Clustering and Density-based Anomaly Detection

COMP90073
Security Analytics

Sarah Erfani, CIS

Semester 2, 2021

Outline

- Anomaly detection with clustering
- Density-Based Spatial Clustering (DBSCAN)
- Local Outlier Factor (LOF)

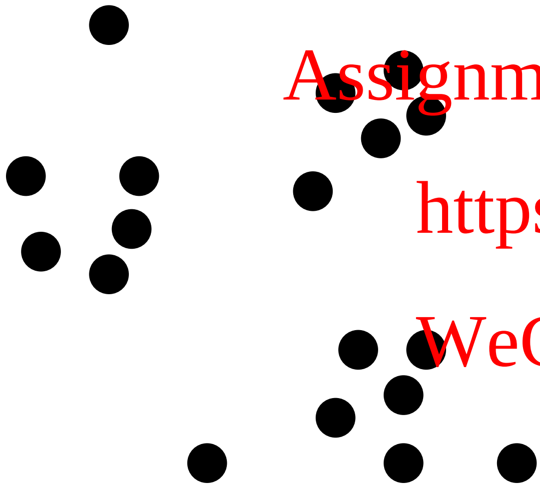
Using Clustering for Anomaly Detection

- **Advantages:**
  - They can detect anomalies without requiring any *labelled* data.
  - They work for many data *types*.
  - Clusters can be regarded as *summaries* of the data.
  - Once the clusters are obtained, clustering-based methods need only compare any object against the clusters to determine whether the object is an anomaly.
  - Test process is typically fast and efficient because the number of clusters is usually small compared to the total number of objects small.

- **Weakness:**
  - Their effectiveness depends highly on the clustering method used. Such methods may not be optimized for outlier detection.
  - They are often costly for large data sets, which can serve as a bottleneck.

