Machine Learning & Deep Learning

Lecture 17

Logistic Regression (로지스틱 회귀)
Sigmoid 함수 (시그모이드 함수)

김희숙 (H.S.Kim)

"예, 아니오로만 대답하세요!"

- 법정 드라마나 영화에서 검사가 피고인을 다그치는 장면의 흔한 대사
- 때로 할 말이 많아도 예 혹은 아니오로만 대답해야 할 때가 있음

→ 이와 같은 상황이 딥러닝에서도 끊임없이 일어남

- 전달받은 정보를 놓고 참과 거짓 중에 하나를 판단해 다음 단계로 넘기는 장치들이 딥러닝 내부에서 쉬지 않고 작동한다.
- 딥러닝을 수행한다는 것은 겉으로 드러나지 않는 **'미니 판단 장치'**들을 이용해서 복잡한 연산을 해낸 끝에 최적의 예측 값을 내놓는 작업!

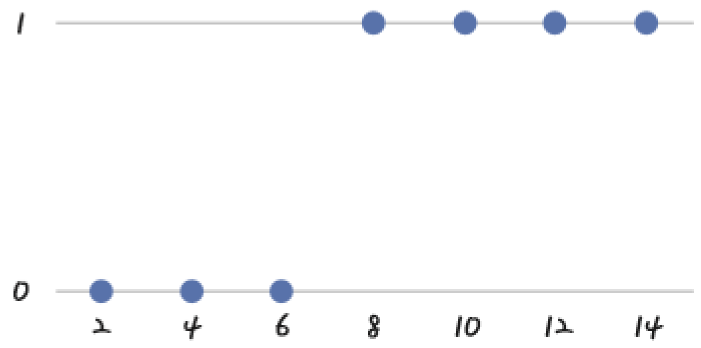
- **로지스틱 회귀(logistic regression)**
  - ✓ 영국의 통계학자인 **데이비드 콕스(D. R. Cox)**가 1958년에 제안한 확률 모델로서 독립 변수의 **선형 결합**을 이용하여 사건의 발생 가능성을 예측하는데 사용되는 통계 기법
  - ✓ 일반적인 **회귀 분석의 목표와 동일**하게 종속 변수와 독립 변수간의 관계를 구체적인 함수로 나타내어 향후 예측 모델에 사용
  - ✓ 독립 변수의 **선형 결합으로 종속 변수를 설명**한다는 관점에서는 **선형 회귀 분석과 유사**
  - ✓ 로지스틱 회귀는 선형 회귀 분석과는 다르게 **입력 데이터가 주어졌을 때 해당 데이터의 결과가 특정 분류로 나뉘기 때문**에 일종의 **분류(classification)** 기법
  - ✓ **참과 거짓 중에 하나**를 내놓는 과정은 **로지스틱 회귀(logistic regression)**의 원리를 거쳐 이루어짐

- 합격과 불합격만 발표되는 시험이 있다고 할때, 공부한 시간에 따른 합격 여부를 조사해 보면 다음과 같다.

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

공부한 시간에 따른 합격 여부

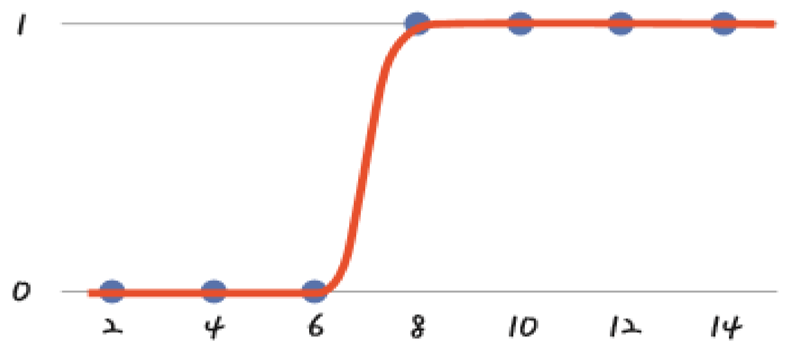
- 합격을 1 불합격을 0 이라 하고, 이를 좌표 평면에 표현하면 다음과 같음



