Symbols, Patterns and Signals: nearest neighbour classification and Clustering

Dr. Iván Palomares Carrascosa, Office 319 Magg's House

Email: i.palomares@bristol.ac.uk

In this lecture…

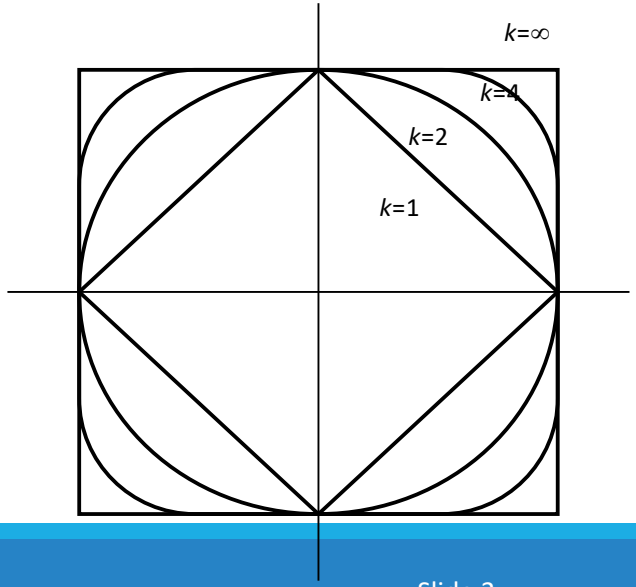
- **(Reminder of) Distance-based methods**
- **Nearest-neighbour classification**
- ***k*-means clustering**

Distance-based methods (reminder)

L_k norm or Minkowski metric for d-dimensional vectors *x,y*

$$L_k(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} \left| x_i - y_i \right|^k \right)^{1/k}$$

- **k=1**: Manhattan distance $L_1(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{d} \left| x_i - y_i \right|$
- **k=2**: Euclidean distance $L_2(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{d} \left(x_i - y_i \right)^2} = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})} = \left\| \mathbf{x}-\mathbf{y} \right\|$
- **k=∞**: $L_\infty(\mathbf{x},\mathbf{y}) = \max_i \left| x_i - y_i \right|$



Plot of all points at distance 1 from the origin for different values of *k* (*d*=2)

Symbolic distance (reminder)

For equal-length bit strings, the Hamming distance is the number of positions in which the two strings differ

- **101100** and **011101** have Hamming distance **3**
- corresponds to the $\mathbf{L_1}$ distance
  - we can also use $\mathbf{L_k}$, by just applying a monotonic transformation in this case

Can be generalised in various ways

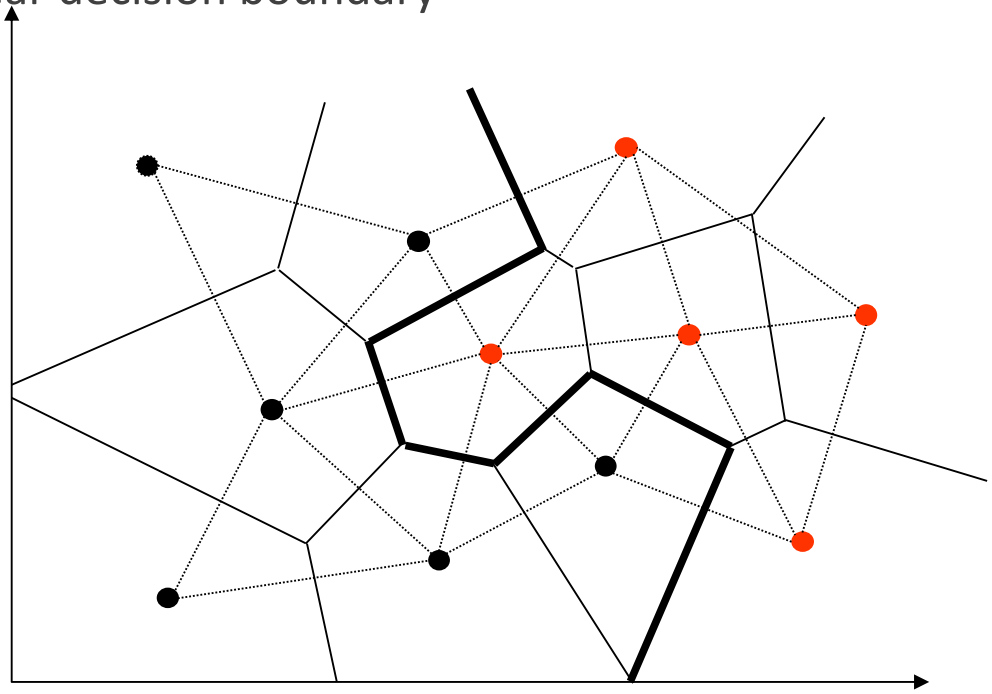
- symbolic unordered features: **0** if same, **1** if different
- symbolic ordered features: represent by (unit-variance) numbers
- strings of different lengths (e.g. words): edit distance

