9.54
Class 13

Unsupervised learning
Clustering

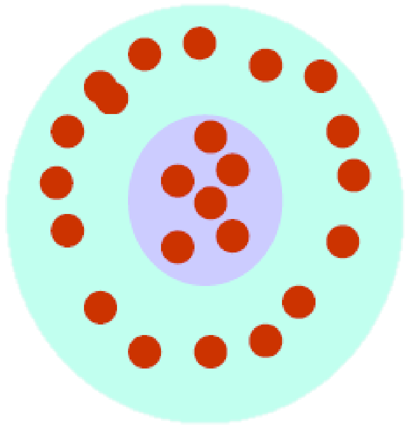
Shimon Ullman + Tomaso Poggio
Danny Harari + Daneil Zysman + Darren Seibert

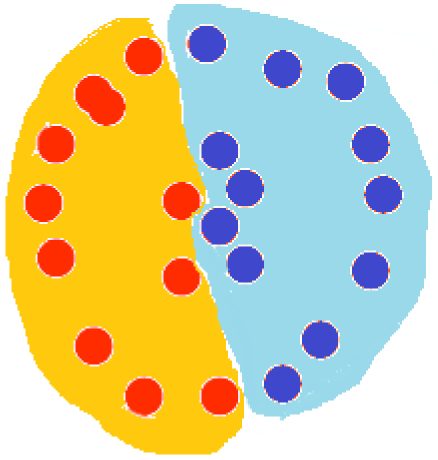
Outline

- Introduction to clustering
- K-means
- Bag of words (dictionary learning)
- Hierarchical clustering
- Competitive learning (SOM)

What is clustering?

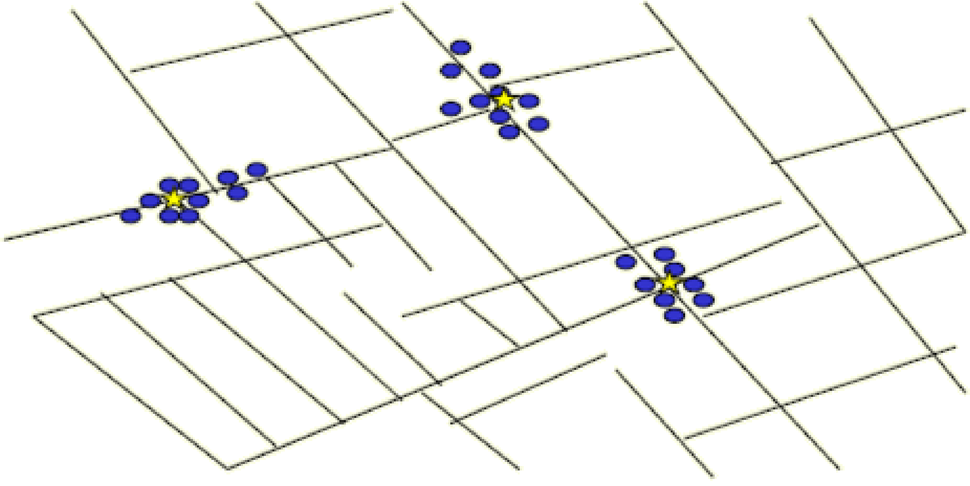
- The organization of unlabeled data into similarity groups called clusters.
- A cluster is a collection of data items which are “similar” between them, and “dissimilar” to data items in other clusters.





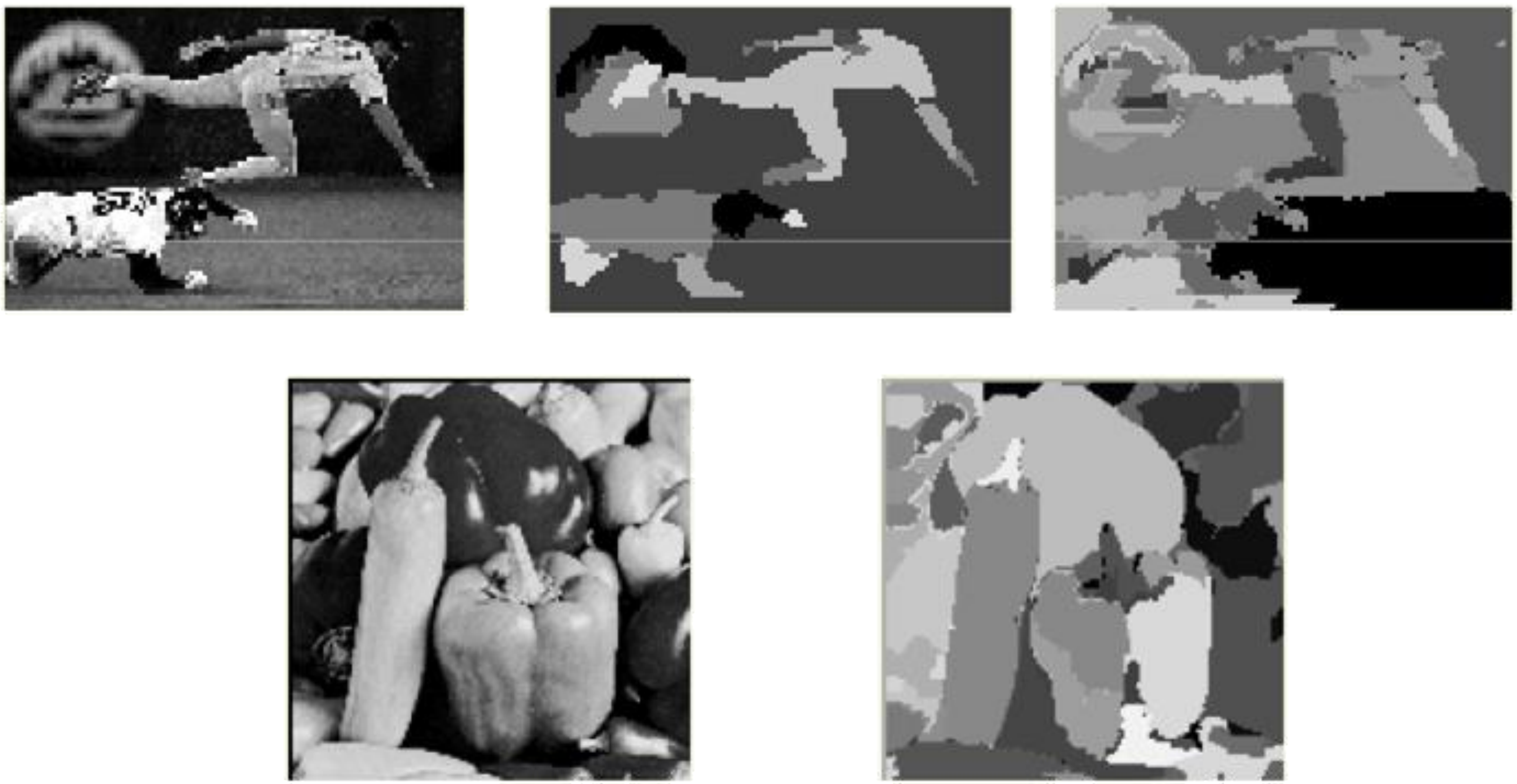
Historic application of clustering

- John Snow, a London physician plotted the location of cholera deaths on a map during an outbreak in the 1850s.
- The locations indicated that cases were clustered around certain intersections where there were polluted wells -- thus exposing both the problem and the solution.



From: Nina Mishra HP Labs

Computer vision application: Image segmentation

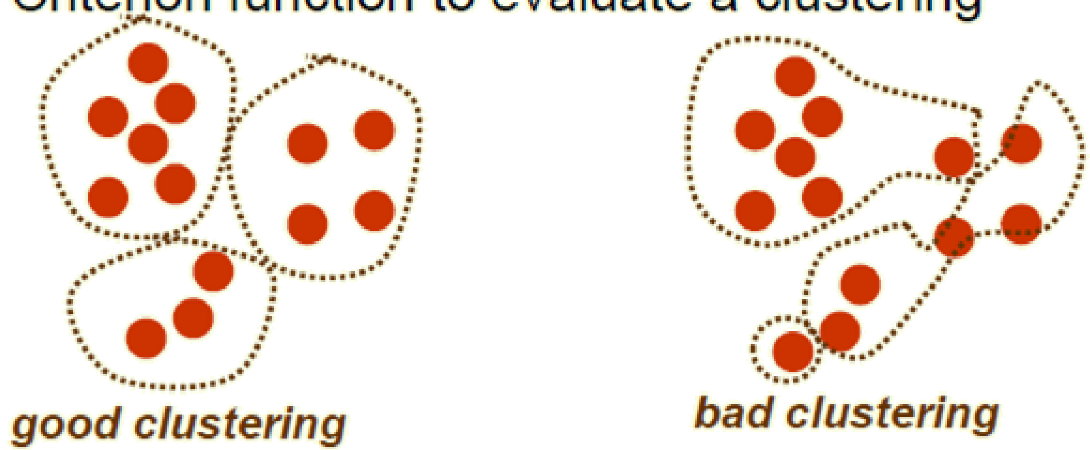


From: Image Segmentation by Nested Cuts, O. Veksler, CVPR2000

What do we need for clustering?

1. Proximity measure, *either*
   - similarity measure $\boldsymbol{s}(\boldsymbol{x}_i,\boldsymbol{x}_k)$: large if $\boldsymbol{x}_i,\boldsymbol{x}_k$ are similar
   - dissimilarity(or distance) measure $\boldsymbol{d}(\boldsymbol{x}_i,\boldsymbol{x}_k)$: small if $\boldsymbol{x}_i,\boldsymbol{x}_k$ are similar

2. Criterion function to evaluate a clustering



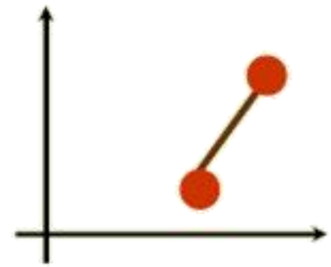
3. Algorithm to compute clustering
   - For example, by optimizing the criterion function

Distance (dissimilarity) measures

- Euclidean distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

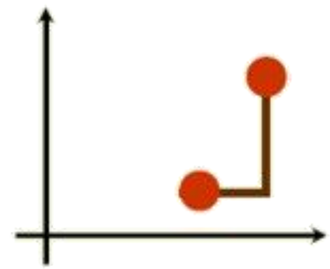
  - translation invariant



- Manhattan (city block) distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

  - approximation to Euclidean distance, cheaper to compute



- They are special cases of **Minkowski distance**:

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|^p \right)^{\frac{1}{p}}$$

(p is a positive integer)

Cluster evaluation (a hard problem)

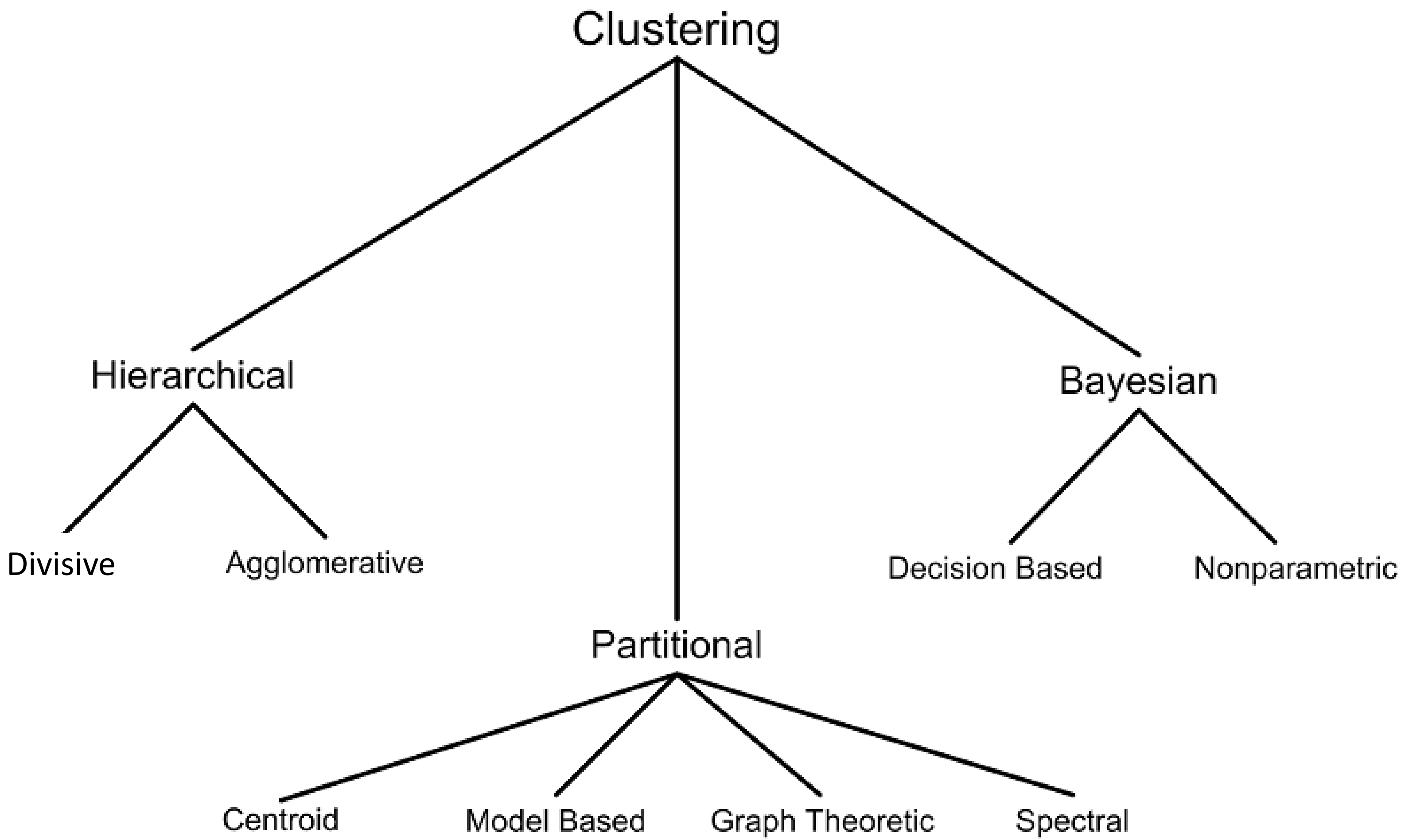
- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster centroid.
  - Sum of squared error (SSE) is a commonly used measure.
- **Inter-cluster separation** (isolation):
  - Separation means that different cluster centroids should be far away from one another.
- In most applications, expert judgments are still the key

How many clusters?

