University of Pittsburgh

ECE 2195: Special Topics – Computers Machine Learning

Unsupervised Learning: Clustering

Mai Abdelhakim, PhD

Assistant Professor of ECE

Swanson School of Engineering

University of Pittsburgh

maia@pitt.edu

This Unit

~10 minutes for presentation
Final: Dec 14
Only 1 needs to upload

- Unsupervised Learning
  - Clustering
    - K-Means
    - Hierarchical (Agglomerative)
    - DBSCAN

Unsupervised Learning

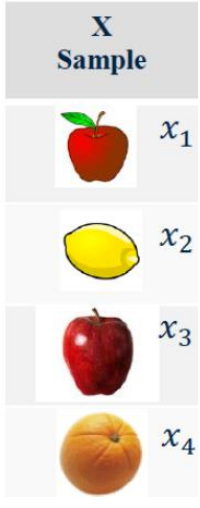
- With unsupervised learning:

  no more labels

  - **Can we find subgroups** among the observations?
    - Are there interesting patterns?

- **Hard to assess the performance**
  - How to do that without ground truth labels?
  - More challenging, and subjective
  - Needs manual assessment

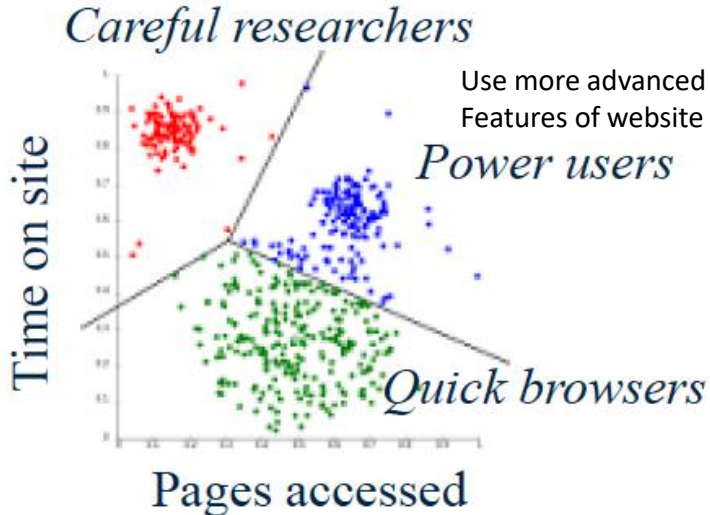
  Look into what clusters are obtained and assess from there



Unsupervised Learning

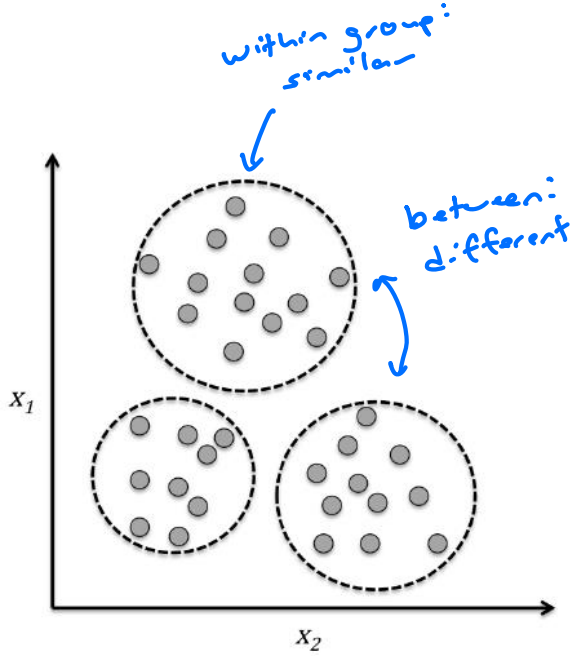
Examples:

- Market Segmentation: Identify subgroups of people who are more receptive to particular advertisement and are likely to purchase a particular product
  - Features could be: household income, occupation, …
- E-commerce: Clustering could also be performed based on browsing activity
  - **Tailor website for each group**



Clustering is based on how similar observations are to one another

- Find **distinct groups** in the data
  - Observation **within a group** are quite **similar**
  - Observations in **different groups** are different from each other



- Similar or different are based on the domain/application and data being studied

- For n observations, what is the minimum and maximum number of clusters?

minimum = 1

maximum = n

Well-known Algorithms

- Algorithms:
  - K means
  - Agglomerative Clustering
  - DBSCAN: Density Based Spatial Clustering of Applications with Noise

K-Means

- Partition the observations into **pre-specified** number of clusters (equal to K)
- Data is partitioned into K clusters, each observation is assigned to one of these clusters

