

K-Means

Class	Algorithmic Methods of Data Mining
Program	M. Sc. Data Science
University	Sapienza University of Rome
Semester	Fall 2018
Slides by	Carlos Castillo http://chato.cl/

Sources:

- Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, May 2014. Example 13.1. [[download](#)]
- Evimaria Terzi: Data Mining course at Boston University <http://www.cs.bu.edu/~evimaria/cs565-13.html>

The k-means problem

- consider set $X = \{x_1, \dots, x_n\}$ of n points in \mathbb{R}^d
- assume that the number k is given
- **problem:**
 - find k points c_1, \dots, c_k (named **centers** or **means**)

so that the **cost**

$$\sum_{i=1}^n \min_j \{L_2^2(x_i, c_j)\} = \sum_{i=1}^n \min_j \|x_i - c_j\|_2^2$$

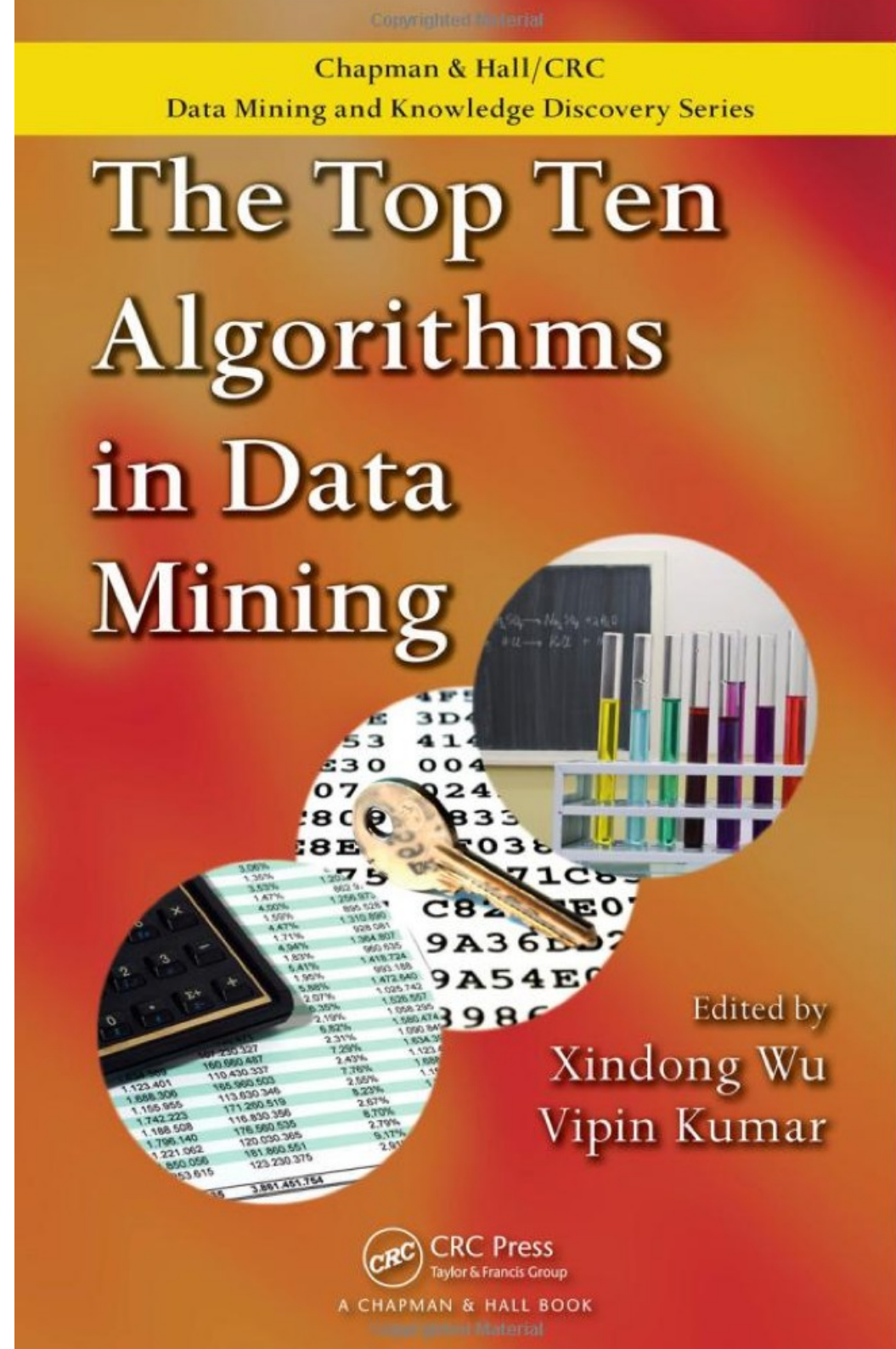
is minimized

The k-means problem

- $k=1$ and $k=n$ are **easy** special cases (*why?*)
- an **NP-hard** problem if the **dimension** of the data is at least 2 ($d \geq 2$)
- in practice, a **simple iterative algorithm** works quite well

The k-means algorithm

- voted among the **top-10 algorithms** in data mining
- **one way** of solving the **k-means** problem