합격과 불합격만 있을 때의 좌표 표현

- 앞장에서 배운 대로 선을 그어 이 점의 특성을 잘 나타내는 일차 방정식을 만들 수 있을까?
- 이 점들은 1과 0 사이의 값이 없으므로 **직선으로 그리기가 어려움**

- 점들의 특성을 정확하게 담아내려면 **직선이 아니라 다음과 같이 S자 형태여야 함**



각 점의 특성을 담은 선을 그었을 때

- 로지스틱 회귀는 선형 회귀와 마찬가지로 적절한 선을 그려가는 과정
- 다만, **직선이 아니라** 참(1)과 거짓(0) 사이를 구분하는 **S자** 형태의 선을 그어 주는 작업

- S자 형태로 그래프가 그려지는 함수가 있다!
  → 시그모이드 함수(sigmoid function)
- 시그모이드 함수를 나타내는 방정식

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

'자연 로그의 밑' or
'오일러의 수'

자연 상수 e

값은 대략 2.718281828... 근사값으로 표현할 수 있음

- 우리가 구해야 하는 값은 결국 지수에 해당하는 $ax + b$

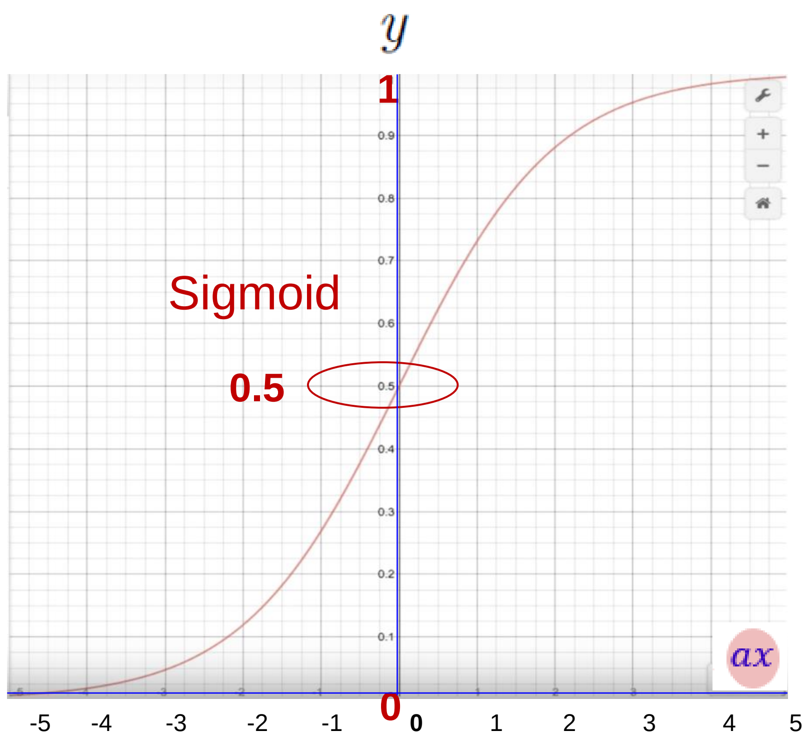
Logistic function or Sigmoid function

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

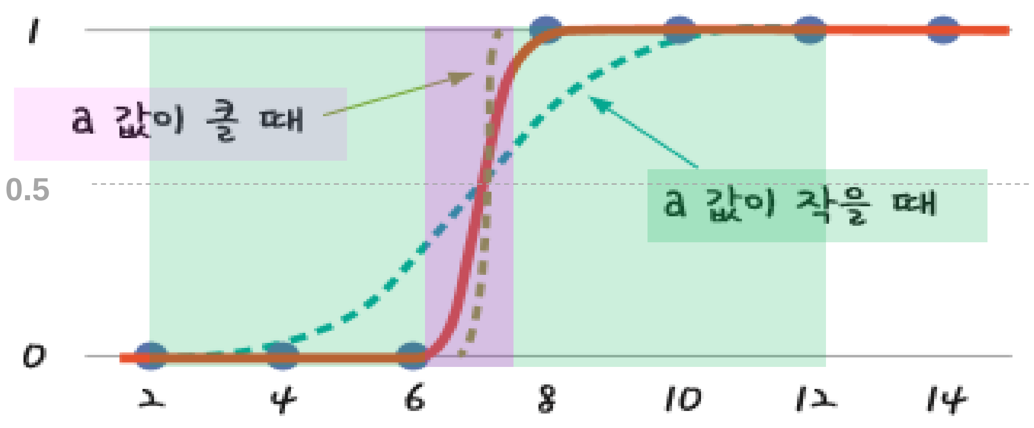
Sigmoid

Curved in two directions,

like the letter "S"



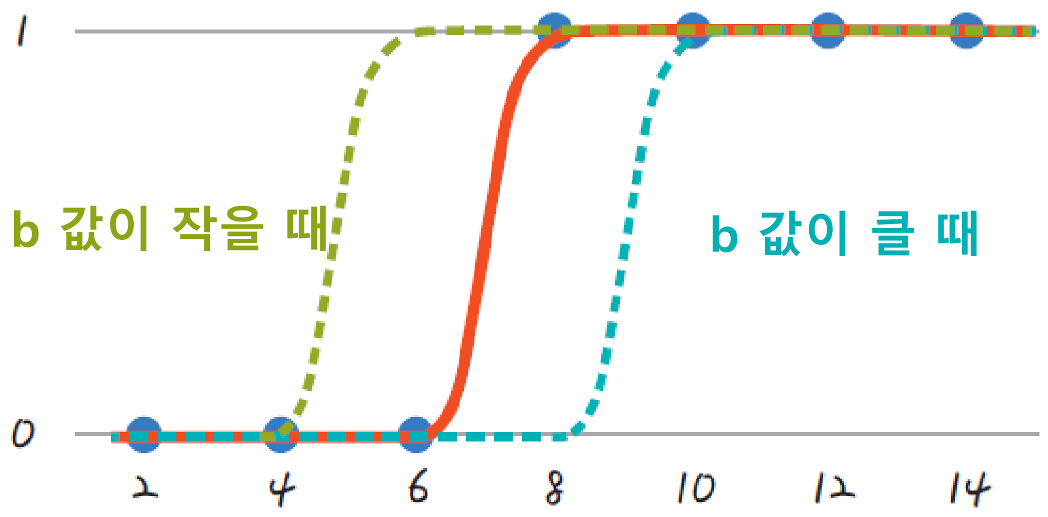
- 선형 회귀에서 우리가 구해야 하는 것이 a 와 b 였듯이 여기서도 마찬가지
- 앞서 구한 직선의 방정식과는 다르게 여기에서 a 와 b 는 어떤 의미를 지니고 있을까?
- **a는 그래프의 경사도를 결정**
- 아래 그림과 같이 **a 값이 커지면 경사가 커지고, a 값이 작아지면 경사가 작아짐**



