# ch03 DBSCAN : density-based clustering applications with noise

### 학습 목표

- DBSCAN 알고리즘을 실습과 이론을 통해 이해하는 것을 목표로 한다.

### 학습 내용

- 01 DBSCAN은 무엇인가?
- 02 DBSCAN의 원리 이해
- 03 알고리즘 동작 이해
- 04 실습해 보기

### 목차



In [1]:

```
from IPython.display import display, Image
```

## 01. DBSCAN은 무엇인가?

목차로 이동하기

- 유용한 군집(Clustering) 알고리즘입니다.
- 데이터의 밀집 지역으로 한 클러스터를 구성한다. 비교적 데이터가 비어 있는 지역을 경계로 다른 클러스터와 구분이 된다.

### DBSCAN에 대한 장점

- DBSCAN의 장점은 클러스터 개수를 미리 지정할 필요가 없다.
- DBSCAN은 병합 군집이나 k-평균보다는 다소 느리지만 비교적 큰 데이터셋에도 적용이 가능하다.
- 어떤 클래스에도 속하지 않는 포인트를 구분할 수 있다.

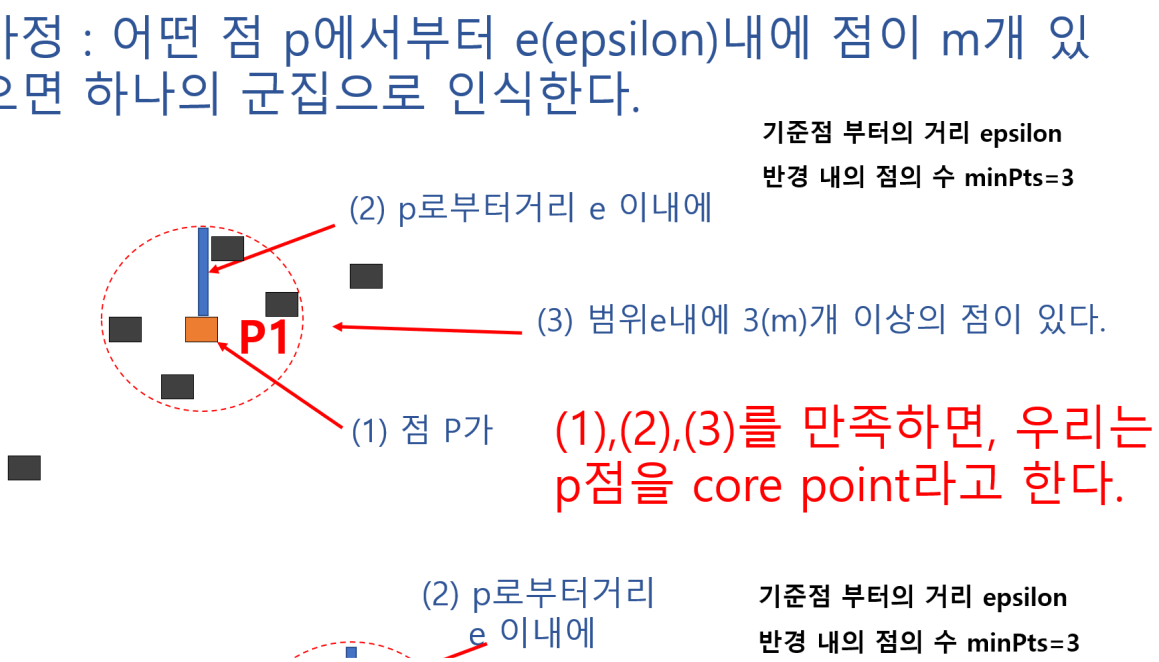
## 02. DBSCAN의 원리 이해

목차로 이동하기

알고리즘은 방문하지 않은 임의의 지점에서 시작한다. 주변 정보는 eps의 매개변수에 의해 검색된다.
(가) 특성 공간에서 가까이 있는 데이터가 많아 붐비는 지역의 포인트를 찾는다.
(나) **밀집 지역(dense region)** - 붐비는 지역을 말한다.
(다) DBSCAN의 아이디어는 데이터의 밀집 지역이 한 클러스터를 구성하며 비교적 비어있는 지역을 경계로 다른 클러스터와 구분된다는 것이다.

In [2]:

```
display(Image(filename='img/DBSCAN01.png'))
display(Image(filename='img/DBSCAN02.png'))
display(Image(filename='img/DBSCAN03.png'))
display(Image(filename='img/DBSCAN04.png'))
display(Image(filename='img/DBSCAN05.png'))
```




In [3]:

```
import mglearn
import matplotlib.pyplot as plt

### 한글
import matplotlib
from matplotlib import font_manager, rc
import platform

path = "C:/Windows/Fonts/malgun.ttf"
if platform.system() == "Windows":
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
elif platform.system()=="Darwin":
    rc('font', family='AppleGothic')
else:
    print("Unknown System")

### 마이너스 설정
from matplotlib import rc
matplotlib.rc("axes", unicode_minus=False)
```

## 03. 알고리즘 동작 이해

목차로 이동하기

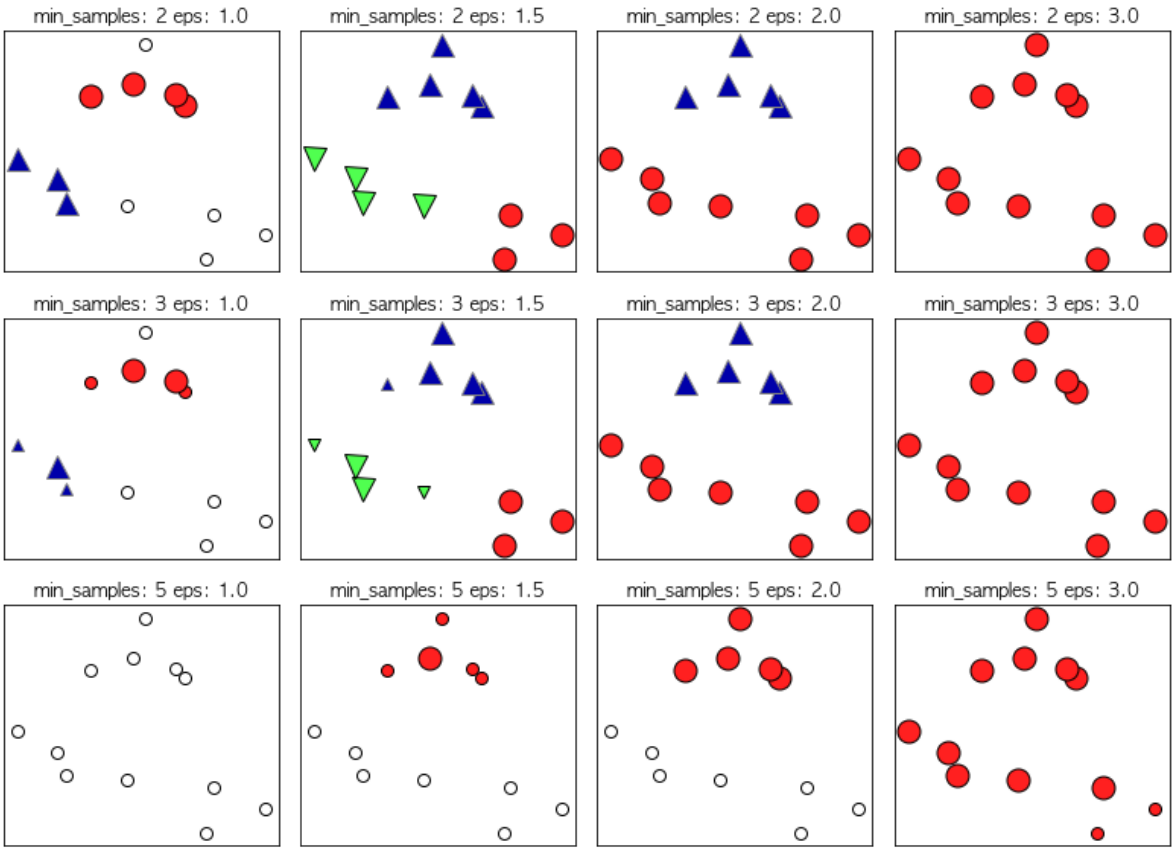
- (가) 시작할 때, 알고리즘이 방문하지 않은 무작위로 포인트를 선택
- (나) 그 포인트에서 eps거리안의 모든 포인트를 찾는다.
- (다) eps 거리안에 있는 포인트 수가 **핵심샘플도 없고, min_samples보다 적다면**
  - 어떤 클래스에도 속하지 않는 **잡음(noise)로 레이블**
- (라) eps 거리안에 **min_samples보다 많은 포인트**가 있다면 그 포인트는 **핵심 샘플(core point)**로 레이블.
  - 새로운 클러스터 레이블로 할당하고 그 포인트의 eps 거리안의 모든 이웃을 살핌.
  - 만약 어떤 클러스터에도 할당되어 있지 않으면 바로 전의 클러스터 레이블로 할당.
  - 만약 핵심 샘플이면 그 포인트의 이웃을 차례로 방문. 이런 형태로 클러스터는 eps 거리 안에 샘플이 없을 때까지 자라난다.
- (마) 그런다음 방문하지 못한 포인트를 방문하여 같은 과정을 반복함.

- DBSCAN은 클러스터의 개수를 지정할 필요는 없지만
  - eps의 값은 간접적으로 몇 개의 클러스터가 만들어질지 제어합니다.
    - eps를 매우 작게 하면 어떤 포인트도 핵심 포인트가 되지 않고, 모든 포인트가 잡음 포인트(noise point)가 될 수 있음.
    - eps를 매우 크게 하면 모든 포인트가 단 하나의 클러스터에 속하게 됨.
  - 적절한 eps값을 찾을 때는 StandardScaler나 MinMaxScaler로 모든 특성의 스케일을 비슷한 범위로 조정해 주는 것이 좋음.

In [4]:

```
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 샘플**로 분류
(2) 잡음 포인트는 흰색으로 표시
(3) eps보다 가까운 핵심 샘플은 DBSCAN에 의해 동일한 클러스터로 합쳐진다.
(4) **핵심 샘플은 크게 표시. 경계 포인트는 작게 표시**
(5) eps를 증가시키면 하나의 클러스터에 더 많은 포인트가 포함.
(6) min_samples설정은 포인트들이 잡음 포인트가 될지, 아니면 하나의 클러스터가 될지 결정하는데 중요 역할

### 포인트의 종류

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

* core point(핵심 샘플) : eps안에 min_samples보다 같거나 많다.
* border point(경계 포인트) : eps안에 min_samples보다 작은데, core point가 있다.
* noise point(잡음 포인트) : eps안에 min_samples보다 작고, 포인트중에 core point 가 없다. 어떤 클래스에소 소속되지 않는다.

## 04. 실습해보기

목차로 이동하기

### 데이터 준비 및 스케일 조정

In [5]:

```
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)

# 평균 0, 분산 1이 되도록 데이터 스케일 조정
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
```

- 병합군집과 마찬가지로 DBSCAN은 새로운 데이터에 대해 예측할 수 없어, fit_predict메소드를 사용

In [6]:

```
dbscan = DBSCAN()  # eps = 0.5, min_samples=5로 기본값
clusters = dbscan.fit_predict(X_scaled)
clusters
```

Out[6]:

```
array([0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
1,
       1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
0,
       0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
0,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0,
0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
1,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
1,
       0, 1])
```

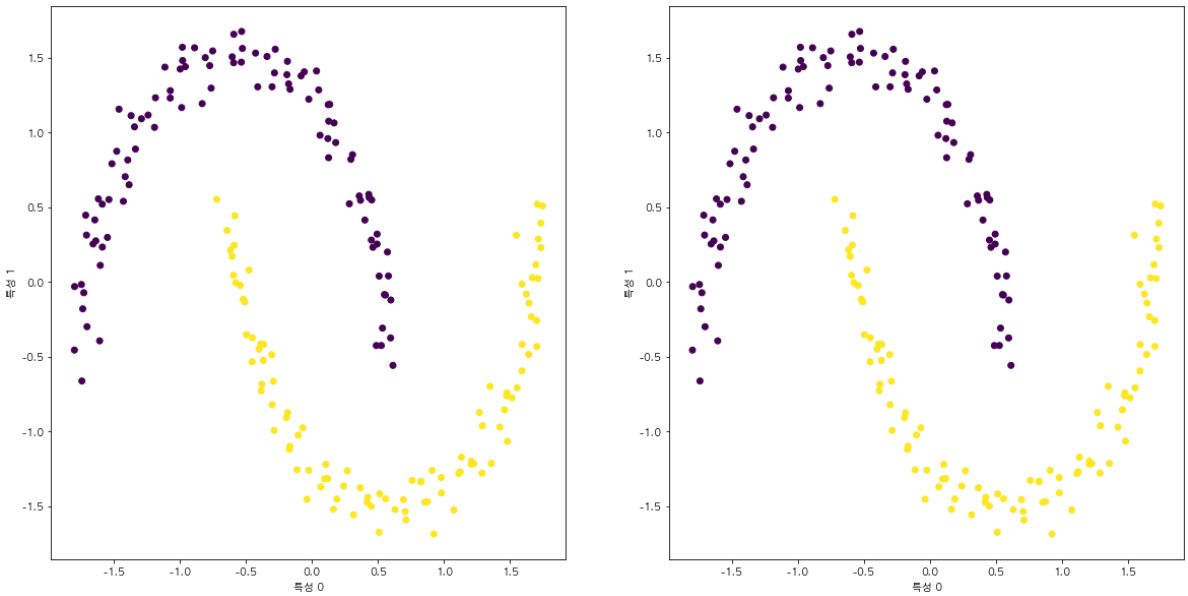
In [7]:

```
# 원래 데이터 그래프
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)    # 2행 1열에 첫번째
f1 = X_scaled[:, 0]  # 특성 첫번째 선택
f2 = X_scaled[:, 1]  # 특성 두번째 선택
plt.scatter(f1, f2, c=y)  # 원본 샘플 데이터
plt.xlabel("특성 0")
plt.ylabel("특성 1")

# DBSCAN 적용한 그래프
plt.subplot(1,2,2)    # 2행 1열에 두번째
f1 = X_scaled[:, 0]  # 특성 첫번째 선택
f2 = X_scaled[:, 1]  # 특성 두번째 선택
plt.scatter(f1, f2, c=clusters) # DBSCAN 적용한 샘플 데이터
plt.xlabel("특성 0")
plt.ylabel("특성 1")
```

Out[7]:

```
Text(0, 0.5, '특성 1')
```




- 예상한 클러스터 개수(2개)를 만들어내므로 매개변수 설정이 맞음.
- eps를 0.7로 올리면 하나의 클러스터를 만들어냄.
- DBSCAN을 사용할 때, 클러스터 할당값을 주의해서 다루어야 함.

## 실습

- K-means 알고리즘(eps 0.7일 때, 을 적용한 y_pred를 구하고, 그래프 표시해 보기(3개)

In [8]:

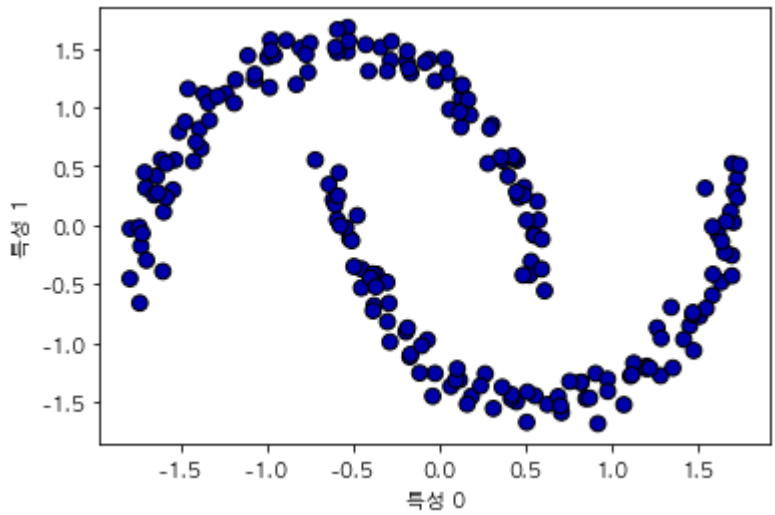
```
dbscan = DBSCAN(eps=0.7, min_samples = 6)
clusters = dbscan.fit_predict(X_scaled)

f1 = X_scaled[:,0] # 첫번째 특성
f2 = X_scaled[:,1] # 두번째 특성

# 클러스터 할당을 표시한다.
plt.scatter(f1, f2, c=clusters,
            cmap=mglearn.cm2, s=60, edgecolors='black')
plt.xlabel("특성 0")
plt.ylabel("특성 1")
```

Out[8]:

```
Text(0, 0.5, '특성 1')
```




## REF ¶

- https://towardsdatascience.com/dbscan-algorithm-complete-guide-and-application-with-python-scikit-learn-d690cbae4c5d (https://towardsdatascience.com/dbscan-algorithm-complete-guide-and-application-with-python-scikit-learn-d690cbae4c5d)

## 실습 풀이

- K-means 알고리즘을 적용한 y_pred를 구하고, 그래프 표시해 보기(3개)