

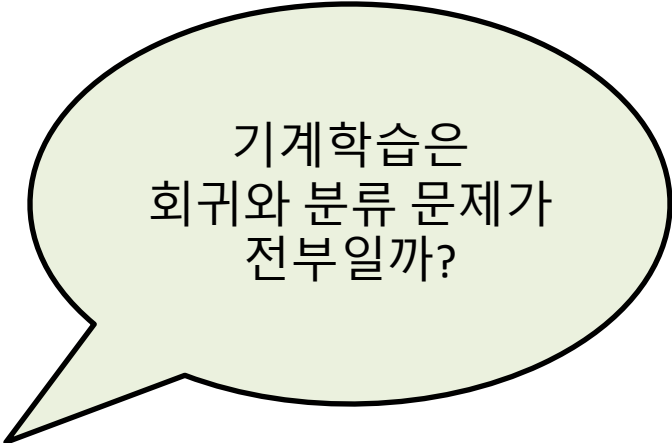
# 판별 분석

---

Discriminant analysis

# 기계학습 분야

- 회기/분류 문제 해결을 위한 솔루션
  - KNN
  - Logistic Regression
  - LDA → 오늘 배웁니다.
  - Bayesian
  - Decision Tree
  - SVM
  - Ensemble
  - Kmeans



기계학습은  
회귀와 분류 문제가  
전부일까?

# 기계학습 분야

## ■ 기계 학습의 연구 분야 카테고리

```
Machine Learning (ML)
├ ML: (Deep) Neural Network Algorithms
├ ML: (Deep) Neural Network Learning Theory
├ ML: Active Learning
├ ML: Adversarial Learning & Robustness
├ ML: Applications
├ ML: Bayesian Learning
├ ML: Bio-inspired Learning
├ ML: Calibration & Uncertainty Quantification
├ ML: Causal Learning
├ ML: Classification and Regression
├ ML: Clustering
├ ML: Dimensionality Reduction/Feature Selection
├ ML: Distributed Machine Learning & Federated Learning
├ ML: Ensemble Methods
├ ML: Ethics -- Bias, Fairness, Transparency & Privacy
├ ML: Evaluation and Analysis (Machine Learning)
├ ML: Evolutionary Learning
├ ML: Feature Construction/Reformulation
├ ML: Graph-based Machine Learning
├ ML: Hyperparameter Tuning / Algorithm Configuration
├ ML: Imitation Learning & Inverse Reinforcement Learning
├ ML: Kernel Methods
├ ML: Learning on the Edge & Model Compression
├ ML: Learning Preferences or Rankings
├ ML: Learning Theory
├ ML: Learning with Manifolds
├ ML: Matrix & Tensor Methods
├ ML: Multi-class/Multi-label Learning & Extreme Classification
├ ML: Multi-instance/Multi-view Learning
├ ML: Multimodal Learning
├ ML: Neural Generative Models & Autoencoders
├ ML: Online Learning & Bandits
├ ML: Optimization
├ ML: Other Foundations of Machine Learning
├ ML: Probabilistic Graphical Models
├ ML: Quantum Machine Learning
├ ML: Reinforcement Learning
├ ML: Relational Learning
├ ML: Representation Learning
├ ML: Scalability of ML Systems
├ ML: Semi-Supervised Learning
├ ML: Structured Prediction
├ ML: Time-Series/Data Streams
├ ML: Transfer/Adaptation/Multi-task/Meta/Automated Learning
├ ML: Unsupervised & Self-Supervised Learning
```