K-means algorithm

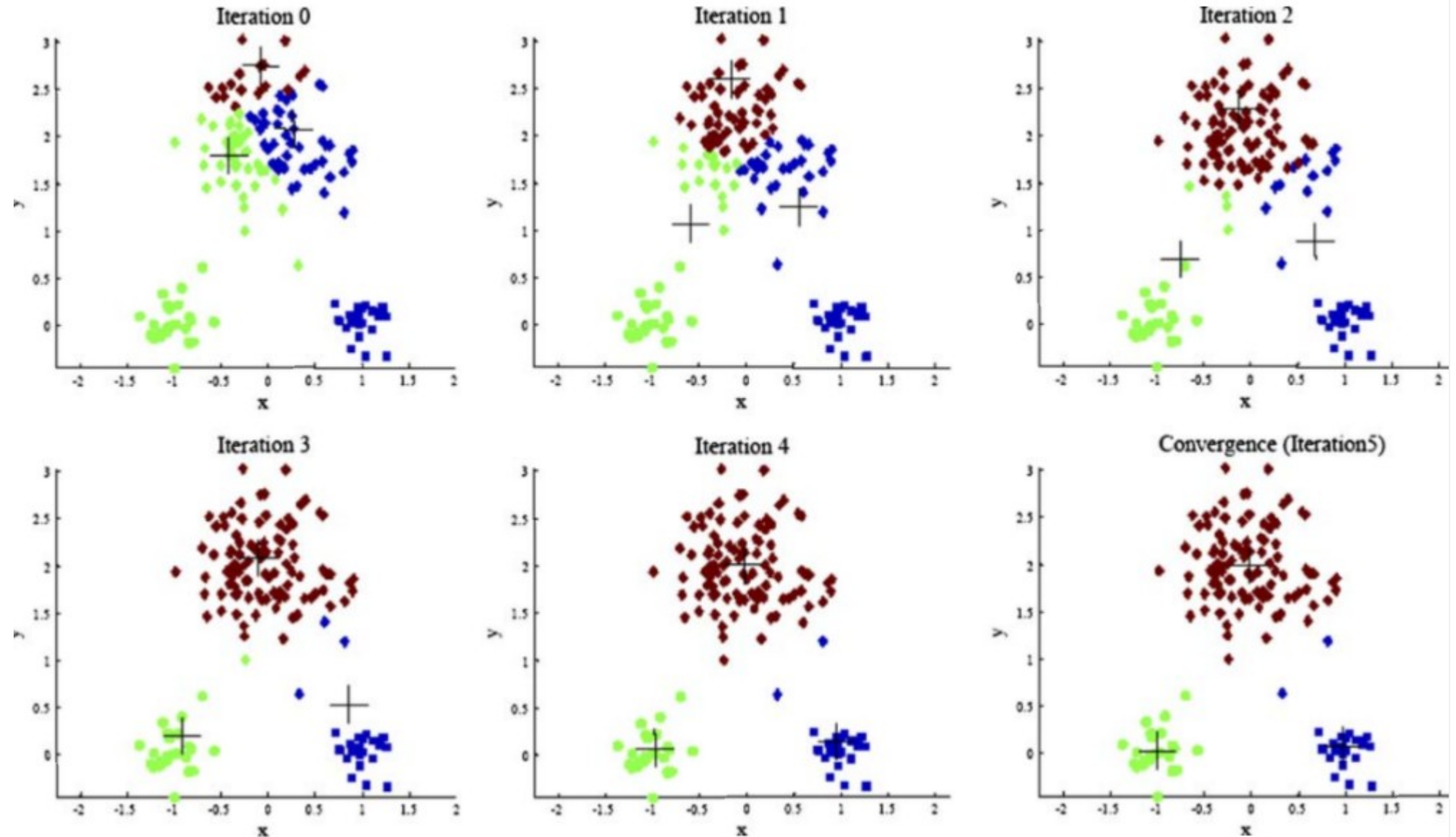
K-MEANS (\mathbf{D}, k, ϵ):

```
1  $t = 0$ 
2 Randomly initialize  $k$  centroids:  $\mu_1^t, \mu_2^t, \dots, \mu_k^t \in \mathbb{R}^d$ 
3 repeat
4    $t \leftarrow t + 1$ 
5    $C_j \leftarrow \emptyset$  for all  $j = 1, \dots, k$ 
   // Cluster Assignment Step
6   foreach  $\mathbf{x}_j \in \mathbf{D}$  do
7      $j^* \leftarrow \operatorname{argmin}_i \left\{ \|\mathbf{x}_j - \mu_i^t\|^2 \right\}$  // Assign  $\mathbf{x}_j$  to closest centroid
8      $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$ 
   // Centroid Update Step
9   foreach  $i = 1$  to  $k$  do
10     $\mu_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$ 
11 until  $\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$ 
```

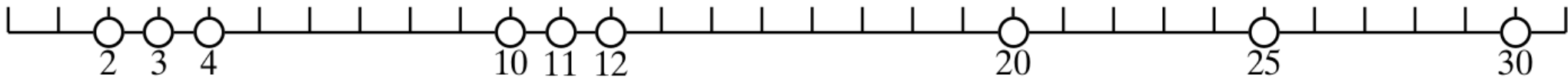
The k-means algorithm

1. **randomly** (or with another method) pick k cluster centers $\{c_1, \dots, c_k\}$
2. for each j , set the cluster X_j to be the set of points in X that are **the closest to center c_j**
3. for each j let c_j be **the center of cluster X_j** (mean of the vectors in X_j)
1. repeat (go to step 2) until convergence

Sample execution



1-dimensional clustering exercise



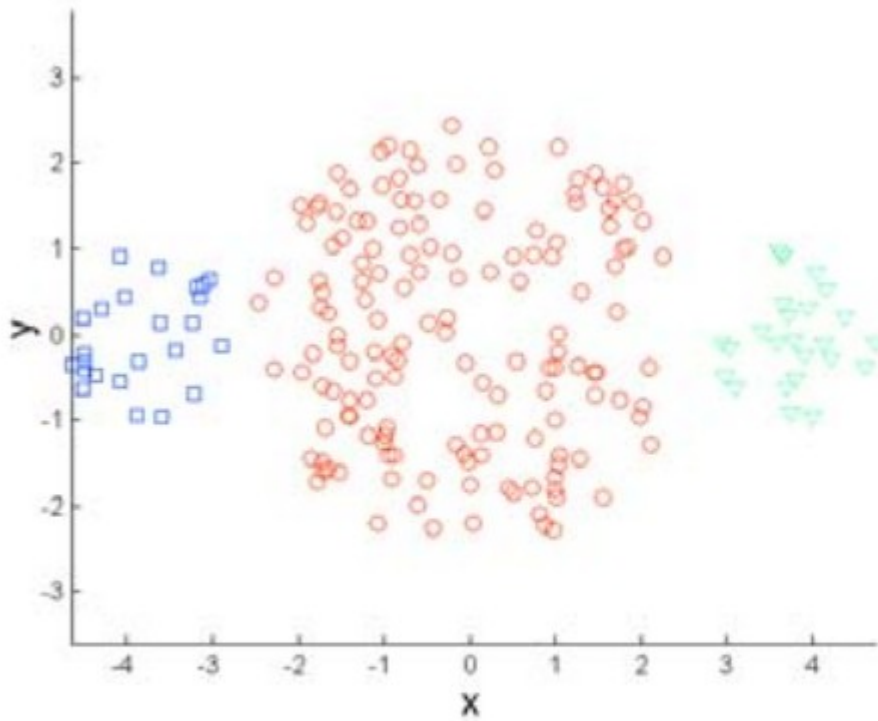
Exercise:

- *For the data in the figure*
 - *Run k-means with $k=2$ and initial centroids $u_1=2$, $u_2=4$ (Verify: last centroids are 18 units apart)*
- *Try with $k=3$ and initialization 2,3,30*

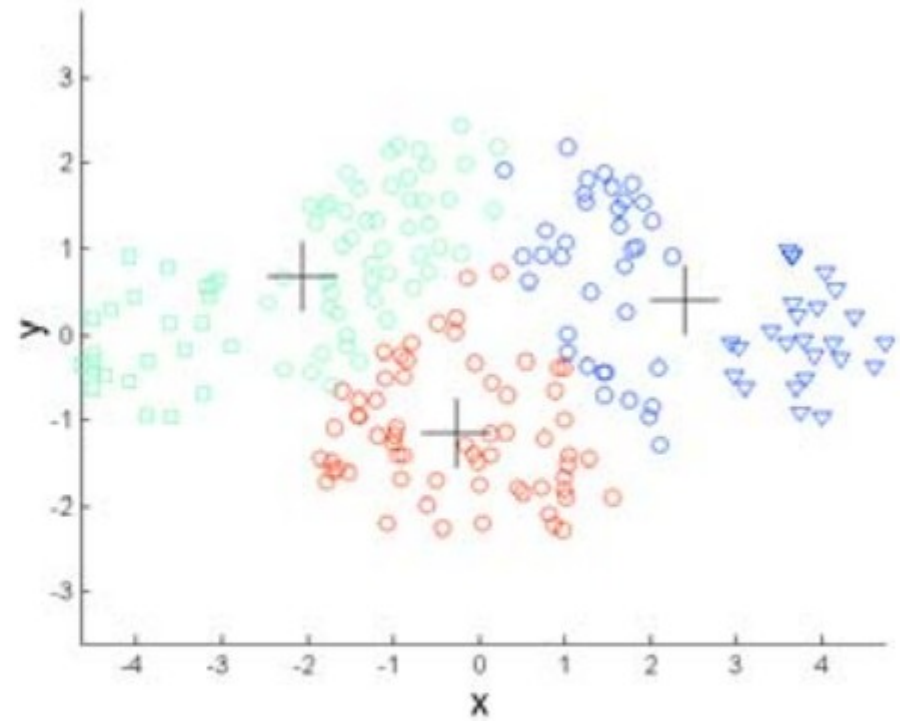
Limitations of k-means

- Clusters of different size
- Clusters of different density
- Clusters of non-globular shape
- Sensitive to initialization

