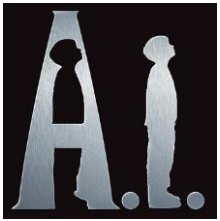


UCCD2063

Artificial Intelligence Techniques

Unit 08:

Unsupervised Learning:

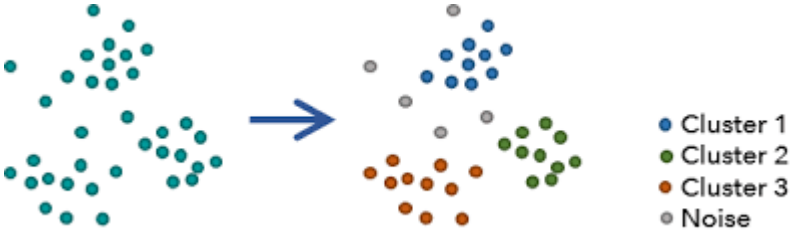
Clustering

Outline

- **Unsupervised Learning - Clustering**
- **Anomaly Detection**

What is Clustering?

- Clustering: the process of grouping data samples that are ***similar*** in some way into classes of similar objects
- Clustering is a form of ***unsupervised learning*** – class labels are not known in advance (e.g. *y* is not provided)
- It is a method of ***data exploration*** – a way of looking for patterns or structure in the data that are of interest



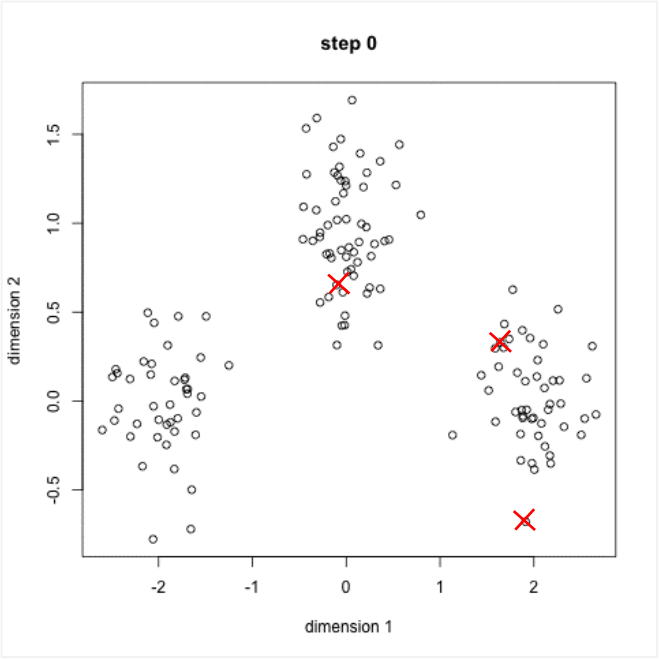
Popular Clustering Algorithms

- **K-means** clustering – trying to separate samples in *k* groups of equal variance.

- **Agglomerative** clustering – a bottom up approach which each observation starts in its own cluster, and clusters are successively merged together.

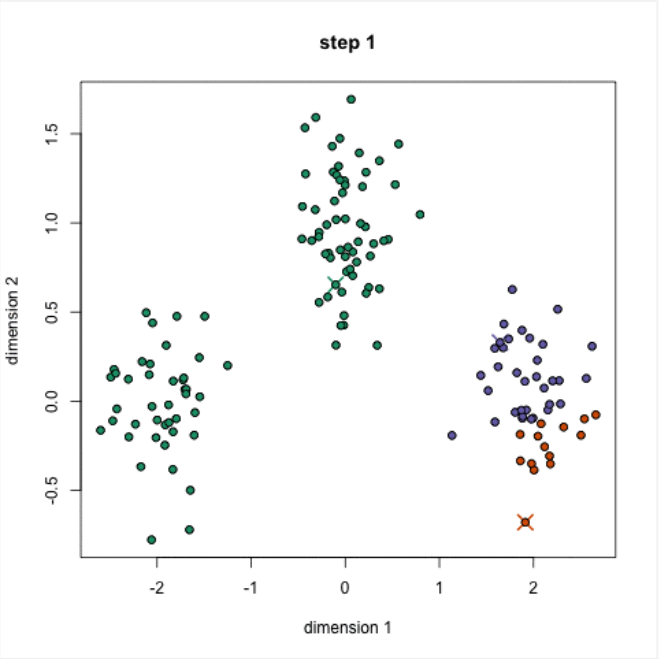
- **DBSCAN** – views clusters as areas of high density separated by areas of low density, clusters found by DBSCAN can be any shape.

