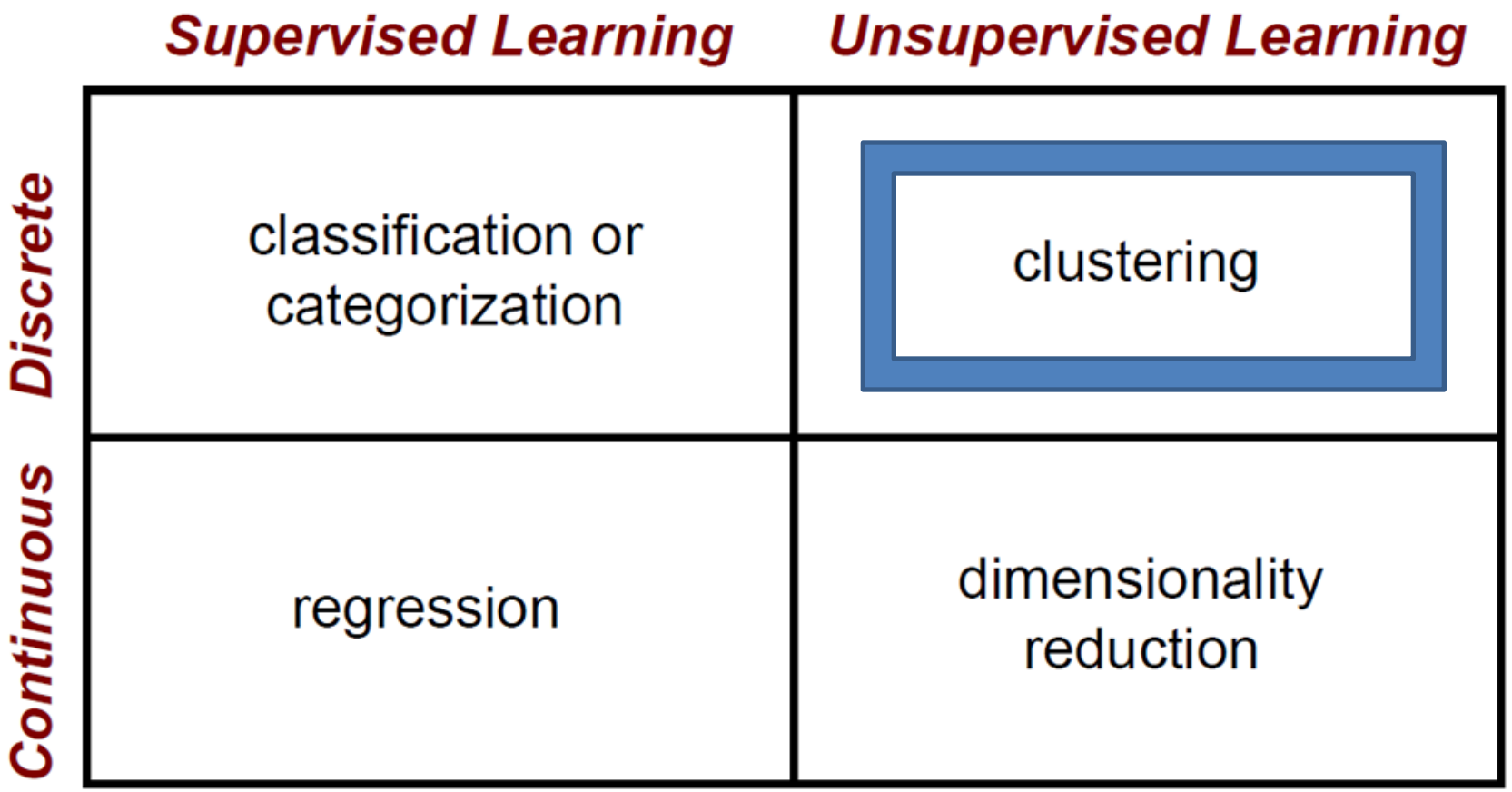
# *Overview of Machine Learning and Pattern Recognition*

*Dipartimento di Automatica e Informatica*
*Politecnico di Torino, Torino, ITALY*

# *4. Machine Learning approaches*

# Summary of ML problems



# Clustering Strategies

- Flat (e.g. k-means)
  - Iteratively re-assign points to the nearest cluster center
- Hierarchical clustering
  - Iteratively merge or split the clusters
- Density based clustering (e.g. mean shift)
  - Estimate modes of pdf

- And many others…

As we go down this chart, the clustering strategies have more tendency to transitively group points even if they are **not** nearby in feature space

# *K*-means algorithm

- For a given cluster assignment *C* of the data points, compute the cluster's centroid $m_k$:

$$m_k = \frac{\sum_{i:C(i)=k} x_i}{N_k},\ k = 1, \ldots, K.$$

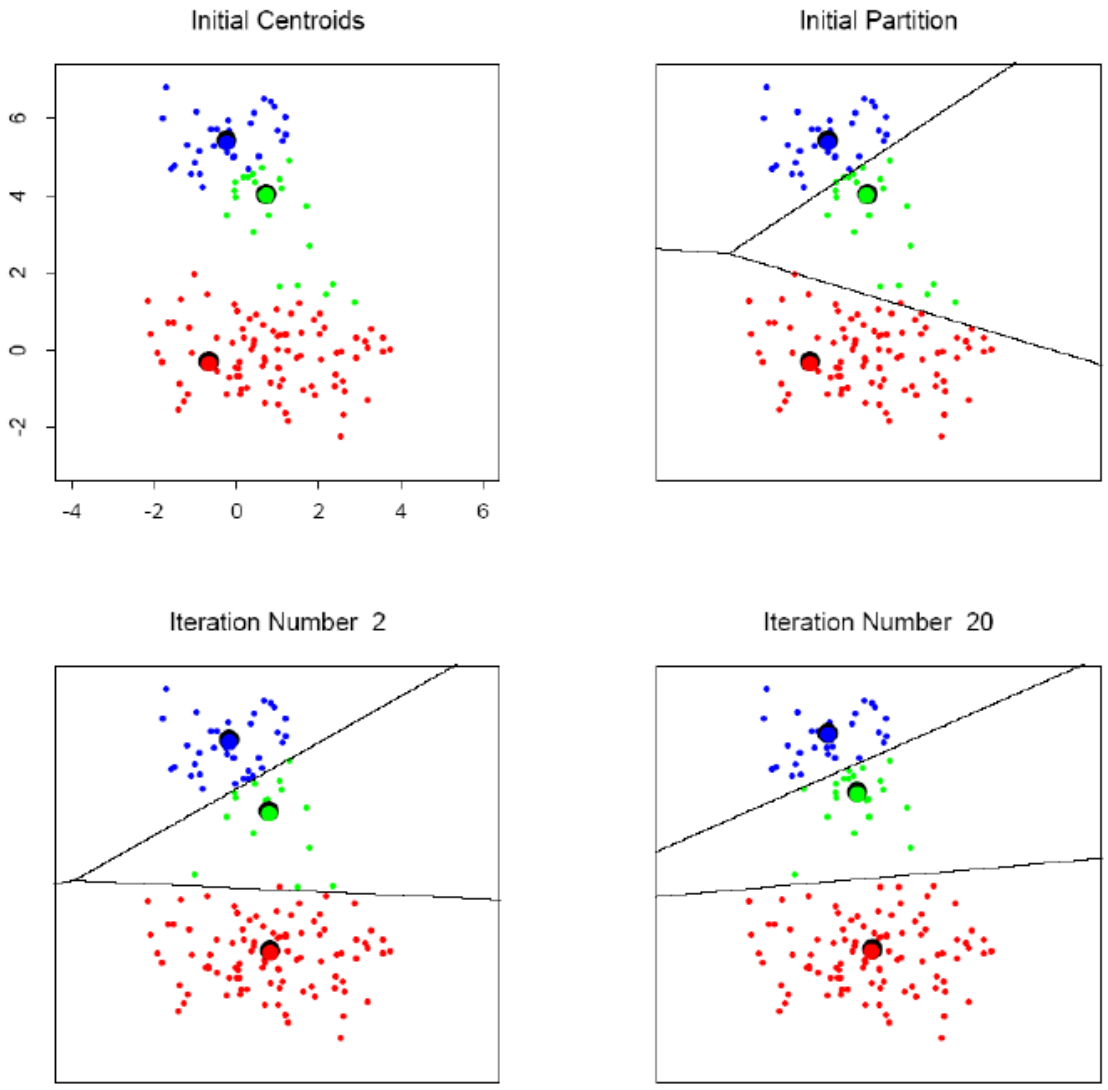
- For a current set of cluster means, assign each observation as:

$$C(i) = \arg\min_{1 \le k \le K} \left\| x_i - m_k \right\|^2,\ i = 1, \ldots, N$$

- Iterate above two steps until convergence

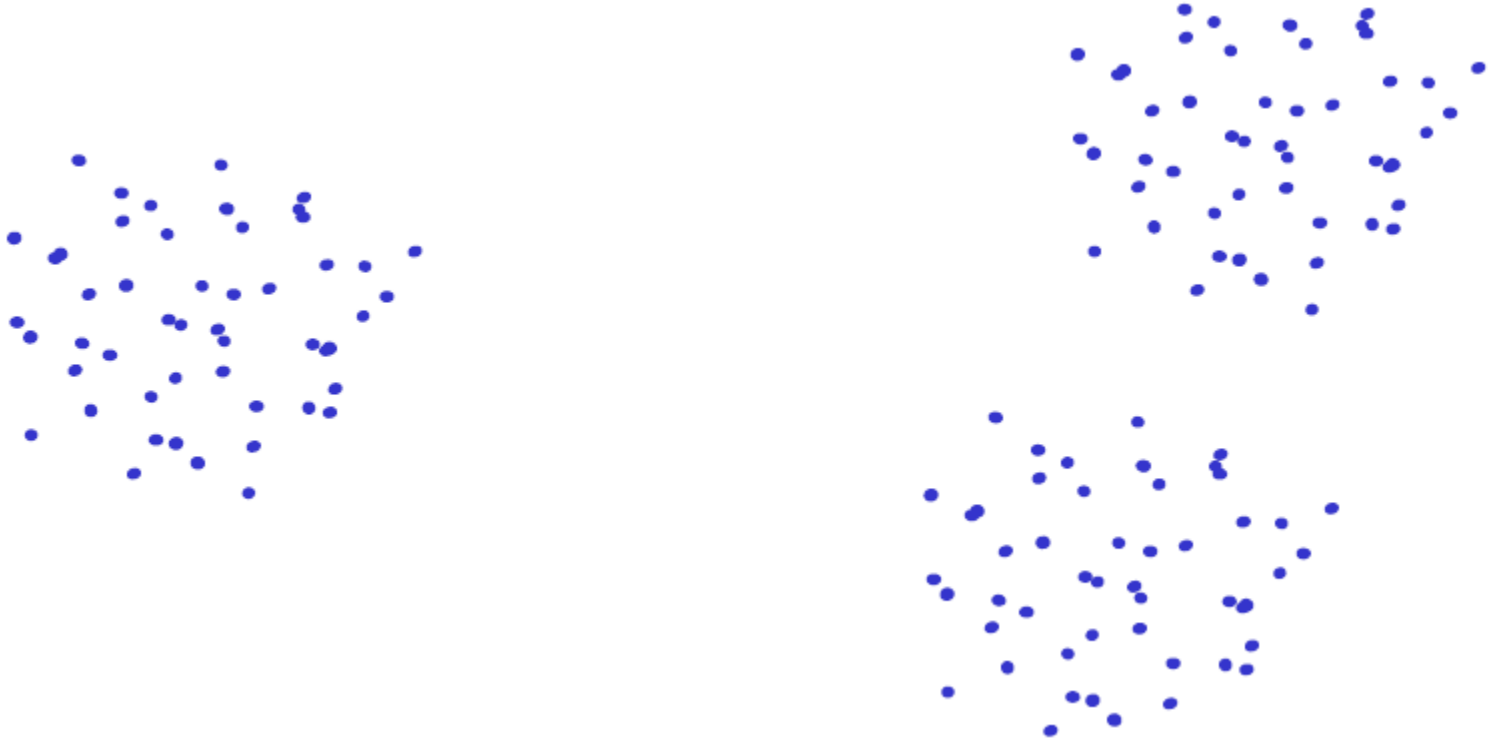
- Works towards the minimization of the **within-cluster scatter** (total sum of point-to-centroid distances)

