머신러닝
(Machine Learning)

LECTURE XVIII: 비지도 학습과 데이터 전처리 2
(Unsupervised Learning and Preprocessing)

Dai-Gyoung Kim

Department of Applied Mathematics

Hanyang University ERICA

비지도 학습과 데이터 전처리 (Unsupervised Learning and Preprocessing)

Contents

■ 차원 축소, 특성 추출, 매니폴드 학습

비지도 학습을 사용한 데이터 변환의 유용한 점은 데이터의 효과적인 시각화, 압축, 정보 추출을 위한 표현 등임.

- 범용적으로 쓰이는 알고리즘
  1) **주성분 분석**(PCA, Principal Component Analysis)
     - ✓ 압축, 시각화
  2) **비음수 행렬 분해**(NMF, Non-negative Matrix Factorization)
     - ✓ 특성 추출
  3) **t-SNE**(t-distributed Stochastic Negative Embedding)
     - ✓ 2차원 산점도를 이용해 시각화

▶ 차원 축소에 관한 고찰

- 많은 경우 테이터셋의 훈련 샘플은 각각이 수천 또는 수백만 개의 특성을 가지고 있어 훈련을 느리게 할 뿐만 아니라 좋은 솔루션을 찾기 어렵게 만듦. 이런 경우를 **차원의 저주**(curse of dimensionality)라고 함.
- **차원 축소**(Dimensionality Reduction)란 고차원 데이터로부터 본질적인 정보만을 추출하여 저차원 표현으로 변환하는 문제임.
  - ✓ 차원을 축소함으로써, 훈련 속도를 높일 수 있고, 또한 데이터를 효율적으로 시각화할 수 있음.

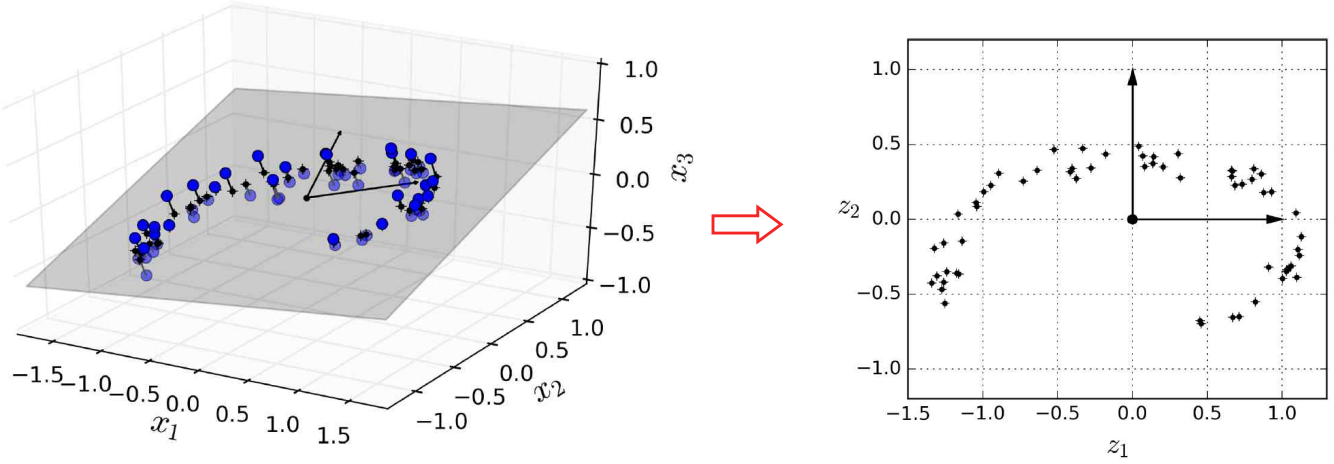
- 차원 축소에 사용되는 두 가지 주요 접근 방법은 **투영**(projection)과 **매니폴드 학습**(Manifold Learning)이 있음.
- 차원 축소 기법에는 PCA, 커널 PCA, LLE 등이 있음.

▶ 차원 축소를 위한 접근 방법

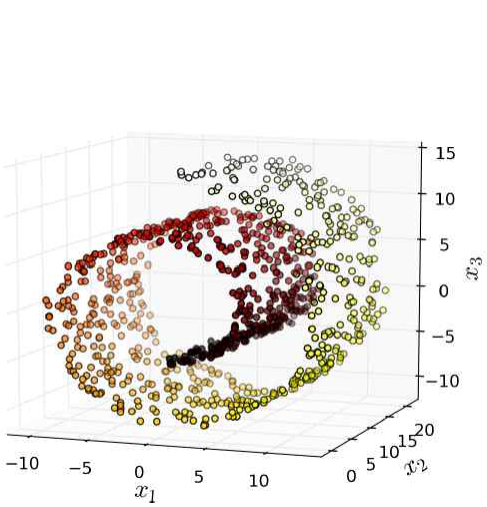
투영(Projection)

