
UKRI
Science and Technology Facilities Council
Hartree Centre
Welcome

Science and Technology Facilities Council

Hartree Centre

Week 6: Clustering

Introduction to key clustering algorithms and approaches

Michalis Smyrnakis
Artificial Intelligence Group Leader, Hartree Centre

Lecture Outline

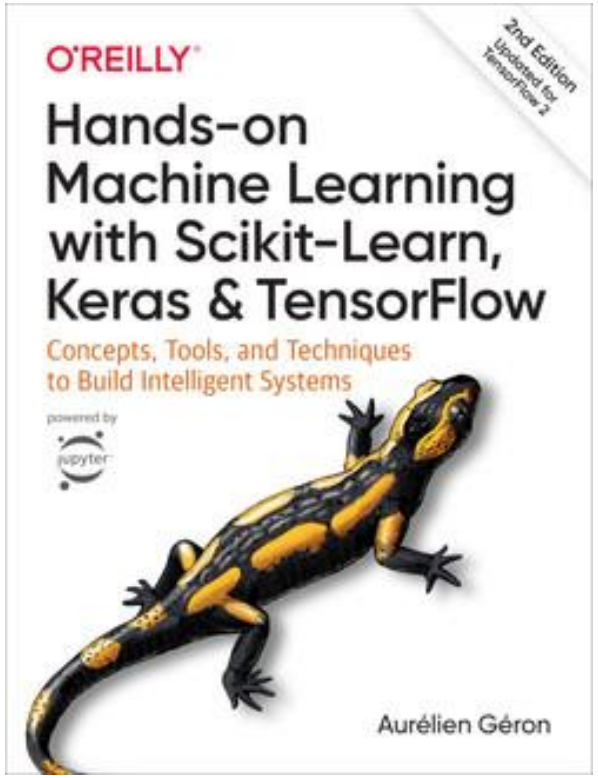
- Today we'll focus on clustering algorithms
- We saw a few examples in week 1
- Clustering is a fairly intuitive concept but it's hard to do well, especially for big datasets
  - Intuitive algorithms cluster well but don't scale well to large datasets
  - More scalable algorithms don't always cluster well.
- Towards the end of the lecture, we'll introduce an advanced algorithm that clusters well and scales well too.
- Lecture outline:
  - Clustering examples
  - Clustering algorithms:
    - Hierarchical agglomerative clustering
      - Intuitive and works well, but only on small datasets
    - K-means clustering:
      - Perhaps the most widely used clustering algorithm in practice
      - Scales well to large datasets, but doesn't always produce sensible clusters
    - DBSCAN
      - Scales well and often produces good clusters

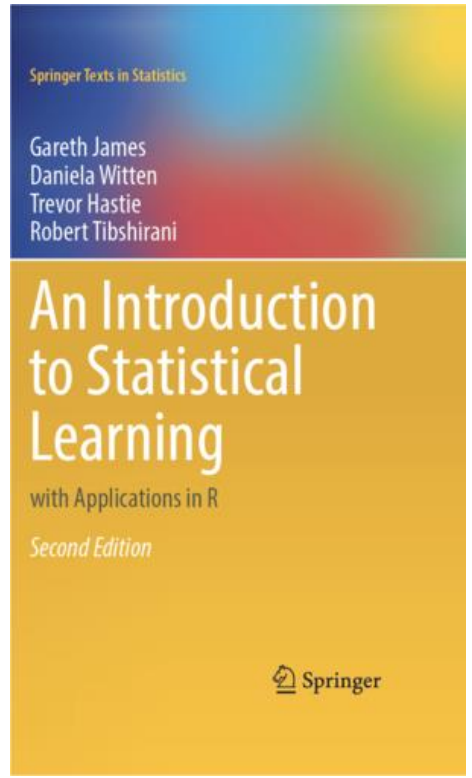
Lecture Outline Continued

- For each clustering algorithm:
  - Informal explanation
  - Algorithm
  - Parameter selection
  - Pros and cons
- 10 min break after the first 50 minutes

Recommended Reading for Clustering

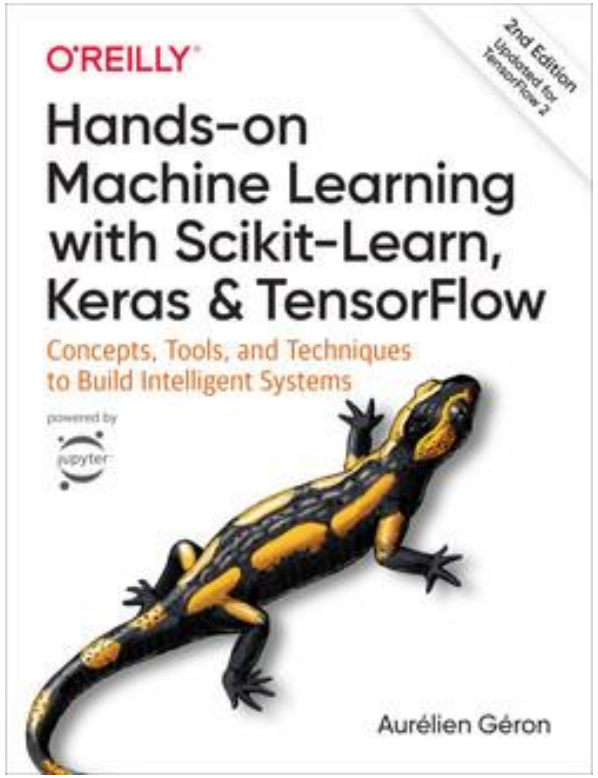
- This lecture is self-contained, but the following might also be useful
  - Hands-on Machine Learning, 2nd Edition 2019
    - Available online from the university library:
    - https://libguides.liverpool.ac.uk/online
    - Chapter 9
    - No maths
    - Examples in Python
  - An Introduction to Statistical Learning
    - Free online: https://www.statlearning.com/
    - Section 12.4
    - Good for mathematical aspects of ML in general (including clustering)
    - Examples in R (not used in this course)
    - Accompanying lectures:
    - https://web.stanford.edu/~hastie/lectures.htm





Code for Today's Lecture

- Most code can be found in Hands-on Machine Learning, Chapter 9
- Extra code is provided in the slides



Unsupervised learning

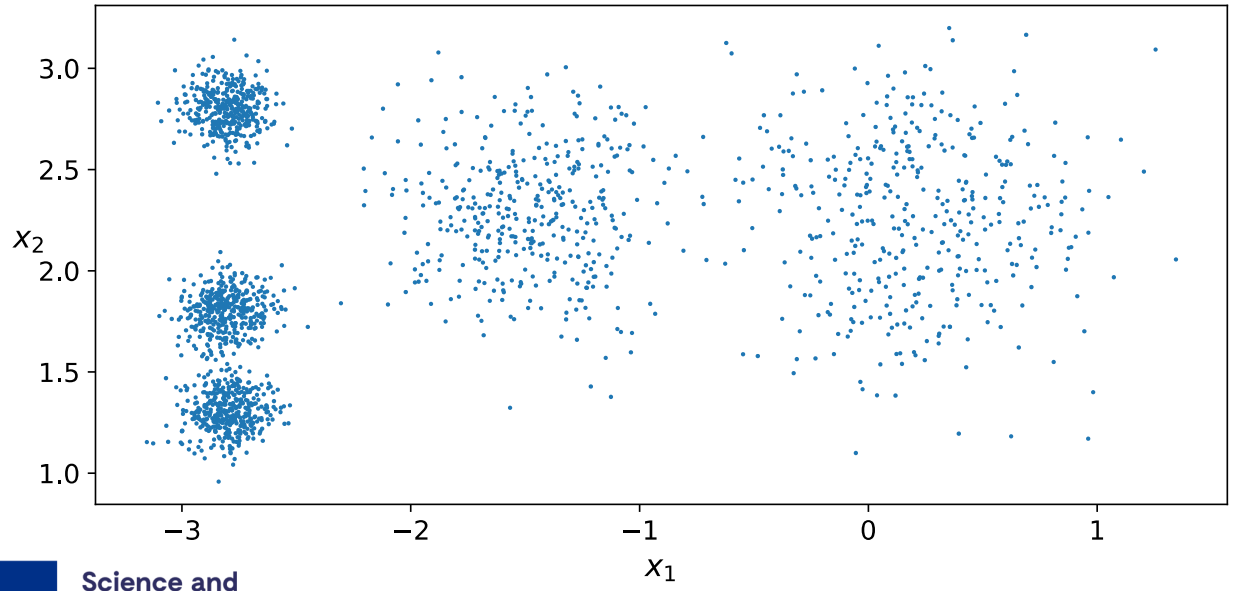
- A type of machine learning where the algorithm learns patterns and structures directly from the data without any labelled examples.
- No ground truth: unsupervised learning algorithms do not have access to predefined class labels or target variables.
- Pattern discovery: the goal is to discover hidden patterns, structures, or relationships within the data.
- Applications: Clustering, dimensionality reduction, anomaly detection.

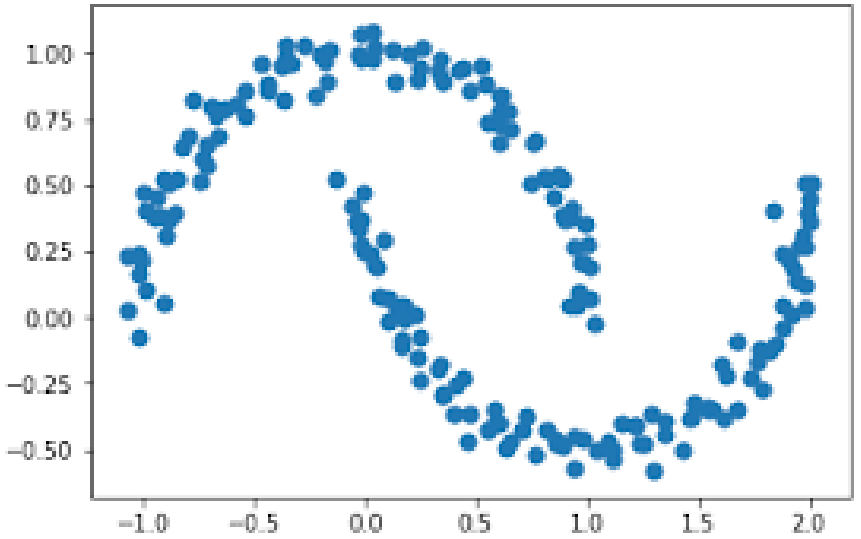
Unsupervised learning advantages and disadvantages

- Advantages:
  - Can discover hidden patterns that humans might not notice.
  - Can be used for exploratory data analysis.
  - Can be applied to large datasets with no labelled data.
- Disadvantages:
  - Can be challenging to evaluate the quality of the results.
  - May require domain knowledge to interpret the findings.

Clustering

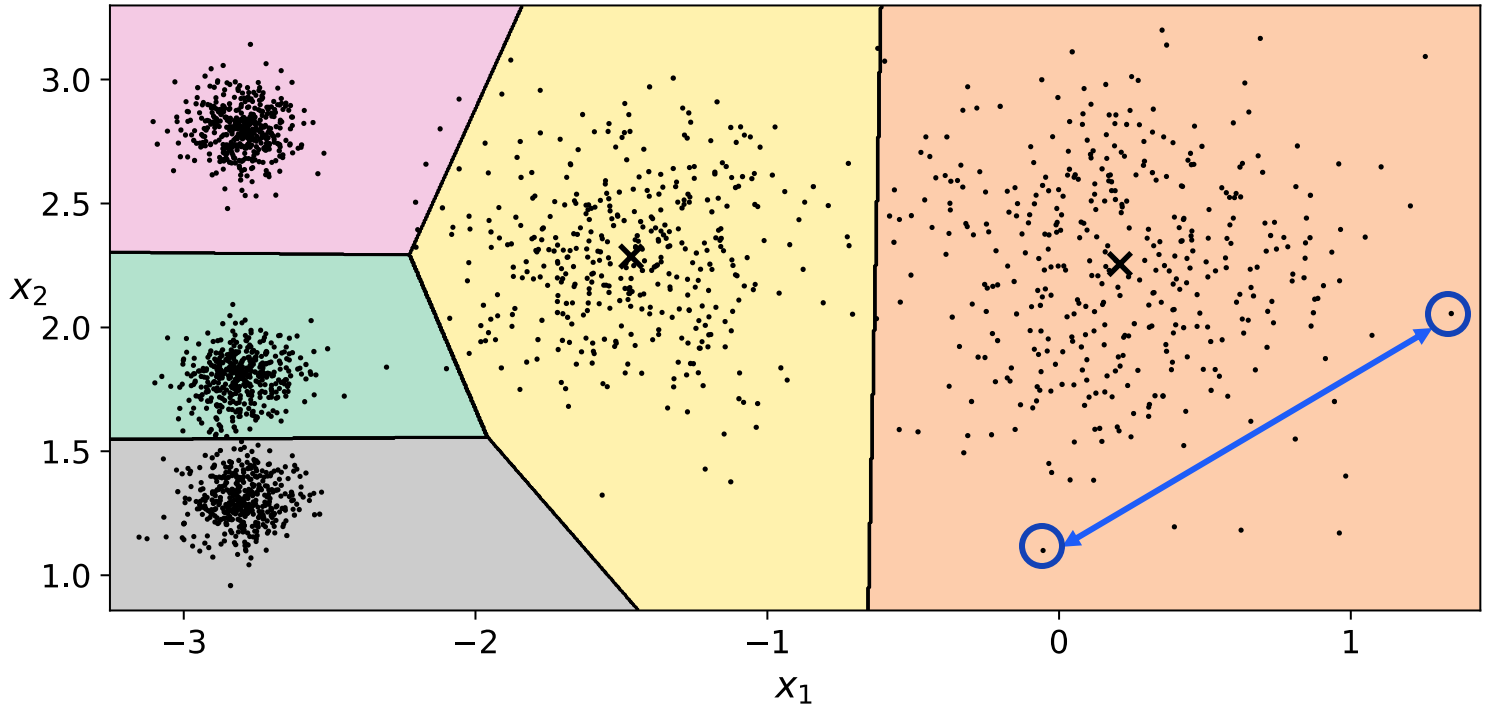
- Clustering algorithms are used to discover patterns of similarity in data
- To illustrate, consider the simple datasets below
  - How many clusters are there?
  - How would you define a cluster?
  - How might you automatically identify these clusters?