a 값이 클 때와 작을 때의 그래프 변화

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

- **b는 그래프의 좌우 이동을 의미**
- 아래 그림과 같이 **b 값이 크고 작아짐에 따라 그래프가 이동함**

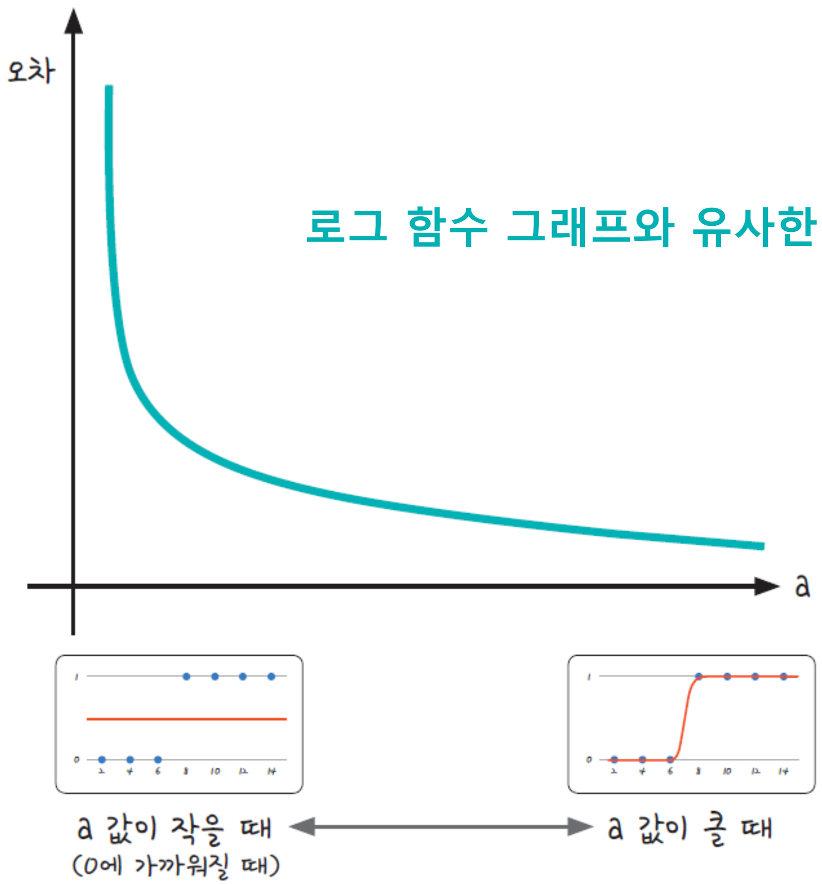


b 값에 따른 그래프 변화

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

- **따라서 a 와 b 의 값에 따라 오차가 변한다.**

a 값에 따라 오차의 그래프는 그림과 같이 변함



로그 함수 그래프와 유사한 형태

a 와 b의 값에 따라 오차가 변한다.

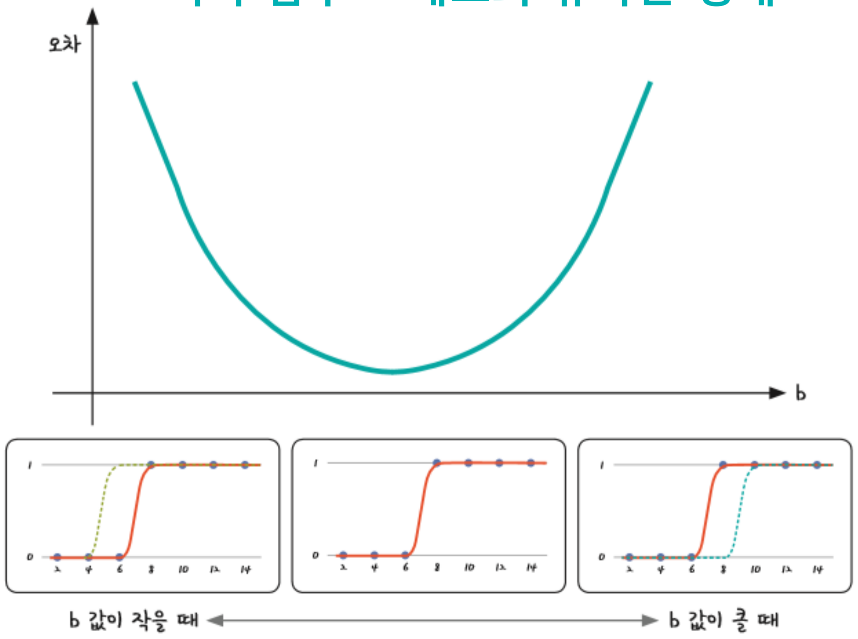
$$y = \frac{1}{1 + e^{-(ax+b)}}$$

a 값이 작아지면 경사가 작아진다.

a 값이 커지면 경사가 커진다.

b 값에 따라 오차의 그래프는 그림과 같이 변함

이차 함수 그래프와 유사한 형태



a 와 b의 값에 따라 오차가 변한다.

