

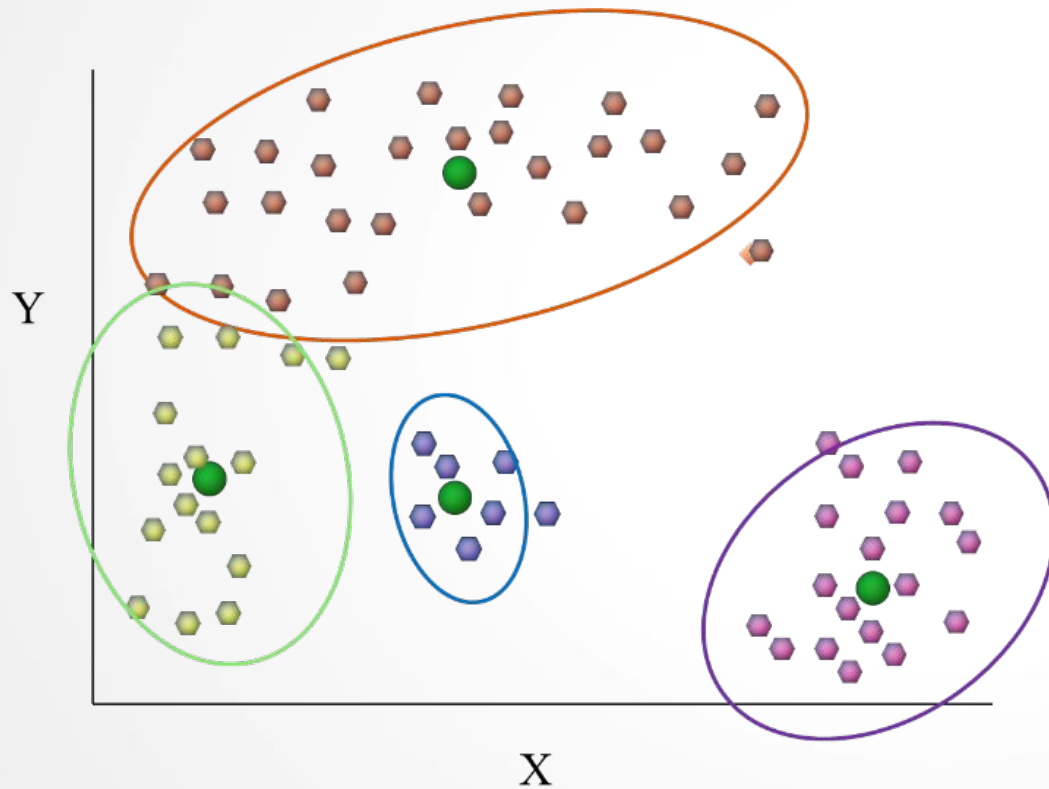
DSC 40A

Theoretical Foundations of Data Science I

Outline

- We'll look at the clustering problem in machine learning and an algorithm that solves this problem.
- Look out for connections to loss functions and risk minimization!

Clustering: Applications



- Bot detection
- Marketing to different subpopulations
- Discovering structure:
 - strains of viruses
 - new species
 - communities in a social network
 - chemicals properties

Clustering: Problem Statement

Given a list of n data points (or vectors) in \mathbb{R}^d

$$x_1, x_2, \dots, x_n$$

and a positive integer, k ,

group the data points into k groups (clusters) of nearby points.

Clustering: Problem Statement

Given a list of n data points (or vectors) in \mathbb{R}^d

$$x_1, x_2, \dots, x_n$$

and a positive integer, k ,

group the data points into k groups (clusters) of nearby points.

Which of these inequalities should be true?

- A. $d < n$
- B. $n < d$
- C. $k < n$
- D. $n < k$

How to define groups?

Pick k cluster centers (centroids),

$$\mu_1, \mu_2, \dots, \mu_k$$

These k centroids define the k groups, by placing each data point in the group corresponding to the nearest centroid.

How to define centroids?

Choose the k cluster centers (centroids) to minimize a cost function.

$\text{Cost}(\mu_1, \mu_2, \dots, \mu_k) =$ total squared distance of each data point x_i
to its nearest centroid μ_j

Lloyds Algorithm, or k-Means Clustering

1. Randomly initialize the k centroids.
2. Keep centroids fixed. Update groups.
Assign each point to the nearest centroid.
3. Keep groups fixed. Update centroids.
Move each centroid to the center of its group.
4. Repeat steps 2 and 3 until done.

Step 1: Randomly initialize the k centroids.

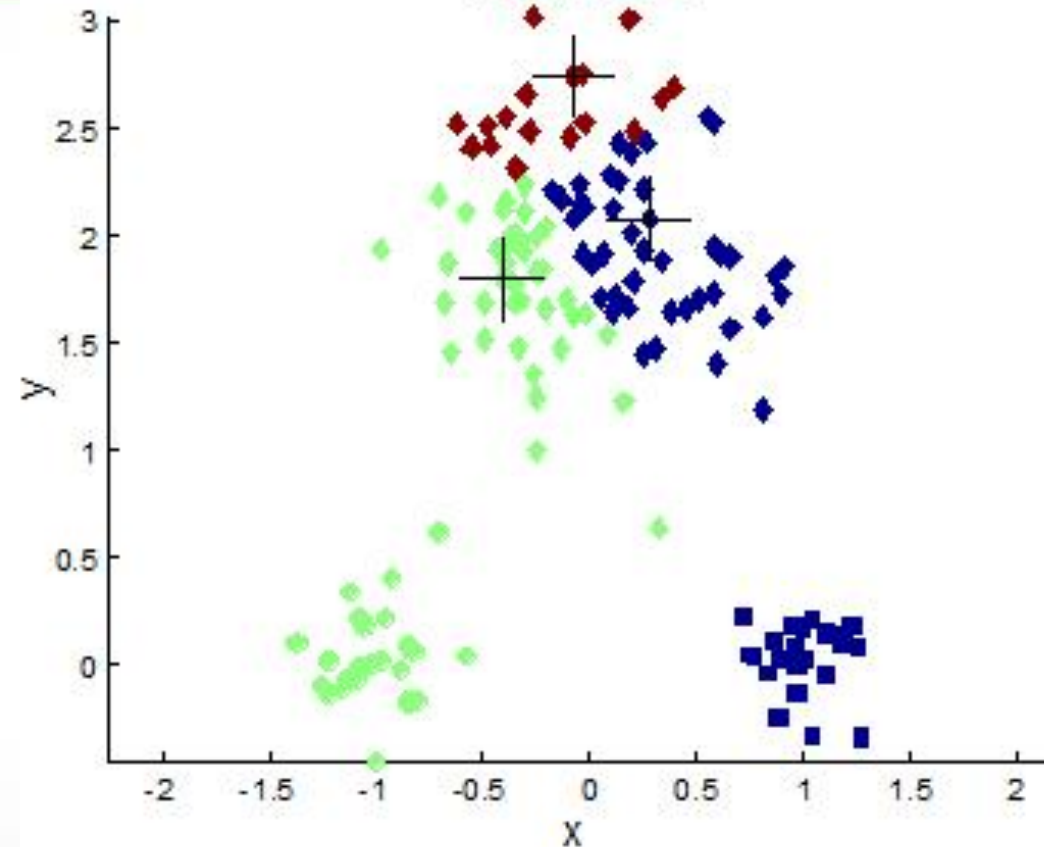
Two common strategies:

- Randomly select k of the data points x_i .
- Randomly assign each data point to one of k groups. Set the centroid of each group to be the center of the points assigned to that group.

Step 2: Keep centroids fixed. Update groups.

For each point,

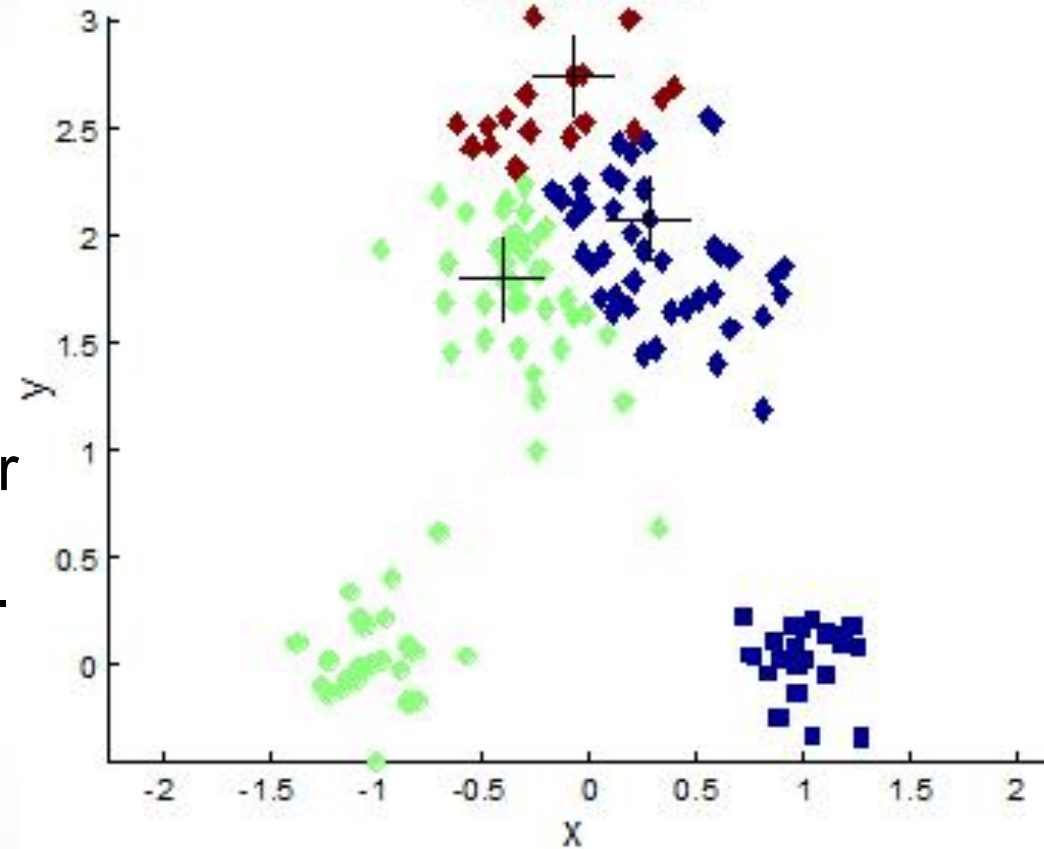
- find the nearest centroid and
- add the point to a group corresponding to that nearest centroid.



Step 3: Keep groups fixed. Update centroids.

For each centroid,

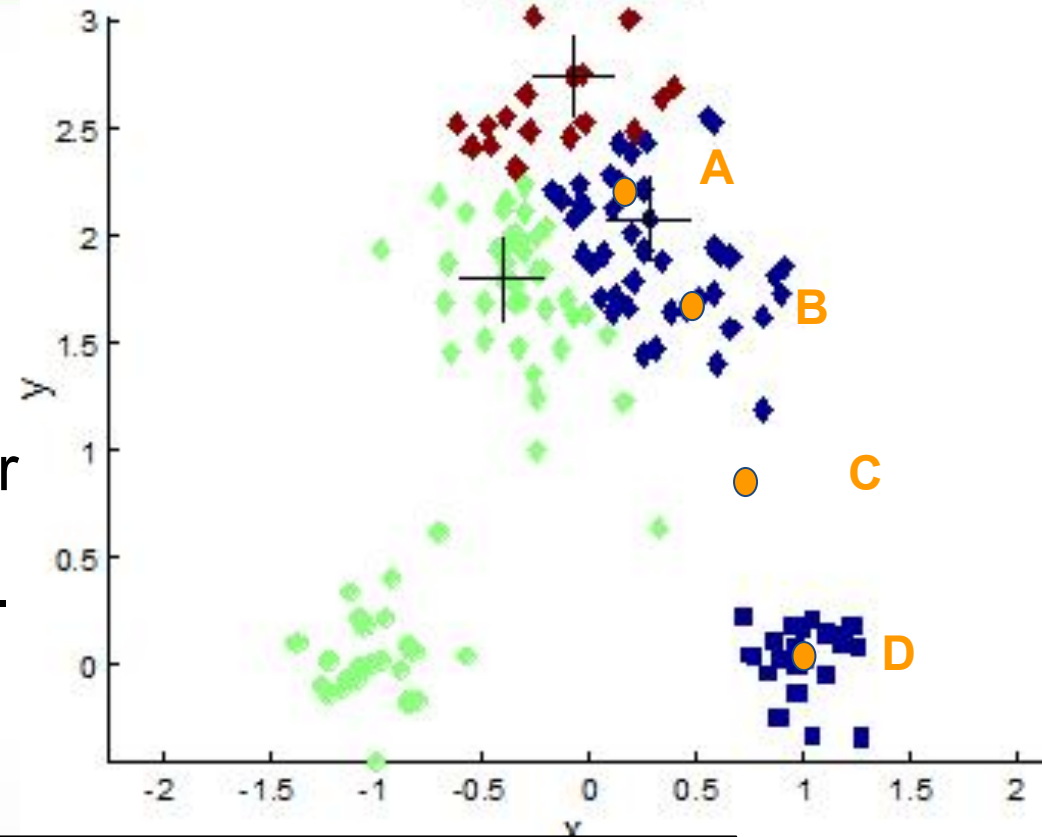
- average the coordinates of all data points in the group, and
- move the centroid to this center point with average coordinates.



