

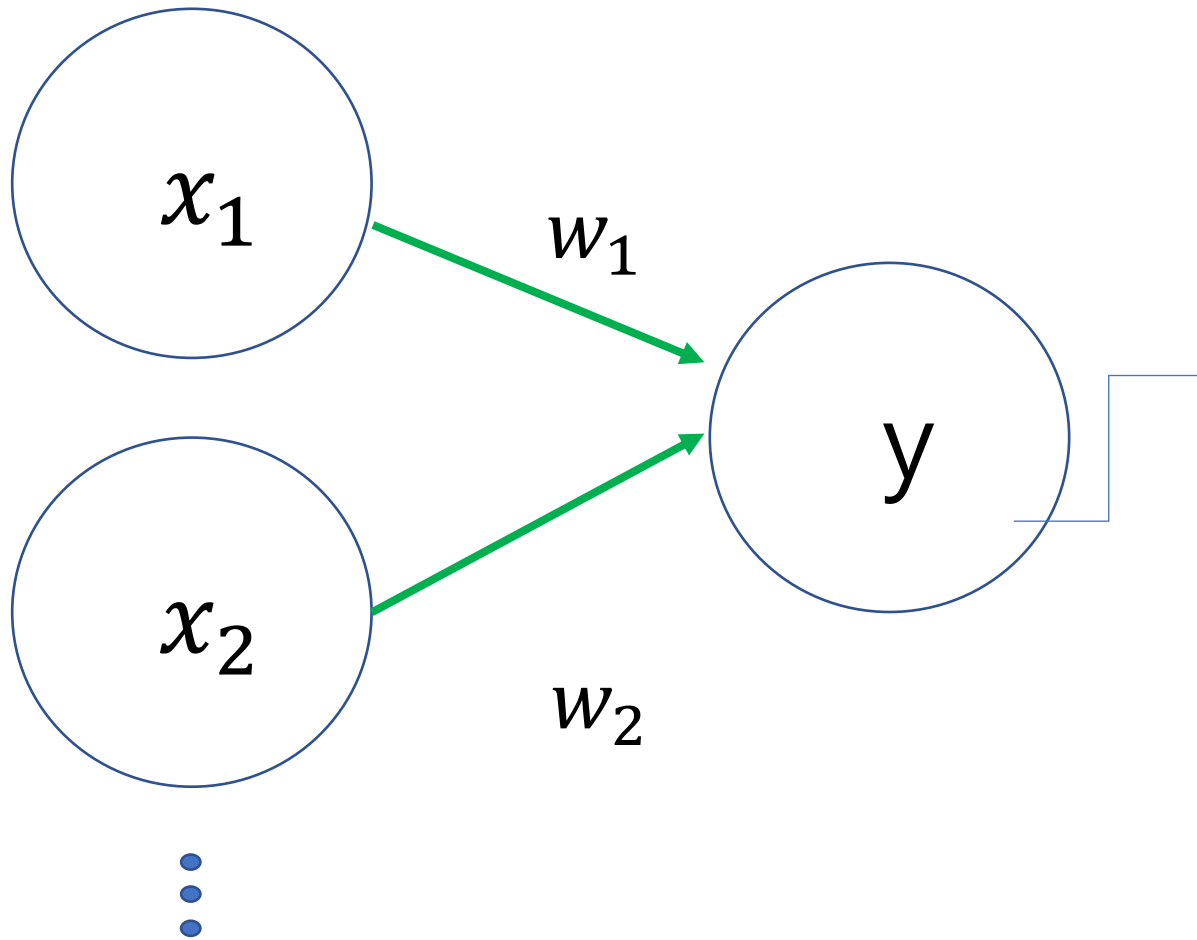
딥러닝(Deep Learning)

기본 알고리즘 알아보기

퍼셉트론 알아보기

- ▶ 가장 간단한 인공 신경망 구조의 하나.
- ▶ 이진 분류의 지도 학습을 위한 알고리즘.(위키 정의)
- ▶ 1957년 프랑크 로젠블라트가 제안.
- ▶ 입력과 출력이 어떤 숫자이고 각각의 입력 연결은 가중치와 연관되어 있음.
- ▶ 입력과 가중치 합을 계산하고, 이어 계산된 합에 계단 함수(step function)을 적용

퍼셉트론 기본 구조

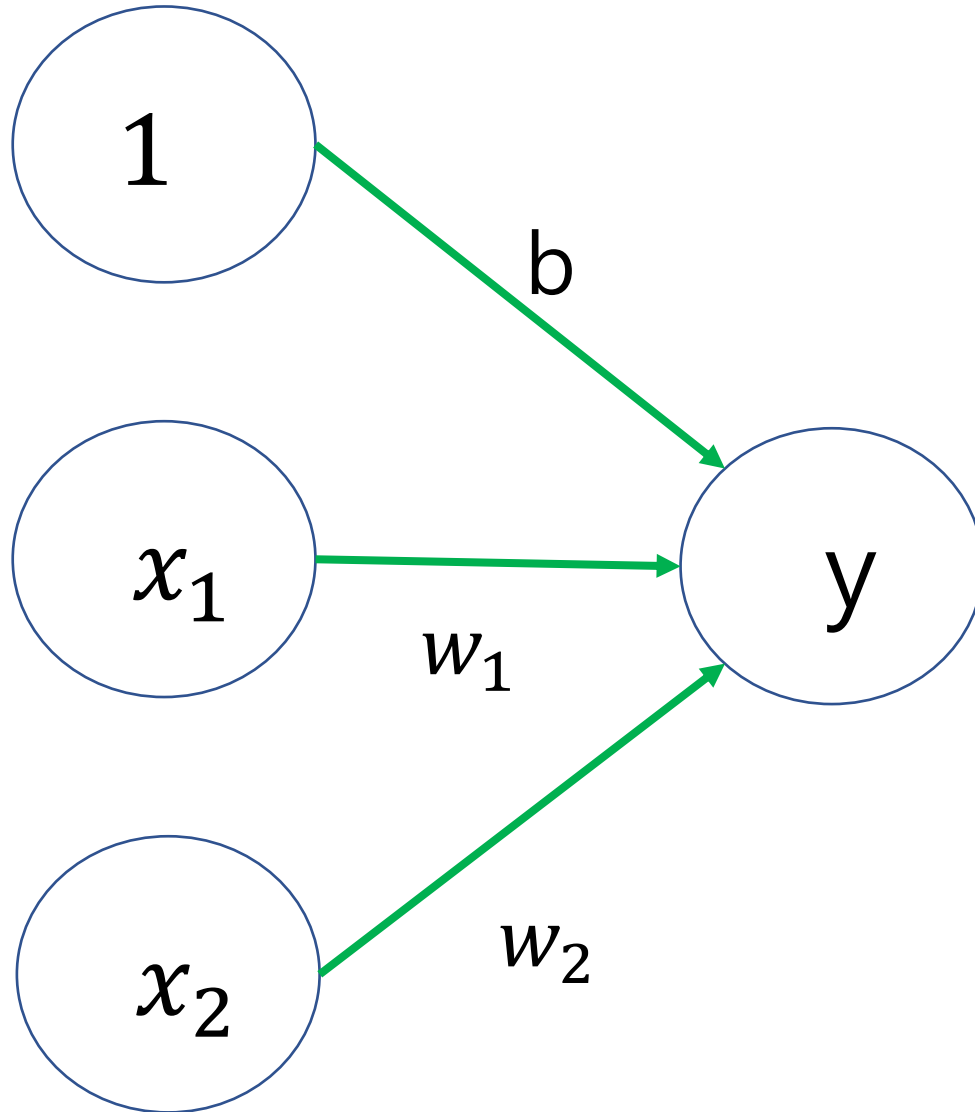


- ▶ 입력값(x_1, x_2)
- ▶ 입력을 가중시키는 가중치(w_1, w_2)
- ▶ 이를 통해 계산된 결과(출력) y
- ▶ y 값에 활성화함수를 통해 임계치(보통) 0보다 크면 뉴런이 활성화되고 결과값이 1이 출력

