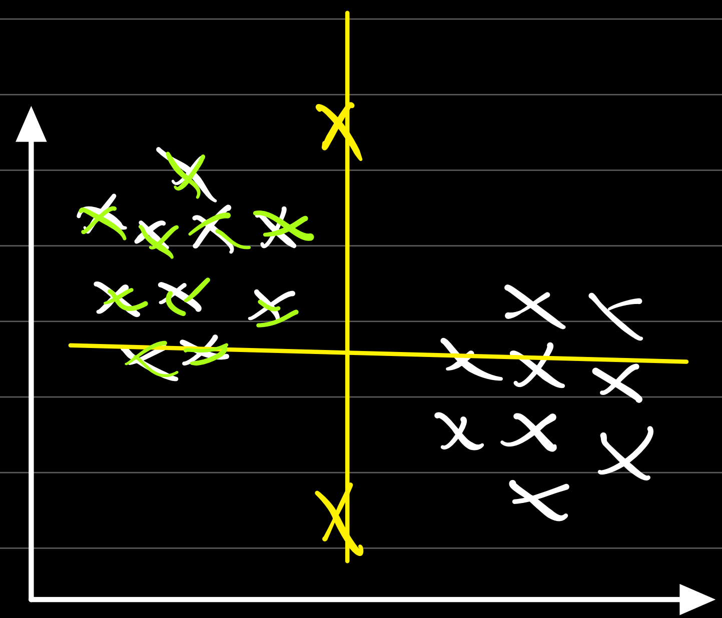
* K-Means Clustering

After chosing optimal value of k:

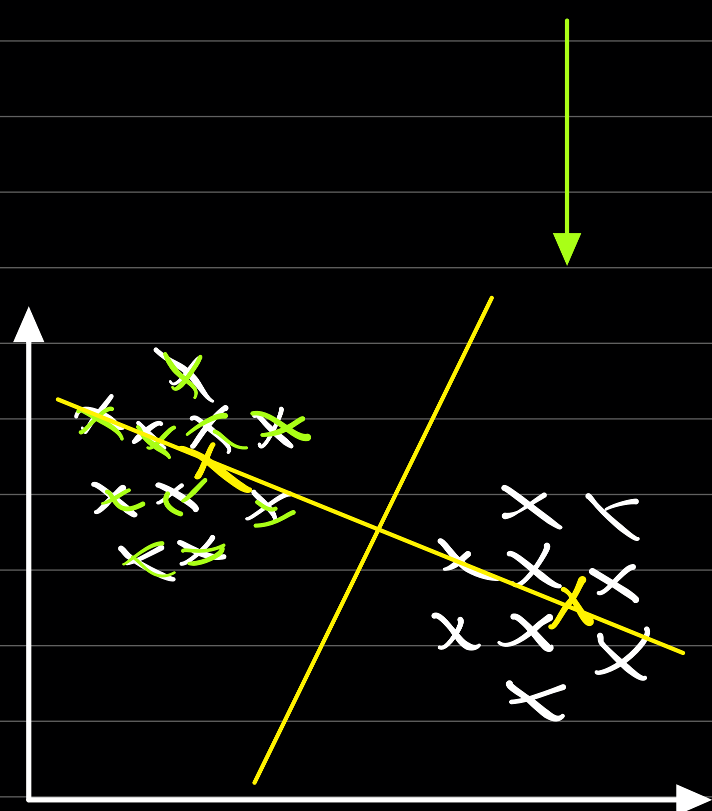
Step 1) We try k values

Step 2) Initilize k number of centroids

Step 3) Compute the avg to update the centroids



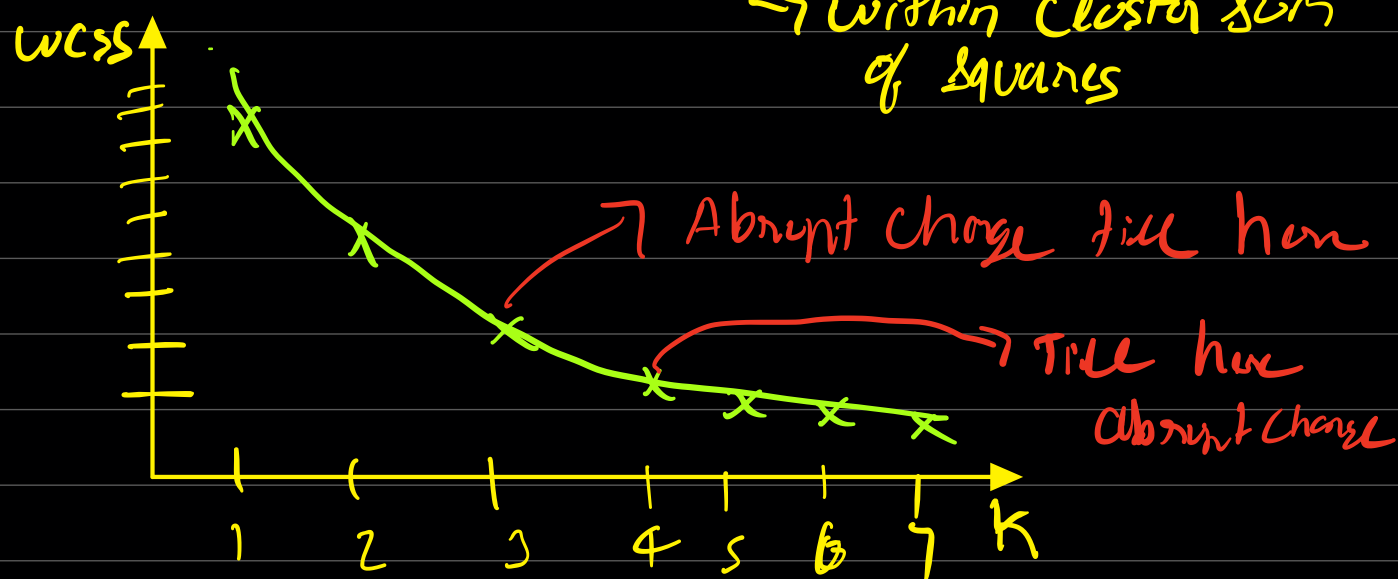
Updated Centroids



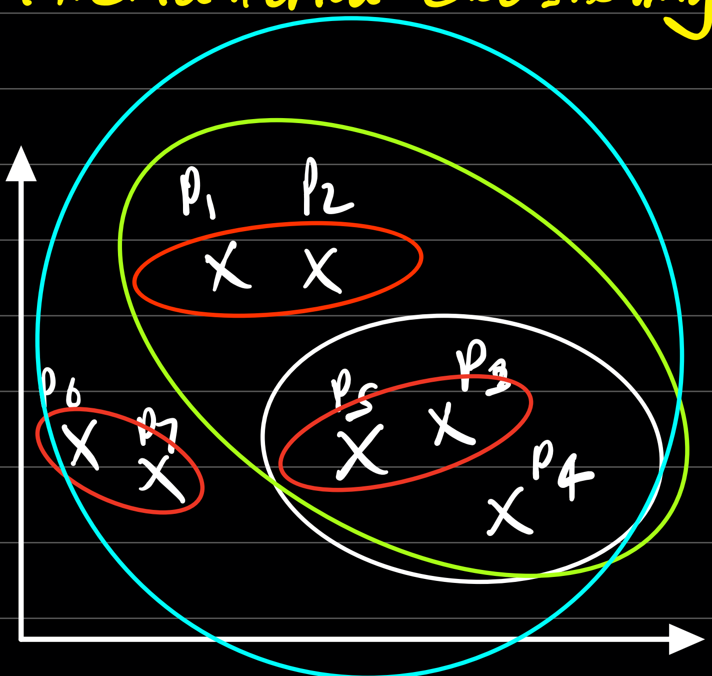
But how to find the optimal K?