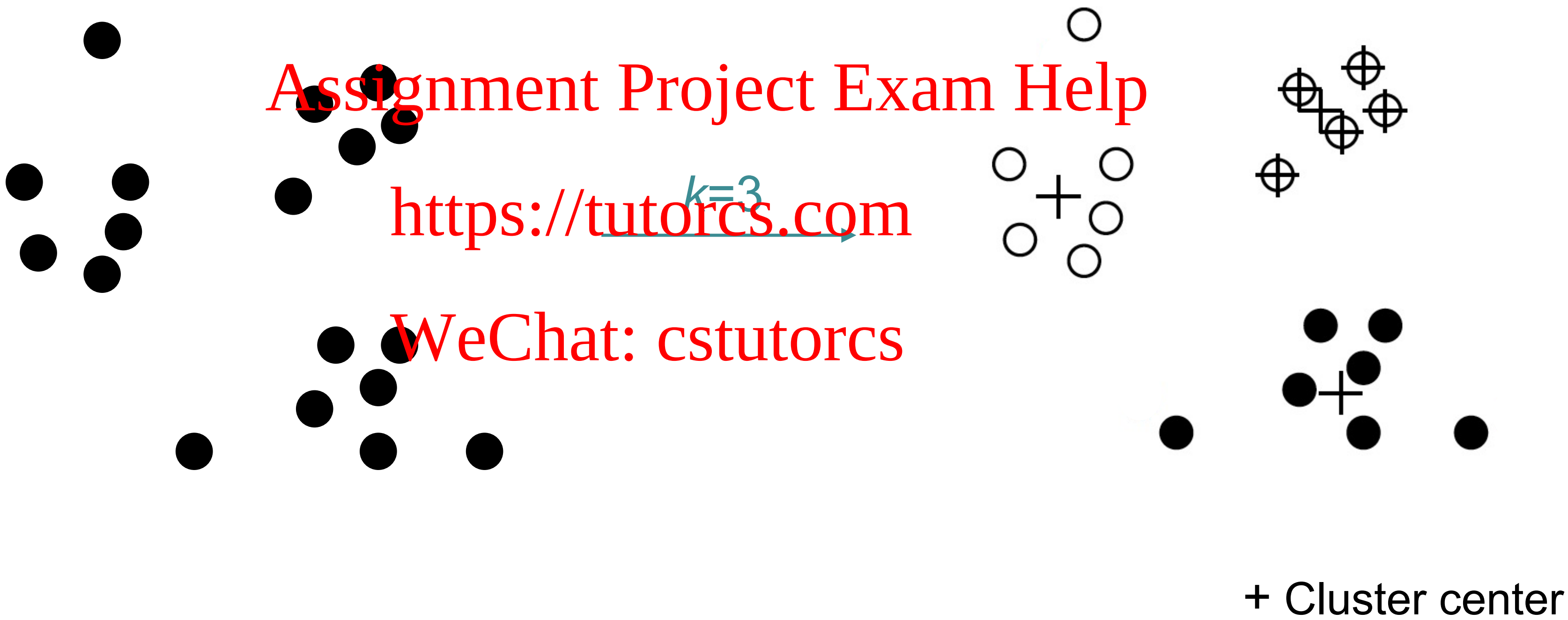
Clustering-based Anomaly Detection

- *k*-means clustering



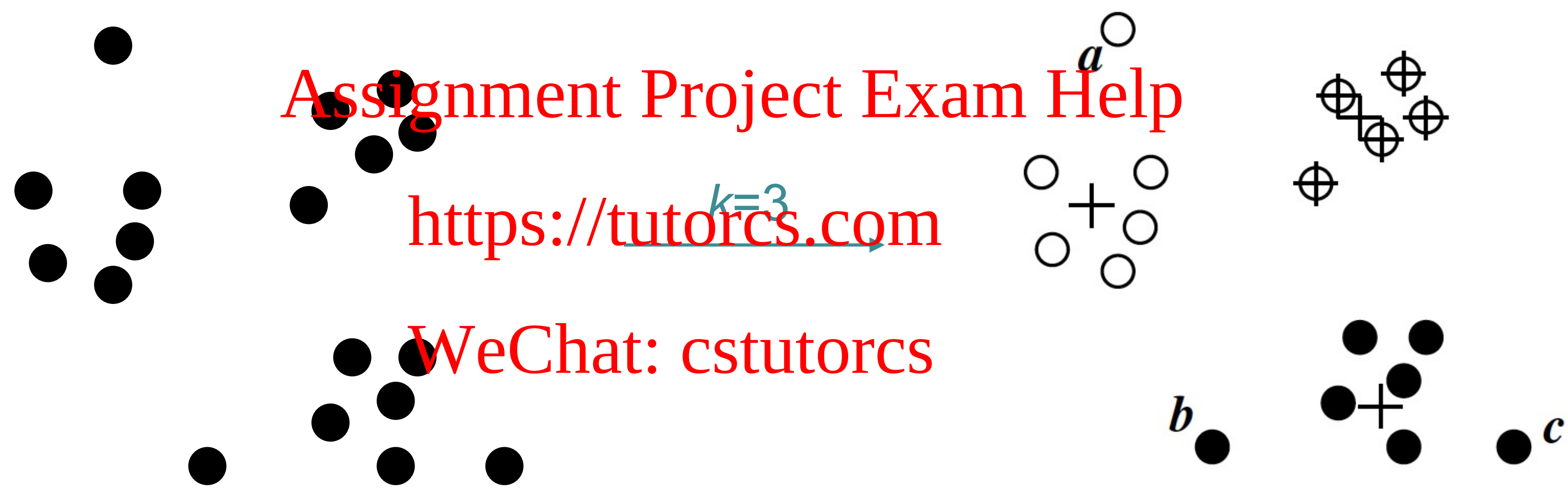
Clustering-based Anomaly Detection

- *k*-means clustering



Clustering-based Anomaly Detection

- *k*-means clustering

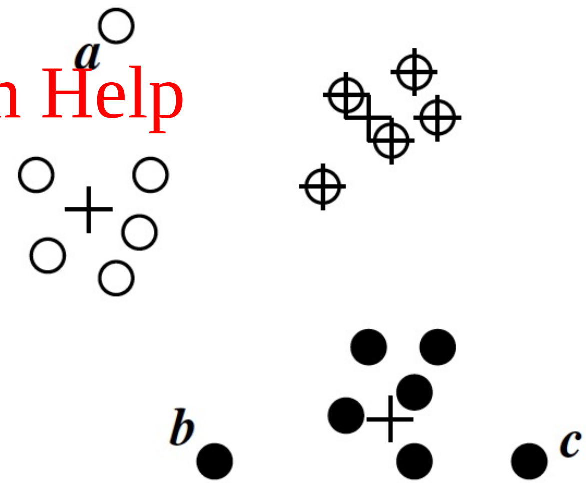


Clustering-based Anomaly Detection

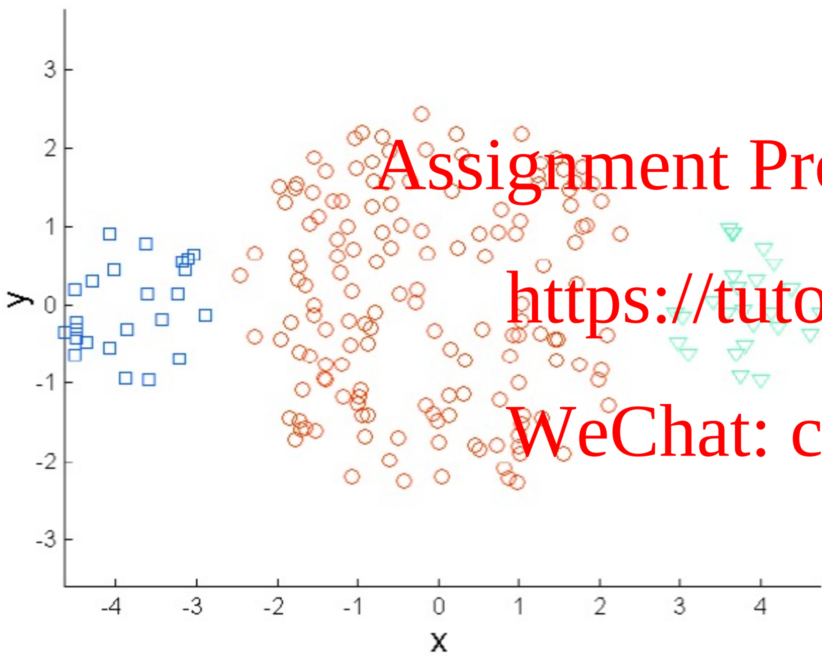
- Assign an anomaly score to each object according to the distance between the object and the centre of closest cluster.

- $Anomaly\ score(p_j) = \frac{dist(p_j, c_o)}{\frac{1}{n}\sum_i dist(p_i, c_o)}$

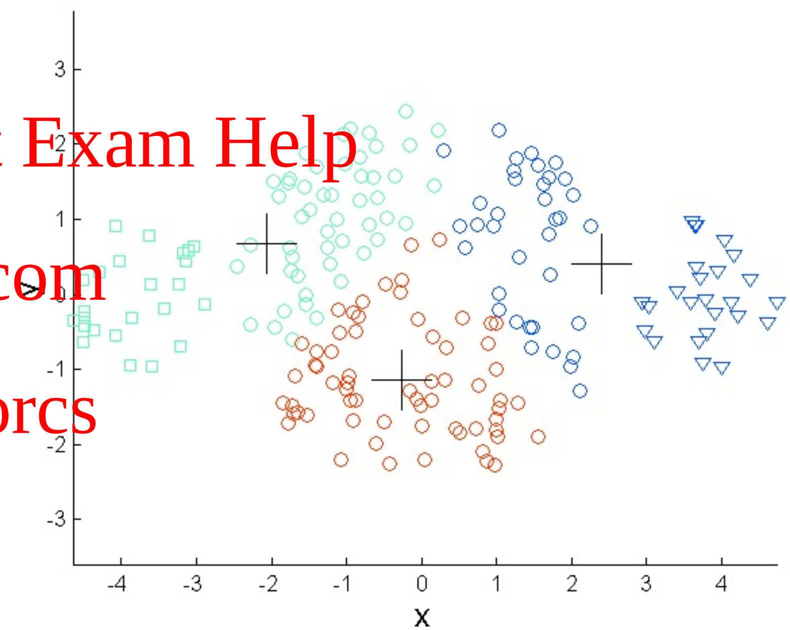
- Anomalies (***a***,***b***,***c***) are far from the clusters to which they are closest (with respect to the cluster centres).



Limitations of *k*-means: Differing Size

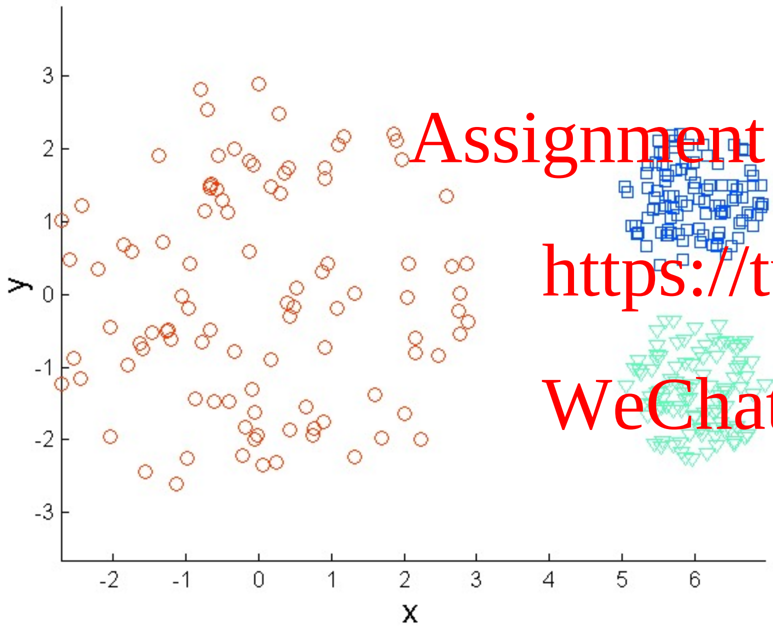


Original Points

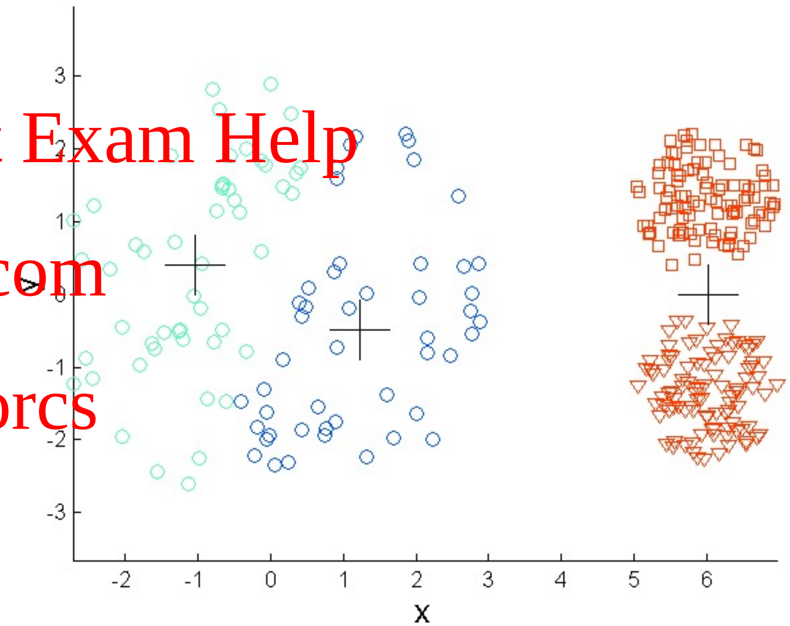


K-means (3 Clusters)

