하루 한시간 교재 읽기 + 하루 한시간 파이썬 프로

그램

인생이 달라집니다.

인공지능 수학

인공지능 전공

안정호 교수

인공지능 수학 기말고사 안내

시간: 2023. 6.13 (화) 15:20 – 16:50

범위: 교재 6, 7장 수업시간에 다룬 내용

힌트: 실습문제와 유사한 문제, 약 90% 출제

답안: 손 계산 문제는 나눠준 종이에 답안 작성하여 대면 제출,

프로그램 문제는 ipynb 파일에 작성하여 이러닝캠퍼스에 제출

유의사항:

-16:50이후 제출 시 1분당 1점 감점, 17:00이후 제출 불가

-Open Book Test, 책과 종이 한 장 지참 가능

-파일 지참/참고 불가. 인터넷 검색 불가. 개인 컴퓨터 사용 가능

-부정행위시, F 학점

언제든,

경천관 404호로 와서 많이 질문하세요~

시그모이드 Sigmoid 함수

(1) 함수 식?

$$y = \frac{1}{1+\exp(-x)}$$

$$0 \le y = \frac{1}{1+\exp(-x)} \le 1$$

(2) 미분?

$$u = 1 + \exp(-x)$$

$$\begin{aligned}
\frac{dy}{dx} &= \frac{dy}{du}\frac{du}{dx} \\
&= \frac{d}{du}(u^{-1})\frac{d}{dx}(1+\exp(-x)) \\
&= (-u^{-2})(-\exp(-x)) \\
&= \frac{\exp(-x)}{(1+\exp(-x))^2} \\
&= \left(\frac{\exp(-x)}{1+\exp(-x)}\right)\left(\frac{1}{1+\exp(-x)}\right) \\
&= \left(\frac{1+\exp(-x)}{1+\exp(-x)} - \frac{1}{1+\exp(-x)}\right)\left(\frac{1}{1+\exp(-x)}\right) \\
&= (1-y)y
\end{aligned}$$

교차 엔트로피 Cross Entropy

교차 엔트로피는 예측 확률 분포와 실제 확률 분포가 얼마만큼 떨어져 있는지 나타내는 척도임.

확률분포 P에 대한 확률분포 Q의 교차 엔트로피는 다음과 같이 정의한다.

$$H(P,Q) = \sum_X P(X) \log \frac{1}{Q(X)} = -\sum_X P(X) \log Q(X)$$

교차 엔트로피의 log는 자연로그

여기서, $P(X)$는 원래 분포(실제 확률분포), $Q(X)$는 새로운 분포(예측 확률분포)

크로스 엔트로피 H(P, Q)는 P=Q일 때 최소가 됨.

따라서, 크로스 엔트로피는 두 확률분포가 얼마만큼 다른 지 알려줌.

분포의 오차(Error 또는 차이 difference)를 나타내는 양으로 사용됨.

크로스 엔트로피 예제

다음 표를 참고하여, 확률변수 X에 대한 확률변수 Y의 크로스 엔트로피를 구하시오.

x	0	1	2023
$P(X = x)$	0.1	0.5	0.4
$P(Y = x)$	0.5	0.25	0.25

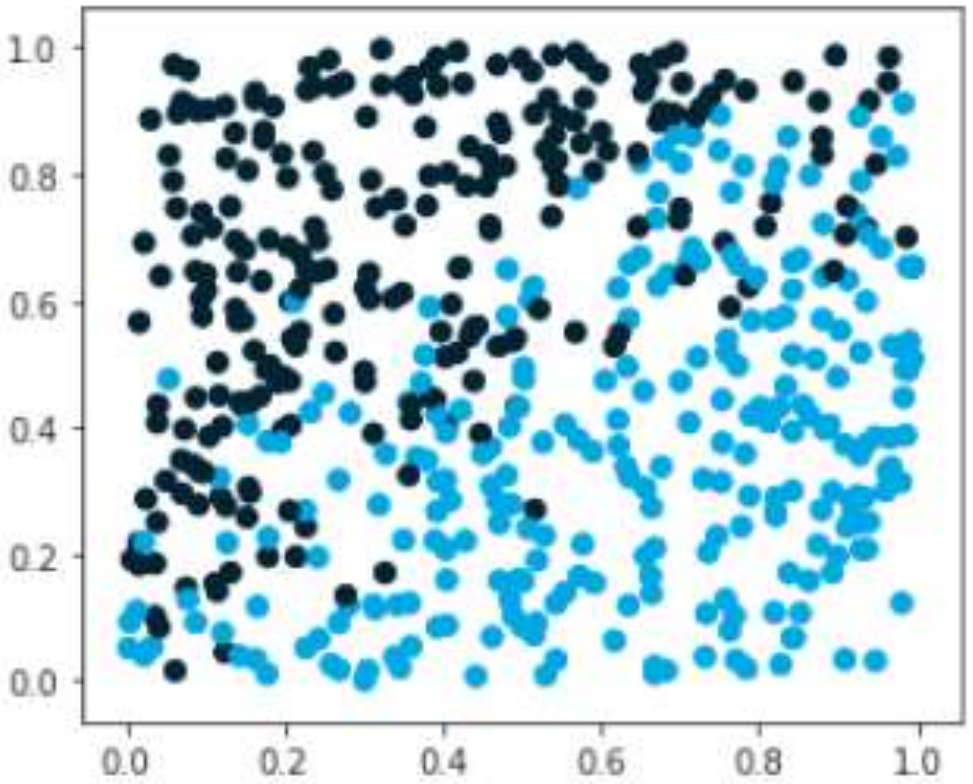
수학을 기계학습에 응용

회귀와 과학습, 분류와 로지스틱회귀, 뉴럴 네트워크의 개요, 학습 메커니즘, 단일 뉴런에 의한 학습의 구현, 딥러닝으로

다음 그림에서, 검은 점과 파란 점을 나누는 경계를

(1) 직선으로 그려보세요.

(2) 곡선으로 그려보세요.



분류 Classification 문제

데이터 분류

사진 분류

특징 분류

sepal_length	sepal_width	petal_length	petal_width	species
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa

<분류 관련 용어>

- Class와 Label
- 이진 분류와 다중 분류

로지스틱 회귀 logistic regression

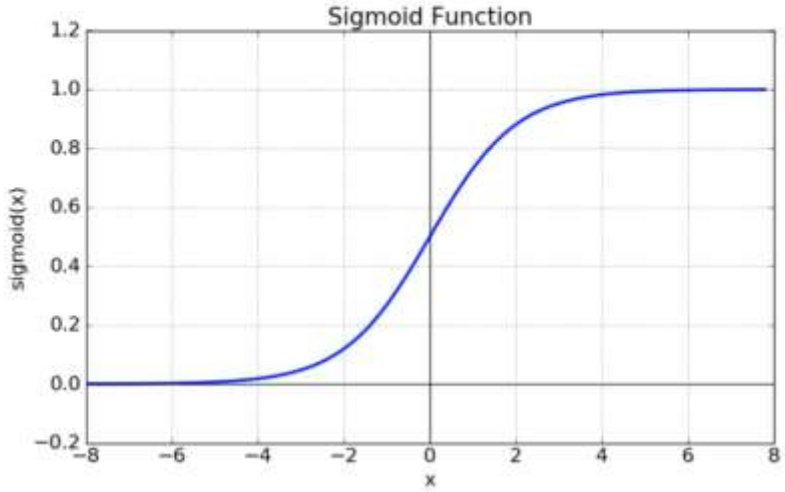
로지스틱 회귀는 대표적인 분류 모델 중 하나. 회귀라는 단어가 이름에 있어 회귀 모델이 아닌가

오해하는데, 이는 분류 모델이다. ^^

로지스틱 회귀는 기본적으로 이진 분류를 다룬다.

Binary classification, two class problem

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



특징: Sigmoid 함수를 이용한다!

Sigmoid 함수 값을 label이 1일 확률로 해석한다.

우리는,

1차원 데이터를 로지스틱 회귀 모델을 적용하여 (이진) 분류하는 예와

2차원 데이터를 로지스틱 회귀 모델로 (이진) 분류하는 예를 살펴 보자.

