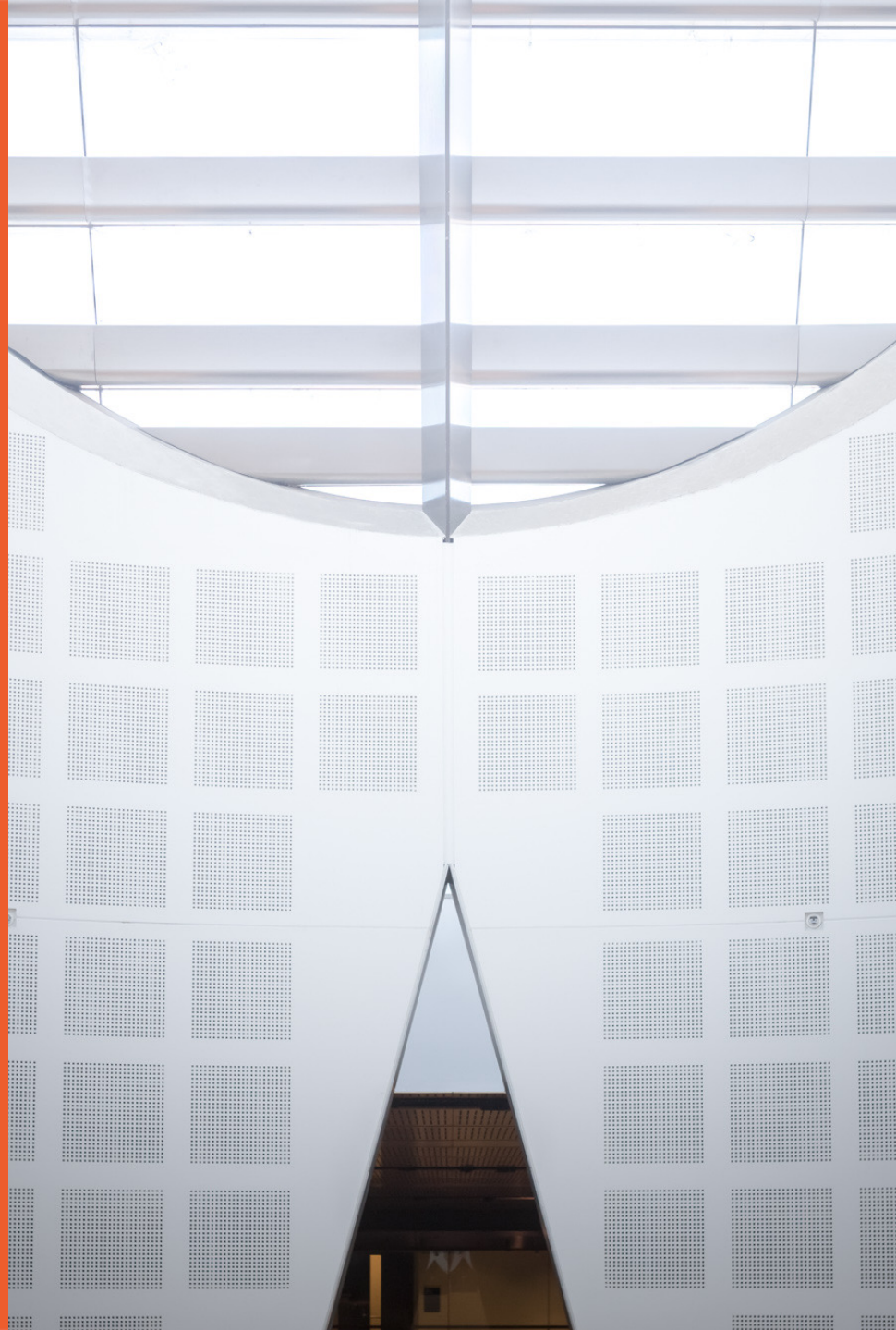
Genomic Data Analysis (Clustering as an example)

Dr Chang Xu

School of Computer Science

Reference: Healthcare Data Analytics, Chapter 6



THE UNIVERSITY OF SYDNEY

Deep or broad immune profiling

To measure:

Pan-leukocyte
- CD45

Myeloid
- CD11b

Monocyte
- *CD14*
- *HLA-DR*

Granulocyte
- *CD16*
- *CD66b*
- *Siglec-8*
- *FceR1a*
- *CD117*

T cells
- CD3
- CD4
- CD8
- CD25
- CD69
- CD45RA
- CD45RO
- CD62L
- CCR7
- PD-1
- Tim-3
- Lag-3
- T-Bet
- GATA-3
- RORgT
- FoxP3

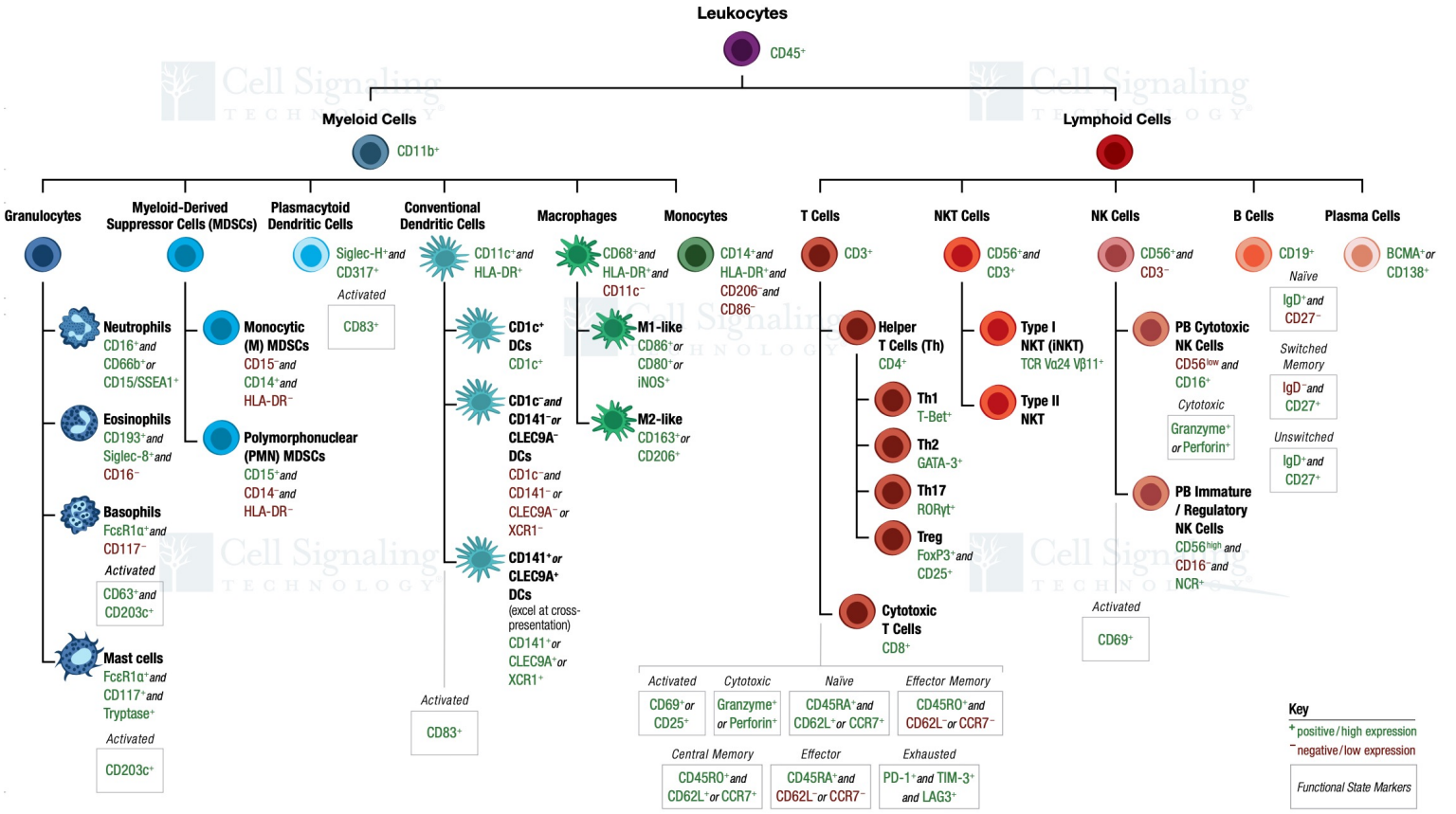
NK cells
- CD56
- TCRva24
- Granzyme
- Perforin

B cells
- CD19
- CD138
- IgD (CD27)

cDC
- **CD11c**
- **(HLA-DR)**

PDC
- **Siglec-H**
- **CD317**

Macrophage
- **CD68**
- **(HLA-DR)**



https://media.cellsignal.com/www/pdfs/science/pathways/Immune-Cell-Markers-Human.pdf

Identify Cell Types

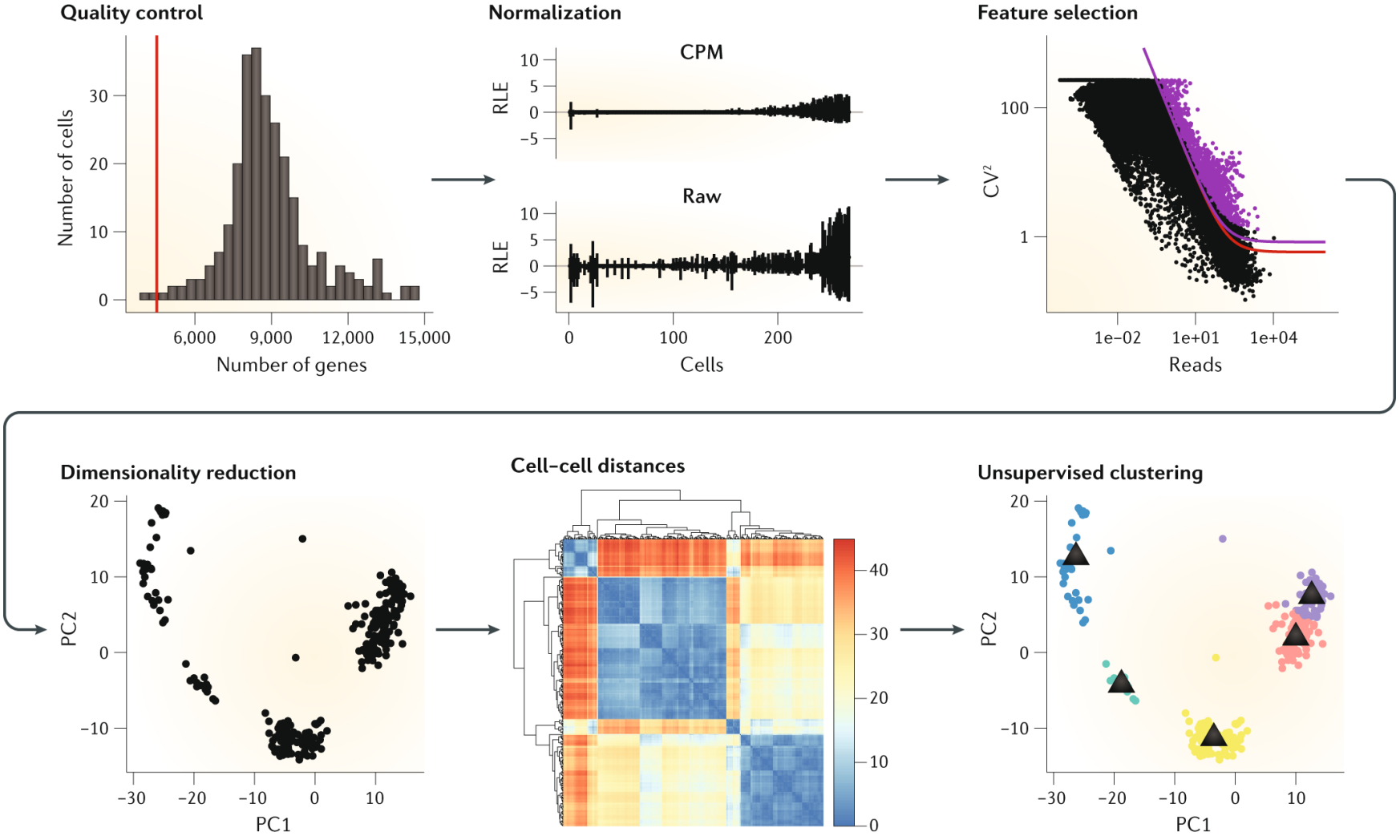
The cell can be considered the fundamental unit in biology. For centuries, biologists have known that multicellular organisms are characterized by a plethora of distinct cell types.

