DBSCAN and Graph Clustering

Mustafa Hajij

Part I
DBSCAN

DBSCAN

In DBSCAN points are classified as follows :

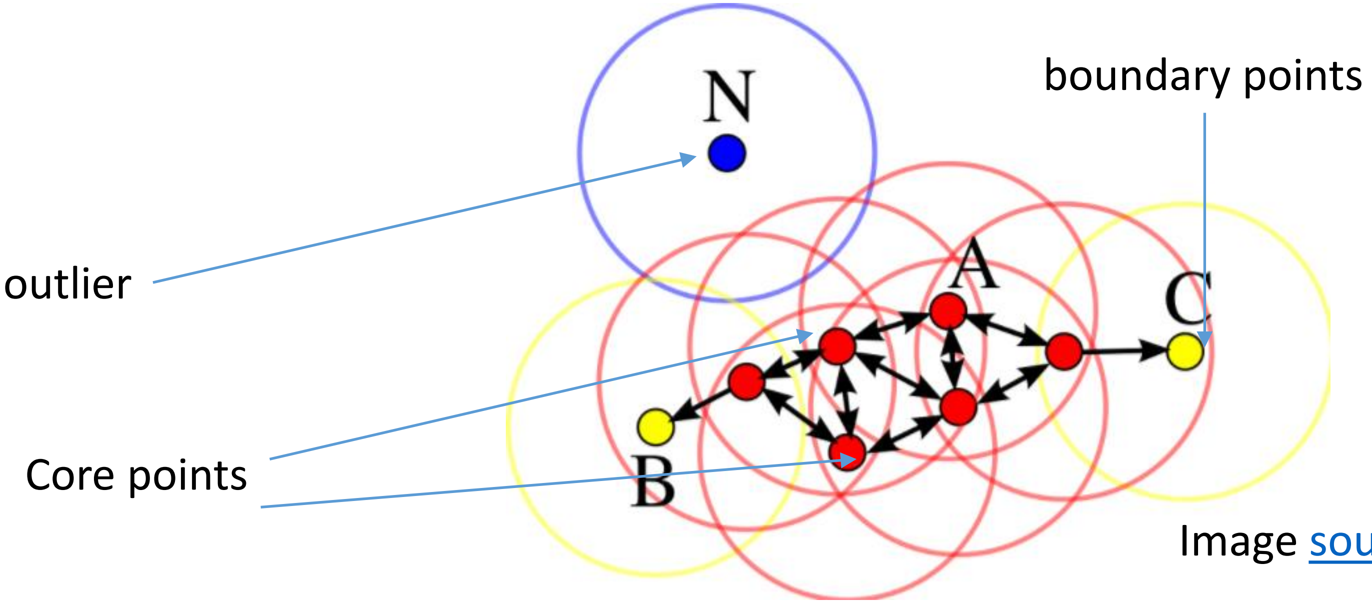


Image source

min_samples=4

DBSCAN

In DBSCAN points are classified as follows :

- *core points* : A point p is said to be a core point if at least **min_samples** points are within a distance ε

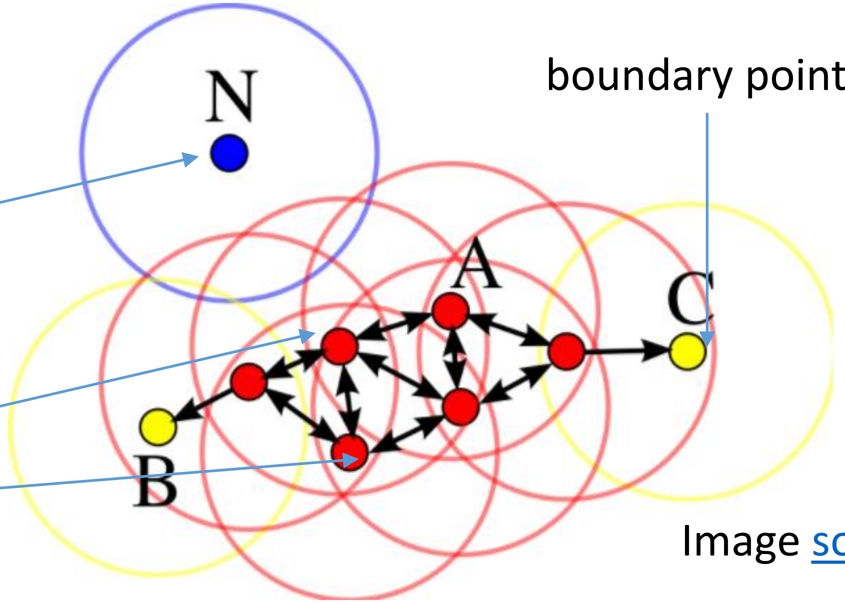


Image source

min_samples=4

DBSCAN

In DBSCAN points are classified as follows :

- *core points* : A point p is said to be a core point if at least **min_samples** points are within a distance ε

- A point q is *directly reachable* from p if the point q is within distance ε from point p and p must be a core point.