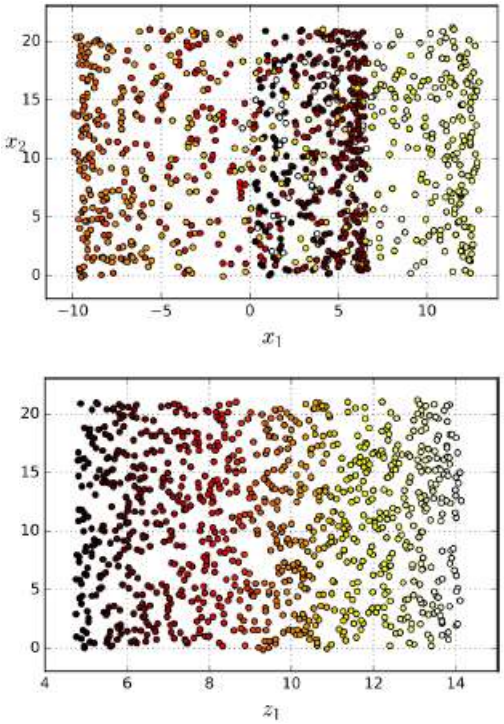
- 대부분의 실제문제에서는 훈련 샘플이 모든 차원에 걸쳐 균일하게 퍼져 있지 않음.
- 많은 특성은 거의 변화가 없는 반면, 다른 특성들은 서로 강하게 연관되어 있음. (예: MNIST 이미지)
- 모든 훈련 샘플은 대부분 고차원 공간 안의 저차원 부분공간(subspace)에 가까이 놓여 있음.

- 다음 그림은 타원 모양을 띤 3D(3차원) 데이터셋으로 거의 2D 평면내의 타원 형태로 볼 수 있음.



- 다음 그림은 3D내의 스위스롤(Swiss roll) 모양의 데이터셋으로, 보통의 투영 방법이 효율적인 차원 축소를 구현하지 못함.



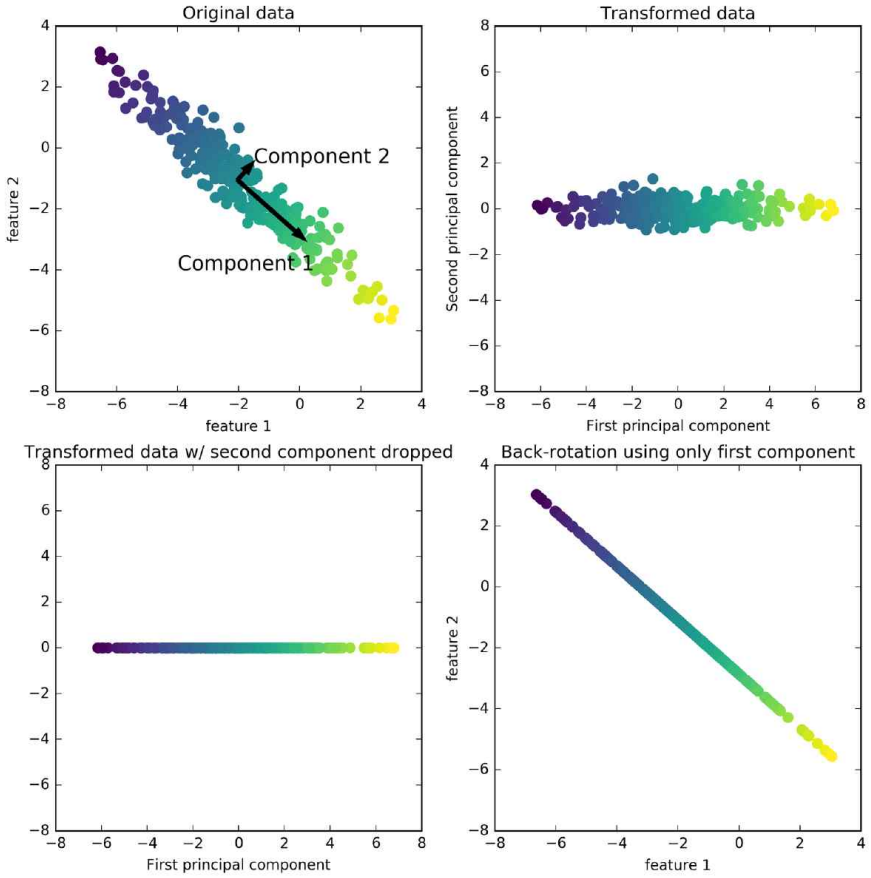


매니폴드 학습(Manifold Learning)

