Anomaly/Outlier detection methods

What are outliers?

- **Outlier**: A data object that **deviates significantly** from the normal objects as if it were **generated by a different mechanism**
  - Eg: Unusual credit card purchases, sports: Michael Jordan, Wayne Gretzky, ...
- Outliers are different from the noise data
- Outliers are interesting: It violates the mechanism that generates the normal data
- Applications:
  - Credit card fraud detection
  - Telecom fraud detection
  - Customer segmentation
  - Medical analysis

Challenges of Outlier Detection

- Modeling normal objects and outliers properly
  - Hard to enumerate all possible normal behaviors in an application
  - The border between normal and outlier objects is often a gray area
- Application-specific outlier detection
  - Choice of distance measure among objects and the model of relationship among objects are often application-dependent
  - E.g., clinic data: a small deviation could be an outlier; while in marketing analysis, larger fluctuations
- Handling noise in outlier detection
  - Noise may distort the normal objects and blur the distinction between normal objects and outliers. It may help hide outliers and reduce the effectiveness of outlier detection

DBSCAN Algorithm

DBSCAN

- DBSCAN is a density-based algorithm.
- DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise
- Density based Clustering locates regions of high density that are separated from one another by regions of low density
  - Density = number of points within a specified radius (Eps)

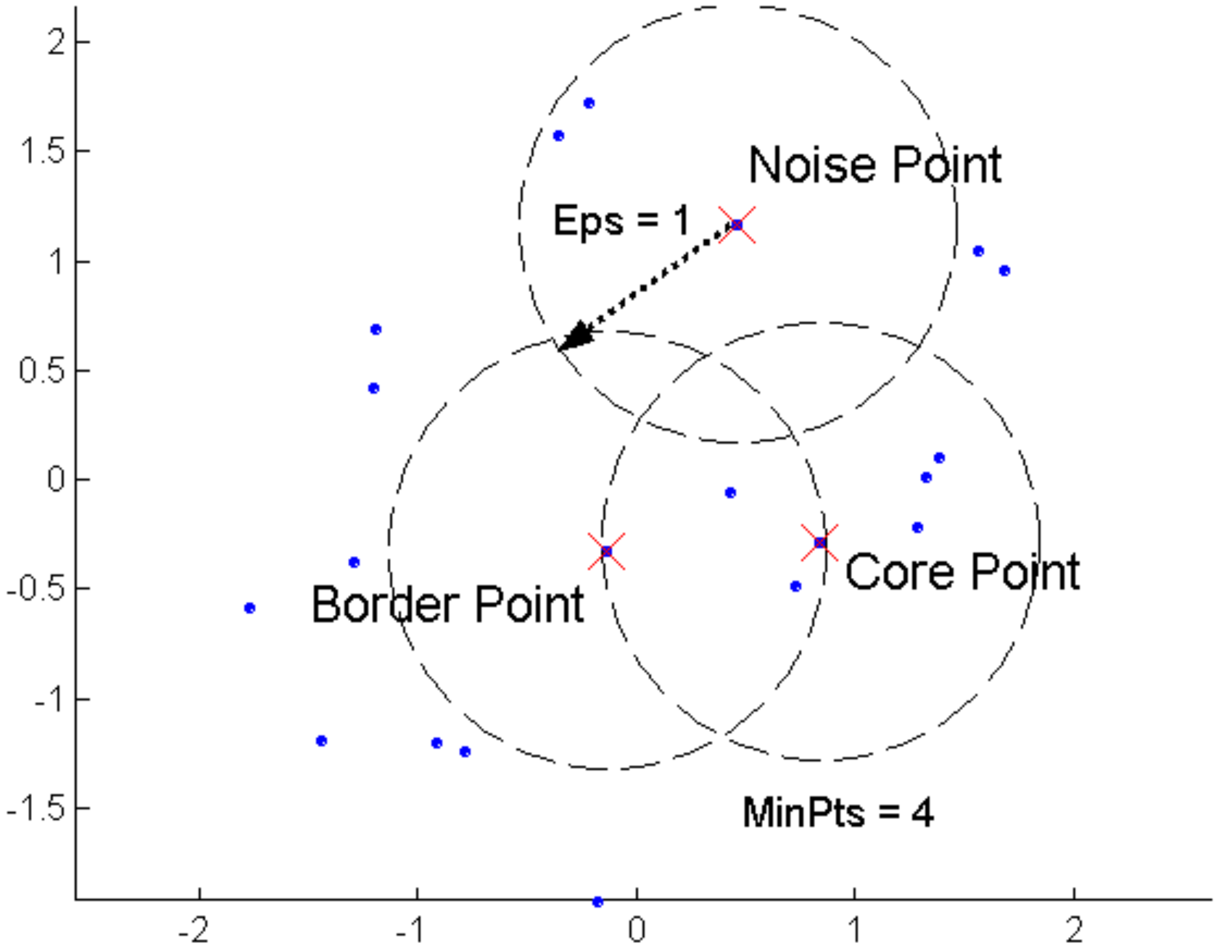
Terms used in DBSCAN

- A point is a ***core point*** if it has more than a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