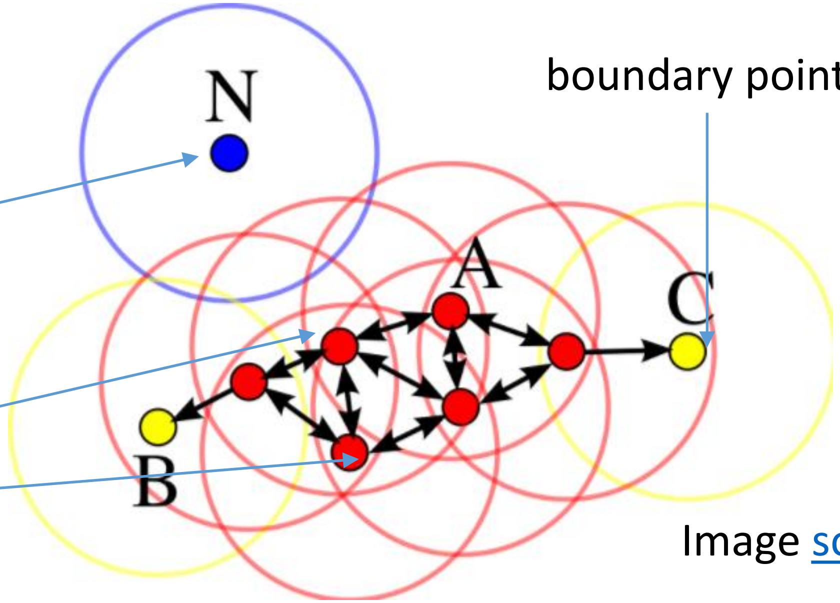


Image [source]

min_samples=4

DBSCAN

In DBSCAN points are classified as follows :

- *core points* : A point p is said to be a core point if at least **min_samples** points are within a distance ε

- A point q is *directly reachable* from p if the point q is within distance ε from point p and p must be a core point.

- *Density-reachable points* : A point q is reachable from p if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$,

where each p_{i+1} is directly reachable from p_i (all the points on the path must be core points, with the possible exception of q).

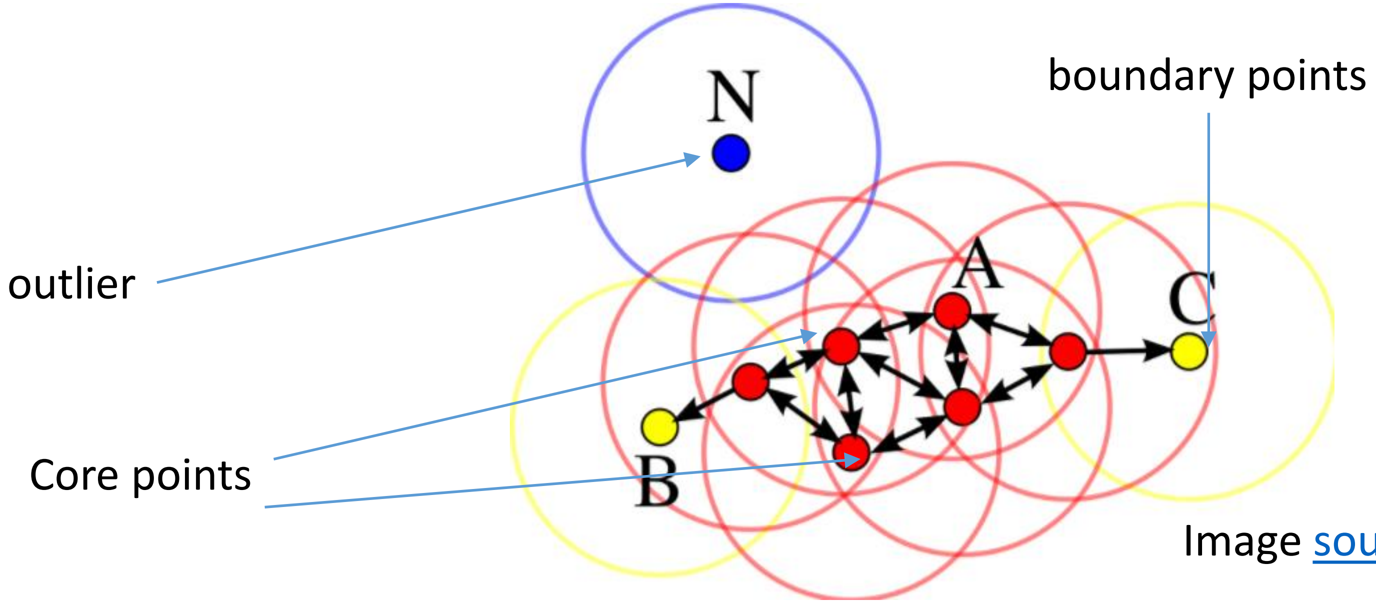


Image source

min_samples=4

DBSCAN

In DBSCAN points are classified as follows :

- *core points* : A point p is said to be a core point if at least **min_samples** points are within a distance ε

- A point q is *directly reachable* from p if the point q is within distance ε from point p and p must be a core point.

- *Density-reachable points* : A point q is reachable from p if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$,
where each p_{i+1} is directly reachable from p_i (all the points on the path must be core points, with the possible exception of q).

- *Outliers :* All points not reachable from any other point are called outliers.