- Cells can be distinguished by their size and shape using a microscope, and attributes based on their physical appearance have traditionally been the primary determinant of cell type.

- Later, discoveries in molecular biology made it possible to characterize cell types on the basis of the presence or absence of surface proteins.

- Advances in microfluidics have made it possible to isolate a large number of cells, and along with improvements in RNA isolation and amplification methods, it is now possible to profile the transcriptome of individual cells using next-generation sequencing technologies.

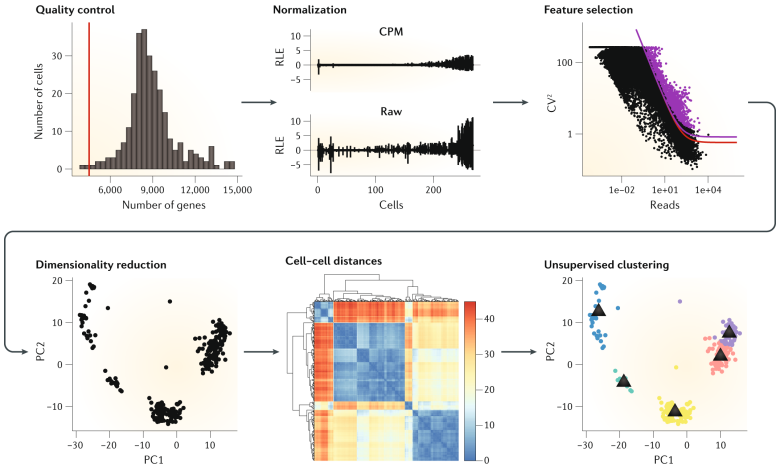
Single-cell RNA sequencing



Overview of the workflow for the computational analysis of single-cell RNA sequencing (scRNA-seq) data leading up to unsupervised clustering.

Single-cell RNA sequencing

There are several steps involved in the computational analysis of scRNA-seq data, including quality control, mapping, quantification, normalization, clustering, finding trajectories and identifying differentially expressed genes



First, unreliable cells (and possible doublets) are removed through quality control. The cleaned data set is then normalized to correct for differences in read coverage and other technical confounders.

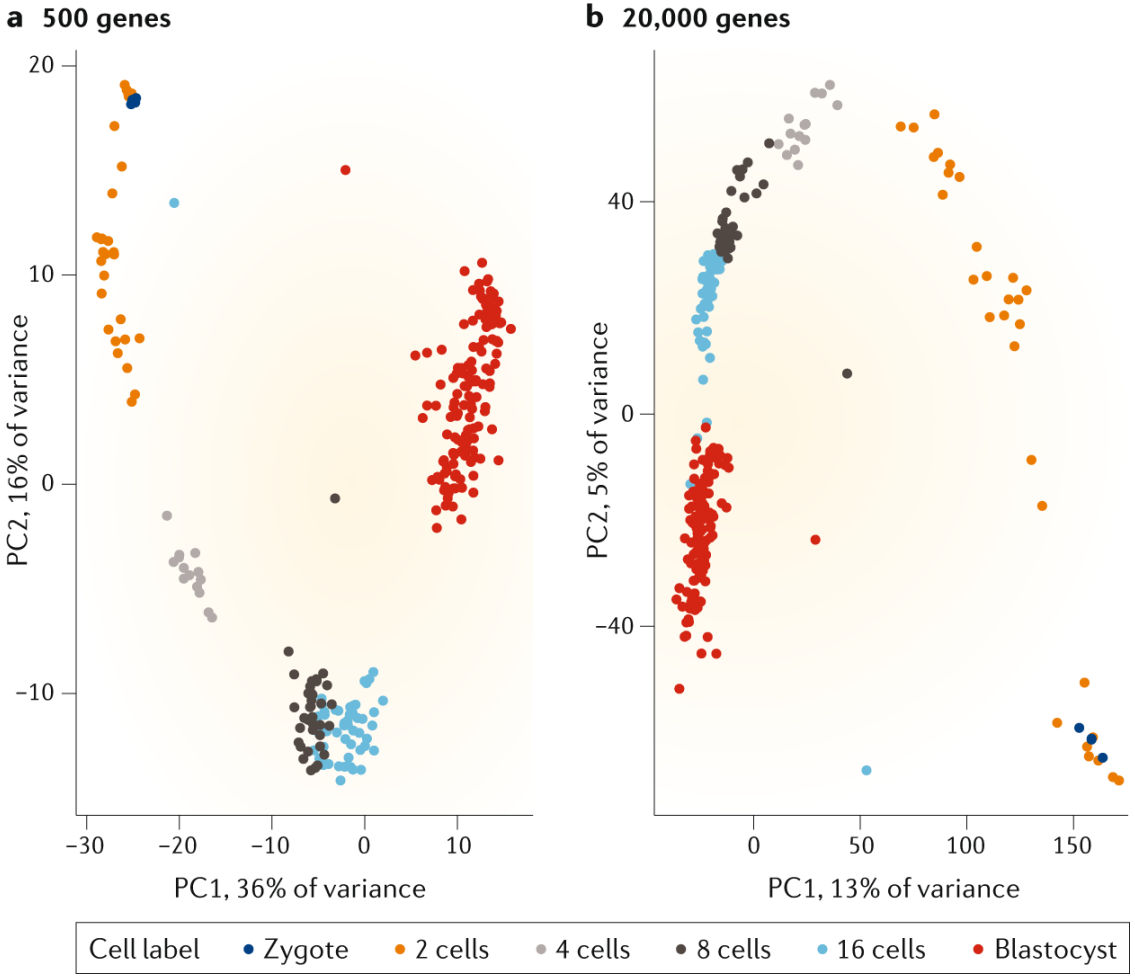
Feature selection and dimensionality reduction isolate the most informative genes and strongest signals from background noise, respectively.

Cell–cell distances are then calculated in the lower dimensional space and used to either construct a cell–cell distance graph or used directly by clustering algorithms to assign cells to clusters.

Some methods will compute the distances before the dimensionality reduction. CPM, counts per million; CV, coefficient of variation; PC, principal component; RLE, relative log expression.

Illustration of the curse of dimensionality

The application of feature selection and/or dimensionality reduction may reduce the noise and speed up calculations. Feature selection involves identifying the most informative genes, for example, the ones with the highest variance, whereas dimensionality reduction, for example, principal component analysis (PCA), projects data into a lower dimensional space.



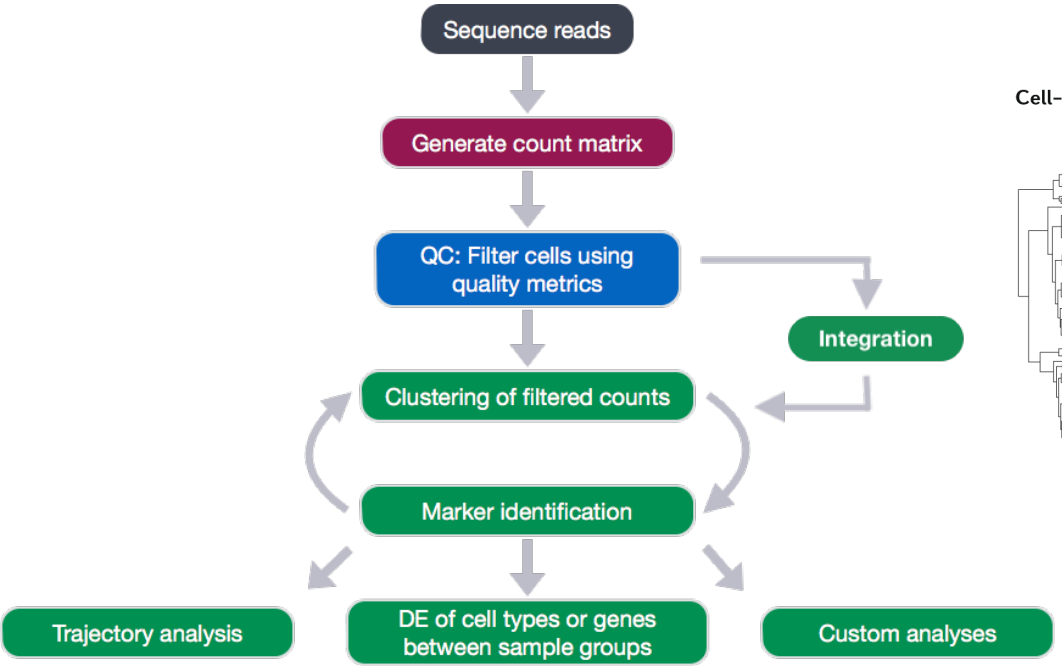
Six separate populations of cells should ideally be distinguishable. Principal component (PC) analysis (PCA) plots of the Deng data set using 500 (part **a**) and 20,000 (part **b**) of the most variable genes. When using a large number of features, clusters are less distinct, as indicated by the shorter distances between clusters (for example, the 4-cell stage is not as isolated). Consequently, unsupervised clustering becomes more challenging.

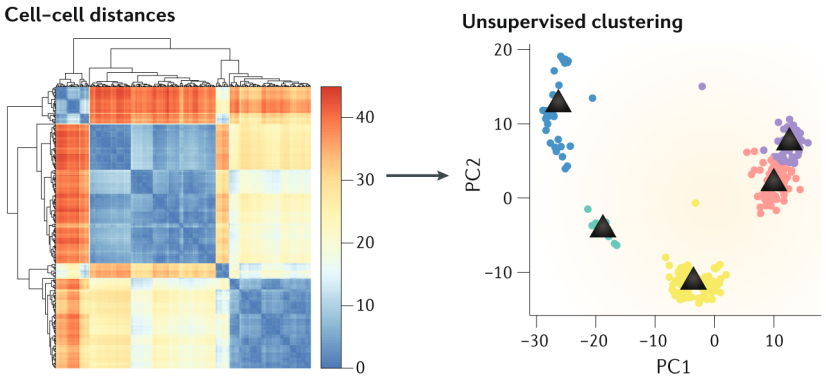
Clustering

The ability to define cell types through unsupervised clustering on the basis of transcriptome similarity has emerged as one of the most powerful applications of scRNA-seq.

