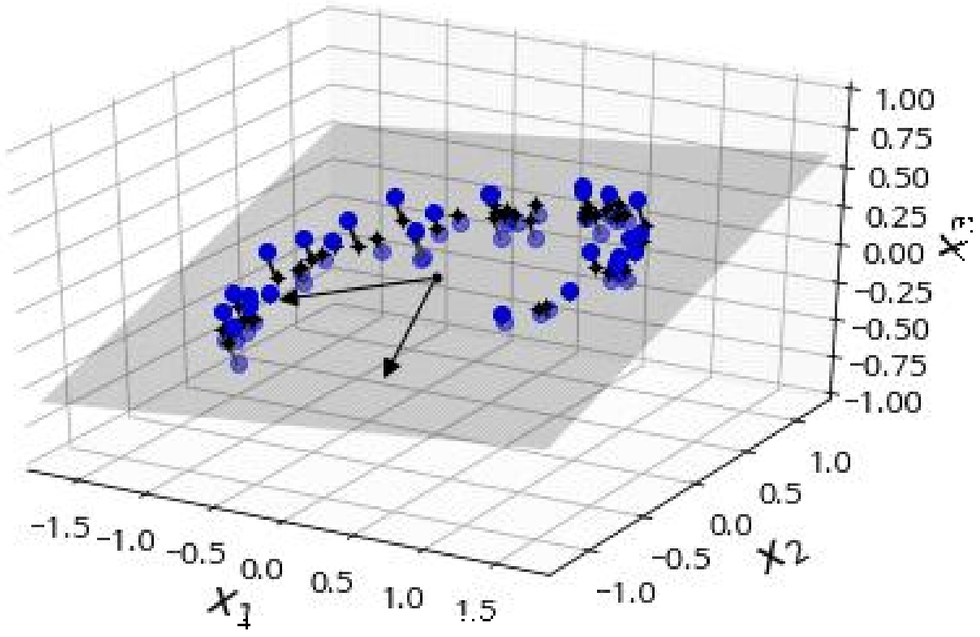
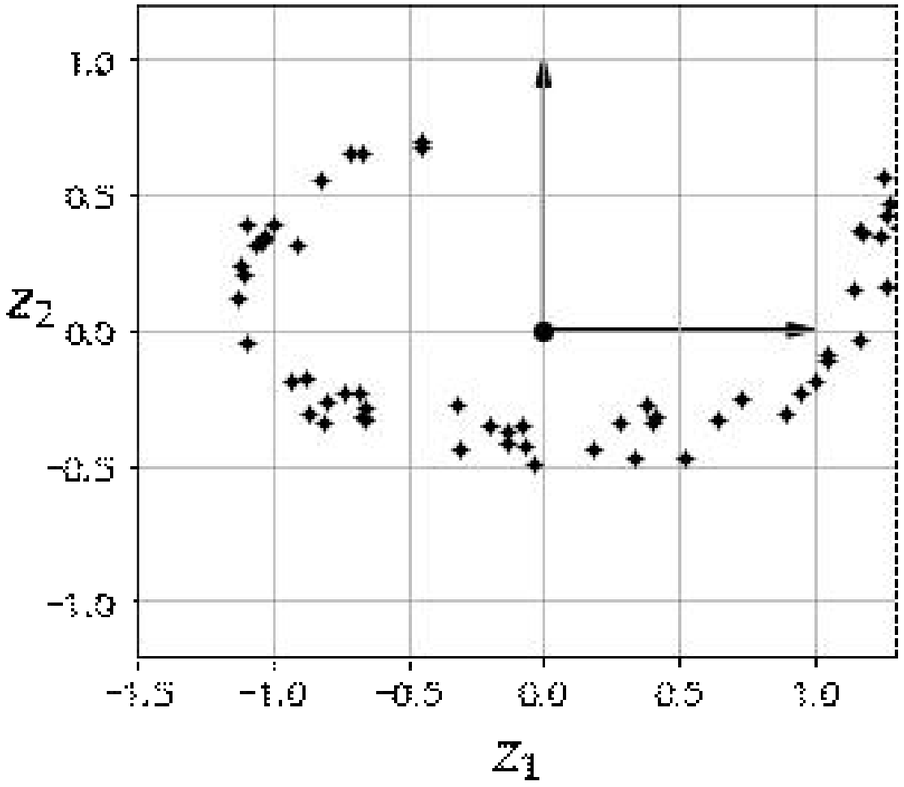