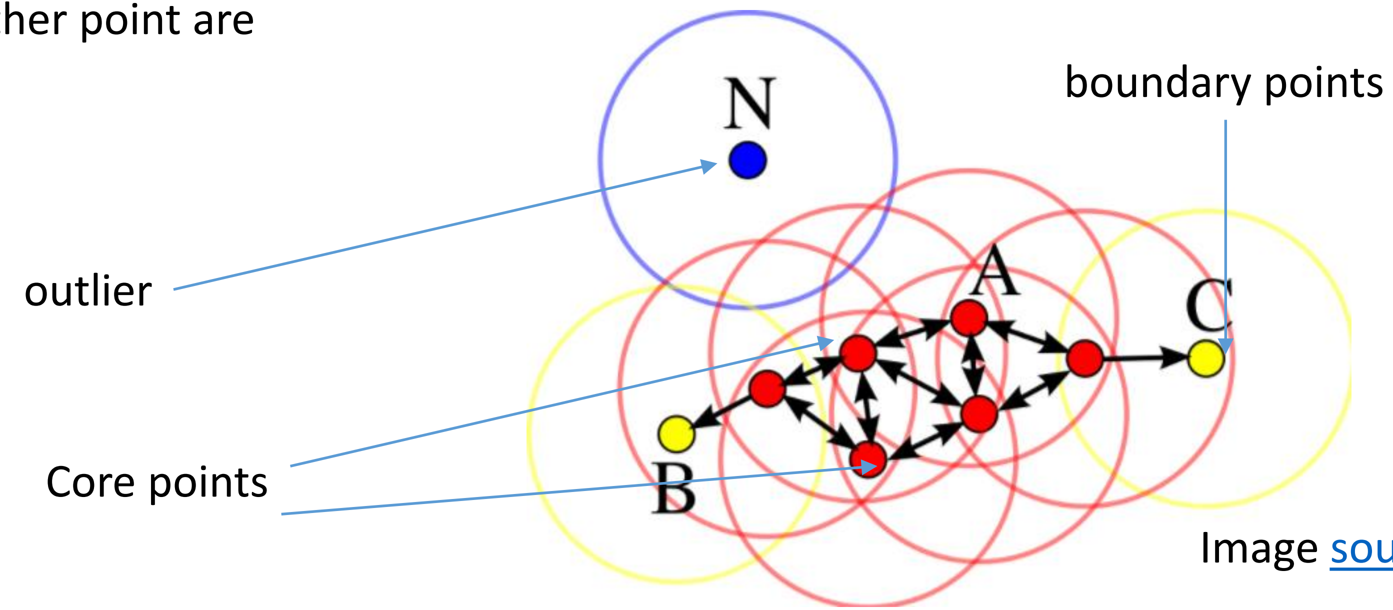


Image source

min_samples=4

DBSCAN

In DBSCAN points are classified as follows :

- *core points* : A point p is said to be a core point if at least **min_samples** points are within a distance ε

- A point q is *directly reachable* from p if the point q is within distance ε from point p and p must be a core point.

- *Density-reachable points* : A point q is reachable from p if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i (all the points on the path must be core points, with the possible exception of q).

- *Outliers :* All points not reachable from any other point are called outliers.

Boundary points cannot reach other points but other points core points can reach them

outlier

Core points

boundary points

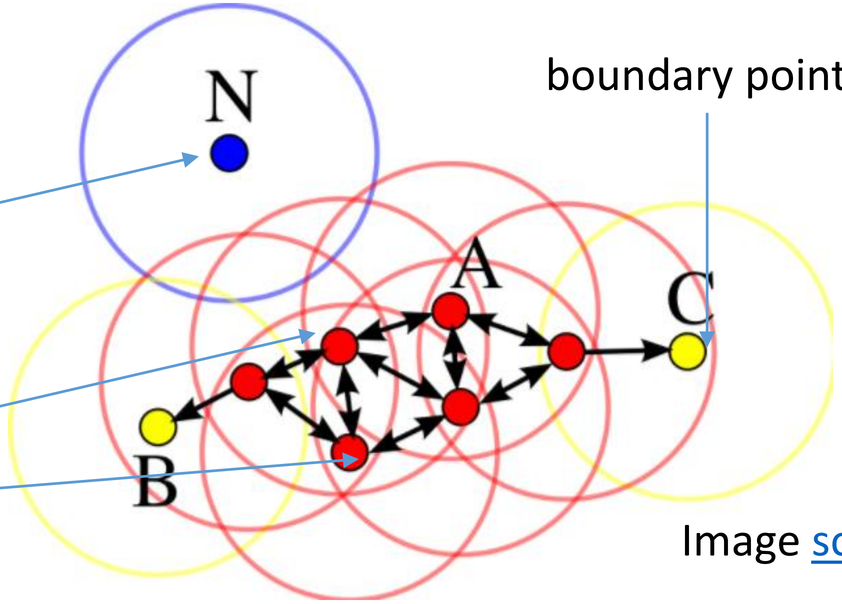


Image source

min_samples=4

DBSCAN-Reachability

If p is reachable from q. Does that mean q is reachable from p ? Explain. Can you give an example from the points below?

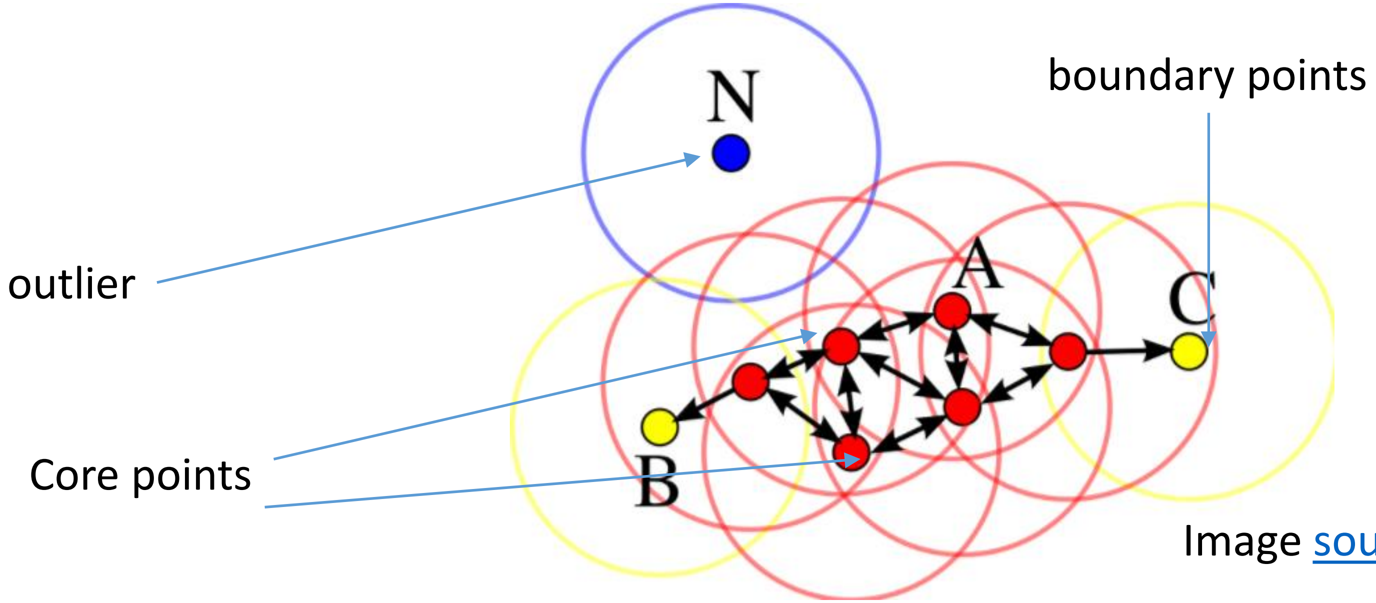


Image source

min_samples=4

DBSCAN-Density-connectedness

Two points *p* and *q* are density-connected if there is a point *o* such that both *p* and *q* are reachable from *o*.