3 clusters or 2 clusters?

- Possible approaches
  1. fix the number of clusters to ***k***
  2. find the best clustering according to the criterion function (number of clusters may vary)

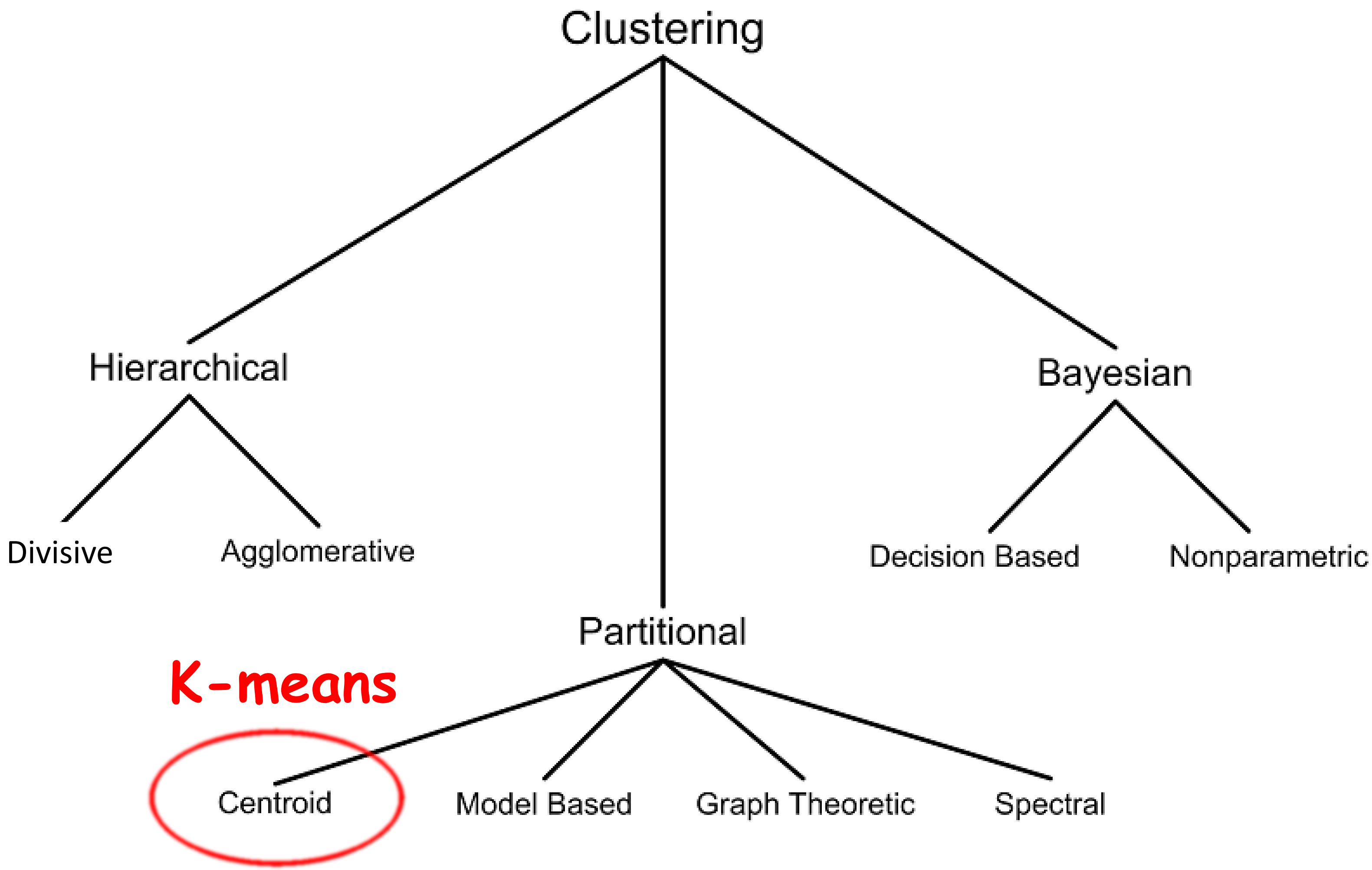
Clustering techniques



Clustering techniques

- **Hierarchical** algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative ("*bottom-up*") or divisive ("*top-down*"):
    1. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters;
    2. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
- **Partitional** algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.
- **Bayesian** algorithms try to generate a *posteriori distribution* over the collection of all partitions of the data.

Clustering techniques



K-Means clustering

- K-means (MacQueen, 1967) is a partitional clustering algorithm
- Let the set of data points D be $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in $X \subseteq R^r$, and r is the number of dimensions.
- The k-means algorithm partitions the given data into k clusters:
  - Each cluster has a cluster **center**, called **centroid**.
  - k is specified by the user

K-means algorithm

- Given *k*, the *k-means* algorithm works as follows:
  1. Choose *k* (random) data points (seeds) to be the initial centroids, cluster centers
  2. Assign each data point to the closest centroid
  3. Re-compute the centroids using the current cluster memberships
  4. If a convergence criterion is not met, repeat steps 2 and 3

K-means convergence (stopping) criterion

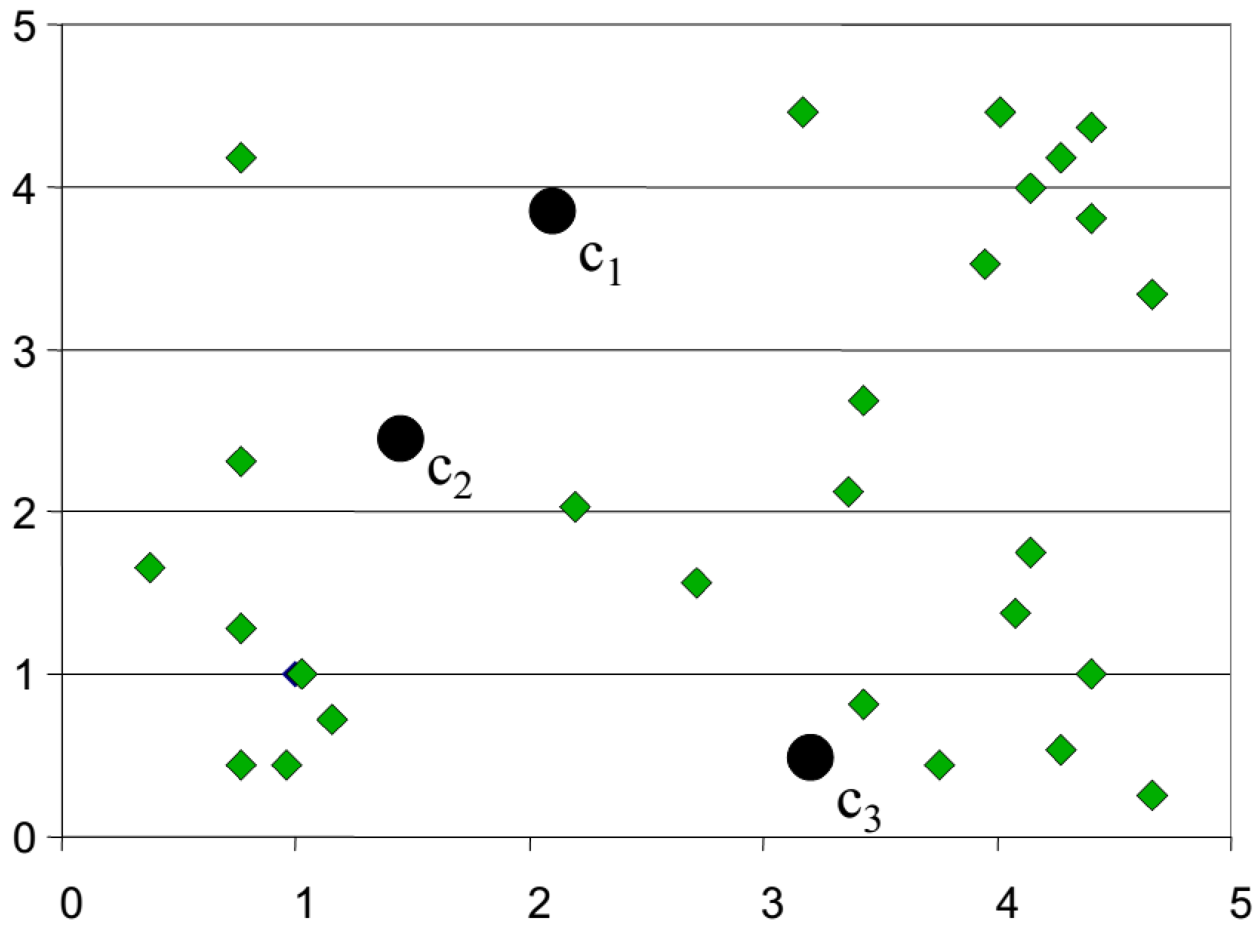
- no (or minimum) re-assignments of data points to different clusters, *or*
- no (or minimum) change of centroids, *or*
- minimum decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mathbf{m}_j)^2$$

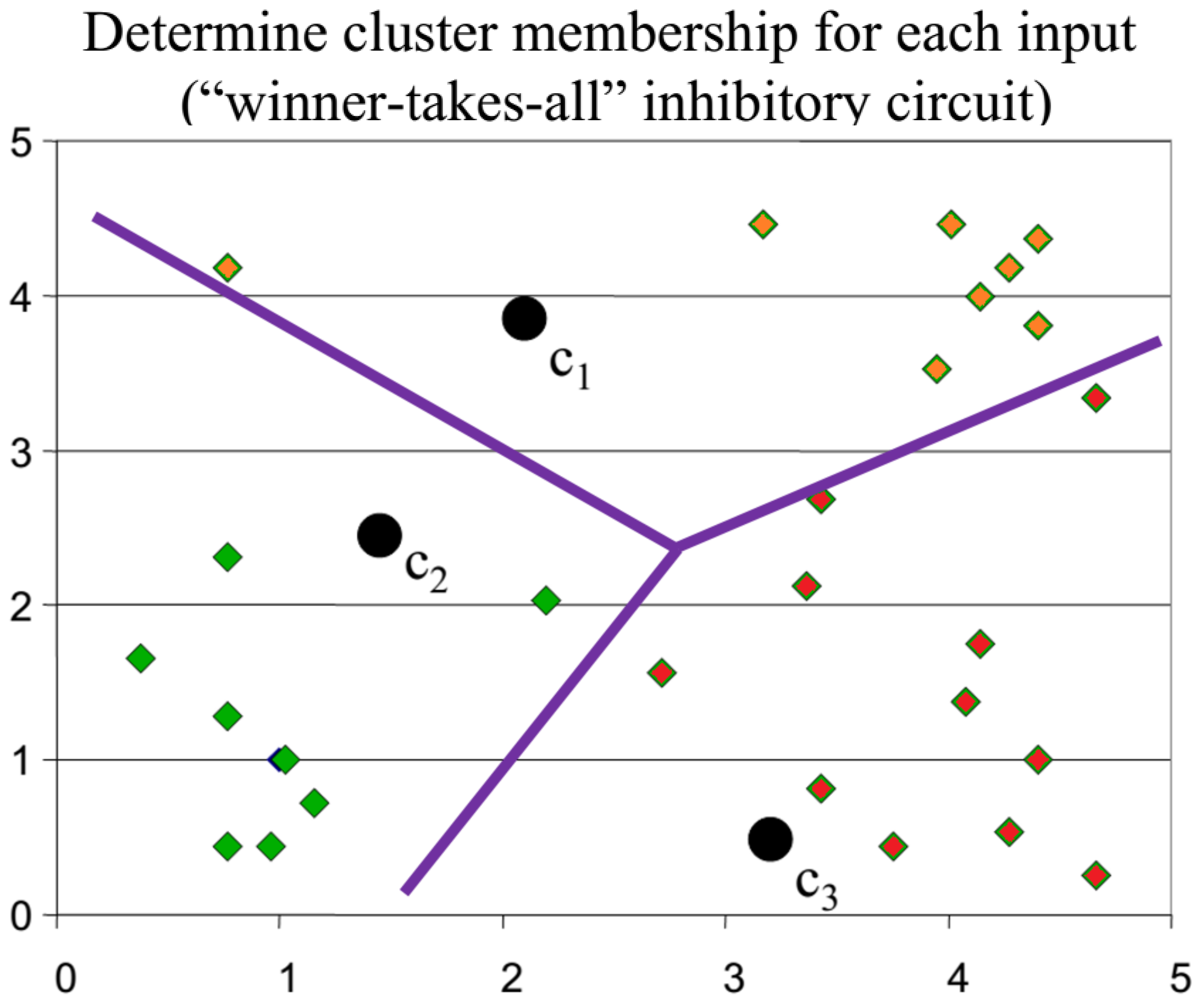
  – C_j is the *j*th cluster,
  – $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j),
  – $d(\mathbf{x}, \mathbf{m}_j)$ is the (Eucledian) distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

