AI and Deep Learning

Linear Regression & Back-propagation

Jeju National University

Yungcheol Byun

Agenda

- Neuron and Regression
- Loss/Error/Cost Function
- Learning and Updating Weights
- Gradient/Slope
- Computation Graph
- Forward Propagation
- Backpropagation


NAT GEO
WILD

“

After spending most of their time in the ocean, salmons **go back** home(river) where they were born.

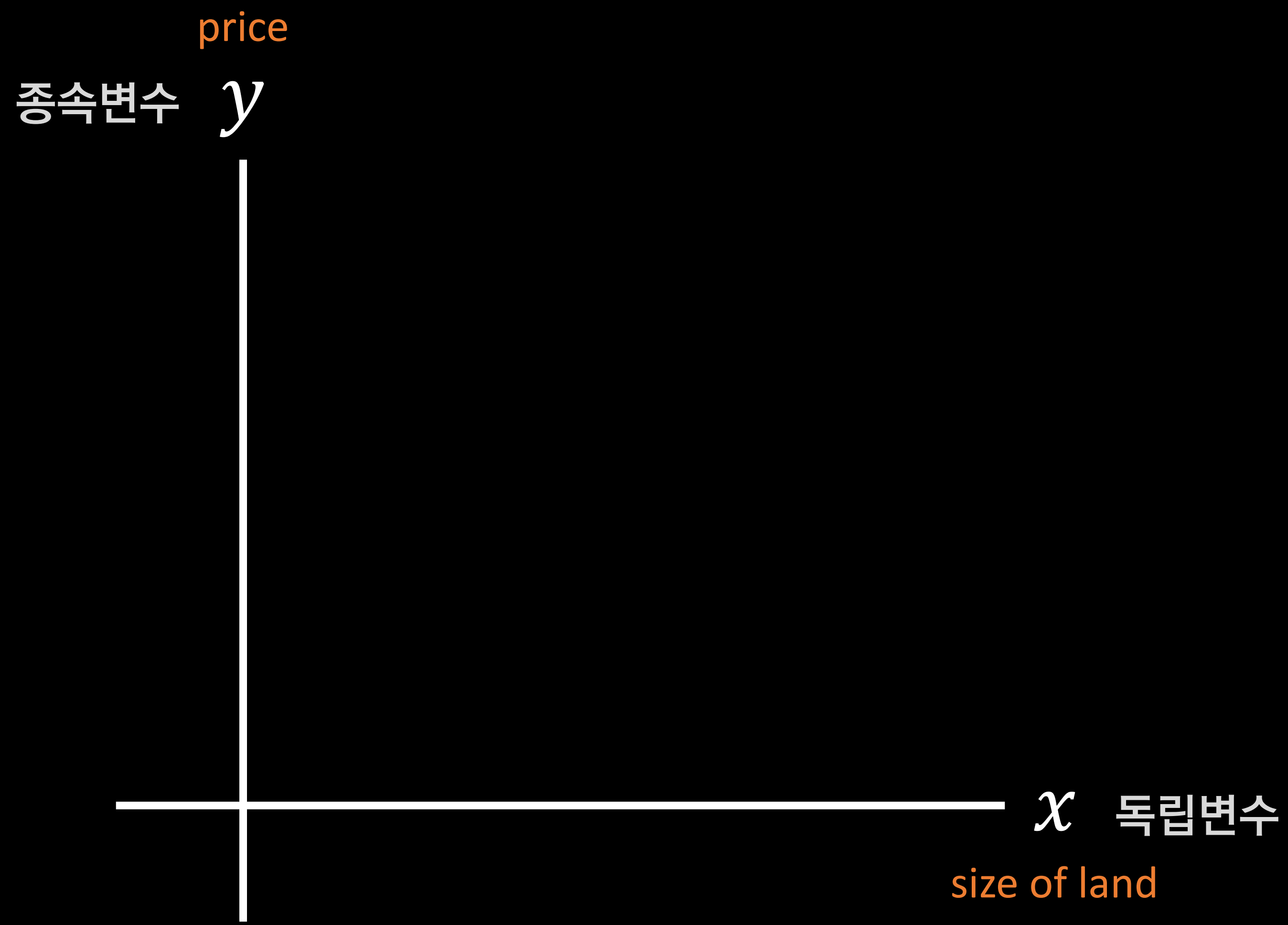
Regression(회귀)

- Going back
- To describe a natural **phenomena**
- A term frequently used in anthropology(**인류학**) to present a natural tendency

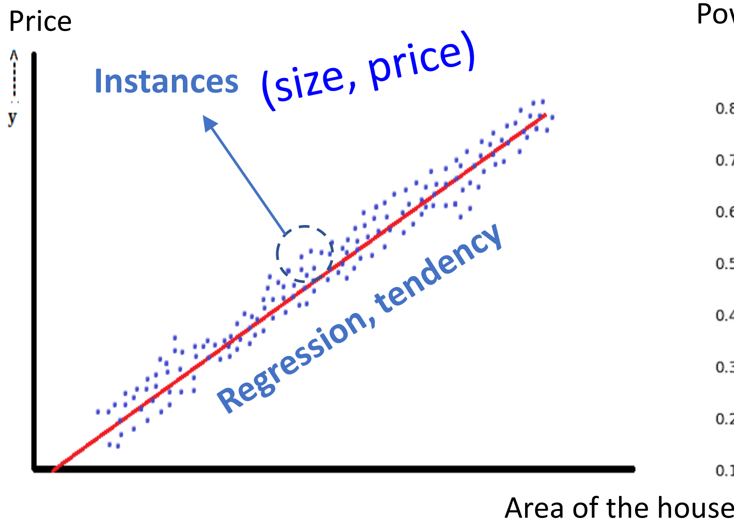
What is ‘a proposed explanation for a phenomenon’?

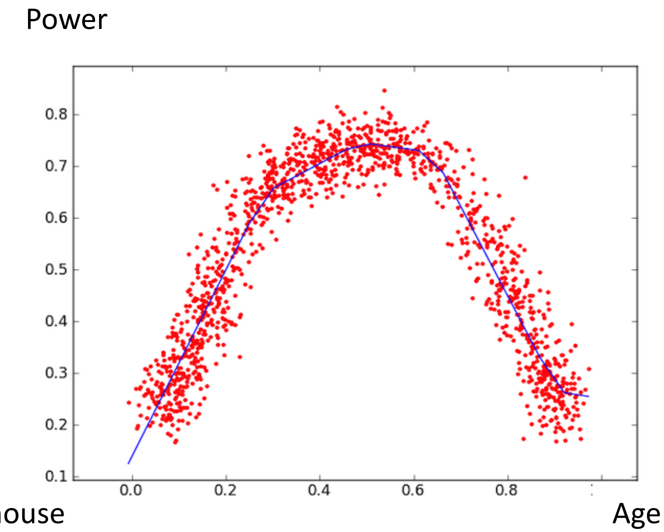
Regression (회귀)

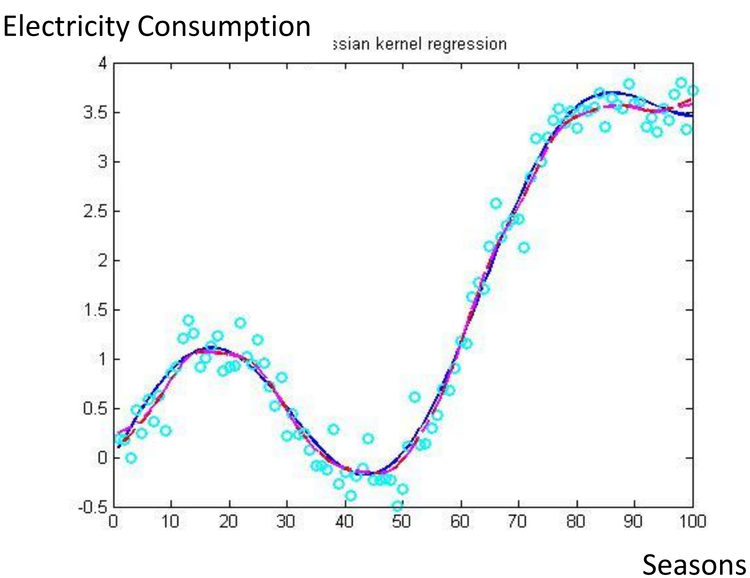
- Statistical measure to determine the relationship between one **dependent** variable (usually denoted by Y, 종속변수) and a series of other **independent** variables (X, 독립변수).

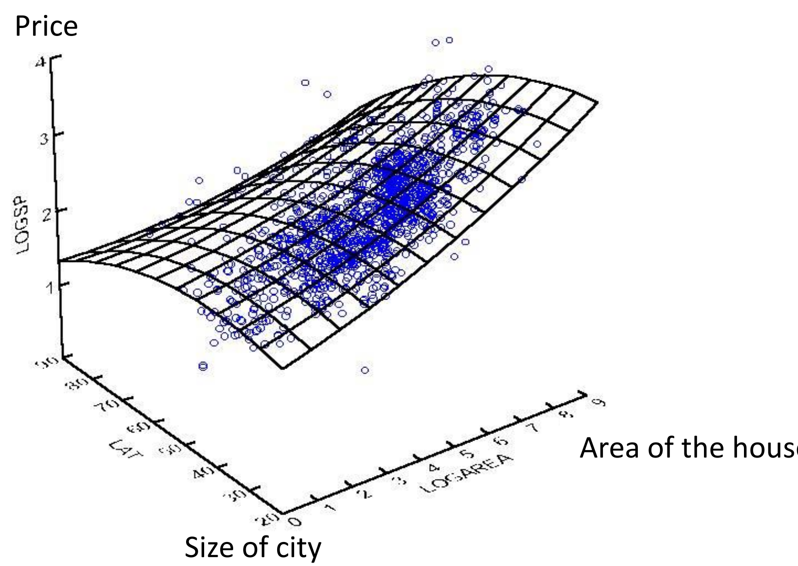

price
종속변수
y
x
독립변수
size of land

Examples of **Regression**

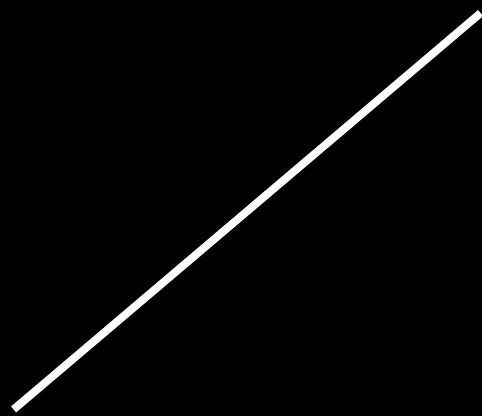








An example of **Linear** Regression



- A **linear** or a **non-linear** regression model?
- It is not about the relationship between the independent variable and the dependent variable.
- If the **hypothesis** is a linear combination of coefficients, then it is a linear model.

$$h = w \cdot x$$

Lab

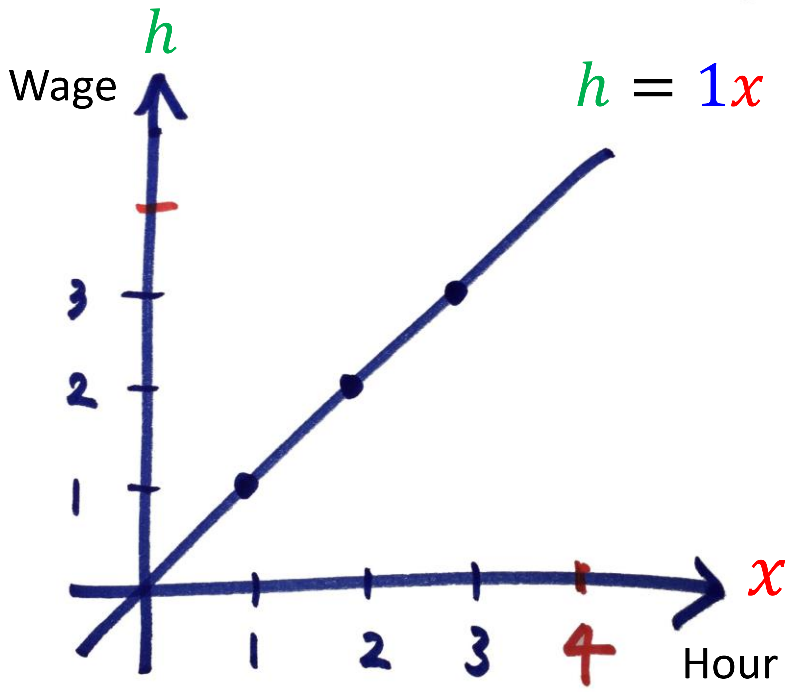
Linear Regression

using desmos

www.**desmos**.com

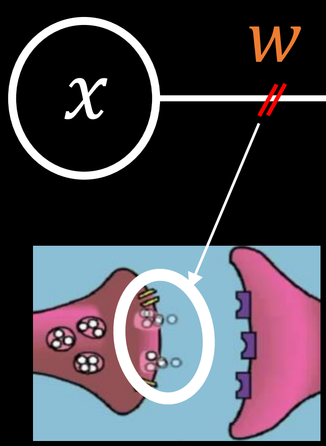
1. Draw a point(data) (1, 1)
2. Add (2, 2), (-1, -1), (-2, -2)
3. $h = x$
4. $h = 2x$
5. $h = wx$ (**rotation**)
6. Move all of the points by adding 1 to y
7. $h = wx + 1$ (**shifting**)
8. $h = wx + b$ (**rotation** and **shifting**)

www.**desmos**.com



$$h = wx$$

Neuron and regression



$$h = wx$$

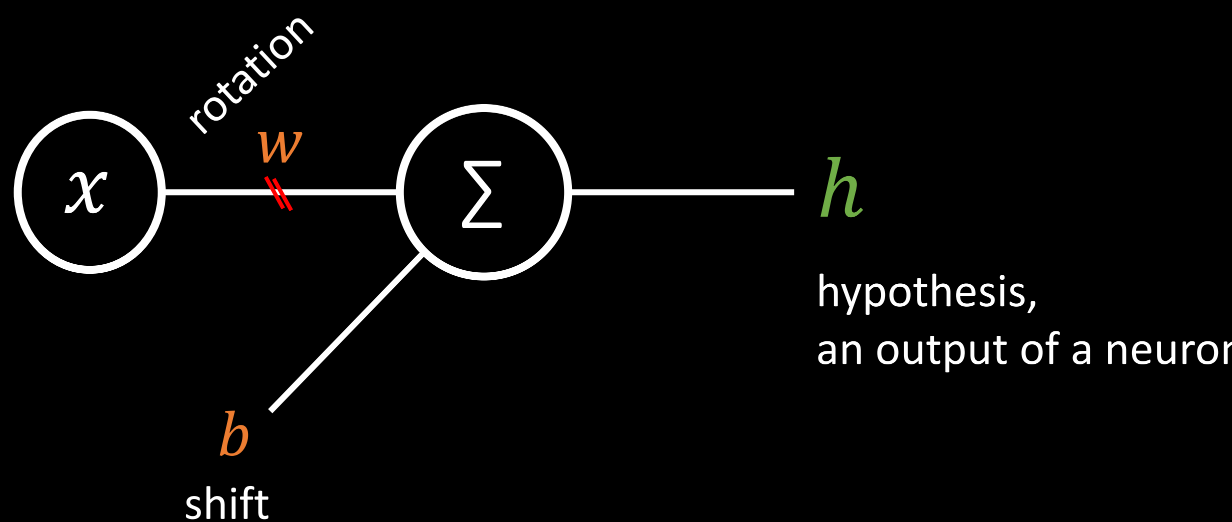
- w : Neuro-transmitter (신경전달물질의 양)
- if large / if small / if not exists (**rotation**)
- A neuron represents a **linear** regression

Hypothesis

$$h = wx$$
$$h = wx + b$$

- An answer by a neuron
- $h$ypothesis : a proposed explanation for a phenomenon (a regression).
- Not proved yet, but it can represent the regression well after updating w.
- b? **shift** for better linear regression representation

The role of *w* and *b*



$$h = \underbrace{wx + b}_{\text{linear combination of coefficients}}$$

⟶ linear regression

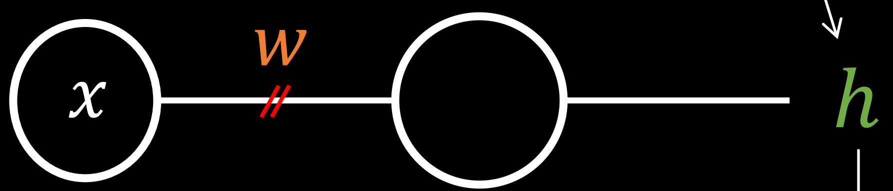
How to learn (update w)

- Scolding or blaming the neuron if it is wrong
- The neuron gets stress and automatically updates w to answer well next time so that the error(difference) decreases.
- How can we calculate the error?

Error function

loss function
difference function

answer by a neuron



ground truth

$$E = |w \cdot x - y|$$

Why absolute?

