AI
Notes

Context

Exam :
Monday 30th , 9:30am – 12:00pm

4 Questions :
1. Clustering
2. Supervised Learning
3. Search
4. Optimisation

Each 15 marks,
total 60 marks
**note: no GMM and weka

Content :

• Clustering	(Week 1 & 2)
• Supervised Learning	(Week 3 - 6)
• Search	(Week 7 & 8)
• Optimisation	(Week 9 & 10)

Clustering :
- Hierarchical Clustering
- K-means
- GMM/EM
- DBSCAN

Supervised Learning :
- Linear Regression
- Gradient Descent
- Logistic Regression
- Neural Network
- Evaluate & Hyperparameter Tuning
- Naive Bayes
- KNN-algorithm

Search :
- Search Problem
- BFS
- DFS
- A*

Optimisation :
- Hill Climbing
- Simulated Annealing
- Formulation
- Constraint handling

Introduction

Basic Introduction:

Assuming a data set with $\boldsymbol{n}$ samples, and each sample has $\boldsymbol{d}$ dimensions :

$$[(x_1^1, x_2^1, \dots x_d^1), (x_1^2, x_2^2, \dots x_d^2), \dots, (x_1^n, x_2^n, \dots x_d^n)] = 1,2..n$$

Instance : One data sample
Dimension : Number of coordinates to specify the samples
Feature / Attribute : The value at each dimension

you can think of $\boldsymbol{n}$ samples as data inputs, and $\boldsymbol{d}$ dimensions as attributes
(or in Objects terms,
n : Objects
d : properties)

There are 3 types of Machine Learning:

- **Supervised Learning**

(Labelled Data)

- **Unsupervised Learning**

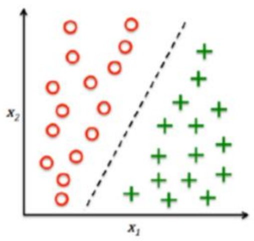
(Unlabeled Data, may be due to data being too big / unclear)

- **Reinforced Learning**

(Learns from rewards, rewards changes by time)

Classification

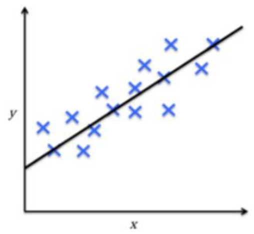
Outputs are **categorical**



eg: handwritten numbers → [1,2,3,..]

Regression

Outputs are **continuous numbers**



eg: Student Score Marks → 83.2%

Clustering

Hierarchical Clustering:

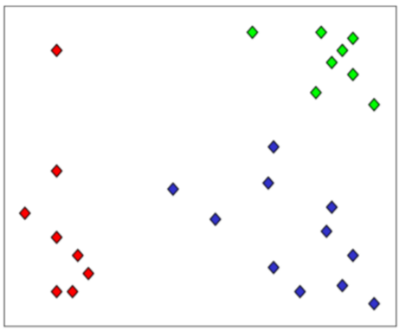
Terminology

Singleton : one data point

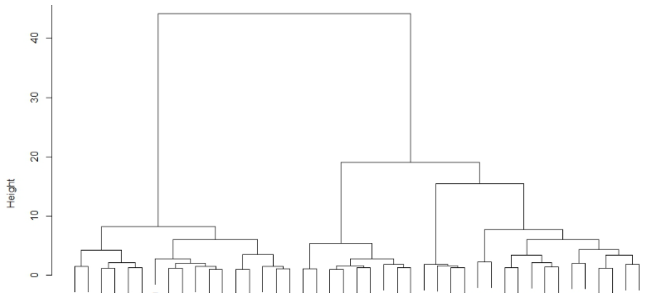
Cluster : multiple data points

Segments data into clusters such that there is a "natural grouping" among objects, it creates a hierarchical decomposition of the set of objects using some criterions

eg :



And produces a ***dendrogram*** *:*



Note : the distance between clusters are noted as height

In the end, there will be only 1 cluster all grouped together

Measuring Distance

we will be using **Euclidean Distance** only

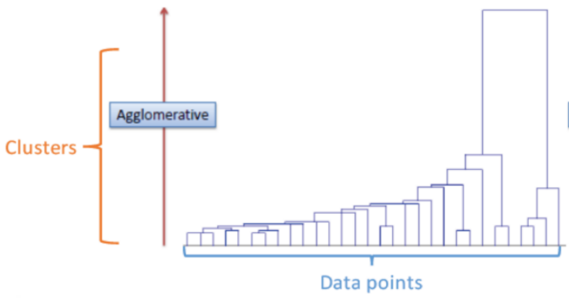
$$distance(a,b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

extra : it is also Minkowski with dimension = 2

There are 2 ways of Hierarchical clustering:

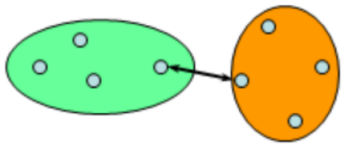
- **Agglomerative** *(bottom-up merging)*
- **Divisive** *(top-down merging)*

Divisive *is not recommended as it can be computationally expensive*



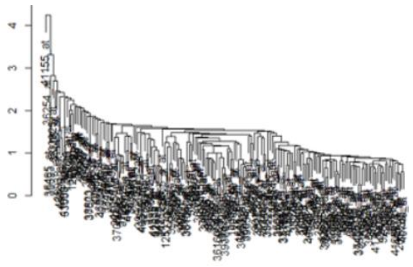
There are 3 ways to measure distance between clusters:

- **Single Linkage**



- ➢ Distance of the **closest pair**
- ➢ Produces **long chain shape** of clusters

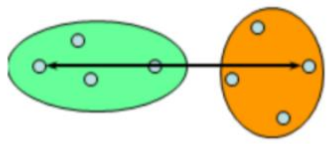
Eg:



$$c_1, c_2 = min\left(\forall distance\left(p_{c_1}, p_{c_2}\right)\right)$$

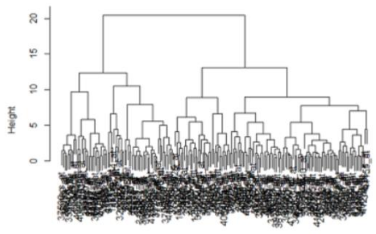
Where c_1 , c_2 are clusters and return minimum distance between 2 points in c_1 and c_2

- **Complete Linkage**



  - Distance of the **furthest pair**
  - Produces **compact shape** clusters
  - *Sensitive to noise*

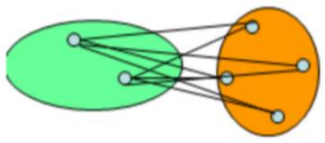
Eg:



$$c_1, c_2 = max\left(\forall distance\left(p_{c_1}, p_{c_2}\right)\right)$$

return maximum distance between 2 points in $c_1$and c_2

- **Group Average**



  - Average of all distances
  - Most used
  - *Robust against noise*

$$c_1, c_2 = \frac{\sum \forall distance\left(p_{c_1}, p_{c_2}\right)}{Num\ \mathrm{p}_{c_1} + Num\ \mathrm{p}_{c_2}}$$

return Average distance of **all the points** in $c_1$and c_2

Advantages

- Deterministic results (same result for any run)
- Does not need to specify number of clusters
- Can create cluster of arbitrary shapes

Disadvantages

- Does not scale up to larger datasets, time complexity is $O(n^2)$
- Will impose hierarchical structure even though data is not appropriate for it
- Once a decision is made, cant be undone

Algorithm (agglomerative):

Step 1 : Find every Euclidean distance

(create distance matrix)

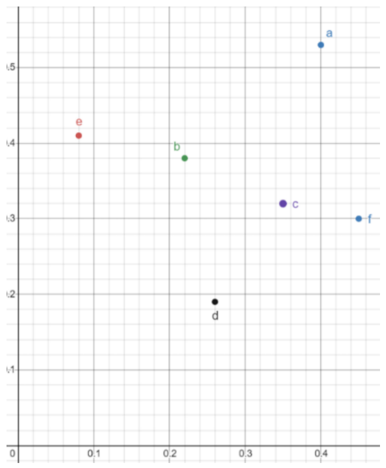
Step 2 : Get the **minimum** distance , which the 2 points/cluster joins becoming the new cluster

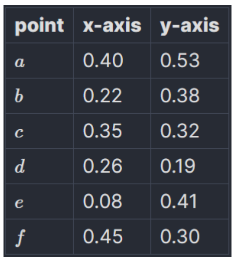
Step 3 : Recalculate the distance matrix with the new cluster

- Complete Linkage → Max
- Single Linkage → Min
- Group Average → Average

Step 4 : Repeat **Step 2**, until only 1 cluster left , Done!