- A ***border point*** has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A ***noise point*** is any point that is not a core point or a border point.

DBSCAN

- Any two core points that are within a distance Eps of one another are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are highlighted

DBSCAN: Core, Border and Noise Points



Parameters

- minPts
  - minPts = 1 does not make sense since every point is already its own cluster
  - minPts should usually be at least 3. If a larger value is possible, it is better.
  - Larger the dataset size, the higher minPts value
- Eps
  - If Eps is chosen too small, a large part of the dataset will not be clustered.
  - If Eps is chosen too big, a large part of the dataset will be in the same cluster.
  - In general, smaller values of Eps are preferred.

DBSCAN Algorithm

Given a set of points $S=\{x_1,x_2,\ldots,x_n\}$ Є R^D

1. Choose values for Minpts > 0 and Eps > 0

2. $A_i=\{x$ Є $S : d(x_i,x) <= Eps\}$; $i=1,2\ldots,n$

3. If $|A_i|<$Minpts ignore the point

4. Take union of A_i and A_j if $A_i \cap A_j$ ‡ Φ

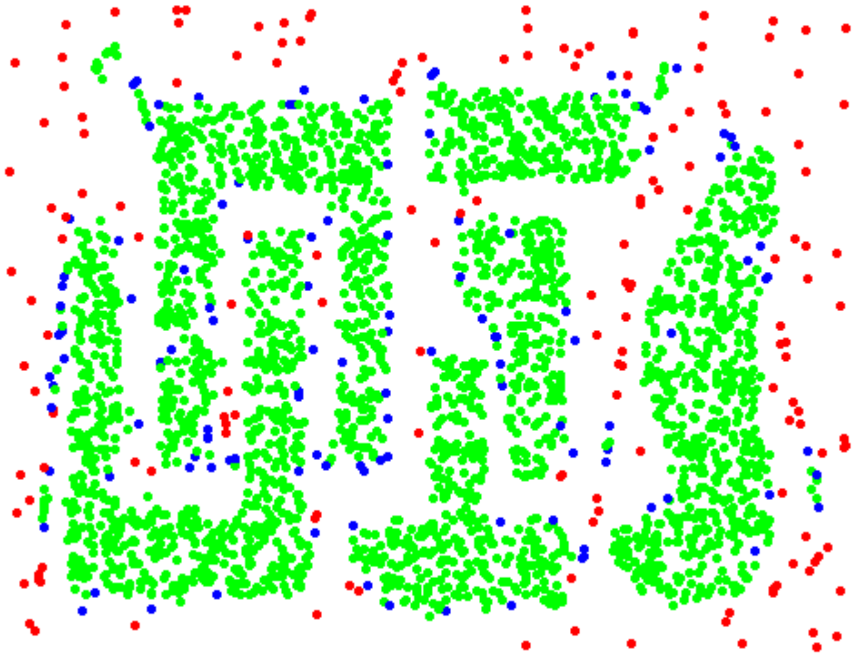
5. Repeat 4 till no union take place

DBSCAN Algorithm

- Eliminate incorrect points
- Perform clustering on the remaining points

```
current_cluster_label ← 1
for all core points do
    if the core point has no cluster label then
        current_cluster_label ← current_cluster_label + 1
        Label the current core point with cluster label current_cluster_label
    end if
    for all points in the Eps-neighborhood, except i^th the point itself do
        if the point does not have a cluster label then
            Label the point with cluster label current_cluster_label
        end if
    end for
end for
```