Clustering Algorithms

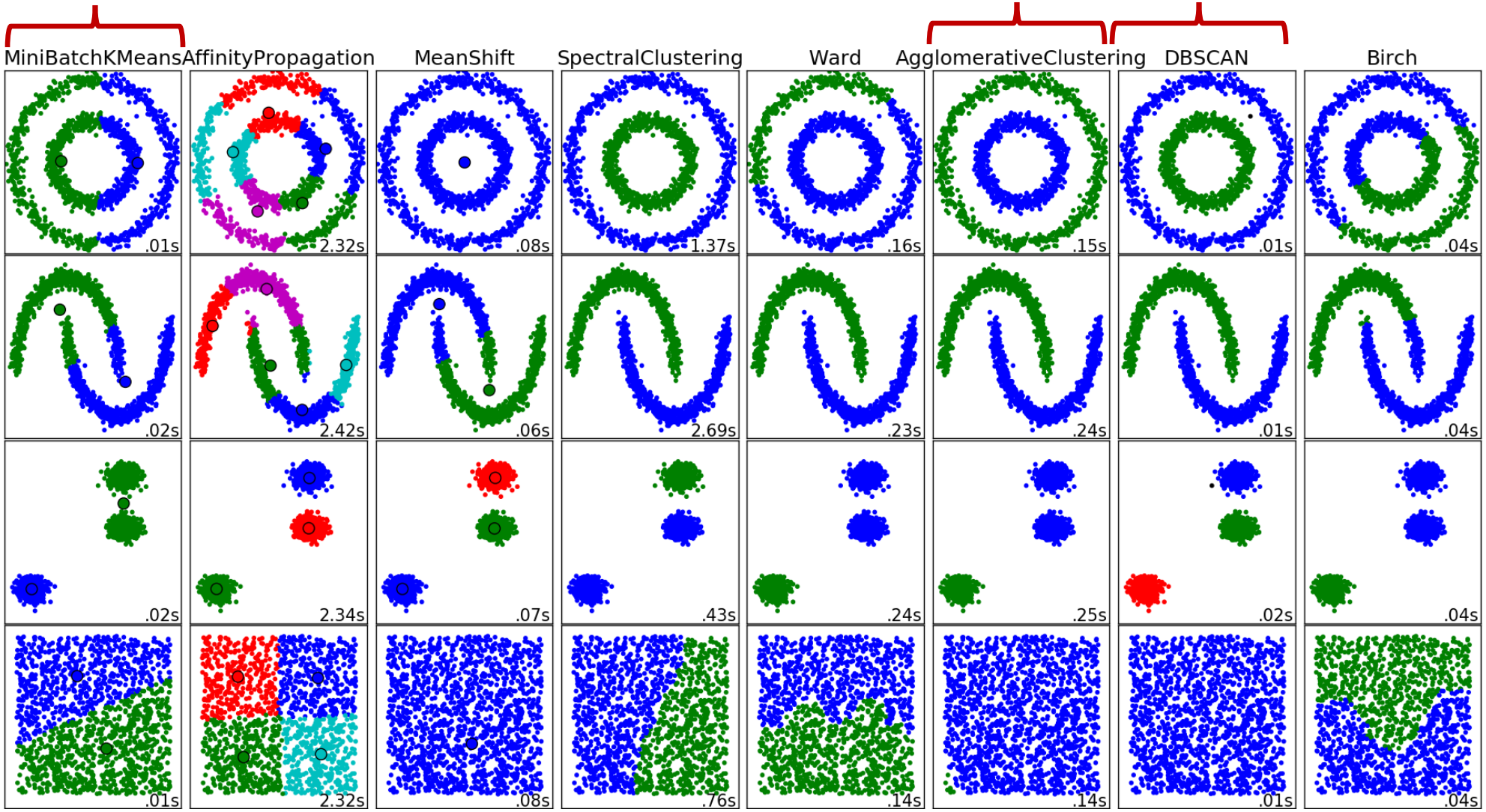
- The clustering below was done by the k-means algorithm, which we'll see shortly
- Why would you want to cluster?
  - Clustering has made the data a bit more decipherable:
    - We know that it is an amalgamation of 5 sub-groups
    - We could look more closely at these 5 sub-groups to understand more about them
  - This dataset is so simple that we could immediately see the clusters, but we can use the same clustering techniques for far more complicated datasets, where the clustering is not at all obvious.
    - E.g. datasets that can't even be visualized



All we need for clustering is a measure of dissimilarity between data samples (in this case, it's just Euclidian distance).

Comparison of Clustering Algorithms

- Plot shows various clustering algorithms available in sklearn
- We'll look at k-means, agglomerative and DBSCAN
- Time and space complexity are also important, especially for big data applications

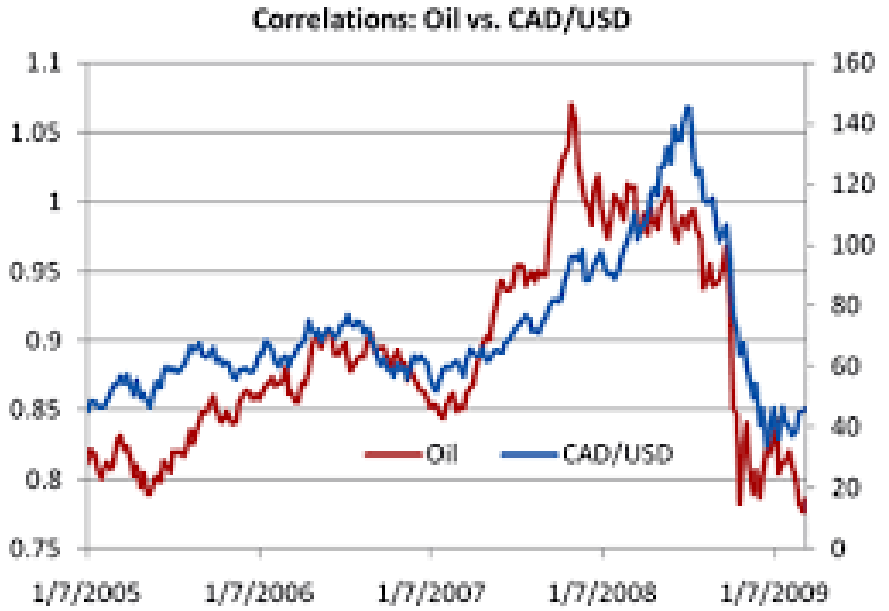


Science and Technology Facilities Council

Hartree Centre

Distance and Dissimilarity

- In the above example, clustering was based on the Euclidian distance between points, but we can view the problem in a more abstract way, which makes clustering more generally applicable
- For this, we use the concept of dissimilarity, which generalizes the concept of distance
- Euclidian distance is a good dissimilarity measure for points in Euclidian space (e.g. the dataset in the previous slide)
- How could we measure dissimilarity for text documents?
- How could we measure the dissimilarity between timeseries?



Science and Technology Facilities Council
Hartree Centre

Distance properties

- Non- negativity $d(x, y) \geq 0$
- Identity of indiscernible $d(x, y) = 0 \; if \; and \; only \; if \; x = y$
- Symmetry $d(x, y) = d(y, x)$
- Triangle inequality $d(x, y) \leq d(x, z) + d(z, y)$

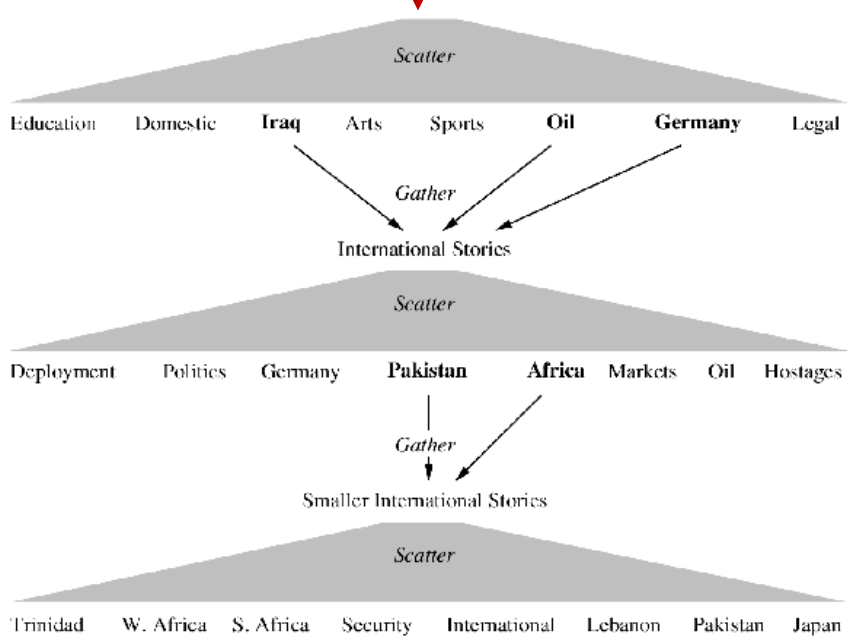
Popular distances and dissimilarities

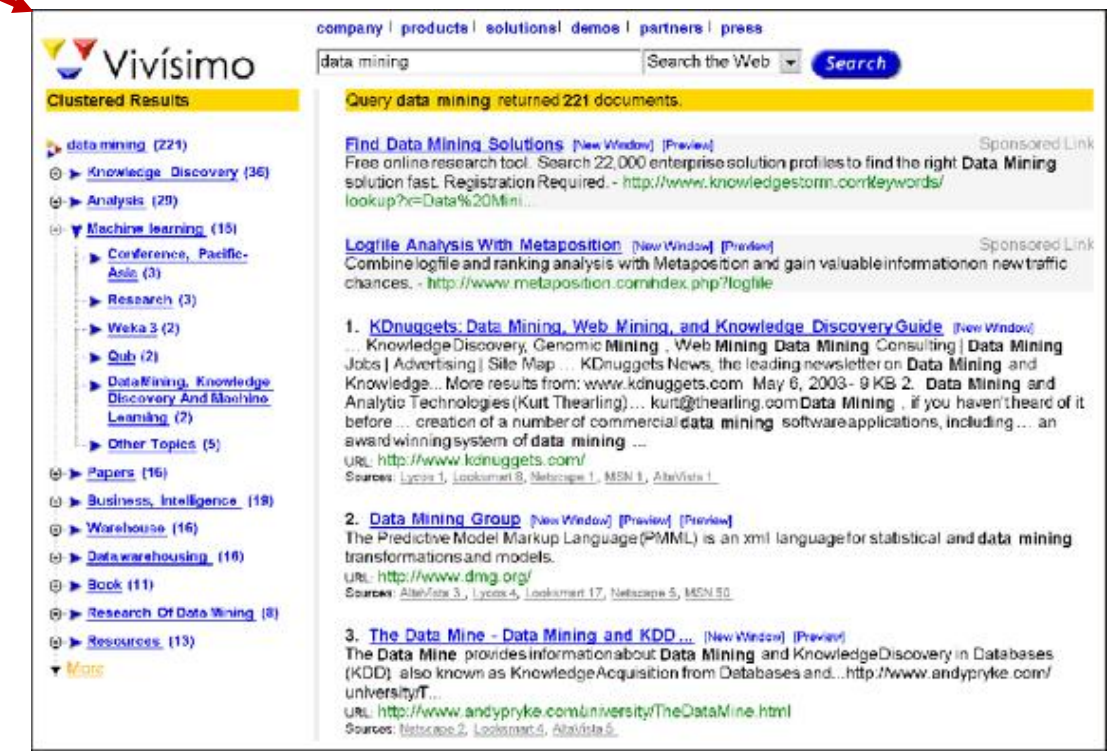
- Euclidean distance: $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
  - Measures straight-line distance between two points in Euclidean space.
- Cosine dissimilarity: $\text{similarity}_{\cos} = \frac{xy}{\|x\|\|y\|}$
  - dissimilarity_measure=1- $\text{similarity}_{\cos}$
  - Measures the angle between two vectors (commonly used in text analysis).
- Mahalanobis distance: $d(x, y) = \sqrt{(x - y)^T S^{-1}(x - y)}$
  - Accounts for correlations between variables, often used for identifying outliers.

Clustering Examples

Clustering for Search Engine Design

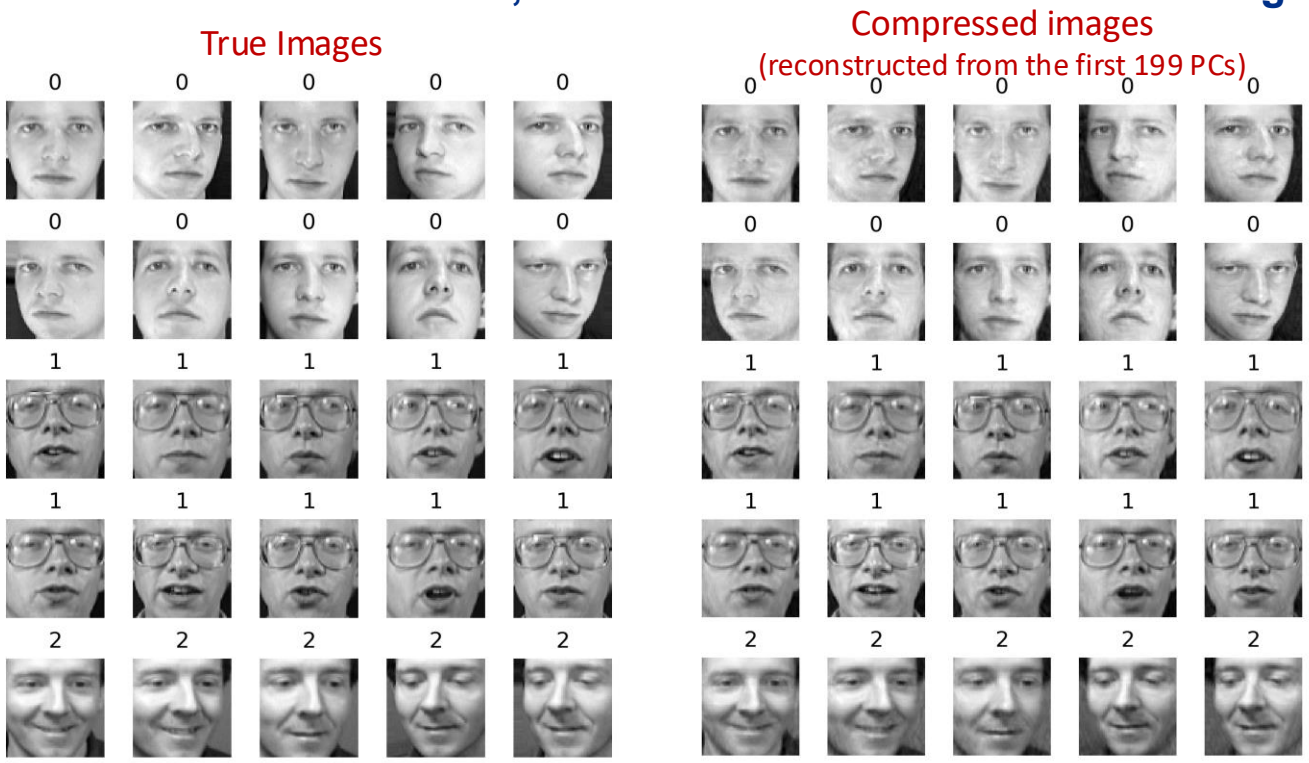
- In week 9, we'll look at applications of clustering for search engine design
  - Search result clustering
  - Scatter-gather search