Error function

The error is the difference between a neuron's answer and it's ground truth.

$$E = |hypothesis - y|$$

$$E = |w \cdot x - y|$$

‘1 hour, then 1 USD’

x_i	y_i
1	1

$$E = |w \cdot 1 - 1|$$

Supervised Learning

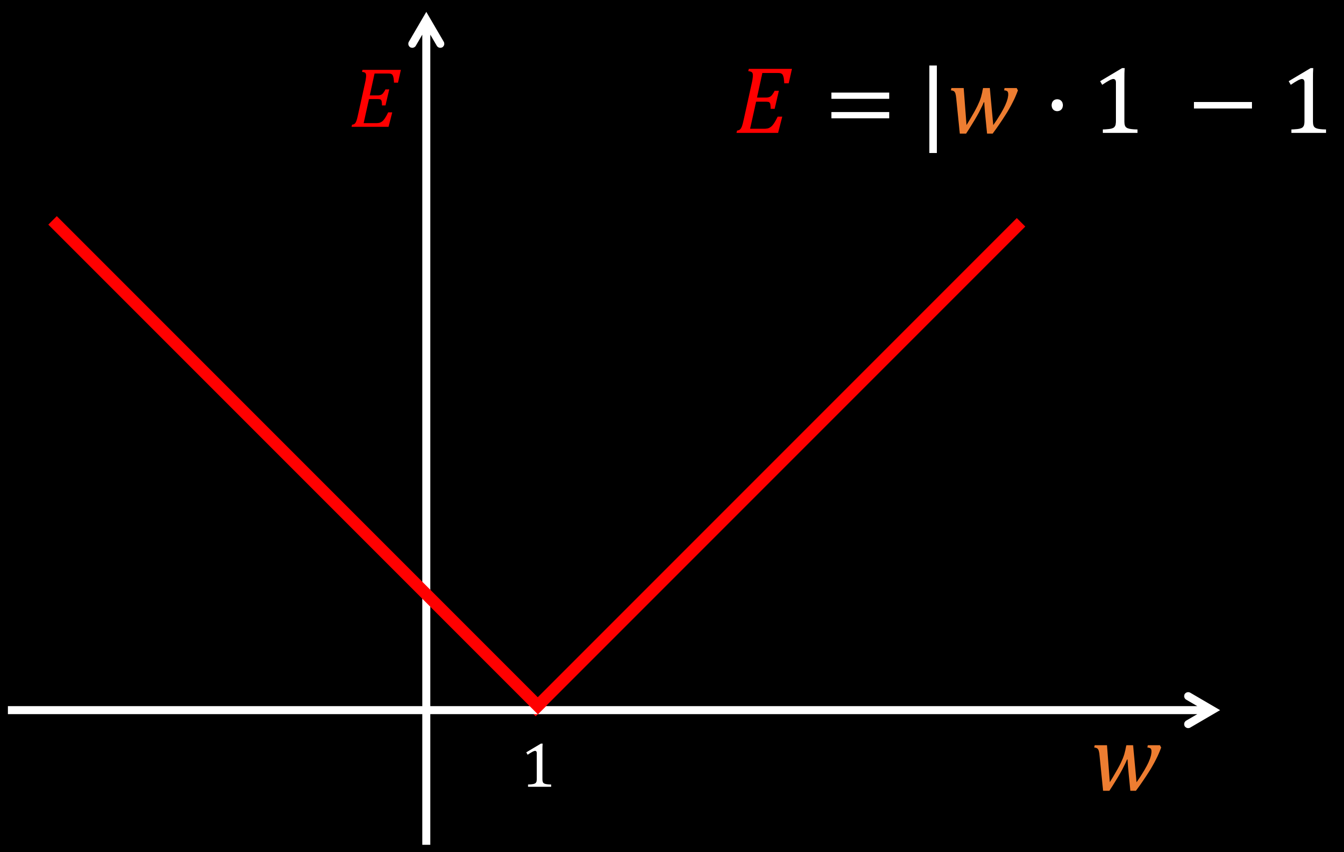
지도학습

www.desmos.com

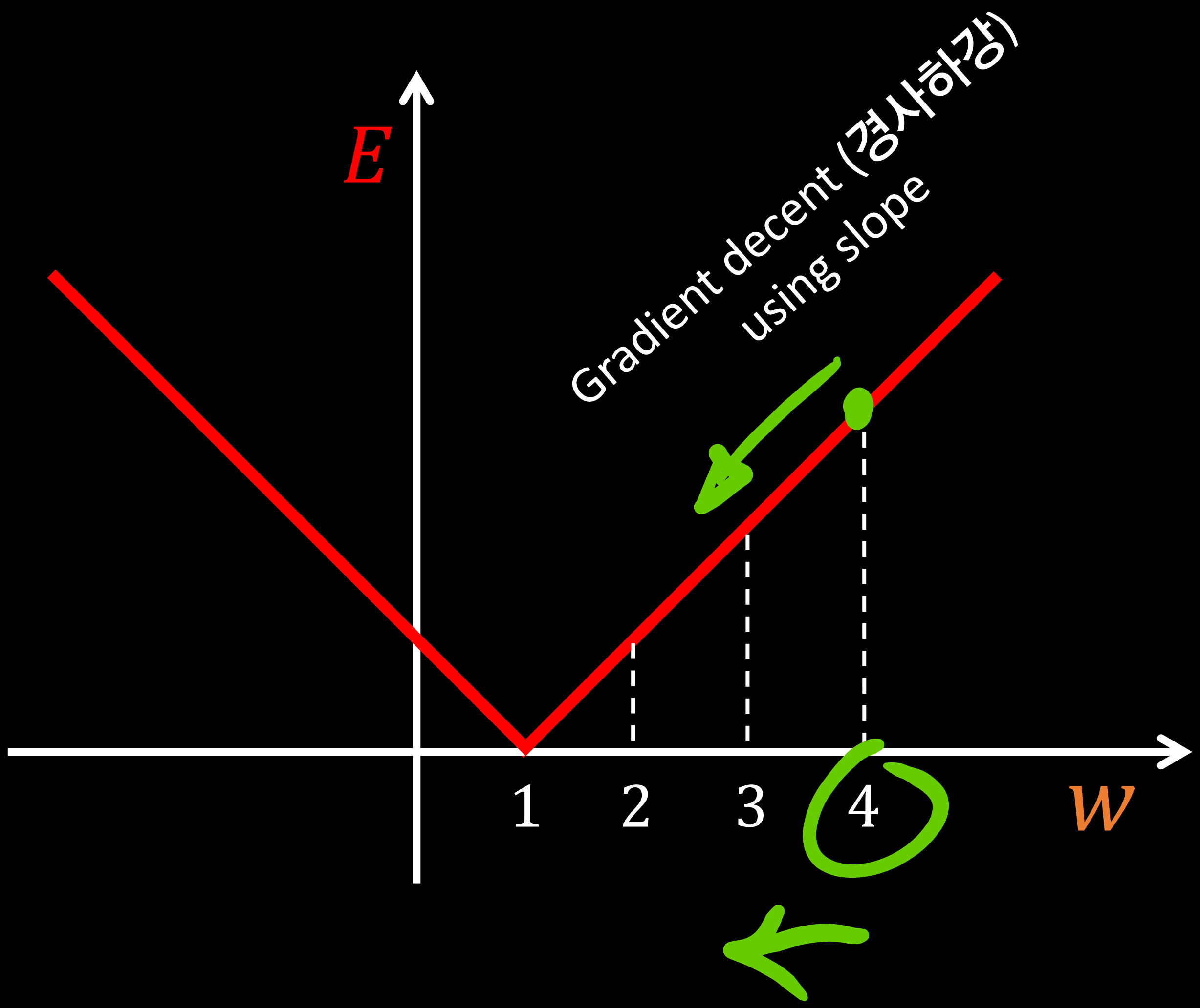
1. Mark (1, 1)
2. $h = w \cdot x$
3. $E = w \cdot 1 - 1$
4. $E = |w \cdot 1 - 1|$
5. (w, E)

desmos

Error Function of w



How to update w

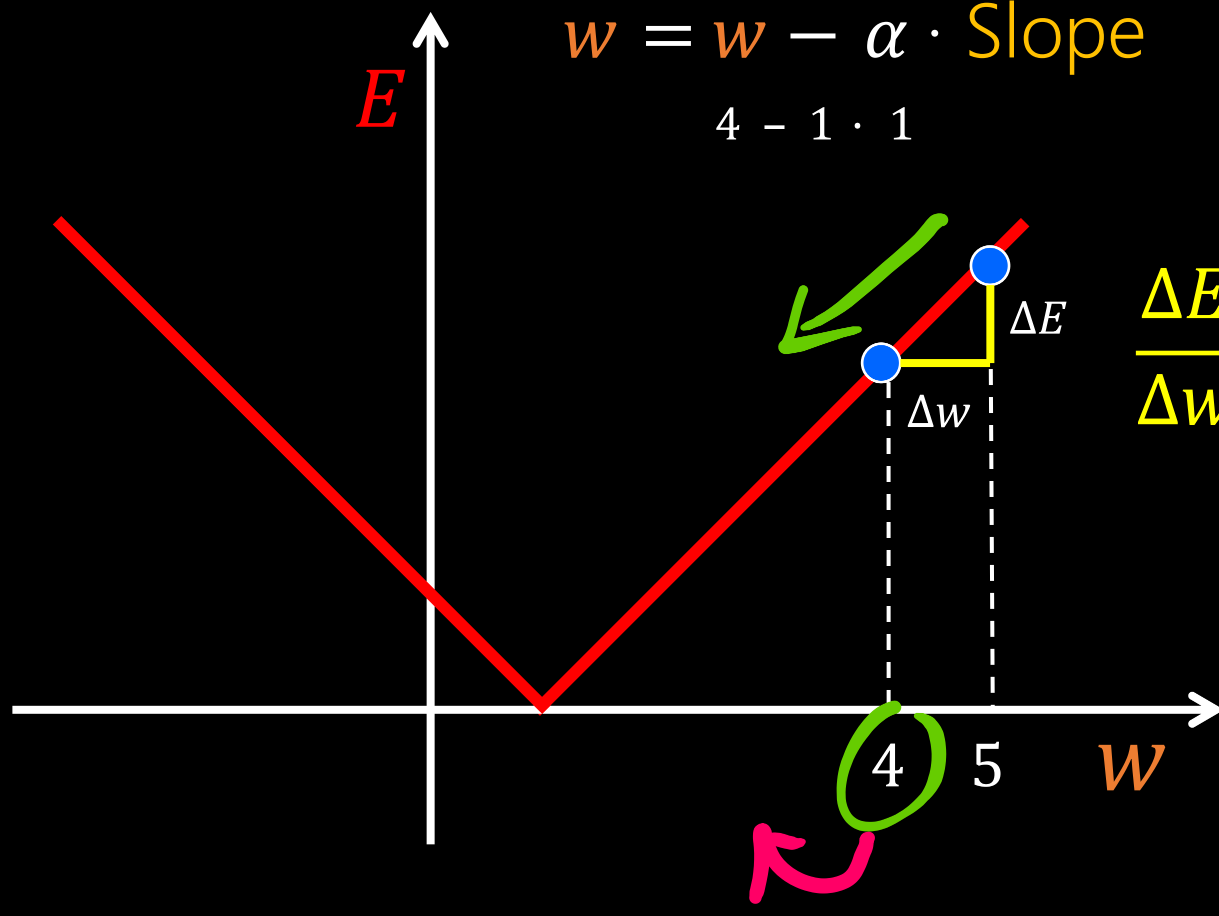


How to update w

learning rate (ex, 1)

$$w = w - \alpha \cdot \text{Slope}$$

$$4 - 1 \cdot 1$$



$$\frac{\Delta E}{\Delta w} = \frac{+1}{+1}$$

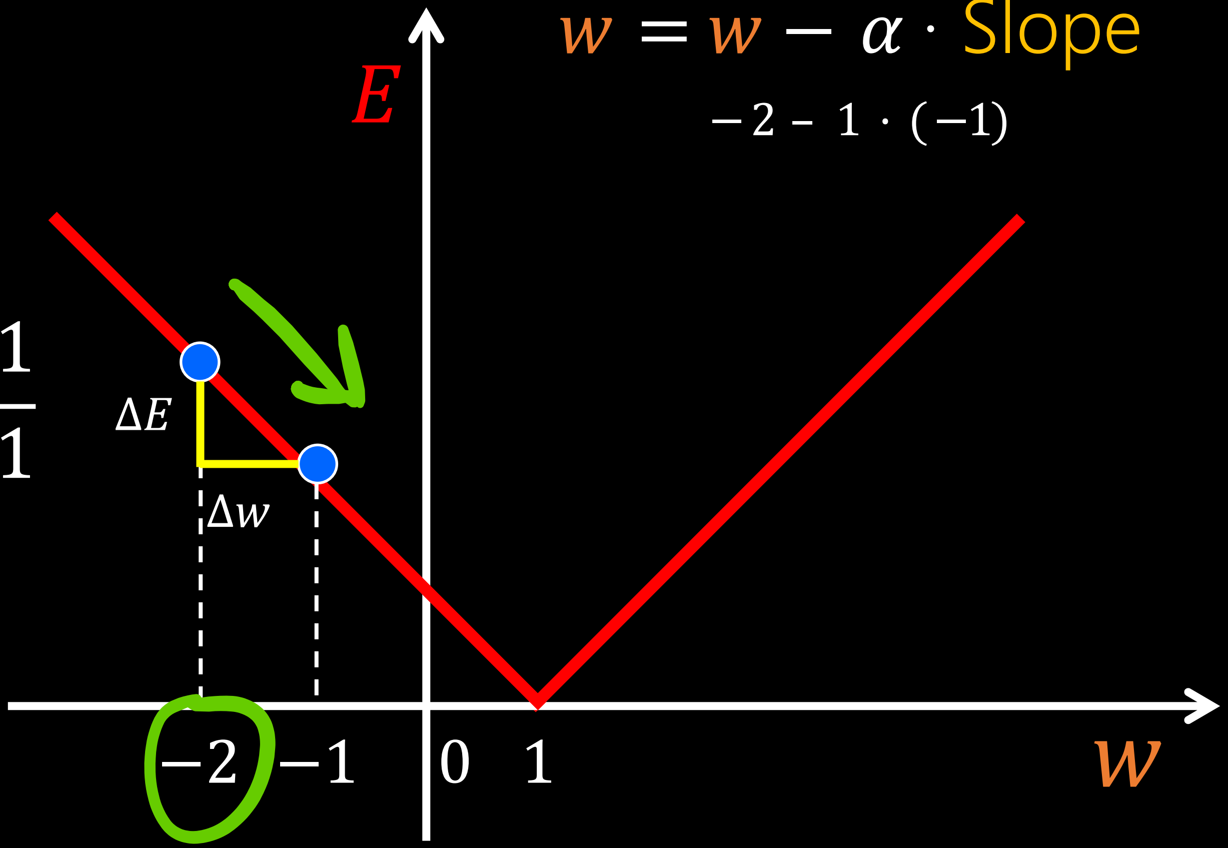
How to update w

learning rate (ex, 1)

$$w = w - \alpha \cdot \text{Slope}$$

$$-2 - 1 \cdot (-1)$$

$$\frac{\Delta E}{\Delta w} = \frac{-1}{+1}$$



$$w = 4, \alpha = 1, Slope = 1$$

$$w = w - \alpha \cdot \text{Slope}$$

$4 - 1 \cdot 1 \longrightarrow 3$ Error E = 2

$3 - 1 \cdot 1 \longrightarrow 2$ Error E = 1

$2 - 1 \cdot 1 \longrightarrow 1$ Error E = 0

$$w = -2, \alpha = 1, Slope = -1$$

$w = w - \alpha \cdot$ Slope

$-2 - 1 \cdot (-1) \longrightarrow -1$ Error E = 2

$-1 - 1 \cdot (-1) \longrightarrow 0$ Error E = 1

$0 - 1 \cdot (-1) \longrightarrow 1$ Error E = 0

$$w = -2, \alpha = 2, Slope = -1$$

$w = w - \alpha \cdot \text{Slope}$

$-2 - 2 \cdot (-1) \longrightarrow 0$ Error $E = 1$

$0 - 2 \cdot (-1) \longrightarrow 2$ Error $E = 1$

$2 - 2 \cdot (1) \longrightarrow 0$ Error $E = 1$

$0 - 2 \cdot (-1) \longrightarrow 2$ Error $E = 1$

$2 - 2 \cdot (1) \longrightarrow 0$ Error $E = 1$

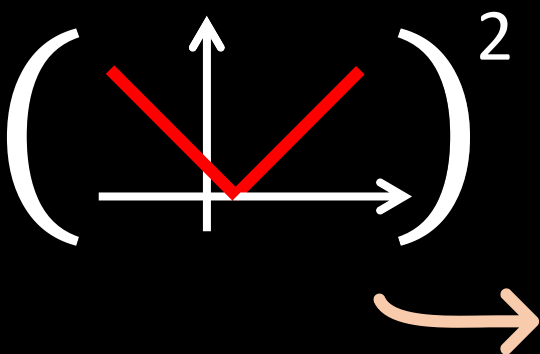
Absolute Error

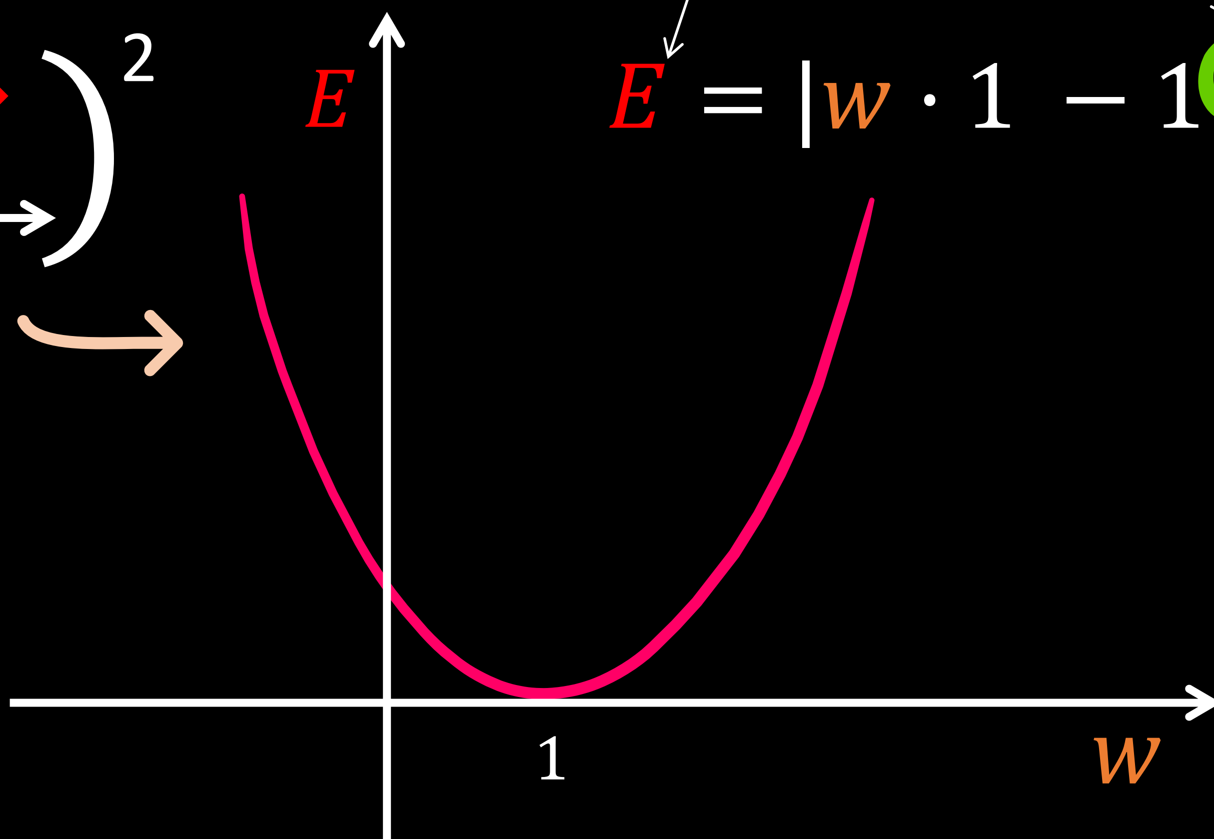
L1 Loss function

Issues in the absolute error

- Always the same slope in the error graph regardless of the value of w
- Therefore, the same speed in movement
- Not guarantee to get the proper value of w which gives 0 or almost 0 error.

