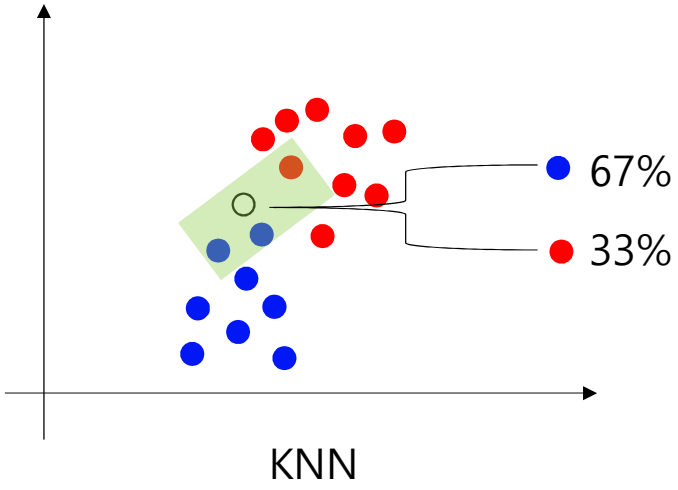
로지스틱회귀-이진분류

로지스틱(Logistic)회귀

- KNN 분류와 비슷하게 확률 기반으로 데이터 분류
- 선형(Linear)회귀와 비슷하게 선형 방정식을 사용하여 데이터를 분류
- 여러 특징을 사용할 경우 다중(Multiple)회귀와 비슷
- 데이터가 어떤 클래스에 속할 확률을 0과 1사이의 수로 계산하여, 높은 확률의 클래스로 분류

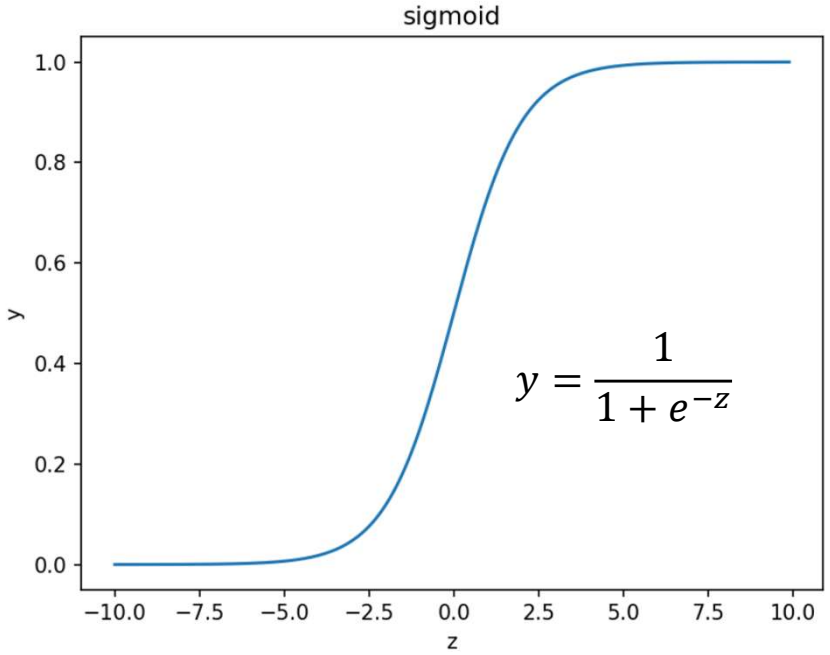


선형회귀 $z = ax + b$

다중회귀 $z = ax_1 + bx_2 + c$

로지스틱(Logistic)함수

- 확률을 0과 1시이의 수로 변환하기 위하여 시그모이드(Sigmoid)함수 사용
- 선형 방정식의 결과값이 큰 음수일 경우 0으로 근접하고, 큰 양수일 경우 1로 근접



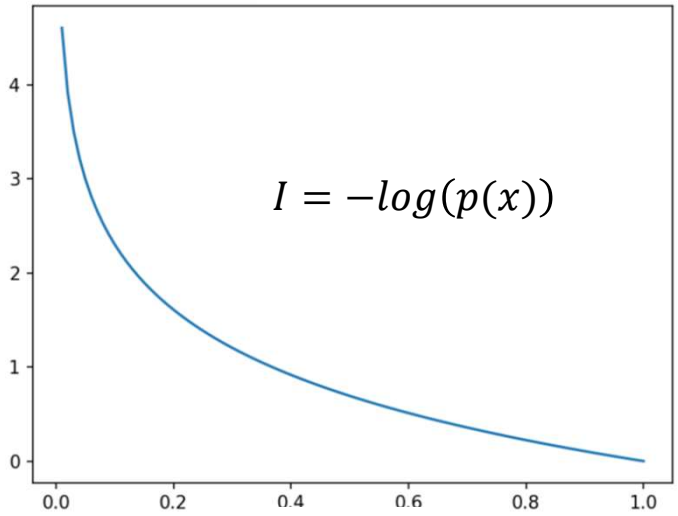

```
import numpy as np
import matplotlib.pyplot as plt
z = np.arange(-10, 10, 0.1)
y = 1 / (1 + np.exp(-z))
plt.plot(z, y)
plt.title("sigmoid"); plt.xlabel("z"); plt.ylabel("y")
plt.show()
```