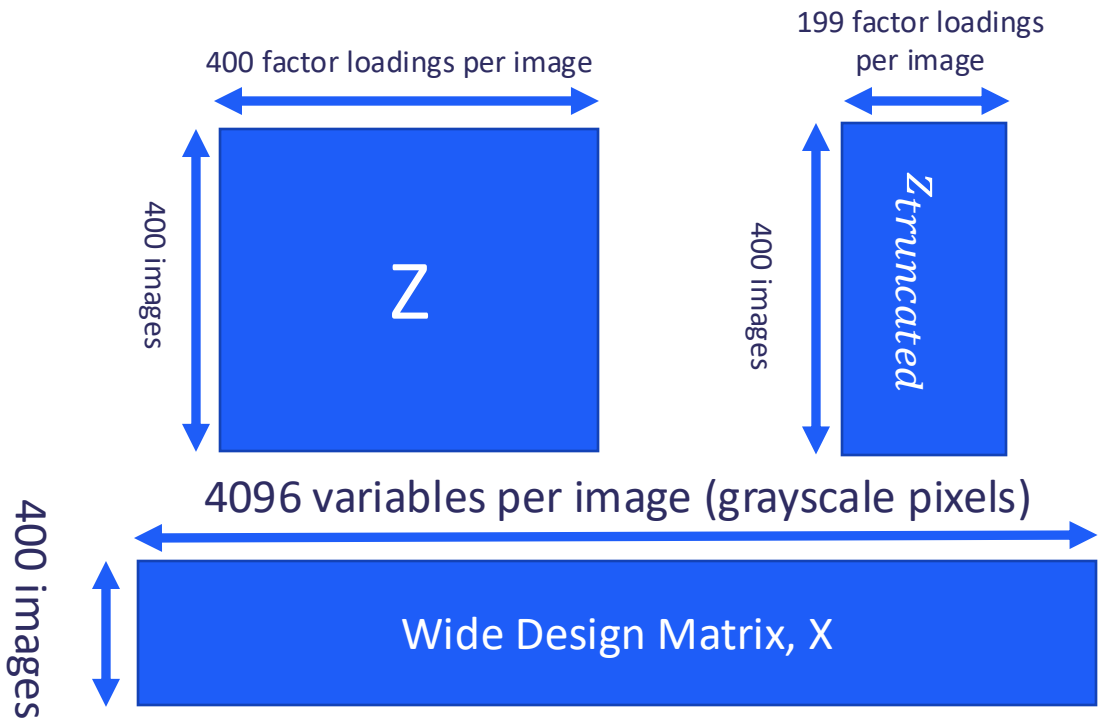




PCA Factor Loadings

- From only the first 199 principal components, we get a very good representation of each face, with 99% of the variance in the original data explained – see the plot below.
- In order to discard the remaining 201 PC and factor loadings, we:
  - Truncated the factor loading matrix F, by removing the last 201 rows.
  - Truncated the PC matrix by removing the last 201 rows.
- Recall that X can approximately be reconstructed from these truncated matrices, $X \approx Z_{truncated} V_{truncated}^{T}$
- Hence, we cluster on 199 factor loadings rather than 4096 pixels
- For more details, see the "Hands-on Machine Learning" book



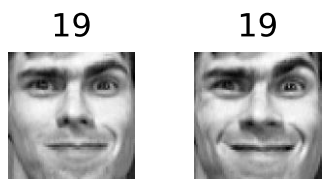


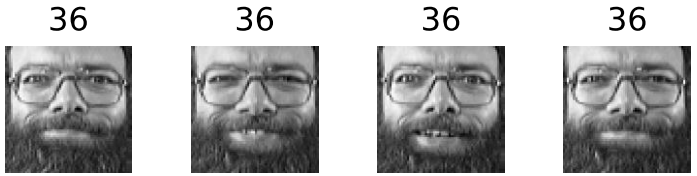
First 25 PCs

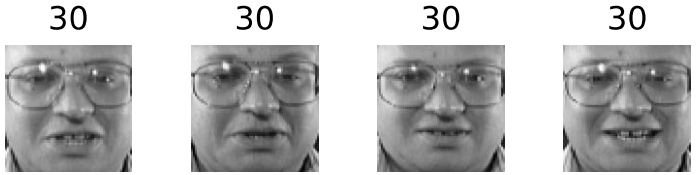
- We saw in the previous slide that PCA preprocessing reduces the dimensionality of the dataset before clustering
- However, there's another more subtle reason to use PCA as a preprocessing step for clustering…
- The plot opposite shows the first 25 principal components of the Olivetti faces dataset
- The first principal component is a generic face
- Subsequent principal components are offsets to provide incrementally better face reconstructions
- See if you can spot the PCs that:
  - Add glasses
  - Add smiles or smirks
  - Add cheekbones
  - Broaden or narrow the face
  - Lengthen or shorten the nose
- It's these features that we're comparing when we compute the Euclidian distance between principal component factor loadings

Clustering the Olivetti Faces Dataset

- We can then run a clustering algorithm using Euclidian distance on a truncated PCA factor loading matrix







Not perfect, but the clusters are a lot more homogeneous than the original dataset

Science and Technology Facilities Council
Hartree Centre

Compression and Image Segmentation

- We can also cluster the colors in images
  - This reduces the size of the images, which simplifies ML, e.g. neural networks
  - Also makes semantic segmentation easier
- Similarity measure is the Euclidian distance between RGB values for each pixel

Compression and Image Segmentation

Original image

10 colors

8 colors

6 colors

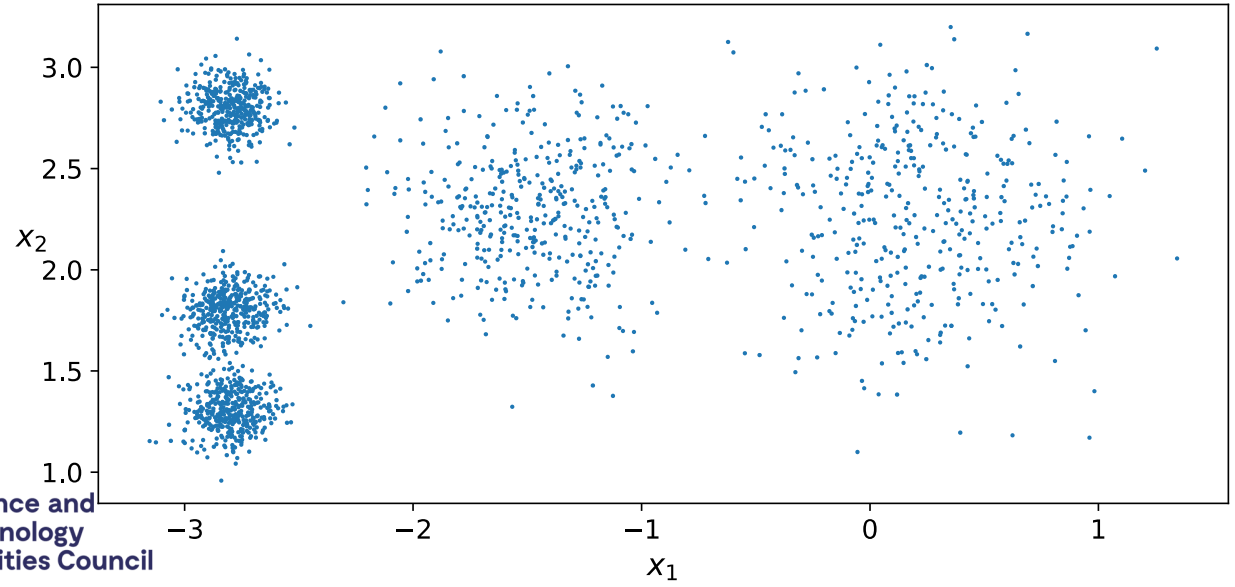
4 colors

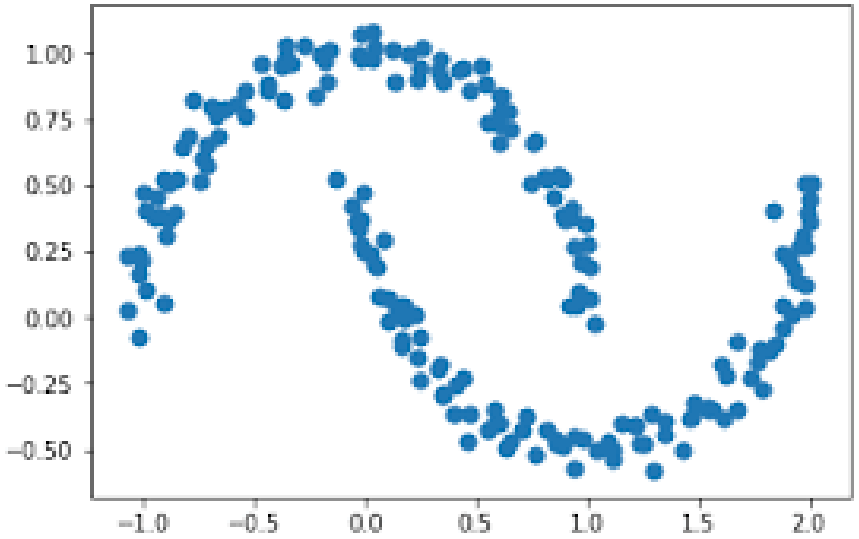
2 colors

Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC)

- We'll start with the most intuitive clustering algorithm, which goes by the name hierarchical agglomerative clustering (HAC)
- How would you cluster these datasets?
  - Maybe start by joining the closest 2 points to form a cluster
  - Then find the next two closest points and join them
  - Take a representative point for each cluster (e.g. the mean), and combine points with clusters if they're close too
  - That's the idea behind HAC





Hierarchical Agglomerative?

Basically means "Bottom Up"…

Merriam-Webster

hierarchical adjective

Save Word

hi·er·ar·chi·cal | \ ˌhī-(ə-)ˈrär-ki-kəl *also* hir-ˈär- \

variants: *or* **hierarchic** \ ˌhī-(ə-)ˈrär-kik *also* hir-ˈär- \

Definition of *hierarchical*

: of, relating to, or arranged in a hierarchy

// a hierarchical society

agglomeration noun

Save Word

ag·glom·er·a·tion | \ ə-ˌglä-mə-ˈrā-shən \

Definition of *agglomeration*

1 : the action or process of collecting in a mass

// the agglomeration of matter into stars and galaxies

2 : a heap or cluster of usually disparate (see DISPARATE sense 1) elements

// … an agglomeration of 100-year-old cottages with gingerbread scroll-saw ornamentation.

— Ira Henry Freeman

3 : a large, densely and contiguously populated area consisting of a city and its suburbs

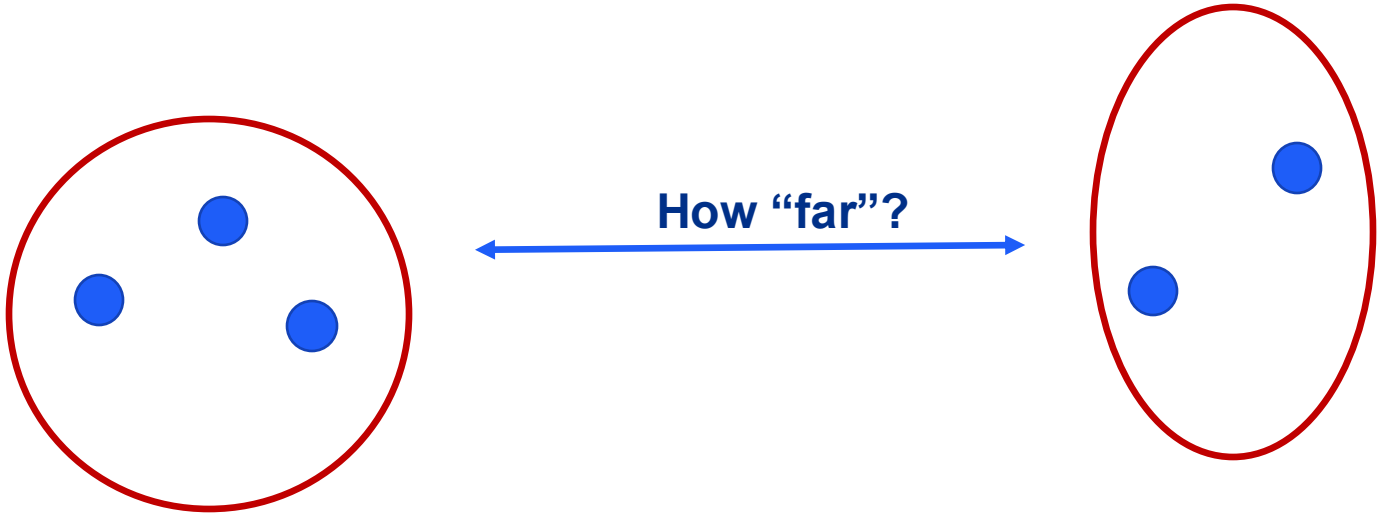
// an urban agglomeration

UKRI Science and Technology Facilities Council

Hartree Centre

Linkages

- When we're comparing clusters in the HAC algorithm, we need a measure of dissimilarity between clusters
  - This isn't a simple as just using the pointwise dissimilarity function, because that's only defined between two points, and we're concerned with dissimilarities between clusters of points
- There are three ways that one can extend the concept of dissimilarity to clusters
- We refer to these as "linkages"…

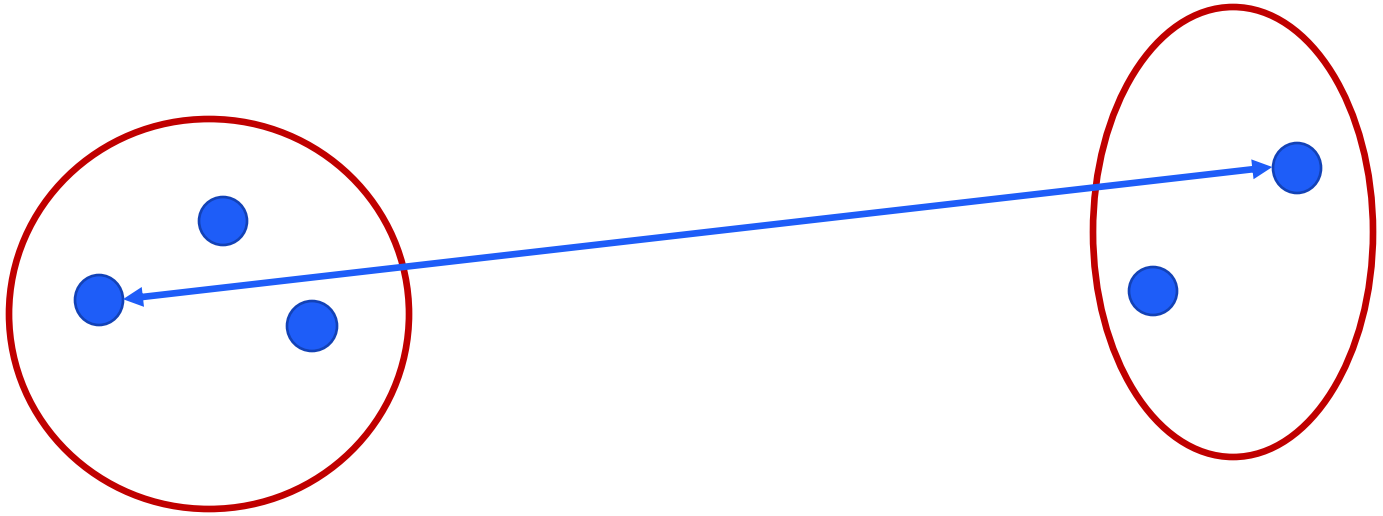


Complete Linkage

- **Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the largest of these dissimilarities.**

- $d(A, B) = max\{d(x, y) | x \in A, y \in B\}$

- **Tends to create more compact, spherical clusters.**

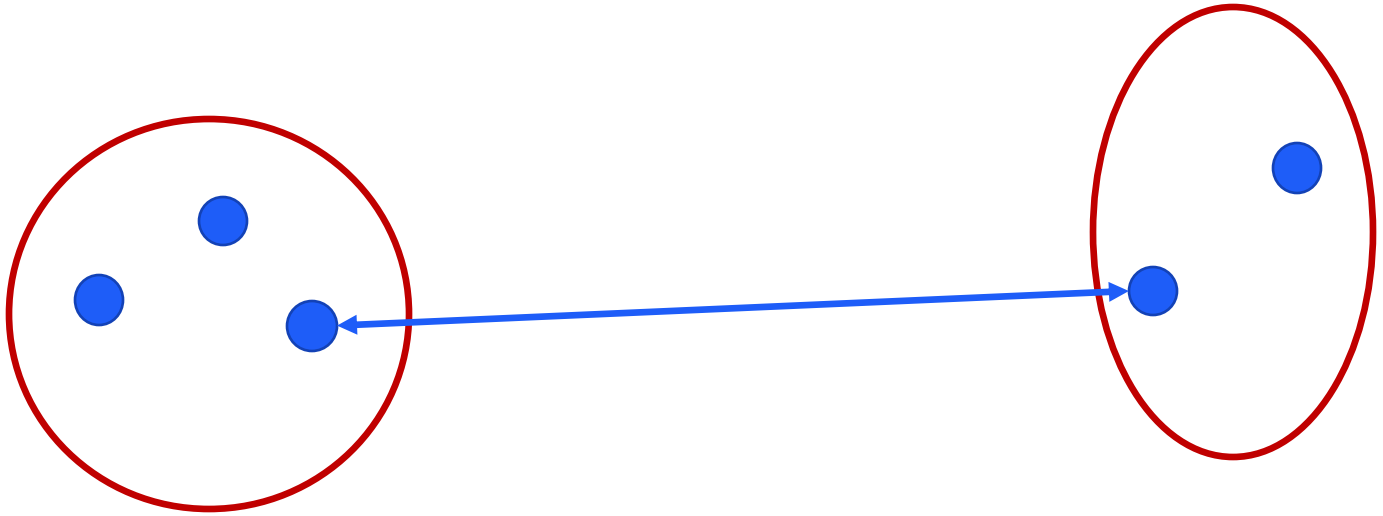


Single Linkage

- Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities

- $d(A, B) = min\{d(x, y) | x \in A, y \in B\}$

- Typically creates long, chain-like clusters.

