로지스틱 회귀

우도경

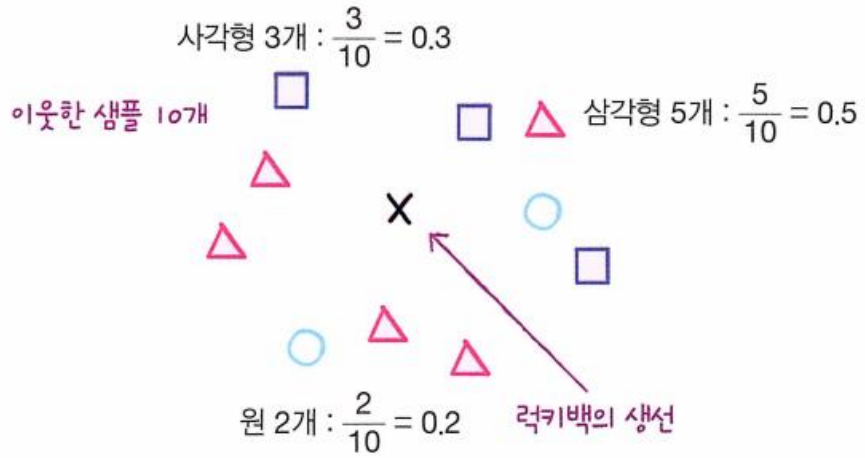
2025-07-16

INDEX

- 기존 모형의 한계
- 로지스틱 회귀
- 마무리

1. 기존 모형의 한계

- 기존 KNN 모델로 생선이 어떤 클래스에 속할지 확률을 계산할 수 있을까?



- 샘플 X 주위에 가장 가까운 이웃 샘플 10개를 선택
- 이웃의 클래스 비율을 확률로 출력

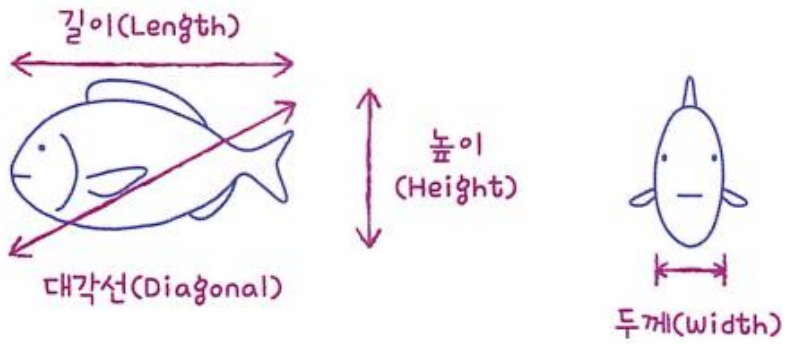
1. 기존 모형의 한계

- 데이터 준비

```
[ ]  import pandas as pd

     fish = pd.read_csv('https://bit.ly/fish_csv_data')
     fish.head()
```

	Species	Weight	Length	Diagonal	Height	Width
0	Bream	242.0	25.4	30.0	11.5200	4.0200
1	Bream	290.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	26.5	31.1	12.3778	4.6961
3	Bream	363.0	29.0	33.5	12.7300	4.4555
4	Bream	430.0	29.0	34.0	12.4440	5.1340



1. 기존 모형의 한계

- 데이터 준비

```
[ ]  print(pd.unique(fish['Species']))
```

```
['Bream' 'Roach' 'Whitefish' 'Parkki' 'Perch' 'Pike' 'Smelt']
```

```
[ ]  fish_input = fish[['Weight','Length','Diagonal','Height','Width']].to_numpy()
```

```
[ ]  print(fish_input[:5])
```

```
[[242.      25.4     30.      11.52     4.02  ]
 [290.      26.3     31.2     12.48     4.3056]
 [340.      26.5     31.1     12.3778   4.6961]
 [363.      29.      33.5     12.73     4.4555]
 [430.      29.      34.      12.444    5.134 ]]
```

```
[ ]  fish_target = fish['Species'].to_numpy()
```

- 타깃 : Species
- 입력 데이터 : Weight, Length, Diagonal, Height, Width

1. 기존 모형의 한계

- 데이터 준비

```
[ ]  from sklearn.model_selection import train_test_split

     train_input, test_input, train_target, test_target = train_test_split(
         fish_input, fish_target, random_state=42)
```

```
[ ]  from sklearn.preprocessing import StandardScaler

     ss = StandardScaler()
     ss.fit(train_input)
     train_scaled = ss.transform(train_input)
     test_scaled = ss.transform(test_input)
```