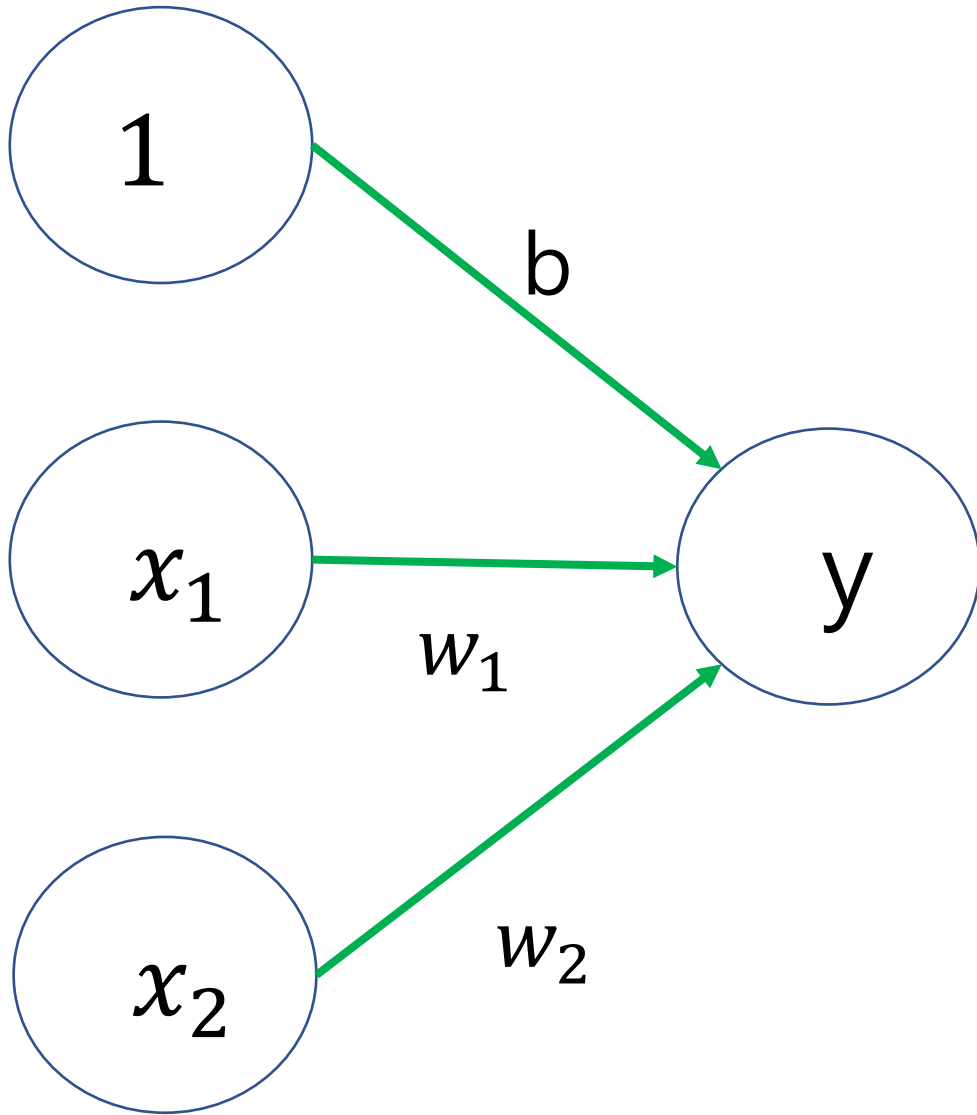
위의 과정을 통틀어, 퍼셉트론이라고 한다.

퍼셉트론 기본 구조

▶ bias는 결정 경계의 위치를 변경



퍼셉트론 동작원리



$$y = \begin{cases} 0 & (w_1 \cdot x_1 + w_2 \cdot x_2 + b \leq \theta) \\ 1 & (w_1 \cdot x_1 + w_2 \cdot x_2 + b > \theta) \end{cases}$$

참고

θ (theta) : 임계값

퍼셉트론과 신경망

- ▶ 다층 퍼셉트론은 퍼셉트론이 여러 층으로 구성된 신경망을 가르킨다.
- ▶ 다층 퍼셉트론은 1986년에 발표되었다.
- ▶ 다층 퍼셉트론은 단층 퍼셉트론이 해결이 어려웠던 XOR 문제를 해결.
- ▶ 다층 퍼셉트론은 (통과) 입력층 하나와 은닉층(hidden layer)라고 불리는 하나 이상의 입력층과 마지막 층인 출력층(output layer)로 구성.
- ▶ 인공 신경망의 은닉층이 2개 이상일 때, 이를 심층 신경망(deep neural network)-DNN이라 한다.

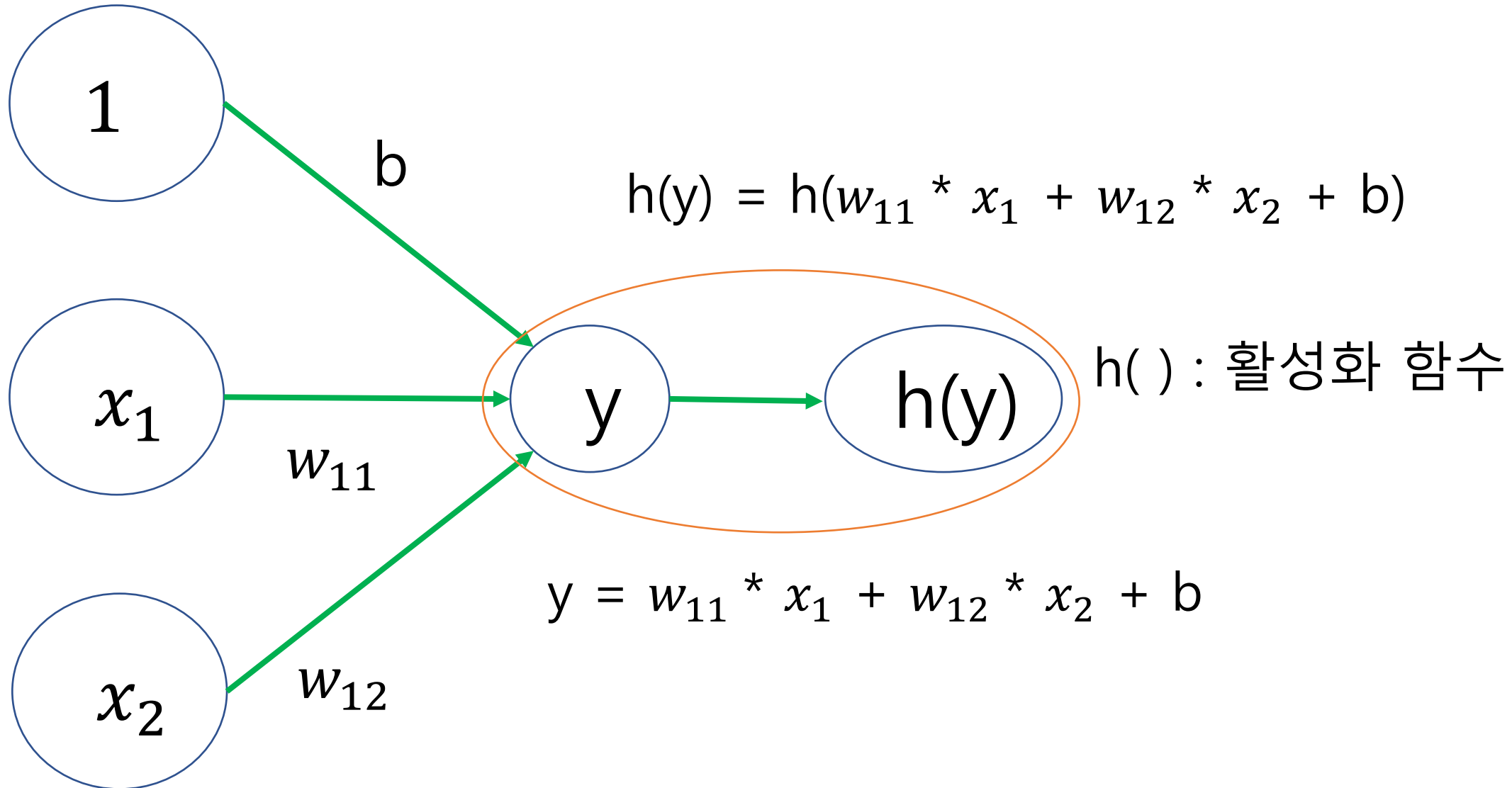
인공 신경망이란?(ANN)