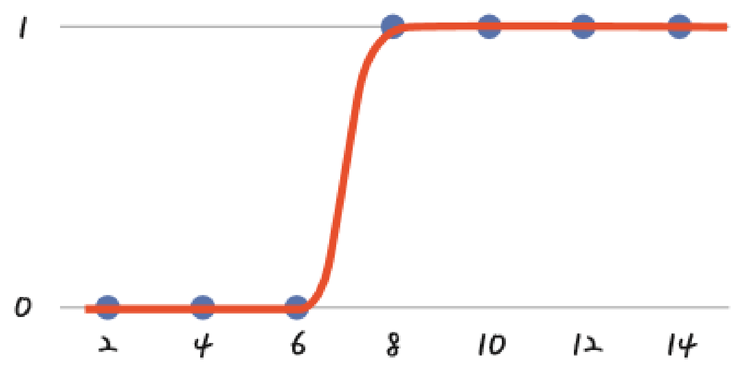
$$y = \frac{1}{1 + e^{-(ax+b)}}$$

- b 와 오차와의 관계
  - ✓ b 값이 너무 작아지거나 커지면 오차도 이에 따라 커진다.

- 이제 우리에게 주어진 과제는 또 다시 a와 b의 값을 구하는 것임
- 시그모이드 함수에서 a와 b의 값을 어떻게 구해야 할까?
- 답은 역시 **경사 하강법**

- 경사 하강법은 먼저 오차를 구한 다음 오차가 작은 쪽으로 이동시키는 방법에 해당함
- 그렇다면 이번에도 예측 값과 실제 값의 차이, 즉 **오차(error)를 구하는 공식**이 필요함

3. 시그모이드 함수의 오차 공식 : 로그 (log) 함수



시그모이드 함수 그래프

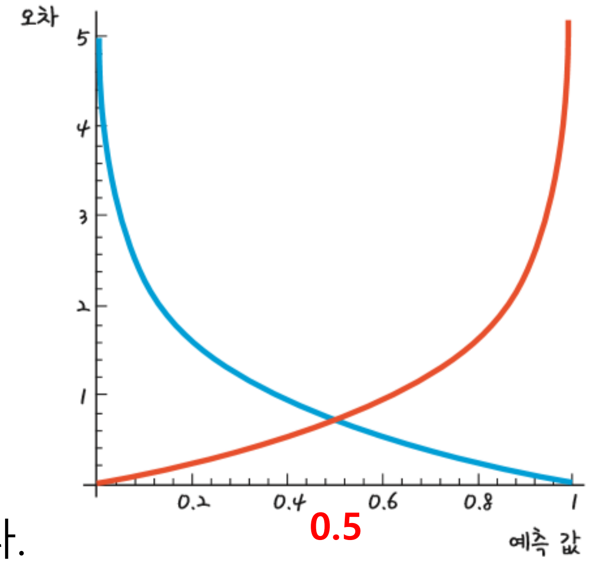
- 시그모이드 함수의 특징은 y 값이 0과 1 사이라는 것
- 따라서 실제 값이 1일 때 예측 값이 0에 가까워지면 오차가 커져야 함
- 반대로, 실제 값이 0일 때 예측 값이 1에 가까워지는 경우에도 오차는 커져야 함
- 이를 공식으로 만들 수 있게 해 주는 함수는 ? **로그 함수**

3. 시그모이드 함수의 오차 공식 : 로그 (log) 함수

- 아래과 같이 **y = 0.5에 대칭**하는 두 개의 로그 함수를 그려보자

실제 값 1 (파란색) vs. 실제 값 0 (빨간색)

