03 - DBSCAN

- We studied two clustering methods; K-Means which is a centroid-based algorithm, and Agglomerative Clustering which is a hierarchical system.
- There's one more approach to clustering known as DBSCAN which is a density-based approach.

- DBSCAN refers to Density-based spatial clustering of applications with noise
- DBSCAN works fairly well with large data and is able to handle noise and outliers very efficiently.
- First things first, here are some key ideas that build the DBSCAN.

Density and Dense Region

- DBSCAN uses a concept of density, which can be defined as;
  - at a certain point P, density at point P is the number of points within a hypersphere centered at P with a radius of $epsilon$
- Now, consider any region around the point P within eps radius, if there are more data points than $minpts$, we call the region a **Dense** region.
- For example, let's say we have eps=1 and $minpts$=10. Consider two points P_1 and P_2, both with a radius of eps
  - Suppose there are 20 points around point P_1, and only 6 points around point P_2, within the radius of eps, then we say the region around point P_1 is dense and the region around point P_2 as non-dense.

Min Points($minpts$) and Epsilon(eps)

- $minpts$ are the minimum number of points that we need in a hypersphere around point P with the radius of eps for considering the region as a **Dense** region.
- $minpts$ acts like a certain threshold and eps are the radius of the hypersphere

Core Point

- If a point P has points $\geq minpts$ within the radius of eps, then P is a core point.
- This also implies that point P has a dense region around it

Border Point

- A point P can be defined as a border point if:
    1. P is not a core point
    2. Point P lies in the neighborhood of point Q such that point Q is a core-point

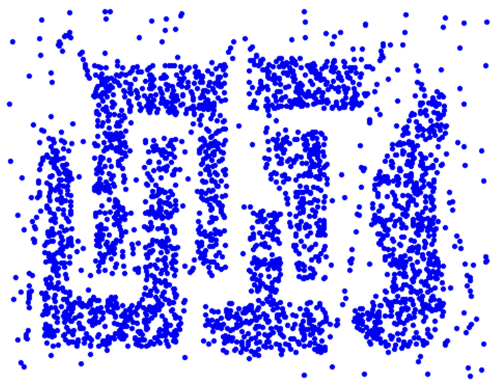
Neighborhood

- A point P is said to be in the neighborhood of point Q if distance between point P and Q is less than eps value; i.e. $dist(P,Q) \leq eps$

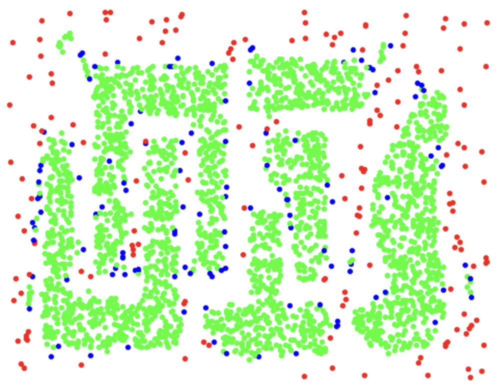
Noise Point

- It is a point that is neither a core point nor a border point.
- Suppose around core point P, a border point Q, and a point R which is in a non-dense region, the point R is said to be a noise point

- One thing to understand is that, when using DBSCAN, we fix two things:
    1. Min Points
    2. Epsilon.
- By fixing these hyperparameters, we get core points, border points, and noise points as well



Original Points



Point types: core, border and noise

Density Edges and Density Connected Points

- If points P and Q are two core points and the distance between point P and Q is less than or equal to eps value, then an edge between point P and Q is known as a **density edge.**

- Points P and Q can be said as density-connected points;
  - if both points are core points
  - if there exist other density edges connecting the points P and Q
  -
- Imagine we have two core points, point P, and Q, and there are other core points connecting point P with point Q; say P_1,P_2,....Pn, where the distance between each point P_1,P_2,....Pn is less than eps
- Then point P and point Q are said to be density connected points.