# *K*-means clustering example



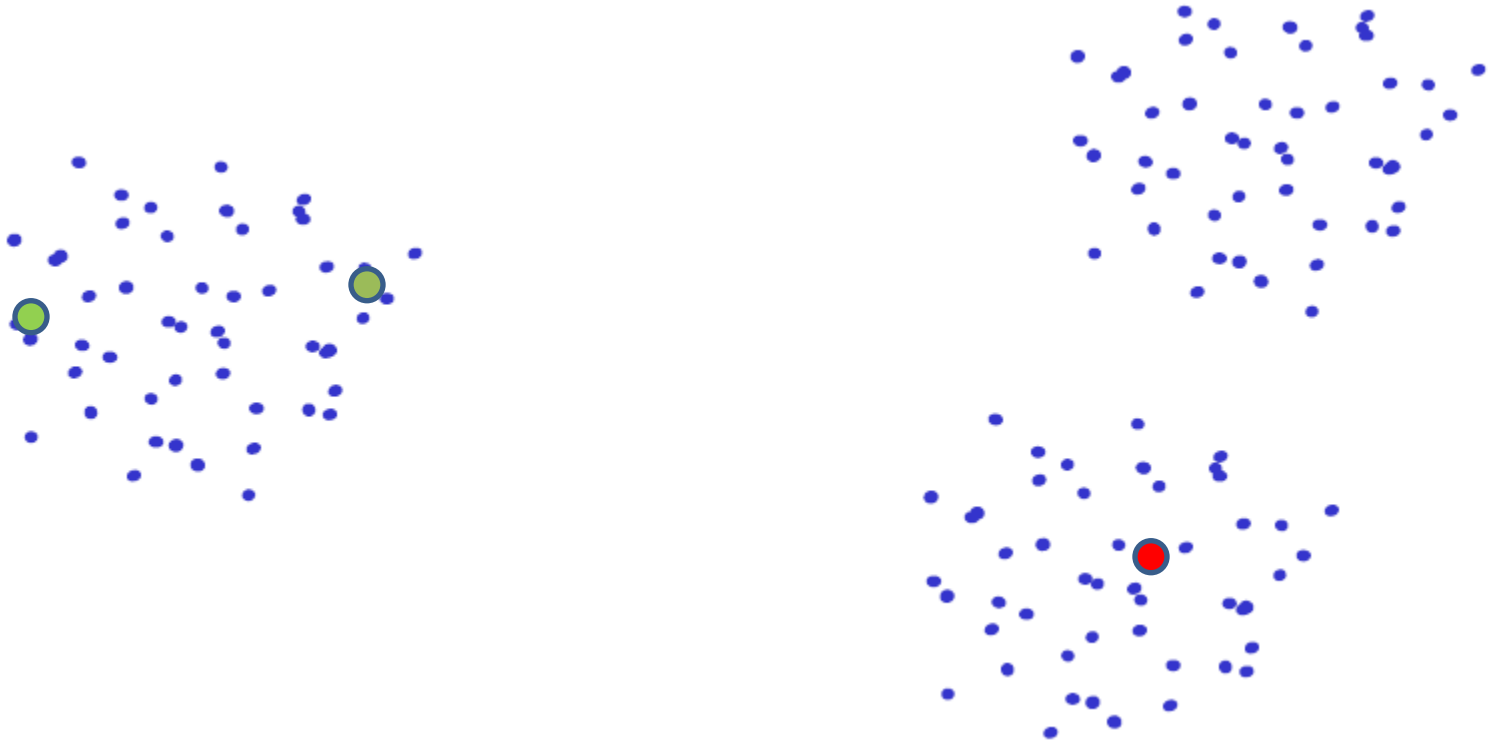
# *K*-means clustering

- Disadvantages
  - Dependent on initialization



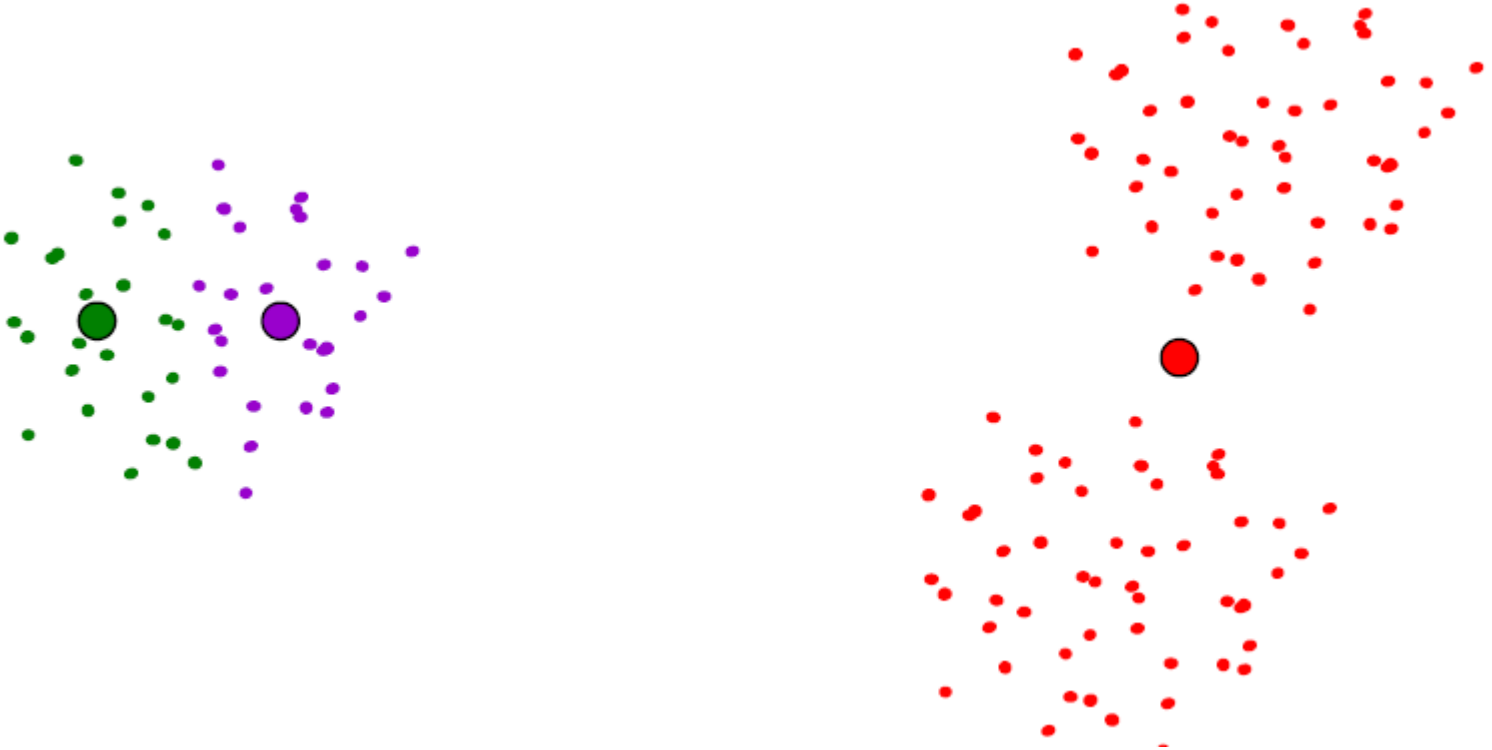
# *K*-means clustering

- Disadvantages
  - Dependent on initialization



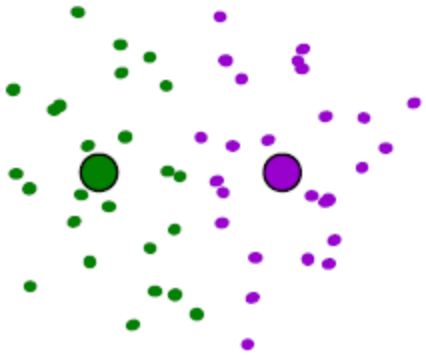
# *K*-means clustering

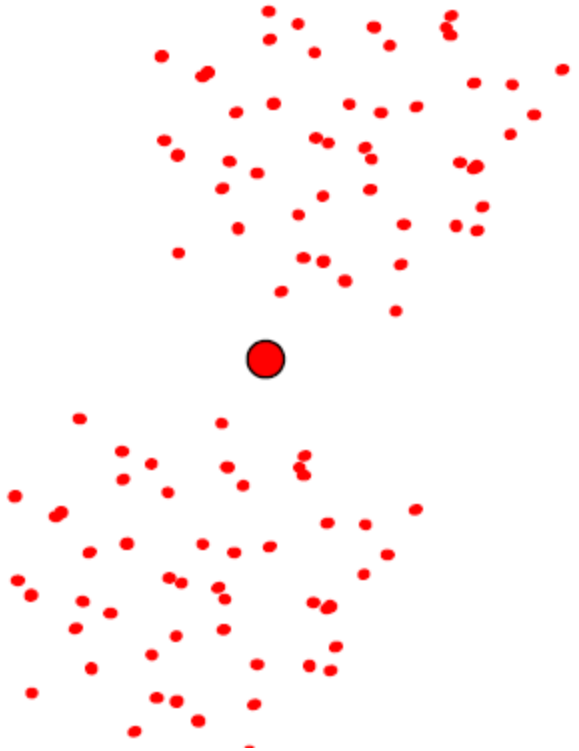
- Disadvantages
  - Dependent on initialization



# *K*-means clustering

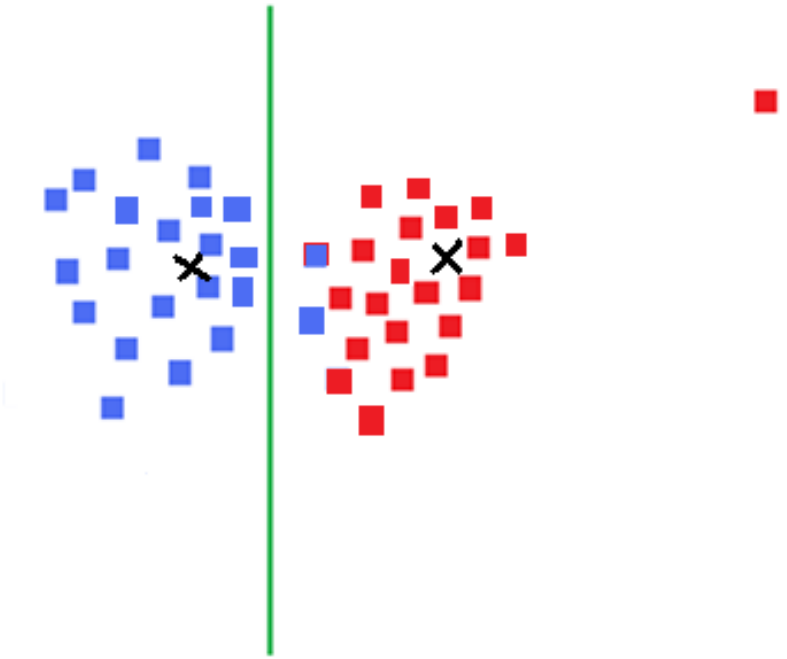
- Disadvantages
  - Dependent on initialization
    - Run the algorithm multiple times





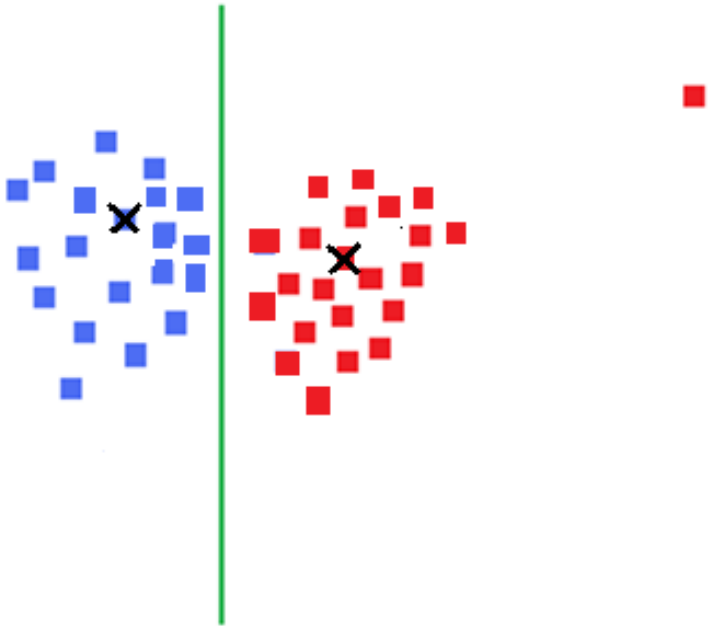
# *K-means clustering*

- Disadvantages
  - Dependent on initialization
    - Run the algorithm multiple times
  - Sensitive to outliers



# *K-means clustering*

- Disadvantages
  - Dependent on initialization
    - Run the algorithm multiple times
  - Sensitive to outliers
    - Use k-medoids

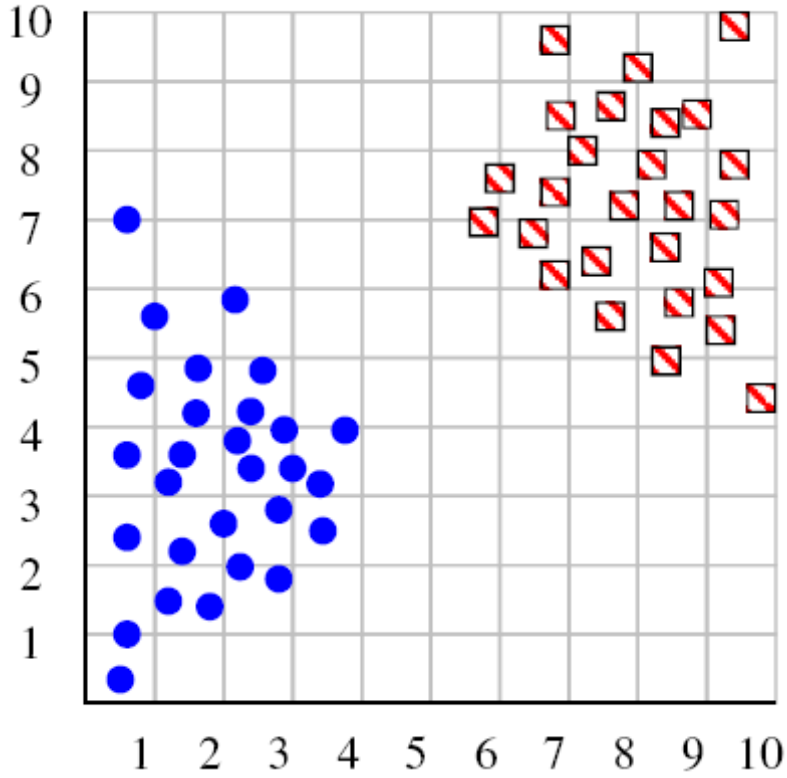


# *K-means clustering*

- Disadvantages
  - Dependent on initialization
    - Run the algorithm multiple times
  - Sensitive to outliers
    - Use k-medoids
  - How to decide k?

