DBSCAN : density-based clustering applications with noise

(가) 장점
- 클러스터의 개수를 미리 지정할 필요가 없다.
- 복잡한 형상도 찾을 수 있다.
- 어떤 클래스에도 속하지 않는 포인트를 구분할 수 있다.
- 병합 군집이나 k-평균보다 다소 느리지만 비교적 큰 데이터 셋에도 적용가능

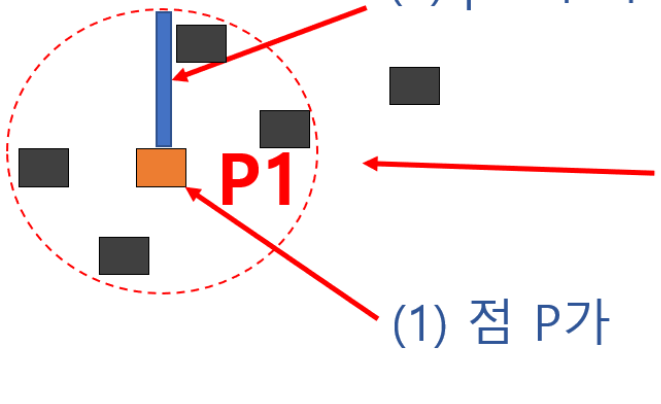
간단한 설명

(가) 특성 공간에서 가까이 있는 데이터가 많아 붐비는 지역의 포인트를 찾는다.
(나) 밀집 지역(dense region) - 붐비는 지역
(다) DBSCAN의 아이디어는 데이터의 밀집 지역이 한 클러스터를 구성하며 비교적 비어있는 지역을 경계로 다른 클러스터와 구분된다는 것이다.

가정 : 어떤 점 p에서부터 e(epsilon)내에 점이 m개 있으면 하나의 군집으로 인식한다.

기준점 부터의 거리 epsilon

반경 내의 점의 수 minPts=3

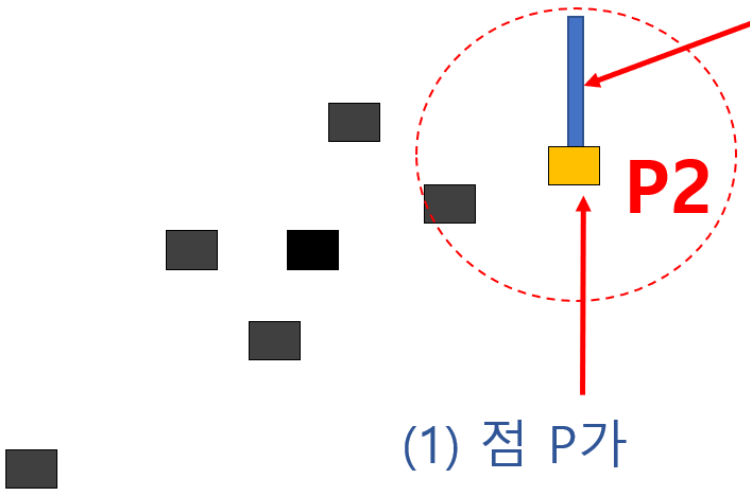


(2) p로부터거리 e 이내에

(3) 범위e내에 3(m)개 이상의 점이 있다.

(1) 점 P가

(1),(2),(3)를 만족하면, 우리는 p점을 core point라고 한다.



기준점 부터의 거리 epsilon

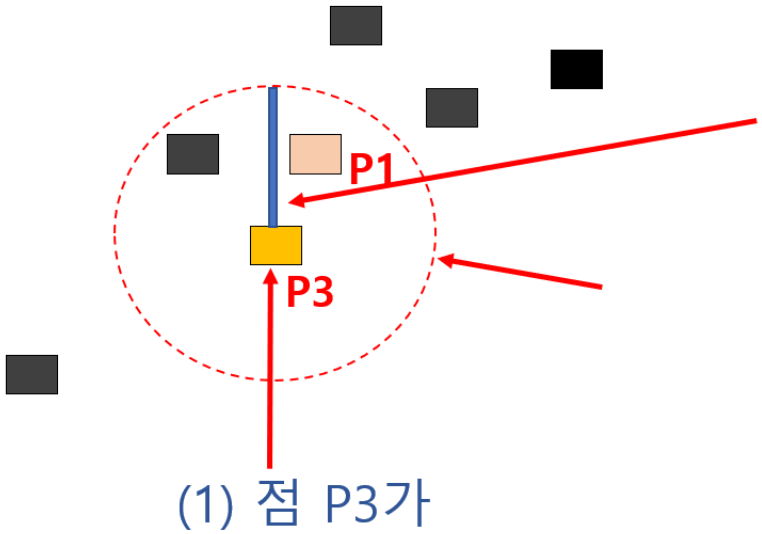
반경 내의 점의 수 minPts=3

(3) 범위e내에 3개가 안되는 점이 있다.
단, core point를 포함하고 있다.