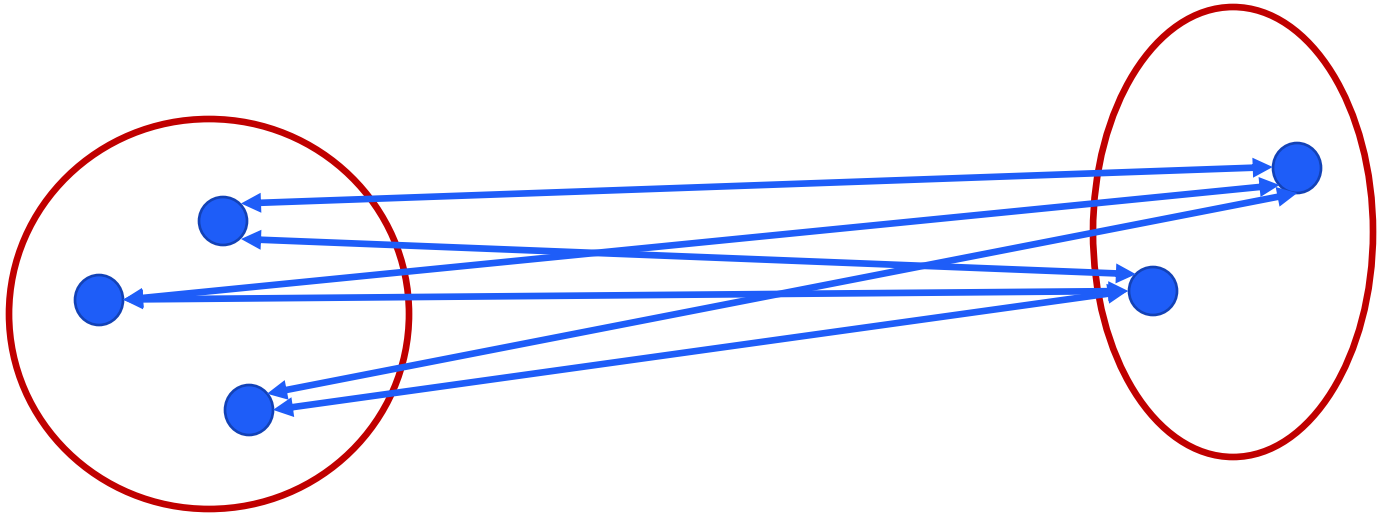


Average Linkage

- Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities.

- $d(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$

- Produces clusters that are intermediate between single and complete linkage.



Ward’s linkage

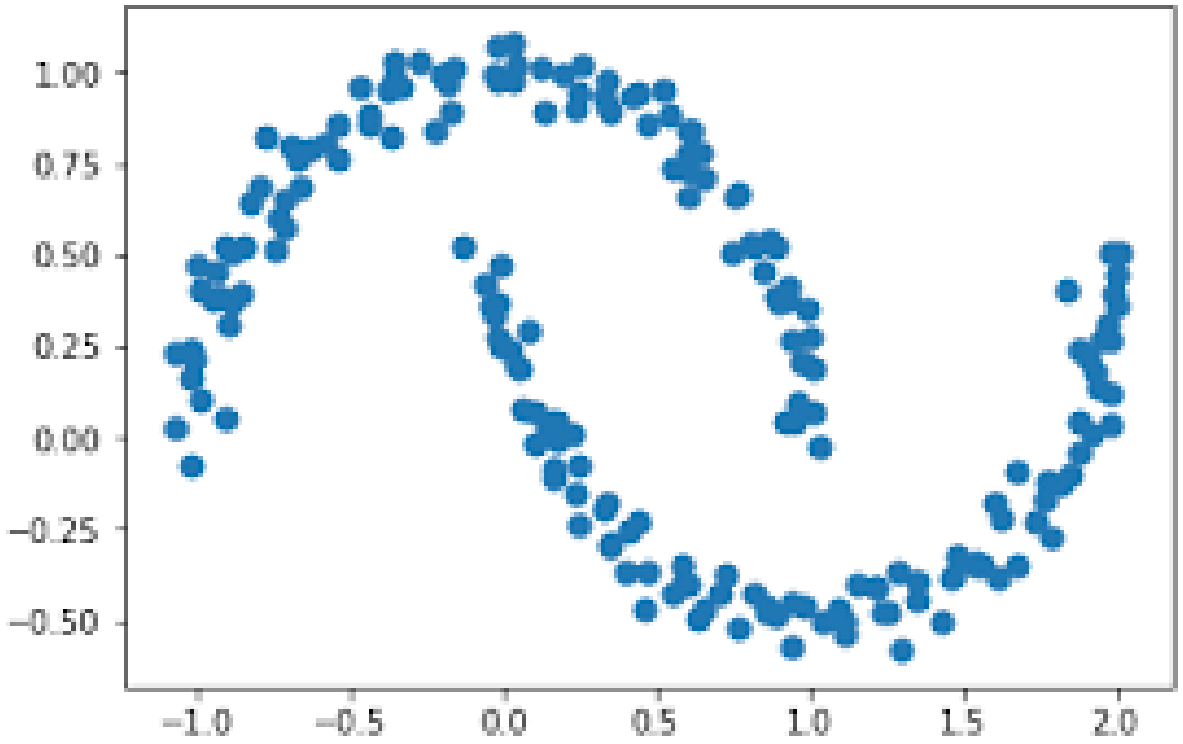
- Minimize the increase in the sum of squared deviations within clusters after each merge.
- Ward's linkage seeks to minimize the variance within clusters, resulting in clusters that are relatively compact and spherical.

$$\Delta(\mathrm{A},\mathrm{B}) = \sum_{i \in A \cup B} \|x_i - c_{A \cup B}\| - \sum_{i \in A} \|x_i - c_A\| - \sum_{i \in B} \|x_i - c_B\| = \frac{|A||B|}{|A| + |B|} \|c_A - c_B\|^2$$

- Outliers can have a significant impact on Ward's linkage clustering, as they can create large clusters that include points that are not closely related.
- Ward's linkage can be computationally expensive for large datasets

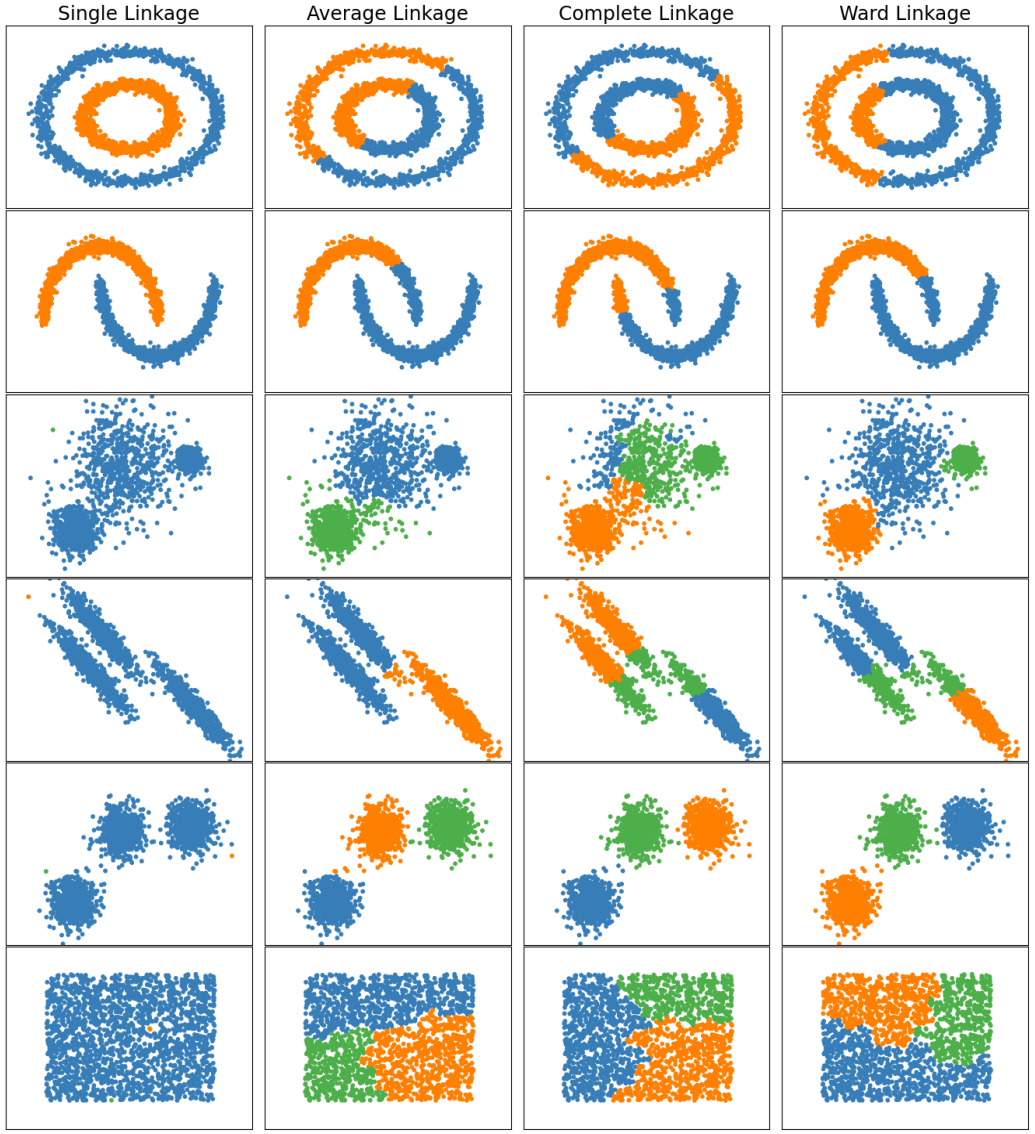
HAC Example

- **Q: How will HAC perform on the half moon dataset?**



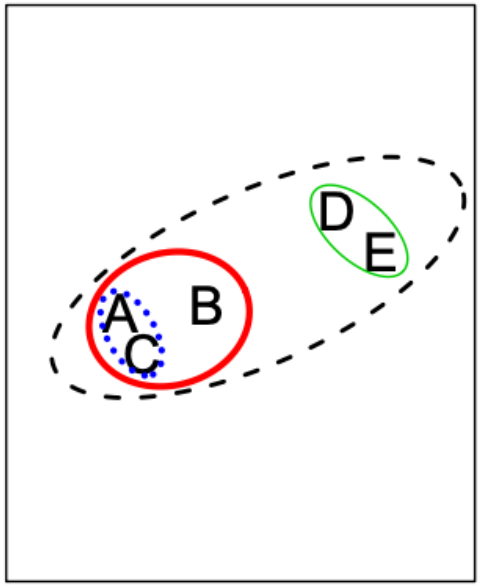
Comparison of Linkages

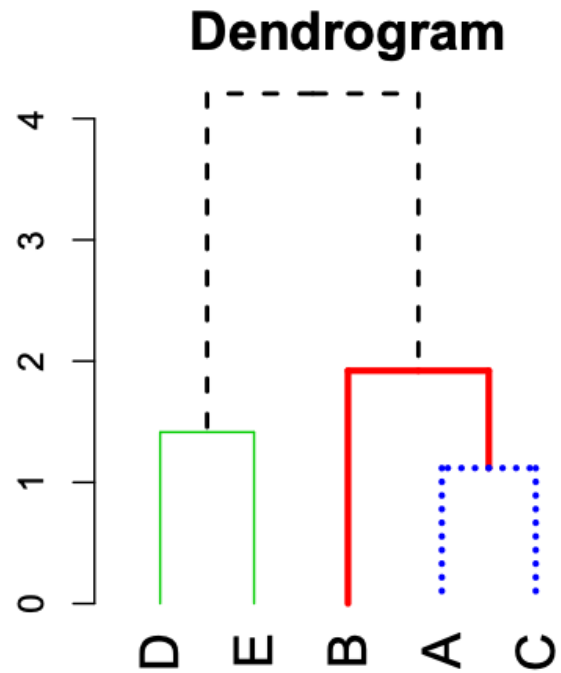
- A: It depends on the linkage. In particular:
  - Single linkage is fast, and can perform well on non-globular data, but it performs poorly in the presence of noise.
  - Average and complete linkage perform well on cleanly separated globular clusters, but have mixed results otherwise.