Limitations of k-means: different sizes

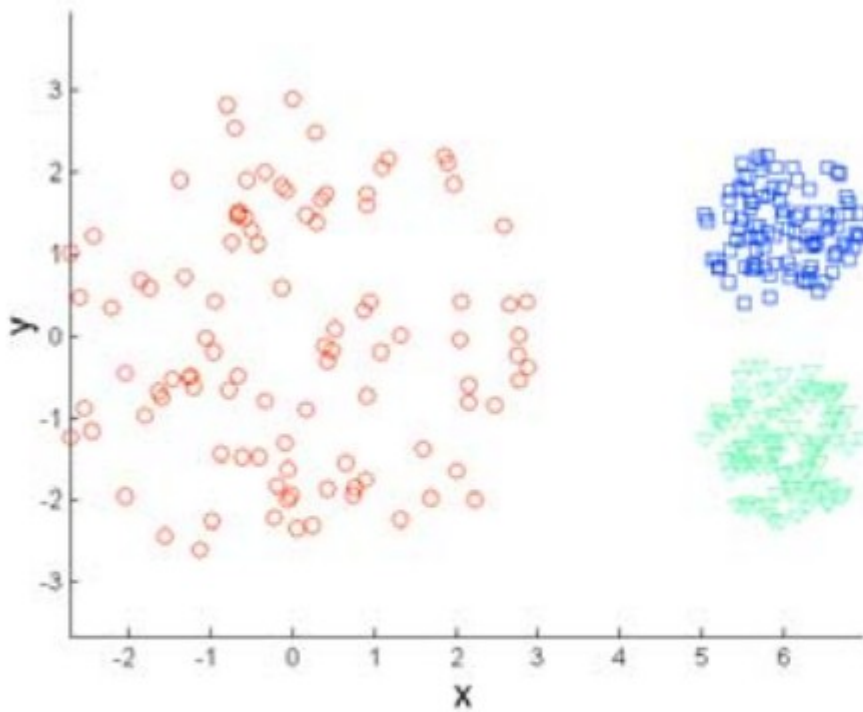


Original Points

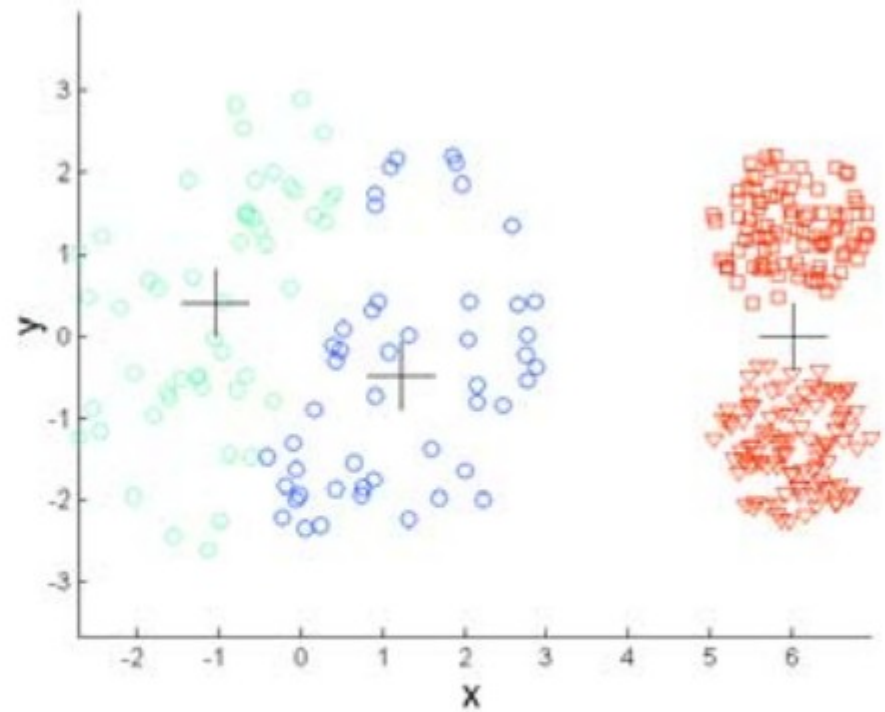


K-means (3 Clusters)