K-means Clustering



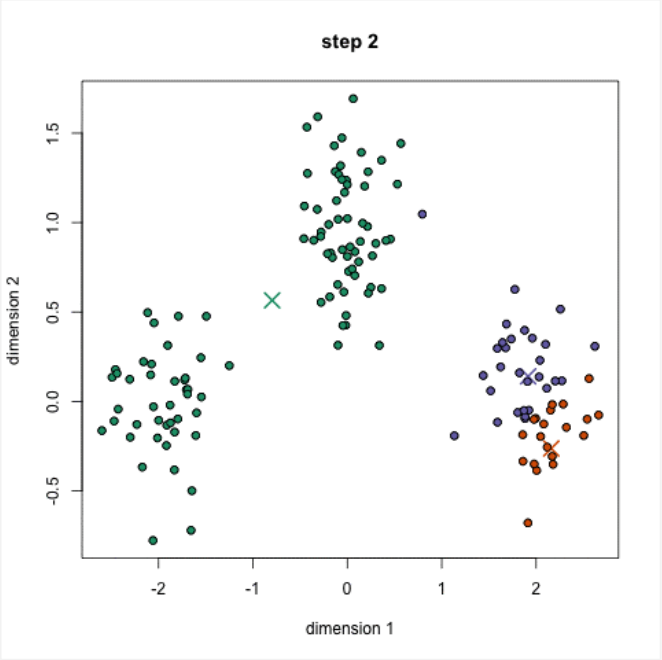
1. Randomly initialize *k* (e.g. 3) cluster centers
2. Assign each data point to the closest cluster center (using some distance measure)
3. Re-compute cluster centers (mean of data points in cluster)
4. Repeat the process and stop when there are no new re-assignments

K-means Clustering



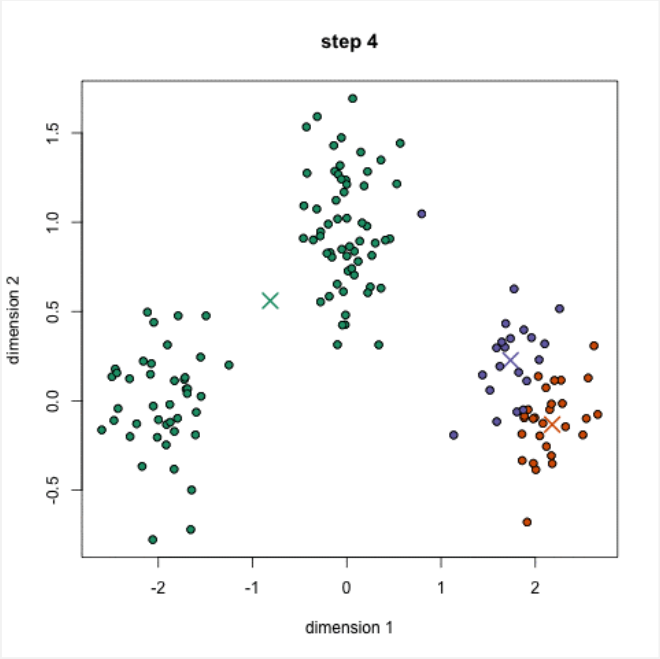
1. Randomly initialize *k* cluster centers
2. Assign each data point to the closest cluster center (using some distance measure)
3. Re-compute cluster centers (mean of data points in cluster)
4. Repeat the process and stop when there are no new re-assignments

K-means Clustering



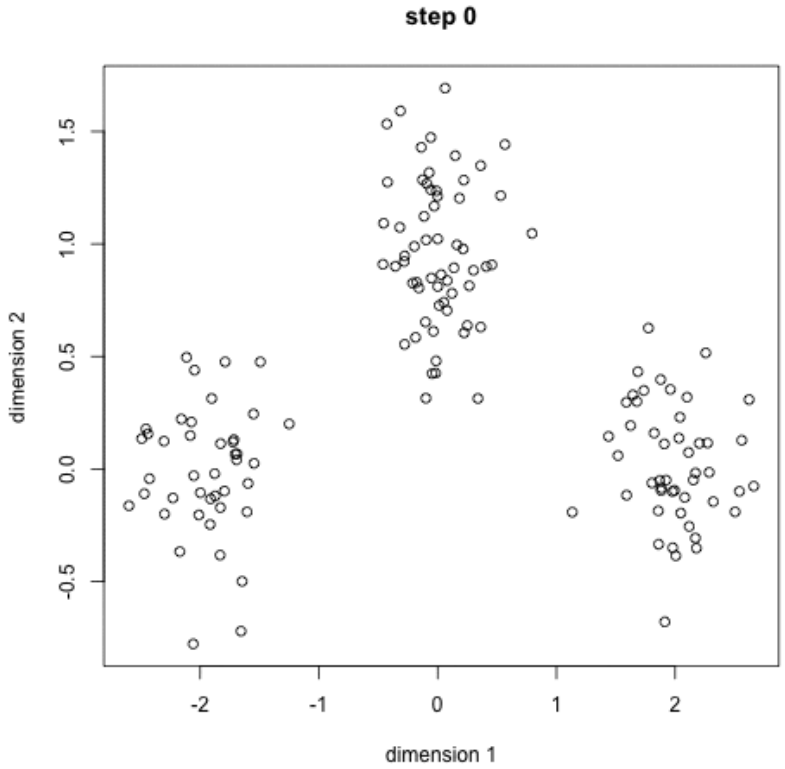
1. Randomly initialize *k* cluster centers
2. Assign each data point to the closest cluster center (using some distance measure)
3. Re-compute cluster centers (mean of data points in cluster)
4. Repeat the process and stop when there are no new re-assignments