엔트로피(Entropy)

- 불확실성, 무질서도, 새로운 정보량, 놀라움 등의 정도를 나타내는 값
- 발생확률(p)이 낮은 사건일수록 정보량이 높음
- 엔트로피가 클수록, 데이터가 불확실하고 무질서하여 데이터가 잘 분리되어 있지 않은 상태
- 엔트로피가 작을수록, 데이터가 잘 분리되어 있는 상태
- 엔트로피는 정보량의 기댓값, 평균 정보량으로 계산

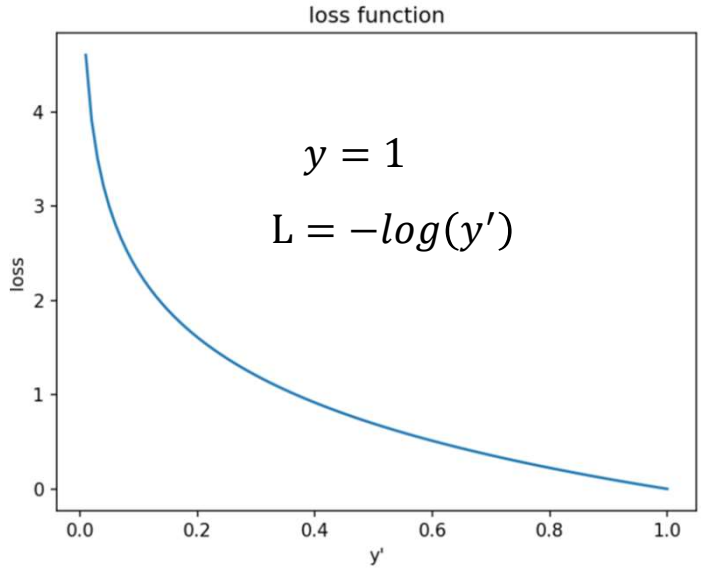
$$I = p(x)(-log(p(x))$$

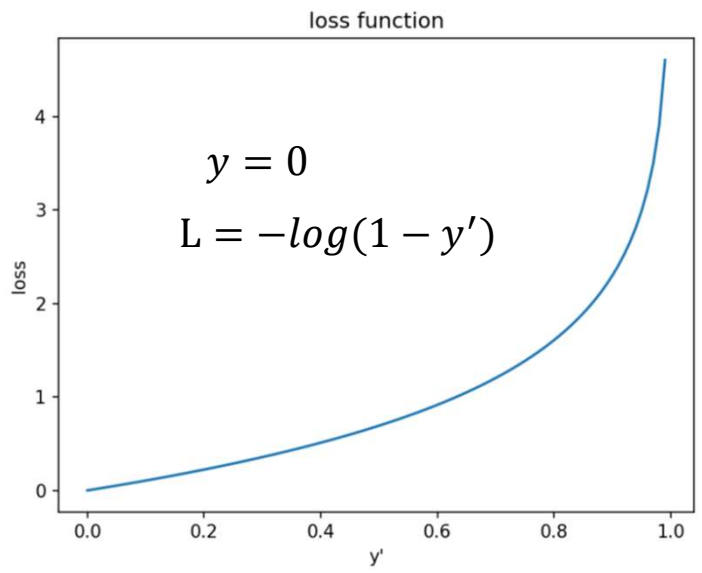
실제확률 정보량



교차-엔트로피(Cross-Entropy) 손실함수

- 로지스틱회귀 모델에서 사용하는 손실함수를 위해 크로스 엔트로피 공식 사용
- 엔트로피에서의 실제(정답) 확률값만 사용하는 대신에, 크로스 엔트로피에서는 예측 확률값 사용
- 정보량 = 실제값과 예측값이 얼마나 근사한지의 정도





$$\mathrm{L} = -ylog(y') - (1 - y)log(1 - y')$$

sklearn.linear_model.LogisticRegression

class sklearn.linear_model.LogisticRegression(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

penalty : *{'l1', 'l2', 'elasticnet', 'none'}, default='l2'*

Specify the norm of the penalty:

- `'none'`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

max_iter : *int, default=100*

Maximum number of iterations taken for the solvers to converge.