Limitations of k-means: different density

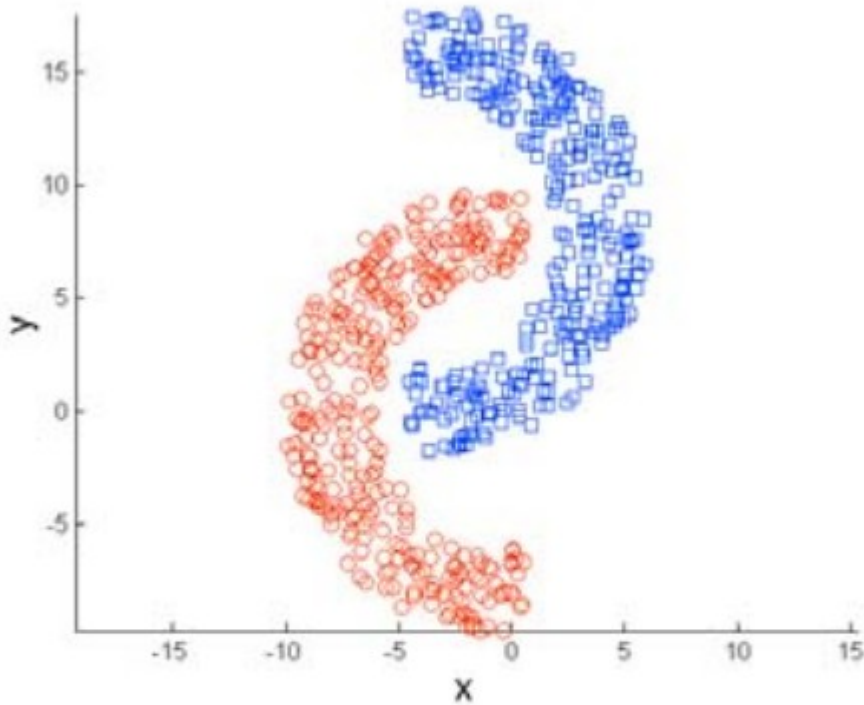


Original Points

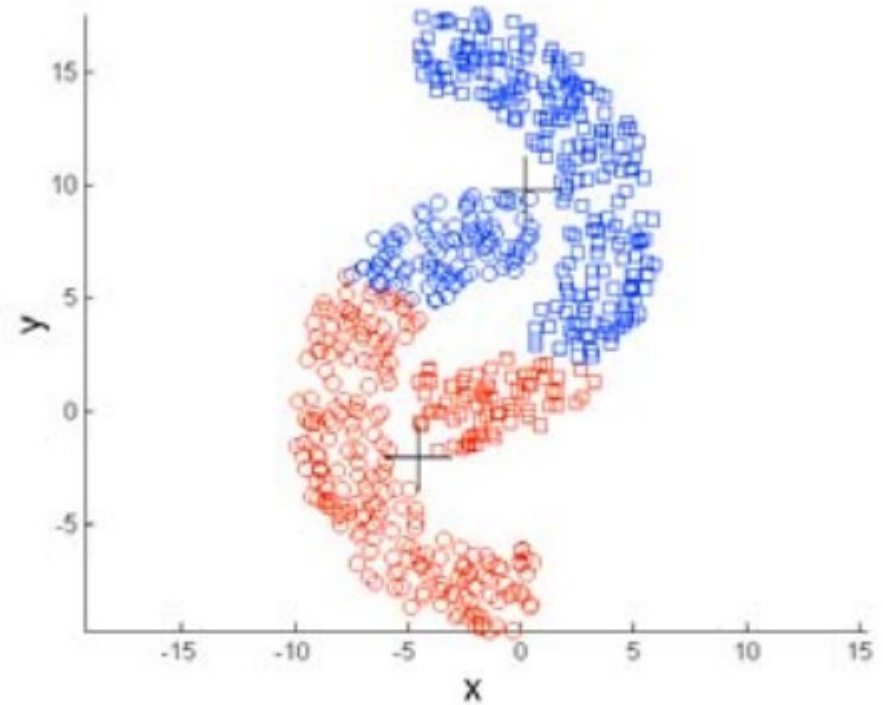


K-means (3 Clusters)

Limitations of k-means: non-spherical shapes

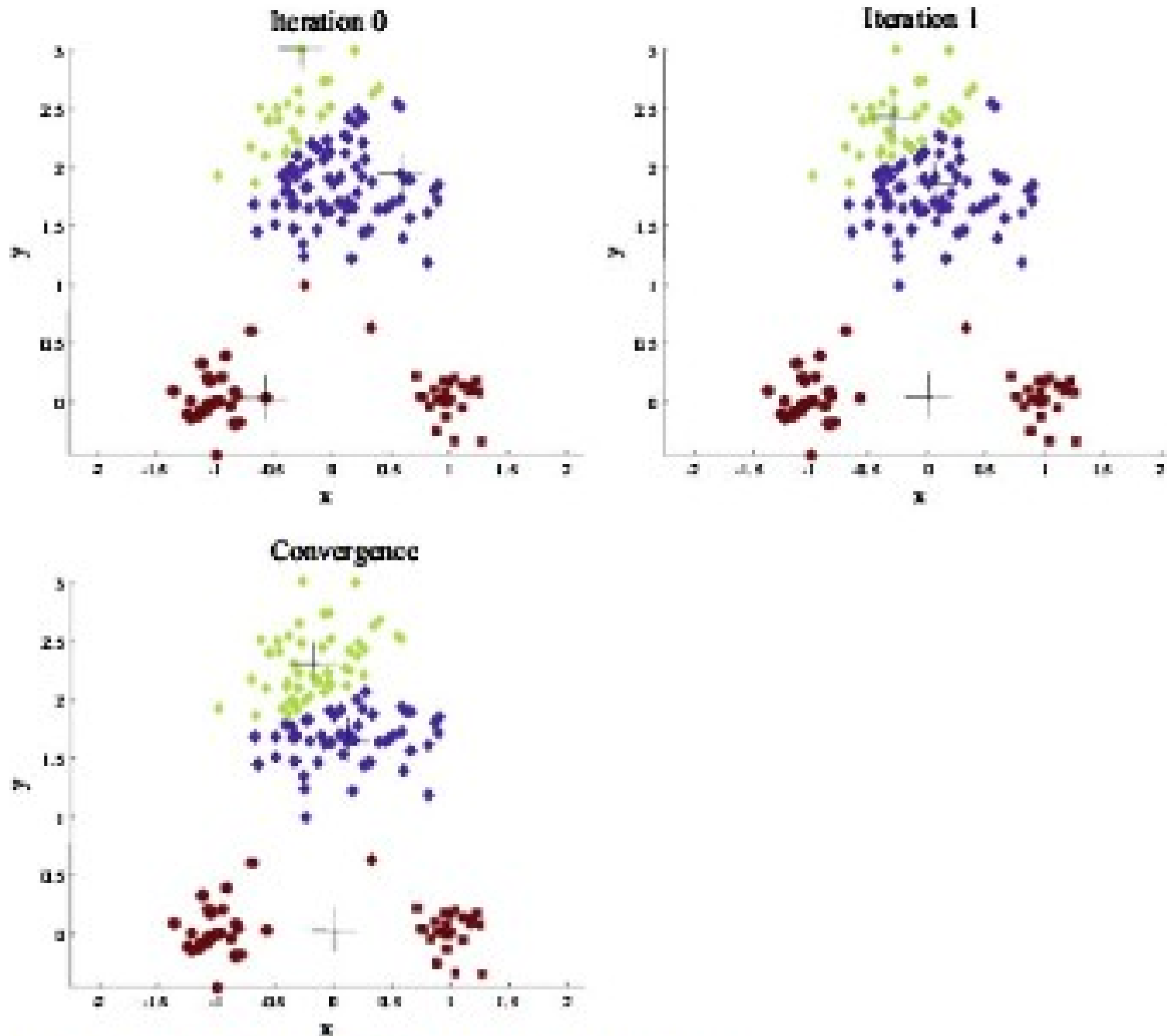


Original Points



K-means (2 Clusters)

Effects of bad initialization



k-means algorithm

- finds a **local optimum**
- often **converges** quickly
but not always
- the **choice of initial points** can have **large influence** in the result
- tends to find **spherical clusters**
- **outliers** can cause a problem
- different **densities** may cause a problem

Advanced: k-means initialization

Initialization

- random initialization
- random, but repeat many times and take the best solution
 - helps, but solution can still be bad
- pick points that are distant to each other
 - k-means++
 - provable guarantees

