

AI and Deep Learning

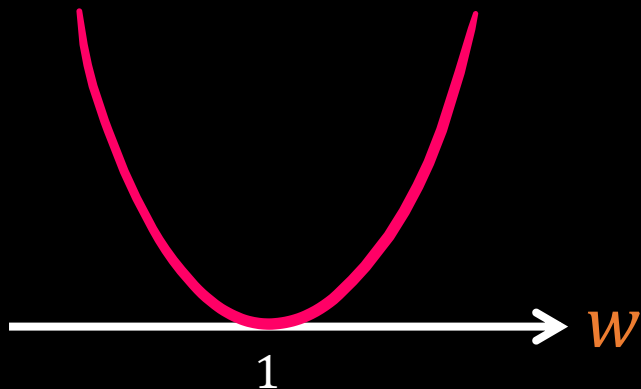
# 선형회귀

오류 계산 그래프에서의 역전파

제주대학교

변영철

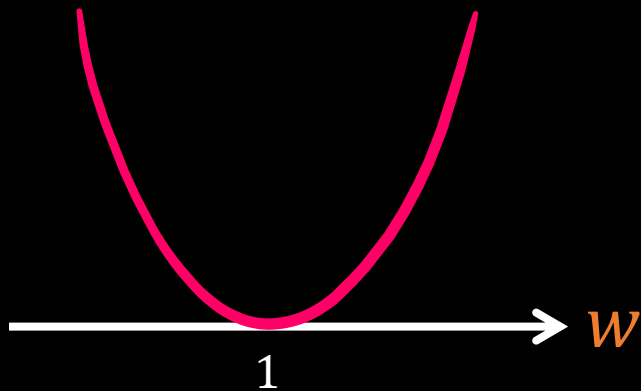
<http://github.com/yungbyun/mllecture>



오류 그래프  $E$ 의  $w$  지점에서의 기울기

$w$  변화가  $E$ 에 미치는 영향

이를 수학 공식으로 표현해보라.



$$\lim_{\Delta w \rightarrow 0} \frac{\Delta E}{\Delta w}$$

$$= \frac{\partial E}{\partial w}$$

# 학습 방법( $w$ 업데이트)

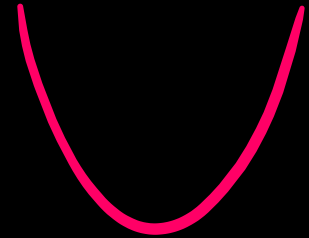


- 시냅스 연결강도 값( $w$ )을 난수 초기화 (ex, 4)
- $w$  값 지점(4)에서 오류 그래프의 기울기(경사) 구함.
- 기울기로  $w$ 를 뺌(경사하강)

$$w = w - \alpha * \text{기울기}$$

반영 비율 (learning rate)

# TensorFlow



- 텐서(Tensor) : ‘잡아당기다’라는 라틴어 ‘tensus’
- 무언가를 잡아당기면 그 주위에 복잡한 변형이 일어나는데, 이를 **기술하는 수학적 언어**가 텐서
- 오류  $E$ 가 줄어들도록 아래로 잡아당기면 그에 따라  $w$ 값이 업데이트 (파라미터 튜닝),  $w$ 가 많을 경우 모든 값에 대해 복잡한 변형이 일어남을 의미
- 텐서는 텐서플로우 내부에서 만들어지는 값(난수),  $w$ 와 같이 텐서 값을 담는 데이터(변수, 벡터, 행렬 형태 등)도 텐서라고 함. 텐서와 연산자로 정의되는 수식도 텐서  $E = (w \cdot 1 - 1)^2$
- TensorFlow는 텐서(Tensor, 데이터)의 흐름(Flow)이며, 오류 계산 그래프의 텐서 흐름을 통해  $w$  파라미터를 튜닝해주는 머신러닝 (응용개발) 프레임워크

# 텐서플로우를 이용한 문제

1시간 일하면 1만원 줄게.

일한 시간만큼 시급을 알아 맞추는 뉴런을  
코딩하시오.

올바르게 대답하도록 뉴런의 시냅스 연결강도( $w$ )를  
업데이트하는 코드를 작성하시오.

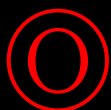
01.py

Finding  $w$  in  
linear regression



- 구글이 제공하는 “주피터 노트북”
- 구글 클라우드를 이용한 머신러닝 응용 개발
- 텐서플로우, 파이썬 등 이미 준비가 되어 있어서 따로 설치할 필요없음
- 구글 계정 생성 및 로그인 후 아래 페이지 방문

<http://colab.research.google.com/>

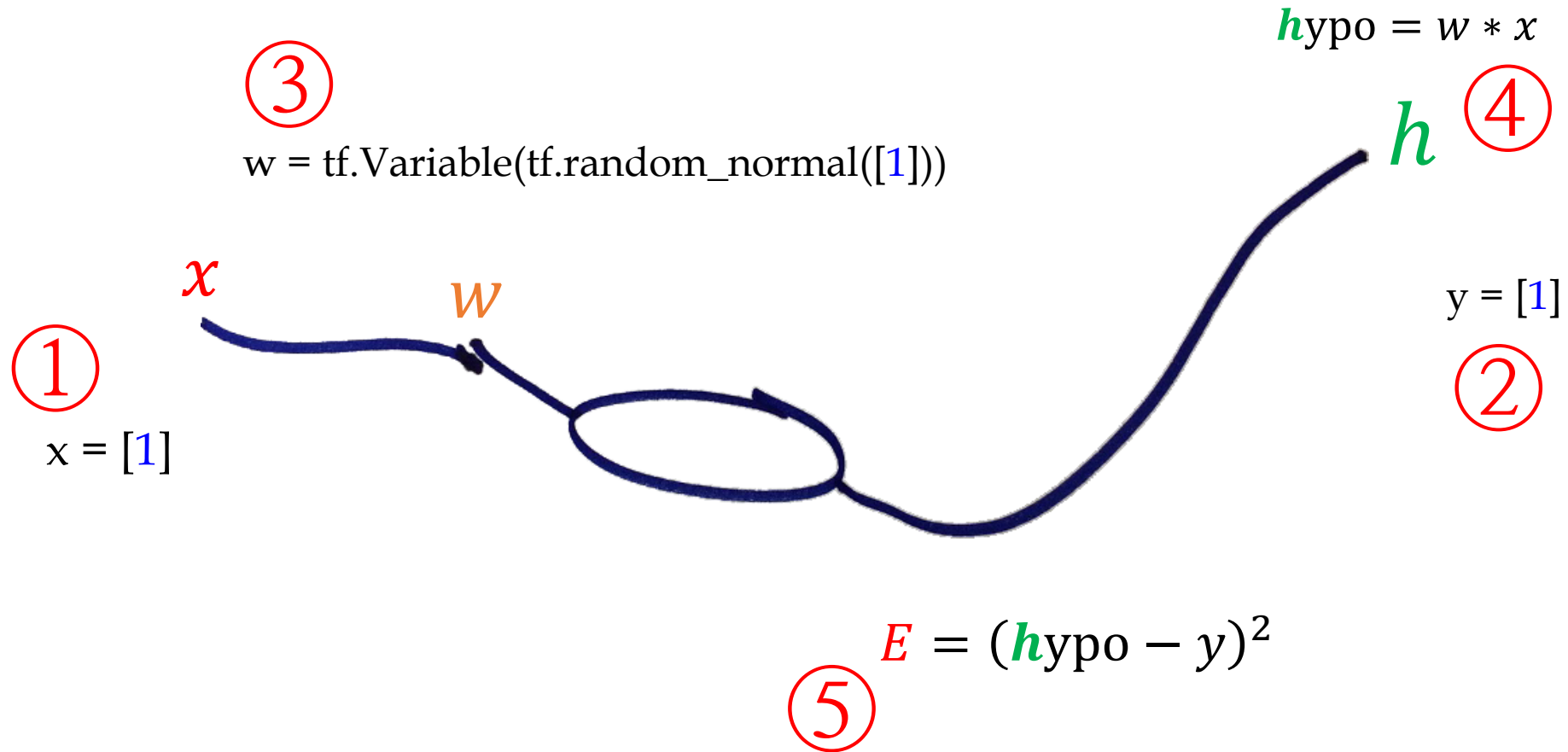


항상 작성해야 하는 기본 코드

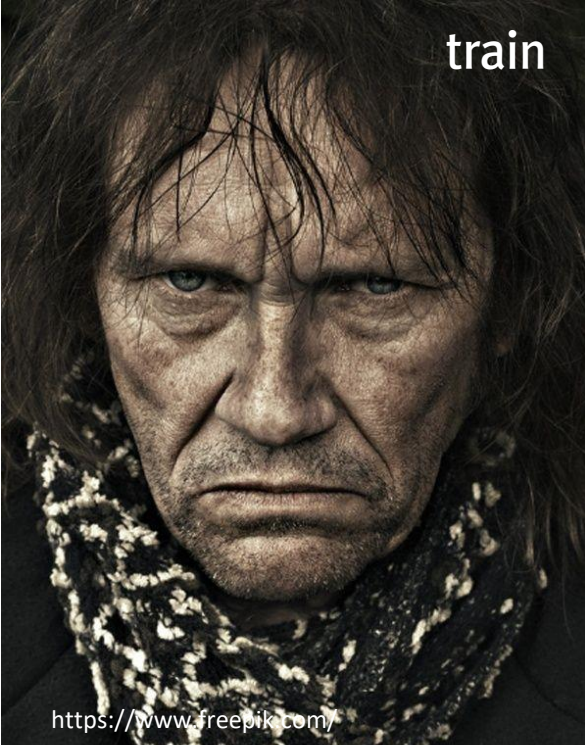
```
%tensorflow_version 1.x  
import tensorflow as tf
```



# TF를 이용한 선형 회귀 학습



이제 남은 코드는?



