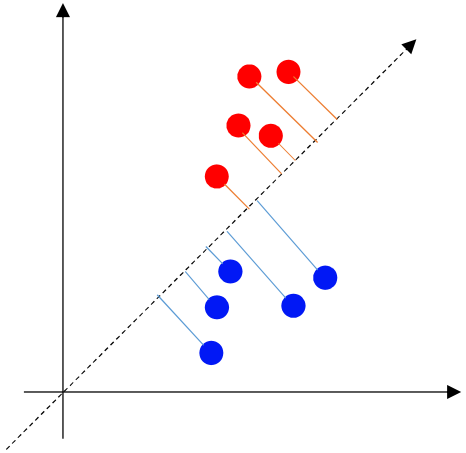
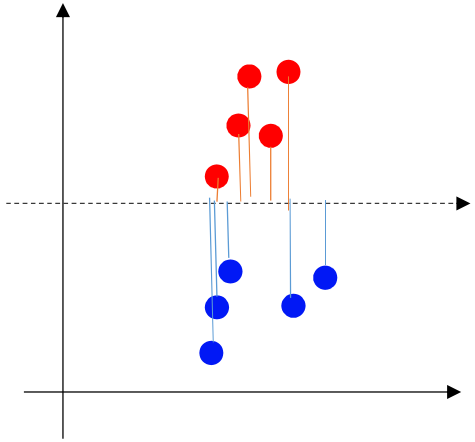
판별분석

판별분석(Discriminant Analysis)

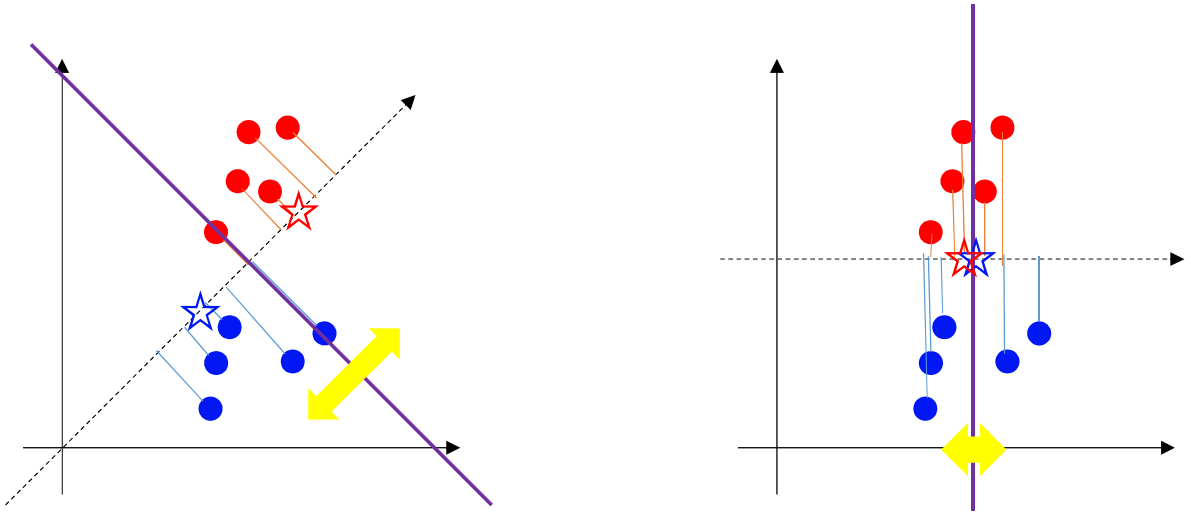
- 분류와 차원축소 2개의 목적을 위해 사용되는 알고리즘
- 클래스별 공분산 구조 비슷하다면 선형판별분석(LDA), 그렇지 않다면 이차판변분석(QDA) 사용
- 클래스 내 분산과 클래스 간 분산의 비율을 최대화하는 방식으로 차원 축소
- 직선 위 투영된 데이터들이 같은 클래스는 가깝게, 다른 클래스는 멀게 위치하도록 학습





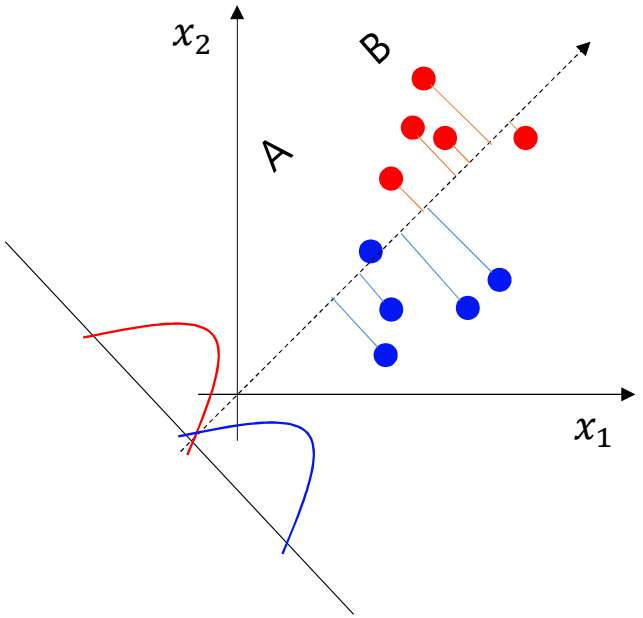
결정경계(Decision Boundary)

- 투영되는 선(축)과 직교하고, 클래스 간 투영된 데이터가 겹치는 영역이 작은 지점
- SB(Between-Class Scatter), 클래스 간 분산이 큰 지점
- SW(Within-Class Scatter), 클래스 내 분산이 작은 지점



선형판별분석(LDA)

- 클래스를 구분하는데 기여할 수 있는 중요도가 높은 판별변수 결정
- 클래스 구분의 기준이 되는 독립변수들의 선형결합으로 구성된 판별함수 도출
- 도출된 판별함수에 의해 학습 데이터 분류의 정확도 분석
- 판별함수를 이용하여 새로운 데이터, 즉 테스트 데이터 예측



$$z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

sklearn.discriminant_analysis.LinearDiscriminantAnalysis

class sklearn.discriminant_analysis.LinearDiscriminantAnalysis(*solver='svd', shrinkage=None, priors=None, n_components=None, store_covariance=False, tol=0.0001, covariance_estimator=None*) [source]

Linear Discriminant Analysis.

n_components : *int, default=None*

Number of components (<= min(n_classes - 1, n_features)) for dimensionality reduction. If None, will be set to min(n_classes - 1, n_features). This parameter only affects the `transform` method.

```
>>> import numpy as np
>>> from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> y = np.array([1, 1, 1, 2, 2, 2])
>>> clf = LinearDiscriminantAnalysis()
>>> clf.fit(X, y)
LinearDiscriminantAnalysis()
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

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

LDA분류

```
import seaborn as sns
iris = sns.load_dataset("iris")

data = iris.drop("species", axis = 1)
t = iris[["species"]].copy()
t[t["species"] == "setosa"] = 0
t[t["species"] == "versicolor"] = 1
t[t["species"] == "virginica"] = 2
t = t["species"].astype("int")

from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(
    data, t, test_size = 0.3, random_state = 42, stratify = t)
```

```
Train-Eval: 0.9714285714285714
Test-Eval : 0.9777777777777777
[[15  0  0]
 [ 0 15  0]
 [ 0  1 14]]
```

데이터 표준화

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
model = LinearDiscriminantAnalysis(n_components = 2)
model.fit(train_data_scaled, train_target)
print("Train-Eval:", model.score(train_data_scaled, train_target))
print("Test-Eval :", model.score(test_data_scaled, test_target))

from sklearn.metrics import confusion_matrix
conf = confusion_matrix(test_target, model.predict(test_data_scaled))
print(conf)
```

차원축소

n_components : *int, default=None*

Number of components (<= min(n_classes - 1, n_features)) for dimensionality reduction. If None, will be set to min(n_classes - 1, n_features). This parameter only affects the `transform` method.

```
train_data_lda = model.transform(train_data_scaled)
print(model.coef_)
print(model.intercept_)

import matplotlib.pyplot as plt
plt.xlabel('LDA1')
plt.ylabel('LDA2')
plt.scatter(                                    , c=train_target)
plt.show()
```

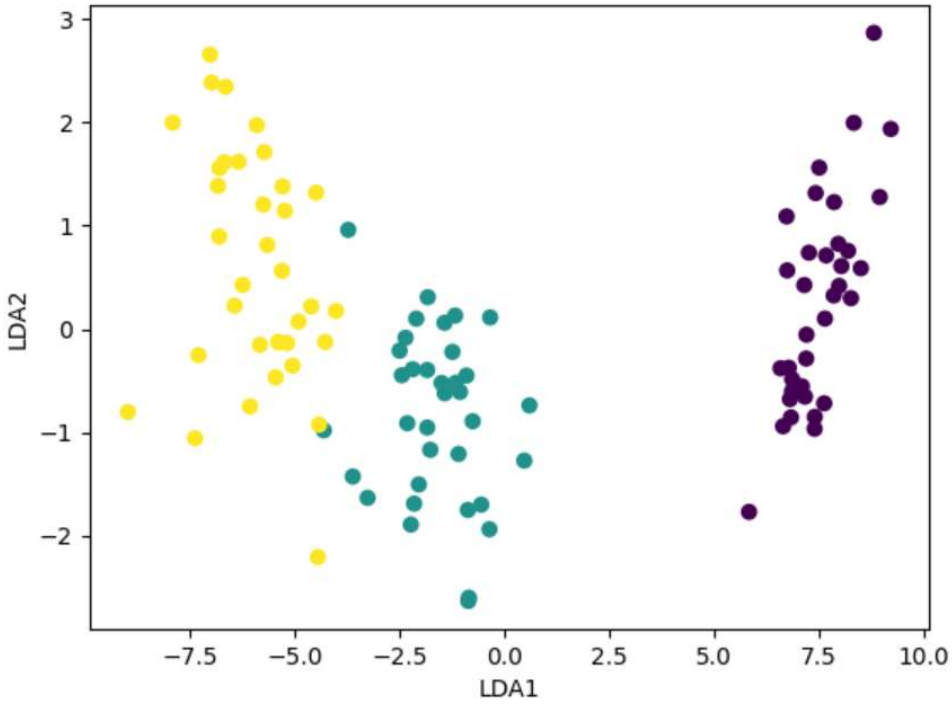
구현

변수 4개

클래스 3개

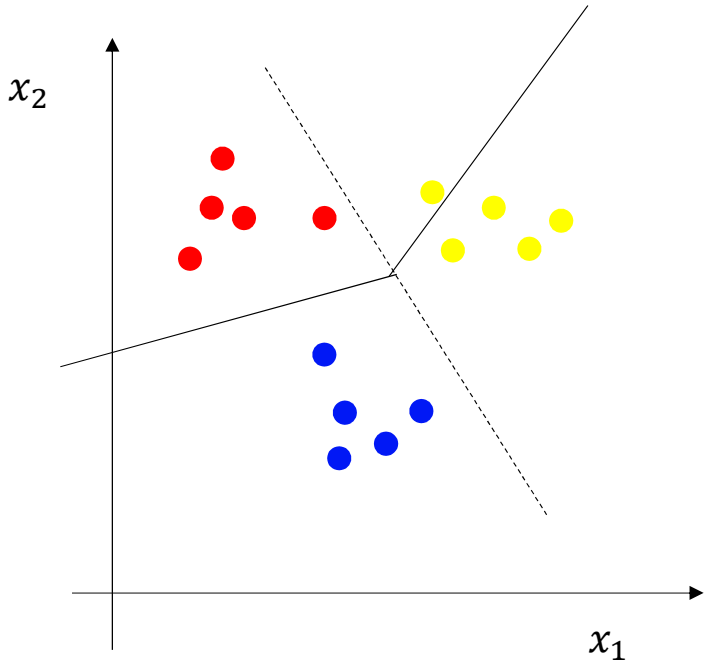
```
[[  5.07607338   4.74964533 -28.0269806  -17.28835989]
 [ -1.19926353  -1.76861284   7.54898745   1.91901225]
 [ -3.87680984  -2.98103249  20.47799315  15.36934765]]
[-29.3503556   -2.76460787 -18.6422177 ]
```

$$z_t = w_{t0} + w_{t1}x_1 + w_{t2}x_2 + \cdots + w_{tn}x_n$$



변수 2개

판별점수



$$z_t = w_{t0} + w_{t1}x_1 + w_{t2}x_2 + \cdots + w_{tn}x_n$$

```
[[  5.07607338   4.74964533 -28.0269806  -17.28835989]
 [ -1.19926353  -1.76861284   7.54898745   1.91901225]
 [ -3.87680984  -2.98103249  20.47799315  15.36934765]]
[-29.3503556   -2.76460787 -18.6422177 ]
```

```
Target
 107    2
63     1
Name: species, dtype: int32
Score -75.51865953602342 8.014278439947056 16.74719992017384
Score -48.41871780123347 1.8953544627921832 -4.233817837461242
```

z_1 z_2 z_3

sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis

class sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis(*, *priors=None*, *reg_param=0.0*, *store_covariance=False*, *tol=0.0001*) [source]

Quadratic Discriminant Analysis.

```
>>> from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> y = np.array([1, 1, 1, 2, 2, 2])
>>> clf = QuadraticDiscriminantAnalysis()
>>> clf.fit(X, y)
QuadraticDiscriminantAnalysis()
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

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

QDA분류

구현

```
from sklearn.metrics import confusion_matrix
conf = confusion_matrix(test_target, model.predict(test_data_scaled))
print(conf)
```

```
Train-Eval: 0.9714285714285714
Test-Eval : 0.9777777777777777
[[15  0  0]
 [ 0 15  0]
 [ 0  1 14]]
```

LDA

```
Train-Eval: 0.9809523809523809
Test-Eval : 0.9777777777777777
[[15  0  0]
 [ 0 15  0]
 [ 0  1 14]]
```

QDA

참고자료

- 지능기전공학부 최유경 교수님 자료, https://github.com/sejongresearch/2021.MachineLearning
- 코랩(Colab), https://colab.research.google.com/
- 파이썬(Python), https://www.python.org/doc/
- 사이킷런(sckit-learn), https://scikit-learn.org/stable/index.html
- 판다스(pandas), https://pandas.pydata.org/
- 맷플롯립(matplotlib), https://matplotlib.org/
- 씨본(seaborn), https://seaborn.pydata.org/
- 캐글(Kaggle), https://www.kaggle.com/
- 넘파이(numpy), https://numpy.org/doc/stable/
- 스택오퍼플러우(stackoverflow), https://stackoverflow.com/