HAC Continued

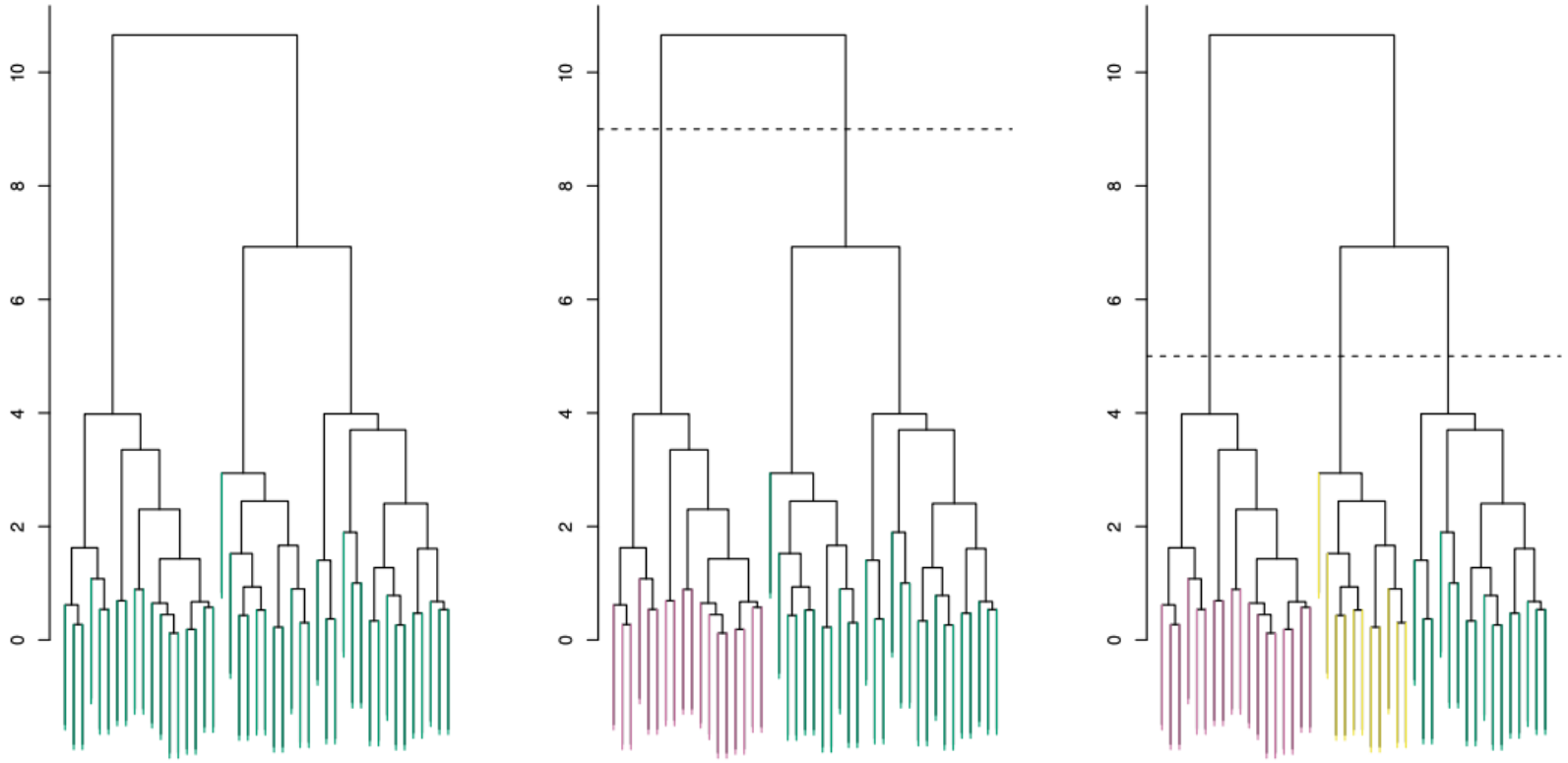
- HAC join points together into clusters sequentially, based on dissimilarity.
  - Start by joining the two points that are least dissimilar to each other, to form the first cluster.
  - Thereafter, repeatedly identify the point (or cluster) which is least dissimilar to another point (or cluster) and join them to form a new, larger, cluster.
- This forms a tree structure for the data, which expresses sensible clustering partitions for a range of possible numbers of clusters
- We depict the tree structure in a ***dendrogram***





Cutting the Dendrogram

- Any horizontal cut through the dendrogram will give a set of clusters.
  - Depending on where you cut, you'll get between 1 and N clusters, with fewer clusters for a higher cut



The HAC Algorithm

- A concrete algorithm for hierarchical agglomerative clustering:
  1. Build an NxN matrix, C, of dissimilarities between all points
  2. Consider all the points to be single-element clusters
  3. For i = 1..N-1:
     a. Traverse the matrix to find the smallest dissimilarity
     b. Merge the two clusters with the smallest dissimilarity
     c. Compute the dissimilarities of this new cluster to all the other clusters

- The loop only needs to be executed N-1 times because there are N clusters initially, and each iteration of the loop results in one less cluster.

HAC Algorithm Complexity

- The time complexity of the above algorithm is $\Theta(N^3)$ because we exhaustively scan the N × N matrix C for the smallest dissimilarity in each of N − 1 iterations.
- Space complexity for the above algorithm is $\Theta(N^2)$, because we must store the N × N matrix C:
  - E.g. 8GB of RAM would be entirely filled by a matrix representing 32 bit float dissimilarities for 23k points (exercise)
- Other HAC algorithms use a more efficient priority-queue search algorithm, giving a time complexity of $\Theta\left(N^2 log(N)\right)$.
  - E.g. see Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

Divisive Clustering: The Top-Down Approach

- Divisive clustering (also known as **Top-Down Hierarchical Clustering**) starts with all data points in a single cluster and recursively splits them into smaller clusters.
  - The process continues until each data point forms its own cluster or a stopping criterion is met (e.g., desired number of clusters).

- **Divisive**: Splits clusters until individual points are reached. When **Agglomerative** merges points until one large cluster is formed.

Science and Technology Facilities Council

Hartree Centre

10 Minute Break

Science and Technology Facilities Council

Hartree Centre

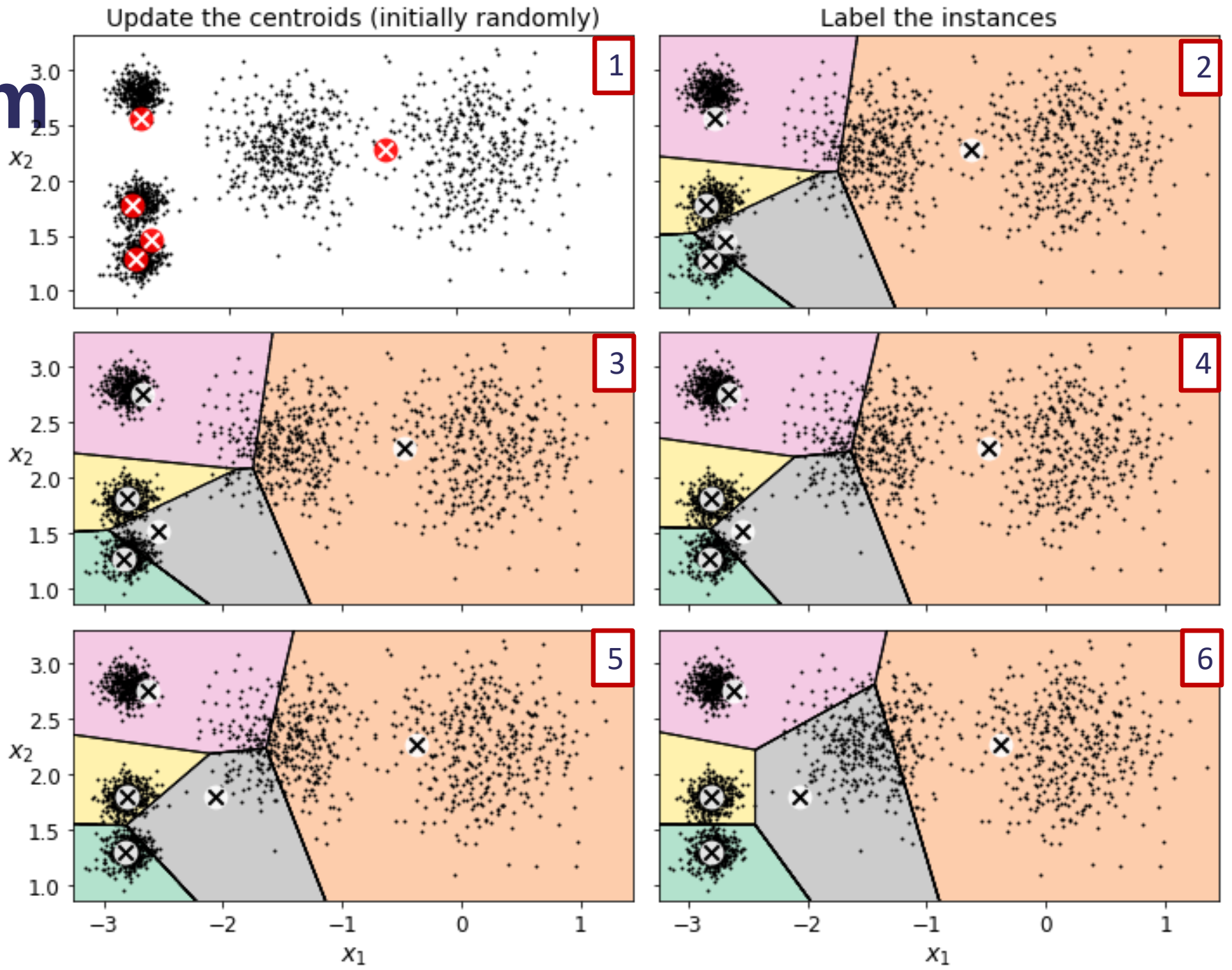
K-Means Clustering

K-Means Clustering

- We saw that HAC works well on some datasets, but doesn't scale well to large datasets
- K-Means takes a different approach to clustering, which scales very well (it's $O(N)$)
- The number of clusters, k, is specified beforehand
- The k-means algorithm the finds k ***cluster centroids***, which represent the centers of the k clusters
- It only support Euclidian distance as a measure of dissimilarity
- Each data sample is assigned to the cluster with the closest centroid (in Euclidian distance)
- The k centroids are mean values for the data points in the clusters, hence the name "k-means"

The Algorithm

a. ***Initialisation:*** Randomly select k samples from the data. These are the initial centroid guesses.
b. ***Loop*** until convergence, or for a fixed number of cycles:
   i. ***Reassignment***: assign each sample to the cluster with the closest centroid
   ii. ***Recentering***: for each cluster, calculate a new centroid as the average of all points in the cluster



Science and Technology Facilities Council
Hartree Centre

Within Cluster Variation

- In what sense does k-means provide a “good” solution?
- We understand this in terms of ***within cluster variation*** (WCV) for a given cluster C_k:

$$WCV(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

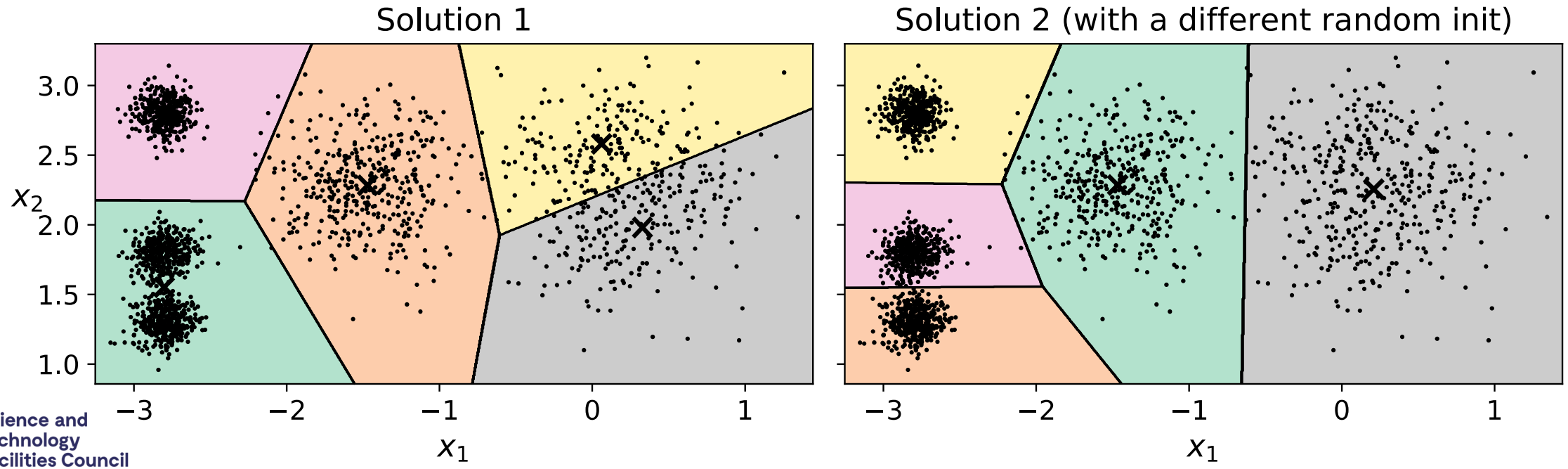
- Each iteration of the k-means reduces the WCV.
- In general, we might seek the global minimum of total WCV (also called inertia):

$$\min_{C_1 \ldots C_k} \left\{ \sum_{k=1}^{K} WCV(C_k) \right\}$$

- However, k-means typically only finds local minima.

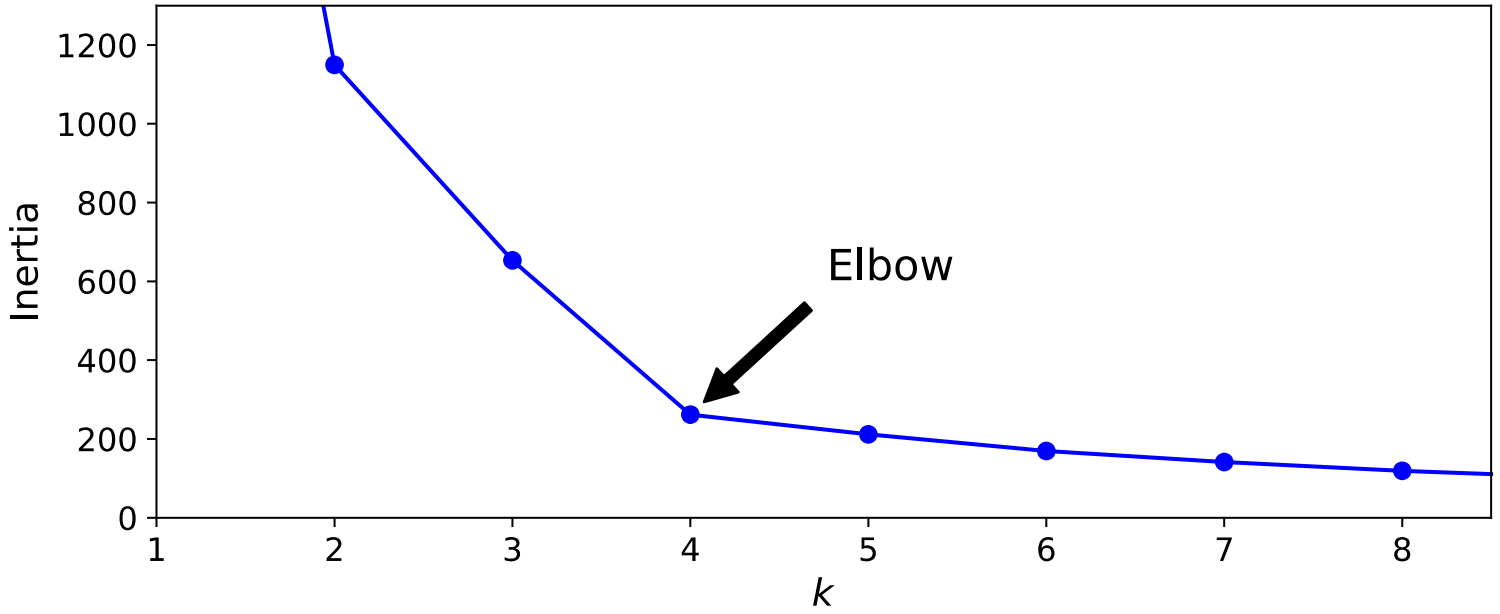
K-Means and Local Minima of Inertia

- The k means algorithm finds local minima of total within cluster variation (AKA inertia)
- For this reason, one typically runs the algorithm several times with different initial conditions (i.e. different choices for the initial centroids)



Choosing the Number of Clusters

- How should we choose k?
- One way is to use a plot of k versus total WCV (AKA inertia): $\sum_{k=1}^{K} WCV(C_k)$
- Inertia will always be smaller for larger k (exercise: show this) so we can't choose the value of k which minimizes it
- However, we can look for an ***elbow*** in the plot
  - This is a common technique for fitting models in machine learning
  - We assume that reductions in inertia to the right of the elbow are the result of overfitting
  - In the plot below, we would choose k = 4



Silhouette method

- Measure of how similar an object is to its own cluster compared to other clusters
- Range of silhouette score is {-1,1}