- 데이터를 훈련 세트와 테스트 세트로 나눔
- 훈련 세트와 테스트 세트를 표준화 전처리
- 훈련 세트의 통계 값으로 테스트 세트 변환

1. 기존 모형의 한계

- KNN의 확률 예측

```
[ ]  from sklearn.neighbors import KNeighborsClassifier

     kn = KNeighborsClassifier(n_neighbors=3)
     kn.fit(train_scaled, train_target)

     print(kn.score(train_scaled, train_target))
     print(kn.score(test_scaled, test_target))
```

```
0.8907563025210085
0.85
```

```
[ ]  print(kn.classes_)
```

```
['Bream' 'Parkki' 'Perch' 'Pike' 'Roach' 'Smelt' 'Whitefish']
```

```
[ ]  print(kn.predict(test_scaled[:5]))
```

```
['Perch' 'Smelt' 'Pike' 'Perch' 'Perch']
```

```
[ ]  import numpy as np

     proba = kn.predict_proba(test_scaled[:5])
     print(np.round(proba, decimals=4))
```

```
[[0.     0.     1.     0.     0.     0.     0.    ]
 [0.     0.     0.     0.     0.     1.     0.    ]
 [0.     0.     0.     1.     0.     0.     0.    ]
 [0.     0.     0.6667 0.     0.3333 0.     0.    ]
 [0.     0.     0.6667 0.     0.3333 0.     0.    ]]
```

```
[ ]  distances, indexes = kn.kneighbors(test_scaled[3:4])
     print(train_target[indexes])
```

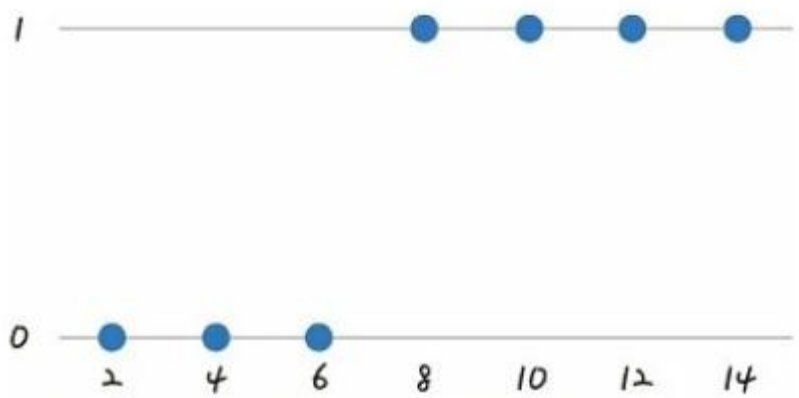
```
[['Roach' 'Perch' 'Perch']]
```

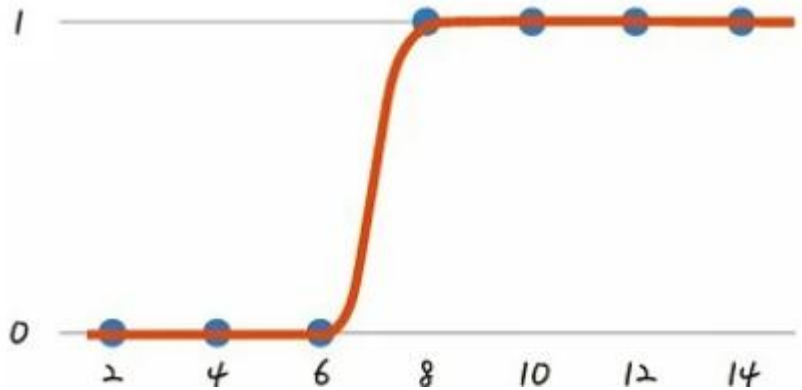
- 가능한 확률은 0/3, 1/3, 2/3, 3/3 이 4가지 값
- 확률을 출력하는 다른 방법이 필요

2. 로지스틱 회귀

- 로지스틱 회귀

공부한 시간	2	4	6	8	10	12	14
합격 여부	불합격	불합격	불합격	합격	합격	합격	합격





- 로지스틱 회귀 또한 선형 회귀와 마찬가지로 점들의 특징을 가장 잘 나타내는 선을 그리는 과정
- 직선으로는 이 점의 특성을 잘 나타내는 일차방정식을 만들기 어려움
- 따라서, 참(1)과 거짓(0) 사이를 구분하는 S자 형태의 선을 그어주는 작업

