딥러닝프로젝트

과목 구성

1. 이미지 인식 기술을 이용한 프로젝트를 수행한다
2. 문장 인식 기술을 이용한 프로젝트를 수행한다.
3. 애니메이션 생성 기술을 이용한 프로젝트를 수행한다.
4. 소리 데이터 인식 기술을 이용한 프로젝트를 수행한다.
5. 언어 인식 기술을 이용한 프로젝트를 수행한다.

사전 필수 지식

1.파이썬
2.Numpy , pandas
3.pytorch

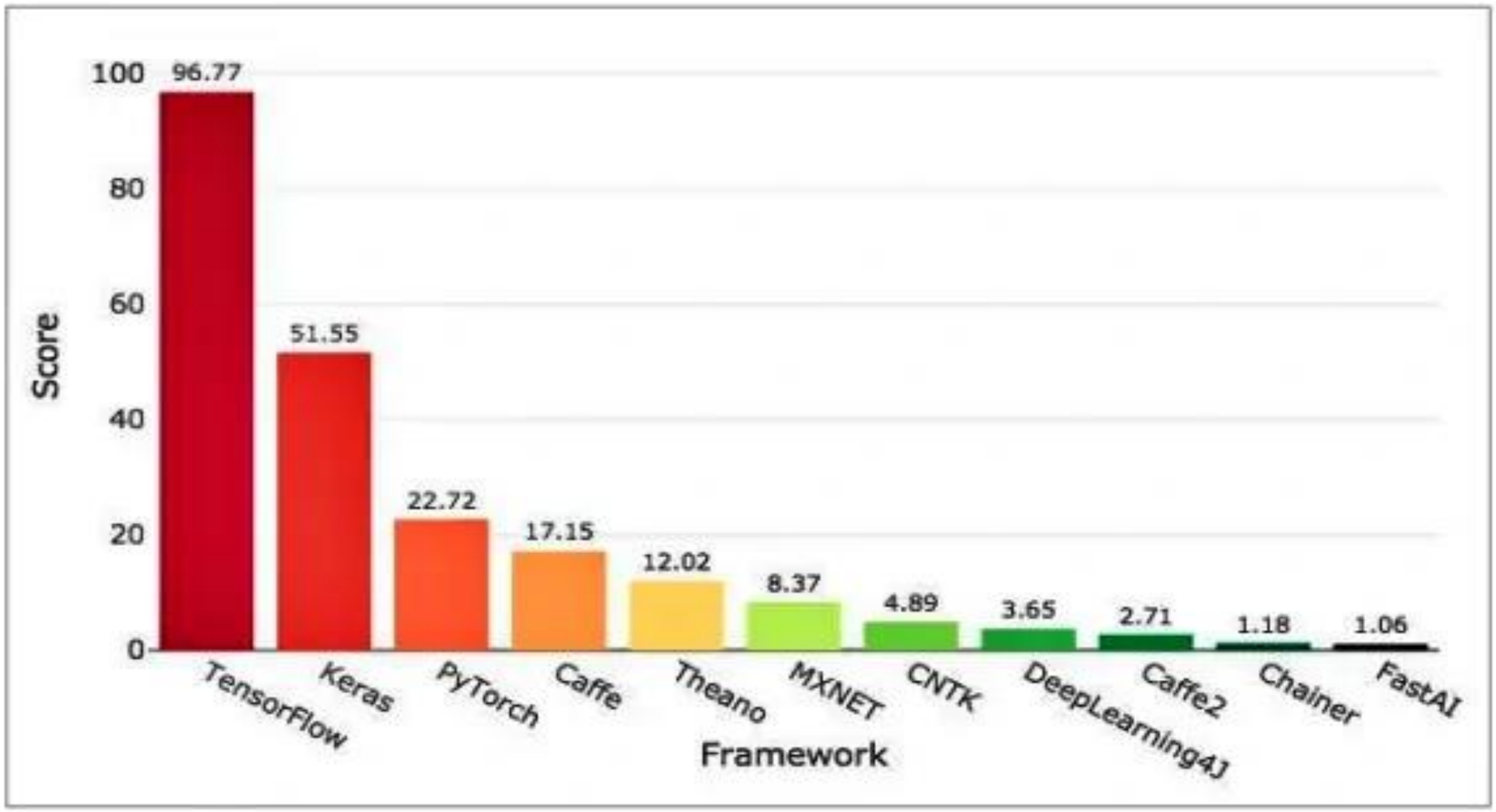
수업시간 :09:30~ 17:30
쉬는 시간: 15분
과제: 16:00~ 17:20
평가: 금요일 오후 1시간


기계학습
알고리즘
목표변수
있음
목표변수
없음
지도학습
비지도학습
분류
(Classification)
추정
(Estimation)
차원축소
(Dimension Reduction)
군집화
(Clustering)
연관성 규칙 발견
(Association Rule)
• KNN (K-nearest neighbors)
• Naive Bayes
• Decision Tree
• Logistic Regression
• Random Forest
• Support Vector Machine
• ANN (Artificial Neural Network)
• Linear Regression (Stepwise)
• Regularized Linear Regression
• Regression Tree
• Random Forest Regression
• Support Vector Regression
• PCA (Principal Component Analysis)
• Factor Analysis
• MDS (Multi-Dimensional Scaling)
• Hierarchical Clustering
• K-means Clustering, K-medios
• SOM (Self-Organizing Map)
• MBA (Market Basket Analysis)
• Sequence Analysis
• Collaborative Filtering

DEEP LEARNING

TENSORF OVERVIEWS

머신러닝 라이브러리 (1)



Deep Learning Framework Power Scores(출처: Towards Data Science)출처: https://engineer-daddy.co.kr/entry/딥러닝-프레임워크-종류-특징-선택가이드

