Machine Learning and Deep Learning
Lecture-03

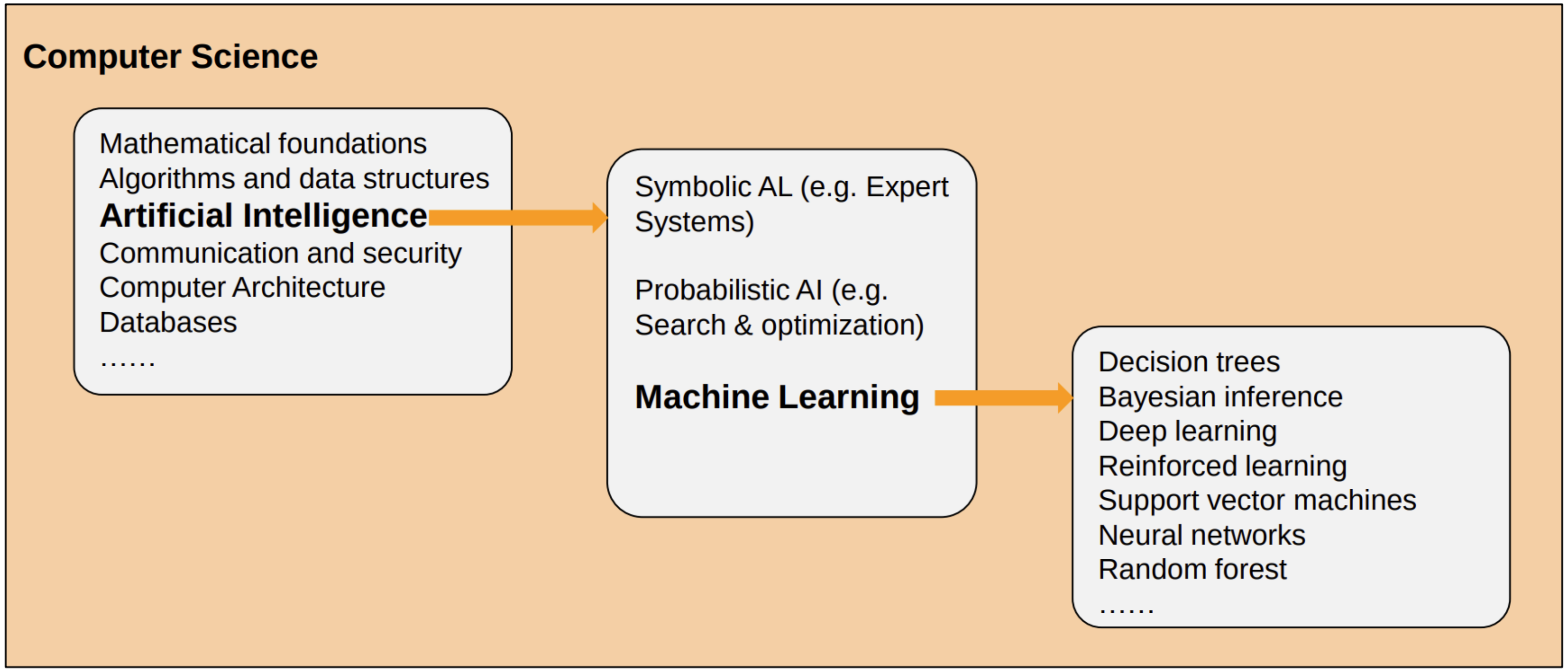
By: Somnath Mazumdar
Assistant Professor
sma.digi@cbs.dk

Outline

- Introduction
- Unsupervised Machine Learning
  - K-Means
  - Density Clustering DBSCAN
  - Hierarchical Clustering

Fundamentals of Machine Learning

- Artificial Intelligence (AI) is a branch or Computer Science that uses algorithms and techniques to mimic human intelligence.
- Machine Learning (ML) is one of several AI techniques for sophisticated cognitive tasks.



Fundamentals of Machine Learning

Traditional AI techniques

Static, Rule Based, No generalization

Machine Learning

Dynamic, Data driven, Generalization

Adpated Slides from Fundamentals of Machine Learning By Ekpe Okorafor

Fundamentals of Machine Learning

- Three Approaches:
  - Example - Excelling at playing chess game.

- Symbolic AI: “Let us sit down with the world’s best chess player, Magnus Carlsen, and put his knowledge into a computer program”.

- Statistical AI: “Let us simulate all the different possible moves and the associated outcomes at each single step and go with the most likely to win”.

- Machine Learning Approach: “Let us show millions of examples or real life and simulated games (won and lost) to the program, and let it learn from experience”.

Fundamentals of Machine Learning

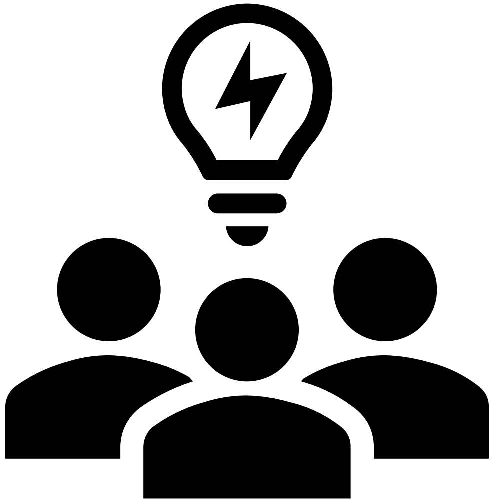
- ML is good at solving 2 types of problems where other AI techniques fail:
  1. Tasks programmers can’t describe (Handwriting, Cognitive Reasoning).
  2. Complex multidimensional problems that can’t be solved by numerical reasoning (Weather Forecasting, Health Care Outcomes).

- Unsupervised learning: Want to find unknown structures or trends.

- Supervised learning: Already know answers we want (found in past or completed data).

What do you think?

- Consider a financial startup, 'X' have a large data set, but it does not have any data scientists or ML-guys but has computer science guys. How should they proceed? They do not know what questions to ask!!
- Condition: They will wait to hire a new data scientist!!
- Hints: Supervised vs Unsupervised model – which one they should apply and why?