DBSCAN Algorithm

Step-1:

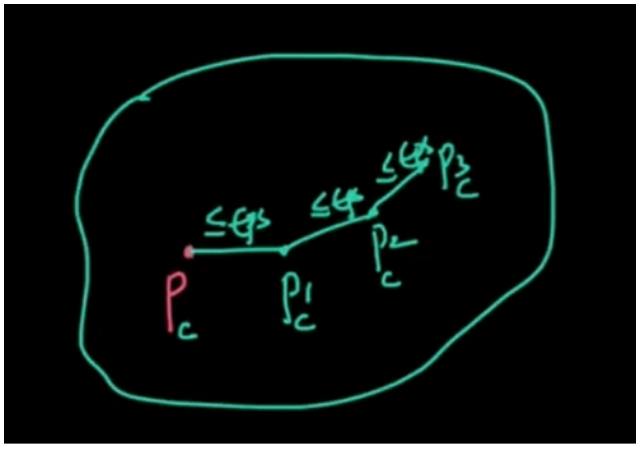
- For each point x_i that belongs to the dataset D, label it as either core point, border point, or noise point.
- Time complexity of this step would be $O(n*logN)$

Step-2:

- Remove all the noise points from the dataset
- Time complexity of this step would be $O(n)$
- This is basically a noise removal step

Step-3:

- For each core point P that is not yet assigned to any clustered:
    - create a new cluster with point P
    - Add all points that are density connected to point P, to the P's cluster
- To understand this with an example, Consider a core point P and there are three core points P_1,P_2 and P_3 which are density connected.
- Then, we group all the three points in the cluster of point P
- Time complexity of this step would be $O(n*logN)$



Step-4:

- For each border point, we assign it to the nearest core points' cluster.

    - For example, if we're having a cluster having core points $P_1, P_2 .... P_9$, and a border point P_{10} which is near the cluster.
    - We merge border point P_{10}, into the cluster of core points $P_1, P_2 .... P_9$
- Time complexity of this step would be $O(n) * logN$

Adjusting Min Points

- So there are some rules of thumb that people have made over the past years, which typically works well. They are:
    - value of $minpts$ should be greater than or equal to $d+1$; where d is dimensionality of the data
    - lot of libraries use the value of $minpt$s approximately equal to $2*d$
- The points mentioned above are typically rules of thumb and these are used because they tend to work fairly good in most of the cases
- Given an epsilon value, if the dataset is noisy, we pick larger $minpts$

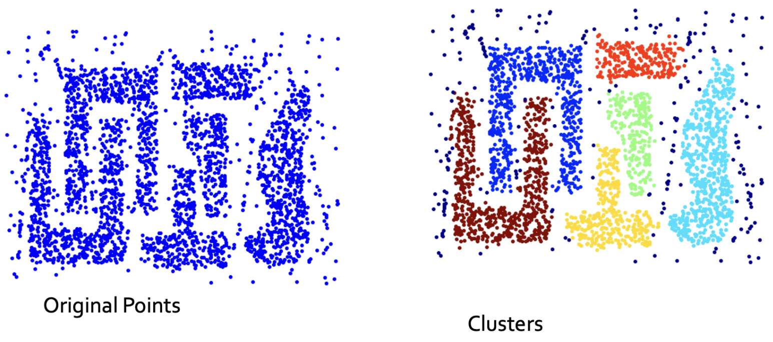
Adjusting Epsilon

- Let's assume we've fixed the value of $minpts$ = 4.
- **Step 1:**
    - for every point x_i in dataset, we compute a distance d_i
    - d_i refers to the distance from x_i to x_i's 4_{th} nearest neighbor (because we've set $minpts$ = 4)
- **Step 2:**
    - Sort the values of d_i's and plot them. You'll notice that the distance will increase graudally and then suddenlly, at a certain point, the value of distance will get boosted
    - So, the index at which the value of d_i distance got boosted will be used as the value of eps
    - The indices having higher values of d_i's will be outliers

Advantages of DBSCAN

- It’s resistant to noise
- Can handle clusters of different shapes and sizes.
- It doesn’t require one to specify the number of clusters a priori.
- It requires only two parameters: MinPts and Epsilon.
- It is designed for use with databases as it’s created by the database community.

