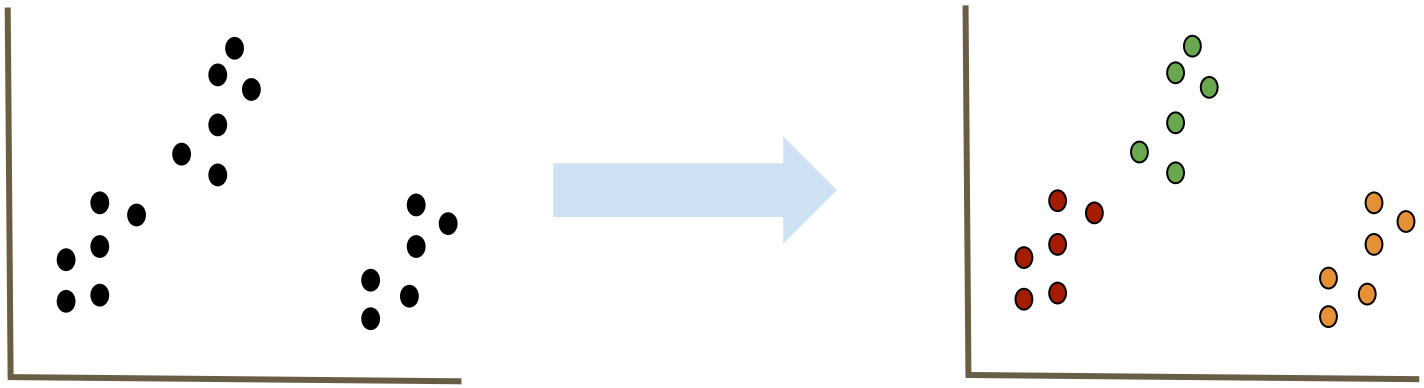
# Clustering - Kmeans

Boston University CS 506 - Lance Galletti

# 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.
- 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.)

# The Clustering Problem

Given a collection of data points

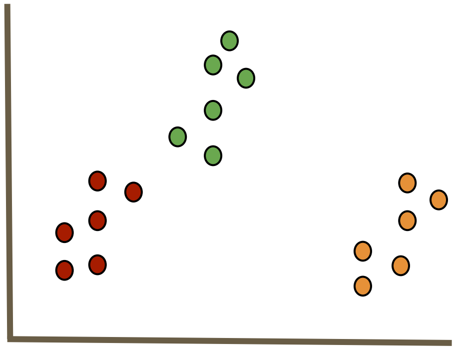
Find a clustering such that:

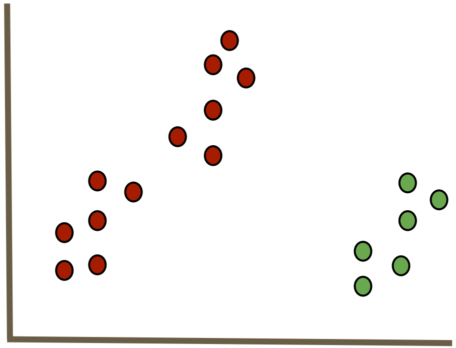
- **Similar** data points are in the **same cluster**
- **Dissimilar** data points are in **different clusters**

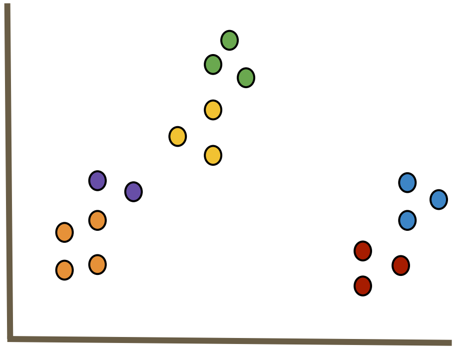
Questions:

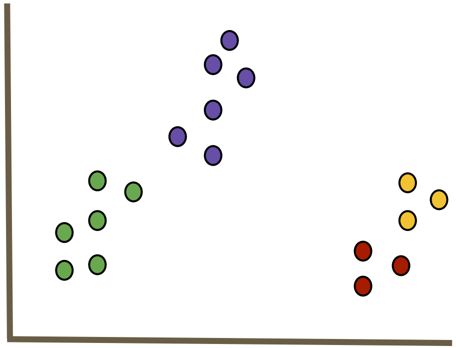
- What does **similar** mean?
- How do we find a **clustering**?
- How do we know if we have found a **good clustering**?

# 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

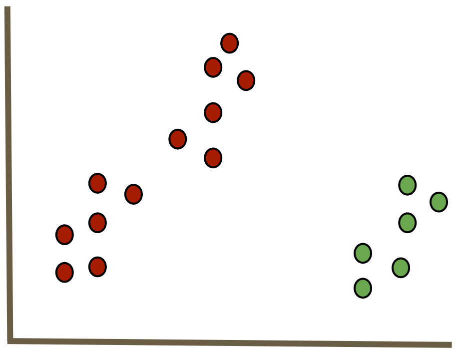
**Density-Based**
Defined based on the local density of points

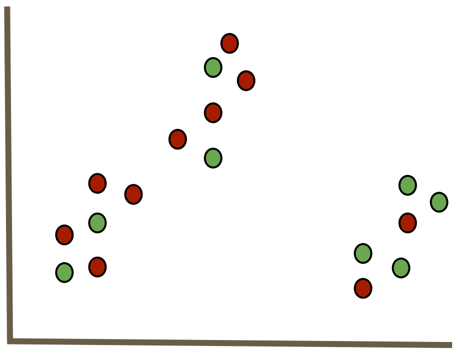
**Soft Clustering**
Each point is assigned to every cluster with a certain probability

# Partitional Clustering

# Partitional Clustering

Given **n** data points and a number **k** of clusters: partition the **n** data points into **k** clusters.

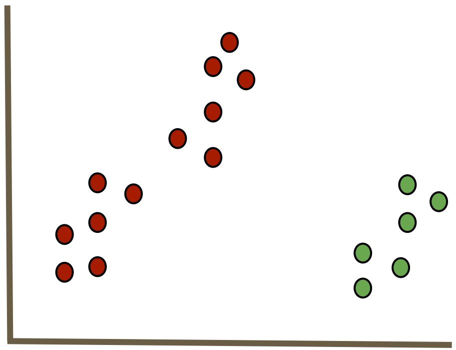




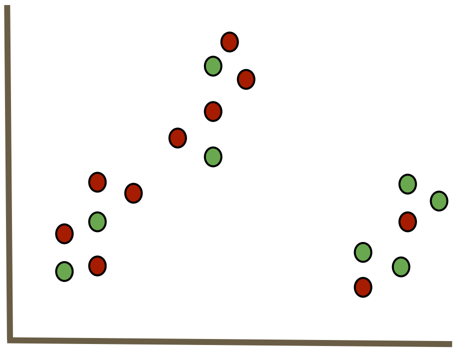
# Partitional Clustering

Suppose we are given all possible ways of distributing these **n** data points into these **k** buckets / clusters. How would we find the best such partition?

# Example



VS

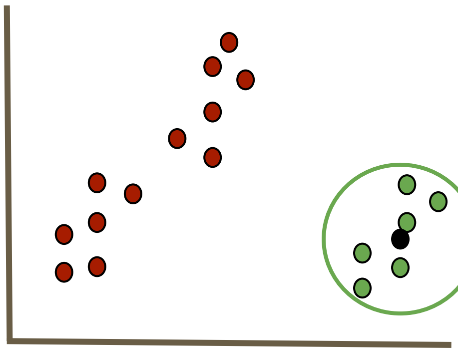


Clearly the clustering on the left has smaller intra-cluster distances than the one on the right. That is:

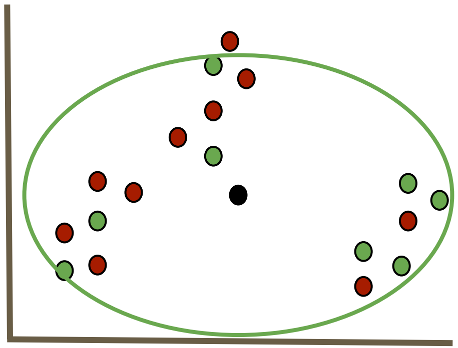
$$\sum_{k}^{K} \sum_{x_i, x_j \in C_k} d(x_i, x_j)$$

Is a smaller quantity

# Example



VS



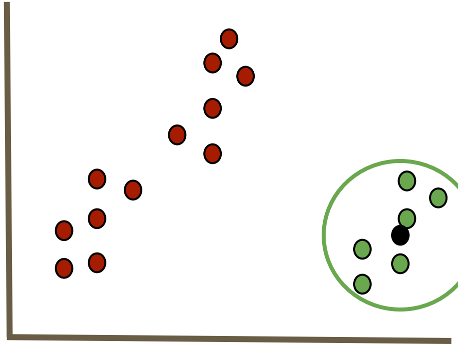
Given a distance function **d**, we can find points (not necessarily part of our dataset) for each cluster called **centroids** that are at the center of each cluster.

# Example

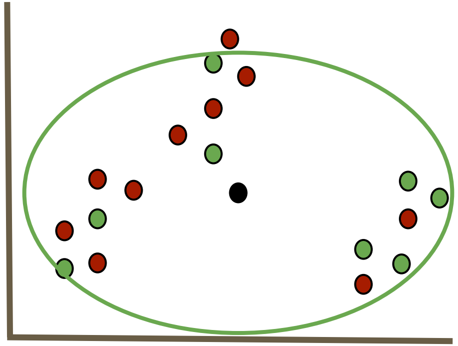
Q: When **d** is Euclidean, what is the **centroid** (also called **center of mass**) of **m** points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ ?

