CH03 비지도학습 - PCA

학습 내용

- 1. 왜 비지도 학습을 사용하는가?
- 2. PCA(Principal component analysis)
- 3. IRIS 데이터 셋을 이용한 PCA 예제 실습

01 왜 비지도학습을 사용하는가?

가. 시각화한다.
나. 데이터를 많은 feature를 몇개의 압축적인 feature(특성)으로 줄인다.
다. 추가적인 처리(주로 지도학습에 이용하기 위해)

02 PCA(Principal component analysis)

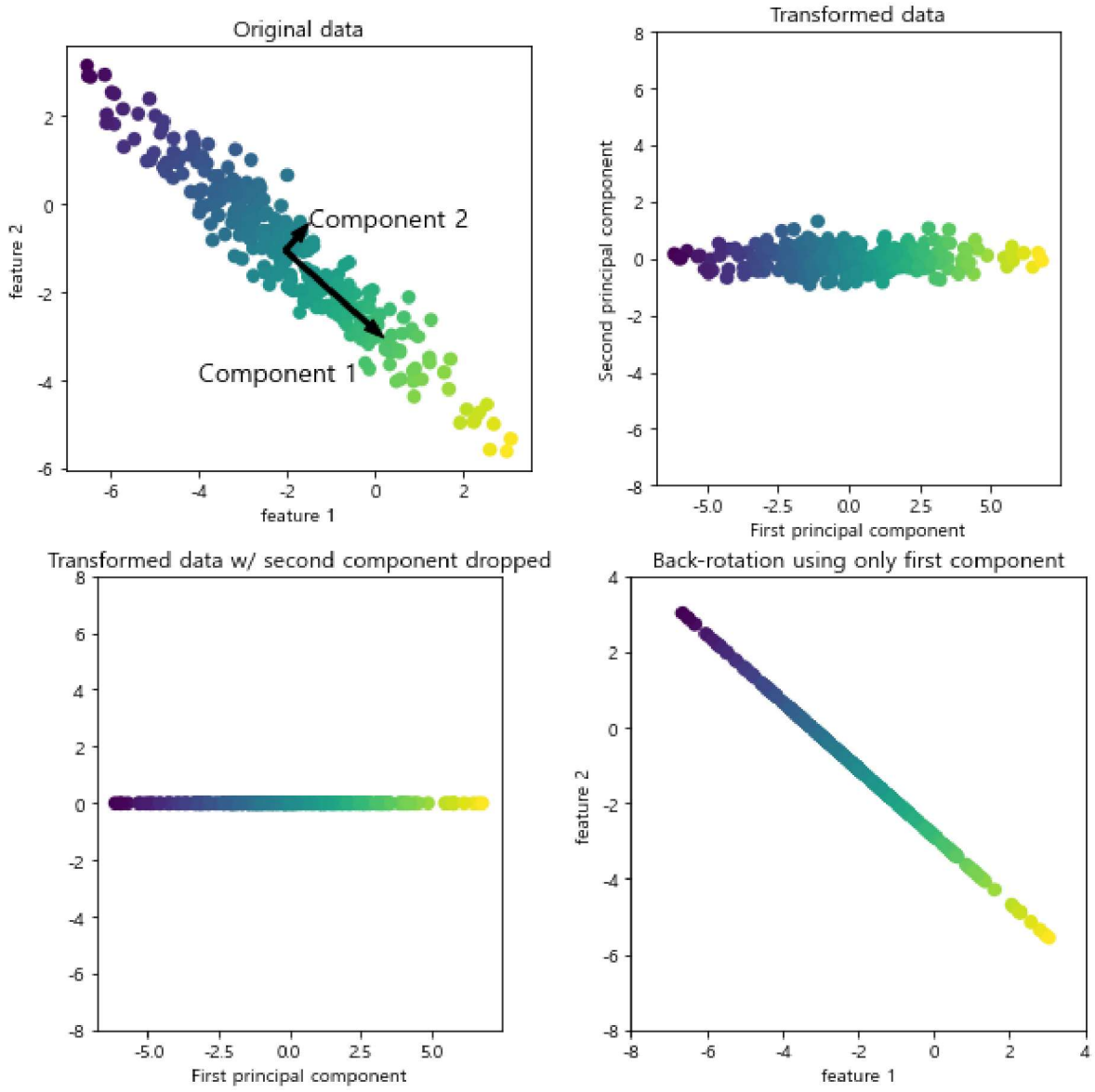
- 주성분 분석은 특성들이 **통계적으로 상관관계가 없도록** 데이터셋을 **회전시키는 기술**이다.
- 데이터의 회전 후에 데이터 설명도에 따라 얼마나 중요한가가 판단되고, 새로운 데이터 중의 **일부 데이터가 선택**된다.

In [6]:

```
import mglearn
```

In [7]:

```
mglearn.plots.plot_pca_illustration()
```



네개의 그래프는 주성분 1을 찾아가는 과정을 보여준다.

첫번째 그래프

- 원본 데이터 포인터를 색으로 구분해 표시
- 그래프상의 화살표의 머리와 꼬리는 의미가 없다.