When DBSCAN Works Well



Limitations of DBSCAN

- Even a small change in the hyperparameters, we can get a completely different type of clusters. So, it’s quite sensitive to the choice of hyperparameters.

DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

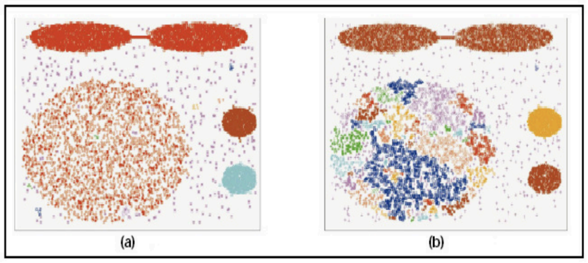
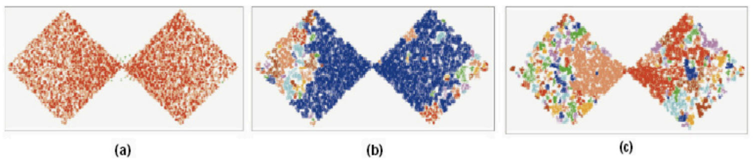
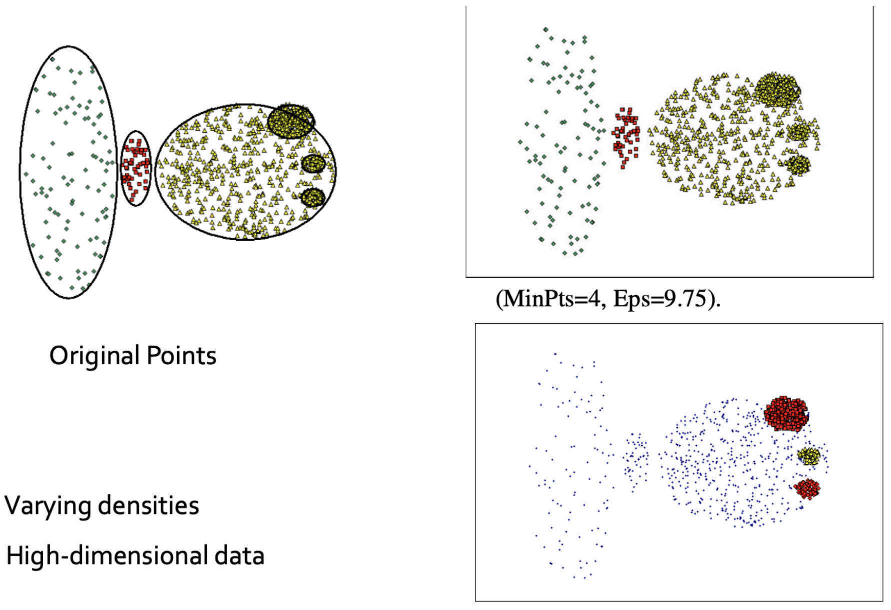


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



- Cannot handle varying densities and data with higher dimensions.



(MinPts=4, Eps=9.75).

(MinPts=4, Eps=9.92)

Anomaly Detection

So far, you have learned many concepts that are there in Machine Learning

The purpose of this lecture is to let you know how you can take the concepts that you already know like Random Forests, SVMs, KNN, etc., and modify them for using them for another purpose other than classification i.e. Anomaly Detection.

What is an Anomaly?

- Anomaly is synonymous with an outlier. These terms are often interchanged and may be called Novelty depending on the context.

What's the difference?