로지스틱 회귀를 이용한 1차원 데이터 분류

학습 데이터

다음과 같은 1차원 데이터가 있다.

이를 로지스틱 회귀 모델로 학습하고, 테스트를 위해 주어진 데이터를 분류하라.

테스트 데이터

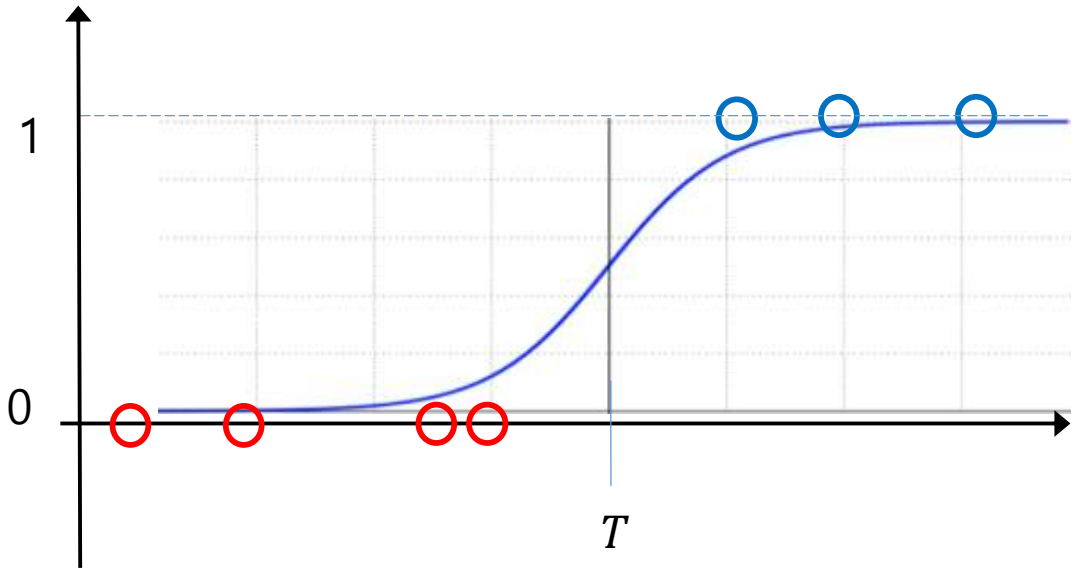
x	2	4	5	8	10	11
y	0	0	0	1	1	1

*데이터 설명: 여기서 x는 공부한 시간이며, y 는 시험의 합격 여부이다. 0은 불합격, 1은 합격을 의미한다.

시그모이드 함수를 데이터에 적합하여 분류!

x	t
x_1	t_1
x_2	t_2
⋮	⋮
x_m	t_m

x_i은 실수, t_i들은 0 또는 1 중 하나.



적합시키는 방법은?

Sigmoid 함수를 옆으로 이동시키고 늘리거나 압축시켜서 데이터에 맞춘다!

적합한 다음에? 뭘 어떻게 하자는 건데?

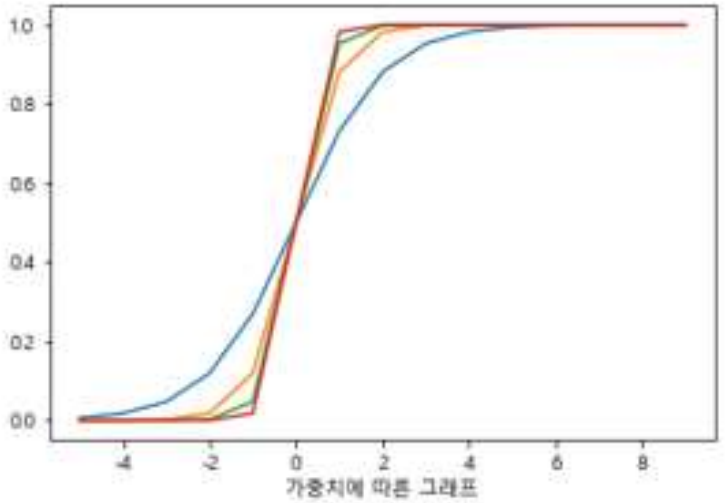
적합 후, 결정 규칙을 만들자!

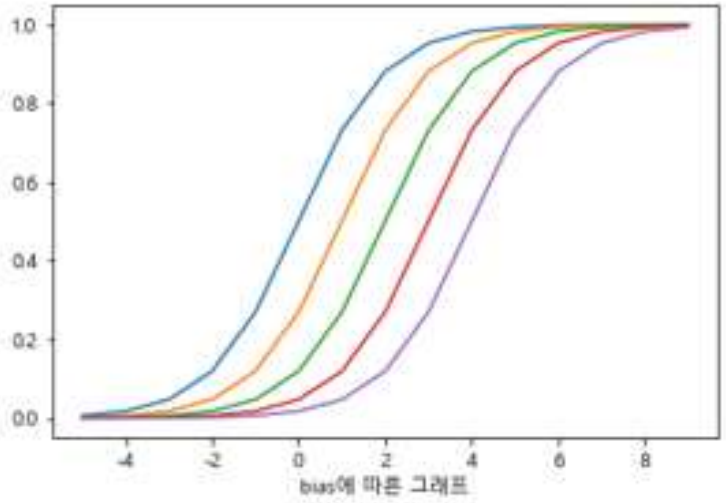
비장의 무기

시그모이드 함수의 파라미터 a, b

이 a, b를 데이터로 학습한다.
학습데이터는 이 a, b를 결정하기 위해서만 사용된다.

$$y = \frac{1}{1 + e^{-(ax+b)}}$$





결정경계 decision boundary, 결정 규칙 decision rule

데이터를 적합시켜서 적절한 파라미터 a, b 를 찾았다고 하자. $f(x) = \frac{1}{1+e^{-(ax+b)}}$

그 다음은?

결정 규칙을 만들자. 이 규칙에 따라 분류하는 것이다. 드디어 분류하는 구나!

어떻게? $f(x) = 0.5$를 만족하는 x를 x_0라 할 때, x_0를 기준으로 좌/우로 분리(분류).

x_0는 무얼까? $ax + b = 0$인 지점! $x_0 = -b/a$.

결정경계: $x = -b/a$

결정규칙: $x < -b/a$

$$d(x) = \begin{cases} 0 & if\ f(x) < 0.5 \\ 1 & o.w. \end{cases}$$

* 중요

$0 \le f(x) \le 1$ 이므로 $f(x)$를 확률로 해석한다. (엄밀하게는 확률은 아니다)

$f(x) = P(x = 1)$

로지스틱 회귀를 이용한 1차원 데이터 분류

손실함수 E에 대해, 최급강하법에 의해 a, b는 다음과 같이 구할 수 있다.

$$a \leftarrow a - \eta \frac{\partial E}{\partial a}$$

$$b \leftarrow b - \eta \frac{\partial E}{\partial b}$$

손실함수 E는 다음과 같이 정의한다.

$$E = -\sum_{j=1}^{m} \left(t_j \log y_j + (1 - t_j) \log (1 - y_j) \right)$$

x	t
x_1	t_1
x_2	t_2
⋮	⋮
x_m	t_m

실험 전에 또는 실험하면서, a, b 초기화, 학습률 계수 η, epoch 수를 어떻게 할지 결정한다.

$$H(P,Q) = \sum_X P(X) \log \frac{1}{Q(X)} = -\sum_X P(X) \log Q(X)$$

손실함수

손실함수 E는 교차 엔트로피를 이용하여 다음과 같이 정의한다.

$$E = -\sum_{j=1}^{m} \left(t_j \log y_j + (1 - t_j) \log (1 - y_j) \right)$$

1	0
t_i	$1 - t_i$
y_i	$1 - y_i$

$$i = 1, 2, \cdots, m$$

x_i에 대하여 $\mathrm{P}(x_i = 1)$값인 t_i값이 주어져 있다. 이 t_i는 0 또는 1이다.

t_i는 $y_i = \frac{1}{1+\exp(-ax_i-b)}$로 추정한다(비슷하길 바란다). 그리고, $1 - t_i$는 $1 - y_i$값으로 추정한다.