Example of a run (Complete Linkage):





point	x-axis	y-axis
a	0.40	0.53
b	0.22	0.38
c	0.35	0.32
d	0.26	0.19
e	0.08	0.41
f	0.45	0.30

Step 1 – Use Euclidean distance to create distance matrix

(in this case we are only calculating point a and b)

$$distance(a, b) = \sqrt{(a_x - b_x)^2 + \left(a_y - b_y\right)^2}$$

$$distance(a, b) = \sqrt{(0.40 - 0.22)^2 + (0.53 - 0.38)^2}$$

$$= \sqrt{(0.18)^2 - (0.15)^2}$$

$$= \sqrt{0.0549}$$

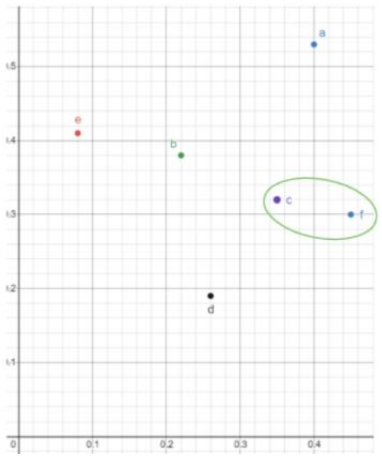
$$= 0.23$$

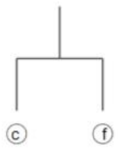
Distance Matrix:

	a	b	c	d	e	f
a	0					
b	0.23	0				
c	0.22	0.15	0			
d	0.37	0.20	0.15	0		
e	0.34	0.14	0.28	0.29	0	
f	0.23	0.25	0.11	0.22	0.39	0

Step 2 – get minimum value, forms the cluster

	a	b	c	d	e	f
a	0					
b	0.23	0				
c	0.22	0.15	0			
d	0.37	0.20	0.15	0		
e	0.34	0.14	0.28	0.29	0	
f	0.23	0.25	0.11	0.22	0.39	0





Step 3 – Recalculate the distance matrix (Complete Link)

to update the table:

$$\text{NewValue} = \max(distance(p_1, n), distance(p_2, n))$$

Which concludes:

$$\max(distance(c, a), distance(f, a)) = 0.23$$

$$\max(distance(c, b), distance(f, b)) = 0.25$$

$$\max(distance(c, d), distance(f, d)) = 0.22$$

$$\max(distance(c, e), distance(f, e)) = 0.39$$

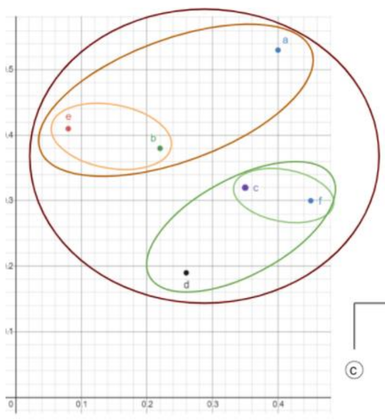
Updated Table :

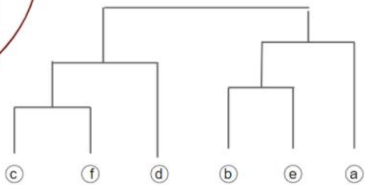
	a	b	c,f	d	e
a	0				
b	0.23	0			
c,f	0.23	0.25	0		
d	0.37	0.20	0.15	0	
e	0.34	0.14	0.28	0.29	0

Step 4 – Repeat Step 2 until 1 cluster left

Will give an output of the distance of the final 2 cluster

**many more steps here but will skip ahead till 1 cluster left as it's just a repeat





Done!

K-Means:

Terminology

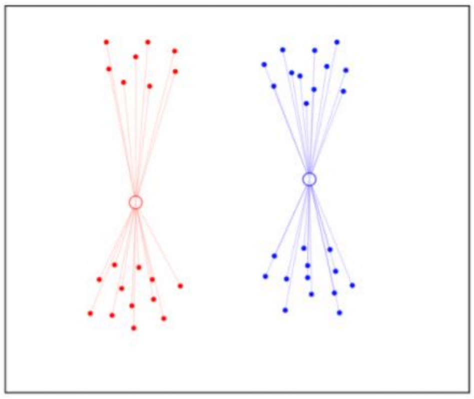
K : number of clusters

Centroid : the center of uniform density

Centroid Based – Describes each cluster by its mean

Its objective is to *minimize the within cluster variances* of all clusters that you set, ***K***