K-means clustering example: step 1

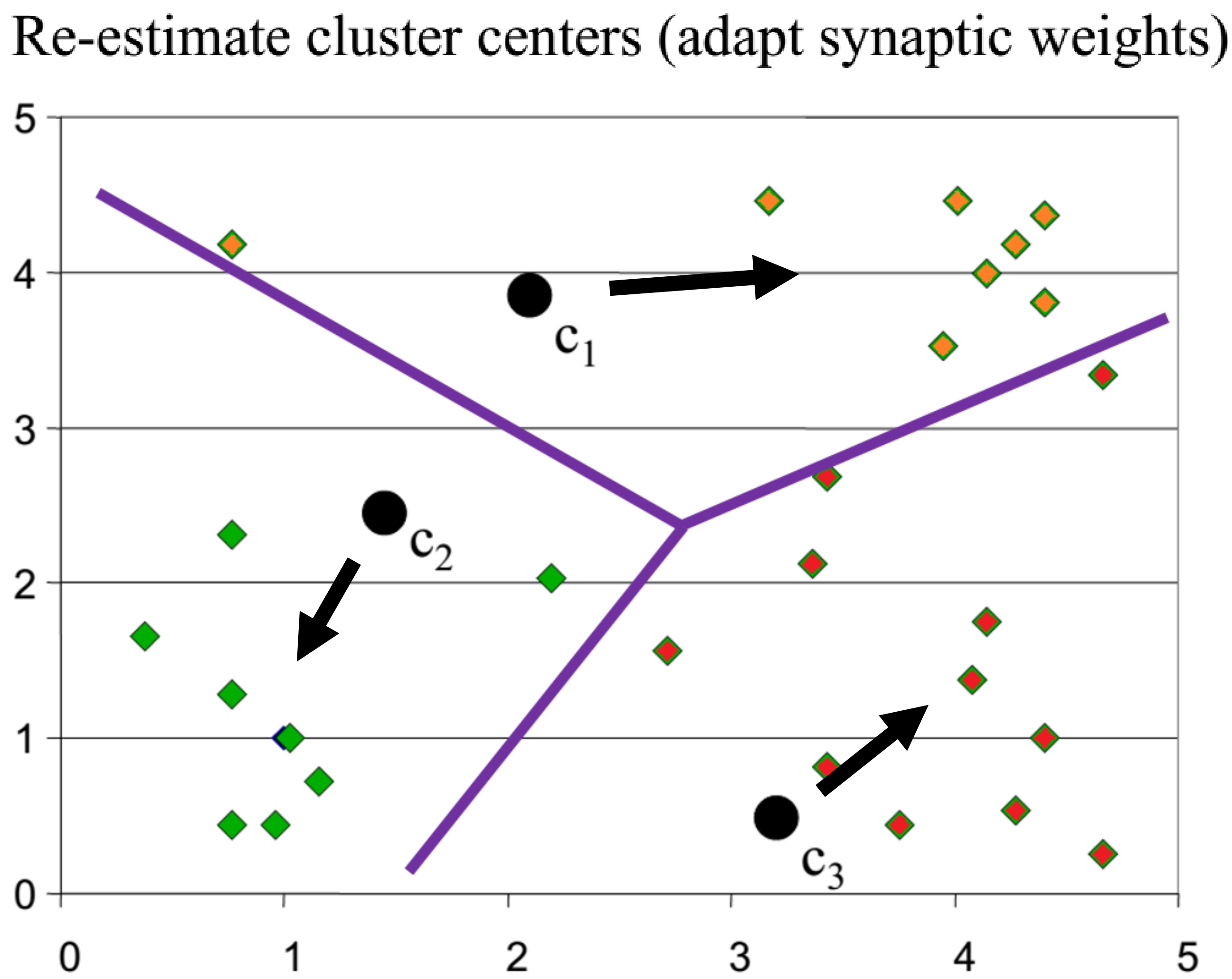
Randomly initialize the cluster centers (synaptic weights)



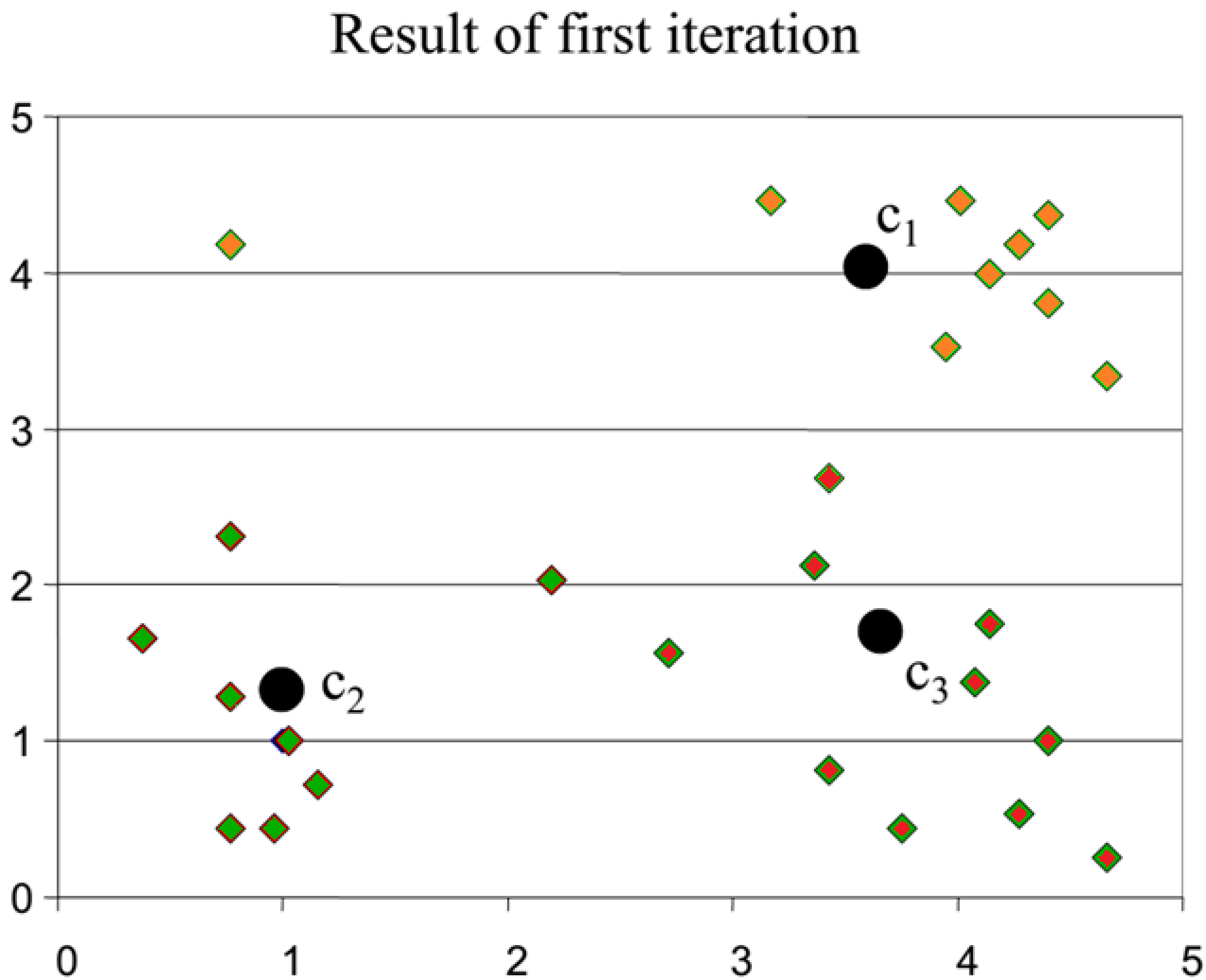
K-means clustering example – step 2



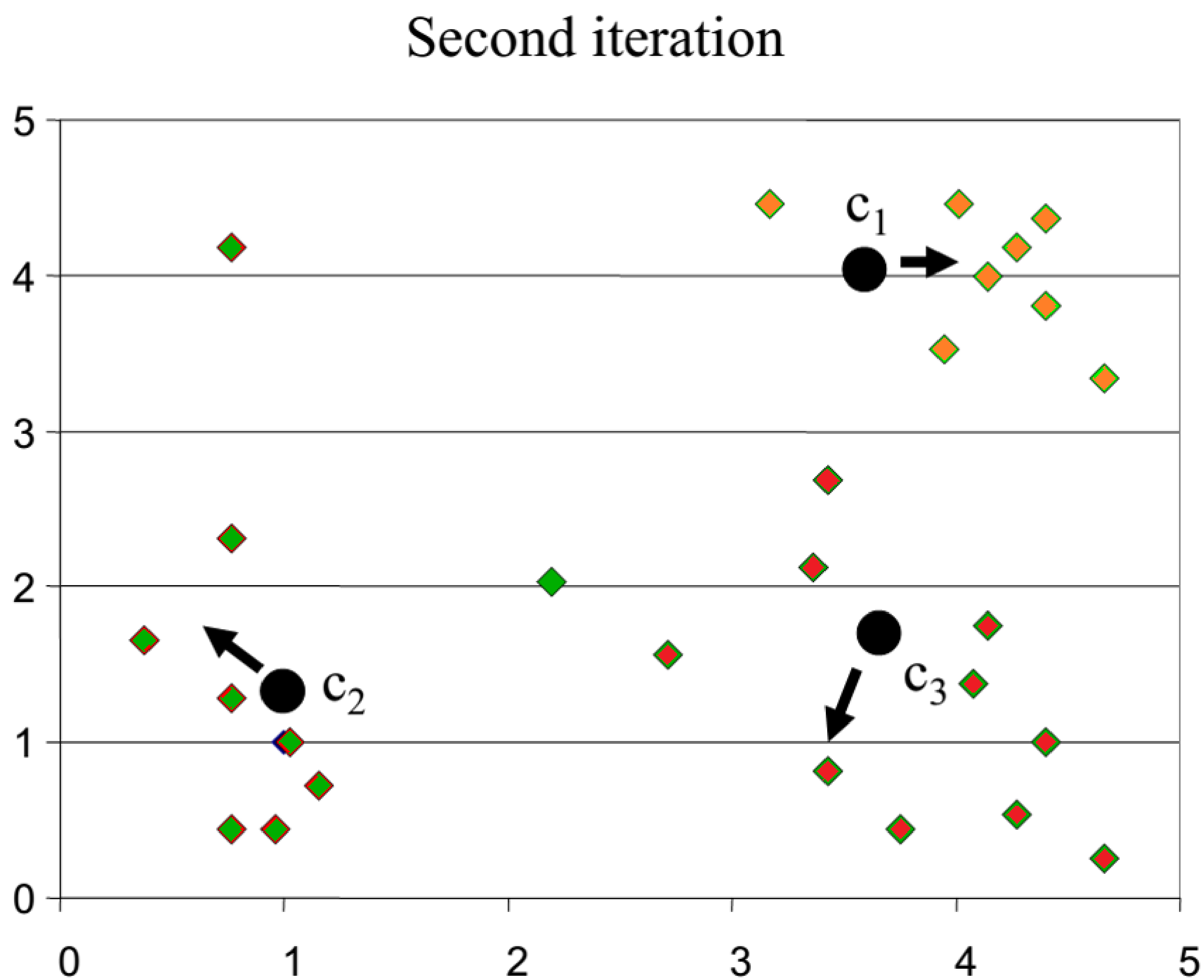
K-means clustering example – step 3



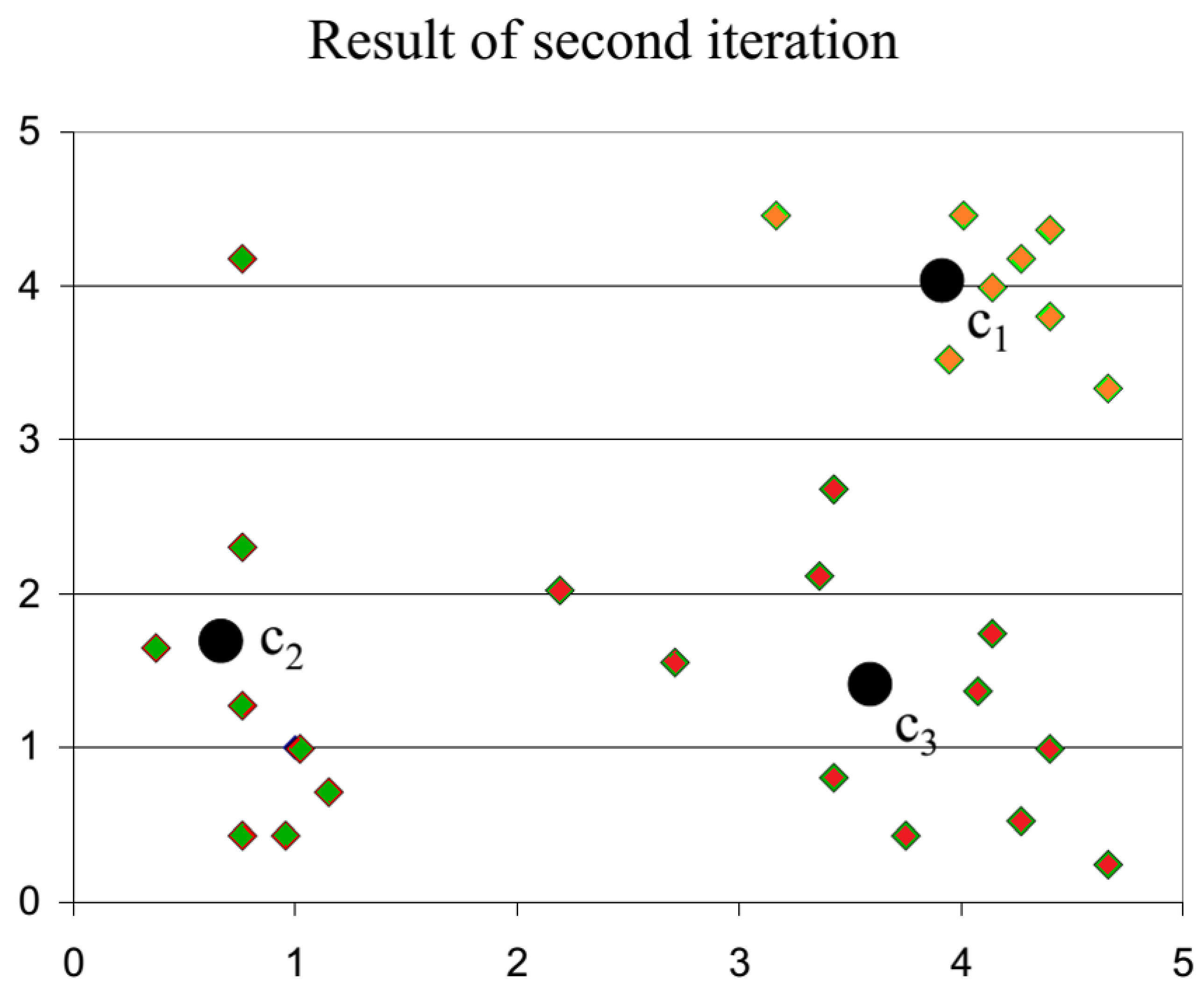
K-means clustering example



K-means clustering example



K-means clustering example



Why use K-means?