$\frac{\partial E}{\partial a}$와 $\frac{\partial E}{\partial b}$ 를 계산한 후, 최급강하법으로 돌리면 학습 끝!

$$a \leftarrow a - \eta \frac{\partial E}{\partial a} \qquad b \leftarrow b - \eta \frac{\partial E}{\partial b}$$

$$E = -\sum_{j=1}^{m}\left(t_j \log y_j + (1-t_j)\log(1-y_j)\right)$$

$$y_i = \frac{1}{1+\exp(-ax_i+b)}$$

$$\frac{\partial E}{\partial a} = \sum_{j=1}^{m} x_j(y_j - t_j)$$

$$\frac{\partial E}{\partial a} = -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial a}\log y_j + (1-t_j)\frac{\partial}{\partial a}\log(1-y_j)\right)$$

$$= -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial y_j}\log y_j \frac{\partial y_j}{\partial a} + (1-t_j)\frac{\partial}{\partial y_j}\log(1-y_j)\frac{\partial y_j}{\partial a}\right) = -\sum_{j=1}^{m}\left(t_j \frac{1}{y_j}\frac{\partial y_j}{\partial a} + (1-t_j)\frac{-1}{1-y_j}\frac{\partial y_j}{\partial a}\right)$$

$$= -\sum_{j=1}^{m}\left(\frac{t_j}{y_j}(1-y_j)\, y_j\, x_j - \frac{1-t_j}{1-y_j}(1-y_j)\, y_j\, x_j\right) = -\sum_{j=1}^{m} x_j\left(t_j(1-y_j) - (1-t_j)\, y_j\right) = \sum_{j=1}^{m} x_j(y_j - t_j)$$

$$\frac{\partial y_j}{\partial a} = \frac{\partial y_j}{\partial u_j}\frac{\partial u_j}{\partial a} = (1-y_j)\, y_j\, x_j$$

$$u_j = ax_j + b$$

$$E = -\sum_{j=1}^{m}\left(t_j \log y_j + (1-t_j)\log(1-y_j)\right)$$

$$y_i = \frac{1}{1+\exp(-ax_i+b)}$$

$$\frac{\partial E}{\partial b} = \sum_{j=1}^{m}(y_j - t_j)$$

$$\frac{\partial E}{\partial b} = -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial b}\log y_j + (1-t_j)\frac{\partial}{\partial b}\log(1-y_j)\right)$$

$$= -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial y_j}\log y_j \frac{\partial y_j}{\partial b} + (1-t_j)\frac{\partial}{\partial y_j}\log(1-y_j)\frac{\partial y_j}{\partial b}\right) = -\sum_{j=1}^{m}\left(t_j \frac{1}{y_j}\frac{\partial y_j}{\partial b} + (1-t_j)\frac{-1}{1-y_j}\frac{\partial y_j}{\partial b}\right)$$

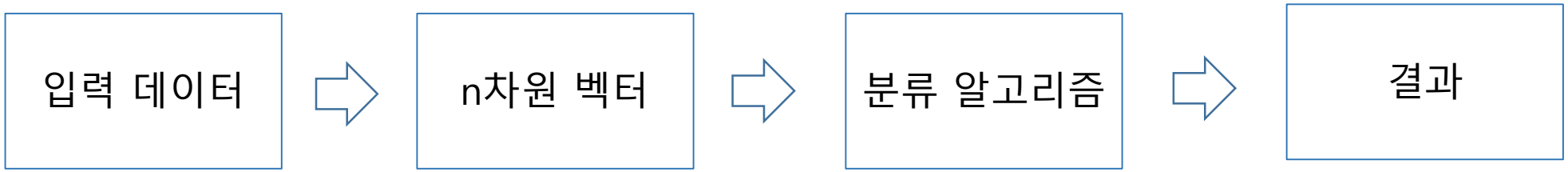
$$= -\sum_{j=1}^{m}\left(\frac{t_j}{y_j}(1-y_j)\,y_j - \frac{1-t_j}{1-y_j}(1-y_j)\,y_j\right) = -\sum_{j=1}^{m}\left(t_j(1-y_j) - (1-t_j)\,y_j\right) = \sum_{j=1}^{m}(y_j - t_j)$$

$$\frac{\partial y_j}{\partial b} = \frac{\partial y_j}{\partial u_j}\frac{\partial u_j}{\partial b} = (1-y_j)\,y_j$$

$$u_j = ax_j + b$$

로지스틱회귀 분류 Classification 정리

학습데이터를 이용하여 $f(x) = \frac{1}{1+e^{-(ax+b)}}$ 학습(fit) 한다. 그리고,



생략 가능~

결정경계: $x = -b/a$

결정규칙:

$$d(x) = \begin{cases} 0 & if\ f(x) < 0.5 \\ 1 & o.w. \end{cases} \qquad x < -b/a$$

* 중요

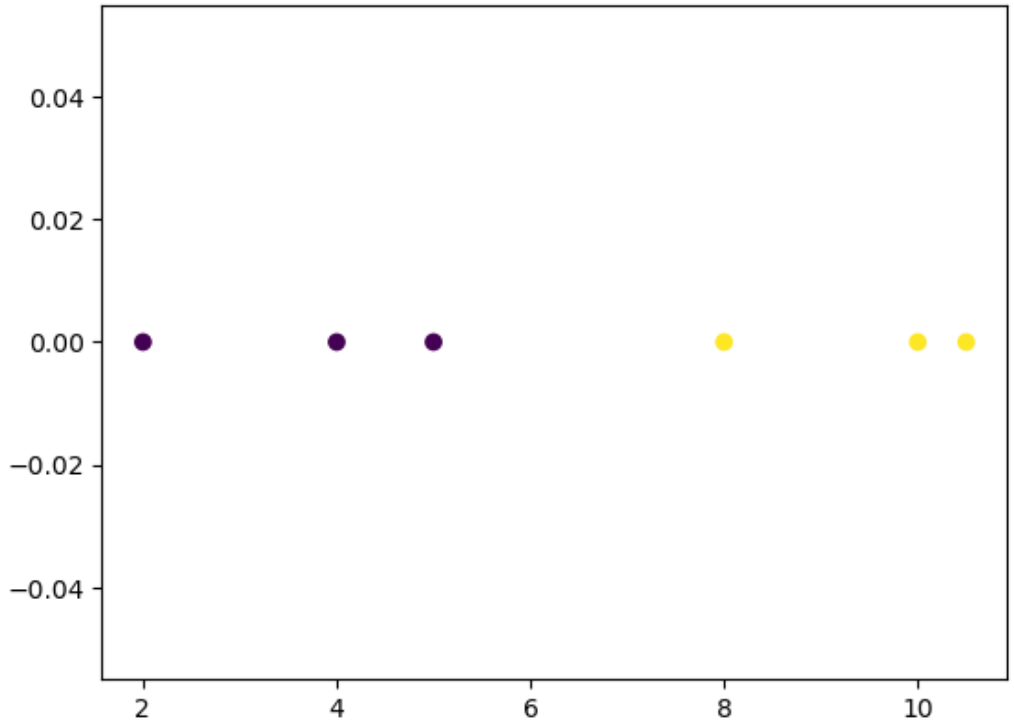
$0 \le f(x) \le 1$ 이므로 $f(x)$를 확률로 해석한다. (엄밀하게는 확률은 아니다)

$$f(x) = P(x = 1)$$

(실습) 다음 데이터를 분류할 수 있는 로지스틱 회귀 알고리즘을 파이썬 코드를 활용하여 작성하시오. 이 모델로 $x = 7$ 일 때, 성공인지 실패인지 예측하시오.