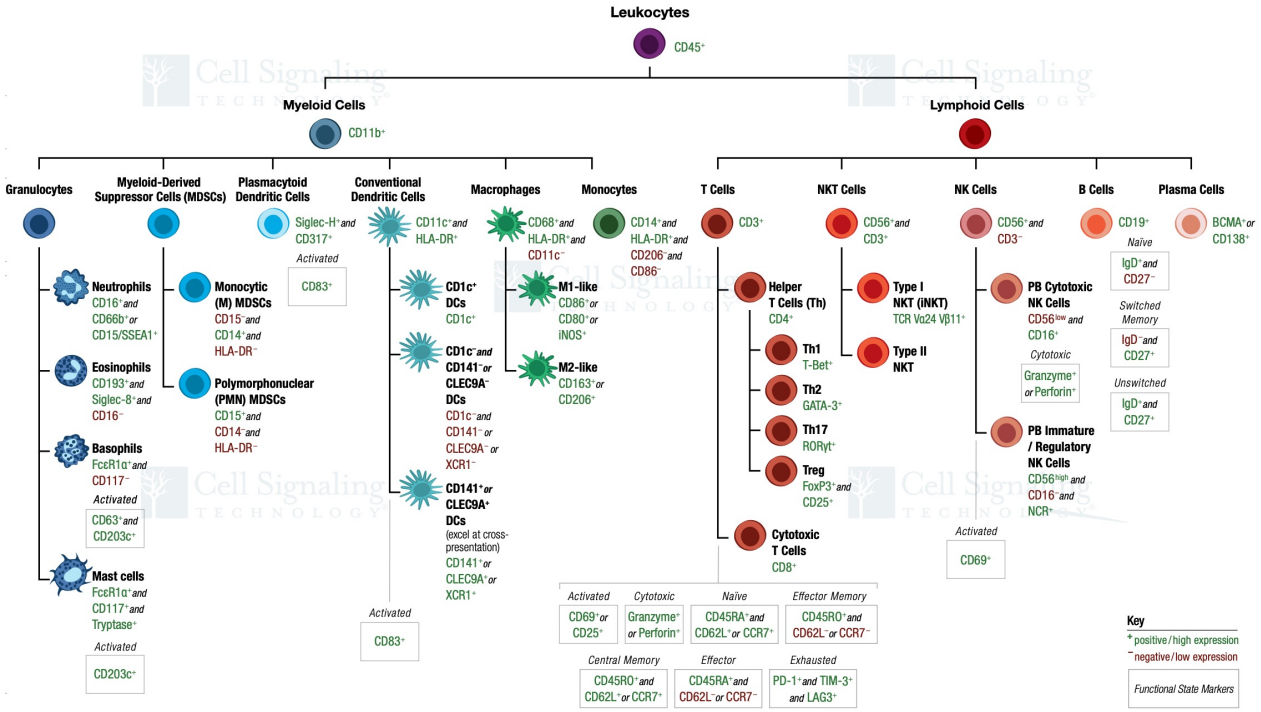
Single-cell RNA-seq clustering analysis

- To generate cell type-specific clusters and use known cell type marker genes to determine the identities of the clusters.

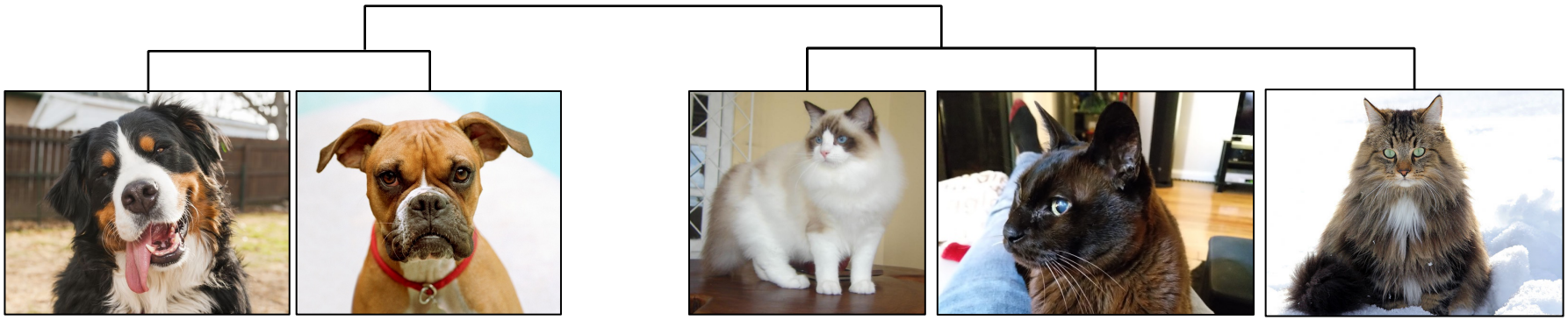




Cell types

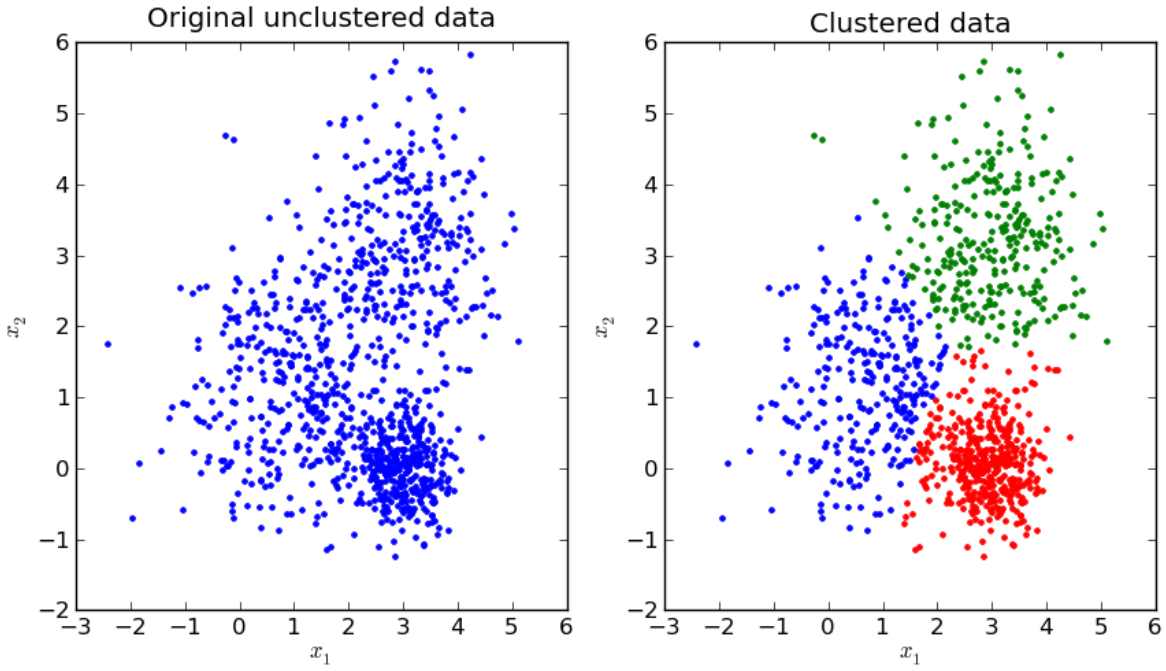


Pet types



Clustering

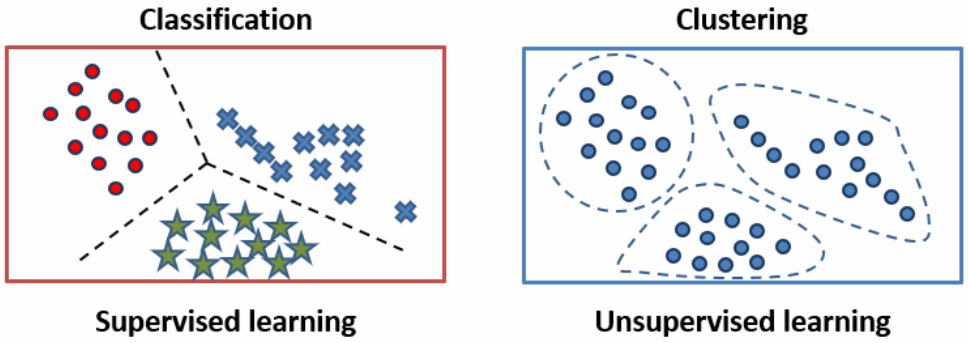
Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters)



Unsupervised Learning v.s. Supervised Learning

- Classification requires (human supervised) predefined label classes. What if we are in the early phases of a study and/or don't have the required resources to manually define, derive or generate these class labels?

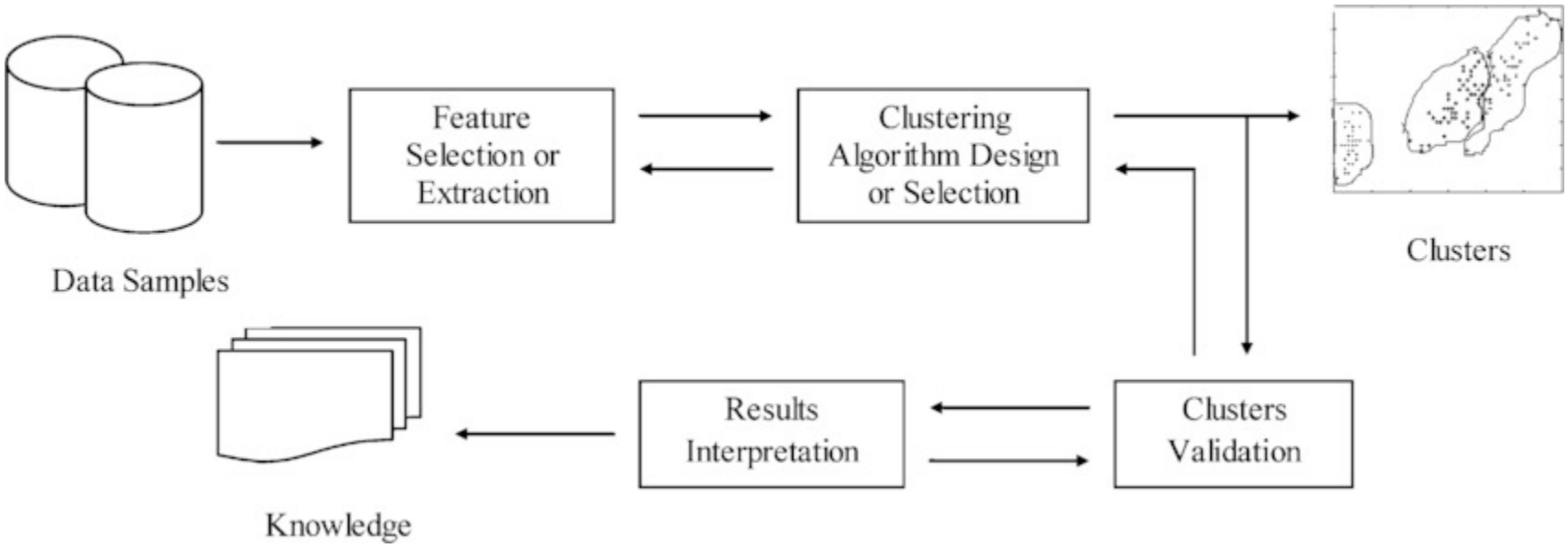
- Clustering can help us explore the dataset and separate cases into groups representing similar traits or characteristics. Each group could be a potential candidate for a class. Clustering is used for exploratory data analytics, rather than for confirmatory analytics or for predicting specific outcomes.