- Anomaly means something which is not a part of the normal behavior
- Novelty means something unique, or something that you haven't seen before(novel)

Applications of Anomaly/Outlier/Novelty Detection:

- Credit Card Fraud Detection
- System Intrusion Detection
- Ecosystem Disturbances in Weather and Environment (alarming about Tsunami, Earthquake, etc.)
- You can read how some startups are thinking out of the box using these techniques from a blog **here**

1. Distribution Based

- The simplest way for detecting an outlier would be to use distribution parameters (mean and standard deviation).
- Consider a feature X in some observations x_1, x_2,...., x_n with some outliers.
- We know that it will follow some distribution which will have parameters θ.
- Let **x** follow a gaussian distribution with some mean(μ) and standard deviation(σ)
- If we know the distribution of the data, we'll try to fit the distribution. But, the problem arises with the distribution parameters.
- While we know the distribution, the parameter estimates of the distribution are often corrupted by the noise/outlier
- Hence, we need to robustly estimate the parameters of the distribution. We do this using an algorithm called RANSAC which stands for Random Sample Consensus

2. Random Sample Consensus (RANSAC)

- RANSAC is a trial-and-error approach that works very well in real life.
- Imagine a dataset X with a number of points having parameters μ and σ. Let's call them collectively θ
- There are mainly three steps involved in RANSAC. They are

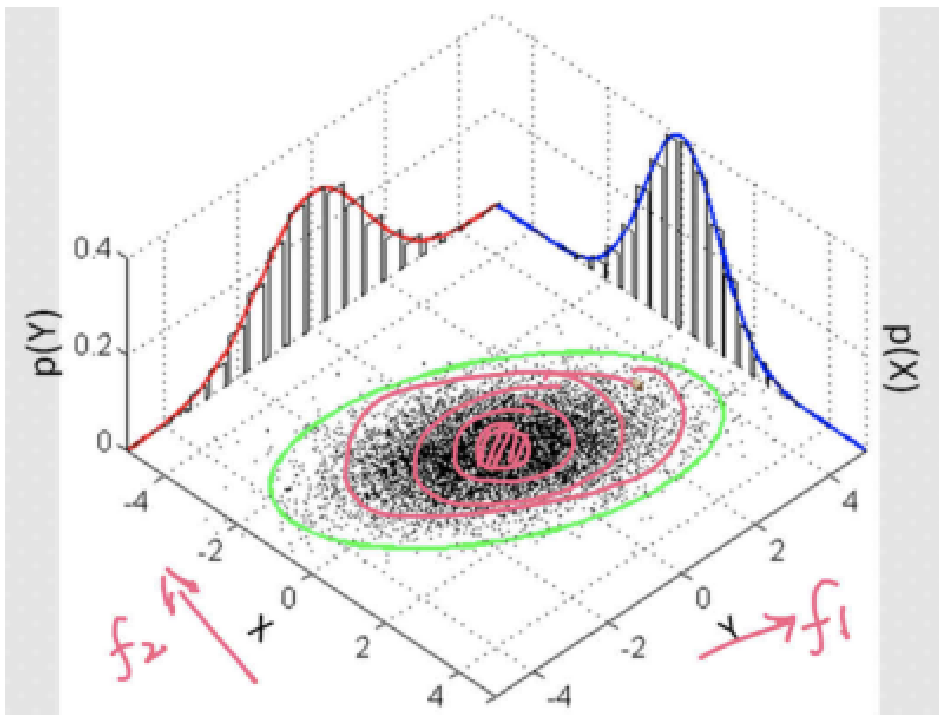
    - Sample a subset of points from the dataset (n'). We consider this point as an inlier.
    - Now, compute a model that estimates the parameters of the sampled points.
    - Score the model which indicates how many points will support the model.
- We repeat these three steps iteratively and then select the model best supported by the data, which then tells which points are inliers and which are outliers.