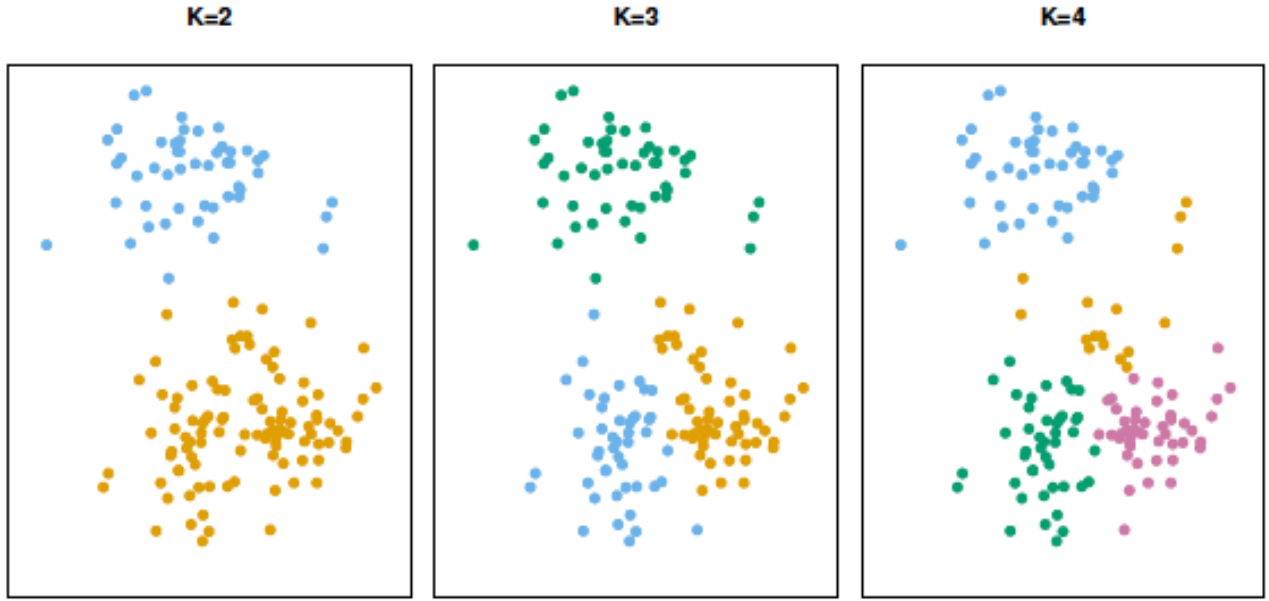
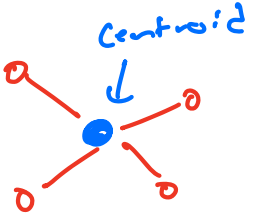


Figure: k=2,3,4 clusters each cluster has different color

Within cluster variation should be small

- **Idea**: find clusters where the **within-cluster variations** is as small as possible
  - Within-cluster variation (WCV) is the amount by which the observations within a cluster differ from each other
- Suppose, K clusters represented by set: $C_1, C_2, \ldots, C_K$
  - Each set contains the indices of observations within the cluster
- **Within-cluster variation** of C_k can be expressed as



Squared Euclidean distance between two samples

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

For every pair of observations, what is the Euclidean distance between them

$|C_k|$ Number of observations in cluster C_k

i and i' are two indicies of two observations in cluster C_k, each has p features

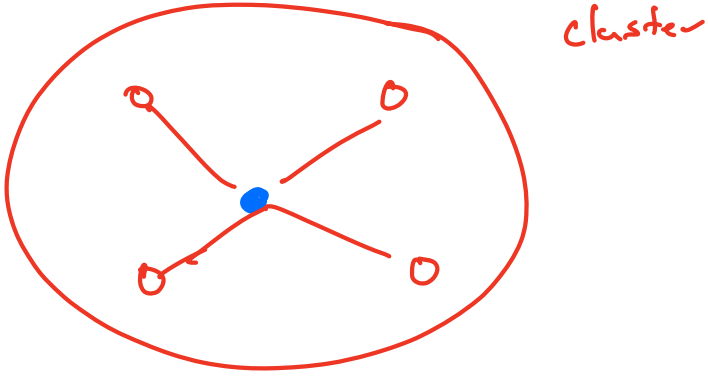
Similarilarity can be measured using distance from cluster centroids

- Within-cluster variation can also be measured by how far observations are form their **cluster centroids**

Squared Euclidean distance between sample and cluster centroid

$$\sum_{i \in C_k} \|x_i - \mu_k\|^2$$

- μ_k: is centroid of a cluster k; it's a vector of length equal to number of features (p)



Define objective function

- Define responsibilities: r_{ik} cluster k is responsible of generating sample i
  - r_{ik}=1 if sample i belongs to cluster k, r_{ik}= 0 otherwise
  - $r_{ik} \in \{0,1\}$
  - $\sum_k r_{ik} = 1$

- Example: if there are 5 data points, & 3 clusters
- $[r_{ik}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

cluster 1 : (1, 3)
cluster 2 : (2, 5)
cluster 3 : (4)

- Cost function $J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|^2$
  - $\mu_k = \frac{\sum_{i=1}^{n} r_{ik} x_i}{\sum_{i=1}^{n} r_{ik}}$

all data points
centroid
all clusters

K-Means problem formulation – Hard to solve

- **Optimization problem**: finding clustering (μ_k for all k) that **minimize the within-cluster variation**

want to get these

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|^2$$

k is given

μ_k depends on r_{ik}, but r_{ik} depends on μ_k → No closed form solution

- Solution is difficult – no-closed form solution

- A simplified algorithm can find sub-optimum solution

Find Solutions via Expectation Maximization Algorithm

- The previous problem suggests an iterative scheme to find the solution

- **Expectation maximization** is an iterative approach that can provide solutions to such problem – via following steps
  1. Make initial guesses for the paratmers (e.g. Centroids) random guesses
  2. Alternate between the following two steps
     - E-step (Expectation): Find the hidden structure via the current parameters
     - M-step (maximization): Update/re-estimate the parameters by observing hidden structure in E-step

E-step: Find clusters using centroids

M-step: Change centroids (take mean of clusters)