x	2	4	5	8	10	10.5
t	0	0	0	1	1	1

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([2, 4, 5, 8, 10, 10.5])
t = np.array([0, 0, 0, 1, 1, 1])

plt.scatter(x, np.zeros(len(x)), c=t)
plt.show()
```



Eta와 epoch 수 변경해 보세요~

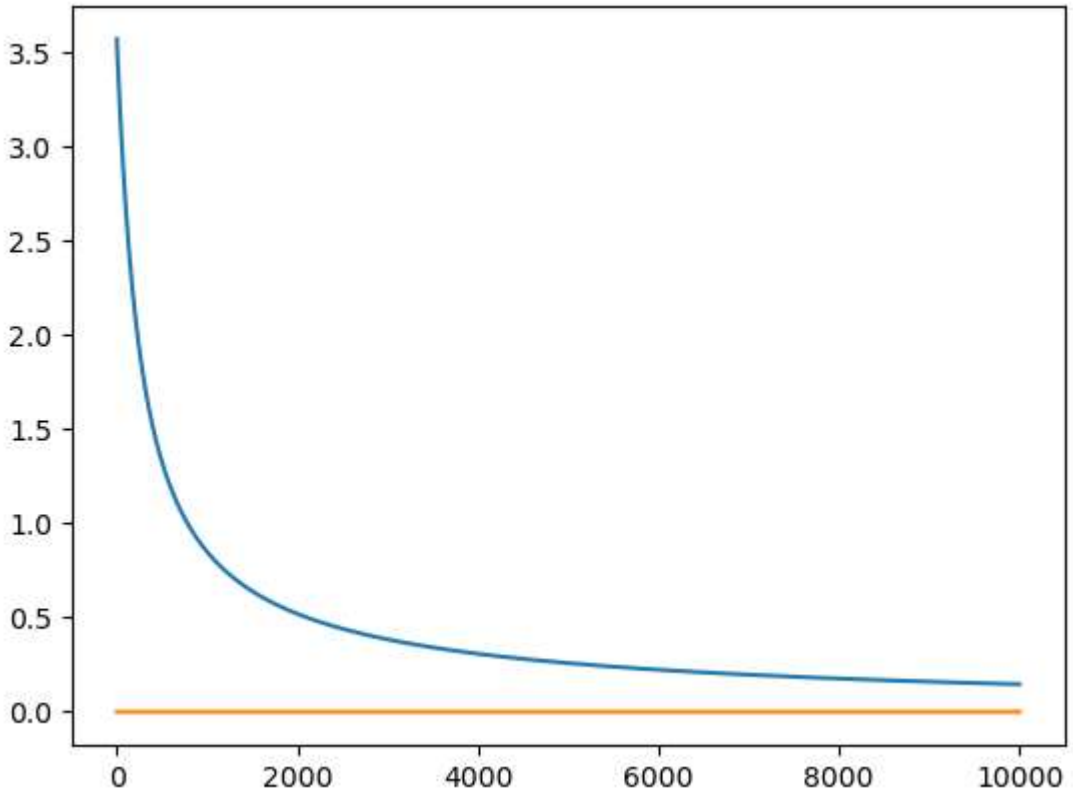
```
eta = 0.01
epoch = 30000

def y(x, a, b):
    u = a*x + b
    return 1/(1+np.exp(-u))

def lr_loss(T, Y): # cross entropy loss function
    delta = 1e-100
    return - np.sum(T*np.log(Y+delta)+(1-T)*np.log(1-
Y+delta))  # 주의! 앞에 – 있음

def dE_da(X, Y, T):
    return np.sum( X*(Y-T))

def dE_db(Y, T):
    return np.sum(Y-T)
```

$$\frac{\partial E}{\partial a} = \sum_{j=1}^{m} x_j(y_j - t_j) \qquad \frac{\partial E}{\partial b} = \sum_{j=1}^{m} (y_j - t_j)$$

$$E = -\sum_{j=1}^{m} \left(t_j \log y_j + (1 - t_j) \log (1 - y_j) \right)$$

$$y_i = \frac{1}{1 + \exp(-ax_i + b)}$$

```
def lr_fit_1D(X, T, eta, epoch):

    E = np.zeros(epoch)
    E_low= np.ones(epoch)*loss(T, T)

    a, b = np.random.randn(2) # 초기값 설정
    Y = y(X, a, b)

    for i in range(epoch):
        dEda = dE_da(X, Y, T)
        dEdb = dE_db(Y, T)
        a = a - eta*dEda
        b = b - eta*dEdb
        Y = y(X, a, b)
        E[i] = loss(T, Y)

    plt.plot(range(epoch), E)
    plt.plot(range(epoch), E_low)
    plt.show()
    return a, b

a, b = lr_fit_1D(x, t, eta, epoch)
print(a, b, -b/a)
```



결정규칙:

$$d(x) = \begin{cases} 0 & if\ f(x) < 0.5 \\ 1 & o.w. \end{cases}$$

$x < -b/a$

```
#def lr_predict_1D(x, a, b):
#     y = 1/(1+np.exp(-a*x-b))
#     if y > 0.5:  return 1
#     else:    return 0


def lr_predict_1D(x, a, b):
    if x > -b/a:
        return 1
    else:
        return 0