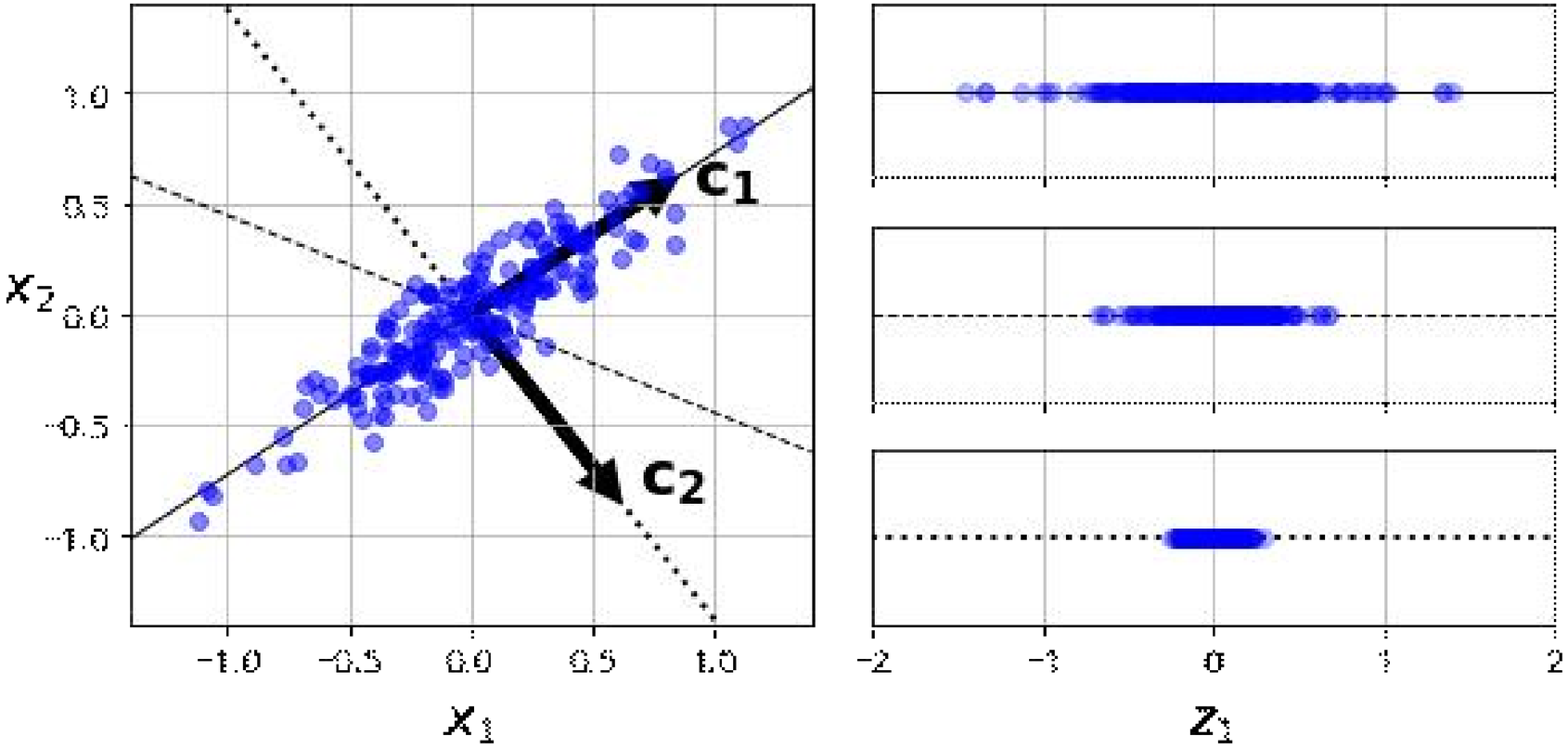
차원 축소