- 스위스롤(Swiss roll)은 2D 매니폴드임. 즉, 국부적으로 2차원 평면으로 보이지만 3차원에서 말려 있음.
- n차원내의 d차원 **매니폴드**는 국부적으로 d차원 초평면으로 보일 수 있는 n차원 공간의 일부임($d < n$).
- 대부분 실제 고차원 데이터셋이 더 낮은 저차원 매니폴드에 가깝게 놓여 있다는 **매니폴드 가정** 하에, 많은 차원 축소 알고리즘이 훈련 샘플이 놓여있는 매니폴드를 모델링하는 식으로 작동함. 이를 **매니폴드 학습**(manifold learning)이라고 함.
- 매니폴드 학습에서 훈련 샘플이 저차원의 매니폴드 공간에 표현되면 더 간단해질 것이라는 가정을 함.
  - ✓ 그러나 이 가정은 항상 유효하지 않음.
- 모델을 훈련시키기 전에 훈련 세트의 차원을 감소시키면 훈련 속도는 빨라지지만 항상 더 나은 솔루션을 제공하지 않음. 전적으로 데이터셋에 달려있음.

❖ 주성분 분석(PCA)

- 주성분 분석(PCA)은 가장 인기 있는 차원 축소 알고리즘임.
- 주성분 분석은 통계적으로 특성들의 상관관계가 없도록 데이터셋을 회전(좌표변환)시켜 주요한 특성을 파악하게 하는 기법임.
- 오른쪽 그림은 가공의 2차원 데이터셋에 PCA를 적용하여 그 효과를 보여주는 예시임.

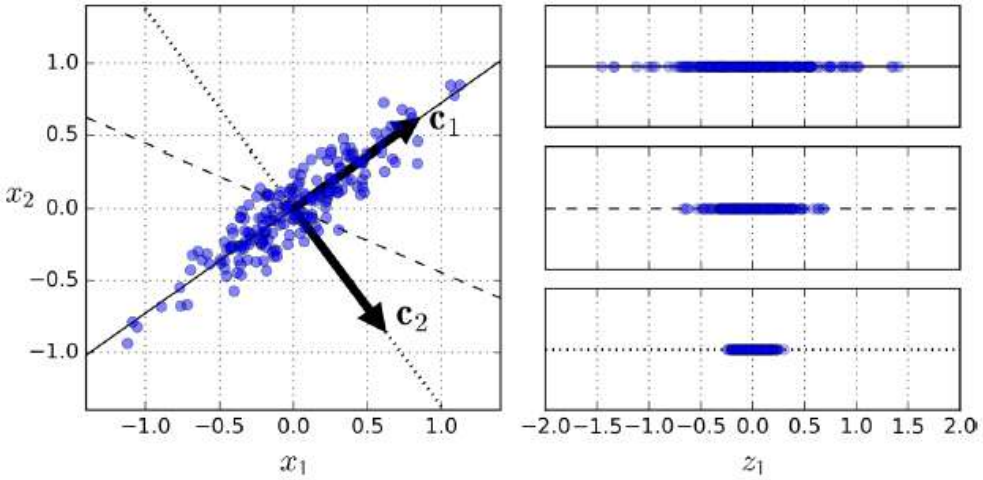


[PCA 알고리즘의 기본 개념]

- PCA 알고리즘은 먼저 데이터에 가장 가까운 초평면을 정의한 다음, 데이터를 이 평면에 투영시킴.

분산 보존

- 저차원의 초평면에 훈련 세트를 투영하기 전에 먼저 올바른 초평면을 선택해야함.
- 왼쪽 그림은 간단한 2D 데이터셋이 세 개의 1D 초평면에 투영된 결과를 보여주고 있음.



- ✓ 위 그림의 오른쪽에서 실선에 투영된 것은 분산을 최대로 보존하며, 반면 점선에 투영된 것은 분산을 매우 적게 유지하고 있음.
- ✓ 분산이 최대로 보존되는 축을 선택하는 것이 정보가 가장 적게 손실됨.
  (이 축이 원본 데이터셋과 투영된 데이터사이의 평균 제곱 거리를 최소화하는 축임.)

PCA 알고리즘

- PCA는 훈련 세트에서 분산이 최대인 축을 찾고, 그다음 첫 번째 축에 직교하고 남은 분산을 최대한 보존하는 두 번째 축을 찾음(이 과정을 차원 수 만큼 진행함).
- i번째 축을 정의하는 단위 벡터를 i번째 주성분(PC)이라고 부름.
- 이런 과정을 거쳐 찾은 축(방향)을 데이터에 있는 주된 분산의 방향이라고 해서 **주성분**(principal component)이라고 함.