A: The mean / average of the points

# Example



VS



Turns out when **d** is Euclidean:

$$\sum_{k}^{K} \sum_{x_i, x_j \in C_k} d(x_i, x_j)^2 = \sum_{k}^{K} |C_k| \sum_{x_i \in C_k} d(x_i, \mu_k)^2$$

# K-means

Given $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ our dataset and **k**

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

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

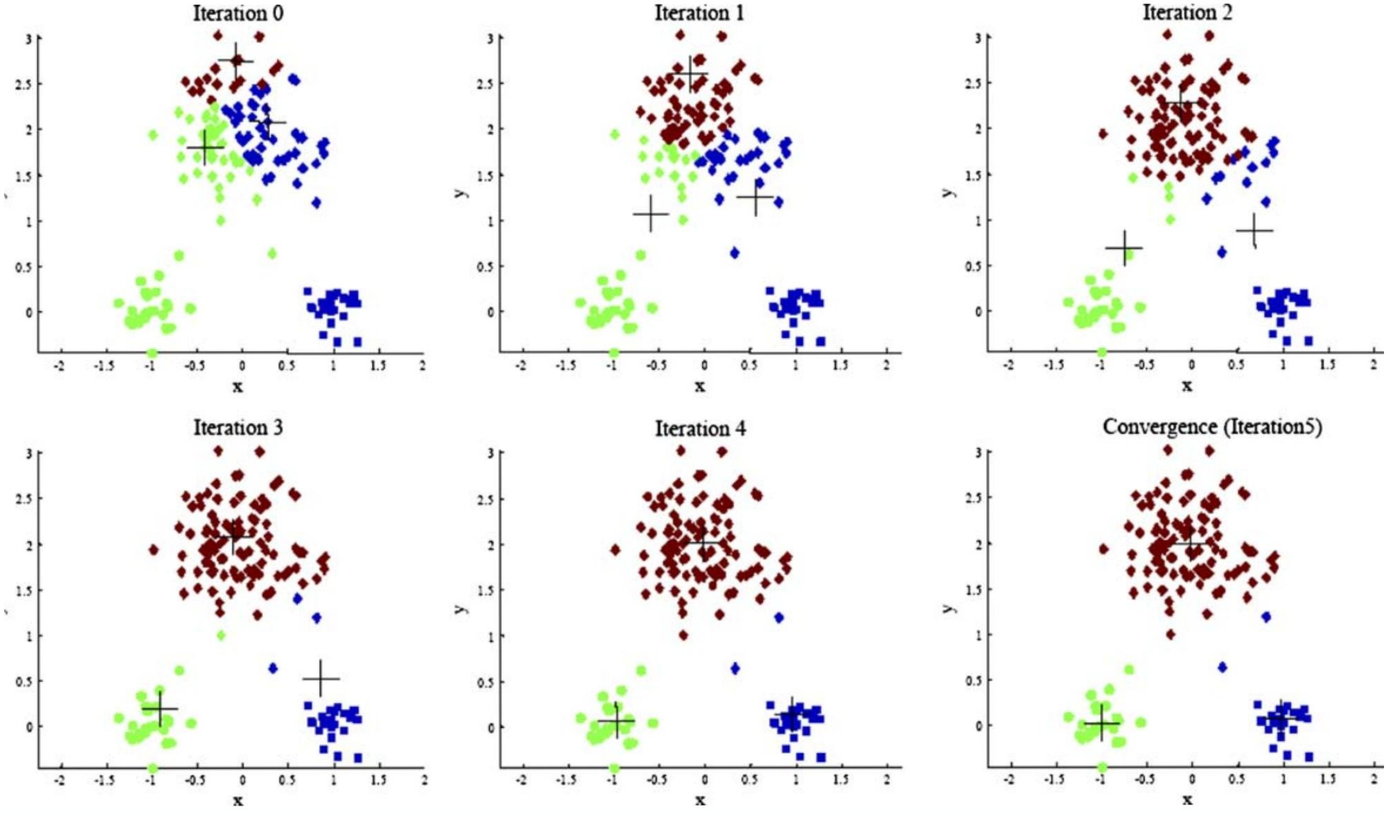
When **k=1** and **k=n** this is easy. Why?

When $\mathbf{x}_i$ lives in more than 2 dimensions, this is a very difficult (**NP-hard**) problem

# K-means - Lloyd's Algorithm

1. Randomly pick **k** centers $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$
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

# K-means - Lloyd's Algorithm



# K-means - Lloyd's Algorithm

Will this algorithm always converge?

**Proof** (by contradiction): Suppose it does not converge. Then, either:

1. The minimum of the cost function is only reached in the limit (i.e. after an infinite number of iterations).
   **Impossible** because we are iterating over a finite set of partitions

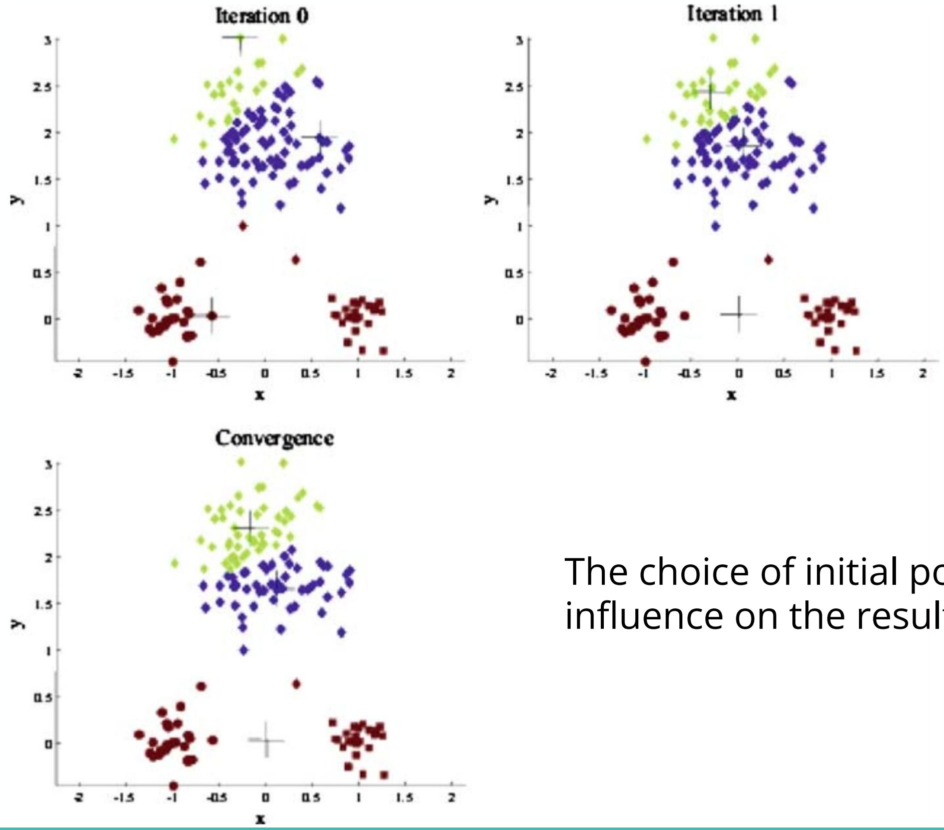
1. The algorithm gets stuck in a cycle / loop
   **Impossible** since this would require having a clustering that has a lower cost than itself and we know:
    - If old ≠ new clustering then the cost has improved
    - If old = new clustering then the cost is unchanged

**Conclusion**: Lloyd's Algorithm always converges!

# K-means - Lloyd's Algorithm

Will this always converge to the optimal solution?

# K-means - Lloyd’s Algorithm




The choice of initial points has a large influence on the resulting clustering

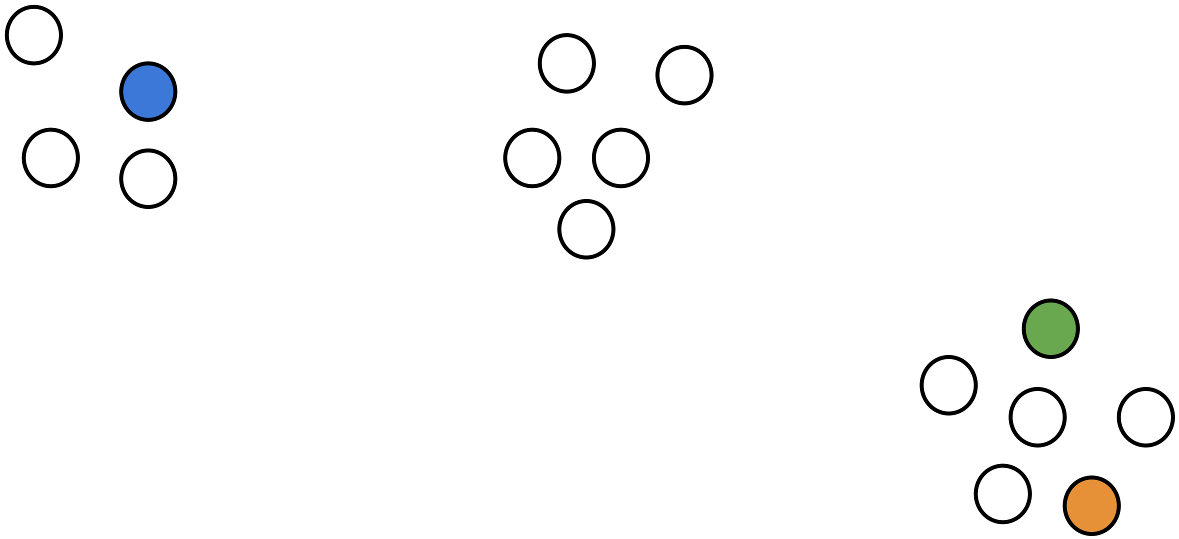
# K-means - Initialization

One solution: Run Lloyd’s algorithm multiple times and choose the result with the lowest cost.