2 minutes to think

What do you think?

- Unsupervised model
  1. Examine various properties of data.
  2. No data labeling.
  3. Simpler model.
  4. Cheaper to run.

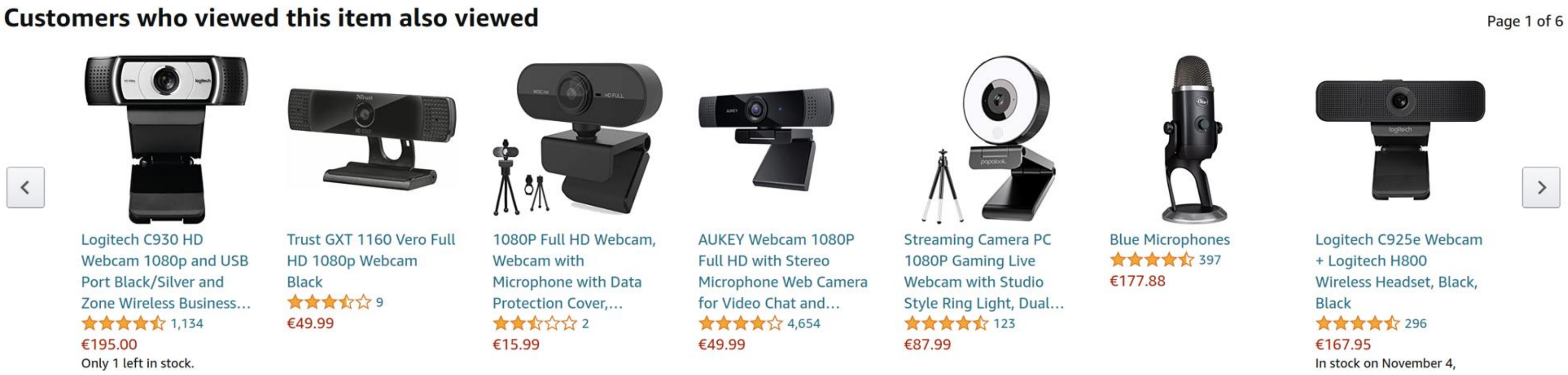
Unsupervised Learning

- Unsupervised Learning: Search for structure in data. Unknown targets.
  - User inputs data with undefined answers.
  - Machine finds useful information hidden in data.
- Example:
  - Cluster analysis: Group into sets: K-Means Clustering, Hierarchical Clustering.
  - Density reduction: Select relevant variables.
  - Dimension Reduction: Principal component analysis.

Three C’s of ML

- Three C’s of ML:
    1. Collaborative filtering
    2. Clustering
    3. Classification
- Collaborative filtering is a technique for recommendations.
    - It’s one primary type of recommender system.
    - Can use the same algorithm to recommend practically anything – Movies (Netflix).

    - Amazon uses it to recommend a variety of products.



Three C’s of ML

- Clustering algorithms discover structure in collections of data.
  - Where no formal structure previously existed.
  - They discover what clusters (‘groupings’), naturally occur in data
    - By examining various properties of input data.
- Clustering is often used for exploratory analysis: Divide and Explore !!

- Applications:
  - Market segmentation –> Targeted marketing.
  - Finding related news articles –> Google News.

Three C’s of ML

- Classification is a form of ‘supervised’ learning.
  - Requires training with data that has known labels
  - Learns how to label new records based on that information.

- Applications:
  - Spam filtering: Train using a set of spam and non/spam messages --> System will eventually learn to detect unwanted e-mail
  - Risk Analysis: Train using financial records of customers who do/don’t default --> System will eventually learn to identify risk customers

No 'BEST!!'

- There are many candidate algorithms for a problem.
- No overall best algorithm.
- Each algorithm has advantages and limitations.
  - Best approach = simple algorithm + lots of data.
- Algorithm choice is often related to data volume <Bigdata>.
- Some scale better than others.
  - Most algorithms offer better results as volume increases.

Applications

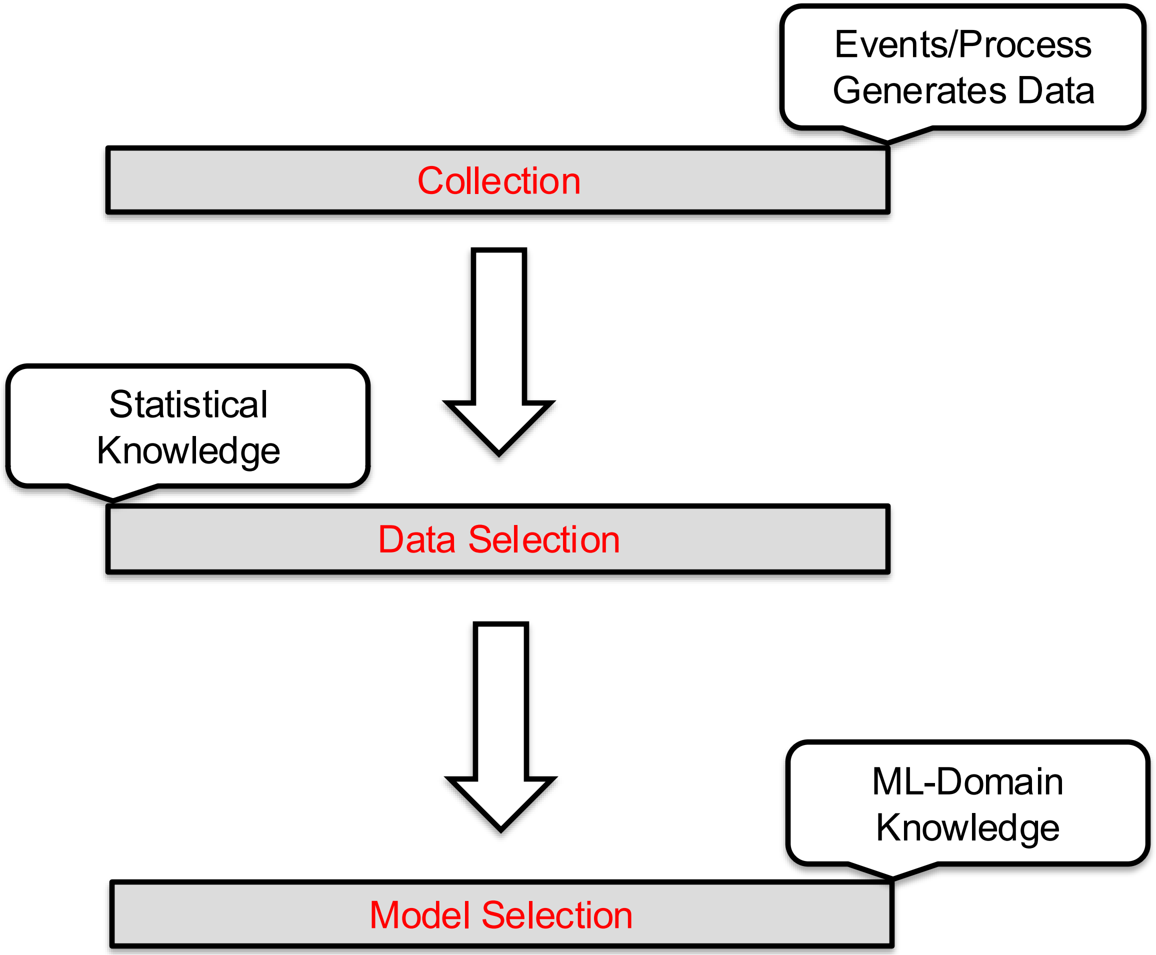
- Dimensionality reduction: As a preprocessing step to reduce number of dimensions while maintaining meaningful properties of input data.