- High value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters

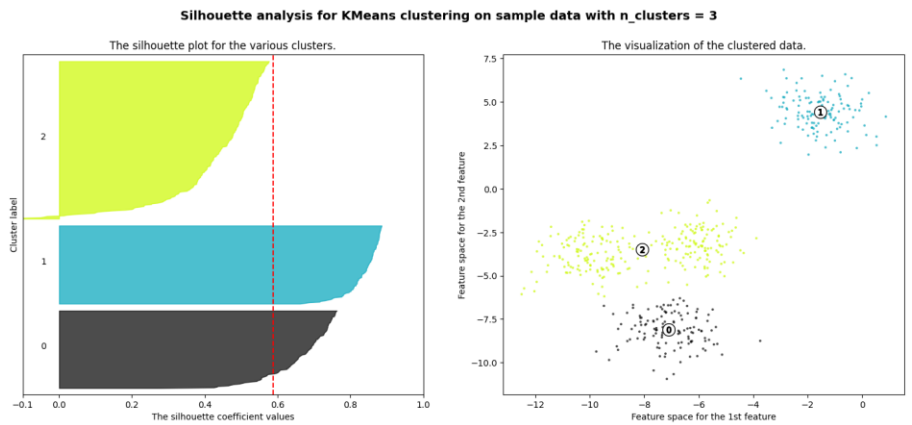
Silhouette methods things to consider

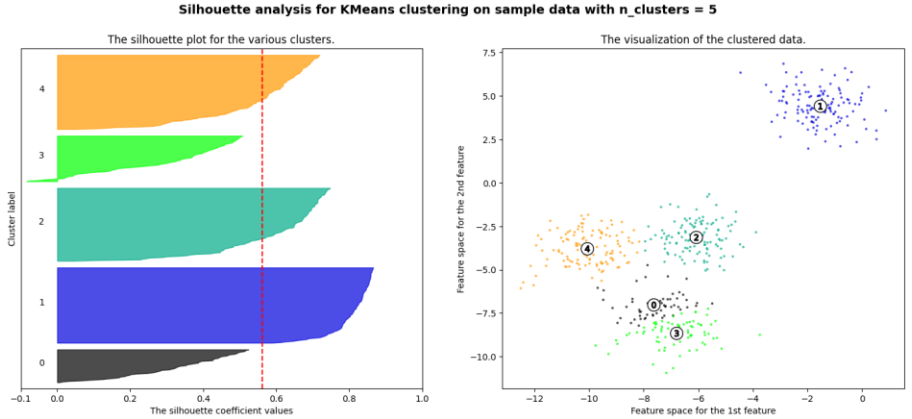
- In high dimensional data distances become similar and therefore it becomes difficult to achieve high values of silhouette score.
- Good results for convex shaped cluster
- Might not perform well with irregular shaped or varying sized clusters
- Can be computed for various distances such as Euclidean and Manhattan

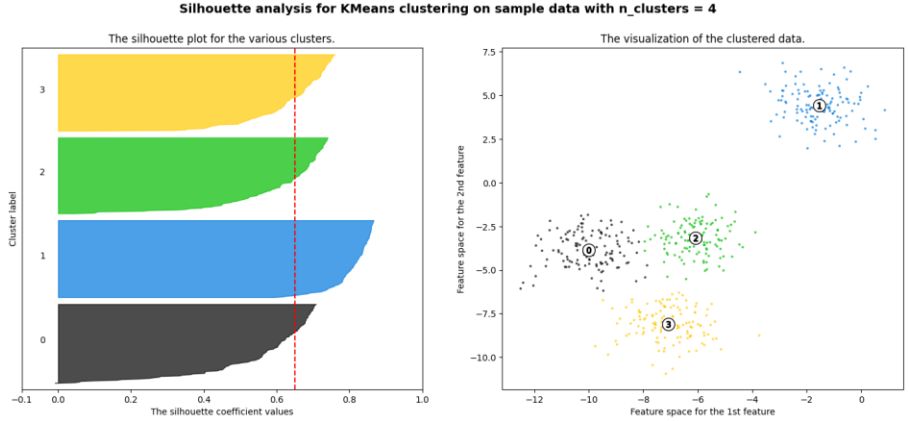
Silhouette coefficient derivation

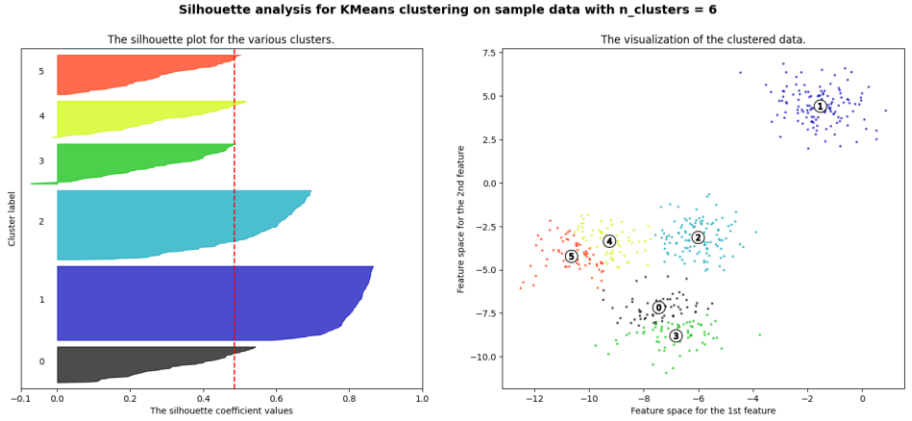
- In cluster distance: $a_i = \frac{1}{|C_A|-1} \sum_{j \in C_A, i \neq j} d(i,j)$
- Between cluster distance: $b_i = \min_{B \neq A} \frac{1}{|C_b|} \sum_{j \in C_B} d(i,j)$
- Silhouette score: $s_i = \begin{cases} \frac{b_i - a_i}{\max\{a_i, b_i\}} & if\ |C_A| > 1 \\ 0 \quad . & if\ |C_A| = 1 \end{cases}, -1 \leq |s_i| \leq 1$
- *Silhouette coefficient:* $SC = \max_k \frac{1}{|C_k|} \sum_{i \in k} s_i$

Example of Silhouette method









Science and Technology Facilities Council

Complexity of K-Means

- What is the time complexity of K-means?
- Most of the time is spent on computing distances between points.
  - One such operation costs $\Theta(d)$ where d is the dimensionality of (i.e. number of variables in) one sample.
- The reassignment step (assigning points to clusters) computes distances between each of the N points and k centroids, so its overall time complexity is $\Theta(Nkd)$
- If we run the loop for a fixed number of iterations I, the overall complexity is therefore $\Theta(NkdI)$.
- The main strength of K-Means: it's fast! K-means is linear in *N*.
- This means that K-means is more efficient than HAC
- We had to fix the number of iterations I.
  - In most cases, K-means quickly converges of approximately converges
- Space complexity for k-means is $\Theta\big((N+k)d\big)$ (exercise: show this)
- For large datasets, one might consider using "minibatch k-means". It's outside the scope of the course, but is implemented in sklearn (also discussed in the "Hands-On Machine Learning" book).

Limitations of K-Means: Dissimilarity

- k-means only supports one dissimilarity measure: Euclidean distance
- Hence, clustering problems that require none-Euclidian dissimilarity measures require a different clustering algorithm
- The Euclidian distance is defined for all real vectors, but that doesn't mean that it's a sensible measure of dissimilarity for all real vectors. E.g.:
  - Timeseries: correlation is a far better dissimilarity measure
  - Word documents: cosine distance between TF-IDF vectors is a better measure of dissimilarity (discussed in week 9)

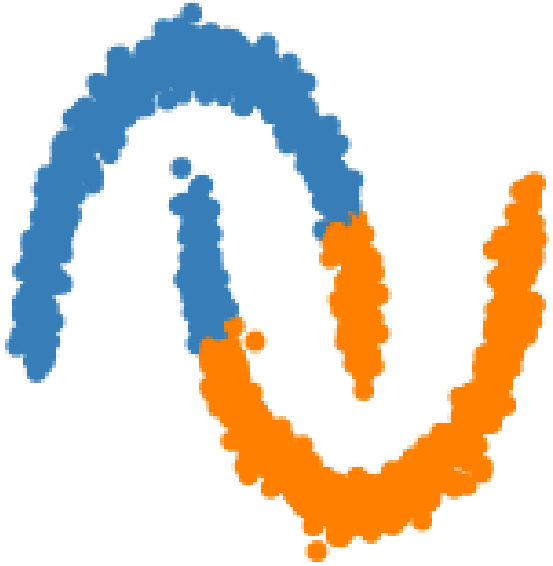
Limitations of K-Means: Cluster Shapes

- Struggles with spread out clusters, interlocking clusters, nested clusters etc.
- Can only create linear cluster boundaries
- Does best on globular clusters
- It is not very robust towards outliers, as it puts squared weight on them

Spread out clusters

Nested clusters

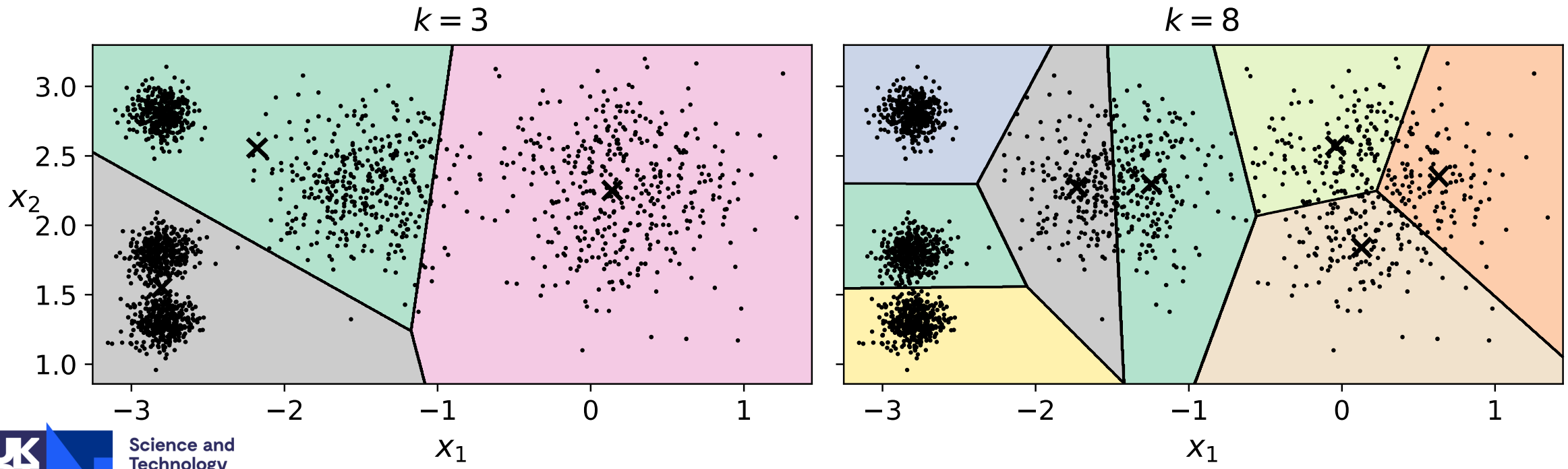
Interlocking clusters



Misspecified k

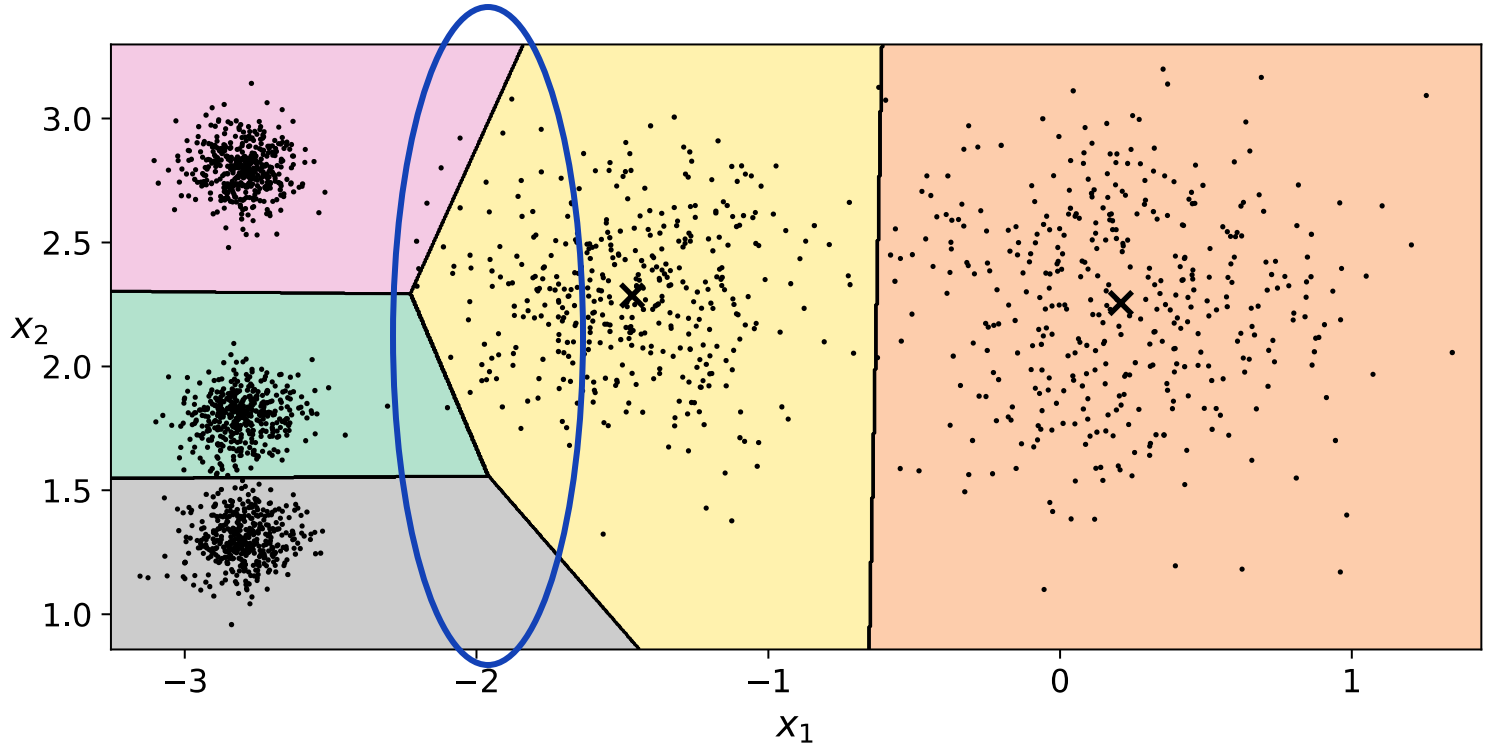
Limitations of K-Means: Misspecified K Value

- **Sensitive to the user-defined number of clusters**



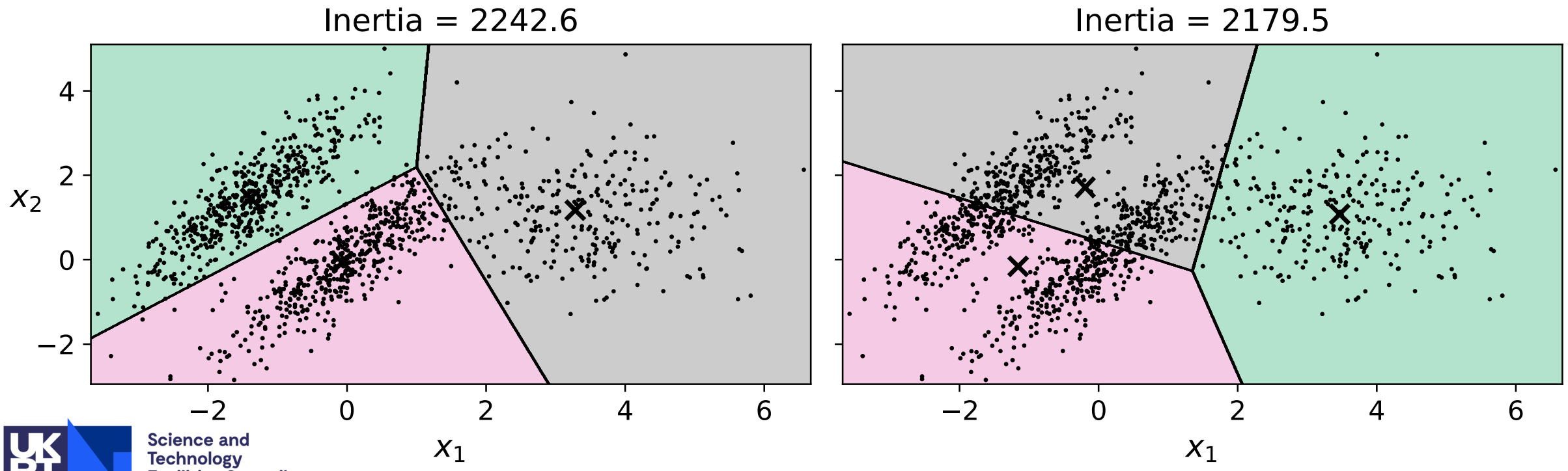
Limitations of K-Means: Cluster Boundaries

- Cluster boundaries are equidistant between centroids, even when one cluster is more disperse (suggesting that its members will generally stray further from its centroid).