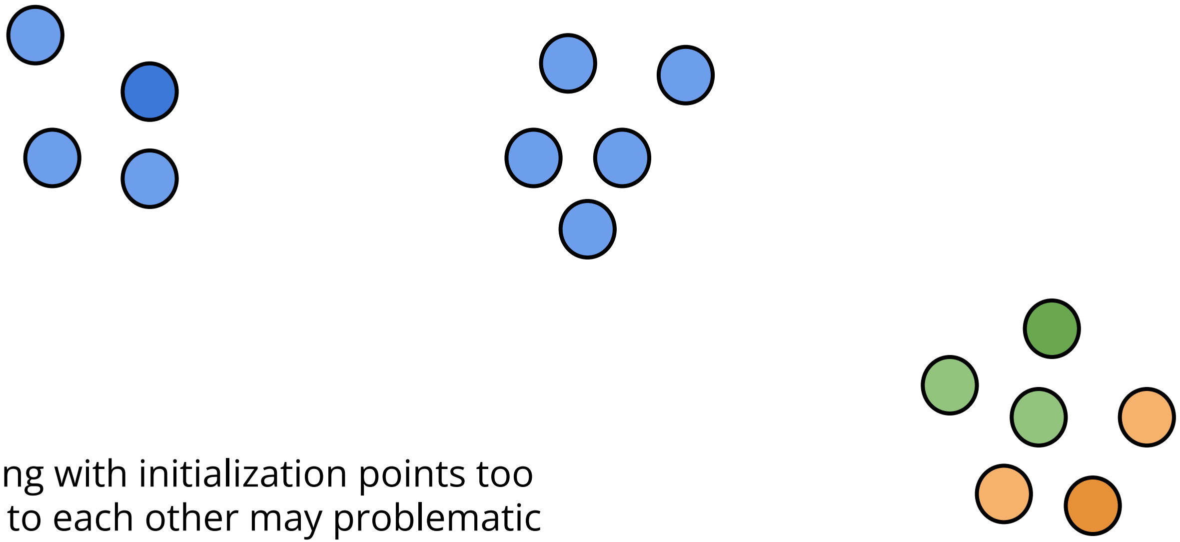
This can still lead to bad results because of randomness.

