DBSCAN

Density-based spatial clustering of applications with noise

(밀도 기반 클러스터링)

21.01.20

Seunghan Lee (이승한)

Contents

1. What is DBSCAN?

- **밀도(Density) 기반**의 클러스터링 방법
- 기본 Idea :

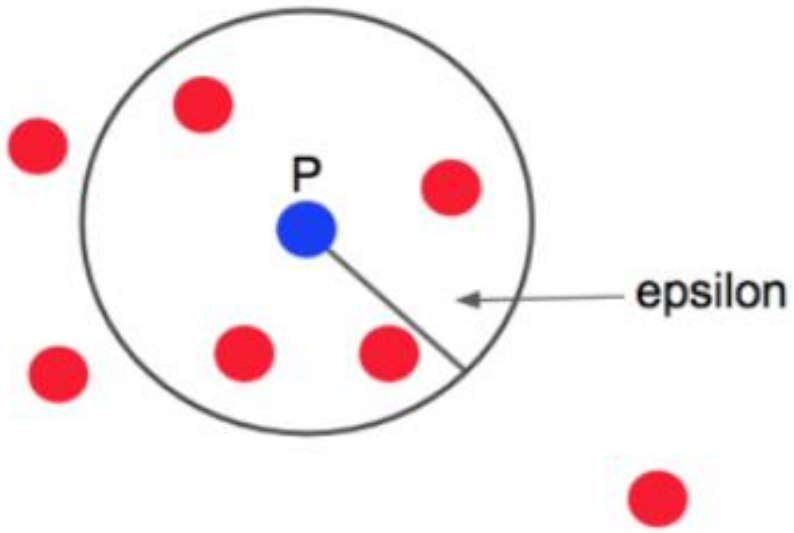
  군집 = (1) 특정 점 기준으로

  (2) "일정 거리" 내에

  (3) "최소 m개의 점"이 있으면

해당 '특정 점' 과 '그 안에 있는 모든 점들 ' 은 **한 개의 cluster를 형성**

1. What is DBSCAN?

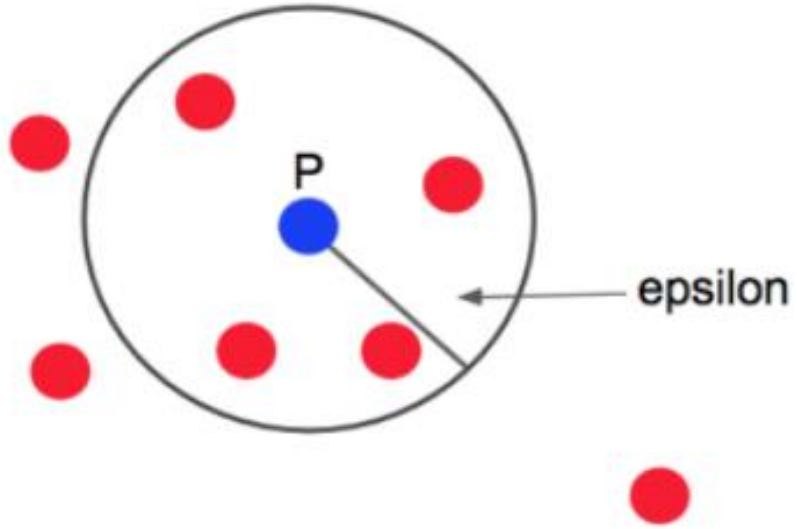


군집 = (1) 특정 점 기준으로

(2) "일정 거리" 내에

(3) "최소 m개의 점"이 있으면

1. What is DBSCAN?



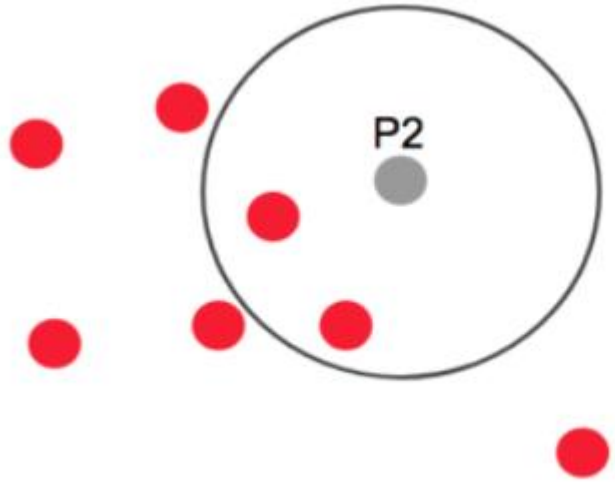
if

- 특정 점 : **파란 점 P**
- 일정 거리 : epsilon
- m = 3

왼쪽의 **점 P**와, epsilon거리 내에 있는 빨간색 **점 4개**, (> m=3)