주성분 찾기

- 행렬의 특이값 분해(SVD, Singular Value Decomposition)의 기법으로 훈련 세트 X를 세 개의 행렬 U, Σ, V의 곱 $X = U\Sigma V^T$으로 분해할 수 있음.
- SVD: $X \in \mathbb{R}^{m \times n}$에 대하여

$$X = U\Sigma V^T$$

  - $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m}$, u_i: left singular vector of X(eigenvector of XX^T)
  - $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times n}$, v_j: right singular vector of X(eigenvector of X^TX)
  - U, V: orthogonal matrices
  - $\Sigma \in \mathbb{R}^{m \times n}$, $k = \min(m,n)$:

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \\ & & 0 \\ & & \vdots \\ & & 0 \end{pmatrix} \ (m \geq n), \qquad \Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_k & 0 \cdots 0 \end{pmatrix} \ (m \leq n)$$

- σ_i: singular values of X(square root of eigenvalues of XX^T or X^TX)
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0$
- $X\boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j, \ X^T\boldsymbol{u}_j = \sigma_j \boldsymbol{v}_j$
- $X = \sum_{j=1}^{k} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T, \ X^\dagger = \sum_{j=1}^{k} \frac{1}{\sigma_j} \boldsymbol{v}_j \boldsymbol{u}_j^T \ (k = \mathrm{rank}(X))$

- $Xv_j = \sigma_j u_j,\ X^T u_j = \sigma_j v_j$

- 공분산 행렬의 고유 벡터는 분산이 어느 방향으로 가장 큰지를 나타냄.
- 중앙에 맞춰진(데이터셋의 평균을 0으로 맞춤) 훈련 세트의 공분산 행렬은 다음과 같음.

$$Cov = \frac{1}{m-1} X^T X = \frac{1}{m-1} (U\Sigma V^T)^T (U\Sigma V^T)$$

$$= \frac{1}{m-1} (V\Sigma U^T)(U\Sigma V^T) = \frac{1}{m-1} (V\Sigma^2 V^T) = V \frac{\Sigma^2}{m-1} V^T$$

- 따라서 v_j가 공분산 행렬의 고유벡터이고 주성분이 됨. 이때 대응되는 분산은 σ_j^2임.

- 다음 코드는 넘파이의 'svd()' 함수를 사용해 훈련 세트의 모든 주성분을 구한 후 처음 두 개의 PC를 추출함.

```
X_centered = X - X.mean(axis=0)
U, s, Vt = np.linalg.svd(X_centered)
c1 = Vt.T[:, 0]
c2 = Vt.T[:, 1]
```

d차원으로 투영하기

- 주성분을 모두 추출했으면 처음 d개의 주성분으로 정의한 초평면에 투영하여 데이터셋의 차원을 d차원으로 축소할 수 있음.
- 이 초평면은 분산을 가능한 한 최대로 보존하는 투영을 보장함.

- 초평면에 훈련 세트를 투영하기 위해 다음 식과 같이 훈련 세트 행렬 X와 첫 d 개의 주성분을 담은 행렬 $W_d = [v_1, \ldots, v_d]$과 곱하면 됨.

$$X_d = X W_d$$

- 다음 코드는 첫 두 개의 주성분으로 정의된 평면에 훈련 세트를 투영한 것임.

```
W2 = V.T[:, :2]
X2D = X_centered.dot(W2)
```

사이킷런 사용하기

- 사이킷런의 PCA 모델은 SVD 분해 방법을 사용하여 구현함. 다음은 PCA 모델을 사용해 데이터셋의 차원을 2로 줄이는 코드임.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X2D = pca.fit_transform(X)
c1 = pca.components_.T[:,0]
```

✓ 위 사이킷런의 PCA 모델은 자동으로 데이터를 중앙에 맞춰줌.

설명된 분산의 비율

- 'explained_variance_ratio_' 변수에 저장된 주성분의 설명된 분산의 비율은 다음과 같은 정보를 줌.

$$\frac{1}{\sum_{i=1}^{k} \sigma_i^2}\left[\sigma_1^2, \ \ldots \ , \sigma_d^2\right]$$

- 다음은 앞의 그림에 나타난 3D 스위스롤(Swiss roll) 모양의 데이터셋의 처음 두 주성분에 대한 설명된 분산의 비율을 보여줌.