Another solution: Try different initialization methods

# K-means - Random

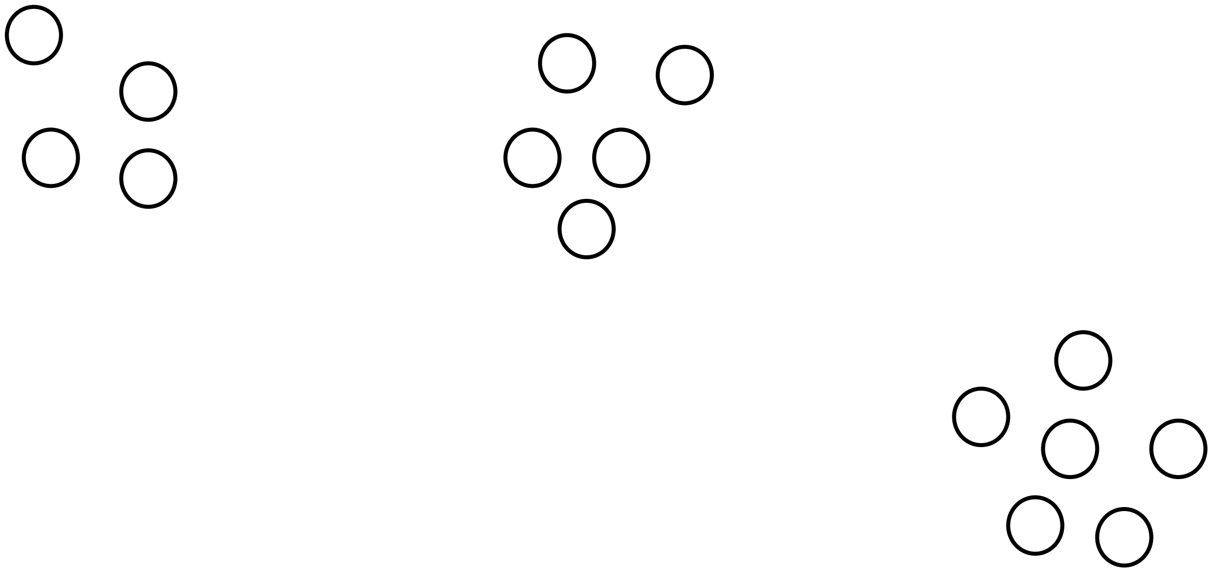


# K-means - Random

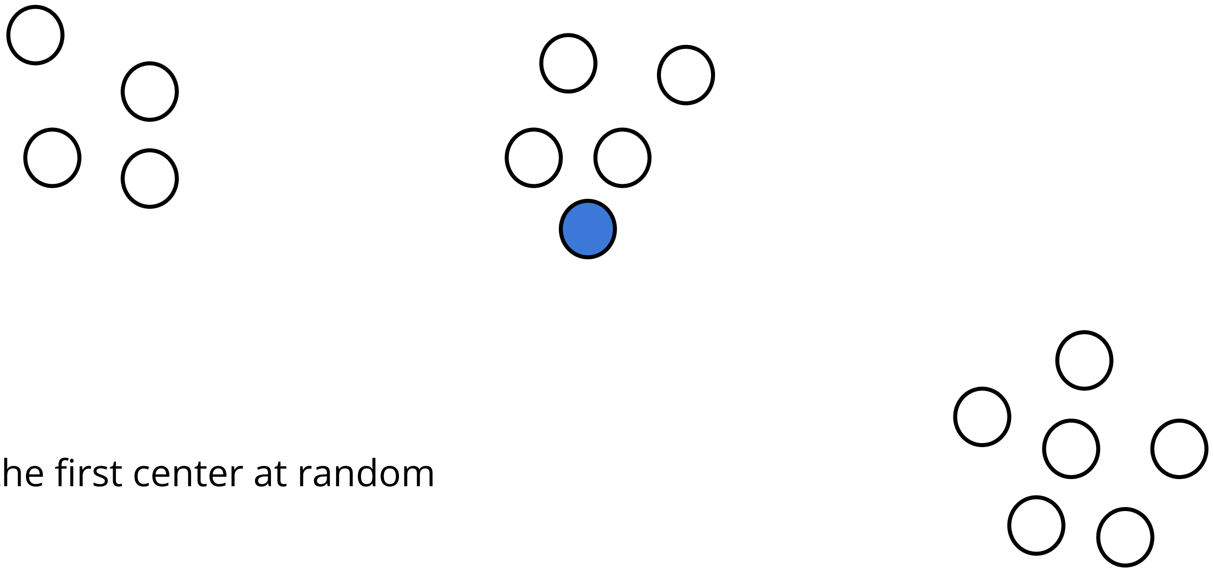


Starting with initialization points too
close to each other may problematic

# K-means - Farthest First Traversal

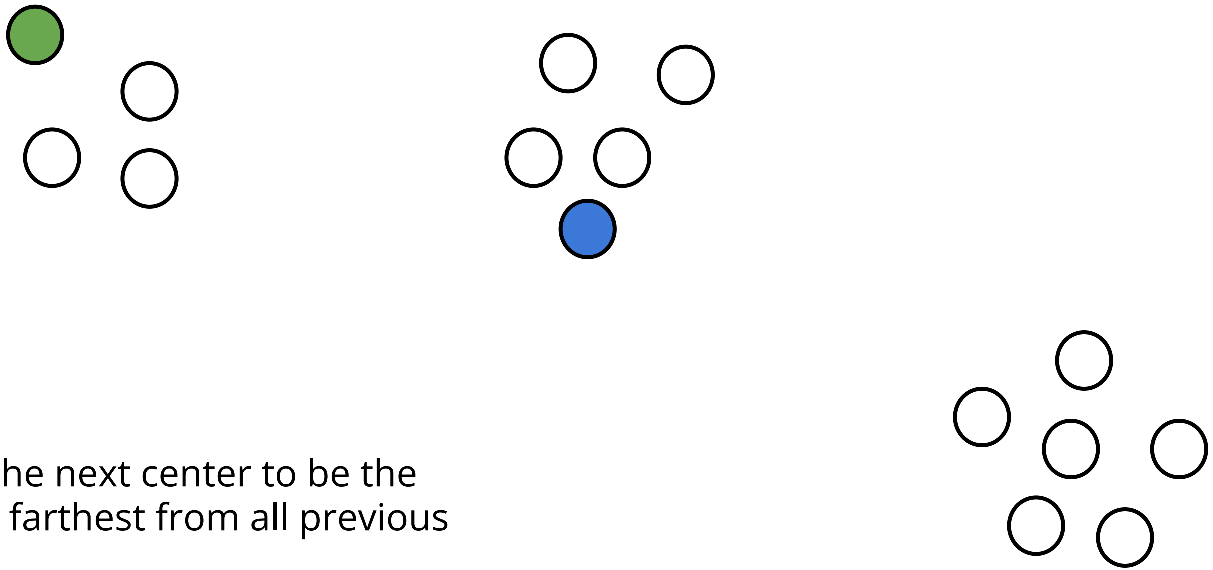


# K-means - Farthest First Traversal



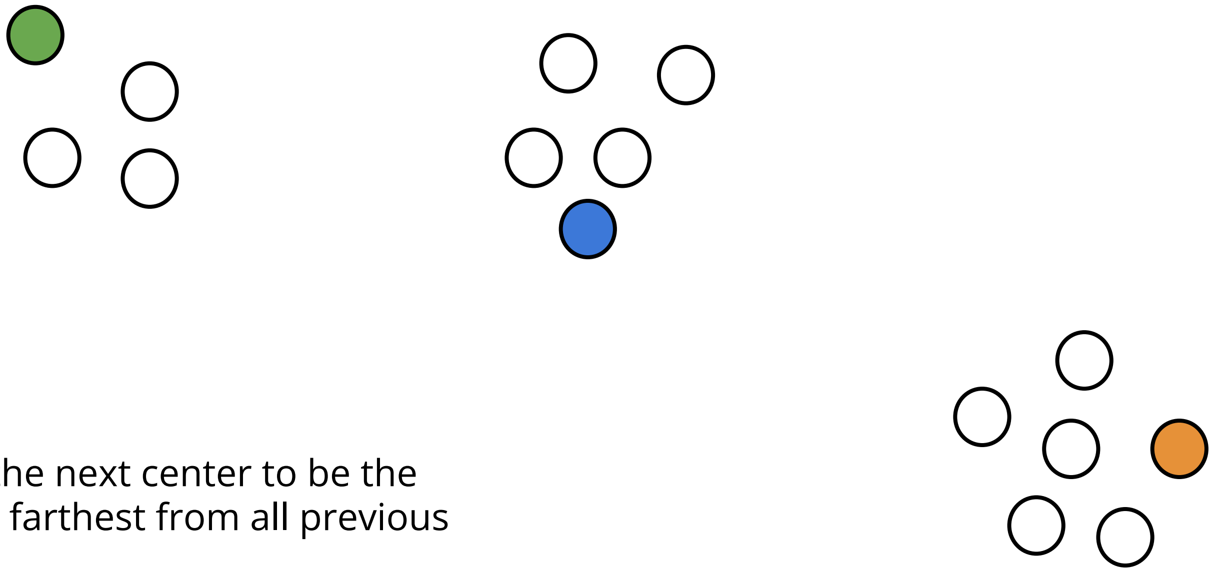
Pick the first center at random

# K-means - Farthest First Traversal



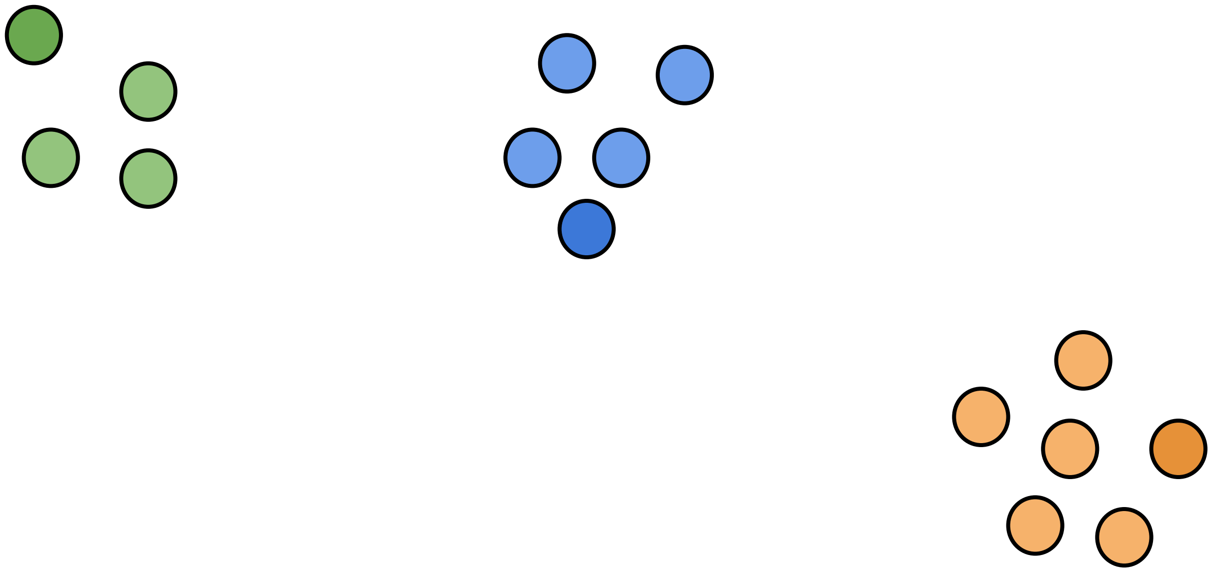
Pick the next center to be the
point farthest from all previous