x = np.linspace(5, 8)
for i in range(len(x)):
    print(x[i], lr_predict_1D(x[i], a, b))
```

$\boldsymbol{x = 7}$ 일 때, 성공? 실패?

각자 코드로 확인.

2차원 로지스틱 회귀

2차원 데이터를 로지스틱 회귀 모델로 분류해 보자.

이진 분류

데이터

2차원 데이터의 이진 분류

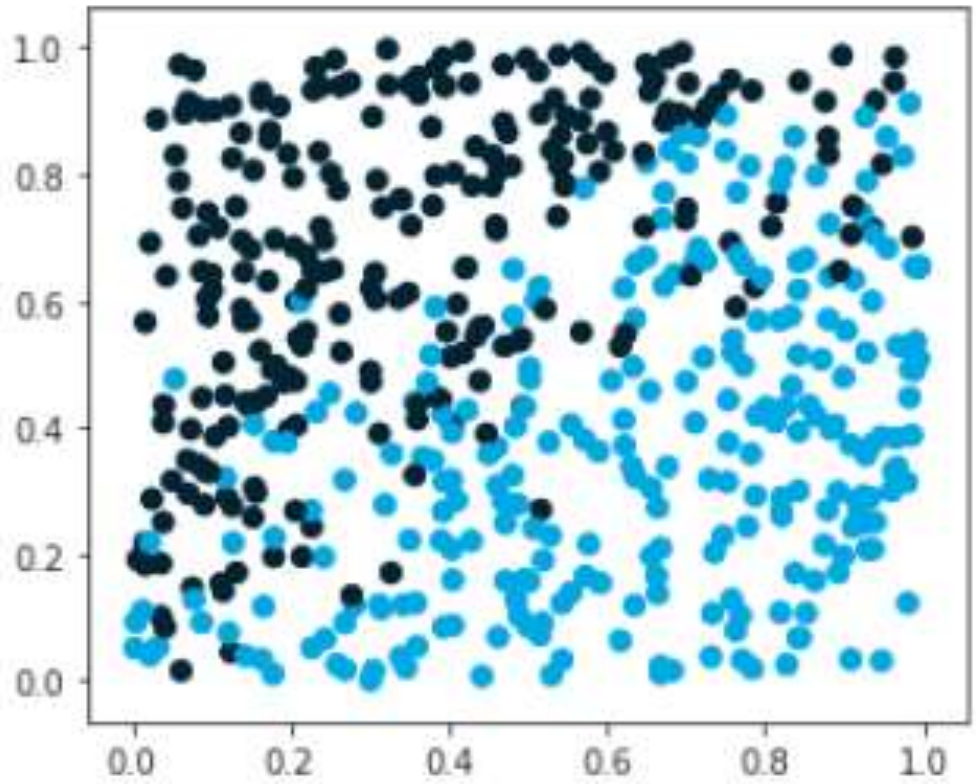
두 클래스의 레이블을 0, 1이라 하자.

x	t
x_1	t_1
x_2	t_2
⋮	⋮
x_m	t_m

x_i은 2차원 벡터, t_i들은 0 또는 1 중 하나.

$$x_k = (x_{k1}, x_{k2})$$

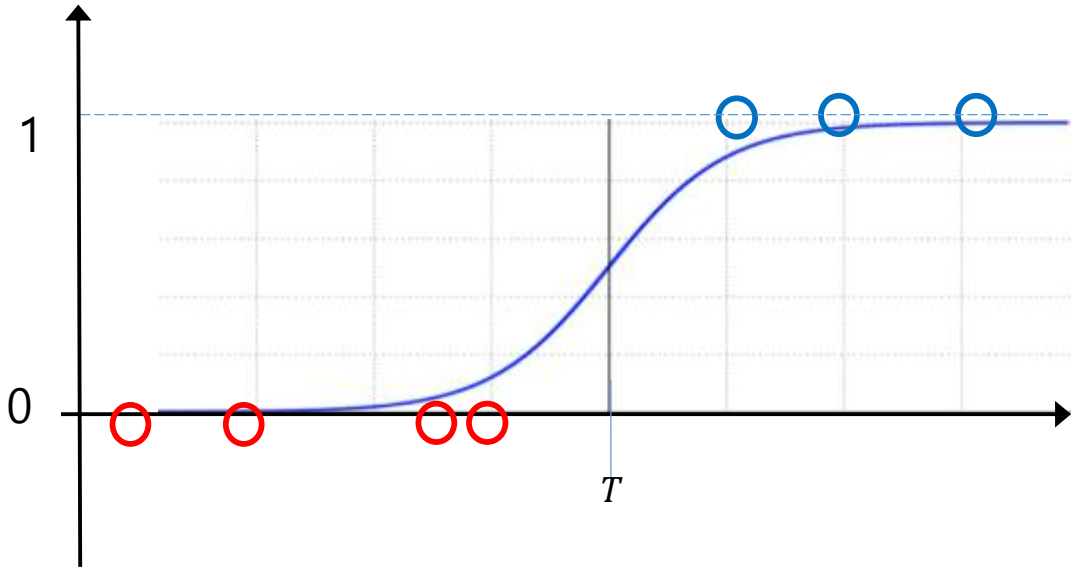
2차원 데이터의 예



시그모이드 함수를 2차원 데이터에 적합하여 분류!

x	t
x_1	t_1
x_2	t_2
⋮	⋮
x_m	t_m

x_i은 2차원 벡터, t_i들은 0 또는 1 중 하나.



$$x_k = (x_{k1},\ x_{k2})$$

$$x = (x_1, x_2)$$

$$f(x) = \frac{1}{1 + e^{-(a_1x_1+a_2x_2+b)}}$$

적합시키는 방법은?

Sigmoid 함수를 옆으로 이동시키고 늘리거나 압축시켜서 데이터에 맞춘다!

$$= \frac{1}{1+\exp(-(a_1x_1+a_2x_2+b))}$$

적합한 다음에 $f(x) = 0.5$ 인 지점을 기준으로 결정규칙을 만든다.

$$= \frac{1}{1+\exp(-(\sum_{k=1}^{2} a_kx_k+b))}$$

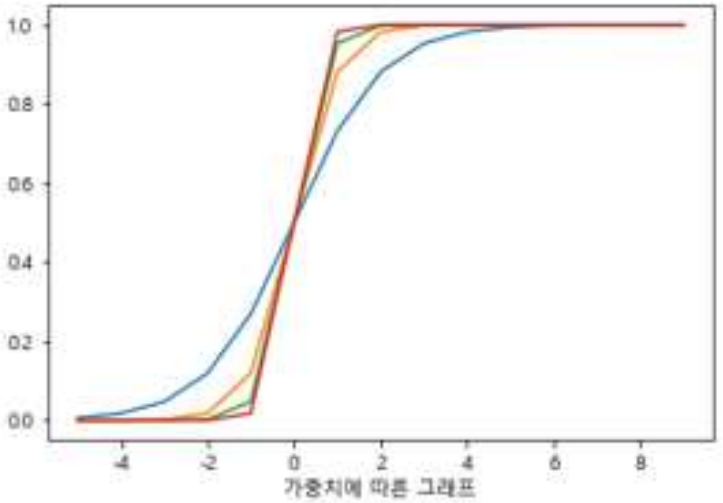
비장의 무기

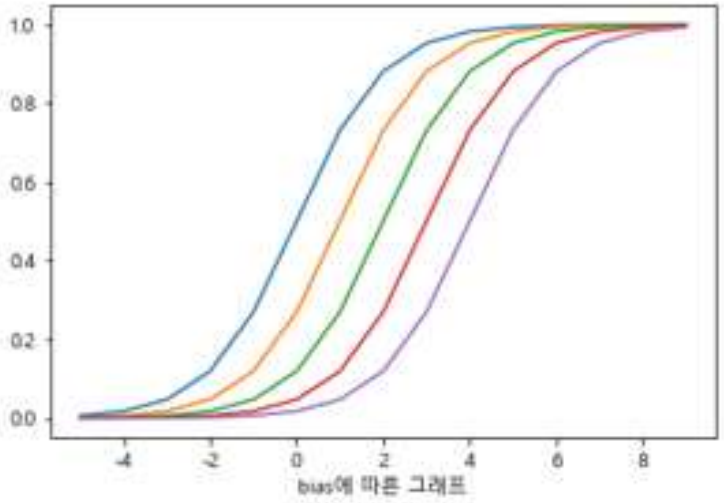
시그모이드 함수의 파라미터 a, b

이 a, b를 데이터로 학습한다.
학습데이터는 이 a, b를 결정하기 위해서만 사용된다.

$$f(x) = \frac{1}{1 + e^{-(ax+b)}}$$

$$f(x) = \frac{1}{1 + e^{-((a_1x_1+a_2x_2)+b)}}$$





결정경계 decision boundary, 결정 규칙 decision rule

데이터를 적합시켜서 적절한 파라미터(숫자) a_1, a_2, b 를 찾았다고 하자.

$$f(x) = \frac{1}{1 + e^{-(a_1 x_1 + a_2 x_2 + b)}}$$

$$x = (x_1, x_2)$$

결정 규칙을 만들자. 이 규칙에 따라 분류할 것이다. 드디어 분류!

어떻게? $f(x) = 0.5$를 만족하는 x를 기준으로 좌/우로 분리(분류).

$f(x) = 0.5$ 인 지점은 무얼까?

$-(a_1 x_1 + a_2 x_2 + b) = 0$인 지점. 즉, $a_1 x_1 + a_2 x_2 + b = 0$인 지점. 직선이다.

결정경계: $a_1 x_1 + a_2 x_2 + b = 0$

결정규칙:

$a_1 x_1 + a_2 x_2 + b < 0$

$$d(x) = \begin{cases} 0 & if\ f(x) < 0.5 \\ 1 & o.w. \end{cases}$$

* 중요

$0 \le f(x) \le 1$ 이므로 $f(x)$를 확률로 해석한다. (엄밀하게는 확률은 아니다)

$$f(x) = P(x = 1)$$

로지스틱 회귀를 이용한 2차원 데이터 분류

$$f(x) = \frac{1}{1 + e^{-(a_1 x_1 + a_2 x_2 + b)}}$$

손실함수 E에 대해, 최급강하법에 의해 다음과 같이 학습한다.

$$a_1 \leftarrow a_1 - \eta \frac{\partial E}{\partial a_1}$$

$$a_2 \leftarrow a_2 - \eta \frac{\partial E}{\partial a_2}$$

$$b \leftarrow b - \eta \frac{\partial E}{\partial b}$$

손실함수 E는 다음과 같이 정의한다.

$$E = -\sum_{j=1}^{m} \left(t_j \log y_j + (1 - t_j) \log (1 - y_j) \right)$$

x	t
x_1	t_1
x_2	t_2
⋮	⋮
x_m	t_m

x_i은 2차원 벡터, t_i들은 0 또는 1 중 하나.

$$x_k = (x_{k1}, x_{k2})$$

실험 전에, a_1, a_2, b 초기화, 학습률 계수 η, epoch 수 결정.

$$H(P,Q) = \sum_X P(X) \log \frac{1}{Q(X)} = -\sum_X P(X) \log Q(X)$$

$$y_i = \frac{1}{1 + \exp(-(a_1 x_{j1} + a_2 x_{j2} + b))}$$

손실함수

손실함수 E는 교차 엔트로피를 이용하여 다음과 같이 정의한다.

$$E = -\sum_{j=1}^{m} \left(t_j \log y_j + (1 - t_j) \log (1 - y_j) \right)$$

1	0
t_i	$1 - t_i$
y_i	$1 - y_i$

$i = 1, 2, \cdots, m$

x_i에 대하여 $\mathrm{P}(x_i = 1)$값인 t_i값이 주어져 있다. 이 t_i는 0 또는 1이다.

t_i는 $y_i = \frac{1}{1+\exp(-(a_1 x_{i1} + a_2 x_{i2} + b))}$로 추정한다는 의미이고, $1 - t_i$는 $1 - y_i$값으로 추정한다는 의미이다.

$\frac{\partial E}{\partial a_1}$와 $\frac{\partial E}{\partial a_2}$, $\frac{\partial E}{\partial b}$ 를 계산한 후, 최급강하법 식으로 업데이트하면서 파라미터를 구하면 학습 끝!

$$E = -\sum_{j=1}^{m}\left(t_j \log y_j + (1-t_j)\log(1-y_j)\right)$$

$$y_i = \frac{1}{1+\exp(-(a_1 x_{j1} + a_2 x_{j2} + b))}$$

$$\frac{\partial E}{\partial a_1} = \sum_{j=1}^{m} x_{j1}(y_j - t_j)$$

$$\frac{\partial E}{\partial a_1} = -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial a_1}\log y_j + (1-t_j)\frac{\partial}{\partial a_1}\log(1-y_j)\right)$$