개발환경

- Python 3.8
- pip install torch torchvision

텐서플로의 특징

1) 기본적으로 텐서를 활용한 그래프 수치 연산을 하는 도구

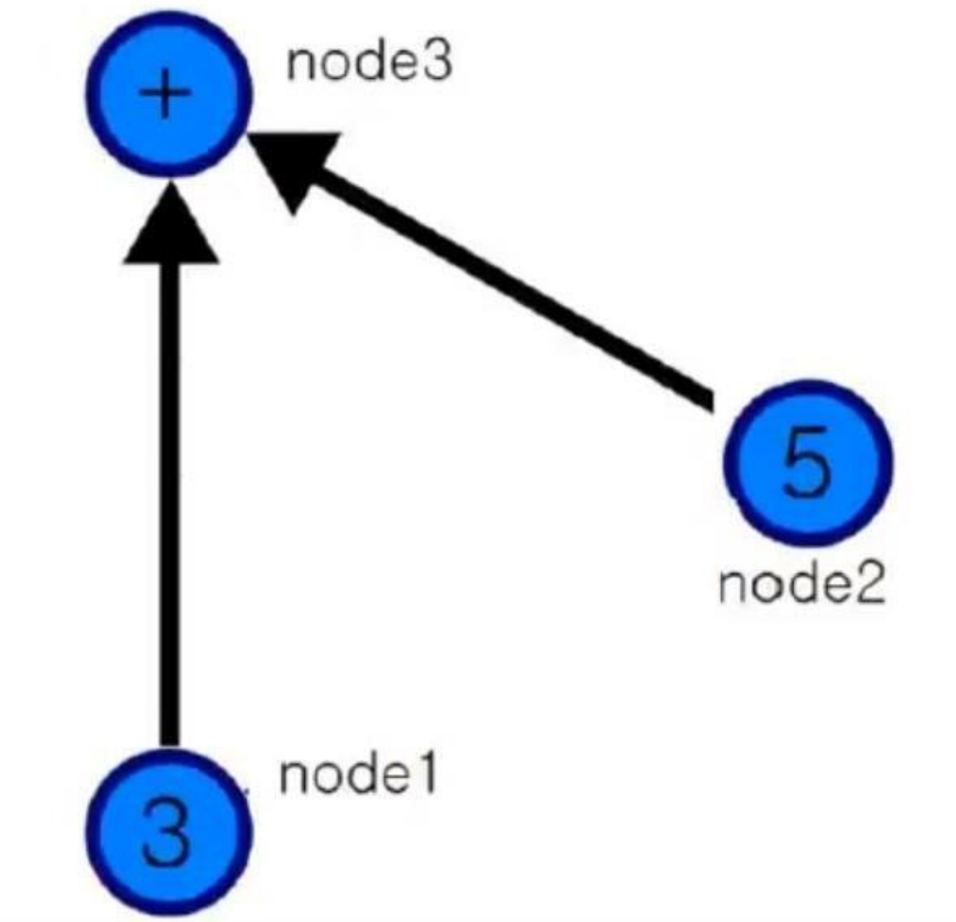
2) 수학적인 의미에서의 그래프 : 노드와 엣지로 구성된 기하 모형

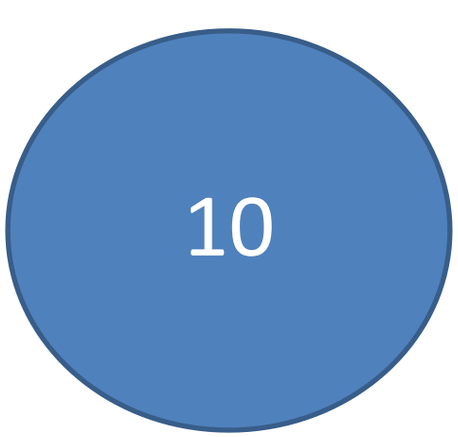
3) 노드(node) : 연산 및 데이터를 정의하는 것

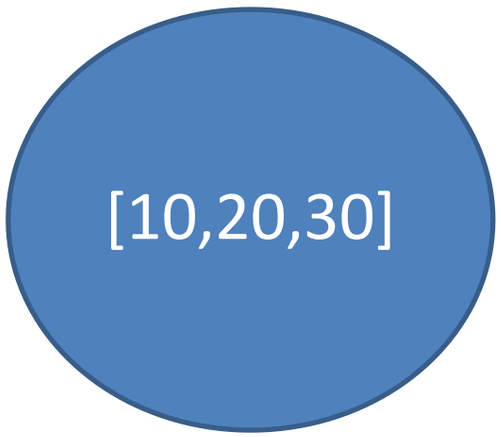
4) 엣지(edge) : 노드들을 연결하는 것 (데이터의 흐름)

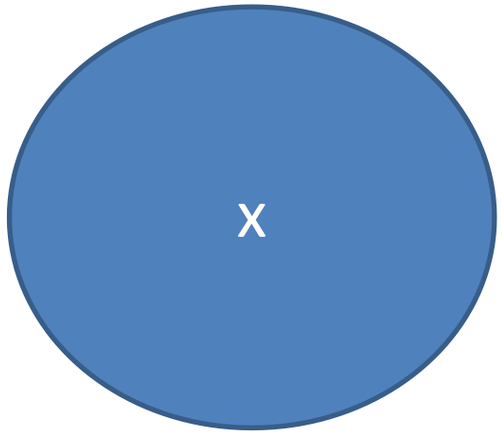
5) 텐서(Tensor) : 다차원 데이터 배열

6) 텐서플로 - 텐서가 노드에서 연산되고 엣지를 통해 돌아다닌다




10


[10,20,30]


x

텐서플로우 속성 : RANK

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Rank	Math entity	Python example
0	Scalar (magnitude only)	`s = 483`
1	Vector (magnitude and direction)	`v = [1.1, 2.2, 3.3]`
2	Matrix (table of numbers)	`m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`
3	3-Tensor (cube of numbers)	`t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]`
n	n-Tensor (you get the idea)	`....`