# K-means - Farthest First Traversal

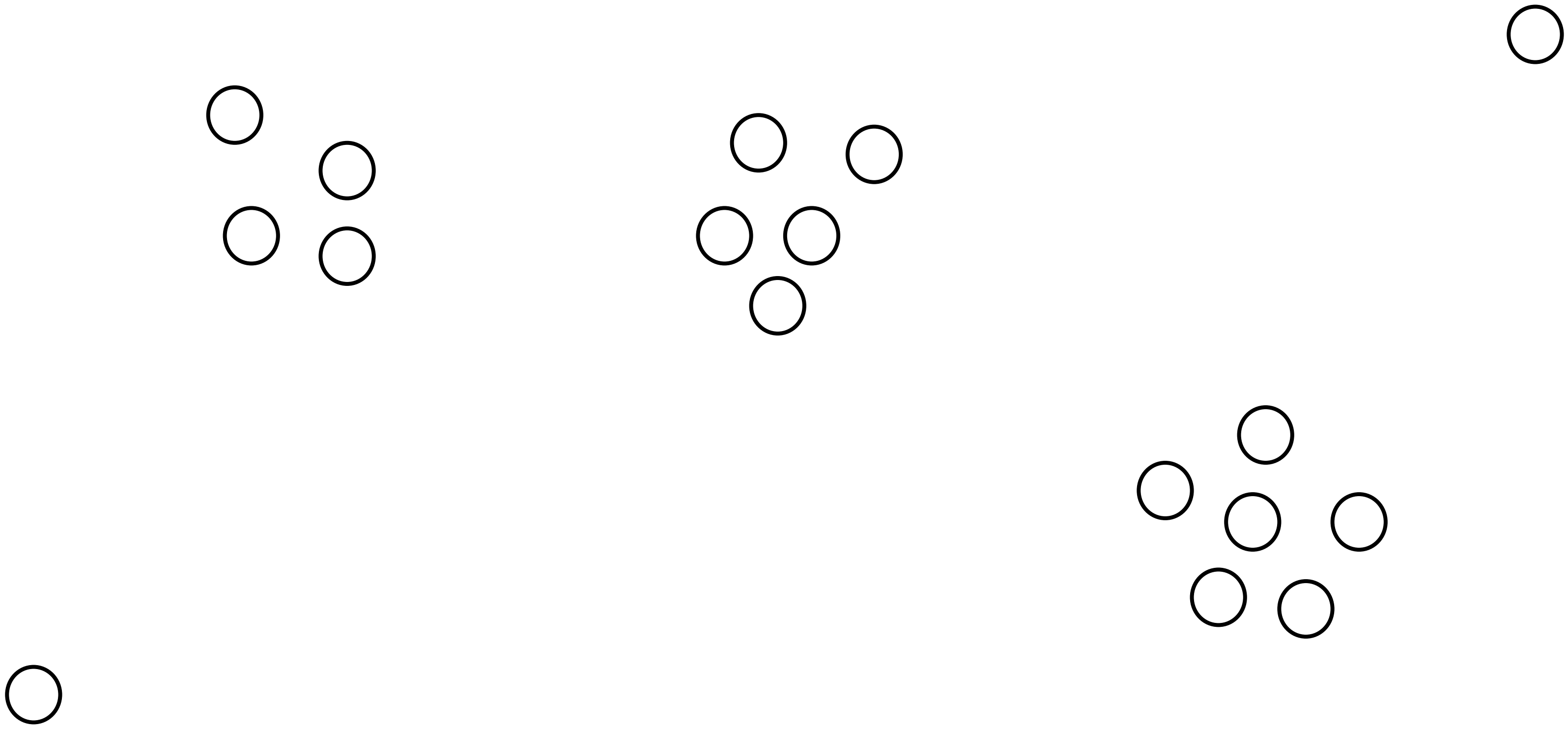


Pick the next center to be the point farthest from all previous

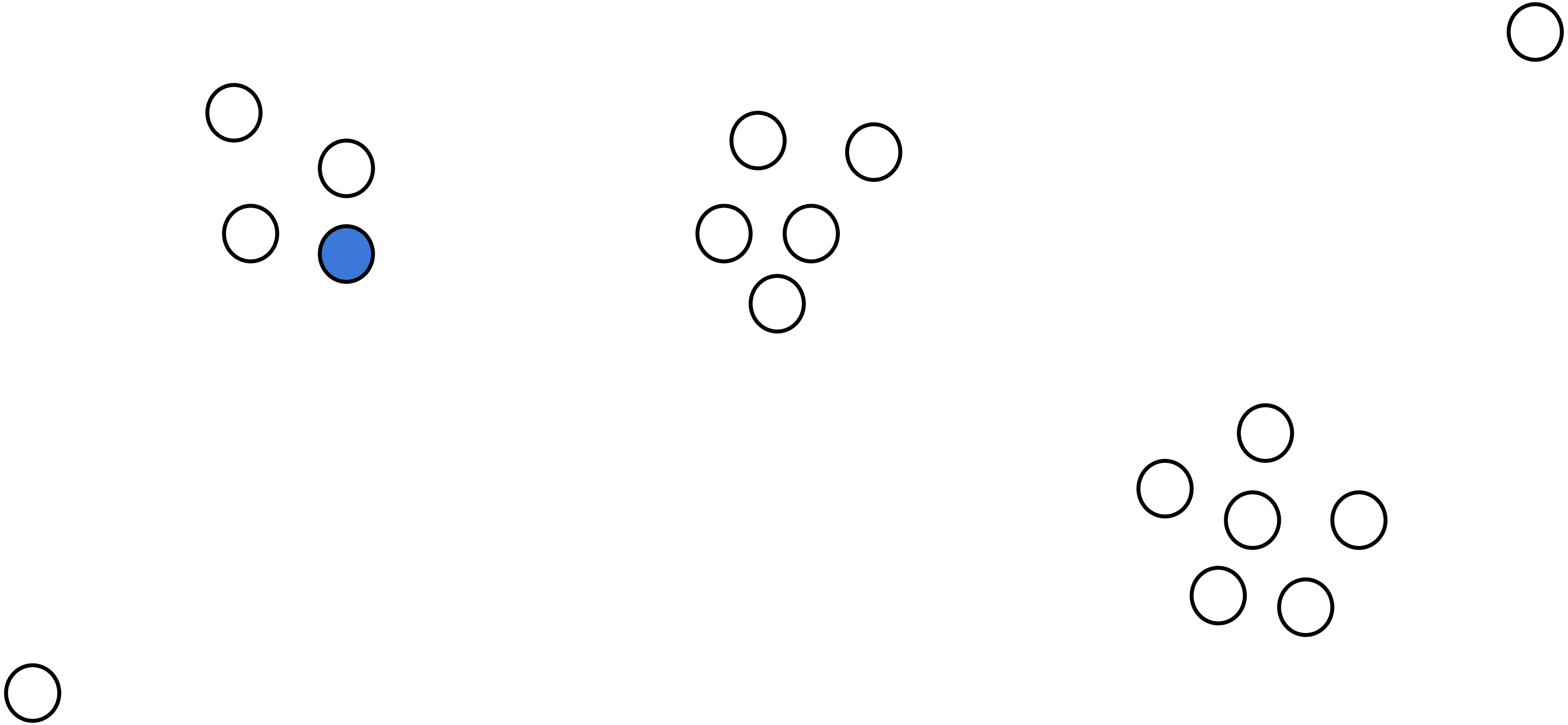
# K-means - Farthest First Traversal



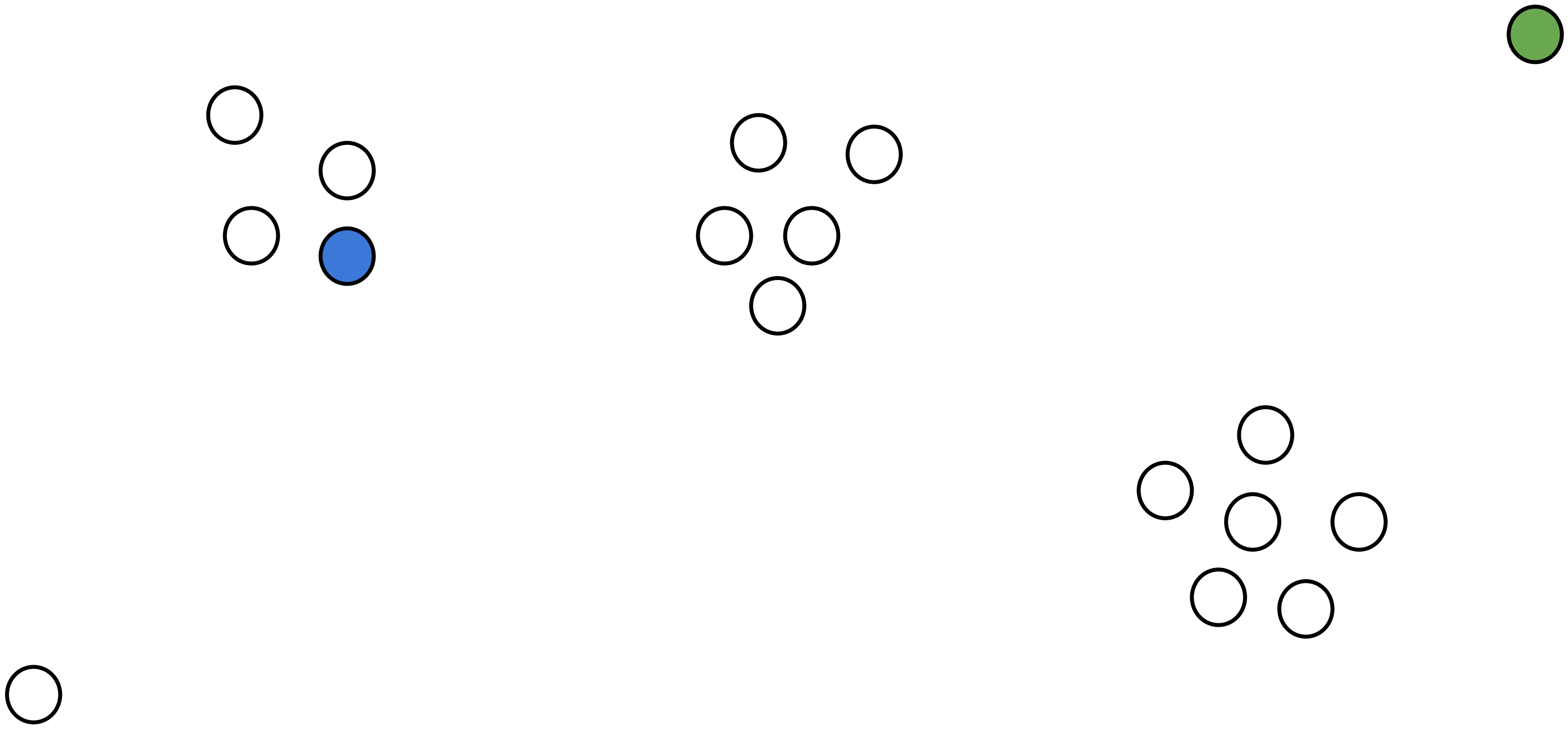
# K-means - FFT and outliers



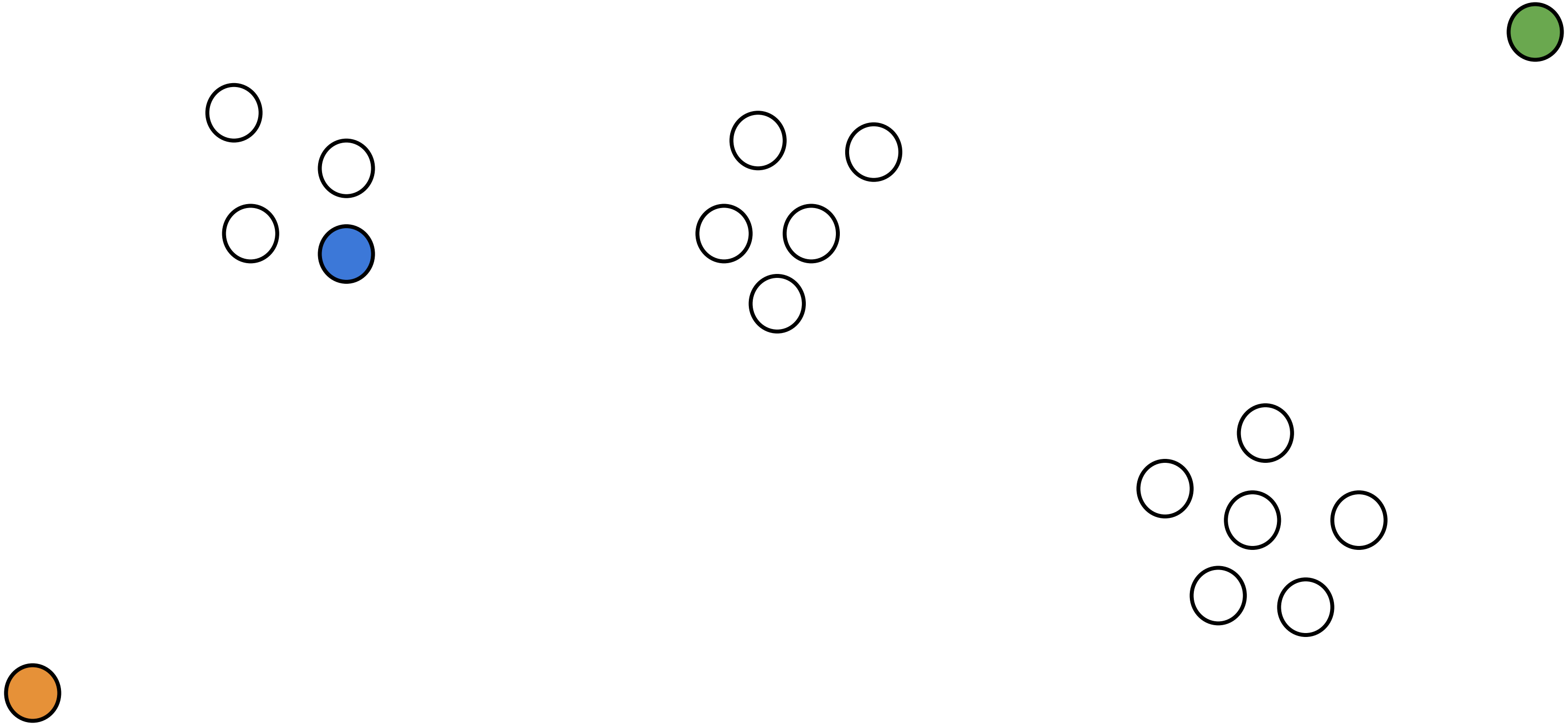
# K-means - FFT and outliers



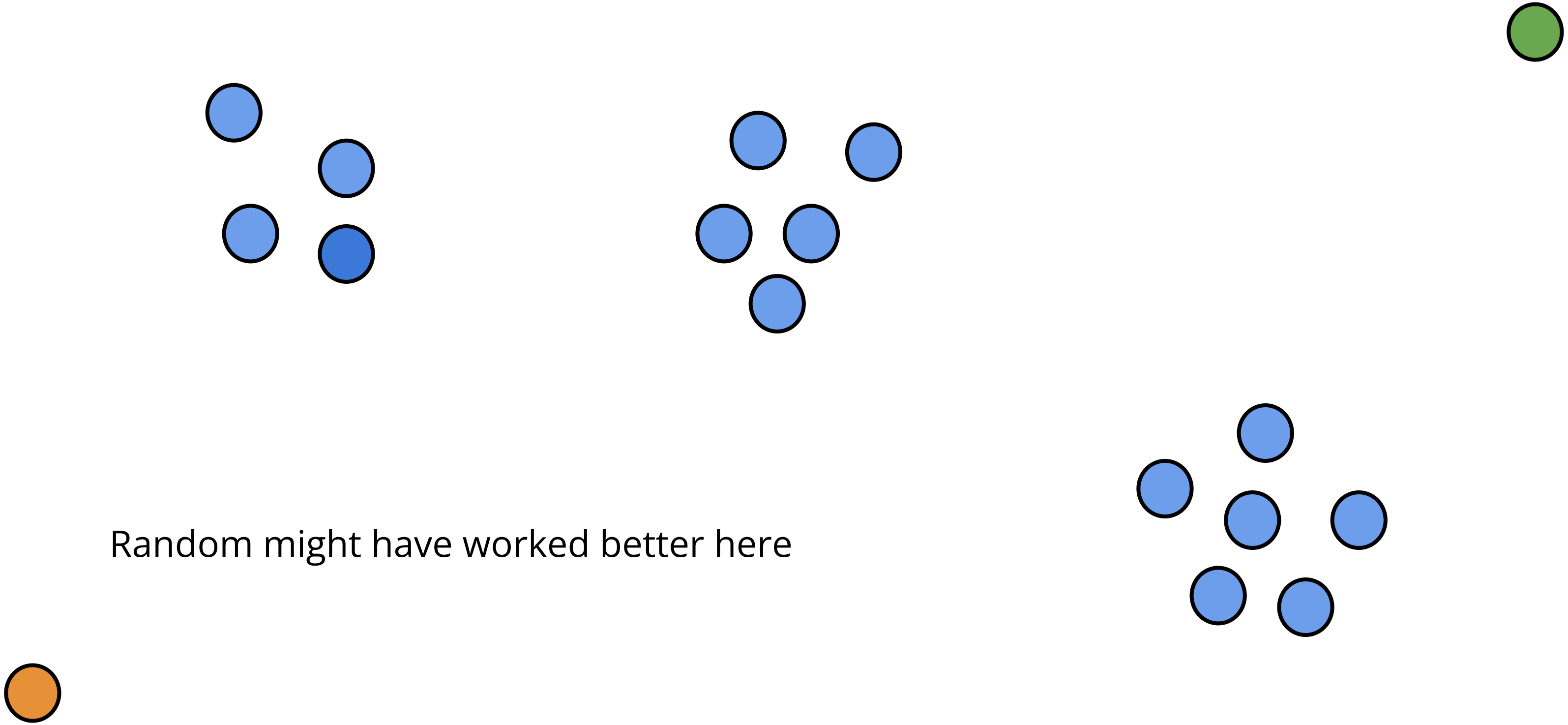
# K-means - FFT and outliers



# K-means - FFT and outliers



# K-means - FFT and outliers



Random might have worked better here

# K-means++

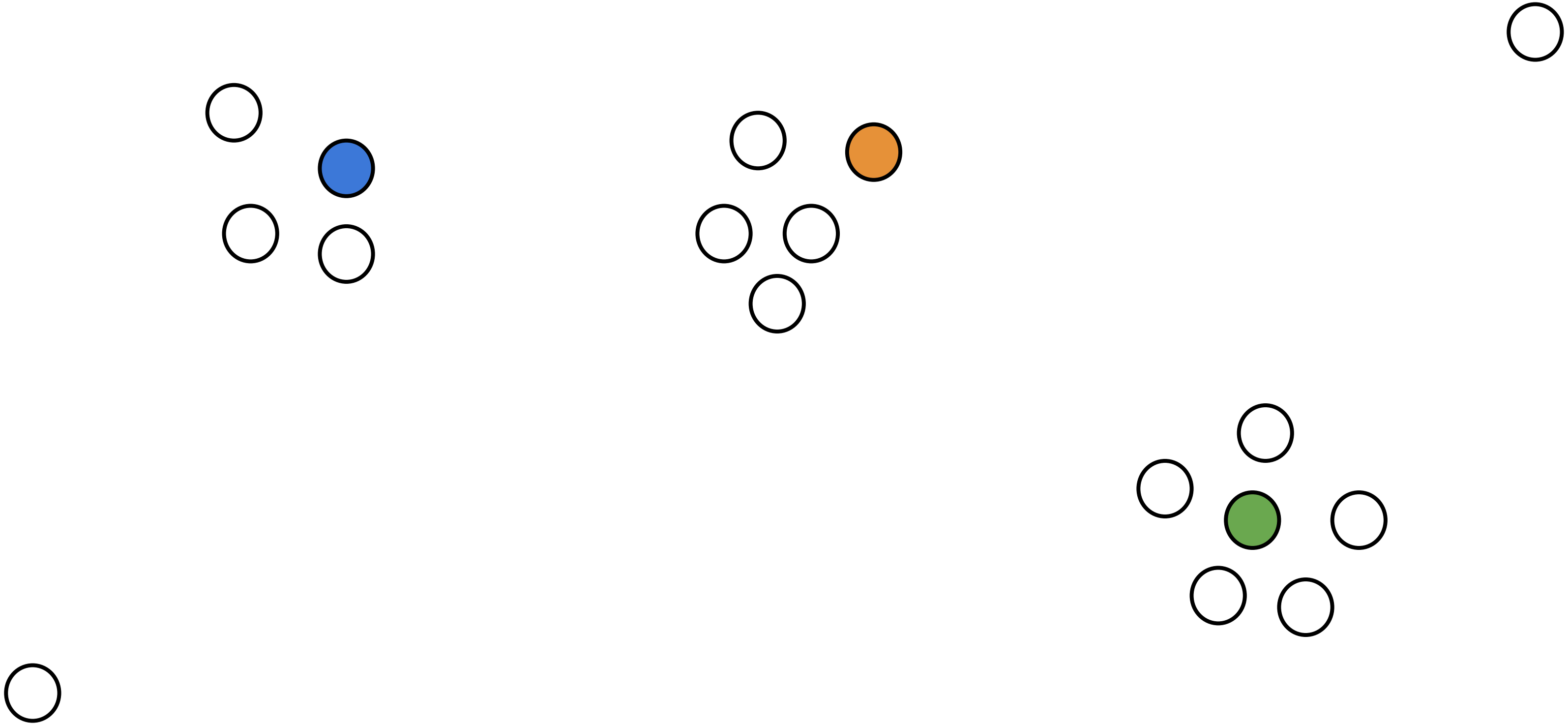