(1),(2),(3)를 만족하면, 우리는
p점을 border point라고 한다.

기준점부터의 거리 epsilon

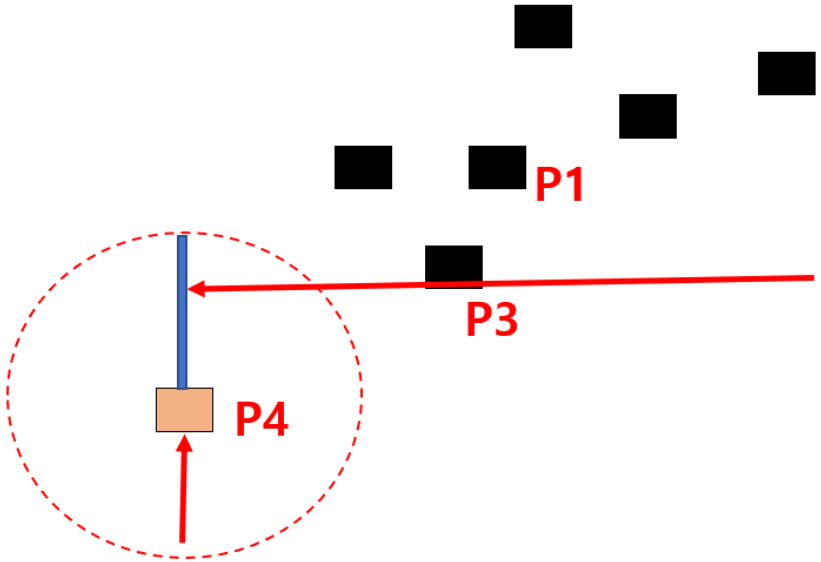
반경 내의 점의 수 minPts=3



(2) p3로부터거리 e 이내에

(3) 범위e내에 3개이상의 점이 있다.
이 포인트도 core point이다.
여기에서 P1도 포함하고 있으므로

P1과 P3는 같은 군집을 갖는다.

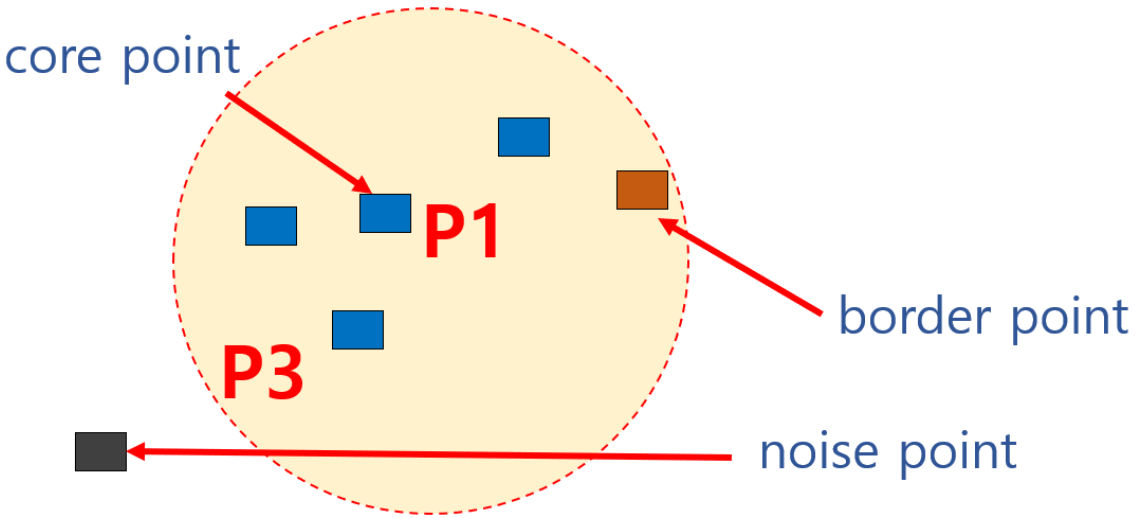


(1) 점 P4가 noise point

(2) p4로부터거리 e 이내에

(3) 범위e내에 하나의 점도 없고, core point도 갖지 않는다.
우리는 이를 noise point라고 한다.

주변의 점들의 밀도(개수)로 판단한다.