로그 함수 그래프



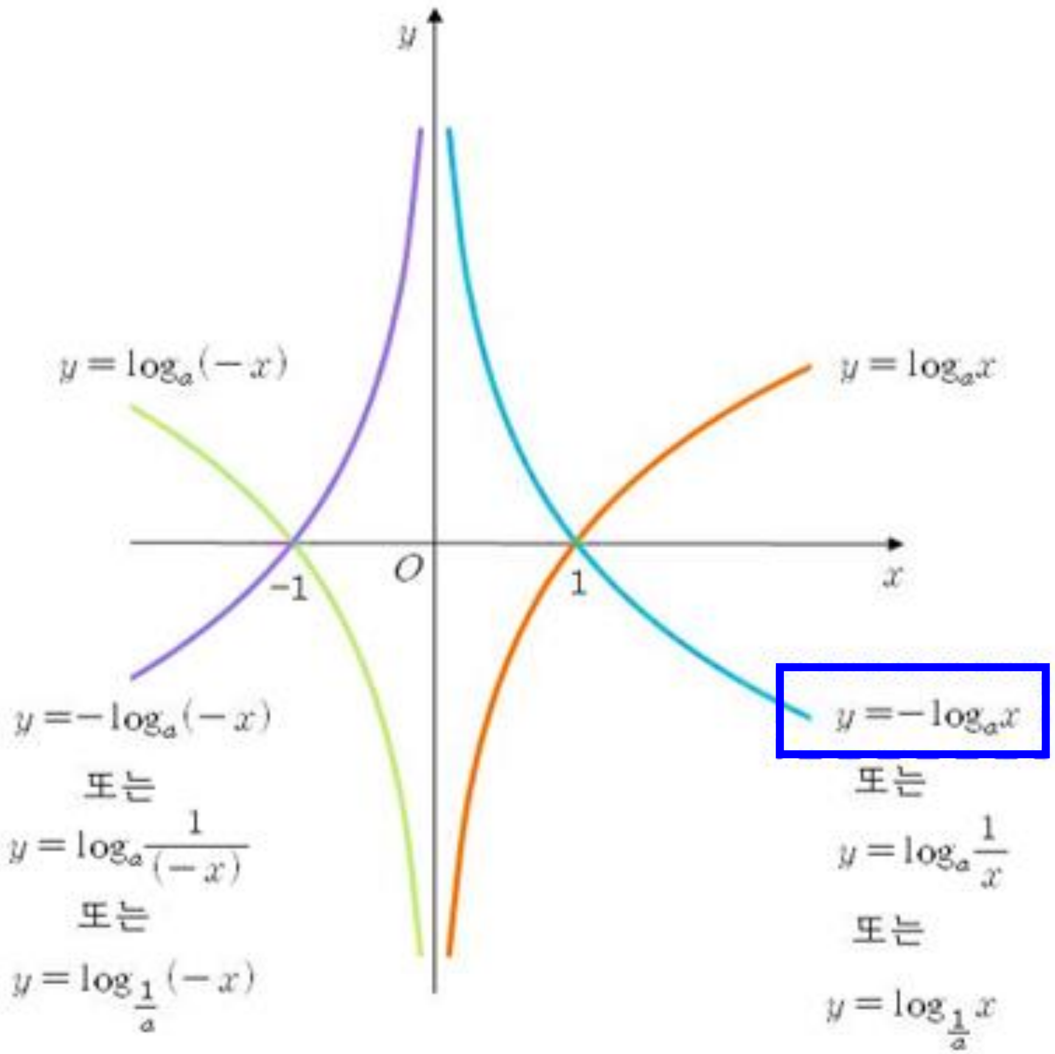
빨간색 선은 실제 값이 0 일때 사용할 수 있는 함수

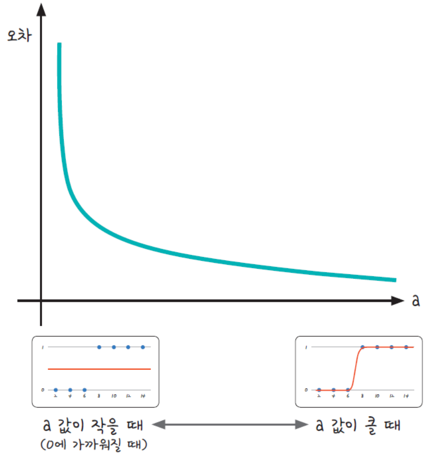
- ✓ 예측 값 **0** → 오차 0
- ✓ 예측 값 1 → 오차가 커진다.