Limitations of *k*-means: Differing Density

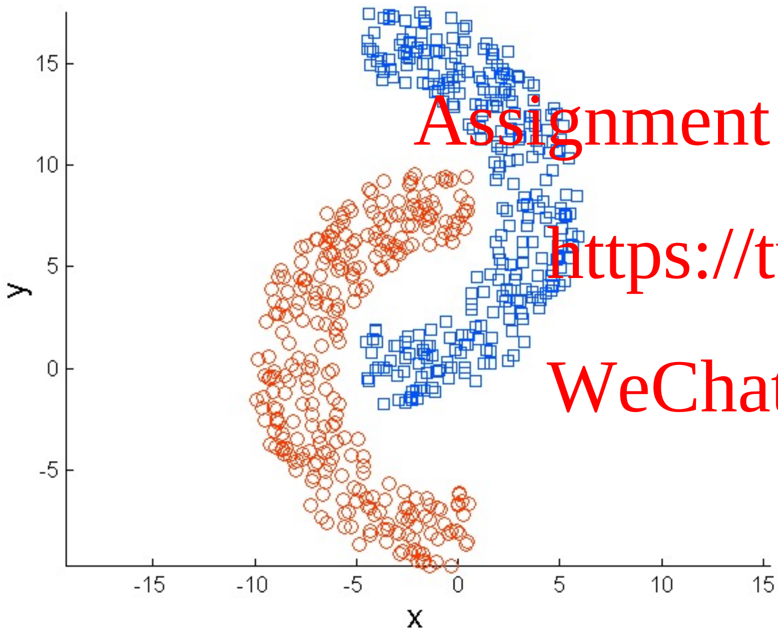


Original Points

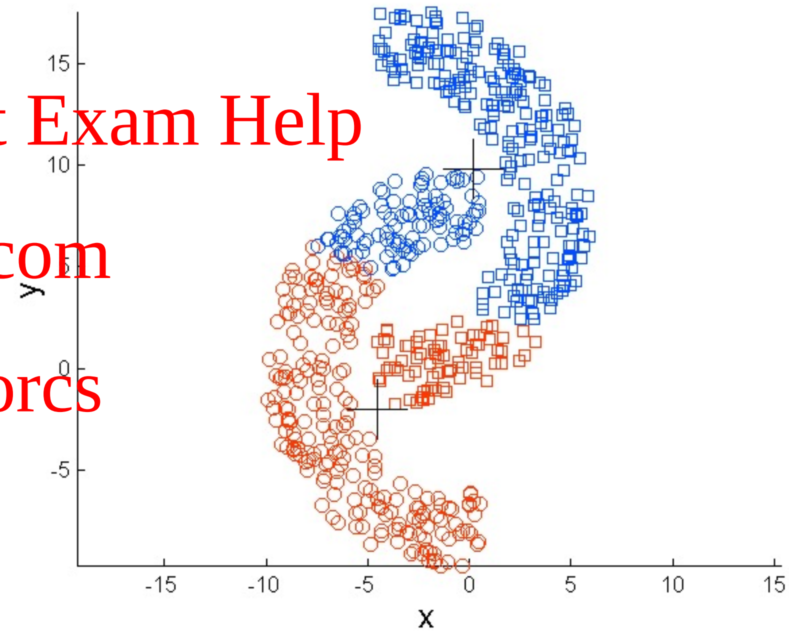


K-means (3 Clusters)

Limitations of *k*-means: Non Globular Shape



Original Points



K-means (2 Clusters)

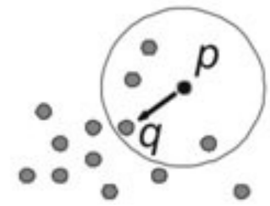
Density based Clustering

- **Density-based clustering:** Model clusters as dense regions in the data space, separated by sparse regions, which can discover clusters of non-spherical shape and avoid outliers.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [2]

- **Objective:** Identify dense regions, which can be measured by the number of objects close to a given point.
  - *Finds core objects*, that is, objects that have dense neighbourhoods. It connects core objects and their neighbourhoods to form dense regions as clusters.

- Important Questions:
  - How do we measure density?
  - What is a dense region?



Eps = 1cm
MinPts = 4

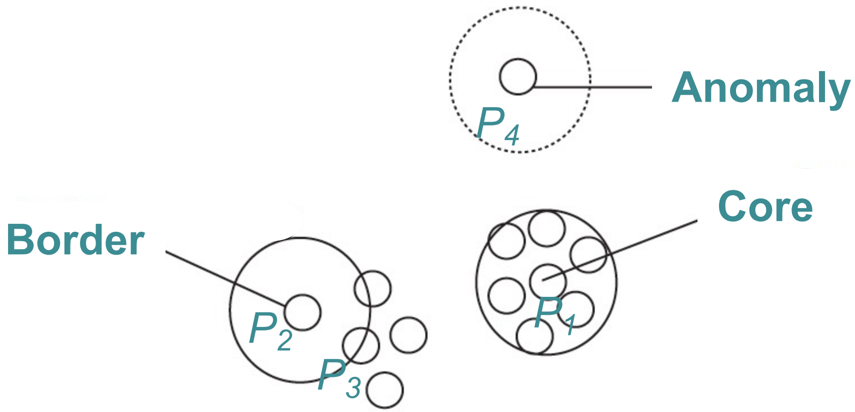
<u>**Parameters**</u>:

- **Density at point *p***: Number of points within a circle of radius **Eps**
- **Dense Region**: A cluster with radius Eps that contains at least **MinPts** points

DBSCAN – Concepts

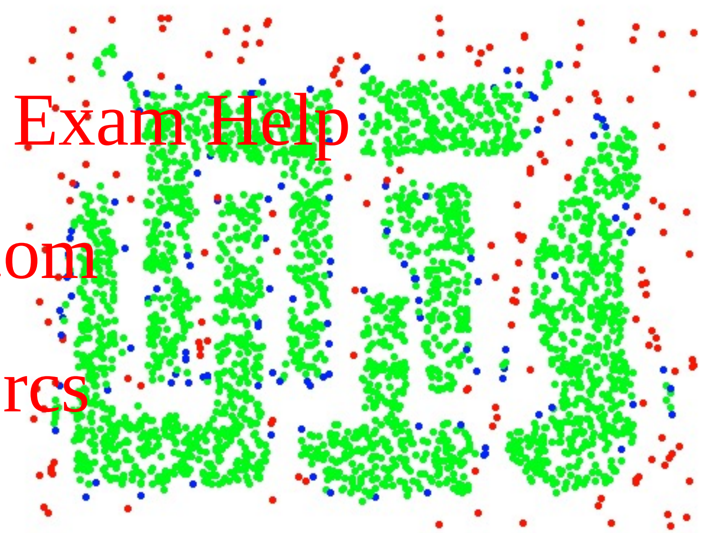
DBSCAN defines different classes of points:

- **Core point:** A point with at least MinPts points within its Eps-neighbourhood (including *itself*).
- **Border point:** A point with fewer points than MinPts in the Eps-neighbourhood, but is in the neighbourhood of a core point.
- **Anomaly (outlier) point:** a point which is neither core nor border.
- E.g., MinPts = 4



DBSCAN – Core, Border, and Anomaly

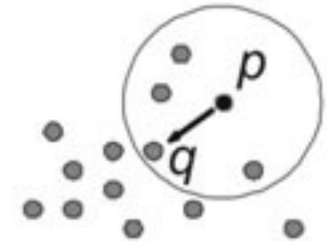
Original data



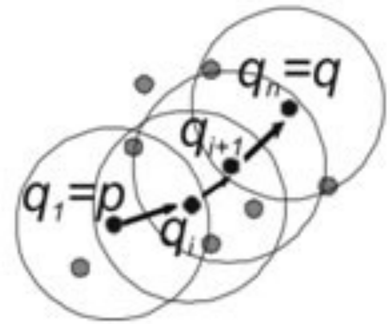
Point types: core, border and anomaly

DBSCAN – Concepts

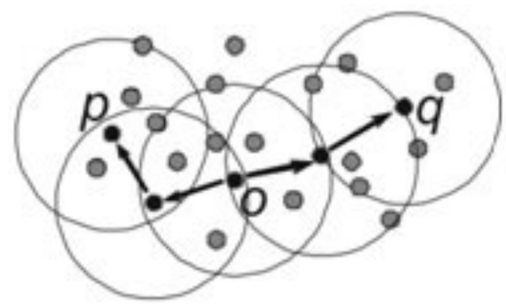
- **Directly Density-reachable:** Point q is directly density-reachable from p (w.r.t. Eps and MinPts) if p is a *core point*, and q is within the Eps-neighbourhood of p.



- **(Indirectly) Density-reachable:** Point q is density-reachable from p (w.r.t. Eps and MinPts) if there is a chain of points $q_1 \ldots q_n$, $q_1 = p$, $q_n = q$, such that q_{i+1} is directly density-reachable from q_i.

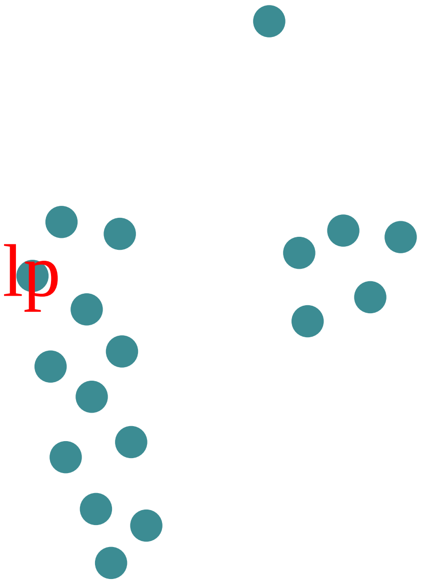


- **Density-connected:** Point q is density-connected to a point p (w.r.t. Eps and MinPts) if there is a point *o* such that both p and q are density reachable from *o* (w.r.t. Eps and MinPts).



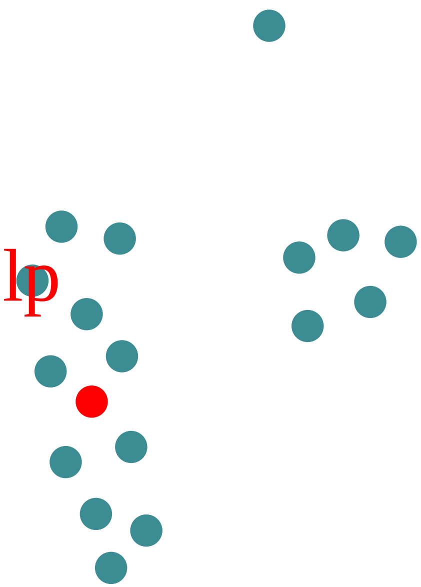
DBSCAN Algorithm

- Randomly select an unvisited object *p*



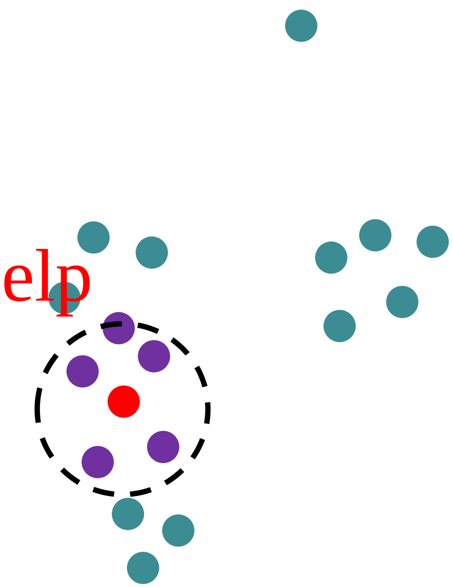
DBSCAN Algorithm

- Randomly select an unvisited object *p*



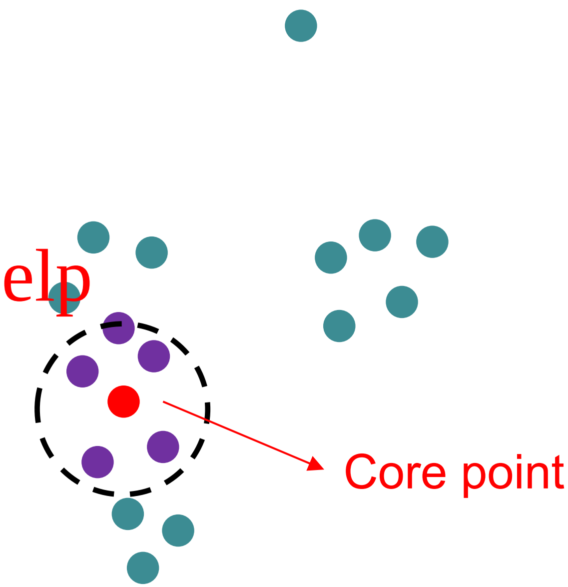
DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and MinPts (e.g., 5)



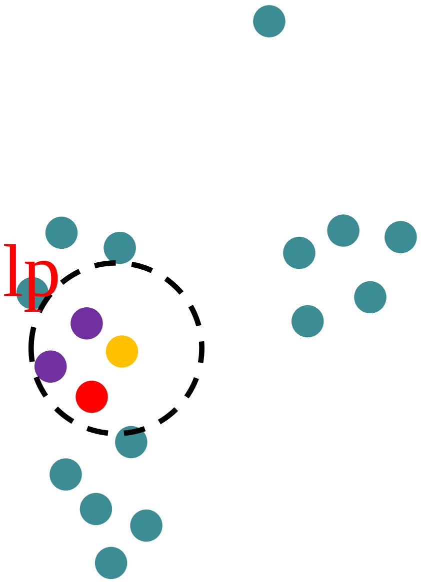
DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and MinPts (e.g., 5)
- If *p* is a core point, create a cluster