Square Error





error

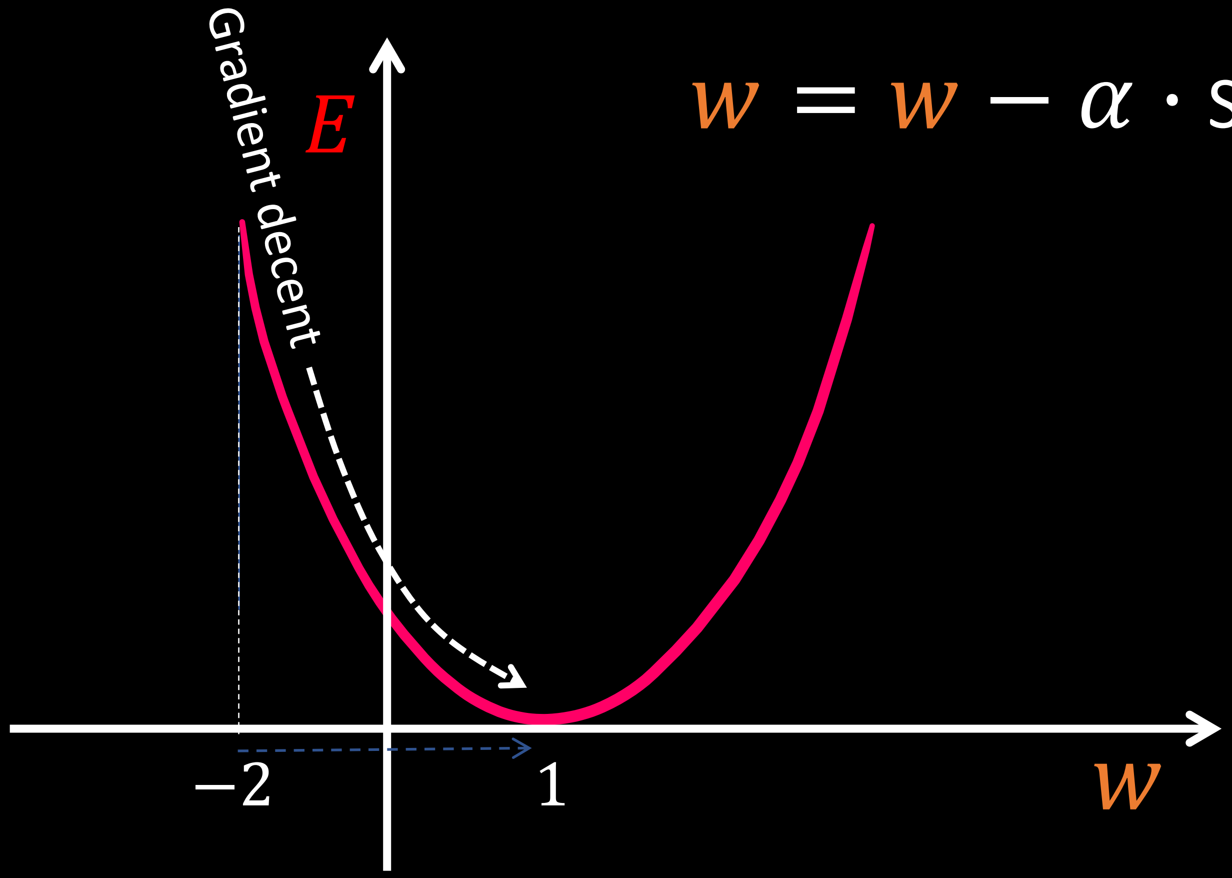
square (제곱)