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,

    where n is the number of data points,

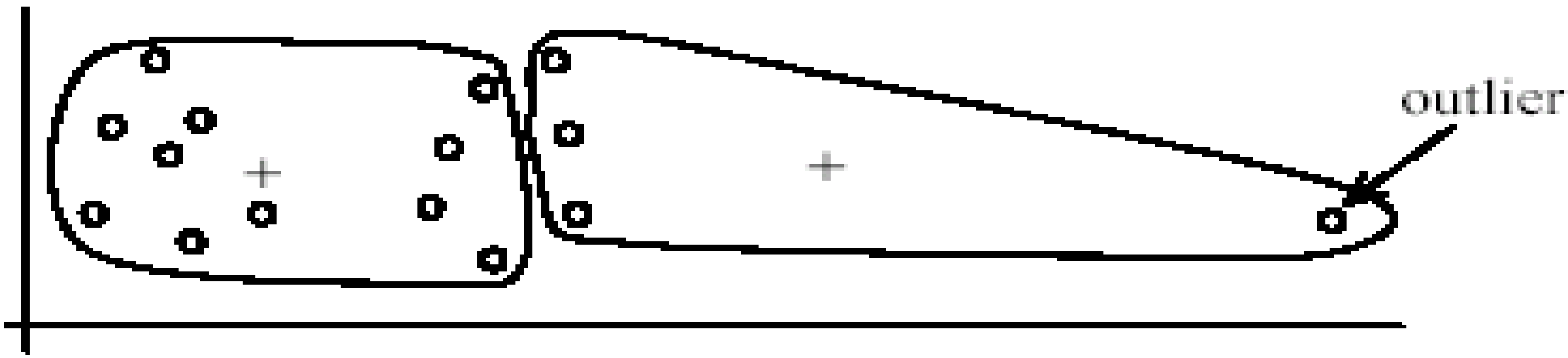
    k is the number of clusters, and

    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

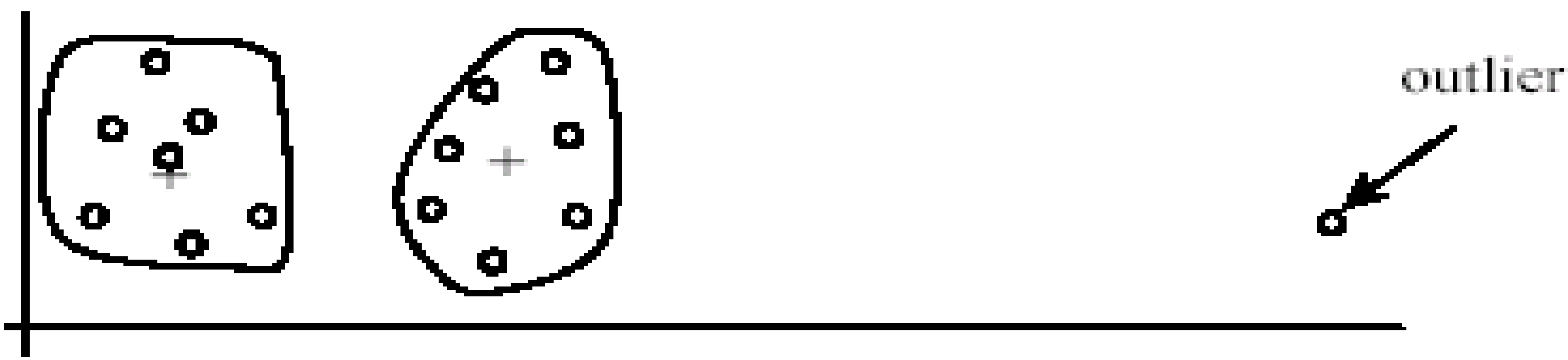
Weaknesses of K-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Outliers



(A): Undesirable clusters

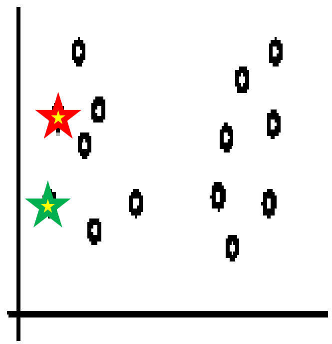


(B): Ideal clusters

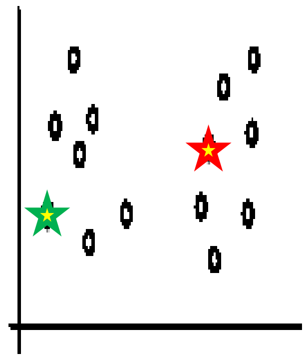
Dealing with outliers

- Remove some data points that are much further away from the centroids than other data points
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Perform random sampling: by choosing a small subset of the data points, the chance of selecting an outlier is much smaller
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

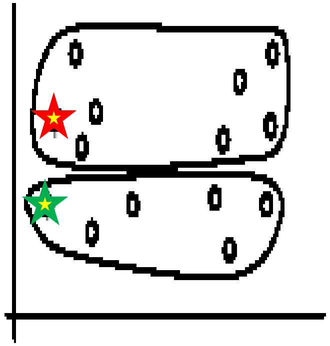
Sensitivity to initial seeds



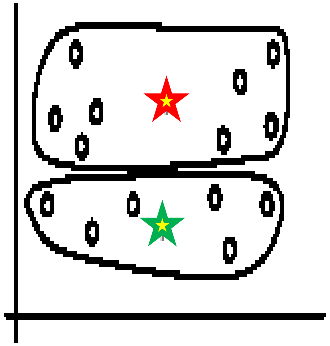
Random selection of seeds (centroids)



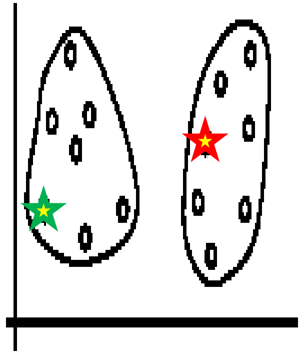
Random selection of seeds (centroids)



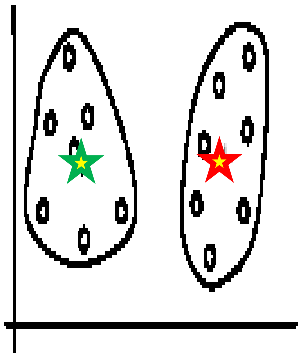
Iteration 1



Iteration 2



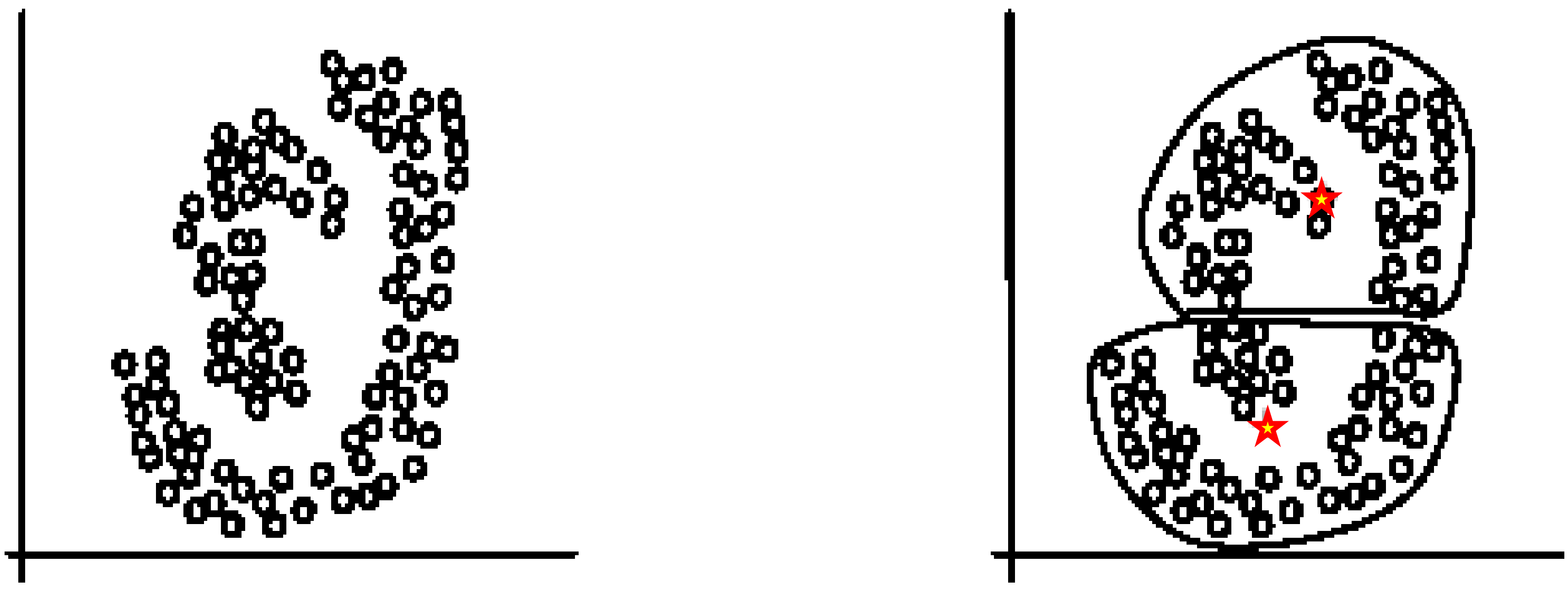
Iteration 1



Iteration 2

Special data structures

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).