- ▶ 시냅스의 결합으로 네트워크를 형성한 인공 뉴런(노드)가 학습을 통해 가중치를 변화시켜, 문제 해결 능력을 가지는 모델 전반을 가리킨다.
- ▶ 좁은 의미에서 오차 역전파법을 이용한 다층 퍼셉트론을 가리키는 경우도 있지만, 이것은 잘못된 것으로 인공신경망은 이에 국한되지 않음.
- ▶ 통계학적 모델이 다음과 같은 특징을 가질 때, 이를 신경(neural)라 부른다.
 - > A. 조정이 가능한 가중치들의 집합. 즉, 학습 알고리즘에 의해 조정이 가능한 숫자로 표현된 매개 변수로 구성
 - > B. 입력의 비선형 함수를 유추 가능.
- ▶ 신경망이라는 단어는 보통 통계학, 인지 심리학, 그리고 인공지능에서 사용되는 모델들을 가리킨다.

인공 신경망이란?(ANN)

- ▶ 딥러닝의 핵심이며 아주 복잡한 문제를 다루는 데 적합.
- ▶ 수백만 개의 이미지를 분류가 가능하다.
- ▶ 음성 인식 서비스의 성능을 높일 수 있다. (애플의 시리 등)
- ▶ 매일 수억 명의 사람들에게 가장 좋은 비디오를 추천해 준다. (예: 유튜브)
- ▶ 수백만 개의 기보를 익히고 자기 자신과 게임. (예 : 딥마인드 알파고)

활성화 함수의 등장



활성화 함수의 기본 이해

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

활성화 함수 - sigmoid(시그모이드 함수)

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

$\exp(-x)$ 는 e^{-x} 를 뜻한다.

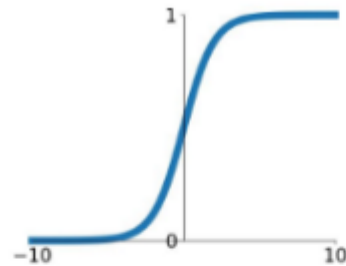
e 는 자연상수로 2.7182의 값을 갖는 실수이다.

$h(1.0) = 0.731$, $h(2.0) = 0.880$

활성화 함수의 종류

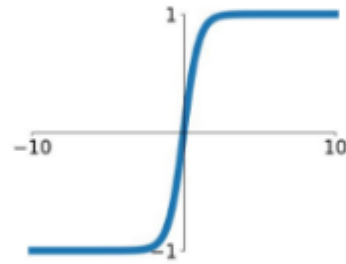
Sigmoid

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



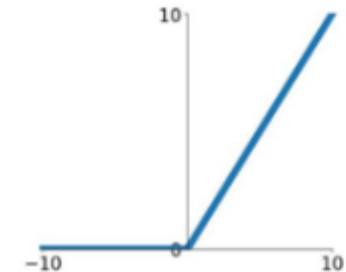
tanh

$$\tanh(x)$$



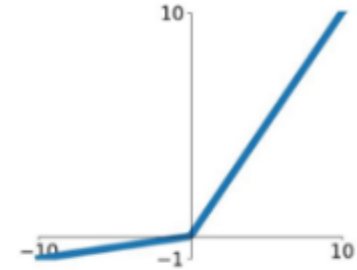
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

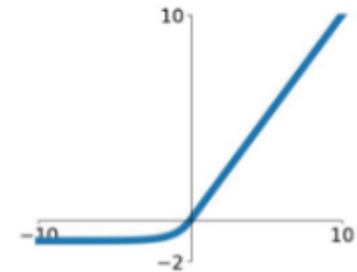


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

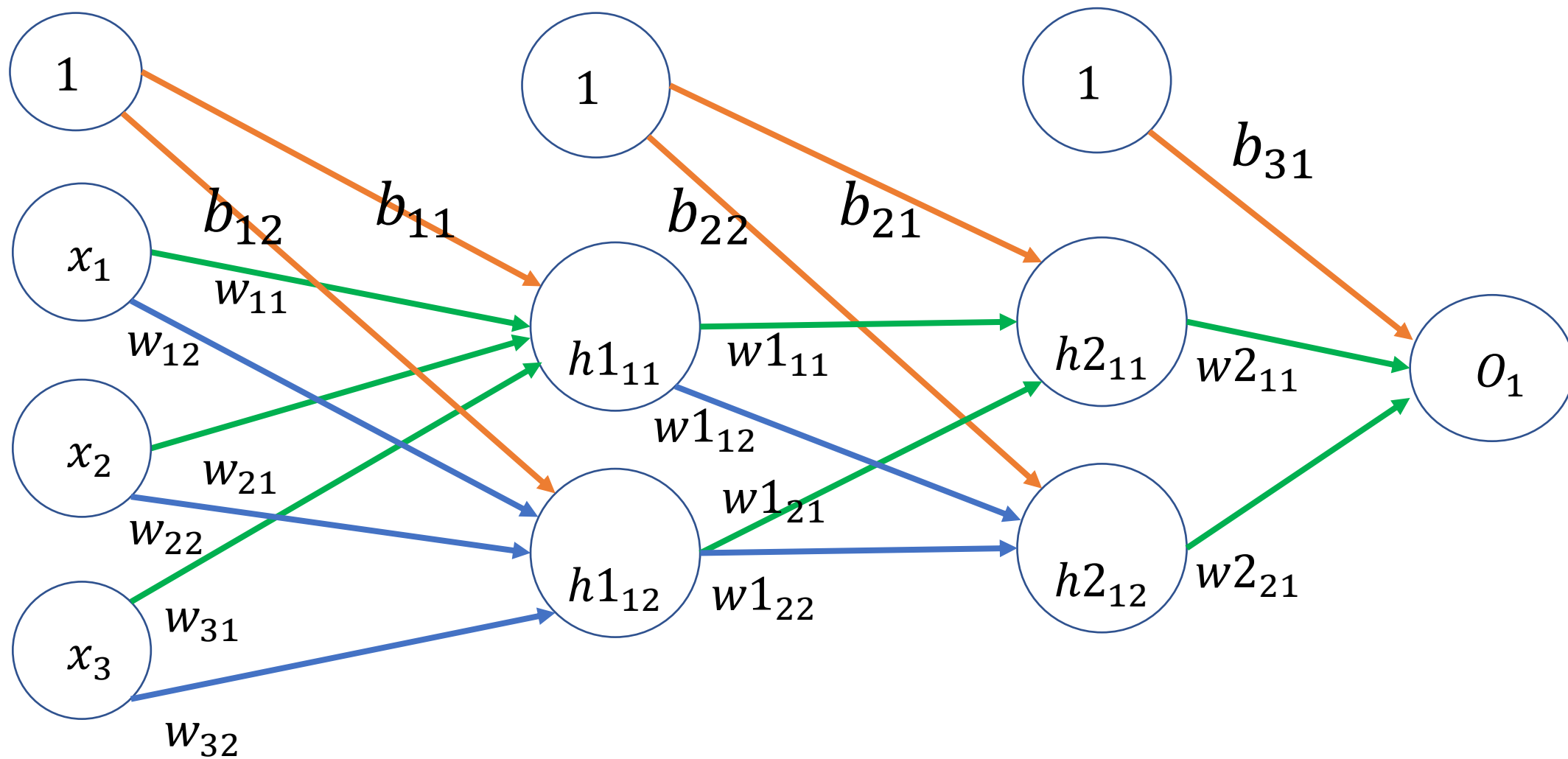
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