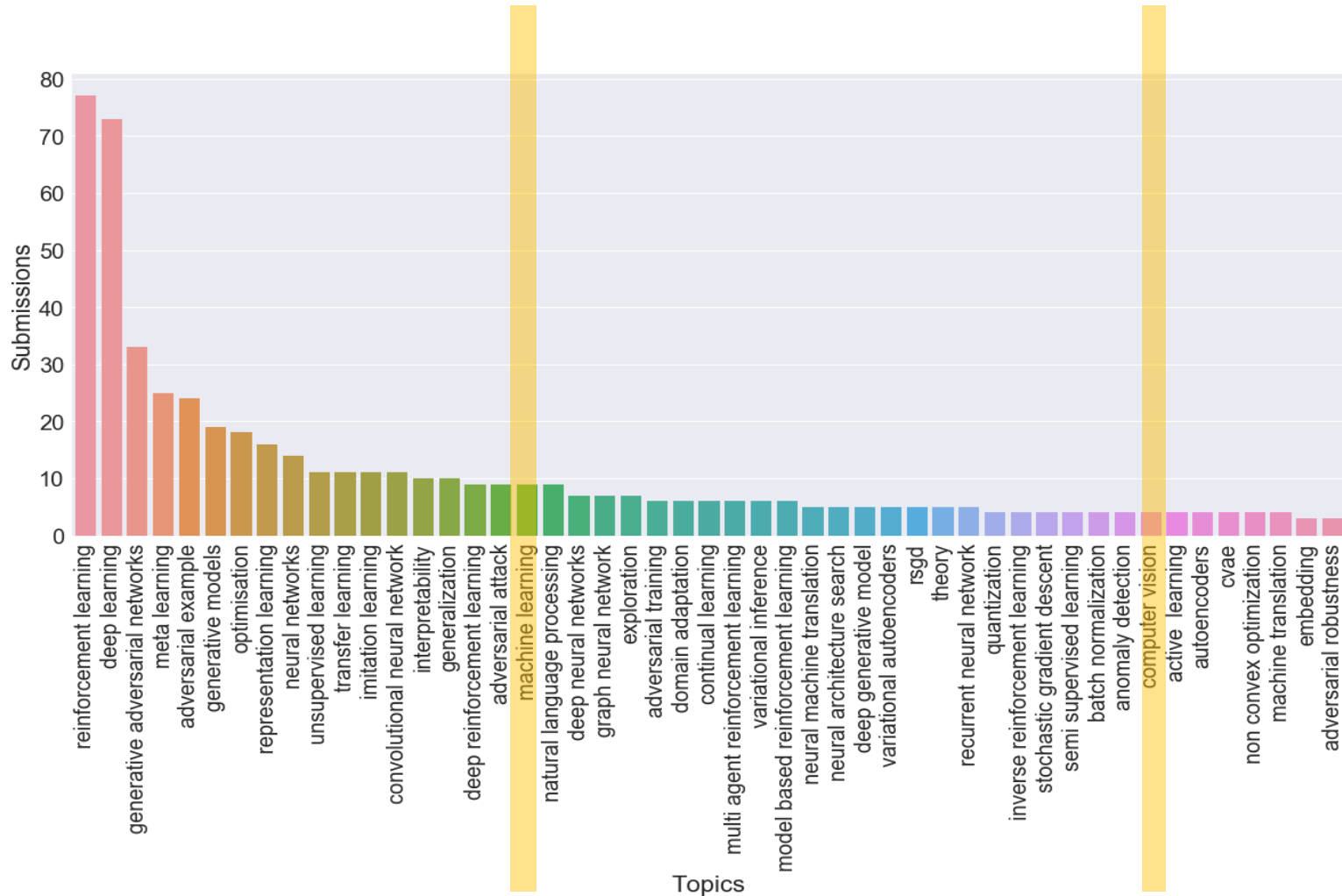
→ 이번학기

*\* Research Areas in Machine Learning  
(From AAAI2021, NeurIPS)*

# 기계학습 분야

## ■ 키워드 통계 (머신 러닝 분야)

ICLR 2019: stats & trends



# 판별 분석 (Discriminant Analysis)

## 판별 분석의 정의

전체집단

연구자가 측정, 관찰한 결과의 집합

- 두 개 이상의 **모 집단**에서 추출된 **표본**들이 지니고 있는 **정보**를 이용하여 이 표본들이 어느 모집단에서 추출된 것인지를 결정해 줄 수 있는 **기준을 찾는 분석법**

분리기준

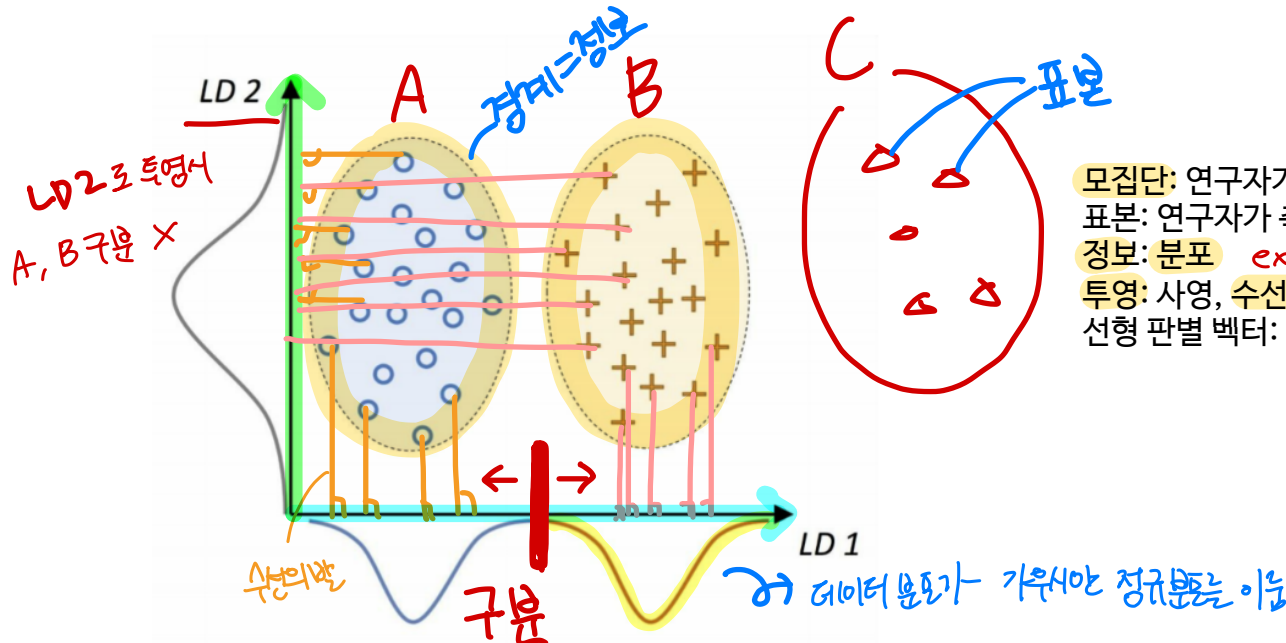
## 좋은 기준/나쁜 기준의 예



x1축(LD1)에 투영된 **선형 판별 벡터**는 두 가지 클래스로 잘 구분해 줌



x2축(LD2)에 투영된 **선형 판별 벡터**는 클래스 판별 정보가 없어 좋은 선형 판별 벡터 아님



모집단: 연구자가 알고 싶어하는 대상 / 집단 전체

표본: 연구자가 측정 또는 관찰한 결과의 집합

정보: 분포 ex) 가우시안 정규분포

투영: 사영, 수선의 발 Projection by 행렬곱

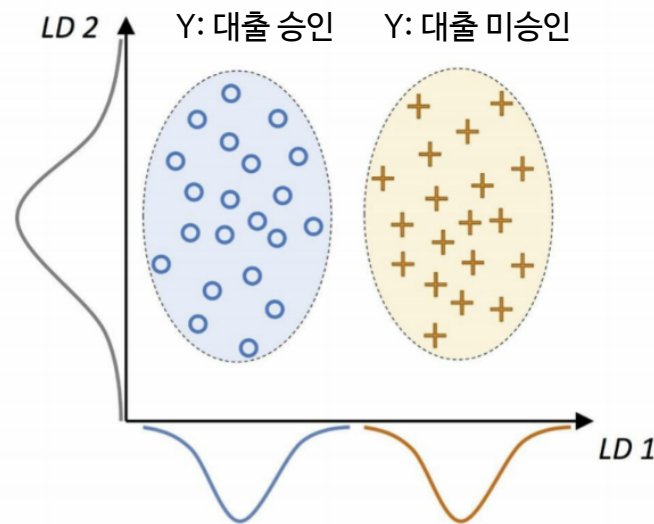
선형 판별 벡터: 지금부터 배울 LDA

# 판별 분석 (Discriminant Analysis)

## ■ 판별 분석의 정의

- 두 개 이상의 **모 집단**에서 추출된 **표본**들이 지니고 있는 **정보**를 이용하여 이 표본들이 어느 모집단에서 추출된 것인지를 결정해 줄 수 있는 **기준을 찾는 분석법**
- 예) 은행에서 부동산 담보 대출을 행하고자 할 경우 채무자가 대출금을 갚을 것인가?
  - 이 경우 과거 대출금을 반환치 않은 **사람의 정보 유형**(연령, 소득, 결혼 유무 등)을 참고하여 담보 신청 시 신청자의 정보 유형을 과거의 유형과 비교하여 **장래 변제 가능성을** 파악할 수 있음. (\*학습 기반 분류 방법의 핵심)