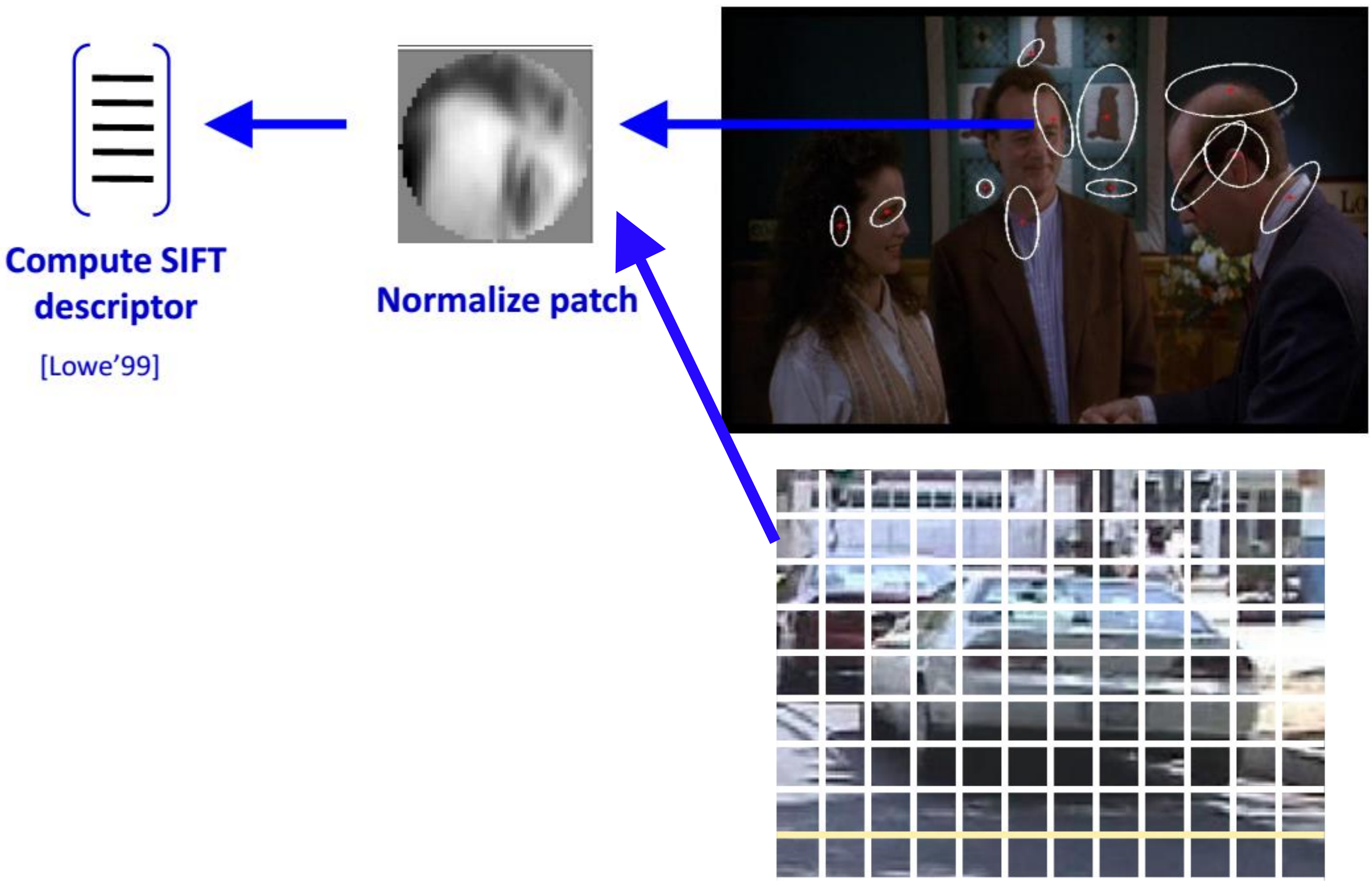


(A): Two natural clusters (B): *k*-means clusters

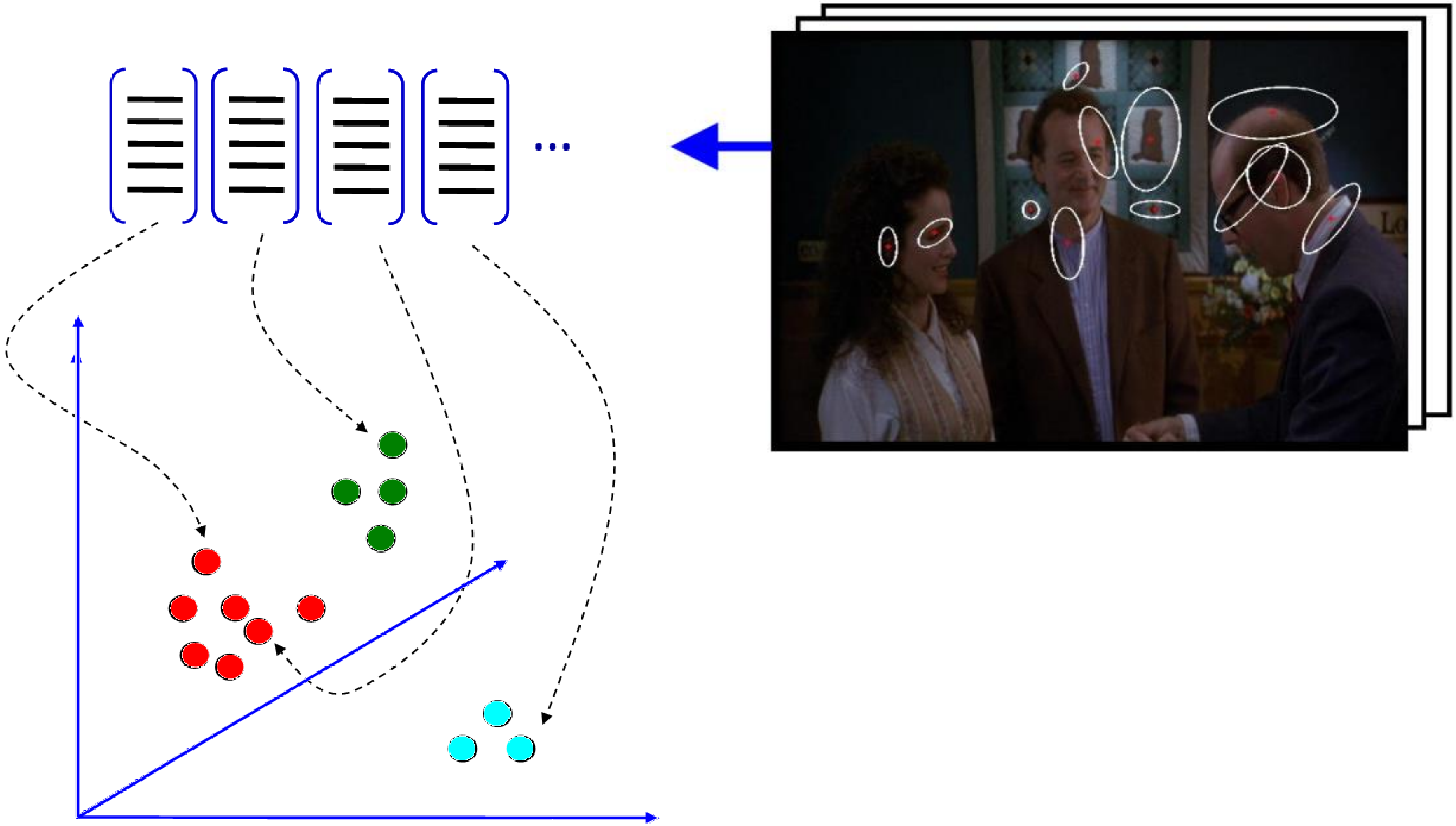
K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity and efficiency
- No clear evidence that any other clustering algorithm performs better in general
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

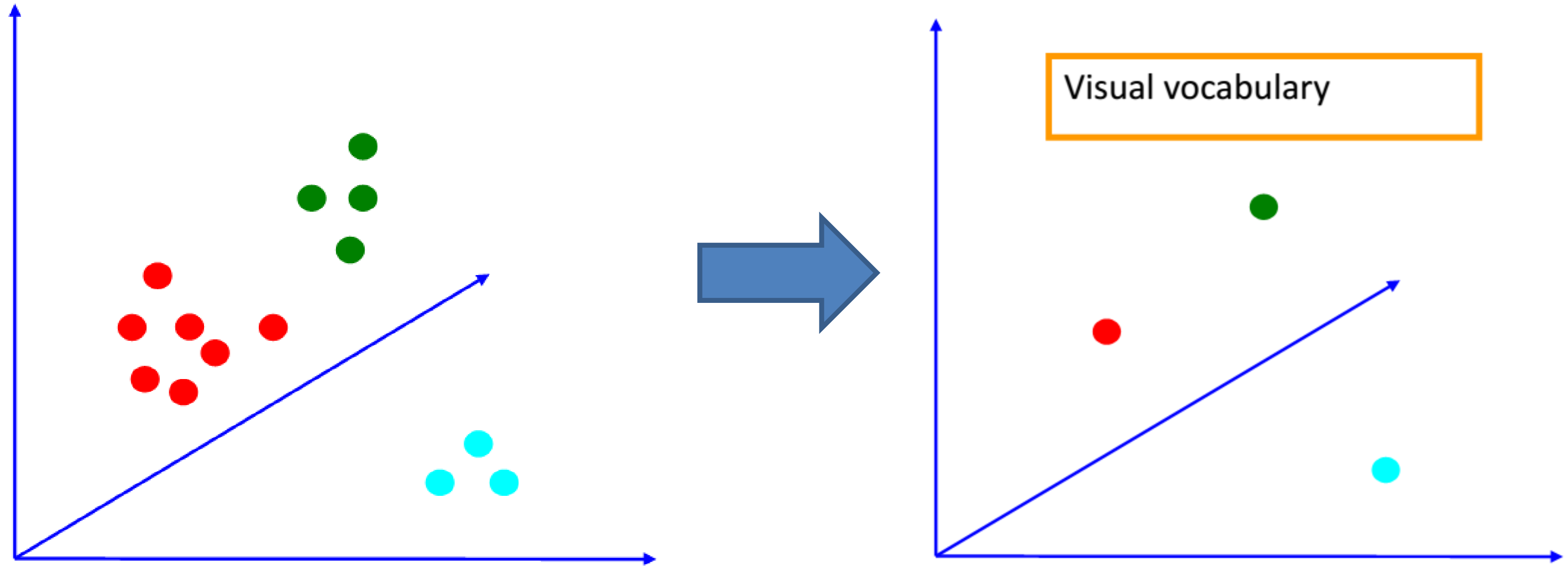
Application to visual object recognition: Dictionary learning (Bag of Words)



