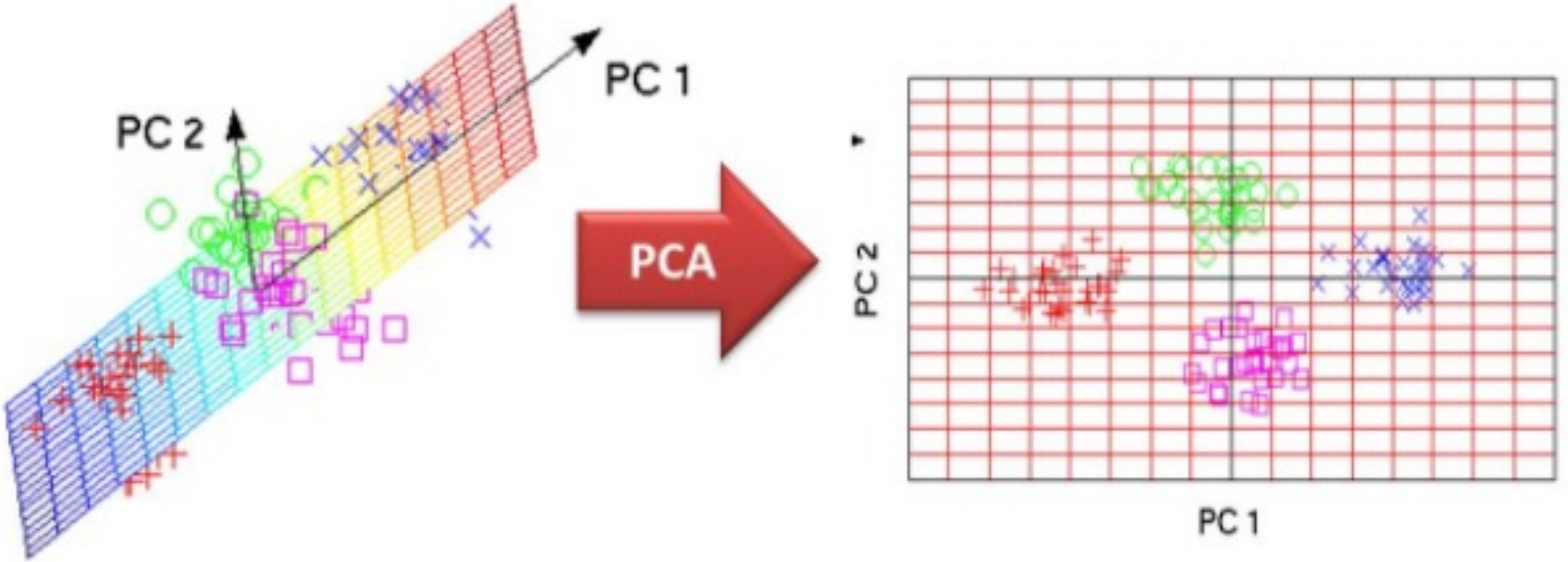
주성분분석(PCA)

PCA(Principal Components Analysis) ?