총 5개의 점은 하나의 cluster를 형성한다

1. What is DBSCAN?



if

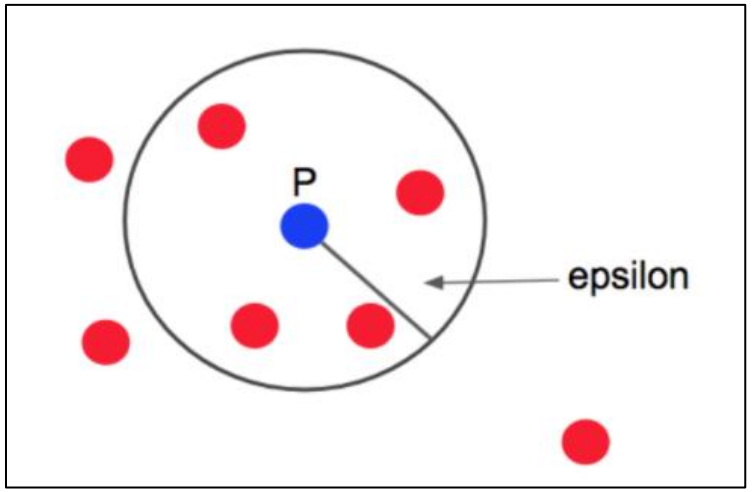
- 특정 점 : **파란 점 P**
- 일정 거리 : epsilon
- m = 3

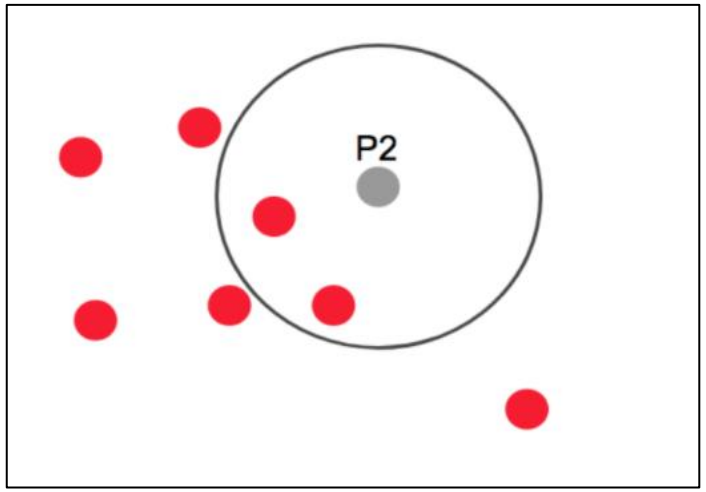
왼쪽의 **점 P**의 epsilon거리 내에는, 빨간색 **점 2개** 밖에 없기 때문에 (< m=3), **cluster를 형성하지 못한다.**

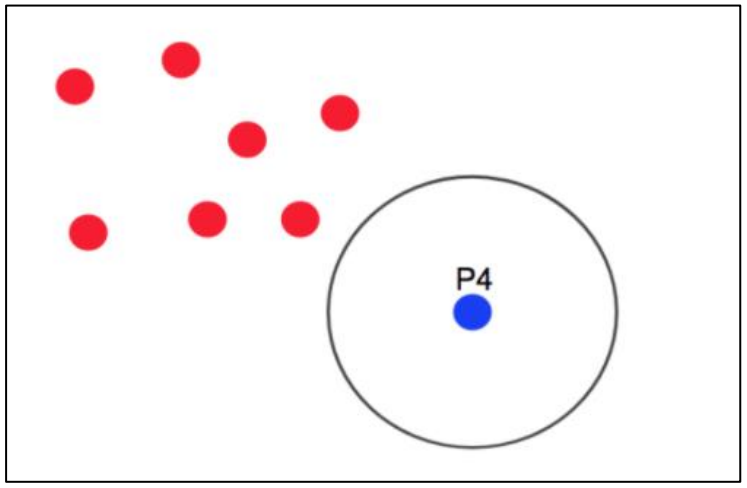
1. What is DBSCAN?