Core, Border and Noise Points



Eps = 10, MinPts = 4

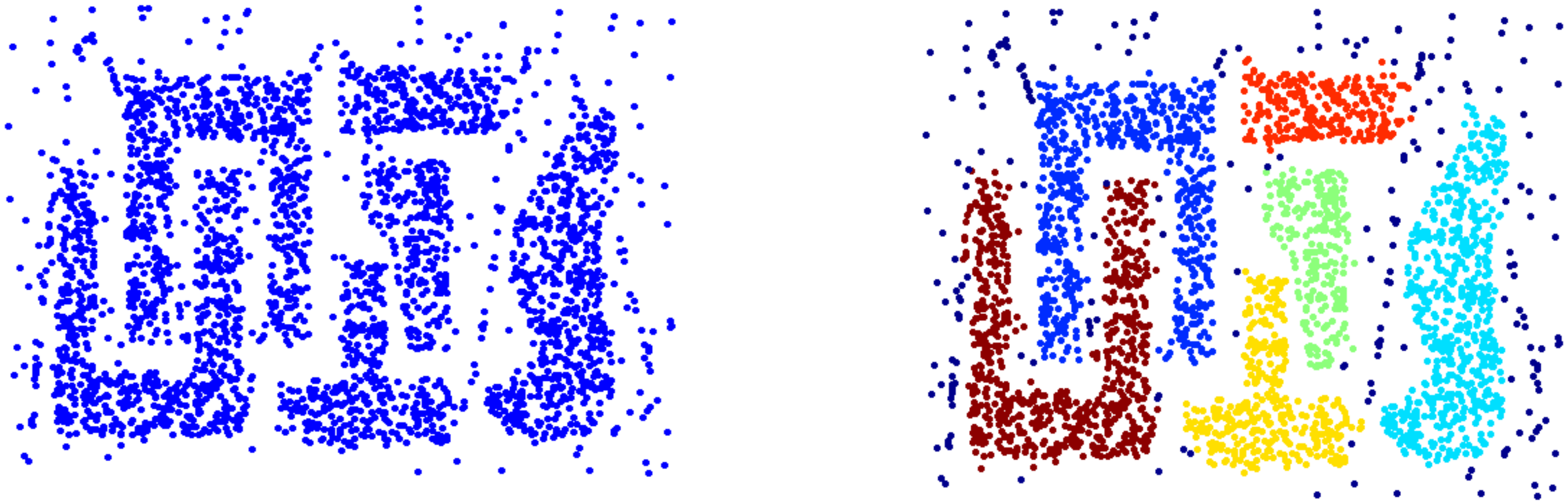
Advantages of DBSCAN

- Need not specify the number of clusters
- Can find arbitrarily shaped clusters and clusters surrounded by other clusters
- Requires just two parameters
- minPts and Eps can be set by a domain expert
- Mostly insensitive to the ordering of points in database

Complexity

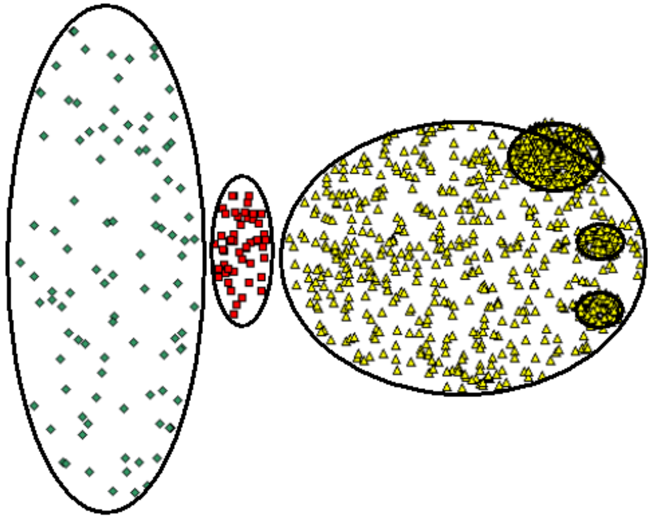
- **<u>Time Complexity</u>**: $O(n^2)$
  - Each point needs to be determined if it’s a core point or not
  - Can be reduced to $O(n\log n)$ in lower dimensions using efficient data structures (n is the number of objects to be clustered)
- **<u>Space Complexity</u>**: $O(n)$

When does DBSCAN work well?