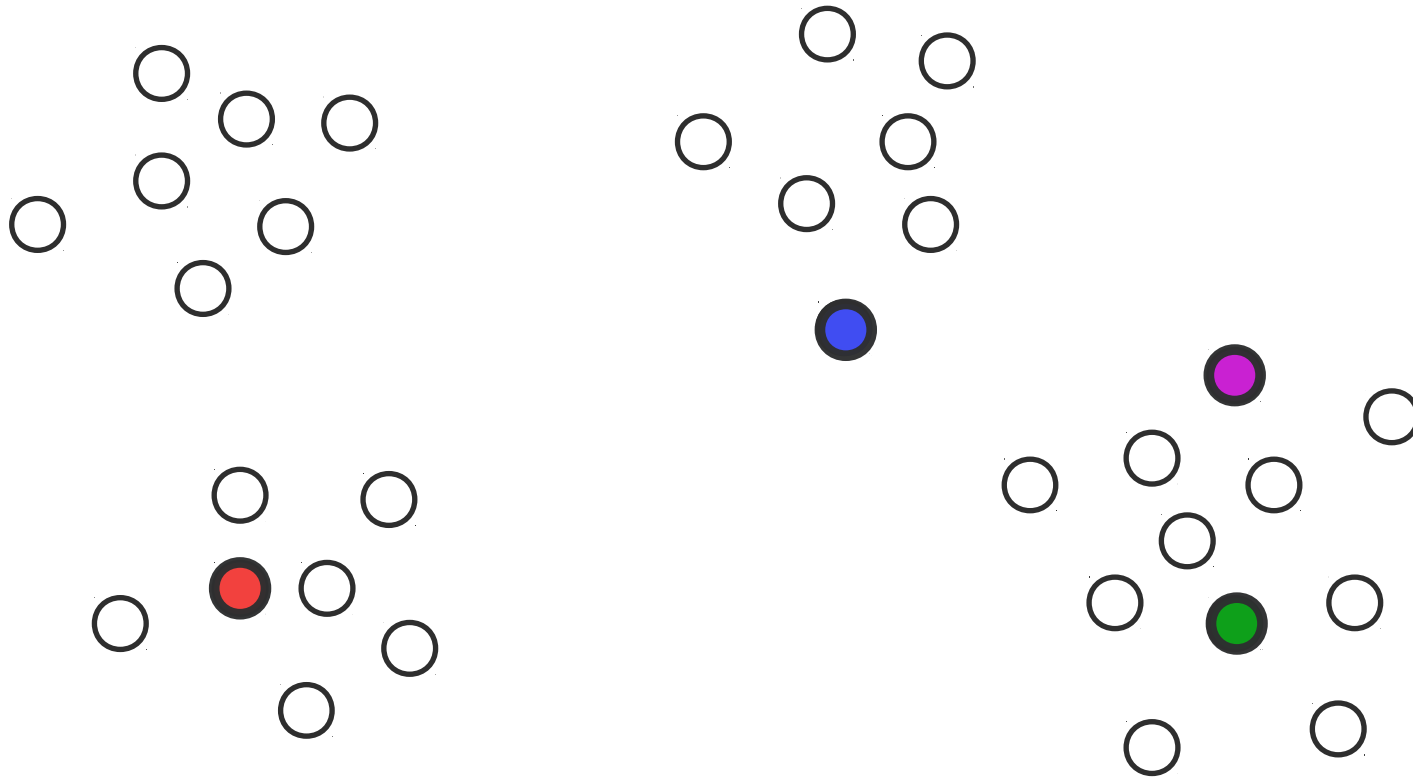
k-means++

David Arthur and Sergei Vassilvitskii

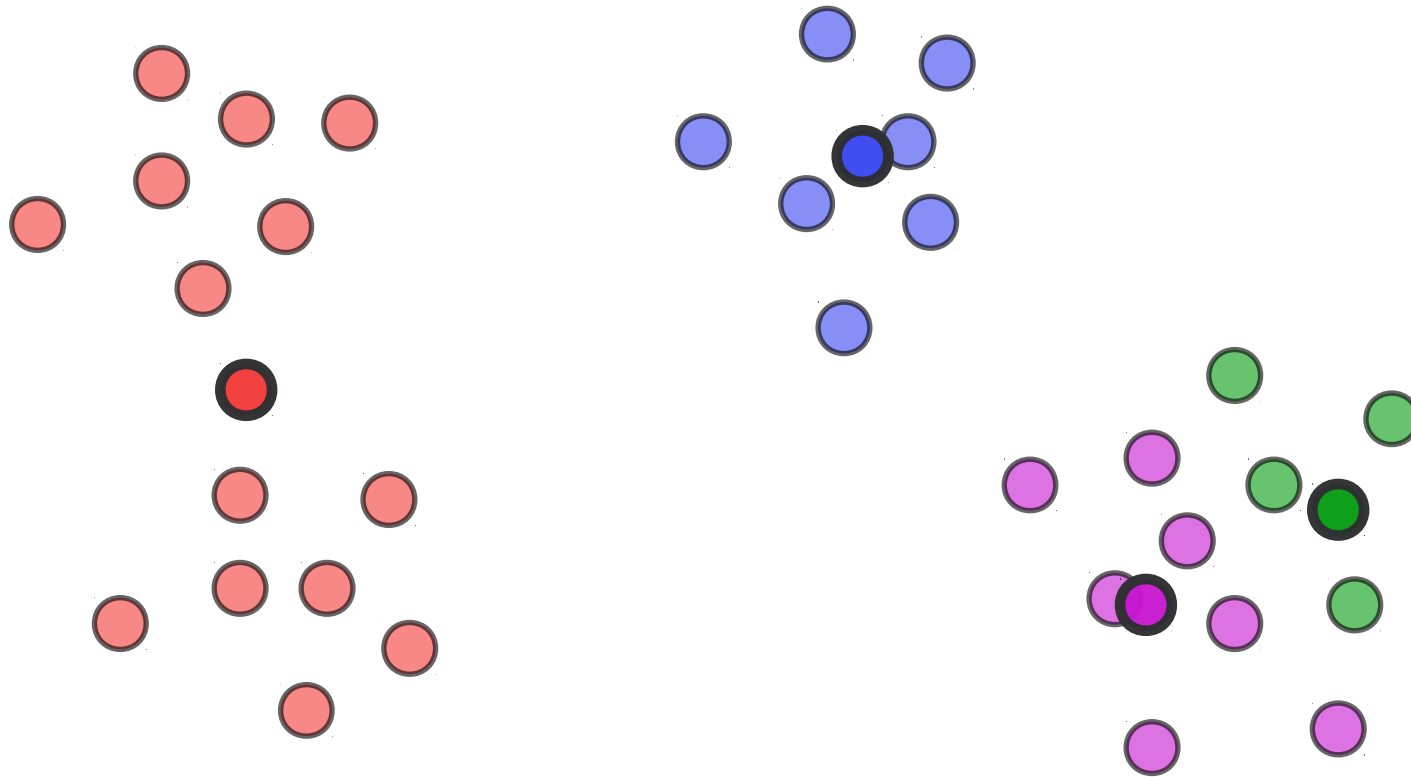
k-means++: The advantages of careful seeding