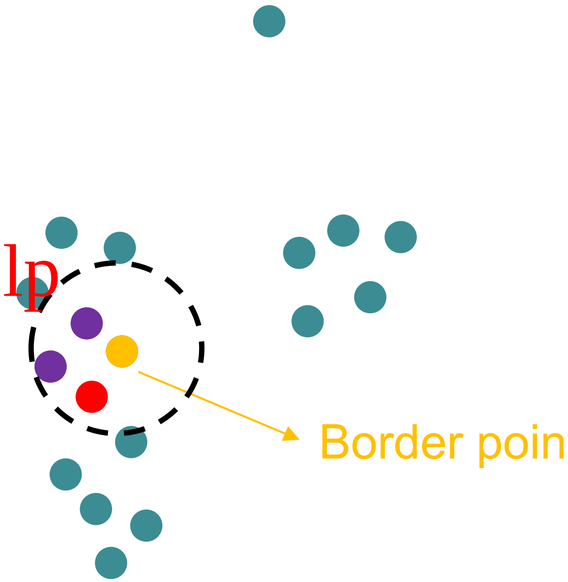
DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and MinPts (e.g., 5)
- If *p* is a core point, create a cluster



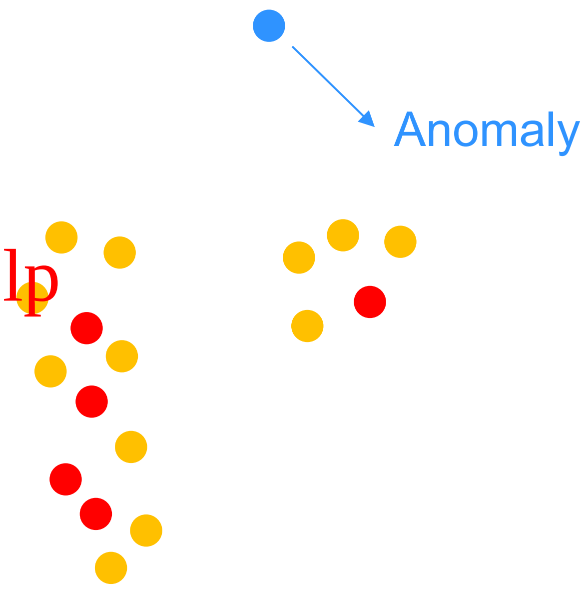
DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and MinPts (e.g., 5)
- If *p* is a core point, create a cluster
- If *p* is a boarder point, no points are density-reachable from *p* and DBSCAN visits the next point of the database



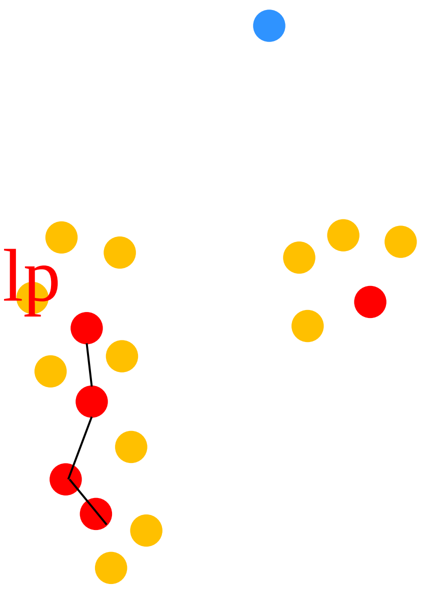
DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and Mint
- If *p* is a core point, create a cluster
- If *p* is a boarder point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Repeat the above steps until all data points have been visited



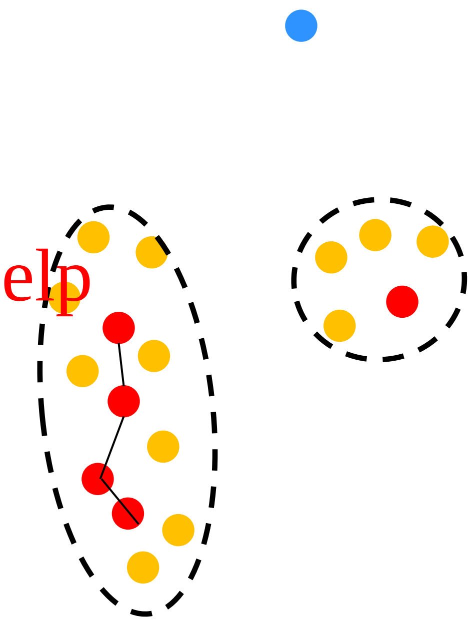
DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and Mint
- If *p* is a core point, create a cluster
- If *p* is a boarder point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Repeat the above steps until all data points have been visited



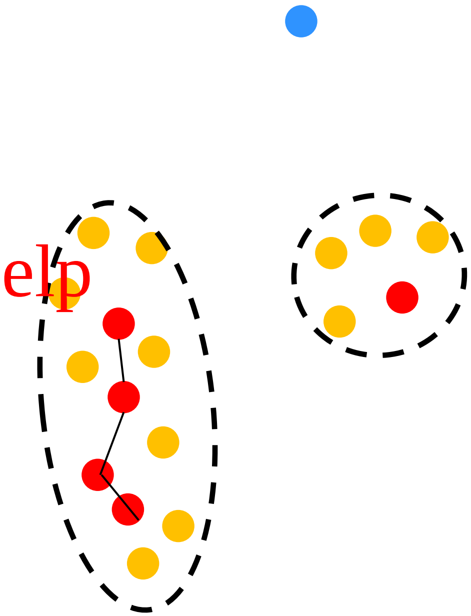
DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and Mint
- If *p* is a core point, create a cluster
- If *p* is a boarder point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Repeat the above steps until all data points have been visited



DBSCAN Algorithm

- Randomly select an unvisited object *p*
- Retrieve all points density-reachable from *p* w.r.t. Eps and Mint
- If *p* is a core point, create a cluster
- If *p* is a boarder point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Repeat the above steps until all data points have been visited

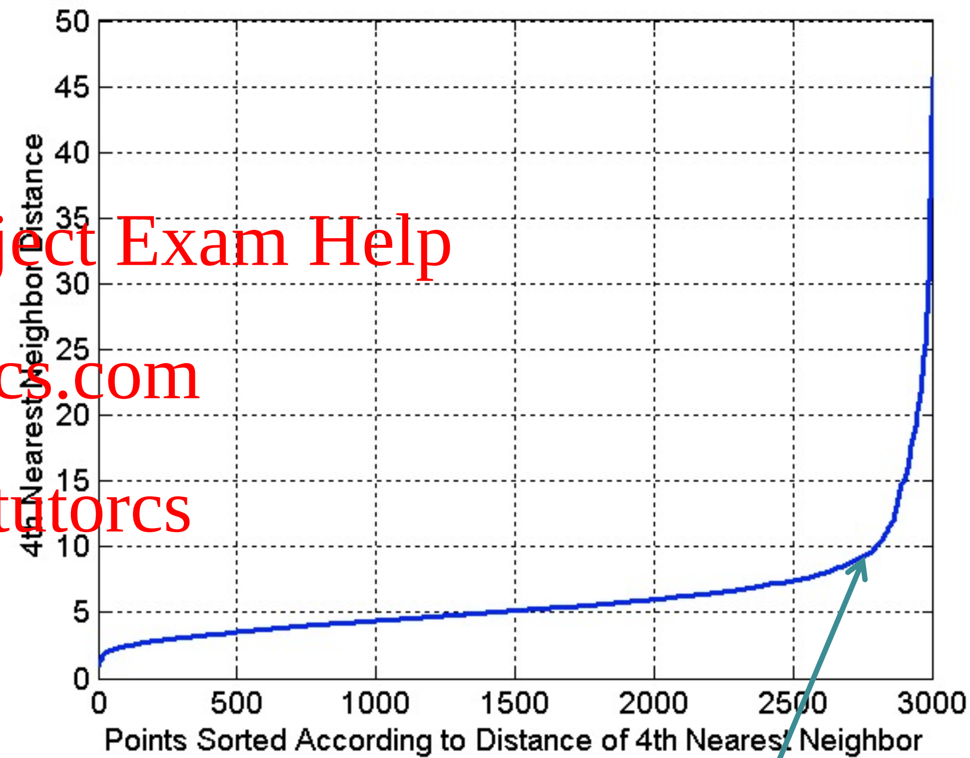


<u>Computational Complexity</u>:

- $O(n^2)$, where *n* is the number of samples.
- If a spatial index is used, $O(n \log n)$.