PCA 알고리즘

- STPE 1. PCA 알고리즘은 제일 먼저 **'성분1'**의 분산이 가장 큰 방향을 찾는다.
  - 이 방향(또는 벡터)이 **데이터에서 가장 많은 정보를 담고 있는 방향이 된다.**
  - 또 다른 말로 특성들의 상관관계가 가장 큰 방향입니다.
- STEP 2. 첫 번째 방향과 직각인 방향 중, 가장 많은 정보를 담는 방향을 찾는다.(두번째 주성분)
  - 만약 특성이 2개인 2차원에서는 직각방향이 하나만 존재,
  - 특성이 여러개인 고차원에서는 첫번째 성분과 직각을 이루는 것은 무한히 많을 수 있다.
- STEP 3. 이어 같은 방법으로 성분을 찾아간다.

- 이런 과정을 거쳐 찾은 방향의 데이터에 있는 주된 분산의 방향이라고 해서 주성분이라 한다.
  - 일반적으로 원본 특성 개수만큼의 주성분이 있다.

두번째 그래프

- (가) 첫번째와 같은 데이터이지만, 주성분1과 2를 각각 x축과 y축에 나란하도록 회전. 회전하기 전에 데이터에서 평균을 빼서 중심을 원점을 맞춤.
- (나) PCA에 의해 회전된 두 축은 연관되어 있지 않다. 독립적이다. 독립적이므로 상관관계 행렬(correlation matrix)이 대각선 방향을 제외하고 0이된다.

PCA는 주성분의 일부만 남기는 차원 축소 용도로 사용.

세번째 그래프

- (가) 세번째 그래프는 첫 번째 주성분만 유지하려고 함. 2차원 -> 1차원으로 감소
  - 주성분 1위로 데이터를 투영시키기
  - 원본 특성중 하나만 남기는 것이 아닌 가장 유용한 방향을 찾은 주성분1만 유지.

네번째 그래프

- (가) 다시 데이터에 평균을 더해서 반대로 회전.
- (나) 이 데이터 포인트들은 원래 특성 공간에 있지만 첫번째 주성분만 담고 있다.

이 변환은 데이터에서 노이즈를 제거하거나 주성분에서 유지되는 정보를 시각화

PCA를 적용한 유방암 데이터셋 시각화

- 특성이 많을 때는 산점도 행렬을 적용하기 어렵다. 이 특성은 435개의 산점도를 그려야 합니다.
- 이 보다 쉬운 방법은 양성과 악성 두 클래스에 대해 각 특성의 히스토그램을 그리는 것이다.

In [8]:

```
# 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)

matplotlib.rcParams['axes.unicode_minus'] = False
```

PCA를 적용하여 유방암 데이터 셋 시각화하기

In [9]:

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
import numpy as np
```

유방암 특성들을 히스토그램 시각화를 이용하여 특성을 확인(악성.양성정보 포함)

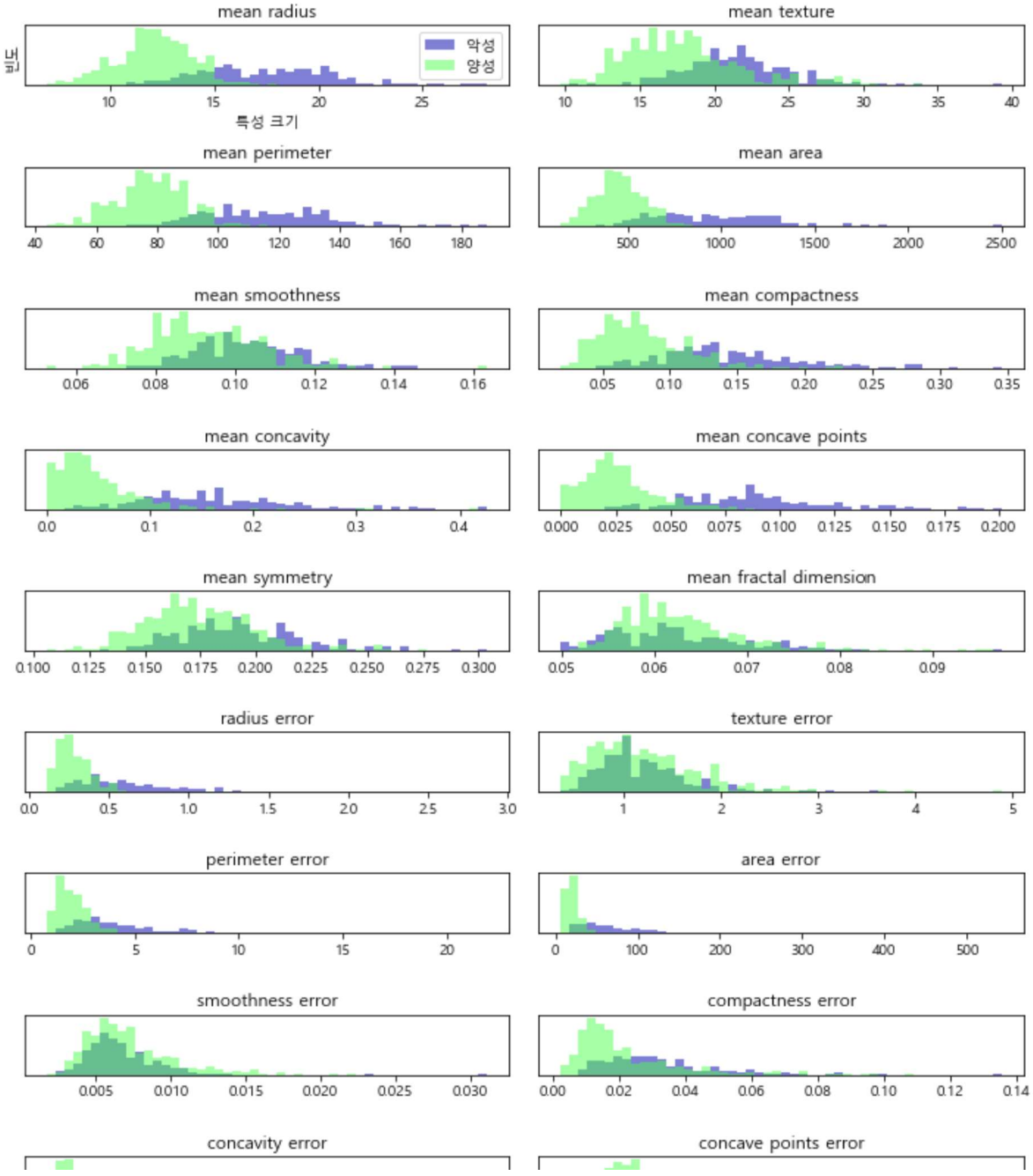
In [10]:

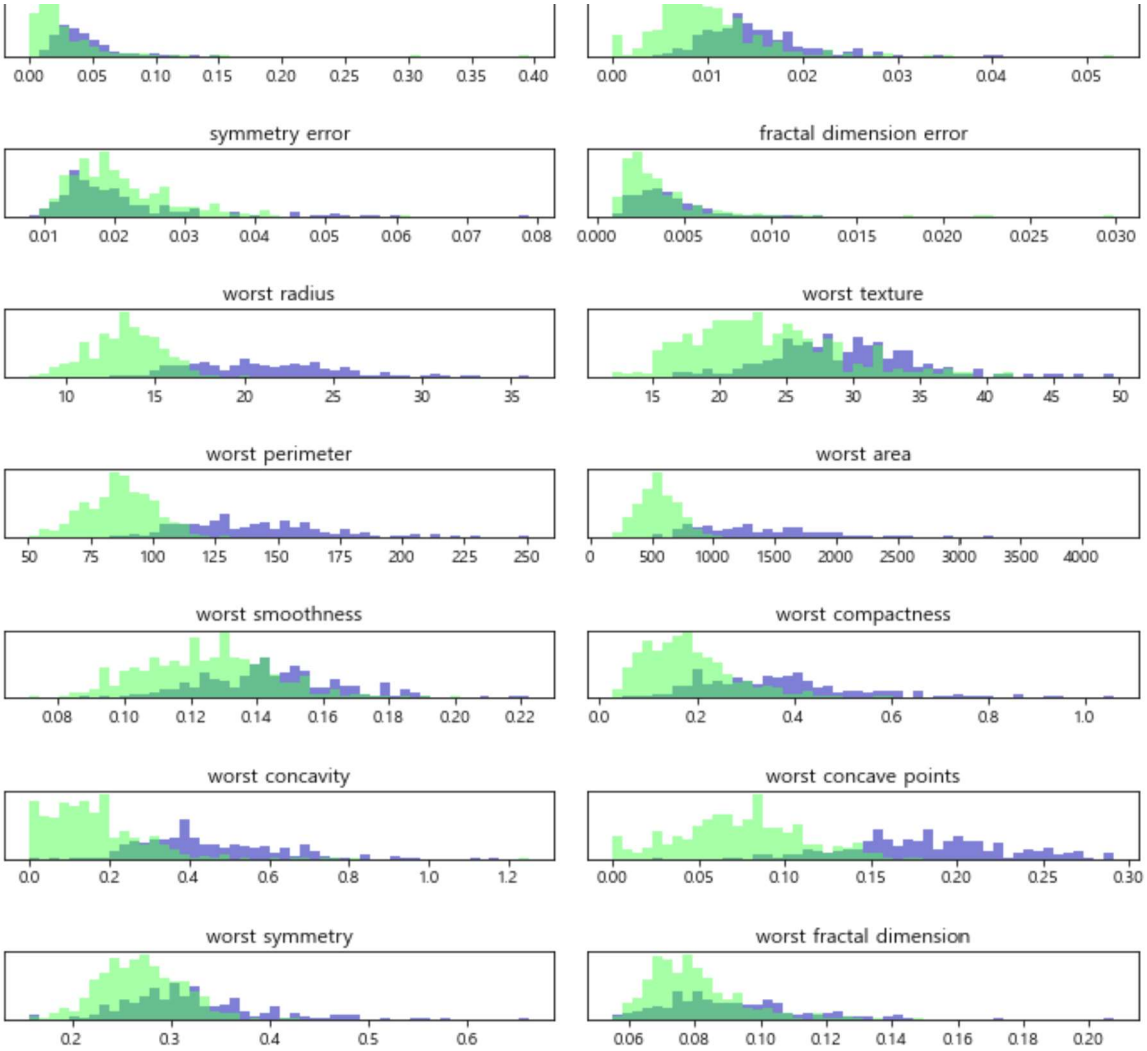
```
cancer = load_breast_cancer()
fig, axes = plt.subplots(15, 2, figsize=(10, 20))
malignant = cancer.data[cancer.target == 0]
benign = cancer.data[cancer.target == 1]