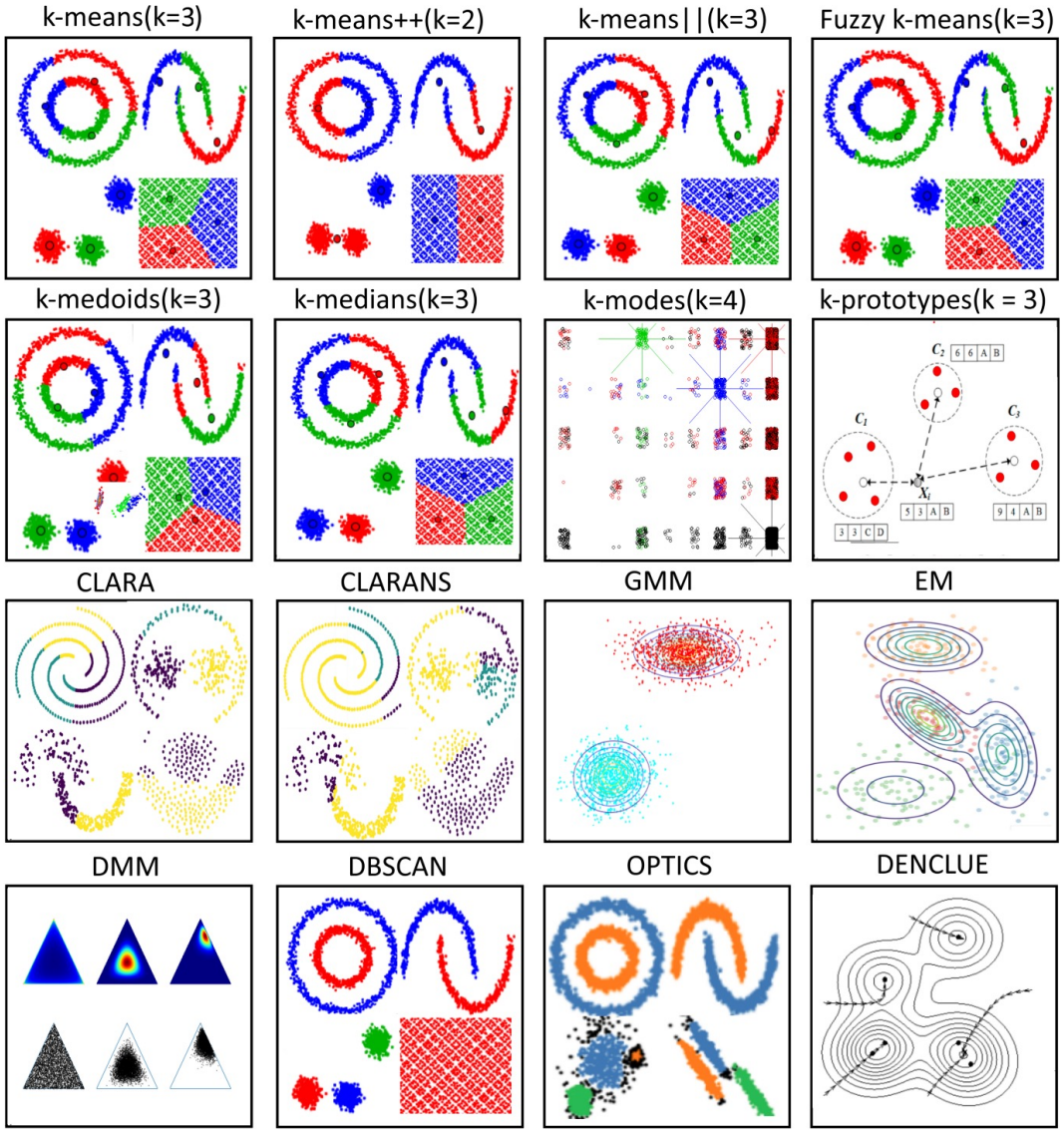
Process of Clustering

4 Main Steps:

1. Feature Selection or Extraction
2. Clustering Algorithm Design or Selection
3. Clusters Validation
4. Results Interpretation



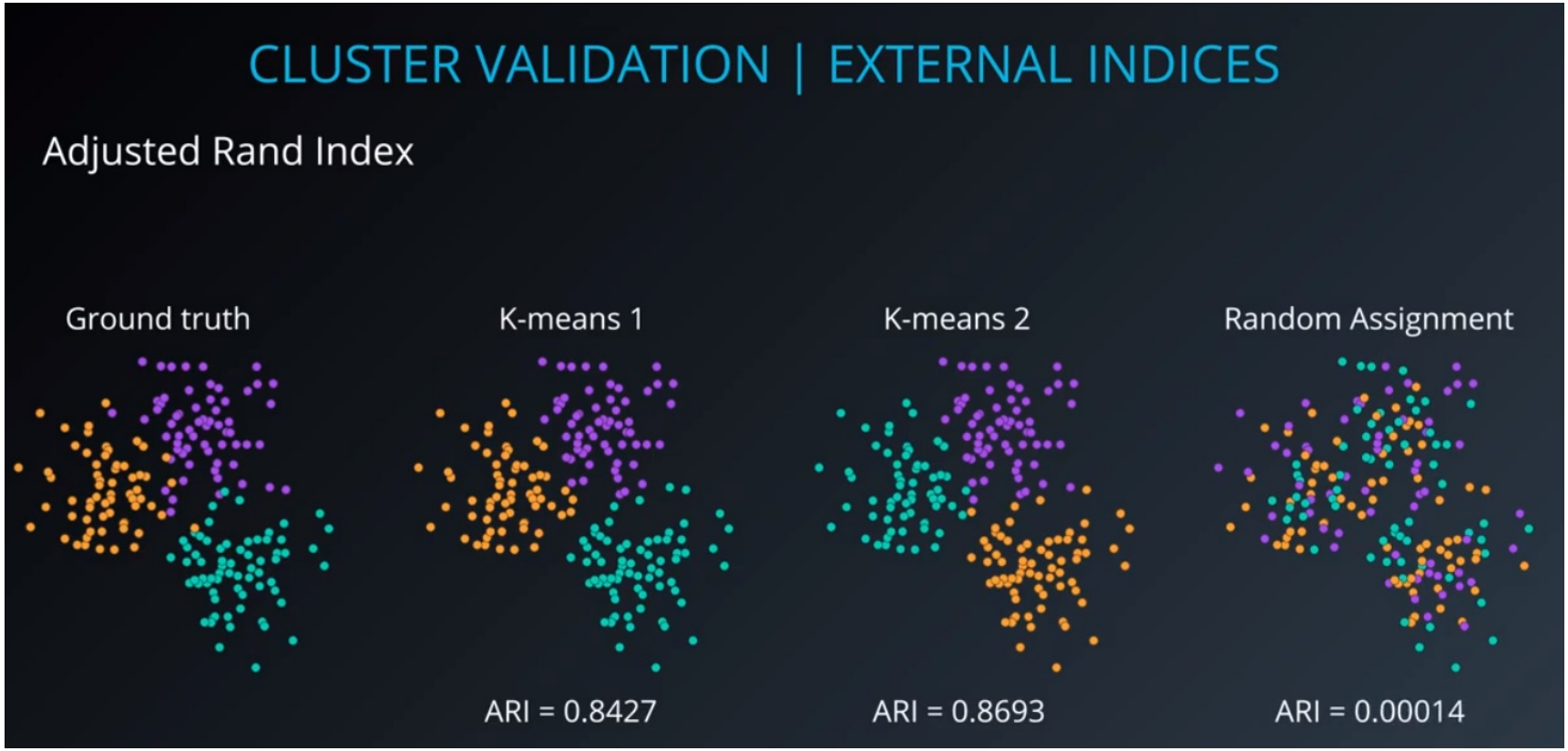
Clustering Algorithm Design or Selection



Cluster Validation

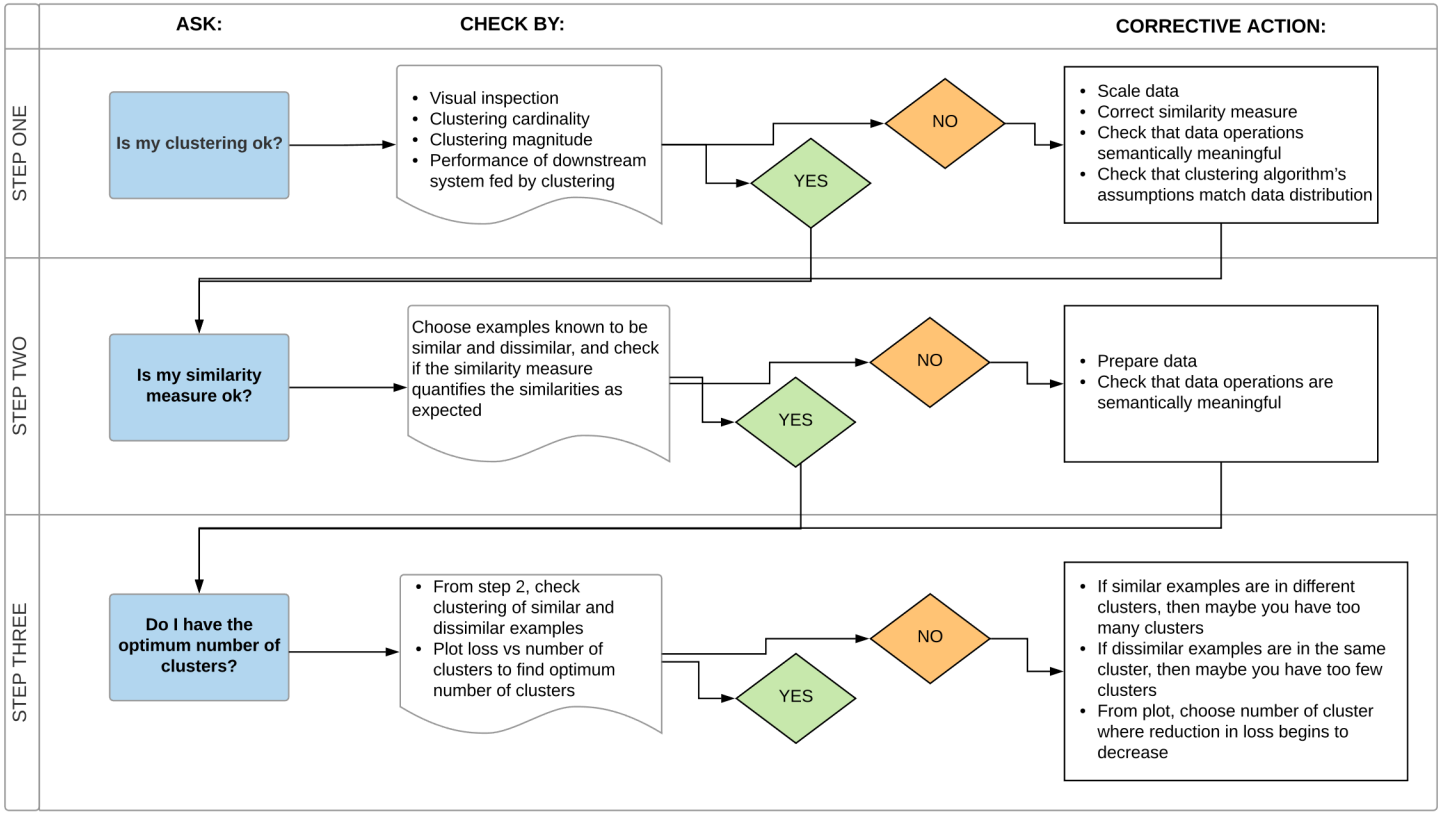
- Given a data set, each clustering algorithm can always generate a division, no matter whether the structure exists or not
- Moreover, different approaches usually lead to different clusters; and even for the same algorithm, parameter identification or the presentation order of input patterns may affect the final results
- Effective evaluation standards and criteria are important to provide the users with a degree of confidence for the clustering results derived from the used algorithms

Cluster Validation



Results Interpretation

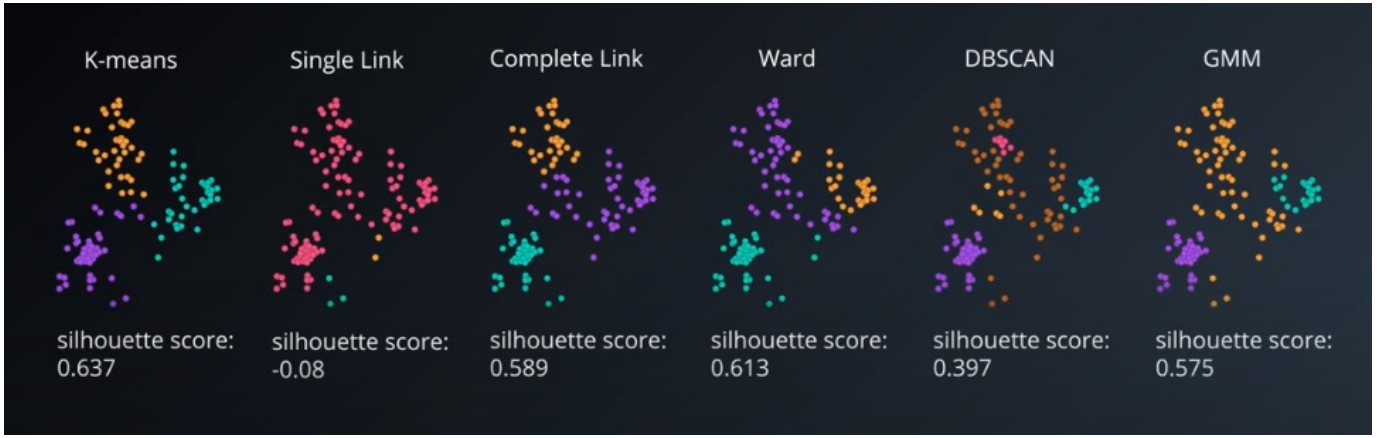
The ultimate goal of clustering is to provide users with meaningful insights from the original data, so that they can effectively solve the problems encountered