```
print(pca.explained_variance_ratio_)
```

```
array([ 0.84248607, 0.14631839])
```

✓ 분산의 84%가 첫 번째 축에 놓여 있고, 14.6%가 두 번째 축에 놓여 있음. 세 번째 축에는 1.2%만 남아 있음.

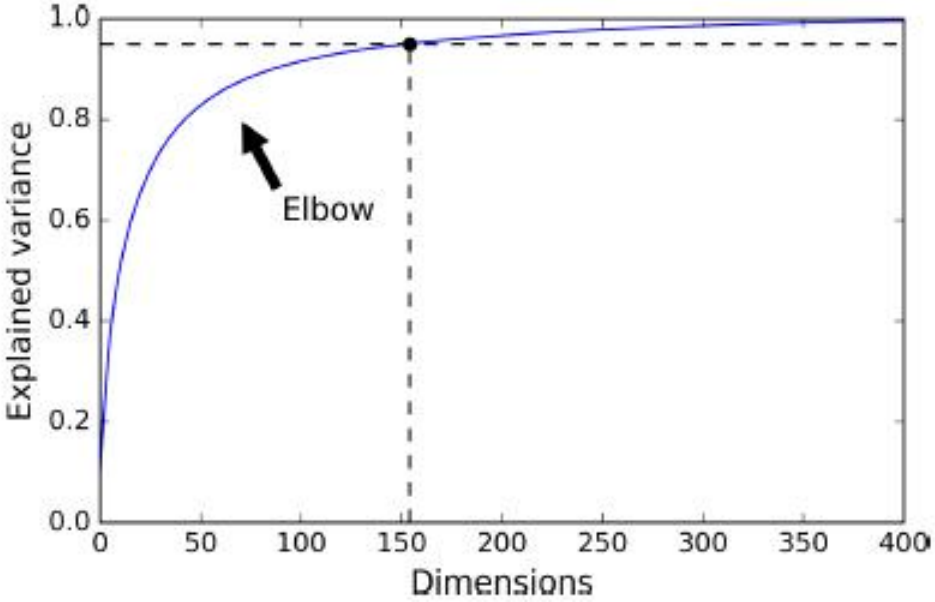
적절한 차원 수 선택하기

- 충분한 분산(예를 들면 95%)이 될 때까지 더해야 할 차원 수를 선택하는 것이 일반적임.
- 데이터 시각화를 위해서는 2 또는 3차원으로 축소함.
- 다음 코드는 차원을 축소하지 않고 PCA를 계산한 뒤 훈련 세트의 분산을 95%로 유지하는 데 필요한 최소한이 차원수를 계산함.

```
pca = PCA()
pca.fit(X)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum >= 0.95) + 1