K-means Clustering



1. Randomly initialize *k* cluster centers
2. Assign each data point to the closest cluster center (using some distance measure)
3. Re-compute cluster centers (mean of data points in cluster)
4. Repeat the process and stop when there are no new re-assignments

K-means Clustering - animation

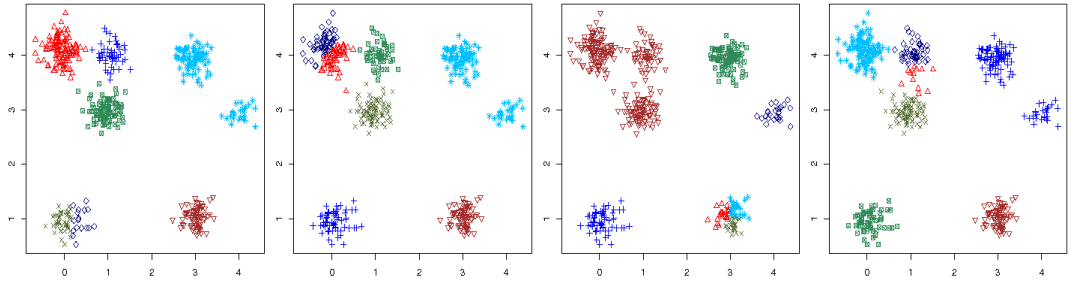


K-means Clustering

Problems with k-means:

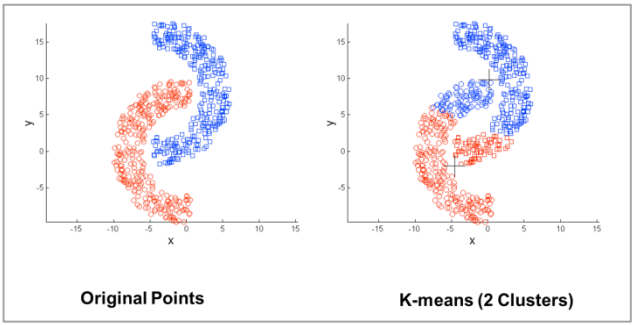
- Have to select *k* centers, can be difficult
- Start with a random choice of cluster centers and may yield different clustering results on different runs
- Assume clusters are convex shaped, cannot deal with complex clusters