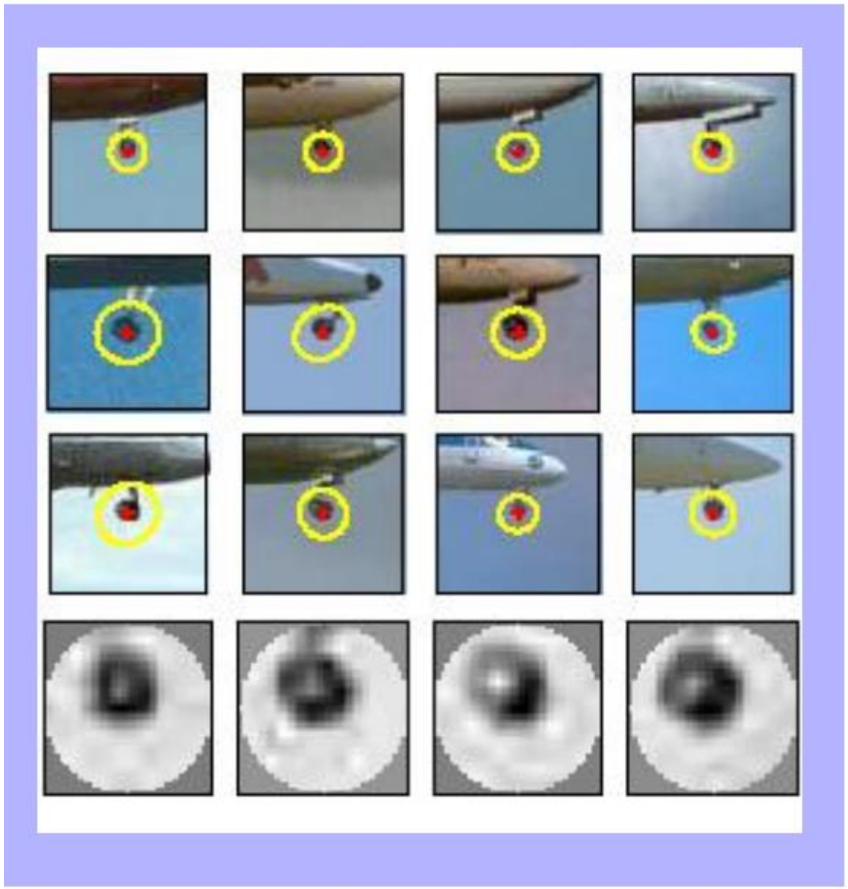
Learning the visual vocabulary

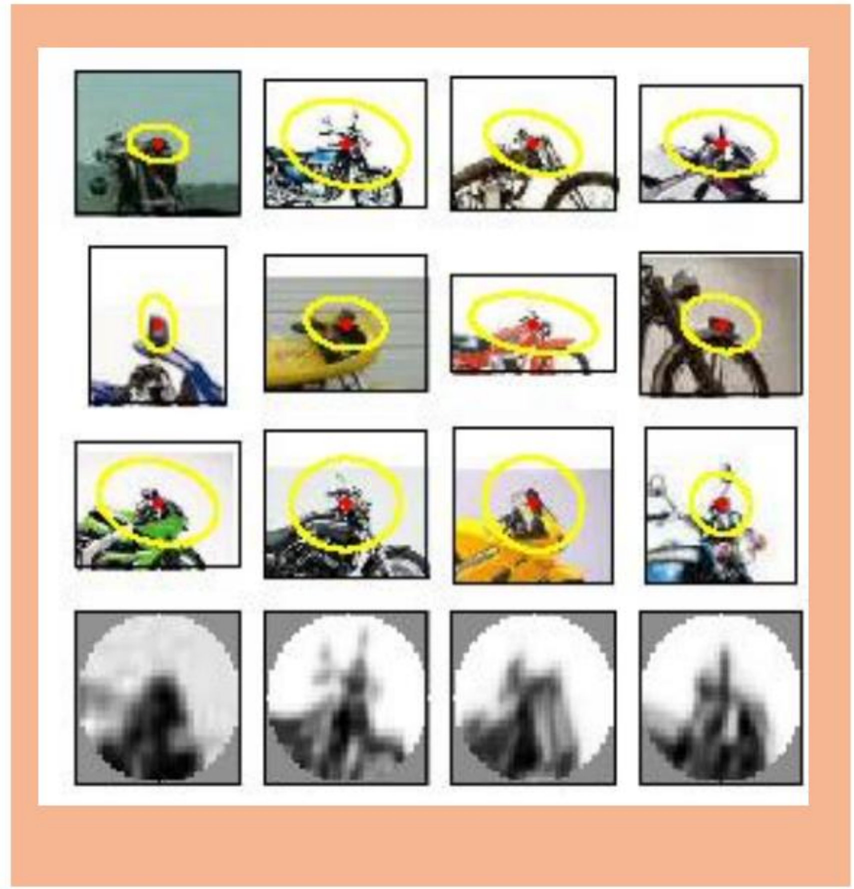


Learning the visual vocabulary

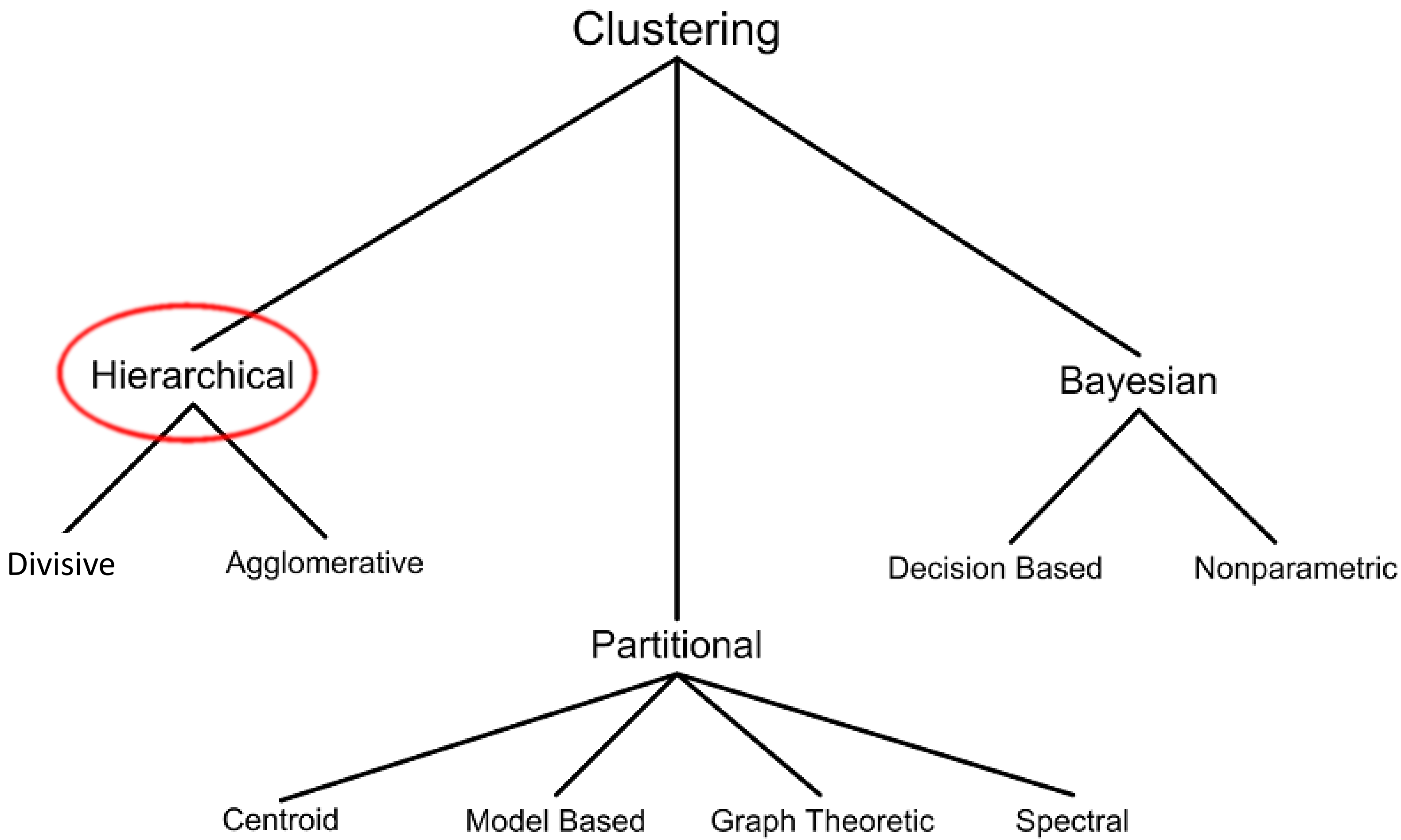


Examples of visual words



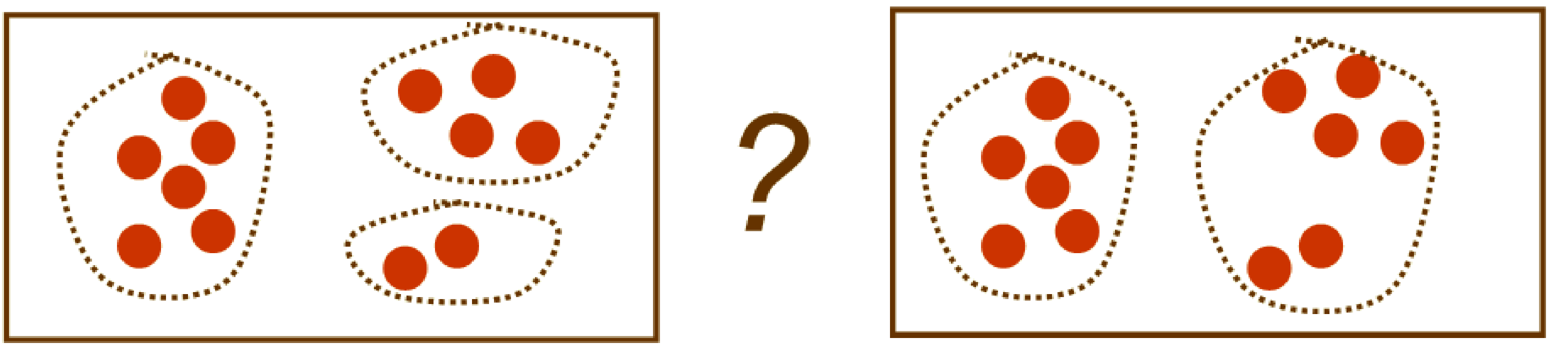


Clustering techniques



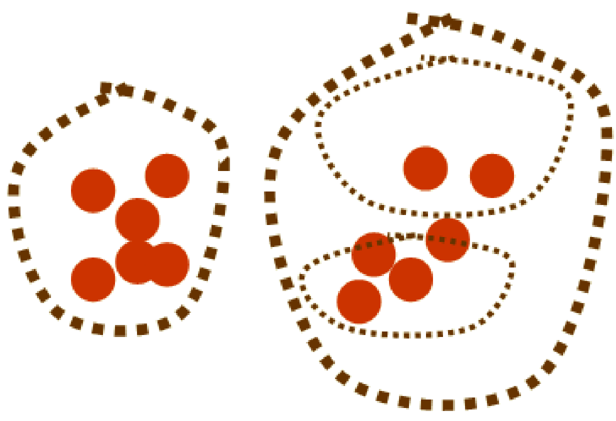
Hierarchical clustering

- Up to now, considered "flat" clustering

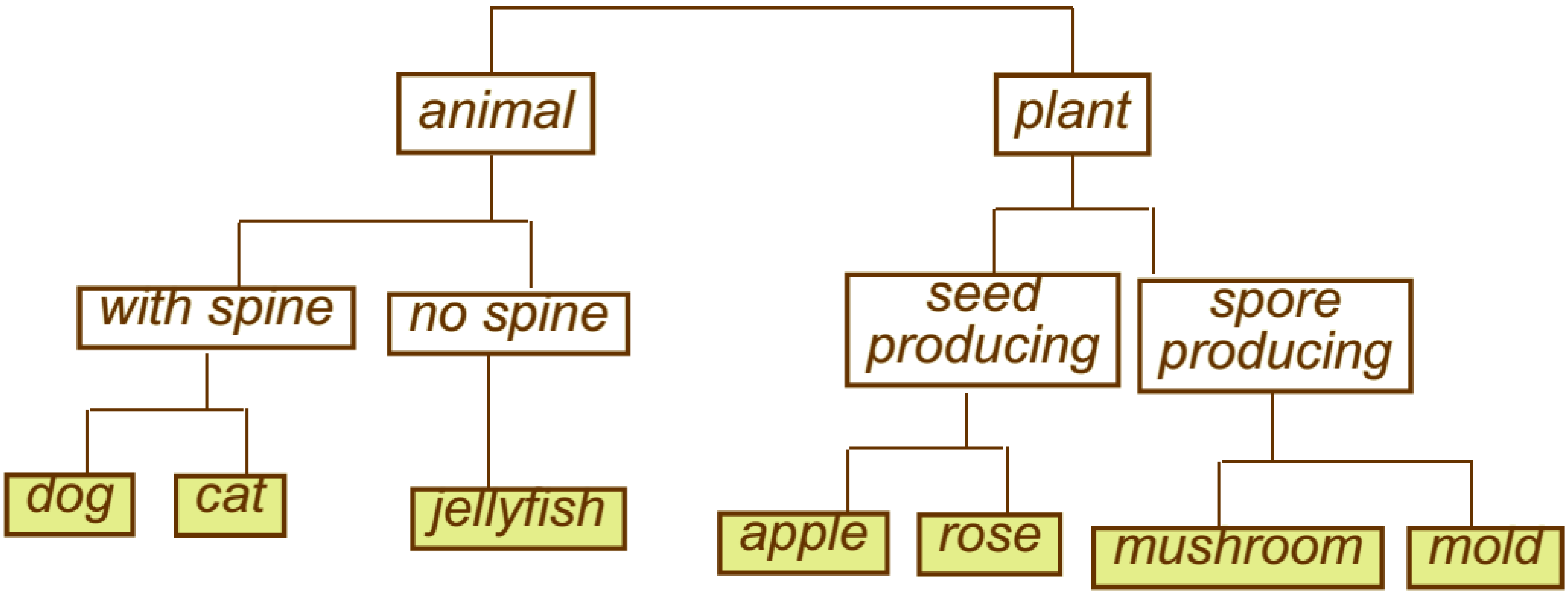


- For some data, hierarchical clustering is more appropriate than "flat" clustering

- Hierarchical clustering

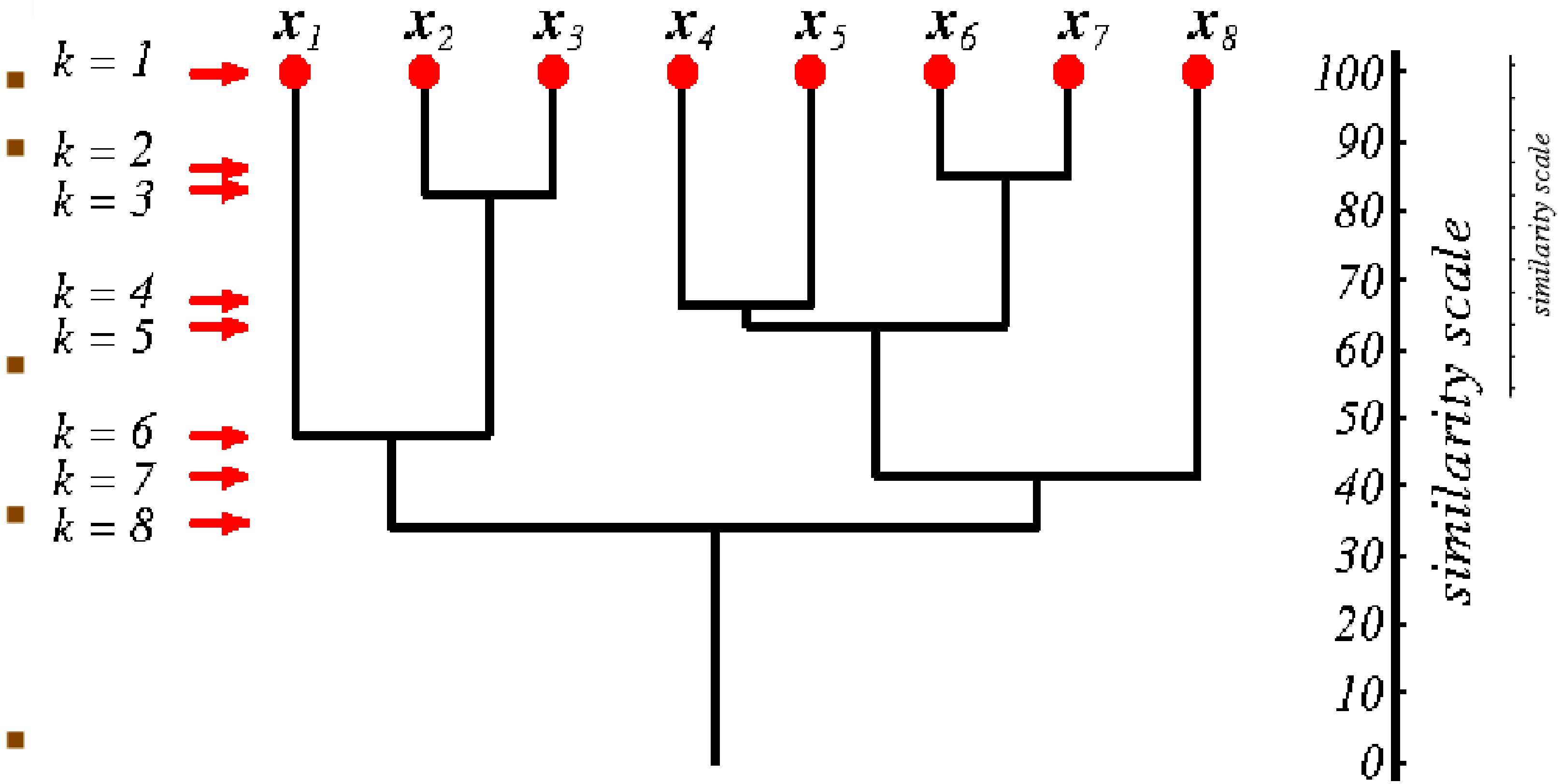


Example: biological taxonomy



A Dendrogram

- preferred way to represent a hierarchical clustering



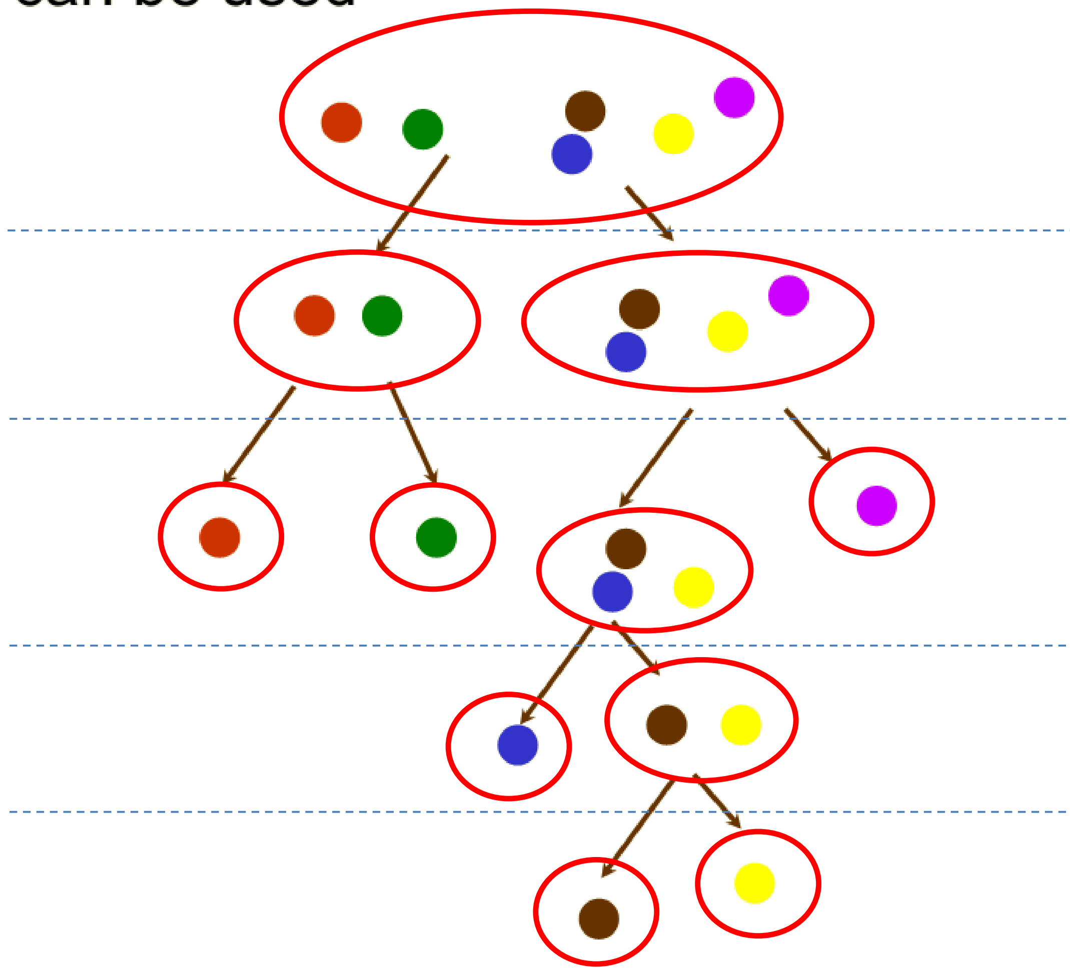
grouped clusters

Types of hierarchical clustering

- Divisive (top down) clustering
  Starts with all data points in one cluster, the root, then
  - Splits the root into a set of child clusters. Each child cluster is recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point
- Agglomerative (bottom up) clustering
  The dendrogram is built from the bottom level by
  - merging the most similar (or nearest) pair of clusters
  - stopping when all the data points are merged into a single cluster (i.e., the root cluster).