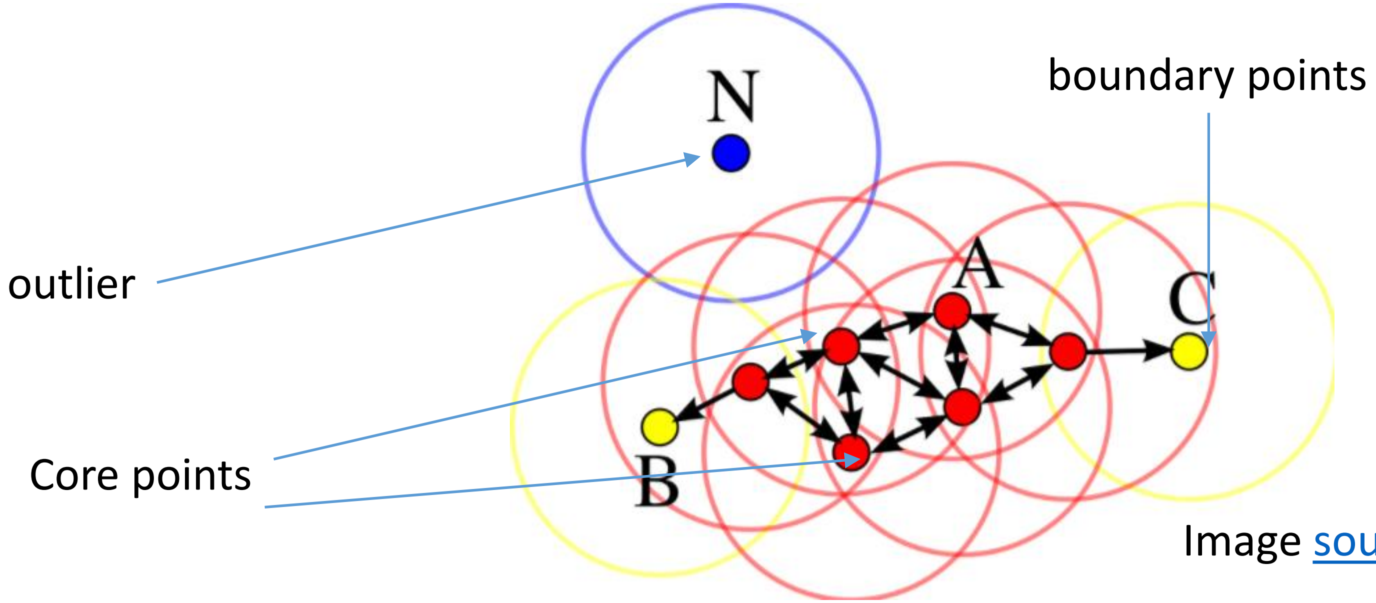


Image source

min_samples=4

DBSCAN-Density-connectedness

Two points *p* and *q* are density-connected if there is a point *o* such that both *p* and *q* are reachable from *o*.

A cluster then satisfies two properties:

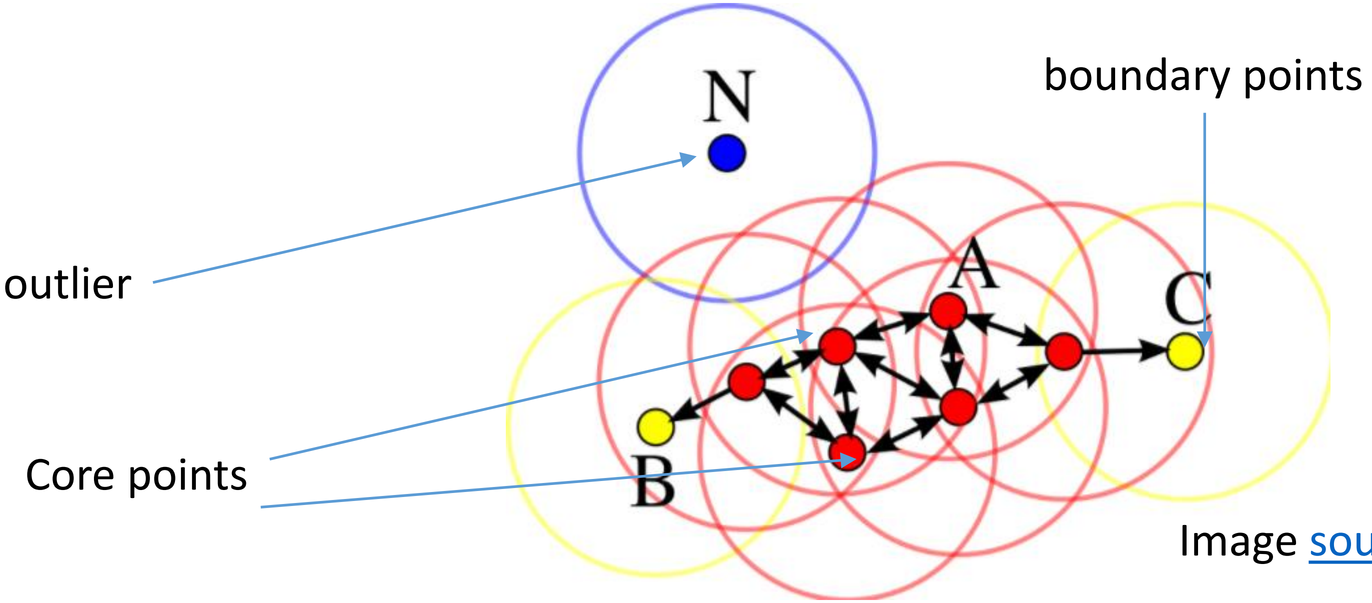


Image source

min_samples=4

DBSCAN-Density-connectedness

Two points *p* and *q* are density-connected if there is a point *o* such that both *p* and *q* are reachable from *o*.

