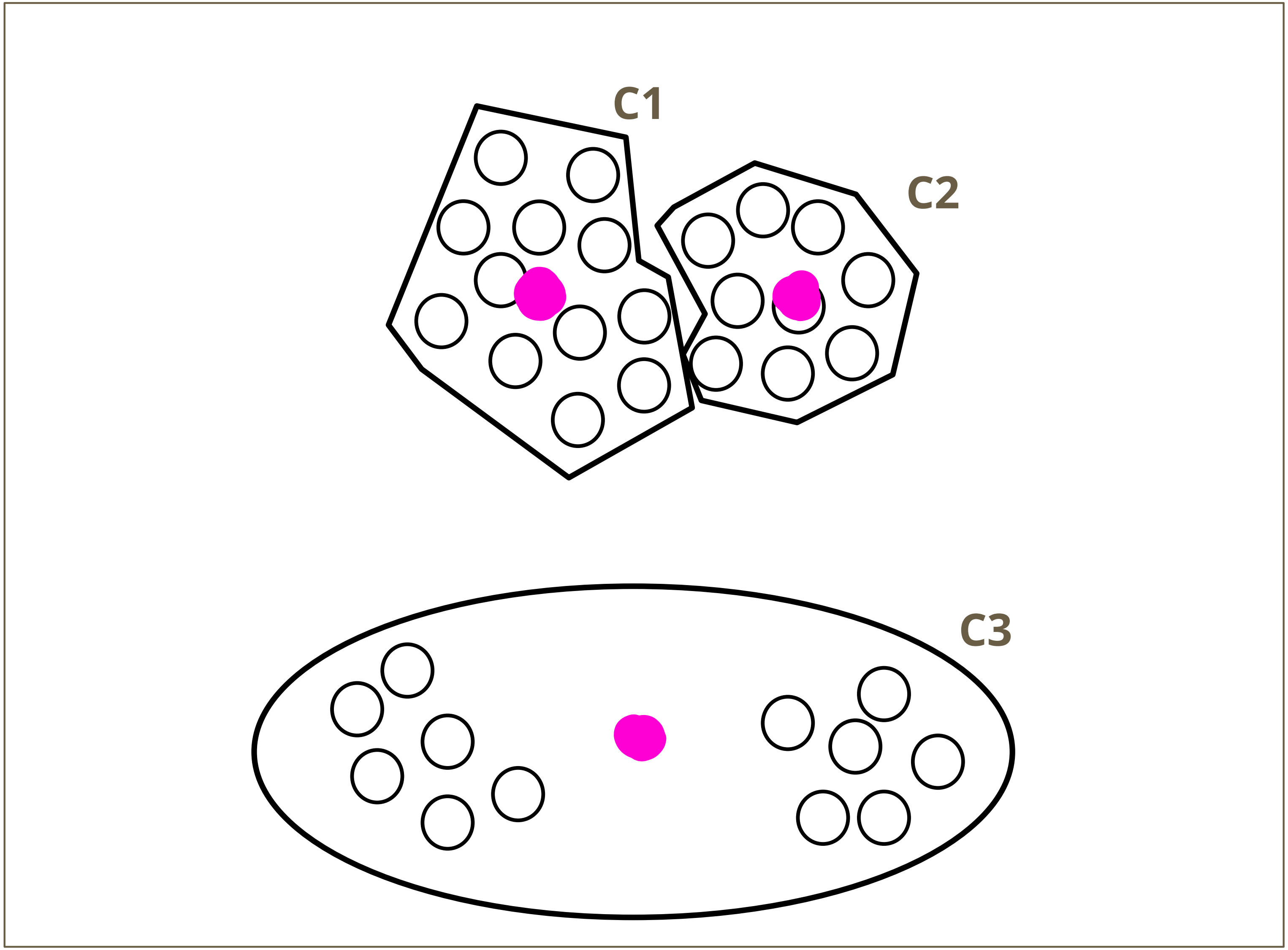
no




these points are closer to the C2 mean

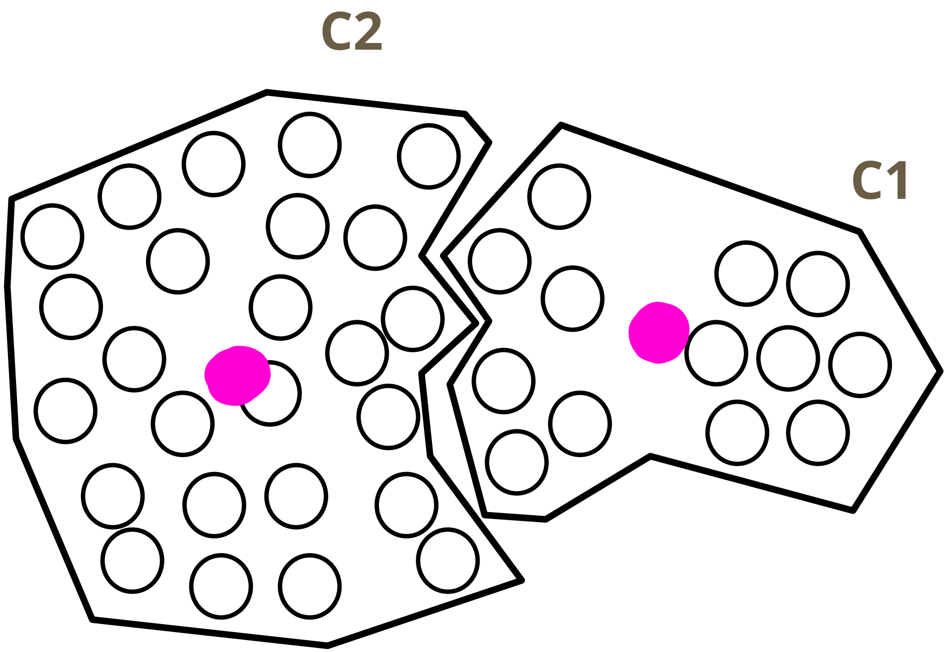
4

yes



S

yes



6

yes