pca = PCA(n_components=d)
X_reduced = pca.fit_transform(X)
```

- 또 다른 방법은 설명된 분산을 차원 수에 대한 함수로 그래프를 그려봄.
- 그래프에서 설명된 분산의 빠른 성장이 멈추는 변곡점을 찾아 데이터에 내재된 고유차원을 찾음.



▶ PCA를 적용해 유방암 데이터셋 시각화하기

- 산점도는 데이터 시각화에 많이 쓰이지만 세 개 이상의 특성을 가진 데이터에 대해서는 두 개의 특성을 짝지어 산점도 행렬을 사용해 시각화하기 때문에 고차원

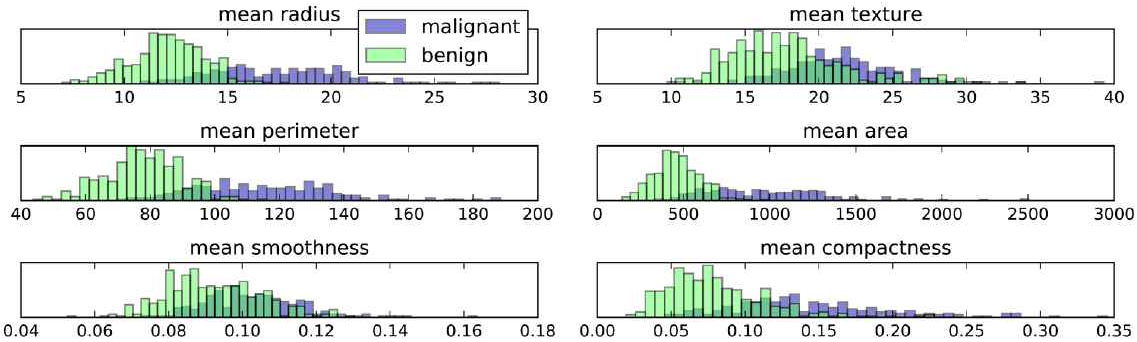
데이터를 시각화하는 데는 비효율적임.

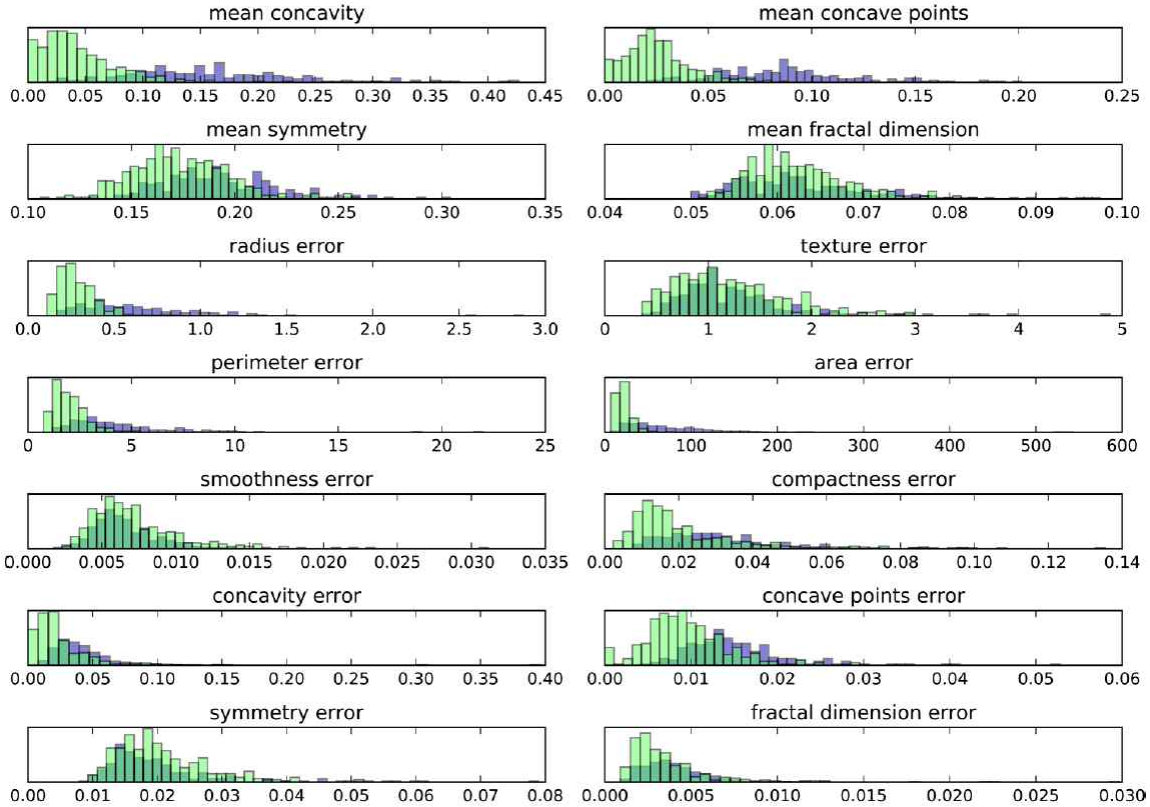
- 예를 들어 유방암 데이터셋은 30개의 특성을 가지고 있어 ${}_{30}C_2 = 435$개의 산점도가 그려짐.
- 이보다 효율적인 방법은 양성과 음성 두 클래스에 대해 각 특성의 히스토스램을 분석하는 것임.

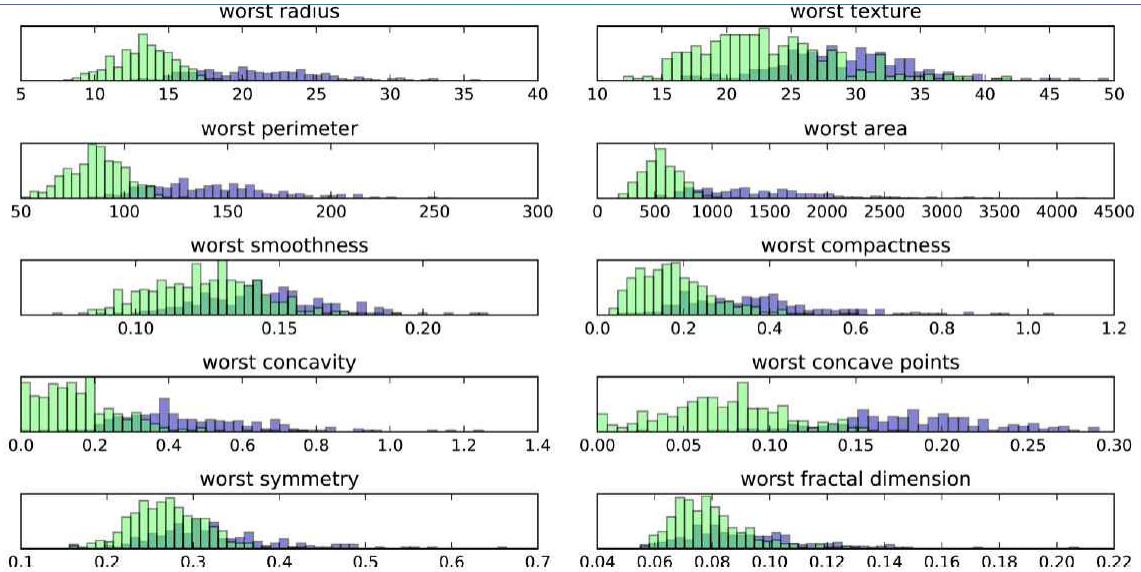
```
fig, axes = plt.subplots(15, 2, figsize=(10, 20))
malignant = cancer.data[cancer.target == 0]
benign = cancer.data[cancer.target == 1]