In what follows we assume some distance metric on feature vectors with possibly mixed numeric and symbolic features

A close neighbour…

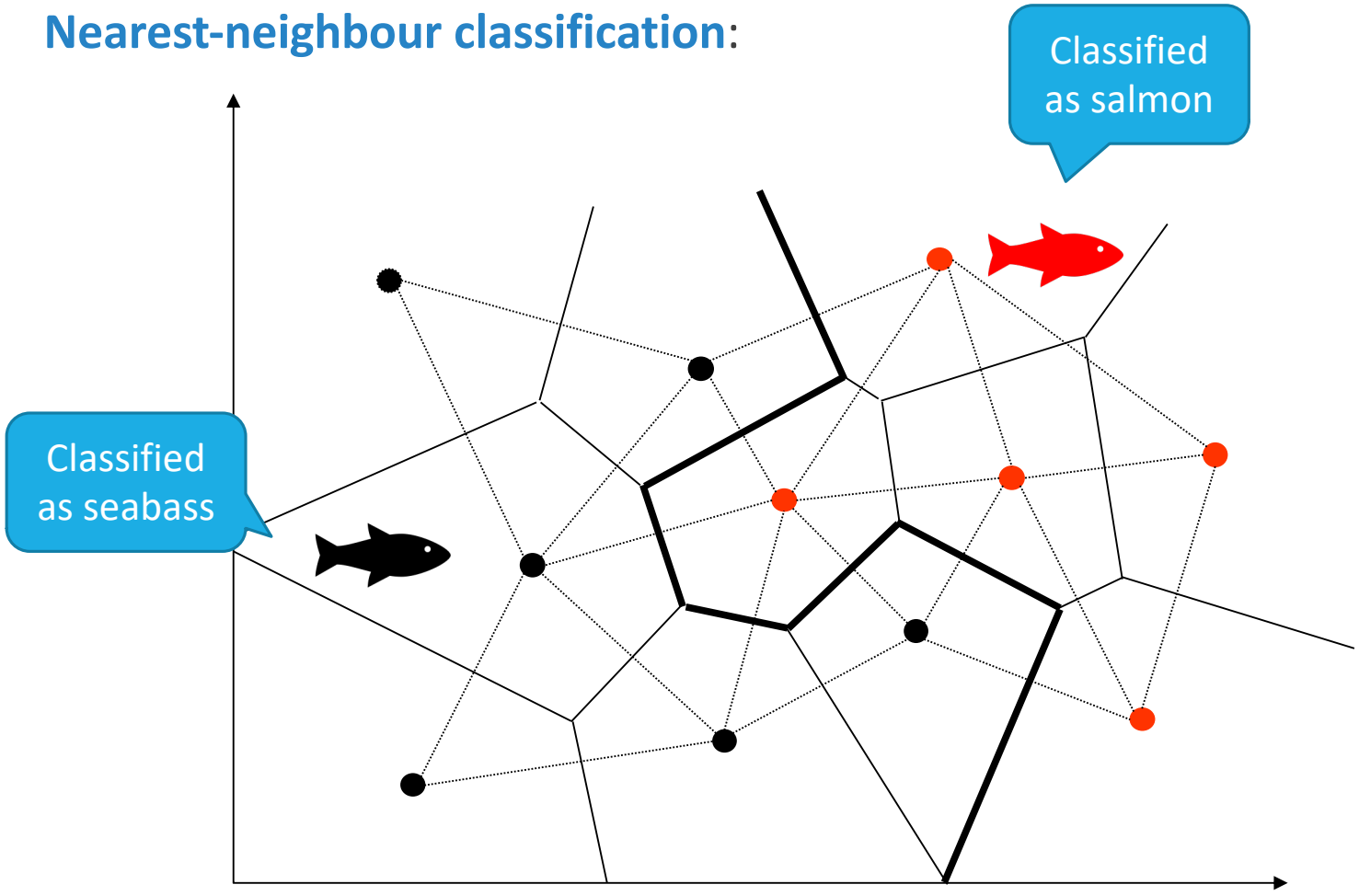
Nearest-neighbour classification: **given a set of labeled observations (e.g. the training data), assign a test instance to the class of its nearest exemplar**