Initialize with a combination of the two methods:

1. Start with a random center
2. Let **D(x)** be the distance between **x** and the centers selected so far. Choose the next center with probability proportional to $\mathbf{D(x)^a}$

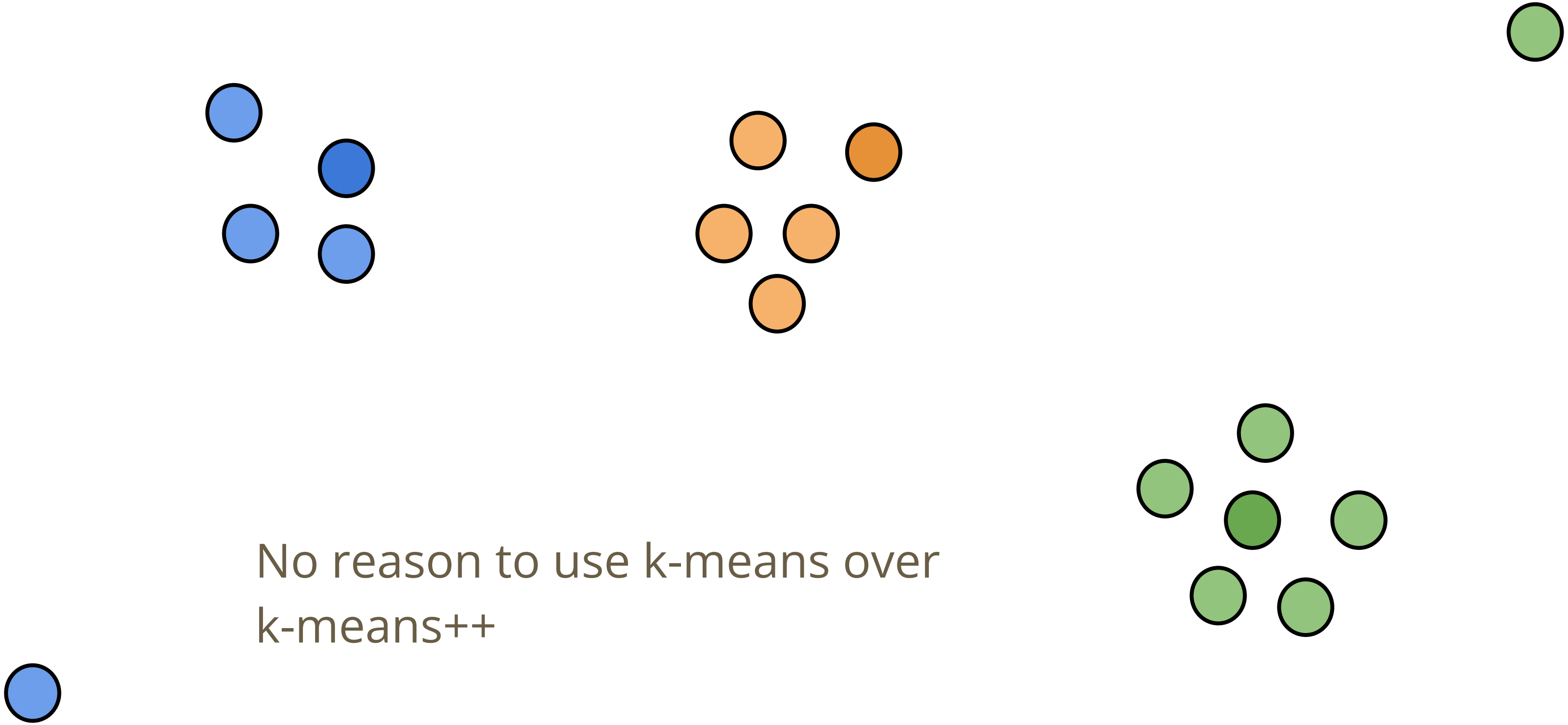
When:

- **a = 0** : random initialization (all points have equal probability)
- **a = ∞** : farthest first traversal
- **a = 2** : K-means++

# K-means++



# K-means++



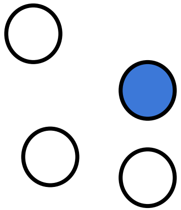
No reason to use k-means over k-means++

# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to $\mathbf{D(x)}^a$?