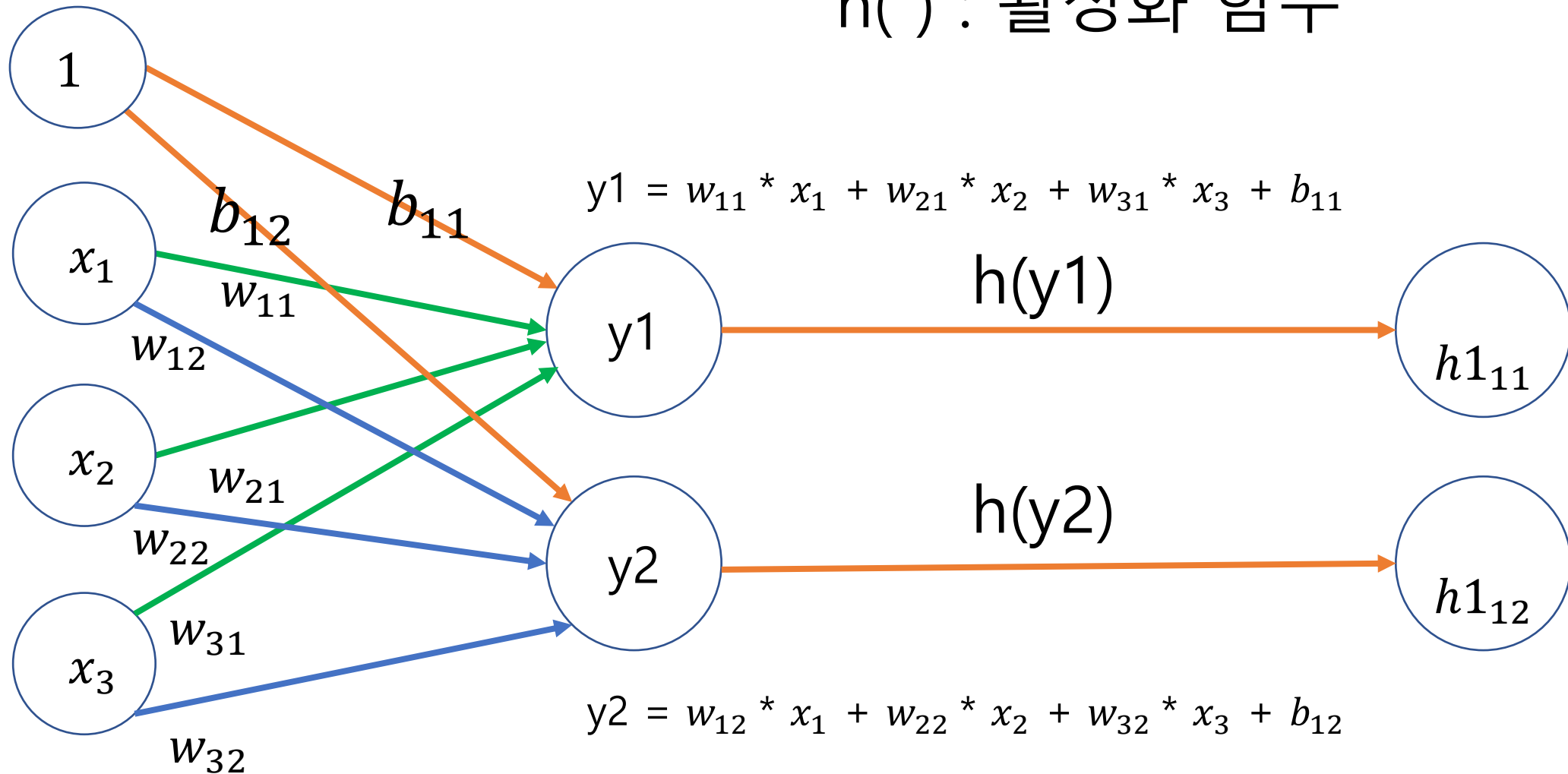
<https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044> 참조

다층 신경망

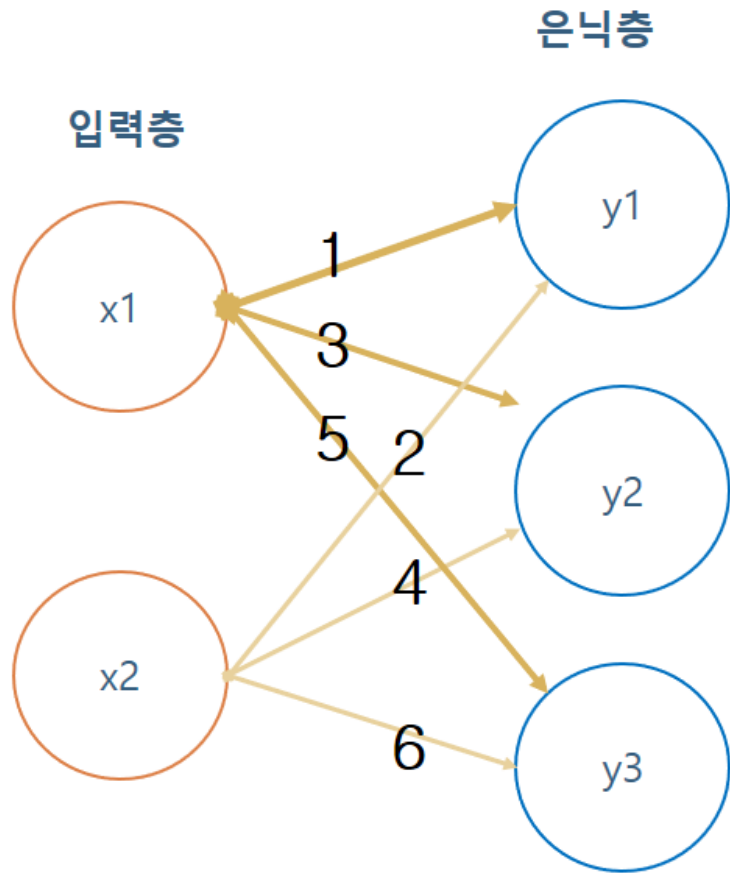


다층 신경망 일부 연산 확인

$h()$: 활성화 함수



신경망을 행렬로 표시(1)



$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

↓

$$\begin{matrix} X & W & = & Y \\ 1 \times 2 & 2 \times 3 & & 1 \times 3 \end{matrix}$$

신경망을 행렬로 표시(2)

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{bmatrix} \times \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} x_{11}w_{11}+x_{12}w_{21}+x_{13}w_{31} & x_{11}w_{12}+x_{12}w_{22}+x_{13}w_{32} \\ x_{21}w_{11}+x_{22}w_{21}+x_{23}w_{31} & x_{21}w_{12}+x_{22}w_{22}+x_{23}w_{32} \\ x_{31}w_{11}+x_{32}w_{21}+x_{33}w_{31} & x_{31}w_{12}+x_{32}w_{22}+x_{33}w_{32} \\ x_{41}w_{11}+x_{42}w_{21}+x_{43}w_{31} & x_{41}w_{12}+x_{42}w_{22}+x_{43}w_{32} \\ x_{51}w_{11}+x_{52}w_{21}+x_{53}w_{31} & x_{51}w_{12}+x_{52}w_{22}+x_{53}w_{32} \end{bmatrix}$$

$[n,3] \qquad \qquad [3, 2] \qquad \qquad [N,2]$

★ 첫번째 행렬의 열, 두번째 행의 행이 같다.

★ 첫번째 행렬의 행, 두번째 행의 열이 결과 행렬의 행렬이 된다.