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("Feature magnitude")
ax[0].set_ylabel("Frequency")
ax[0].legend(["malignant", "benign"], loc="best")
fig.tight_layout()
```




mean concavity
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45
mean concave points
0.00 0.05 0.10 0.15 0.20 0.25
mean symmetry
0.10 0.15 0.20 0.25 0.30 0.35
mean fractal dimension
0.04 0.05 0.06 0.07 0.08 0.09 0.10
radius error
0.0 0.5 1.0 1.5 2.0 2.5 3.0
texture error
0 1 2 3 4 5
perimeter error
0 5 10 15 20 25
area error
0 100 200 300 400 500 600
smoothness error
0.000 0.005 0.010 0.015 0.020 0.025 0.030 0.035
compactness error
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14
concavity error
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40
concave points error
0.00 0.01 0.02 0.03 0.04 0.05 0.06
symmetry error
0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08
fractal dimension error
0.000 0.005 0.010 0.015 0.020 0.025 0.030



- ✓ 위 그림은 각 특성에 대한 히스토그램으로 특정 간격(bin)에 얼마나 많은 데이터 포인트가 있는지 횟수를 센 것임.
- ✓ 각 특성마다 양성, 음성 클래스 히스토그램의 겹치는 정도를 파악하여 두 클래스의 구분을 가늠해 볼 수 있음.
- ✓ 그러나 특성 간의 상호 작용이나 이 상호 작용이 클래스와 어떤 관련이 있는지는 파악할 수 없음.

- PCA를 사용하면 주요 상호작용을 파악할 수 있는 시각화를 구현할 수 있음.
  - ✓ 예를 들어 처음 두 개의 주성분을 찾아 2차원 공간에 하나의 산점도로 데이터를 시각화할 수 있음.

[PCA를 사용한 시각화]

- 먼저 PCA를 적용하기 전에 'StandardScaler'를 사용해 각 특성의 분산이 1이 되도록 데이터의 스케일을 조정함.

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()

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

- 다음은 PCA 객체를 생성하고, 'fit' 메서드를 호출하여 주성분을 찾고, 'transform' 메서드를 호출하여 데이터를 변환하여 차원을 축소함.

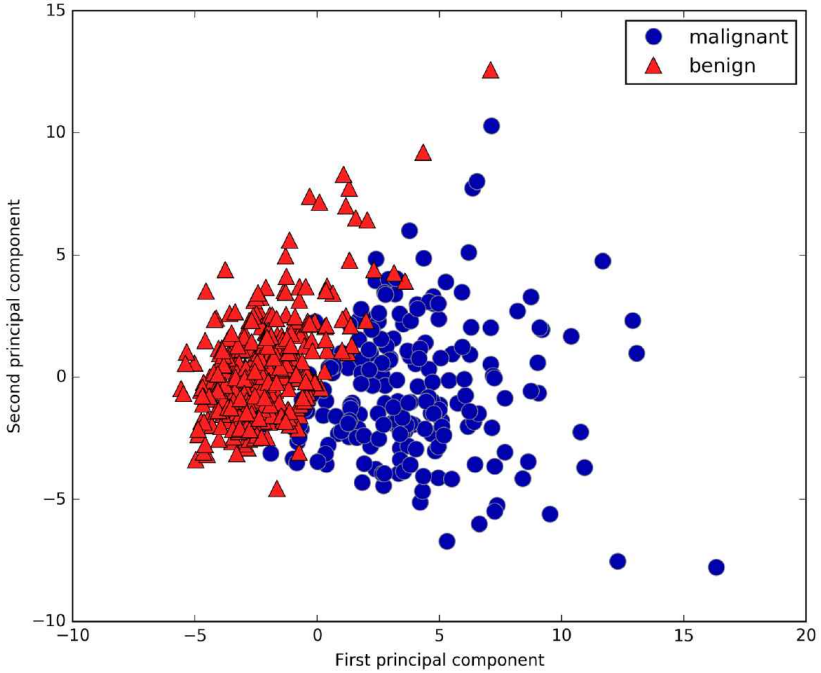
```
from sklearn.decomposition import PCA
# keep the first two principal components of the data
pca = PCA(n_components=2)
# fit PCA model to breast cancer data
pca.fit(X_scaled)

# transform data onto the first two principal components
X_pca = pca.transform(X_scaled)
print("Original shape: {}".format(str(X_scaled.shape)))
print("Reduced shape: {}".format(str(X_pca.shape)))
```

```
Original shape: (569, 30)
Reduced shape: (569, 2)
```

- 처음 두 개의 주성분을 사용해 유방암 데이터셋의 2차원 산점도로 시각화함.