SODA 2007

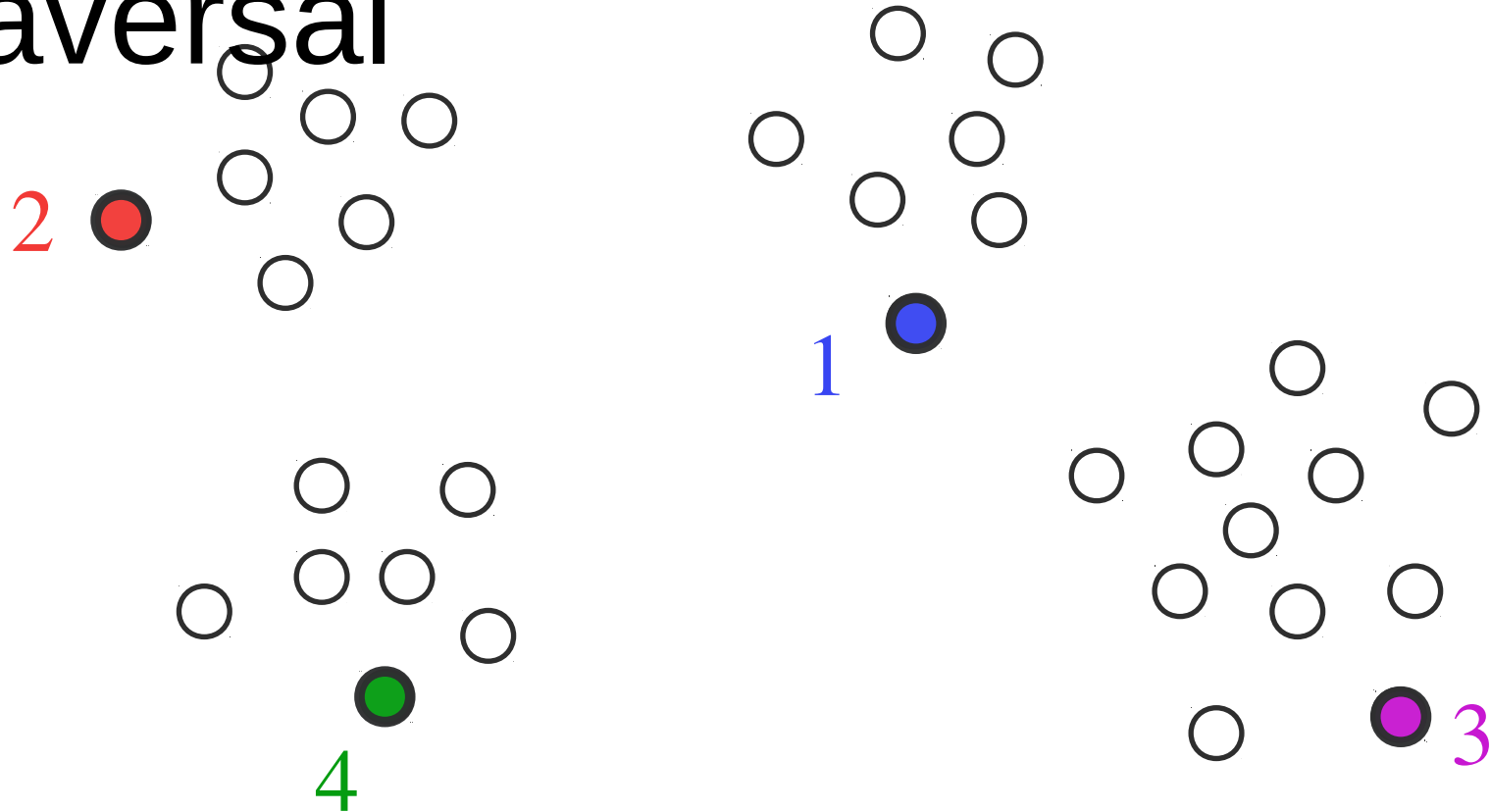
k-means algorithm: random initialization



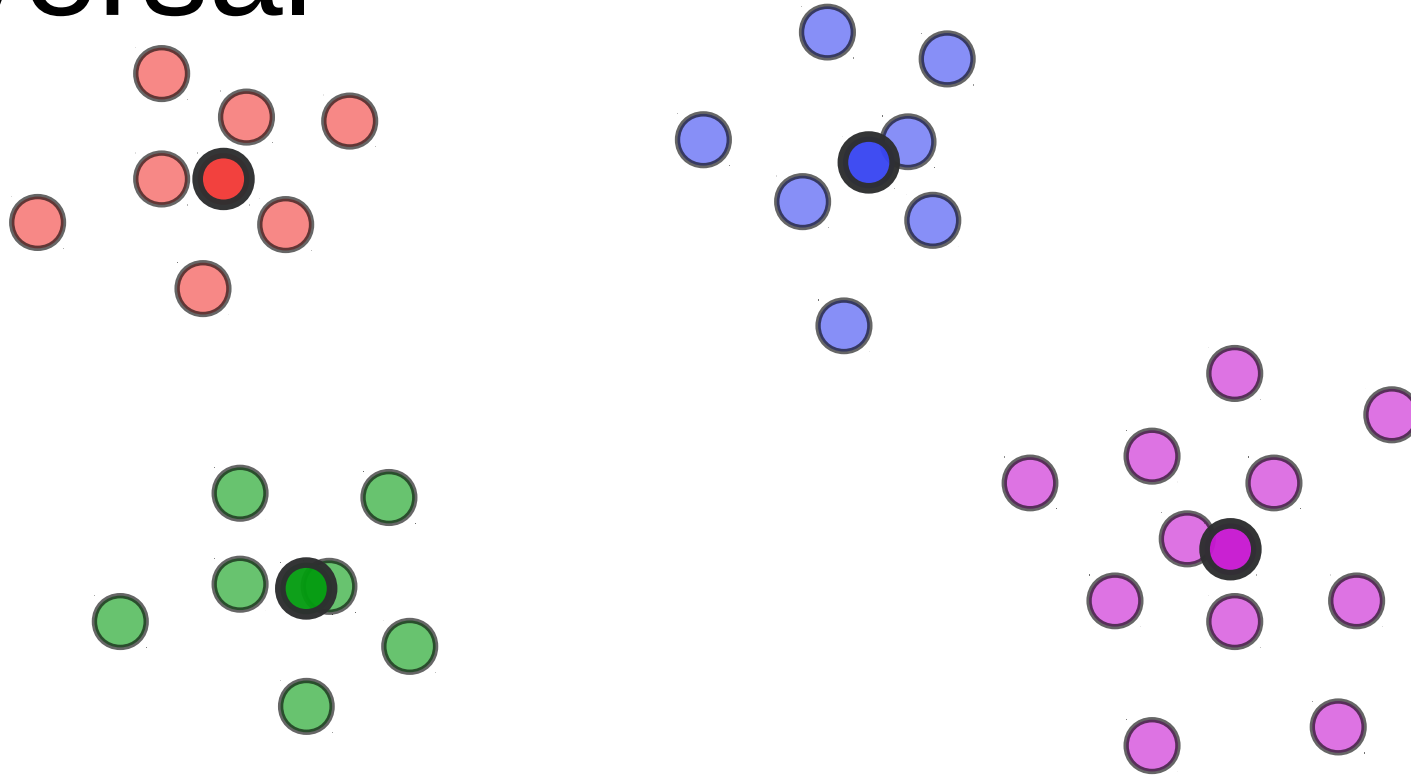
k-means algorithm: random initialization