텐서플로우 속성 : SHAPE

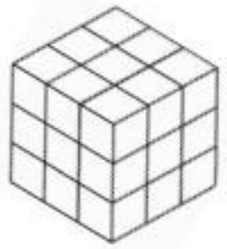
```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

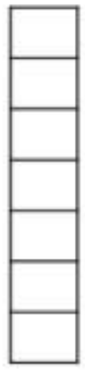
Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

텐서

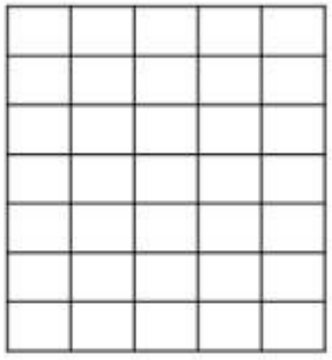
- 텐서는 기본 자료구조이며, 그래프에서 edge를 따라 흐르는 값을 의미함
- 텐서는 다차원 배열과 리스트로 구성됨
- 텐서는 rank, shape, type 세 가지의 매개변수를 가짐

- Tensor = rank + shape + type
  - 차원 (rank)
  - 행, 열 (shape)
  - 타입 (type)

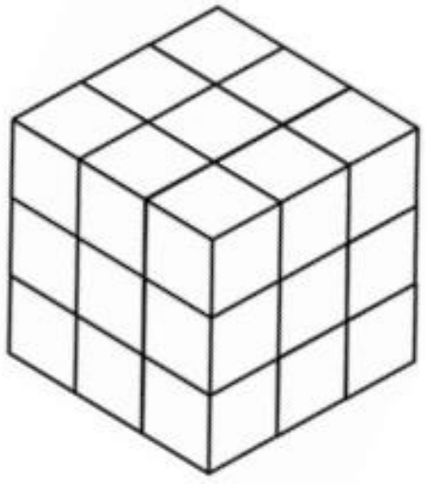




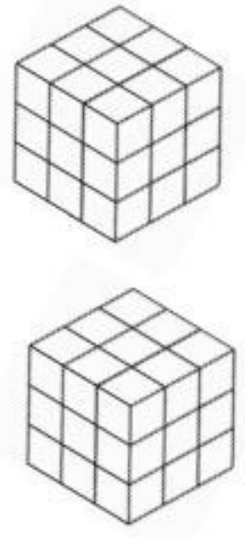
Rank = 1



Rank = 2



Rank = 3



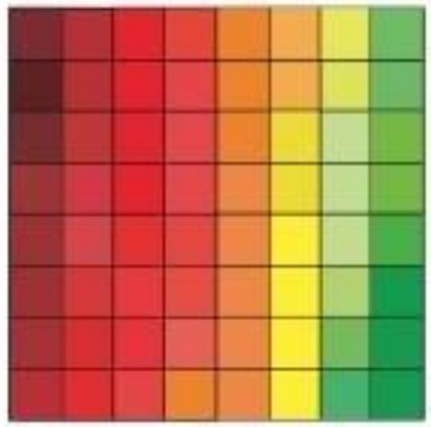
Rank = 4

tensor = multidimensional array

vector