Elbow method

We plot a graph of WCSS v/s K
↳ within cluster sum of squares

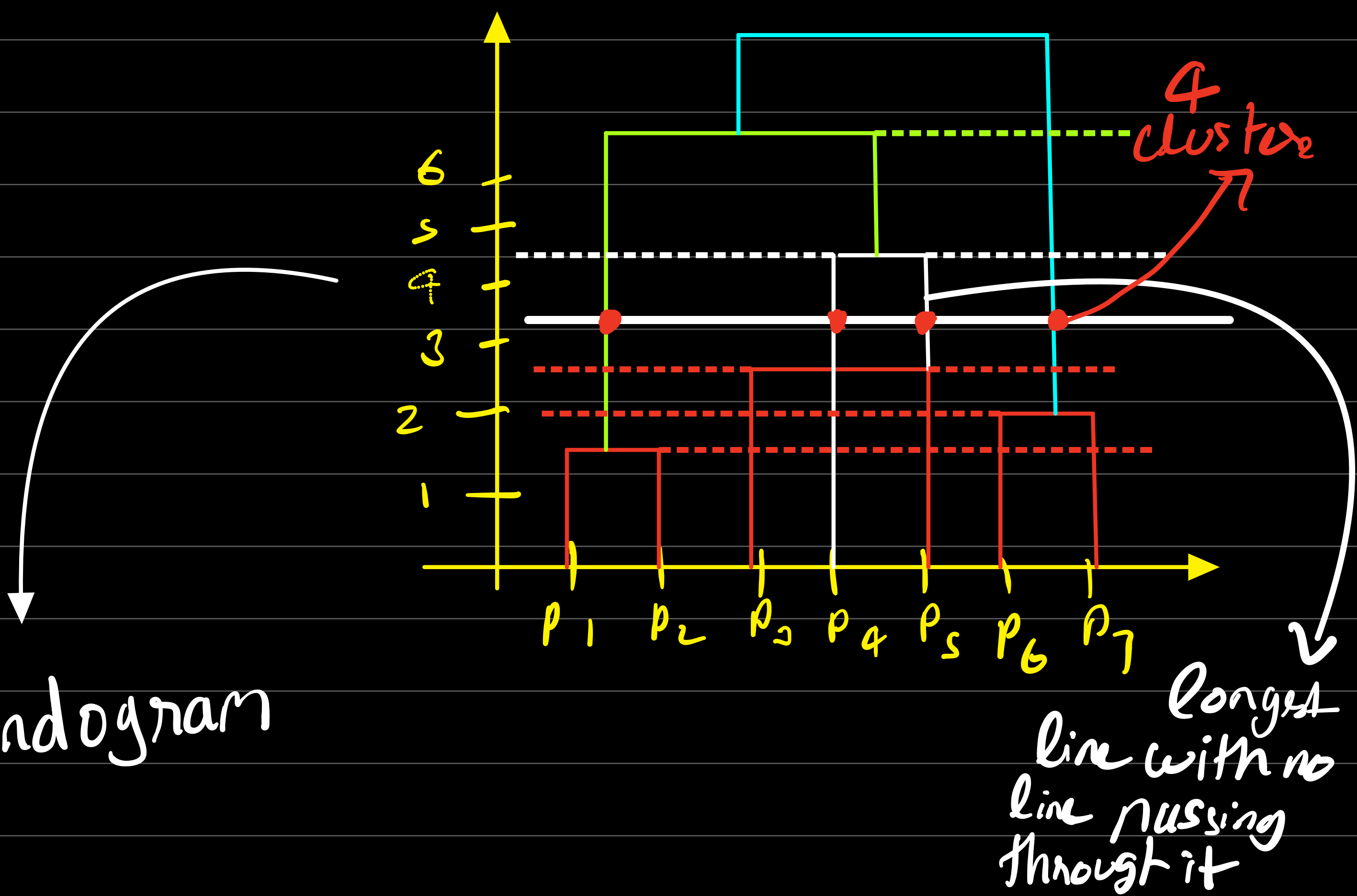


* Hierarchical Clustering



1) It combines all two nearest points

2) It combines two nearest clusters or a nearest point and a cluster



Dendogram

How to find the optimal K value or no. of cluster??