$$= -\sum_{j=1}^{m}\left(t_j \frac{\partial \log y_j}{\partial y_j}\frac{\partial y_j}{\partial a_1} + (1-t_j)\frac{\partial \log(1-y_j)}{\partial y_j}\frac{\partial y_j}{\partial a_1}\right) = -\sum_{j=1}^{m}\left(t_j \frac{1}{y_j}\frac{\partial y_j}{\partial a_1} + (1-t_j)\frac{-1}{1-y_j}\frac{\partial y_j}{\partial a_1}\right)$$

$$= -\sum_{j=1}^{m}\left(\frac{t_j}{y_j}(1-y_j)\, y_j x_{j1} - \frac{1-t_j}{1-y_j}(1-y_j)\, y_j x_{j1}\right) = -\sum_{j=1}^{m} x_{j1}\left(t_j(1-y_j) - (1-t_j)\, y_j\right) = \sum_{j=1}^{m} x_{j1}(y_j - t_j)$$

$$\frac{\partial y_j}{\partial a_1} = \frac{\partial y_j}{\partial u_j}\frac{\partial u_j}{\partial a_1} = (1-y_j)\, y_j\, x_{j1}$$

$$u_j = a_1 x_{j1} + a_2 x_{j2} + b$$

$$E = -\sum_{j=1}^{m}\left(t_j \log y_j + (1-t_j)\log(1-y_j)\right)$$

$$y_i = \frac{1}{1+\exp(-(a_1 x_{j1} + a_2 x_{j2} + b))}$$

$$\frac{\partial E}{\partial a_2} = \sum_{j=1}^{m} x_{j2}(y_j - t_j)$$

$$\frac{\partial E}{\partial a_2} = -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial a_2}\log y_j + (1-t_j)\frac{\partial}{\partial a_2}\log(1-y_j)\right)$$

$$= -\sum_{j=1}^{m}\left(t_j \frac{\partial \log y_j}{\partial y_j}\frac{\partial y_j}{\partial a_2} + (1-t_j)\frac{\partial \log(1-y_j)}{\partial y_j}\frac{\partial y_j}{\partial a_2}\right) = -\sum_{j=1}^{m}\left(t_j \frac{1}{y_j}\frac{\partial y_j}{\partial a_2} + (1-t_j)\frac{-1}{1-y_j}\frac{\partial y_j}{\partial a_2}\right)$$

$$= -\sum_{j=1}^{m}\left(\frac{t_j}{y_j}(1-y_j)\, y_j x_{j2} - \frac{1-t_j}{1-y_j}(1-y_j)\, y_j x_{j2}\right) = -\sum_{j=1}^{m} x_{j2}\left(t_j(1-y_j) - (1-t_j)\, y_j\right) = \sum_{j=1}^{m} x_{j2}(y_j - t_j)$$

$$\frac{\partial y_j}{\partial a_2} = \frac{\partial y_j}{\partial u_j}\frac{\partial u_j}{\partial a_2} = (1-y_j)\, y_j\, x_{j2}$$

$$u_j = a_1 x_{j1} + a_2 x_{j2} + b$$

$$E = -\sum_{j=1}^{m}\left(t_j \log y_j + (1-t_j)\log(1-y_j)\right)$$

$$y_i = \frac{1}{1+\exp(-(a_1 x_{j1} + a_2 x_{j2} + b))}$$

$$\frac{\partial E}{\partial b} = \sum_{j=1}^{m}(y_j - t_j)$$

1차원 데이터때와 동일. 단 y_j계산법이 다르다.

$$\frac{\partial E}{\partial b} = -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial b}\log y_j + (1-t_j)\frac{\partial}{\partial b}\log(1-y_j)\right)$$

$$= -\sum_{j=1}^{m}\left(t_j \frac{\partial}{\partial y_j}\log y_j \frac{\partial y_j}{\partial b} + (1-t_j)\frac{\partial}{\partial y_j}\log(1-y_j)\frac{\partial y_j}{\partial b}\right) = -\sum_{j=1}^{m}\left(t_j \frac{1}{y_j}\frac{\partial y_j}{\partial b} + (1-t_j)\frac{-1}{1-y_j}\frac{\partial y_j}{\partial b}\right)$$

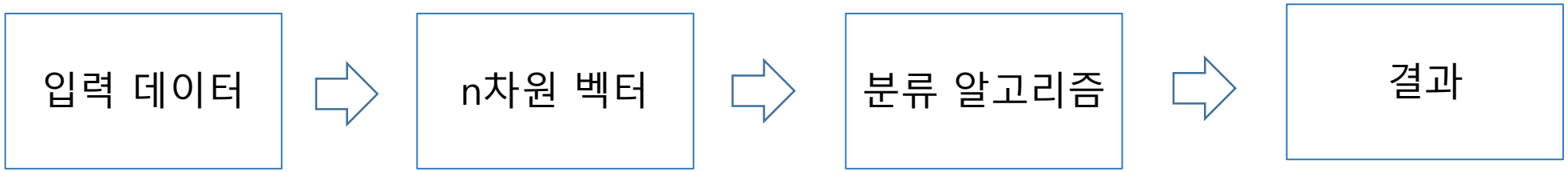
$$= -\sum_{j=1}^{m}\left(\frac{t_j}{y_j}(1-y_j)\,y_j - \frac{1-t_j}{1-y_j}(1-y_j)\,y_j\right) = -\sum_{j=1}^{m}\left(t_j(1-y_j) - (1-t_j)\,y_j\right) = \sum_{j=1}^{m}(y_j - t_j)$$

$$\frac{\partial y_j}{\partial b} = \frac{\partial y_j}{\partial u_j}\frac{\partial u_j}{\partial b} = (1-y_j)\,y_j$$

$$u_j = a_1 x_{j1} + a_2 x_{j2} + b$$

로지스틱회귀 분류 Classification 정리

학습데이터를 이용하여 $f(x) = \frac{1}{1+e^{-(a_1x_1+a_2x_2+b)}}$ 학습(fit) 한다. 그리고,



생략 가능~

결정경계: $a_1\,x_1 + a_2x_2 + b = 0$

결정규칙:

$$d(x) = \begin{cases} 0 & if\ f(x) < 0.5 \\ 1 & o.w. \end{cases}$$

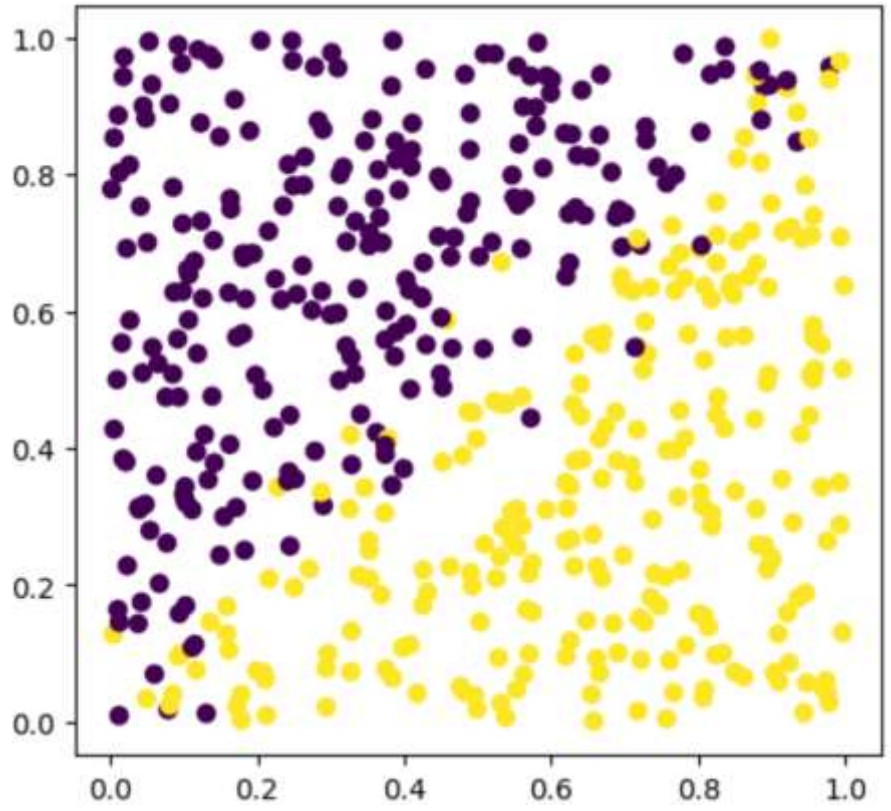
: $a_1\,x_1 + a_2x_2 + b < 0$

* 중요

$0 \le f(x) \le 1$ 이므로 $f(x)$를 확률로 해석한다. (엄밀하게는 확률은 아니다)

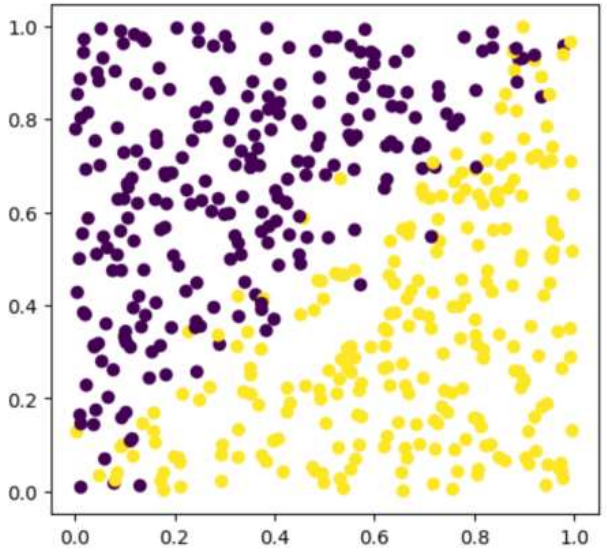
$$f(x) = P(x = 1)$$

(실습) 다음 2차원 데이터를 분류할 수 있는 로지스틱 회귀 알고리즘을 파이썬 코드를 활용하여 작성하시오.



(실습) 다음 2차원 데이터를 분류할 수 있는 로지스틱 회귀 알고리즘을 파이썬 코드를 활용하여 작성하시오.