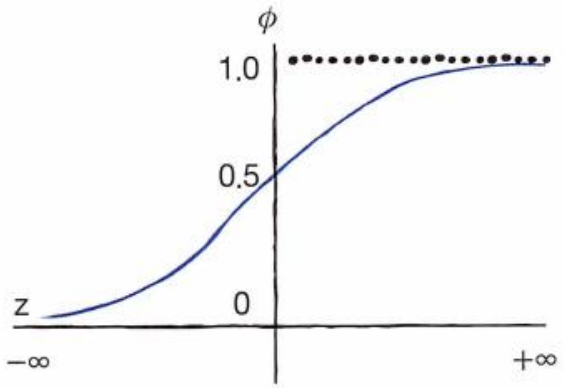
2. 로지스틱 회귀

- 시그모이드 함수

$$z = a \times (Weight) + b \times (Length) + c \times (Diagonal) + d \times (Height) + e \times (Width) + f$$

$$\phi = \frac{1}{1+e^{-z}}$$

시그모이드 함수



시그모이드 그래프

- z가 무한하게 큰 음수일 경우 함숫값은 0에 가까워지고, z가 무한하게 큰 양수일 때는 1에 가까워짐
- z가 어떤 값이 되더라도 ø는 0~1사이의 범위를 가짐
- 따라서, 0~1 사이의 값을 0~100%까지 확률로 해석 가능

2. 로지스틱 회귀

- 로지스틱 회귀로 이진 분류 수행하기

```
[ ]  char_arr = np.array(['A', 'B', 'C', 'D', 'E'])
     print(char_arr[[True, False, True, False, False]])
```

```
['A' 'C']
```

```
[ ]  bream_smelt_indexes = (train_target == 'Bream') | (train_target == 'Smelt')
     train_bream_smelt = train_scaled[bream_smelt_indexes]
     target_bream_smelt = train_target[bream_smelt_indexes]
```

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(train_bream_smelt, target_bream_smelt)
```

```
[ ]  print(lr.predict(train_bream_smelt[:5]))
```

```
['Bream' 'Smelt' 'Bream' 'Bream' 'Bream']
```

```
[ ]  print(lr.predict_proba(train_bream_smelt[:5]))
```

```
[[0.99760007 0.00239993]
 [0.02737325 0.97262675]
 [0.99486386 0.00513614]
 [0.98585047 0.01414953]
 [0.99767419 0.00232581]]
```

```
[ ]  print(lr.classes_)
```

```
['Bream' 'Smelt']
```

- 훈련 세트에서 도미(Bream)와 빙어(Smelt)의 행만 추출
- 모델 훈련 및 예측

2. 로지스틱 회귀

- 로지스틱 회귀로 이진 분류 수행하기

```
[ ]  print(lr.coef_, lr.intercept_)
```

```
[[-0.40451732 -0.57582787 -0.66248158 -1.01329614 -0.73123131]] [-2.16172774]
```

```
[ ]  decisions = lr.decision_function(train_bream_smelt[:5])
     print(decisions)
```

```
[-6.02991358  3.57043428 -5.26630496 -4.24382314 -6.06135688]
```

```
[ ]  from scipy.special import expit

     print(expit(decisions))
```

```
[0.00239993 0.97262675 0.00513614 0.01414953 0.00232581]
```

- 로지스틱 회귀모델이 학습한 방정식

$$z = -0.404 \times (Weight) - 0.576 \times (Length) - 0.663 \times (Diagonal) - 1.013 \times (Height) - 0.732 \times (Width) - 2.161$$

2. 로지스틱 회귀

- 로지스틱 회귀로 다중 분류 수행하기

```
[ ]  lr = LogisticRegression(C=20, max_iter=1000)
     lr.fit(train_scaled, train_target)

     print(lr.score(train_scaled, train_target))
     print(lr.score(test_scaled, test_target))
```

```
0.9327731092436975
0.925
```

```
[ ]  print(lr.predict(test_scaled[:5]))
```

```
['Perch' 'Smelt' 'Pike' 'Roach' 'Perch']
```

```
[ ]  proba = lr.predict_proba(test_scaled[:5])
     print(np.round(proba, decimals=3))
```