You need to find longest vertical line that has no vertical line passed through it.

MAX Time taken by K-Means or Hierarichal?

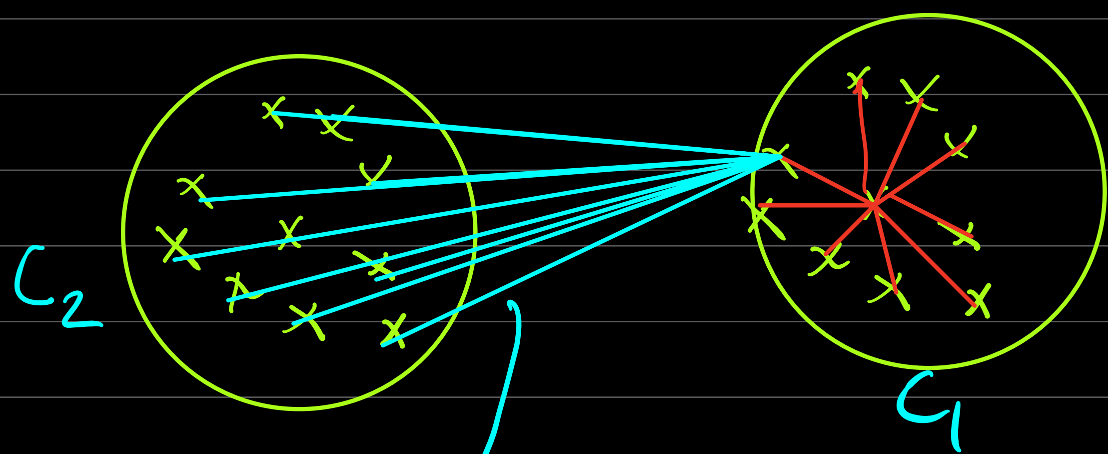
If dataset is large it will take a lot of time

* Silhouette (clustering) → for validation of models

$$a_i = \frac{1}{|C_I| - 1} \sum_{j \in C_I,\, i \neq j} d(i, j)$$

↓

Average Distance



$$b(i) = \min \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

This for every point in C_1 and then calculate avg distance

For a good model $a(i) << b(i)$

Silhouette (value) for one datapoint i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad , \text{if } |C_I| > 1$$