K-means Clustering Problems



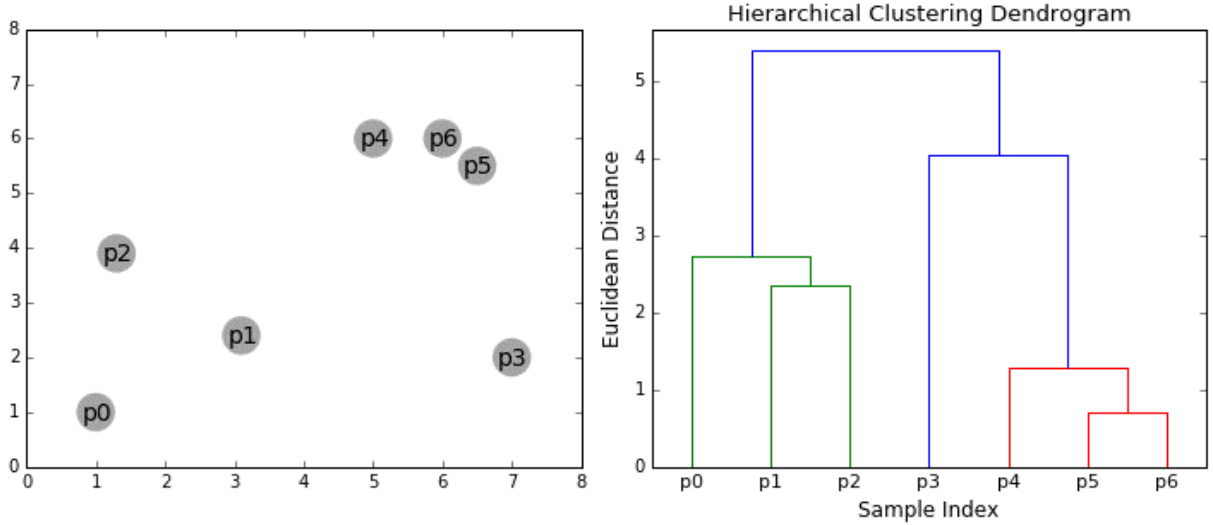
(a) Different starting cluster centers

(b) Incorrect number of clusters



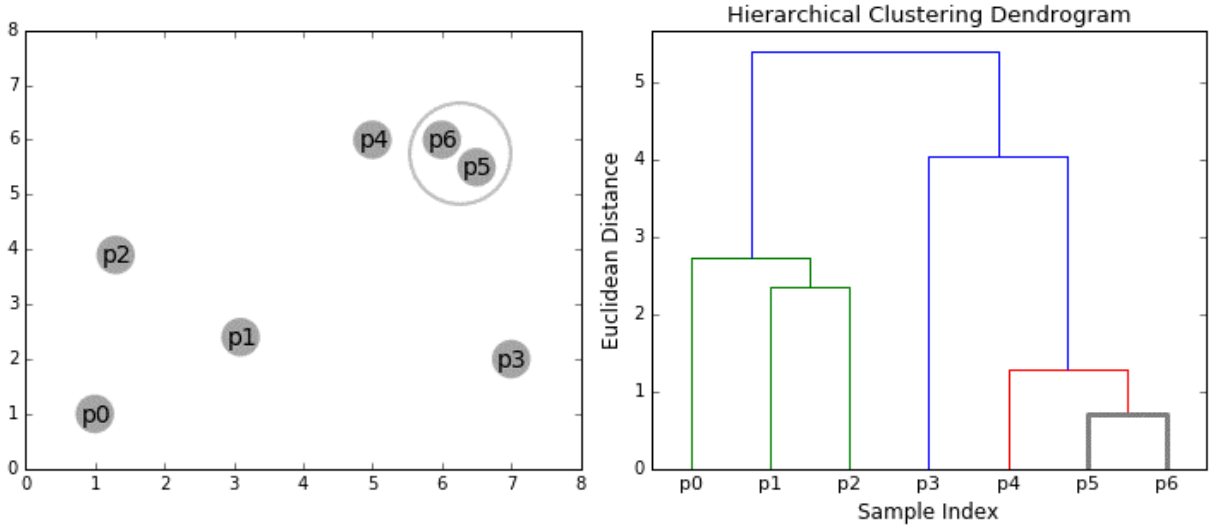
(c) Non-convex clusters

Agglomerative Clustering



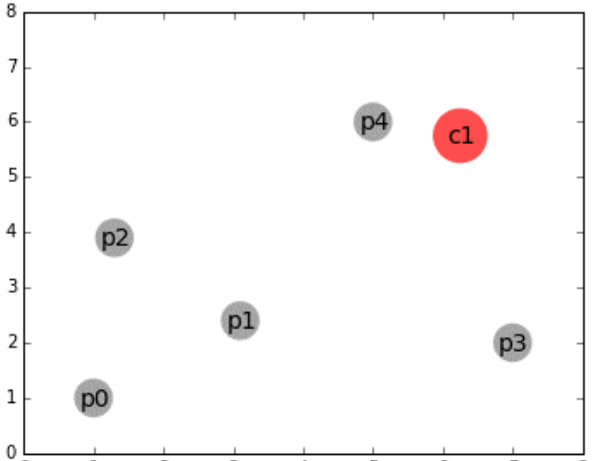
- Each points are initialized as its own cluster
- Compute the *linkage* between clusters
- Merge the two clusters with smallest linkage
- Repeat the process until desirable number of clusters obtained

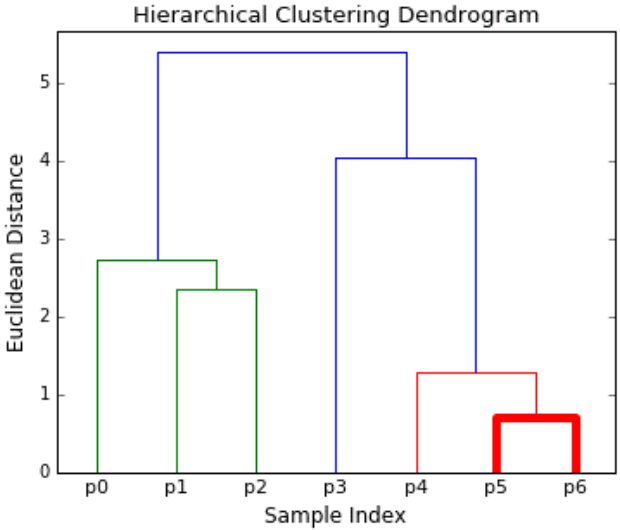
Agglomerative Clustering



- Each points are initialized as its own cluster
- Compute the *linkage* between clusters
- Merge the two clusters with smallest linkage
- Repeat the process until desirable number of clusters obtained