$$E = |w \cdot 1 - 1|^2$$

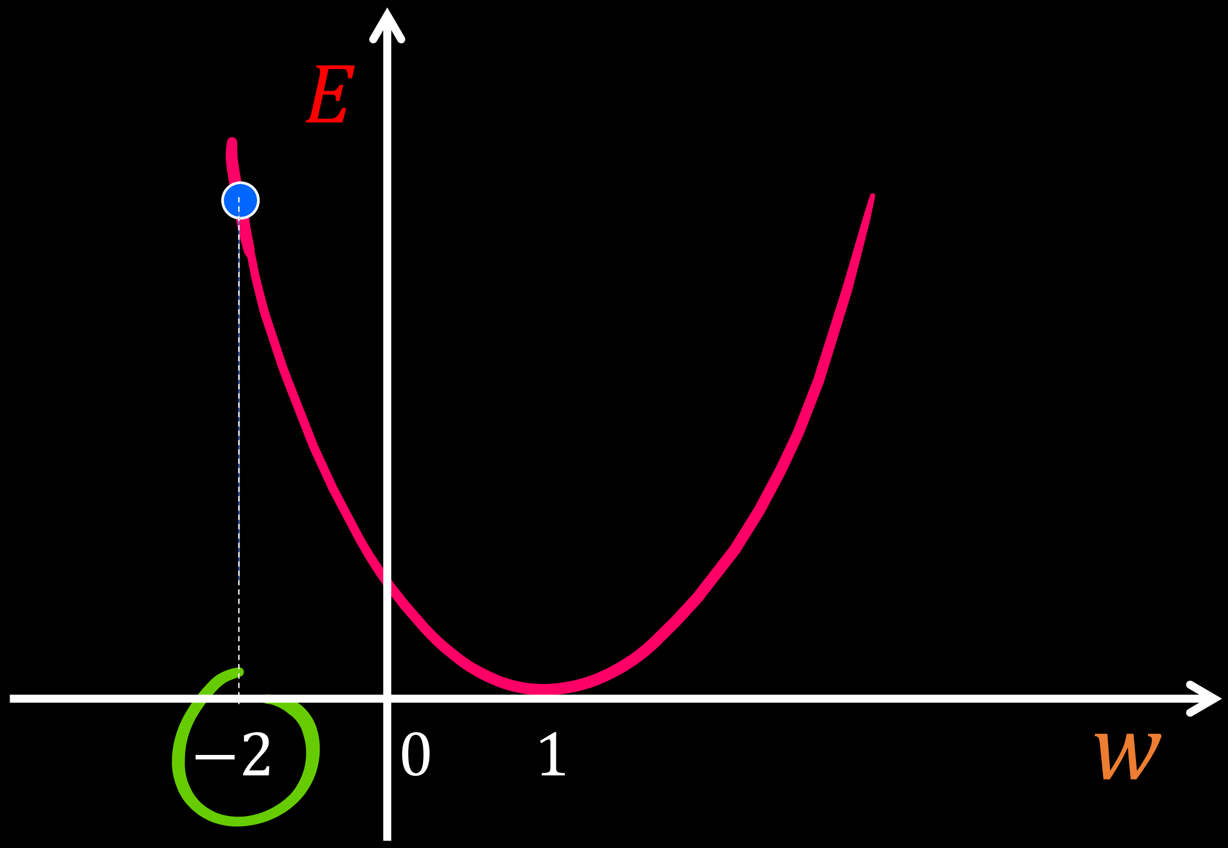
How to update w

$$\frac{\Delta E}{\Delta w}$$

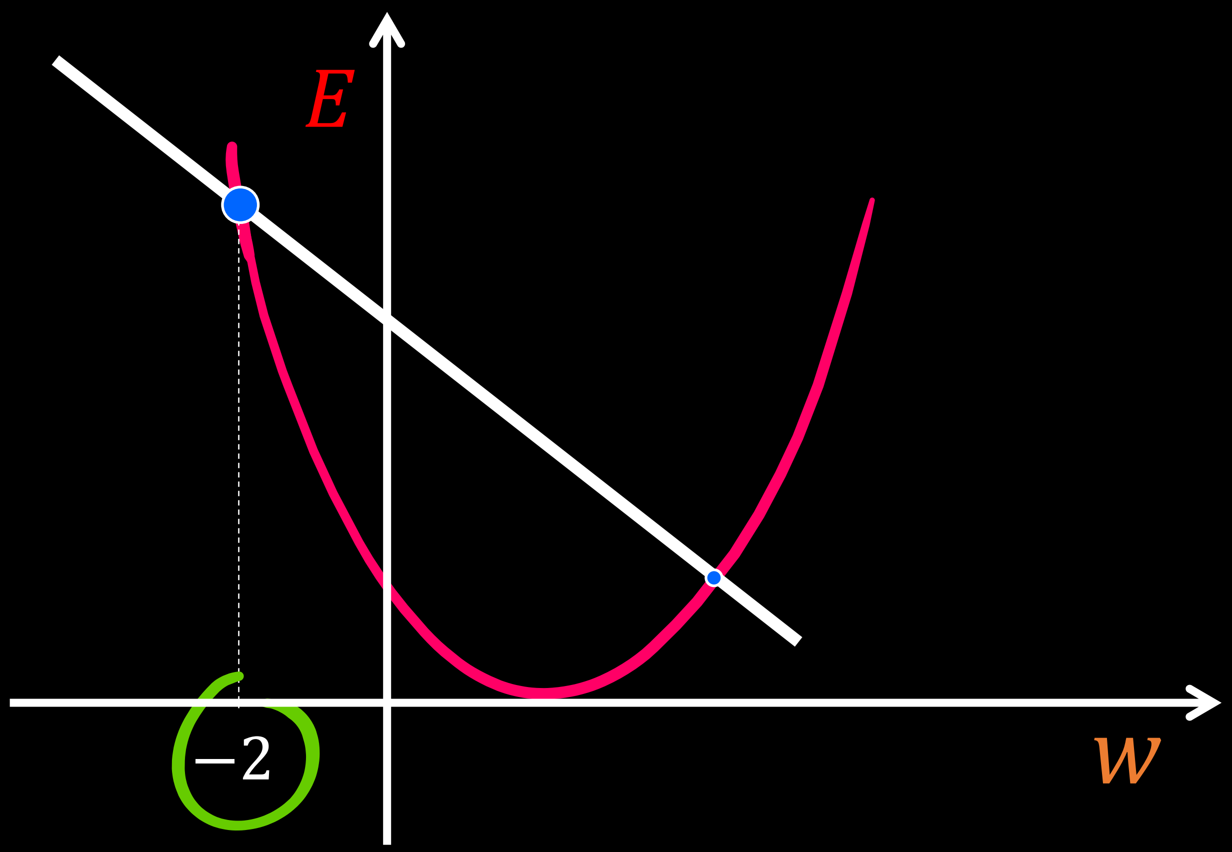
$$w = w - \alpha \cdot \text{Slope}$$



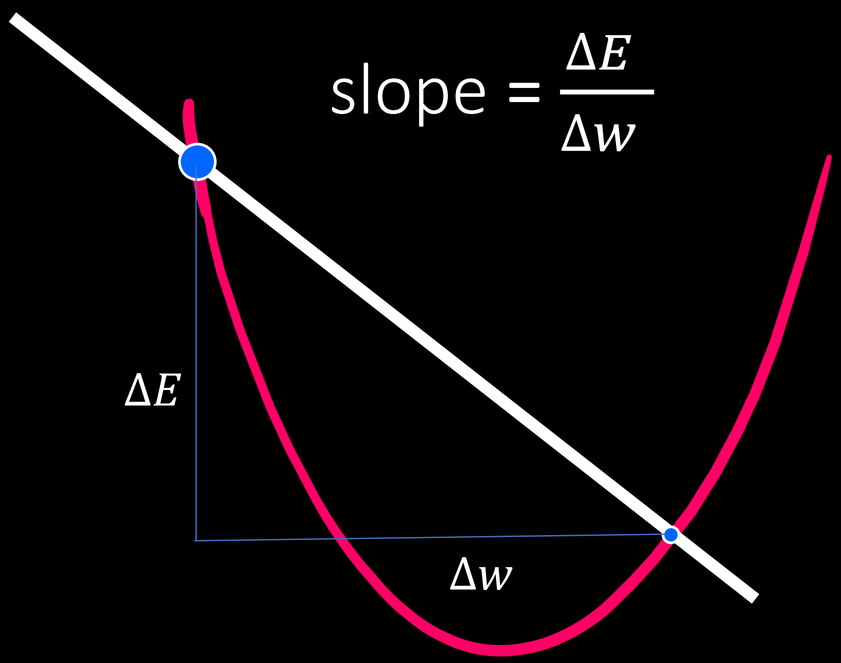
How to update w



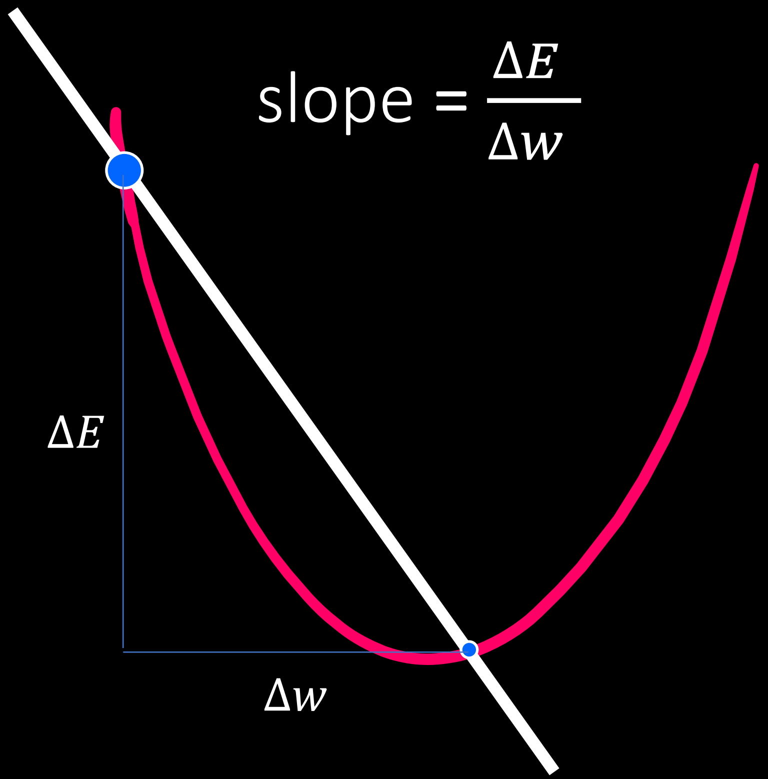
How to update w



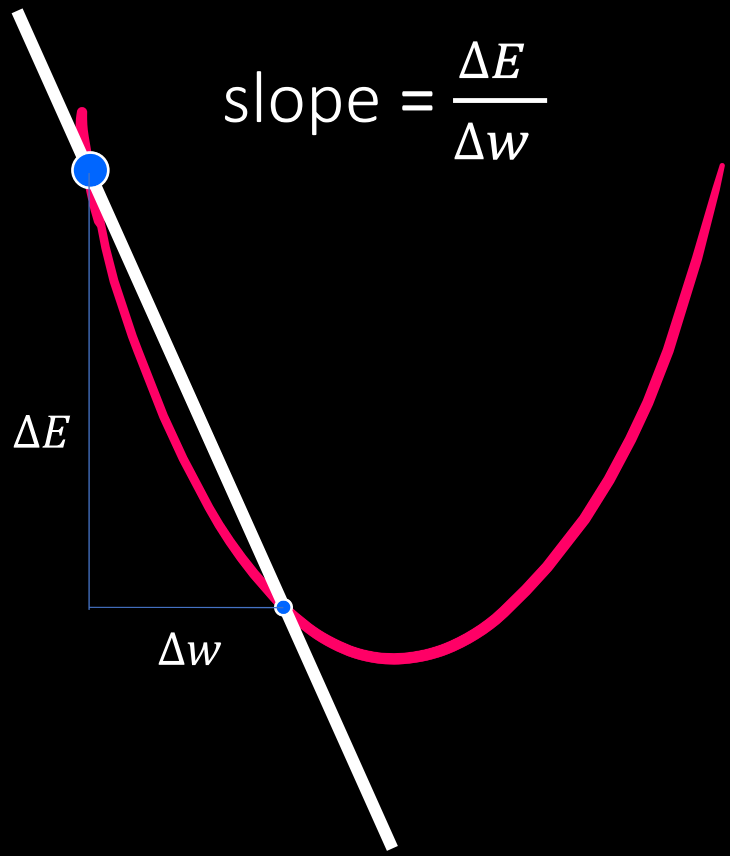
How to update w



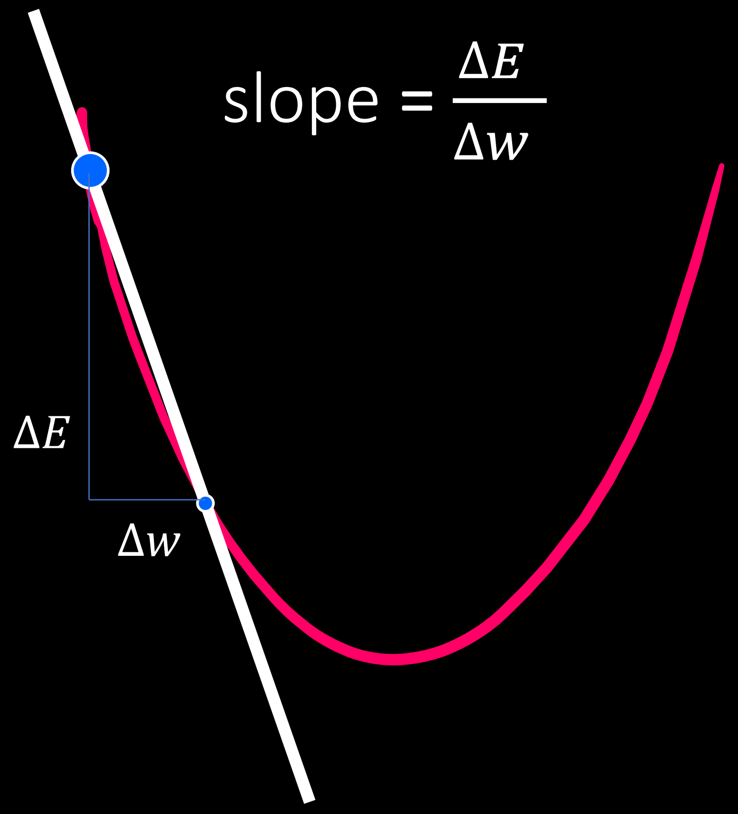
How to update w



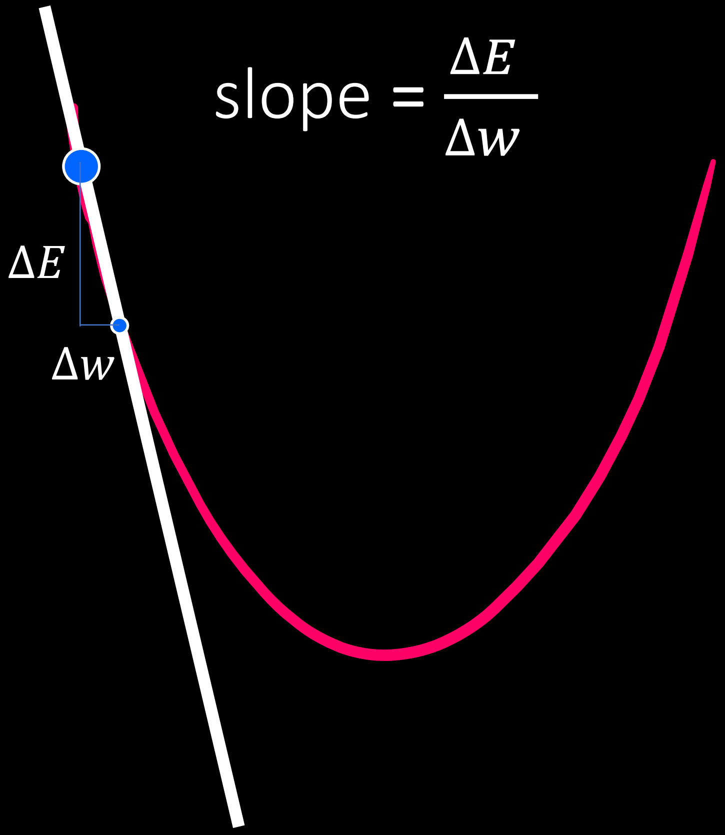
How to update w



How to update w



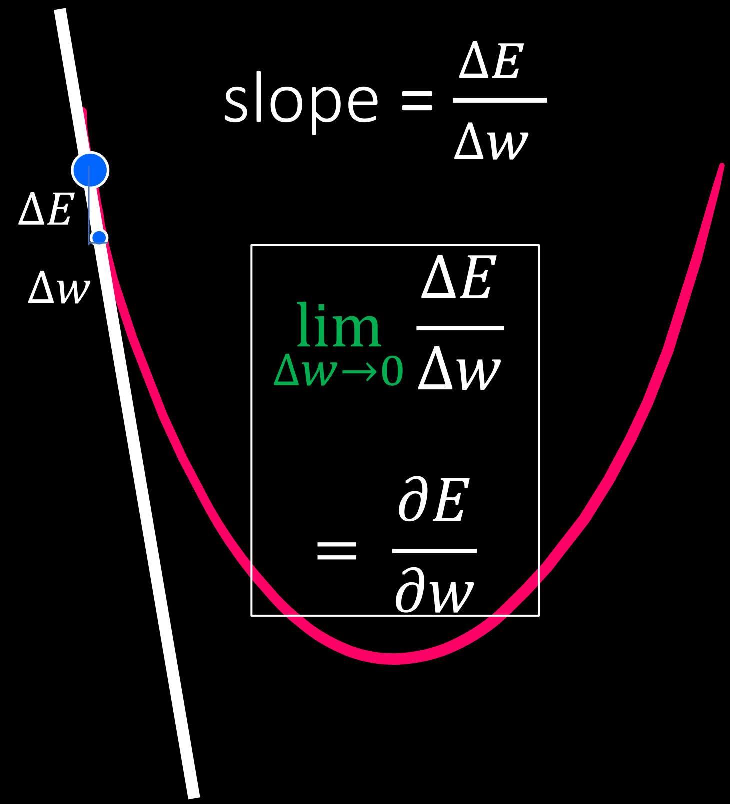
How to update w



slope = $\frac{\Delta E}{\Delta w}$

How to update w

접선·Tangent line

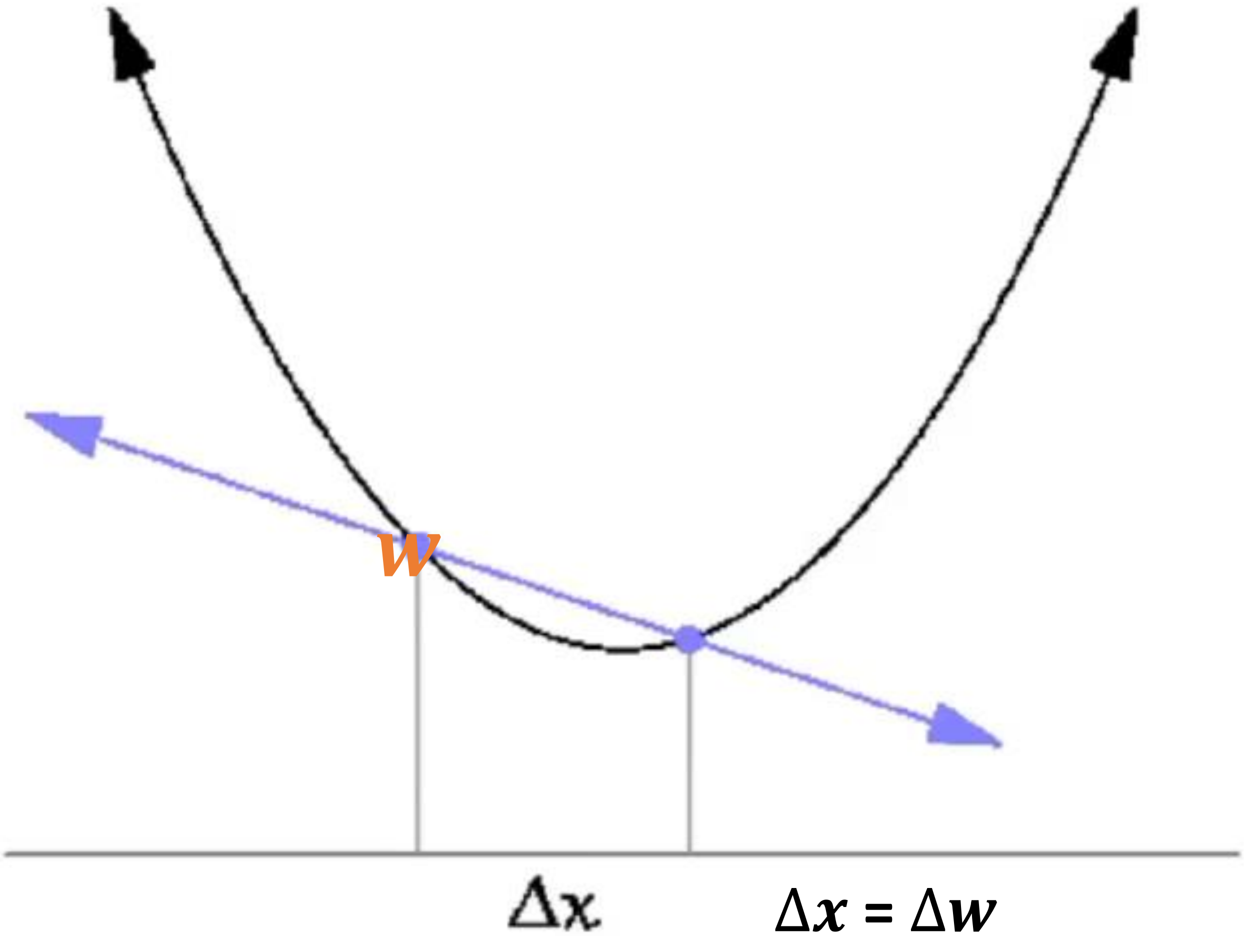


$$\text{slope} = \frac{\Delta E}{\Delta w}$$

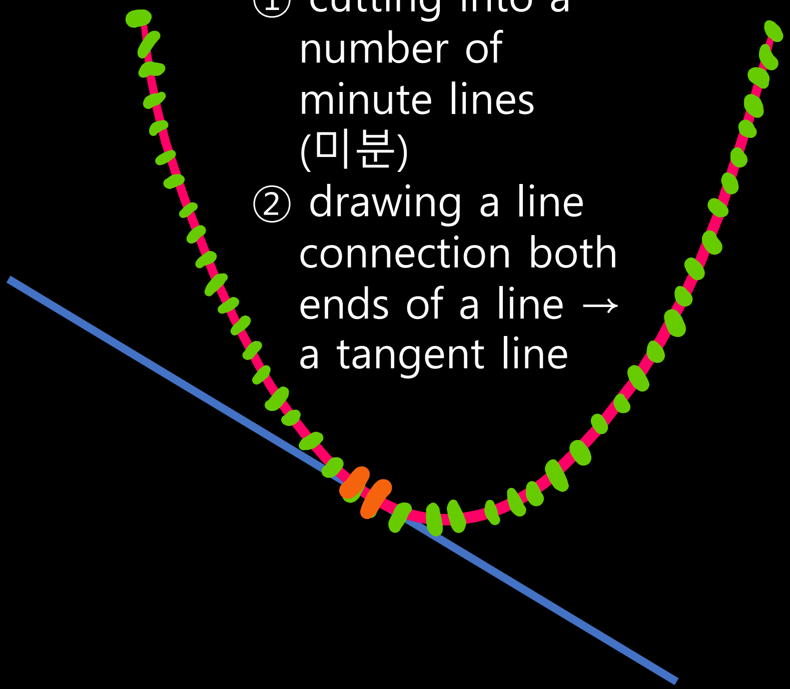
$$\lim_{\Delta w \to 0} \frac{\Delta E}{\Delta w}$$

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

How to update w



How to update *w*



Numerical differentiation

① cutting into a number of minute lines (미분)
② drawing a line connection both ends of a line → a tangent line

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

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

= 미분

How to update w

$$w = w - \alpha * \text{Slope}$$

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

$\alpha =$ learning rate (ex, 0.1)

Squared Error

L2 Loss function

Advantages

- Fast movement from both sides and fine tuning at the valley(center) area
- Different slope/gradient according to the value of w
- Steep slope means that the error is big and w is far from the optimal area.
- We can get the slope(gradient) at any place(differentiable).

In case of Absolute Error

- Always the same slope in the error graph regardless of the value of w
- Therefore, the same speed in the movement
- Not sure to get the w value which gives 0 error or almost 0
- No way to guess the current value of w
- Not differentiable when w is 1

Multiple Data

For 3 instances of data

x_i	y_i
1	1
2	2
3	3

$$E = \frac{1}{3}\sum_{i=1}^{3}(wx_i - y_i)^2$$

desmos

Add (2, 2), (3, 3)

$$E = \frac{1}{3}\sum_{i=1}^{3}(wx_i - y_i)^2$$

Draw (w, E)

Multiple Data

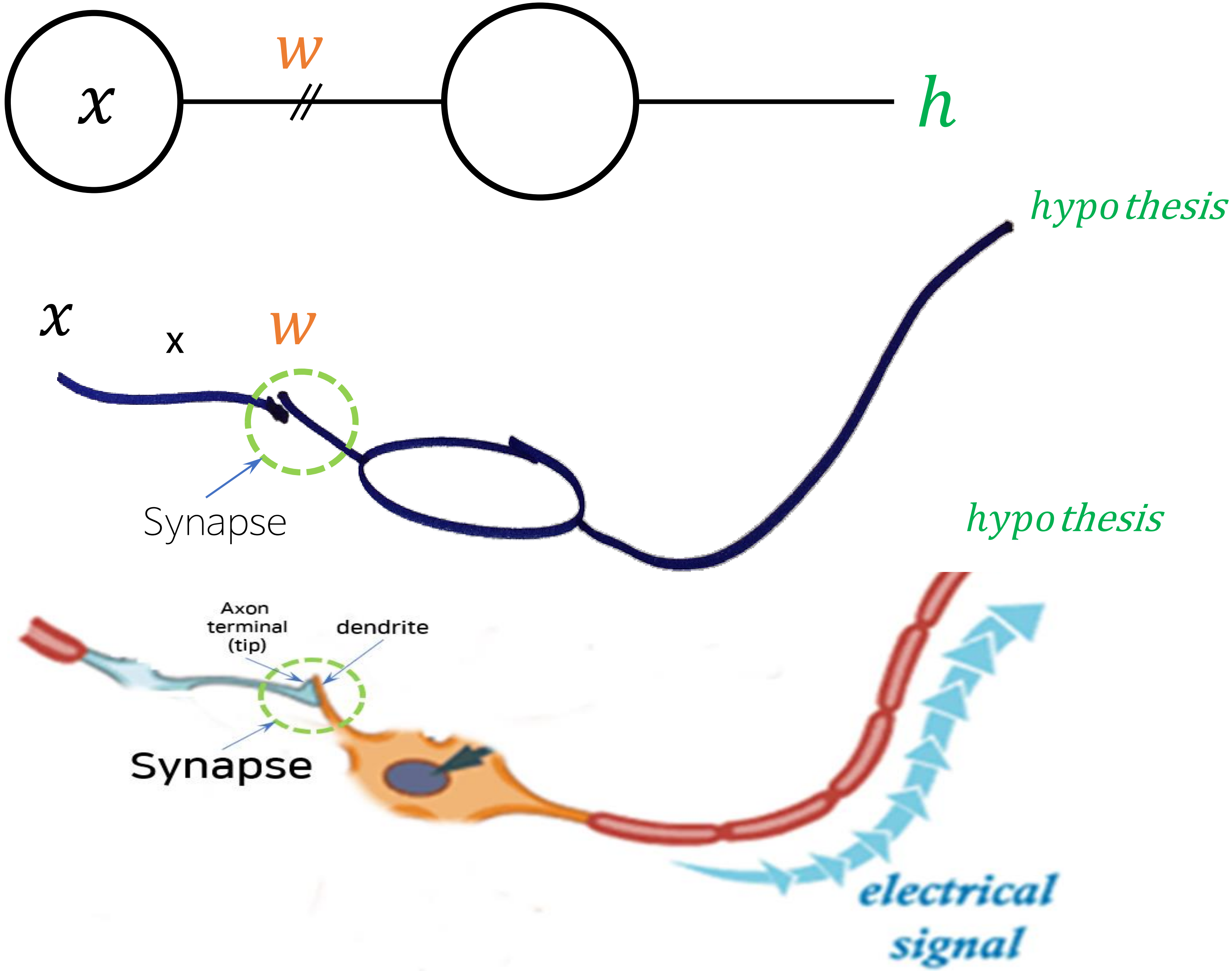
In case of m instances,

Mean Square Error

An answer by a neuron

$$E = \frac{1}{m}\sum_{i=1}^{m}(wx_i - y_i)^2$$

Ground truth



x
w
h
hypo thesis
x
x
w
Synapse
hypo thesis
Axon terminal (tip)
dendrite
Synapse
electrical signal

The meaning of slope

Steep slope $\frac{\Delta E}{\Delta w}$ $\frac{8}{1}$

The error E will change drastically when we change w.

Gentle slope $\frac{\Delta E}{\Delta w}$ $\frac{1}{10}$

The error E changes just a little bit when we change w.

$$\lim_{\Delta w \to 0} \frac{\Delta E}{\Delta w} \to \frac{\partial E}{\partial w}$$

Therefore,

Slope(gradient)

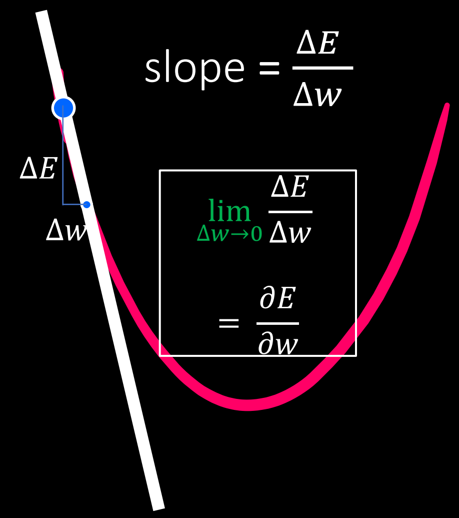
means the influence of w change on error E.

(Q) Compute the influence.

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

Let's assume that data (x, y) is $(1, 1)$, then compute the influence of w change on E when the current w is equal to 3.

Method1 numerical gradient



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

w: 3 -> E: 4

w: 3.00001 -> E: 4.00004

Δw=0.00001

ΔE=0.00004

Slope =

Influence of w change = 4

$$\frac{\Delta E}{\Delta w} = \frac{0.00004}{0.00001} = 4$$

Method2 derivative, differential equation

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

$$\lim_{\Delta w \to 0} \frac{\Delta E}{\Delta w} = \frac{\partial E}{\partial w} = \frac{\partial}{\partial w}(w \cdot 1 - 1)^2$$

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

Therefore, when $w = 3$,
the gradient is $2(3 - 1) = 4$

How to update w (Learning)

1. Initialize w with a random value (*ex, 3*)

Loop

2. Get the influence(slope) of w on E
3. To decrease the error, update w using below eq:

Parameter Tuning

$$W = W - \alpha * \text{slope}$$

4. Go to step 2

TensorFlow

- Machine learning framework by Google
- Tuning parameters including w automatically instead of us
- Define w inside of TensorFlow to be tuned (managed) by it
- Hypothesis and cost_function(E)

Linear Regression using TF

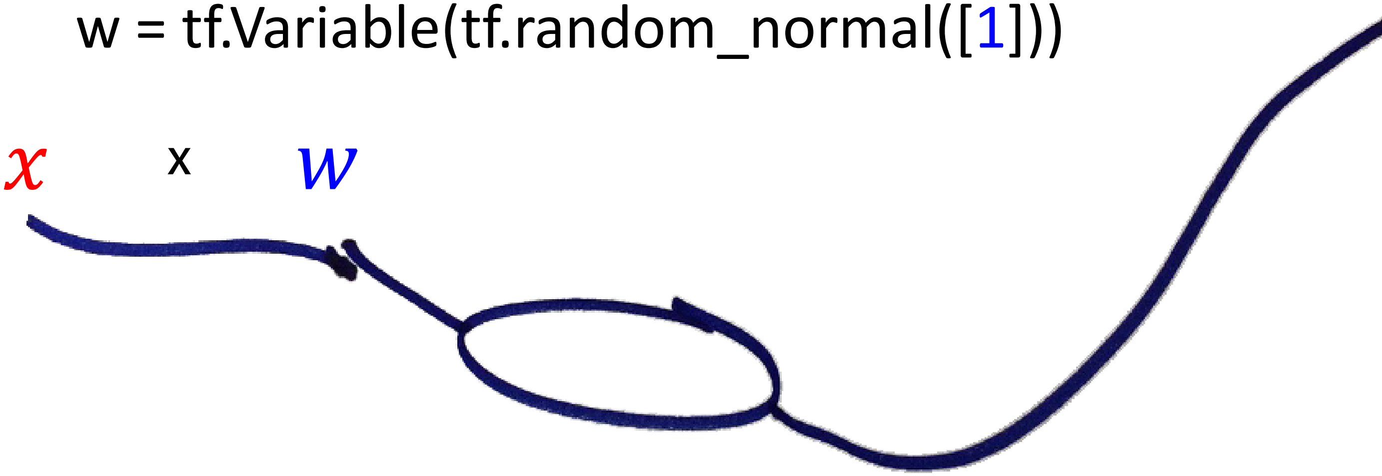
③

w = tf.Variable(tf.random_normal([1]))

hypo=$w * x$

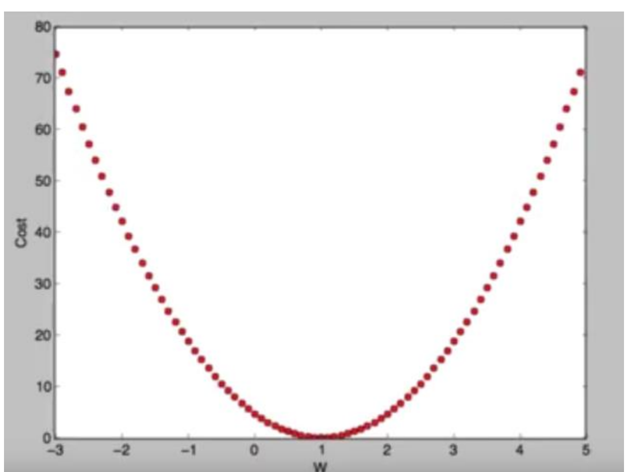
h ④

x x w



① x = [1]

y = [1] ②



cost_function = (hypo – y) ** 2

⑤

$$E = (\text{hypo} - y)^2$$

Download myml.git

https://github.com/yungbyun/myml.git

1) Run DOS prompt
2) git clone https://github.com/yungbyun/myml.git
3) Open using PyCharm (File | Open...)