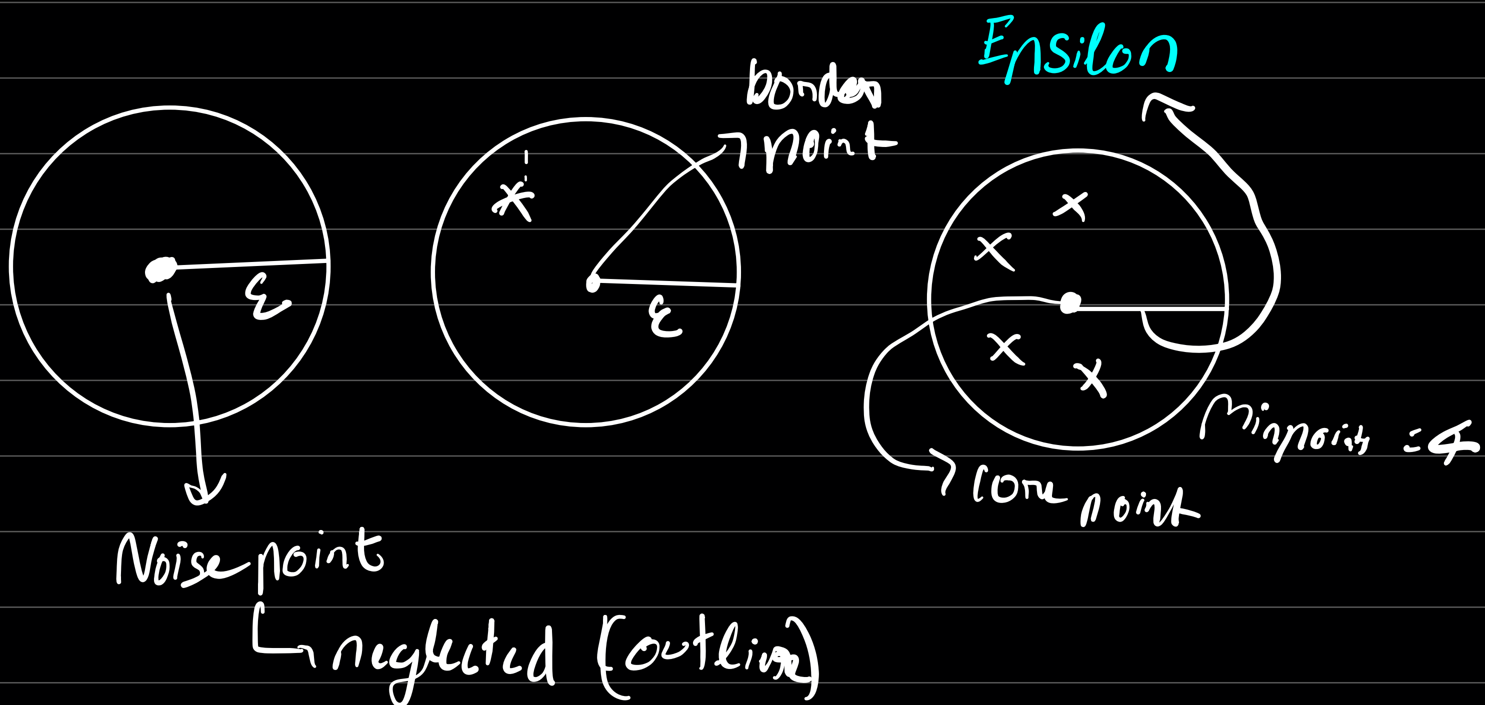
$[-1, 1]$

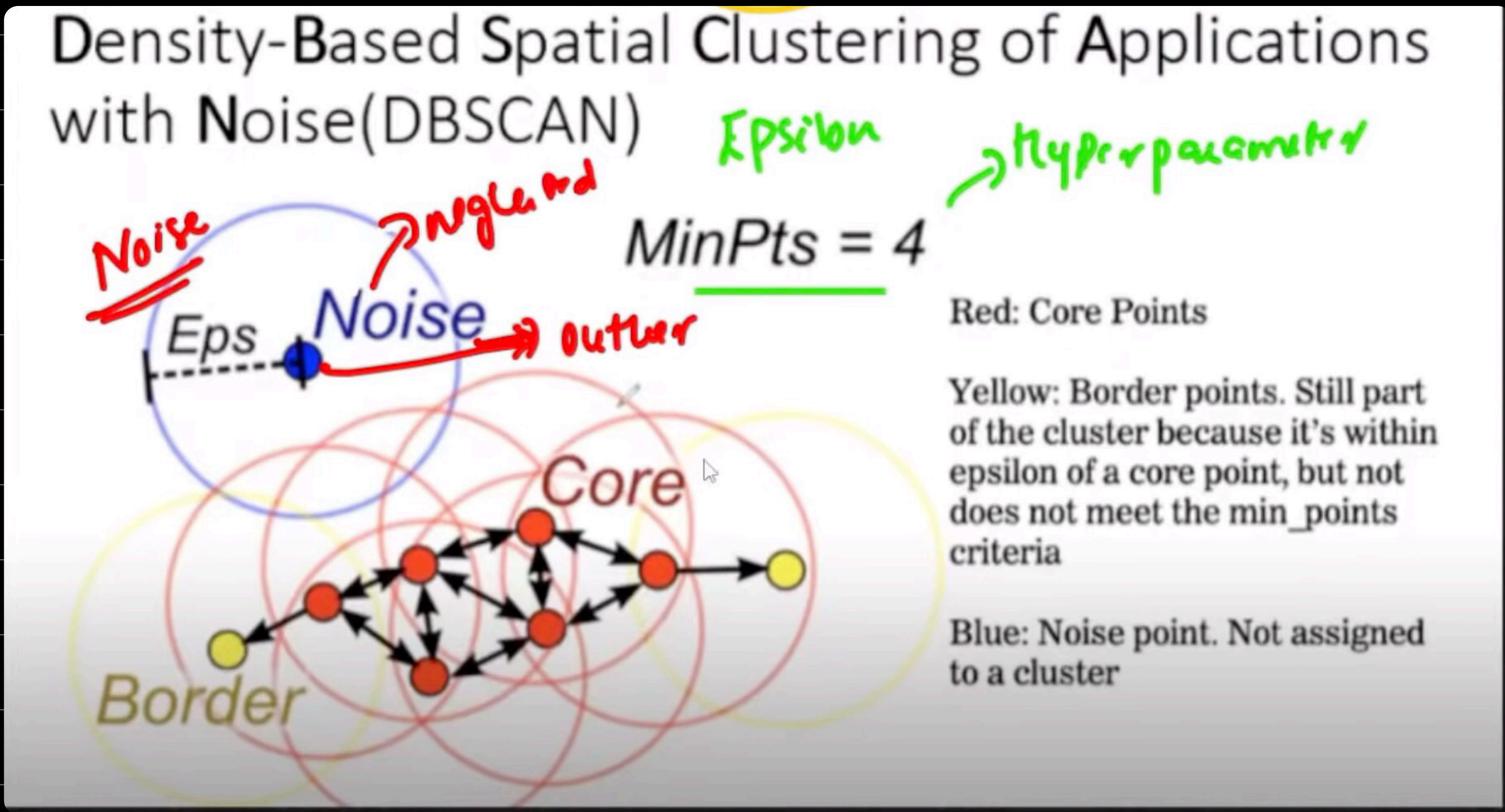
More towards -1 means that the model is not good i.e. $b(i) > a(i)$. For a good Clustering model score will be near 1

* DBSCAN Clustring (Density Based Spatial Clustring of Applications with Noise)

Hyperparameter

1. Min points
2. Core points
3. Border points
4. Noise points





Rechability - If a datapoint can be accessed from another data point directly or indirectly.

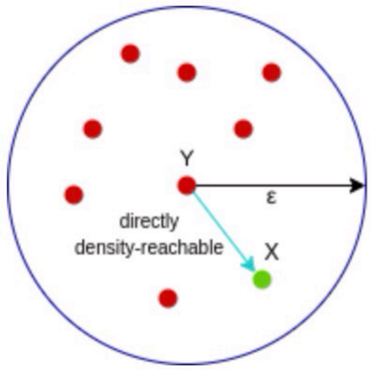
Connectivity - Whether two data points belong to the same cluster or not.