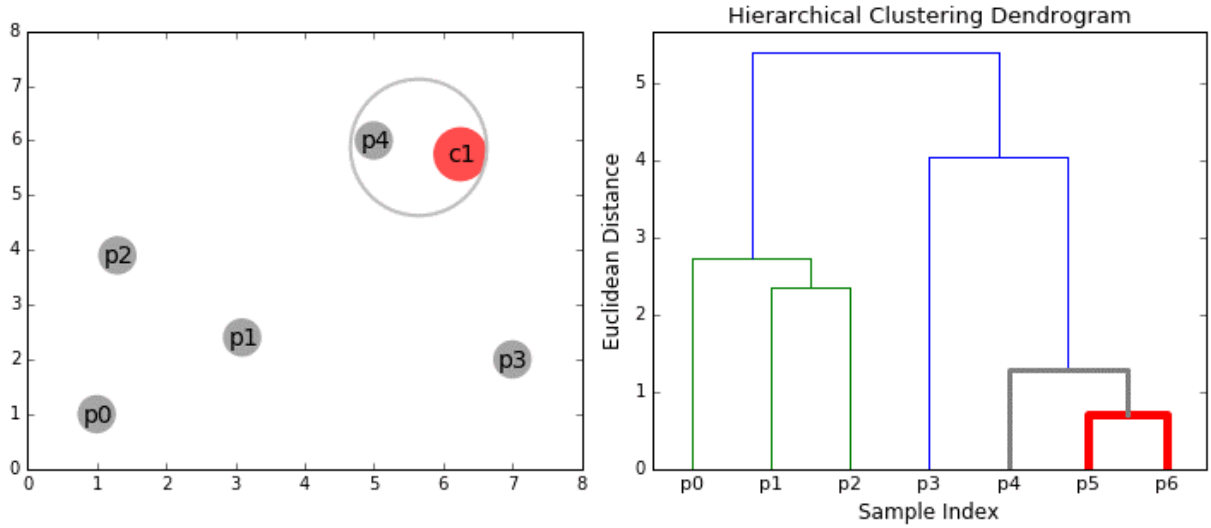
Agglomerative Clustering





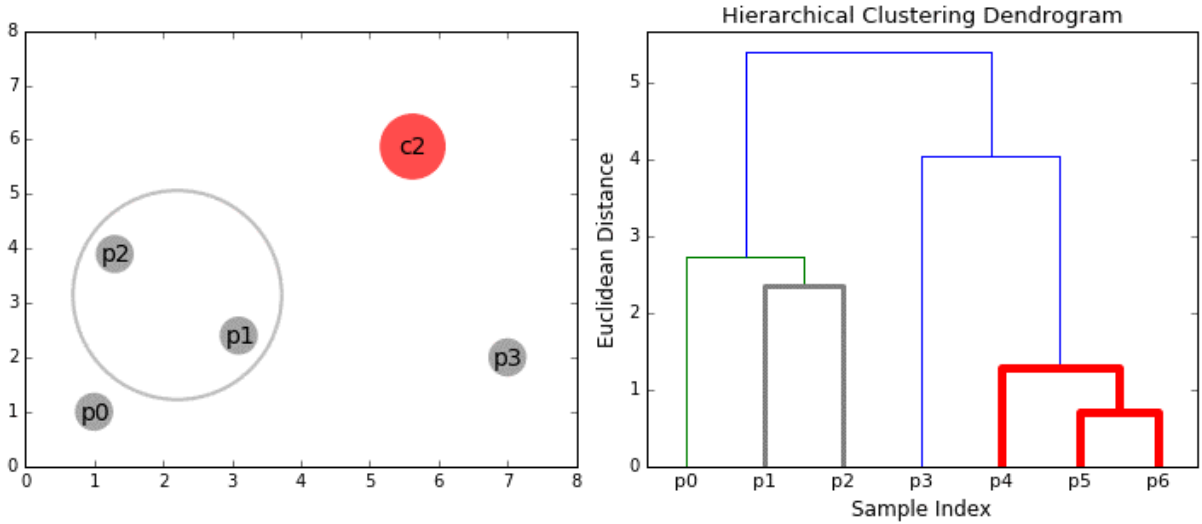
- Each points are initialized as its own cluster
- Compute the *linkage* between clusters
- Merge the two clusters with smallest linkage
- Repeat the process until desirable number of clusters obtained