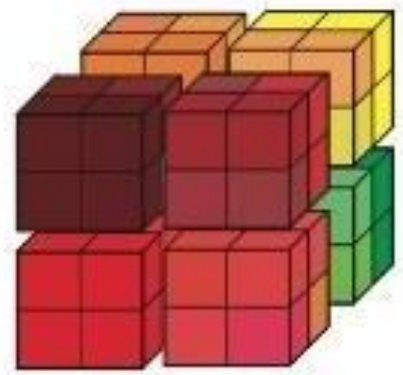
$\mathbf{v} \in \mathbb{R}^{64}$

matrix



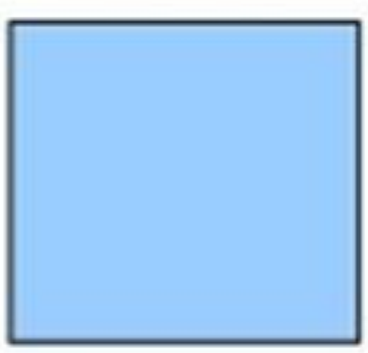
$X \in \mathbb{R}^{8x8}$

tensor

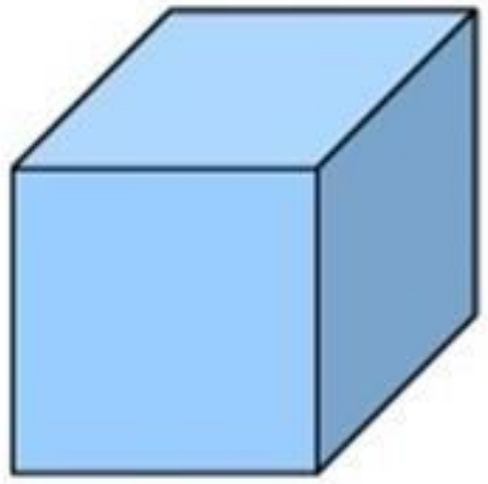


$\mathcal{X} \in \mathbb{R}^{4x4x4}$

1d-tensor

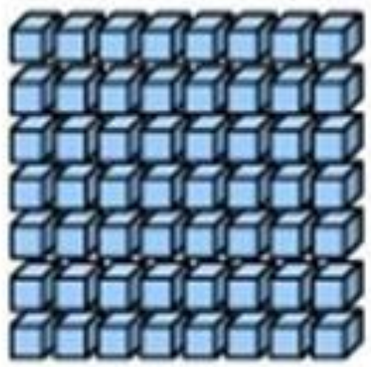


2d-tensor

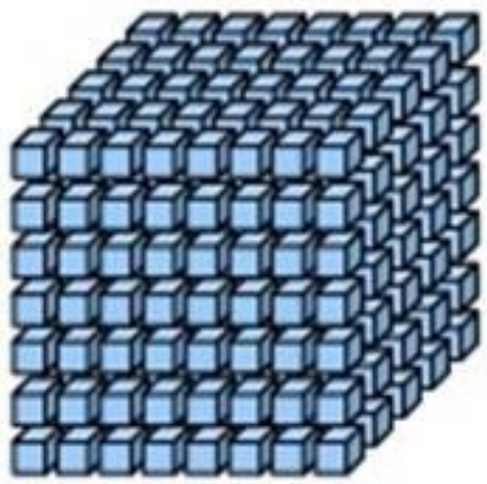


3d-tensor

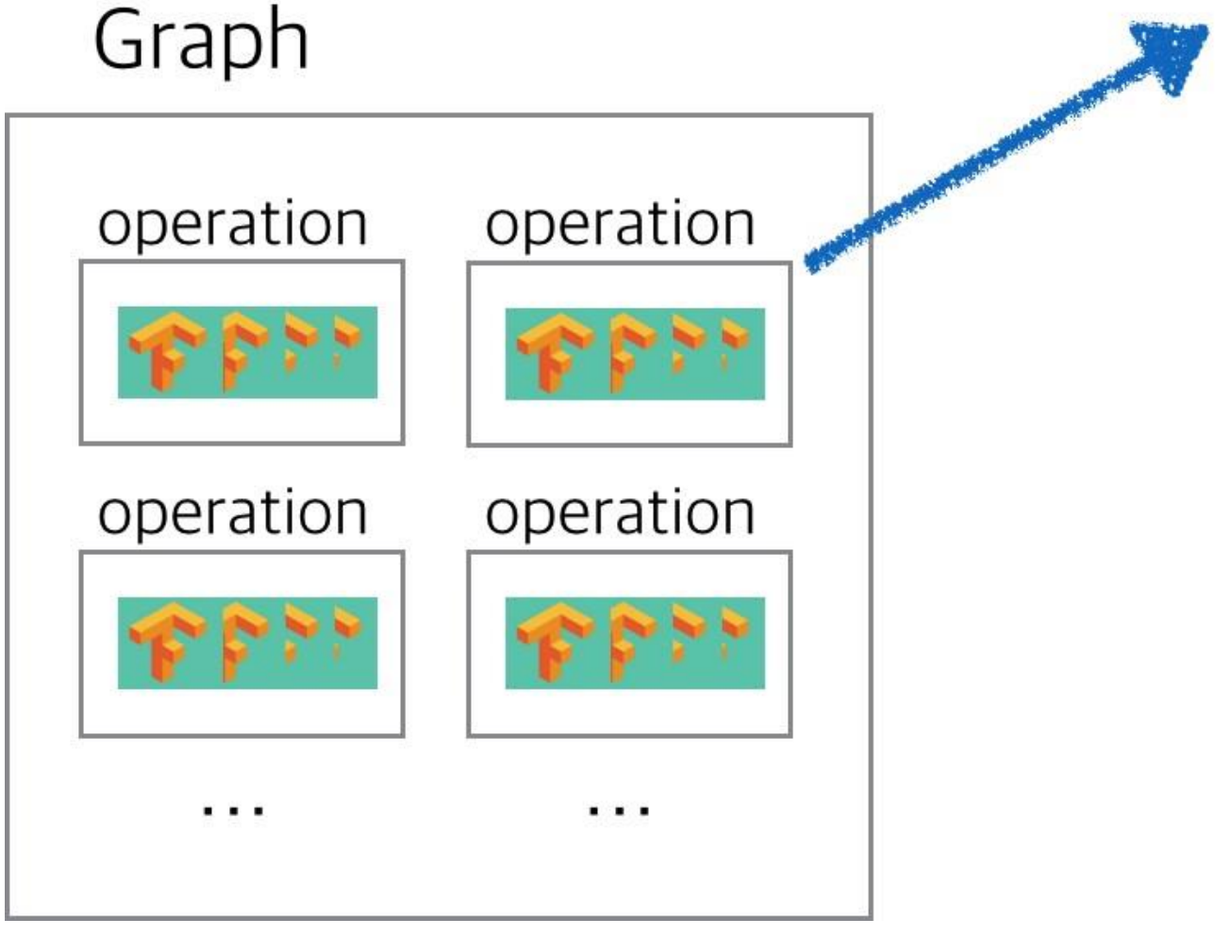
4d-tensor

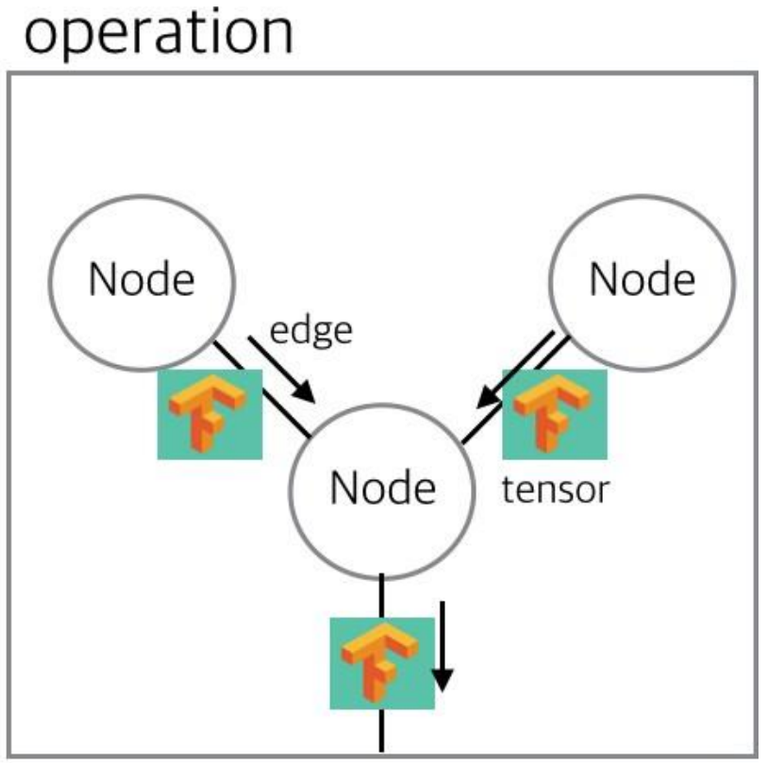


5d-tensor

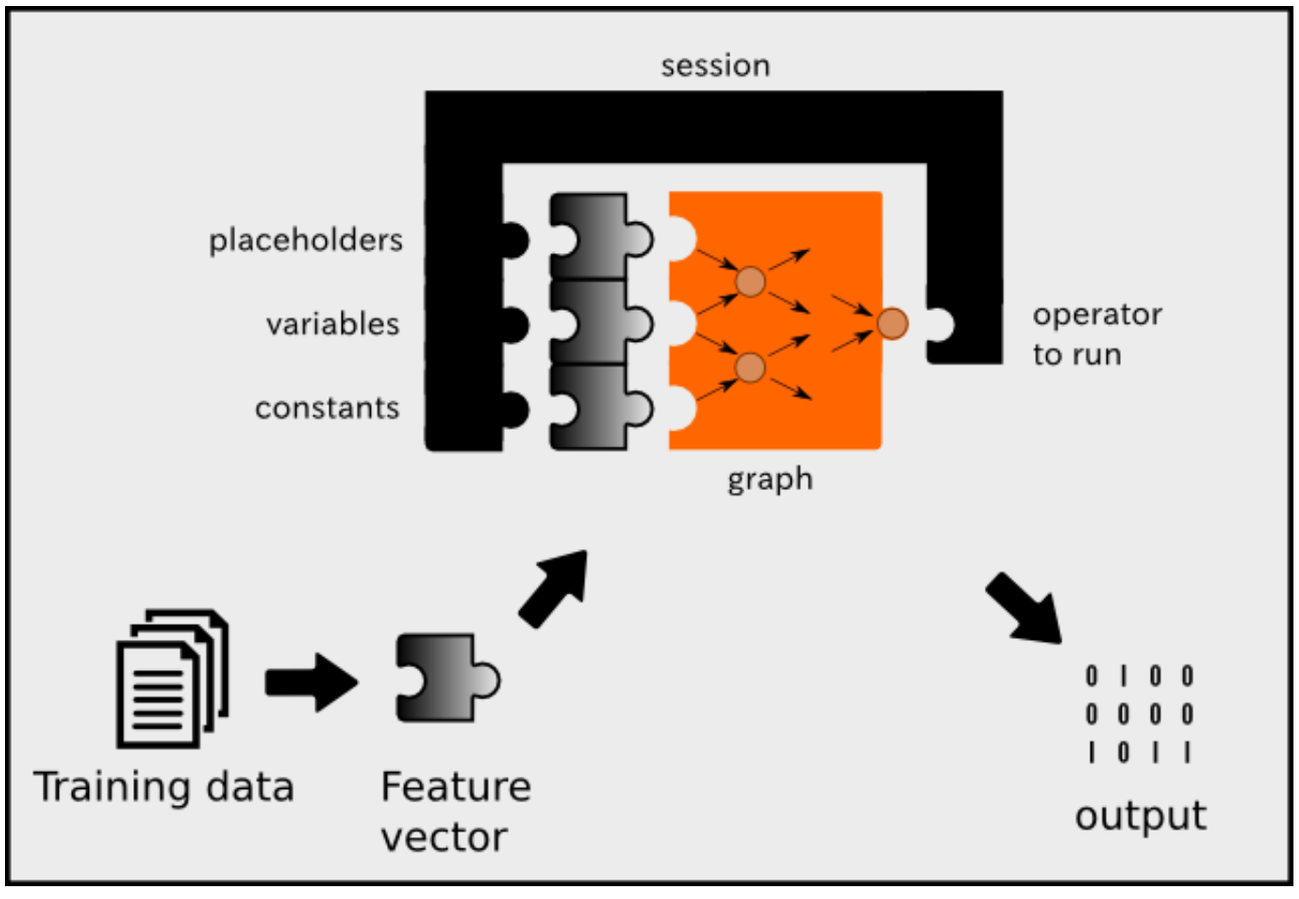


6d-tensor


Graph
operation
operation
operation
operation
...
...


operation
Node
edge
Node
Node
tensor

rank	type	example
0	scalar	[1]