Limitations of K-Means: Cluster Shape

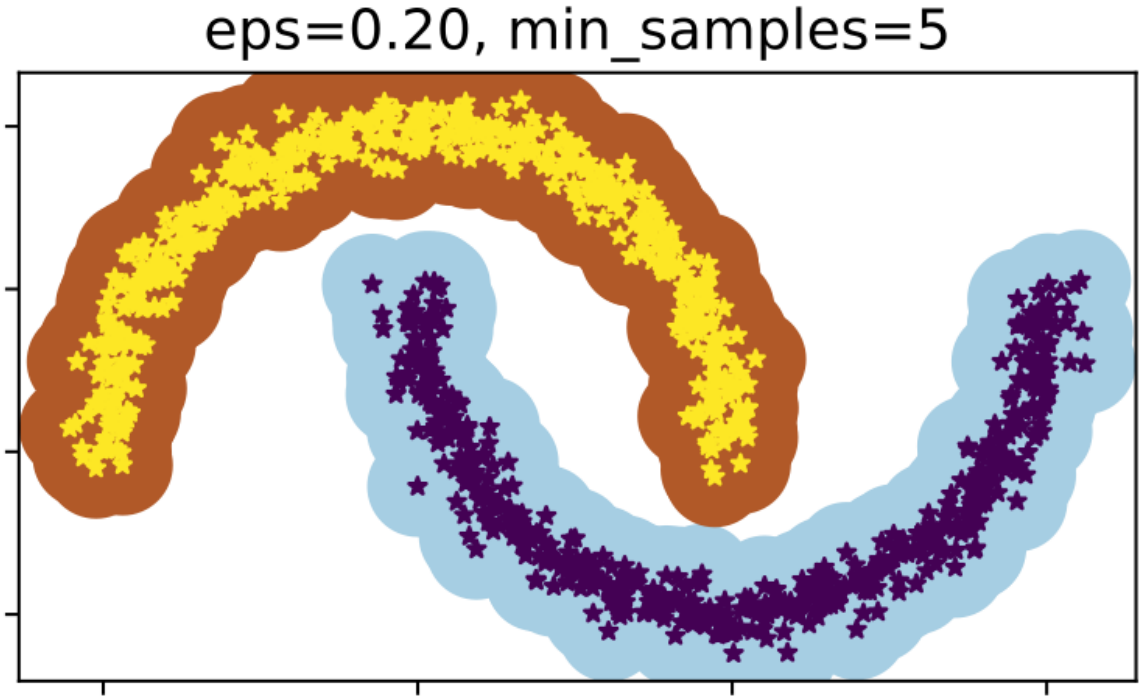
- **Struggles with spread out (non-globular) clusters**



DBSCAN

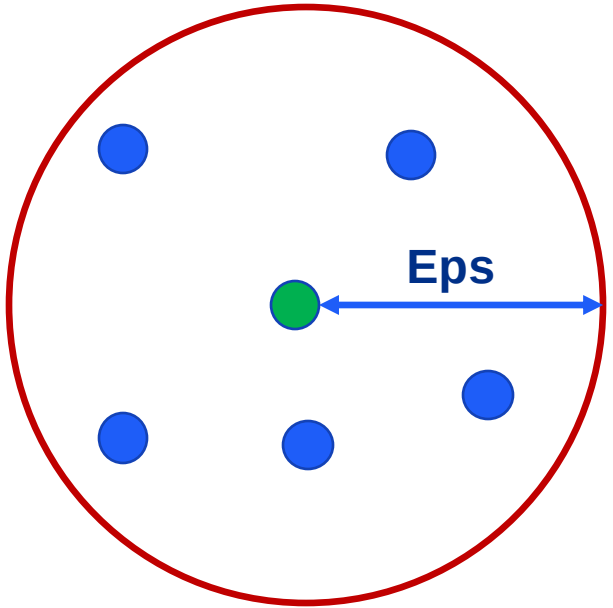
DBSCAN: Density-Based Clustering

- DBSCAN is yet another approach to clustering
- It identifies regions in the sample space which are densely populated with samples, and tries to join these dense regions together to form larger clusters
  - Hence, it can discover clusters of arbitrary shape
    - E.g. it performs well on the half moon dataset below, with which both HAC and k-means struggled
- DBSCAN is an acronym for “Density-based spatial clustering of applications with noise”
  - But it also has another meaning: the algorithm SCANs a database (DB) of samples in order to discover the dense regions, and then joins these dense regions to form clusters



DBSCAN Parameters

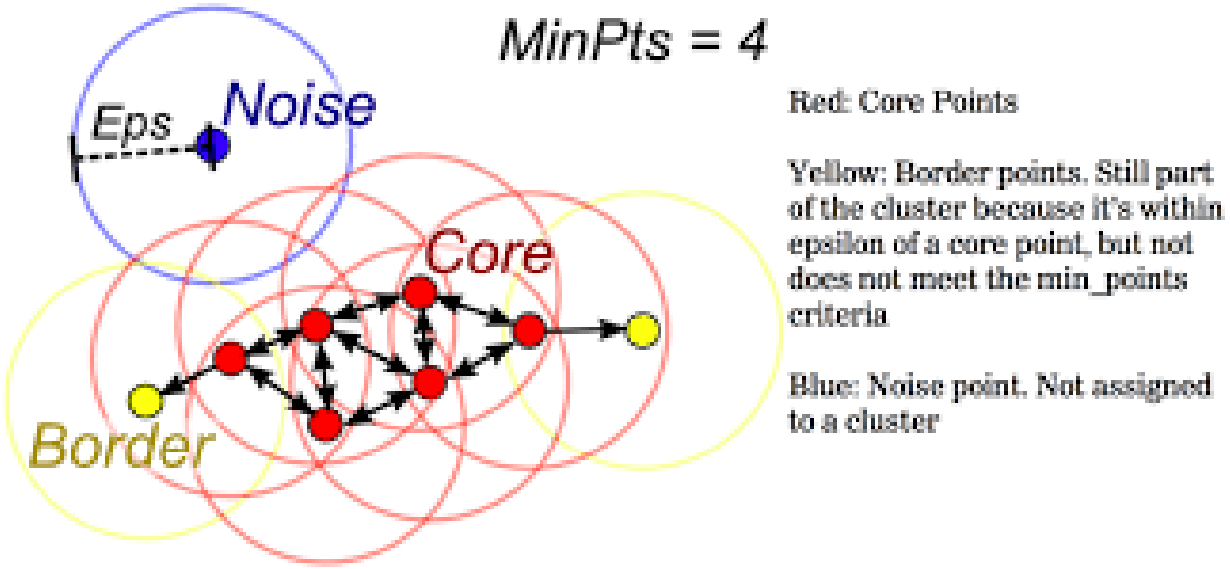
- DBSCAN has two parameters:
  - Eps: the distance (more generally, the dissimilarity) around a given point p that we look for neighboring points
    - We call this neighborhood an ε-neighborhood $N_\varepsilon(p)$
  - MinPts: the minimum number of points that are required within an ε-neighborhood $N_\varepsilon(p)$ for p to be considered a ***core point***, which represents a dense region of the sample space
- Points that are in the ε-neighborhood of a core point (but are not themselves a core point) are called ***border points***
- For example, in the plot below, the green point would be a core point, so long as $MinPts \leq 5$



DBSCAN Algorithm

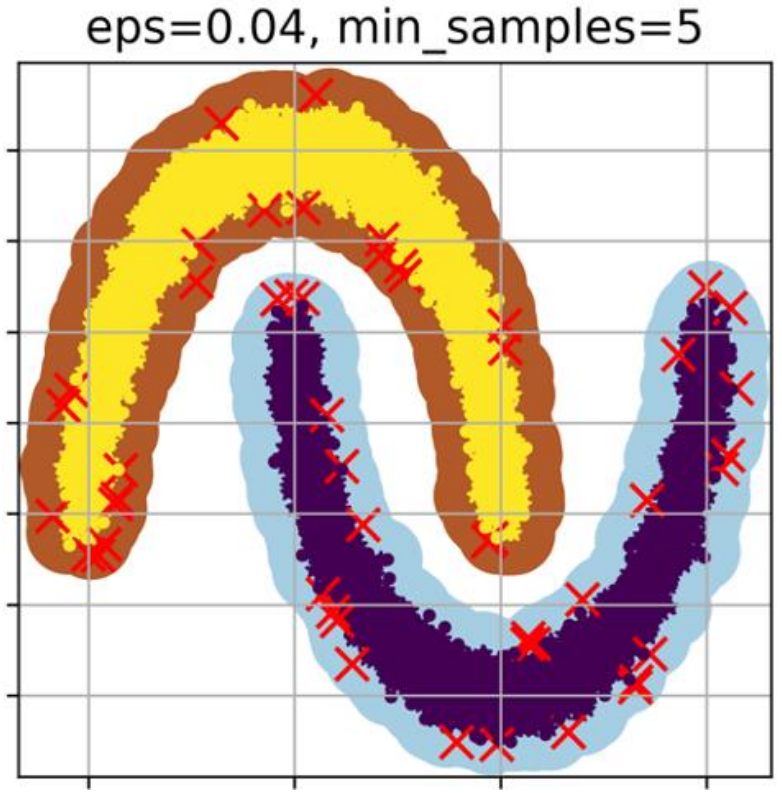
The DBSCAN algorithm:

1. **Arbitrarily select an unvisited point p**
2. **Retrieve all points in the epsilon-neighborhood of p (call this function *regionQuery*)**
3. **If p is a core point, then for each unvisited point q in the epsilon-neighborhood of p:**
   a. **Add q to the same cluster as p**
   b. **Loop to step (2), calling regionQuery on the new point q**
4. **If there are any unvisited points remaining in the dataset, loop to step (1)**
5. **Any clustered, none core points are labelled as border points**
6. **Any unclustered points are labelled as anomalies (or “noise”)**



DBSCAN Anomaly Detection

- Any points that remain unclustered after application of the DBSCAN algorithm are considered to be ***anomalies***
- These points are not in the epsilon neighborhood of a core point, although them may be in the epsilon neighborhood of a boundary point (as shown in the plot)

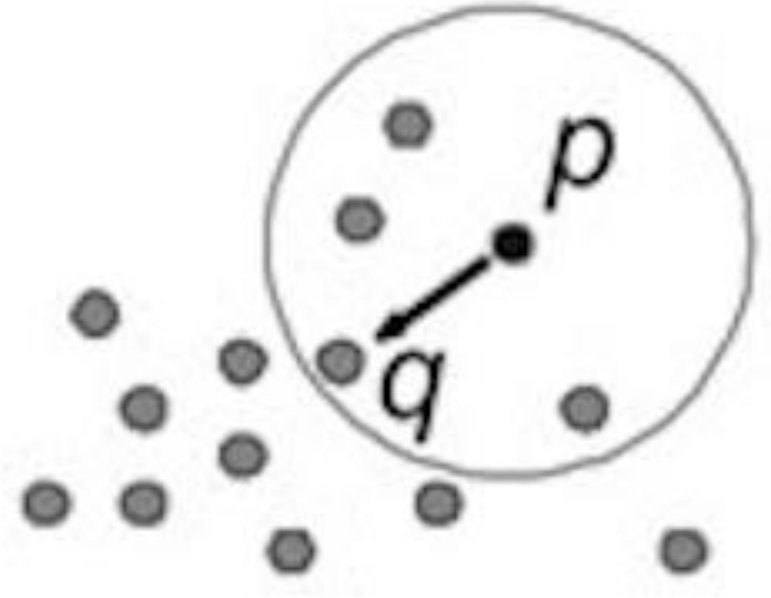


DBSCAN Algorithm Complexity

- For a naive implementation of regionQuery, distances between points are exhaustively computed and stored in a matrix, which is then traversed for each call of regionQuery.
  - Time complexity is $\Theta(N^2)$
    - Time complexity for computing the NxN distance matrix is $\Theta(N^2)$
    - A single call to regionQuery is $\Theta(N)$, because it retrieves and compares the (N-1) distances between the given point p and all other points
    - regionQuery is called N times, giving an overall time complexity for all regionQuery calls of $\Theta(N^2)$
  - Space complexity is $\Theta(N^2)$, because the NxN matrix of distances must be stored in memory.
- More advanced implementations of DBSCAN have time complexity of $O(N\,log(N))$ and space complexity of $\Theta(N)$
  - This is achieved by using a spatial index for regionQuery (e.g. R*-trees, which are outside the scope of this lecture).
    - This gives a time complexity of $O(N\,log(N))$ for a single call to regionQuery
    - R*-trees have space complexity of $\Theta(N)$

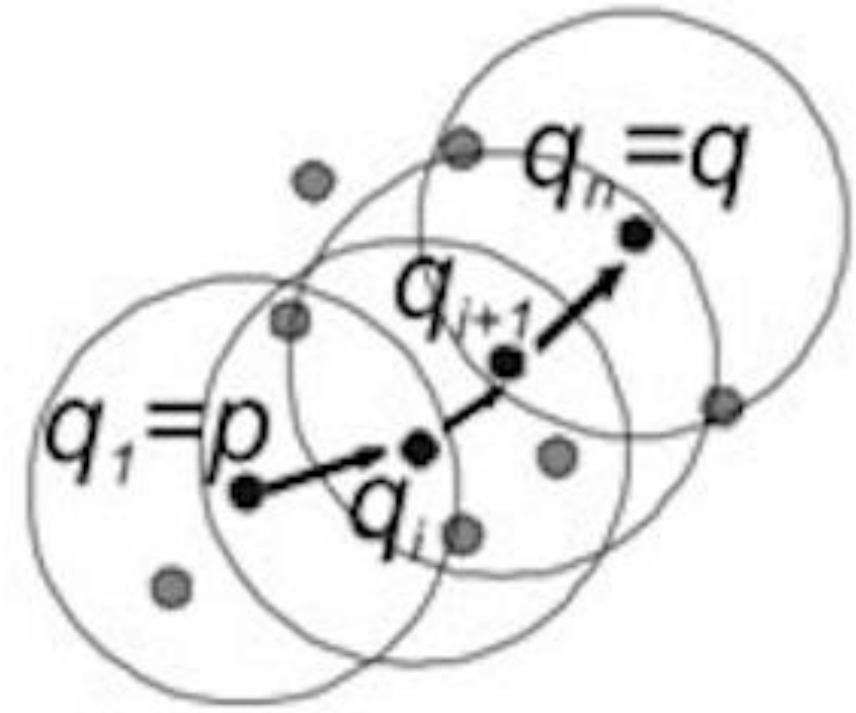
Directly Density-Reachable Points

- In what sense is DBSCAN optimal? The short answer is:
  - DBSCAN efficiently discovers clusters which are maximal sets of density connected points
- To unpack that answer, we'll need to introduce a bit of DBSCAN jargon…
- A point q is said to be ***directly density-reachable*** from point p w.r.t. Eps, MinPts if:
  - q is in the epsilon neighborhood of p
  - p is a core point