```
[[0.    0.014 0.842 0.    0.135 0.007 0.003]
 [0.    0.003 0.044 0.    0.007 0.946 0.   ]
 [0.    0.    0.034 0.934 0.015 0.016 0.   ]
 [0.011 0.034 0.305 0.006 0.567 0.    0.076]
 [0.    0.    0.904 0.002 0.089 0.002 0.001]]
```

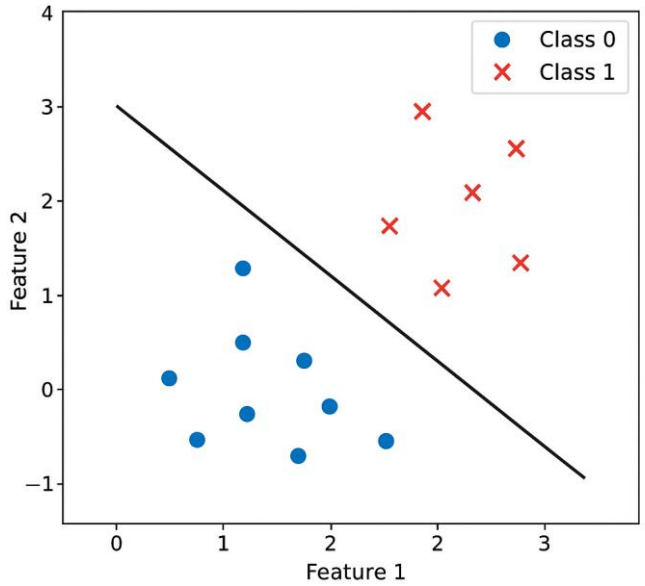
```
[ ]  print(lr.classes_)
```

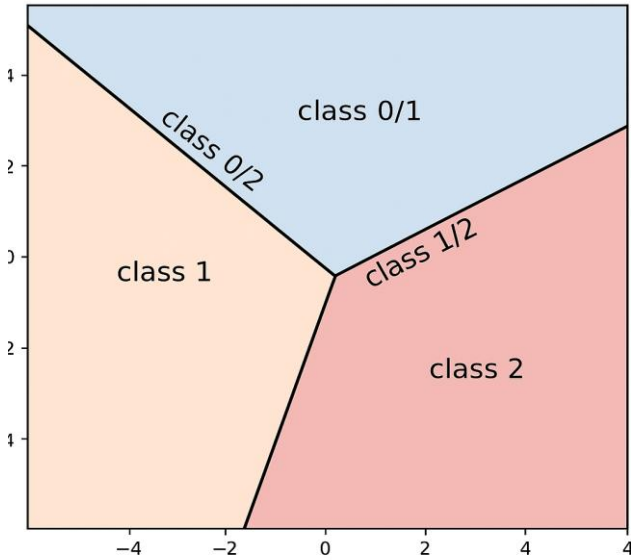
```
['Bream' 'Parkki' 'Perch' 'Pike' 'Roach' 'Smelt' 'Whitefish']
```

- 로지스틱 회귀는 기본적으로 릿지 회귀와 같이 계수의 제곱을 규제
- 규제를 제어하는 매개변수는 C이며, 값이 작을수록 규제가 커짐. (기본값 : 1)
- 7개 클래스에 대한 확률을 계산하므로 7개의 열이 출력

2. 로지스틱 회귀

- 이진 vs 다중 분류의 결정 경계





- 이진분류는 이 샘플이 클래스 1인지, 0인지에 대한 하나의 질문 → 하나의 결정 경계를 기준으로 왼쪽은 클래스 0, 오른쪽은 클래스 1로 분류
- 다중 분류는 각각의 클래스에 대해 "이 클래스인지, 아닌지"를 판별하는 하나의 경계선이 필요. 결과적으로 클래스 개수만큼의 경계선이 만들어짐

2. 로지스틱 회귀

- 로지스틱 회귀로 다중 분류 수행하기

$$z = a \times (Weight) + b \times (Length) + c \times (Diagonal) + d \times (Height) + e \times (Width) + f$$

$$e\_sum = e^{z1} + e^{z2} + e^{z3} + e^{z4} + e^{z5} + e^{z6} + e^{z7}$$

$$s1 = \frac{e^{z1}}{e\_sum},\ s2 = \frac{e^{z2}}{e\_sum},\ \cdots,\ s7 = \frac{e^{z7}}{e\_sum}$$

- 클래스마다 z값을 하나씩 계산
- $z^1 \sim z^7$까지의 값을 사용해 지수함수 $e^{z^1} \sim e^{z^7}$계산해서 모두 더한 값을 e_sum이라고 함
- $e^{z^1} \sim e^{z^7}$ 을 e_sum으로 나눈 값이 각 클래스에 대한 확률 값

3. 마무리

- 로지스틱 회귀로 확률 예측
  - 분류 모델은 예측 뿐만 아니라 예측의 근거가 되는 확률을 출력할 수 있음
  - 로지스틱 회귀는 선형 방정식을 사용해서 계산한 값을 0~1 사이의 값으로 변환해서 출력함.
  - 다중 분류일 경우에는 클래스 개수만큼 방정식이 생성되고, 각 방정식의 출력 값을 각 클래스에 대한 확률로 이해할 수 있음

Thank You

참고문헌

- 박해선. 혼자 공부하는 머신러닝 + 딥러닝. 한빛미디어, 2025.
- 조태호. 모두의 딥러닝 - 누구나 쉽게 이해하는 딥러닝 개정3판. 길벗, 2022.