A cluster then satisfies two properties:

1. All points within the cluster are mutually density-connected.

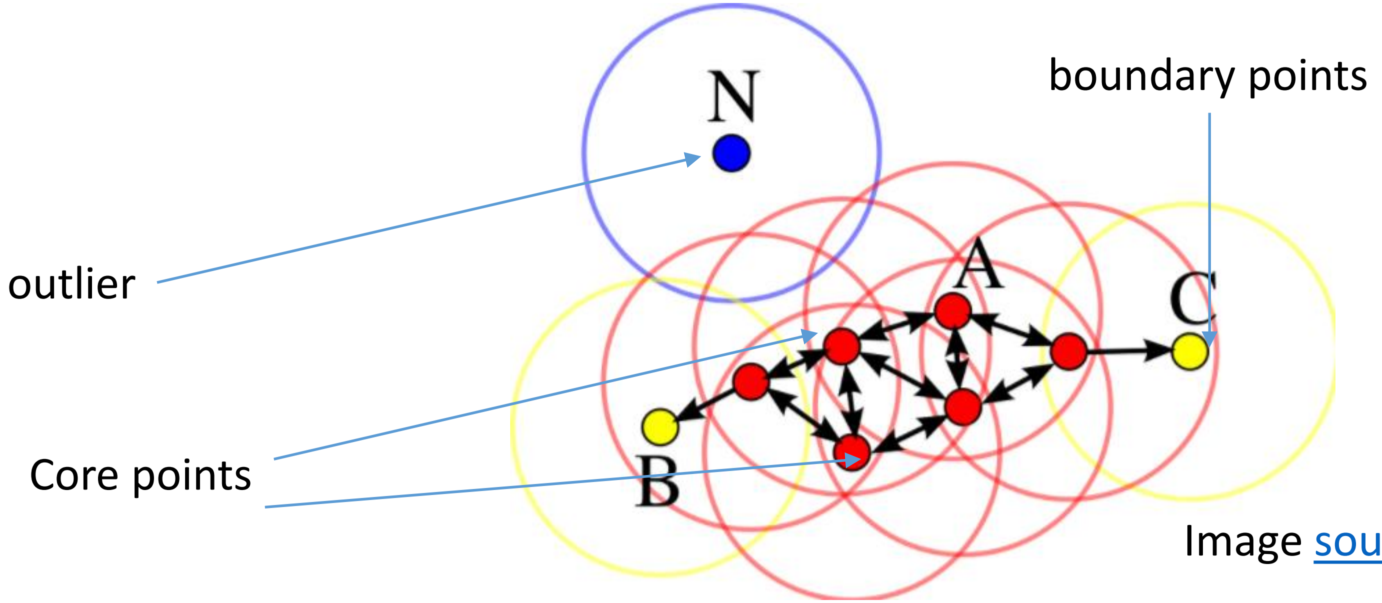


Image source

min_samples=4

DBSCAN-Density-connectedness

Two points *p* and *q* are density-connected if there is a point *o* such that both *p* and *q* are reachable from *o*.

A cluster then satisfies two properties:

1. All points within the cluster are mutually density-connected.
2. If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

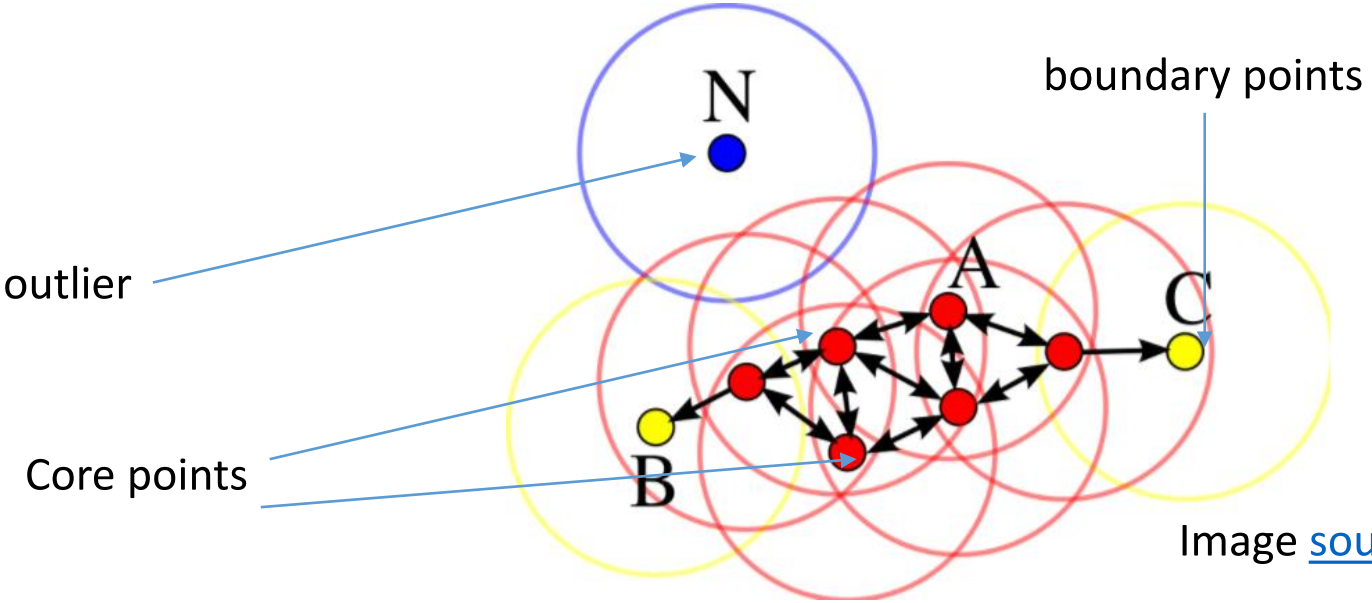


Image source

min_samples=4

DBSCAN-Algorithm

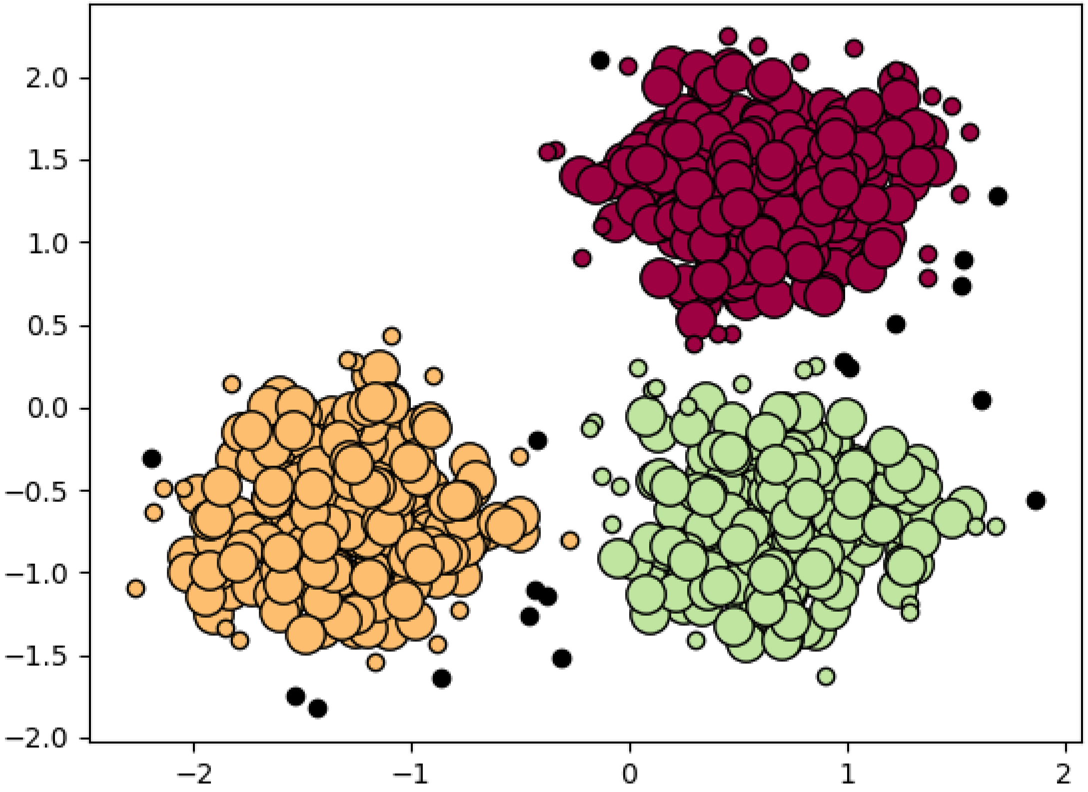
Input a data X, a positive real number ε and a positive integer **min_samples**

1. Find the ε neighbors graph.
2. Identify the core points with more than **min_samples** neighbors.

3. Find the connected components of core points on the neighbor graph, ignoring all non-core points.

4. Assign each non-core point to a nearby cluster if the cluster is an ε neighbor, otherwise assign it to noise.

DBSCAN-sklearn



Sklearn [example]

Sklearn [DBSCAN]

Comparison between DBSCAN and other clustering algorithms

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points

source

Part II
Graph Clustering

Scan Algorithm (A Structural Clustering Algorithm for Networks)

Scan algorithm is similar to DBSCAN algorithm. The only difference here is that we cluster the nodes of a graph instead Of the points in a point cloud.

Let G(V,E) be a graph. Let u, v be two vertices in V. Define the following similarity measure between the nodes u, v :

$$Sim(u, v) = \frac{|H(u) \cap H(v)|}{\sqrt{|H(u)||H(v)|}}$$

Scan Algorithm (A Structural Clustering Algorithm for Networks)

Scan algorithm is similar to DBSCAN algorithm. The only difference here is that we cluster the nodes of a graph instead Of the points in a point cloud.

Let G(V,E) be a graph. Let u, v be two vertices in V. Define the following similarity measure between the nodes u, v :

$$Sim(u,v) = \frac{|H(u) \cap H(v)|}{\sqrt{|H(u)||H(v)|}}$$

Here $H(v)$ *is the set of direct neighbors of* v : $H(v)$={u ∈ V|(u,v) ∈E} ∪ {v}.

Scan Algorithm (A Structural Clustering Algorithm for Networks)

Scan algorithm is similar to DBSCAN algorithm. The only difference here is that we cluster the nodes of a graph instead
Of the points in a point cloud.

Let G(V,E) be a graph. Let u, v be two vertices in V. Define the following similarity measure between the nodes u, v :

$$Sim(u, v) = \frac{|H(u) \cap H(v)|}{\sqrt{|H(u)||H(v)|}}$$

Here $H(v)$ *is the set of direct neighbors of* v : $H(v)$={u ∈ V|(u,v) ∈E} ∪ {v}.