Agglomerative Clustering



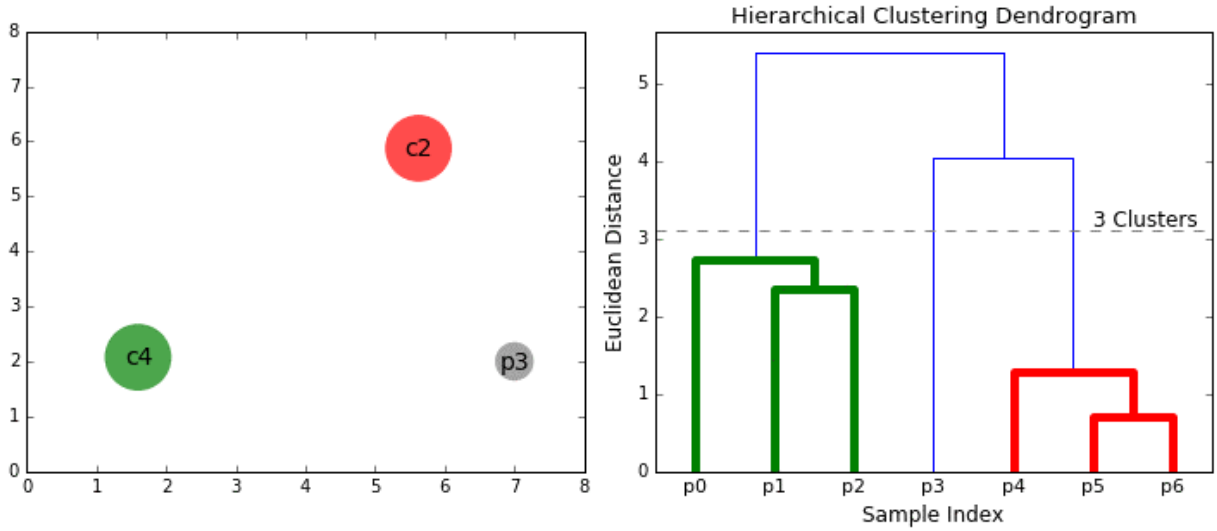
- Each points are initialized as its own cluster
- Compute the *linkage* between clusters
- Merge the two clusters with smallest linkage
- Repeat the process until desirable number of clusters obtained

Agglomerative Clustering



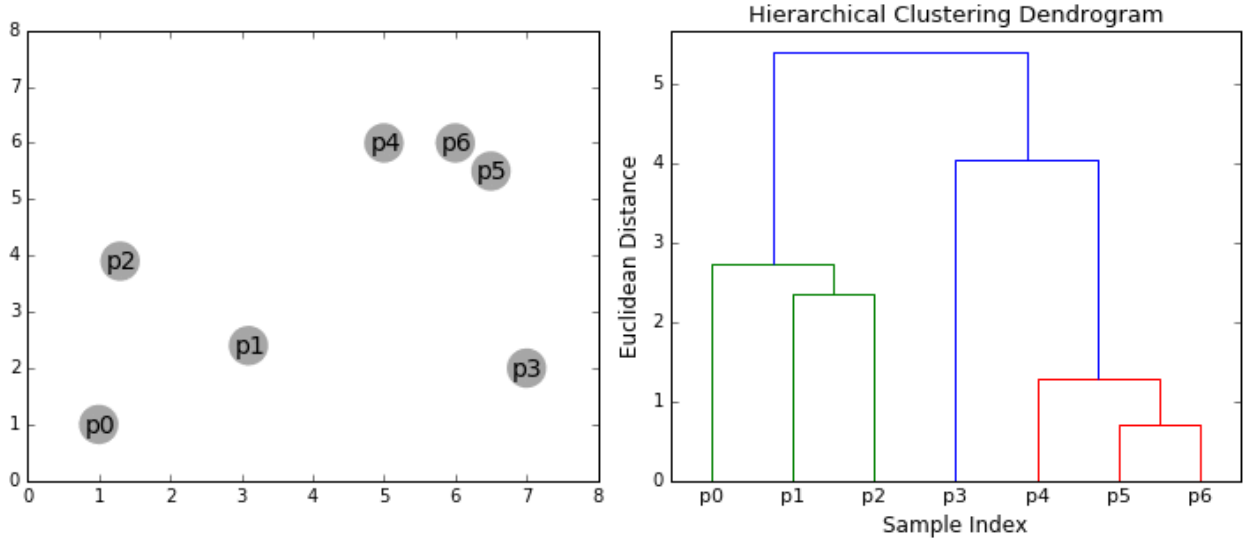
- Each points are initialized as its own cluster
- Compute the *linkage* between clusters
- Merge the two clusters with smallest linkage
- Repeat the process until desirable number of clusters obtained