Step 3: Keep groups fixed. Update centroids.

For each centroid,

- average the coordinates of all data points in the group, and
- move the centroid to this center point with average coordinates.



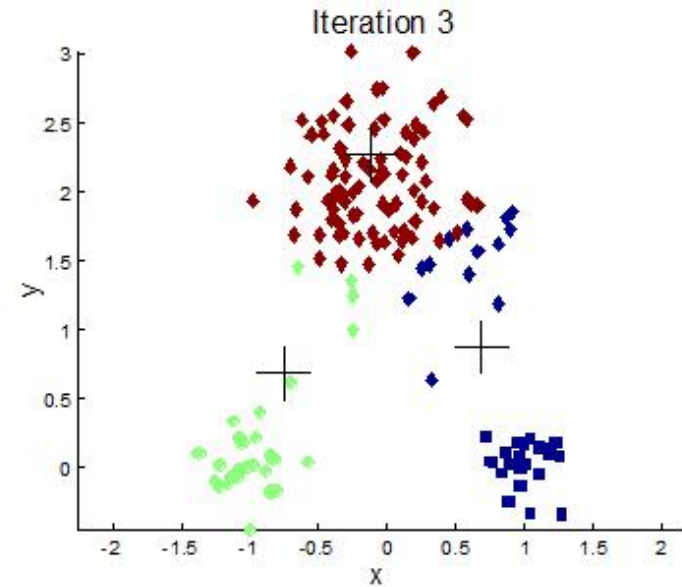
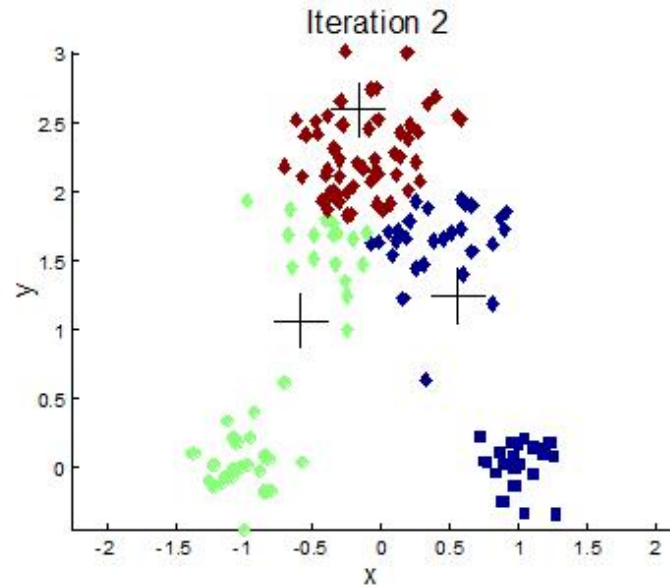
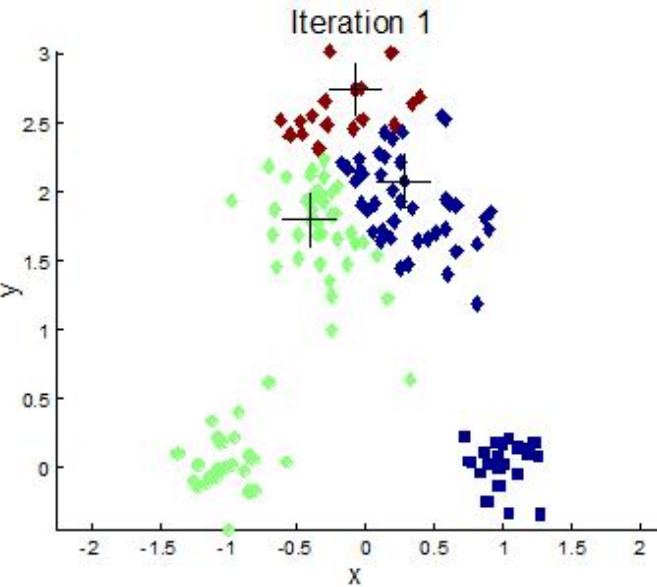
For the blue group of points, approximately where will the centroid move to?

Step 4: Repeat steps 2 and 3 until done.