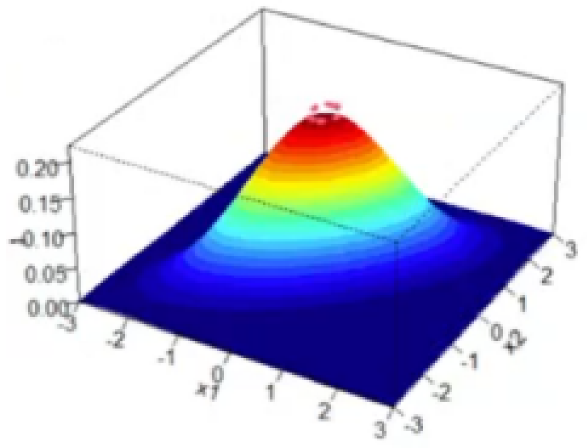
Extending the idea to higher dimensions

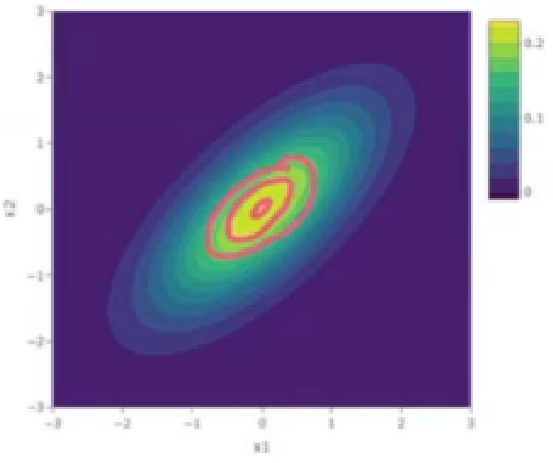
- Till now, we assumed that we have only a single feature X which was following Gaussian Distribution.
- Now, imagine we have d-dimensional data where each point $x_i \in R^d$ and the data is not labeled.
- If we know the data points x_is follow multivariate gaussian distribution(unimodal), then X follows normal distribution; X ~ ($\vec{u}$, Σ), where is $\vec{u}$ mean vector and Σ is a covariance matrix
- Here we'll consider ($\vec{u}$, Σ) as θ
- In GMMS, in multidimensional space, the shape of gaussian was similar to a hill where the density of the points was highest in the middle contour, and it keeps getting low as we move away from the center
- In this case too, RANSAC can be applied. Farther away from centroid, we'll know that it is an outlier.
- Similar principle is used in another method called Elliptical Envelope.

3. Elliptic Envelope

- We know that a Unimodal Multivariate Gaussian Distribution on a single plane will look like ellipses if visualized on a plane. This idea can be extended to find out an outlier







- Given some data X where x_is Є R^d and X follows Normal Distribution being unimodal, Elliptical Envelope robustly estimates the parameters ($\vec{u}$, Σ).
- The term robustly means without getting impacted by outliers
- Next, then we remove the points that are outliers which are very far away from the centroid

What if the distribution is non-gaussian? Do we need to convert it into gaussian distribution?

- While the elliptical envelope method makes an assumption that the distribution is gaussian, the strategy can be applied to any distribution.

- As long as we know any distribution and its parameters θ, we can extend our strategy to use RANSAC and estimate parameters θ.

- These can be other distributions such as multivariate Poissons, multivariate log normals, etc., but we don't use them as much.

- One other strategy is to convert them into Gaussians but we don't necessarily have to.

- As long as there is any distribution we can calculate the probability of $x_i \in X$ using PMF/PDF.

Sklearn walkthrough:

Scikit-learn implements **EllipticEnvelope** as a part of the **covariance** module. Let's walk through the parameters that are important:

1. **assume_centered**: It is for assuming that the data is centered at 0, i.e. $\vec{\mu}$ =0.
By default, it is set as $False$. If set to $True$, it will just estimate the covariance matrix.
It uses the FastMCD approach and not the RANSAC approach. MCD refers to the Minimum Covariance Determinant.
FastMCD performs in a similar way to RANSAC where it takes a subset of points and using them tries to estimate the distribution parameters robustly.

2. **support_fraction**: It tells how many points to use to estimate the parameters.

3. **contamination**: It says what percentage of our data we think are outliers.
It takes values from 0 to 0.5, where 0.5 represents that 50 % of our data is noisy.
The default assumption value is 0.1 ~ 10 %

Disadvantages:

- It cannot be used for non-unimodal data

- It is specifically for multivariate Gaussians

- If the data fails to meet the assumptions of unimodal and multivariate gaussian, the whole thing crashes.