Agglomerative Clustering



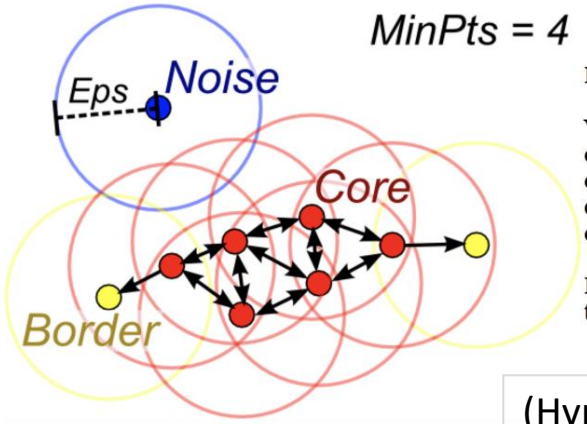
- Each points are initialized as its own cluster
- Compute the *linkage* between clusters
- Merge the two clusters with smallest linkage
- Repeat the process until desirable number of clusters obtained

Agglomerative Clustering - animation



DBSCAN

- **DBSCAN** (**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise)
- **Core point**: a data point that has more than a specified number of *MinPts* within *Eps* radius around it
- **Border point**: a data point that has fewer than *MinPts* within *Eps*, but is in the neighborhood of a core point
- **Noise point**: any point that is not a *core* point or a *border* point



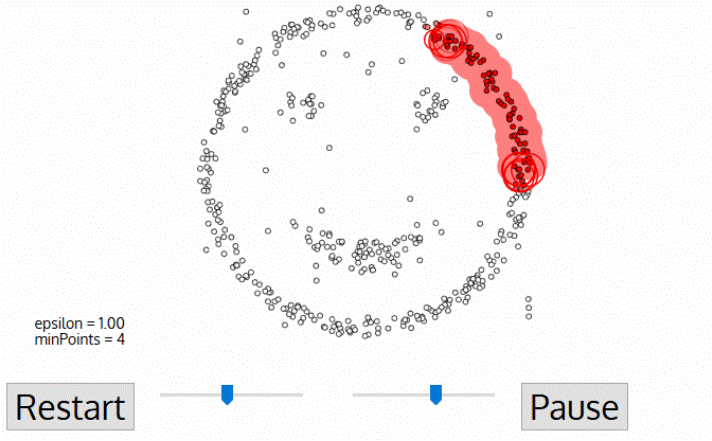
Red: Core Points

Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

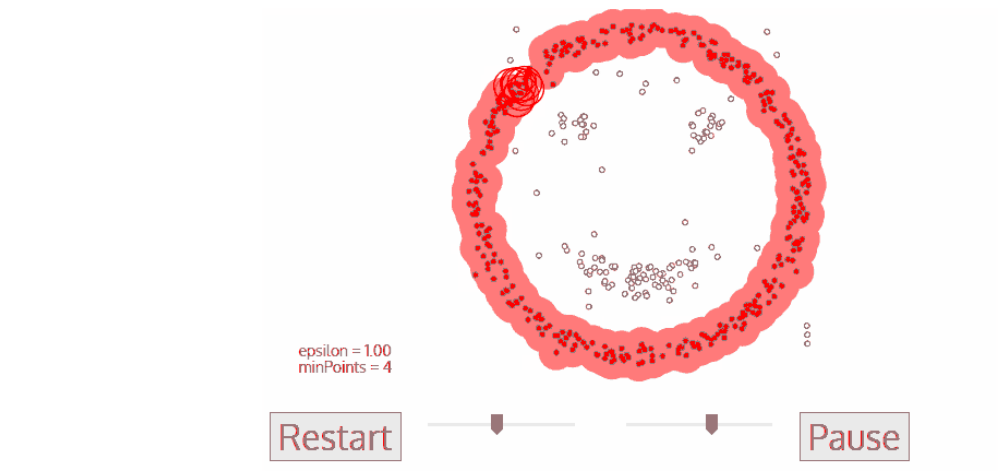
(Hyperparameters: *MinPts* and *Eps*)

DBSCAN



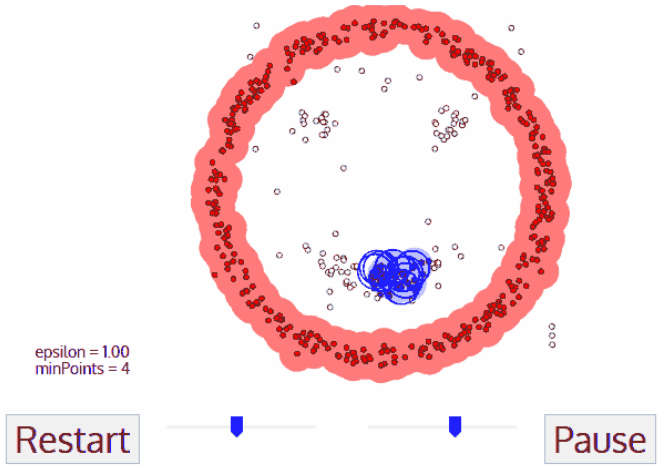
- Randomly select a **core point** to start a new cluster
- Iteratively add all points (**core** and **border**) within the *Eps* distance to the cluster
- Stop when no more points are within the *Eps* neighborhood
- Repeat the procedure with an unvisited core point, and stop when no more unvisited core point