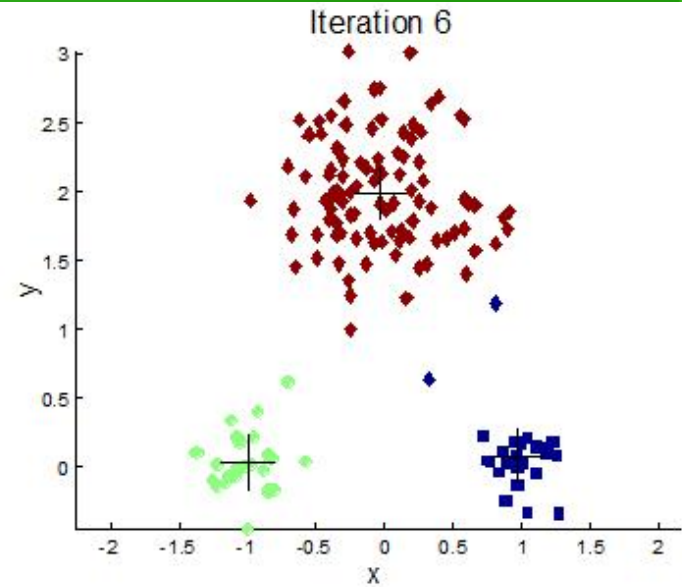
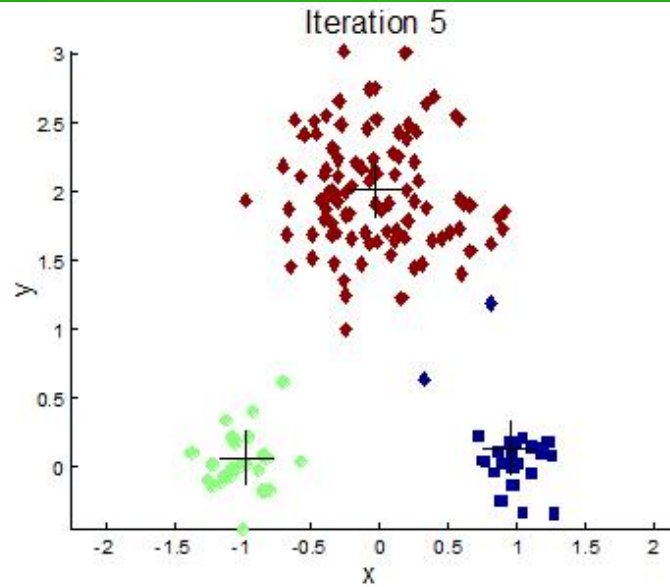
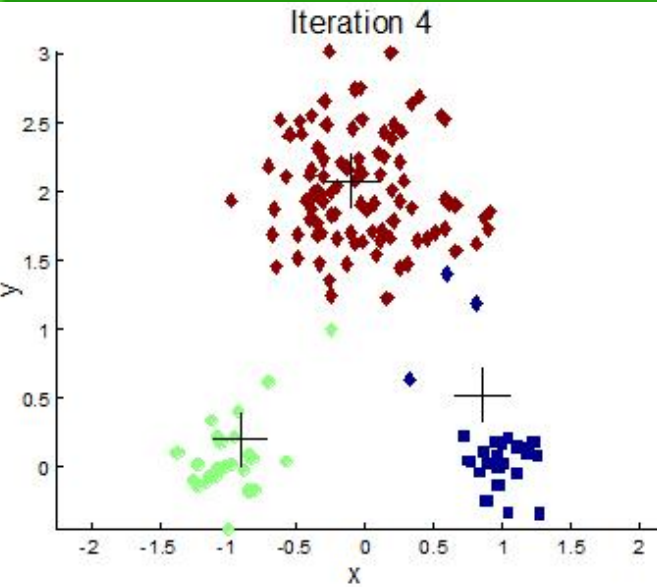
Done when:

- max number of iterations is reached, or
- centroids don't move (at all, or very much), or
- groups don't change (at all, or very much)

k-Means Clustering Example



k-Means Clustering Example



Summary

- We described the clustering problem and the k-means algorithm, which solves this problem.
- **Next time:** We'll see that updating the centroids according to this algorithm reduces the cost with each iteration.

$\text{Cost}(\mu_1, \mu_2, \dots, \mu_k) =$ total squared distance of each data point x_i to its nearest centroid μ_j

Lloyds Algorithm, or k-Means Clustering

1. Randomly initialize the k centroids.
2. Keep centroids fixed. Update groups.
Assign each point to the nearest centroid.
3. Keep groups fixed. Update centroids.
Move each centroid to the center of its group.
4. Repeat steps 2 and 3 until done.

Outline

- Why does k-means clustering work?
- What are some practical considerations when using this algorithm?

Why does k-means clustering work?

$\text{Cost}(\mu_1, \mu_2, \dots, \mu_k) =$ total squared distance of each data point x_i to its nearest centroid μ_j

- Argue why updating the groups and centroids according to the algorithm reduces the cost with each iteration.
- With enough iterations, cost will be sufficiently small.

Why does k-means clustering work?

$\text{Cost}(\mu_1, \mu_2, \dots, \mu_k) =$ total squared distance of each data point x_i to its nearest centroid μ_j


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

Why does k-means clustering work?

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

1. Randomly initialize the k centroids.

sets initial cost
(before the
process begins)



Why does k-means clustering work?

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

2. Fix the centroids. Update the groups.

consider an
arbitrary iteration

Certainly $\text{Cost}(\mu_1, \mu_2, \dots, \mu_k)$ decreases in this step because assigning each point to the **closest** centroid is best.

Why does k-means clustering work?

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

3. Fix the groups. Update the centroids.

consider an
arbitrary iteration

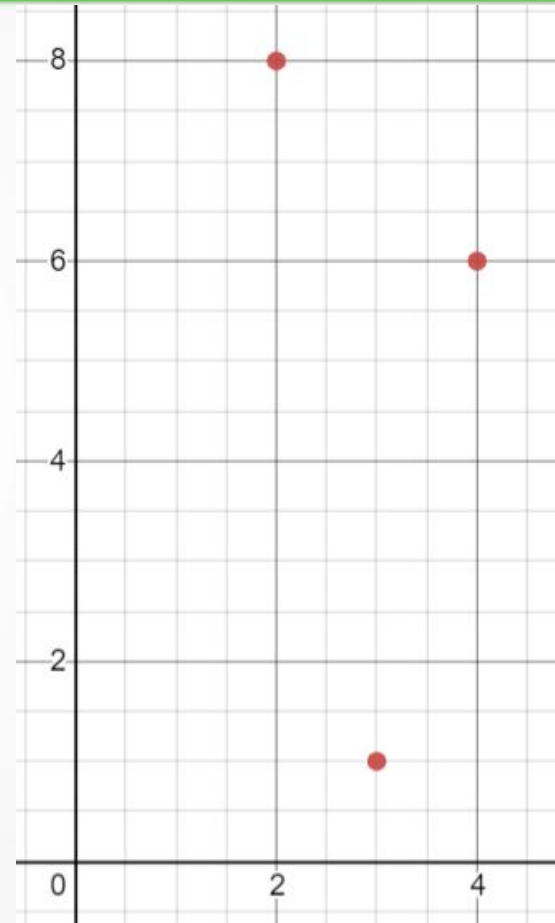
Argue that $\text{Cost}(\mu_1, \mu_2, \dots, \mu_k)$ decreases in this step because for each group j , $\text{Cost}(\mu_j)$ is minimized when we update the centroid.