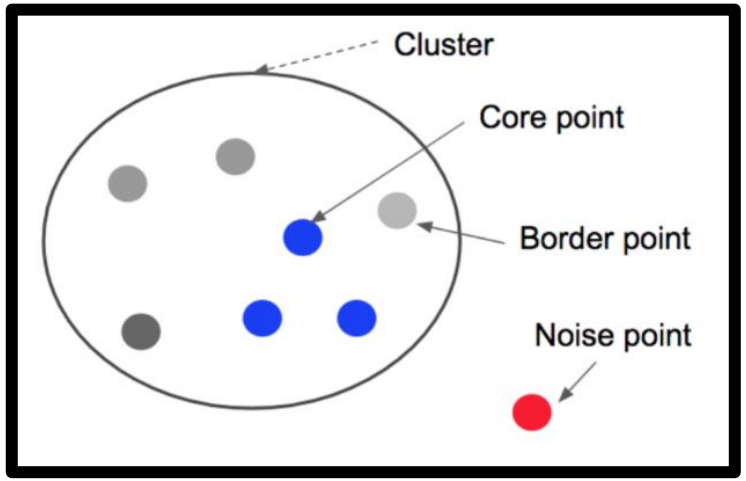
용어 설명

- 1) Core point
- 2) Border point
- 3) Noise Point







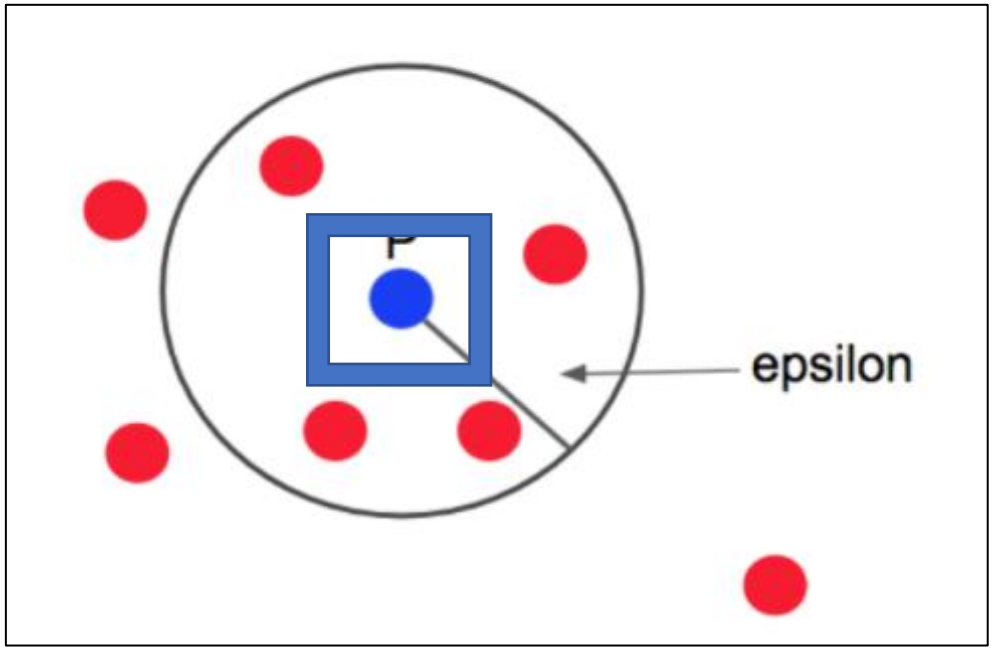


1. What is DBSCAN?

1) Core point

어떤 점이 특정 반경(epsilon) 내에, cluster를 형성할 수 있게끔 만큼의

주변 점들을 가지고 있는 경우!



1. What is DBSCAN?

1) Core point

어떤 Core point (P1)의 일정 epsilon 거리 내에,

다른 Core point (P2)가 속하게 된다면,

두 개의 cluster (각각 p1 & p2가 중심이 되는 cluster)는 **하나의 cluster로 합쳐진다!**

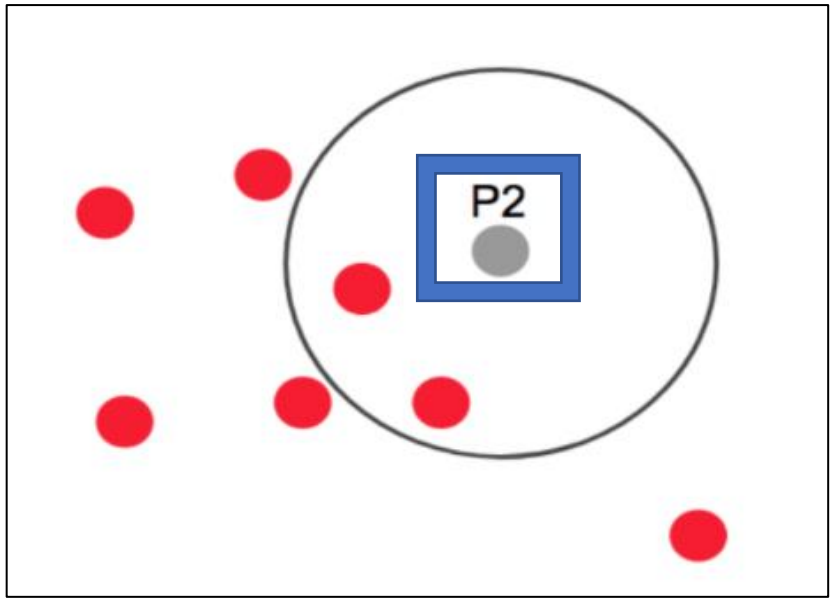
1. What is DBSCAN?

2) Border point

어떤 점이 특정 cluster에 속하긴 하지만, 스스로가 중심으로써 특정 반경

(epsilon) 내에, cluster를 형성할 수 있게끔

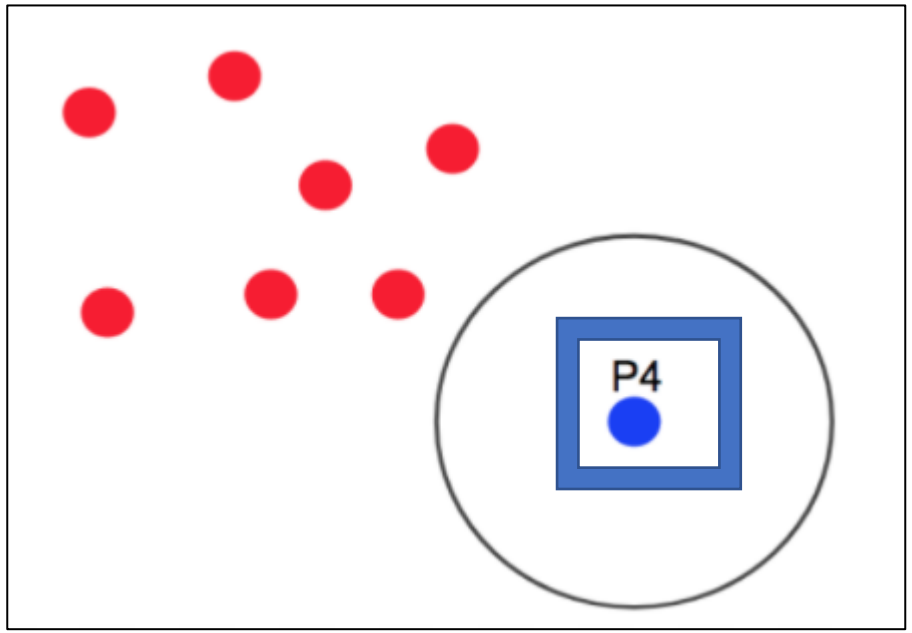
만큼의 주변 점들을 가지지 못한 경우!