- instance space divided by *Voronoi tesselation*
- piecewise linear decision boundary



A close neighbour…

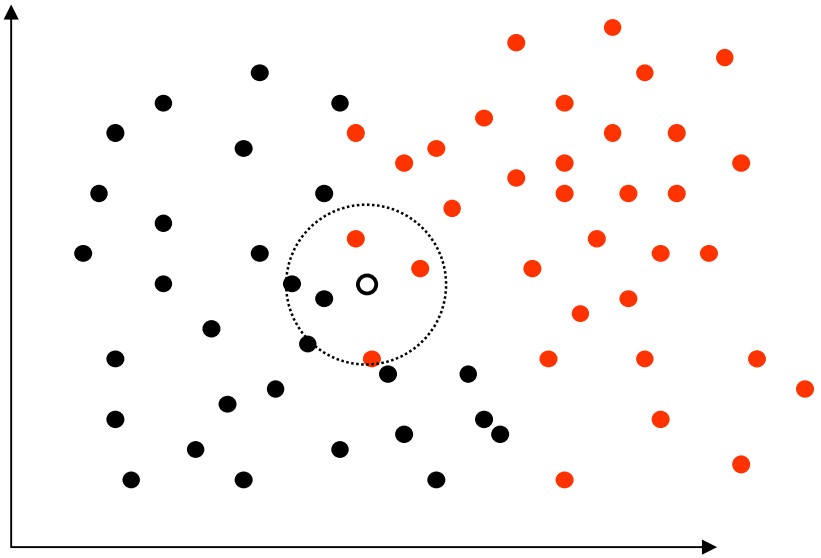
Nearest-neighbour classification:



Increasing the number of neighbours

***k*-Nearest-neighbour classification**: **assign a test instance to the majority class among its *k* nearest neighbours**

- smoother decision boundary than with **k=1**
- **k**$\rightarrow\infty$: all instances are assigned most common class in training set (i.e., using prior distribution only)
- **k** can be tuned using separate test set



To classify a test instance, we draw the smallest hypersphere around it that contains ***k*** neighbours; then assign majority class among those