- Outlier detection: Identify instances with low affinity to all clusters.

- Semi-supervised learning: When few labels are available, perform clustering and propagate the labels to all the instances in the same cluster.

	Supervised Learning	***Unsupervised Learning***
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Before you Start..



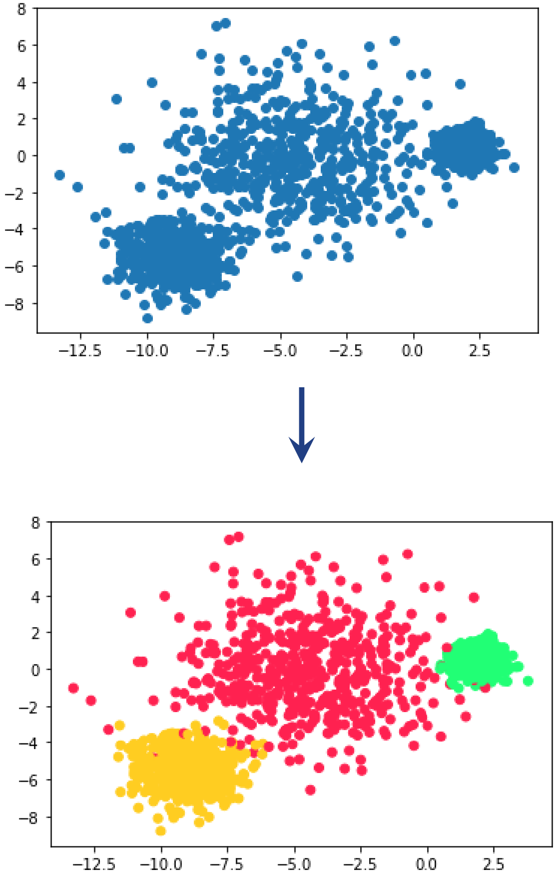
Clustering

Unsupervised learning

- Labels are not provided, and the algorithm learns how to group the data without being explicitly told what the groups are

- E.g., grouping a webstore's customers into segments or finding an outlier in transactions

- Example algorithms:
  - Clustering: K-means, DBSCAN, Hierarchical
  - Dimensionality reduction: Principal components analysis (PCA).



Example

- Often data comes without labels.
- Producing labeled training data can be difficult and time consuming.

Exclusive: OpenAI Used Kenyan Workers on Less Than $2 Per Hour to Make ChatGPT Less Toxic

https://time.com/6247678/openai-chatgpt-kenya-workers/

To build that safety system, OpenAI took a leaf out of the playbook of social media companies like Facebook, who had already shown it was possible to build AIs that could detect toxic language like hate speech to help remove it from their platforms. The premise was simple: feed an AI with labeled examples of violence, hate speech, and sexual abuse, and that tool could learn to detect those forms of toxicity in the wild. That detector would be built into ChatGPT to check whether it was echoing the toxicity of its training data, and filter it out before it ever reached the user. It could also help scrub toxic text from the training datasets of future AI models.

Clustering

- Input: Set of objects described by features x_i.

- Output: An assignment of objects to “groups”.

- Unlike classification, we are not given “groups”
  - Clustering algorithm must discover "groups".

- Examples of groups when clustering the webstore’s customers:
  – Wealthy customers
  – Discount hunting customers

Clustering

- Goals:
  - Objects in the same group should be ‘similar’.
  - Objects in different groups should be ‘different’.

- ‘Best’ clustering is hard to define:
  - No test error.
  - No ‘best’ method.

- Why cluster?
  - To know what the groups are.
  - A ‘prototype’ example for each group.
  - To find the group for a new example $x_{i.}$
  - To find objects related to a new example x_i.