ax = axes.ravel()

for i in range(30):
    _, bins = np.histogram(cancer.data[:, i], bins=50)
    ax[i].hist(malignant[:, i], bins=bins, color=mglearn.cm3(0), alpha=.5)
    ax[i].hist(benign[:, i], bins=bins, color=mglearn.cm3(2), alpha=.5)
    ax[i].set_title(cancer.feature_names[i])
    ax[i].set_yticks(())

ax[0].set_xlabel("특성 크기")
ax[0].set_ylabel("빈도")
ax[0].legend(["악성", "양성"], loc="best")
fig.tight_layout()
```





- 히스토그램을 두개 겹쳐 놓은 것으로
  - 초록색은 양성 클래스의 포인트, 푸른색은 악성 클래스의 포인트를 나타낸다.
- 이를 통해 어떤 특성이 양성과 악성을 구분하는 데 더 좋은지 가늠해 볼 수 있습니다.
  - 'smoothness error'는 거의 겹쳐서 별로 쓸모가 없음.
  - 'worst concave points'는 두 히스토그램이 확실히 구분되어 매우 유용한 특성

데이터 전처리 및 PCA 변환

데이터 변환 - 표준화

In [11]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler()
scaler.fit(cancer.data)
X_scaled = scaler.transform(cancer.data)
```

PCA 적용

- PCA 변환을 학습하고 적용하는 것.

- (1) PCA 객체를 생성하고, 몇개의 성분을 사용할 것인가?
- (2) fit 메서드를 호출하여 주성분을 찾고,
- (3) transform 메서드를 호출하여 데이터를 회전시키고 차원 축소
  - 기본값일 때는 PCA는 데이터를 회전만(이동)만 시키고 모든 주성분을 유지한다.

In [12]:

```
# 데이터의 처음 두 개 주성분만 유지
pca = PCA(n_components=2)

# 유방암 데이터로 PCA 모델을 생성
pca.fit(X_scaled)

# 주어진 PCA 이용하여 두 개의 주성분을 가져온다.
X_pca = pca.transform(X_scaled)
print("원본 데이터 형태: {}".format(str(X_scaled.shape)))
print("축소된 데이터 형태: {}".format(str(X_pca.shape)))
```

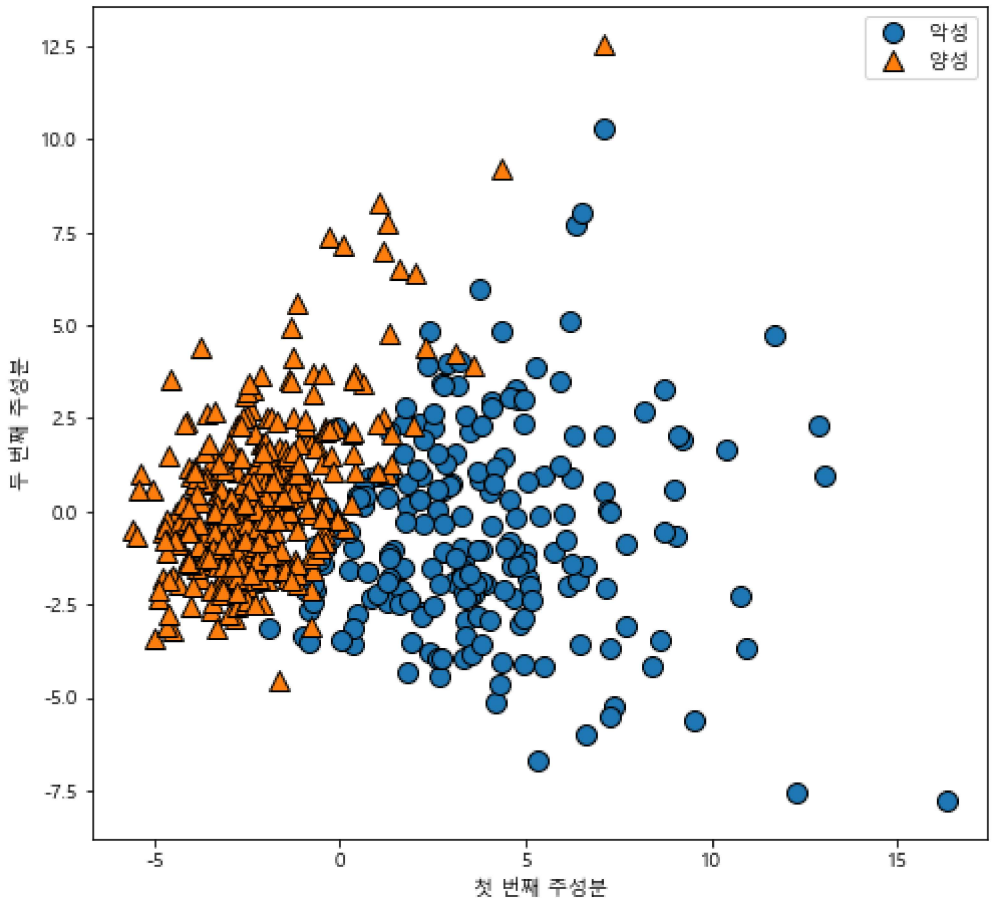
```
원본 데이터 형태: (569, 30)
축소된 데이터 형태: (569, 2)
```