<https://www.freepik.com/>



<https://www.freepik.com/>

```
train = tf.train.GradientDescentOptimizer(learning_rate  
=0.1).minimize(E)
```

```
sess = tf.Session()
```

```
sess.run(tf.global_variables_initializer()) #w값 초기화
```

```
err_list = []
```

```
for i in range(101):
```

```
    w_val = sess.run(w)
```

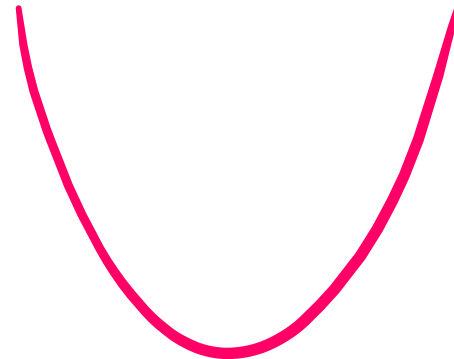
```
    err = sess.run(E)
```

```
    print(i, 'w:', w_val, 'cost:', err)
```

```
    err_list.append(err)
```

```
    sess.run(train) #한번 경사하강(w 업데이트)
```

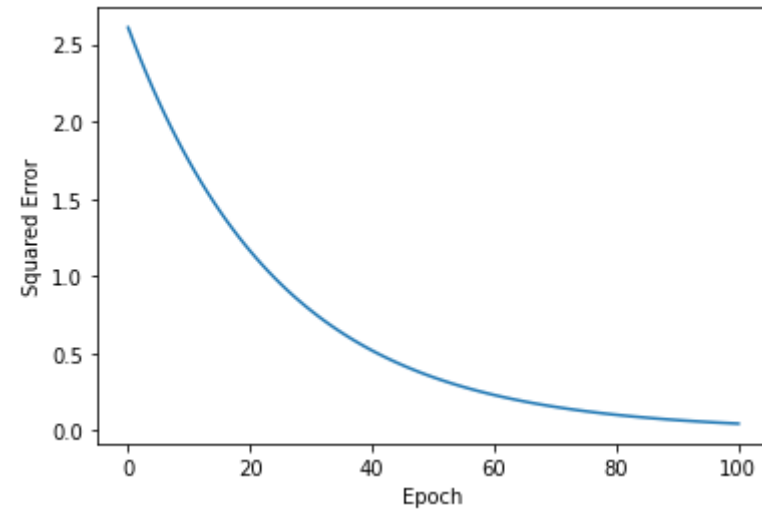
학습이 끝났다. 남은 것은?



6

⑦ `# ---- test(prediction)`  
`print(sess.run(w * [3, 4, 6, 9]))`

⑧ `# show an error graph`  
`import matplotlib.pyplot as plt`  
`plt.plot(err_list)`  
`plt.xlabel('Epoch')`  
`plt.ylabel('Square Error')`  
`plt.show();`



```
import tensorflow as tf
```

```
#----- 학습 데이터 설정
```

```
x = [1]
```

```
y = [1]
```

- import : 라이브러리 모듈을 불러오고, 필요할 경우 초기화도 수행함. 어떤 이름으로 사용될지 지정함(as tf).
- x: 입력 데이터를 보관하는 리스트
- y: 정답을 보관하는 리스트

```
xxx = [1, 'hello', 2, 'world']  
print (xxx)  
xxx[1] = 'hi'  
xxx.append('Jeju')  
print (xxx)
```

#----- 신경세포 만들기

$w = \text{tf.Variable}(\text{tf.random\_normal}([1]))$

$h = w * x$

- random\_normal: 난수 텐서를 생성하여 반환하는 함수
- Variable: 텐서를 담는 그릇(변수)  $w$  을 만듦. 난수 텐서를 담는 그릇  $w$  도 텐서
- $h$ : 입력 데이터( $x$ )와  $w$  텐서를 곱한 것. 이것도 텐서

#----- 학습 시키기

$$E = (h - y) ** 2$$

```
train = tf.train.GradientDescentOptimizer  
(learning_rate=0.1).minimize(E)
```



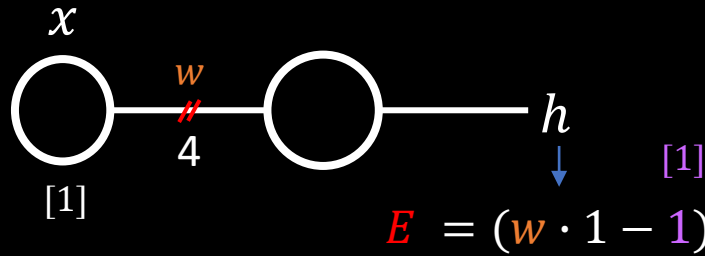
- **E** : 경사하강할 오류/에러/차이/로스(loss) 함수. 텐서 **h**가 들어있는 **E**도 텐서
- GradientDescentOptimizer : 경사하강 알고리즘을 추상화한 클래스
- train 객체: 오류 그래프 **E**에서 경사하강하도록 **w** 업데이트 (학습)
- **E** 는 계산 그래프(computational graph)로 구성됨.

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())
for i in range(101):
    print(sess.run( $w$ ), sess.run( $E$ ))
    sess.run(train)
```



- Session: run 함수로 텐서를 평가(eval)하여 값을 구하기도 하고, train 객체에게 경사하강 하도록 시키기도 하는 객체
- sess.run(tf.global\_variables\_initializer()) global\_variables\_initializer 함수가 반환하는 변수 초기화 오퍼레이션으로 변수들을 초기화
- for 루프: i를 0에서 100까지 (101번) 바꾸면서 반복문을 수행
- sess.run(train): sess가 train에게 한번 경사하강하도록 일을 시킴.  
 $w = w - \alpha \cdot \text{기울기}$
- 101번 경사하강이 끝나면 업데이트된  $w$ 가 최종 학습결과

시냅스 초기값 난수 4, 학습상수 0.1일 때



#----- training data

x\_data = [1]

y\_data = [1]

#----- a neuron

w = tf.Variable(tf.random\_normal([1]))

h = w \* x\_data

$E = (h - y) ** 2$

기울기 =  $2(w \cdot 1 - 1)^1$

for 루프  
101번

$w = 4$	$(4 - 1)^2$	기울기 = 6	$w = 4 - 0.1 \cdot 6$	→ 3.4
$w = 3.4$	$(3.4 - 1)^2$	기울기 = 4.8	$w = 3.4 - 0.1 \cdot 4.8$	→ 2.92
$w = 2.92$	$(2.92 - 1)^2$	기울기 = 3.84	$w = 2.92 - 0.1 \cdot 3.84$	→ 2.53
$w = 2.53$	$(2.53 - 1)^2$	기울기 = 3.06	$w = 2.53 - 0.1 \cdot 3.06$	→ 2.22



#----- 학습 후 테스트하기

```
x = [2]
```

```
print(sess.run(x * w))
```

- 학습이 끝났을 때 결과는 무엇? 바로 여러 번 반복해서 업데이트된  $w$
- 뉴런의 출력은? 학습된  $w$  값  $\times$  입력( $x$ )

#----- 학습 후 테스트하기