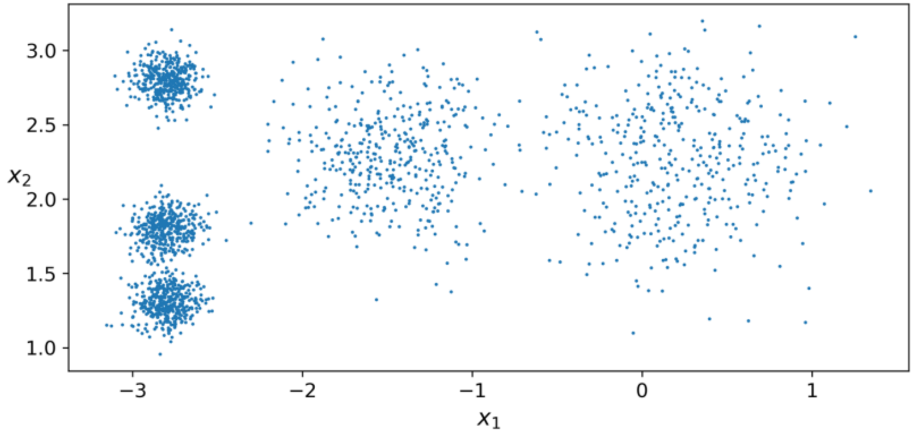
Applications of Clustering

- Clustering is used in a wide variety of applications:
  - For customer segmentation
  - For data analysis
  - As a dimensionality reduction technique
  - For anomaly detection (outlier detection)

K-Means

K-Means

- K-Means algorithm is a simple and efficient algorithm.
- It clusters dataset quickly, often in few iterations.
- Proposed by Stuart Lloyd at Bell Labs, 1957 but published in 1982.
- In 1965, Edward W. Forgy published virtually the same algorithm, it is referred to as Lloyd–Forgy.



An unlabeled dataset composed of five blobs of instances

Stuart P. Lloyd, “Least Squares Quantization in PCM,” IEEE Transactions on Information Theory 28, no. 2 (1982): 129–137

K-Means Algorithm

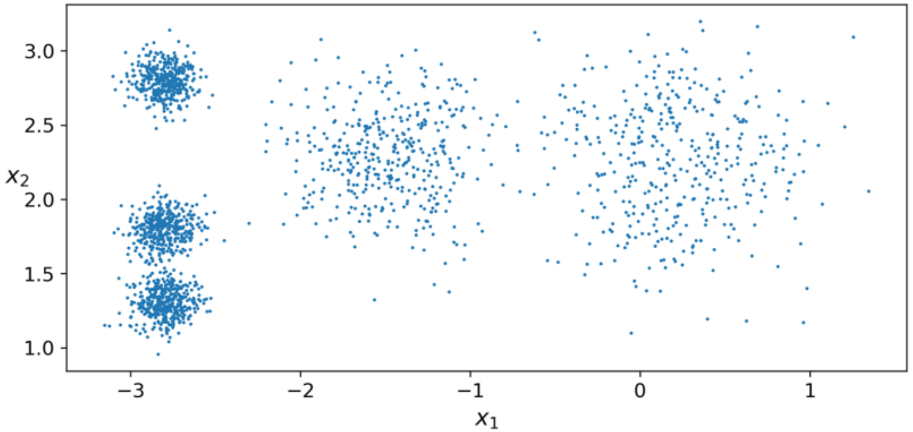
- Suppose centroids are given:
    - Can easily label all instances in dataset by assigning each of them to cluster whose centroid is closest.

- All instance labels were given:
    - Easily locate all centroids by computing mean of instances for each cluster.

- If neither labels nor centroids are given?
    - Start by placing centroids randomly (k-instances).
    - Label instances, update centroids, continue until centroids stop moving.

- It guaranteed to converge in a finite number of steps.

K-Means Algorithm



Unlabeled dataset composed of five blobs of instances

Q: Find each blob's center

```python
from sklearn.cluster import KMeans
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
```

Train a K-Means

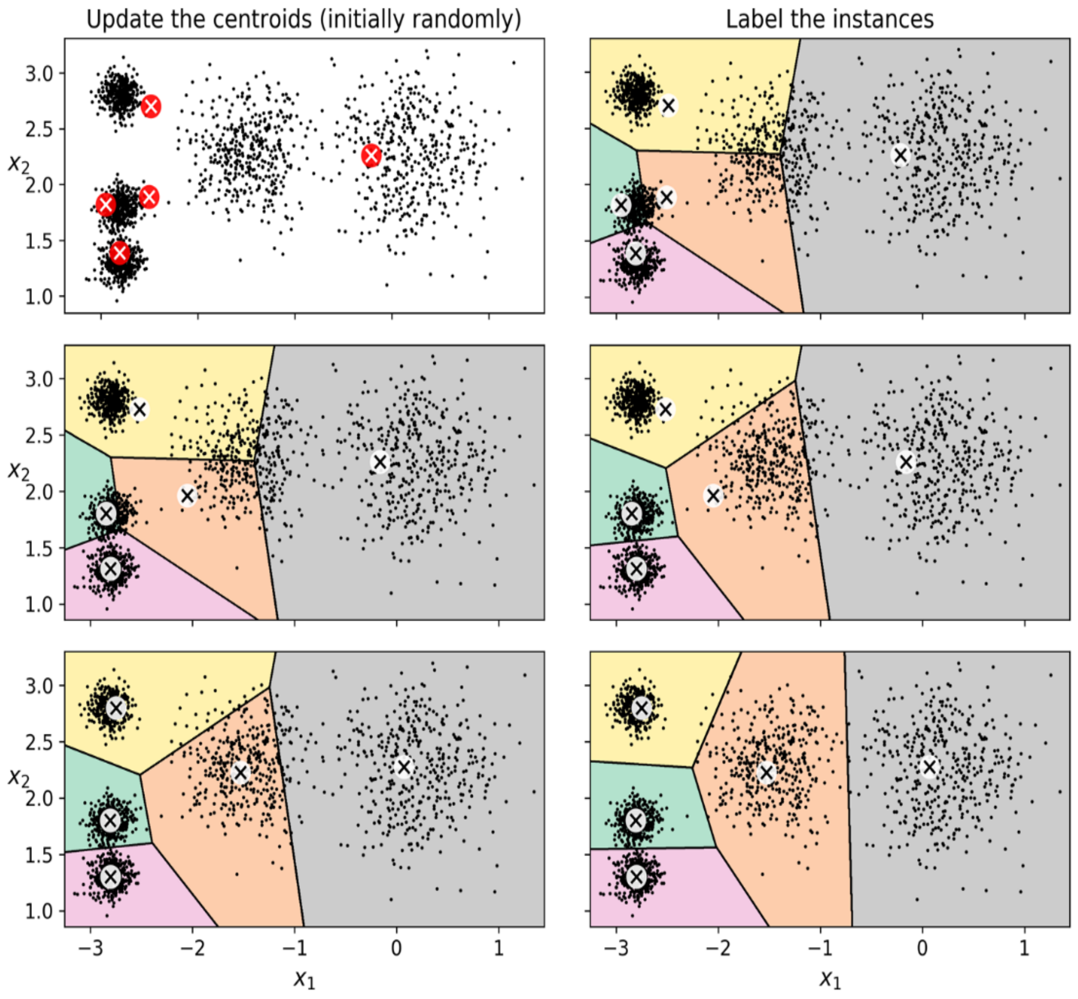
```python
>>> kmeans.cluster_centers_
array([[-2.80389616,  1.80117999],
       [ 0.20876306,  2.25551336],
       [-2.79290307,  2.79641063],
       [-1.46679593,  2.28585348],
       [-2.80037642,  1.30082566]])
```