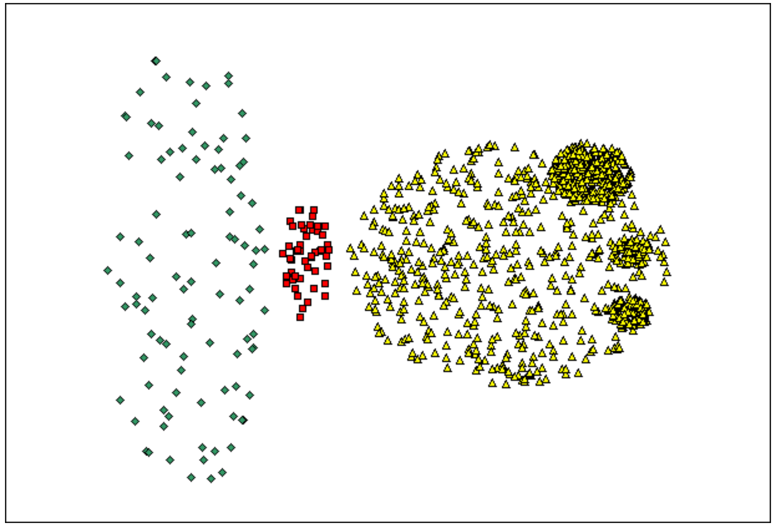
Can handle clusters of different shapes and sizes

When does DBSCAN NOT work well?

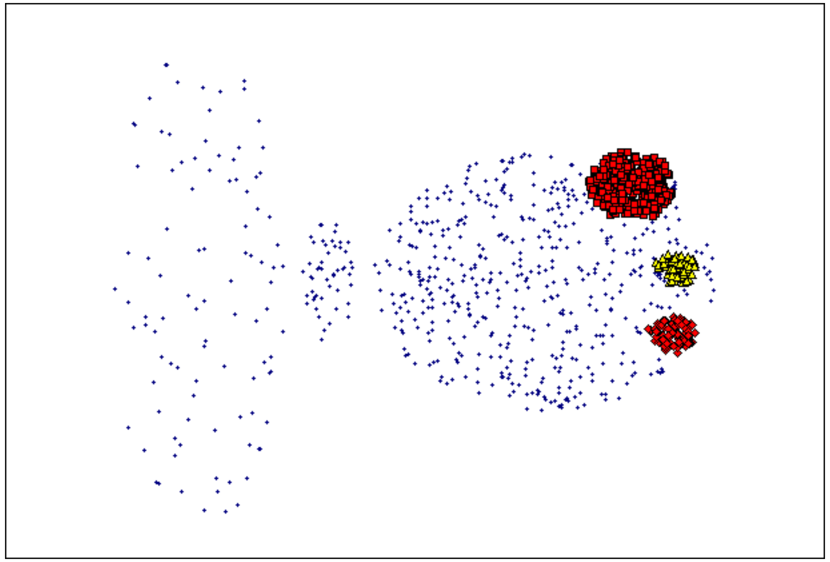


Original Points

(MinPts=4, Eps=9.75)



(MinPts=4, Eps=9.92)



- Varying densities
- High-dimensional data

Summary of DBSCAN: The Good

- Can detect arbitrary shapes
- Not very sensitive to noise
- Great at outlier detection
- Complexity is not too bad
- The most commonly used clustering algorithm besides k-Means

Summary of DBSCAN: The Bad

- Does not work well with high dimensional data
- Parameter selection can be tricky
- Needs domain knowledge to an extent

DBSCAN code from scratch in Python

- https://github.com/eriklindernoren/ML-From-Scratch/blob/master/mlfromscratch/unsupervised_learning/dbscan.py
- Code along with detailed explanation: https://becominghuman.ai/dbscan-clustering-algorithm-implementation-from-scratch-python-9950af5eed97

Thank you!