```
x = [2]
```

```
print(sess.run(h))
```

- 텐서  $h = w \cdot x$  가 처음 (계산 그래프로) 만들어질 때 딱 한번만  $x$ 에 값이 설정되며, 이후에는  $x$ 에 새로운 값을 넣어도 계산 그래프에 반영되지 않음.
- 따라서 이전에  $x$ 에 할당된 값 [1]이 여전히 사용됨.
- 따라서 우리가 원하는 값을 넣는 방법이 필요함.
- 이를 위한 것이 **Placeholder**(자리표시기) 방법임.

02.py

Placeholder

자리표시기

$x \rightarrow x\_data$

$y \rightarrow y\_data$

$h \rightarrow hypo$

$E \rightarrow cost$

# 소스코드 다운로드

- 1) 명령 프롬프트 실행
- 2) 명령 프롬프트에서 코드를 저장하고자 하는  
폴더(C:\)로 이동
- 3) `git clone https://github.com/yungbyun/myml.git`

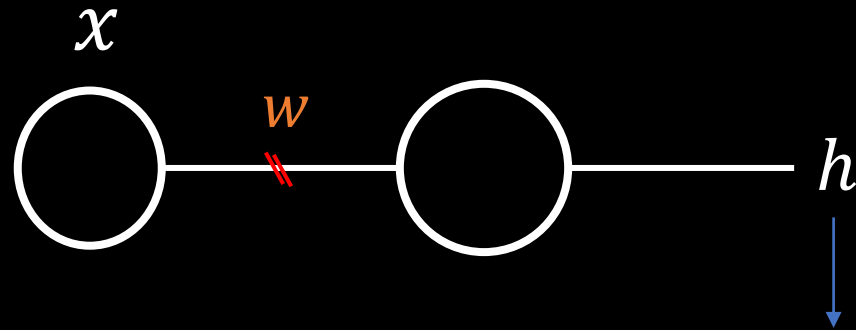
03.py

Drawing  
cost function

# 오류 함수 생각하기

$$E = (wx - y)^2$$

- 어느 부분이 뉴런인가?
- 시냅스는?
- 입력 데이터는?
- 뉴런의 출력(대답)
- 가설(hypothesis)은?
- 정답(ground truth)은?
- 오류 함수  $E$ 의 의미는?
- 뉴런 입력이 여러 개일 경우
- 데이터로 주어지는 것은(csv 파일)?
- 우리가 튜닝해야 하는 것은?



$$E = |h - y|$$

$$E = (h - y)^2$$

$$E = (w \cdot x - y)^2$$

$$E = (w \cdot 1 - 1)^2$$

Mean Square Error  
평균 제곱 오류 함수

$$E = \frac{1}{N} \sum_{i=1}^N (w x_i - y_i)^2$$



오류 함수  $E$

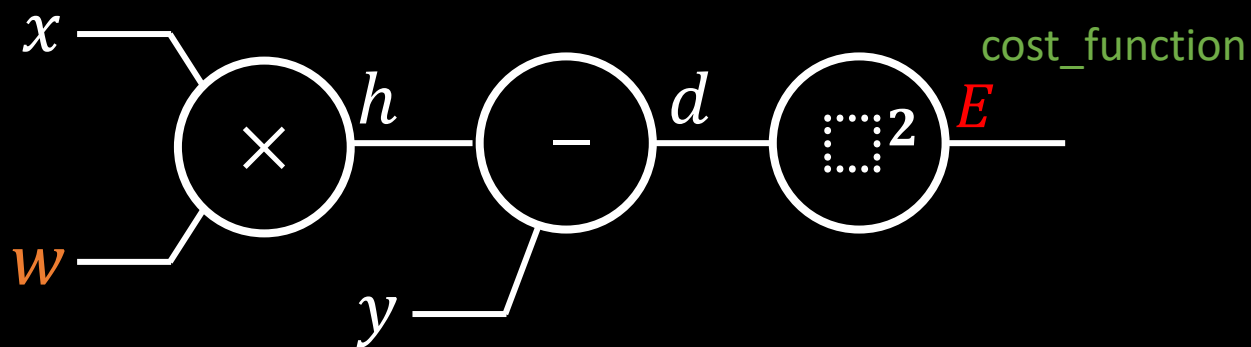
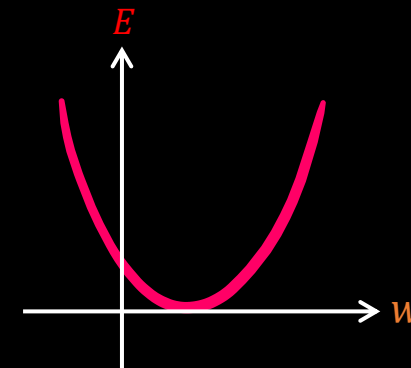
오류 계산 그래프  