행렬의 내적(예제)

```
A = np.array([[1,2],[3,4]])  
A.shape
```

```
(2, 2)
```

```
B = np.array([[5,6],[7,8]])  
B.shape
```

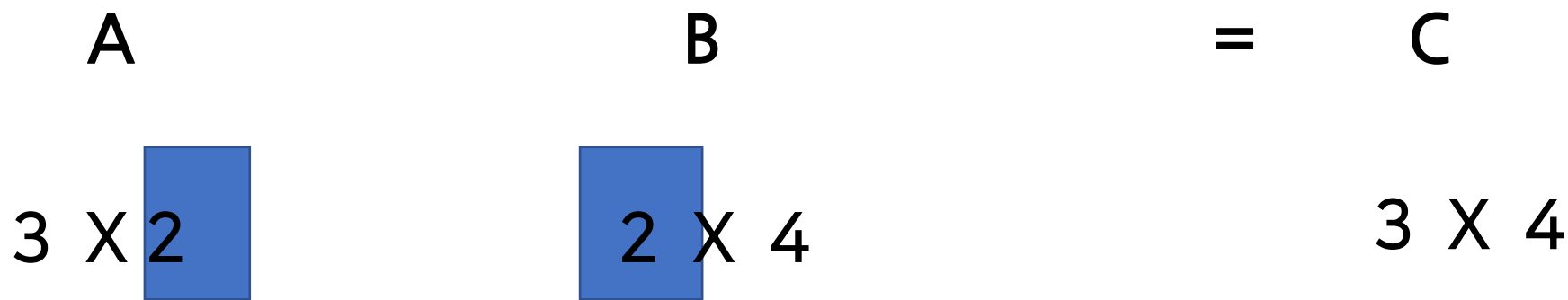
```
(2, 2)
```

```
np.dot(A,B)
```

```
array([[19, 22],  
       [43, 50]])
```

행렬의 곱에서 유의하기

▶ 행렬의 곱에서는 대응하는 차원의 원소 수를 일치 시켜야 함.

$$\begin{array}{ccc} A & B & = C \\ 3 \times 2 & 2 \times 4 & 3 \times 4 \end{array}$$


다차원 배열 연산 예제

```
import numpy as np
A = np.array([1,2,3,4])
print(A)
```

[1 2 3 4]

```
np.ndim(A)
```

배열의 차원수 확인

1

```
A.shape
```

배열의 형상

(4,)

```
A.shape[0]
```

4

```
import numpy as np
B = np.array([[1,2],[3,4],[5,6]])
print(B)
```

[[1 2]
 [3 4]
 [5 6]]

```
np.ndim(B)
```

배열의 차원수 확인

2

```
B.shape
```

배열의 형상

(3, 2)

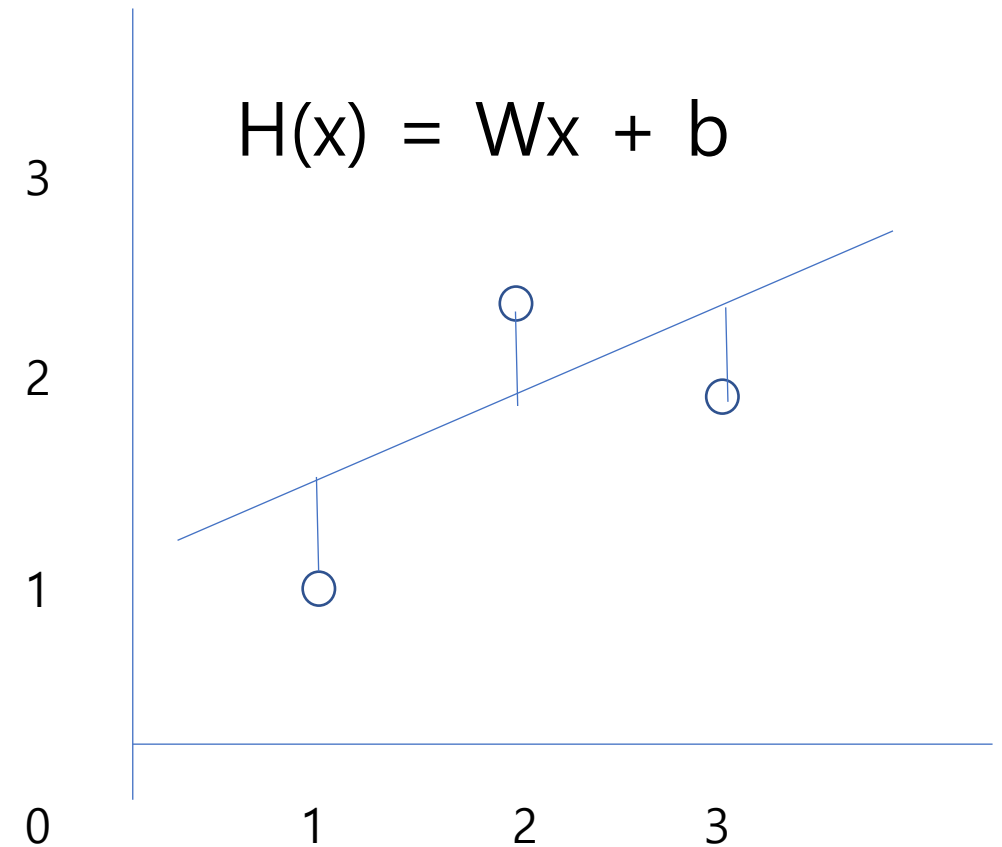
Cost function, loss function

- ▶ 비용 함수(cost function)는 전체 훈련 세트(또는 미니 배치 경사 하강법의 경우, 미니 배치)에 대한 것.
- ▶ 손실 함수(loss function)는 학습용 샘플의 하나의 예제에 대한 것.

Cost Function에 대해 알아보기

▶ 비용함수는 주어진 훈련 샘플과 예상 출력과 관련하여 신경망이 얼마나 좋은가를 측정한 것.

$$\text{Cost} = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$



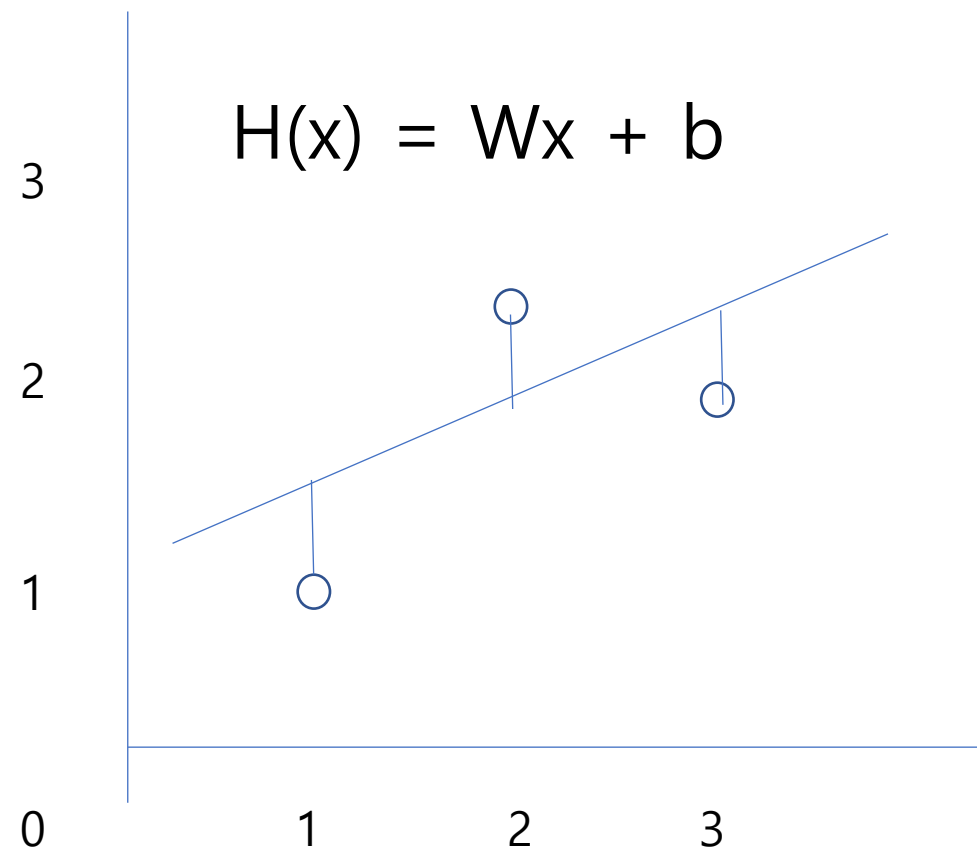
예측 모델과 cost function

▶ 모델(Hypothesis)과 cost Function

$$\text{Cost} = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx + \cancel{b}$$

$$\text{Cost}(W, b) = \frac{1}{m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$