1	vector	[1,1]
2	matrix	[[1,1], [1,1]]
3	3 tensor	[[[1,1], [1,1]], [[1,1], [1,1]]]
n	N tensor	


session
placeholders
variables
constants
operator
to run
graph
Training data
Feature
vector
output

텐서플로우 속성 : DTYPE

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.

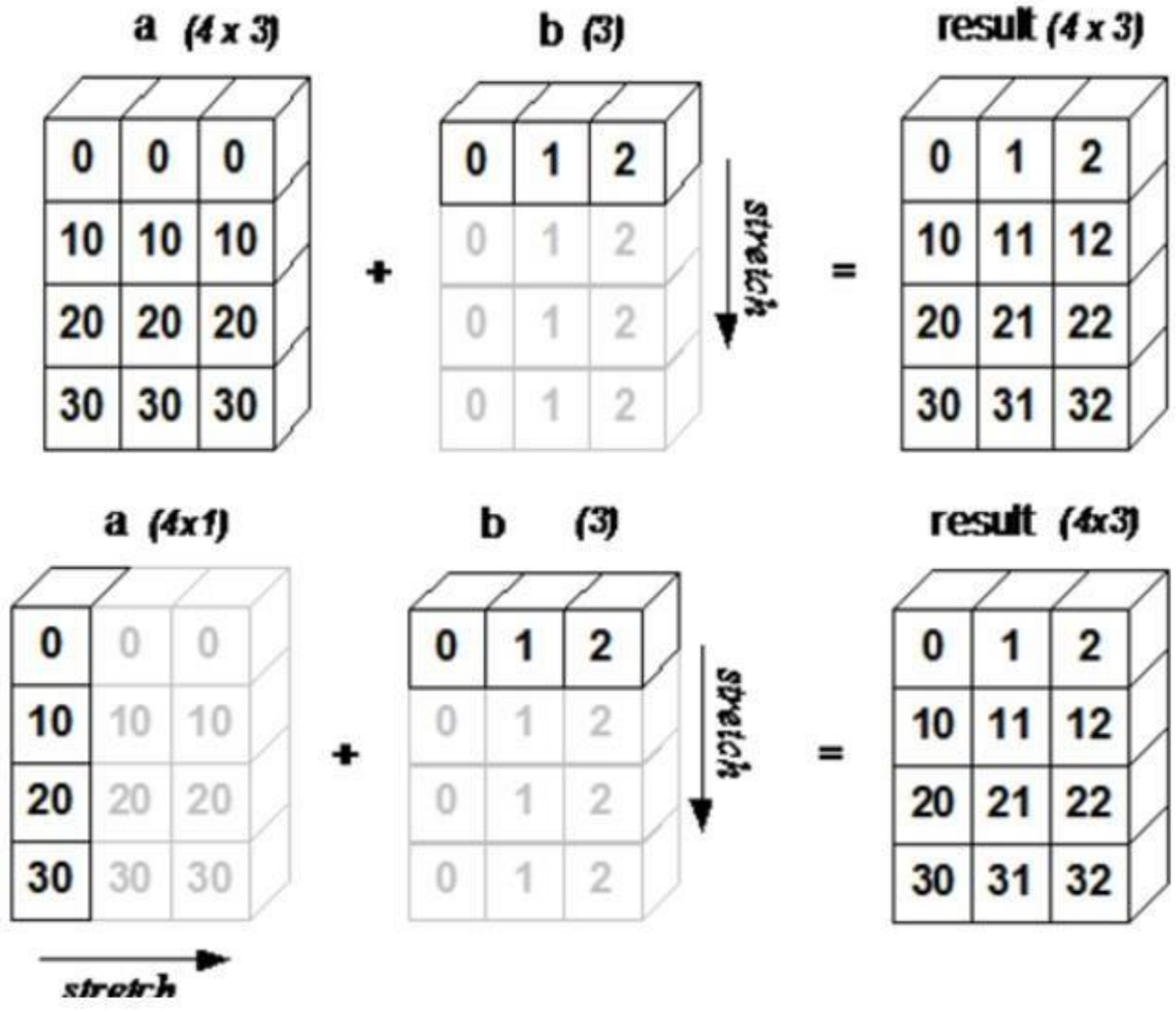
...

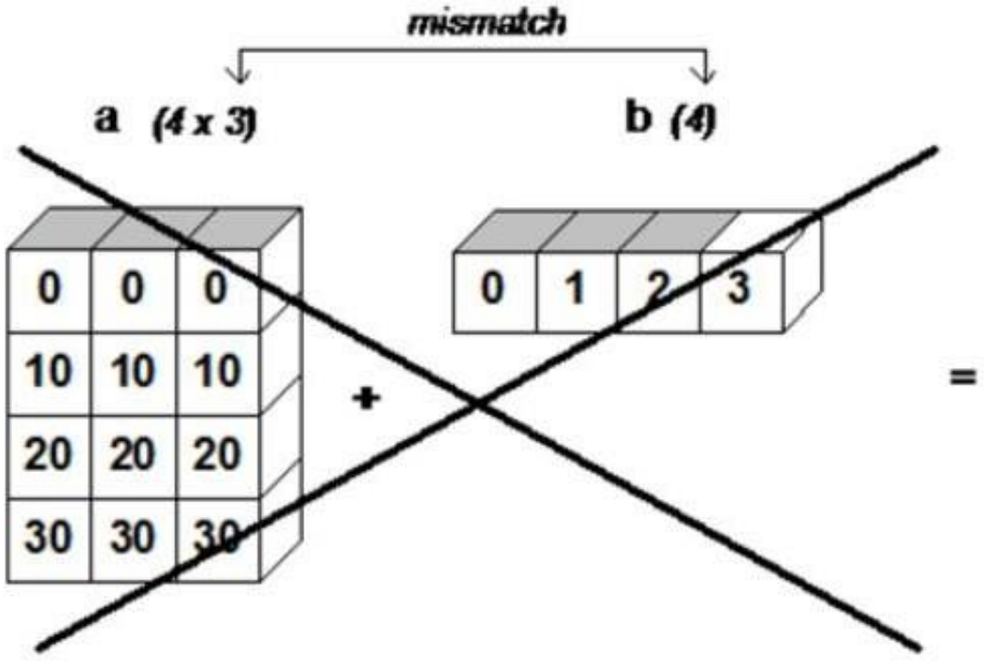
텐서플로우 속성 : operation

- **Operation**
  - operation은 이름을 가지며 추상적인 연산을 의미함
  - Graph-construction time에 operation이 생성(노드)
  - 딥 러닝에 사용되는 대표적인 operation 및 class로는 constant, Variable, placeholder, Matrix operation, neural-net building block 등이 있음

Category	**Examples**