(computational graph)

# 오류 계산 그래프

$$E = (wx - y)^2$$

hypo = w \* x

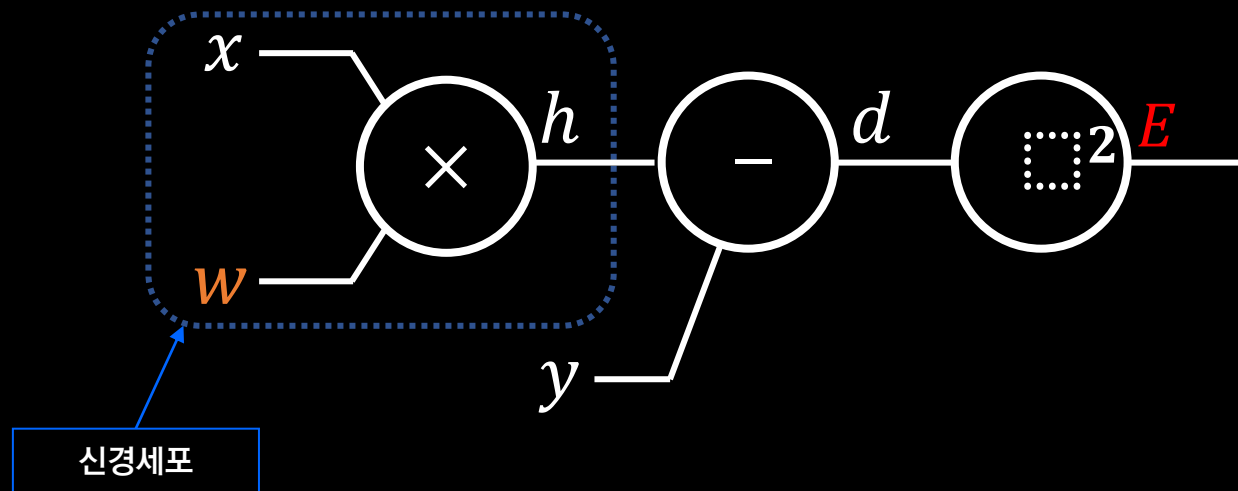
cost\_function(E) = (hypo - y) \*\* 2



텐서란 무엇이고, 텐서 플로우란 무엇인가?  
텐서플로우 (응용개발) 프레임워크가 파라미터 튜닝

# 오류 계산 그래프와 미치는 영향

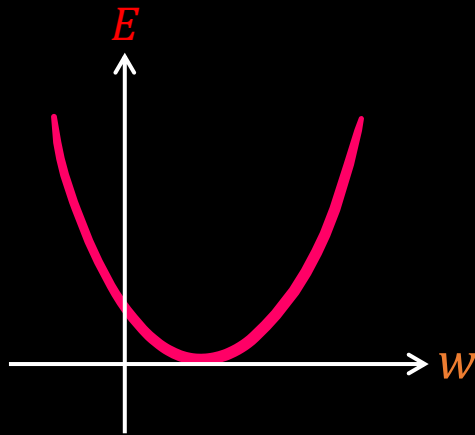
$w$ 의 변화가  $E$ 에 미치는 영향(기울기)을 구한 후  
경사하강하도록  $w$ 를 조절



$w$ 의 변화가  $E$ 에 미치는 영향  
 $E$ 함수에서  $w$ 위치에서의 기울기

“수학공식으로 표현해보라.”

# 미치는 영향과 기울기



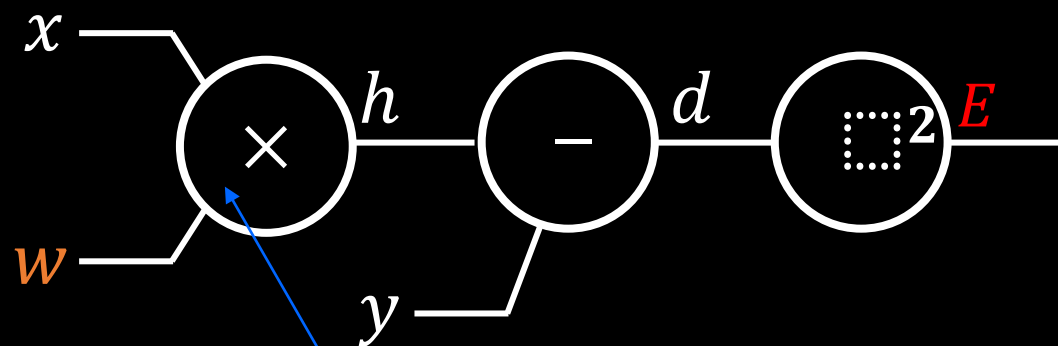
$$\lim_{\Delta w \rightarrow 0} \frac{\Delta E}{\Delta w}$$

$$= \frac{\partial E}{\partial w}$$

$$\lim_{\Delta \text{일} \rightarrow 0} \frac{\Delta \text{스트레스}}{\Delta \text{일}}$$

$$= \frac{\partial \text{스트레스}}{\partial \text{일}}$$

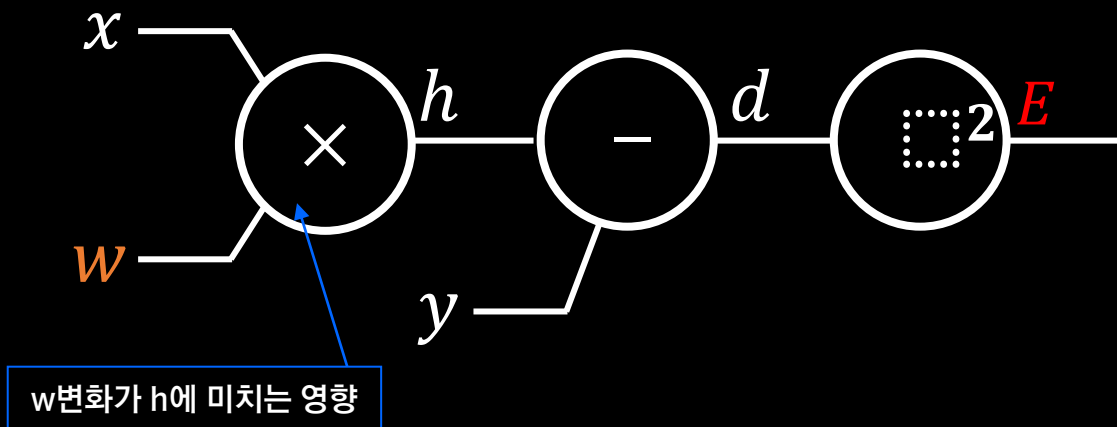
# 오류 계산 그래프와 미치는 영향



$w:1 \rightarrow h:1x$   
 $w:2 \rightarrow h:2x$

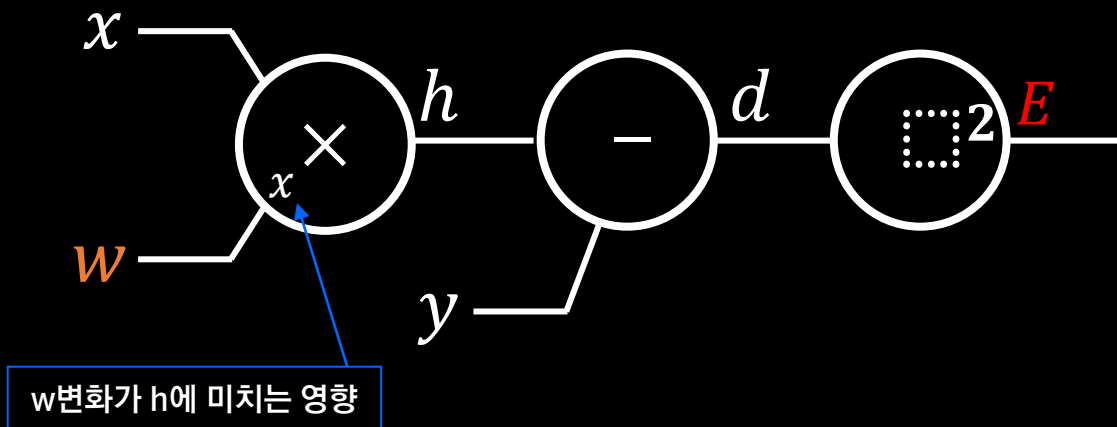
$$\frac{\Delta h}{\Delta w} = \frac{x}{1}$$

# 오류 계산 그래프와 미치는 영향

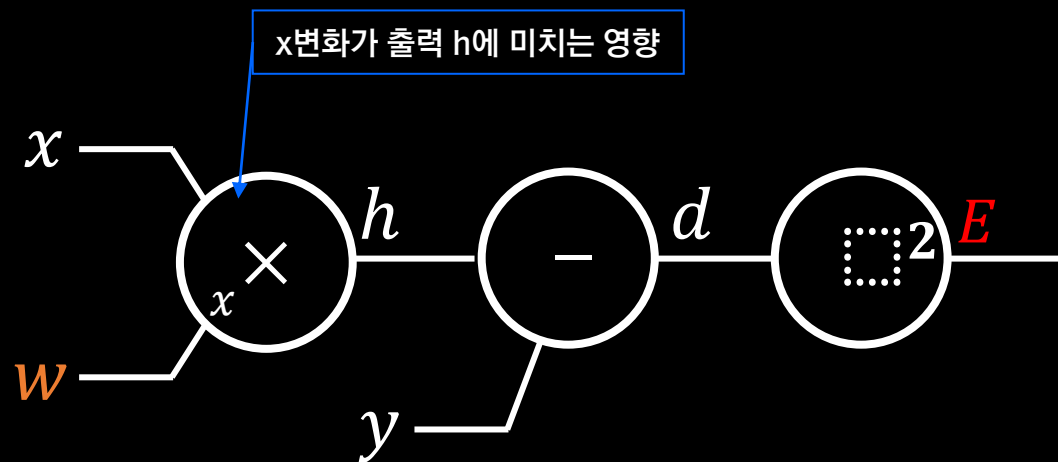




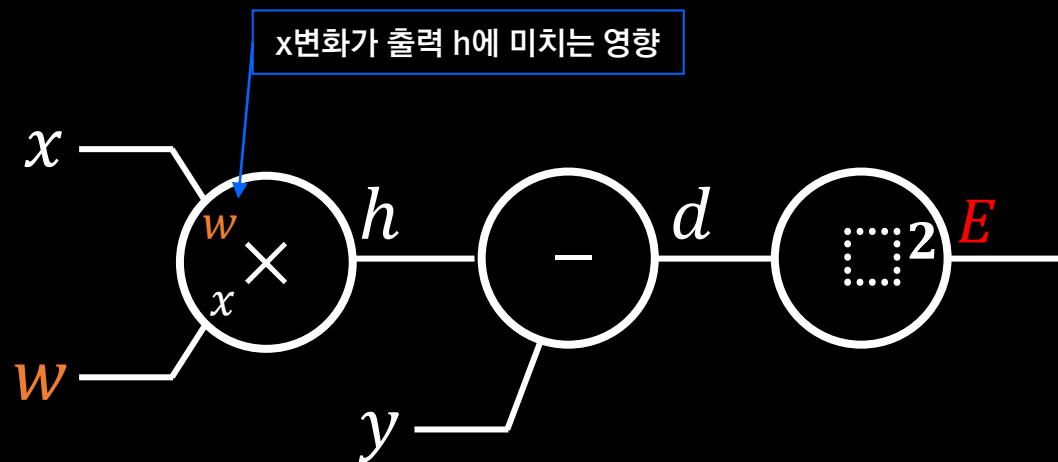
# 오류 계산 그래프와 미치는 영향



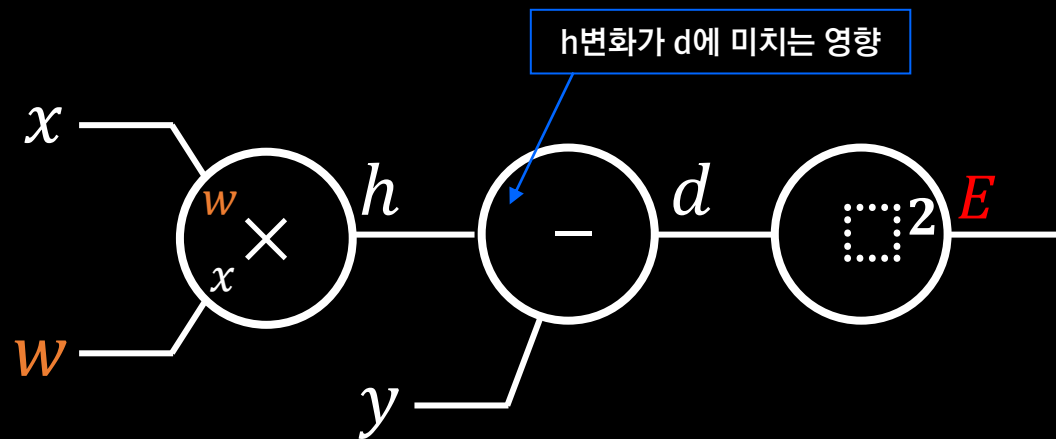