모델의 목표는 비용을 최소화

$$\text{cost}(W,b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

Minimize cost(W,b)

- ▶ 만약 모델이 비용함수를 MSE(Mean Squared Error)로 쓴다고 가정.
- ▶ W와 b를 통해 cost의 값을 최소화하는 알고리즘 구하기

W값에 따른 cost 값 구하기

$$H(x) = Wx + b$$

$$\text{cost}(W,b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

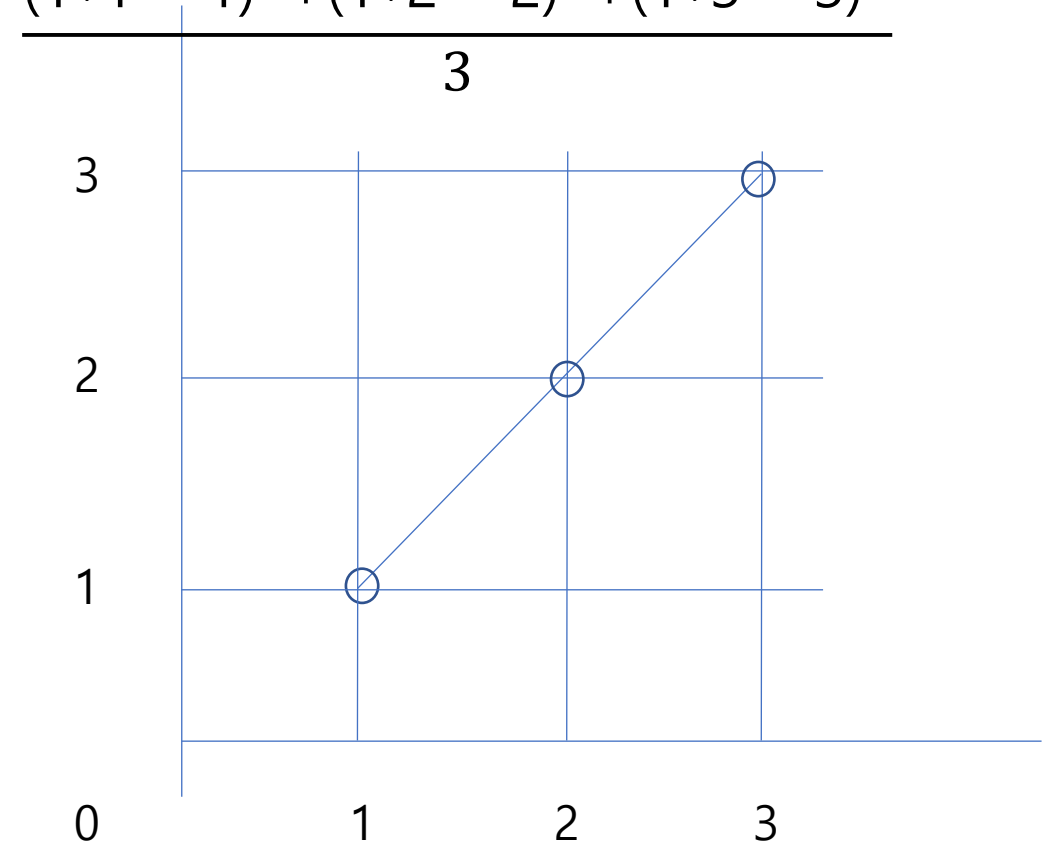
$$\text{Cost}(W,b) = \frac{1}{m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$

X	Y
1	1
2	2
3	3

$$H(x) = Wx \rightarrow H(x) = 1 * x$$

W가 1일때,

$$\frac{(1*1 - 1)^2 + (1*2 - 2)^2 + (1*3 - 3)^2}{3}$$



W값에 따른 cost 값 구하기

$$H(x) = Wx + b$$

$$\text{cost}(W,b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

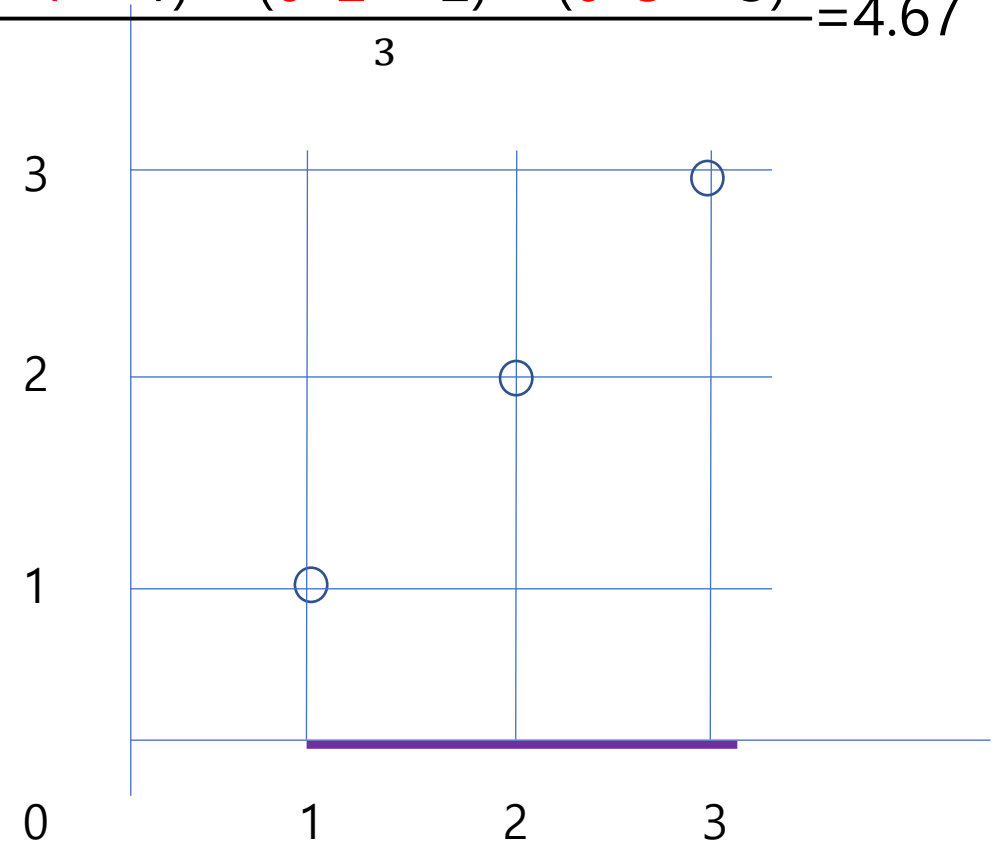
X	Y
1	1
2	2
3	3

$$\text{Cost}(W,b) = \frac{1}{m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx \rightarrow H(x) = 0 \cdot x$$