The ε neighbor of a node v $N_{\varepsilon}(v)$ is given as the set of node in V whose similarity exceeds the value ε :

$$N_{\varepsilon}(v) = \{u \in H(v) \,|\, Sim(u, v) \geq \varepsilon\}$$

Scan Algorithm (A Structural Clustering Algorithm for Networks)

Using this notion of $N_{\varepsilon}(v)$ the same definitions that we defined on DBSCAN can be adapted on graph and in this way we Obtain a clustering for the nodes of the graph that is analogous to the DBSCAN on point clouds.

Scan Algorithm (A Structural Clustering Algorithm for Networks)

Using this notion of $N_{\varepsilon}(v)$ the same definitions that we defined on DBSCAN can be adapted on graph and in this way we Obtain a clustering for the nodes of the graph that is analogous to the DBSCAN on point clouds.

For instance, a vertex said to be a core vertex if its ε neighborhood as a cardinality at least min_samples.
In a similar fashion we can define the notion of reachability and other definitions we defined earlier.

Zahn’s algorithm

Zahn’s algorithm that we used to obtain a clustering algorithm on point cloud can be simply used to obtain a clustering algorithm on graphs as follows.

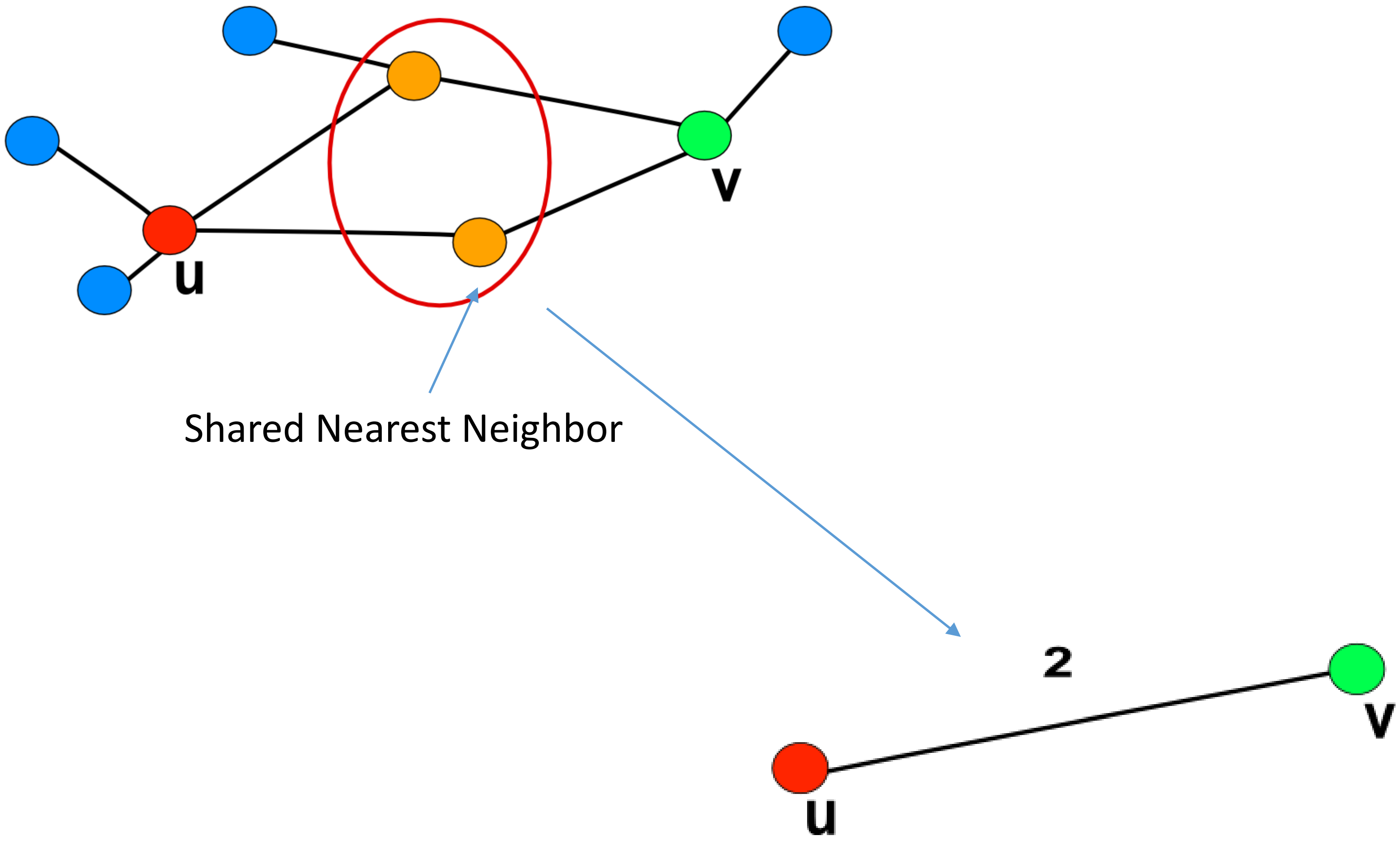
Suppose that we are given a set of a weighted graph G.

1. Construct the MST of G.
2. Remove the *inconsistent edges* to obtain a collection of connected components (clusters).
3. Repeat step (2) as long as the termination condition is not satisfied.