C : *float, default=1.0*

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

어린이와 청소년 구분

```python
import pandas as pd
data3 = pd.read_csv("data/health.csv")
data = data3[["H", "W"]]
t = data3["T"]

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)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(train_data, train_target)
print("Train-Eval:", lr.score(train_data, train_target))
print("Test-Eval :", lr.score(test_data, test_target))
print(lr.coef_, lr.intercept_)
```

```
Train-Eval: 1.0
Test-Eval : 1.0
[[-0.2669508  -0.21960901]] [46.11595006]
```

$$z = -0.27x_1 - 0.22x_2 + 46.12$$

학습

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

테스트

$y' \geq 0.5, y' = 1$

$y' < 0.5, y' = 0$

$$\mathrm{L} = -ylog(y') - (1-y)log(1-y')$$

결과 검증

	H	W
5	152.8	49.9
8	156.6	62.2
4	126.0	25.8

5	0
8	0
4	1

테스트 데이터

$$z = -0.27x_1 - 0.22x_2 + 46.12$$

```
print(lr.coef_[0][0] * test_data["H"] + lr.coef_[0][1] * test_data["W"] + lr.intercept_[0])
z = lr.decision_function(test_data)
print(z)
```

5	-5.632622
8	-9.348226
4	6.814236

```
[-5.63262229 -9.34822618  6.8142364 ]
```

```
import numpy as np
print(np.round(lr.predict_proba(test_data), 3))
print(lr.predict(test_data))
```

$y' \geq 0.5, y' = 1$

$y' < 0.5, y' = 0$

```
[[0.996 0.004]
 [1.    0.   ]
 [0.001 0.999]]
[0 0 1]
```

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

```
y = np.round(1 / (1 + pow(np.e, -z)), 3)
print(1 - y, y)
```

```
[0.996 1.    0.001] [0.004 0.    0.999]
```

데이터(Iris) 실수화, 분할

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

idx = iris[iris["species"] == "virginica"].index
iris.drop(idx, inplace = True)
```

구현

https://en.wikipedia.org/wiki/Iris_flower_data_set

Setosa

Versicolor

붓꽃 품종

로지스틱회귀 정확도

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(train_data, train_target)
print("Train-Eval:", lr.score(train_data, train_target))
print("Test-Eval :", lr.score(test_data, test_target))
print(lr.coef_, lr.intercept_)

from sklearn.metrics import confusion_matrix
conf = confusion_matrix(test_target, lr.predict(test_data))
print(conf)
```

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

```
(30, 4) (30,)
(1, 4) (1,)
```

```
Train-Eval: 1.0
Test-Eval : 1.0
[[ 0.52354909 -0.78468851  2.11753181  0.90085533]] [-6.89382217]
[[15  0]
 [ 0 15]]
```

$$z = ax_1 + bx_2 + cx_3 + dx_4 + e$$

```
import numpy as np
z = lr.decision_function(test_data)
print(z)
y = np.round(1 / (1 + pow(np.e, -z)), 3)
print(1 - y)
print(y)
```

```
[-4.01130926 -3.09392238  4.45945356 -2.76296213 -4.16824696  3.82215345
  4.59328492  4.54979314  4.35308426 -2.95462285  4.82606495  1.80785004
 -3.99943853 -3.69459221  5.2378276   5.19776344 -3.02134799 -3.61353576
  4.78587376 -4.06587069 -3.72083318 -4.27478222  4.56416338  3.32564641
 -4.01118224 -5.01267209  5.20883309  1.67702628  6.37900435 -3.58754885]
[0.982 0.957 0.011 0.941 0.985 0.021 0.01  0.01  0.013 0.95  0.008 0.141
 0.982 0.976 0.005 0.005 0.954 0.974 0.008 0.983 0.976 0.986 0.01  0.035
 0.982 0.993 0.005 0.157 0.002 0.973]
[0.018 0.043 0.989 0.059 0.015 0.979 0.99  0.99  0.987 0.05  0.992 0.859
 0.018 0.024 0.995 0.995 0.046 0.026 0.992 0.017 0.024 0.014 0.99  0.965
 0.018 0.007 0.995 0.843 0.998 0.027]
```

수렴(Convergence)

- 모델이 충분이 학습되지 않으면 성능이 수렴되지 않음 (수렴되면 성능변화 거의 없음)
- 과대적합과 과소적합을 회피하기 위하여 절절한 학습과정이 필요함

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1)
lr.fit(train_data, train_target)
print("Train-Eval:", lr.score(train_data, train_target))
print("Test-Eval :", lr.score(test_data, test_target))
```

```
C:\Users\user\Desktop\PythonTest\venv\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Train-Eval: 0.5
Test-Eval : 0.5
```

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/