Clustering Algorithms

There are different types of clustering algorithms that handle all kinds of unique data

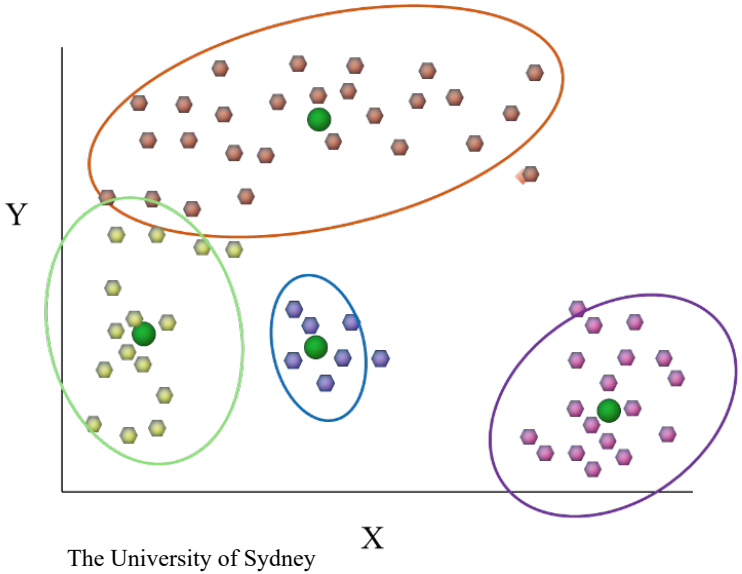
- Centroid-based: K-Means
- Connectively-based
- Distribution-based: DBSCAN
- Density-based: GMM
- Hierarchical-based
- …



Centroid-Based Clustering

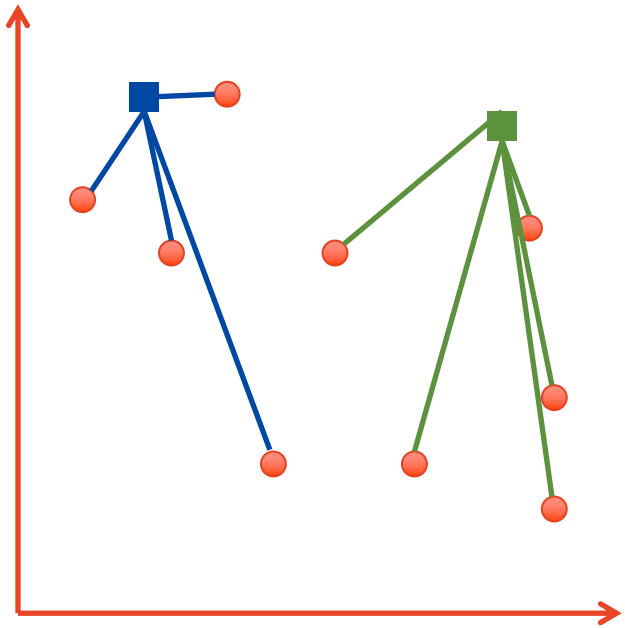
Centroid-based clustering is the one you probably hear about the most. It's a little sensitive to the initial parameters you give it, but it's fast and efficient.

These types of algorithms separate data points based on multiple centroids in the data. Each data point is assigned to a cluster based on its squared distance from the centroid. This is the most commonly used type of clustering.



K-means clustering is the most commonly used clustering algorithm. It's a centroid-based algorithm and the simplest unsupervised learning algorithm.

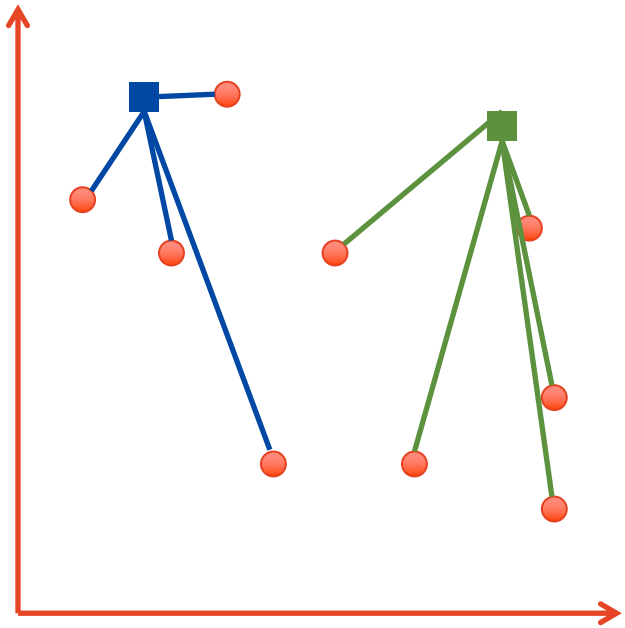
Clustering – unsupervised (machine) learning



k-means clustering:

1. Pick how many clusters you want, “k”
2. Place each cluster at a random point in space (“centroids”)
3. Assign each data point to the nearest centroid

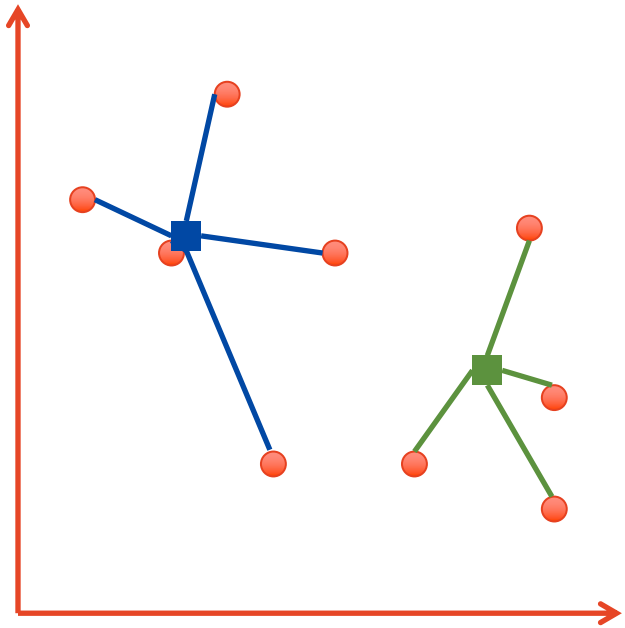
Clustering – unsupervised (machine) learning



k-means clustering:

1. Pick how many clusters you want, “k”
2. Place each cluster at a random point in space (“centroids”)
3. Assign each data point to the nearest centroid
4. Update centroids, mean location of their data
5. When centroids don’t move (much) stop. Else, go back to #3.

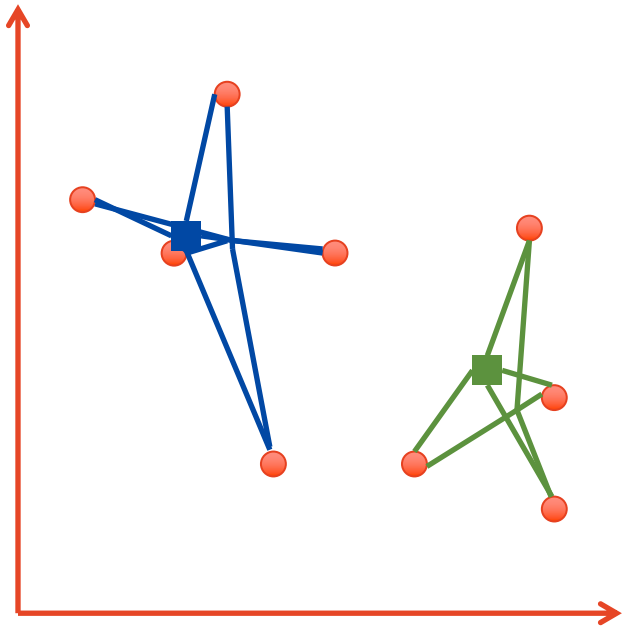
Clustering – unsupervised (machine) learning



k-means clustering:

1. Pick how many clusters you want, “k”
2. Place each cluster at a random point in space (“centroids”)
3. Assign each data point to the nearest centroid
4. Update centroids, mean location of their data
5. When centroids don’t move (much) stop. Else, go back to #3.

Clustering – unsupervised (machine) learning



k-means clustering:

1. Pick how many clusters you want, “k”
2. Place each cluster at a random point in space (“centroids”)
3. Assign each data point to the nearest centroid
4. Update centroids, mean location of their data
5. When centroids don’t move (much) stop. Else, go back to #3.

k is predefined. Final clusters depend on k’s starting points.

Centroid-Based Clustering - K-Means

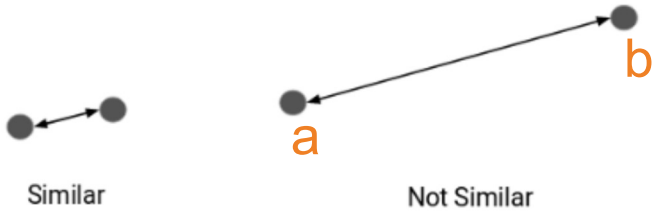
Given an input parameter k and a set of points in a N-dimensional space, K-means seeks to divide the points into k clusters so as to minimize a cost function. In particular, the objective is to partition a set of observations $x_1, x_2, ..., x_N$ into k groups $S = S_1, S_2, ..., S_N$ so as to minimize the within-cluster sum of squares:

$$\underset{s}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||_2$$

where μ_j is the mean of cluster j.

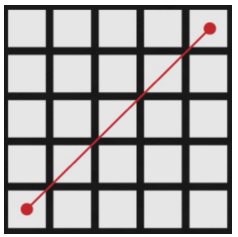
Measuring "distance"/similarity between records.

- Euclidean
  - 'Straight line' ('as the crow flies')
  - Only works on numeric data
  - In equation: '*f*' means feature, add up for all features
- Manhattan
  - Distance along each axis/feature
  - Like walking around city blocks (can't go through walls)
  - Only works on numeric data
- Minowski
  - Generalised form of the other two
  - Higher **orders** further punish dissimilarities
  - Only works on numeric data
- Hamming
  - Count similarities in each feature
  - *Can accommodate categorical data*



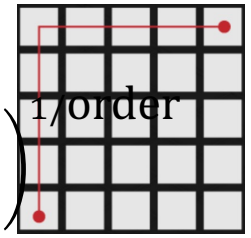
$$D(a,b) = \left(\sum^{f} \left(|a_f - b_f|\right)^2\right)^{1/2}$$

Euclidean



$$D(a,b) = \sum^{f} |a_f - b_f|$$

Manhattan



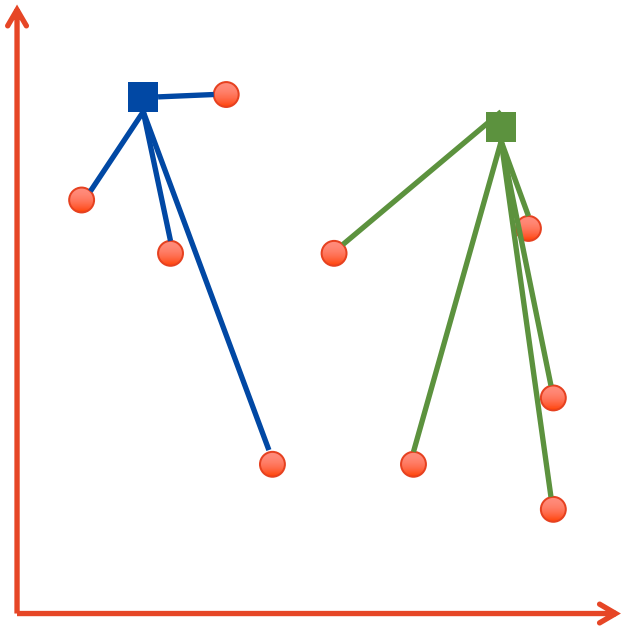
$$D(a,b) = \left(\sum^{f} \left(|a_f - b_f|\right)^{\text{order}}\right)^{1/\text{order}}$$

Hamming

(assume each character is a feature)

Hyperparameters for K-means clustering

- K – the number of clusters
- The starting position of the cluster centroids
  - There are variant algorithms that try to pick this more sensibly

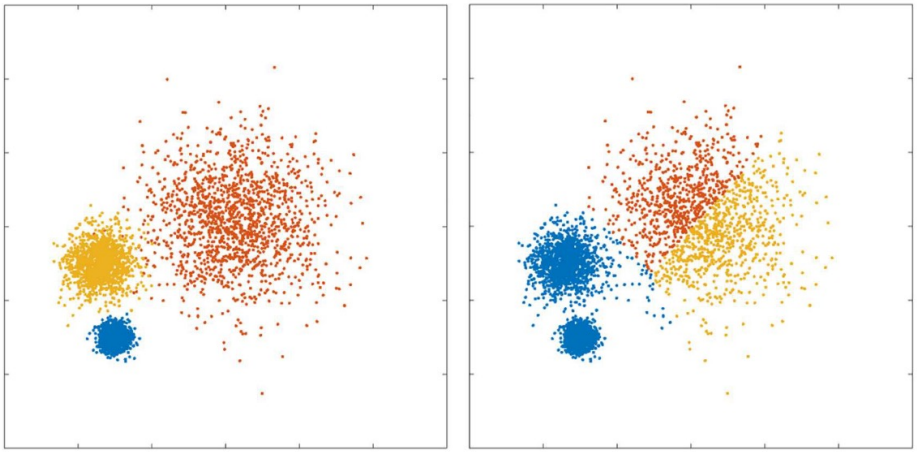


Pros and cons of k-means

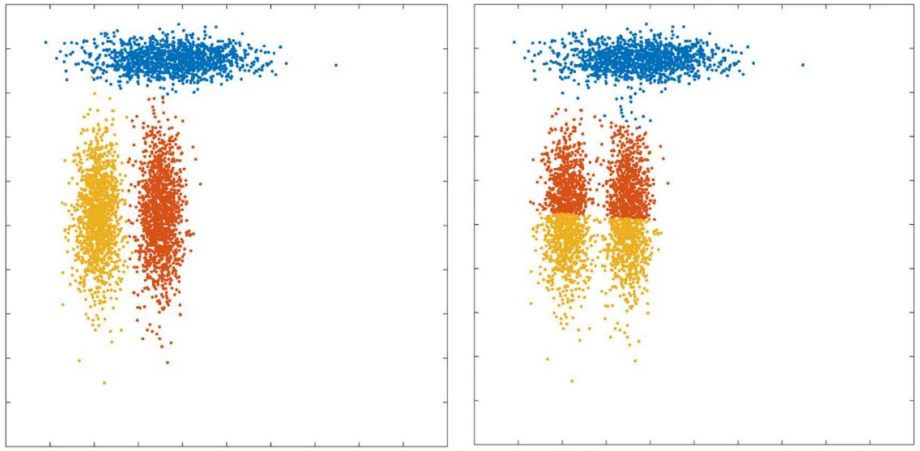
- Pros: Simple, and very popular.

Quite a few cons…

- Necessitates numerical data
- Can be "tripped up" by outliers.
- Clusters are always spherical (represented by their centroids)
- Clusters assumed to have equal radius
  - It ignores differences in density.
- Must know *k* in advance. Even thought we usually don't!
- *K*-means is sensitive to the starting position of the cluster centroids.
  - Can get stuck in *local optimum*.



(a) Generated synthetic data (b) K-means



(a) Generated synthetic data (b) K-means

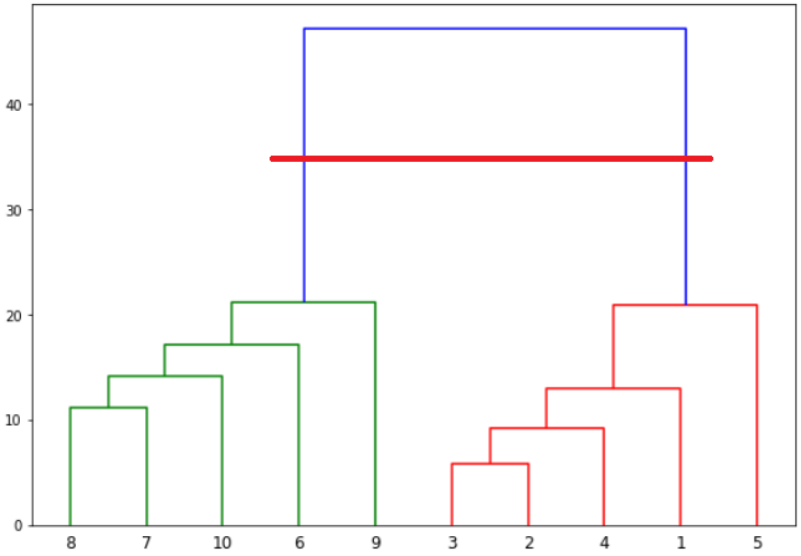
[Raykov 2016]

Hierarchical-Based Clustering

Hierarchical-Based Clustering

Hierarchical-based clustering is typically used on hierarchical data, like you would get from a company database or taxonomies. It builds a tree of clusters so everything is organized from the top-down.

This is more restrictive than the other clustering types, but it's perfect for specific kinds of data sets.

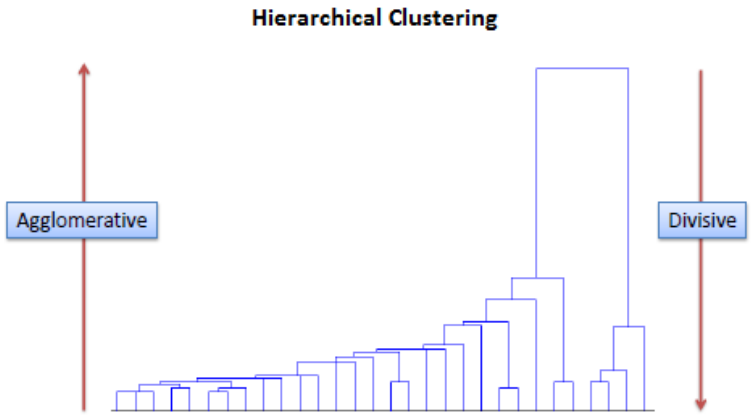


Hierarchical-Based Clustering

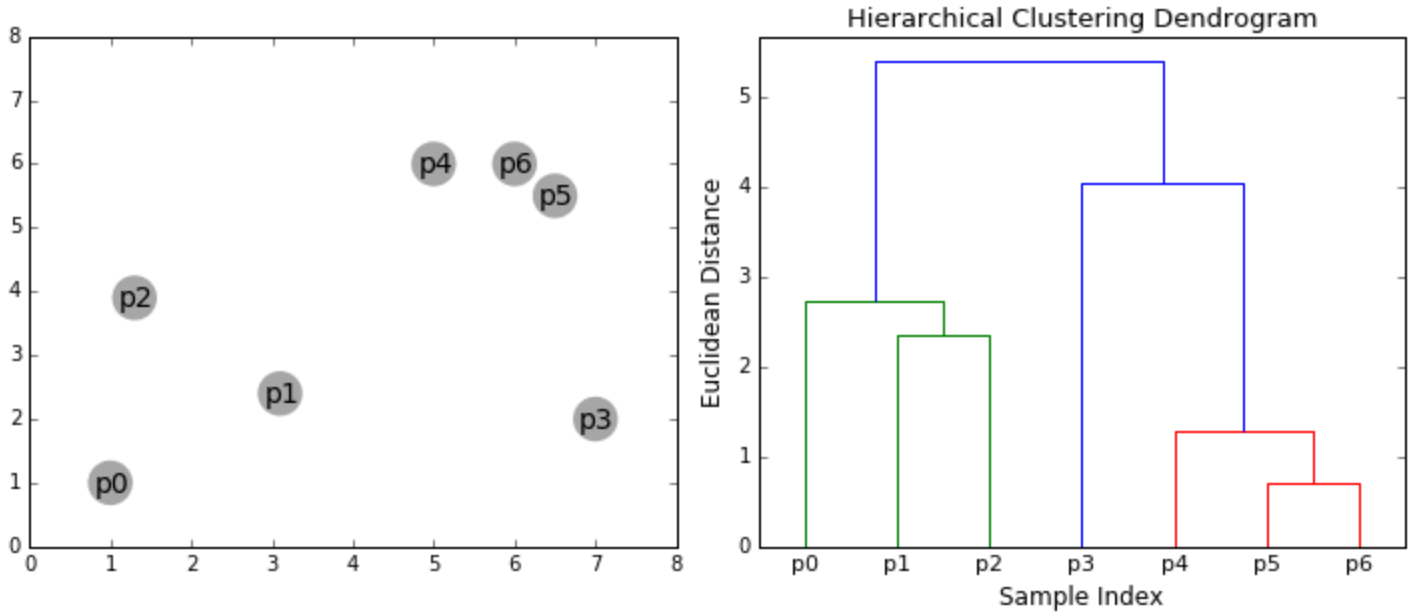
Hierarchical clustering algorithms group similar objects into groups called clusters. There are two types of hierarchical clustering algorithms:

- Agglomerative — Bottom up approach. Start with many small clusters and merge them together to create bigger clusters.

- Divisive — Top down approach. Start with a single cluster than break it up into smaller clusters.



Hierarchical-Based Clustering



Hierarchical-Based Clustering

Some pros and cons of Hierarchical Clustering:

Pros

- No assumption of a particular number of clusters (i.e. k-means)
- May correspond to meaningful taxonomies

Cons

- Once a decision is made to combine two clusters, it can't be undone
- Too slow for large data sets, O(n^2 log(n))

Density-Based Clustering

DBSCAN

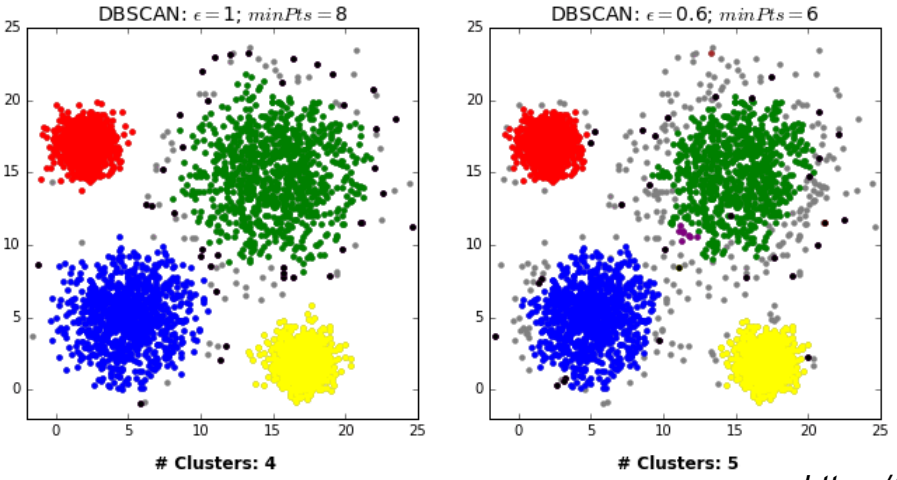
–Density-based Clustering Method: DBSCAN

–Discovers clusters of arbitrary shape. Clustering method

DBSCAN (Ester et al., KDD 96)

Density-Based Spatial Clustering of Applications with Noise

DBSCAN is a simple and effective density-based clustering algorithm that illustrates a number of important concepts that are important for any density-based clustering approach.



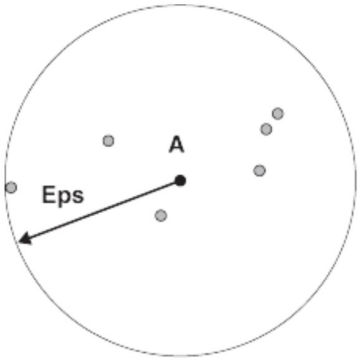
https://dashee87.github.io/images/DBSCAN_search.gif

DBSCAN

– **DBSCAN: D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (Ester, 96)

– **A density-based notion of cluster**

- *A cluster is defined as maximal set of density-connected points*
- *Density = number of points within a specified radius (Eps)*

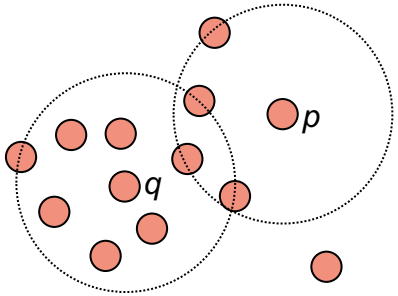


– **Two parameters:**

- **Epsilon (Eps - Ɛ):** Maximum radius of the neighbourhood
  E.g., the number of points within a radius of *Eps* of point *A* is 7, including *A* itself.
- **Minimum Points (MinPts):** Minimum number of points in the Eps-neighourhood of a point

The Eps(ε)-neighborhood of a point q:

$N_{Eps}(q) : \{p\ belongs\ to\ D \mid dist(p,q) \leq Eps\}$



MinPts = 5

If the radius is large enough, then all points will have a density of *m*, the number of points in the data set. Likewise, if the radius is too small, then all points will have a density of 1. 😭

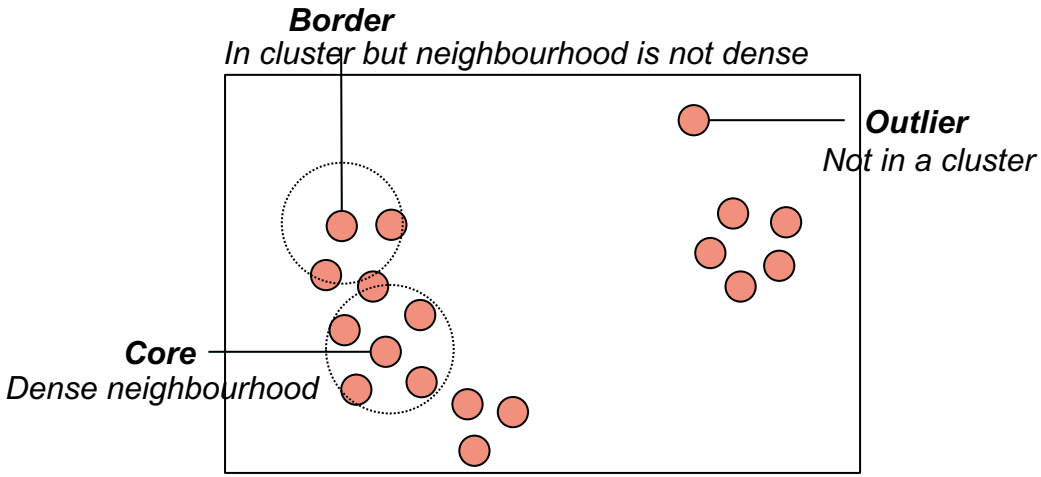
DBSCAN

–DBSCAN: Density-Based Spatial Clustering of Applications with Noise (Ester, 96)

–The center-based approach to density allows us to classify a point as being

- A **core point** if it has at least a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
  - Counts the point itself
- A **border point** is not a core point, but is in the neighborhood of a **core point**
- A **noise/outlier point** is any point that is not a core point or a border point

Any two core points that are close enough—within a distance *Eps* of one another—are put in the same cluster.
Any border point that is close enough to a core point is put in the same cluster as the core point. (Ties need to be resolved if a border point is close to core points from different clusters.)
Noise points are discarded.

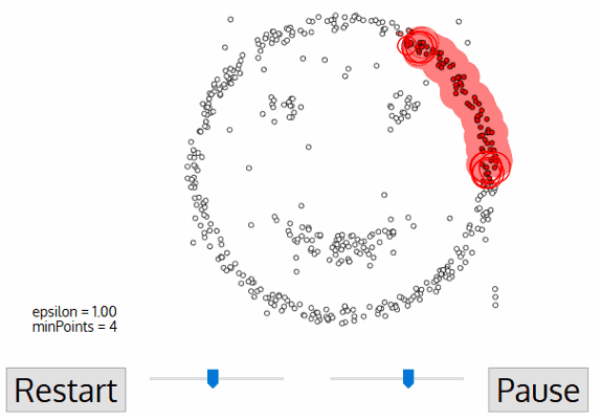


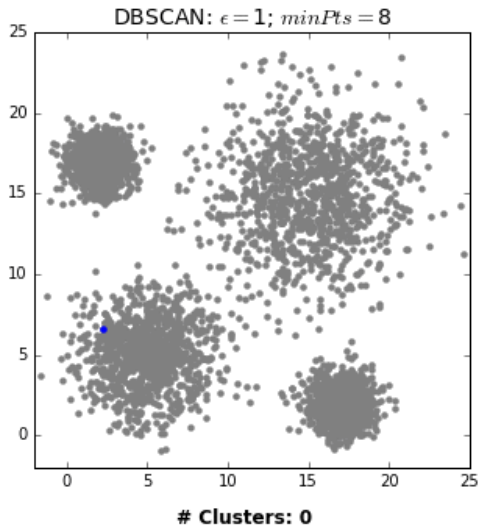
DBSCAN

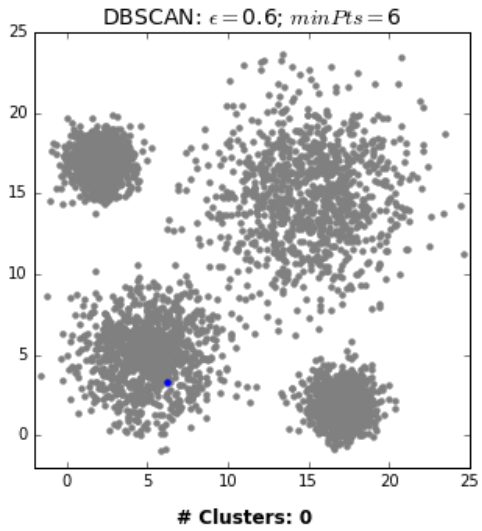
–DBSCAN: Density-Based Spatial Clustering of Applications with Noise (Ester, 96)

–Algorithm

- Arbitrarily select a point p
- Retrieve all points density-reachable from p with Eps (ε), MinPts
  - –If p is a core point, a cluster is formed
  - –If p is a border point, no points are density-reachable from p, and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed







- *If the radius is too large, then all points are core points*
- *If the radius is too small, then all points are outliers*

Measuring clustering performance

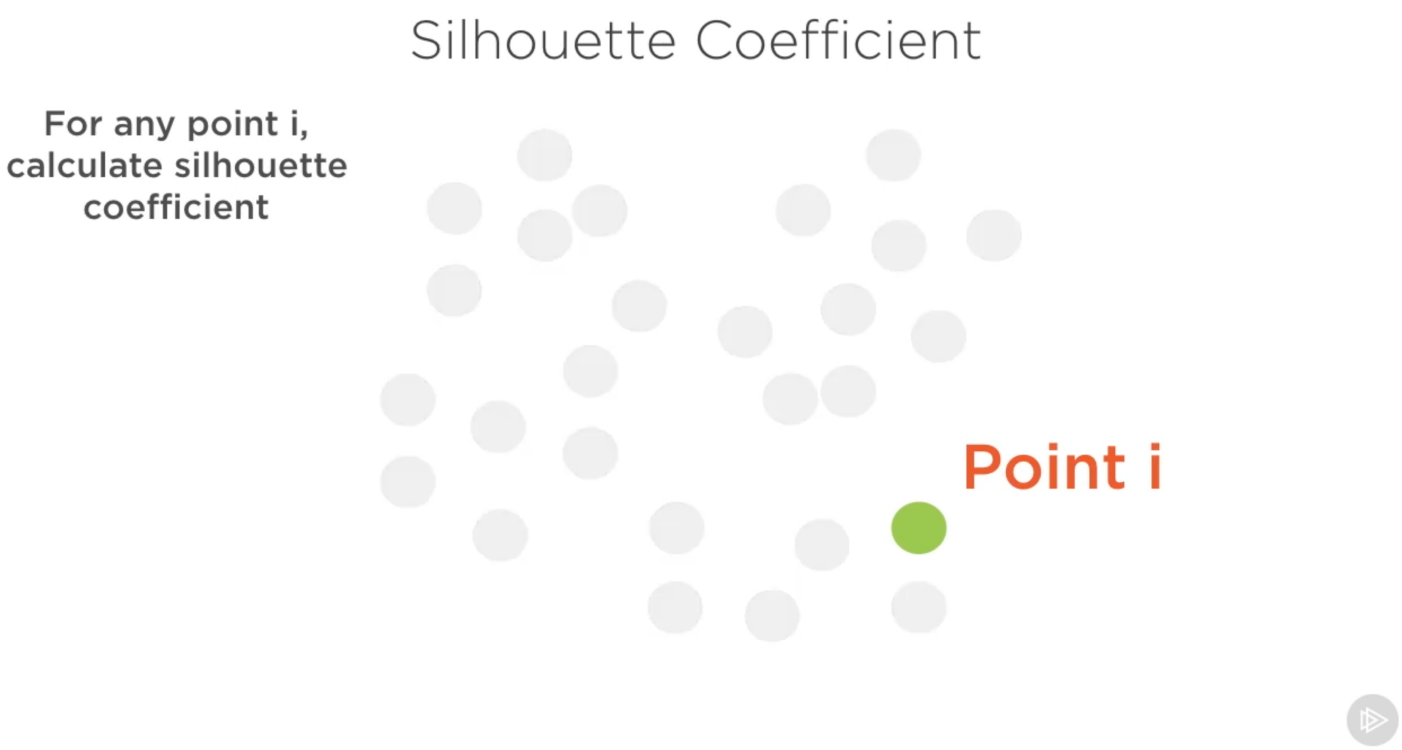
How good is your clustering?

- Silhouette coefficient.
  - Best value = 1; worst value = -1
- Each data point gets a score.
- Score = how consistent the data-point is with its current cluster, relative to the next nearest cluster.
- Can be used to spot outlier data-points, and examine the quality of each cluster.
- Can take average over all data-points to evaluate the whole clustering effort.