Determining Eps and MinPts

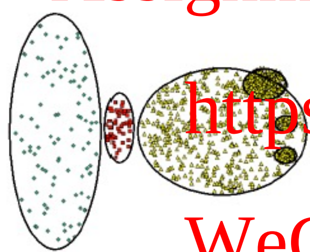
- Idea is that for points in a cluster, their k^{th} nearest neighbours are at roughly the same distance
- Noise points have the k^{th} nearest neighbour at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbour
- Find the distance *d* where there is a "knee" in the curve
  - Eps = *d*, MinPts = *k*
- Demo: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/



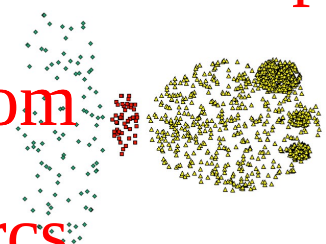
Eps=7~10
MinPts=4

Advantages and Disadvantages of DBSCAN

- **Advantages:**
  - Resistant to Noise
  - Can handle clusters of different shapes and sizes
- **Disadvantages:**
  - Varying densities

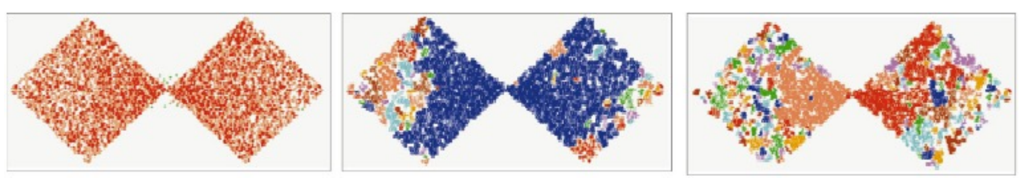


Original data



MinPts=4, Eps=9.75

  - Sensitive to parameter setting



Eps = 5 Eps = 3.5 Eps = 3

  - High-dimensional data

Using Clustering for Anomaly Detection

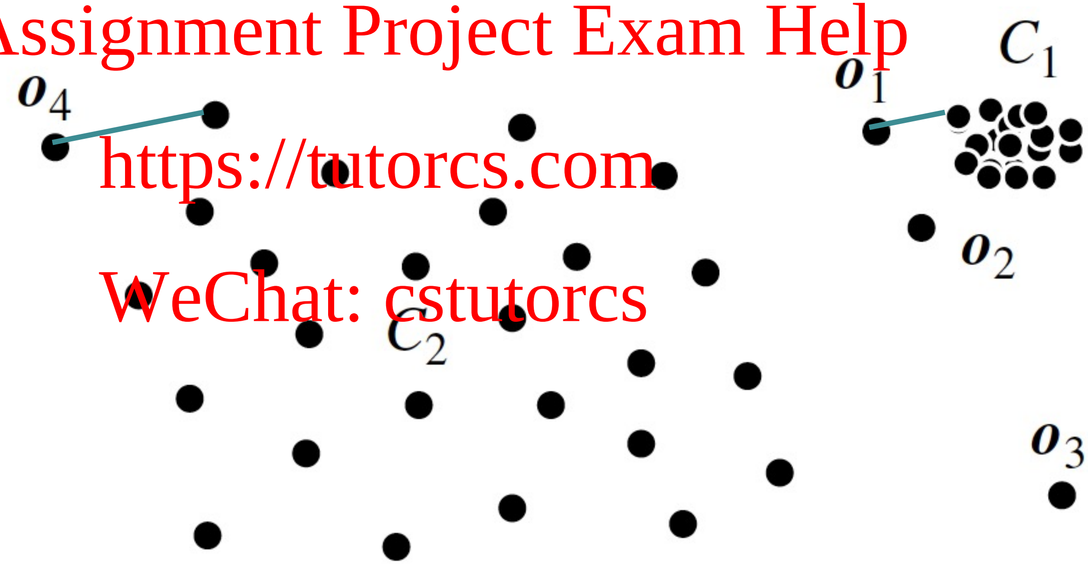
- **Advantages:**
  - They can detect anomaly without requiring any labelled data.
  - They work for many data types.
  - Clusters can be regarded as summaries of the data.
  - Once the clusters are obtained, clustering-based methods need only compare any object against the clusters to determine whether the object is an anomaly.
  - Test process is typically fast and efficient because the number of clusters is usually small compared to the total number of objects small.

- **Weakness:**
  - Their effectiveness depends highly on the clustering method used. Such methods may not be optimized for anomaly detection.
  - They are often costly for large data sets, which can serve as a bottleneck.

Local Proximity-based Outliers

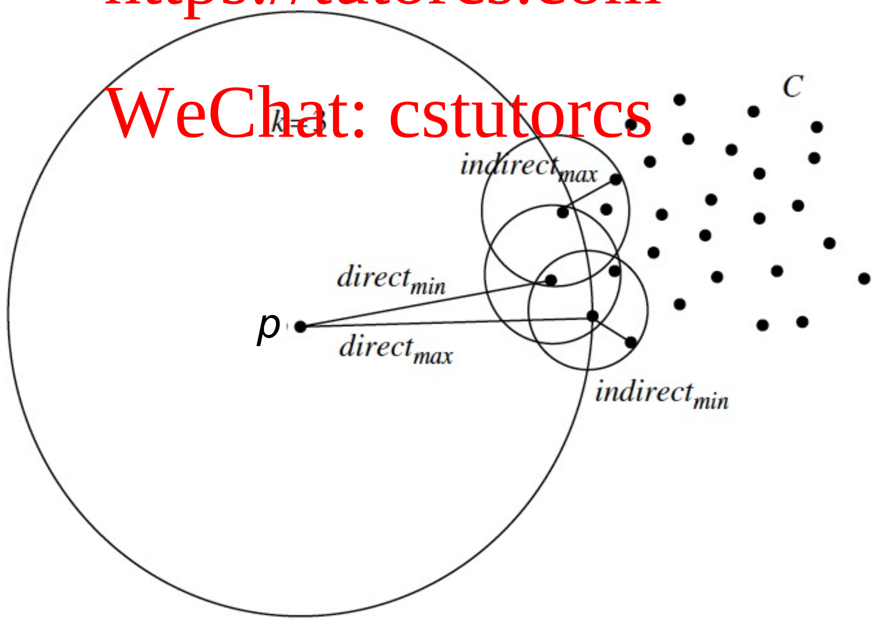
In the following figure which of the following instances are anomalies?

- o_1?
- o_2?
- o_3?
- o_4?



