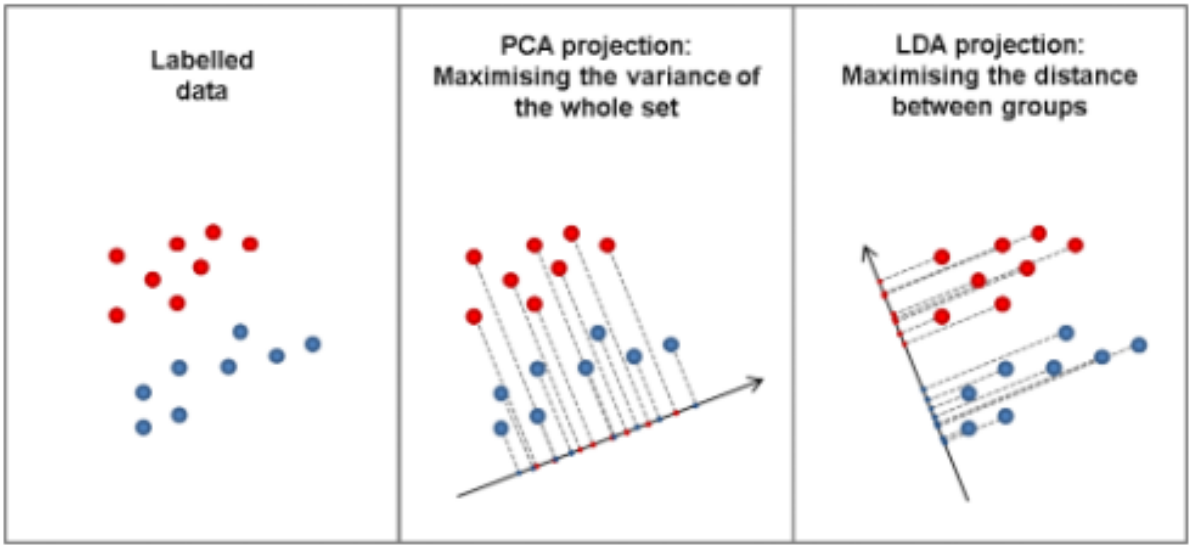
🧡

CH5 Kaggle

캐글의 심장병 예측 노트북을 필사하고 새로운 모델이나 기법에 대해 조사

[확률적 생성 모형]

LDA (LinearDiscriminantAnalysis)



- 선형판별분석
- PCA와 같은 차원 축소 기법
  → PCA는 전체 데이터 분포에 미미한 방향을 찾아서 해당 방향을 제외해 더 낮은 차원에서 분석이 이루어 질 수 있도록 분산이 최대가 되는 방향을 우선순위에 두어 아래 순위 변수는 모형에서 제외하려는 것(모형축소)이 목적
- 지도학습 분류에서 사용 - 범주가 많을 때 로지스틱보다 좋다고 하기도?
- **타겟값 클래스를 최대한 분류할 수 있는 축**을 탐색
  → 클래스 간 분산은 최대한 크게, 클래스 내의 분산은 최대한 작게
  → 클래스 간 평균(클래스 영역의 중심) 차이는 크게
- 기본 가정
  1) 각 집단이 정규분포 형태의 확률 분포를 가짐
  2) 각 집단은 비슷한 형태의 공분산 구조를 가짐

- 장단점
  - 변화 민감 x, 비교적 안정
  - 가정 위반되어도 결과는 비교적 괜찮음
  - 정규분포 크게 벗어난 경우 잘 설명 못 함
  - 공분산 구조 많이 다르다면 잘 설명 못 함 → QDA 등장

▼ 참고

https://nirpyresearch.com/classification-nir-spectra-linear-discriminant-analysis-python/

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline


# 붓꽃 데이터 로드
iris = load_iris()

# 데이터 정규 스케일링
iris_scaled = StandardScaler().fit_transform(iris.data)

# 2개의 클래스로 구분하기 위한 LDA 생성
lda = LinearDiscriminantAnalysis(n_components=2)

# fit()호출 시 target값 입력
lda.fit(iris_scaled, iris.target)
iris_lda = lda.transform(iris_scaled)


lda_columns=['lda_component_1','lda_component_2']
irisDF_lda = pd.DataFrame(iris_lda, columns=lda_columns)
irisDF_lda['target']=iris.target

#setosa는 세모, versicolor는 네모, virginica는 동그라미로 표현
markers=['^', 's', 'o']

#setosa의 target 값은 0, versicolor는 1, virginica는 2. 각 target 별로 다른 shape으로 scatter plot
for i, marker in enumerate(markers):
    x_axis_data = irisDF_lda[irisDF_lda['target']==i]['lda_component_1']
    y_axis_data = irisDF_lda[irisDF_lda['target']==i]['lda_component_2']

    plt.scatter(x_axis_data, y_axis_data, marker=marker,label=iris.target_names[i])