In terms of rechability and connectivity two points in DBSCAN

- Direct Density Reachable
- Density - Reachable
- Density - Connected

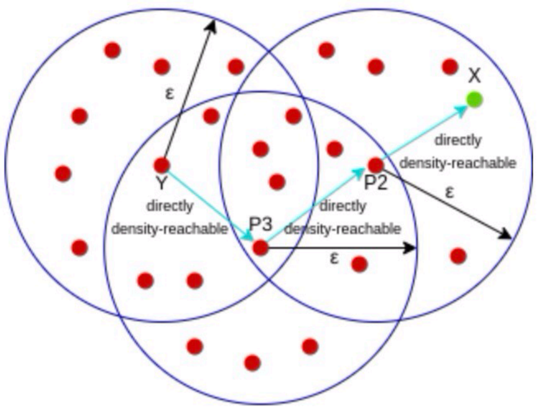
A point **X** is **directly density-reachable** from point **Y** w.r.t *epsilon, minPoints* if,

1. **X** belongs to the neighborhood of **Y**, i.e, *dist(X, Y) <= epsilon*
2. **Y** is a core point



Here, **X** is directly density-reachable from Y, but vice versa is not valid.

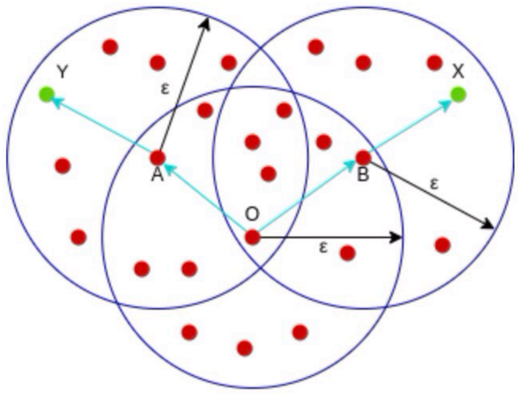
A point **X** is **density-reachable** from point **Y** w.r.t *epsilon, minPoints* if there is a chain of points p1, p2, p3, …, pn and p1=**X** and pn=**Y** such that pi+1 is directly density-reachable from pi.



Here, **X** is density-reachable from **Y** with **X** being directly density-reachable from **P2**, **P2** from **P3**, and **P3** from **Y**. But, the inverse of this is not valid.

Multiple direct-density collection

A point **X** is **density-connected** from point **Y** w.r.t *epsilon and minPoints* if there exists a point **O** such that both **X** and **Y** are density-reachable from **O** w.r.t to *epsilon and minPoints.*



Here, both **X** and **Y** are density-reachable from **O**, therefore, we can say that **X** is density-connected from **Y**.

DBSCAN is very sensitive to the value of epsilon and minpoints

minpoints ≥ Dimensions + 1

minpoints must be atleast 3, generally it is twice the dimentions. But domain knowledge also decides the value

Value of ε can be decided from the K-distance graph

* Mean Shift Clustering (One of the best algos used in image processing and CV)

- Kernal Density Estimation

KDE is the application of kernal smoothing for probability density estimation.

Chat GPT:

Here's how KDE works:

1. Data Representation: You start with a dataset containing continuous data points.
2. Kernel Functions: KDE uses kernel functions (such as Gaussian kernels) to represent each data point as a probability distribution. These kernels are centered at each data point and have a bandwidth parameter that determines their spread.
3. Summation: The kernel functions for all data points are summed up, creating a smooth estimate of the underlying probability density function. This smoothed density function provides an estimate of how likely data points are to occur at different values along the continuous feature space.
4. Visualization: KDE can be used to create density plots or contour plots that visualize the estimated probability density across the feature space. These plots often show peaks or regions of high density, which can be indicative of clusters in the data.

Let $(x_1, x_2, x_3 \ldots x_n)$ be independent and identically distributed samples drawn from some univariate distribution with unknown density f at any given point x.

each random variable has same probability distribution as the others and all are mutually independent.

probability distribution of only one random variable

We are intrested in estimating the shape of this function f.