*eg (**K** = 2) :*



Advantages

- Easy to implement
- Efficient

Disadvantages

- Non-deterministic results (different result for every run)
- May result in local optima (multiple restart to get global optima)
- Require number of clusters in advance

Algorithm :

Step 1 : Set number of cluster , **K**

(K will usually have coordinates, if not → set random)

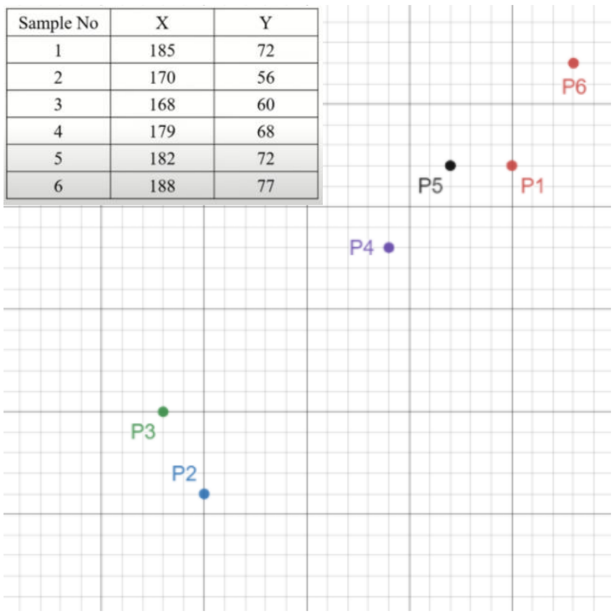
Step 2 : Calculate the distance matrix of every points to **K centroid**

Step 3 : Assign the points to the closest **K centroid** (minimum)

Step 4 : Recalculate the cluster centroid mean

Step 5 : Repeat **Step 2**, until **K**-mean stop changing, Done!

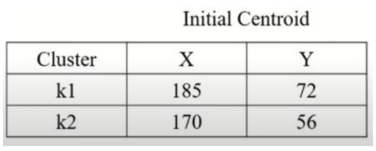
Example of a run :



Sample No	X	Y
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77

Step 1 : Set number of cluster , K

Set $K = 2$ and *in this case:* $K_1 = P_1, K_2 = P_2$



Initial Centroid

Cluster	X	Y
k1	185	72
k2	170	56

Step 2 : Calculate the distance matrix of every points to K centroid

Using Euclidean Distance formula :

$$\boxed{distance(a,b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}}$$

Gives :

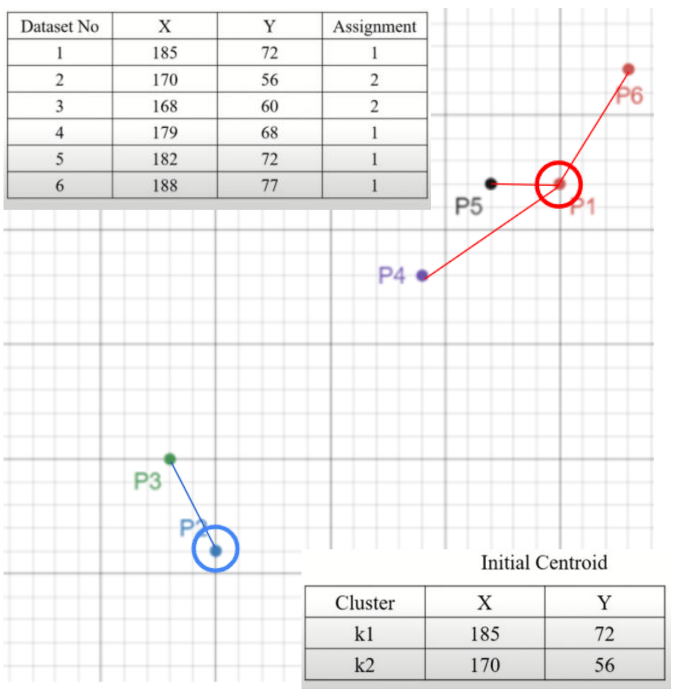
Sample No	K1	K2
3	20.809	4.472
4	7.211	15
5	3	20
6	5.831	27.659

Step 3 : Assign the points to the closest K centroid

Using formula :

$$\boxed{\min\left(distance(P_n, \mathrm{K}_1), distance(P_n, \mathrm{K}_2)\right)}$$

"assign to **K** closest to it"



Dataset No	X	Y	Assignment
1	185	72	1
2	170	56	2
3	168	60	2
4	179	68	1
5	182	72	1
6	188	77	1