In [13]:

```
# 클래스를 색깔로 구분하여 처음 두 개의 주성분을 그래프로 표시
plt.figure(figsize=(8, 8))
mglearn.discrete_scatter(X_pca[:, 0], X_pca[:, 1], cancer.target)
plt.legend(["악성", "양성"], loc="best")
plt.gca().set_aspect("equal")
plt.xlabel("첫 번째 주성분")
plt.ylabel("두 번째 주성분")
```

Out[13]:

```
Text(0, 0.5, '두 번째 주성분')
```



주성분 확인

In [16]:

```
print("PCA 주성분 형태 ", pca.components_.shape)
print("PCA 주성분 :", pca.components_) # 두개의 주성분에 대한 결합- 회귀 계수
```

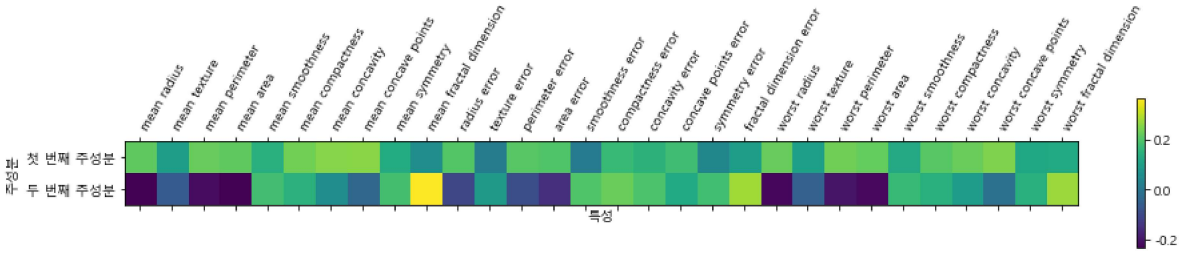
```
PCA 주성분 형태  (2, 30)
PCA 주성분 : [[ 0.21890244  0.10372458  0.22753729  0.22099499  0.14258969  0.239285
35
   0.25840048  0.26085376  0.13816696  0.06436335  0.20597878  0.01742803
   0.21132592  0.20286964  0.01453145  0.17039345  0.15358979  0.1834174
   0.04249842  0.10256832  0.22799663  0.10446933  0.23663968  0.22487053
   0.12795256  0.21009588  0.22876753  0.25088597  0.12290456  0.13178394]
 [-0.23385713 -0.05970609 -0.21518136 -0.23107671  0.18611302  0.15189161
   0.06016536 -0.0347675   0.19034877  0.36657547 -0.10555215  0.08997968
  -0.08945723 -0.15229263  0.20443045  0.2327159   0.19720728  0.13032156
   0.183848    0.28009203 -0.21986638 -0.0454673  -0.19987843 -0.21935186
   0.17230435  0.14359317  0.09796411 -0.00825724  0.14188335  0.27533947]]
```

In [17]:

```
plt.matshow(pca.components_, cmap='viridis')
plt.yticks([0, 1], ["첫 번째 주성분", "두 번째 주성분"])
plt.colorbar()
plt.xticks(range(len(cancer.feature_names)),
           cancer.feature_names, rotation=60, ha='left')
plt.xlabel("특성")
plt.ylabel("주성분")
```

Out[17]:

```
Text(0, 0.5, '주성분')
```



03 IRIS 데이터 셋을 이용한 PCA 예제 실습

In [31]:

```
from sklearn.decomposition import PCA
import seaborn as sns
%matplotlib inline
```

In [32]:

```
iris = sns.load_dataset("iris")

X_iris = iris.drop("species", axis=1)
y_iris = iris['species']

print(X_iris.shape, y_iris.shape)
```

```
(150, 4) (150,)
```

In [33]:

```
X_iris.head()
```

Out[33]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [34]:

```
model = PCA()
X_pca_dat = model.fit(X_iris).transform(X_iris)
print(X_pca_dat.shape, type(X_pca_dat) )
```

```
(150, 4) <class 'numpy.ndarray'>
```