기준점부터의 거리 epsilon

반경 내의 점의 수 minPts=3

In [4]:
```
import mglearn
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
import pandas as pd
%matplotlib inline

### 한글
import matplotlib
from matplotlib import font_manager, rc
font_loc = "C:/Windows/Fonts/malgunbd.ttf"
font_name = font_manager.FontProperties(fname=font_loc).get_name()
matplotlib.rc('font', family=font_name)
```

In [3]:
```
mglearn.plots.plot_dbscan()
```

```
min_samples: 2 eps: 1.000000  cluster: [-1  0  0 -1  0 -1  1  1  0  1 -1 -1]
min_samples: 2 eps: 1.500000  cluster: [0 1 1 1 1 0 2 2 1 2 2 0]
min_samples: 2 eps: 2.000000  cluster: [0 1 1 1 1 0 0 0 1 0 0 0]
min_samples: 2 eps: 3.000000  cluster: [0 0 0 0 0 0 0 0 0 0 0 0]
min_samples: 3 eps: 1.000000  cluster: [-1  0  0 -1  0 -1  1  1  0  1 -1 -1]
min_samples: 3 eps: 1.500000  cluster: [0 1 1 1 1 0 2 2 1 2 2 0]
min_samples: 3 eps: 2.000000  cluster: [0 1 1 1 1 0 0 0 1 0 0 0]
min_samples: 3 eps: 3.000000  cluster: [0 0 0 0 0 0 0 0 0 0 0 0]
min_samples: 5 eps: 1.000000  cluster: [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
min_samples: 5 eps: 1.500000  cluster: [-1  0  0  0  0 -1 -1 -1  0 -1 -1 -1]
min_samples: 5 eps: 2.000000  cluster: [-1  0  0  0  0 -1 -1 -1  0 -1 -1 -1]
min_samples: 5 eps: 3.000000  cluster: [0 0 0 0 0 0 0 0 0 0 0 0]
```