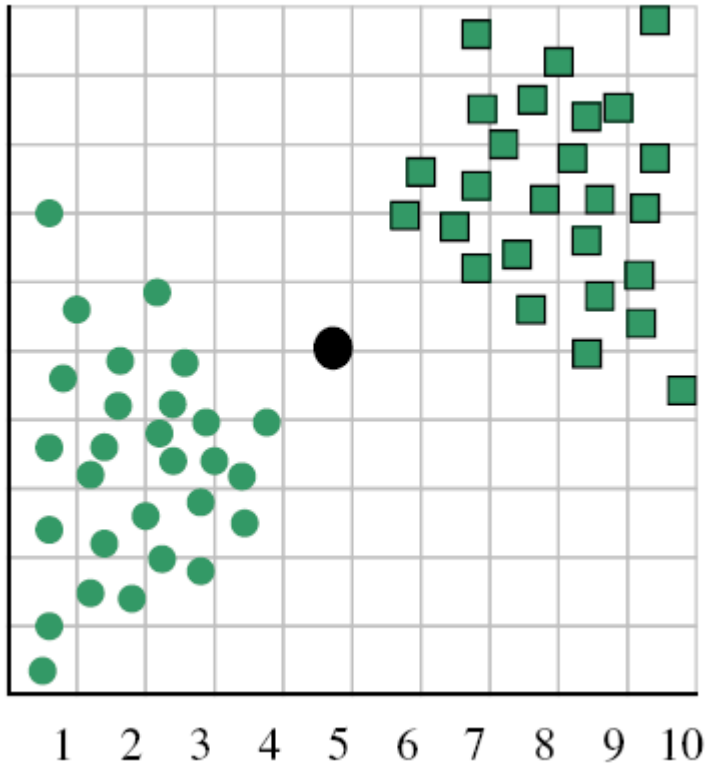
# Deciding K

- Try a couple of K



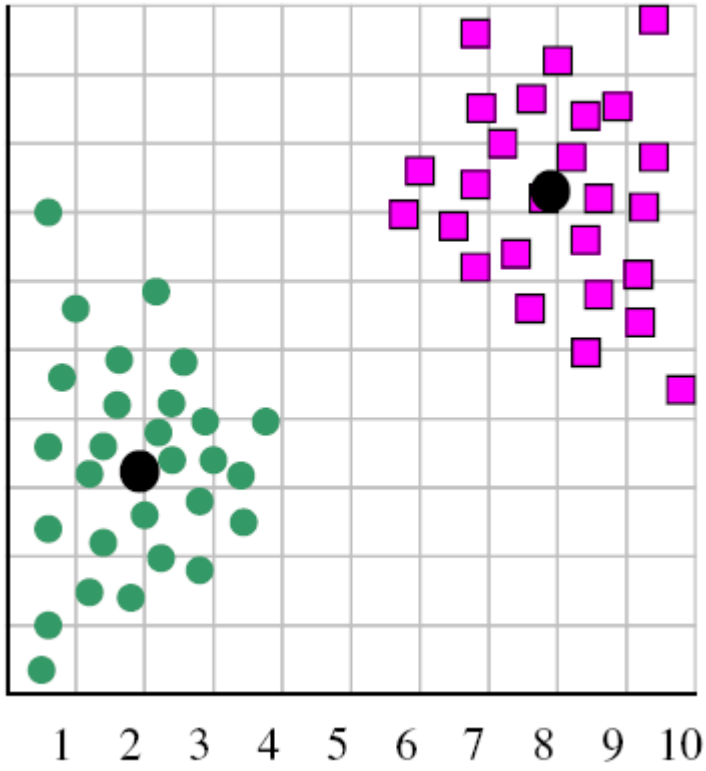
# Deciding K

- When k = 1, the objective function is 873.0



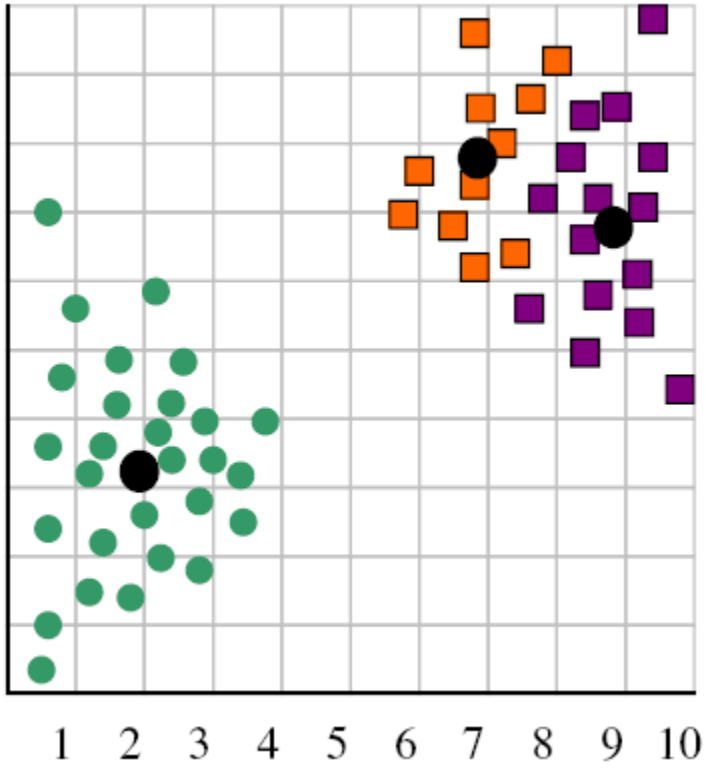
# Deciding K

- When k = 2, the objective function is 173.1



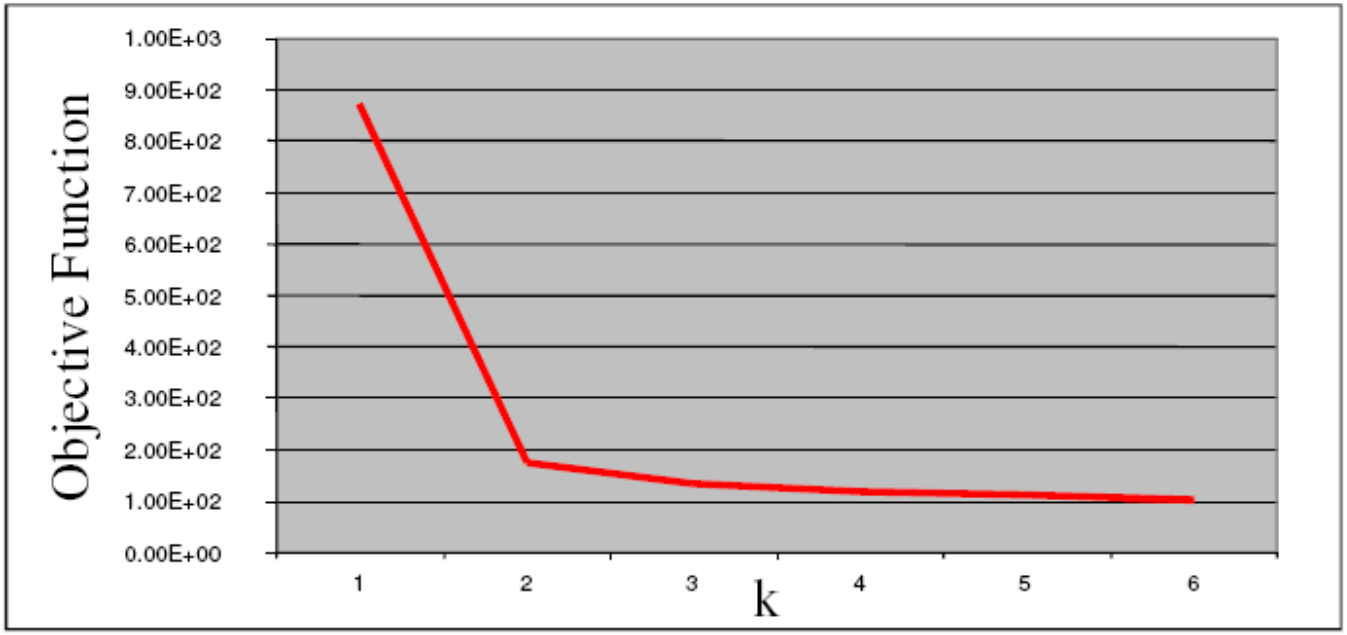
# Deciding K

- When k = 3, the objective function is 133.6



# Deciding K

- We can plot objective function values for k=1 to 6
- The abrupt change at k=2 is highly suggestive of two clusters
- “knee finding” or “elbow finding”
- Note that the results are not always as clear cut!



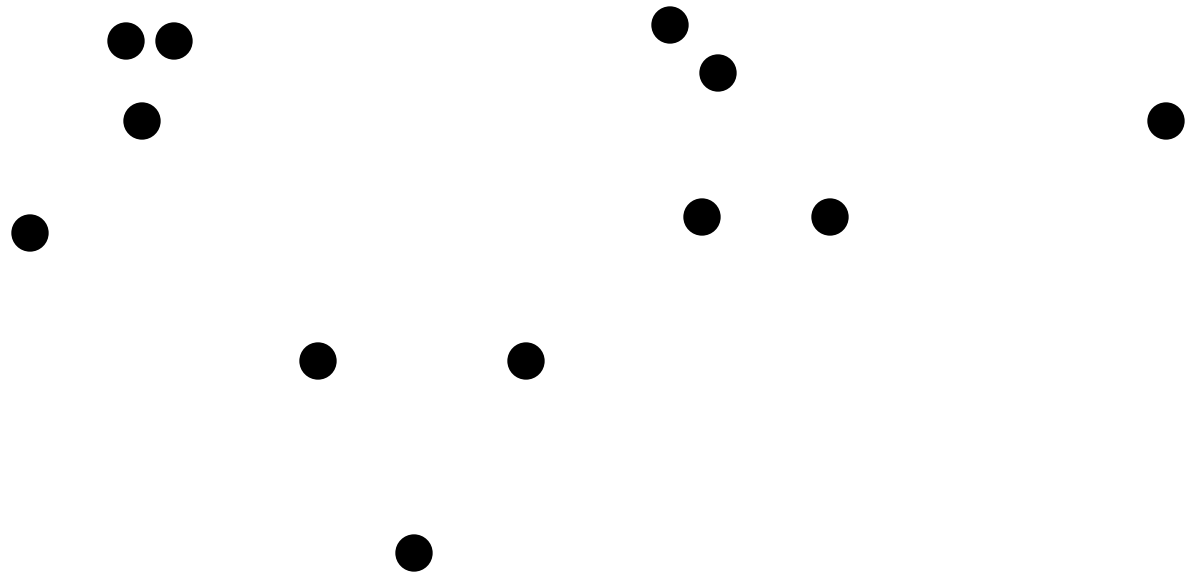
# Hierarchical clustering

- Agglomerative (Bottom-up)
  - Compute all pair-wise pattern-pattern similarity coefficients
  - Place each of *n* patterns into a class of its own
  - Merge the two most similar clusters into one
    - Replace the two clusters into the new cluster
    - Re-compute inter-cluster similarity scores w.r.t. the new cluster
  - Repeat the above step until there are *k* clusters left (*k* can be 1)

Slide credit: Min Zhang

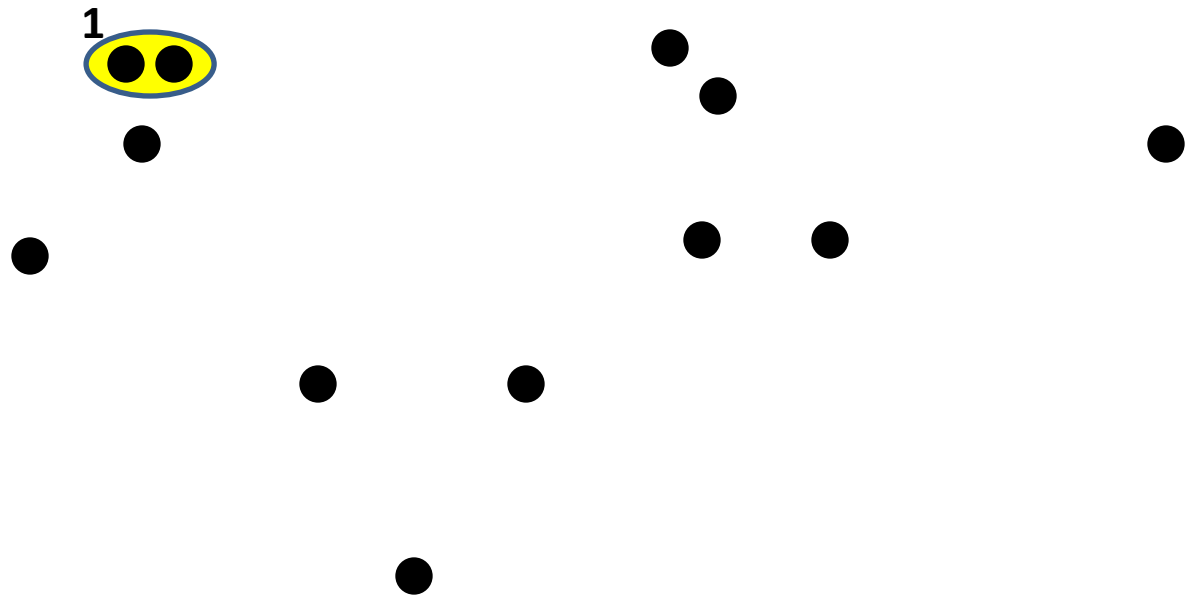
# Hierarchical clustering

- Agglomerative (Bottom up)