# K-means++

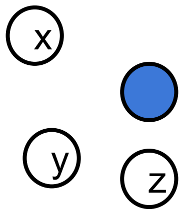
Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to $\mathbf{D(x)^a}$?



# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to $\mathbf{D(x)^2}$?

Let's set **a = 2**

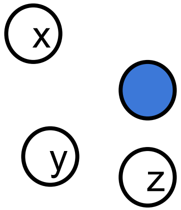



$\mathbf{D(x)^2 = 3^2 = 9}$
$\mathbf{D(y)^2 = 2^2 = 4}$
$\mathbf{D(z)^2 = 1^2 = 1}$

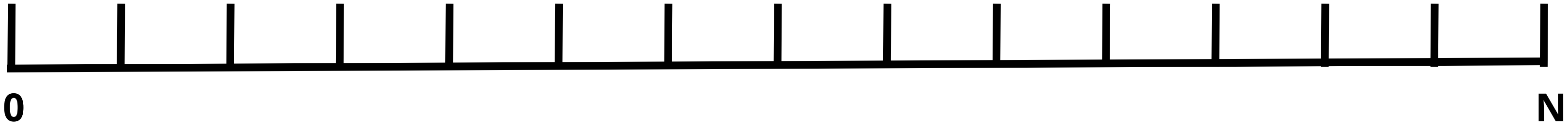
# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to $\mathbf{D(x)^2}$?

Let's set $\mathbf{a = 2}$



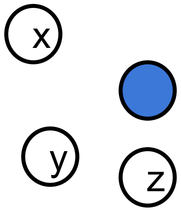

$$\mathbf{D(x)^2 = 3^2 = 9}$$
$$\mathbf{D(y)^2 = 2^2 = 4}$$
$$\mathbf{D(z)^2 = 1^2 = 1}$$



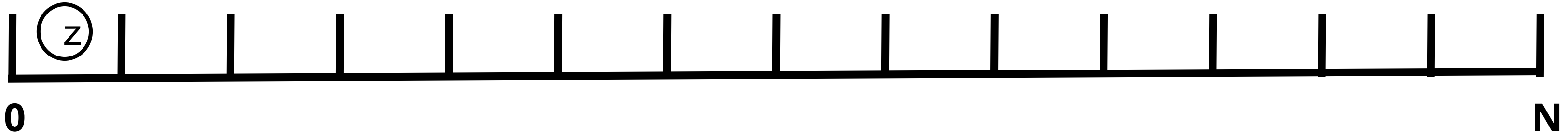
# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to **$D(x)^2$**?

Let's set **a = 2**



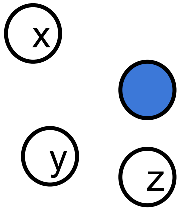

**$D(x)^2 = 3^2 = 9$**
**$D(y)^2 = 2^2 = 4$**
**$D(z)^2 = 1^2 = 1$**



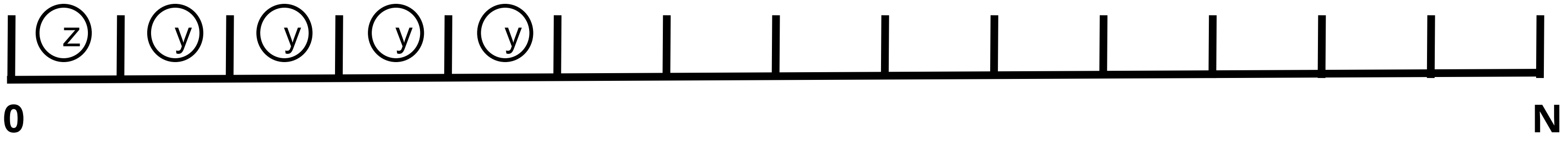
# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to $\mathbf{D(x)^2}$?

Let's set $\mathbf{a = 2}$



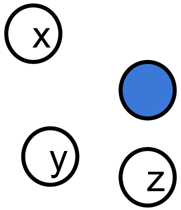

$$D(x)^2 = 3^2 = 9$$
$$D(y)^2 = 2^2 = 4$$
$$D(z)^2 = 1^2 = 1$$



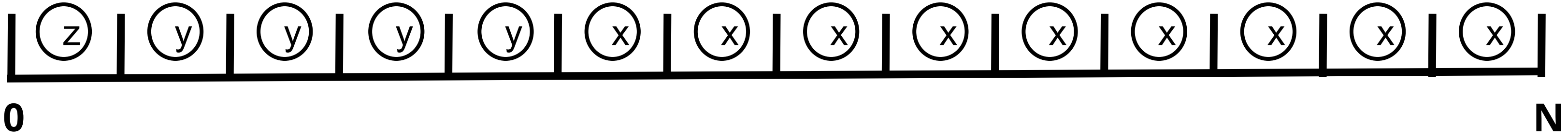
# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to **$D(x)^2$**?

Let's set **a = 2**



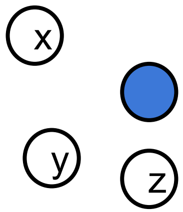

**$D(x)^2 = 3^2 = 9$**
**$D(y)^2 = 2^2 = 4$**
**$D(z)^2 = 1^2 = 1$**



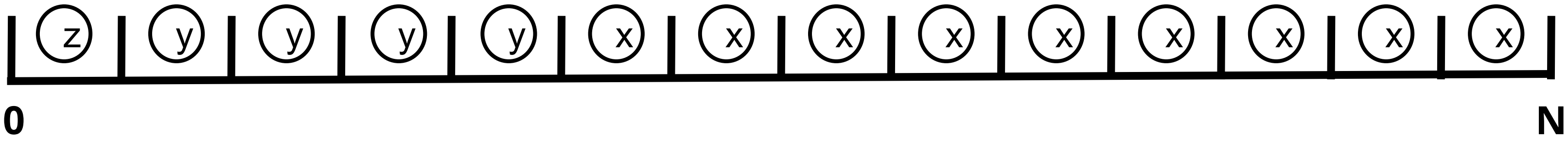
# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to $\mathbf{D(x)^2}$?

Let's set $\mathbf{a = 2}$




$$\mathbf{D(x)^2 = 3^2 = 9}$$
$$\mathbf{D(y)^2 = 2^2 = 4}$$
$$\mathbf{D(z)^2 = 1^2 = 1}$$

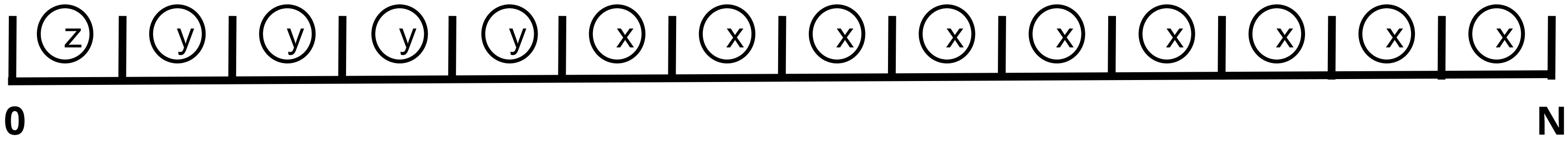



$\mathbf{= D(x)^2 + D(y)^2 + D(z)^2 = 14}$

# K-means++

Suppose we are given a black box that will generate a uniform random number between 0 and any **N**. How can we use this black box to select points with probability proportional to $\mathbf{D(x)^2}$?

Using the black box, we can generate a number between 0 and N to determine which point to pick next. It will be chosen with probability proportional to $\mathbf{D(x)^2}$.