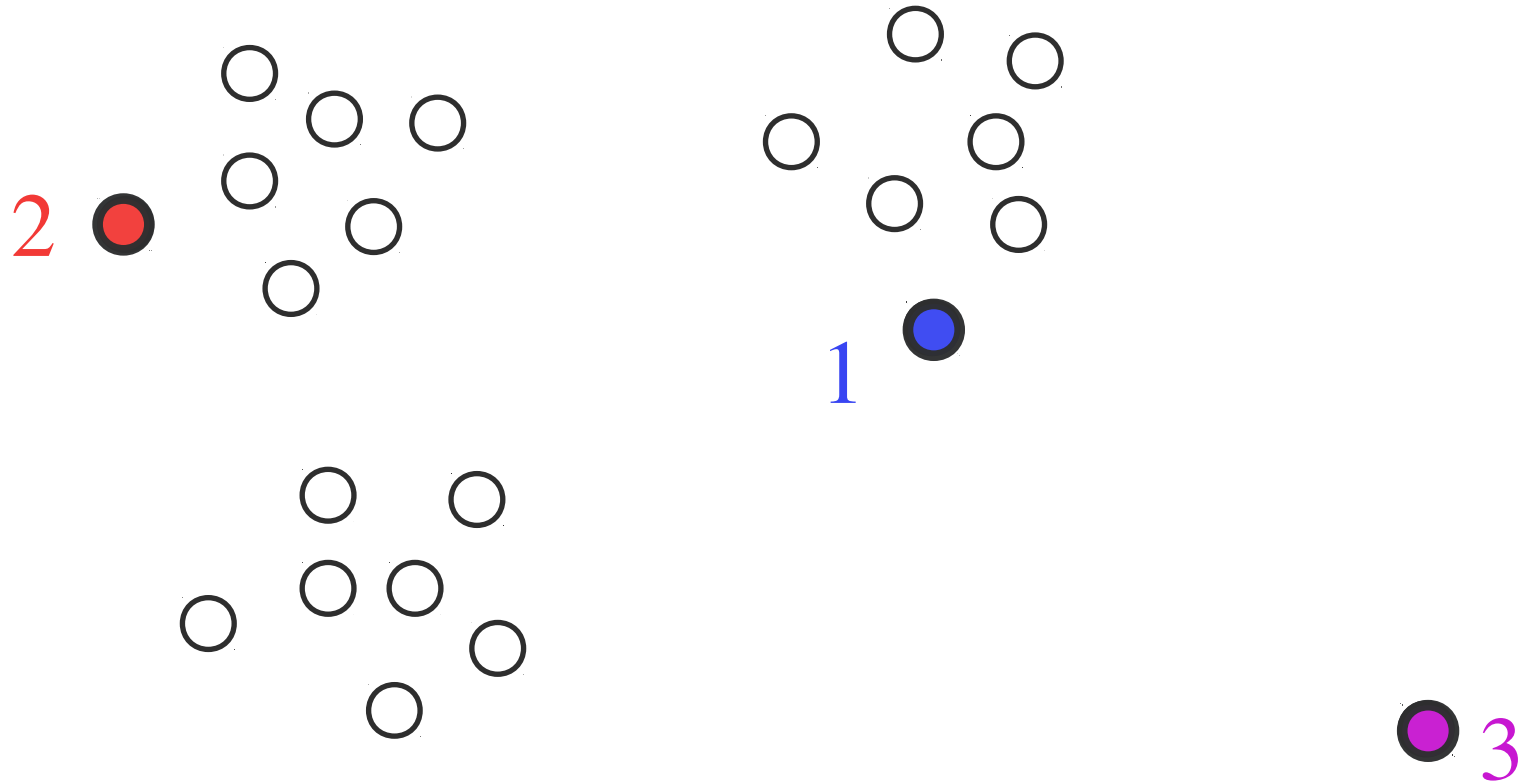
k-means algorithm: initialization with further-first traversal



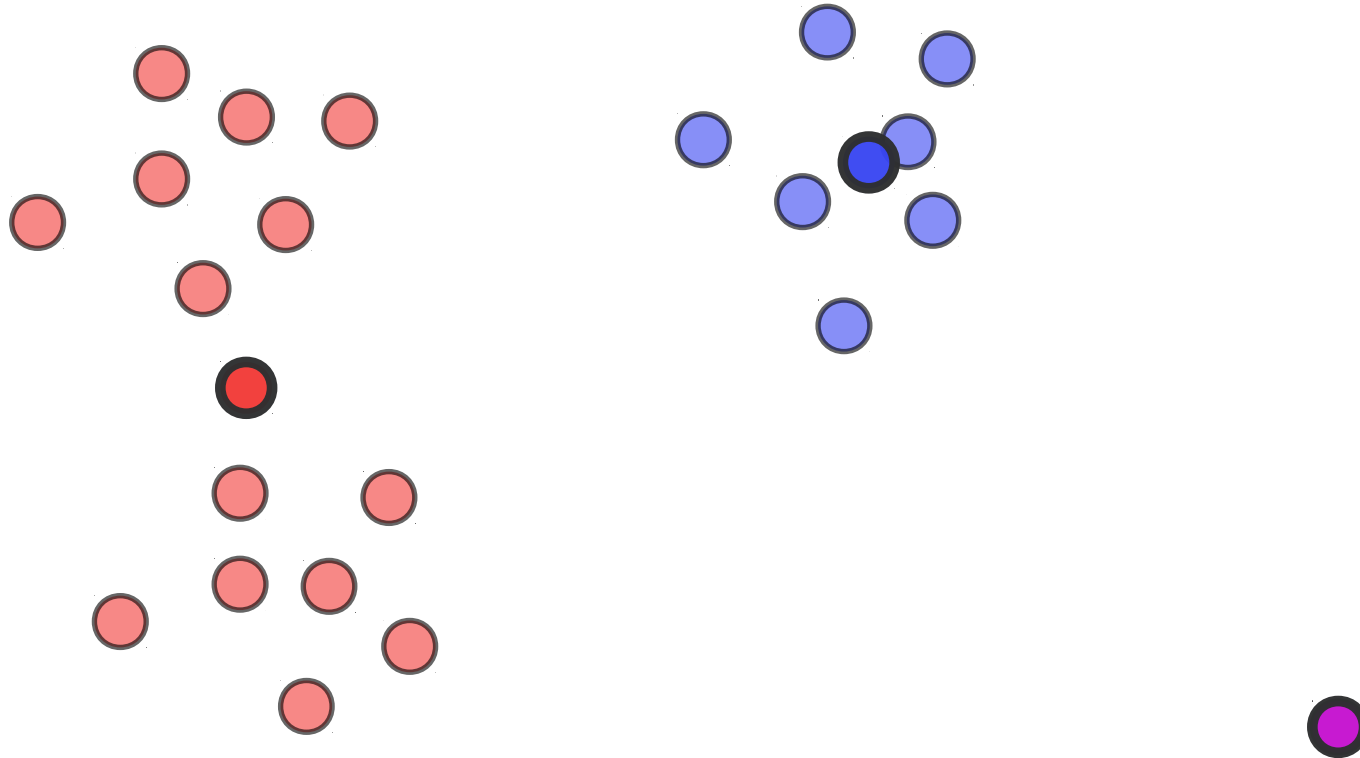
k-means algorithm: initialization with further-first traversal