W가 0일때,

$$\frac{(0 \cdot 1 - 1)^2 + (0 \cdot 2 - 2)^2 + (0 \cdot 3 - 3)^2}{3} = 4.67$$



W값에 따른 cost 값 구하기

$$H(x) = Wx + b$$

$$\text{cost}(W,b) = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$

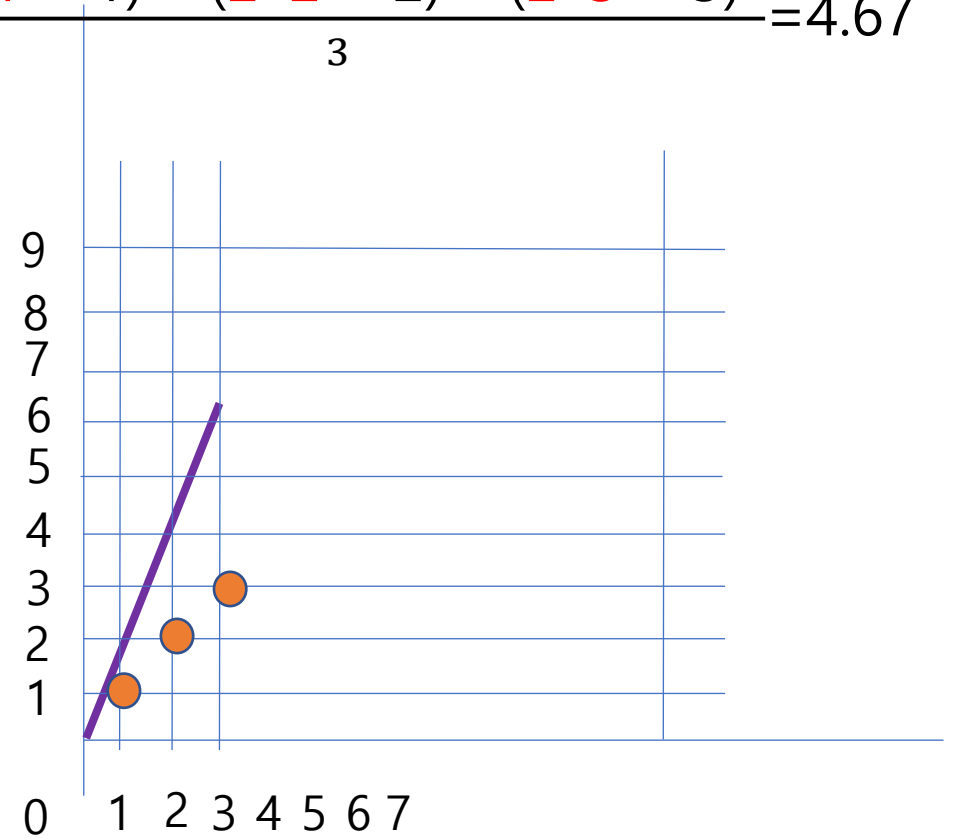
X	Y
1	1
2	2
3	3

$$\text{Cost}(W,b) = \frac{1}{m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx \rightarrow H(x) = 2 * x$$

W가 2일때,

$$\frac{(2*1 - 1)^2 + (2*2 - 2)^2 + (2*3 - 3)^2}{3} = 4.67$$



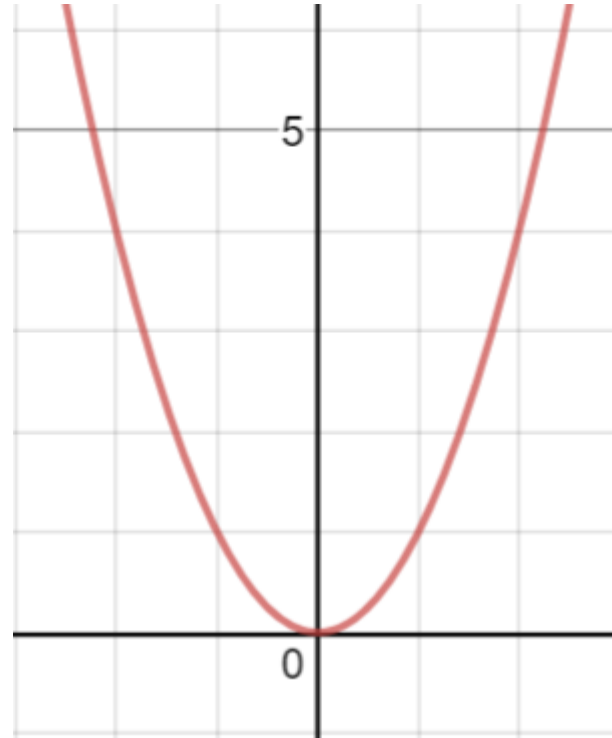
W값에 따른 cost 값 구하기

$$H(x) = Wx + b$$

W가 0일때, $\text{cost}(W)=4.67$

W가 1일때, $\text{cost}(W)=0$

W가 2일때, $\text{cost}(W)=4.67$



Loss function - binary crossentropy/log loss

▶ 이 함수는 분류 과제에 사용된다.

(예) 예를 들어 개와 고양이 분류에 사용된다.

$$\text{Loss} = -\frac{1}{m} \sum_{i=1}^m y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i)$$

y_i : 실제 y 값

\hat{y}_i : 예측 y 값

m : 샘플의 개수

Cost Function에 대해 알아보기

▶ Binary Cross Entropy - 공식을 이용하여 구할 수 있다.

$$\text{Log loss} = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)$$

p_i : 등급 1의 확률

$(1-p_i)$: 등급 0의 확률

Cost Function에 대해 알아보기

▶ Binary Cross Entropy

ID	Actual	Predicted probabilities
ID6	1	0.94
ID1	1	0.90
ID7	1	0.78
ID8	0	0.56
ID2	0	0.51
ID3	1	0.47
ID4	1	0.32
ID5	0	0.10

Cost Function에 대해 알아보기

▶ Binary Cross Entropy

ID	Actual	Predicted probabilities	Corrected Probabilities
ID6	1	0.94	0.94
ID1	1	0.90	0.90
ID7	1	0.78	0.78
ID8	0	0.56	0.44
ID2	0	0.51	0.49
ID3	1	0.47	0.47
ID4	1	0.32	0.32
ID5	0	0.10	0.90

* Corrected Probabilities는 Actual이 1일 때는 그대로 값이 전달

* Corrected Probabilities는 Actual이 0일 때는 (1- Predicted Probabilities)

Cost Function에 대해 알아보기

▶ Binary Cross Entropy

ID	Actual	Predicted probabilities	Corrected Probabilities	\log_{10}
ID6	1	0.94	0.94	-0.0268721464
ID1	1	0.90	0.90	-0.0457574905
ID7	1	0.78	0.78	-0.1079053973
ID8	0	0.56	0.44	-0.3565473235
ID2	0	0.51	0.49	-0.3098039199
ID3	1	0.47	0.47	-0.32790214206
ID4	1	0.32	0.32	-0.49485002168
ID5	0	0.10	0.90	-0.0457574905

* Corrected Probabilities는 Actual이 1일 때는 그대로 값이 전달

* Corrected Probabilities는 Actual이 0일 때는 (1 - Predicted Probabilities)

Cost Function에 대해 알아보기

▶ Binary Cross Entropy

ID	Actual	Predicted probabilities	Corrected Probabilities	\log_{10}
ID6	1	0.94	0.94	-0.0268721464
ID1	1	0.90	0.90	-0.0457574905
ID7	1	0.78	0.78	-0.1079053973
ID8	0	0.56	0.44	-0.3565473235
ID2	0	0.51	0.49	-0.3098039199
ID3	1	0.47	0.47	-0.32790214206
ID4	1	0.32	0.32	-0.49485002168
ID5	0	0.10	0.90	-0.0457574905

* 수정된 모든 확률에 대한 로그 값을 계산

* 보정된 모든 확률은 0과 1 사이에 있다.

* 모든 로그 값은 음수가 된다.

$$-\frac{1}{N} \sum_{i=1}^N (\log(p_i))$$

로그 손실 또는 이항 교차 엔트로피
0.214

Loss function – categorical crossentropy

▶ 이 함수는 분류 과제에 사용된다.

(예) 예를 들어 MNIST의 손 글씨 분류에 사용된다.

$$\text{Loss} = - \sum_{i=1}^m y_i \cdot \log \hat{y}_i$$

경사 하강 알고리즘

▶ 어떻게 비용 함수가 최소가 되는 W 파라미터를 구할 수 있을까?