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural-net building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

Element wise operation broad casting: (요소별 연산 브로드 캐스팅)





행렬곱셈:

행렬 곱셈 예

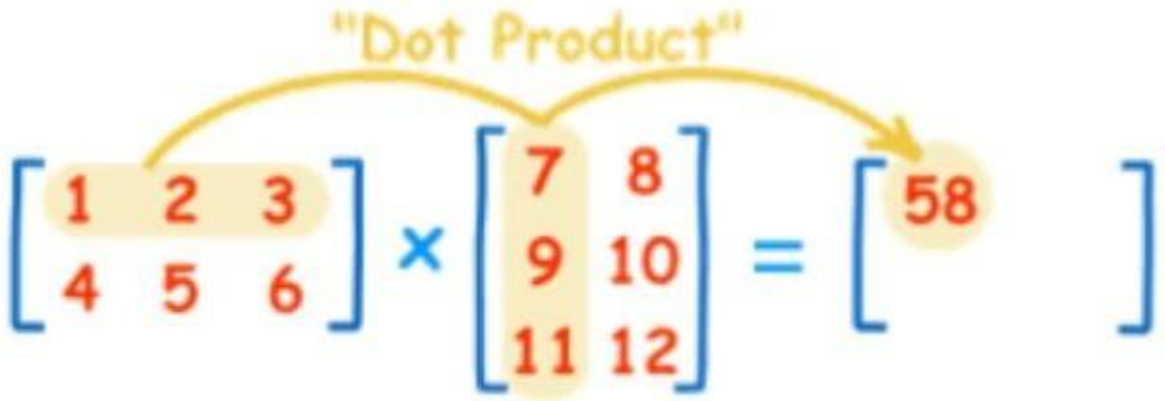
$$A = \begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix} . B = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} \text{ 일 때}$$

$$AB = \begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 3 \cdot 2 & 1 \cdot 2 + 3 \cdot 5 \\ 5 \cdot 1 + 7 \cdot 2 & 5 \cdot 2 + 7 \cdot 5 \end{bmatrix} = \begin{bmatrix} 7 & 17 \\ 19 & 45 \end{bmatrix}$$