Example: 2-nearest-neighbour

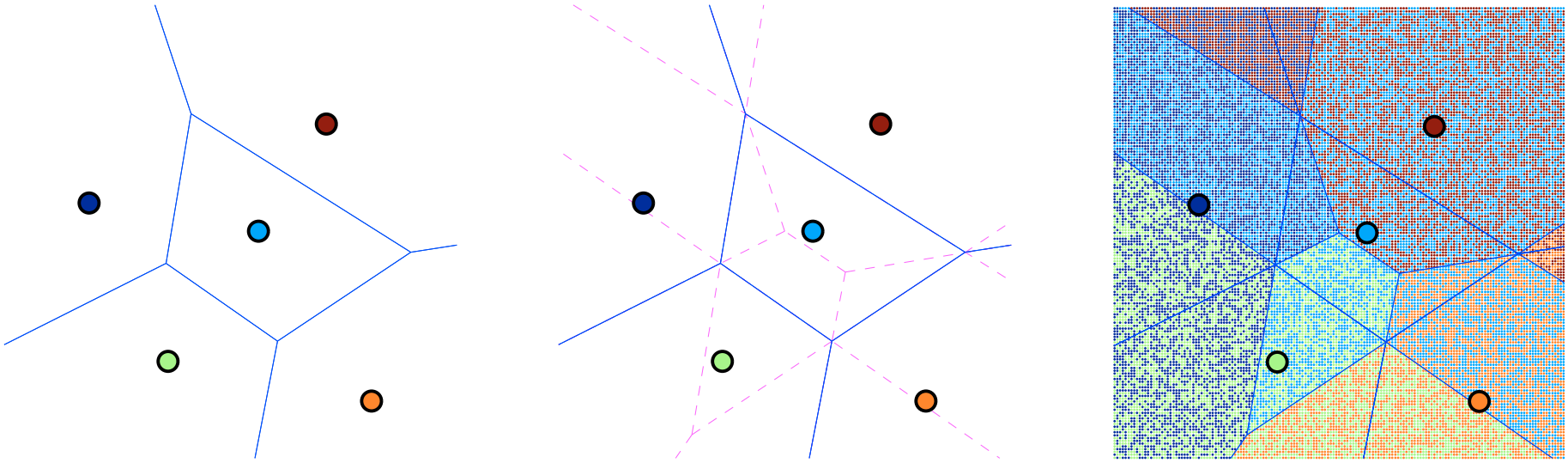


Figure 8.8. **(left)** Voronoi tesselation for five exemplars. **(middle)** Taking the two nearest exemplars into account leads to a further subdivision of each Voronoi cell. **(right)** The shading indicates which exemplars contribute to which cell.

k-nearest neighbour: variations

In *k-NN*, each of the *k* nearest neighbours x_i contributes equally to the vote for the most likely class of the test instance *x*

Distance-weighted *k-NN*: weight the vote with a kernel $K(x-x_i)$ that decreases with the distance

- e.g. Gaussian kernel

$$K(\mathbf{d}) \propto e^{-\|\mathbf{d}\|^2}$$

Using distance-weighting, we can also let k→∞ and have all training instances contribute to the vote

- global rather than local method

All these nearest neighbor methods can also be used for estimation, where the dependent variable is a scalar rather than a class

Discussion

k-nearest neighbour methods take *all* features into account. With large numbers of features this causes problems

- irrelevant features may dominate distance calculations
- training set covers only a fraction of instance space, which causes overfitting: *'curse of dimensionality'*

> *"As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially."*
> *- Prof. Charles Isbell,, Georgia Tech*

This can be addressed in several ways

- stretch/shrink dimensions if it improves performance on a separate test set
- perform feature selection in a pre-processing step

Clustering

Clustering involves segmenting the instance space into regions of *similar* objects

- no guidance through labelled training instances –> **unsupervised learning**

We can use the idea of a distance metric for clustering → *K*-means clustering (this lecture) and hierarchical clustering

We can also use the class as a 'hidden' variable → Gaussian mixture models (next lecture)

K-means clustering

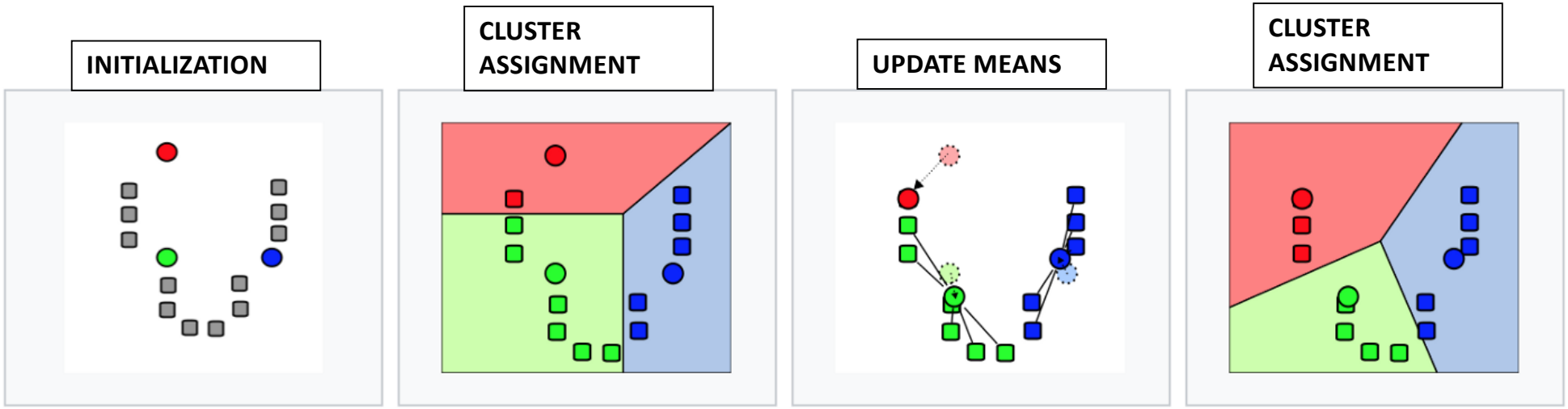
Notes:

- different initialization schemes are possible for the centroids
- alternatively, we can partition the instances into K groups and calculate the initial means from those

```
function KMeans(Instances,K)
randomly initialise K vectors μ1 … μK;
repeat
   assign each x∈Instances to the nearest μj;
   recompute each μj as the mean of the
      instances assigned to it;
until no change in μ1 … μK;
return μ1 … μK;
```

K-means in action!

Source: https://towardsdatascience.com/clustering-using-k-means-algorithm-81da00f156f6



INITIALIZATION

CLUSTER ASSIGNMENT

UPDATE MEANS

```
function KMeans(Instances,K)
randomly initialise K vectors μ1 … μK;
repeat
    assign each x∈Instances to the nearest μj;
    recompute each μj as the mean of the
        instances assigned to it;
until no change in μ1 … μK;
return μ1 … μK;
```

K-means in action!

https://cdn-images-1.medium.com/max/1600/1*4LOxZL6bFl3rXlr2uCiKlQ.gif

Analysis

What does K-means do?

- **within-cluster scatter**: total squared distance between points $\mathbf{x}_i$ and centroid $\boldsymbol{\mu}_k$ in k-th cluster: $\sum_i \left\| \mathbf{x}_i - \boldsymbol{\mu}_k \right\|^2$
- **K**-means attempts to find a configuration $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_K$ that minimises within-cluster scatter summed over all clusters

Does it work?

- The algorithm terminates (converges towards "stable" means/centroids)
- It finds a local optimum from which no further improvement is possible by making local changes.
- It does not necessarily find a global optimum.