Divisive hierarchical clustering

- Any "flat" algorithm which produces a fixed number of clusters can be used
  - set ***c* = *2***

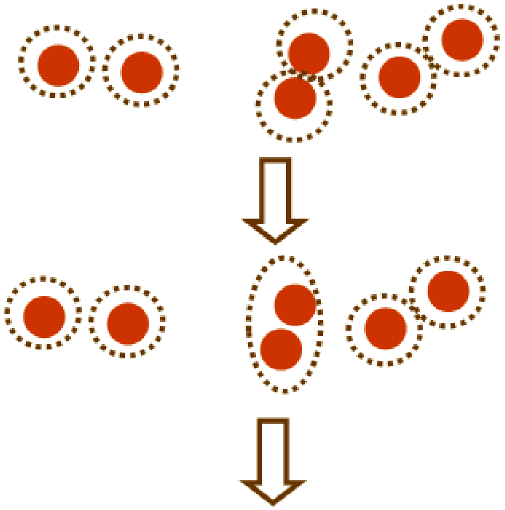


Agglomerative hierarchical clustering

initialize with each example in singleton cluster

while there is more than ***1*** cluster

1. find 2 nearest clusters
2. merge them



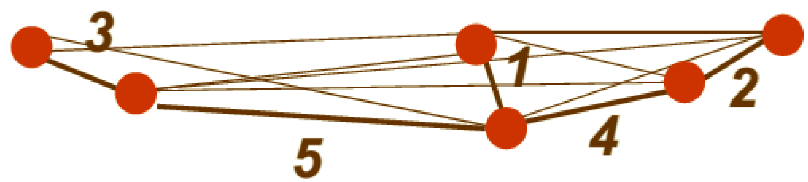
- Four common ways to measure cluster distance

1. minimum distance $$d_{min}(D_i, D_j) = \min_{x \in D_i, y \in D_j} \| x - y \|$$
2. maximum distance $$d_{max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \| x - y \|$$
3. average distance $$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{x \in D_i} \sum_{y \in D_j} \| x - y \|$$
4. mean distance $$d_{mean}(D_i, D_j) = \| \mu_i - \mu_j \|$$

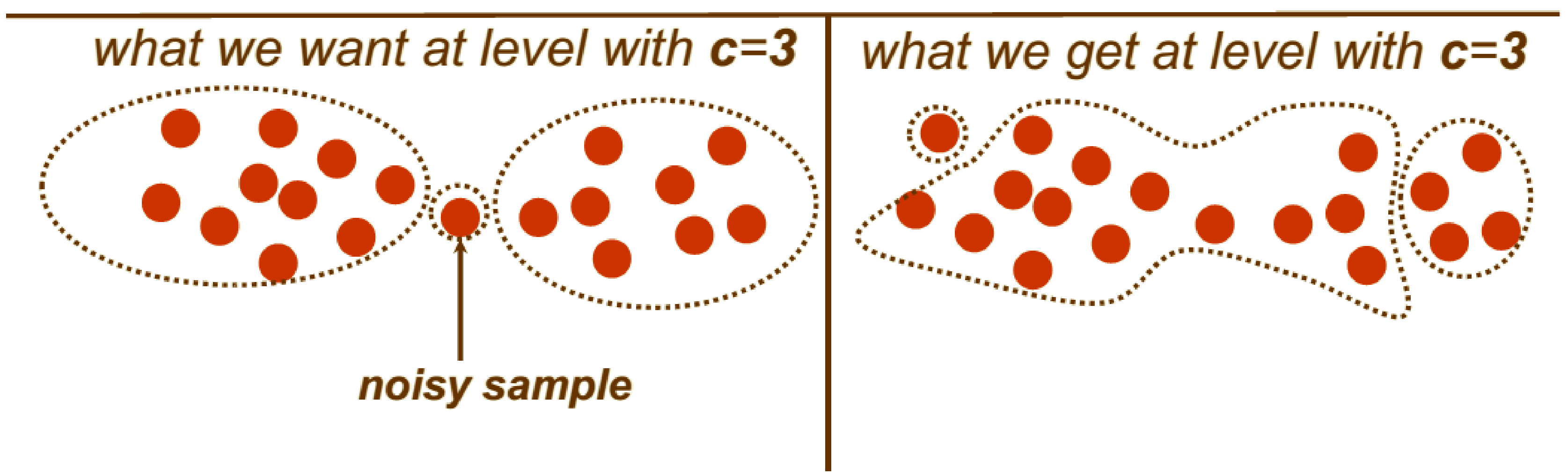
Single linkage or Nearest neighbor

- Agglomerative clustering with minimum distance

$$d_{\min}(D_i, D_j) = \min_{x \in D_i, y \in D_j} \| x - y \|$$



- generates minimum spanning tree
- encourages growth of elongated clusters
- disadvantage: very sensitive to noise

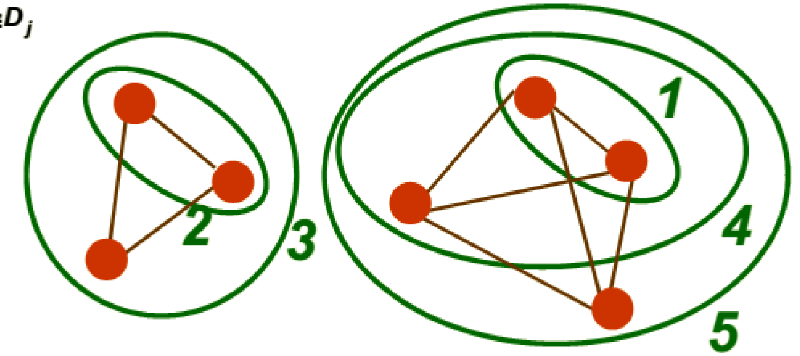


Complete linkage or Farthest neighbor

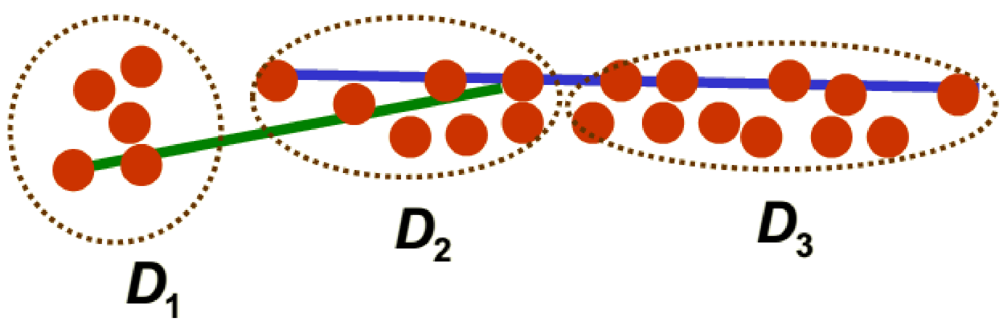
- Agglomerative clustering with maximum distance

$$d_{\max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \| x - y \|$$

- encourages compact clusters



- Does not work well if elongated clusters present

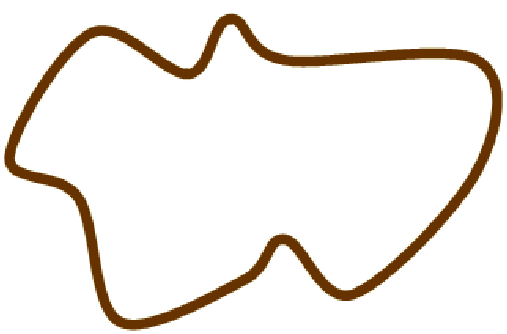


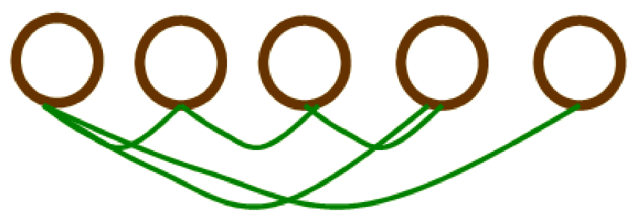
  - $d_{\max}(D_1, D_2) < d_{\max}(D_2, D_3)$
  - thus D_1 and D_2 are merged instead of D_2 and D_3

Divisive vs. Agglomerative

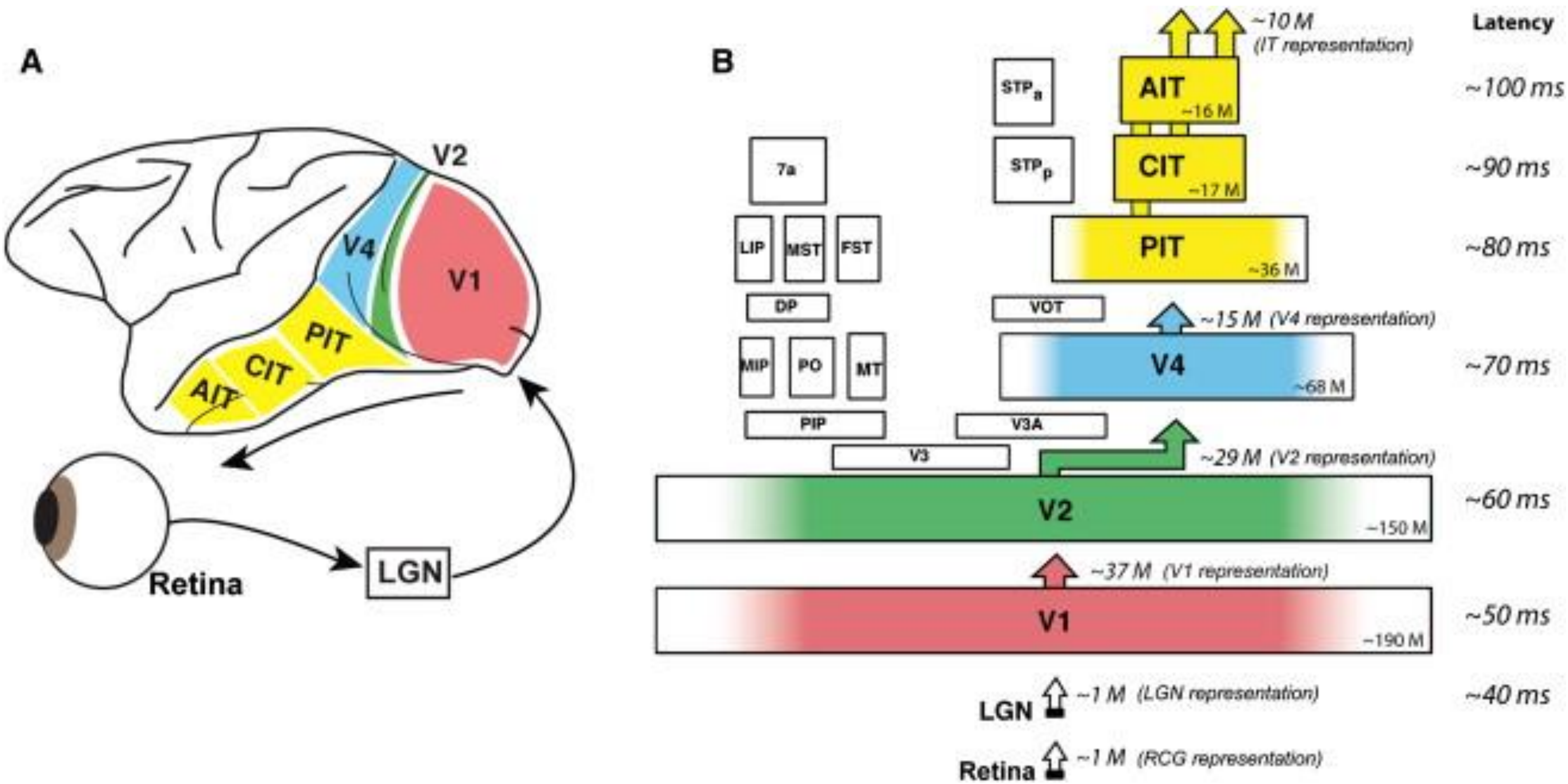
- Agglomerative is faster to compute, in general
- Divisive may be less "blind" to the global structure of the data

Divisive	***Agglomerative***
when taking the first step (split), have access to all the data; can find the best possible split in 2 parts	when taking the first step merging, do not consider the global structure of the data, only look at pairwise structure