K-Means Clustering Algorithm

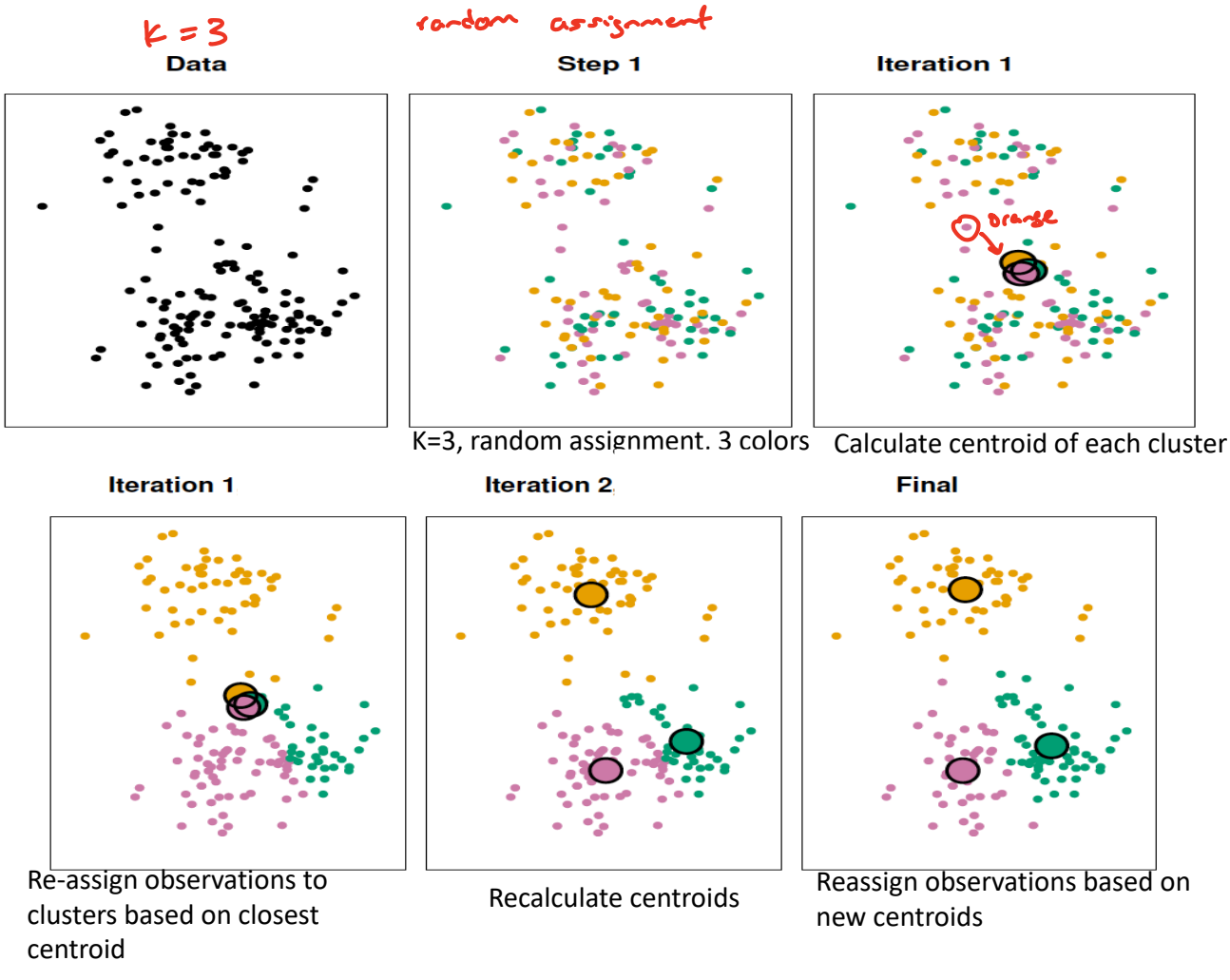
Set value of K!

- Step 1: Random Initialization of clusters – Get the initial centroids
  - randomly select centroids from training data
  - Other implementation: Randomly assign each observation to a cluster (1,2,… K), then find the centroids based on this random assignment

- Step 2
  a) (E-step): Assign each point to a cluster based on the nearest centroid

  b) (M-step): Re-evaluate centroids
     - Compute the cluster center (centroid) == mean of the observations assigned to that cluster.
     - Assign each observation to the cluster with closest centroid

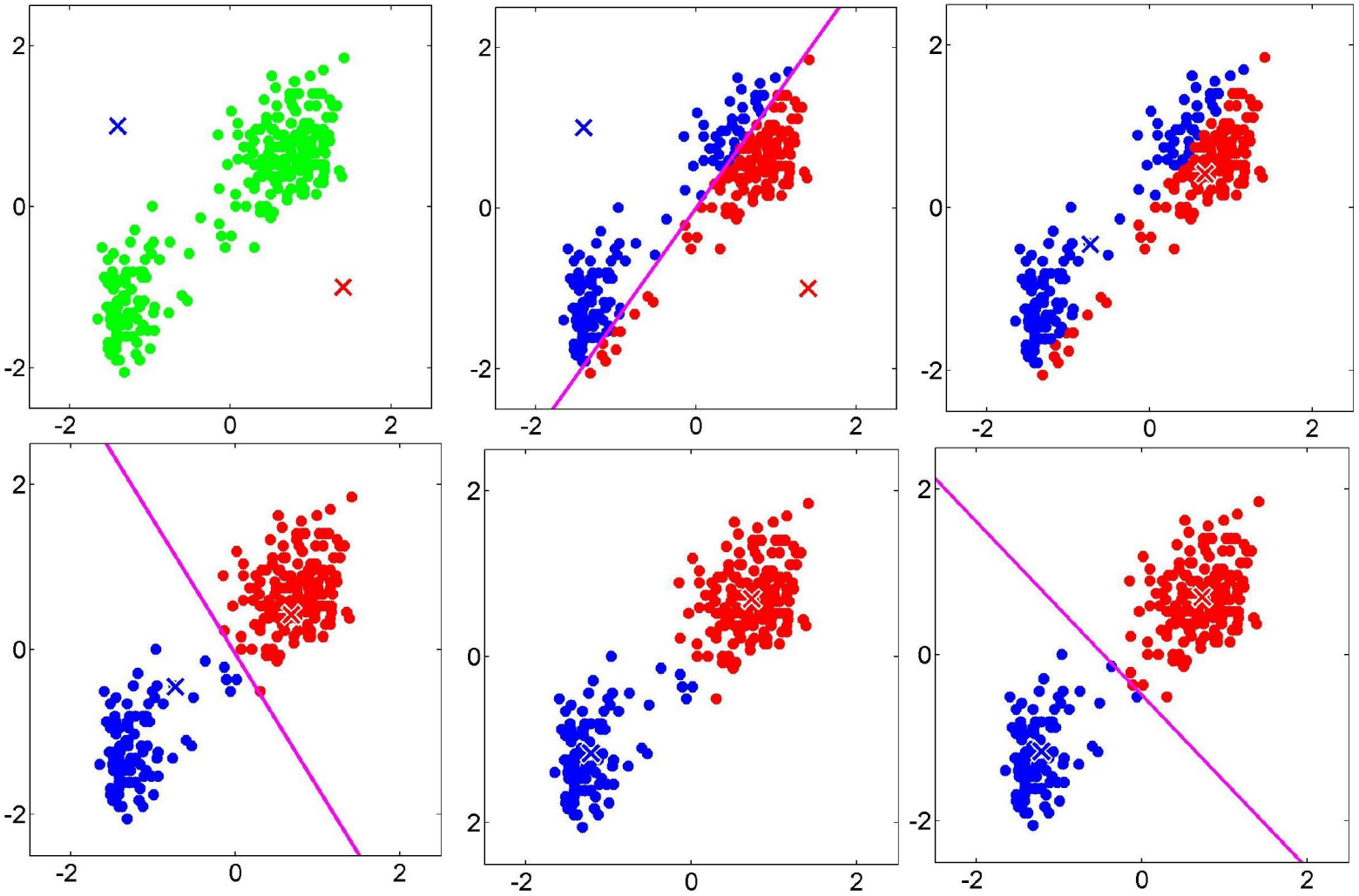
- Iterate over a and b until cluster assignment stops changing