갚을지 안갚을지



2차원 데이터에서

$X_1$ : 연령,  $X_2$ : 소득

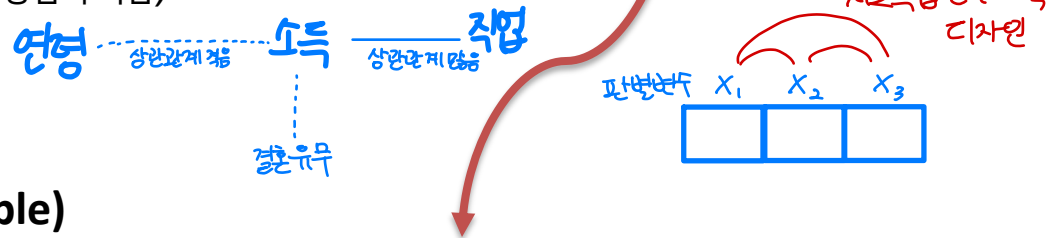
$Y$ : 대출 여부

$Y =$  0 대출 X - 안갚을 것으로 예측한 사람  
1 대출 O - 갚을 것으로 " "

# 판별 분석 (Discriminant Analysis)

## ■ 판별 분석의 정의

- 두 개 이상의 모 집단에서 추출된 표본들이 지니고 있는 정보를 이용하여 이 표본들이 어느 모집단에서 추출된 것인지를 결정해 줄 수 있는 기준을 찾는 분석법
- 예) 은행에서 부동산 담보 대출을 행하고자 할 경우 채무자가 대출금을 갚을 것인가?
  - 이 경우 과거 대출금을 반환치 않은 **사람의 정보 유형** (연령, 소득, 결혼 유무 등)을 참고하여 담보 신청 시 신청자의 정보 유형을 과거의 유형과 비교하여 **장래 변제 가능성**을 파악할 수 있음. (\*학습 기반 분류 방법의 핵심)

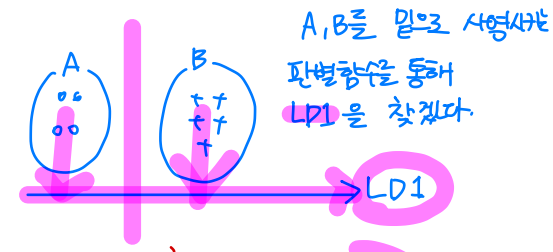


## ■ 판별 분석의 기초 개념

### ■ ① 판별 변수 (Discriminant variable)

판별 변수는 어떤 집단에 속하는지 판별하기 위한 변수로서 **독립 변수** 중 판별력이 높은 변수를 뜻함. 판별 변수를 선택하는데 판별 기여도 외에 고려해야 할 사항은 다른 독립 변수들과의 **상관관계**이며, 상관관계가 높은 두 독립변수를 선택하는 것보다는 **두 독립변수 중 하나를 판별 변수로 선택하고 그것과 상관관계가 적은 독립변수를 선택**함으로써 효과적인 판별 함수를 만들 수 있음

# 판별 분석 (Discriminant Analysis)



## ② 판별 함수 (Discriminant function)

판별 분석이 이용되기 위해서는 각 개체는 여러 집단 중에서 어느 집단에 속해 있는지 알려져 있어야 하며 (지도 학습) 소속 집단이 이미 알려진 경우에 대하여 변수들(X)을 측정하고 이들 변수들을 이용하여 각 집단을 가장 잘 구분해 낼 수 있는 판별식을 만들어 분별하는 과정을 포함하게 됨.

Like 최적화 방법, 라그랑주 곱  $\sum_{i=1}^N |Y_i - \bar{Y}_i|_2$  이 최소값으로 갖도록 해준다.

즉, 판별 함수를 이용하여 각 개체들이 소속 집단에 얼마나 잘 판별되는가에 대한 판별력을 측정하고 새로운 대상을 어느 집단으로 분류할 것이냐를 예측하는데 주요 목적이 있음

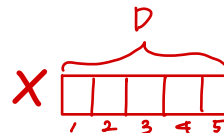
## ③ 판별 점수 (Discriminant score)

- 판별 점수는 어떤 대상이 어떤 집단에 속하는지 판별하기 위하여 그 대상의 판별 변수들의 값을 판별 함수에 대입하여 구한 값을 뜻함

## ④ 표본의 크기

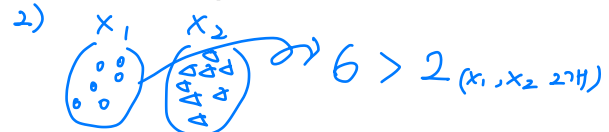
$N$  (학습 데이터수)

$N > 3D$



- 1) 전체 표본의 크기는 통상적으로 독립변수의 개수보다 3배(최소 2배) 이상 되어야 함
- 2) 종속 변수의 집단 각각의 표본의 크기 중 최소 크기가 독립변수의 개수보다 커야 함 (판별력을 좌우하는 것이 전체 표본의 수가 아니라 가장 적은 집단의 표본 수이기 때문임)

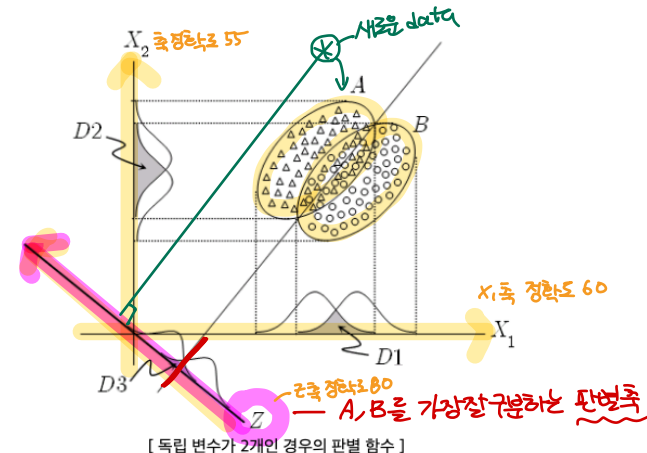
→ 사람 150명 / 독립 변수  $X_1 = \text{결혼 여부}$ ,  $X_2 = \text{소득}$   $150 > 3 \times 2$





A diagram illustrating a 2D array (matrix) with dimensions  $N$  (rows) and  $D$  (columns). A red 'X' is placed above the array, and a red 'Y' is placed above a vertical vector to its right.