Local optimum with K-means

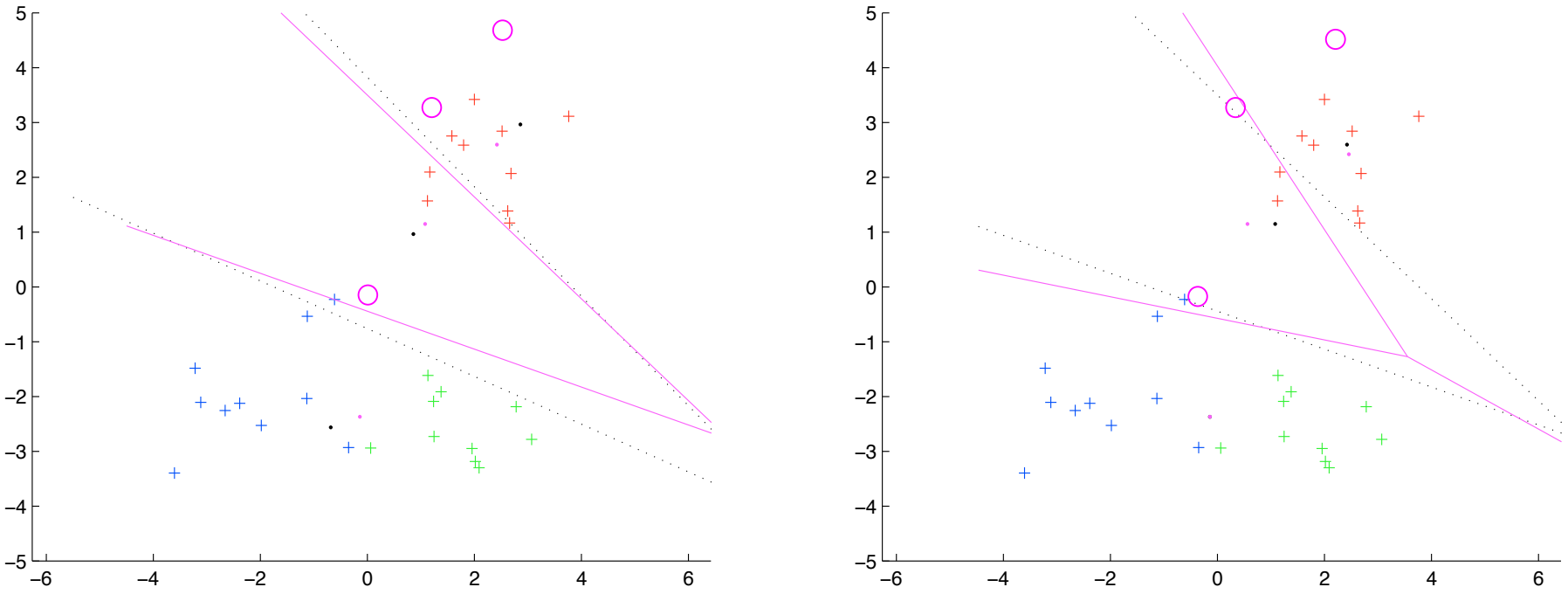


Figure 8.12. **(left)** First iteration of 3-means on the same data as Figure 8.11 with differently initialised centroids. **(right)** 3-means has converged to a sub-optimal clustering.

Discussion

To be reasonably sure that we're not too far from the global optimum, we should run *K*-means a number of times with different initial configurations

- or use more "clever" and/or domain-dependent initialization schemes

***K*-medoids is a variant of *K*-means in which cluster center is a data point (rather than the mean of data points) that minimises within-cluster scatter**

- allows to use distance metrics other than Euclidean distance, which tends to be sensitive to outliers

We can also use "soft" assignment of data points to cluster centers –> Gaussian mixture models

Hierarchical Clustering

Agglomerative hierarchical clustering: iteratively merging the closest pair of points/clusters

- Given: an *n-by-n* matrix **D** of all pairwise distances (e.g., Euclidean) between **n** data points
- Let $\mathbf{d}_{ij}$ be the minimum of **D**, i.e., $\mathbf{x}_i$ and $\mathbf{x}_j$ are the two closest data points
- Merge $\mathbf{x}_i$ and $\mathbf{x}_j$ into a new cluster $\mathbf{x}'$, compute distances of all other points to $\mathbf{x}'$ (see next slides), and compute a new *(n-1)-by-(n-1)* distance matrix $\mathbf{D}'$
- Iterate until only a single cluster is left
- Output a dendrogram

Hierarchical Clustering

Agglomerative hierarchical clustering: iteratively merging the closest pair of points/clusters

Advantage: no need to choose number of clusters (*k*) in advance

- can obtain any number of clusters from dendrogram

Disadvantages: doesn't scale well

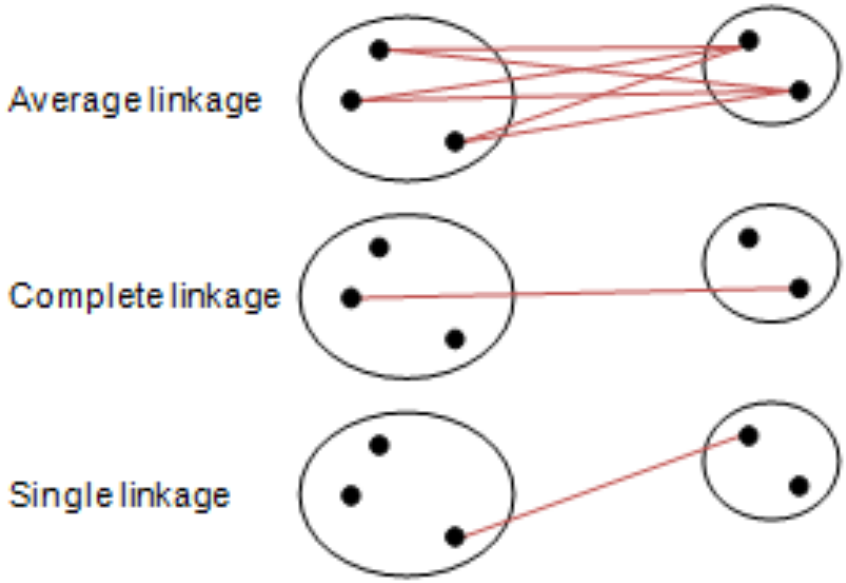
- time complexity $\mathbf{O(n^3)}$ to $\mathbf{O(n^2 \log n)}$