We can improve initialization step!

Another example with different initialization of centeroids

$k = 2$



K-Means, Initialization

- The algorithm may converge to a **local minimum**
- The result **depends on the initial random assignment**
  - Run algorithm multiple times with different initial configuration, then select solution with **smallest within-cluster variations**

Example: Different Initial Setting Results in Different Clustering

K=3, each cluster is represented by different color

Each figure represents an output of K-Means clustering with **different random assignment** of observations at the beginning

Different local optima are obtained

Best solution results in objective (lowest within-cluster variation) of 235.8

minimizing objective function

Variants of K-Means

- **K-means ++:** Chooses the **initial centroids as far as possible** from each other

- **Soft K-means:**
  - Typical K-means is a hard clustering, where each sample is assigned to one cluster
  - **Soft clustering uses probabilities of membership of each sample to one of the clusters**
    Helps when there are outliers
  - Example: Hard cluserting, with K=3 clusters, a sample (i) in 2nd cluster can be represented by a

    weight vector (responsibility) $r_i = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$,

    In soft clustering, responsibility contains probabilities that depend on how close/far the sample is from the centroids,

    e.g. $r_i = \begin{bmatrix} 0.1 \\ 0.85 \\ 0.05 \end{bmatrix}$,

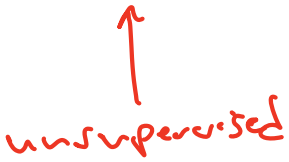
    $$\mu_k = \frac{\sum_{i=1}^{n} r_{ik} x_i}{\sum_{i=1}^{n} r_{ik}}$$

K-means in Python

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

```
from sklearn.cluster import Kmeans
kmeans = KMeans(n_clusters=NumberOfClusters, random_state=0).fit(X)
```

- To use Kmeans++: set *init='k-means++'*



- Use .predict to predict cluster of an observation
- Attributes:
  - **cluster_centers_**
  - **labels_:** has cluster assignment of each point
  - **inertia_**: is the sum of squared distances of samples to their closest cluster center

If labels are know we can easily evalute!

Evaluation: **if labels are known**, we can use metrics such as adjusted_rand_score

- adjusted_rand_score output 0 for random clustering and 1 for perfect clustering

```
from sklearn.metrics.cluster import adjusted_rand_score

kmeans_assignment = KMeans(n_clusters=2, random_state=0).fit_predict(X_train_pca)

print("Clustering score:", adjusted_rand_score(kmeans_assignment,Y_train))
```

Clustering doesn't care about actual label - only the grouping

Agglomerative (Hierarchical) Clustering

- Disadvantage of K-means: need to specify number of clusters
  - We may not know how many clusters in advance

- Hierarchical Clustering: Produce a **dendrogram**, which is a tree-based representation of data that shows the **clustering obtained for each possible number of clusters**

- **Agglomerative clustering is common type of hierarchical clustering**
  - **bottom-up: tree is built starting from leaves**

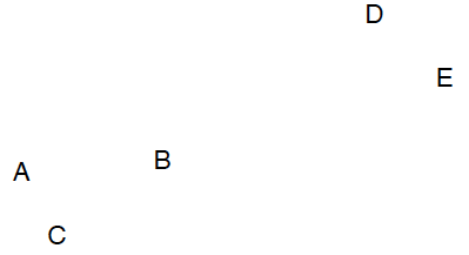
Example: Hierarchical Clustering with n=5 observations

$\frac{n(n-1)}{2}$ pair-wise comparisons

Calculate all n(n-1)/2 pair-wise dissimilarities, and merge the least dissimilar (most similar)