plt.legend(loc='upper right')
plt.xlabel('lda_component_1')
plt.ylabel('lda_component_2')
plt.show()
```

QDA (Quadratic Discriminant analysis)

observation $X = x$ to the class for which is largest.

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\mathbf{\Sigma}_k^{-1}x + x^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log\pi_k\end{aligned} \tag{4.28}$$

- LDA처럼 각 클래스의 관측값이 가우시안 분포에서 가져온 것이라고 가정하고 예측을 수행하기 위해 매개변수에 대한 추정치를 베이즈 정리에 연결한 결과
- BUT 각 클래스에 고유한 공분산 행렬이 있다고 가정
- 설명 변수 개수가 많으면 추정해야 하는 공분산 구조가 많아지기 때문에 샘플 많이 필요
- 다수의 X가 2차 형태로 나타나기 때문에 이차판별분석이라고 함

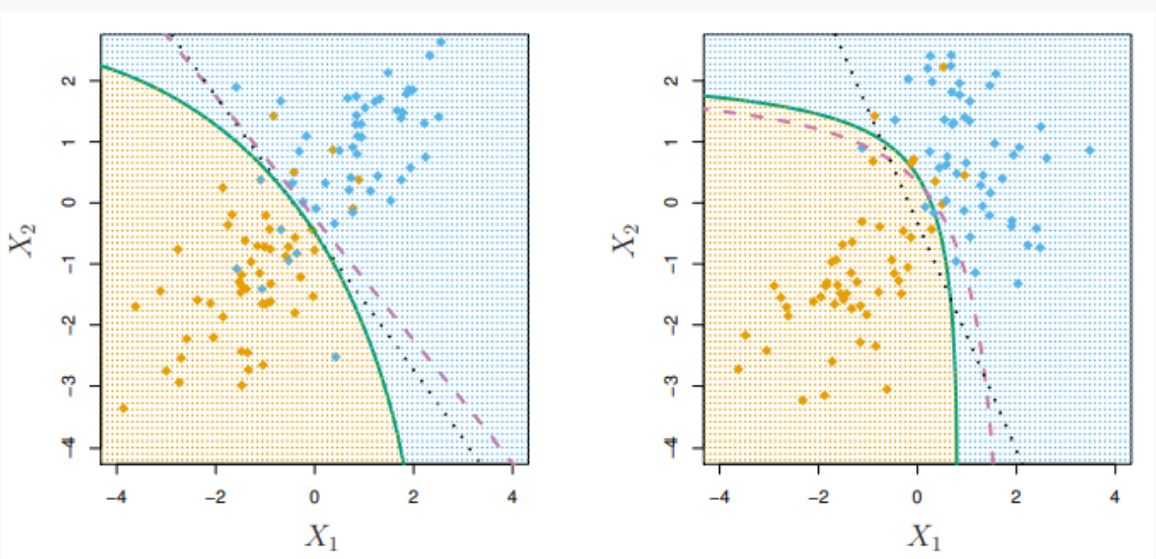


FIGURE 4.9. Left: *The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with* $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$. *The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA.* Right: *Details are as given in the left-hand panel, except that* $\mathbf{\Sigma}_1 \neq \mathbf{\Sigma}_2$. *Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.*

- QDA: 훈련 데이터가 많고, 편향이 줄어드는 것이 중요하다면
  OR 클래스 간 등분산이라는 가정이 명확하지 않다면 우세

↔ LDA : 훈련 데이터가 많고, 분산을 줄이는 것이 중요하다면 우세

```
# 메소드만
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

▼ 참고

https://blog.naver.com/tjdtjdgus99/222188143161

Naive Bayes

Then *Bayes' theorem* states that

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}. \quad (4.15)$$

- 특성의 독립을 가정한 베이즈 정리를 적용한 확률 분류기
- k번째 클래스 내에서 p 예측변수는 모두 **독립이라고 가정**

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p), \quad (4.29)$$

- 따라서 4.29가 성립 → 4.15에 대입하면? 사후확률은 4.30

$$\Pr(Y = k|X = x) = \frac{\pi_k \times f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)}{\sum_{l=1}^{K} \pi_l \times f_{l1}(x_1) \times f_{l2}(x_2) \times \cdots \times f_{lp}(x_p)} \quad (4.30)$$

for $k = 1, \ldots, K$.

To estimate the one-dimensional density function f_{kj} using training data $x_{1j}, \ldots, x_{nj}$, we have a few options.

- Naive Bayes의 가정은 일부 편향을 도입하고 있지만, 분산을 감소시켜 bias-var trade off의 결과로서는 상당히 잘 작동함
- 종류 : 설명 변수가 무엇이냐
  - 연속형 : 정규분포 가정하여 진행
  - 범주형 : 다항분포 이용
  - 이분형 : 이항분포 이용

1. 가우시안

$$= \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left(-\frac{(x_i - \mu_c)^2}{2\sigma_c^2}\right) P(y = c)$$

2. 멀티노미얼

$$P(X_k = x_k | y = c) = \frac{(\sum_{k=1}^{K} x_k)!}{\prod_{k=1}^{K} x_k!} \prod_{k=1}^{K} p_{ck}^{x_k}$$

3. 베르누이(x = 0,1)

$$P(X = x | y = c) = p_c^x (1 - p_c)^{1-x}$$

```
# 메소드만
GaussianNB()
MultinomialNB()
```

▼ 참고

https://blog.naver.com/tjdtjdgus99/222188143161

성능 비교

		True default status		
		No	Yes	Total
Predicted default status	No	9644	252	9896
	Yes	23	81	104
	Total	9667	333	10000

TABLE 4.4. *A confusion matrix compares the LDA predictions to the true default statuses for the* 10,000 *training observations in the* `Default` *data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for* 23 *individuals who did not default and for* 252 *individuals who did default.*

		True default status		
		No	Yes	Total
Predicted default status	No	9320	128	9448
	Yes	347	205	552
	Total	9667	333	10000

TABLE 4.9. *Comparison of the naive Bayes predictions to the true default status for the* $10,000$ *training observations in the* `Default` *data set, when we predict default for any observation for which* $P(Y = \texttt{default}|X = x) > 0.2$.

- LDA가 네이브 베이즈보다 정확도가 높지만, 네이브 베이즈가 참을 참으로 예측하는 능력은 좋음
- 변수 개수가 많거나 데이터 수가 작은 경우 네이브 베이즈 성능이 나머지 방법보다 좋을 것