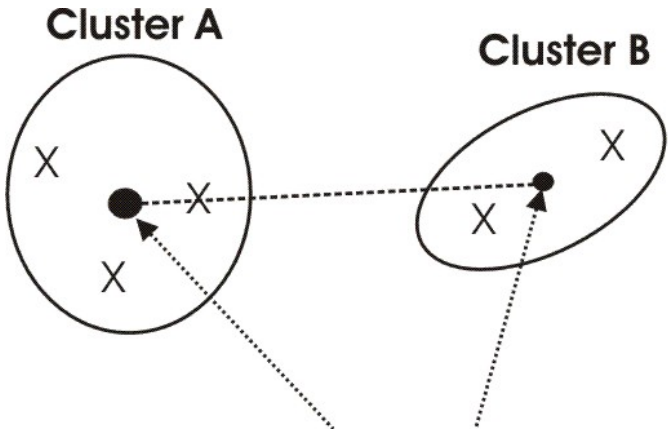
Clusters only apply to given data

Linkage

Distance between clusters can be calculated as:

- the minimum distance between pairs from each cluster (single linkage)
- the maximum distance between pairs from each cluster (complete linkage)
- the average distance between pairs from each cluster (average linkage)
- the distance between the centroids of each cluster (centroid linkage)

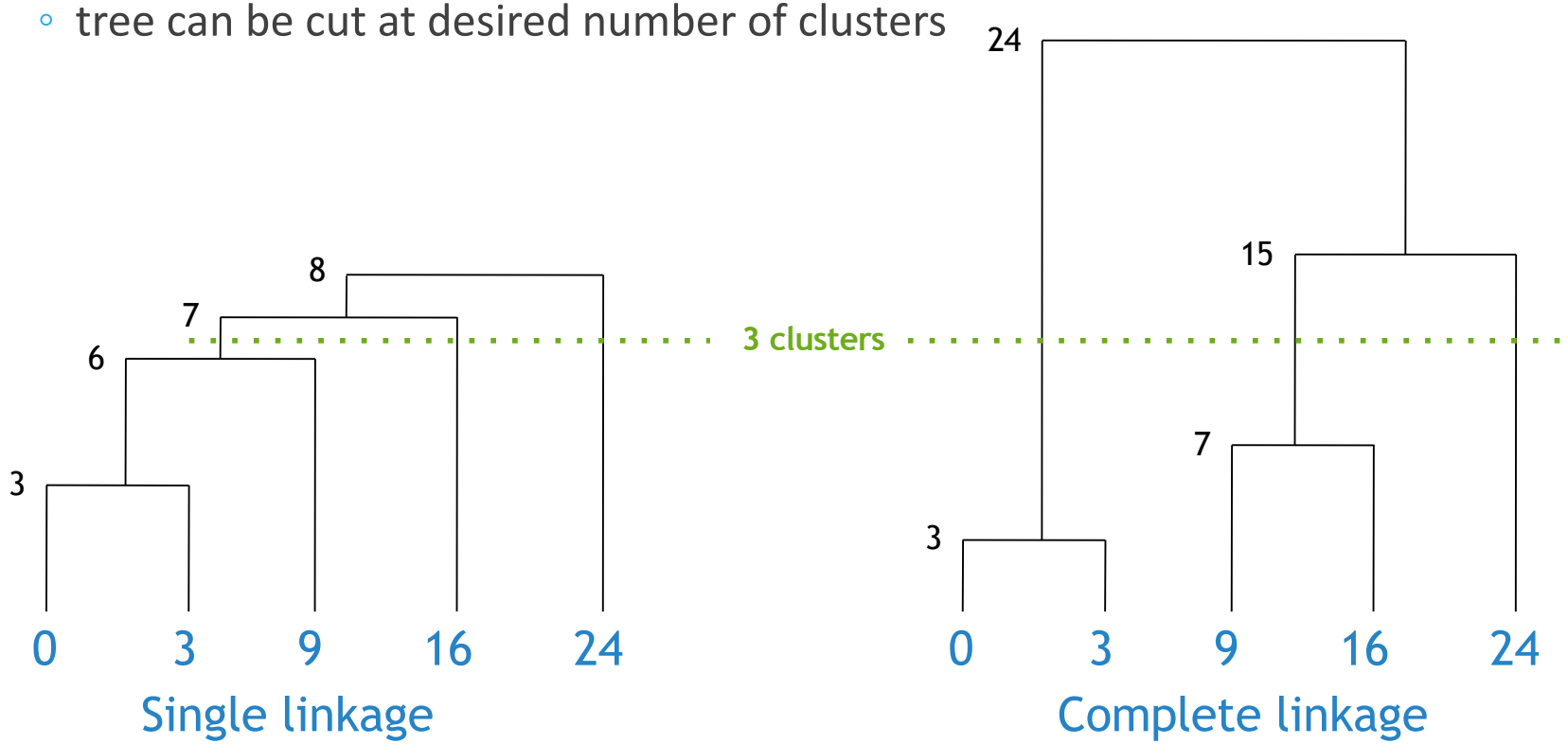




Dendrogram

Tree where each internal node corresponds to a pair of clusters merged in an iteration

- the height of each node indicates distance between clusters
- tree can be cut at desired number of clusters



Silhouettes

a(x) is average distance from a point, to points in its own cluster