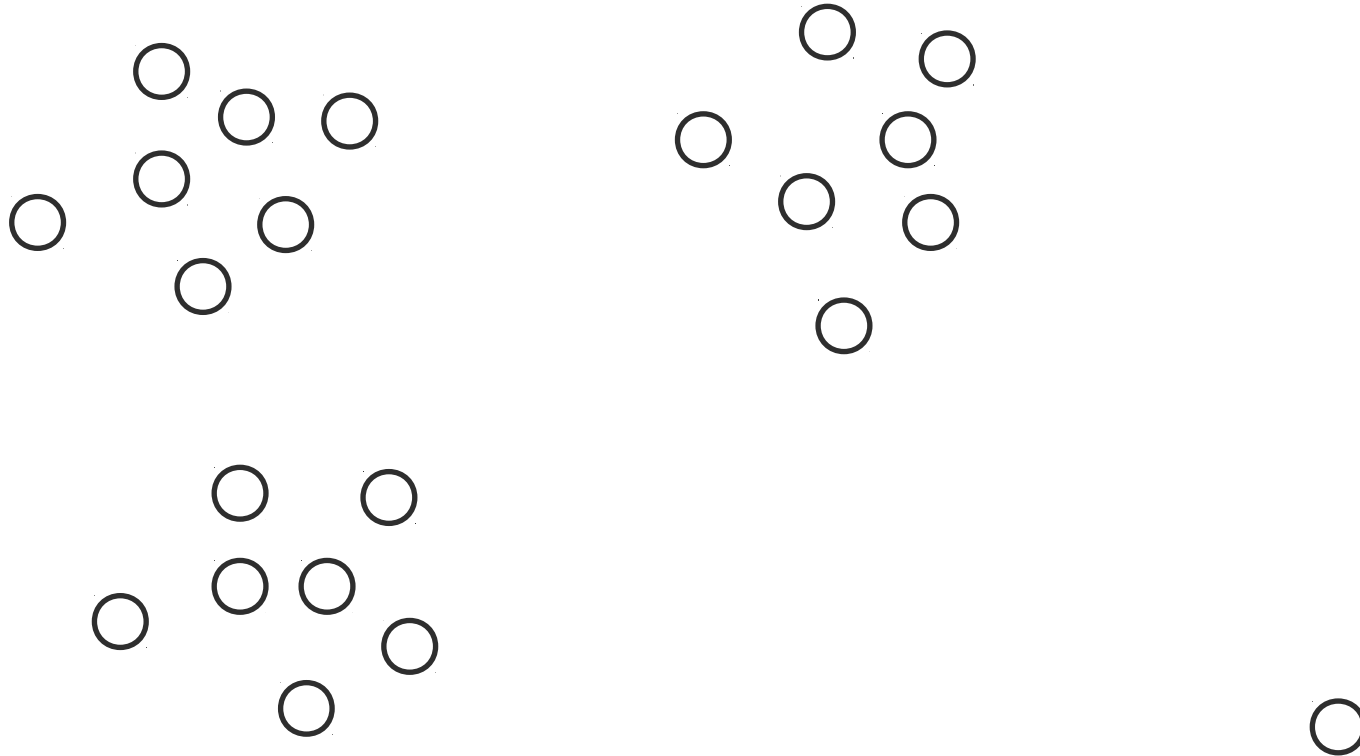
but... sensitive to outliers



but... sensitive to outliers



Here random may work well



k-means++ algorithm

- **interpolate** between the two methods
- let $D(x)$ be the distance between x and the nearest center selected so far
- choose next center **with probability proportional to**

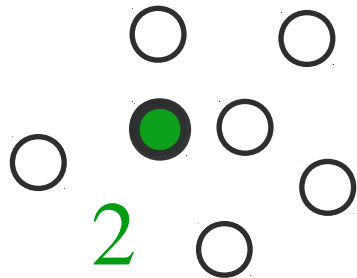
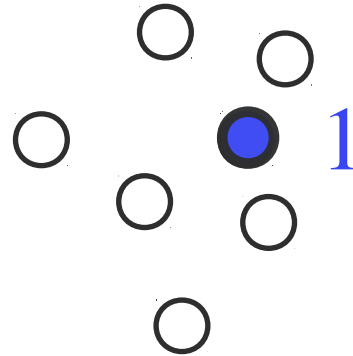
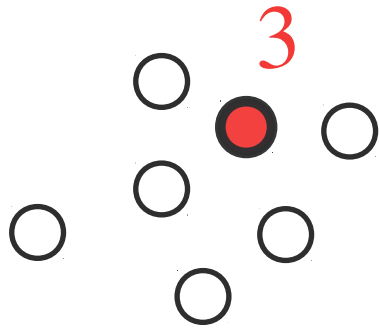
$$(D(x))^a = D^a(x)$$

- ✦ $a = 0$ random initialization
- ✦ $a = \infty$ furthest-first traversal
- ✦ $a = 2$ k-means++

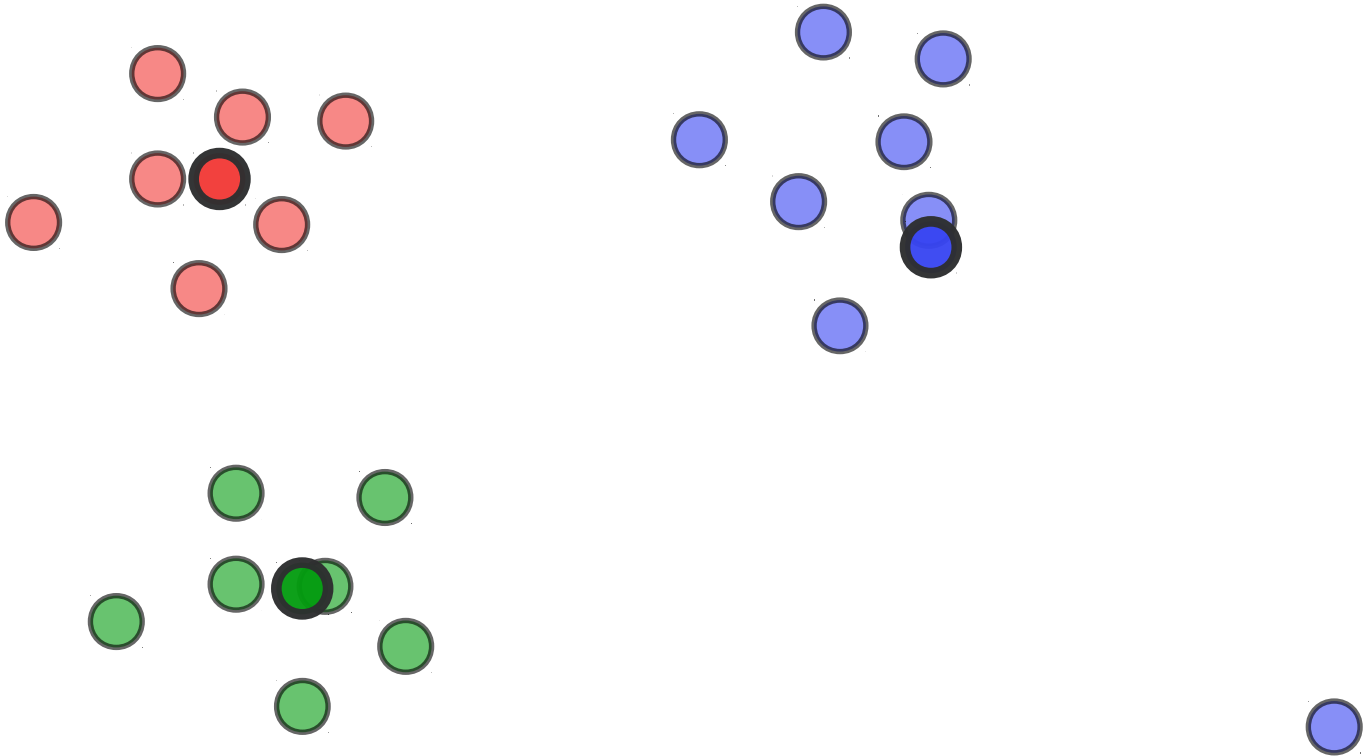
k-means++ algorithm

- initialization phase:
 - choose the first center uniformly at random
 - choose next center with probability proportional to $D^2(x)$
- iteration phase:
 - iterate as in the k-means algorithm until convergence

k-means++ initialization



k-means++ result



k-means++ provable guarantee

- approximation guarantee comes just **from the first iteration** (initialization)
- subsequent iterations **can only improve** cost

Lesson learned

- no reason to use **k-means** and not **k-means++**
- **k-means++** :
 - easy to implement
 - provable guarantee
 - works well in practice

k-means--

- Algorithm 4.1 in [[Chawla & Gionis SDM 2013](#)]