$$BA = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 2 \cdot 5 & 1 \cdot 3 + 2 \cdot 7 \\ 2 \cdot 1 + 5 \cdot 5 & 2 \cdot 3 + 5 \cdot 7 \end{bmatrix} = \begin{bmatrix} 11 & 17 \\ 27 & 41 \end{bmatrix}$$

MATRIX AND TRANSPOSE

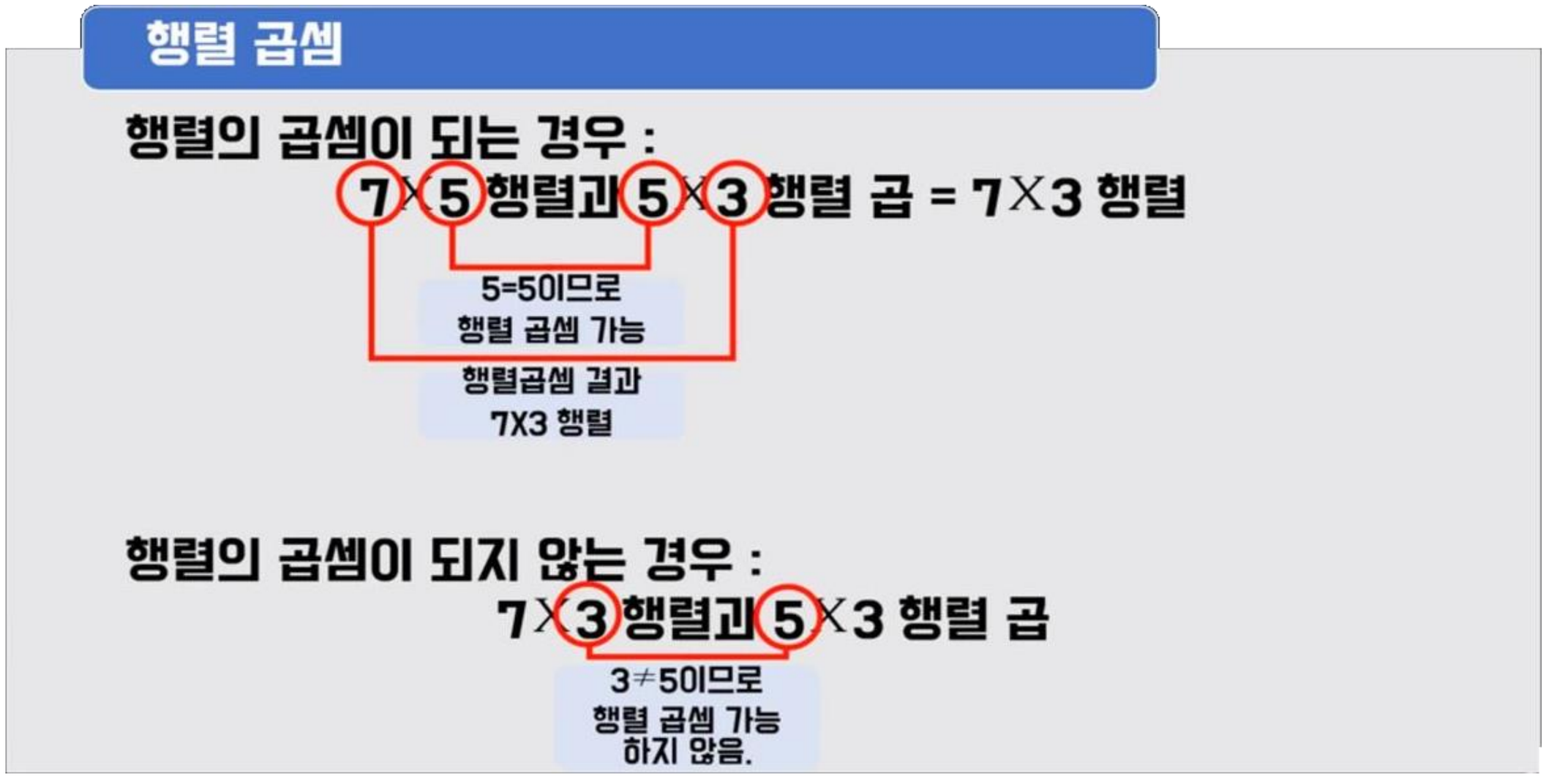
Matrix multiplication



Transpose

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

행렬곱셈:


행렬 곱셈
행렬의 곱셈이 되는 경우 :
7X5 행렬과 5X3 행렬 곱 = 7X3 행렬
5=5이므로
행렬 곱셈 가능
행렬곱셈 결과
7X3 행렬
행렬의 곱셈이 되지 않는 경우 :
7X3 행렬과 5X3 행렬 곱
3≠5이므로
행렬 곱셈 가능
하지 않음.

DEEP LEARNING

LINEAR REGRESSION

Cost function

모델이 추측한 답이 정답과 얼마나 차이나는가?

목표 : 모델이 추측한 답과 정답이 비슷해지게 하는것!
(Cost를 최소화한다고 표현)

가설

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

가설을 수정해 나가면서 가장 합리적인 식을 찾아낸다

Input