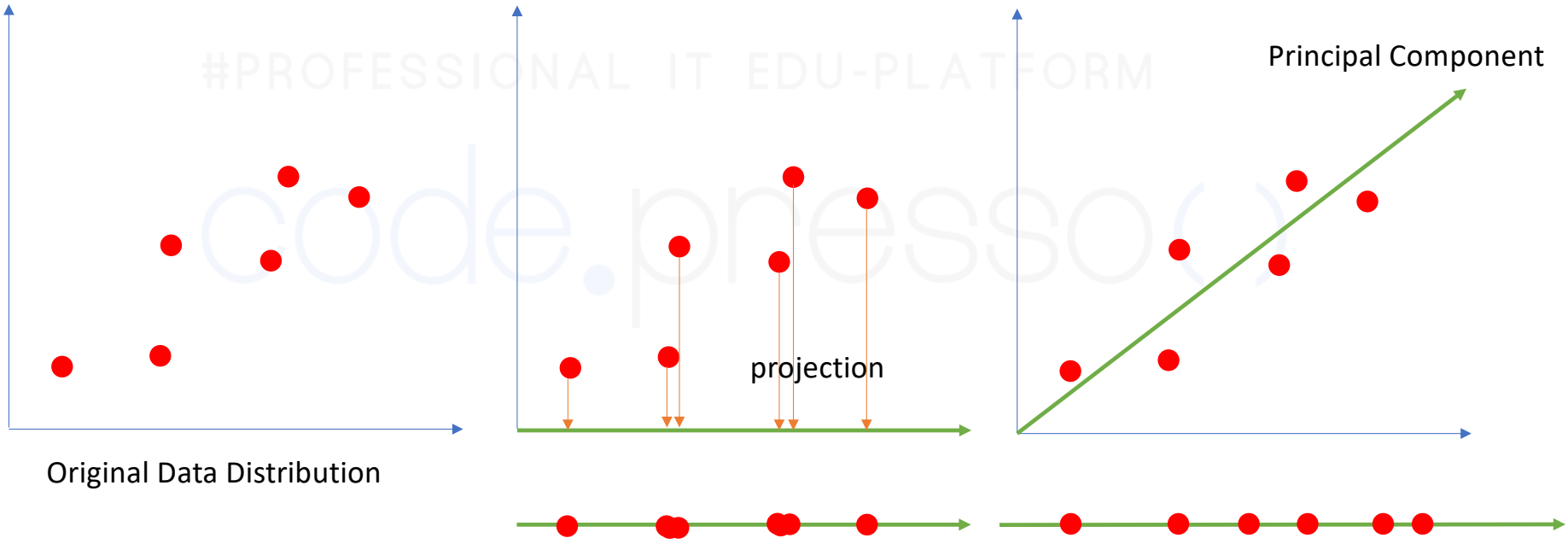
- 고차원 데이터를 효과적으로 분석하기 위한 대표적인 차원축소 기법



https://ashutoshtripathi.com/2019/07/11/a-complete-guide-to-principal-component-analysis-pca-in-machine-learning/

PCA 핵심원리

- 데이터 차원을 축소할 경우, 어떤 벡터에 데이터들을 정사영시켜야 데이터 구조(분산)이 제일 잘 유지되는가?



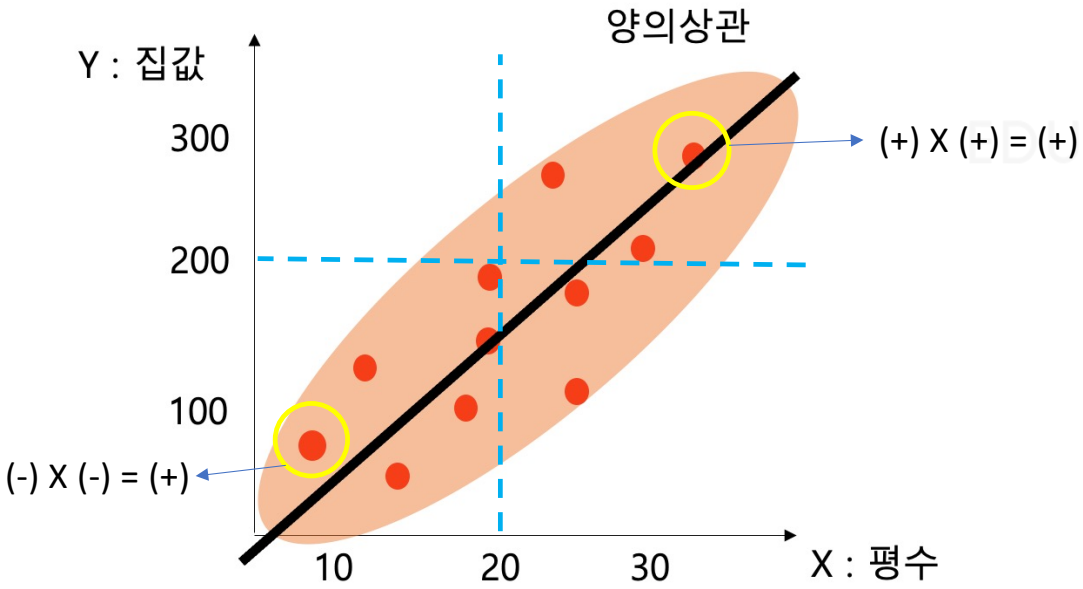
PCA 과정

- 차원축소를 하기 위한 데이터 변수들에 Scaling(Z-score)
- Covariance matrix를 구성(nxn 정방행렬, n = # of feature)

$$\begin{pmatrix} cov(X1,X1),\dots,cov(X1,Xn) \\ cov(X2,X1),\dots,cov(X2,Xn) \\ cov(X3,X1),\dots,cov(X3,Xn) \\ \dots \\ cov(Xn,X1),\dots,cov(Xn,Xn) \end{pmatrix}$$

- Cov. Matrix의 고유벡터(Eigenvector)와 고유값(eigenvalue)를 산출
- Result : Projection on Eigenvector

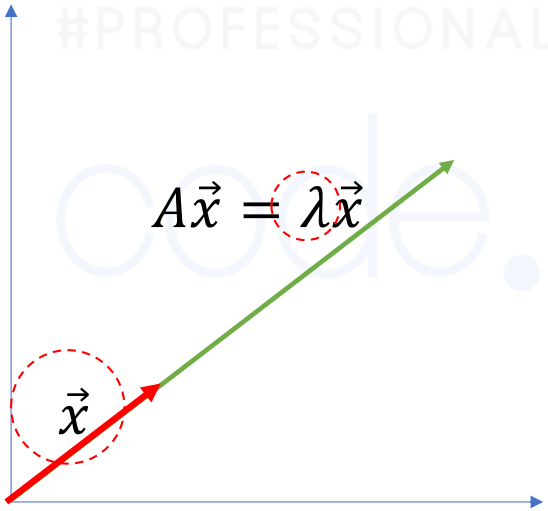
공분산(Covariance)의 의미



$$S_{XY} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

공분산 vs 고유값(Eigenvalue), 고유벡터(Eigenvector)

- 데이터 차원을 축소할 경우, 어떤 벡터에 데이터들을 정사영 시켜야 데이터 구조(분산)이 제일 잘 유지되는가?



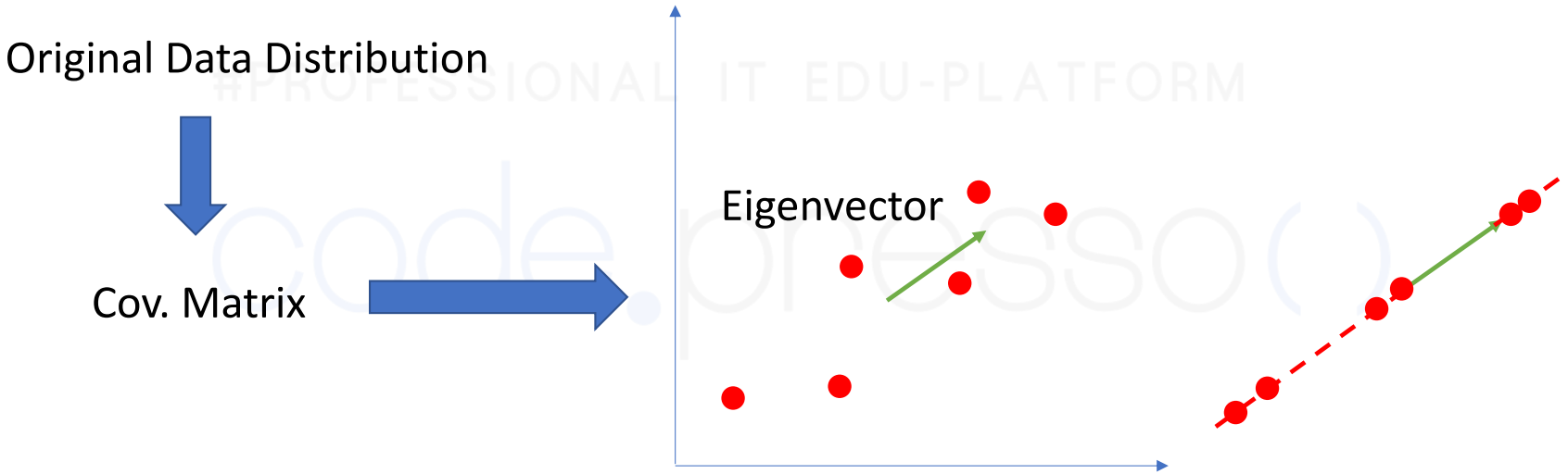
A 행렬이 주어질 때,

$$A\vec{x} = \lambda\vec{x} \qquad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \lambda I\vec{x}$$

$$(A - \lambda I)\vec{x} = 0$$
$$\vec{x} \neq 0, \det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \text{ad-bc}$$

Covariance Matrix와 Eigenvector의 관계



주성분분석(PCA) 실습

실습에 사용할 데이터세트

sklearn.datasets 의 load_iris() 사용

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

load_boston() Load and return the boston house-prices dataset (regression).

load_iris() Load and return the iris dataset (classification).

load_diabetes() Load and return the diabetes dataset (regression).

load_digits() Load and return the digits dataset (classification).

load_linnerud() Load and return the physical excercise linnerud dataset.

load_wine() Load and return the wine dataset (classification).

load_breast_cancer() Load and return the breast cancer wisconsin dataset (classification)

Dict 포맷을 제공하는 데이터세트

sklearn.datasets 의 load_iris() 사용

- .DESCR : 데이터셋의 description
- .feature_names : feature 인덱스명
- .data : feature 데이터
- .target_names : class(target) 인덱스명
- .target : class(target) 데이터

- sklearn.datasets 모듈의 load_iris() 로 데이터 로딩

Code

```
from sklearn.datasets import load_iris
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#iris 데이터셋 로드와 Dict 포맷의 키 확인하기
data = load_iris()
print("iris dataset format and keys\n",data.keys())

#feature name과 관측값 가져오기
iris_data = data['data']
iris_cols = data['feature_names']
print("iris dataset columns\n",iris_cols)
```

- **sklearn.datasets 모듈의 load_iris() 로 데이터 로딩**

Result	iris dataset format and keys dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename']) iris dataset columns ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

- **Column name 재설정 및 데이터프레임 생성, 품종별 카운팅 체크**

Code

```
#column name을 재설정
iris_cols=['sep_len', 'sep_wt', 'pet_len', 'pet_wt']

#데이터프레임 생성
iris_df = pd.DataFrame(data= iris_data, columns= iris_cols)
iris_df['target'] = data['target']
print(iris_df.head(5))

#품종별 데이터 카운팅 체크
target_cnt_df = iris_df.groupby(by='target').count()
print(target_cnt_df)
```

- **Column name 재설정 및 데이터프레임 생성, 품종별 카운팅 체크**

Result

sep_len	sep_wt	pet_len	pet_wt	target
5.1	3.5	1.4	0.2	0.0
4.9	3.0	1.4	0.2	0.0
4.7	3.2	1.3	0.2	0.0
4.6	3.1	1.5	0.2	0.0
5.0	3.6	1.4	0.2	0.0

target	sep_len	sep_wt	pet_len	pet_wt
0	50	50	50	50
1	50	50	50	50
2	50	50	50	50

0 : setosa,
1 : versicolor,
2 : virginica

STEP.3 PCA 수행하기

- 표준화(스케일링) 수행

Code

```
27 #PCA 수행
28 #표준화(스케일링)
29 from sklearn.preprocessing import StandardScaler
30
31 X_train = iris_df.iloc[:, :4]
32 iris_z_score = StandardScaler().fit_transform(X_train)
33
34 iris_z_df = pd.DataFrame(data= iris_z_score, columns= iris_cols)
35 print(iris_z_df.head(5))
```

STEP.3 PCA 수행하기

- 표준화(스케일링) 수행

Result

```
     sep_len       sep_wt     pet_len       pet_wt
-0.9006811..1.01900435..-1.3402265..-1.3154442..
-1.1430169..-0.1319794..-1.3402265..-1.3154442..
-1.3853526..0.32841405..-1.3970639..-1.3154442..
-1.5065205..0.09821728..-1.2833890..-1.3154442..
-1.0218490..1.24920111..-1.3402265..-1.3154442..
```

STEP.3 PCA 수행하기

- PCA 오브젝트 생성과 환경구성 후, 주성분(고유벡터) 찾기

Code

```
#주성분 분석(Feature 4 --> 2 axes)
from sklearn.decomposition import PCA
pca = PCA(n_components=2)

pca.fit(iris_z_df)

#주성분 찾기 : 고유벡터
print('PCA Shape:\n', pca.components_.shape)
print('PCA eigenvectors:\n',pca.components_)

#고유벡터에 데이터를 투영시키는 과정이 transform이다.
X_pca = pca.transform(iris_z_df)
print('PCA Projection result(shape)\n',X_pca.shape)
```

STEP.3 PCA 수행하기

- **PCA 오브젝트 생성과 환경구성 후, 주성분(고유벡터) 찾기**

Result

```
PCA Shape:
 (2, 4)
PCA eigenvectors:
 [[ 0.52106591 -0.26934744  0.5804131   0.56485654]
 [ 0.37741762  0.92329566  0.02449161  0.06694199]]
PCA Projection result(shape)
 (150, 2)
```

STEP.3 PCA 수행하기

- **PCA 수행 후, 주성분의 분산설명력**

Code

```
51 #각 주성분이 분산을 얼마나 잘 설명하는지를 나타냄
52 import numpy as np
53 print('variance :\n',pca.explained_variance_ratio_)
54 print('total variance :\n', np.sum(pca.explained_variance_ratio_))
55 print('\n')
56
57 #projection 된 결과를 데이터프레임으로 구성
58 pca_cols = ['pca_com_1', 'pca_com_2']
59 pca_df = pd.DataFrame(data= X_pca, columns= pca_cols)
60 pca_df['target'] = data['target']
```

STEP.3 LDA 수행하기

- **PCA 수행 후, 주성분의 분산설명력**

Result

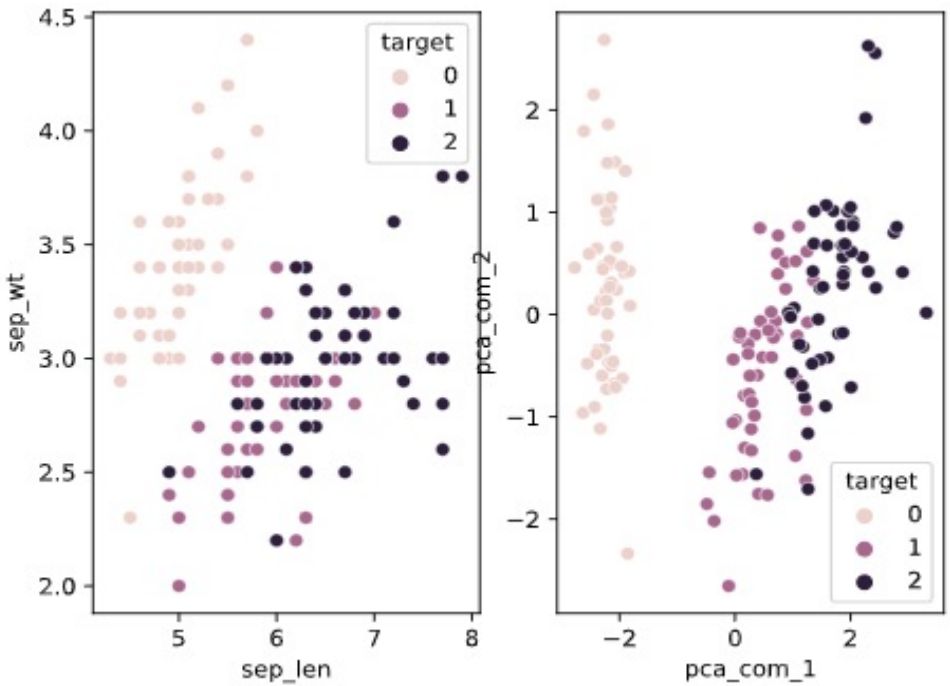
```
variance :
 [0.72962445 0.22850762]
total variance :
 0.9581320720000165


   pca_com_1   pca_com_2      target
-2.2647028..0.48002659..         0.0
-2.0809611..-0.6741335..         0.0
-2.3642290..-0.3419080..         0.0
-2.2993842..-0.5973945..         0.0
-2.3898421..0.64683538..         0.0
```

- 시각화로 PCA 수행 전, 후 비교하기

Code

```
63 #주성분선결과 시각화
64 fig, ax = plt.subplots(ncols=2)
65
66 sns.scatterplot(iris_df['sep_len'], iris_df['sep_wt'],
67                  hue=iris_df['target'], ax=ax[0])
68 sns.scatterplot(pca_df['pca_com_1'], pca_df['pca_com_2'],
69                  hue=pca_df['target'], ax=ax[1])
70 plt.show()
```

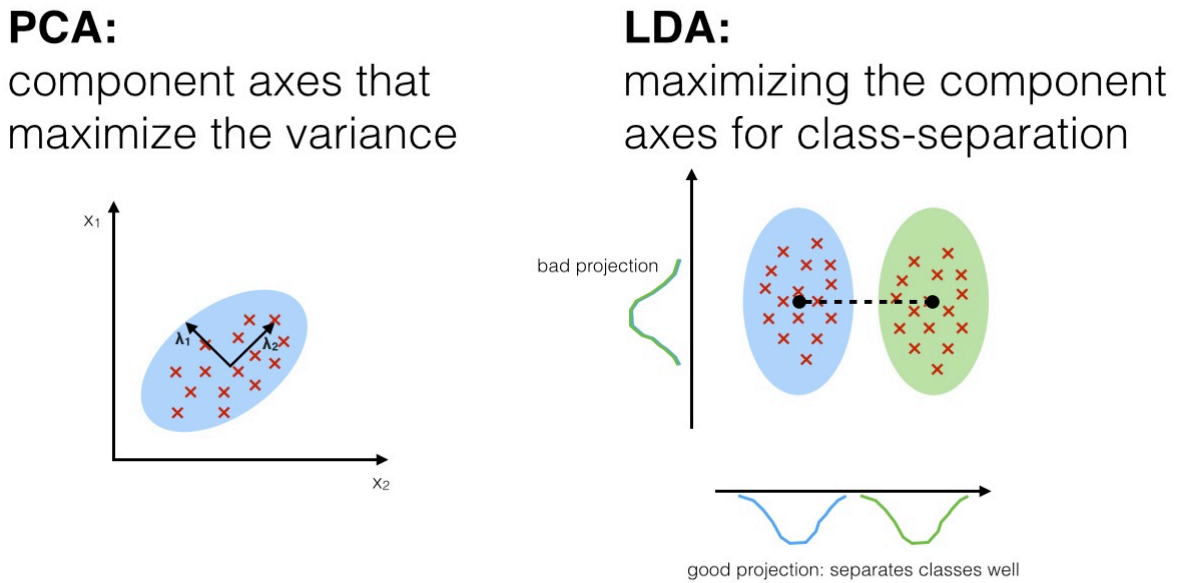
- 시각화로 PCA 수행 전, 후 비교하기

Result	

선형판별분석(LDA)

선형판별분석(Linear Discriminant Analysis, LDA) ?

- 두 개 이상의 모집단에서 표집된 표본들의 데이터분포를 이용하여 이 표본들이 어느 모집단에서 추출된 것지 분류 예측을 할 수 있도록 기준을 찾는 분석 법을 의미함.



https://sebastianraschka.com/Articles/2014_python_lda.html

선형판별식

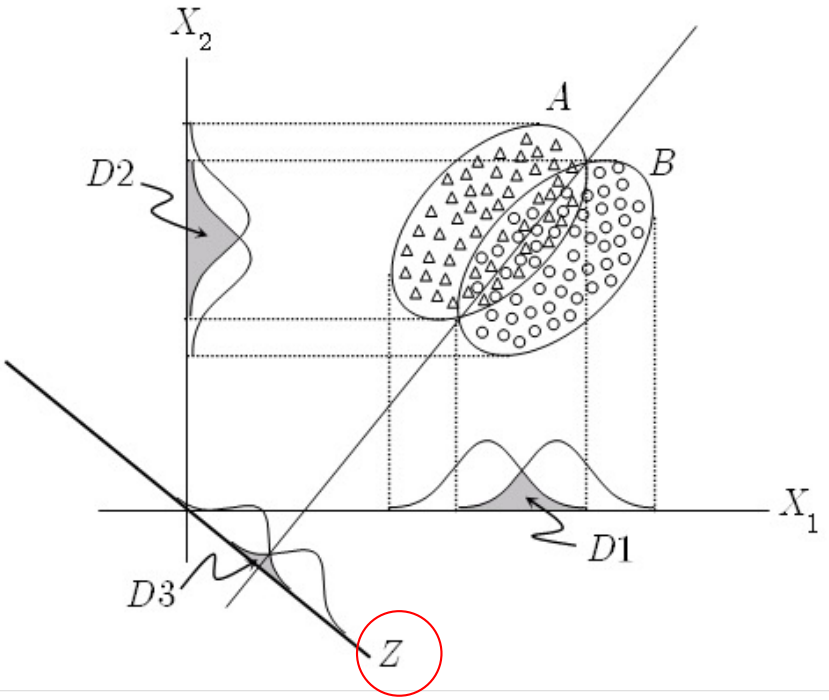
- **집단을 구분할 수 있는 독립(설명)변수를 통하여 집단 구분 함수식(판별식)을 도출하고, 소속된 집단을 예측하는 것이 목적**

$$Z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

X_p : 판별변수

β_p : 판별계수

Z = 판별점수



선형판별분석 제약조건

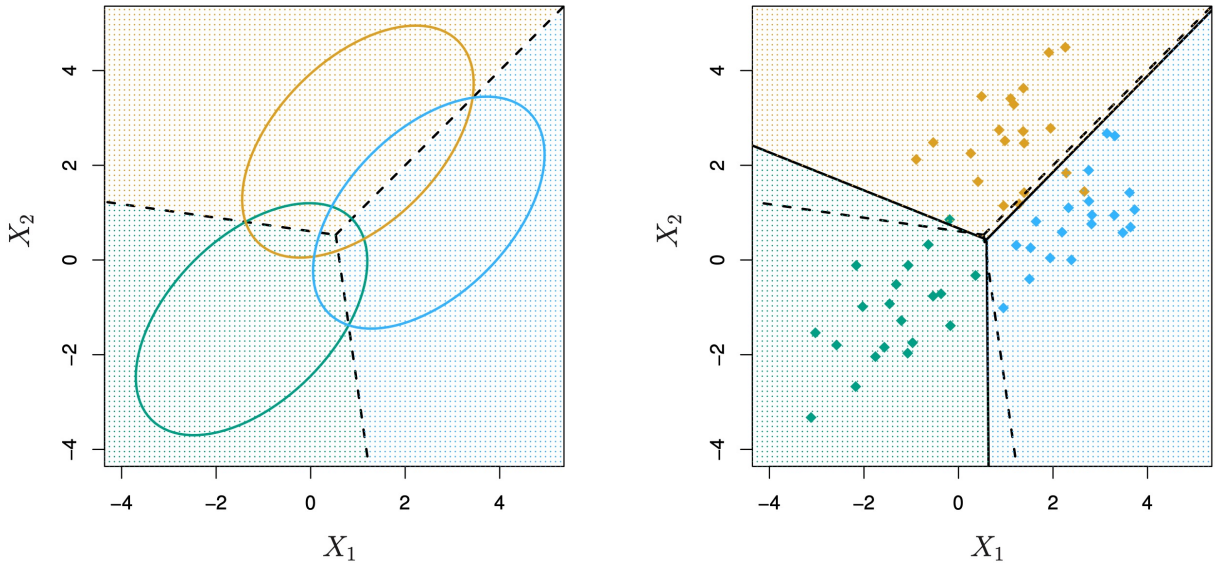
- 종속변수가 범주형 데이터, 독립변수의 척도가 등간이나 비율척도
- 독립변수는 정규분포를 따라야 함
- 종속변수와 독립변수는 상관관계를 가져야 함
- 독립변수들 사이에 상관관계가 작거나 없어야 함
- 종속변수로 구분되는 각 집단 별 공분산 행렬이 유사해야 함

$$\begin{pmatrix} cov(X1,X1),\dots,cov(X1,Xn) \\ cov(X2,X1),\dots,cov(X2,Xn) \\ cov(X3,X1),\dots,cov(X3,Xn) \\ \dots \\ cov(Xn,X1),\dots,cov(Xn,Xn) \end{pmatrix}$$

→ 2개의 확률변수의 상관정도를 나타내는 값

선형판별분석 제약조건 예시

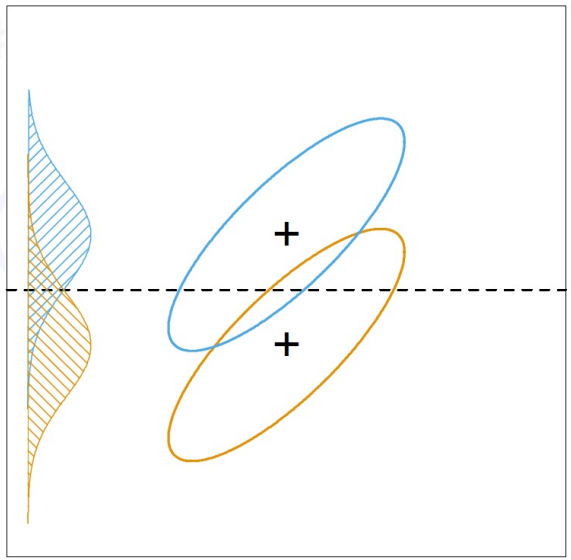
- 독립변수는 정규분포를 따라야 함
- 종속변수로 구분되는 각 집단 별 공분산 행렬이 유사해야 함



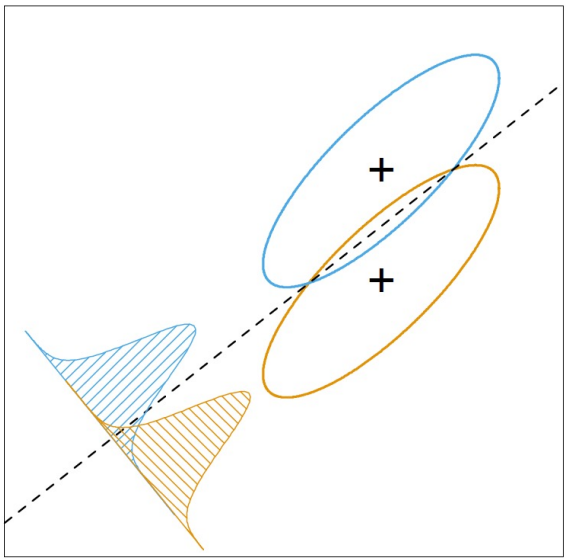
https://web.stanford.edu/class/stats202/notes/Classification/LDA.html

선형판별분석의 팁

- 분류하고자 하는 두 범주의 중심(평균)이 서로 멀수록 좋다
- 분류하고자 하는 두 범주의 분산이 작을수록 좋다



(a)



(b)

선형판별분석(LDA) 실습

실습에 사용할 데이터세트

sklearn.datasets 의 load_iris() 사용

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

load_boston() Load and return the boston house-prices dataset (regression).

load_iris() Load and return the iris dataset (classification).

load_diabetes() Load and return the diabetes dataset (regression).

load_digits() Load and return the digits dataset (classification).

load_linnerud() Load and return the physical excercise linnerud dataset.

load_wine() Load and return the wine dataset (classification).

load_breast_cancer() Load and return the breast cancer wisconsin dataset (classification)

Dict 포맷을 제공하는 데이터세트

sklearn.datasets 의 load_iris() 사용

- .DESCR : 데이터셋의 description
- .feature_names : feature 인덱스명
- .data : feature 데이터
- .target_names : class(target) 인덱스명
- .target : class(target) 데이터

선형판별분석의 주요함수

https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

class sklearn.discriminant_analysis.LinearDiscriminantAnalysis

decision_function(X)	Apply decision function to an array of samples.
fit(X, y)	Fit LinearDiscriminantAnalysis model according to the given
fit_transform(X[, y])	Fit to data, then transform it.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict class labels for samples in X.
predict_log_proba(X)	Estimate log probability.
predict_proba(X)	Estimate probability.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(**params)	Set the parameters of this estimator.
transform(X)	Project data to maximize class separation.

- sklearn.datasets 모듈의 load_iris() 로 데이터 로딩

Code

```
from sklearn.datasets import load_iris
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#iris 데이터셋 로드와 Dict 포맷의 키 확인하기
data = load_iris()
print("iris dataset format and keys\n",data.keys())

#feature name과 관측값 가져오기
iris_data = data['data']
iris_cols = data['feature_names']
print("iris dataset columns\n",iris_cols)
```

- **sklearn.datasets 모듈의 load_iris() 로 데이터 로딩**

Result	iris dataset format and keys dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename']) iris dataset columns ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

- **Column name 재설정 및 데이터프레임(독립변수, 종속변수) 생성**

Code

```
#column name을 재설정
iris_cols=['sep_len', 'sep_wt', 'pet_len', 'pet_wt']

#데이터프레임 생성하기 ==> 학습시키기 위한 데이터(독립변수들)
iris_df = pd.DataFrame(data= iris_data, columns= iris_cols)
print(iris_df.head(5))

#데이터프레임에 학습 데이터의 정답값(라벨) 데이터 추가(종속변수)
iris_df['label'] = data['target']
print(iris_df.head(5))
```

- **Column name 재설정 및 데이터프레임(독립변수, 종속변수) 생성**

Result

sep_len	sep_wt	pet_len	pet_wt
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

sep_len	sep_wt	pet_len	pet_wt	label
5.1	3.5	1.4	0.2	0.0
4.9	3.0	1.4	0.2	0.0
4.7	3.2	1.3	0.2	0.0
4.6	3.1	1.5	0.2	0.0
5.0	3.6	1.4	0.2	0.0

0 : setosa,
1 : versicolor,
2 : virginica

- **Groupby를 이용한 데이터 카운팅**

Code

```
26 #종속변수 각 그룹에 대해 데이터 카운팅 해보기
27 check_df = iris_df.groupby(by='label').count()
28 print(check_df)
```

- **Groupby를 이용한 데이터 카운팅**

Result

label	sep_len	sep_wt	pet_len	pet_wt
0	50	50	50	50
1	50	50	50	50
2	50	50	50	50

0 : setosa,
1 : versicolor,
2 : virginica

STEP.3 LDA 수행하기

- LDA 오브젝트 생성과 환경구성

Code

```
30 #LDA 수행
31 #LDA 패키지 import
32 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
33
34 #학습시키기 위한 독립변수와 종속변수 분할하기
35 X_train = iris_df[iris_cols]
36 y_train = iris_df['label']
37
38 #LDA 오브젝트생성 및 독립변수와 종속변수를 이용해 LDA 환경구성
39 lda = LDA().fit(X_train, y_train)
```

- LDA 오브젝트 생성과 환경구성

Code

```
print("판별식 선형계수\n", lda.coef_)
print("판별식 상수\n", lda.intercept_)
y_pred = pd.DataFrame(lda.predict(X_train))
print("예측결과\n", y_pred.head(5))
y_pred_score = pd.DataFrame(lda.predict_proba(X_train))
print("예측스코어\n", y_pred_score.head(5))
print("예측정확도\n", lda.score(X_train,y_train))
```

- LDA 오브젝트 생성과 환경구성 결과 확인하기

Result

```
판별식 선형계수
 [[  6.31475846  12.13931718 -16.94642465 -20.77005459]
 [ -1.53119919  -4.37604348   4.69566531   3.06258539]
 [ -4.78355927  -7.7632737   12.25075935  17.7074692 ]]
판별식 상수
 [-15.47783673  -2.02197415 -33.53768674]
```

$$Z1 = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Z2, Z3는 위와 동일

▪ LDA 오브젝트 생성과 환경구성 결과 확인하기

Result

```
예측결과
    0
0   0
1   0
2   0
3   0
4   0
예측스코어
        0             1             2
0   1.0   3.896358e-22   2.611168e-42
1   1.0   7.217970e-18   5.042143e-37
2   1.0   1.463849e-19   4.675932e-39
3   1.0   1.268536e-16   3.566610e-35
4   1.0   1.637387e-22   1.082605e-42
예측정확도
 0.98
```

0 : setosa,
1 : versicolor,
2 : virginica

STEP.3 LDA 수행하기

- **LDA를 통한 분류결과 확인하기**

Code

```
# 분류 결과 확인하기
from sklearn.metrics import confusion_matrix

conf_df = pd.DataFrame(confusion_matrix(y_train, lda.predict(X_train)))
conf_df.columns=['pred 0', 'pred 1', 'pred 2']#setosa,versicolor,virginica
conf_df.index = ['real 0', 'real 1', 'real 2']
print('Confusion Matrix \n',conf_df)
```

- **LDA를 통한 분류결과 확인하기**

Result

```
Confusion Matrix
          pred 0  pred 1  pred 2
real 0        50       0       0
real 1         0      48       2
real 2         0       1      49
```

0 : setosa,
1 : versicolor,
2 : virginica

- 시각화로 LDA 수행 전, 후 비교하기

Code

```
#시각화로 확인해보기
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
cld=LinearDiscriminantAnalysis()

X_lda = cld.fit_transform(X_train, y_train)
print(X_lda.shape)
```

```
#데이터셋 시각화 해보기
fig, ax = plt.subplots(ncols=2)

sns.scatterplot(iris_df['sep_len'], iris_df['sep_wt'],
                hue=iris_df['label'], ax=ax[0])
sns.scatterplot(X_lda[:,0], X_lda[:,1], hue=y_train, ax=ax[1])
plt.show()
```

- 시각화로 LDA 수행 전, 후 비교하기

Result