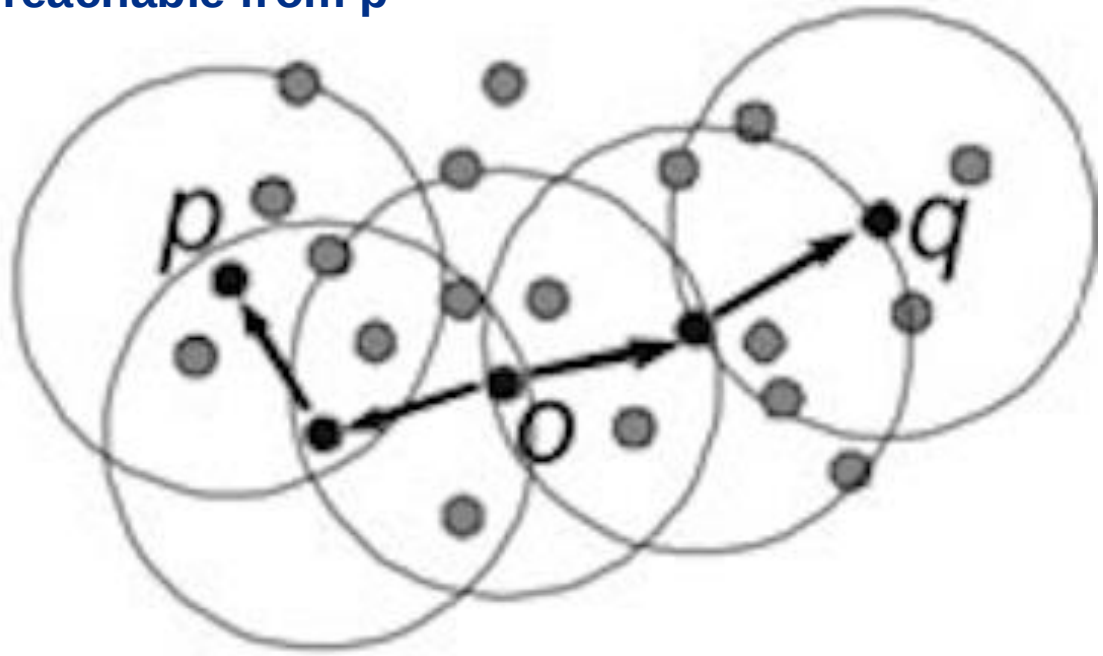
Density-Reachable Points

- A point q is said to be ***density-reachable*** from point p w.r.t. Eps, MinPts if there is a chain of points $p_1, p_2, \ldots, p_n$ with $p_1 = p$, $p_n = q$, such that p_{i+1} is directly density reachable from p_i



Density-Connected Points

- A point p is said to be ***density-connected*** to a point q w.r.t. Eps, MinPts if there is a point o such that both p and q are density reachable from o
- As mentioned earlier, DBSCAN efficiently discovers clusters which are maximal sets of density connected points
  - So, DBSCAN finds ***connected directed graphs*** in the dataset, where:
    - Leaf nodes are boundary points
    - Non-leaf nodes are core points
    - (p, q) is a directed edge if q is directly density reachable from p
  - These connected graphs are the clusters



DBSCAN Parameter Selection: MinPts

- DBSCAN has two model parameters: MinPts and Eps.
- It's difficult to determine optimal values for these parameters, as there's no objective measure of optimality for the different clusterings that can be formed for a given dataset
- In practice, the parameters are often chosen using "rules of thumb"
- Choosing MinPts:
  - For 2-dimensional data, use DBSCAN's default value of MinPts = 5 (Ester et al., 1996)
  - If your data has more than 2 dimensions, choose MinPts = 1 + 2*(number of dimensions) (Sander et al., 1998).

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise." In kdd, vol. 96, no. 34, pp. 226-231. 1996.

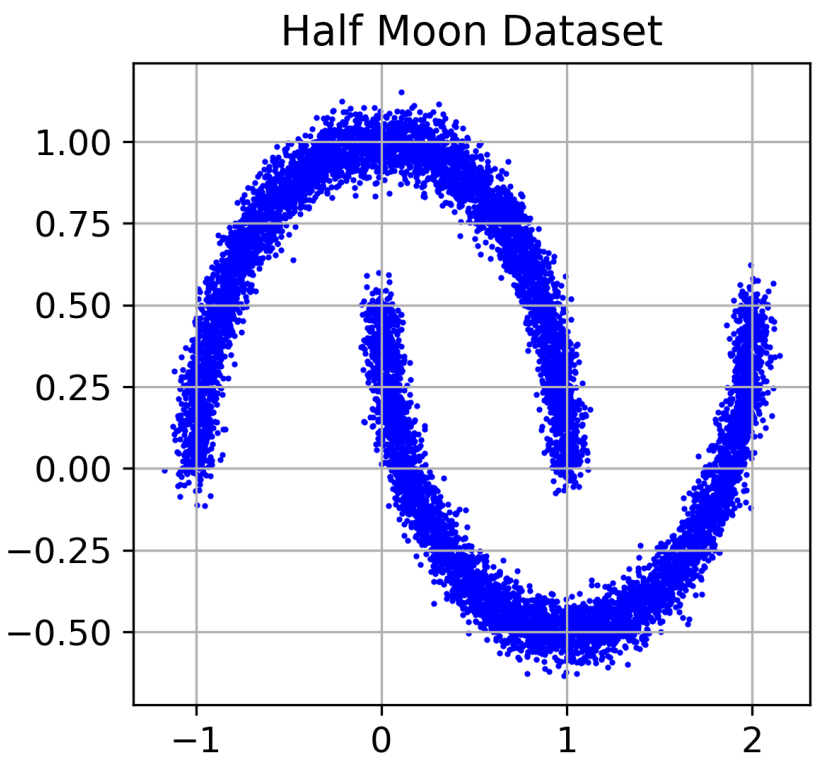
Sander, J., Ester, M., Kriegel, HP. et al. "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications." Data Mining and Knowledge Discovery 2, 169–194 (1998). https://doi.org/10.1023/A:1009745219419

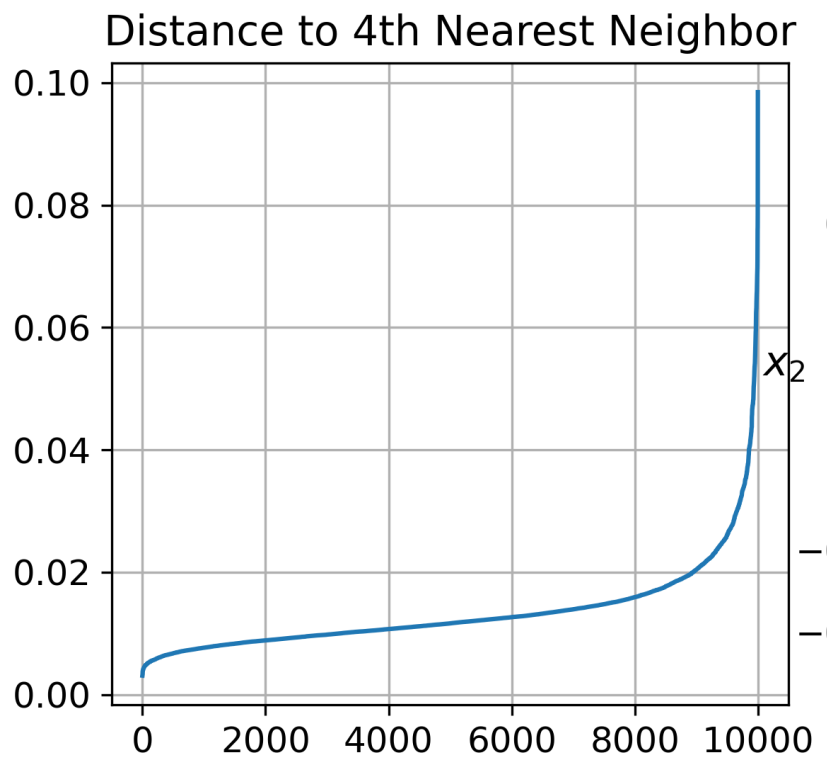
DBSCAN Parameter Selection: Eps

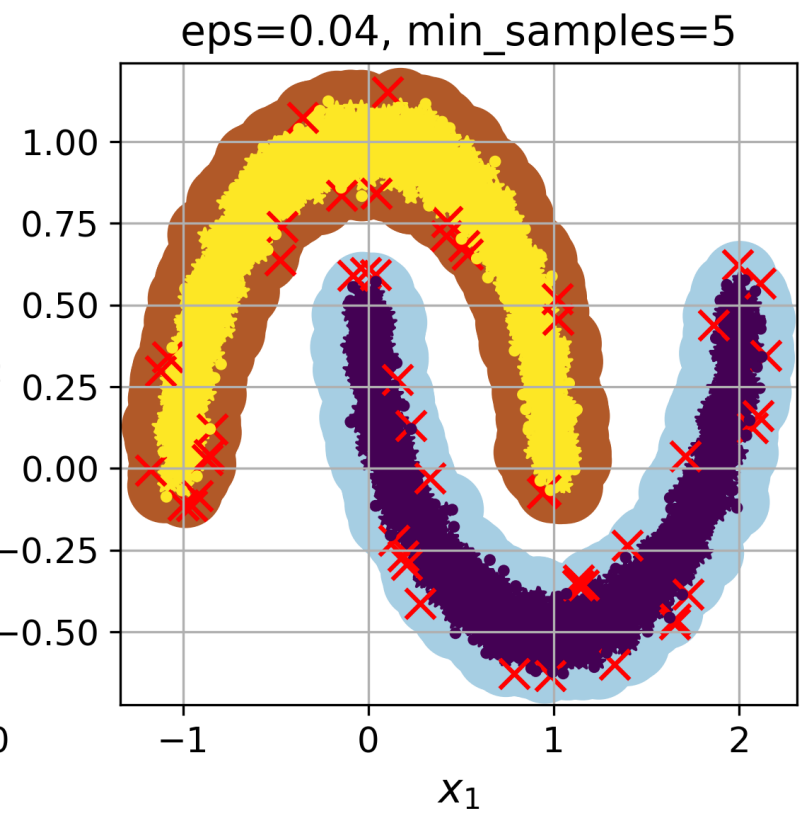
- Eps can be selected with a rule of thumb from Rahmah et al. 2016 (see also code on the next slide):
  - Calculate the dissimilarity between each point and its k^{th} nearest neighbor, where k = MinPts - 1 (which was selected in the previous slide).
    - The "-1" is necessary because MinPts includes the point in question, whereas the set of a point's k nearest neighbors doesn't include the point.
  - Plot these dissimilarities in ascending order (see the plot on the next slide – in this case, dissimilarity is Euclidian distance).
  - Find the elbow in this plot, and read off the corresponding dissimilarity value (in the plot on the next slide,, it's somewhere in the range 0.02 - 0.04)
  - Alternatively, Eps can be set using domain-specific knowledge for the data (i.e. you might know that dissimilarities smaller than a certain value indicate that samples are in the same cluster).
- The red crosses are anomalous (i.e. unclustered) points.

Rahmah, Nadia, and Imas Sukaesih Sitanggang. "Determination of optimal epsilon (eps) value on DBSCAN algorithm to clustering data on peatland hotspots in sumatra." In IOP conference series: earth and environmental science, vol. 31, no. 1, p. 012012. IOP Publishing, 2016.

DBSCAN Parameter Selection: Eps







Science and Technology Facilities Council

Hartree Centre

Eps Estimation Code

```
from sklearn.neighbors import NearestNeighbors

from sklearn.datasets import make_moons
X, y = make_moons(n_samples=100000, noise=0.05, random_state=42)

neighbors = NearestNeighbors(n_neighbors=5).fit(X)
distances, ix = neighbors.kneighbors(X)
distances = np.sort(distances, axis=0)
print(distances[:,4])
distances = distances[:,4]
dbscan2 = DBSCAN(eps=0.2)
dbscan2.fit(X)

plt.figure(figsize=(9, 4))

plt.subplot(131)
plt.scatter(X[:, 0], X[:, 1], marker='o', s=1, c='b')

plt.grid(True)
plt.title("Half Moon Dataset", fontsize=14)
```

```
plt.subplot(132)

plt.title("Distance to 4th Nearest Neighbor", fontsize=14)
plt.plot(distances)
plt.annotate("Elbow", xy=(95000, 0.012), xytext=(70000, 0.02),
             arrowprops=dict(arrowstyle="->"), fontsize=16)
plt.grid(True)

plt.subplot(133)
plot_dbscan(dbscan2, X, size=600, show_ylabels=False)

save_fig("DBSCAN_distances")
```

HDBSCAN

- HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that builds a hierarchy of clusters based on density.
- Unlike DBSCAN, which identifies clusters in a single pass, HDBSCAN constructs a hierarchical tree structure, allowing for flexibility in choosing the optimal number of clusters.

How HDBSCAN Works

- **Density Estimation:**
  - **Core Distance:** For each data point, the core distance is calculated, which is the minimum radius required to form a cluster around that point.
  - **Reachability Distance:** The reachability distance between two points is the maximum of their core distances.
- **Connectivity Graph:**
  - A connectivity graph is constructed where edges connect points that are within a certain reachability distance of each other.
- **Minimal Spanning Tree:**
  - A minimal spanning tree (MST) is extracted from the connectivity graph. This tree represents the hierarchical structure of the data.
- **Cluster Formation:**
  - The MST is traversed from the root to the leaves. At each level, clusters are formed based on the density threshold. Points with a high density are grouped together, while points with low density are considered noise.
- **Hierarchical Clustering:**
  - The process of cluster formation is repeated at different density thresholds, creating a hierarchy of clusters.
- **Optimal Cluster Selection:**
  - The optimal number of clusters is determined based on a metric such as the silhouette coefficient or the validity index. This metric helps to identify the level of the hierarchy that best represents the underlying structure of the data.

Benefits of HDBSCAN

- **Flexibility:** HDBSCAN allows for the exploration of different clustering solutions at various levels of the hierarchy.
- **Noise Handling:** It effectively handles noise and outliers in the data.
- **Scalability:** HDBSCAN can handle large datasets efficiently.
- **Robustness:** It is less sensitive to parameter tuning compared to some other clustering algorithms.

Summary

Summary

- Today we looked at the following clustering algorithms:
  - Hierarchical (aka agglomerative) clustering forms a hierarchy of clusters
  - K-Means clustering picks k "mean" points and forms clusters around these points to minimize the variation within clusters
  - DBSCAN grows clusters incrementally
- All these algorithms have their strengths and weaknesses, making them applicable in different situations

Science and Technology Facilities Council

Hartree Centre

Questions?

UKRI Science and Technology Facilities Council

Hartree Centre

Thank you