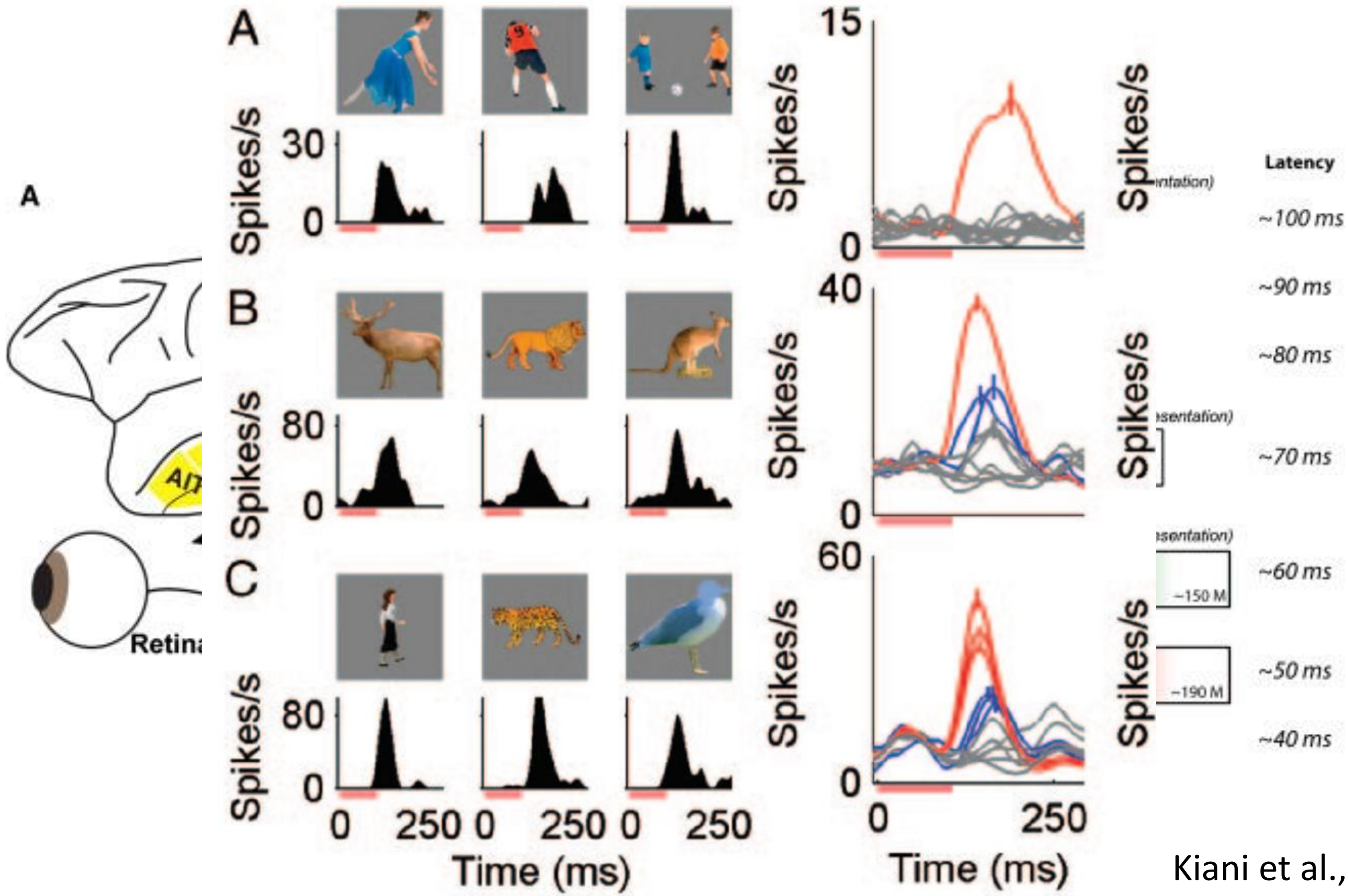




Object category structure in monkey inferior temporal (IT) cortex

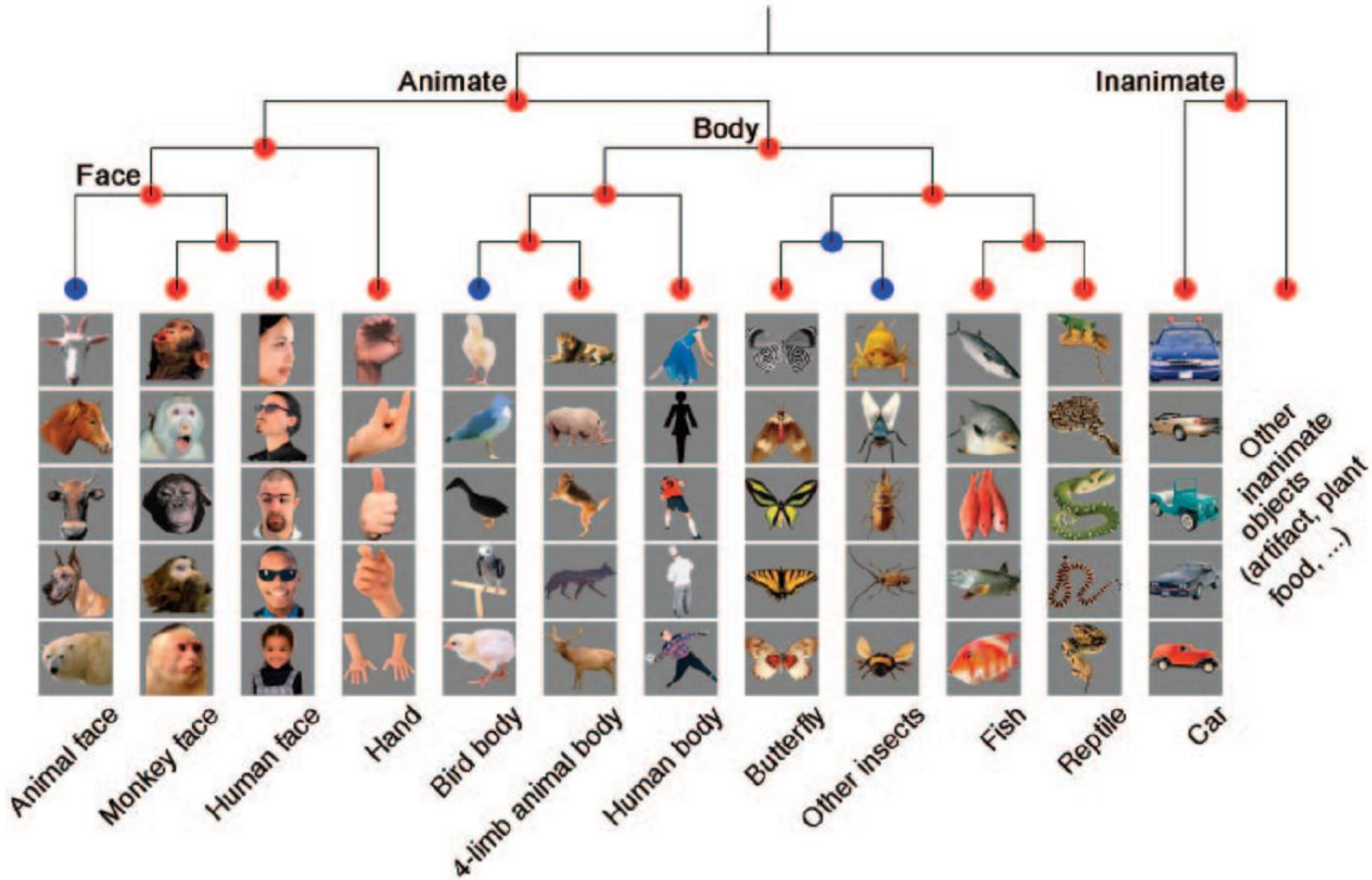


Object category structure in monkey inferior temporal (IT) cortex



Kiani et al., 2007

Hierarchical clustering of neuronal response patterns in monkey IT cortex

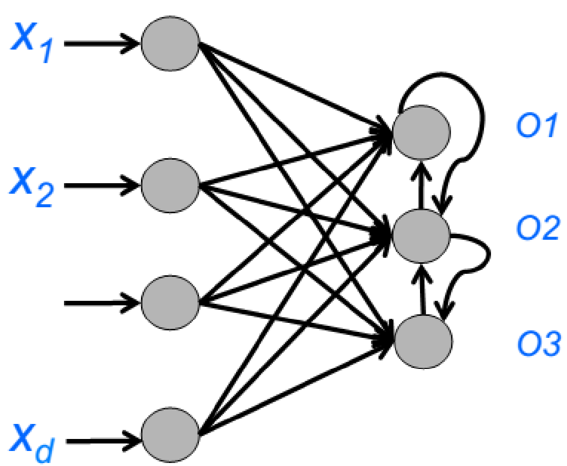


Kiani et al., 2007

Competitive learning

A form of unsupervised training where output units are said to be in competition for input patterns

- During training, the output unit that provides the highest activation to a given input pattern is declared the winner and is moved closer to the input pattern, whereas the rest of the neurons are left unchanged
- This strategy is also called winner-take-all since only the winning neuron is updated
  - Output units may have lateral inhibitory connections so that a winner neuron can inhibit others by an amount proportional to its activation level

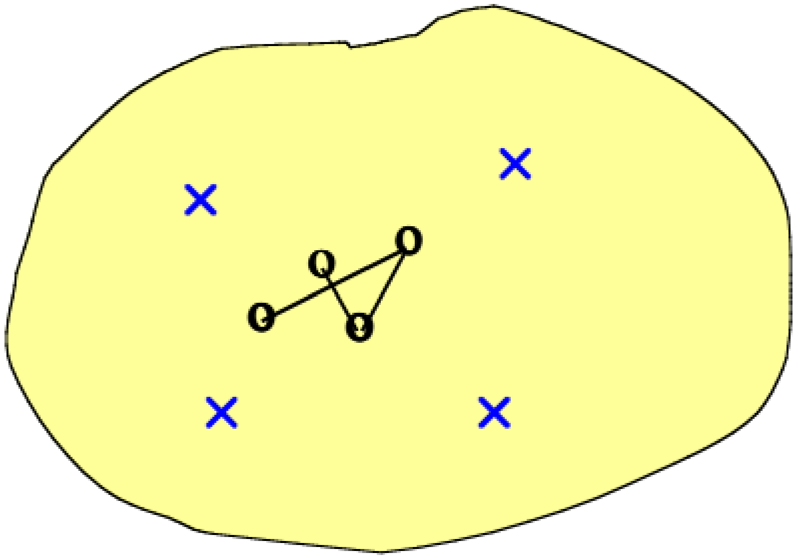


Competitive learning algorithm: Kohonen Self Organization Maps (K-SOM)

- Initialize the units to have random weights
- Repeat
  - Find the weight vector which is closest to the presented input vector. Call this the winner or the winning vector.
  - Modify the winner so as to move closer to the input vector
    - modifying weights so as to make them more similar to the values in the input vector.

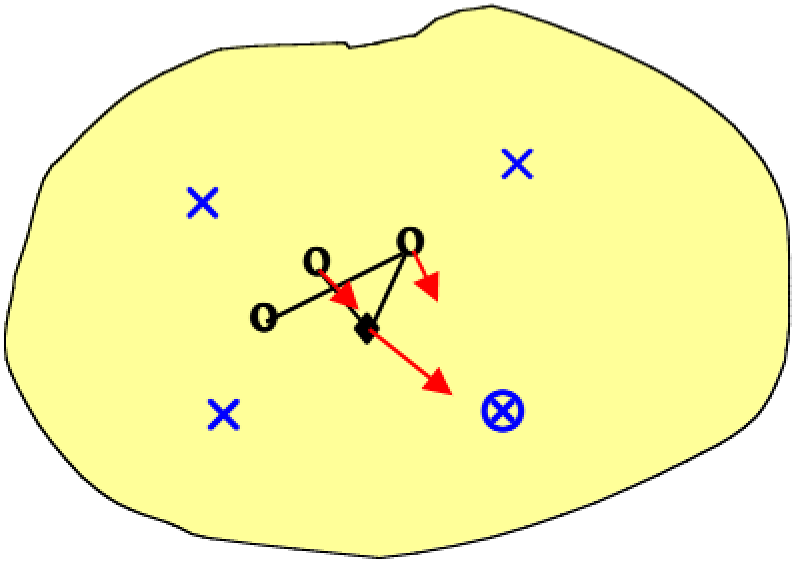
K-SOM example

- Four input data points (crosses) in 2D space.
- Four output nodes in a discrete 1D output space (mapped to 2D as circles).
- Random initial weights start the output nodes at random positions.



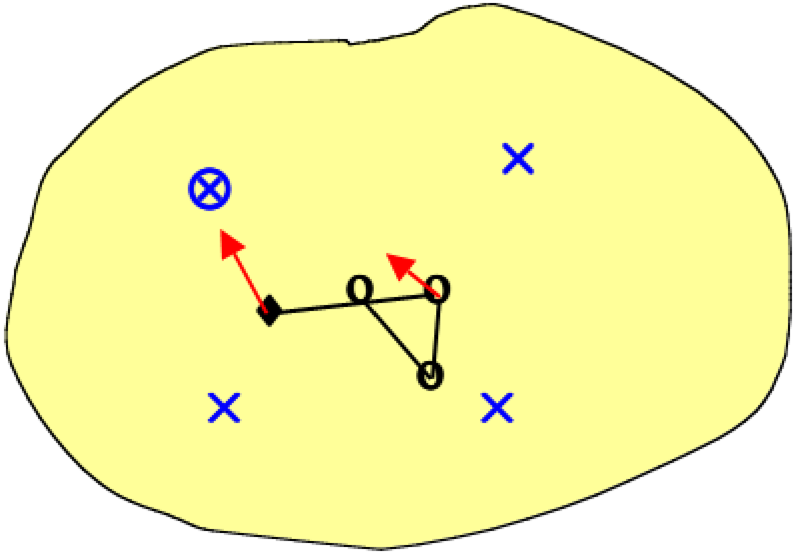
K-SOM example

- Randomly pick one input data point for training (cross in circle).
- The closest output node is the winning neuron (solid diamond).
- This winning neuron is moved towards the input data point, while its two neighbors move also by a smaller increment (arrows).



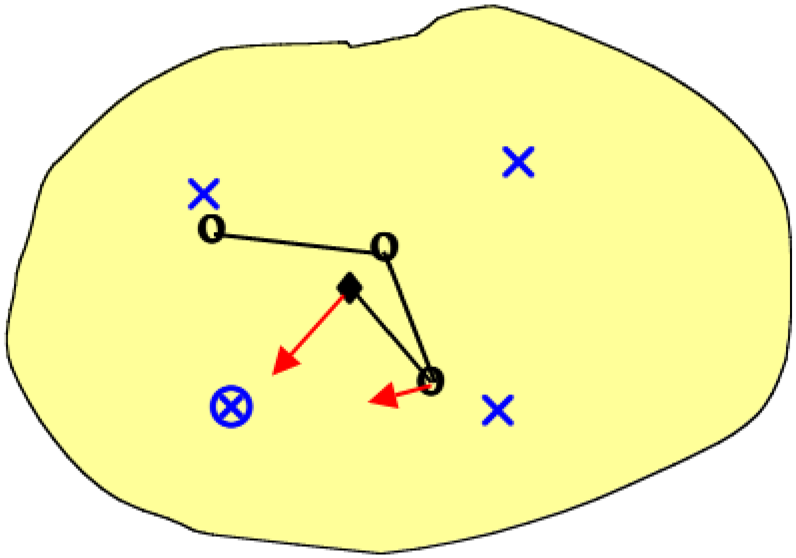
K-SOM example

- Randomly pick another input data point for training (cross in circle).
- The closest output node is the new winning neuron (solid diamond).
- This winning neuron is moved towards the input data point, while its single neighboring neuron move also by a smaller increment (arrows).



K-SOM example

- Continue to randomly pick data points for training, and move the winning neuron and its neighbors (by a smaller increment) towards the training data points.
- Eventually, the whole output grid unravels itself to represent the input space.



Competitive learning claimed effect

- Overtime the weight vectors move towards the centers of clusters of input vectors.
- Final state (convergence) finds one weight vector over the center of each cluster of the input vectors.
- It has been claimed this performs cluster analysis

Hebbian vs. Competitive learning

- H networks are used to extract information globally from the input space.
- H networks requires all weights to be updated at each epoch.
- H networks implement associative memory while C networks are selectors – only one can win!
- C networks are used to clusters similar inputs.
- C networks compete for resources.
- C networks, only the winner's weight is updated each epoch.

Note: epoch – one complete presentation of the input data to the network being trained.

Summary

- Clustering has a long history and still is in active research
  - There are a huge number of clustering algorithms, among them: Density based algorithm, Sub-space clustering, Scale-up methods, Neural networks based methods, Fuzzy clustering, Co-clustering …
  - More are still coming every year
- Clustering is hard to evaluate, but very useful in practice
- Clustering is highly application dependent (and to some extent subjective)
- Competitive learning in neuronal networks performs clustering analysis of the input data