Lab 01.py

Finding *w* in linear regression

You also can find the source code here at Kaggle.com.
https://www.kaggle.com/yungbyun/a-simple-neuron

[FYI] Change the first line as below:

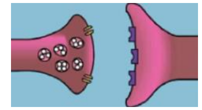
```
import tensorflow as tf
```

↓

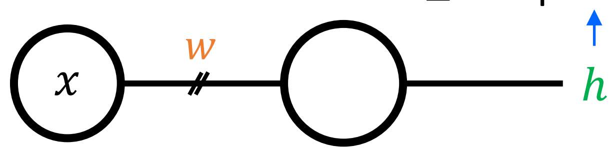
```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

```python
import tensorflow as tf
```

```python
#----- training data
x_data = [1]
y_data = [1]
```



$$E = |w \cdot x - y| ** 2$$




```python
#----- a neuron / neural network
w = tf.Variable(tf.random_normal([1]))
hypo = w * x_data
```

train operation to update w to minimize error(E)

```python
#----- learning
cost = (hypo - y_data) ** 2

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(1001):
    sess.run(train) #1-run, 1-update of w -> 1001 updates

    if i % 100 == 0:
        print('w:', sess.run(w), 'cost:', sess.run(cost))
```

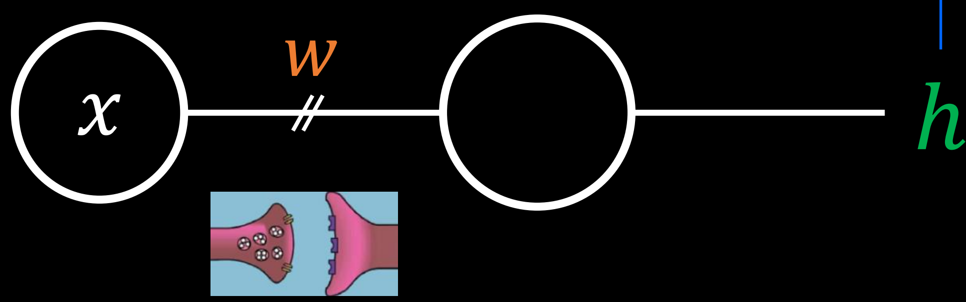
```python
#----- testing(prediction)
x_data = [2]
print(sess.run(x_data * w))
```