```
import numpy as np
import matplotlib.pyplot as plt

n_data = 500  # 데이터 수
X = np.zeros((n_data, 2))  # 입력
T = np.zeros((n_data))  # 정답

for i in range(n_data):
    # x、y좌표를 랜덤으로 설정한다
    x_rand = np.random.rand()  # x좌표
    y_rand = np.random.rand()  # y좌표
    X[i, 0] = x_rand
    X[i, 1] = y_rand

    # x가 y보다 큰 영역에서는 정답 라벨을 1로 한다. 경계는 정규분포를 사용해서 약간 불명료하게
    if x_rand > y_rand + 0.1*np.random.randn():
        T[i] = 1

plt.scatter(X[:, 0], X[:, 1], c=T);  plt.colorbar();  plt.show()
```

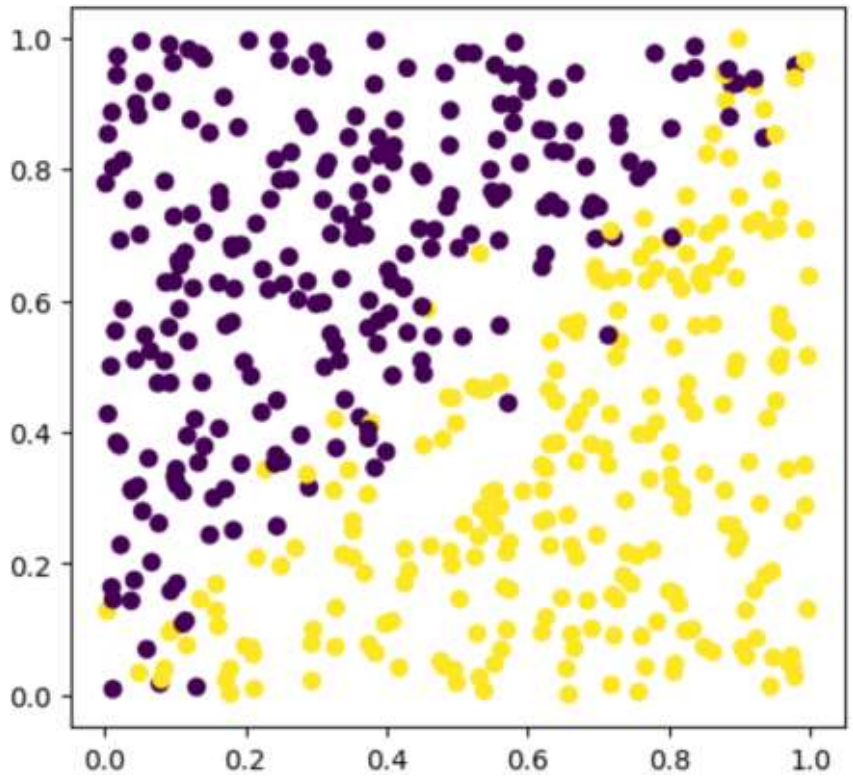



```
# 변수 X, T 저장
np.save('X', X)  # npy 파일로 저장
np.save('T', T)
```

데이터 로드 (npy 파일 로드)

```
# 데이터 로드

X = np.load('X.npy')

T = np.load('T.npy')

print(X.shape, T.shape)
```

```
(500, 2) (500,)
```



y, loss 계산

```
def y(x, a, b):        #a는 각 데이터와 같은 차원
    u = np.dot(x, a) + b
    return 1/(1+np.exp(-u))


def lr_loss(T, Y):
    delta = 1e-100  # 이론적으로 이 delta는 필요없음. 기술적으로 필요.
    return -np.sum(T*np.log(Y+delta)+(1-T)*np.log(1-Y+delta))
)
```

$$E = -\sum_{j=1}^{m}\Big(t_j \log y_j + (1-t_j)\log(1-y_j)\Big)$$

$$y_i = \frac{1}{1+\exp(-(a_1 x_{j1} + a_2 x_{j2} + b))}$$

$\frac{\partial E}{\partial a_1}$, $\frac{\partial E}{\partial a_2}$, $\frac{\partial E}{\partial b}$ 계산