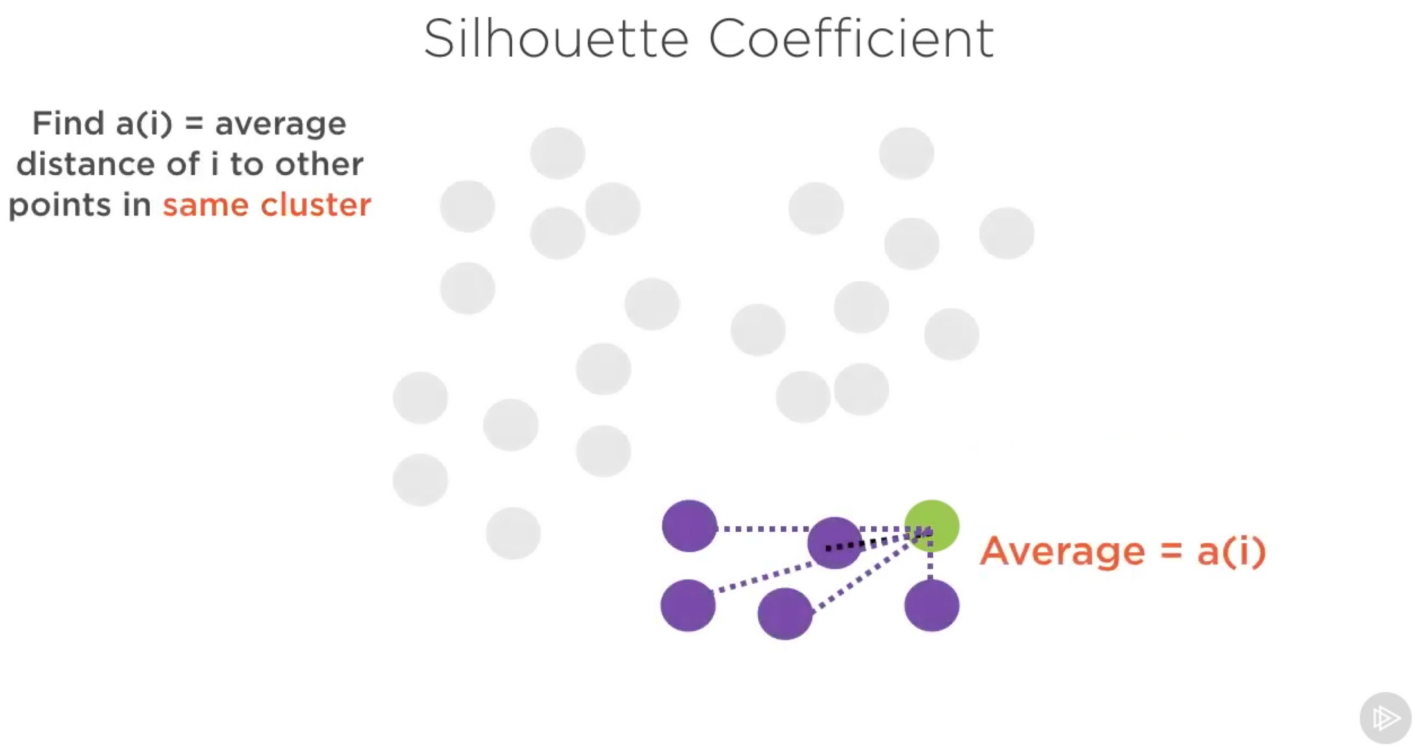
Silhouette Coefficient

For any point i, calculate silhouette coefficient

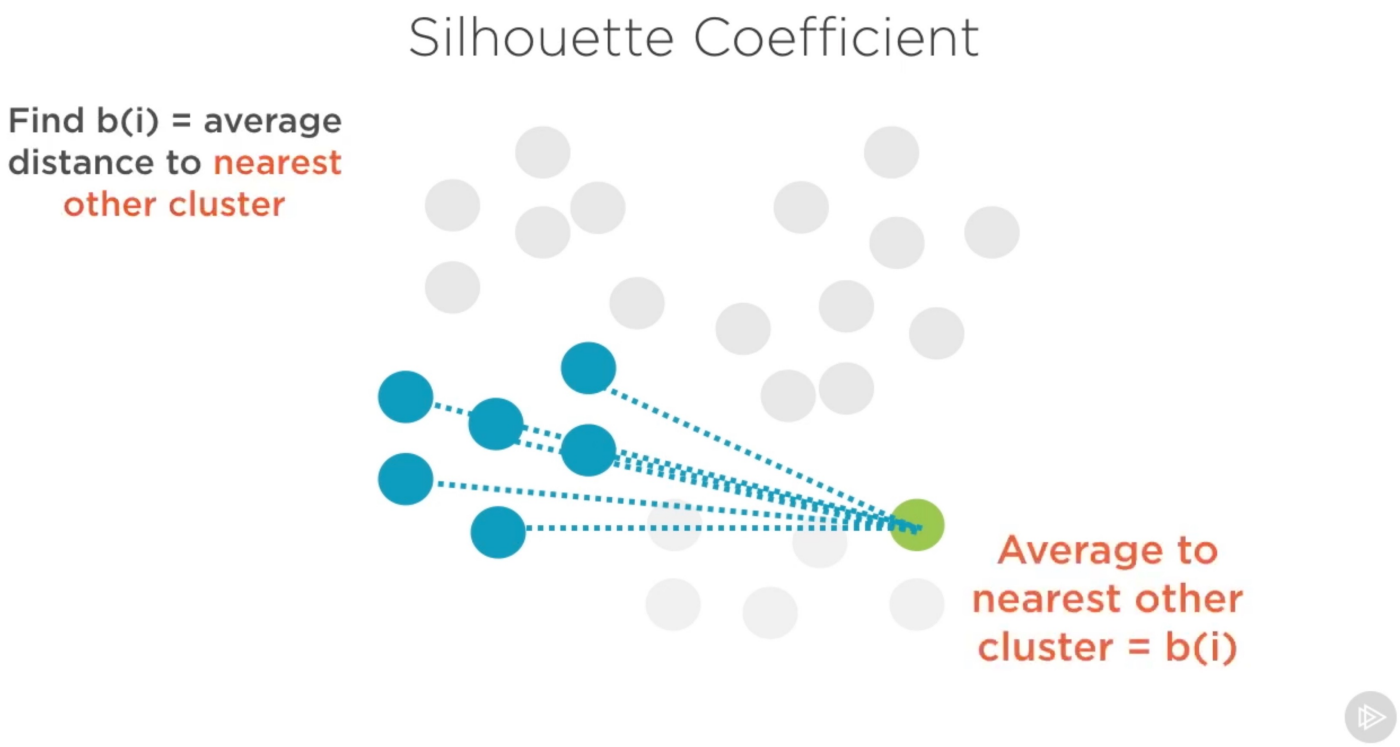
[https://www.youtube.com/watch?v=AtxQ0rvdQIA]



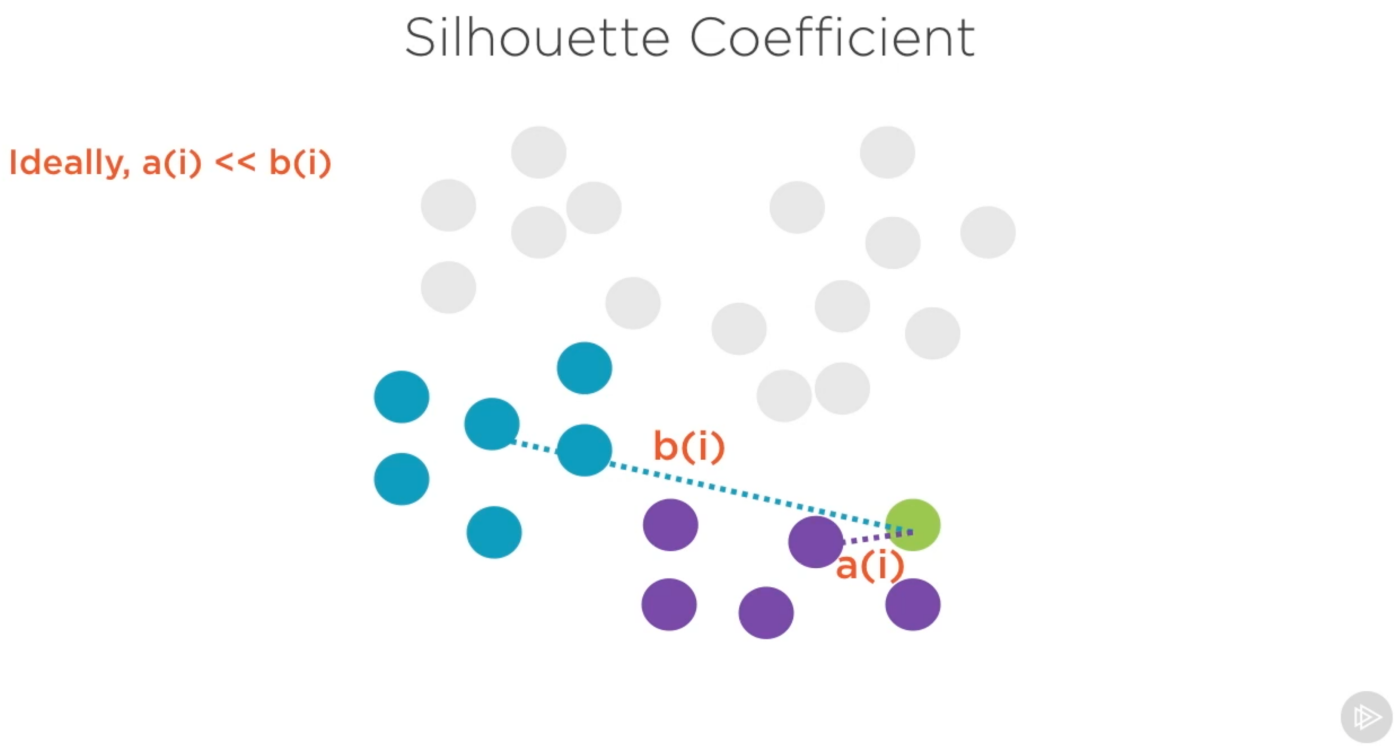
[https://www.youtube.com/watch?v=AtxQ0rvdQIA]



[https://www.youtube.com/watch?v=AtxQ0rvdQIA]



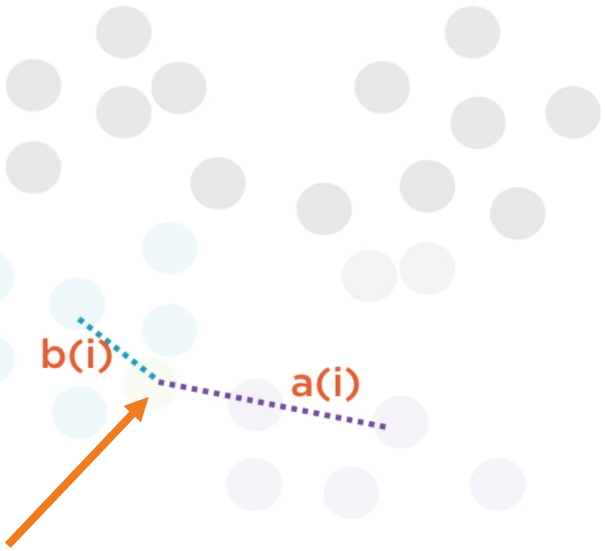
[https://www.youtube.com/watch?v=AtxQ0rvdQIA]



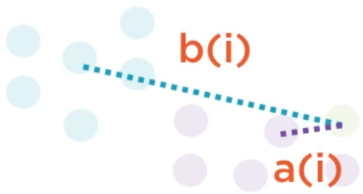
[https://www.youtube.com/watch?v=AtxQ0rvdQIA]

Silhouette Coefficient

If a(i) > b(i), i is likely misclassified



Pretend/imagine that the datapoint was here instead, and belonged to cluster A. That would be incorrect, because it is closer to cluster B.
[https://www.youtube.com/watch?v=AtxQ0rvdQIA]



Silhouette Coefficient

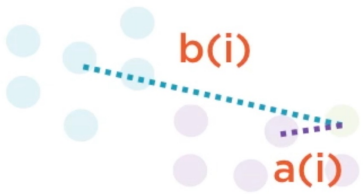
For any point i

$$s(i) = \frac{b(i) - a(i)}{\text{Larger of } b(i) \text{ and } a(i)}$$

a(i) = Average distance inside cluster

b(i) = Average distance to nearest other cluster

[https://www.youtube.com/watch?v=AtxQ0rvdQIA]

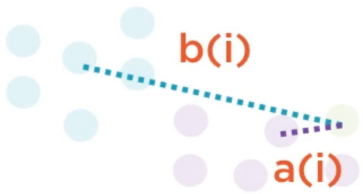


Worst-case s(i) = -1

Worst case, a(i) = Infinity, b(i) = 0

$$s(i) = \frac{b(i) - a(i)}{\text{Larger of } b(i) \text{ and } a(i)} = -1$$

[https://www.youtube.com/watch?v=AtxQ0rvdQIA]



Ideally s(i) = 1

Ideally, a(i) = 0, b(i) = Infinity

$$s(i) = \frac{b(i) - a(i)}{\text{Larger of } b(i) \text{ and } a(i)}$$

[https://www.youtube.com/watch?v=AtxQ0rvdQIA]

Thank you!