- $$Z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$
- Z: 판별점수,  $\beta_0$ : 판별상수,  $X_1, X_2, \dots, X_p$ : 판별변수,  $\beta_1, \beta_2, \dots, \beta_p$ : 판별계수



# 선형 판별 분석 배경 LDA

## 가정 (Assumptions)

- ① 각 클래스 집단은 **정규분포(normal distribution)** 형태의 **확률분포**를 가짐
- ② 각 클래스 집단은 **비슷한 형태의 공분산(covariance)** 구조를 가짐 ... ?
  - 공분산: 2개의 확률변수의 상관 정도를 나타내는 값

완전 공분산 정규 분포

$$\Sigma = \begin{bmatrix} \sigma_1^2 & c_{12} \\ c_{12} & \sigma_2^2 \end{bmatrix}$$

타원축에

대각 공분산 정규 분포

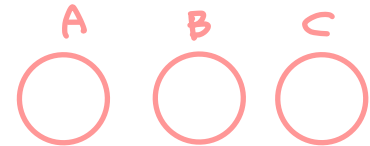
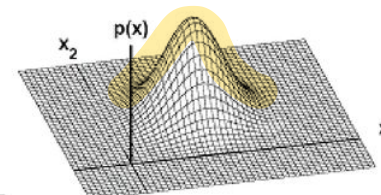
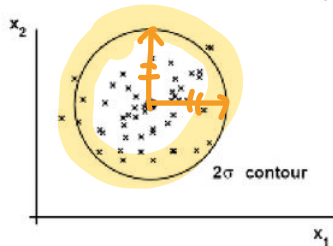
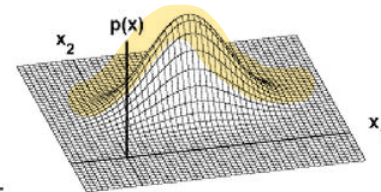
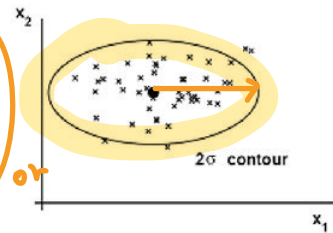
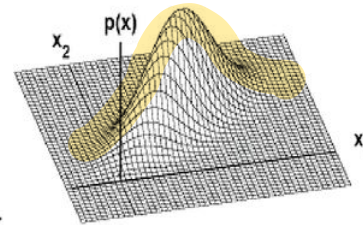
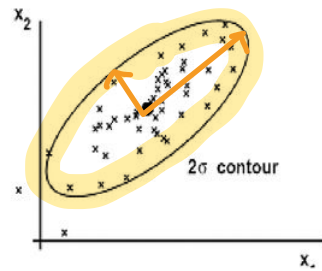
$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

타원

구형 공분산 정규 분포

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

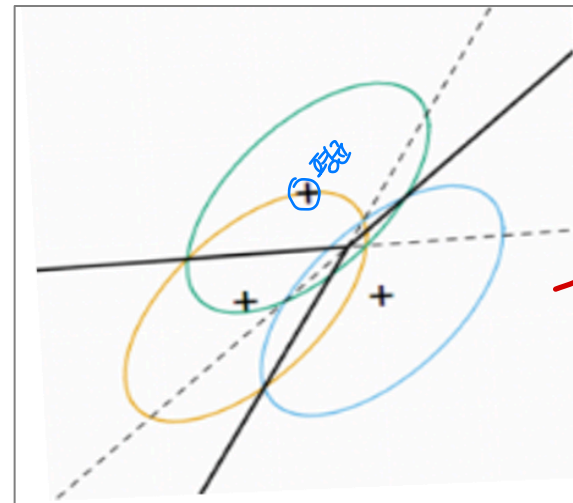
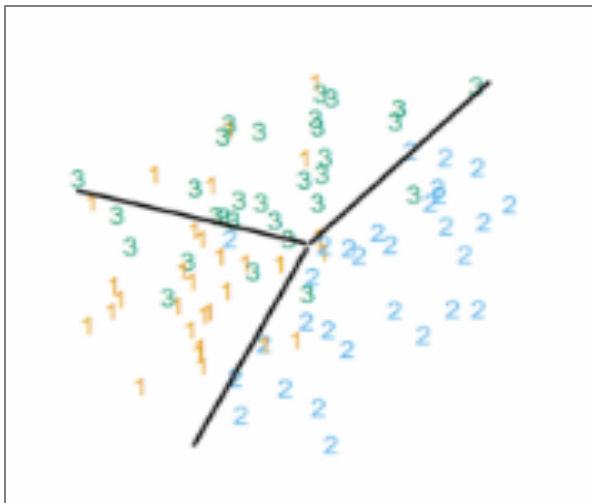
원



# 선형 판별 분석 배경

- 가정 (Assumptions)

- 각 클래스 집단은 정규분포(normal distribution) 형태의 확률분포를 가짐
- 각 클래스 집단은 비슷한 형태의 공분산(covariance) 구조를 가짐
  - 공분산: 2개의 확률변수의 상관 정도를 나타내는 값