```
sess.run(train)
```

How to update w in TensorFlow

Computation Graph

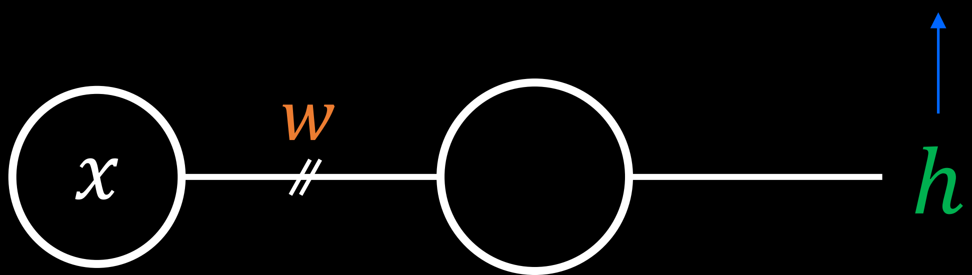
$$E = (h - y)^2$$



x	y
1	1
2	2

a.csv

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



Loss/Error function

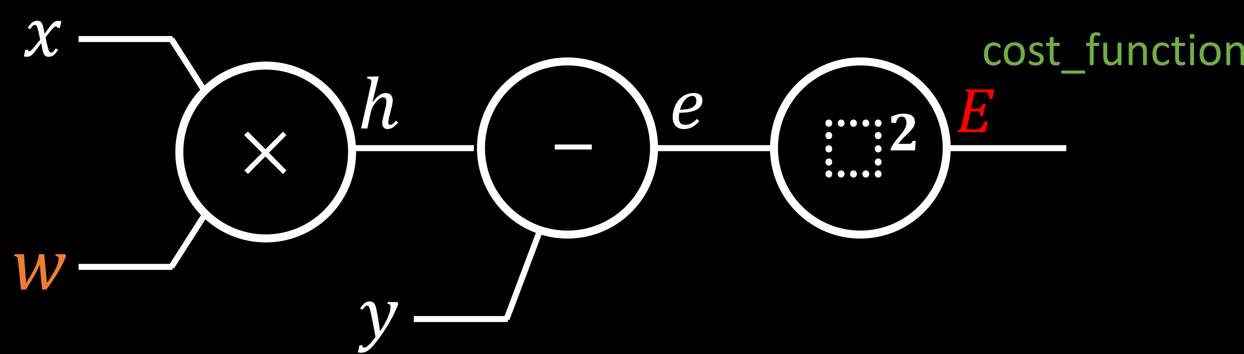
$$E = (wx - y)^2$$

- The part representing a neuron
- Where is a synapse?
- Which one is an input data?
- The output of a neuron
- **Find hypothesis**
- Find a correct answer or ground truth.
- Imagine E having multiple inputs.

Computation Graph of E

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

hypo = w * x
cost_function(E) = (hypo - y) ** 2)

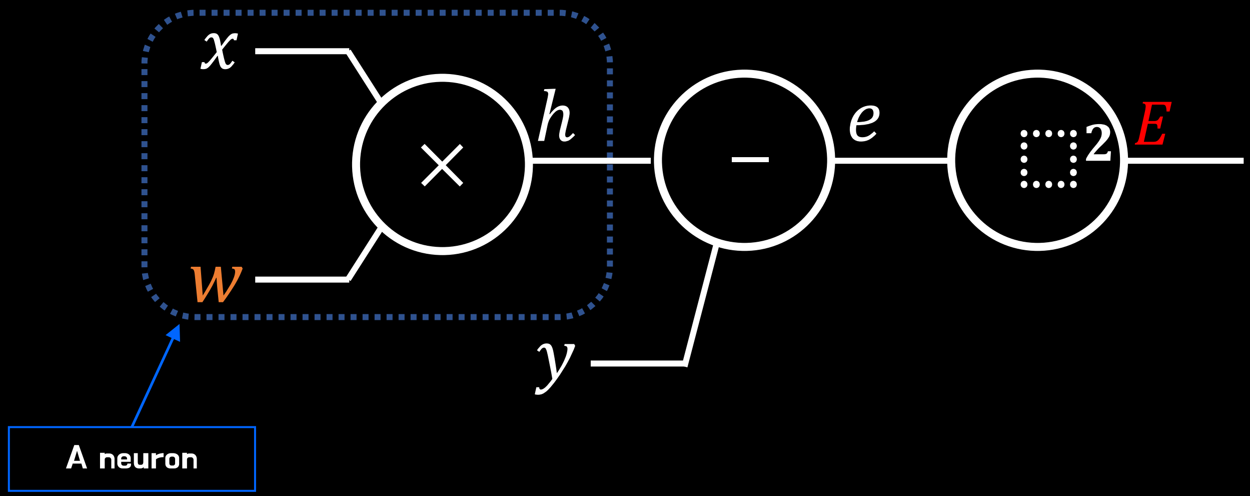


x	y
1	1
2	2

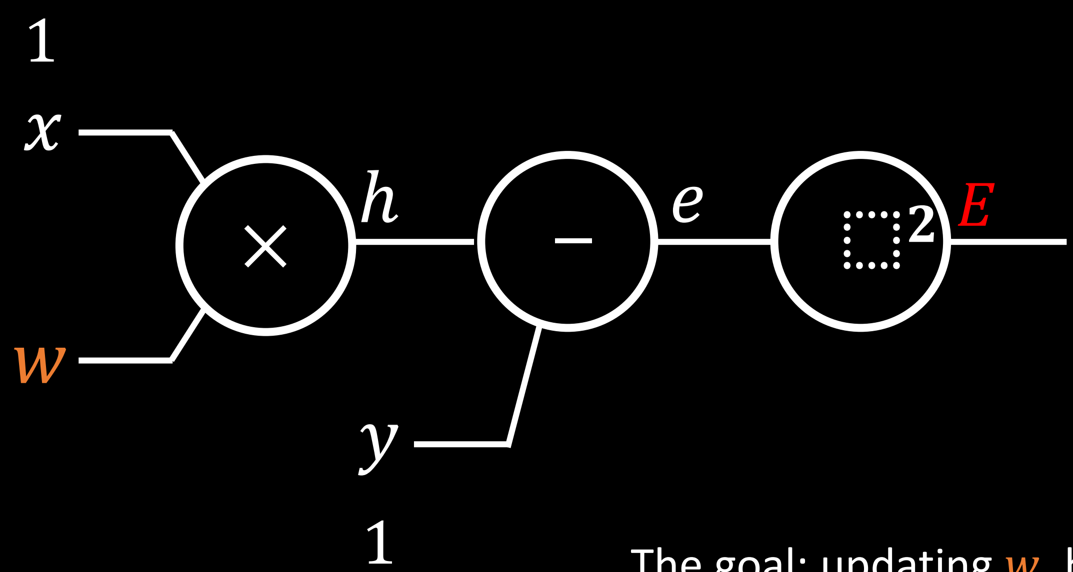
a.csv

Computation Graph of E

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



Computation Graph of E

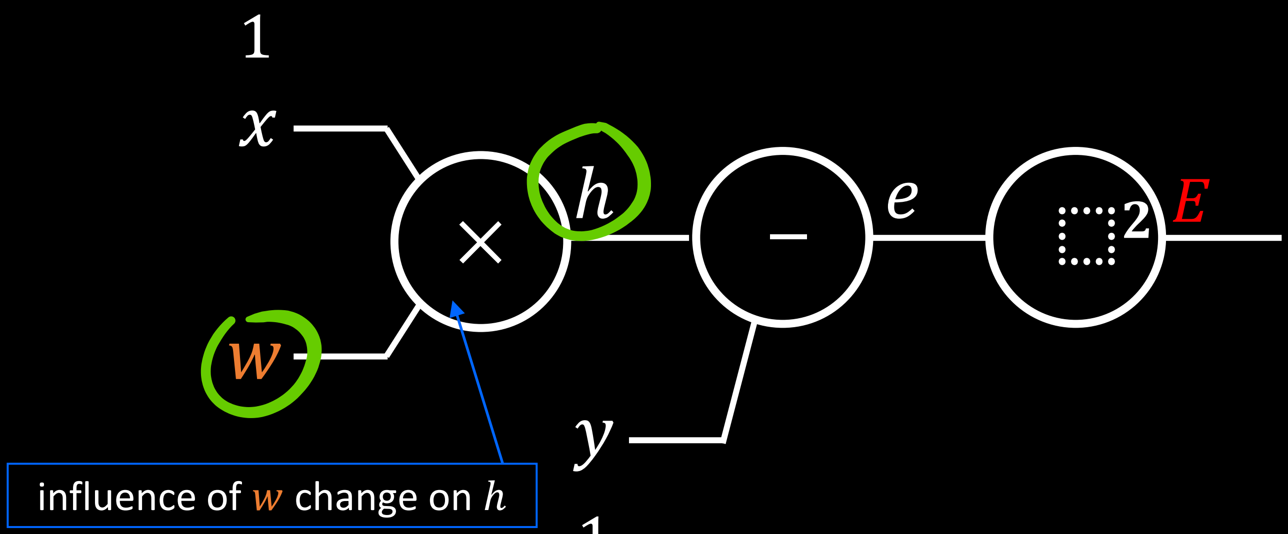


The goal: updating w, how?

x	y
1	1

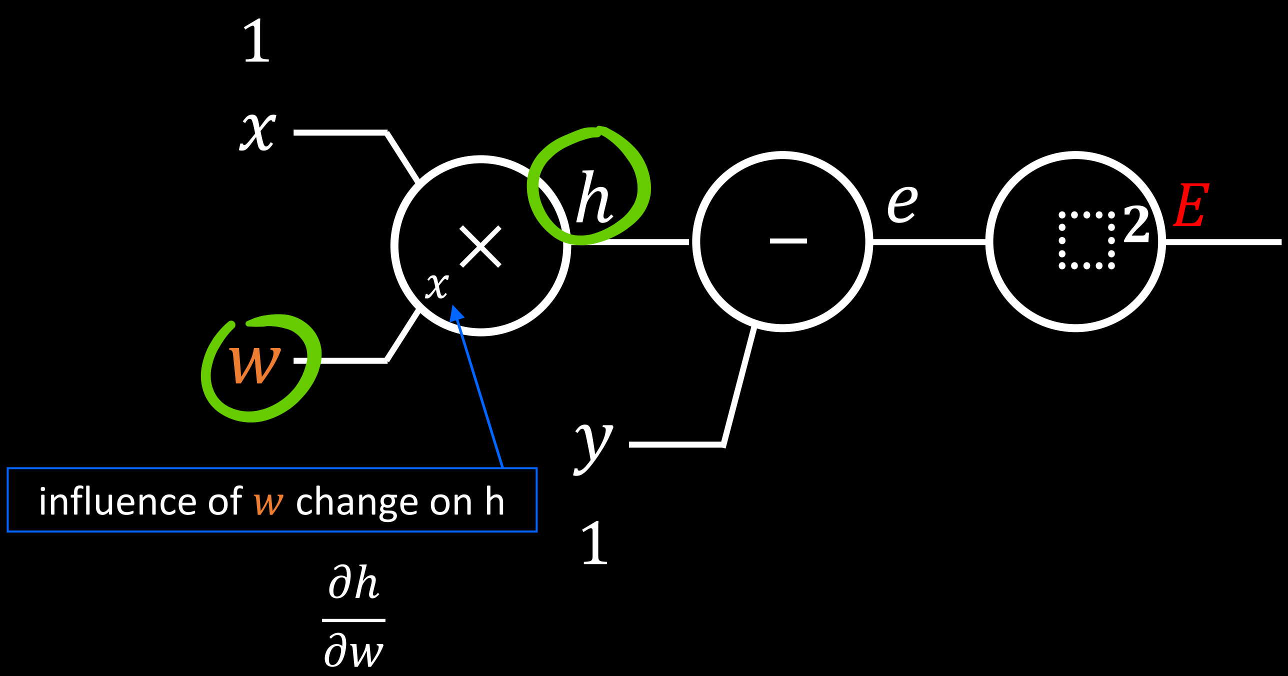
a.csv

Computation Graph of E



$$\frac{\partial h}{\partial w}$$

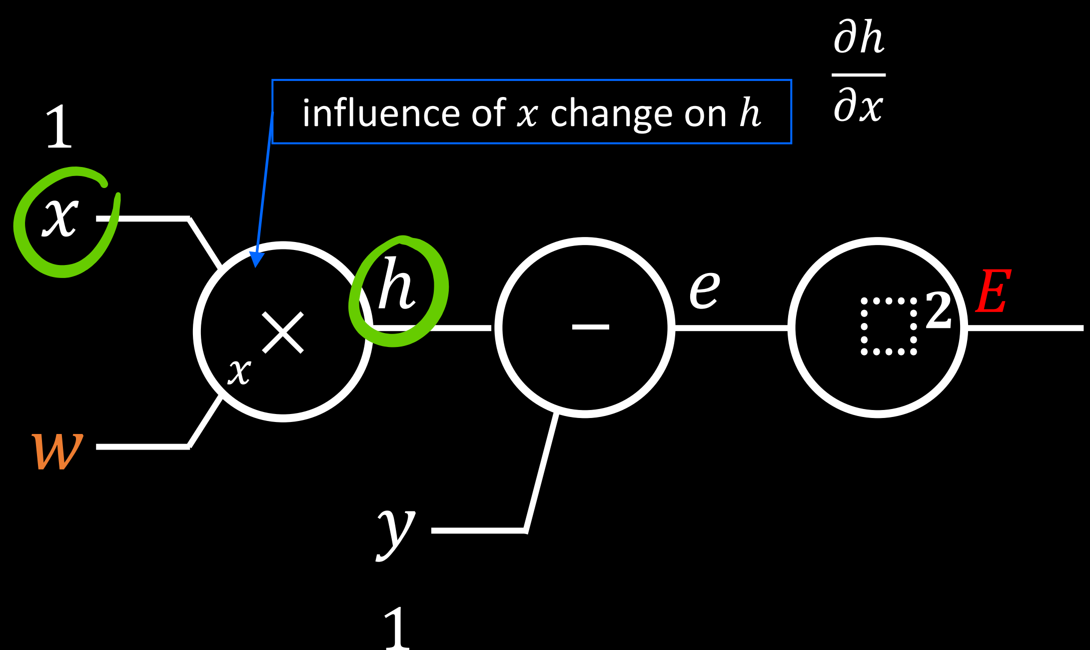
Computation Graph of E



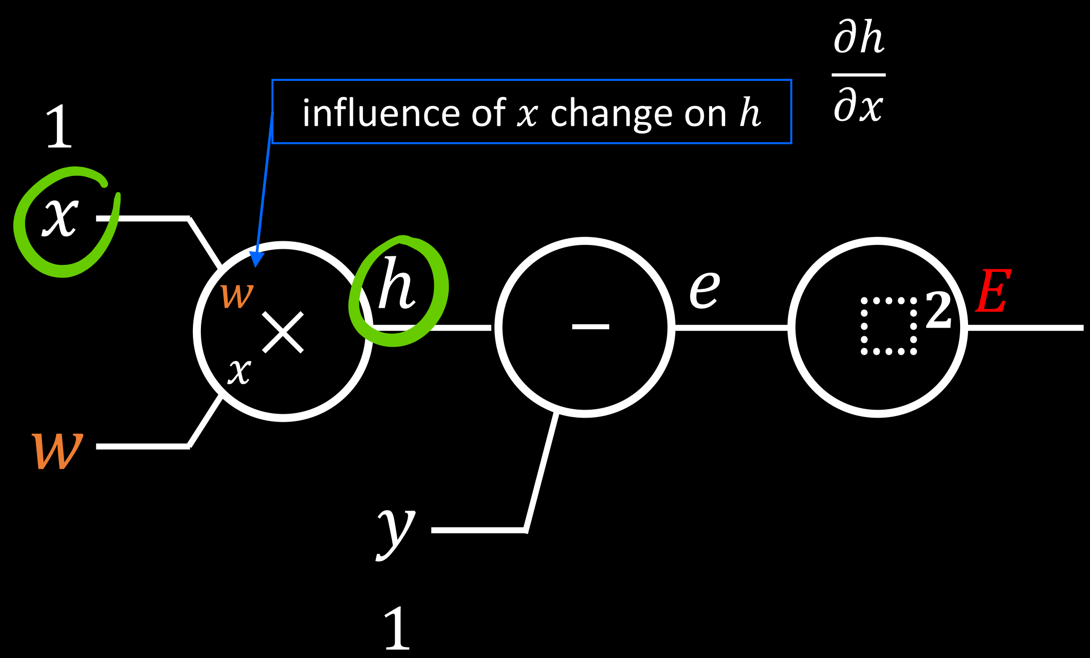
Local gradient

지역 기울기

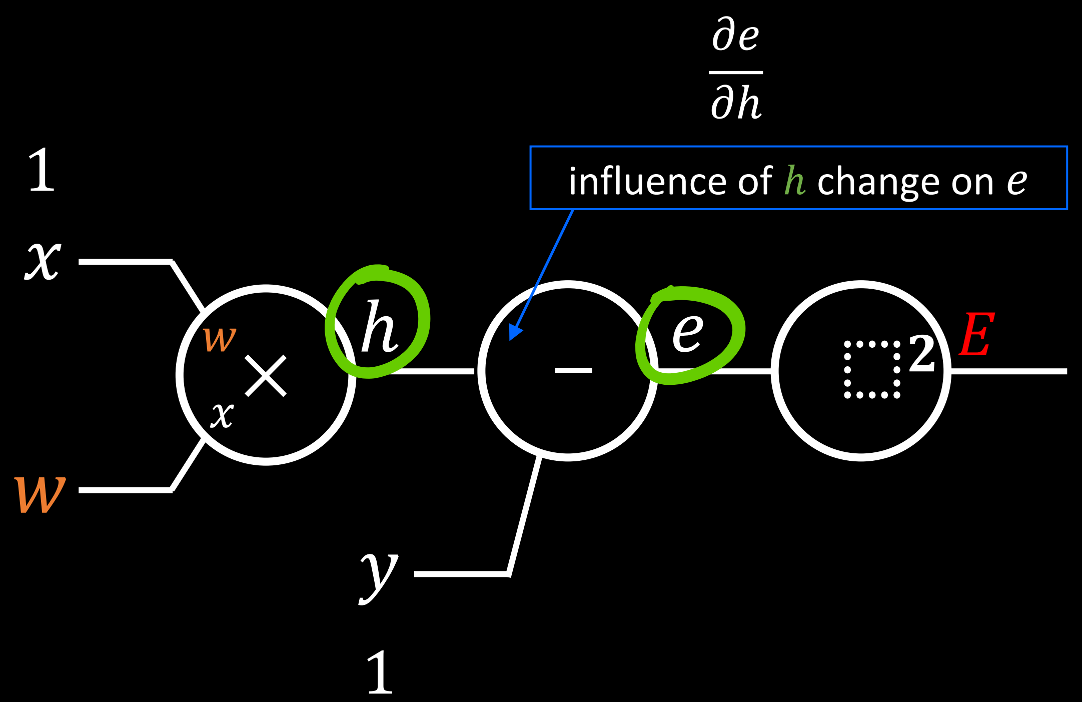
Computation Graph of E



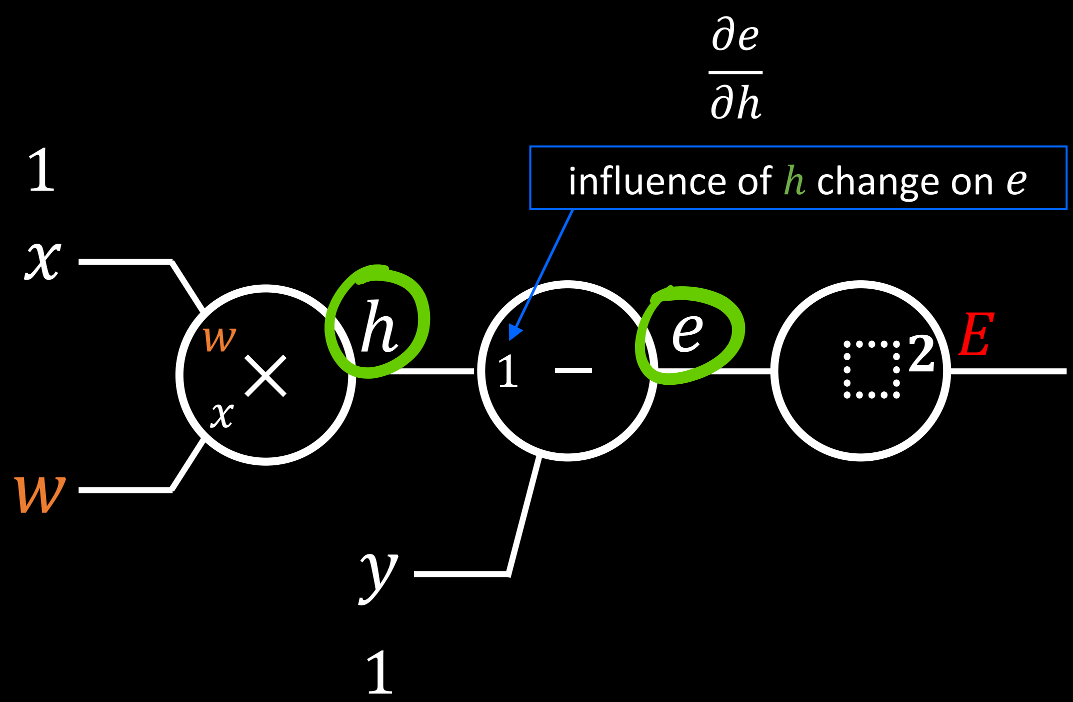
Computation Graph of E



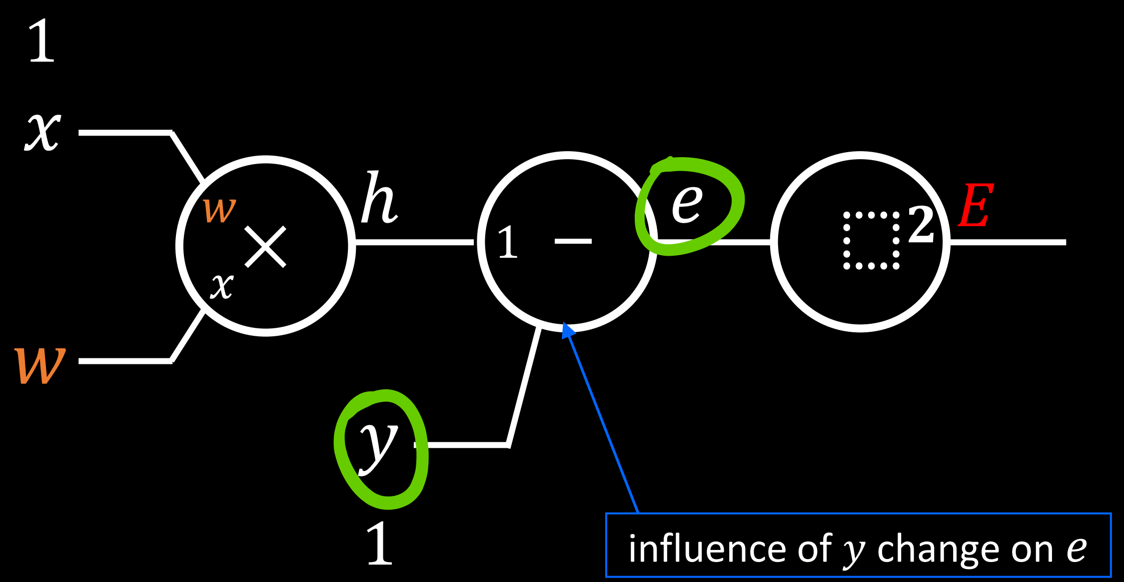
Computation Graph of E



Computation Graph of E

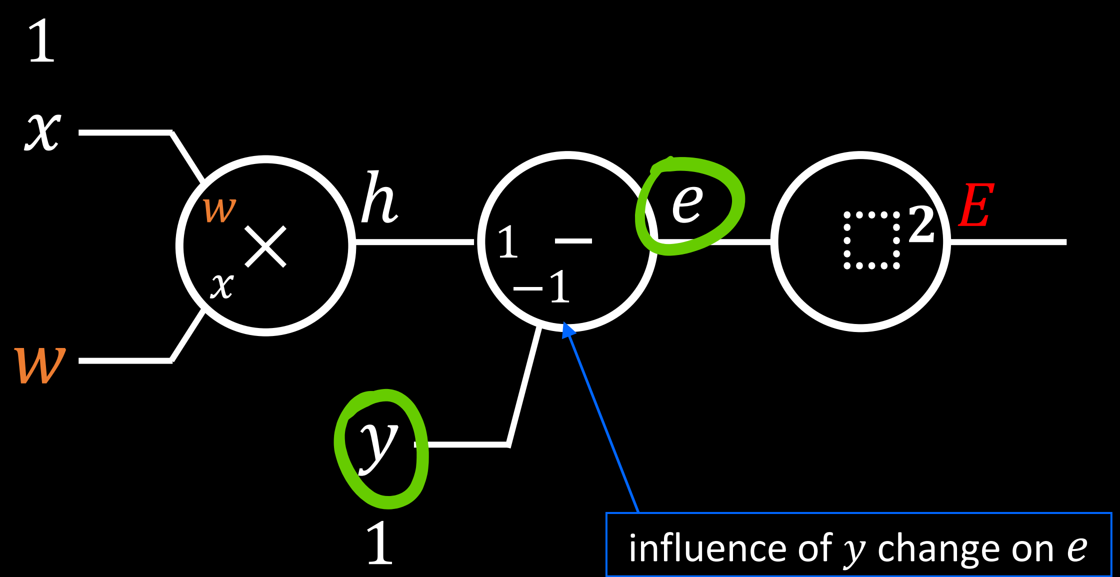


Computation Graph of E



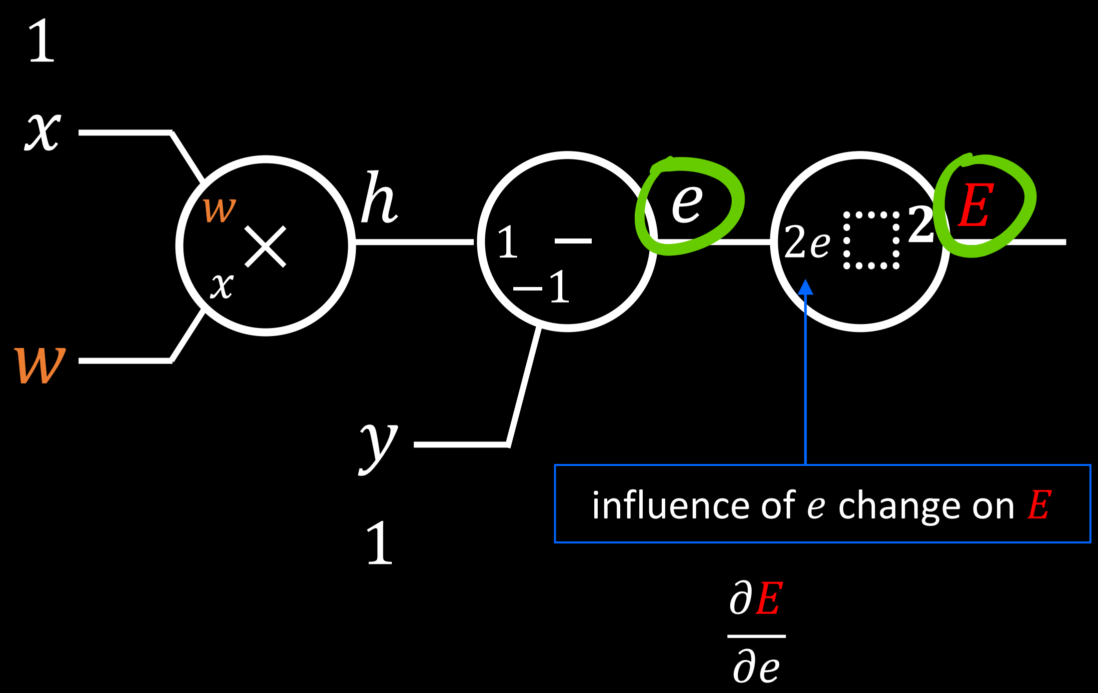
$$\frac{\partial e}{\partial y}$$

Computation Graph of E

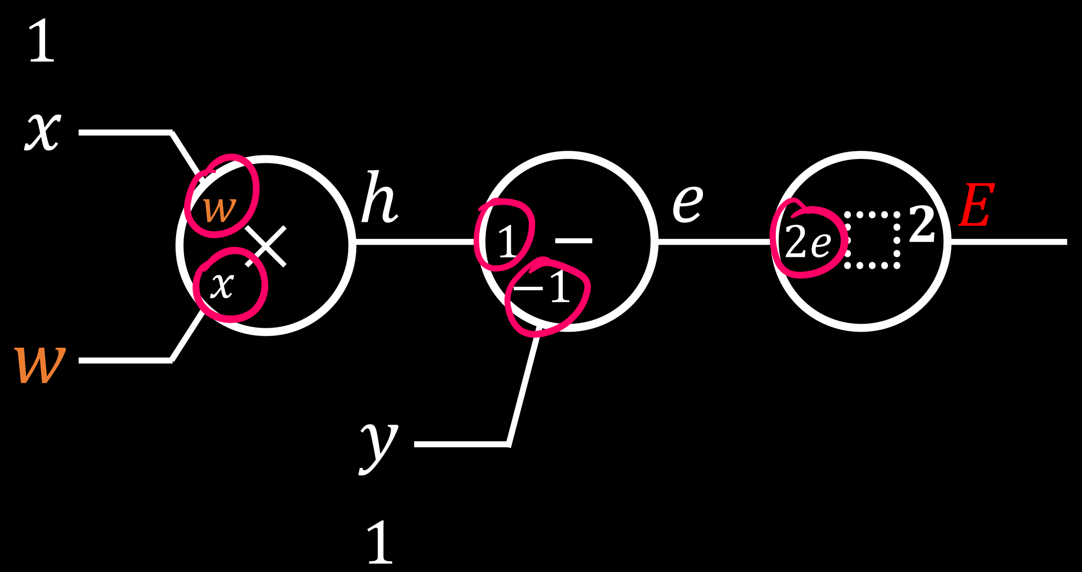


$$\frac{\partial e}{\partial y}$$

Computation Graph of E



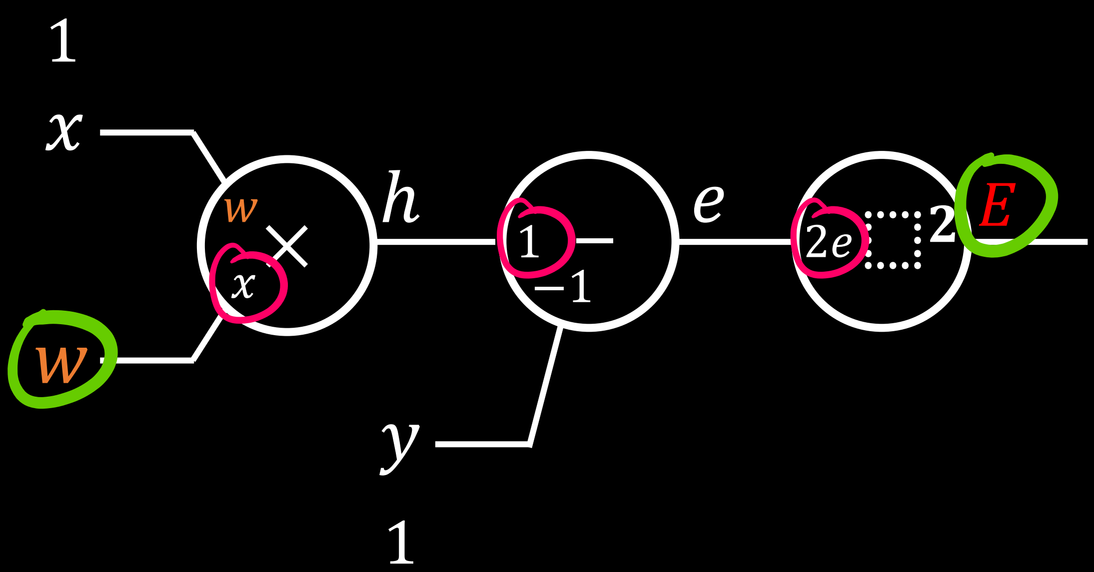
5 Local Gradients in gates



“

How can we get the influence of w change on E?

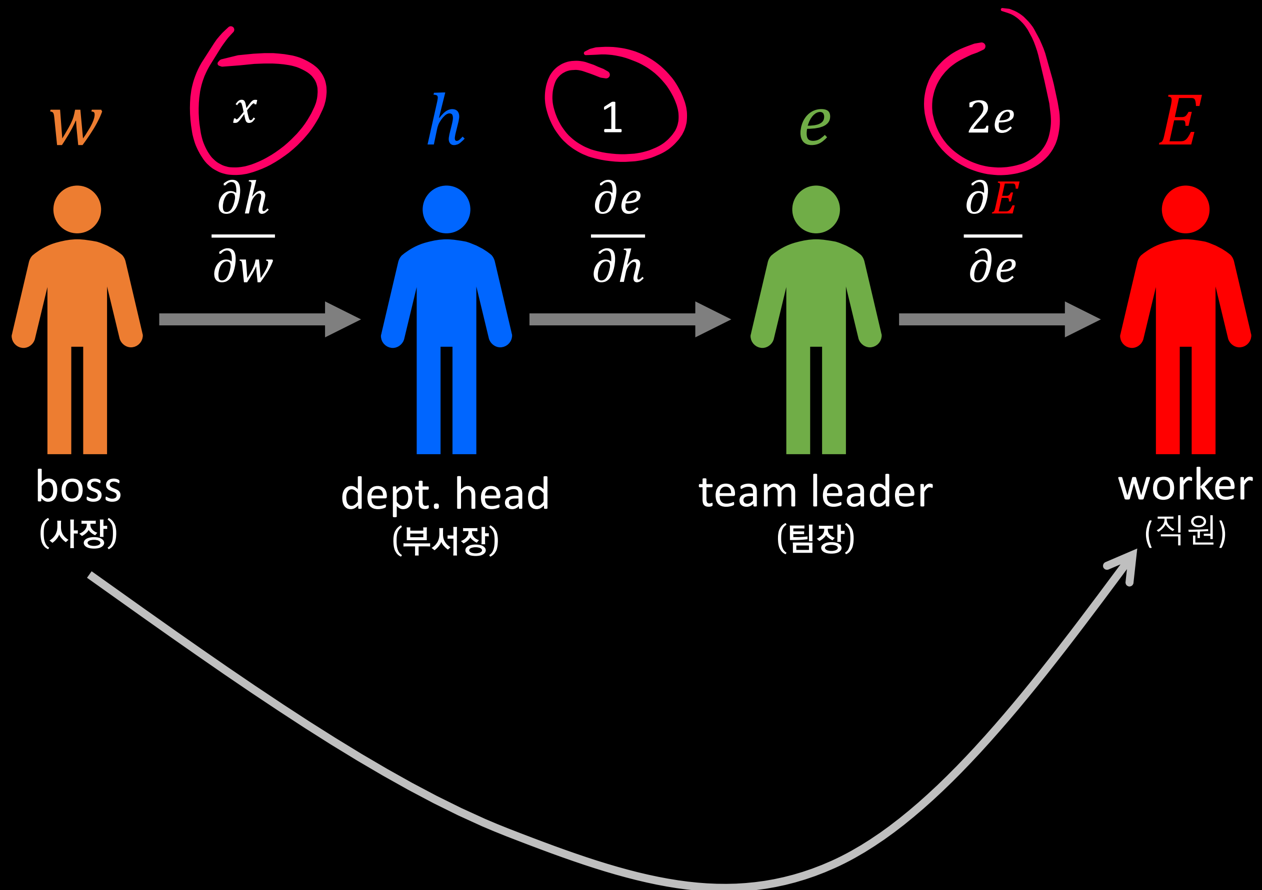
3 Local Gradients in gates



“

How can we get the influence of w change on E?

Influence between persons