파란색 선은 실제 값이 1 일때 사용할 수 있는 그래프

- ✓ 예측 값 **1** → 오차 0
- ✓ 예측 값 0 → 오차가 커진다.

3. 시그모이드 함수의 오차 공식 : 로그 (log) 함수





$y =$ 실제 값

$h =$ 예측된 값

$c =$ cost (에러, 즉 오차를 의미)

실제 값 1 (파란색)

$$-log(h(x))$$

실제 값 0 (빨간색)

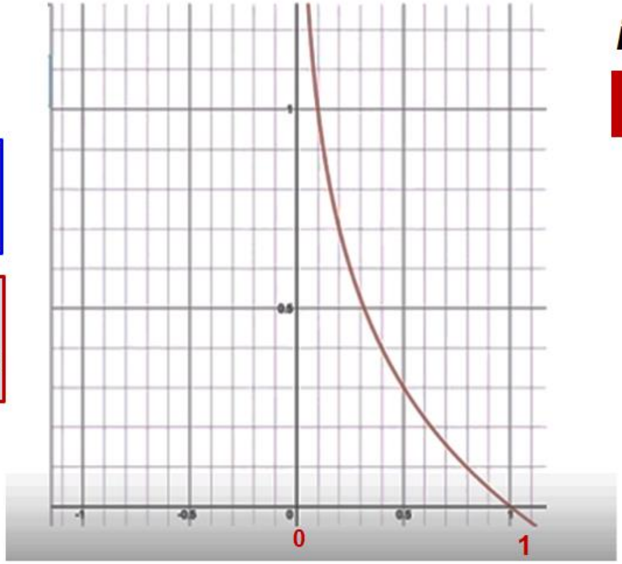
$$-log(1 - h(x))$$

cost

if

y = 1

- **h(x) = 1** → -log(h(x)) = -log(1) = 0 → h(x) = 1, cost = 0
- **h(x) =0** → -log(h(x)) = -log(0) = ∞ → h(x) = 0, cost = ∞



$y = -\log_a x$

if

y = 0

- **h(x) = 1** → -log(1-h(x)) = -log(1-1) = -log(0) = ∞ → h(x) = 1, cost = ∞
- **h(x) =0** → -log(1-h(x)) = -log(1-0) = -log(1) = 0 → h(x) = 0, cost = 0

- 앞의 **실제 값 1 (파란색)**과 **실제 값 0 (빨간색)** 그래프의 식은 각각

$$\boxed{-log(H(x))} \qquad \boxed{-log(1 - H(x))}$$

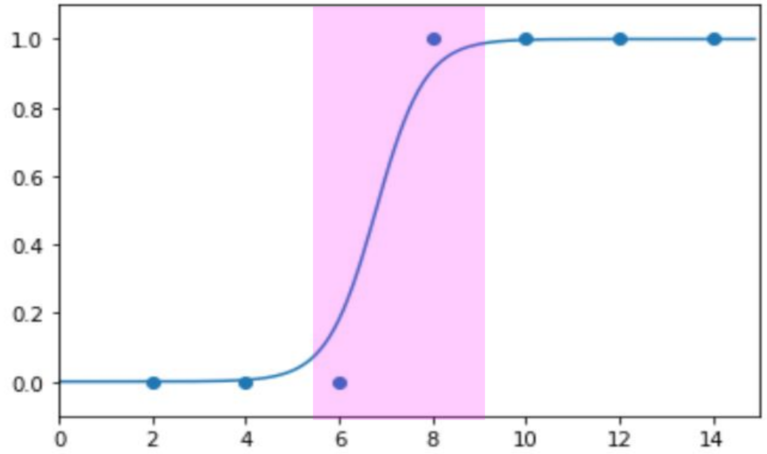
- 이는 다음과 같은 방법으로 해결할 수 있음

$$\boldsymbol{cost\ (H(x), y)} = \quad -\{\underbrace{y \log h}_{\mathrm{A}} + \underbrace{(1-y)\log(1-h)}_{\mathrm{B}}\}$$