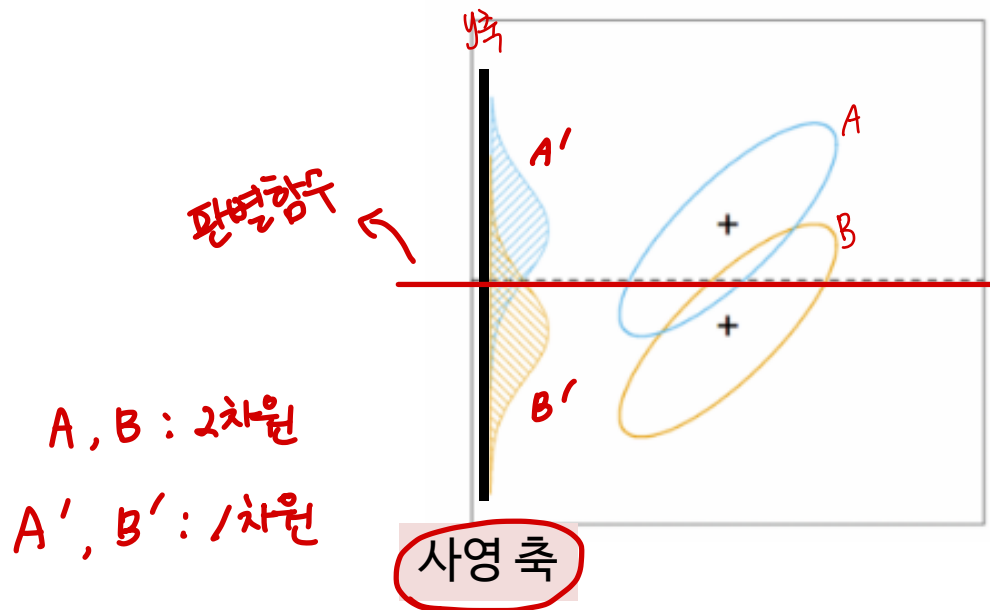


→ 완전공분산  
정규분포

가정의 예시

# 선형 판별 분석 배경

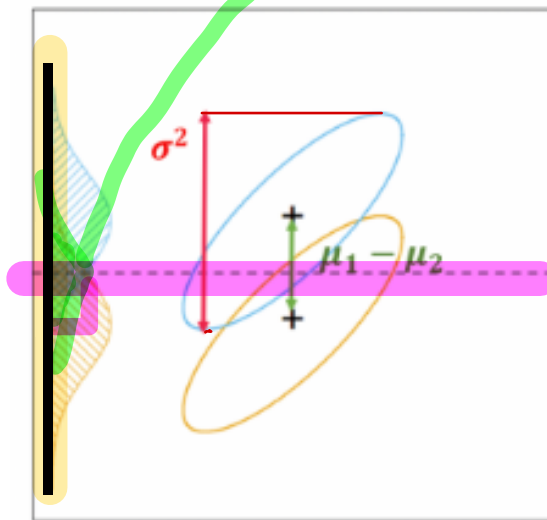
- ① LDA는 판별과 ② 차원 축소의 기능
  - 2차원(두 가지 독립변수)의 두 가지 범주(주황, 파랑)를 갖는 데이터를 분류하는 문제에서 LDA는 먼저 하나의 차원(1d)에 projection을 하여 차원을 축소시킴
    - LDA는 차원 축소의 개념을 포함함
    - 2차원 자료들을 판별 축에 정사영 시킨 분포의 형태를 고려



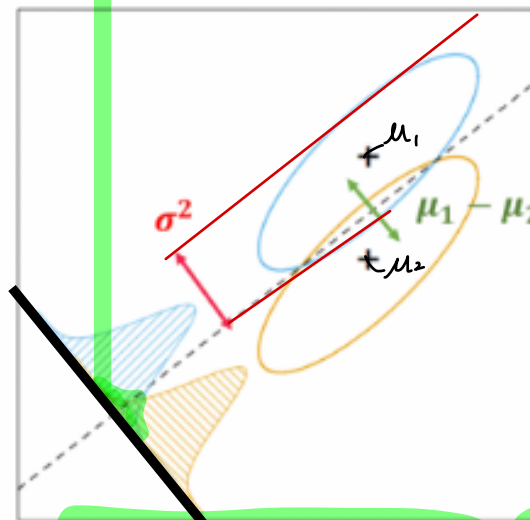
X		Y
N		A
		B
		A
		A
		B
$x_1$ $x_2$		

# 선형 판별 분석 배경

- LDA의 결정 경계(decision boundary)의 특징
  - Projection 축(실선)에 직교하는 축(점선)
  - 정사영(projection)은 두 분포의 특징이 아래의 목표를 달성해야 함
    - 각 클래스 집단의 **평균의 차이가 큰 지점**을 결정 경계로 지정
    - 각 클래스 집단의 **분산이 작은 지점**을 결정 경계로 지정
  - 즉, **분산 대비 평균의 차이를 극대화 하는 결정 경계를 찾고자 하는 것**
    - [팁] 사영 데이터의 분포에서 겹치는 영역이 작은 결정 경계를 선택하면 됨



LDA 결정 경계 후보 #1



LDA 결정 경계 후보 #2

$|\mu_1 - \mu_2|$  클수록  
좋음  
 $\sigma^2$  작을수록 좋음

최적

# 선형 판별 분석(LDA) 요약

## ■ 결정 경계

- 분산 대비 평균의 차이를 극대화 하는 경계

## ■ 가정

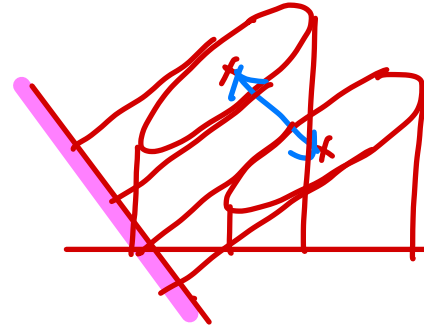
- 각 클래스 집단은 정규분포(normal distribution) 형태의 확률분포를 가짐
- 각 클래스 집단은 비슷한 형태의 공분산(covariance) 구조를 가짐

## ■ 장점

- 변수 간(x) 공분산 구조를 반영
- 공분산 구조 가정이 살짝 위반되더라도 비교적 Robust 하게 동작함

## ■ 단점

- 가장 작은 그룹의 샘플 수가 설명변수의 개수보다 많아야 함
- 정규분포 가정에 크게 벗어나는 경우 잘 동작하지 못함
- 범주 사이(y)에 공분산 구조가 많이 다른 경우를 반영하지 못함 (다음 슬라이드!!)  
→ 이차판별분석법(QDA)를 통해 해결 가능  
비선형일때