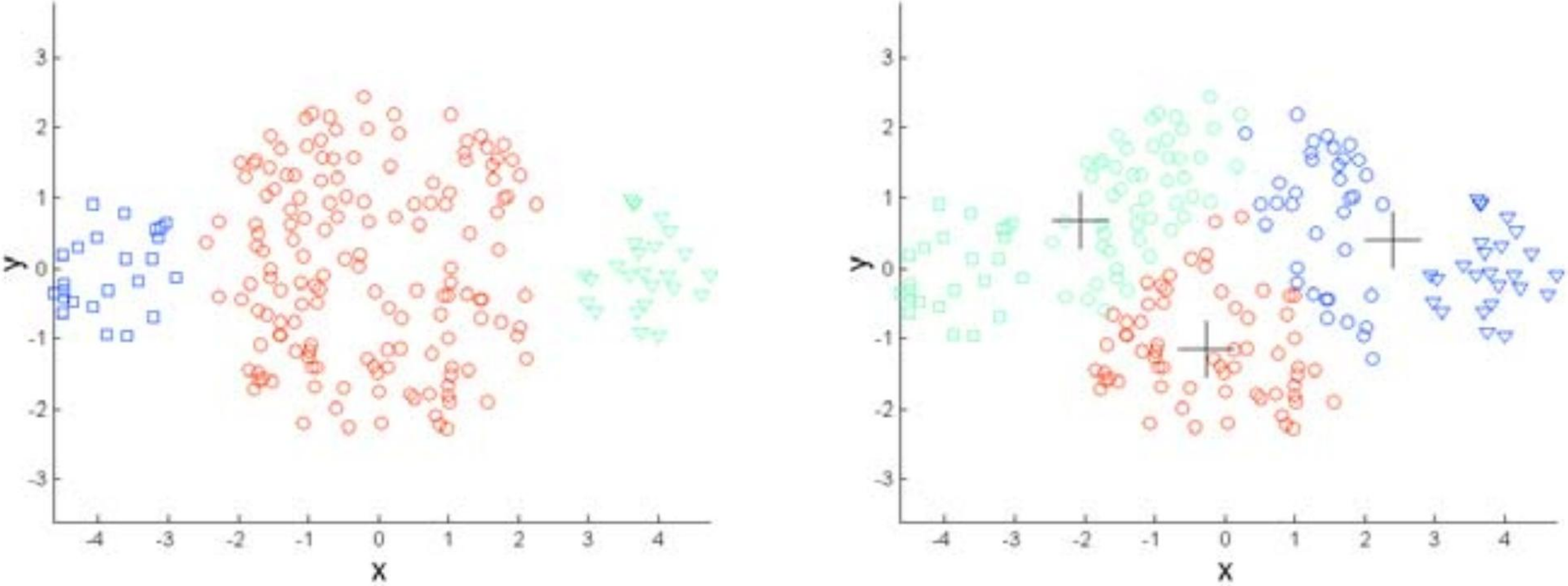



$$= D(x)^2 + D(y)^2 + D(z)^2 = 14$$

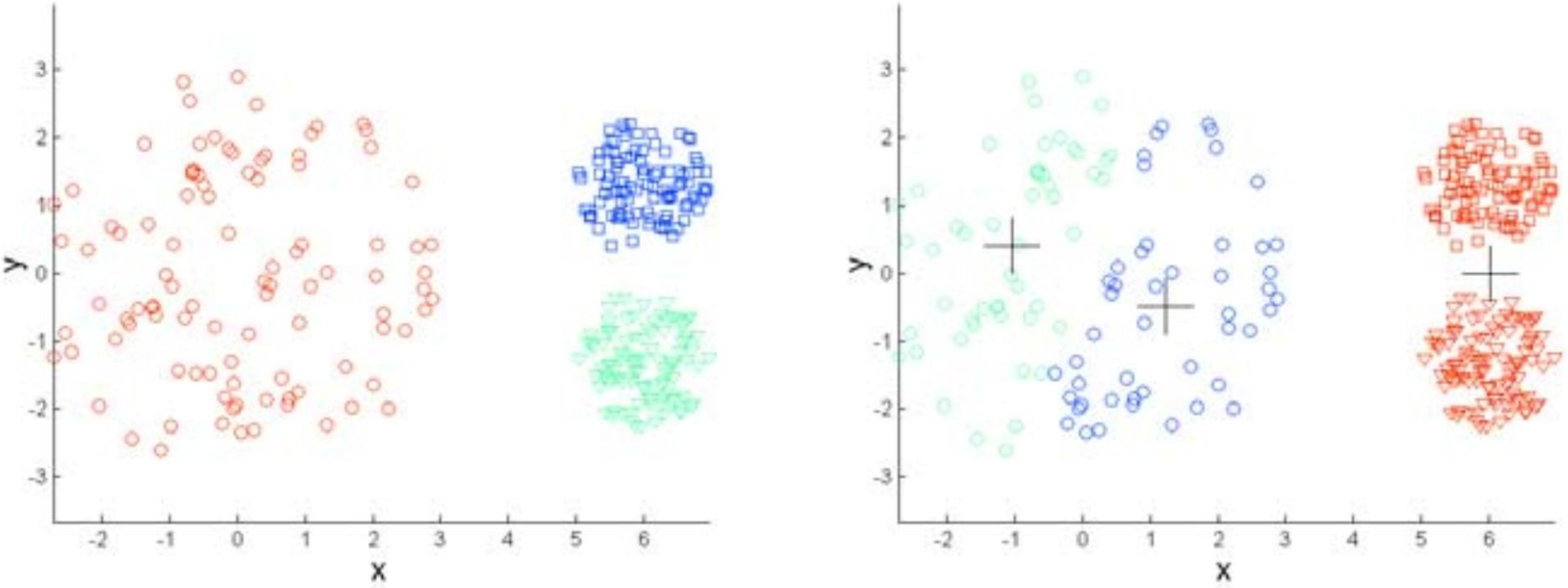
# K-means++

What happens if the black box can only generate numbers between 0 and 1?

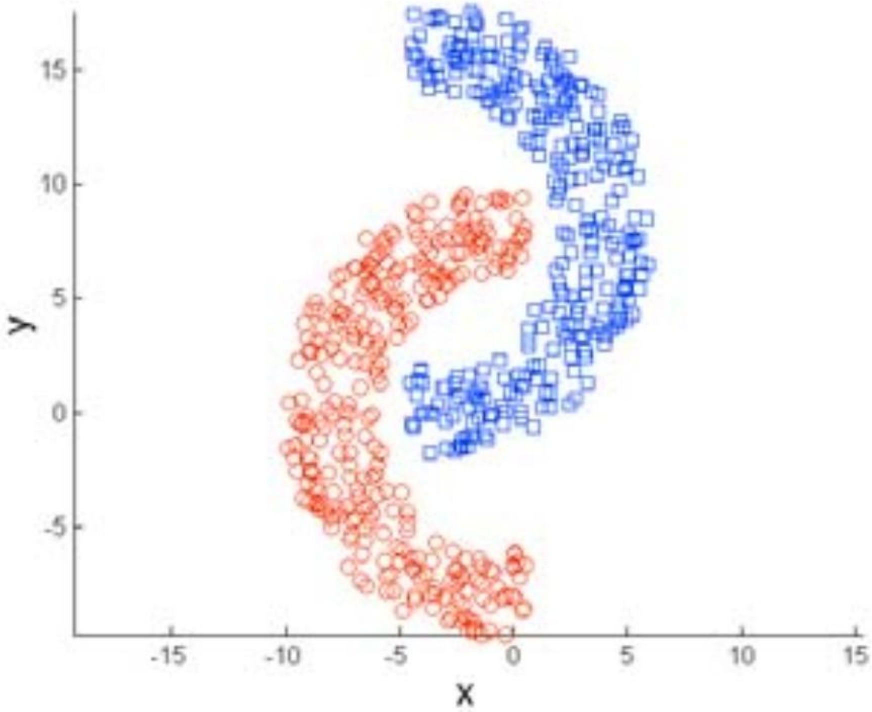
# K-means - Limitations

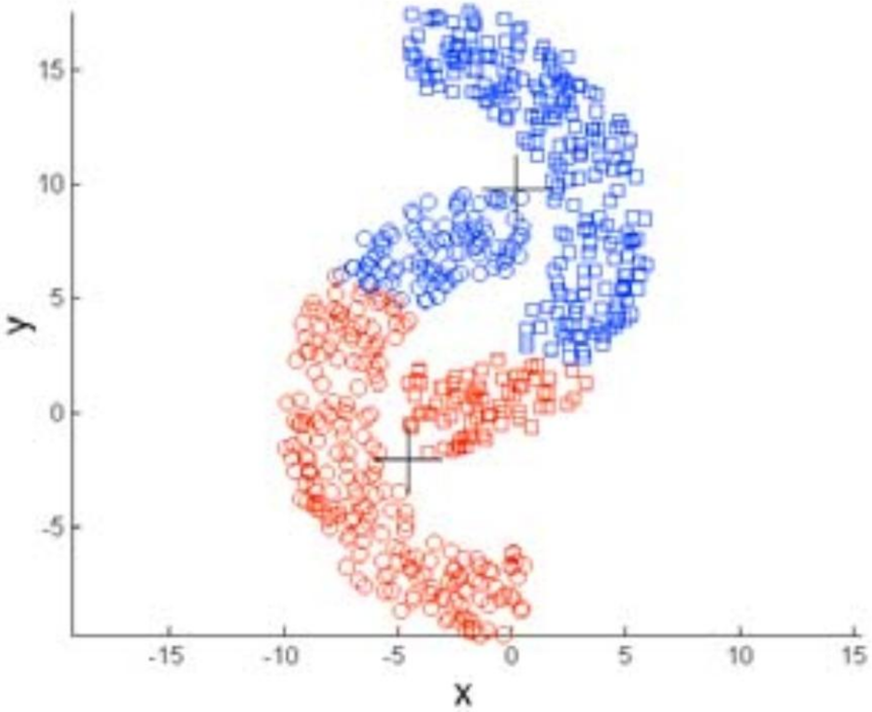


# K-means - Limitations



# K-means - Limitations



# How to choose the right k?

1. Iterate through different values of k (elbow method)
2. Use empirical / domain-specific knowledge
   Example: Is there a known approximate distribution of the data? (K-means is good for spherical gaussians)
3. Silhouette scores

# K-means Variations

1. K-medians (uses the $L_1$ norm / manhattan distance)
2. K-medoids (any distance function + the centers must be in the dataset)
3. Weighted K-means (each point has a different weight when computing the mean)