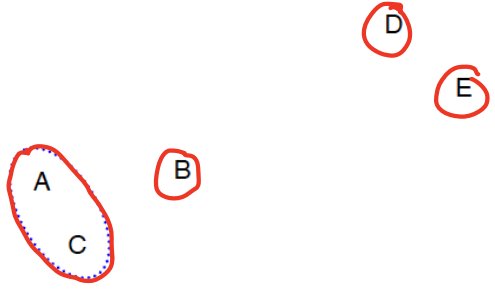
(1) n = 5 clusters

Start with each point in its own cluster



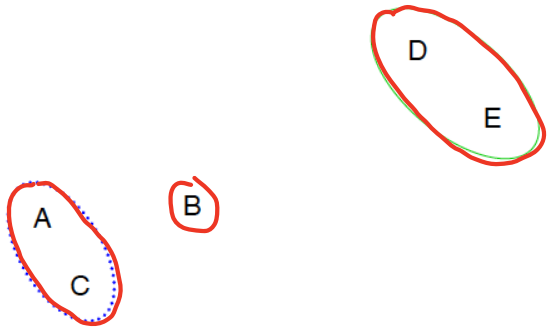
(2) n-1 = 4 clusters

Identify two closest clusters and merge them



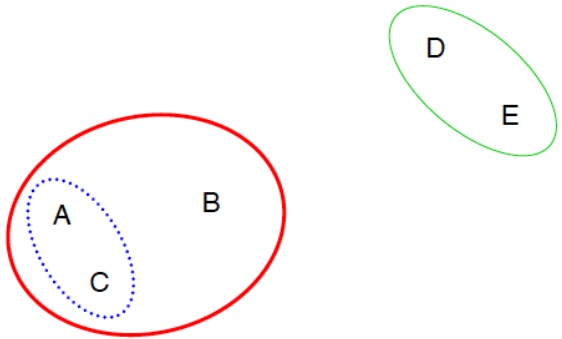
(3) n-2 = 3 clusters

Identify two closest clusters and merge them



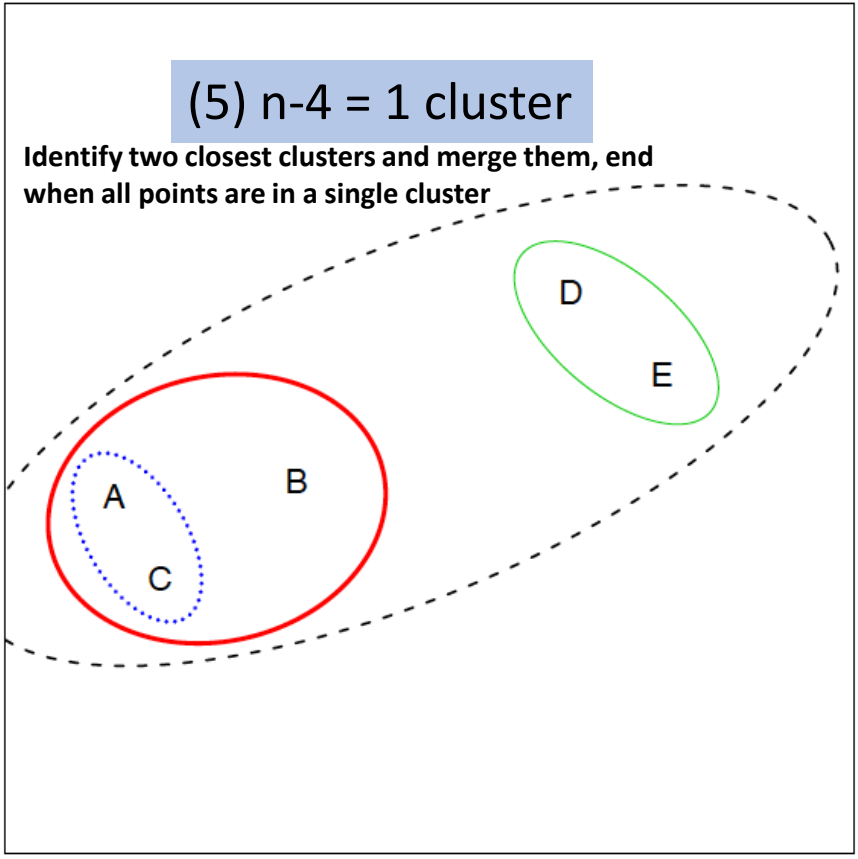
(4) n-3 = 2 clusters

Identify two closest clusters and merge them



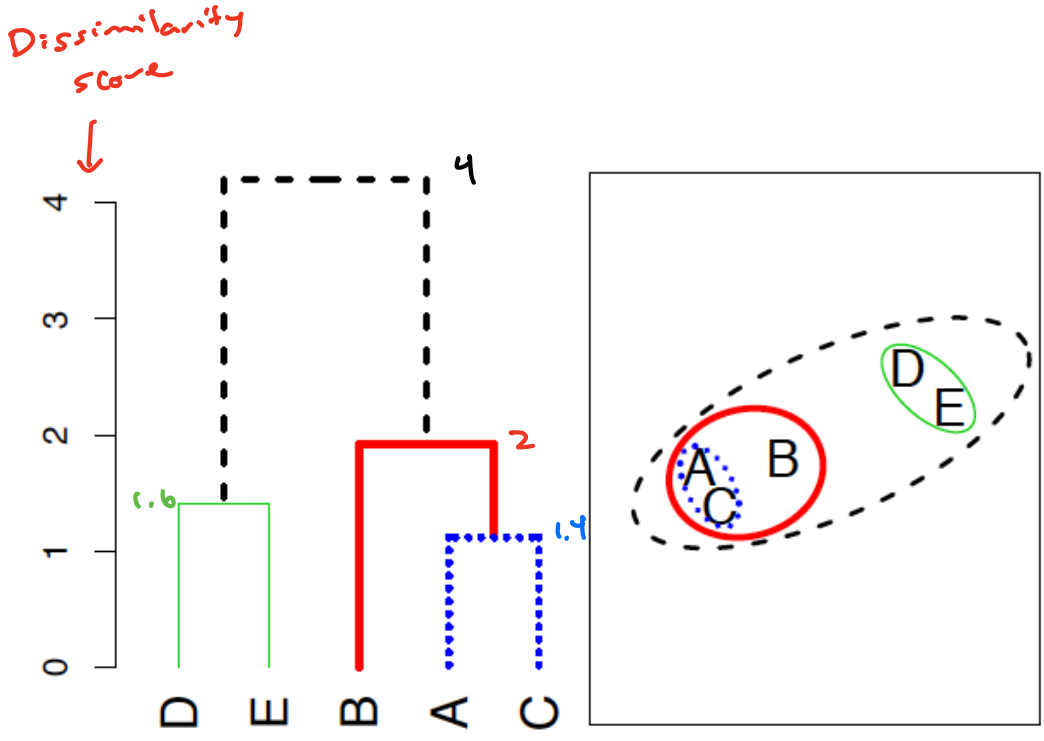
- Hierarchical clustering provides information about clustering obtained with different number of clusters