# 이차 판별 분석(QDA)

## Data Augmentation

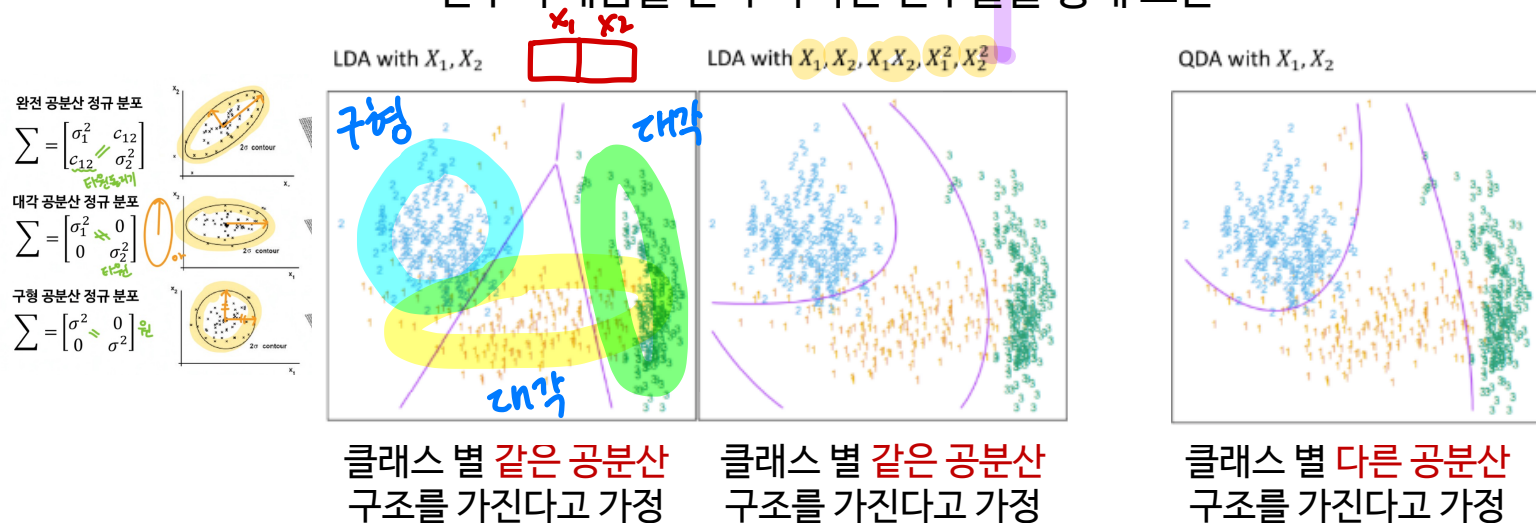
딥러닝 모델을 충분히 훈련하는데 필요한 데이터를 확보하는 기법으로  
적은 양의 훈련 데이터에 임의적인 변형을 가해 새로운 훈련 데이터를 대량 확보하는 방법론

### 이차판별분석의 정의

- K(범주의 수)와 관계없이 공통 공분산 구조에 대한 가정을 만족하지 못하면 QDA 적용
  - 즉, Y의 범주 별로 서로 다른 공분산 구조를 가진 경우에 활용 가능

### LDA와 QDA의 비교

- LDA의 결정 경계는 선형으로 가정하고 있어 서로 다른 공분산 분류에 어려움이 있음  
(첫번째 그림 참고)
  - 단, LDA도 같은 공분산의 비선형 분류 가능 (두번째 그림 참고)
    - 변수의 제곱을 한 추가적인 변수들을 통해 보완



# 이차 판별 분석(QDA)



- QDA는 서로 다른 공분산 데이터 분류 가능 (세번째 그림 참고)

- 상대적 장점: 비선형 분류 가능

- QDA는 서로 다른 공분산 데이터를 분류를 위해 샘플이 많이 필요

- 상대적 단점: 설명변수의 개수가 많을 경우, 추정해야 하는 모수가 많아짐 ?

- 즉, 연산량이 큼

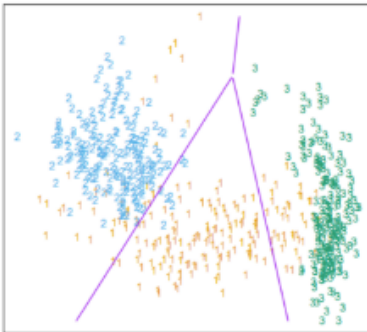
$$LDA: y = b + b_1 x_1 + b_2 x_2 + b_3 x_3$$

$$QDA: y = b + b_1 x_1 + b_2 x_1^2 + b_3 x_1^3$$

[11-2] 7:33

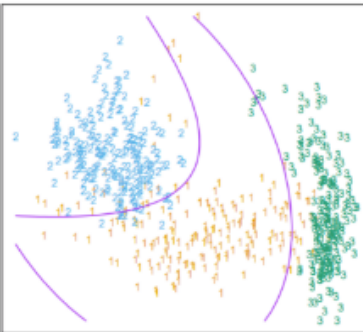
→ 파라미터 수가  
늘어난다?

LDA with  $X_1, X_2$



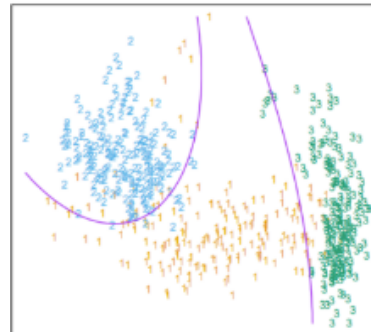
클래스 별 같은 공분산  
구조를 가진다고 가정

LDA with  $X_1, X_2, X_1 X_2, X_1^2, X_2^2$



클래스 별 같은 공분산  
구조를 가진다고 가정

QDA with  $X_1, X_2$



클래스 별 다른 공분산  
구조를 가진다고 가정



# 이차 판별 분석(QDA) 예시

공분산 ... 공복 ...

## QDA: 클래스 별 서로 다른 모수를 갖는 정규분포 분석

- 예를 들어  $y$ 가 1,2,3 이라는 3개의 클래스를 가지고 각 클래스에서의  $x$ 의 확률분포가 다음과 같은 기대값 및 공분산 행렬을 가진다고 가정하자.
- $y$ 의 사전 확률은 다음과 같이 동일하다.