# 오류 계산 그래프와 미치는 영향



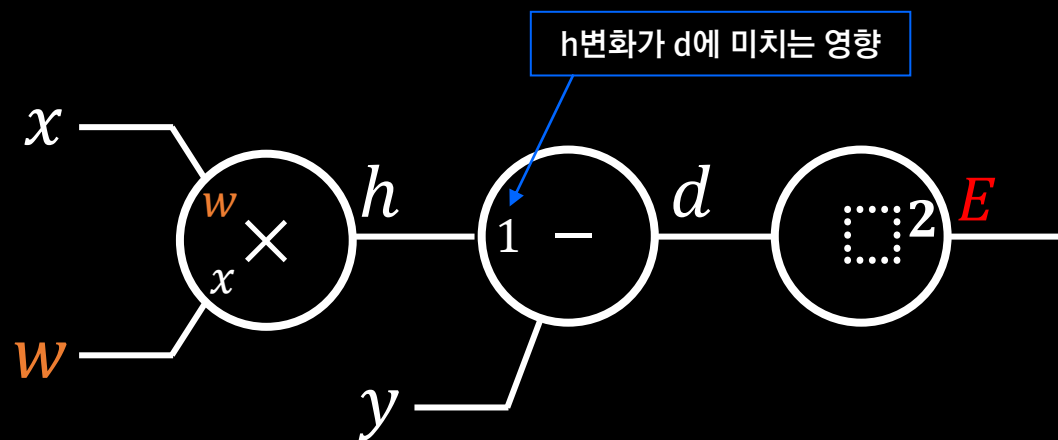
# 오류 계산 그래프와 미치는 영향



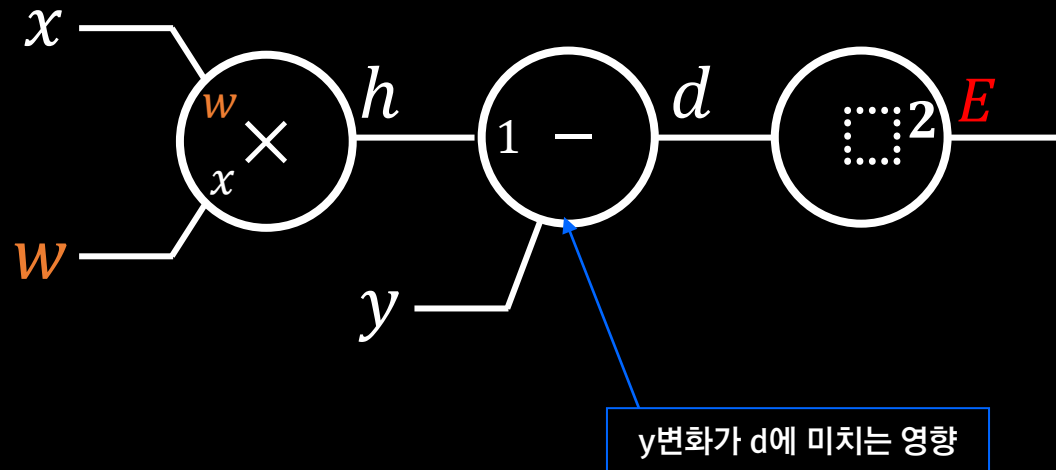
# 오류 계산 그래프와 미치는 영향



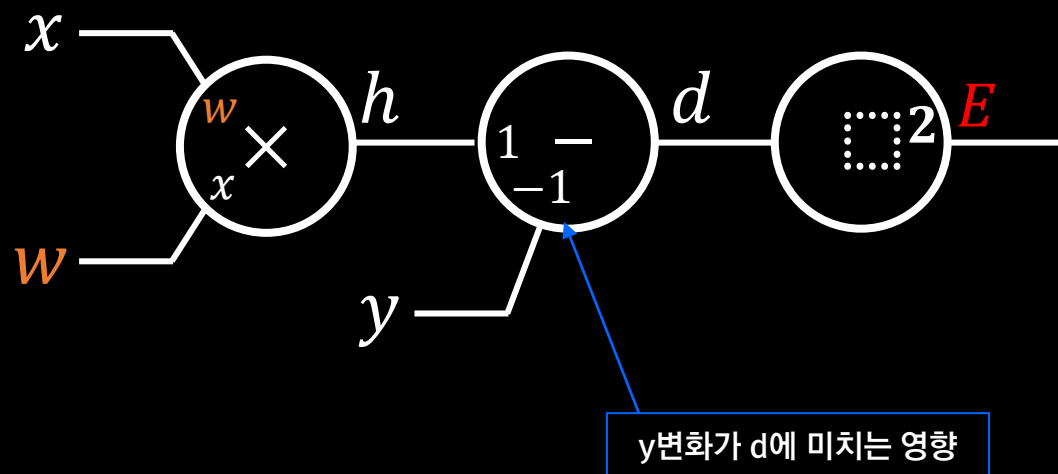
# 오류 계산 그래프와 미치는 영향



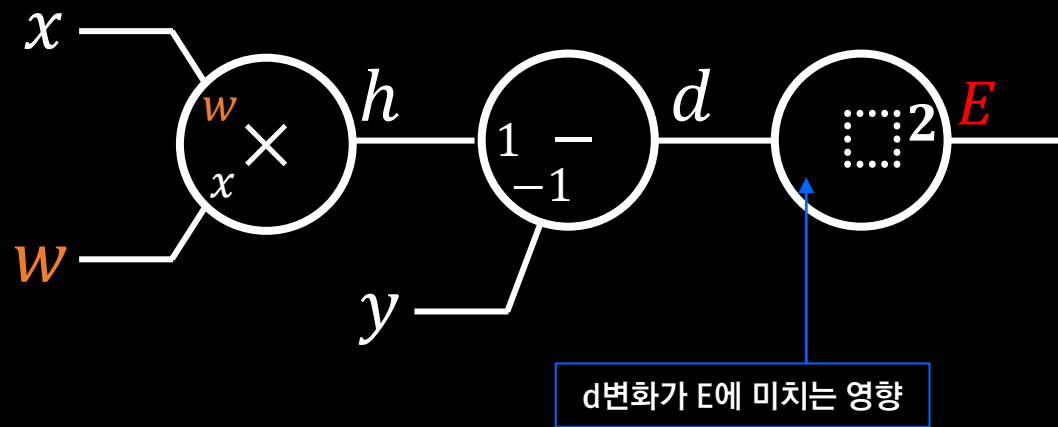
# 오류 계산 그래프와 미치는 영향



# 오류 계산 그래프와 미치는 영향

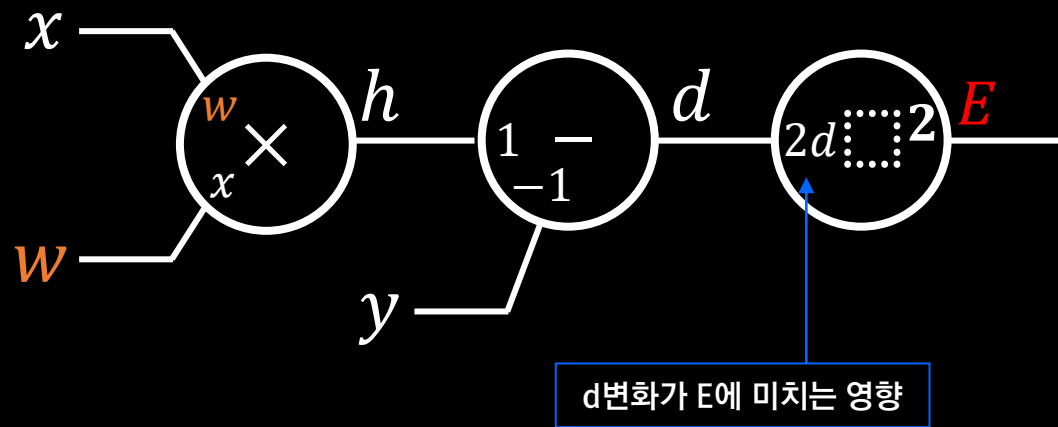


# 오류 계산 그래프와 미치는 영향



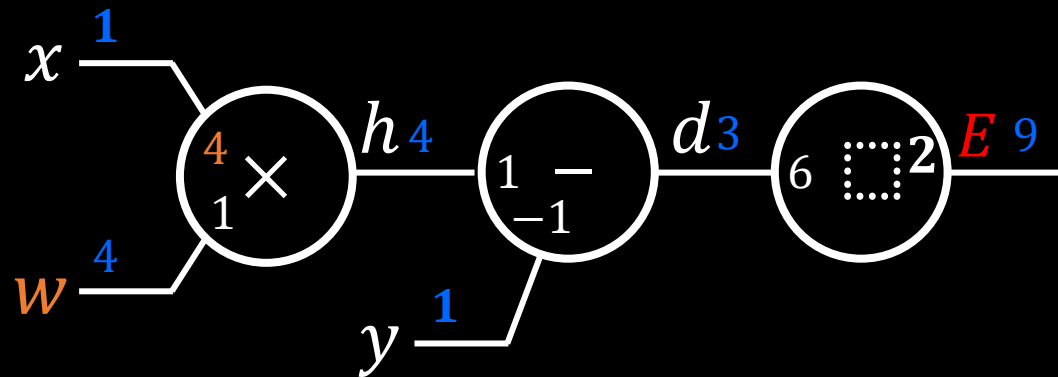


# 오류 계산 그래프와 미치는 영향

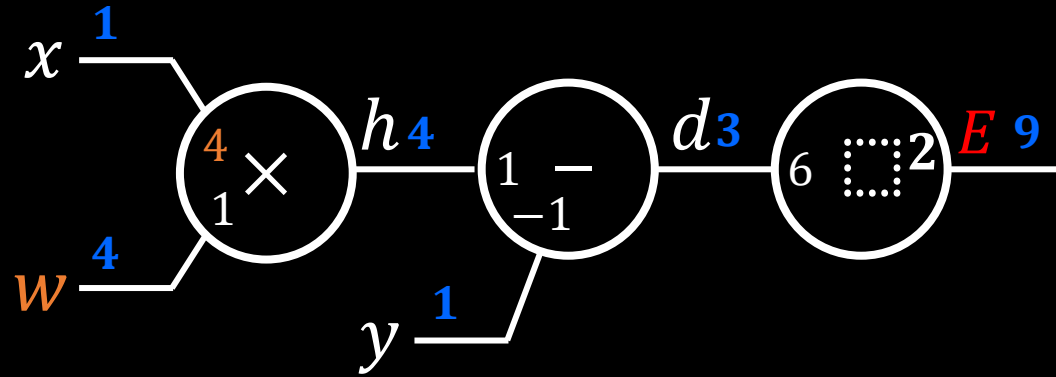


# 오류 계산 그래프와 미치는 영향

데이터  $(x, y) = (1, 1)$ 이고,  $w$  초기값이 4일 때



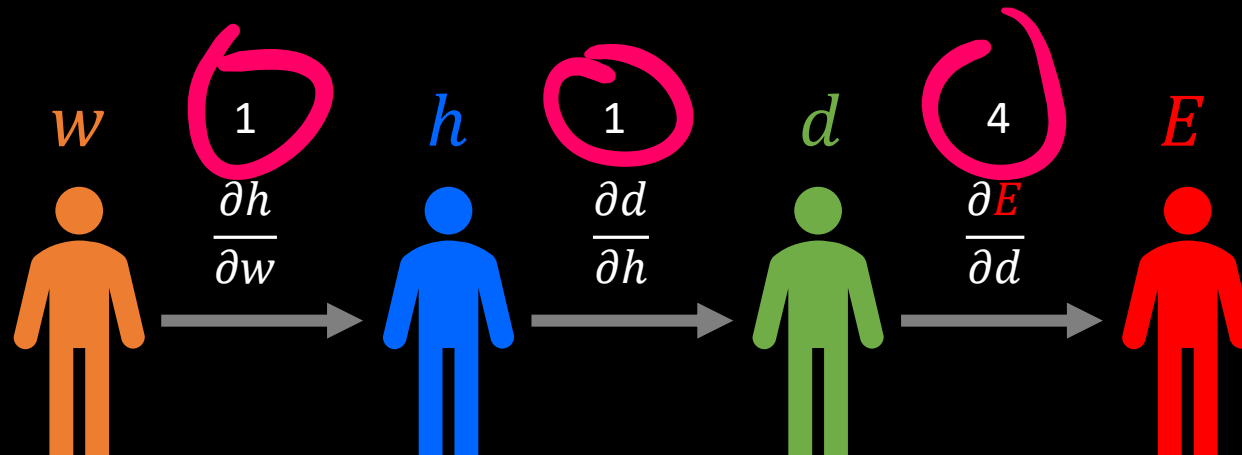
텐서플로 (TensorFlow)!



각 게이트(연산) 별  
지역적으로(local) 미치는 영향은 알았다.

그러면  $w$  변화가  $E$ 에 미치는 영향력은  
어떻게 알 수 있을까?

# 사람 사이의 영향력



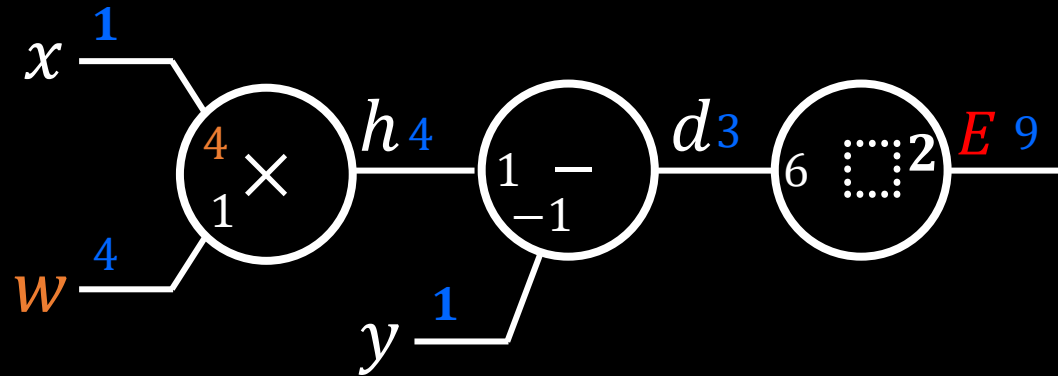
$$\frac{\partial E}{\partial w}$$

$$\frac{\partial E}{\partial w} = \frac{\partial h}{\partial w} \cdot \frac{\partial d}{\partial h} \cdot \frac{\partial E}{\partial d} = 1 \cdot 1 \cdot 4 = 4$$

체인 룰(chain rule)

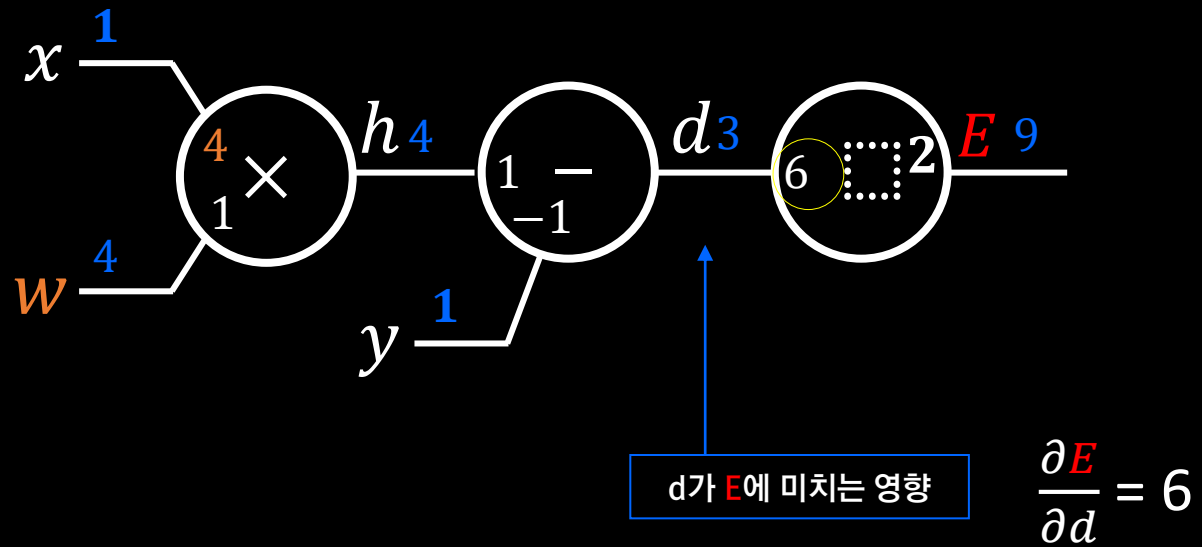
# 앞으로 전파(forward propagation)

$(x, y) = (1, 1)$ 이고  $w$ 는 4일 경우 **에러** 값은?