In [35]:

```
X_pca_dat[0:4]
```

Out[35]:

```
array([[-2.68412563e+00,  3.19397247e-01, -2.79148276e-02,
        -2.26243707e-03],
       [-2.71414169e+00, -1.77001225e-01, -2.10464272e-01,
        -9.90265503e-02],
       [-2.88899057e+00, -1.44949426e-01,  1.79002563e-02,
        -1.99683897e-02],
       [-2.74534286e+00, -3.18298979e-01,  3.15593736e-02,
         7.55758166e-02]])
```

In [36]:

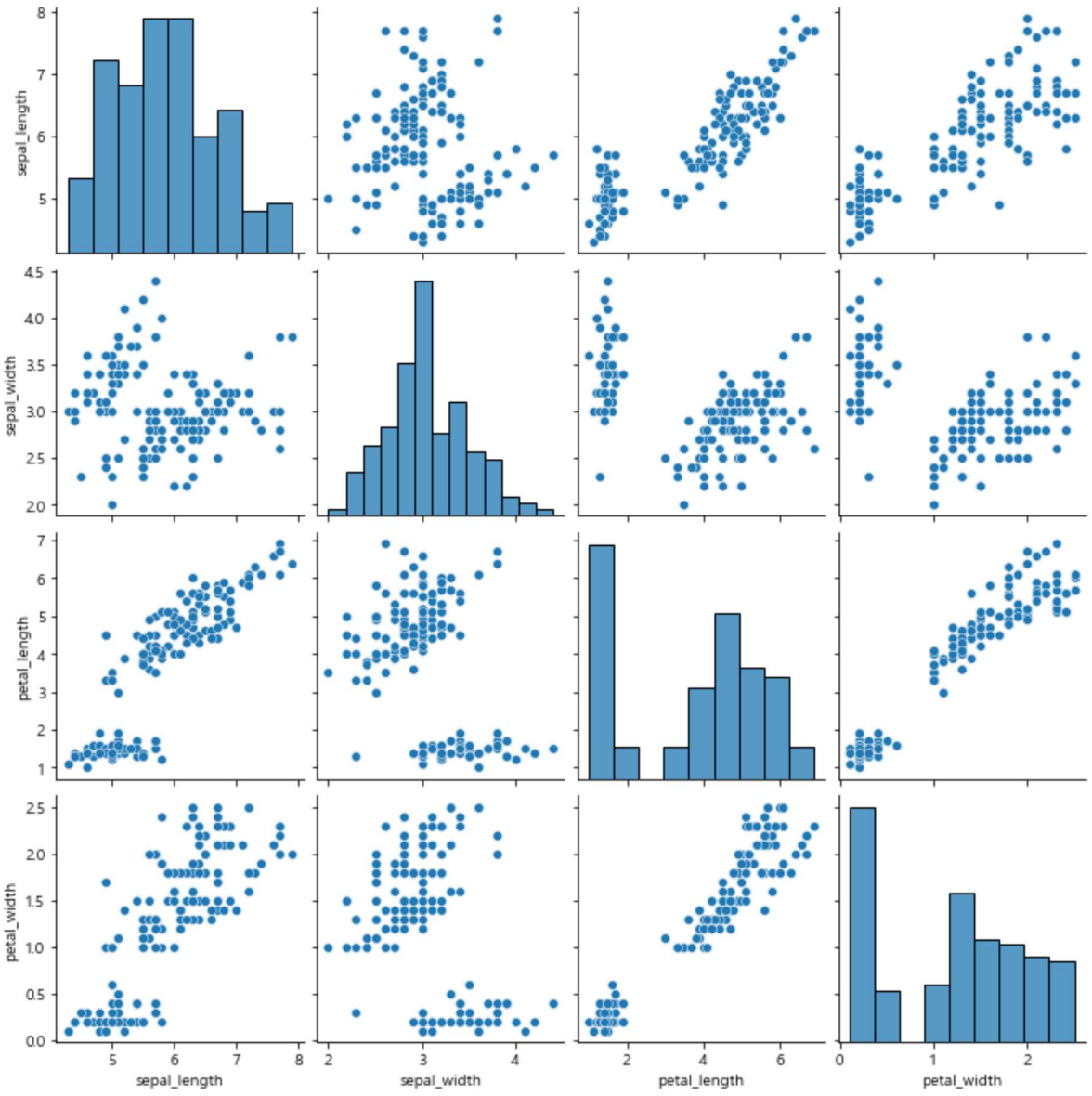
```
import seaborn as sns
```

In [37]:

```
sns.pairplot(iris)
```

Out[37]:

```
<seaborn.axisgrid.PairGrid at 0x1bbfb820c70>
```



In [38]:

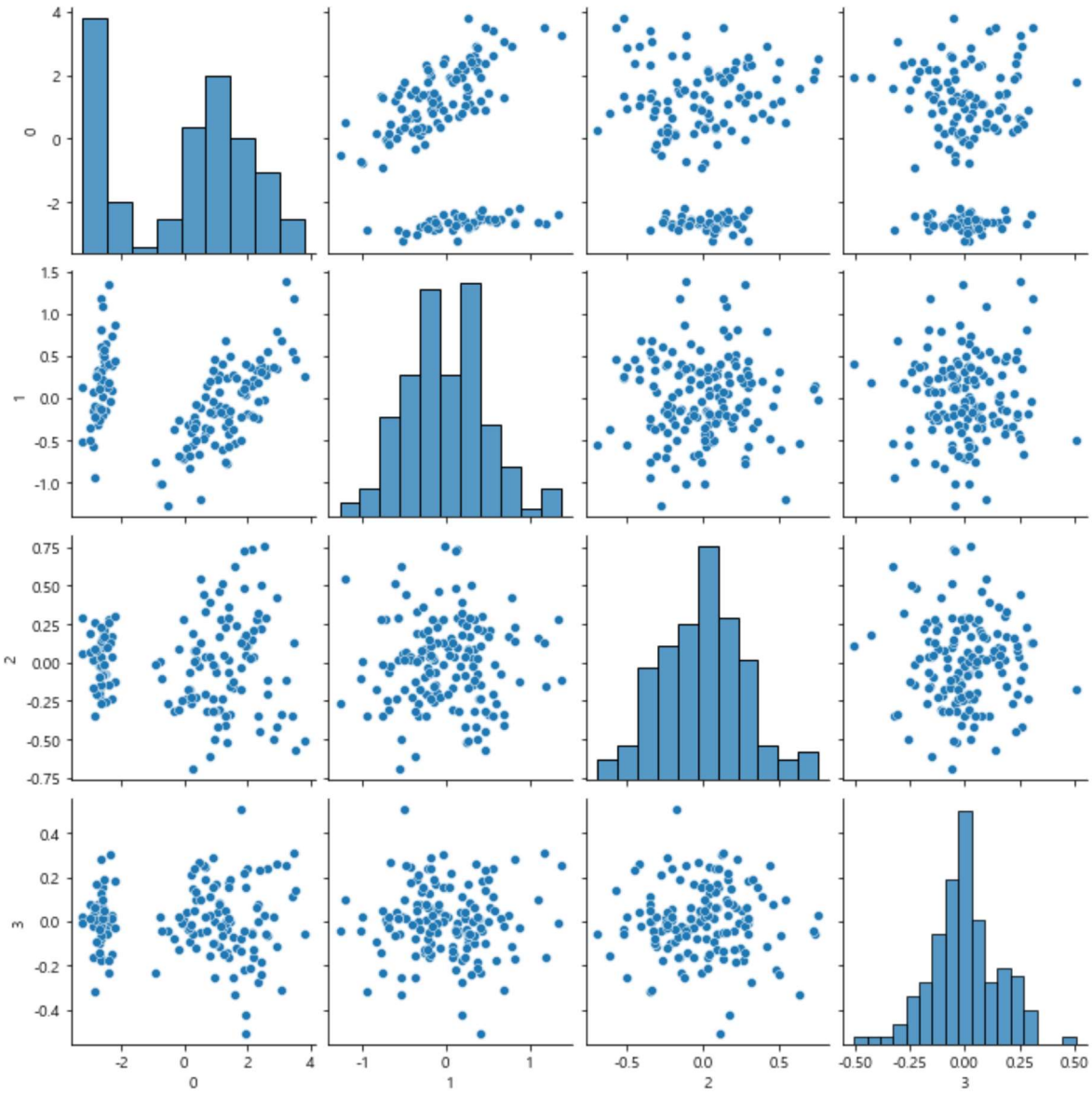
```
import pandas as pd
```

In [39]:

```
df = pd.DataFrame(X_pca_dat[0:,0:])
sns.pairplot(df)
```

Out[39]:

<seaborn.axisgrid.PairGrid at 0x1bbfc0fa8b0>



2개의 feature로 줄여보기

- n_components 를 사용

In [40]:

```
model = PCA(n_components=2)
X_pca2 = model.fit(X_iris).transform(X_iris)
print(X_pca2.shape, type(X_pca_dat) )
print(X_pca2[0:3])
```