Local Outlier Factor (LOF)

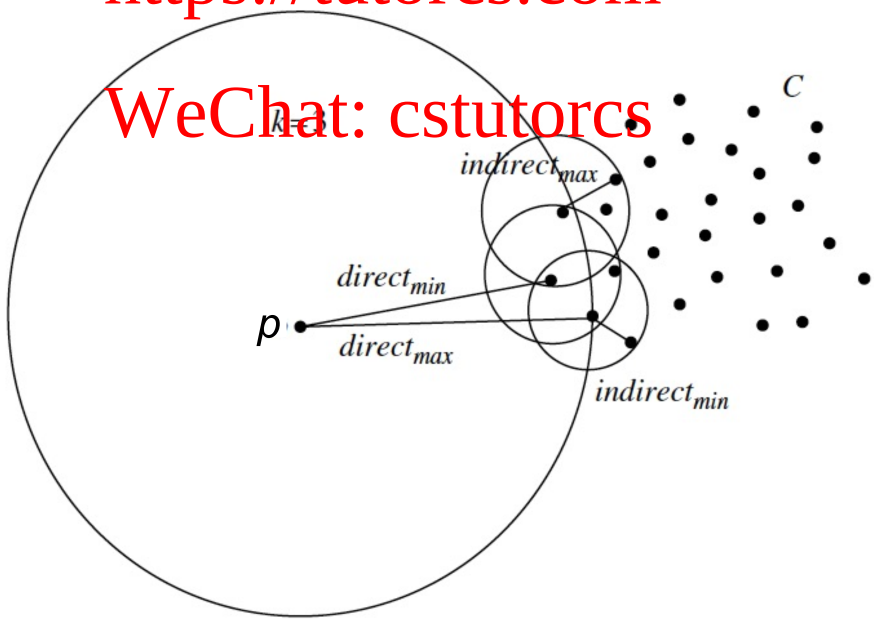
- **Objective:** Quantify the *relative* density about a particular data point.
- **Intuition:** The anomalies should be more *isolated* compared to "normal" data points.
- LOF uses the relative density of an object against its neighbours to indicate the degree to which an object is an anomaly.



THE UNIVERSITY OF MELBOURNE

K-Distance

- $kdist$: distance between p and its k^{th} NN
- Meta-heuristic: The $kdist$ gives us a notion of “volume”
- The more isolated a point is, the larger its $kdist$



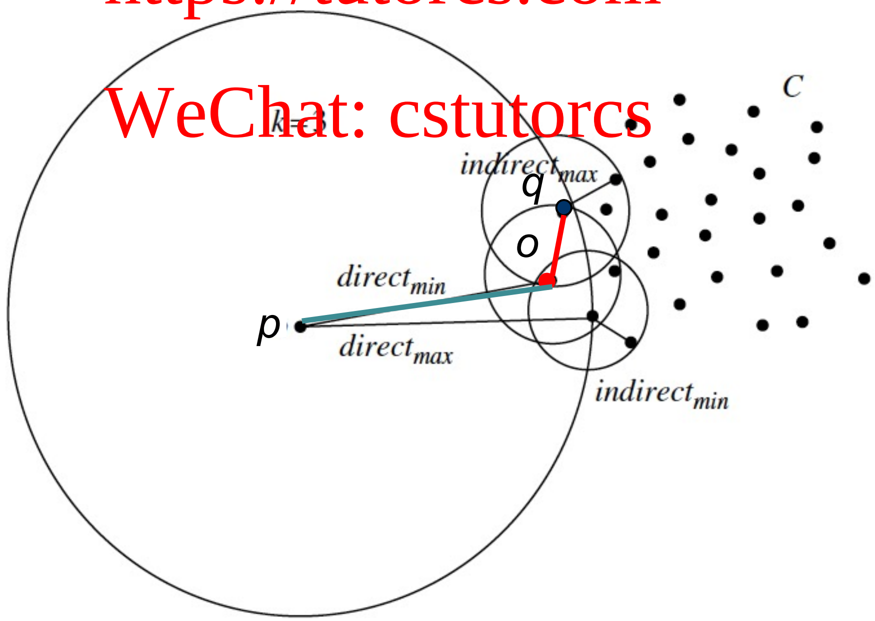
Reachability Distance

- **Reachability Distance** of p with respect to o:

Not symmetric

$$reachdist_k(p,o) = \max\{kdist(o), dist(p,o)\}$$

- **Intuition:** "Do your close neighbours see you as one of their close neighbours"



Local Reachability Density

- **Local Reachability Density** of p:

$$lrd_k(p) = \left(\frac{1}{k} \sum_{o \in N_{(p,k)}} reachdist_k(p, o) \right)^{-1}$$

k nearest neighbours of *p*

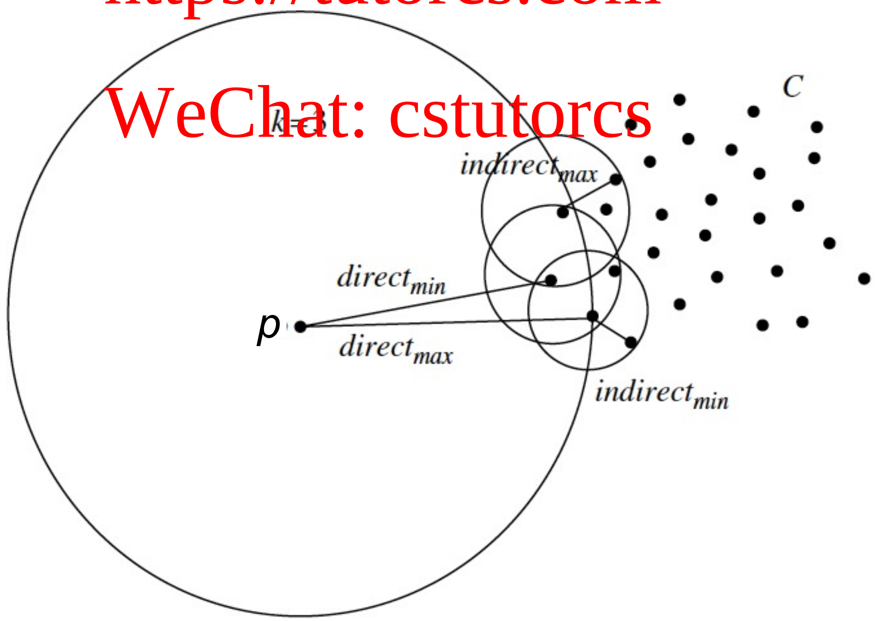
- **Intuition:** How far we have to travel from our point to reach the next point or cluster of points.
  - The lower it is, the less dense it is, the longer we have to travel.

THE UNIVERSITY OF MELBOURNE

LOF

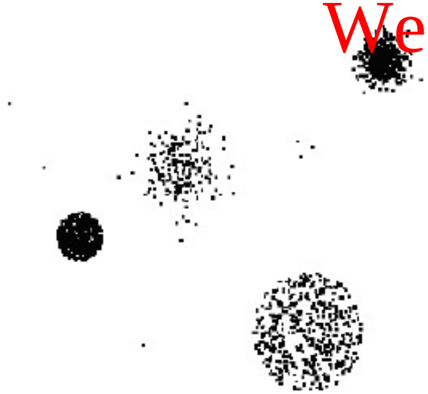
- **LOF** of an object p is the average of the ratio of local reachability of p and those of o's k-nearest neighbours
- The anomalies are coming from less dense area, so the ratio is higher for anomalies

$$LOF_k(p) = \frac{1}{k} \sum\nolimits_{o \in N_{(p,k)}} \frac{lrd_k(o)}{lrd_k(p)}$$