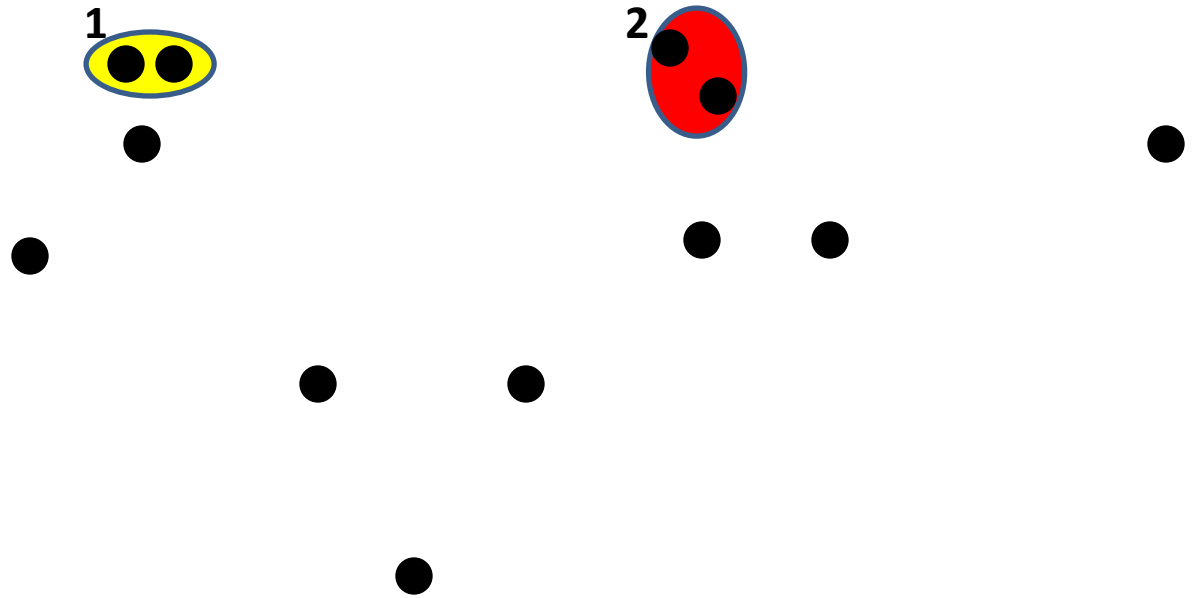
# Hierarchical clustering

- Agglomerative (Bottom up)
- 1st iteration



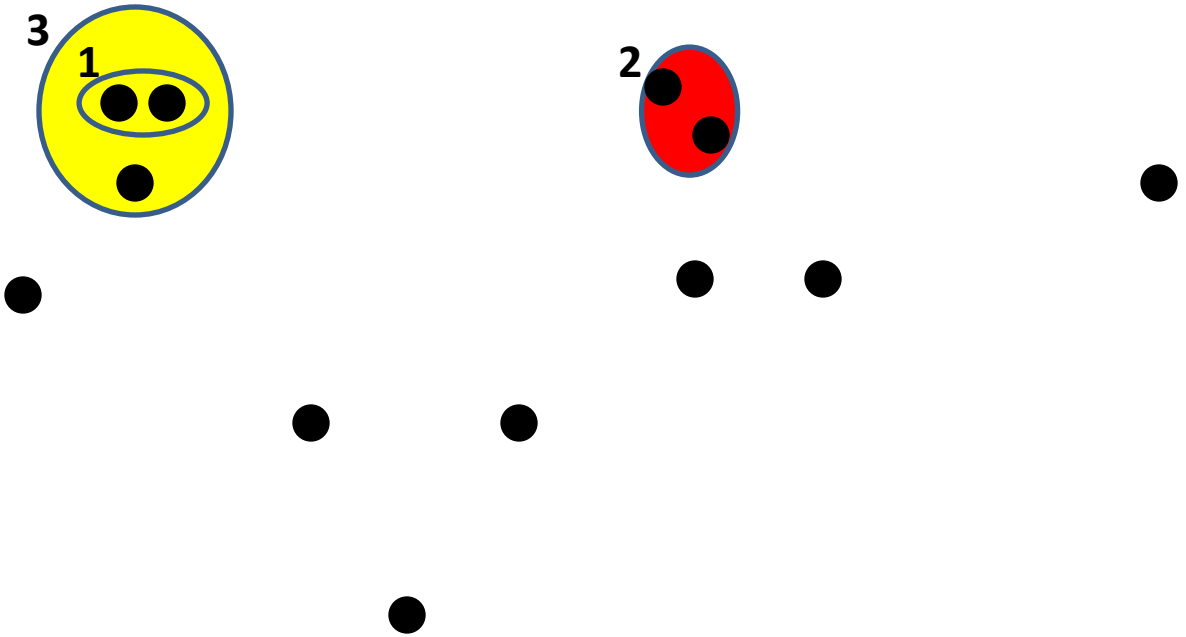
# Hierarchical clustering

- Agglomerative (Bottom up)
- 2nd iteration



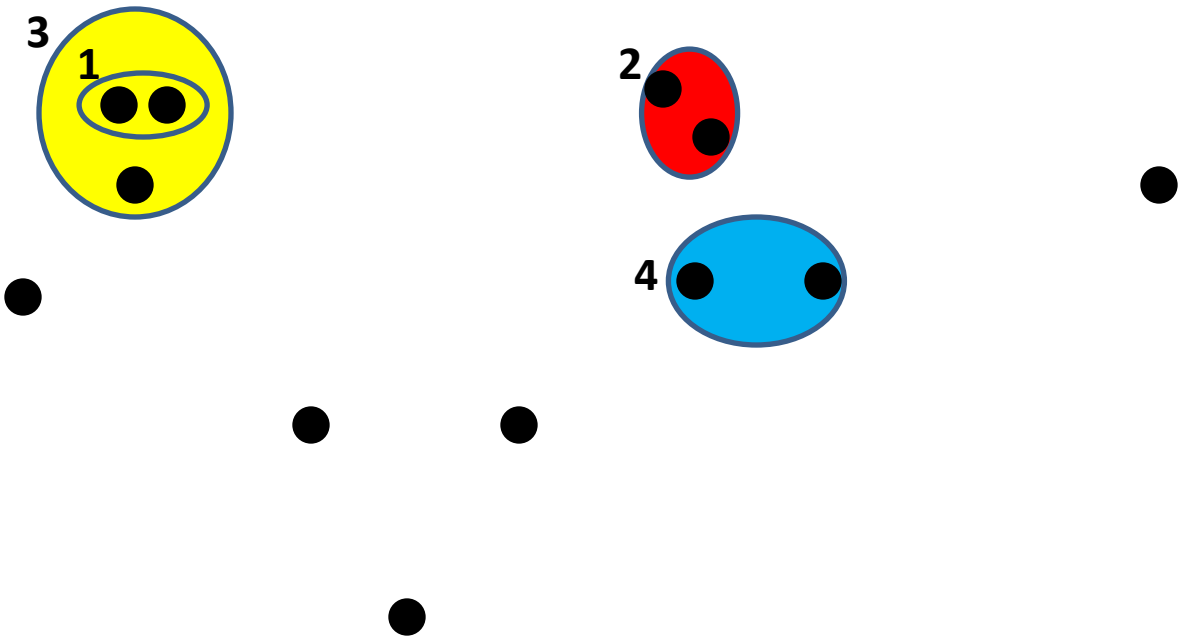
# Hierarchical clustering

- Agglomerative (Bottom up)
- 3rd iteration



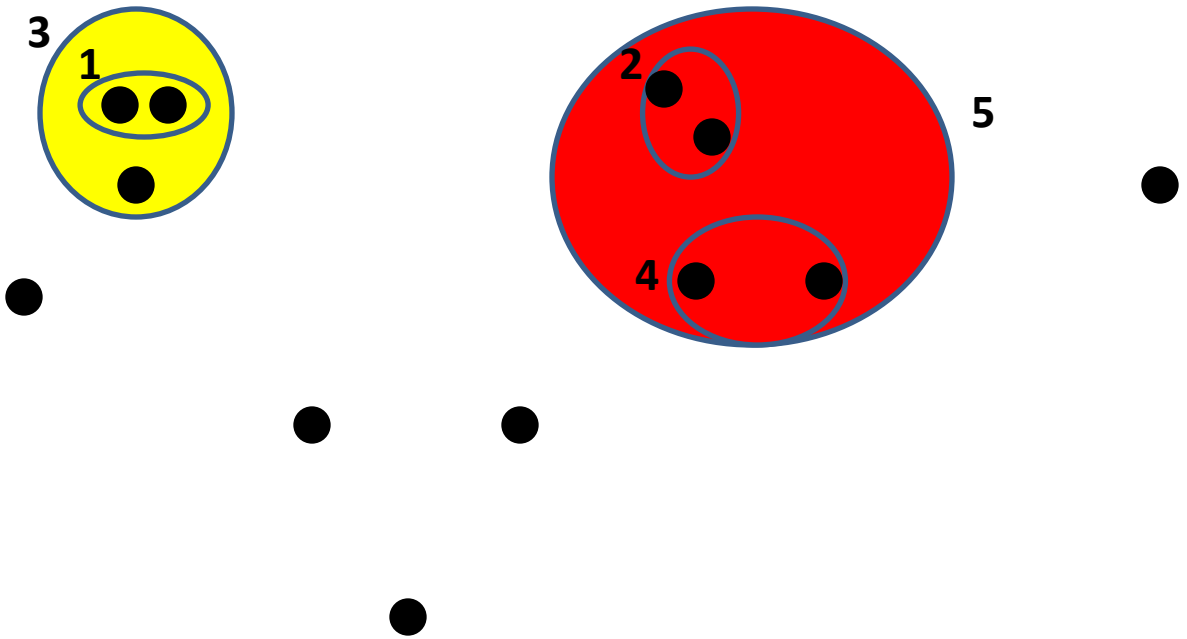
# Hierarchical clustering

- Agglomerative (Bottom up)
- $4^{th}$ iteration



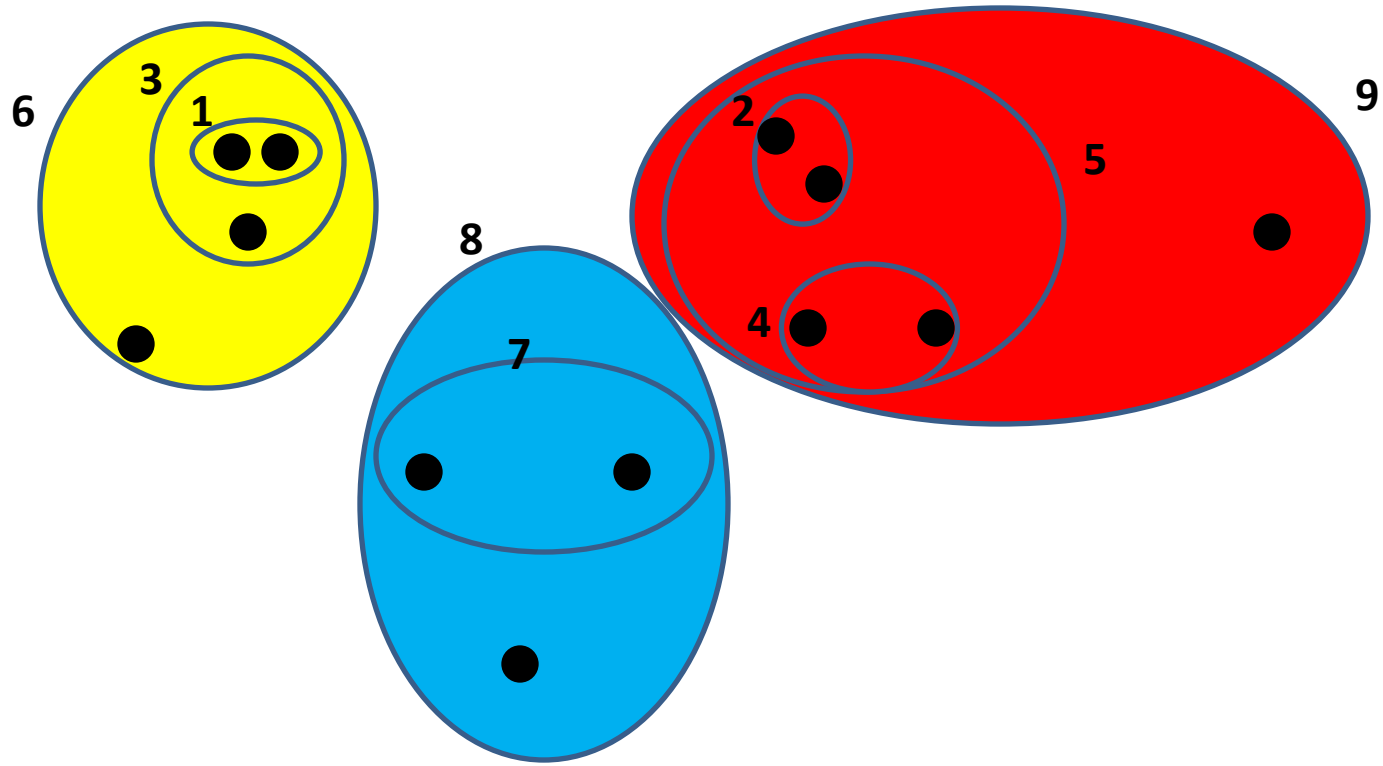
# Hierarchical clustering

- Agglomerative (Bottom up)
- 5th iteration



# Hierarchical clustering

- Agglomerative (Bottom up)