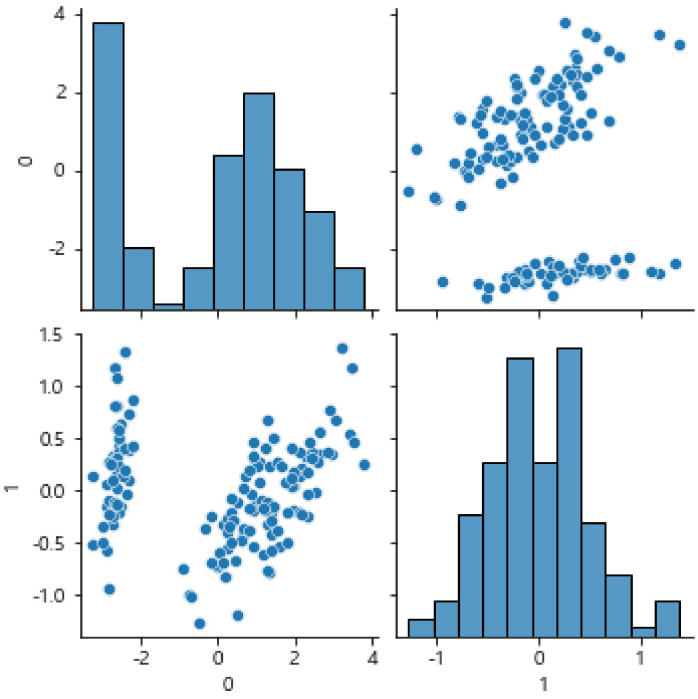
```
(150, 2) <class 'numpy.ndarray'>
[[-2.68412563  0.31939725]
 [-2.71414169 -0.17700123]
 [-2.88899057 -0.14494943]]
```

In [41]:

```
df = pd.DataFrame(X_pca2[0:,0:])
sns.pairplot(df)
df.head(10)
```

Out[41]:

	0	1
0	-2.684126	0.319397
1	-2.714142	-0.177001
2	-2.888991	-0.144949
3	-2.745343	-0.318299
4	-2.728717	0.326755
5	-2.280860	0.741330
6	-2.820538	-0.089461
7	-2.626145	0.163385
8	-2.886383	-0.578312
9	-2.672756	-0.113774



In [42]:

```
df.iloc[:,0].head()
```

Out[42]:

```
0   -2.684126
1   -2.714142
2   -2.888991
3   -2.745343
4   -2.728717
Name: 0, dtype: float64
```

In [43]:

```
iris['PCA1'] = df.iloc[:,0]  # feature 생성
iris['PCA2'] = df.iloc[:,1]  # feature 생성
iris.head()
```

Out[43]:

	sepal_length	sepal_width	petal_length	petal_width	species	PCA1	PCA2
0	5.1	3.5	1.4	0.2	setosa	-2.684126	0.319397
1	4.9	3.0	1.4	0.2	setosa	-2.714142	-0.177001
2	4.7	3.2	1.3	0.2	setosa	-2.888991	-0.144949
3	4.6	3.1	1.5	0.2	setosa	-2.745343	-0.318299
4	5.0	3.6	1.4	0.2	setosa	-2.728717	0.326755

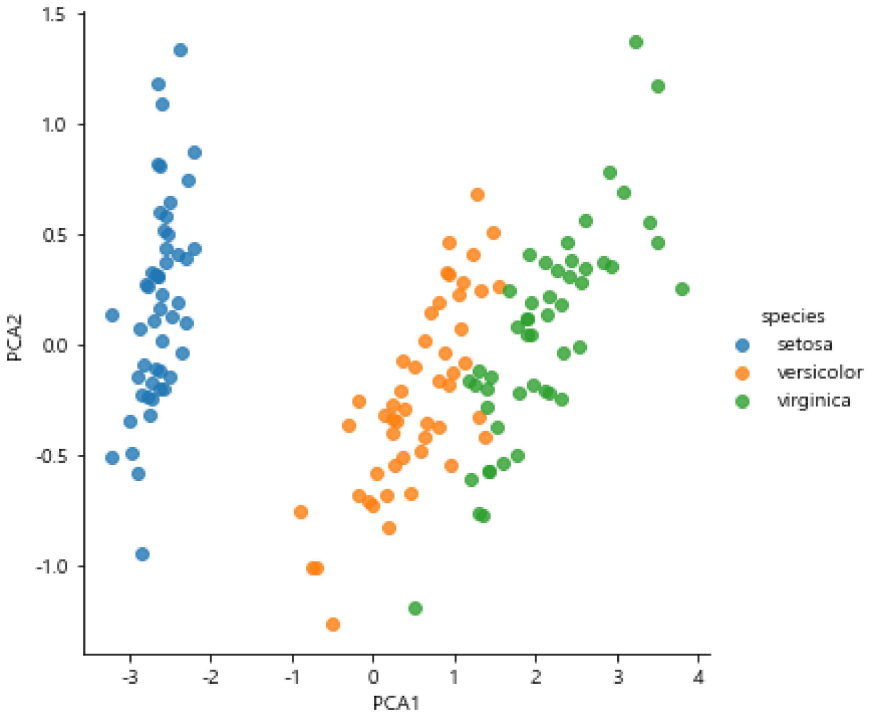
In [44]:

```
sns.lmplot('PCA1', 'PCA2', hue="species", data=iris, fit_reg=False)
```

```
C:₩Users₩front₩Anaconda3₩lib₩site-packages₩seaborn₩_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only vali
d positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[44]:

```
<seaborn.axisgrid.FacetGrid at 0x1bbf94296d0>
```



붓꽃에 대한 기본 정보가 없음에도 PCA를 활용하여 2차원으로 표현한 내용이 잘 구분되어 있음을 볼 수 있다.

REF

- Introduction to Machine Learning with Python 참조
- scikit learn PCA : https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html)