Five centroids that the algorithm found

K-Means Algorithm

1. Centroids are initialized randomly (top left)
2. Instances are labeled (top right)
3. Centroids are updated (center left)
4. Instances are relabeled (center right)
5. and so on.

- Algorithm has reached an optimal clustering in 3 iterations.

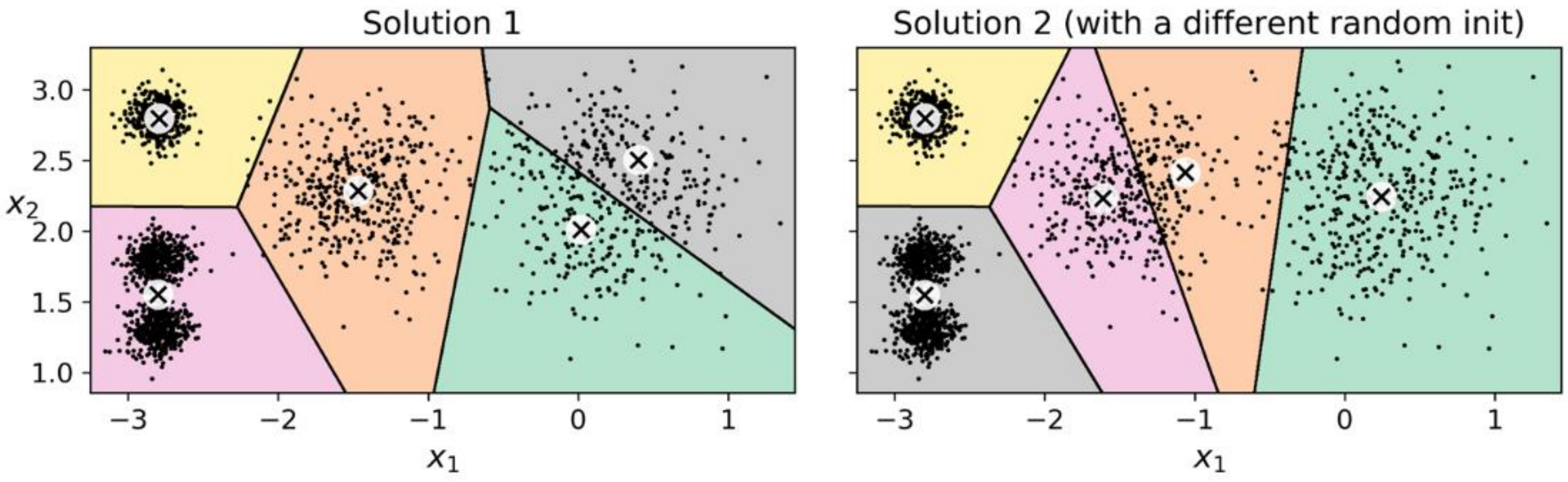


K-Means Algorithm

- Computational complexity is generally linear regarding number of:
  - m (instances), k (number of clusters), and n (number of dimensions).
  - True when data has a clustering structure.
  - If it does not, then in worst-case scenario complexity can increase exponentially with number of instances.
- K-Means is one of the fastest clustering algorithms.

K-Means Algorithm

- Algorithm is guaranteed to converge, BUT may not converge to right solution (i.e., converge to a local optimum):
  - Depends on centroid initialization.



K-Means Algorithm

- Centroid initialization methods:
  - Set init hyperparameter to a NumPy array containing the list of centroids.
  - Set n_init to 1: --> if you know approximately where the centroids should be.
  - Run the algorithm multiple times with different random initializations and keep the best solution.

```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1)
```

- n_init is hyperparameter.
- Algorithm will run 10 times

By default, n_init hyperparameter set to 10

K-Means Algorithm: Mini-Batch

- Mini-batch: Instead of using full dataset at each iteration, use mini-batches, moving the centroids just slightly at each iteration.
  - Speeds up the algorithm typically by a factor of 3x or 4x.
  - Possible to cluster huge datasets that do not fit in memory.

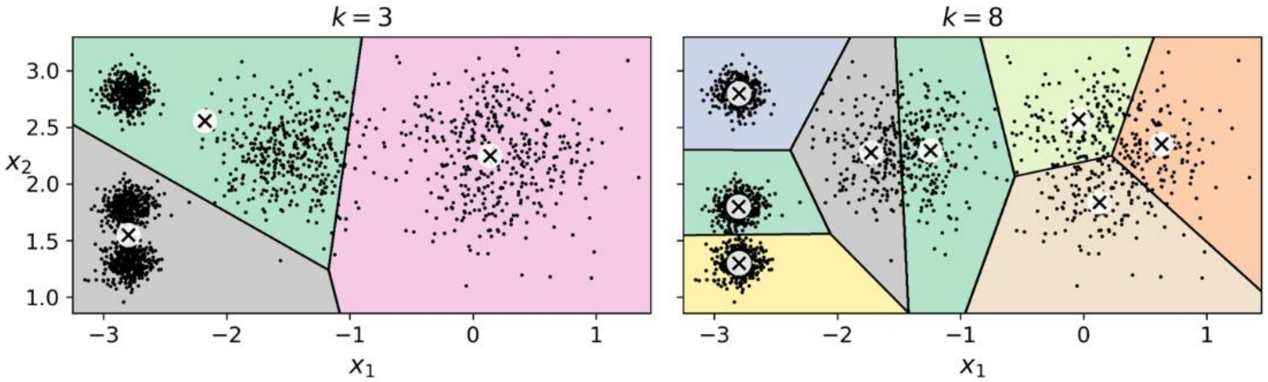
```
from sklearn.cluster import MiniBatchKMeans

minibatch_kmeans = MiniBatchKMeans(n_clusters=5)
minibatch_kmeans.fit(X)
```