1. What is DBSCAN?

3) Noise point

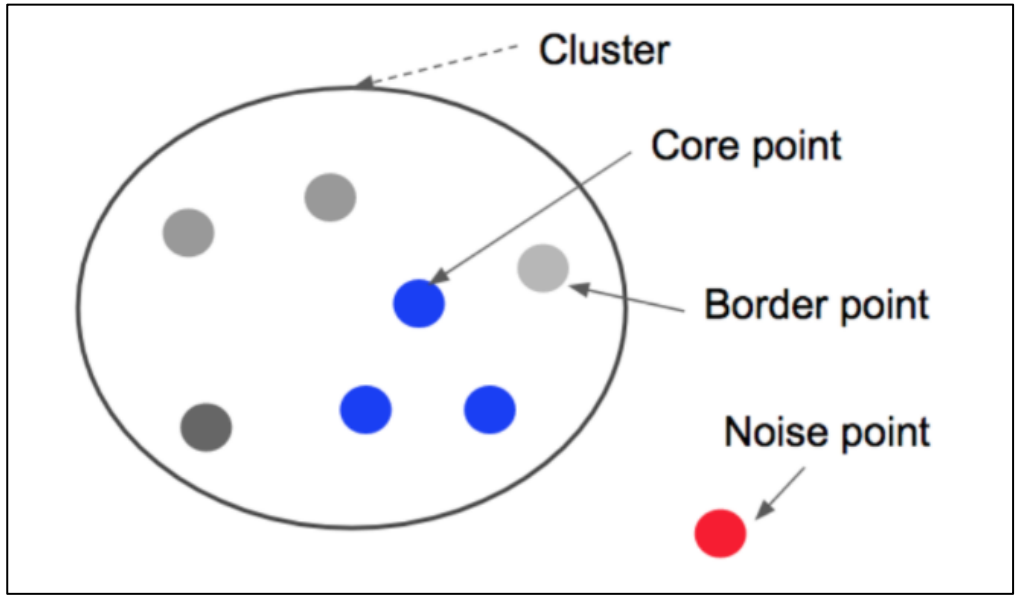
어떤 점을 중심으로 하더라도, cluster내에 속하지 못하게 되는 점



1. What is DBSCAN?

Summary

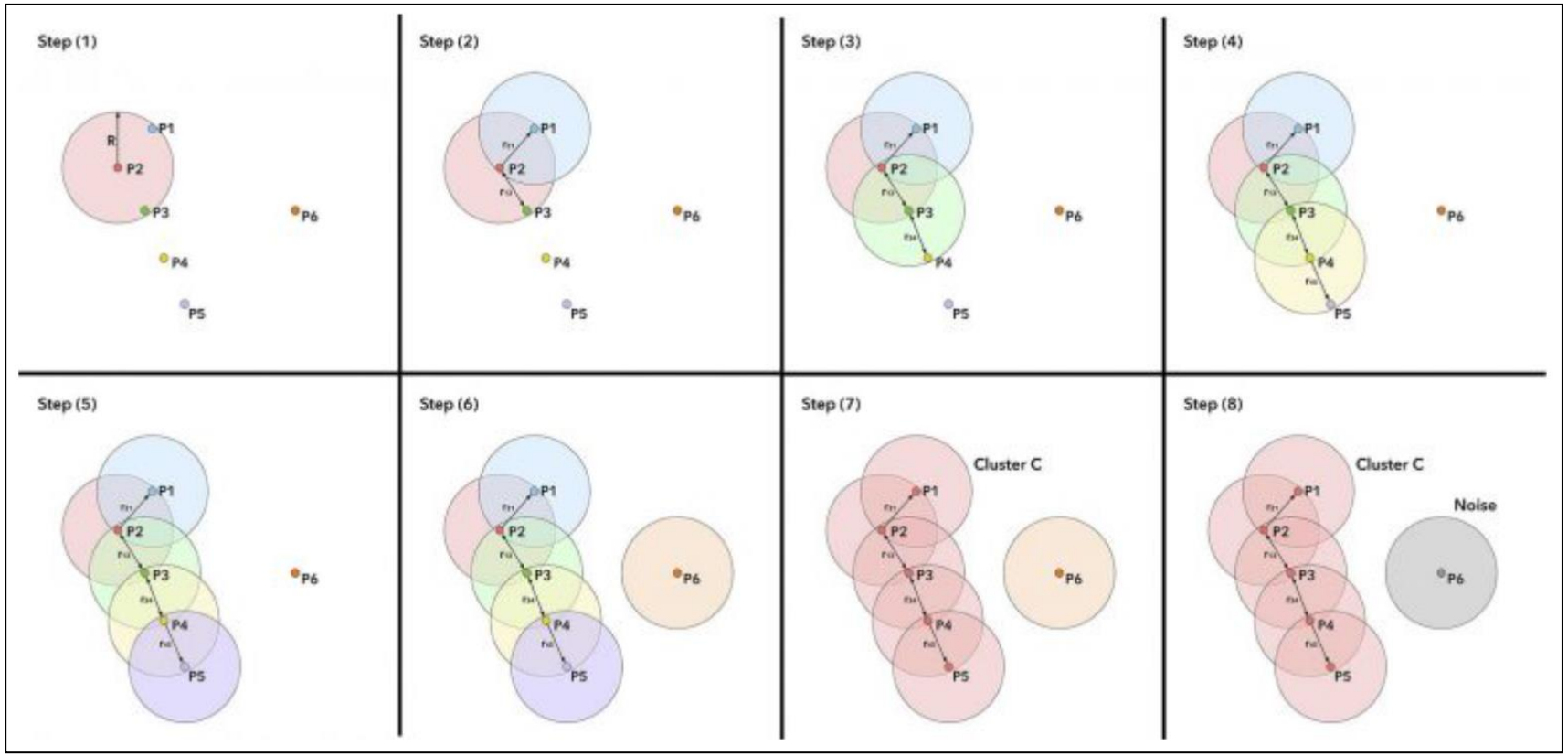
모든 점은, 세 개의 point 중 하나에 반드시 속한다!