It's kernel density estimator is :-

Nothing but a function

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

$K \rightarrow$ Kernal

$h \rightarrow$ $h > 0$, smoothing parameter called bandwidth

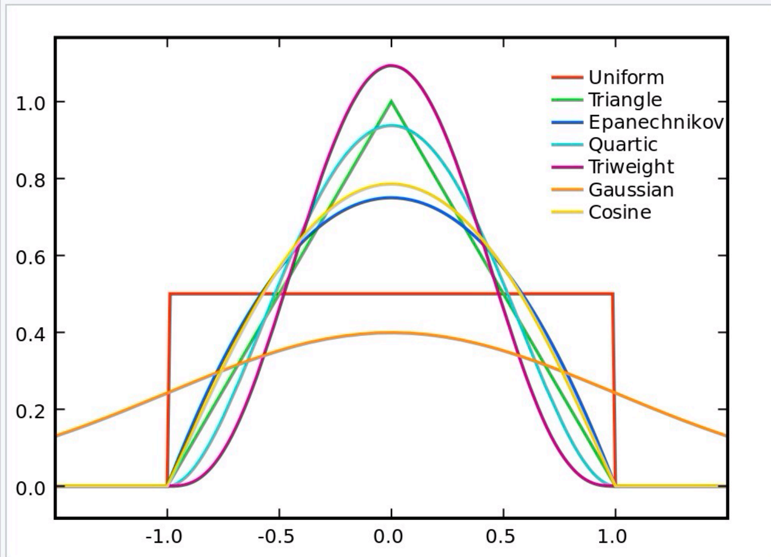
Kernal with subscript h is called scaled kernal and is defined as.

$$K_h(x) = \frac{1}{h} K(x/h)$$

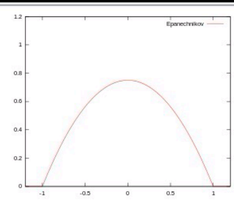
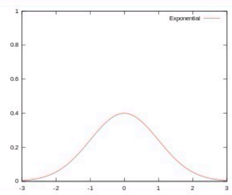
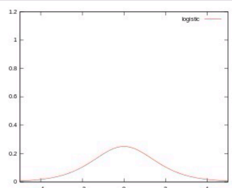
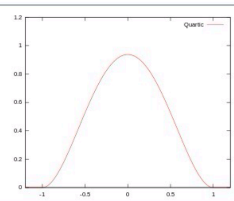
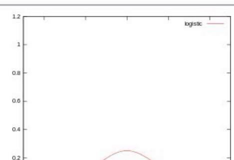
Kernel functions in common use [edit]

Several types of kernel functions are commonly used: uniform, triangle, Epanechnikov,[1] quartic (biweight), tricube,[2] triweight, Gaussian, quadratic[3] and cosine.

In the table below, if K is given with a bounded support, then $K(u) = 0$ for values of u lying outside the support.



All of the kernels below in a common coordinate system.

Epanechnikov (parabolic)	$K(u) = \frac{3}{4}(1-u^2)$ Support: $\lvert u\rvert \le 1$		$\frac{1}{5}$	$\frac{3}{5}$	100%
Gaussian	$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$		1	$\frac{1}{2\sqrt{\pi}}$	95.1%
Logistic	$K(u) = \frac{1}{e^u + 2 + e^{-u}}$		$\frac{\pi^2}{3}$	$\frac{1}{6}$	88.7%
Quartic (biweight)	$K(u) = \frac{15}{16}(1-u^2)^2$ Support: $\lvert u\rvert \le 1$		$\frac{1}{7}$	$\frac{5}{7}$	99.4%
Sigmoid function	$K(u) = \frac{2}{\pi} \frac{1}{e^u + e^{-u}}$		$\frac{\pi^2}{4}$	$\frac{2}{\pi^2}$	84.3%

The bandwidth parameter used to make the KDE surface varies on the different sizes. For example, we have a tall skinny kernel which means a small kernel bandwidth and in a case where the size of the kernel is short and fat, which means a large kernel bandwidth. A small kernel bandwidth makes the KDE surface hold the peak for every data point more formally, saying each point has its cluster; on the other hand, large kernel bandwidth results in fewer kernels or fewer clusters.