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mu_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix} \Rightarrow \text{QDA적합}$$

(공분산 행렬의 대각선 요소는 0.7, 0.8, 0.8로 각각 다르며, 비대각선 요소는 0, 0.2, 0.2로 각각 다르다.)

$$P(Y = 1) = P(Y = 2) = P(Y = 3) = \frac{1}{3}$$

완전 공분산 정규 분포

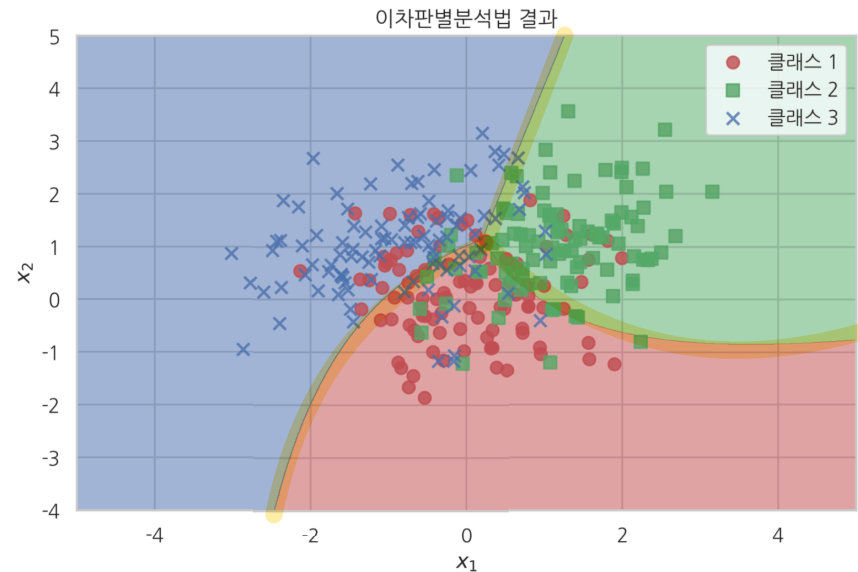
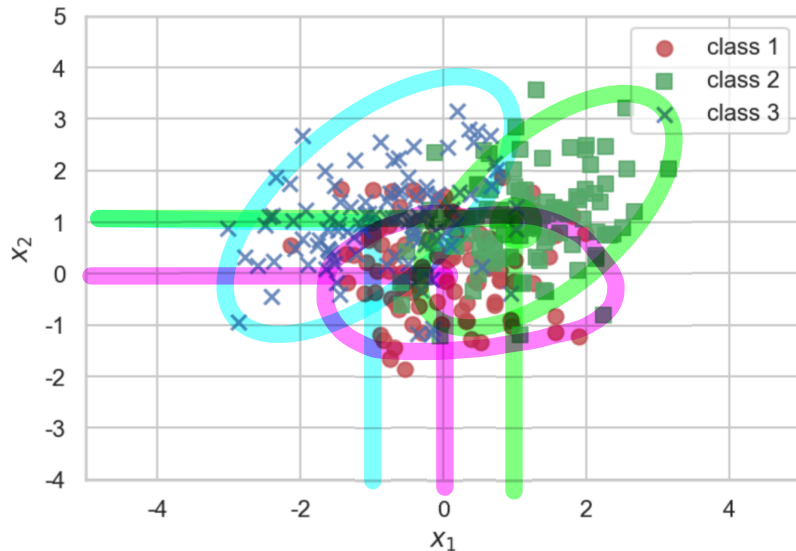
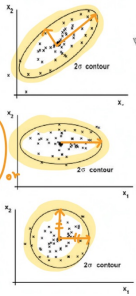
$$\Sigma = \begin{bmatrix} \sigma_1^2 & c_{12} \\ c_{12} & \sigma_2^2 \end{bmatrix}$$

대각 공분산 정규 분포

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

구형 공분산 정규 분포

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$



# 판별 분석 실습

---

Discriminant analysis

# 선형판별분석(LDA) 실습

- 실습 코드: [링크](#)
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x\_train/x\_test
- 학습/시험 데이터 라벨: y\_train/y\_test → (0,1,2)
- 데이터 로더, 범주형 데이터 변환, 데이터 분할
  - 3, 4장 실습과 동일

```
1 # Iris data 불러오기
2 import seaborn as sns # seaborn을 불러옴.
3 iris=sns.load_dataset('iris') # iris라는 변수명으로 Iris data를 download함.
4 X=iris.drop('species',axis=1) # 'species'열을 drop하고 특성변수 X를 정의함.
5 y_=iris['species'] # 'species'열을 label y를 정의함.
6
7 from sklearn.preprocessing import LabelEncoder # LabelEncoder() method를 불러옴
8 classle=LabelEncoder()
9 y=classle.fit_transform(iris['species'].values) # species 열의 문자형을 범주형 값으로 전환
10
11 from sklearn.model_selection import train_test_split
12 X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.3, random_state=123, stratify=y)
13
```

# 선형판별분석(LDA) 실습

- 실습 코드: [링크](#)
- 데이터셋: Iris 데이터 → (150 x 5) 즉, n\_samples = 150, n\_features = 5
- 학습/시험 데이터: x\_train/x\_test
- 학습/시험 데이터 라벨: y\_train/y\_test → (0,1,2) 즉, n\_classes = 3
- 선형판별분석 API: [Manual](#)
  - LinearDiscriminantAnalysis 클래스 호출
    - n\_components : 사영할 축의 차원

```
1 # Iris data에 대한 LDA 적합
2 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
3 cld=LinearDiscriminantAnalysis(store_covariance=True) → Covariance 값 (판별함수에서 추정하고자 하는 모두) 짚어보기
4
5 cld.fit(X_train, y_train) # LDA 적합
6 y_train_pred=cld.predict(X_train)
7 y_test_pred=cld.predict(X_test)
8
9
```

default 값 : False

# 선형판별분석(LDA) 실습 #1

- 실습 코드: [링크](#)
- 데이터셋: Iris 데이터 → (150 x 5) 즉, n\_samples = 150, n\_features = 5
- 학습/시험 데이터: x\_train/x\_test
- 학습/시험 데이터 라벨: y\_train/y\_test → (0,1,2) 즉, n\_classes = 3
- 선형판별평가 API: Accuracy & Confusion Matrix
  - 3, 4장 실습과 동일

```
>
10 from sklearn.metrics import accuracy_score
11 print(accuracy_score(y_train, y_train_pred)) # train data에 대한 accuracy
12 print(accuracy_score(y_test, y_test_pred)) # test data에 대한 accuracy
```

```
0.9714285714285714
0.9777777777777777
```