Why does k-means clustering work?

$\text{Cost}(\mu_j) =$ total squared distance of each data point x_i in group j to centroid μ_j

Example: group j contains $(4, 6)$, $(2, 8)$, $(3, 1)$

How to place centroid $\mu_j = (c_1, c_2)$ to minimize cost?

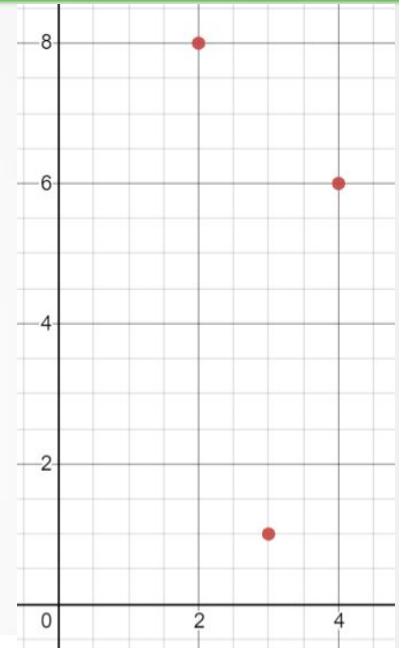


Why does k-means clustering work?

$\text{Cost}(\mu_j) =$ total squared distance of each data point x_i in group j to centroid μ_j

Example: group j contains $(4, 6)$, $(2, 8)$, $(3, 1)$

How to place centroid $\mu_j = (c_1, c_2)$ to minimize cost?



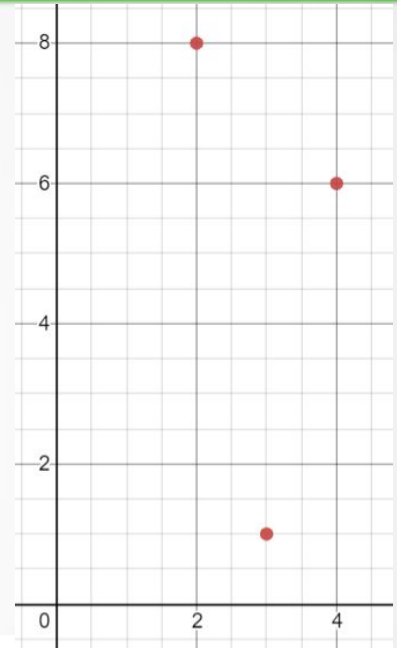
$$\begin{aligned}\text{Cost}(\mu_j) &= \left(\sqrt{(4 - c_1)^2 + (6 - c_2)^2} \right)^2 + \left(\sqrt{(2 - c_1)^2 + (8 - c_2)^2} \right)^2 + \left(\sqrt{(3 - c_1)^2 + (1 - c_2)^2} \right)^2 \\ &= (4 - c_1)^2 + (6 - c_2)^2 + (2 - c_1)^2 + (8 - c_2)^2 + (3 - c_1)^2 + (1 - c_2)^2\end{aligned}$$

Why does k-means clustering work?

$\text{Cost}(\mu_j) =$ total squared distance of each data point x_i in group j to centroid μ_j

Example: group j contains $(4, 6)$, $(2, 8)$, $(3, 1)$

How to place centroid $\mu_j = (c_1, c_2)$ to minimize cost?



$$\begin{aligned}\text{Cost}(\mu_j) &= \left(\sqrt{(4 - c_1)^2 + (6 - c_2)^2} \right)^2 + \left(\sqrt{(2 - c_1)^2 + (8 - c_2)^2} \right)^2 + \left(\sqrt{(3 - c_1)^2 + (1 - c_2)^2} \right)^2 \\ &= (4 - c_1)^2 + (6 - c_2)^2 + (2 - c_1)^2 + (8 - c_2)^2 + (3 - c_1)^2 + (1 - c_2)^2\end{aligned}$$

$$\frac{\partial \text{Cost}(\mu_j)}{\partial c_1} = 2(c_1 - 4) + 2(c_1 - 2) + 2(c_1 - 3)$$

$$\frac{\partial \text{Cost}(\mu_j)}{\partial c_2} = 2(c_2 - 6) + 2(c_2 - 8) + 2(c_2 - 1)$$

Why does k-means clustering work?

$\text{Cost}(\mu_j) =$ total squared distance of each data point x_i in group j to centroid μ_j

Example: group j contains $(4, 6)$, $(2, 8)$, $(3, 1)$

How to place centroid $\mu_j = (c_1, c_2)$ to minimize cost?

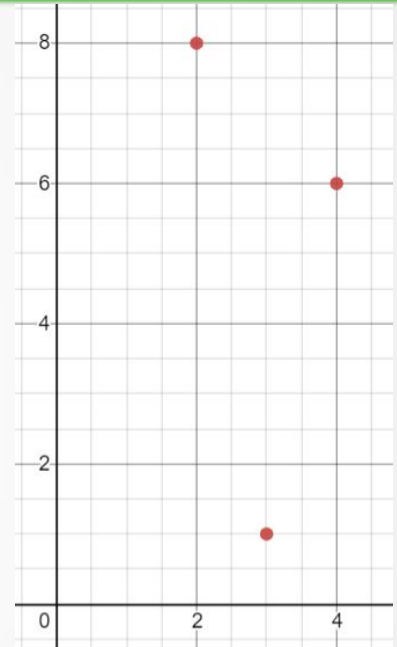
$$\frac{\partial \text{Cost}(\mu_j)}{\partial c_1} = 2(c_1 - 4) + 2(c_1 - 2) + 2(c_1 - 3)$$

$$0 = 2(c_1 - 4) + 2(c_1 - 2) + 2(c_1 - 3)$$

$$0 = c_1 - 4 + c_1 - 2 + c_1 - 3$$

$$3c_1 = 4 + 2 + 3$$

$$c_1 = \frac{4 + 2 + 3}{3}$$



Why does k-means clustering work?