b(x) is average distance from a point, to points in nearest cluster

s(x) = b(x)-a(x)/max(a(x),b(x)); should be large

Silhouette plots **s(x)** for each **x**, grouped by cluster

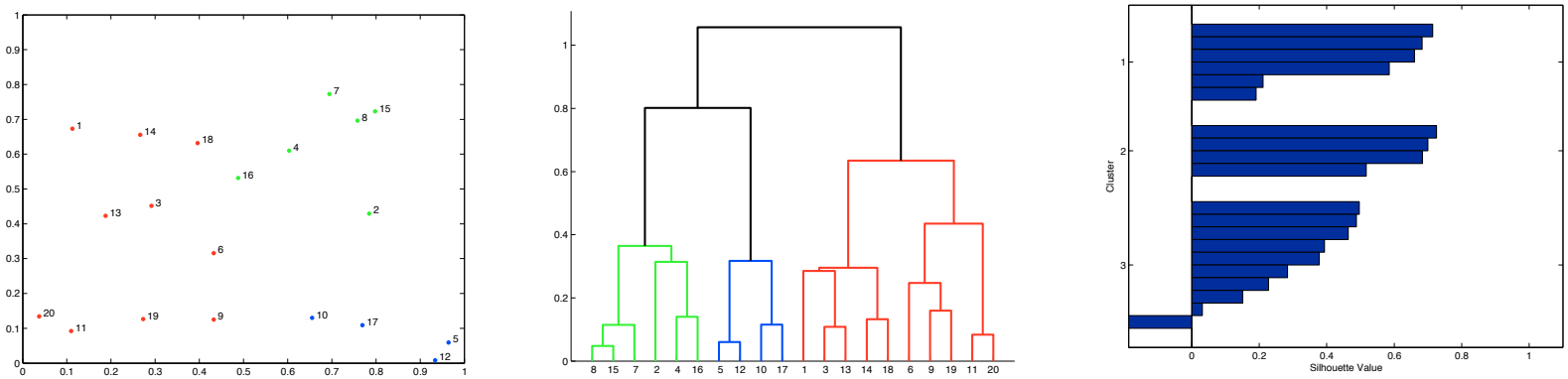


Figure 8.18. (left) 20 data points, generated by uniform random sampling. **(middle)** The dendrogram generated from complete linkage. The cluster structure suggested by the dendrogram is mostly spurious as it cannot be observed in the data. **(right)** The rapidly decreasing silhouette values in each cluster confirm the absence of a strong cluster structure. Point 18 has a negative silhouette value as it is on average closer to the green points than to the other red points.