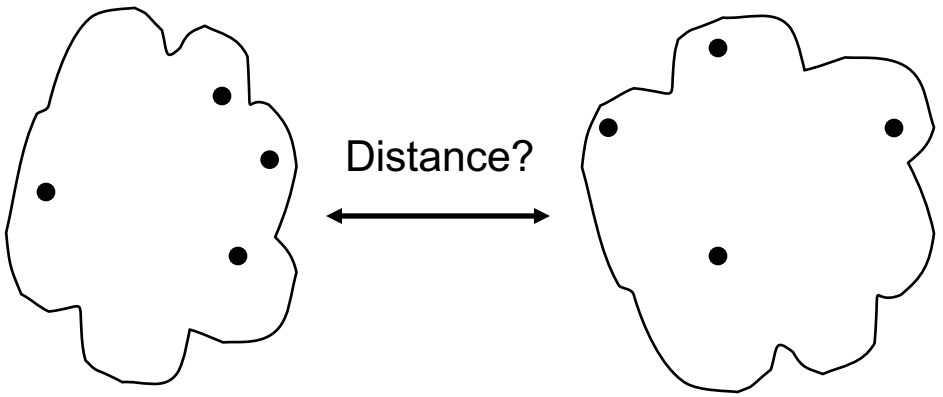

- Finally k clusters left

# Hierarchical clustering

- Divisive (Top-down)
  - Start at the top with all patterns in one cluster
  - The cluster is split using a flat clustering algorithm (e.g. K-means)
  - This procedure is applied recursively until each pattern is in its own singleton cluster

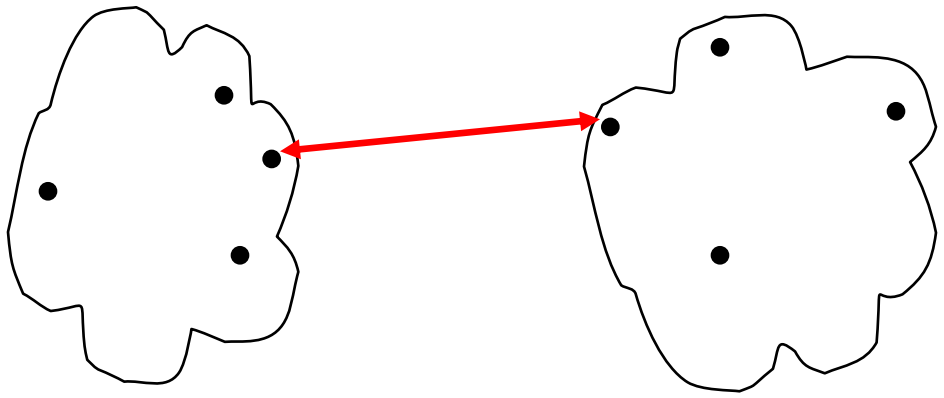
# How to measure clusters' distance?

**Single-link**
**Complete-link**
**Average-link**
**Centroid distance**




Hint: *Distance between clusters* is usually defined on the basis of *distance between objects (data points) in the feature space*

# How to measure clusters' distance?



**Single-link**

**Complete-link**

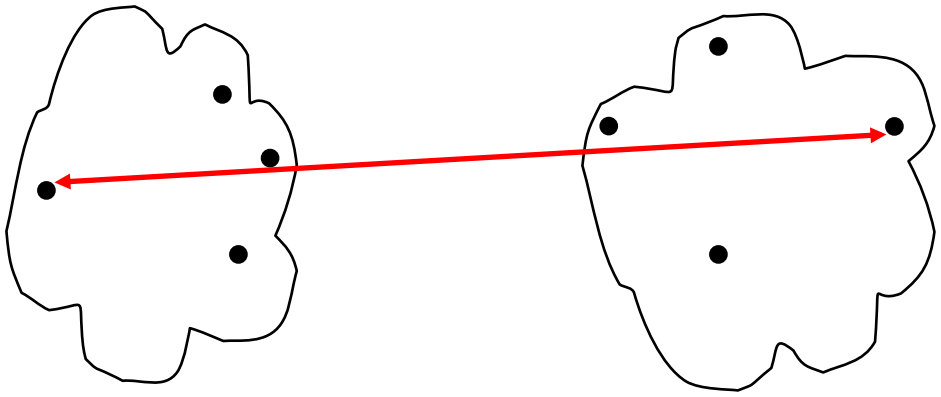
**Average-link**

**Centroid distance**

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$$

The distance between two clusters is represented by the distance of the *closest pair of data objects* belonging to different clusters.

# How to measure clusters' distance?



**Single-link**

**Complete-link**

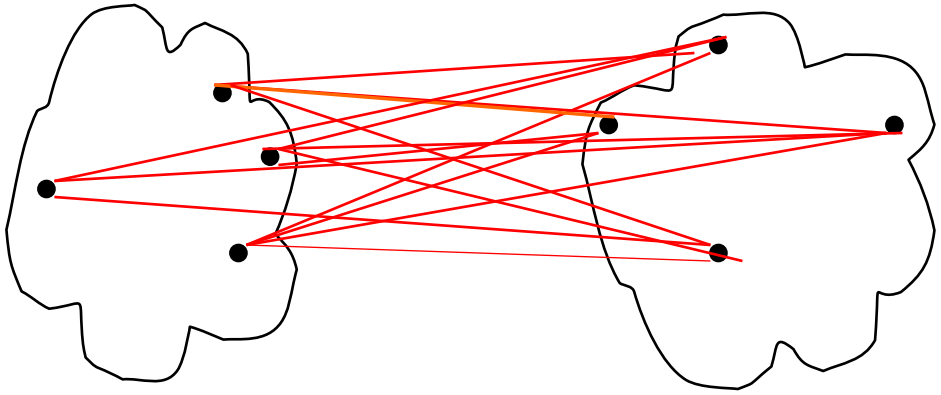
**Average-link**

**Centroid distance**

$$d_{\min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$$

The distance between two clusters is represented by the distance of the *farthest pair of data objects* belonging to different clusters.

# How to measure clusters' distance?



Single-link

Complete-link

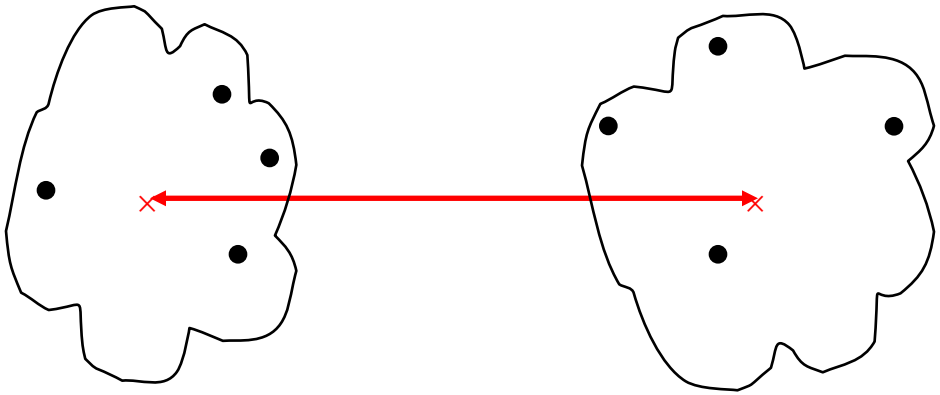
Average-link

Centroid distance

$$d_{\min}(C_i, C_j) = \underset{p \in C_i, q \in C_j}{avg} d(p, q)$$

The distance between two clusters is represented by the *average* distance of *all pairs of data objects* belonging to different clusters.

# How to measure clusters' distance?



$m_i$,$m_j$ are the means of $C_i$, $C_j$,

**Single-link**

**Complete-link**

**Average-link**

**Centroid distance**

$$d_{mean}(C_i, C_j) = d(m_i, m_j)$$

The distance between two clusters is represented by the distance between *the means of the cluters.*

# Which is better?

- Each method has both advantages and disadvantages; application-dependent, **single-link and complete-link** are the most common methods

- Single-link
  - Can find irregular-shaped clusters
  - Sensitive to outliers

- Complete-link, Average-link, and Centroid distance
  - Robust to outliers
  - Prefer spherical clusters

# Mean Shift clustering

- builds upon the concept of **kernel density estimation (KDE)**.
- The data points (i.e. the feature vectors of the data to be clusterized) are interpreted as they were sampled from a probability distribution. KDE is a method to estimate the underlying probability distribution.

# Kernel Density Estimation

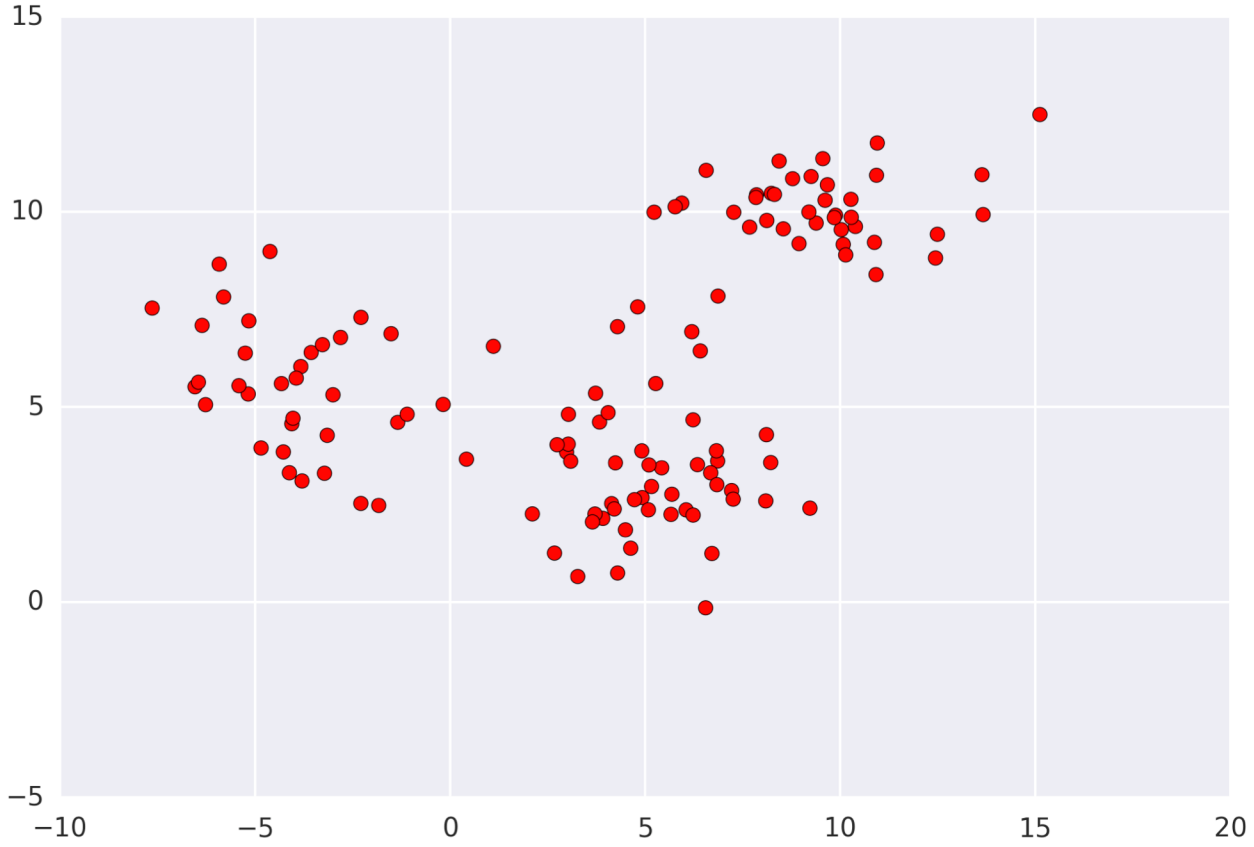
- It works by placing a **kernel** (i.e. a weighting function) on each point of the dataset in the feature space. Adding all of the individual kernels up generates a probability surface (e.g., **density function**).