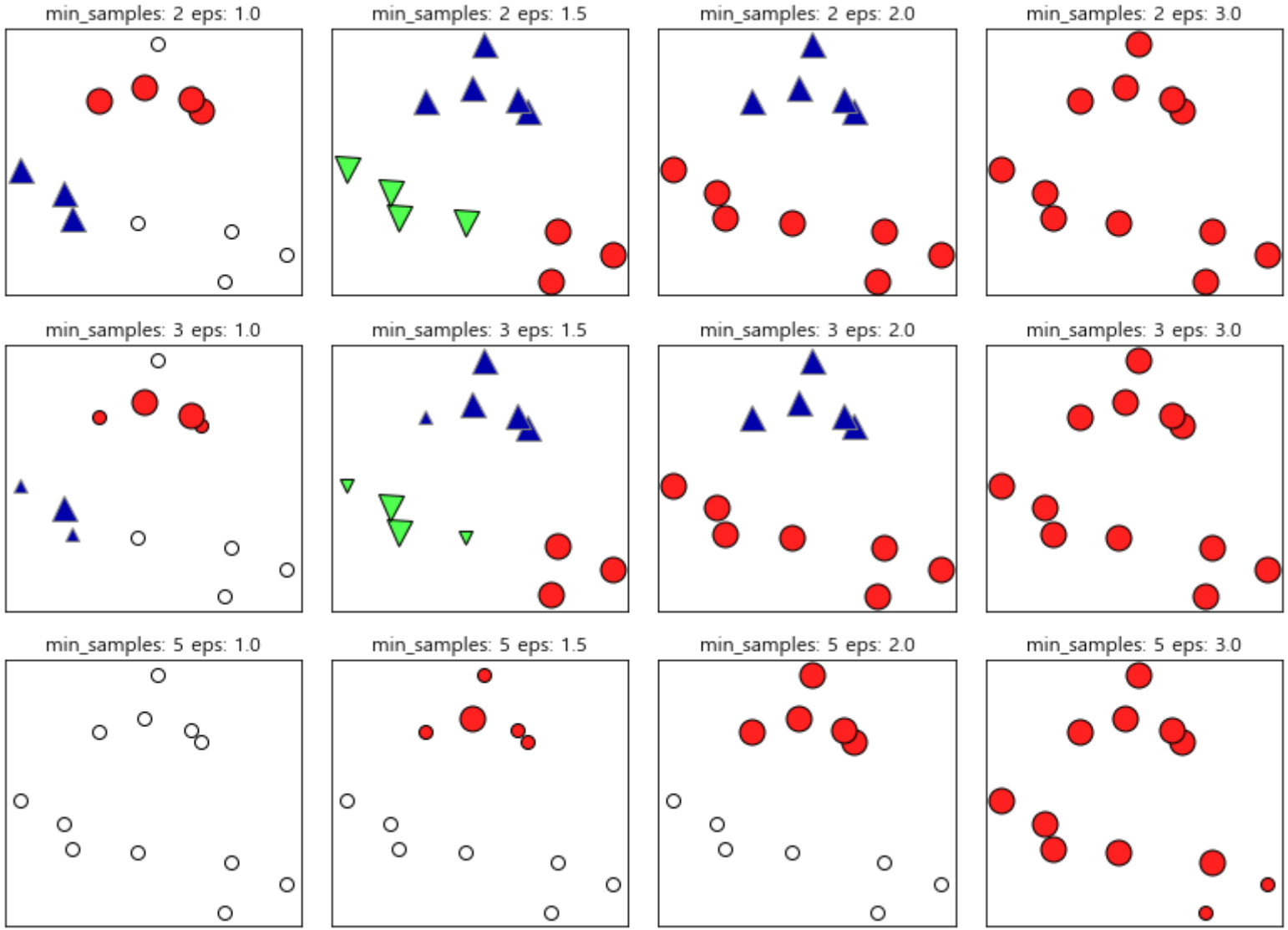


그림 설명

(1) core point(핵심 포인트) : 밀집 지역에 있는 포인트
  한 데이터 포인트에서 eps 거리 안에 데이터가 min_samples 개수만큼 들어 있다면.. 이 데이터를 핵심 샘플로 분류
(2) eps보다 가까운 핵심 샘플은 DBSCAN에 의해 동일한 클러스터로 합쳐진다.

동작

(가) 알고리즘을 시작 시에 무작위로 포인트를 선택
(나) 포인트에서 eps 거리 안의 모든 포인트를 찾는다.
(다) 만약 eps 거리안의 포인트 수가 min_samples보다 적다면 그 포인트는 어떤 클래스에도 속하지 않는 잡음(noise)로 레이블
(라) 핵심 샘플로 레이블 후, 그 포인트의 모든 이웃을 살펴보다.
(마) 만약 어떤 클러스터에도 할당되지 않았다면 바로 전의 만든 클러스터로 레이블을 할당. 핵심 샘플이면 그 포인트의 이웃을 차례로 방문
(바) 이런 식으로 계속 진행하여 클러스터는 eps 거리 안에 더 이상 핵심 샘플이 없을 때까지 자라난다.

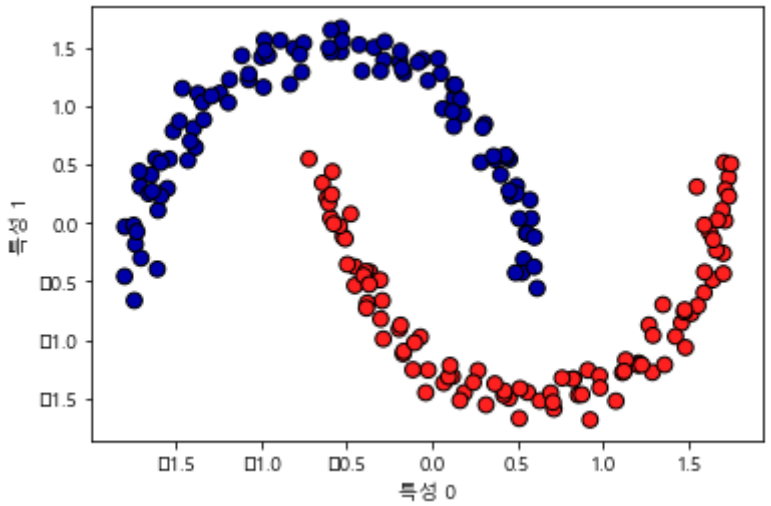
종류

핵심 샘플
경계 포인트 (핵심 포인트에서 eps 거리 안에 있는 포인트)
잡음 포인트

연습해보기

In [2]:
```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_moons
from sklearn.preprocessing import StandardScaler

X, y = make_moons(n_samples=200, noise=0.05, random_state=0)

scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)

dbscan = DBSCAN()
clusters = dbscan.fit_predict(X_scaled)

# 클러스터 할당을 표시한다.
plt.scatter(X_scaled[:,0], X_scaled[:,1],
            c=clusters,
            cmap=mglearn.cm2, s=60, edgecolors='black')
plt.xlabel("특성 0")
plt.ylabel("특성 1")
```

Out[2]: Text(0,0.5,'특성 1')



In [ ]: