CH03 군집(Clustering) - Kmeans

학습 내용

- 01 Clustering(군집)의 목적
- 02 k-평균 알고리즘으로 찾은 클러스터 중심과 클러스터 경계
- 03 k-means 알고리즘 적용
- 04 생성된 데이터의 K-means 군집 모델 적용
- 05 군집 모델(K-means)가 주의점

목차

라이브러리 불러오기 및 데이터 준비

목차로 이동하기

In [1]:

```
### 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)
```

01. Clustering(군집)의 목적

목차로 이동하기

- 한 클러스터 안의 데이터 포인트끼리는 매우 비슷. 다른 클러스터의 데이터 포인터와는 구분되도록 데이터를 나누는 것이 목표입니다.

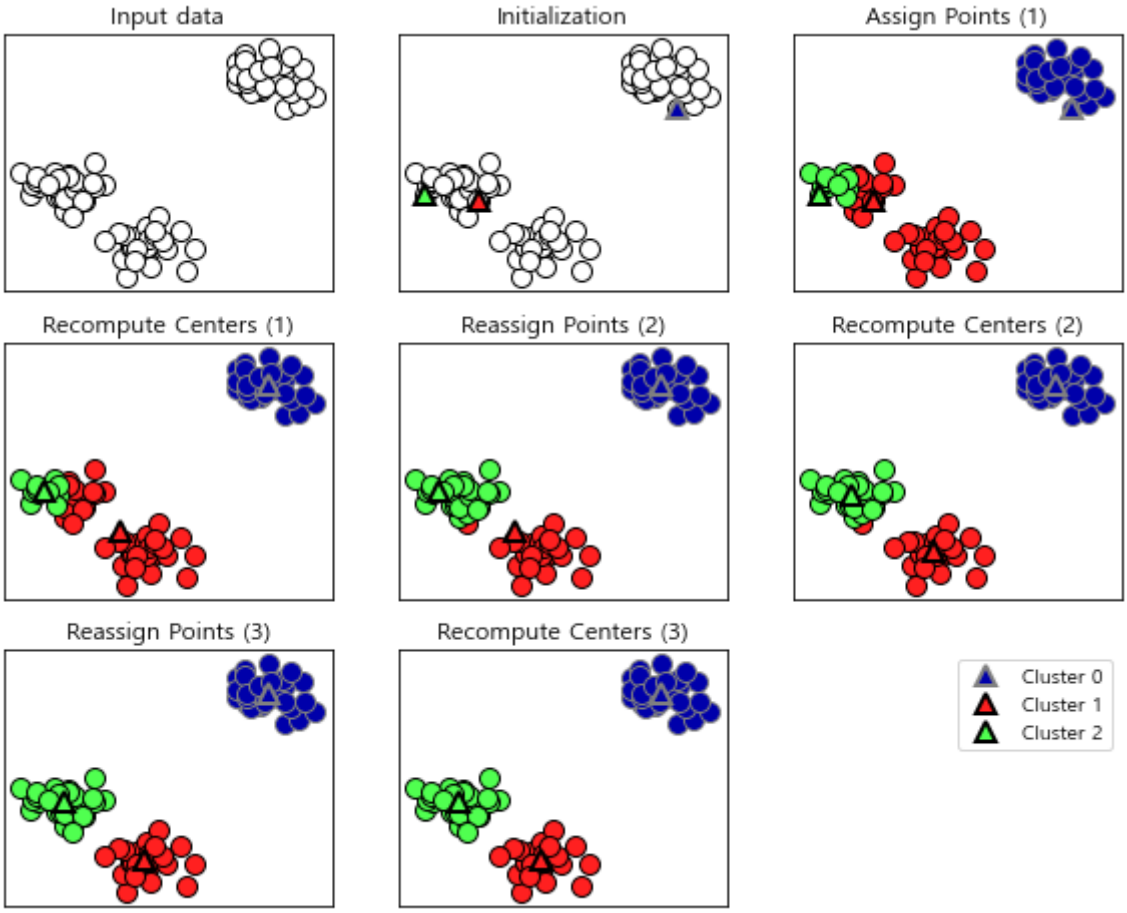
- 분류 알고리즘과 비슷하게 군집 알고리즘은 각 데이터 포인트가 어느 클러스터에 속하는지 할당(또는 예측)합니다.

In [2]:

```
import mglearn
%matplotlib inline
```

In [3]:

```
mglearn.plots.plot_kmeans_algorithm()
```



- 삼각형은 클러스터의 중심이고 원은 데이터 포인트
- 클러스터는 색으로 구분

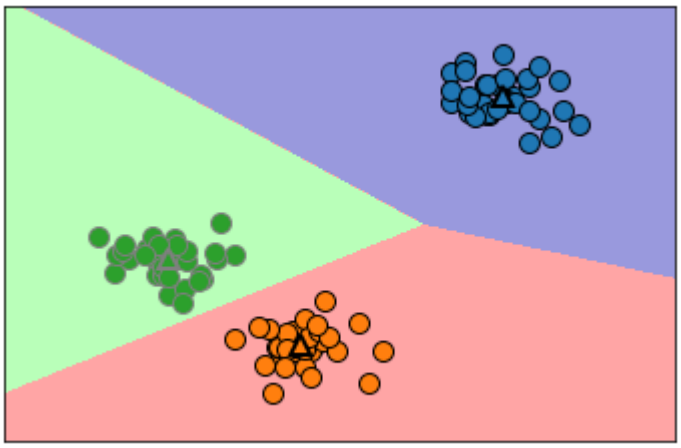
02. k-평균 알고리즘으로 찾은 클러스터 중심과 클러스터 경계

목차로 이동하기

- k-평균 군집은 가장 간단하며 널리 사용하는 군집 알고리즘
- 이 알고리즘은 데이터의 어떤 영역을 대표하는 클러스터 중심(cluster center)을 찾는다. 학습을 시킨 클러스터 중심의 경계

In [4]:

```
mglearn.plots.plot_kmeans_boundaries()
```



03. k-means 알고리즘 적용

목차로 이동하기

In [5]:

```
from sklearn.datasets import make_moons
from sklearn.cluster import KMeans

# 데이터 만들기(2차원 데이터)
X, y = make_moons(n_samples=200, noise=0.05, random_state=0)
print(X.shape, y.shape)
```

```
(200, 2) (200,)
```

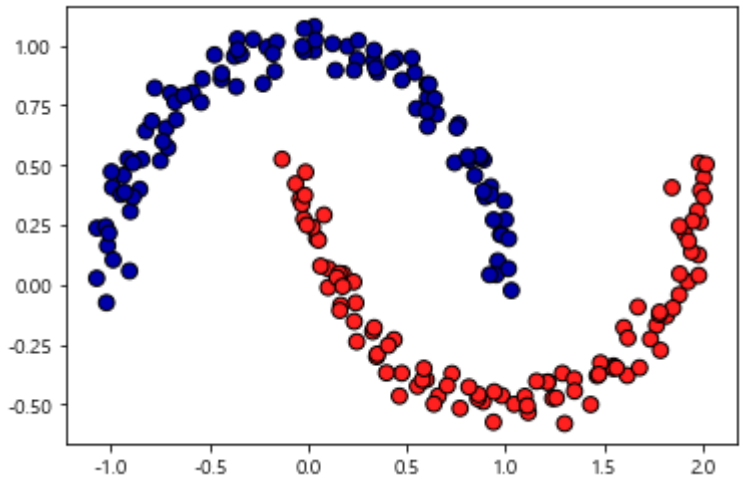
데이터 시각화

In [6]:

```
import matplotlib.pyplot as plt

# 클러스터 할당과 클러스터 중심을 표시한다.
# 특성 1, 특성 2
X1 = X[: ,0]
X2 = X[: ,1]
plt.scatter(X1, X2,
            c=y,
            cmap=mglearn.cm2, s=60, edgecolors='k')
```

Out[6]:

```
<matplotlib.collections.PathCollection at 0x211684b16d0>
```



K-Means 알고리즘 적용

In [7]:

```
# 두 개의 클러스터로 데이터에 KMeans 알고리즘 적용
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
y_pred = kmeans.predict(X)
y_pred
```

Out[7]:

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

In [8]:

```
print(X.shape, y.shape)
print(X[1:5])
print(X[:, 0][1:5])   # X의 첫번째 열에서 1행부터~4행까지
print(X[:, 1][1:5])   # X의 두번째 열에서 1행부터~4행까지
```

```
(200, 2) (200,)
[[ 1.61859642 -0.37982927]
 [-0.02126953  0.27372826]
 [-1.02181041 -0.07543984]
 [ 1.76654633 -0.17069874]]
[ 1.61859642 -0.02126953 -1.02181041  1.76654633]
[-0.37982927  0.27372826 -0.07543984 -0.17069874]
```

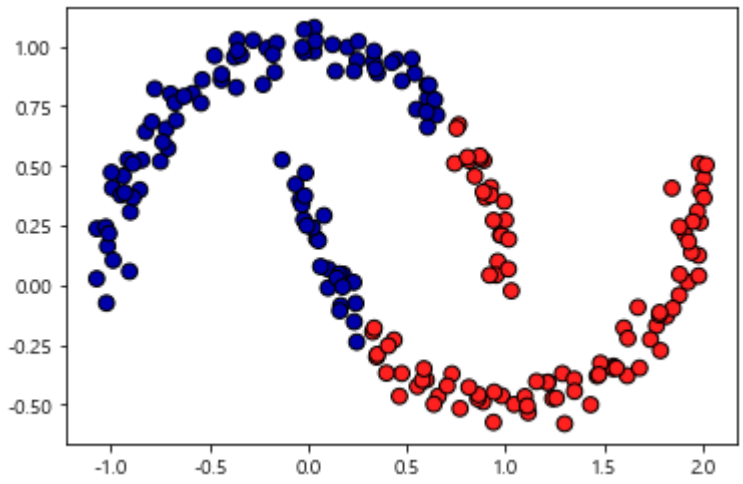
K-means 을 적용하여 할당한 클러스터 시각화

In [9]:

```
import matplotlib.pyplot as plt

# 클러스터 할당과 클러스터 중심을 표시한다.
X1 = X[: ,0]
X2 = X[: ,1]
plt.scatter(X1, X2, c=y_pred,
            cmap=mglearn.cm2, s=60, edgecolors='k')
```

Out[9]:

```
<matplotlib.collections.PathCollection at 0x211685c4b80>
```



In [10]:

```
# 클러스터의 중심
print(kmeans.cluster_centers_)
print(kmeans.cluster_centers_[ 0 , :])  # 클러스터1 의 중심 X, Y
print(kmeans.cluster_centers_[ 1 , :])  # 클러스터2 의 중심 X, Y
```

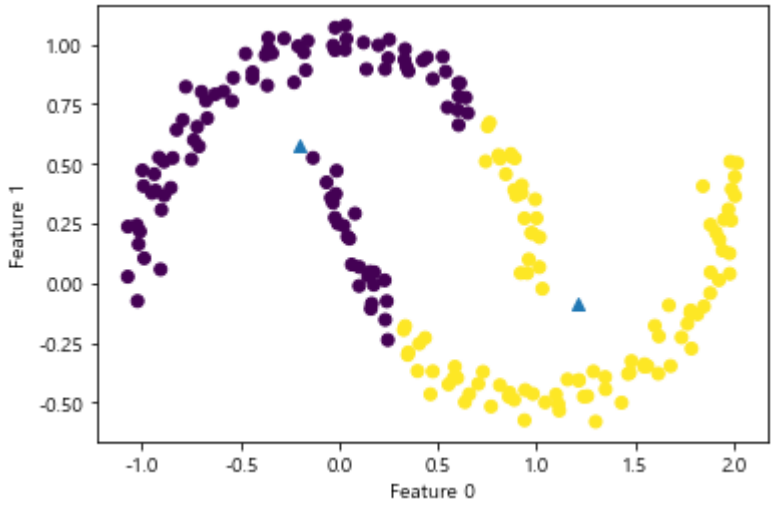
```
[[-0.2003285   0.58035606]
 [ 1.20736718 -0.0825517 ]]
[-0.2003285   0.58035606]
[ 1.20736718 -0.0825517 ]
```

In [11]:

```
## 그래프 위에 클러스터의 중심 표시
centerX = kmeans.cluster_centers_[ : , 0] # 중심 X좌표 세트
centerY = kmeans.cluster_centers_[ : , 1] # 중심 Y좌표 세트

plt.scatter(X1, X2, c=y_pred)
plt.scatter(centerX, centerY, marker="^")
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```

Out[11]:

```
Text(0, 0.5, 'Feature 1')
```



04. 생성된 데이터의 K-means 군집 모델 적용

목차로 이동하기

In [12]:

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import numpy as np
```

make_blobs의 데이터 셋은 3개의 그룹

- 이를 이용하여 실제 K-Means를 적용시키고 이를 확인해 보자.

In [13]:

```
# 인위적으로 2차원 데이터 생성
X, y = make_blobs(random_state=1)

# 데이터 확인
print(X.shape)
print(y.shape)
print(np.unique(y))  # y : 0,1,2를 갖는 값.
```

```
(100, 2)
(100,)
[0 1 2]
```

In [14]:

```
# 군집 모델 만들기 (그룹이 3개)
model = KMeans(n_clusters=3)
model.fit(X)

# 레이블 확인
print("클러스터 레이블:₩n{}".format(model.labels_))

print("예측값")
# 군집화 시키기
print(model.predict(X))
```

```
클러스터 레이블:
[1 0 0 0 2 2 2 0 1 1 0 0 2 1 2 2 2 1 0 0 2 0 2 1 0 2 2 1 1 2 1 1 2 1 0 2 0
 0 0 2 2 0 1 0 0 2 1 1 1 1 0 2 2 2 1 2 0 0 1 1 0 2 2 0 0 2 1 2 1 0 0 0 2 1
 1 0 2 2 1 0 1 0 0 2 1 1 1 1 0 1 2 1 1 0 0 2 2 1 2 1]
예측값
[1 0 0 0 2 2 2 0 1 1 0 0 2 1 2 2 2 1 0 0 2 0 2 1 0 2 2 1 1 2 1 1 2 1 0 2 0
 0 0 2 2 0 1 0 0 2 1 1 1 1 0 2 2 2 1 2 0 0 1 1 0 2 2 0 0 2 1 2 1 0 0 0 2 1
 1 0 2 2 1 0 1 0 0 2 1 1 1 1 0 1 2 1 1 0 0 2 2 1 2 1]
```

현 데이터와 Kmean으로 예측한 결과 비교

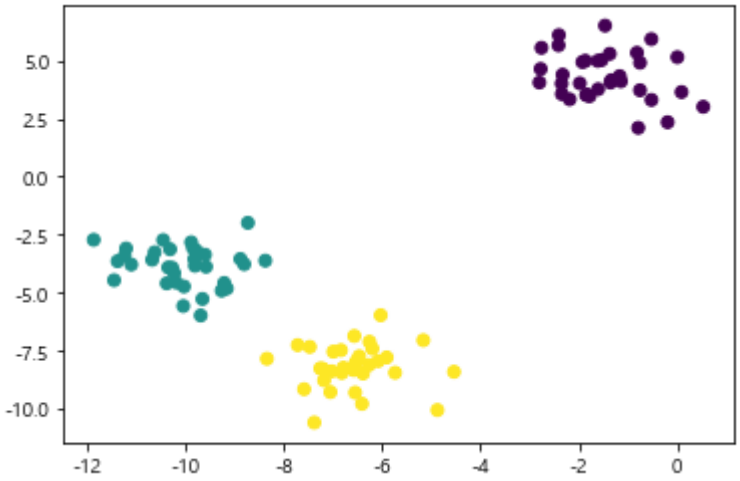
원본 데이터

In [15]:

```
X1 = X[:, 0] # 첫번째 열
X2 = X[:, 1] # 두번째 열

plt.scatter(X1, X2, c=y)
```

Out[15]:

```
<matplotlib.collections.PathCollection at 0x2116818bdc0>
```



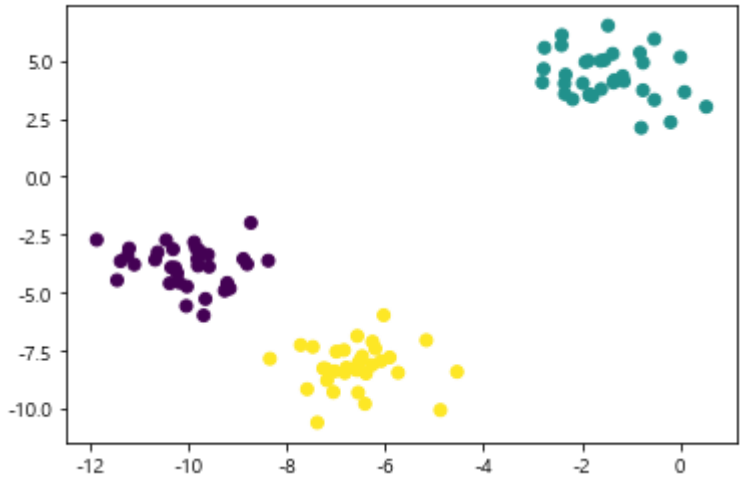
K-mean 적용한 결과 시각화

In [16]:

```
X1 = X[:, 0] # 첫번째 열
X2 = X[:, 1] # 두번째 열

y_pred = model.predict(X)

plt.scatter(X1, X2, c=y_pred)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x21168051670>
```



- 원본 데이터와 거의 일치되도록 예측 수행

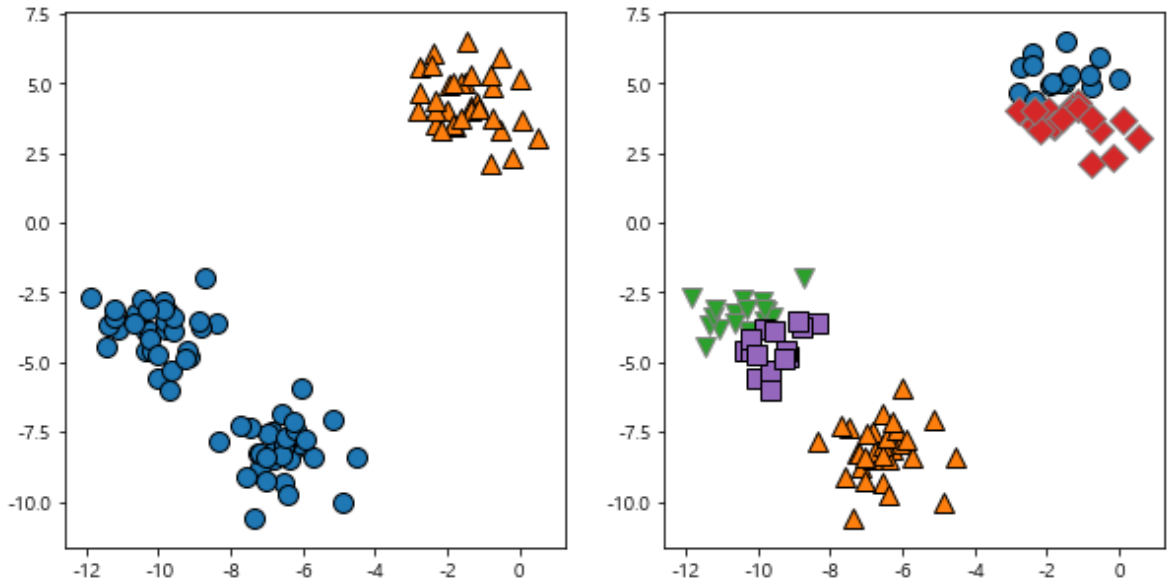
클러스터 개수의 설정을 줄이고, 늘리기

In [17]:

```
fig, axes = plt.subplots(1,2, figsize=(10,5))

# 두개의 클러스터 중심을 사용.
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
assignments = kmeans.labels_

mglearn.discrete_scatter(X[:,0], X[:,1],
                         assignments, ax=axes[0])

# 다섯개의 클러스터 중심을 사용.
kmeans = KMeans(n_clusters=5)
kmeans.fit(X)
assignments =  kmeans.labels_

mglearn.discrete_scatter(X[:,0], X[:,1],
                         assignments, ax=axes[1])
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x2116801a2b0>,
 <matplotlib.lines.Line2D at 0x2116801a610>,
 <matplotlib.lines.Line2D at 0x2116801a970>,
 <matplotlib.lines.Line2D at 0x2116801acd0>,
 <matplotlib.lines.Line2D at 0x21168023070>]
```



05. 군집 모델(K-means)의 주의점

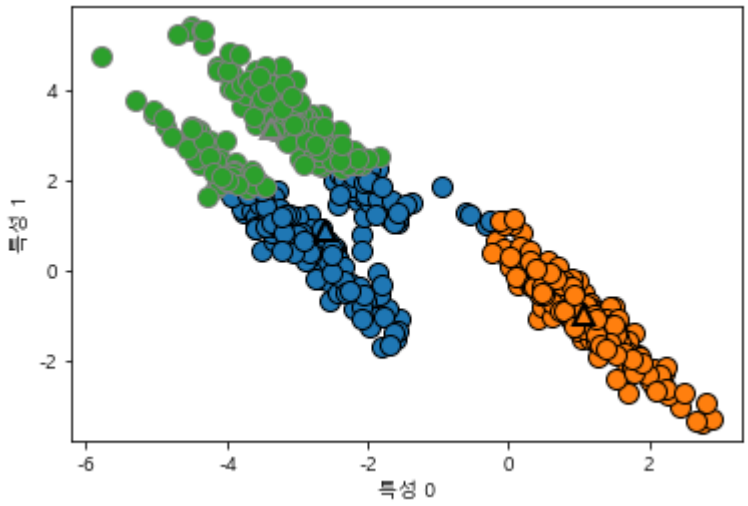
목차로 이동하기

In [18]:

```
# 무작위로 데이터 생성
X, y = make_blobs(random_state=170, n_samples=600)
rng = np.random.RandomState(74)

# 데이터가 길게 늘어지도록 변경한다.
transformation = rng.normal(size=(2,2))
X = np.dot(X, transformation)

# 세 개의 클러스터로 데이터에 KMeans 알고리즘을 적용.
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
y_pred = kmeans.predict(X)

# 클러스터 할당과 클러스터 중심을 나타낸다.
mglearn.discrete_scatter(X[:,0],
                         X[:,1],
                         kmeans.labels_,
                         markers='o')

mglearn.discrete_scatter( kmeans.cluster_centers_[:,0],
                          kmeans.cluster_centers_[:,1], [0,1,2],
                        markers="^",
                        markeredgewidth=2)

plt.xlabel('특성 0')
plt.ylabel('특성 1')
```

Out[18]:

```
Text(0, 0.5, '특성 1')
```



- 거리가 가까운 것을 클러스터로 만들어, 원형이 아닐경우, 클러스터를 구분하지 못하는 k-means(평균) 알고리즘