Ch4. 다양한 분류 알고리즘

동아대 IT 취업동아리 – 머신러닝 스터디

경영정보학과 김예나

럭키백

길이, 높이, 두께, 대각선, 무게

=> 7개 생선에 대한 확률 출력

다중분류 - k-최근접 이웃 분류기

```
from sklearn.neighbors import KNeighborsClassifier
kn = KNeighborsClassifier(n_neighbors = 3)
kn.fit(train_scaled, train_target)
```

classes_ : 타깃값

```
print(kn.classes_)
```

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

-> 알파벳순 !

predict() : 예측 출력

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

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

predict_proba() : 클래스별 확률값 반환

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

<- kn.classes_

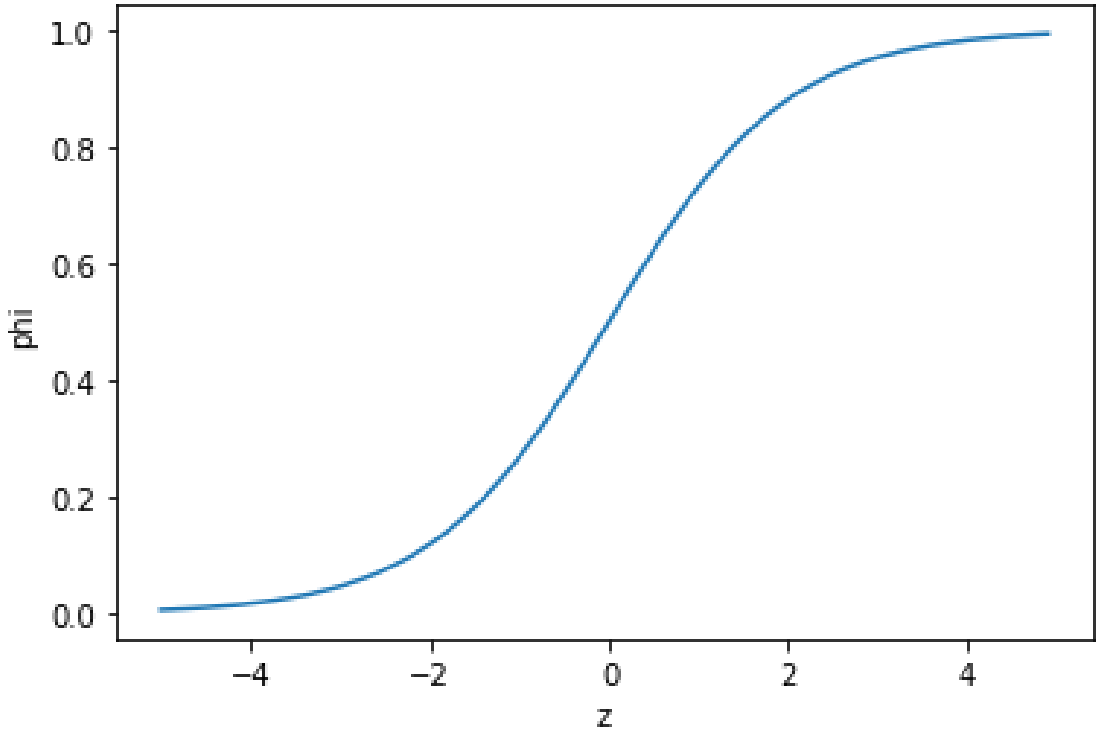
로지스틱 회귀(logistic regression)

- 분류 모델
- 선형방정식 학습

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

시그모이드 함수(sigmoid function)

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



Ø > 0.5 => 양성 클래스 Ø < 0.5 => 음성 클래스

이진 분류 - 로지스틱 분류

불리언 인덱싱 - 도미, 빙어만

```
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.classes_)
```

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

이진 분류 - 로지스틱 분류

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

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

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

```
[[0.99759855 0.00240145]
 [0.02735183 0.97264817]
 [0.99486072 0.00513928]
 [0.98584202 0.01415798]
 [0.99767269 0.00232731]]
```

이진 분류 - 로지스틱 분류

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

```
[[-0.4037798  -0.57620209 -0.66280298 -1.01290277 -0.73168947]] [-2.16155132]
```

desicion_function() : z값 출력

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

```
[-6.02927744  3.57123907 -5.26568906 -4.24321775 -6.0607117 ]
```

expit() : 시그모이드 함수

```
from scipy.special import expit
print(expit(decision))
```

```
[0.00240145 0.97264817 0.00513928 0.01415798 0.00232731]
```

다중 분류 - 로지스틱 분류

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

```
print(lr.classes_)
```

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

다중 분류 - 로지스틱 분류

```
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 0.841 0.     0.136 0.007 0.003]
 [0.     0.003 0.044 0.     0.007 0.946 0.    ]
 [0.     0.    0.034 0.935 0.015 0.016 0.    ]
 [0.011 0.034 0.306 0.007 0.567 0.    0.076]
 [0.     0.    0.904 0.002 0.089 0.002 0.001]]
```

다중 분류 - 로지스틱 분류

```
print(lr.coef_.shape, lr.intercept_.shape)
```

```
(7, 5) (7,)
```

```
decision = lr.decision_function(test_scaled[:5])
print(np.round(decision, decimals = 2))
```

```
[[ -6.5    1.03   5.16  -2.73   3.34   0.33  -0.63]
 [-10.86   1.93   4.77  -2.4    2.98   7.84  -4.26]
 [ -4.34  -6.23   3.17   6.49   2.36   2.42  -3.87]
 [ -0.68   0.45   2.65  -1.19   3.26  -5.75   1.26]
 [ -6.4   -1.99   5.82  -0.11   3.5   -0.11  -0.71]]
```

다중 분류 - 로지스틱 분류

소프트맥스 함수

```
from scipy.special import softmax
proba = softmax(decision, axis = 1)
print(np.round(proba, decimals = 3))
```

```
[[0.     0.014 0.841 0.     0.136 0.007 0.003]
 [0.     0.003 0.044 0.     0.007 0.946 0.    ]
 [0.     0.    0.034 0.935 0.015 0.016 0.    ]
 [0.011 0.034 0.306 0.007 0.567 0.    0.076]
 [0.     0.    0.904 0.002 0.089 0.002 0.001]]
```

감사합니다