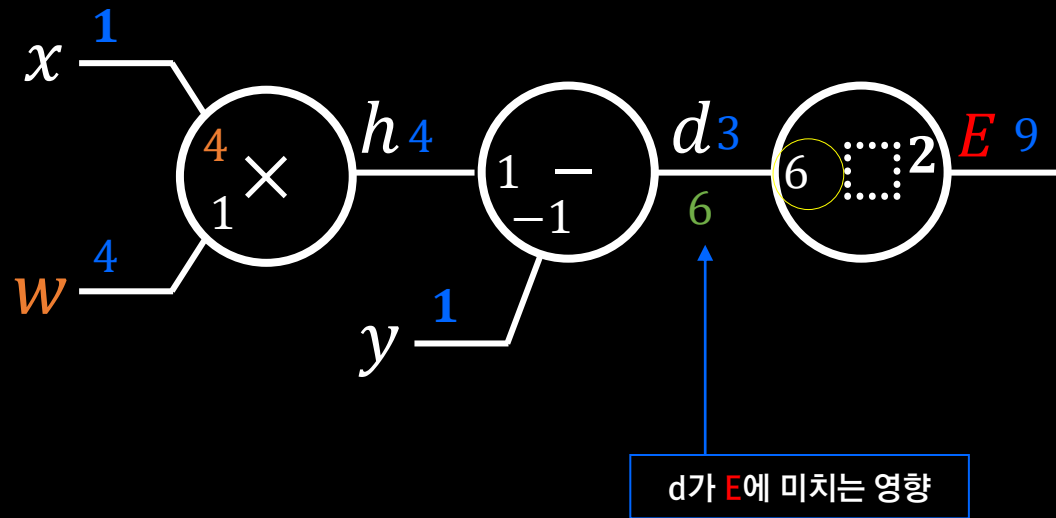


$w$ 변화가  $E$ 에 미치는 영향(기울기)는 어떻게 구할 수 있을까?  $\frac{\partial E}{\partial w}$   
 $w = w - \alpha * \text{기울기}$

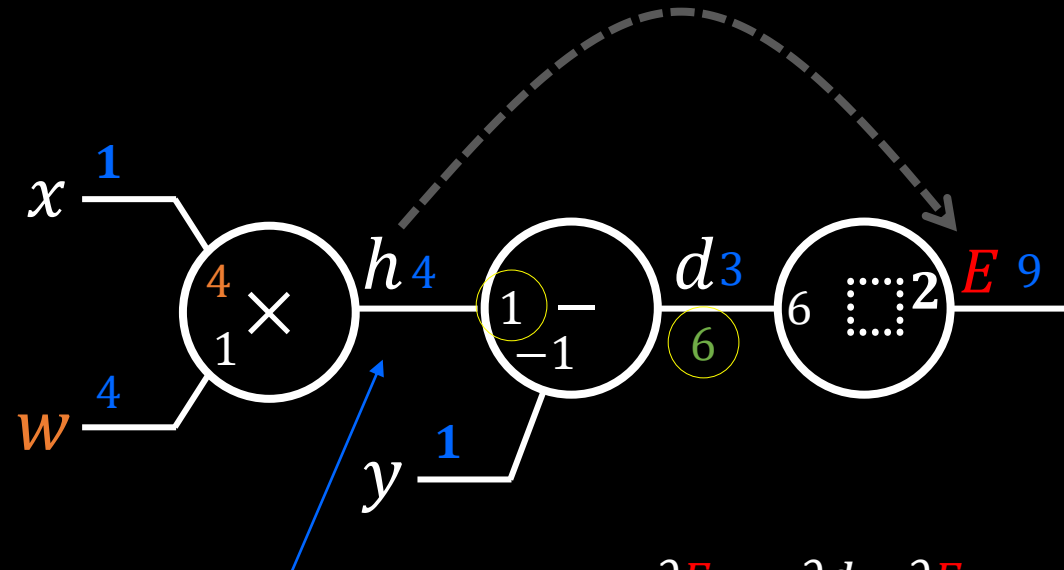
# 역전파(back-propagation)와 체인룰



# 역전파(back-propagation)와 체인룰



# 역전파(back-propagation)와 체인룰

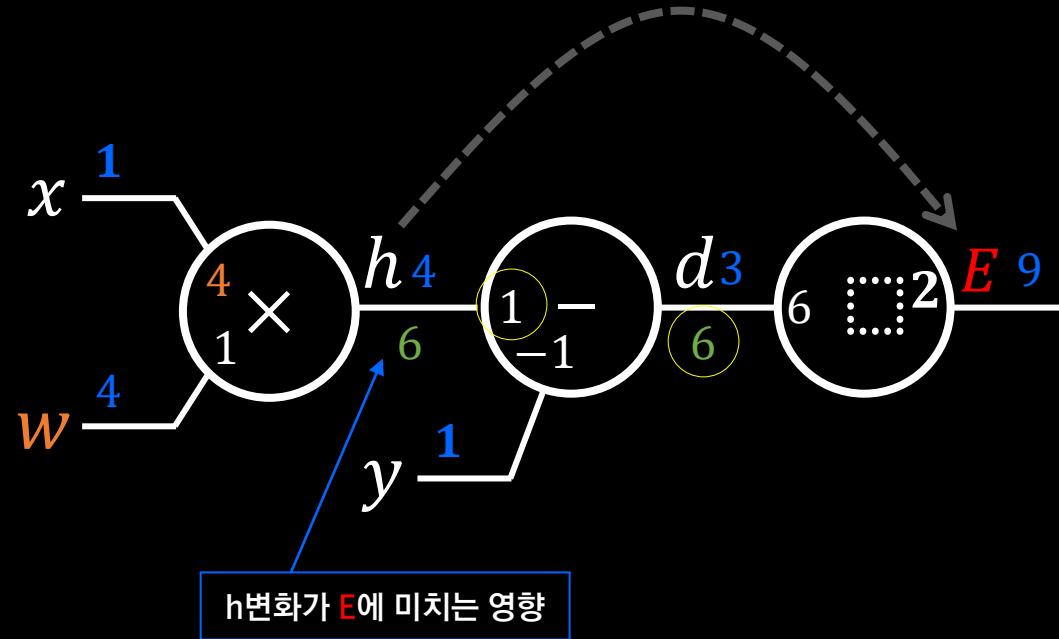


h변화가 E에 미치는 영향

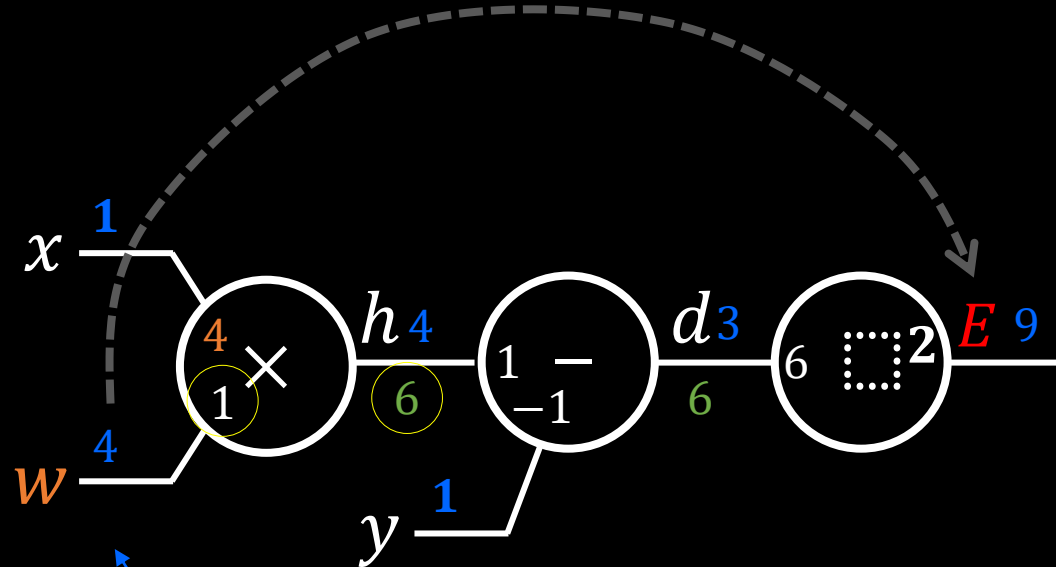
$$\frac{\partial E}{\partial h} = \frac{\partial d}{\partial h} \cdot \frac{\partial E}{\partial d} = 1 \cdot 6$$



# 역전파(back-propagation)와 체인룰



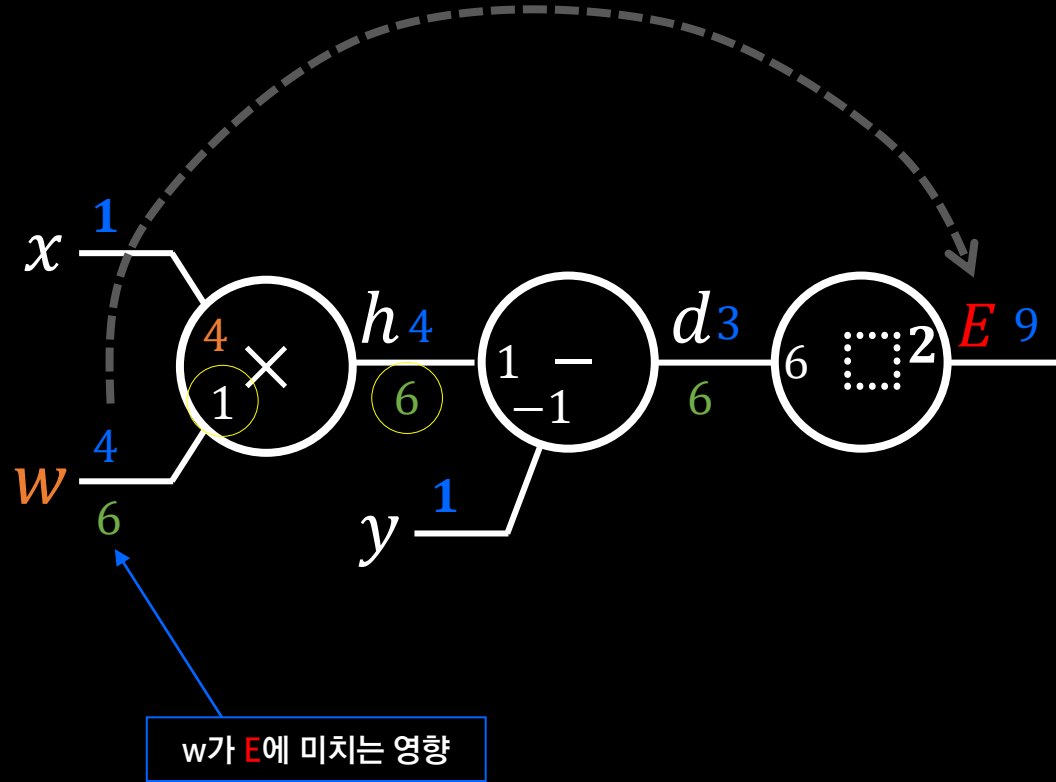
# 역전파(back-propagation)와 체인룰



w가 E에 미치는 영향

$$\frac{\partial E}{\partial w} = \frac{\partial h}{\partial w} \cdot \frac{\partial E}{\partial h} = 1 \cdot 6$$

# 역전파(back-propagation)와 체인룰



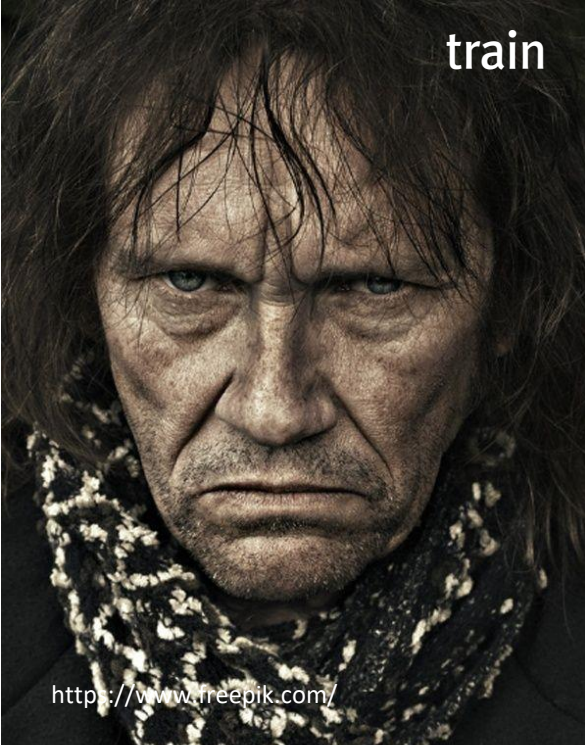
# 따라서 역전파 (back-propagation)

체인 룰(chain rule)을 적용하여  
 $w$ 의 변화가 오류  $E$ 에 얼마나 영향을  
미치는지(기울기)를 알아내는 과정

$$\frac{\partial E}{\partial w}$$

$$w = 4 - 0.1 * 6 \frac{\partial E}{\partial w}$$
$$w = 3.4$$

Tuned parameter  
after 1 step learning.



train



sess

⑥