$\text{Cost}(\mu_j) =$ total squared distance of each data point x_i in group j to centroid μ_j

Example: group j contains $(4, 6)$, $(2, 8)$, $(3, 1)$

How to place centroid $\mu_j = (c_1, c_2)$ to minimize cost?

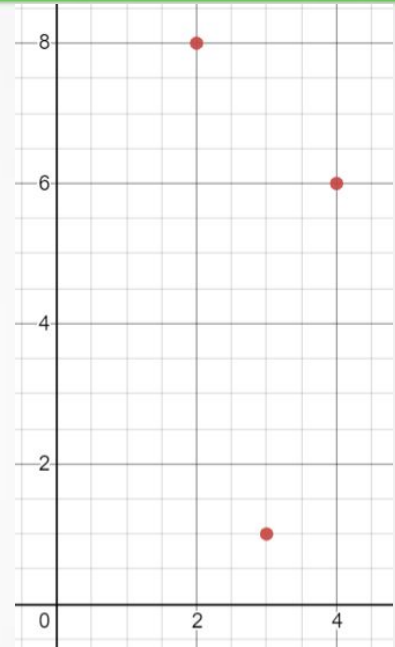
$$\frac{\partial \text{Cost}(\mu_j)}{\partial c_2} = 2(c_2 - 6) + 2(c_2 - 8) + 2(c_2 - 1)$$

$$0 = 2(c_2 - 6) + 2(c_2 - 8) + 2(c_2 - 1)$$

$$0 = c_2 - 6 + c_2 - 8 + c_2 - 1$$

$$3c_2 = 6 + 8 + 1$$

$$c_2 = \frac{6 + 8 + 1}{3}$$



Why does k-means clustering work?

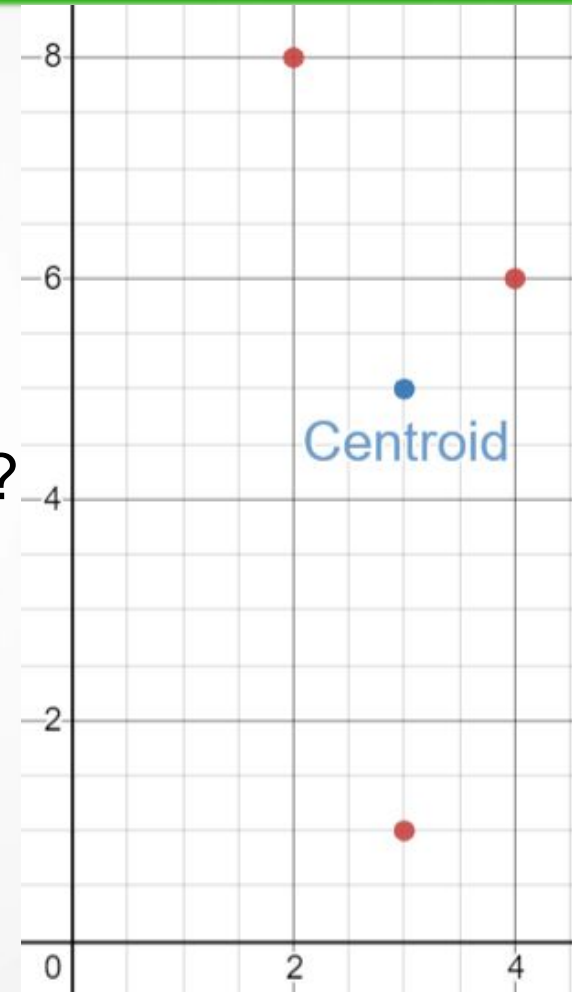
$\text{Cost}(\mu_j) =$ total squared distance of each data point x_i in group j to centroid μ_j

Example: group j contains $(4, 6)$, $(2, 8)$, $(3, 1)$

How to place centroid $\mu_j = (c_1, c_2)$ to minimize cost?

$$(c_1, c_2) = \left(\frac{4+2+3}{3}, \frac{6+8+1}{3} \right) = (3, 5)$$

Minimize cost by averaging in each coordinate.



Cost, Loss, and Risk

The cost of placing the centroid at (c_1, c_2) is

$$\begin{aligned}\text{Cost}(\mu_j) &= \left(\sqrt{(4 - c_1)^2 + (6 - c_2)^2} \right)^2 + \left(\sqrt{(2 - c_1)^2 + (8 - c_2)^2} \right)^2 + \left(\sqrt{(3 - c_1)^2 + (1 - c_2)^2} \right)^2 \\ &= (4 - c_1)^2 + (6 - c_2)^2 + (2 - c_1)^2 + (8 - c_2)^2 + (3 - c_1)^2 + (1 - c_2)^2\end{aligned}$$

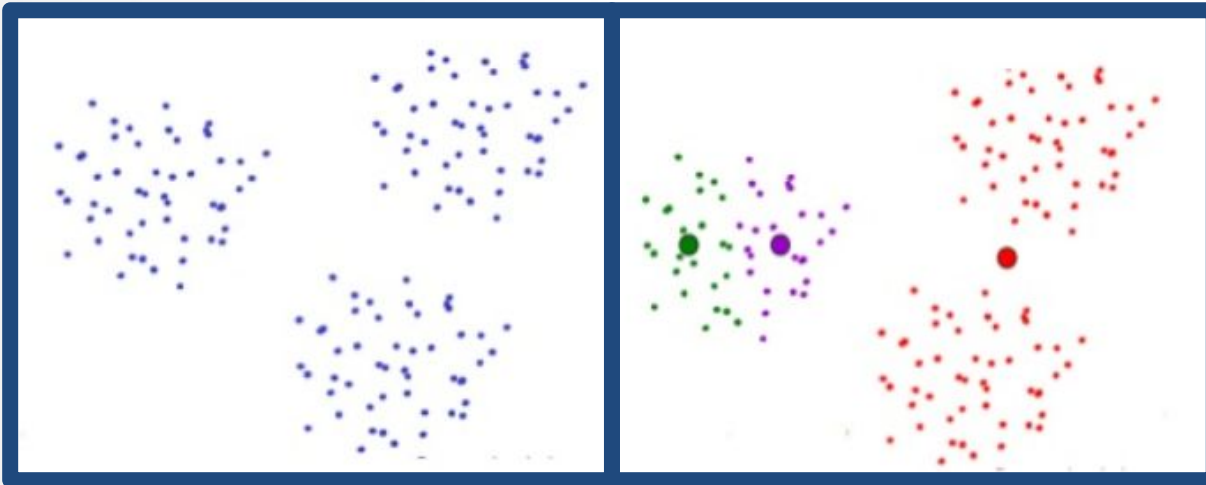
Why does k-means clustering work?