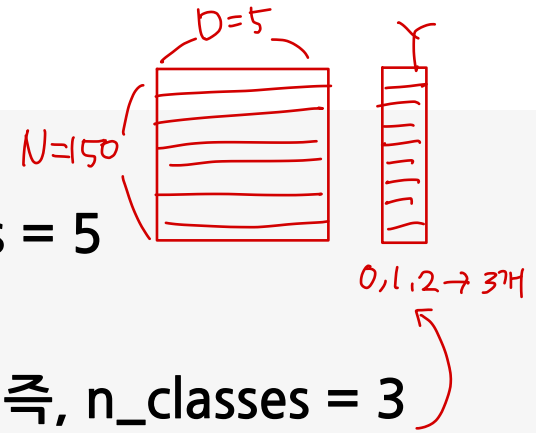
```
[24] 1 # 분류 결과
      2 from sklearn.metrics import confusion_matrix
      3 print(confusion_matrix(y_test, y_test_pred)) # 각 행은 setosa, versicolor, virginica
```

```
0.9777777777777777
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
```

→ 1개 틀림

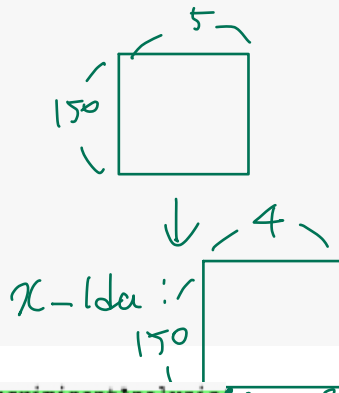
# 선형판별분석(LDA) 실습 #1

- 실습 코드: [링크](#)
- 데이터셋: Iris 데이터 → (150 x 5) 즉,  $n\_features = 5$
- 학습/시험 데이터:  $x\_train/x\_test$
- 학습/시험 데이터 라벨:  $y\_train/y\_test$  → (0,1,2) 즉,  $n\_classes = 3$
- Plot: transformed data

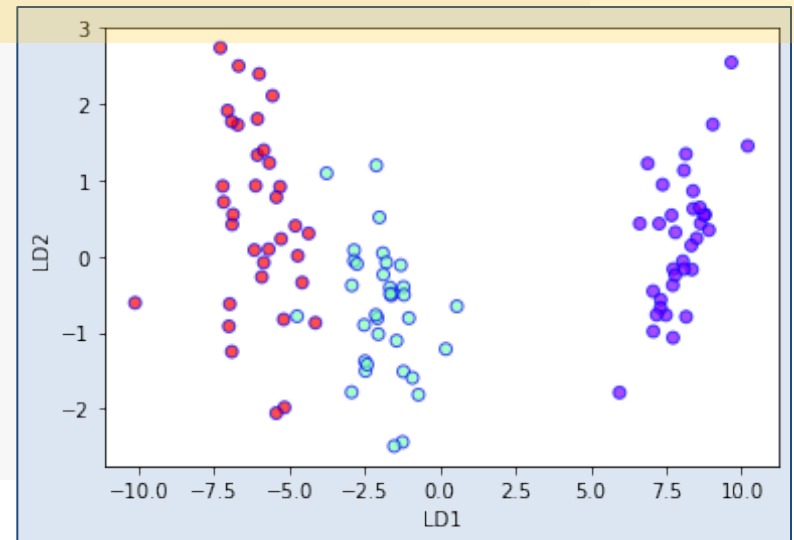


$n\_components: \min(n\_classes - 1, n\_features) \rightarrow 2$   
default 값  
 $\min(2, 5)$

```
1 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
2 cld=LinearDiscriminantAnalysis()
3 X_lda = cld.fit_transform(X_train, y_train)
4
5 from matplotlib import pyplot as plt
6 plt.xlabel('LD1')
7 plt.ylabel('LD2')
8 plt.scatter(
9     X_lda[:,0],
10    X_lda[:,1],
11    c=y_train,
12    cmap='rainbow',
13    alpha=0.7,
14    edgecolors='b'
15 )
```



`LinearDiscriminantAnalysis(n_components=4)`



# 이차판별분석(QDA) 실습 #1

- 실습 코드: [링크](#)
- 데이터셋: Iris 데이터 → (150 x 5) 즉, n\_features = 5
- 학습/시험 데이터: x\_train/x\_test
- 학습/시험 데이터 라벨: y\_train/y\_test → (0,1,2) 즉, n\_classes = 3
- 데이터 로더, 범주형 데이터 변환, 데이터 분할
  - 3, 4장 실습과 동일

## 이차판별분석 API: [Manual](#)

- QuadraticDiscriminantAnalysis 클래스 호출

```
1 from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
2
3 cqd=QuadraticDiscriminantAnalysis(store_covariance=True)
4
5 cqd.fit(X_train, y_train) # QDA 적합
6 y_train_pred=cqd.predict(X_train)
7 y_test_pred=cqd.predict(X_test)
8
```

비교  $Y_{train}$  실제값 (GT)  
 $\bar{Y}_{train}$  예측값

$Y_{test}$  : test 실제값  
 $\bar{Y}_{test}$  : 예측값  
정확도 accuracy\_score

# 이차판별분석(QDA) 실습 #1

- 실습 코드: [링크](#)
- 데이터셋: Iris 데이터 → (150 x 5) 즉, n\_features = 5
- 학습/시험 데이터: x\_train/x\_test
- 학습/시험 데이터 라벨: y\_train/y\_test → (0,1,2) 즉, n\_classes = 3
- Accuracy & Confusion Matrix

```
9 from sklearn.metrics import accuracy_score
10 print(accuracy_score(y_train, y_train_pred)) # train data에 대한 accuracy
11 print(accuracy_score(y_test, y_test_pred)) # test data에 대한 accuracy
```

```
0.9809523809523809
0.9777777777777777
```

```
[28] 1 # 분류 결과
      2 from sklearn.metrics import confusion_matrix
      3 print(confusion_matrix(y_test, y_test_pred))
```

```
 / 2 3
0 / [[15  0  0]
1   [ 0 14  1] 2번째 클래스를 3번째 클래스 잘못 판단함
2   [ 0  0 15]]
```