- Number of cluster ranges from: n to 1



Hierarchical Clustering: Dendrogram

- Dendrogram: read from bottom to up
- Dissimilarity can be Euclidian distance, correlation,..
- **Height of a branch** represents **how far, i.e., dissimilar,** the merged clusters are
  - Observations that fuse at bottom of the tree are more similar to those merged at the top



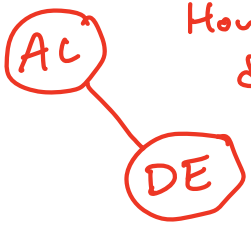
Dendrogram

can set threshold for dissimilarity scores

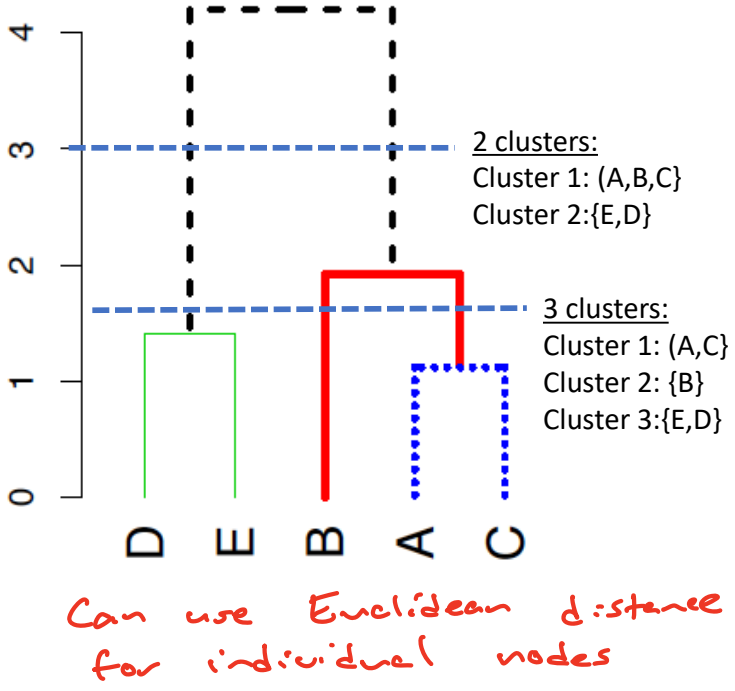
Using the Dendrogram

- Cut dendrogram at a certain height to get clusters

- In previous example:
  Cut at height 3 ➔ get 2 clusters
  Cut at height 1.6 ➔ get 3 clusters



How to measure dissimilarity?



Can use Euclidean distance for individual nodes

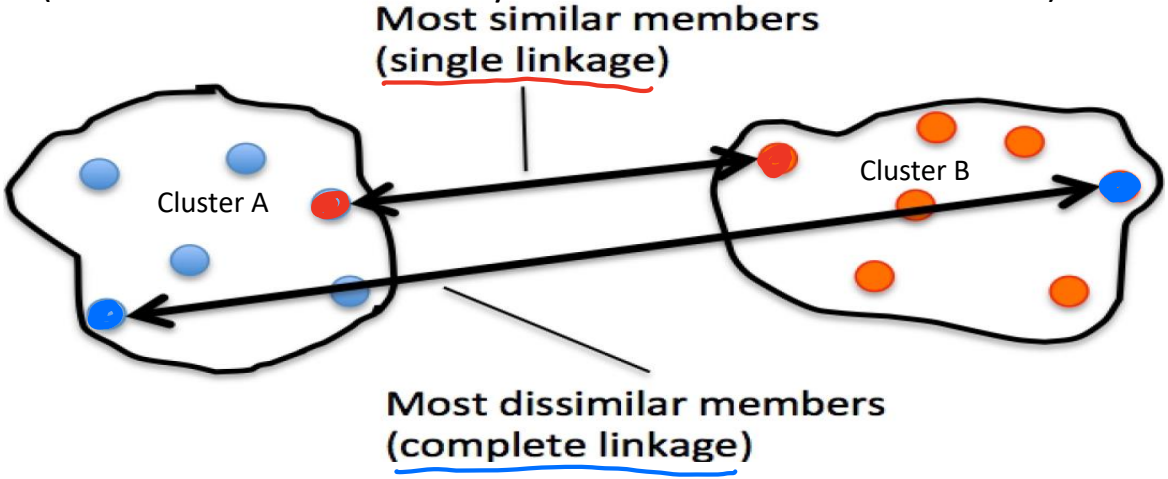
How to Measure Dissimilarity Between Clusters?

- We need to compute the dissimilarity
  - To know which clusters to be merged and at what height
  - **Merge clusters** that are **least dissimilar (most similar)**

- **Linkage**: defines the **dissimilarity between two groups/clusters**
- **Types of linkage are:**
  - **Complete**
  - **Single**
  - **Average**
  - **Centroid**
  - **Ward**
    - Ward merges clusters that lead to the minimum increase of the total within-cluster variation.

Types of Linkage

Single linkage (minimal inter-cluster dissimilarity): compute all pairwise dissimilarity between samples in cluster A and B, and record **smallest** of them (This will be the dissimilarity between cluster A and Cluster B)



Complete linkage (maximal inter-cluster dissimilarity): compute all pairwise dissimilarity between samples in cluster A and B, and **largest** of them will be the dissimilarity between the two groups

Types of Linkage ... Cont.