The connected components of the remaining forest are the clusters of the graph

In this case, an edge in the tree is called inconsistent if it has a length more than a certain given length L

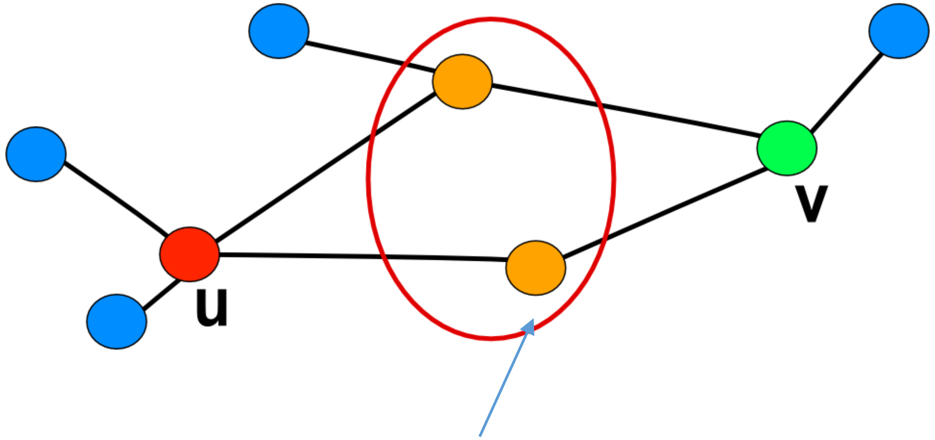
Shared Nearest Neighbor (SNN)



Shared Nearest Neighbor graph

Given a graph $G = G(V, E)$ $we\ can\ contruct\ a\ weighted\ graph\ called\ the\ SNN\ as\ follows$

Add the number of shared edges between every two vertices as a weight for the edge insert edge. If the number of shared edges is zero then do not insert an edge.



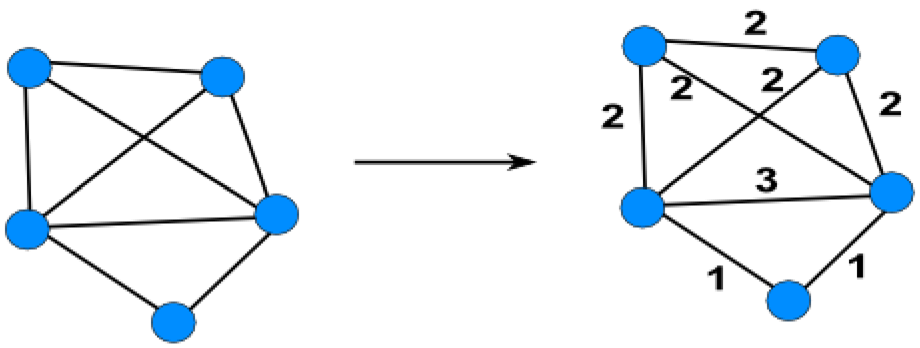
Shared Nearest Neighbor

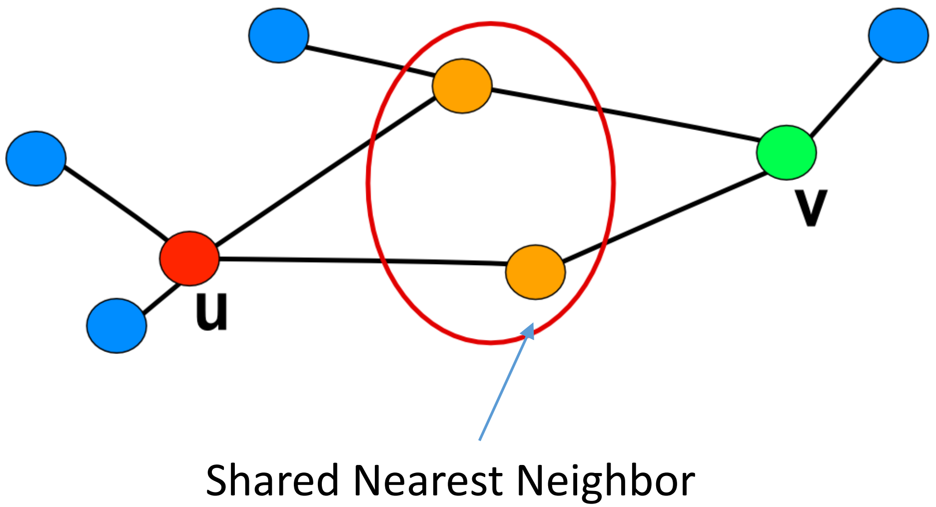
Shared Nearest Neighbor graph

Given a graph $G = G(V, E)$ $we\ can\ contruct\ a\ weighted\ graph\ called\ the\ SNN\ as\ follows$

Add the number of shared edges between every two vertices as a weight for the edge insert edge. If the number of shared edges is zero then do not insert an edge.

Example

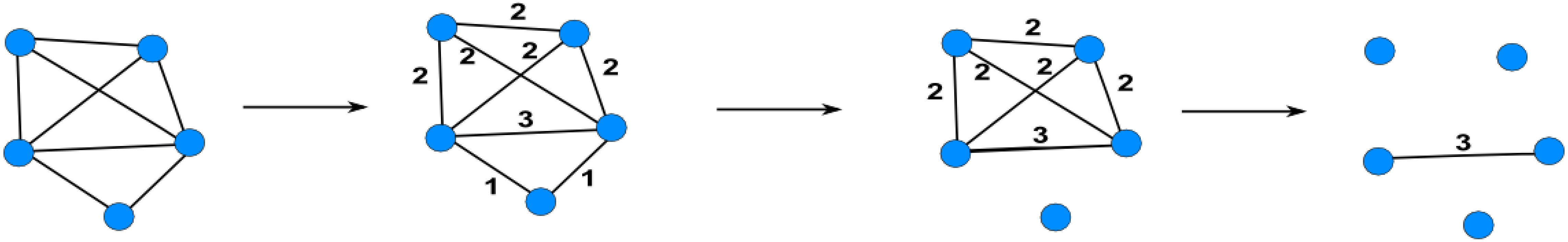




Shared Nearest Neighbor (SNN) Clustering

Input : a graph $G = G(V, E)$ *and a positive integer* τ

- Calculate the Shared Nearest Neighbor Graph of input graph G
- Removes edges from the SNN with weight less than τ



τ= 2 τ= 3