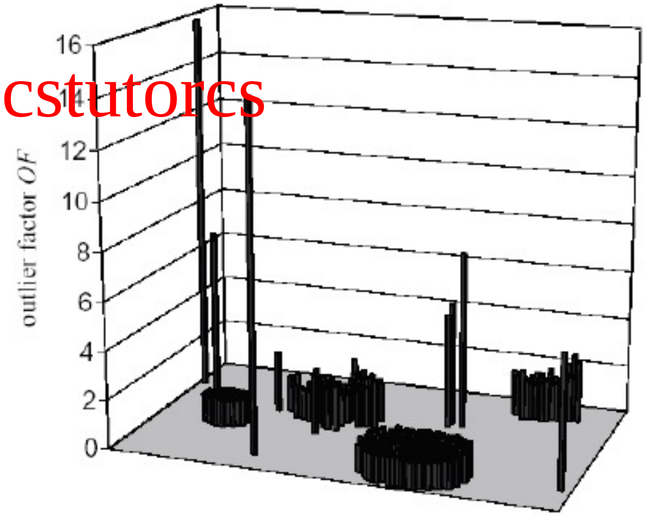


Interpretation of LOF Score

- The lower the local reachability density of p, and the higher the local reachability density of the kNN of p, the higher LOF
  - $LOF_k(p) \sim 1$: Comparable density to neighbours,
  - $LOF_k(p) < 1$: Higher density than neighbours
  - $LOF_k(p) > 1$: Lower density than neighbours



Data set



LOFs

LOF – Example

- Consider the following 4 data points:

  a(0, 0), b(0, 1), c(1, 1), d(3, 0)

- Calculate the LOF for each point and show the top 1 outlier, set k = 2 and use Manhattan Distance.

LOF – Example

Step 1: Calculate all the distances between each two data points

- There are 4 data points:

a(0, 0), b(0, 1), c(1, 1), d(3, 0)

dist(a, b) = 1

dist(a, c) = 2

dist(a, d) = 3

dist(b, c) = 1

dist(b, d) = 3+1=4

dist(c, d) = 2+1=3

LOF – Example

Step 2: Calculate $dist_k$ (o), distance between o and its k-th NN (k-th nearest neighbour)

k=2:

$dist_2$ (a) = dist(a, c) = 2 (c is the 2nd nearest neighbour)

$dist_2$ (b) = dist(b, a) = 1 (a/c is the 2nd nearest neighbour)

$dist_2$ (c) = dist(c, a) = 2 (a is the 2nd nearest neighbour)

$dist_2$ (d) = dist(d, a) = 3 (a/c is the 2nd nearest neighbour)

LOF – Example

Step 3: Calculate all the N_k (p), k-distance neighborhood of p, $N_k(p) = \{p' | p' \text{ in } D, \text{dist}(p, p') \leq \text{dist}_k(p)\}$

$N_2(a) = \{b, c\}$

$N_2(b) = \{a, c\}$

$N_2(c) = \{b, a\}$

$N_2(d) = \{a, c\}$

LOF – Example

Step 4: Calculate all the $lrd_k(p)$

For example:

$$lrd_k(a) = \frac{\|N_2(a)\|}{reachdist(a,b) + reachdist(a,c)}$$

- $\|N_2(a)\| = \|\{b, c\}\| = 2$
- $reachdist(a,b) = \max\{dist_2(b)dist(b,a)\} = \max\{1{,}1\} = 1$
- $reachdist(a,c) = \max\{dist_2(c)dist(c,a)\} = \max\{2{,}2\} = 2$

$$lrd_k(a) = \frac{2}{1+2} = 0.67$$

LOF – Example

Step 4: Calculate all the $lrd_k(p)$

Similarly,

- $lrd_k(b) = \frac{\|N_2(b)\|}{reachdist(b,a)+reachdist(b,c)} = \frac{2}{2+2} = 0.5$

- $lrd_k(c) = \frac{\|N_2(c)\|}{reachdist(c,b)+reachdist(c,a)} = \frac{2}{1+2} = 0.67$

- $lrd_k(d) = \frac{\|N_2(d)\|}{reachdist(d,a)+reachdist(d,c)} = \frac{2}{3+3} = 0.33$

THE UNIVERSITY OF MELBOURNE

LOF – Example

Step 5: calculate all the $LOF_k(p)$

- $LOF_2(a) = \big(lrd_2(b) + lrd_2(c)\big)\times\big(reachdist_2(a,b) + reachdist_2(a,c)\big) =$
$(0.5 + 0.67)\times(1 + 2) = 3.51$

- $LOF_2(b) = \big(lrd_2(a) + lrd_2(c)\big)\times\big(reachdist_2(b,a) + reachdist_2(b,c)\big) =$
$(0.67 + 0.67)\times(2 + 2) = 5.36$

- $LOF_2(c) = \big(lrd_2(b) + lrd_2(a)\big)\times\big(reachdist_2(c,b) + reachdist_2(c,a)\big) =$
$(0.5 + 0.67)\times(1 + 2) = 3.51$

- $LOF_2(d) = \big(lrd_2(a) + lrd_2(c)\big)\times\big(reachdist_2(d,a) + reachdist_2(d,c)\big) =$
$(0.67 + 0.67)\times(3 + 3) = 8.04$

LOF – Example

Step 6: Sort all the $LOF_k(p)$

The sorted order is:

- $LOF_2(d) = 8.04$
- $LOF_2(d) = 5.36$
- $LOF_2(d) = 3.51$
- $LOF_2(d) = 3.51$

Obviously, top 1 anomaly is point *d*

THE UNIVERSITY OF MELBOURNE

LOF – Properties

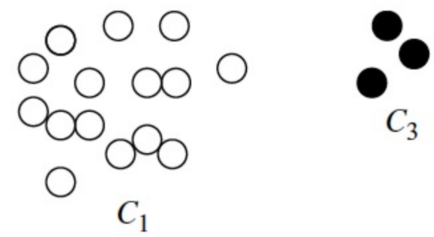
- LOF captures a *local anomaly* whose local density is relatively low comparing to the local densities of its *k*NN
- Outputs a *scoring* (assigns an LOF value to each point)
- Choice of *k* specifies the reference set
- Originally implements a local approach (resolution depends on the user's choice for *k*)

Cluster-based Local Outlier Factor (CBLOF)[1]

1) Find clusters in a data set (using k-means)

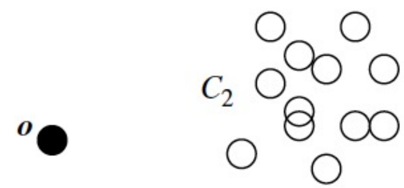
2) Sort them according to decreasing size.

   – Any cluster that contains at least a percentage (e.g., 90%) of the data set is considered a “large cluster.” The remaining clusters are referred to as “small clusters.”



3) To each data point, assign a cluster-based local outlier factor (CBLOF), which computed as the *product of the cluster’s size and the similarity between the point and the cluster*.

   – For a point belonging to a small cluster, its CBLOF is calculated as the product of the size of the small cluster and the similarity between the point and the closest large cluster.



Summary

- What are the advantages of clustering for anomaly detection?
- How distance and density based clustering perform differently?
- How to identify local anomalies?
- How to identify group anomalies?

Next: Anomaly Detection in Evolving Data Streams

References

1. Jiawei Han, Micheline Kamber, Jian Pei, “Data Mining: Concepts and Techniques”, 3rd ed, 2012. Chapters 10.4 and 12

2. Martin Ester , Hans-Peter Kriegel , Jörg Sander , Xiaowei Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.”, KDD, 1996.

3. Density-Based Clustering http://www.cse.buffalo.edu/faculty/azhang/cse601/density-based.ppt