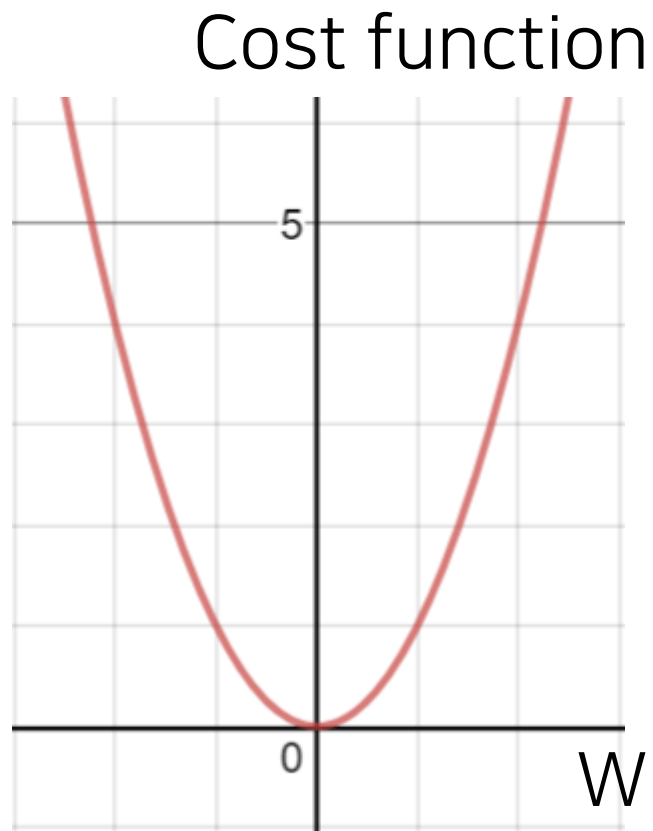
경사를 따라 내려가는 알고리즘 중의 하나이다.

가. Cost function을 최소화시키기

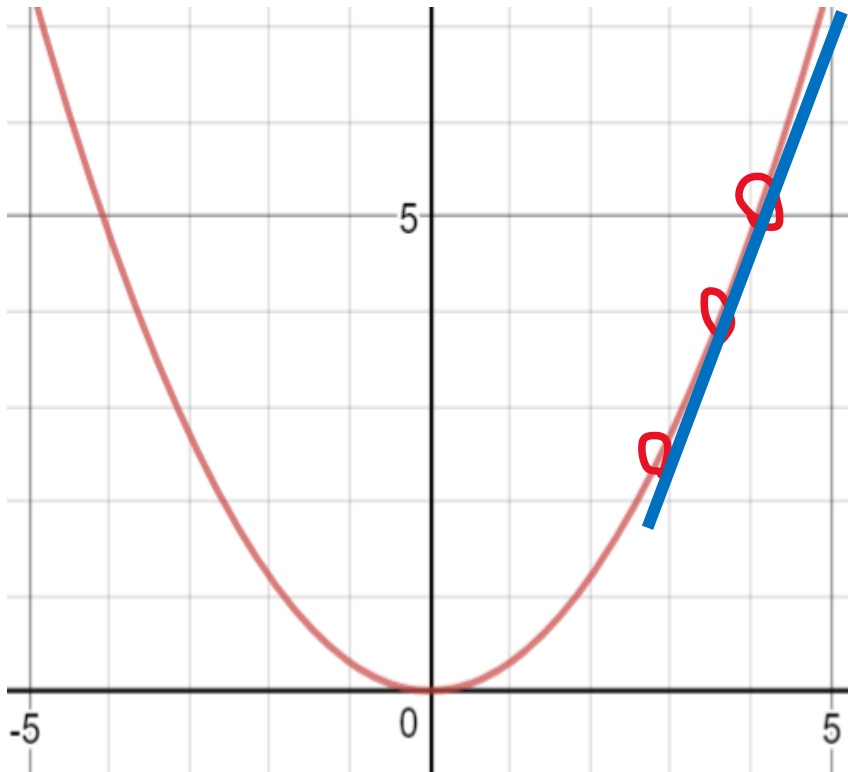
나. Gradient descent 는 여러 최소화 문제를 풀기 위해 사용된다.

다. Cost function이 주어지고, 이 알고리즘은 cost를 최소화시키는 W 와 b 를 찾을 것이다.

라. $W_1, W_2, W_3 \dots$



경사 하강 알고리즘(Gradient descent algorithm)



- ▶ 점진적으로 반복적인 계산을 통해 W파라미터 값을 업데이트
- ▶ 오류 값이 최소가 되도록 W 파라미터를 구하는 방식
- ▶ 예측값과 실제 값의 차이가 작아지는 방향성을 가지고 W파라미터를 보정해 간다.
- ▶ 오류 값이 더 작아지지 않으면 그 오류 값을 최소 비용으로 판단하고 그때의 W값을 최적 파라미터로 반환.
- ▶ “어떻게 하면 오류가 작아지는 방향으로 W값을 보정할 수 있을까?”
- ▶ 예를 들어, 비용함수가 그림과 같은 포물선 형태의 2차 함수라면 경사 하강법은 최초 w에서부터 미분을 적용한 뒤, 미분값을 감소시키는 방향으로 w값을 업데이트.

비용함수를 이용해서 어떻게 w값을 업데이트 할까?

$$\text{Cost}(W) = \frac{1}{m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$



$$\text{Cost}(W) = \frac{1}{2m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$

★ 앞의 1/m이나 1/2m 같은 의미를 지닌다.

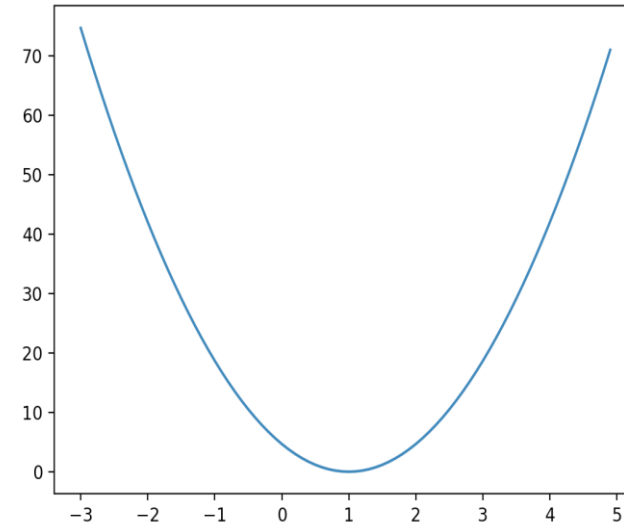
비용함수를 이용해서 어떻게 w값을 업데이트 할까?

$$\text{Cost}(W) = \frac{1}{2m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$



Cost 함수를 미분하면 기울기를 구할 수 있다.

$$W := W - \alpha \frac{\partial}{\partial W} \text{cost}(W)$$



- ▶ W가 최소지점의 영역의 오른쪽에 있으면 -방향으로
- ▶ W가 최소지점의 왼쪽에 있으면 + 방향으로 움직이기 위해 -를 붙여준다.

비용함수를 이용해서 어떻게 w값을 업데이트 할까?

$$\text{Cost}(W) = \frac{1}{2m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$

$$W := W - \alpha \frac{\partial}{\partial W} \text{cost}(W)$$



$$W := W - \alpha \frac{\partial}{\partial W} \frac{1}{2m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$$

$$W := W - \alpha \frac{1}{2m} \sum_{i=1}^m 2(Wx^{(i)}) - y^{(i)})x^{(i)}$$

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)}) - y^{(i)})x^{(i)}$$

비용함수를 이용해서 어떻게 w값을 업데이트 할까?

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})x^{(i)}$$

Learning rate

데이터 개수

★ 우리는 위와 같이 Gradient descent 알고리즘이 수식으로 나오고, 우리는 이를 기계적으로 적용만 시키면 이 Cost Function을 최소화하는 W를 구해내고, 이것이 바로 linear regression인 학습과정을 통해서 모델을 만든다고 할 수 있다.

정리해 보기

- **Hypothesis** $H(x) = Wx + b$

- **Cost function** $\text{Cost}(W,b) = \frac{1}{m} \sum_{i=1}^m ((Wx^{(i)}) - y^{(i)})^2$

- **Gradient descent algorithm**