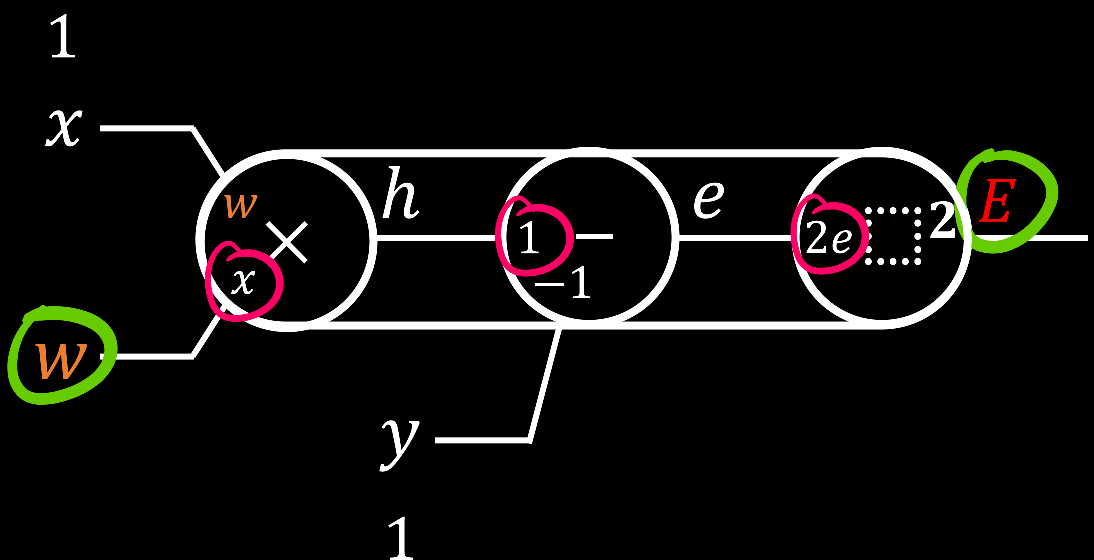
$$\frac{\partial E}{\partial w} = \frac{\partial h}{\partial w} \cdot \frac{\partial e}{\partial h} \cdot \frac{\partial E}{\partial e} = x \cdot 1 \cdot 2e$$

The influence of w change on E

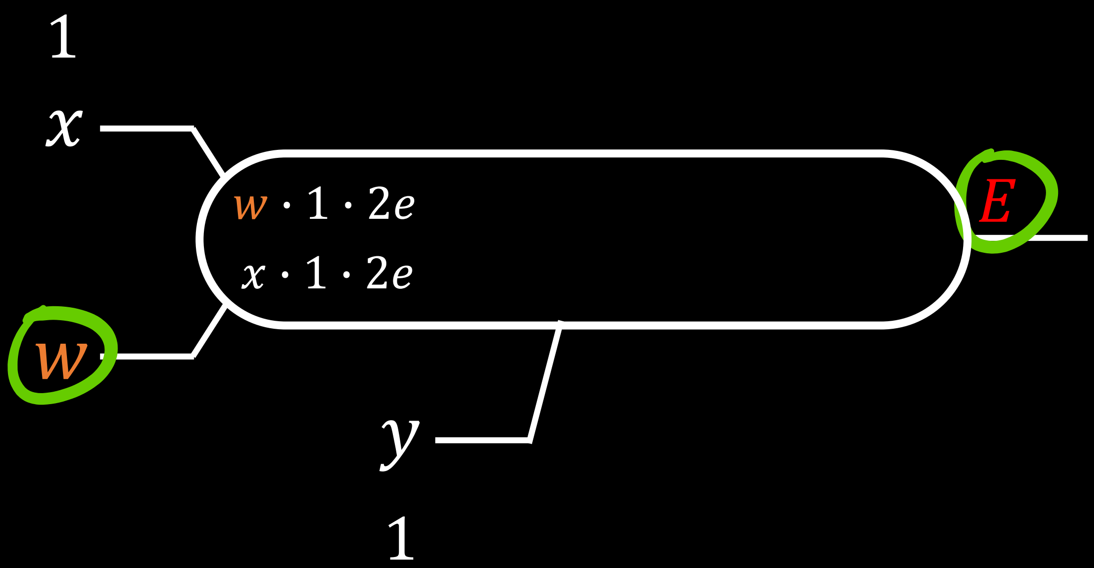
$$\frac{\partial E}{\partial w} = \frac{\partial h}{\partial w} \times \frac{\partial e}{\partial h} \times \frac{\partial E}{\partial e}$$

Chain rule

Merging gates

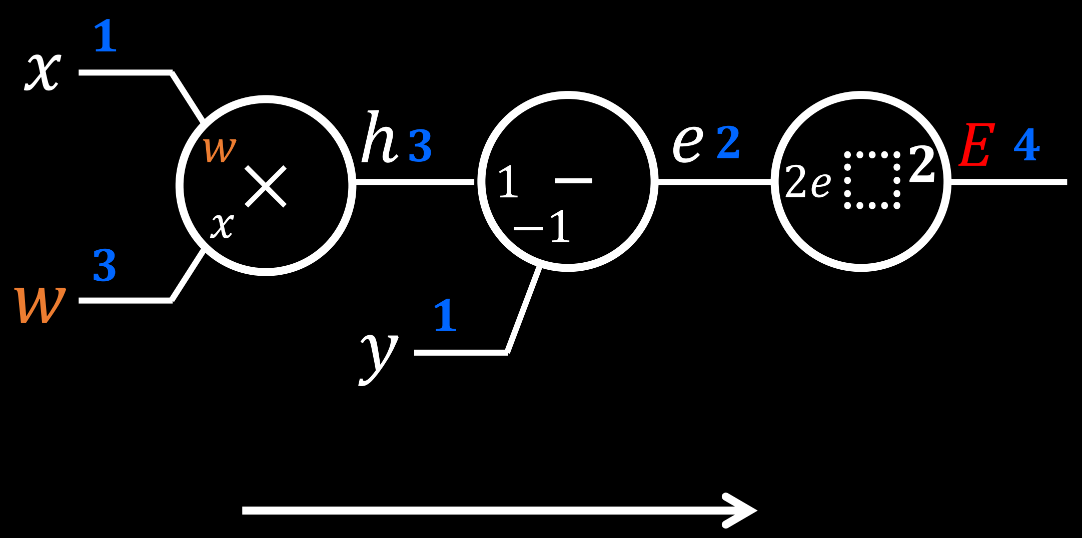


Composite gates



$Forward$ propagation

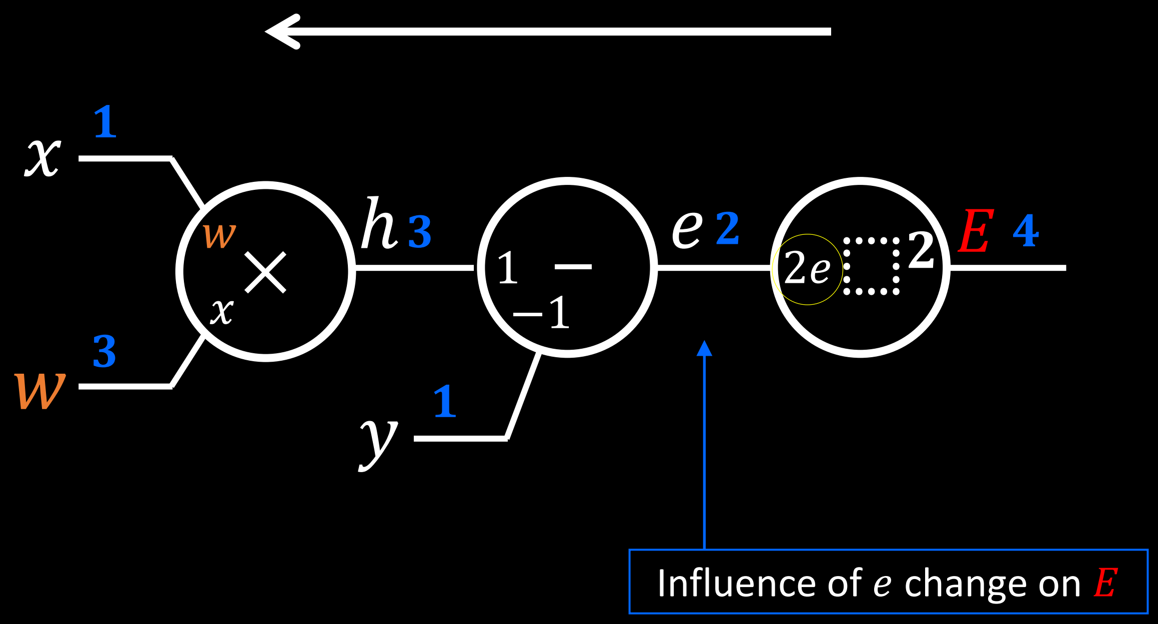
Let $(x, y) = (1, 1)$ and $w = 3$, then compute E.



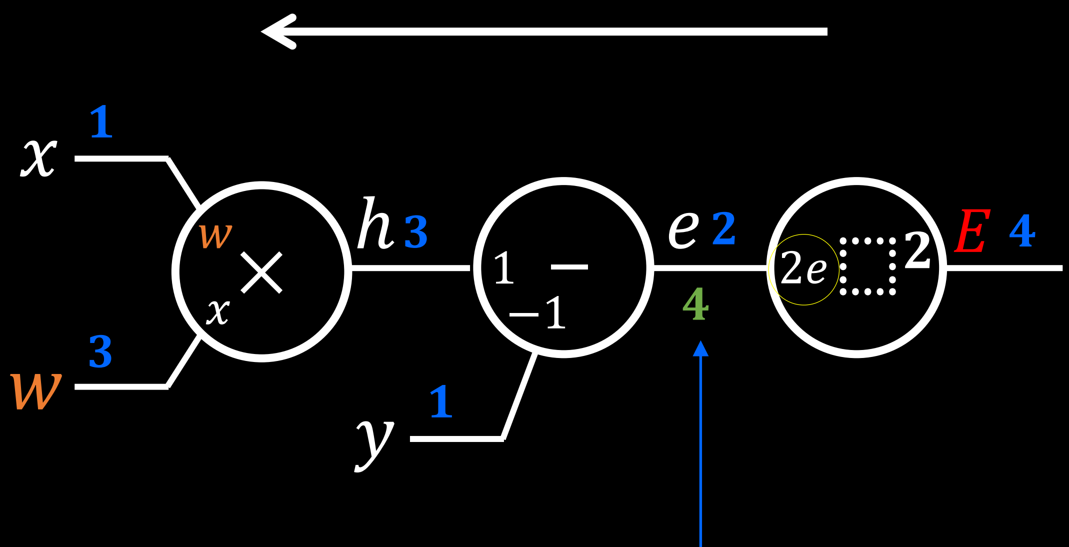
Error is big (4) ,
so, let's update w

using *back*-propagation.

$Back$-propagation



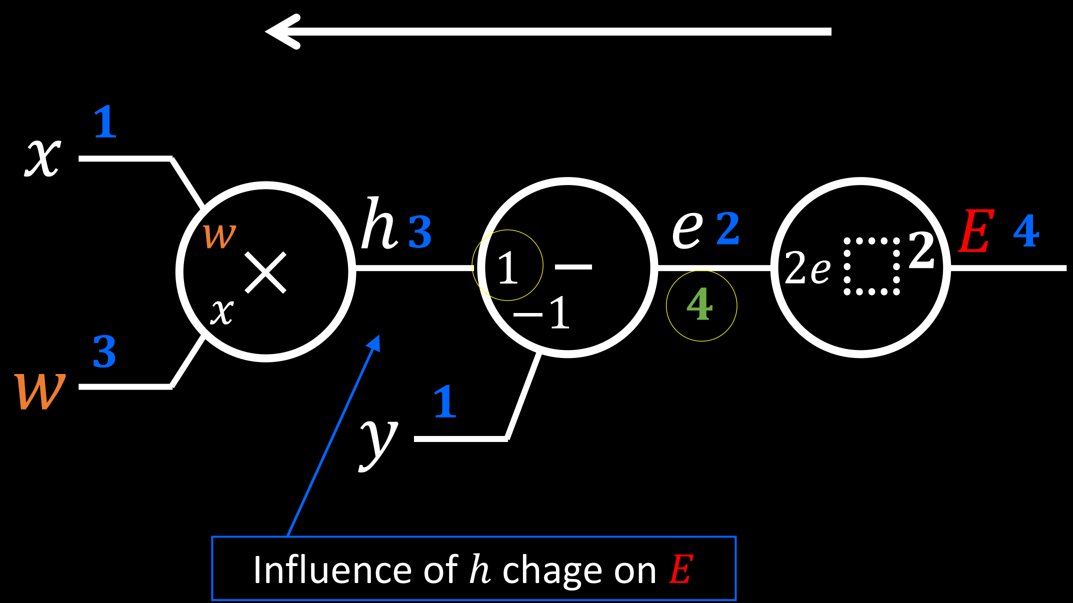
$Back$-propagation



Influence of e change on E

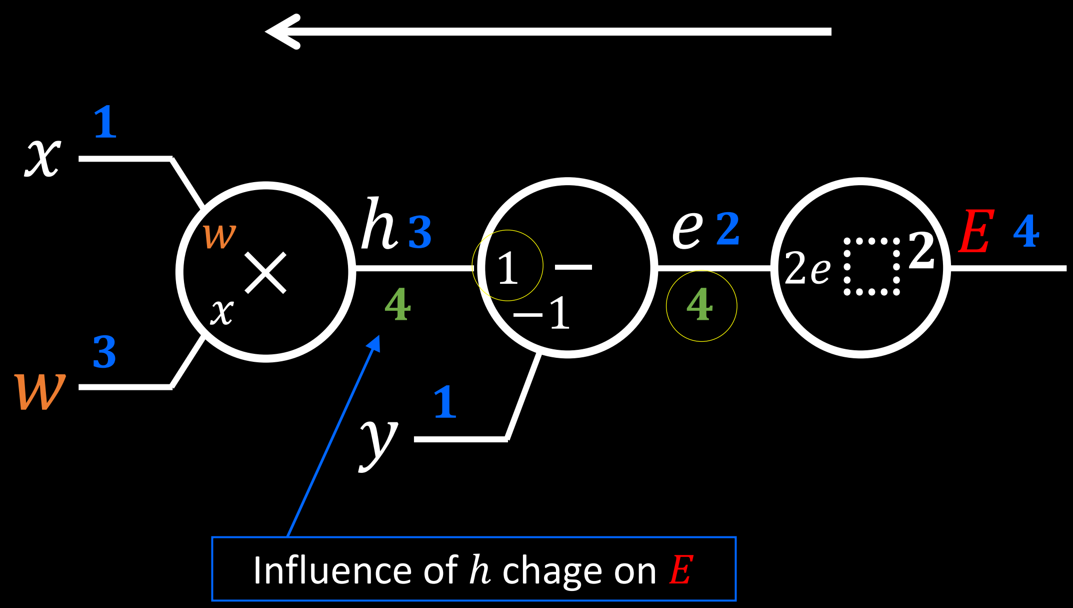
$$\frac{\partial E}{\partial e} = 2e$$

$Back$-propagation

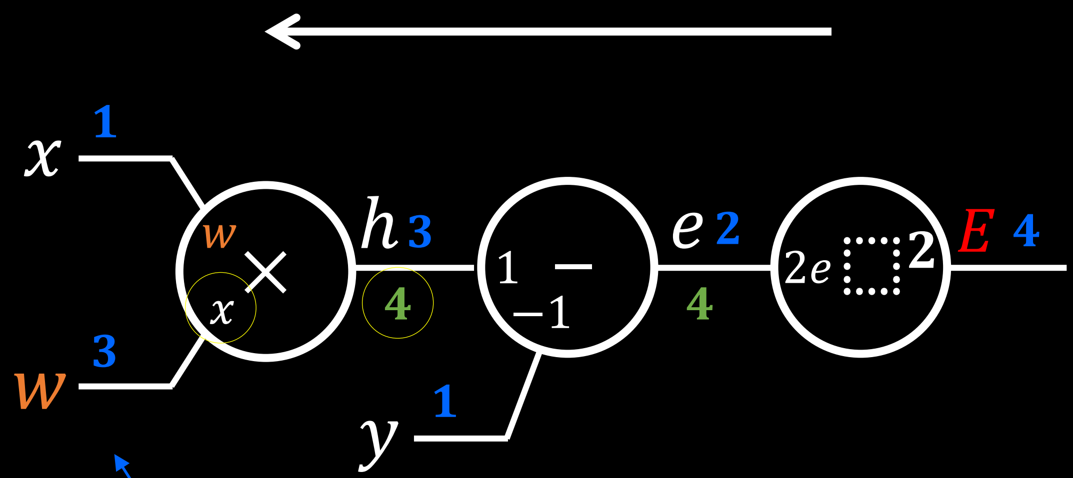


$$\frac{\partial E}{\partial h} = \frac{\partial e}{\partial h} \cdot \frac{\partial E}{\partial e} = 1 \cdot 4$$

$Back$-propagation

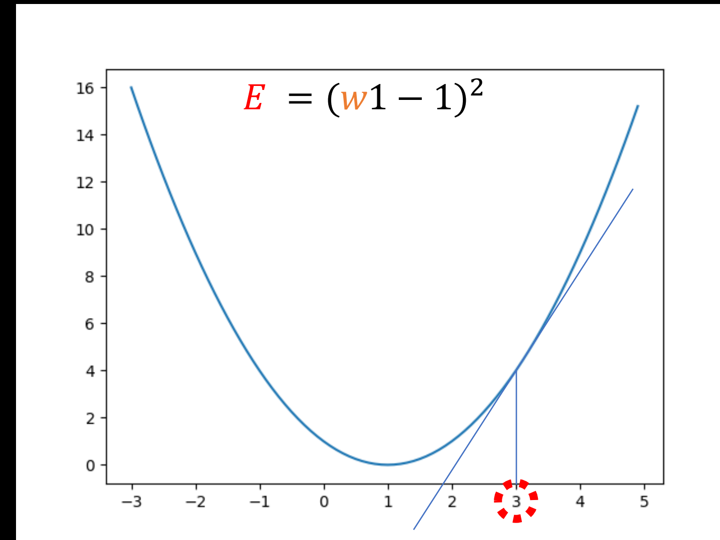


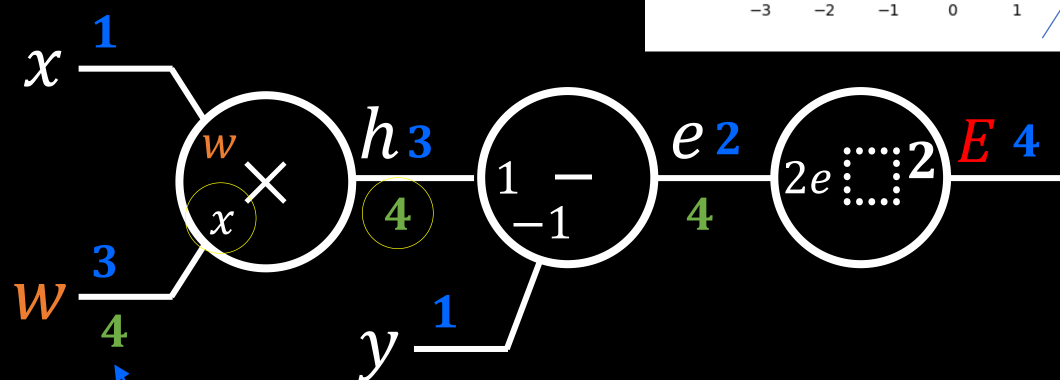
$Back$-propagation



Influence of w change on E

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





Influence of w change on E

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

$Back$ − propagation, the process to apply chain rules

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

Gradient decent (경사하강)
using slope

$$w = 3 - 0.1 * 4 \quad \frac{\partial E}{\partial w}$$

$$w = 2.6$$

Tuned parameter after 1 step learning

After enough number of steps(epochs), the parameter w will be optimized properly.

$Back$–propagation

by Paul Webros (1974, 1982) and Geoffrey Hinton (1986)

Prof. Univ. of Toronto,
Google Brain
Yann LeCun (his post doc)

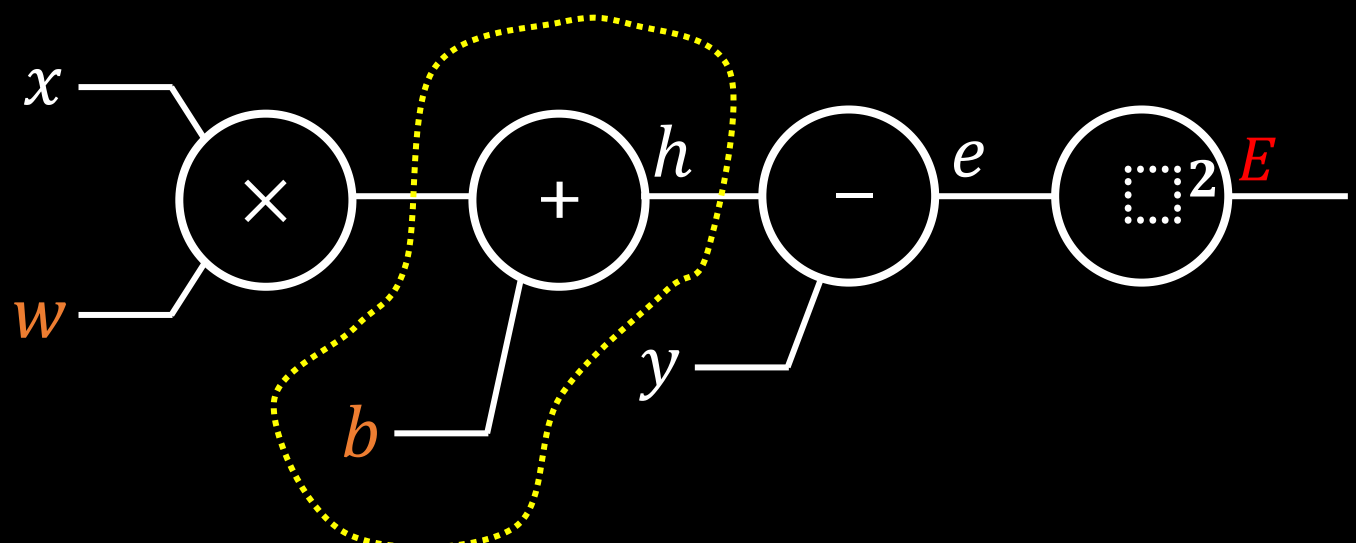
```python
import tensorflow as tf