```
def dE_da(X, Y, T, d):  # 주어진 d값에 따라, ∂E/∂a2 또는 ∂E/∂a2 계산
    return np.sum(X[:,d]*(Y-T))
'''
def dE_da(X, Y, T):
    return np.dot(X.T, (Y-T))
'''
def dE_db(Y, T):
    return np.sum(Y-T)
```

파라미터 a를 한꺼번에 업데이트하기 위한 코드!

$$\frac{\partial E}{\partial a_1} = \sum_{j=1}^{m} x_{j1}(y_j - t_j)$$

$$\frac{\partial E}{\partial a_2} = \sum_{j=1}^{m} x_{j2}(y_j - t_j)$$

$$\frac{\partial E}{\partial b} = \sum_{j=1}^{m} (y_j - t_j)$$

학습 learning 코드

```
def lr_fit_2D(X, T, eta, epoch):
    E = np.zeros(epoch)
    E_low = np.ones(epoch)*lr_loss(T, T)

    a1, a2, b = np.random.randn(3) # 최기값 설정
    Y = y(X, [a1, a2], b)

    for i in range(epoch):
        dEda1 = dE_da(X, Y, T, 0)
        dEda2 = dE_da(X, Y, T, 1)
        dEdb = dE_db(Y, T)
        a1 = a1 - eta*dEda1
        a2 = a2 - eta*dEda2
        b = b - eta*dEdb
        Y = y(X, [a1, a2], b)
        E[i] = lr_loss(T, Y)

return a1, a2, b
```

```
def predict_2D(x, a, b): # 레이블 예측
    u = np.dot(x, a)+b
    y = 1/(1+np.exp(-u))
    return np.int32(y > 0.5) # 결정 규칙
```

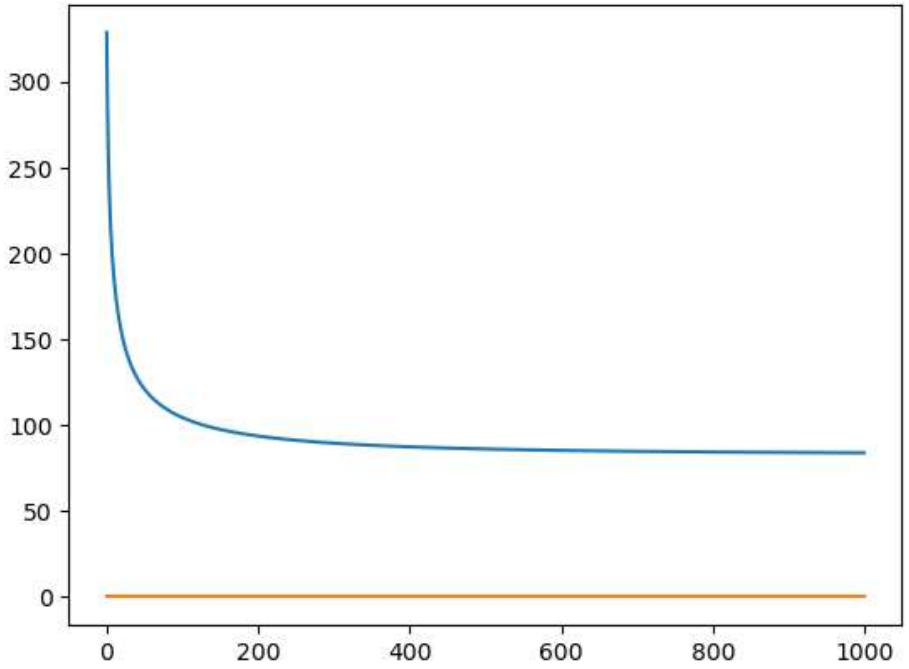
```
plt.plot(range(epoch), E)
plt.plot(range(epoch), E_low)
plt.show()

plt.scatter(X[:,0], X[:,1], c=T)
xmin = min(X[:,0]); xmax = max(X[:,0])
plt.plot([xmin, xmax], [-(a1*xmin+b)/a2, -(a1*xmax+b)/a2])
plt.show()
```

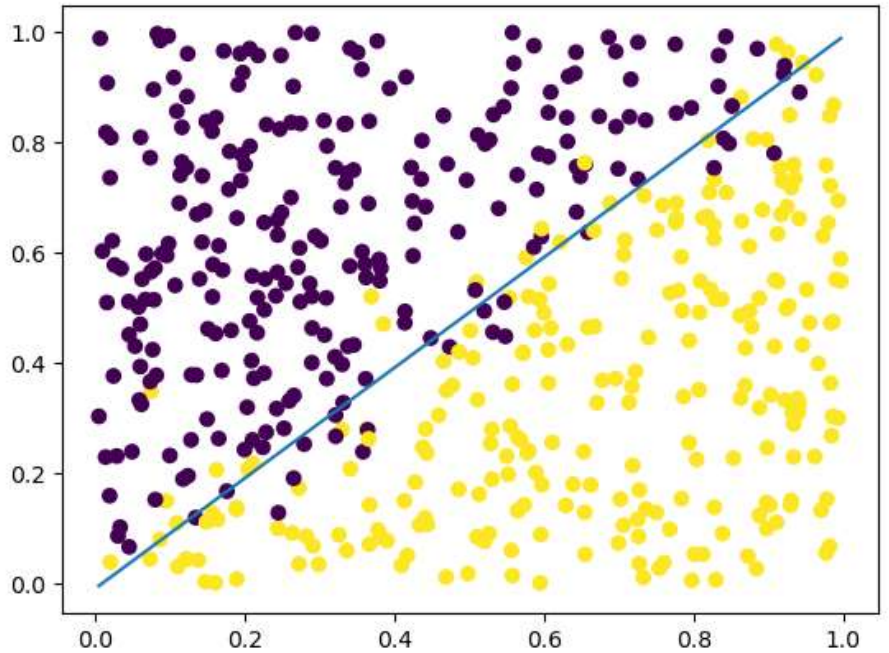
학습 실행, 테스트

```
eta = 0.01
epoch = 1000
a1, a2, b = lr_fit_2D(X, T, eta, epoch)
print(a1, a2, b)
# 학습데이터와 학습결과 그림 확인


# 테스트
X_test = np.array([[0.1, 0.2], [0.9,1], [0.5, 0.55]])
print(predict_2D(X_test, [a1, a2], b))
```

```
16.352740811102064 -16.35858404077616 -0.1363469159253384
[0 0 0]
```

eta와 epoch 수 변경해 보세요~





2차원 데이터 로지스틱 회귀 정리

1. 로지스틱 회귀모델은 이진 분류 문제를 다룬다.

2. 2차원 데이터에 대한 로지스틱 회귀 분류를 수행했다.

3. 학습데이터를 학습하는 최급강하법 식을 유도하시오. 크로스 엔트로피 손실함수 이용.

4. 로지스틱 회귀 알고리즘을 프로그래밍 하시오. 학습과 예측.

5. 벡터 기호를 사용하여, 파라미터 a를 한꺼번에 업데이트 하시오.

참고자료

파라미터 a를 벡터 기호를 이용하여 한꺼번에 계산하는 코드

```
'''
def dE_da(X, Y, T, d):
    return np.sum(X[:,d]*(Y-T))
'''
def dE_da(X, Y, T):
    return np.dot(X.T, (Y-T))
def dE_db(Y, T):
    return np.sum(Y-T)
```

```
def lr_fit_2D(X, T, eta, epoch):
    E = np.zeros(epoch)
    E_low = np.ones(epoch)*lr_loss(T, T)

    # 초기값 설정
    a = np.random.randn(X.shape[1]) # X에서 row vector 가정
    b = np.random.randn(1)
    Y = y(X, a, b)

 for i in range(epoch):
        dEda = dE_da(X, Y, T)
        dEdb = dE_db(Y, T)
        a = a - eta*dEda
        b = b - eta*dEdb
        Y = y(X, a, b)
        E[i] = lr_loss(T, Y)
return a, b
```

```
eta = 0.01
epoch = 1000
a, b = lr_fit_2D(X, T, eta, epoch)

X_test = np.array([[0.1, 0.2], [0.9,1], [0.5, 0.55]])
print(predict_2D(X_test, a, b))
```