Initial Centroid

Cluster	X	Y
k1	185	72
k2	170	56

Step 4 : Recalculate the cluster centroid mean

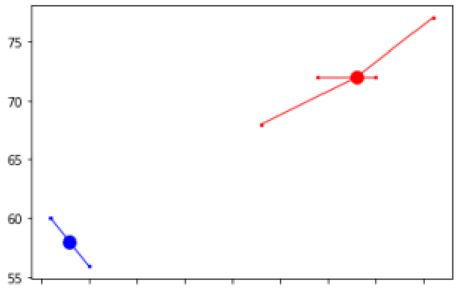
Formula for new K-mean if $\text{dimension} = 2$:

$$\boxed{\frac{\sum P_x}{n}, \frac{\sum P_y}{n}}$$

n = number of points in that cluster

Cluster	X	Y
k1	= (185+179+ 182+188) / 4 = 183	= (72+68+ 72+77) / 4 = 72
k2	= (170 + 168) / 2 = 169	= (60 + 59) / 2 = 58

Step 5 : Repeat Step 2, until K-mean stop changing



Done!

GMM/EM:

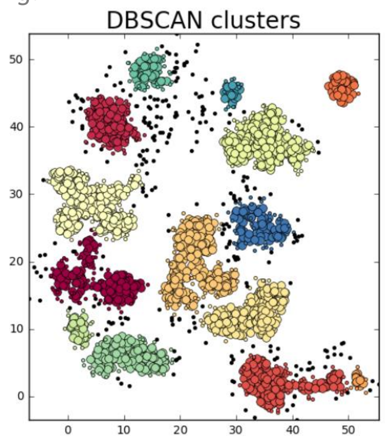
NO

DBSCAN:

Density Based Spatial Clustering of Application with Noise (you do not need to memorize the name)

Discover clusters by ***density*** of regions

Eg:



Usually, there are 2 parameters :

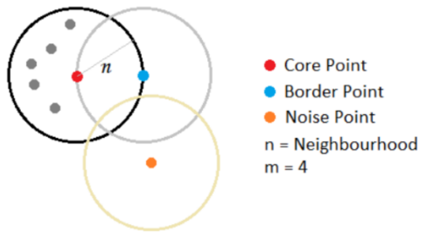
- Radius of circle (*ϵ*)
- Minimum number of points in that circle (*minPts*)

Points are categorized into 3 terms:

Core : point has *minPts* and within radius *ϵ*

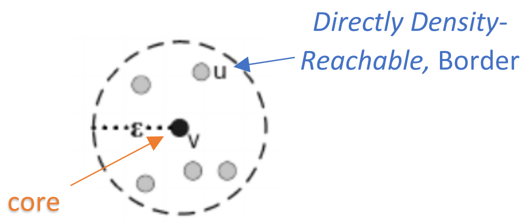
Border : does not have *minPts* and point lies within radius *ϵ of* core

Noise/Outlier : point which are not border or core



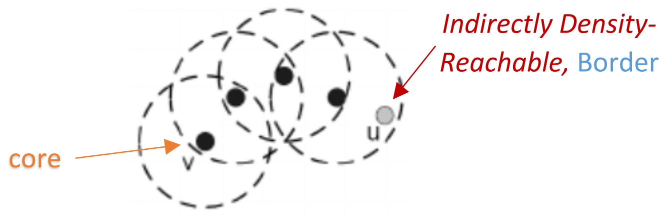
Forming Clusters

Directly Density-Reachable



A point, u is *Directly Density-Reachable* if it is within a core's radius of point v

Indirectly Density-Reachable



A point, u is *Indirectly Density-Reachable* if it is a border but **not within** a core's radius of point v

Advantages

- Can discover arbitrary shape clusters
- Robust to noise/outliers
- Doesn't need number of cluster

Disadvantages

- Non-deterministic results (border points may have different clusters depending on code)
- If data is not well-explained, choosing *ϵ* and *minPts* may be difficult
- May fail if data too sparse

Algorithm :

Step 1 : Label all the points , **core/border/noise**

Step 2 : Remove all the noise/outlier points

(or set them as *Noise*)

Step 3 : ***For every core point***, Assign points that are *density-Reachable* from that core and make it a cluster

Step 4 : Reassign border points belonging to more than 1 cluster to be the one closest to it