```
train = tf.train.GradientDescentOptimizer(learning_rate  
=0.1).minimize(E)
```

```
sess = tf.Session()
```

```
sess.run(tf.global_variables_initializer()) #w값 초기화
```

```
err_list = []
```

```
for i in range(101):
```

```
    w_val = sess.run(w)
```

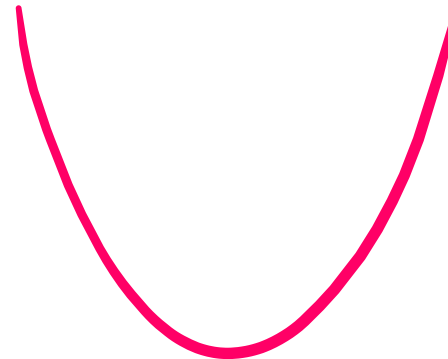
```
    err = sess.run(E)
```

```
    print(i, 'w:', w_val, 'cost:', err)
```

```
    err_list.append(err)
```

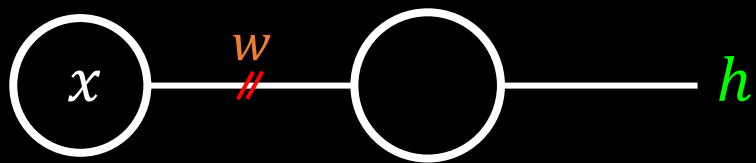
```
    sess.run(train) #한번 경사하강(w 업데이트)
```

학습이 끝난 후 남는 것은?

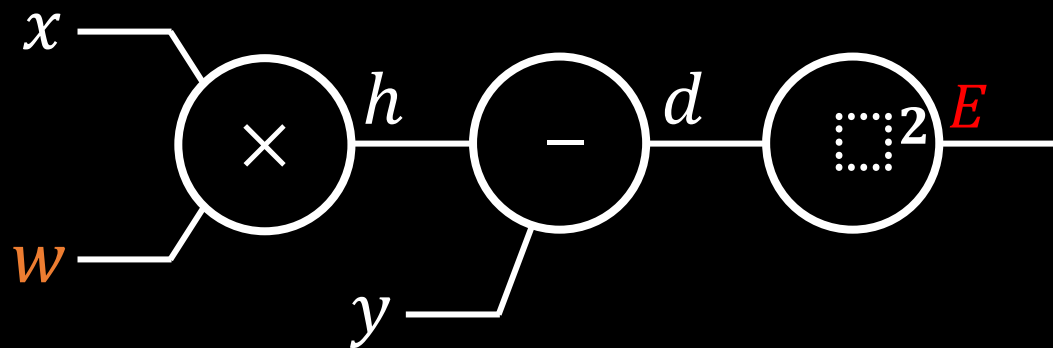


# 계산 그래프 확장

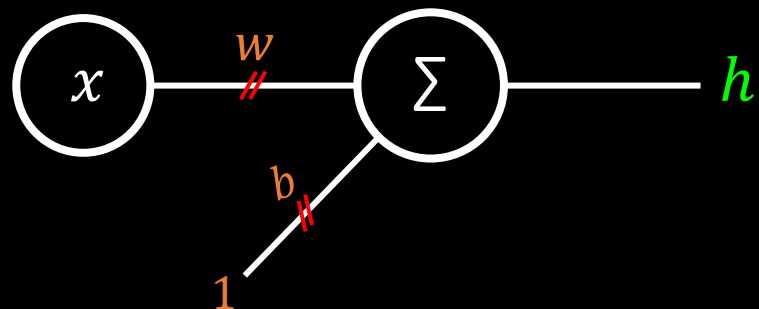
- bias가 있을 경우 (+ 게이트)
- 뉴런 입력이 3개일 때 (+ 게이트)
- 뉴런이 2개일 때
- 튜닝할 파라미터는 모두 몇 개?



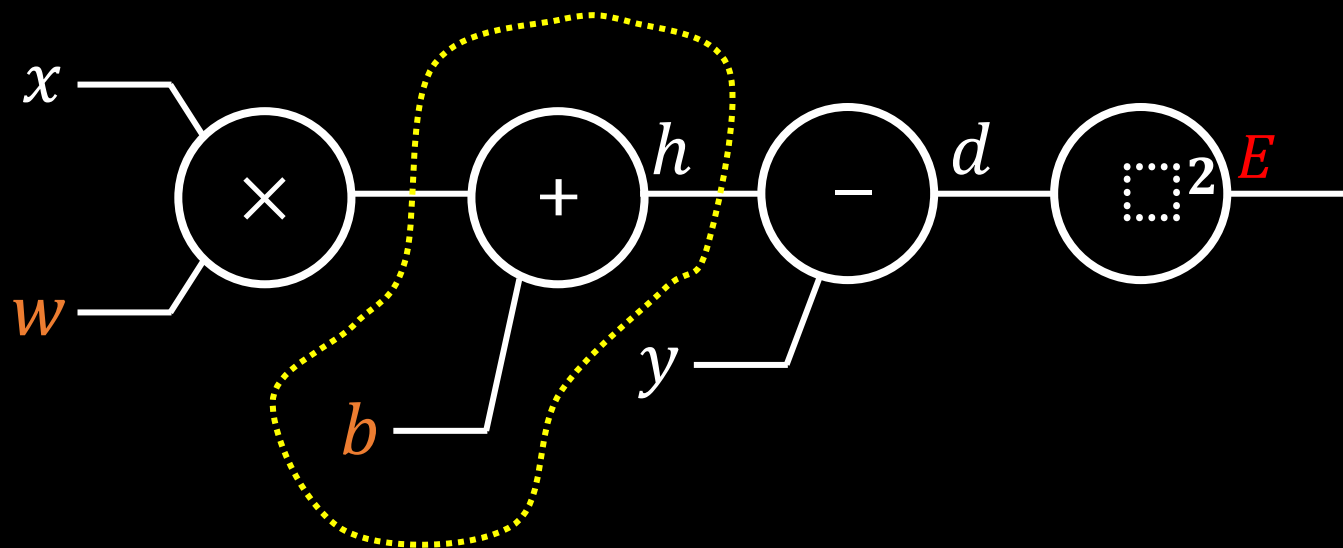
$$E = (wx - y)^2$$

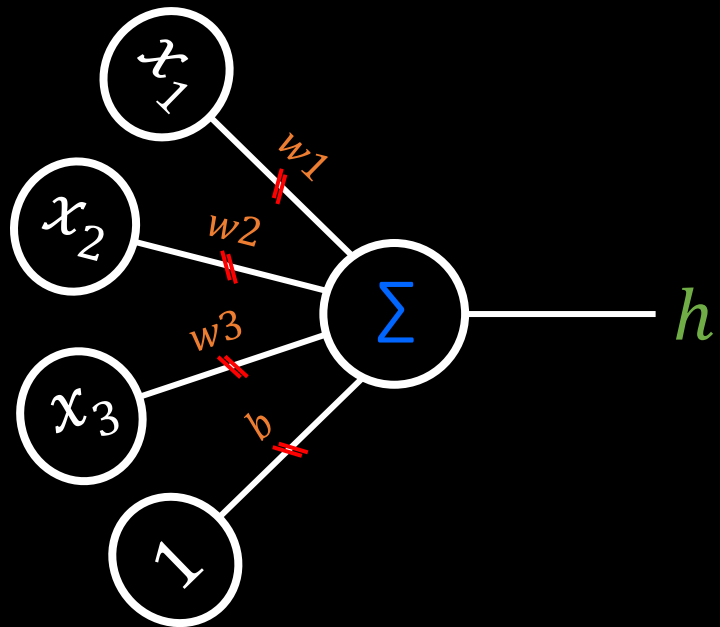




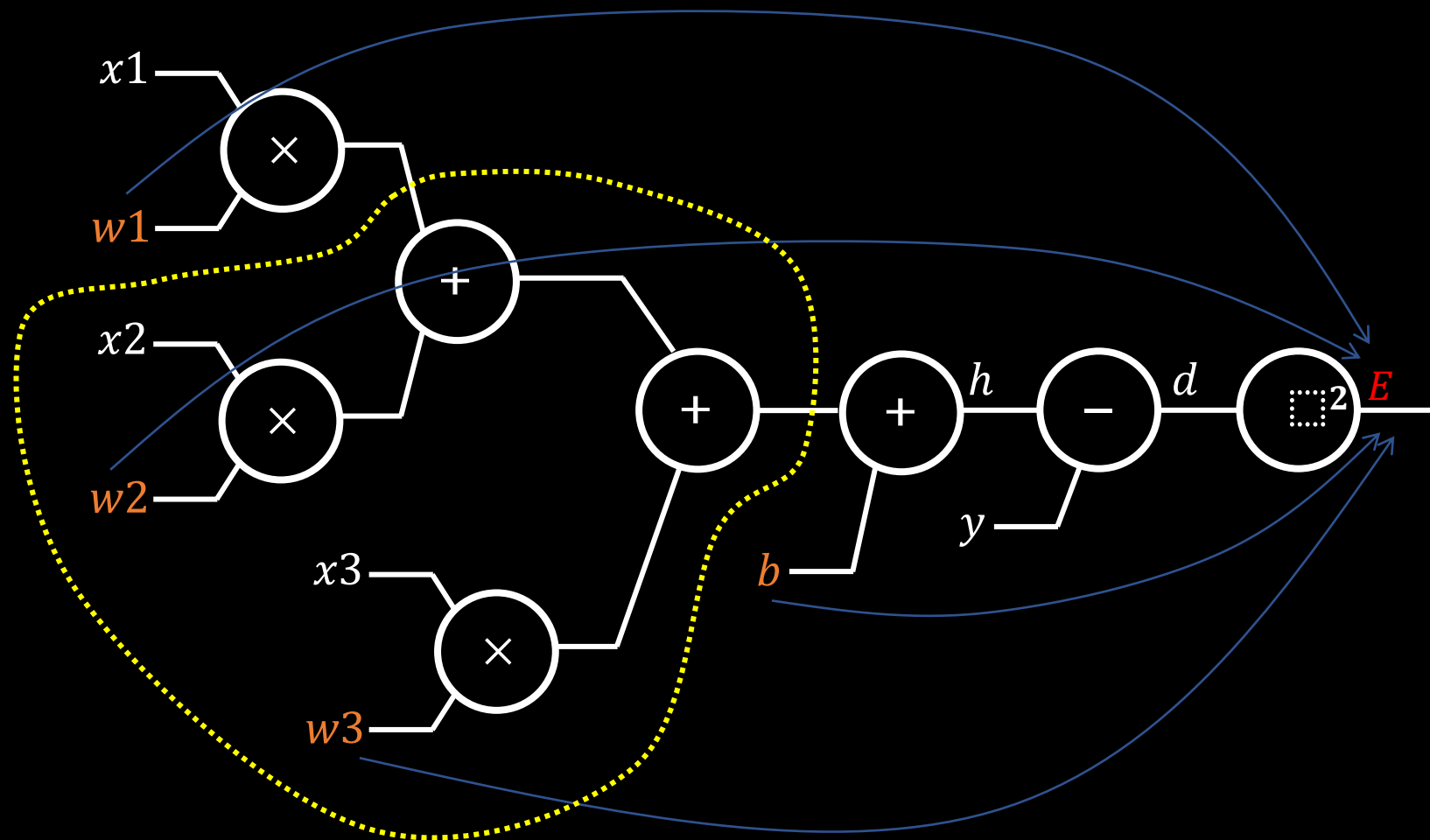


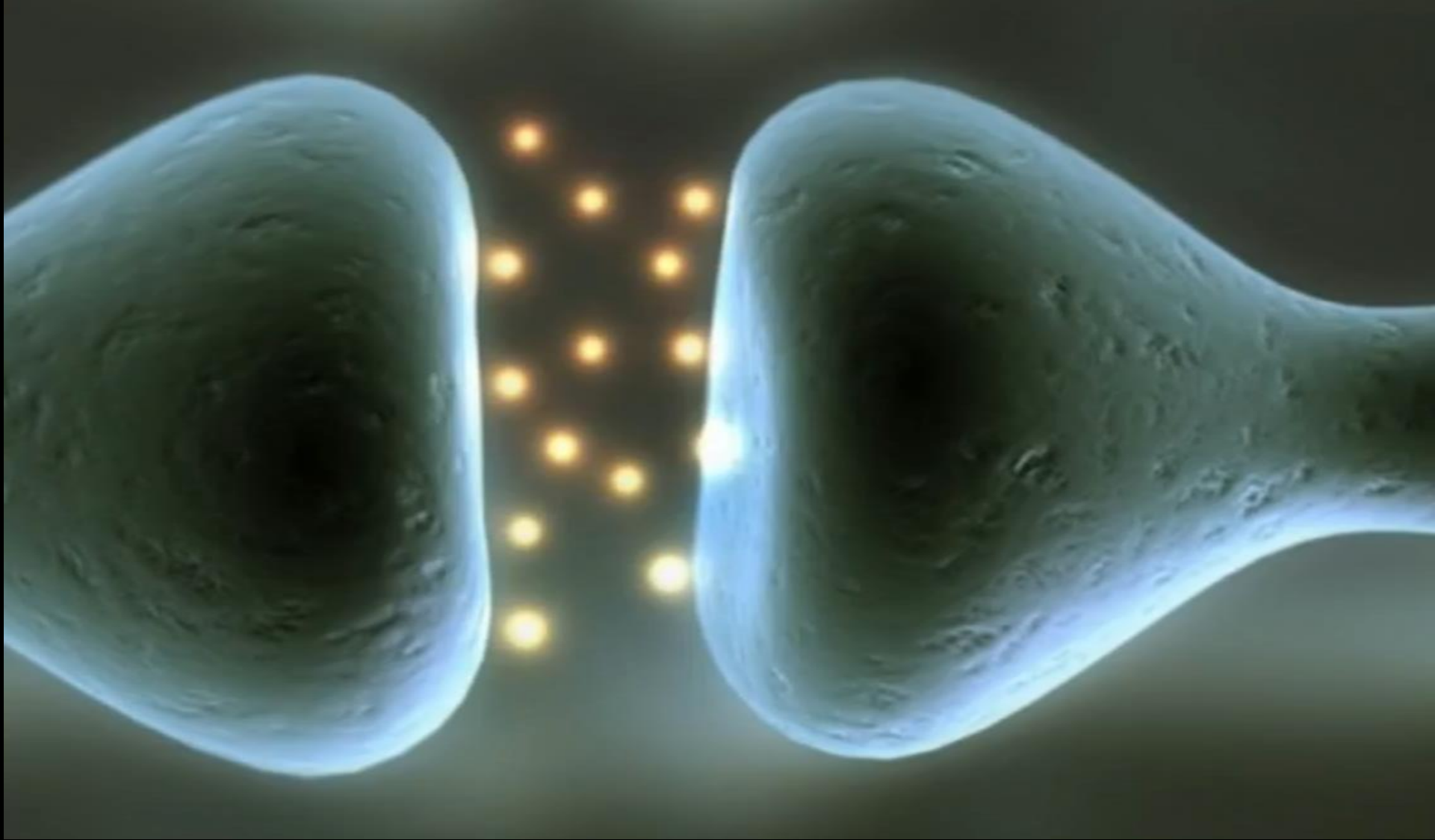
$$E = ((wx + b) - y)^2$$





$$E = ((w_1x_1 + w_2x_2 + w_3x_3 + b) - y)^2$$