1. What is DBSCAN?

Summary

http://www.francescogrigoli.it/tutorial/how-dbscan-clustering-algorithm-works-an-easy-guide-thorugh-sketches.html



2. Training DBSCAN

필요한 파라미터

- 1) ε (eps) : cluster를 형성하기 위한 최소의 거리
- 2) minPts : cluster를 형성하기 위해 필요한 최소의 점 개수

Input: N objects to be clustered and global parameters *Eps*, *MinPts*.
Output: Clusters of objects.

Algorithm:

1) Arbitrary select a point *P*.
2) Retrieve all points density-reachable from *P* wrt ***Eps*** and ***MinPts***.
3) If *P* is a core point, a cluster is formed.
4) If *P* is a border point, no points are density-reachable from *P* and **DBSCAN** visits the next point of the database.
5) Continue the process until all of the points have been processed.

2. Training DBSCAN

Parameter Estimation : (1) ε (eps)

(1) 너무 작으면, 대부분의 데이터가 cluster에 속하지 않게 될 것!

(대부분이 noise point가 될 것)

(2) 너무 크면, 대부분의 데이터가 하나의 cluster가 될 것

2. Training DBSCAN

Parameter Estimation : (2) minPts

(1) D차원일 경우, minPts >= D+1로! (Rule of Thumb)

(2) 최소 3 이상

- minPts =1 : 각각의 점이 결국 하나의 cluster
- minPts = 2 : Hierarchical Clustering
- 따라서, minPts는 3 이상으로 하는 것이 좋음

(3) Outlier가 많은 데이터 일수록, minPts는 크게!

2. Training DBSCAN

Time & Space Complexity

(1) Time complexity : **O(n^2)**

- 각각의 점이 core point인지 확인하는 과정 &
- 해당 점으로부터, 나머지 점들이 일정 거리 내에 있는지 계산
- efficient data structure로 O(n * log(n))까지 줄일 수 있음

(2) Space complexity : **O(n)**

- 특정 점을 기준으로, 나머지 n-1개의 점들에 대한 거리 계산!

3. Pros & Cons of DBSCAN

Pros

- 1) outlier에 강함

  (+ outlier detection에도 사용될 수 있음)

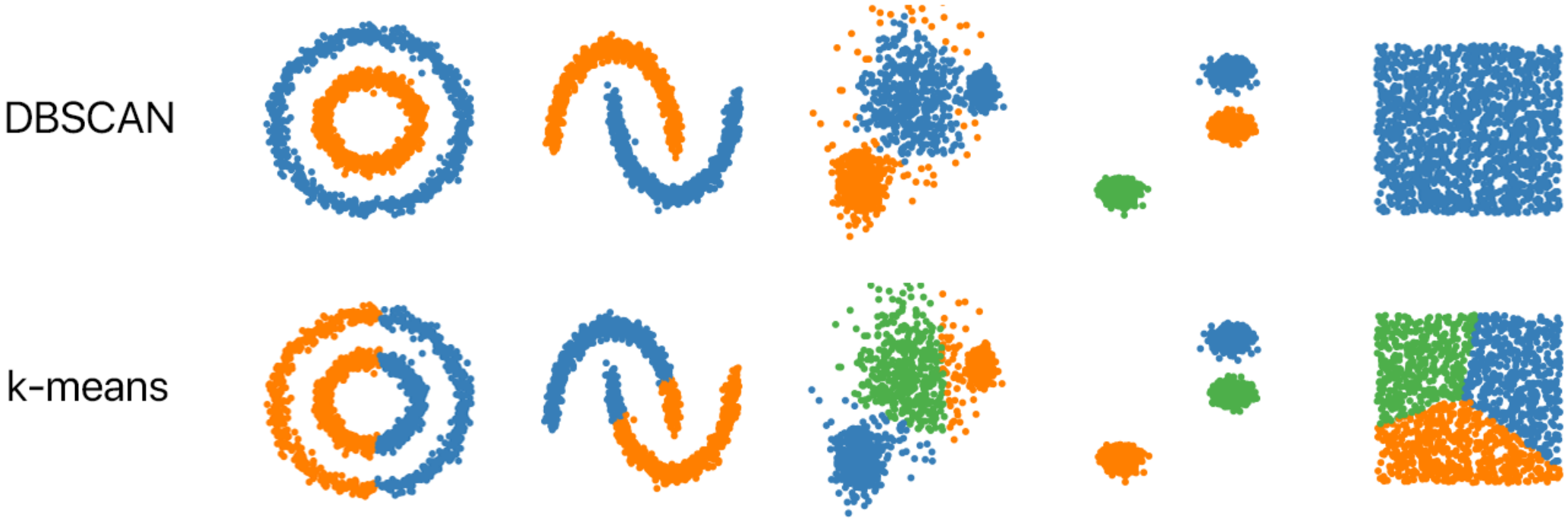
- 2) (K-means와 다르게) 사전에 몇 개의 cluster를 생성할지 정해줄 필요 X