이미지 출처: 오헬리앙 제롱, 핸즈온 머신러닝 1판(한빛미디어, 2018), 272-273

주성분 분석 (principal component analysis, PCA)



이미지 출처: 오헬리앙 제롱, 핸즈온 머신러닝 1판(한빛미디어, 2018), 277

Singular Value Decomposition (SVD)

A가 m×n 행렬이라면

$A = U\Sigma V^T$ 로 분해 가능

(m×n) (m×m) (m×n) (n×n)

U: AA^T 의 고유벡터행렬 ($AA^T = U(\Sigma\Sigma^T)U^T$), 직교행렬

(m×m)

V: A^TA 의 고유벡터행렬 ($A^TA = V^T(\Sigma^T\Sigma)V$), 직교행렬

(n×n)

Σ: 고유값들의 제곱근을 대각원소로 하는 벡터, 나머지 성분은 모두 0

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

평균이 원점에 맞춰진 경우

$$Cov = \frac{1}{n-1} A^T A = \frac{1}{n-1} V^T (\Sigma^T \Sigma) V$$

공분산 행렬의 고유벡터행렬이 V고 주성분이 된다.

$$\begin{bmatrix} | & | & | & \\ c_1 & c_2 & c_3 & \cdots \\ | & | & | & \end{bmatrix}$$

numpy 의 linalg.svd()

```
[ ]  # SVD 분해
     fruits_centered = fruits_2d - fruits_2d.mean(axis = 0)
     U, s, Vt = np.linalg.svd(fruits_centered)
     print(U.shape, s.shape, Vt.shape)
```

```
(300, 300) (300,) (10000, 10000)
```

```
[ ]  print(fruits_centered.shape)
```

```
(300, 10000)
```

투영(Projection)

$$A_{d-proj} = A \cdot W_d \qquad W_d = \begin{bmatrix} | & | & | & \\ c_1 & c_2 & c_3 & \cdots \\ | & | & | & \end{bmatrix}$$

```
[ ]  # 주성분 2개만 사용하여 2차원 초평면 만들고 투영하기
     W2 = Vt.T[:, :2]
     fruits_2d_svd = fruits_centered.dot(W2)
```

```
[ ]  print(fruits_pca.shape)
     print(fruits_2d_svd.shape)
```

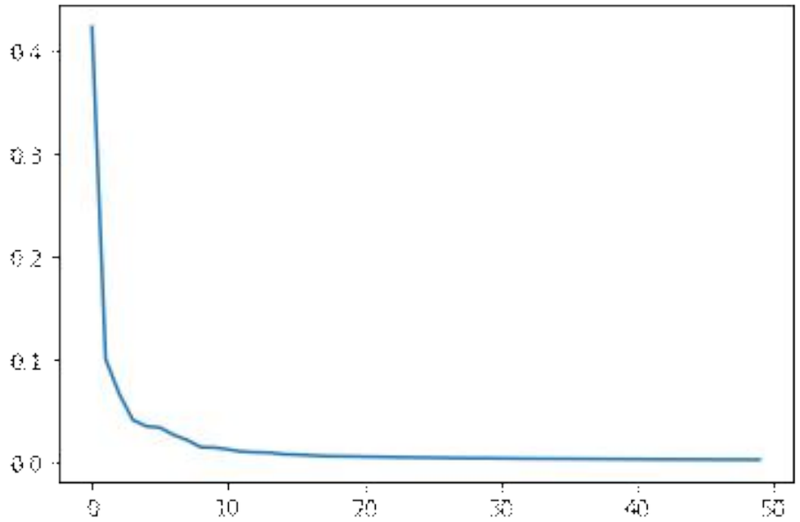
```
(300, 2)
(300, 2)
```

설명된 분산 (explained variance)

주성분이 원본 데이터의 분산을 얼마나 잘 나타내는지 기록한 값

PCA 클래스의 explained_variance_ratio: 각 주성분의 설명된 분산 비율

```
[ ]  # 설명된 분산 비율 그래프 그리기
     plt.plot(pca.explained_variance_ratio_)
     plt.show()
```




```
[ ]  # 설명된 분산의 비율이 50%를 만족하는 주성분 개수 찾기
     pca = PCA(n_components = 0.5)
     pca.fit(fruits_2d)
```

```
PCA(copy=True, iterated_power='auto', n_components=0.5, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

```
[ ]  print(pca.n_components_)
```

```
2
```

축소할 차원 수 정하기 < 일정 비율(ex. 0.95)이 될 때까지 더해야 할 차원 수 선택하기

복원하기

$$A_{d-proj} = A \bullet W_d$$

(m×d) (m×n) (n×d)

$$A_{recovered} = A_{d-proj} \bullet W_d^T \quad (= A W_d W_d^T, \text{ 데이터 손실 발생})$$

```
[ ]  # 복원하기
     fruits_inverse = pca.inverse_transform(fruits_pca)
     print(fruits_inverse.shape)
```

```
(300, 10000)
```

```
[ ]  # 차원 축소된 데이터 복원하기(데이터 손실 발생)
     fruits_recovered = fruits_2d_svd.dot(W2.T)
     print(fruits_recovered.shape)
     print(fruits_centered.shape)
```

```
(300, 10000)
(300, 10000)
```

참고 자료

- 박해선, *혼자 공부하는 머신러닝 + 딥러닝* (한빛미디어, 2021), 318-337
- 오헬리앙 제롱, *핸즈온 머신러닝 1판* (한빛미디어, 2018), 272-283
- Howard Anton, Chris Rorres, *Elementary Linear Algebra 10th edition* (Wiley, 2011), 509
- [선형대수학 #4] 특이값 분해(Singular Value Decomposition, SVD)의 활용, *다크 프로그래머*, 2013년 10월 23일 수정, 2021년 3월 25일 접속, https://darkpgmr.tistory.com/106