$$\widehat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

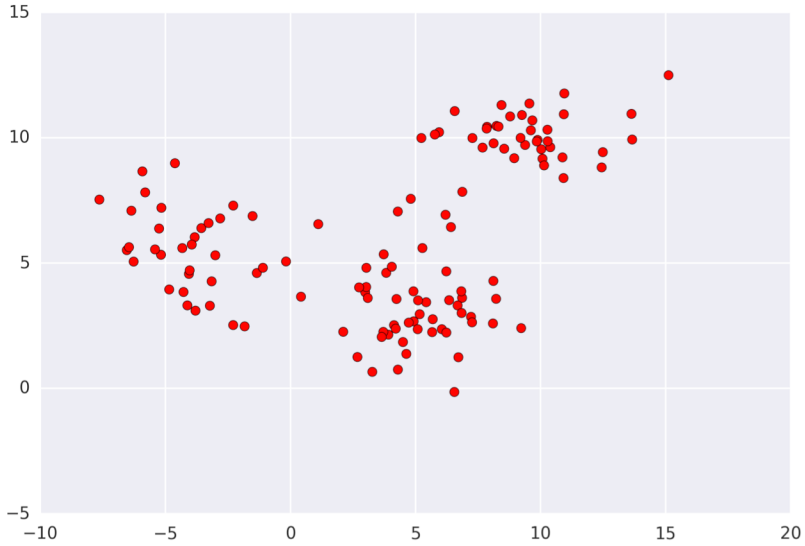
- The most popular kernel function is the Gaussian kernel

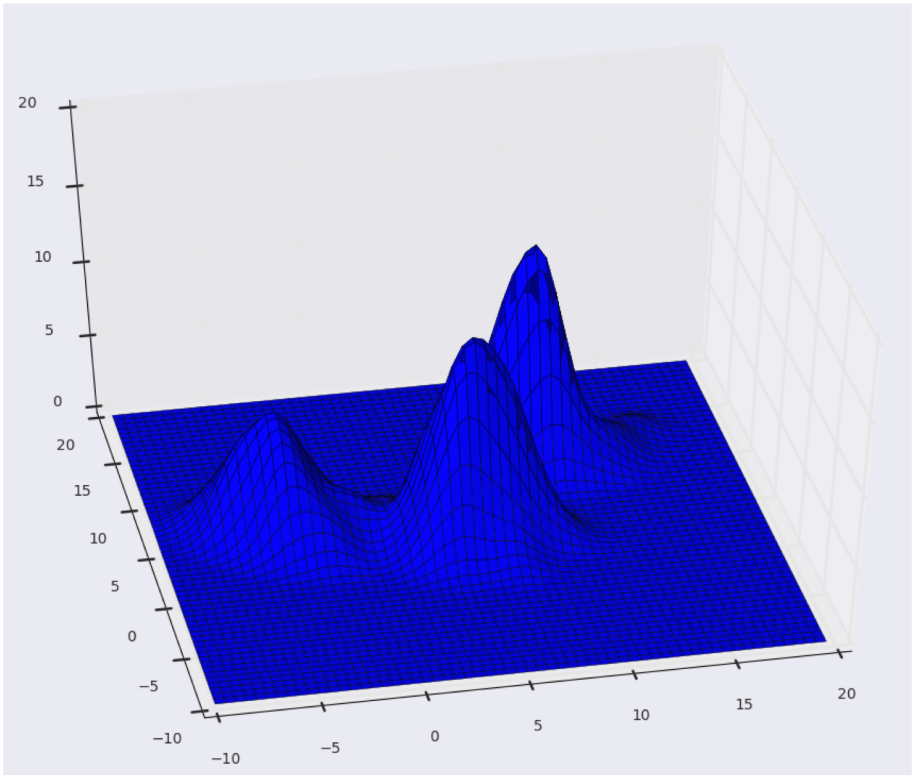
$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}.$$

# KDE: example

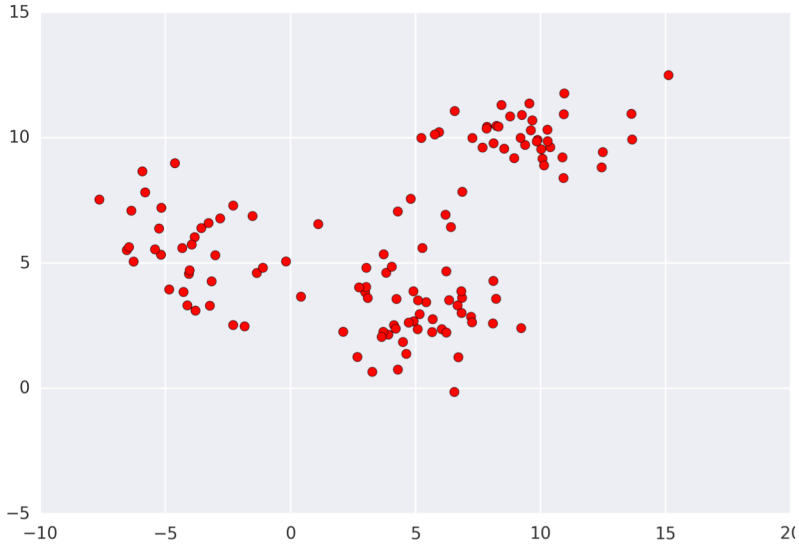


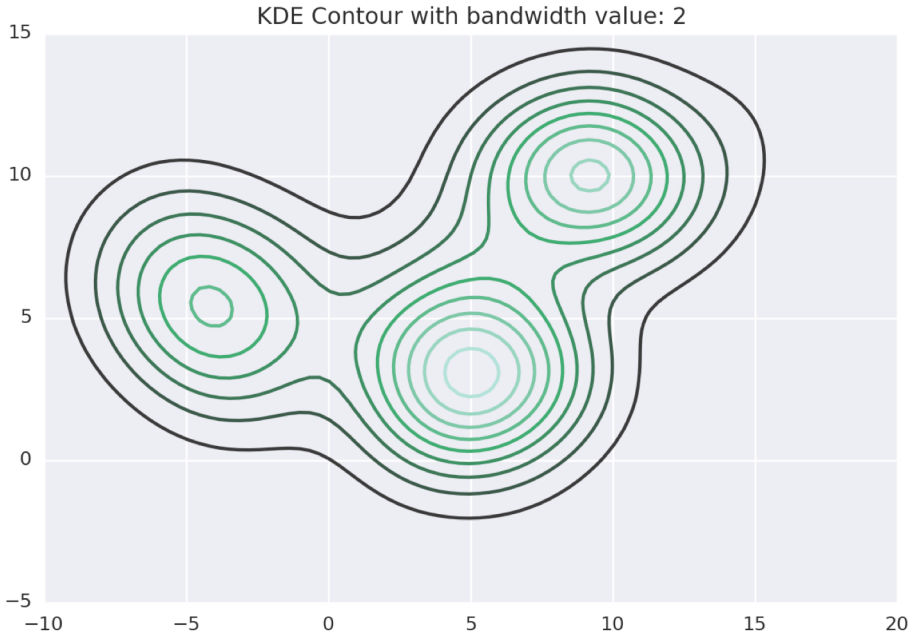
# KDE: example



# KDE: example

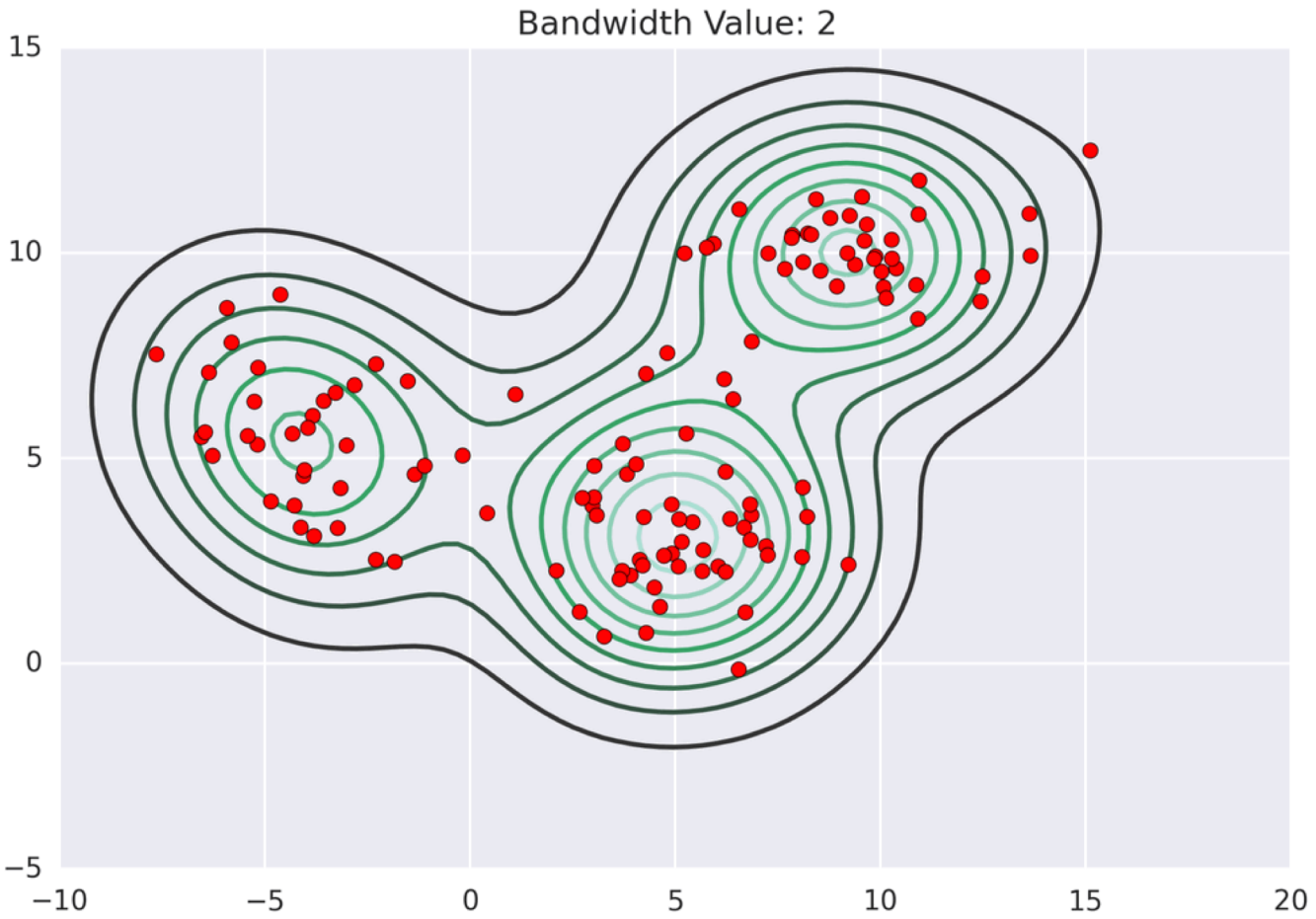

15
10
5
0
−5
−10
−5
0
5
10
15


KDE Contour with bandwidth value: 2
15
10
5
0
−5
−10
−5
0
5
10
15
20

# From KDE to clustering

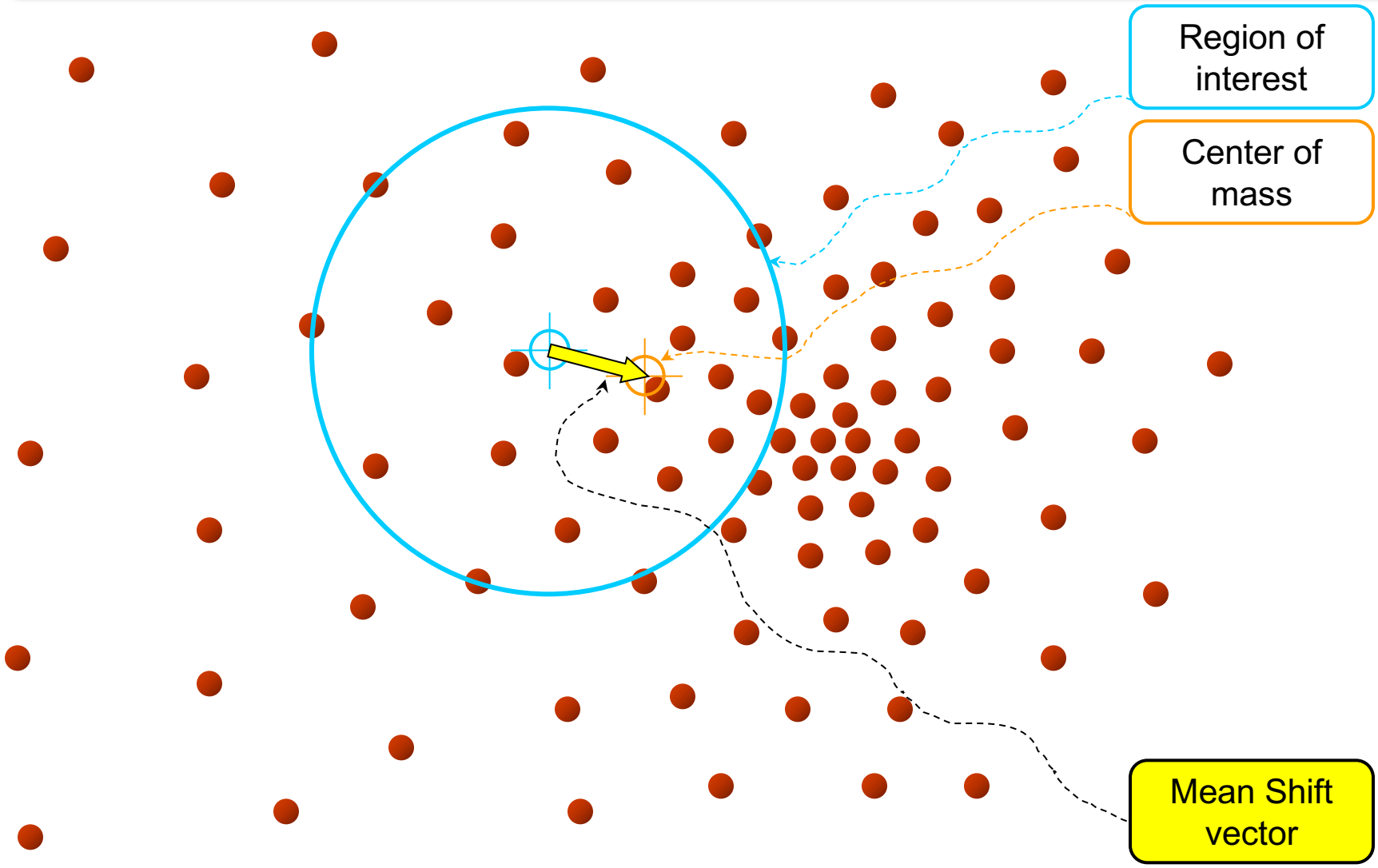
- **Mean shift** exploits this KDE idea by imagining what the points would do if they all climbed up hill to the nearest peak on the KDE surface. It does so by iteratively shifting each point uphill until it reaches a peak.
- Hence, all the points are clustered depending on the peak they are shifted to.