David Sculley, “Web-Scale K-Means Clustering,” Proceedings of the 19th International Conference on World Wide Web (2010): 1177–1178.

Silhouette Score

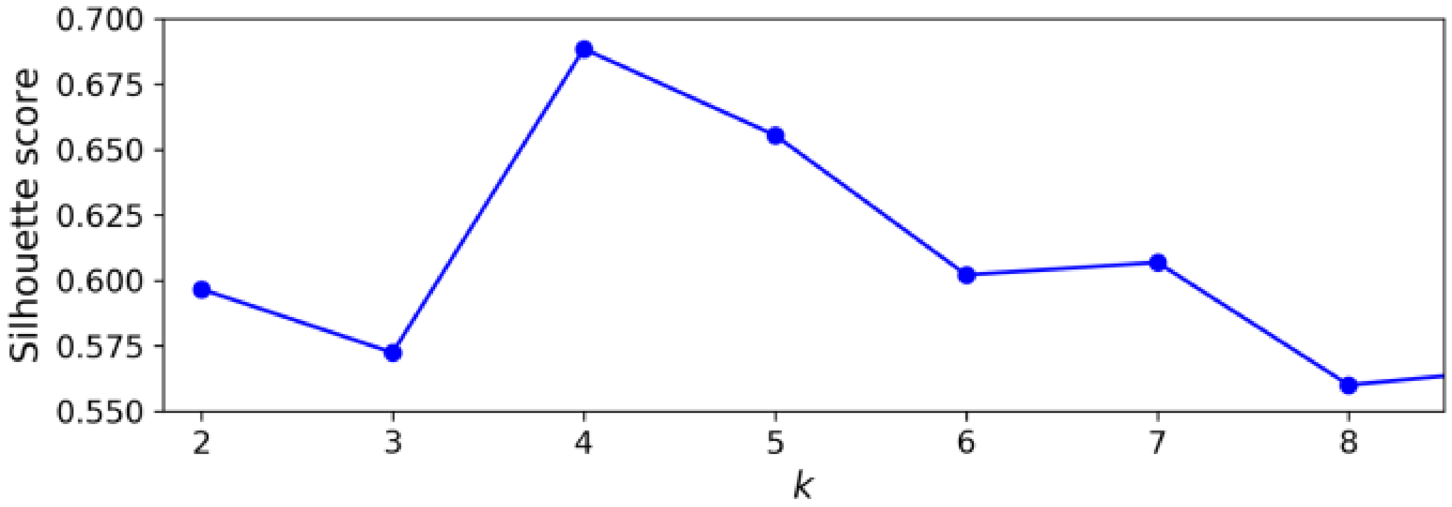
- Silhouette score to find optimal number of clusters.
- It is the mean silhouette coefficient over all the instances.
- (b–a)/max(a,b) where a: mean distance to other instances in same cluster, b: mean nearest-cluster distance.
- Silhouette coefficient vary between –1 and +1.



- k is too small, separate clusters get merged (left)
- k is too large, clusters get chopped into multiple pieces (right)

Silhouette Score

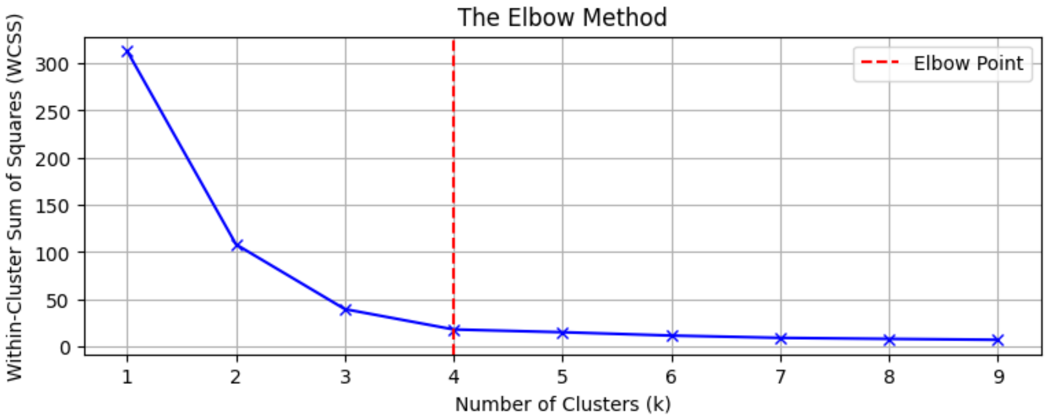
```
from sklearn.metrics import silhouette_score
silhouette_score(X, kmeans.labels_)
```



k = 4 is very good, k = 5 is quite good much better than k = 6 or 7.

Elbow Method

- Finds optimal K value.
  - It is the point at which the graph forms an elbow.
- Steps:
  - Iterate over a range (1 to n) of k values, n is a hyper-parameter.
  - For each k, calculate the Within-Cluster Sum of Squares (WCSS).
    - WCSS tells us how well data points are clustered around their respective centroids.



- Prefer the Silhouette score because:
  - Silhouette score is more effective to find number of K when the elbow method does not show the elbow point.
  - Elbow method can produce more ambiguous result.