$\text{Cost}(\mu_1, \mu_2, \dots, \mu_k) =$ total squared distance of each data point x_i to its nearest centroid μ_j

- Argue why updating the groups and centroids according to the algorithm reduces the cost with each iteration.
- With enough iterations, cost will be sufficiently small.

k-Means Clustering in Practice: Initialization

Can get unlucky with random initialization.



In general, how do we assess which result is the best?

- A. Clusters appear how we expect them to
- B. Clusters are evenly sized
- C. Cost function is lowest

k-Means Clustering in Practice: Initialization

Can get unlucky with random initialization.



In general, how do we assess which result is the best?

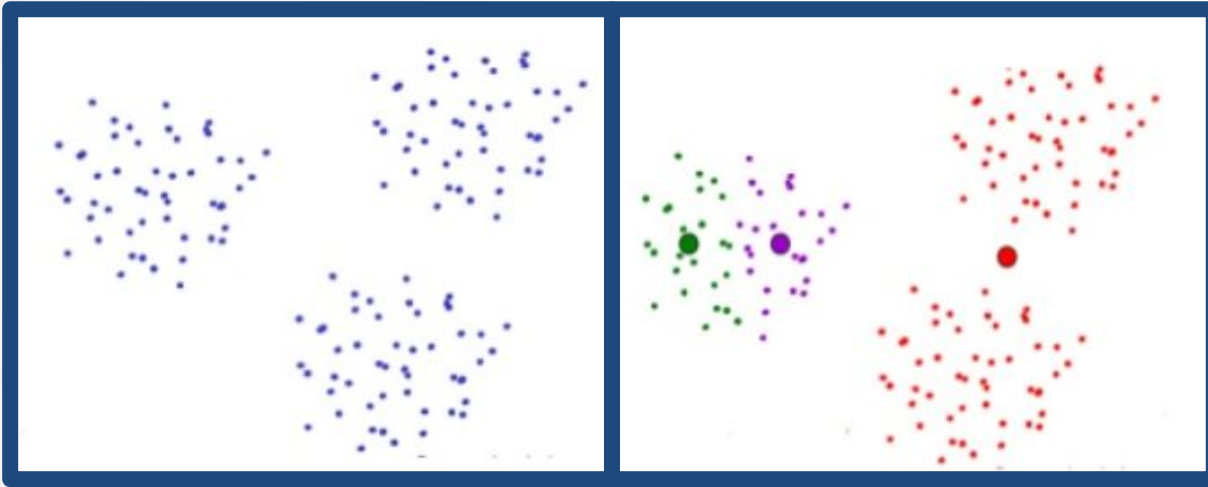
- A. Clusters appear how we expect them to
- B. Clusters are evenly sized
- C. Cost function is lowest

Solution?

- Try algorithm several times, pick the best result.
- Similar approach used in gradient descent.

k-Means Clustering in Practice: Initialization

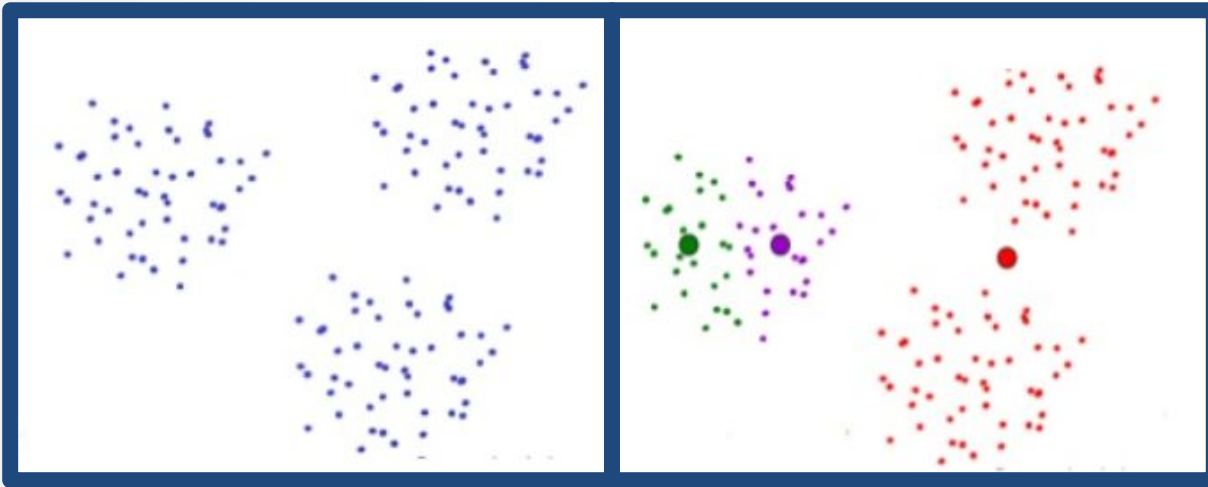
Can get unlucky with random initialization.



- No guarantees of a satisfactory solution with this algorithm.
- Brute force algorithm would try all assignments of points to clusters and choose the one with the lowest cost.

k-Means Clustering in Practice: Initialization

Can get unlucky with random initialization.