# Mean shift: example

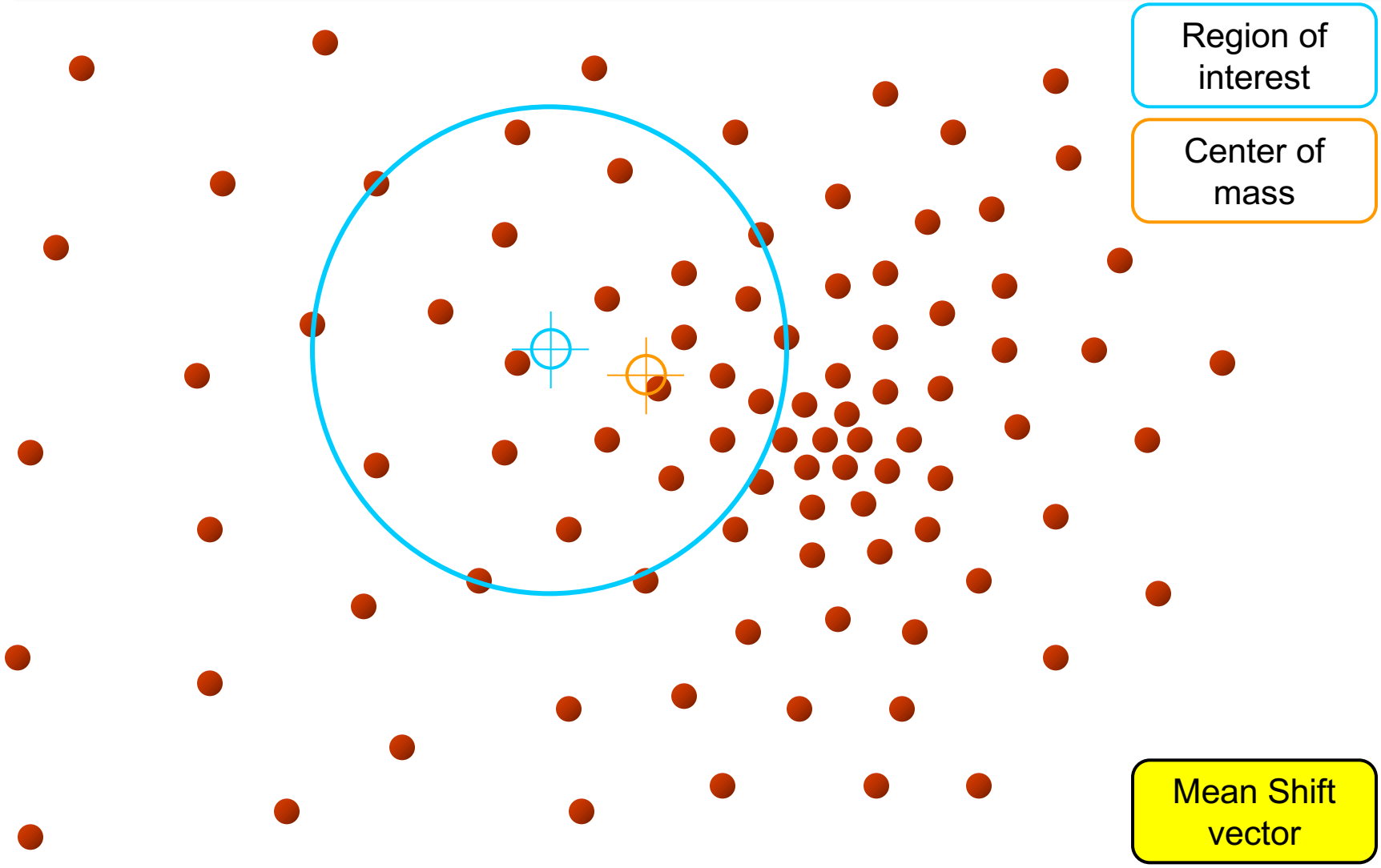



Three peaks → three clusters

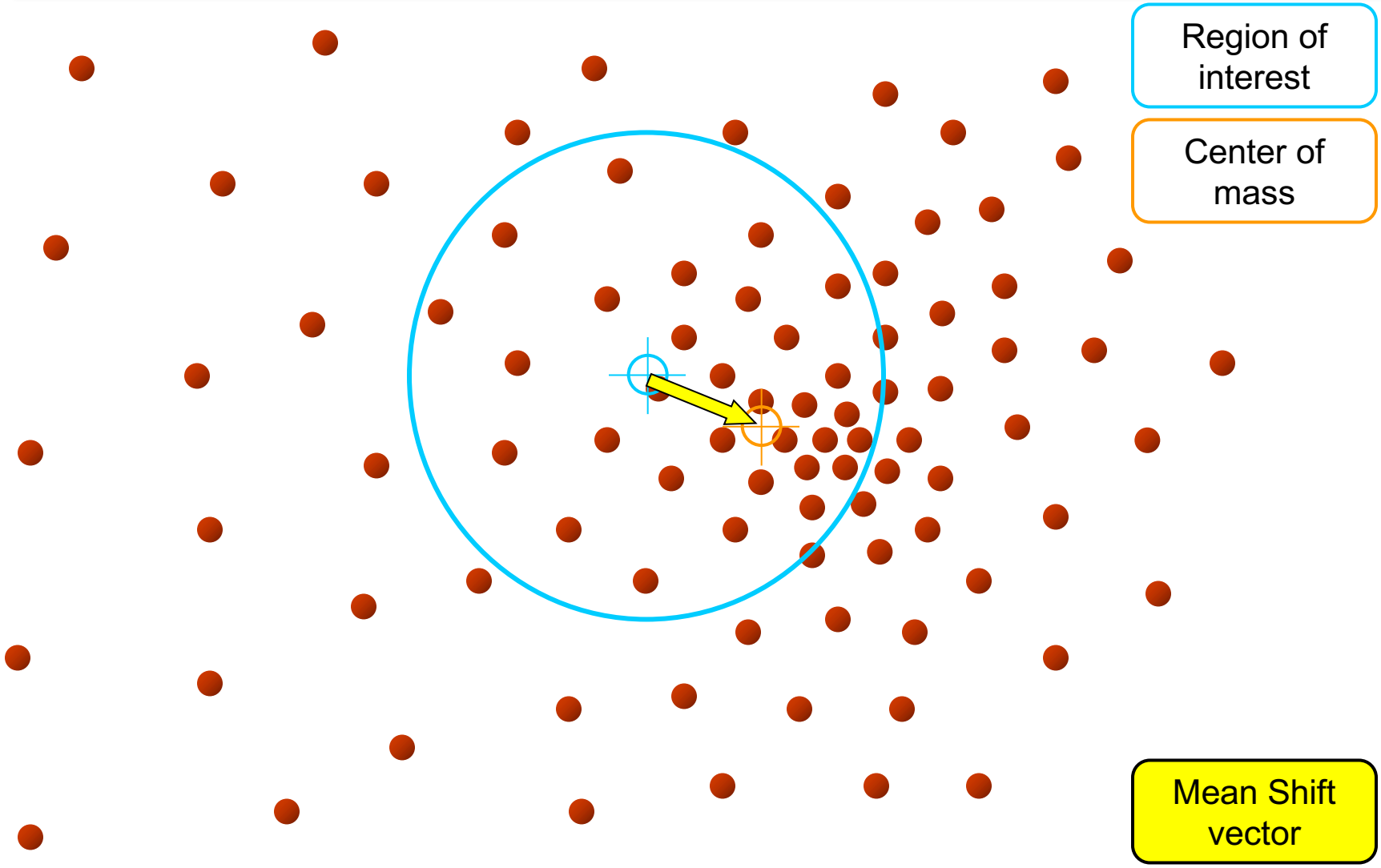
# Mean shift



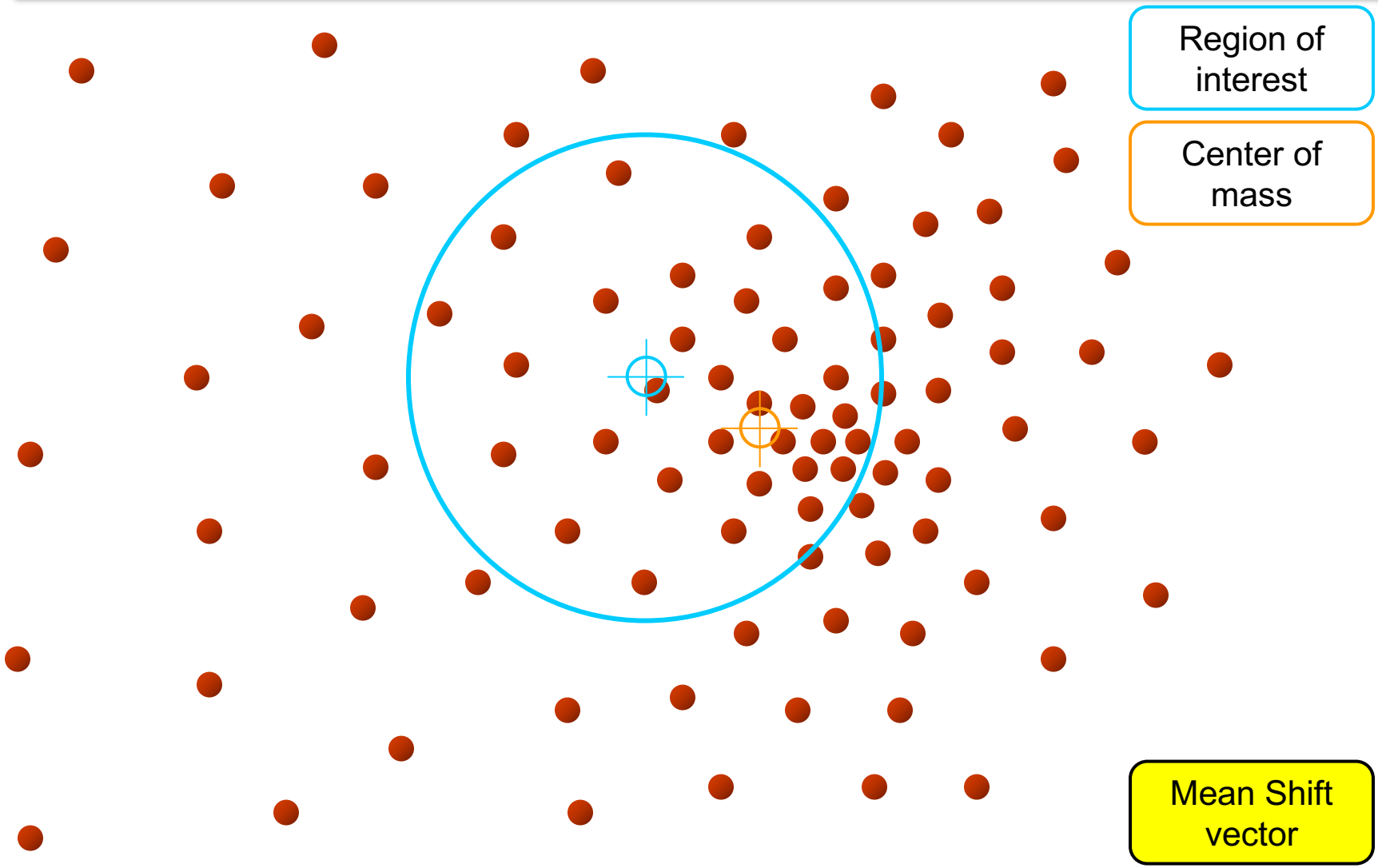
# Mean shift



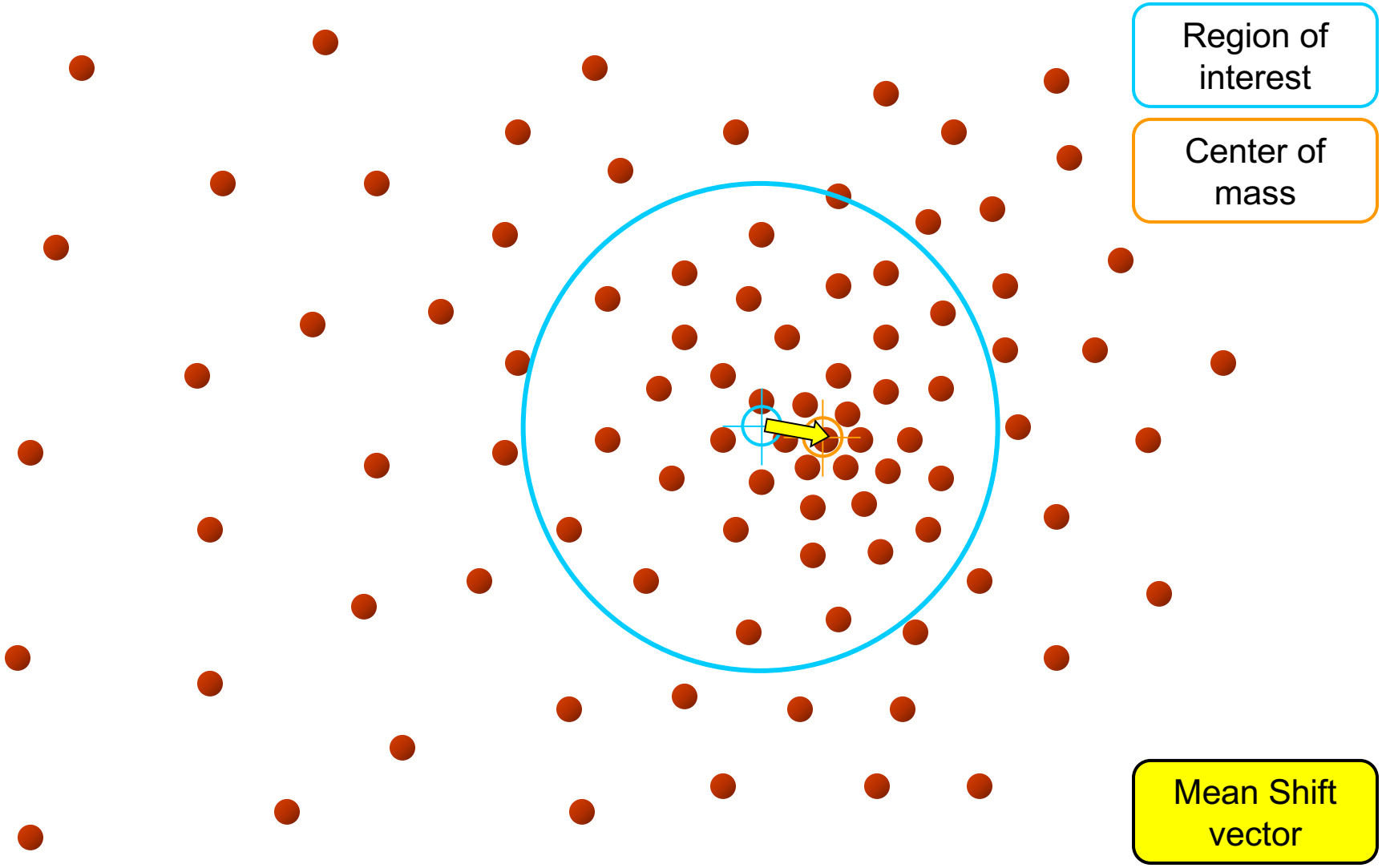
# Mean shift



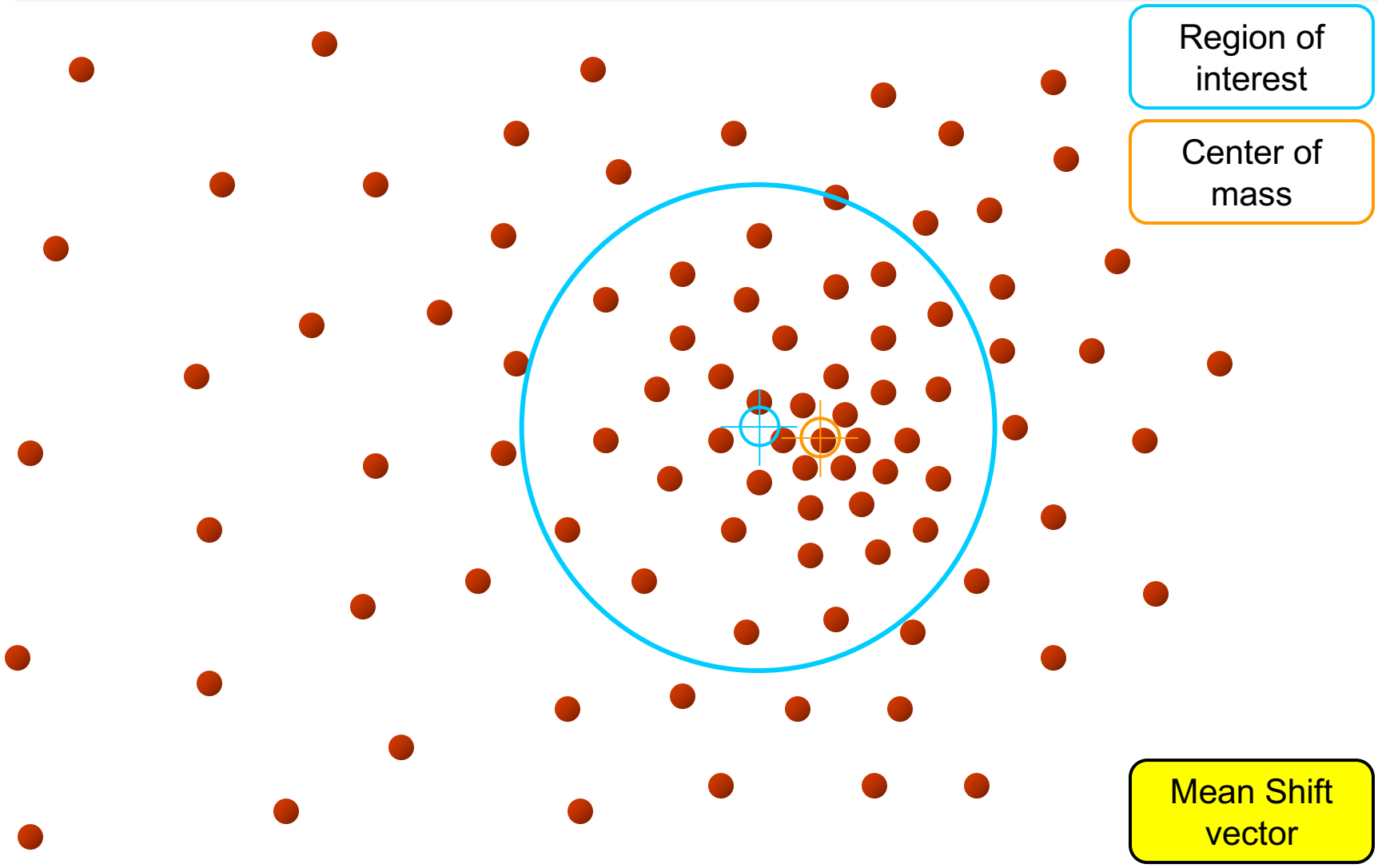
# Mean shift



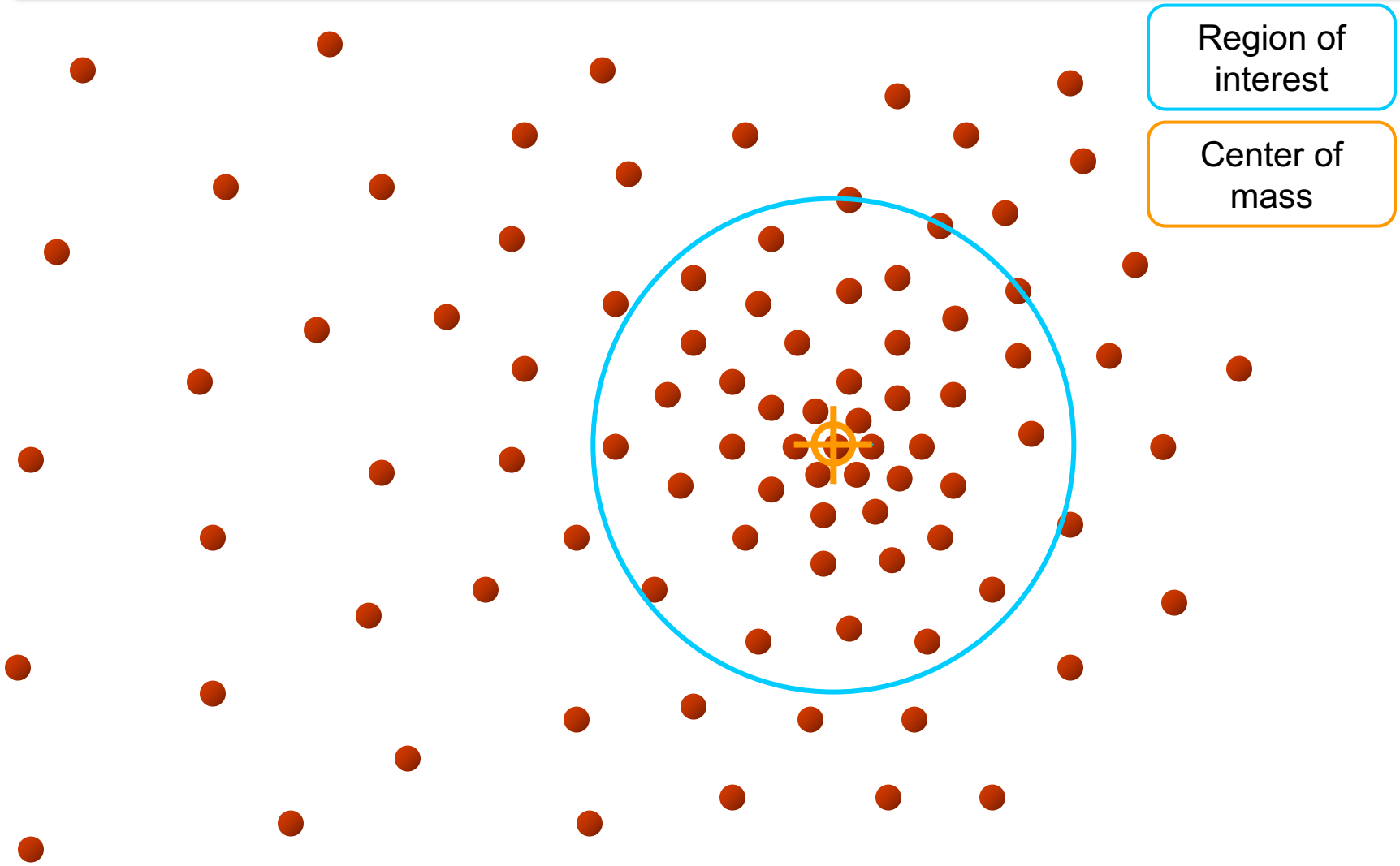
# Mean shift



# Mean shift



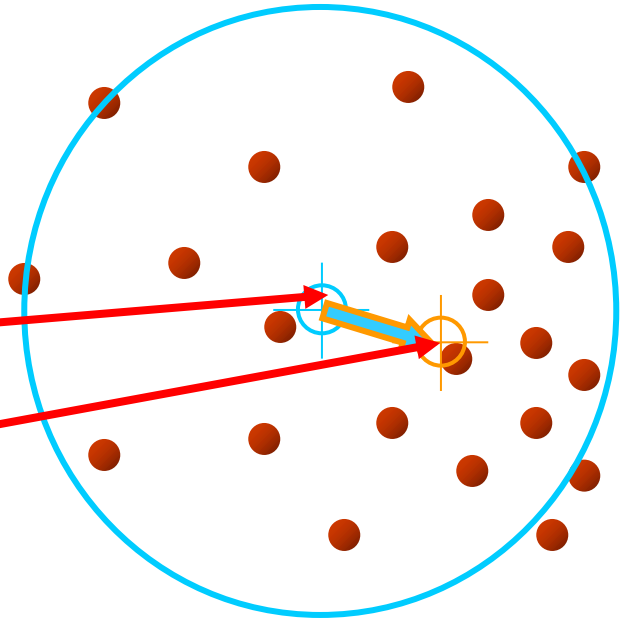
# Mean shift



# Simple Mean Shift procedure

- Compute mean shift vector m(x). The mean shift vector always points toward the direction of the maximum increase of density
- Translate the Kernel window by m(x)

$$\mathbf{m}(\mathbf{x}) = \left[ \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left( \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)}{\sum_{i=1}^{n} g\left( \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)} \right] - \mathbf{x}$$



# Mean shift clustering

- The mean shift algorithm seeks *modes* of the given set of points

1. Choose a kernel and a bandwidth value
2. For each point:
   a) Center a (density estimator) window on that point
   b) Compute the mean shift vector *m* of the data in the search window
   c) Move the density estimation window by *m*
   d) Repeat (b,c) until convergence
3. Assign points that lead to the same mode to the same cluster

# Mean shift clustering

- Pros
  - There is no need to apriori decide the number of clusters
  - Robust to outliers
  - Quite flexible, no assumptions on the data distribution
- Cons
  - Computationally intensive
  - Not feasible for high-dimensional dataset
  - Results depend on kernel/bandwith