- 3) arbitrarily-shaped cluster도 잘 잡아낼 수 있음

- 4) domain 지식을 바탕으로, minPts / eps를 잘 알 경우 사용하기 좋다

3. Pros & Cons of DBSCAN

Pros



3. Pros & Cons of DBSCAN

Cons

- 1) 고차원의 데이터의 경우에 적합하지 않을 수도
  (이는 사실 DBSCAN만의 특징 뿐만 아니라, Euclidean distance를 지표로써 사용하는 모든 알고리즘에 해당 되는 것)
- 2) cluster 개수를 사전에 지정할 필요는 없지만,
  결국 적절한 minPts / eps를 찾아서 지정해줘야함

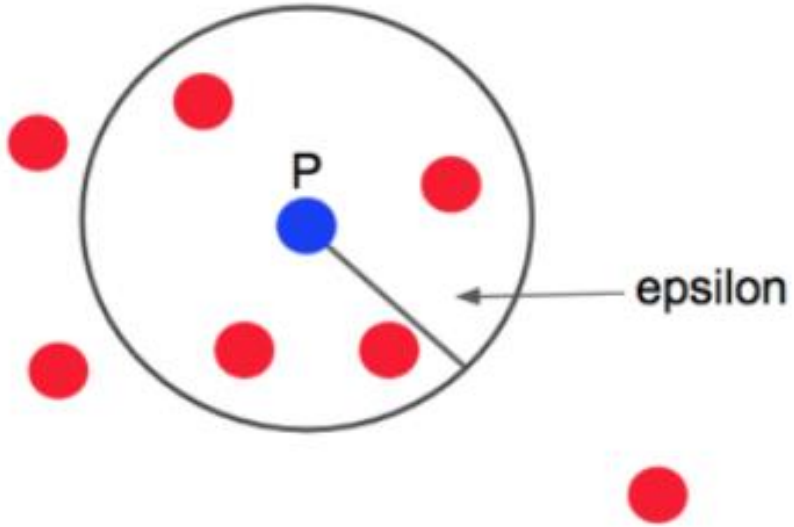
4. Python Code for DBSCAN

(1) 사용할 Package Import

```
from sklearn.cluster import DBSCAN
```

hyperparameter 후보

- 1) `min_samples_list` : cluster형성을 위해 필요한 최소의 data 개수
- 2) `eps_list` : cluster형성을 위한 최대의 거리



4. Python Code for DBSCAN

(2) 최적의 hyperparameter 탐색

```
min_samples_list = [2*i for i in range(2,10)]
eps_list=[10*i for i in range(1,20)]
```

(Min_sample 후보군) (eps 후보군)

```
def dbscan_hyperparameter(X,min_samples_list,eps_list,top_N):
    ss_dict=dict()
    for min_samples in min_samples_list:
        for eps in eps_list:
            dbscan = DBSCAN(eps=eps, min_samples=min_samples).fit(X)
            try :
                silhouette = silhouette_score(X,dbscan.labels_)
                ss_dict[(min_samples,eps)]= silhouette
                num_of_clusters = len(set(dbscan.labels_))
                if num_of_clusters>2:
                    print('Silhouette score of "min_samples={}" and "eps={}" is {}, and number of clusters is {}'.
                          format(min_samples,eps,silhouette,num_of_clusters))
            except Exception:
                pass
    return ss_dict
```