✓ **실제 값 y가 1 이면 B** 부분이 없어짐**, 반대로 0 이면 A** 부분이 없어짐

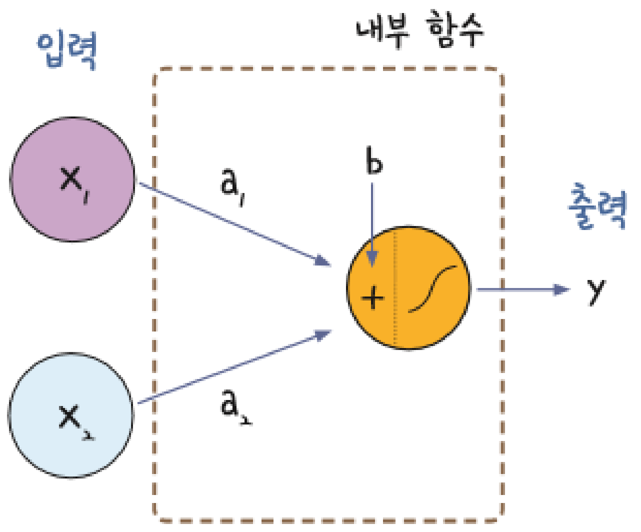
로지스틱 회귀를 코딩으로 구현해 보자.

17_(20 page) 강의용 로지스틱 회귀.ipynb



- 시그모이드 형태의 함수가 잘 만들어지도록 a와 b의 값이 수렴된 것을 알 수 있음
- 만약 여기에 **입력 값이 추가되어 세 개 이상의 입력 값을 다룬다면** 시그모이드 함수가 아니라 **소프트맥스(softmax)**라는 함수를 써야 함

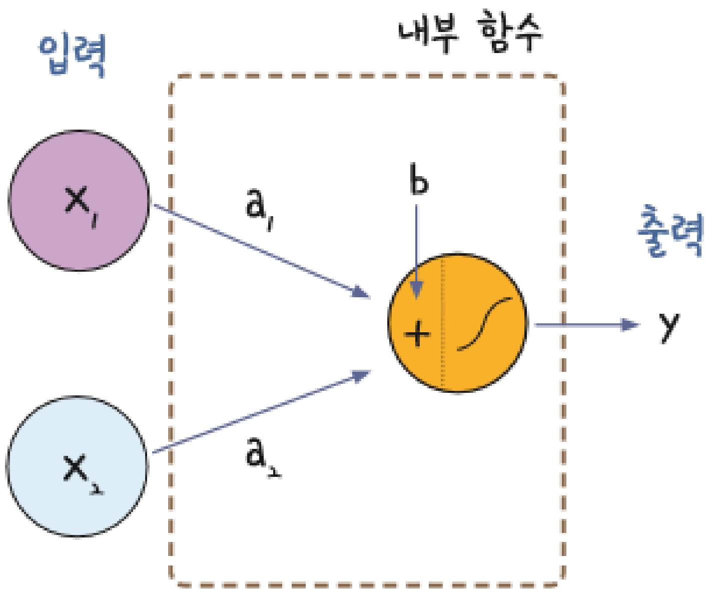
- **Logistic Regression**을 그림으로 나타내면 아래 그림과 같음



$$y = a_1x_1 + a_2x_2 + b$$

- x_1과 x_2가 입력되고, 각각 가중치 **a_1, a_2**를 만남
- 여기에 **b** 값을 더한 후 **시그모이드 함수를 거쳐 1 또는 0 의 출력 값 y 를 출력**

4. 로지스틱 회귀(Logistic Regression)에서 퍼셉트론(Perceptron) 으로



$$y = a_1x_1 + a_2x_2 + b$$

로지스틱 회귀를
퍼셉트론 방식으로 표현한 예

- 이 그림의 개념이 1957년, 코넬 항공 연구소의 '**프랑크 로젠블라트(**Frank Rosenblatt)' 라는 사람이 발표한 **'퍼셉트론(perceptron)'**이다.
- 이 **퍼셉트론**은 그 후 여러 학자들의 노력을 통해 **인공 신경망**, **오차 역전파** 등의 발전을 거쳐 지금의 **딥러닝**으로 이어지게 됨