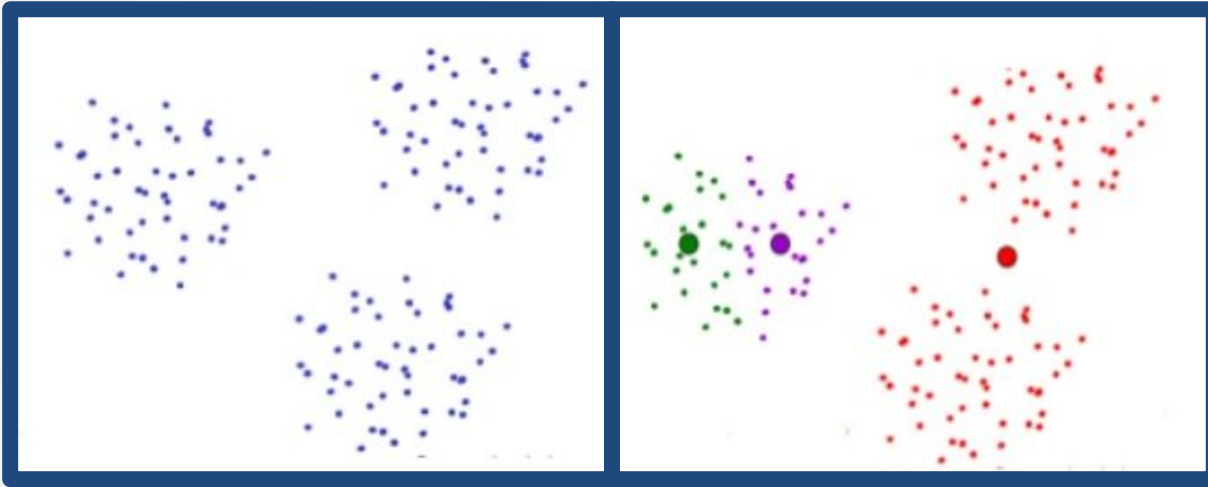


How many ways to assign n points to k clusters?

- No guarantees of a satisfactory solution with this algorithm.
- Brute force algorithm would try all assignments of points to clusters and choose the one with the lowest cost.

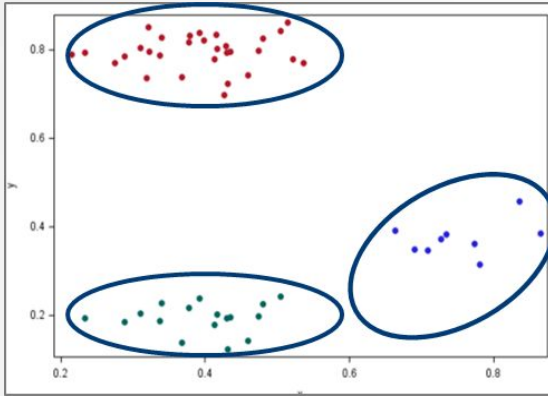
k-Means Clustering in Practice: Initialization

Can get unlucky with random initialization.

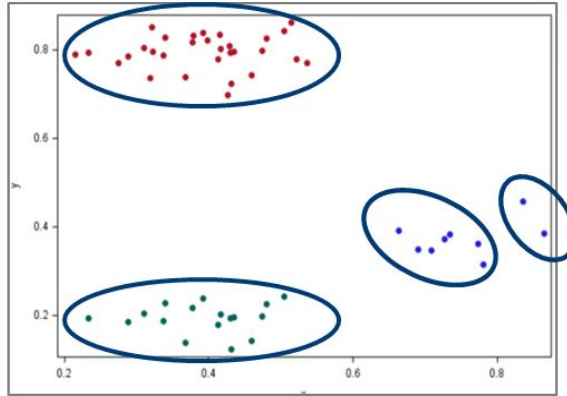


- No guarantees of a satisfactory solution with this algorithm.
- Any algorithm that is guaranteed to find the best coloring of data points takes exponential time (computationally infeasible).

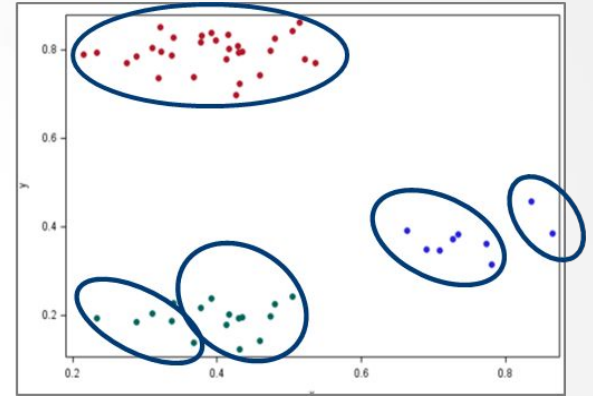
k-Means Clustering in Practice: Choosing k



k=3

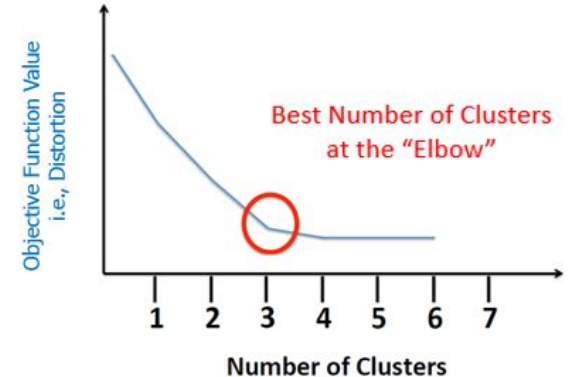


k=4



k=5

- Most commonly done by hand (visualizations, trial and error)
- Elbow method
- Context or domain knowledge
- Use a different clustering algorithm



What if a centroid has no points in its group?

What should we do if a centroid has no points in its group?

- A. Terminate the algorithm.
- B. Wait for points get added to the group in a subsequent iteration.
- C. Set the centroid to be a data point, chosen at random.
- D. Set the centroid to be one of the other centroids, chosen at random.

What if a centroid has no points in its group?

What should we do if a centroid has no points in its group?

- A. Terminate the algorithm.
- B. Wait for points get added to the group in a subsequent iteration.
- C. Set the centroid to be a data point, chosen at random.
- D. Set the centroid to be one of the other centroids, chosen at random.

Two options:

- Eliminate that centroid and find $k-1$ clusters instead
- Randomly re-initialize that centroid

Summary

- We saw that k-means clustering works because each step of the algorithm reduces the cost function, which measures the quality of a set of centroids.
- We discussed some practical considerations, including random initialization and choice of k .