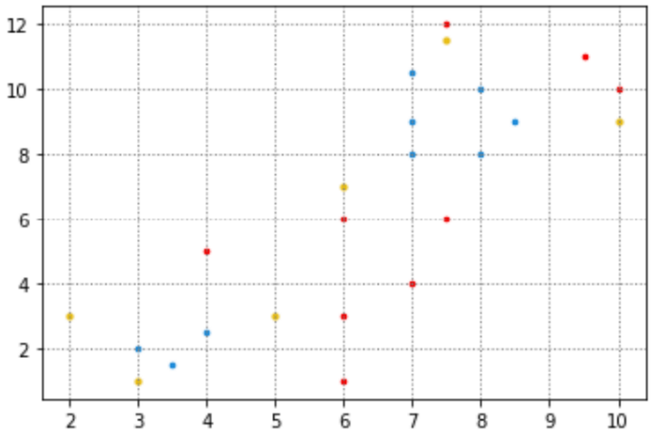
Done!

Example of a run :

Step 1 : Label all the points , **core**/**border**/**noise**

have set:

- Radius of Circle (ϵ) → *1.5*
- Number of Neighbors (minPts) → *4*

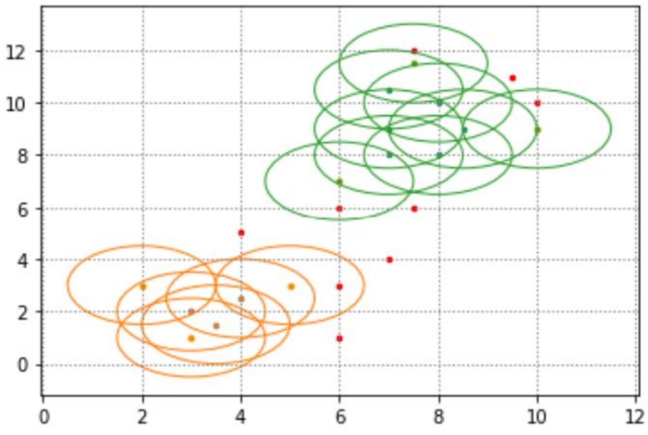


Step 2 : Remove all the noise/outlier points

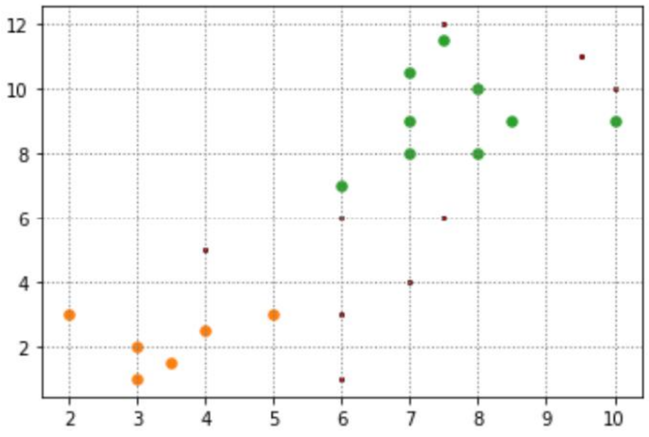
(in code, we can just ignore them to increase efficiency of program)

```
onlyList = core + border
```

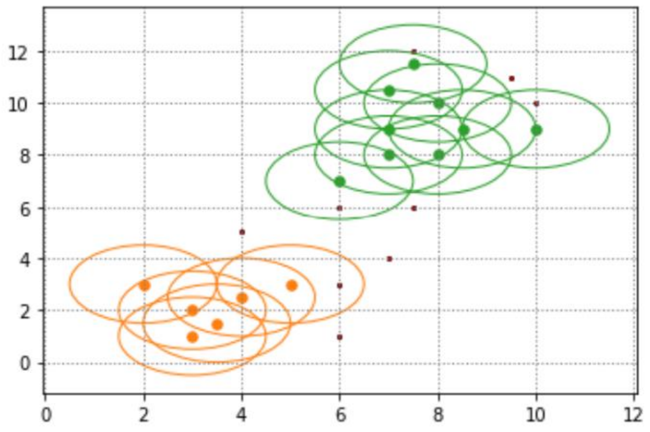
Step 3 : ***For every core point***, Assign points that are *density-Reachable* from that core and make it a cluster



Without radius shown :



Step 4 : Reassign border points belonging to more than 1 cluster to be the one closest to it



(In this case, we don't need to as border doesn't belong to more than 1 cluster)

Done!