Issues of K-Means

- Necessary to run algorithm several times to avoid sub-optimal solutions.
- Need to specify the number of clusters.
- Does not behave very well when the clusters have:
  - Varying sizes
  - Different densities
  - Non-spherical shapes.
- Must scale input features before you run K-Means.
- Scaling features improve the performance.

Density Clustering

Density Clustering

- It is an unsupervised learning method.
- Identify distinctive groups/clusters in input data.
- Idea: A cluster in a data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density.
- Data points in separating regions of low point density are typically considered noise/outliers.

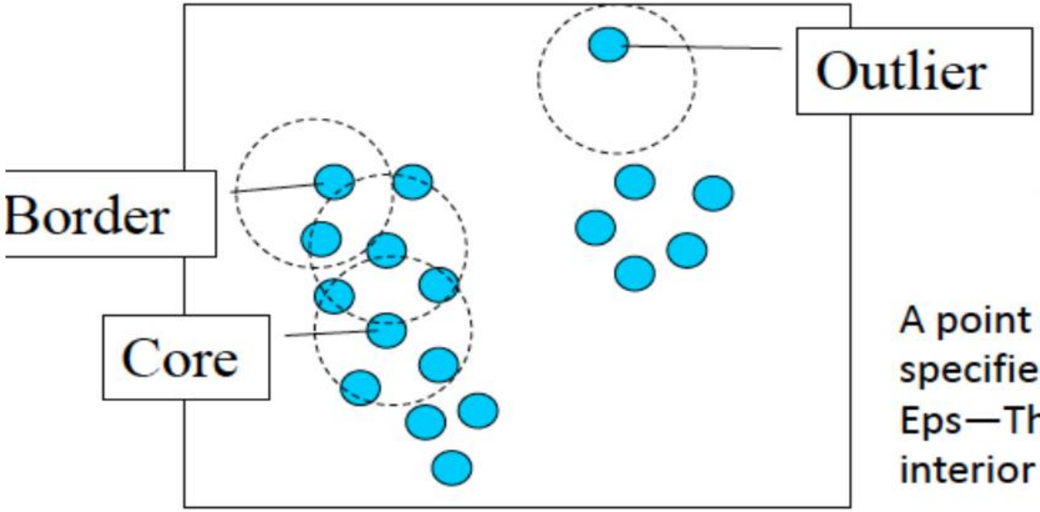
Density-Based Clustering By Joerg Sander, https://doi.org/10.1007/978-0-387-30164-8_211

Density Clustering

- Locates regions of high density that are separated from one another by regions of low density.

- Main principles
  - Maximum radius of the neighbourhood (Eps)
  - Minimum number of points (MinPts) in an Eps neighbourhood of a point: $N_{Eps}(p) : \{q \in D \text{ s.t. } dist(p, q) \leq Eps\}$
  - Idea: Density of neighbourhood must exceed some threshold.
  - Shape of a neighbourhood depends on dist function.

Core, Border and Noise/Outlier



ε = 1unit, MinPts = 5

Given ε and *MinPts*, categorize the objects into three exclusive groups.

A point is a core point if it has more than a specified number of points (MinPts) within Eps—These are points that are at the interior of a cluster.

A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

Density Clustering

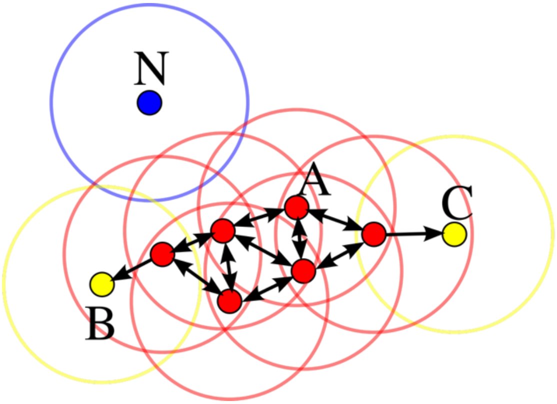
+ Discovers clusters of arbitrary shapes.
+ Handles noise.
+ Needs density parameters as termination condition.

- Cannot handle varying densities.
- Sensitive to parameters.
- Sampling affects density measures.

DBSCAN

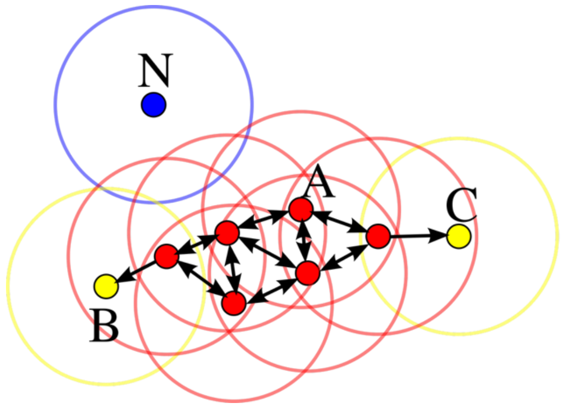
- DBSCAN - Density-Based Spatial Clustering of Applications with Noise.
  - Two parameters Eps and Minpts are set according to experience !!
  - Finds core samples of high density and expands clusters from them.
  - Good for data which contains clusters of similar density.
  - Worst case memory complexity is $O(n^2)$ [when eps is large and minPts is small].

minPts = 4. Red points are core points, because the area surrounding these points in an ε radius contain at least 4 points (including the point itself). Yellow are not core points. Point N is a noise point.



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.

DBSCAN Algorithm



- Defines clusters as continuous regions of high density.
- How DBSCAN works?
  1. For each instance, algorithm counts how many instances are located within a small distance ε (epsilon) from it. Region is called instance's ε - neighborhood.
  2. If an instance has at least min_samples instances in its ε-neighborhood, then it is considered a core instance.
  3. All instances in the neighborhood of a core instance belong to same cluster. Neighborhood may include other core instances.
  4. Any instance that is not a core instance and does not have one in its neighborhood is considered an anomaly.

It works well if all the clusters are dense enough!!