- **Average linkage:** compute all pairwise dissimilarity between samples in cluster A and B, and record **average** of these dissimilarities

- **Centroid linkage:** compute the dissimilarity between centroid for cluster A and cluster B. Centroid for cluster A - average of all points in cluster A

Dendrogram of Hierarchical Clustering depends on the type of linkage used

Example: Single linkage

DISSIMILARITY between observations	Observation 1	Observation 2	Observation 3	Observation 4	Observation 5
Observation 1	0				
Observation 2	0.1	0			
Observation 3	0.5	0.25	0		
Observation 4	0.8	0.2	0.15	0	
Observation 5	0.3	0.75	0.7	0.4	0

smaller of D_{13} , D_{23}

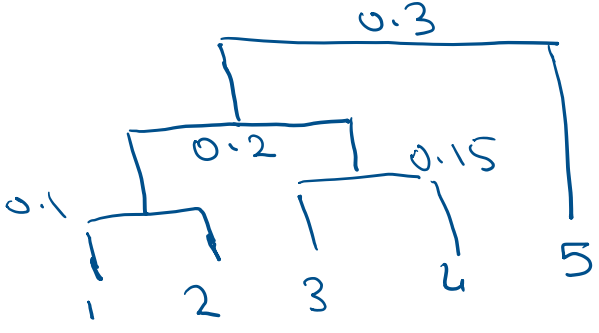
	1,2	3	4	5
1, 2	0			
3	0.25	0		
4	0.2	0.15	0	
5	0.3	0.7	0.4	0

	1,2	3,4	5
1, 2	0		
3,4	0.2	0	
5	0.3	0.4	0

	1,2,3,4	5
1, 2,3,4	0	
5	0.3	0

Example: Single linkage

DISSIMILARITY between observations	Observation 1	Observation 2	Observation 3	Observation 4	Observation 5
Observation 1	0				
Observation 2	0.1	0			
Observation 3	0.5	0.25	0		
Observation 4	0.8	0.2	0.15	0	
Observation 5	0.3	0.75	0.7	0.4	0



Example: Single linkage and Complete linkage

DISSIMILARITY between observations	Observation 1	Observation 2	Observation 3	Observation 4	Observation 5
Observation 1	0				
Observation 2	0.1	0			
Observation 3	0.5	0.25	0		
Observation 4	0.8	0.2	0.15	0	
Observation 5	0.3	0.75	0.7	0.4	0

Single linkage:
Step 1: G12=Group (1,2), dissimilarity 0.1
Step 2: G34= Group (3,4), dissimilarity 0.15
Step 3: G1234=Group (G12, G34), dissimilarity 0.2
Step 5: Group (G1234 , 5) dissimilarity 0.3

Complete linkage:
Step 1: G12=Group (1,2), dissimilarity 0.1
Step 2: G34=Group (3,4), dissimilarity 0.15
Step 3: Group (G34, 5), dissimilarity 0.7
Step 4: Group all, max height of dendrogram 0.8

Scaling

- When features have different units (cm & km), scaling helps
- When features are of same units
  - Scaling ensures that **features are given equal importance**
  - However, this depends on the application
    - Choose most interpretable solution

DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN)

- Reference: Introduction to Machine Learning with Python, Chapter 3

- DBSCAN can **capture clusters of complex shapes and without identifying number of clusters a priori**

- **Idea: clusters form dense regions of data followed by regions that are relatively empty**

DBSCAN algorithm

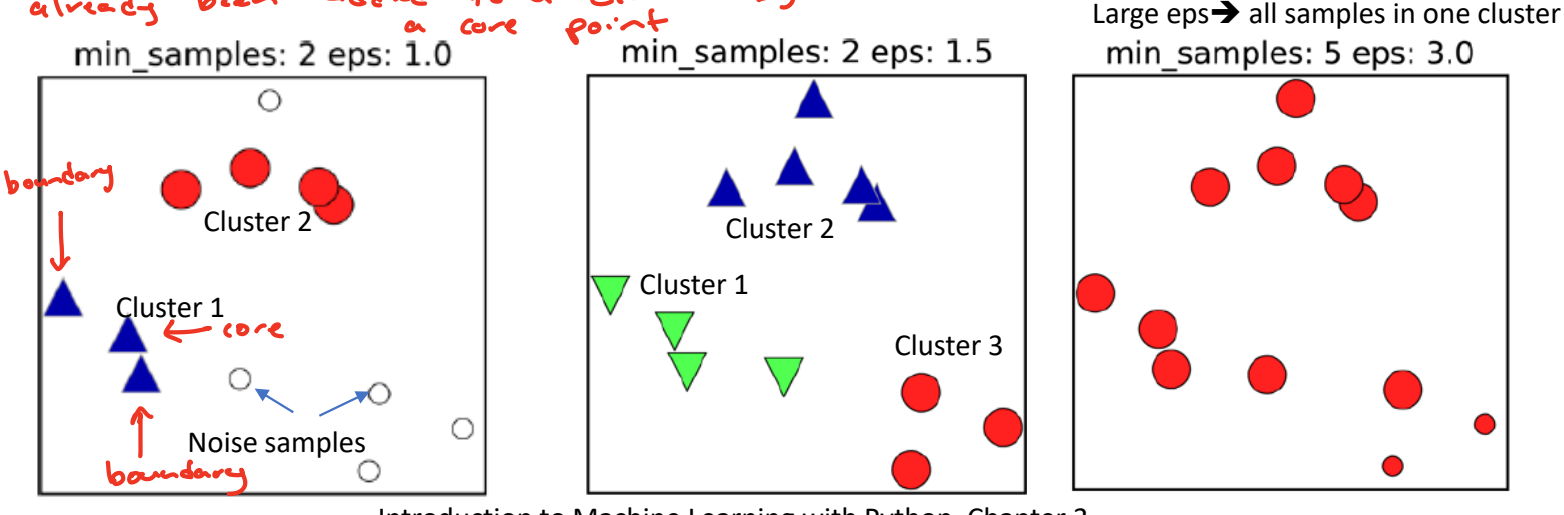
Core observation
Boundary observation
Noise