학습, 더 새롭고 좋은 연결( $w$ )을 만드는 것

# 오류 함수의 의미

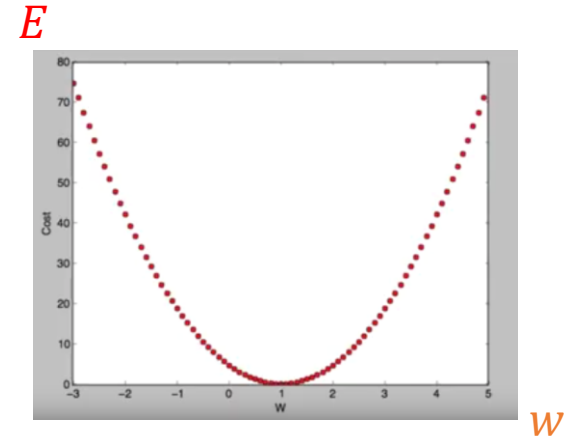
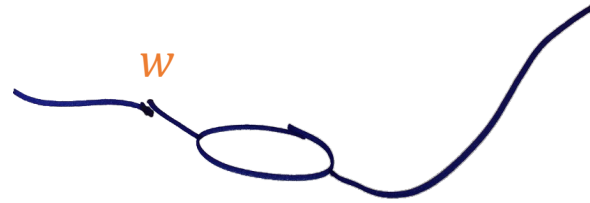
- 기울기가 큼 → 최저점에서 멀리 떨어짐 → 오류가 큼 → **bad!** → 매우 큰 야단 → 스트레스/고통이 큼 → big update( $w$ )
- 기울기가 작음 → 최저점에 가까움 → 오류가 작음 → **not bad!** → 작은 야단 → 스트레스/고통이 작음 → small update( $w$ )
- 기울기 0 → 최저점! → 정답을 맞춤! → **great!** → 야단치지 않음 → no update( $w$ ) → learning ended!

‘좋다’, ‘나쁘다’를 느끼게 하는 기저

우리 마음 속의  
오류함수 *E*

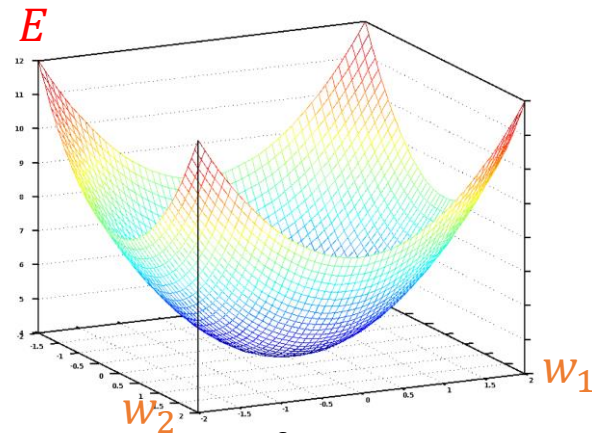
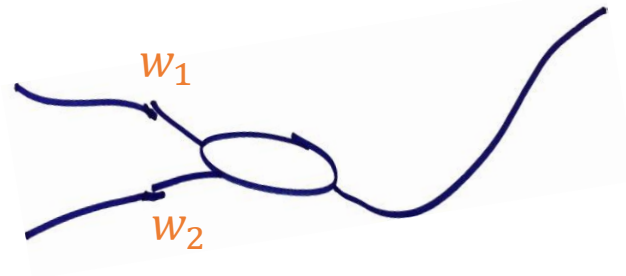
# Cost(Error) Graph

$$E = (w \cdot 1 - 1)^2$$



convex function

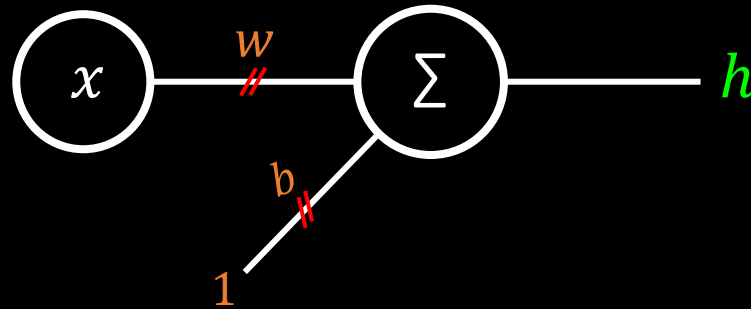
$$E = (W \cdot 1 - 1)^2$$



convex function

# 04.with\_bias.py

## Parameter tuning including bias



05.py

Using multiple data



06.py

Training a neuron  
having multiple  
inputs

# 이번 학습에서는

- 텐서플로우 프로그램의 기본 구조를 이해할 수 있다.
- 오류 계산 그래프 직접 그릴 수 있다.
- 오류 계산 그래프에서 게이트(연산)별 **local** 미치는 영향을 구할 수 있다.
- 역전파와 체인룰을 이용하여 **global** 미치는 영향을 구할 수 있다.
- 학습 메카니즘을 이해할 수 있다.