#----- training data
x_data = [1]
y_data = [1]

#----- a neuron / neural network
w = tf.Variable(tf.random_normal([1]))
hypo = w * x_data

#----- learning
cost = (hypo - y_data) ** 2

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(1001):
    sess.run(train) # 1-run, 1-update of w -> 1001 updates

    if i % 100 == 0:
        print('w:', sess.run(w), 'cost:', sess.run(cost))

#----- testing(prediction)
x_data = [2]
print(sess.run(x_data * w))
```

train operation to update w to minimize cost(error)

Extension of the Graph

- adding bias/shift b (one more plus gate)

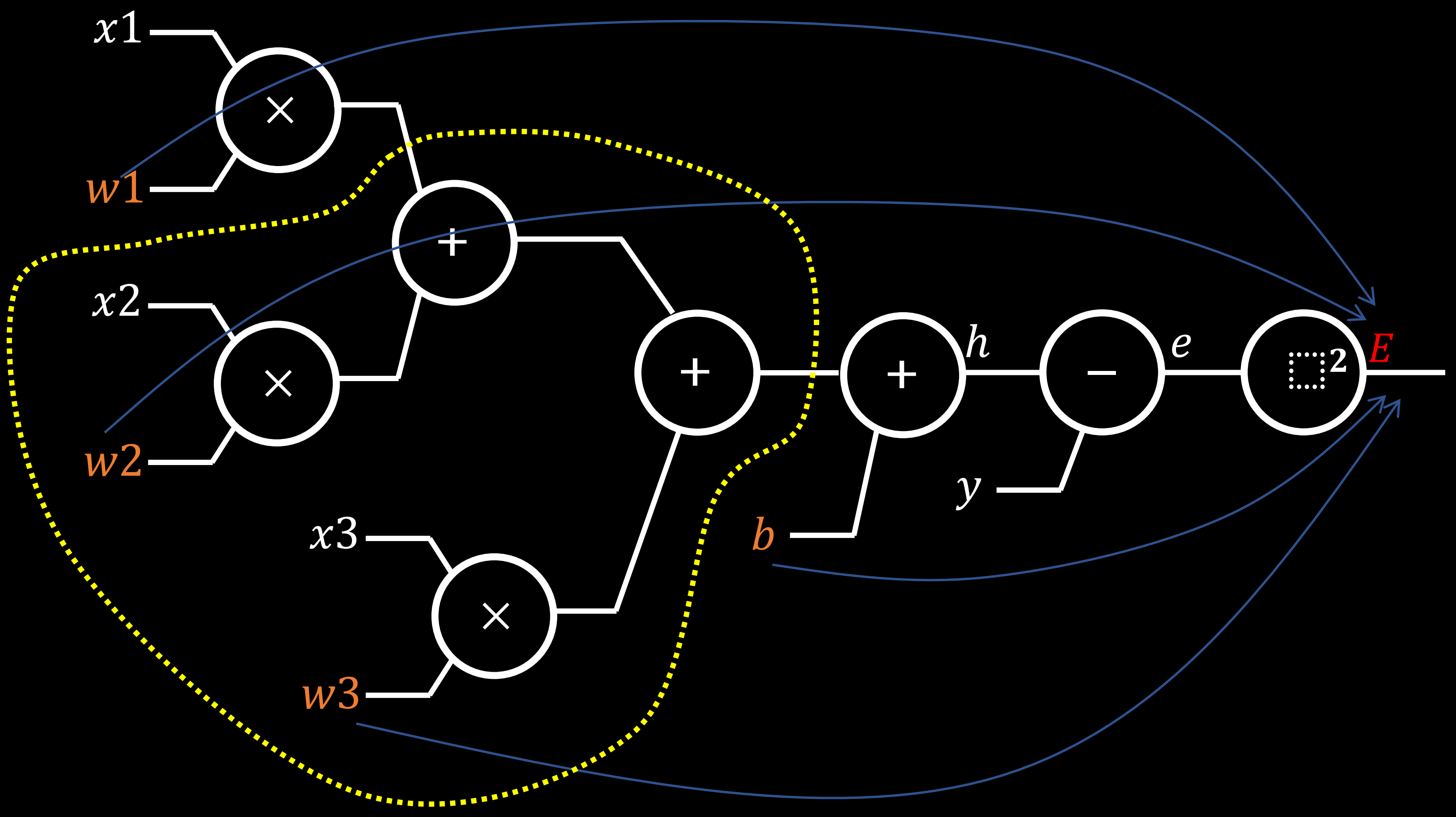
$$E = ((wx \boxed{+ b}) - y)^2$$



Extension of the Graph

- a neuron with 3 inputs (2 more + gate)

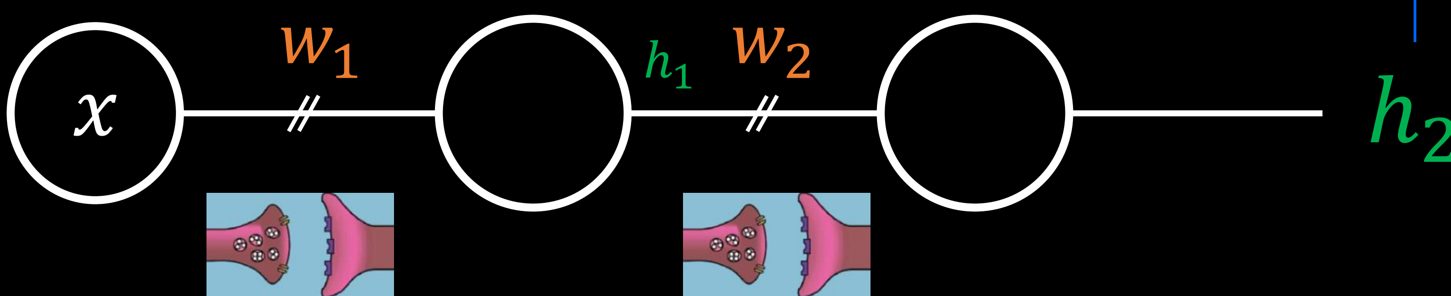
$$E = ((w1x1 \boxed{+ w2x2 + w3x3} + b) - y)^2$$



Extension of the Graph

- Two neurons, 3-layer

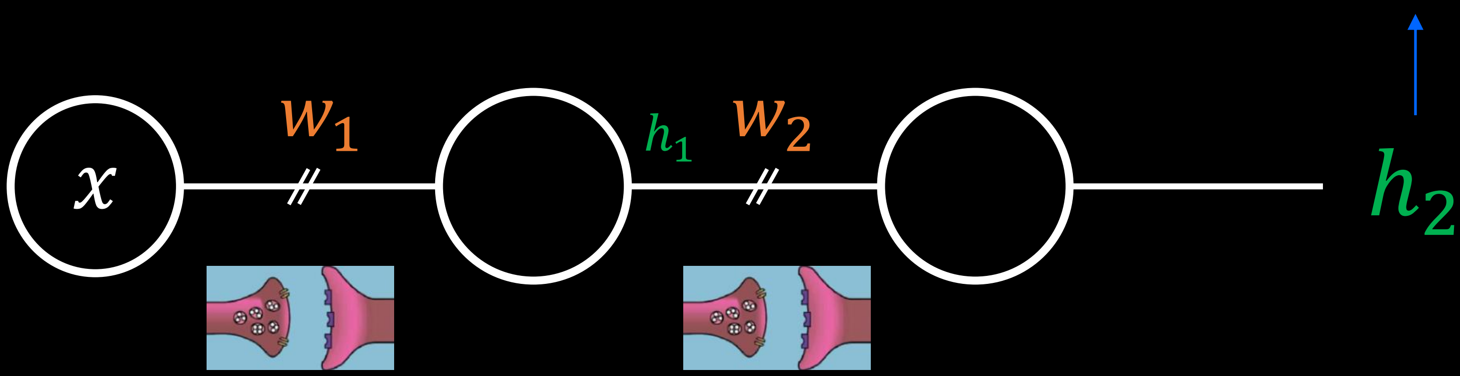
$$E = (w_2 h_1 - y)^2$$



Extension of the Graph

- Two neurons, 3-layer

$$E = (w_2 \cdot (w_1 x) - y)^2$$

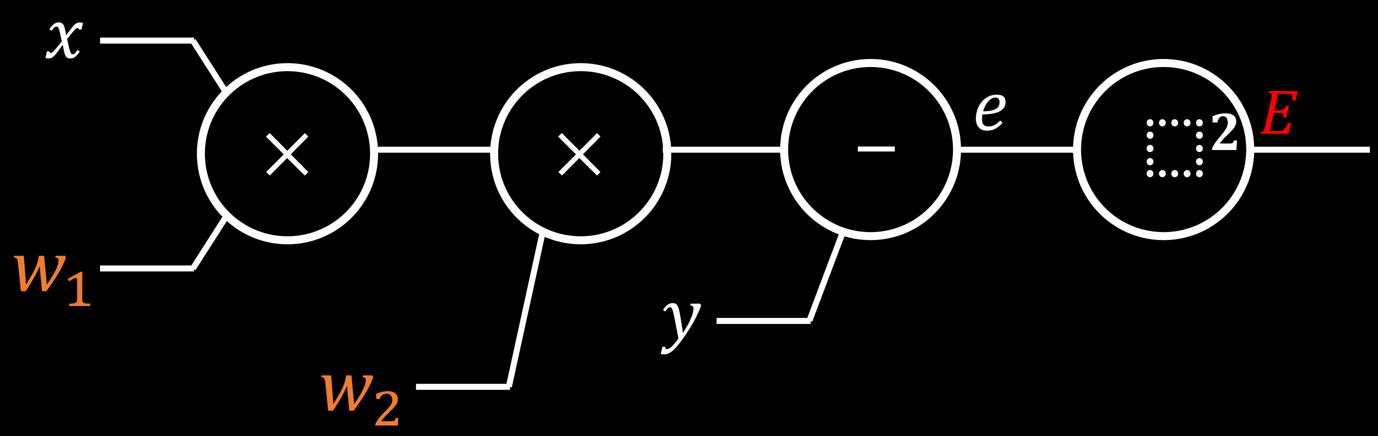


If the hypothesis(h_2) is the linear combination of coefficients(w_1, w_2), so it is a linear model.

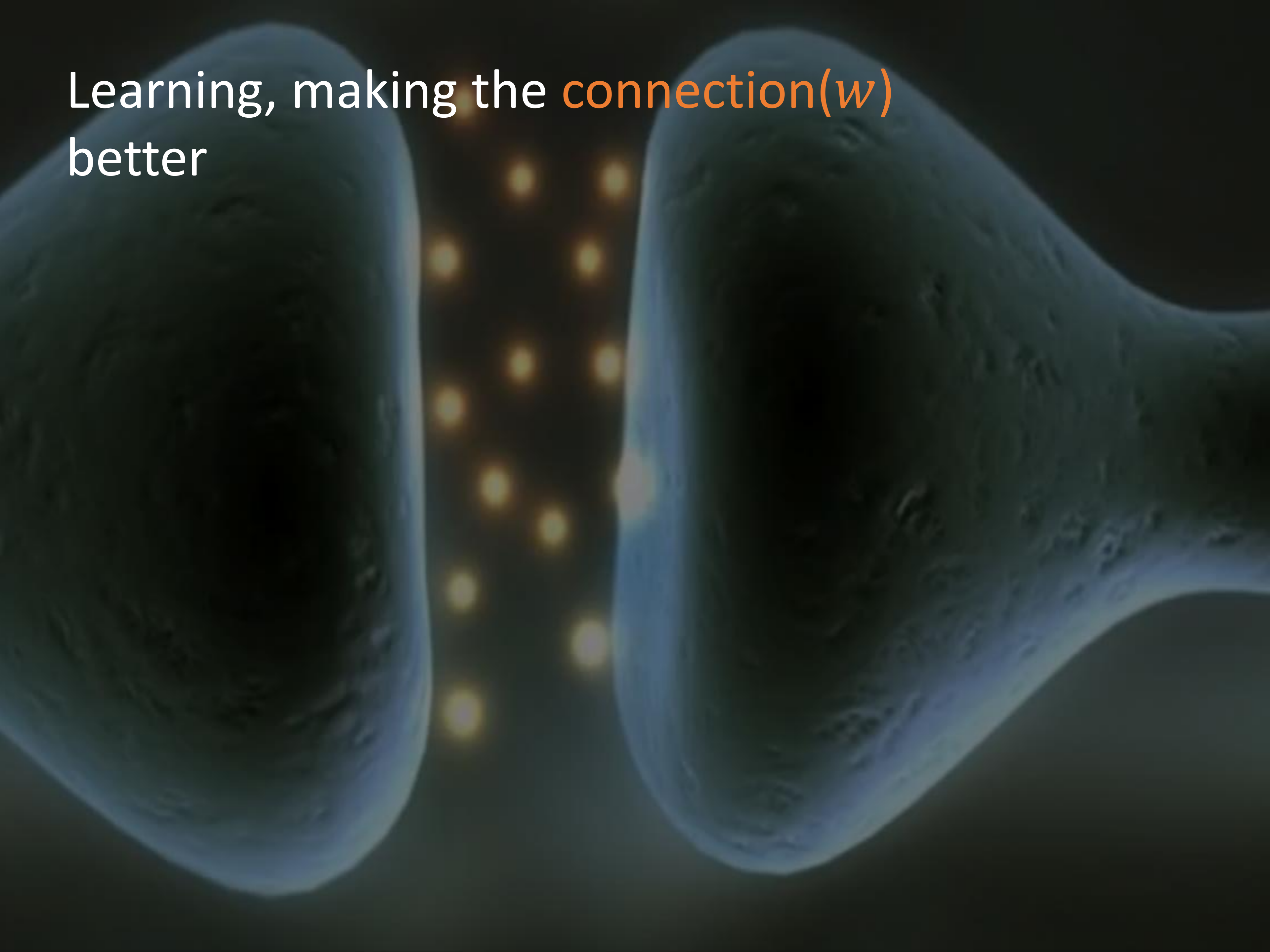
Extension of the Graph

- Two neurons, 3-layer

$$E = (w_2 \cdot (w_1 x) - y)^2$$

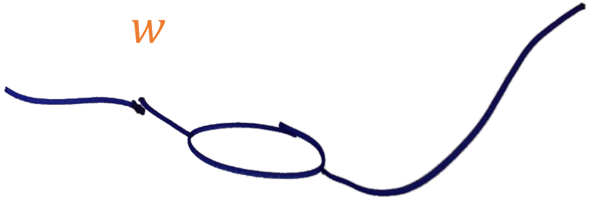


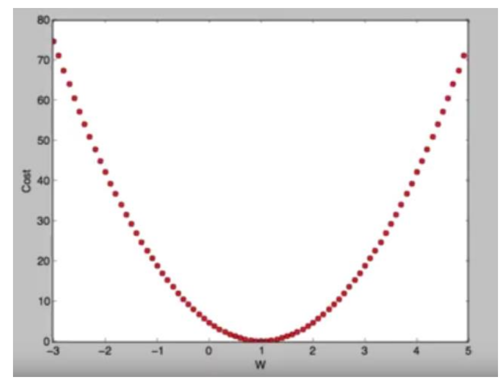
Learning, making the connection(w) better



Cost(Error) graph

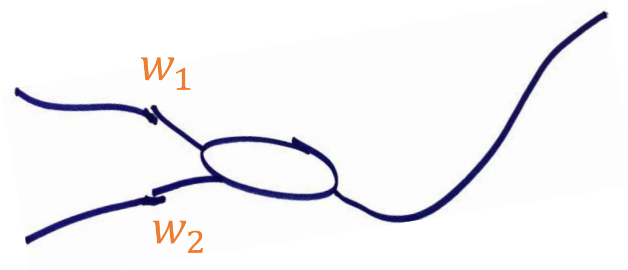
$$E = (w \cdot 1 - 1)^2$$

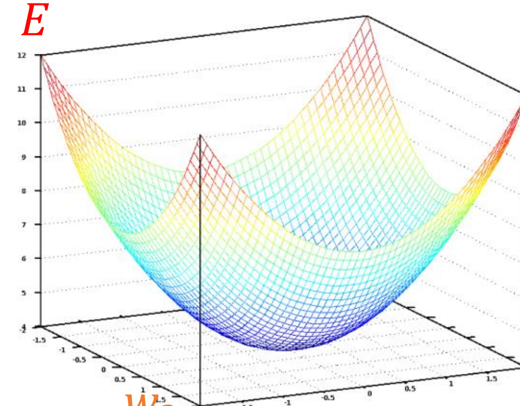




convex function

$$E = (w_1 \cdot 1 + w_2 \cdot 1 - 1)^2$$





convex function

Lab 02.with_bias.py

Parameter tuning including bias

Lab 03.py

Using multiple data

Lab 04.py

Training a neuron having multiple inputs