- Two parameters that DBSCAN depends on: **min_samples**, **eps**
  - Min_samples is minimum number of samples around a core cluster point that are within a distance eps

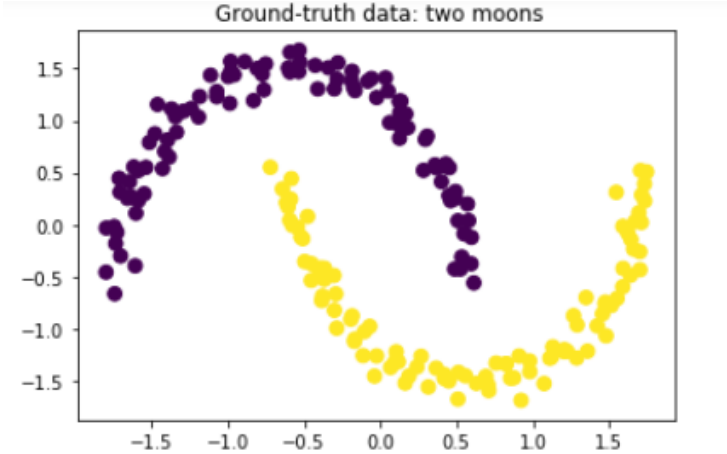
- **Identify points** that are in "crowded"/"**dense**" regions in the feature space
  - These observations are called **core points**
  - Starts with an arbitrary point, check if it is a **core point**
    - If there are "**min_samples**" number of observations within a distance "**eps**" form that point, then this is identified as a core point
      - points within a distance eps are in the same cluster
  - A cluster grows until there is no more core samples within distance "eps", then new points are visited to create other clusters.

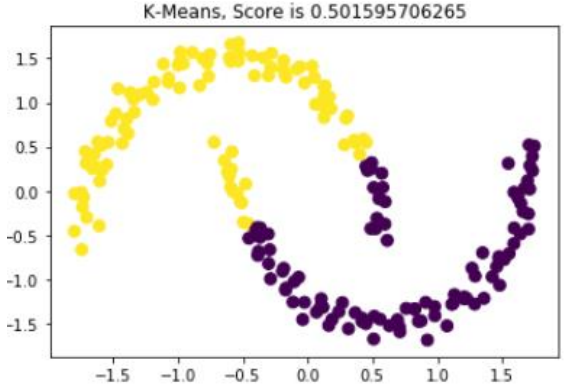
- **Noise**: observation points that do not belong to any cluster
  - Have less than **min_samples** within **eps**
- Points are either: **core, boundary, noise**
- **Result of clustering depends on the parameters min_samples and eps**
  - **When eps increase ➔ less number of clusters**
  - **min_samples** determines the smallest cluster size

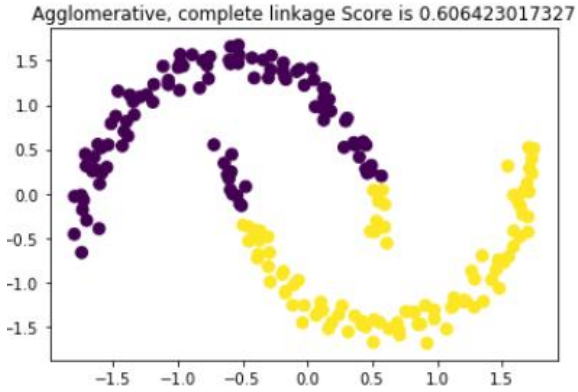
Boundary instead of noise because it's already been added to a cluster by a core point

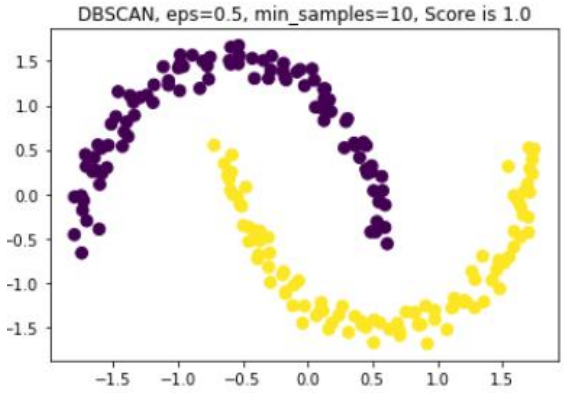


Example


Ground-truth data: two moons


K-Means, Score is 0.501595706265


Agglomerative, complete linkage Score is 0.606423017327


DBSCAN, eps=0.5, min_samples=10, Score is 1.0

Practical Issues: Parameter Setting

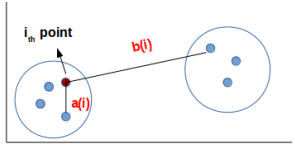
- What are the best parameters to use in a clustering algorithm?
  - How many clusters to choose if K-means is used?
  - What type of dissimilarity & linkage to use in hierarchical clustering?
  - What is eps and min_samples for DBSCAN?
- Scale features or not?
- How to evaluate?

Evaluation of Clustering

- When labels are known (typically not the case), results of clustering can be compared:
  - Example: adjusted_rand_score, Normalized Mutual Information: value of maximum 1 means perfect clustering

- There is **no single agreed-upon performance metric** for unsupervised learning

- Some metrics measure compactness of clustering
  - Ex: within cluster variations or silhouette score
    - Silhouette score: Difference between mean intra-cluster variation (a) and the mean nearest cluster distance (b) ➔ b-a/max(b,a)



- The **only way to know whether clustering worked well is to have a domain expert analyze it manually**