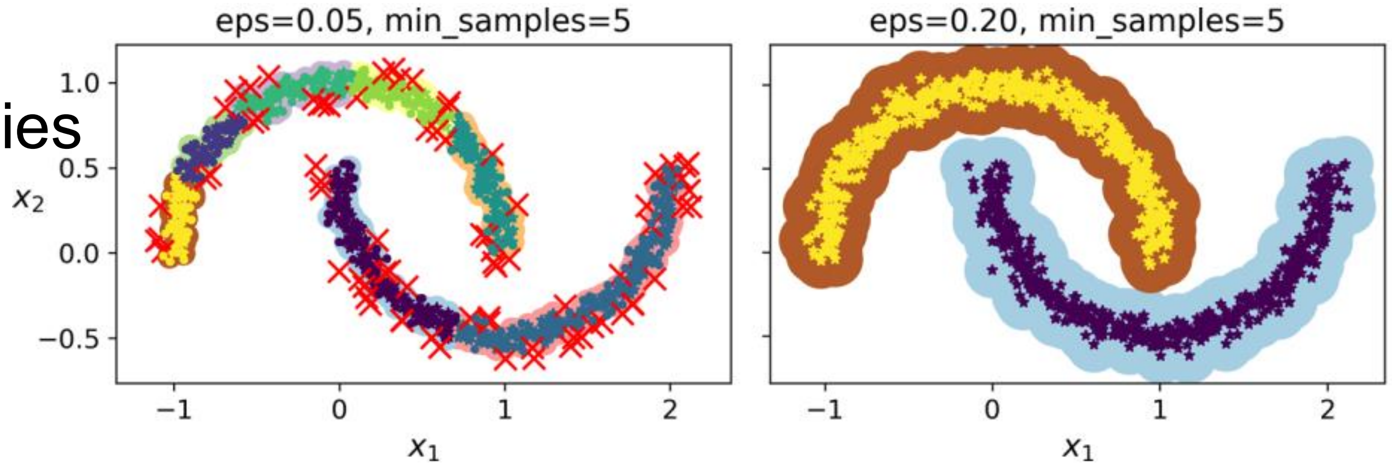
Example

```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_moons
```
<--moons dataset

```
X, y = make_moons(n_samples=1000, noise=0.05)
dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)
```

- Showing anomalies & 7 different clusters.

Better



- Widen each instance's neighborhood by increasing eps to 0.2

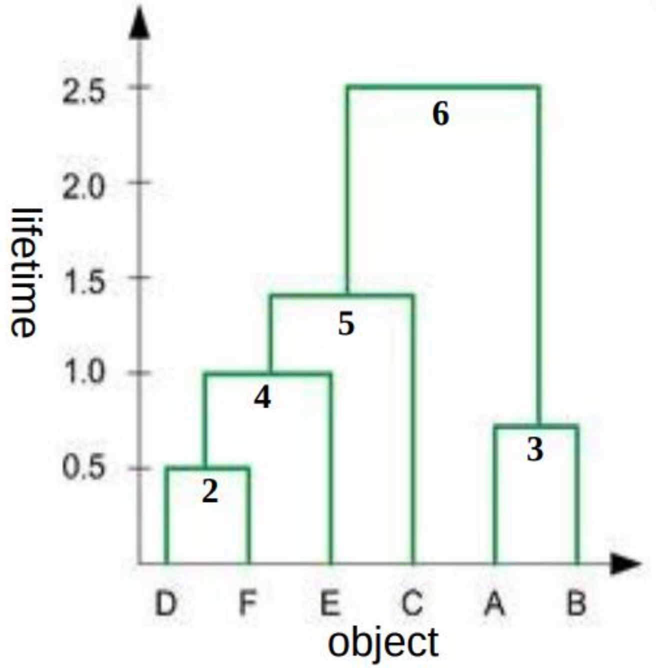
Hierarchical Cluster

Hierarchical Cluster

- Hierarchical algorithm is a sequential clustering algorithm.
- Use distance matrix to construct a tree of clusters (dendrogram).
- Hierarchical representation without need of knowing number of clusters.
- You can set termination condition with known # of clusters.

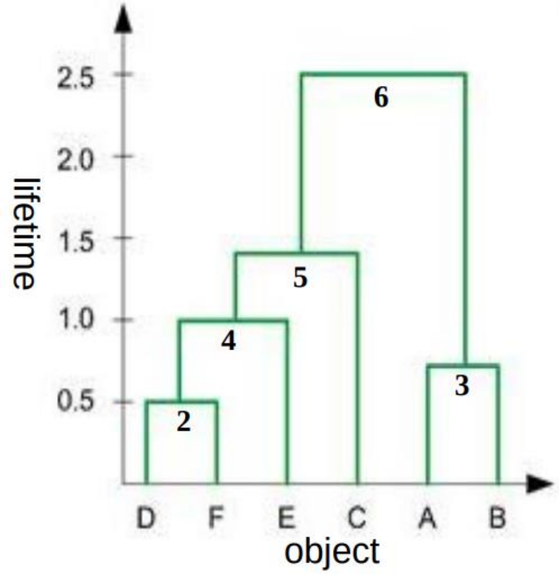
Hierarchal Clustering

- Dendrogram tree representation:
  - At start, we have 6 clusters: A, B, C, D, E and F.
  - We merge clusters D and F into cluster (D, F) at distance 0.50.
  - We merge cluster A and cluster B into (A, B) at distance 0.71



Hierarchal Clustering

- We merge clusters E and (D, F) into ((D, F), E) at distance 1.00
- We merge clusters ((D, F), E) and C into (((D, F), E), C) at distance 1.41
- We merge clusters (((D, F), E), C) and (A, B) into ((((D, F), E), C), (A, B)) at distance 2.50
- Last cluster contain all objects, thus conclude computation.



Hierarchal Clustering: Lifetime

- Lifetime: Distance between that a cluster is created and that it disappears (merges with other clusters during clustering).
- Example: lifetime of A, B, C, D, E and F are 0.71, 0.71, 1.41, 0.50, 1.00 and 0.50, respectively.
- Then lifetime of (A, B) is 2.50 – 0.71 = 1.79