DBSCAN



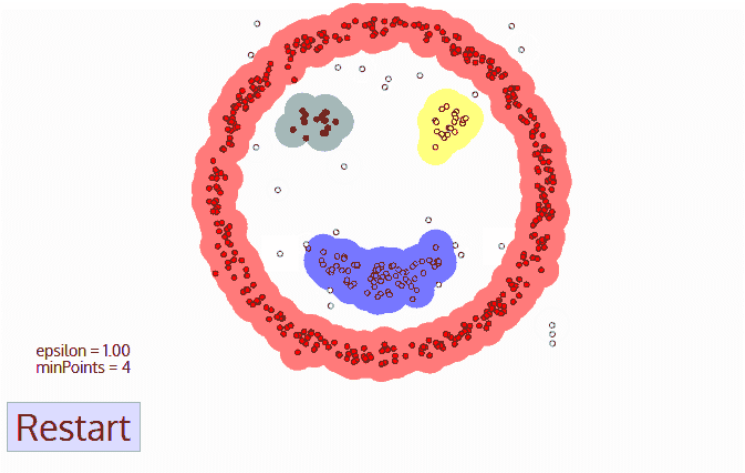
- Randomly select a **core point** to start a new cluster
- Iteratively add all points (**core** and **border**) within the *Eps* distance to the cluster
- Stop when no more points are within the *Eps* neighborhood
- Repeat the procedure with an unvisited core point, and stop when no more unvisited core point

DBSCAN



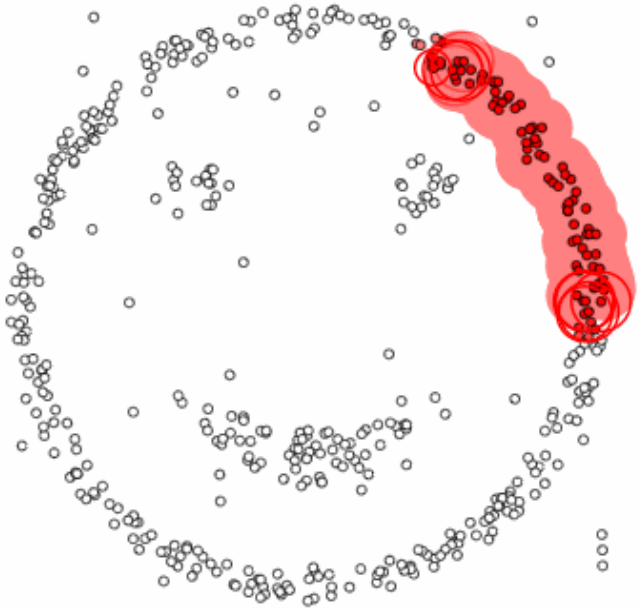
- Randomly select a **core point** to start a new cluster
- Iteratively add all points (**core** and **border**) within the *Eps* distance to the cluster
- Stop when no more points are within the *Eps* neighborhood
- Repeat the procedure with an unvisited core point, and stop when no more unvisited core point

DBSCAN



- Randomly select a **core point** to start a new cluster
- Iteratively add all points (**core** and **border**) within the *Eps* distance to the cluster
- Stop when no more points are within the *Eps* neighborhood
- Repeat the procedure with an unvisited core point, and stop when no more unvisited core point

DBSCAN - animation



epsilon = 1.00
minPoints = 4

Restart

Pause

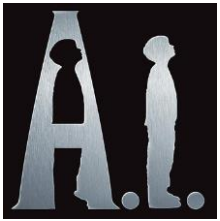
DBSCAN

- Strength :
  - Can handle noise (outliers) very well
  - Can handle clusters of different shapes and sizes

- Weakness :
  - Does not work well when dealing with clusters of varying densities
  - Sensitive to the hyperparameters
  - May not work well in high dimensionality of data

Clustering in sklean

```
from sklearn.cluster import Kmeans
from sklearn.cluster import DBSCAN

kmeans = KMeans(
    init= "random", # initialization method
    n_clusters=3,   # cluster number
    n_init=10,      # number of initialization runs
    max_iter=300,   # max iteration
    random_state=42
)
dbscan = DBSCAN(eps=0.3, min_samples=5)

kmeans.fit(input_features)  # train
dbscan.fit(input_features)

kmeans.inertia_  # lowest SSE among the runs

kmeans.cluster_centers_  # the k cluster centers

kmeans.labels_[:] # assigned labels for input features
dbscan.labels_[:]
```



Next:

Reinforcement Learning