4. Python Code for DBSCAN

(2) 최적의 hyperparameter 탐색

```
min_samples_list = [2*i for i in range(2,10)]
eps_list=[10*i for i in range(1,20)]
```

```
ss_dict = dbscan_hyperparameter(pca_df_temp.values,min_samples_list,eps_list)
```

executed in 1.57s, finished 12:22:35 2021-01-18

```
Silhouette score of "min_samples=4" and "eps=50" is -0.2075477431665968, and number of clusters is 8
Silhouette score of "min_samples=4" and "eps=60" is -0.0906100714107247, and number of clusters is 4
Silhouette score of "min_samples=6" and "eps=60" is -0.05005686133077893, and number of clusters is 3
Silhouette score of "min_samples=6" and "eps=80" is 0.18362328384681317, and number of clusters is 3
Silhouette score of "min_samples=10" and "eps=60" is -0.10149286272332543, and number of clusters is 3
Silhouette score of "min_samples=12" and "eps=60" is -0.10221356952203912, and number of clusters is 3
```

Silhouette score : (Bad) -1 ~ 1 (Good)

```
dbscan = DBSCAN(eps=80, min_samples=6).fit(pca_df_temp.values)
```

4. Python Code for DBSCAN

(3) 최적의 hyperparameter를 사용한 DBSCAN 학습

```
dbscan = DBSCAN(eps=80, min_samples=6).fit(pca_df_temp.values)
```

```
dbscan.labels_
```

executed in 10ms, finished 12:22:49 2021-01-18

```
array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1,  0,
        0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  1,  1,  1,  1,  1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0, -1, -1,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, -1,  0, -1, -1, -1,  0,  0,  0, -1,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
      dtype=int64)
```

-1 : Noise

군집화 결과

4. Python Code for DBSCAN

(4) 결과

```
unique_elements, counts_elements = np.unique(dbscan.labels_, return_counts=True)
cluster_count = dict(zip(unique_elements,counts_elements))
```

executed in 5ms, finished 12:23:00 2021-01-18

```
cluster_count
```

executed in 8ms, finished 12:23:01 2021-01-18

```
{-1: 24, 0: 290, 1: 6}
```

좋지 않은 군집화 결과

4. Python Code for DBSCAN

(4) 결과

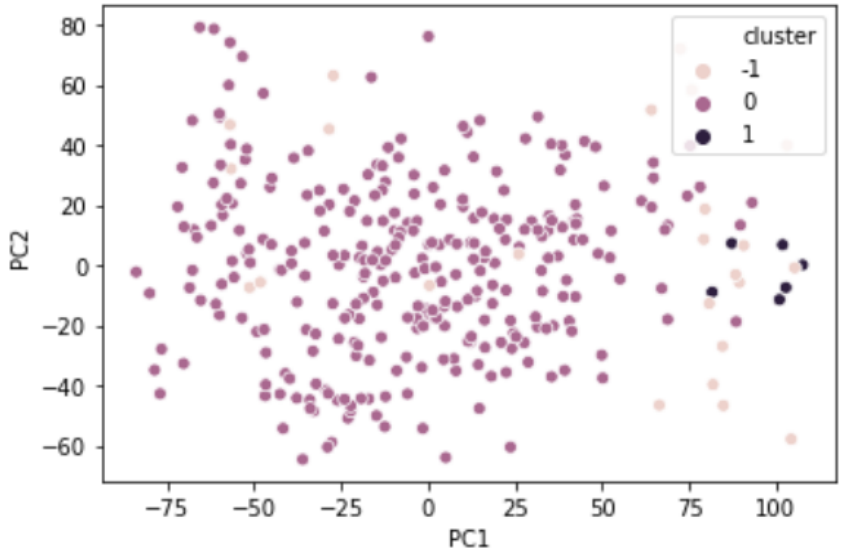
```
pca_df_temp['cluster']=dbscan.labels_
```

executed in 6ms, finished 12:23:01 2021-01-18

```
sns.scatterplot(x='PC1',y='PC2',hue='cluster',legend='full',data=pca_df_temp)
```

executed in 157ms, finished 12:23:01 2021-01-18

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe946d68b0>
```



위 데이터에는, DBSCAN보다는 K-means가 더 적합한 알고리즘으로 판단됨