```
# plot first vs. second principal
# component, colored by class
plt.figure(figsize=(8, 8))
mglearn.discrete_scatter(X_pca[:, 0],
                         X_pca[:, 1], cancer.target)
plt.legend(cancer.target_names,
                      loc="best")
plt.gca().set_aspect("equal")
plt.xlabel("First principal component")
plt.ylabel("Second principal component")
```



- PCA는 비지도 학습이므로 회전축을 찾을 때 어떤 클래스 정보도 사용하지 않고, 단순히 데이터에 있는 상관관계만을 고려함.
- 위 산점도는 첫 번째 주성분과 두 번째 주성분을 사용하여 만들어졌고 클래스 정보를 이용하여 포인트 모양을 구분하였음.
  - ✓ 두 클래스가 2차원 공간에서 잘 구분되는 것을 볼 수 있음 (선형 분류기를 적용하여 두 클래스를 구분 가능).

- PCA의 단점은 그래프의 두 축(주성분)을 해석하기 어렵다는 점. 주성분은 원본 데이터의 어떤 방향에 대응하는 여러 특성이 조합된 형태임.

```
print("PCA component shape: {}".format(pca.components_.shape))
```

```
PCA component shape: (2, 30)
```

  - ✓ PCA 객체가 학습될 때 'components_' 속성에 주성분이 저장됨.
  - ✓ 'components_'의 각 행은 주성분 하나씩을 나타내며 중요도에 따라 정렬되어 있으며, 각 열은 원본 데이터의 특성에 대응하는 값임.

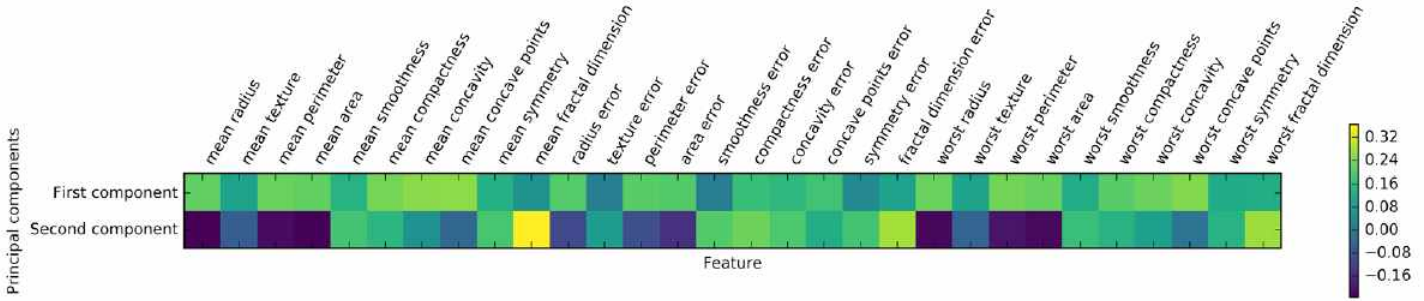
- 다음은 'components_' 값을 출력한 것임.

```
print("PCA components:\n{}".format(pca.components_))
```

```
PCA components:
[[ 0.219 0.104 0.228 0.221 0.143 0.239 0.258 0.261 0.138 0.064
   0.206 0.017 0.211 0.203 0.015 0.17  0.154 0.183 0.042 0.103
   0.228 0.104 0.237 0.225 0.128 0.21  0.229 0.251 0.123 0.132 ]
 [-0.234  -0.06  -0.215 -0.231 0.186 0.152 0.06  -0.035 0.19  0.367
  -0.106   0.09  -0.089 -0.152 0.204 0.233 0.197  0.13  0.184 0.28
  -0.22   -0.045 -0.2   -0.219 0.172 0.144 0.098 -0.008 0.142 0.275]]
```

- 다음은 위의 결과를 히트맵으로 시각화한 것임.

```
plt.matshow(pca.components_, cmap='viridis')
plt.yticks([0, 1], ["First component", "Second component"])
plt.colorbar()
plt.xticks(range(len(cancer.feature_names)), cancer.feature_names, rotation=60, ha='left')
plt.xlabel("Feature")
plt.ylabel("Principal components")
```



- ✓ 첫 번째 주성분의 모든 특성의 부호가 같음. 이는 모든 특성 사이에 공통의 상호관계가 있다는 뜻임.
- ✓ 두 번째 주성분은 부호가 섞여 있으며 이는 특성 사이의 또 다른 세부 상호관계가 있다는 뜻임.
- ✓ 위 경우 두 주성분 모두 30개의 특성을 포함하고 있으므로 각 주성분 축이 가지는 의미를 설명하기 힘듦.