```
def function(C):
```

Algorithm → `F = C * 1.8 + 32`

```
return F
```

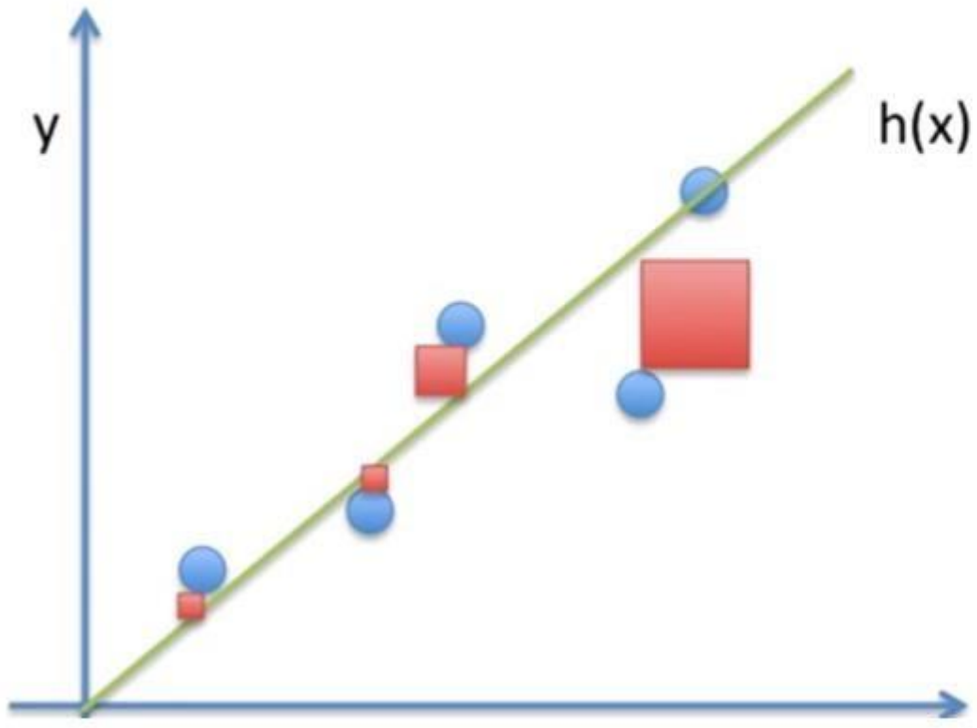
Output

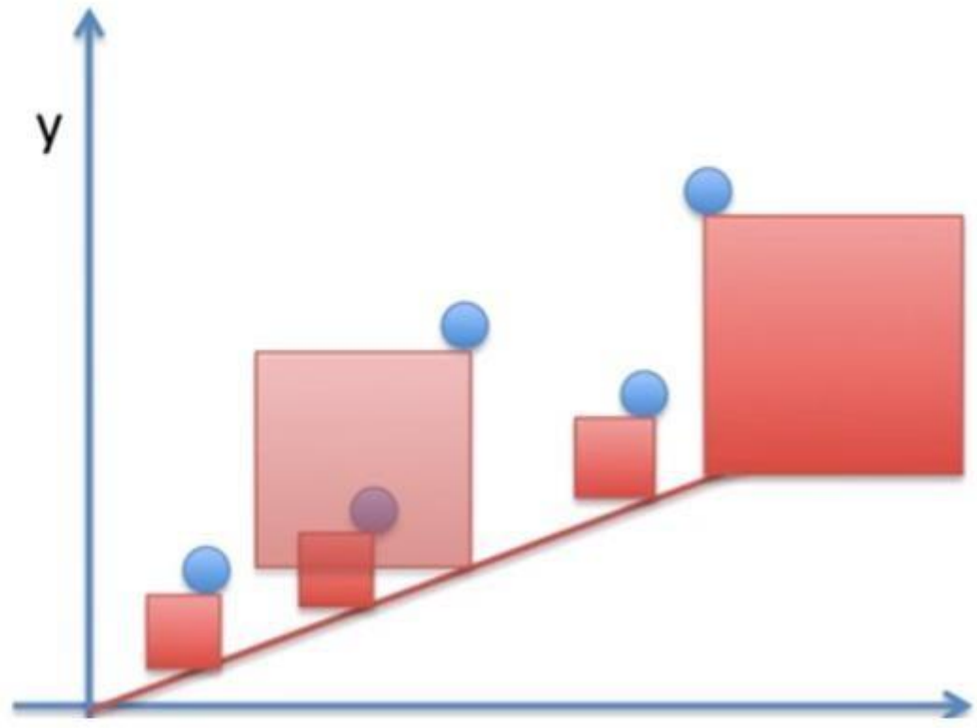
Least mean Square Error(LMS)

Error = h(x) - y

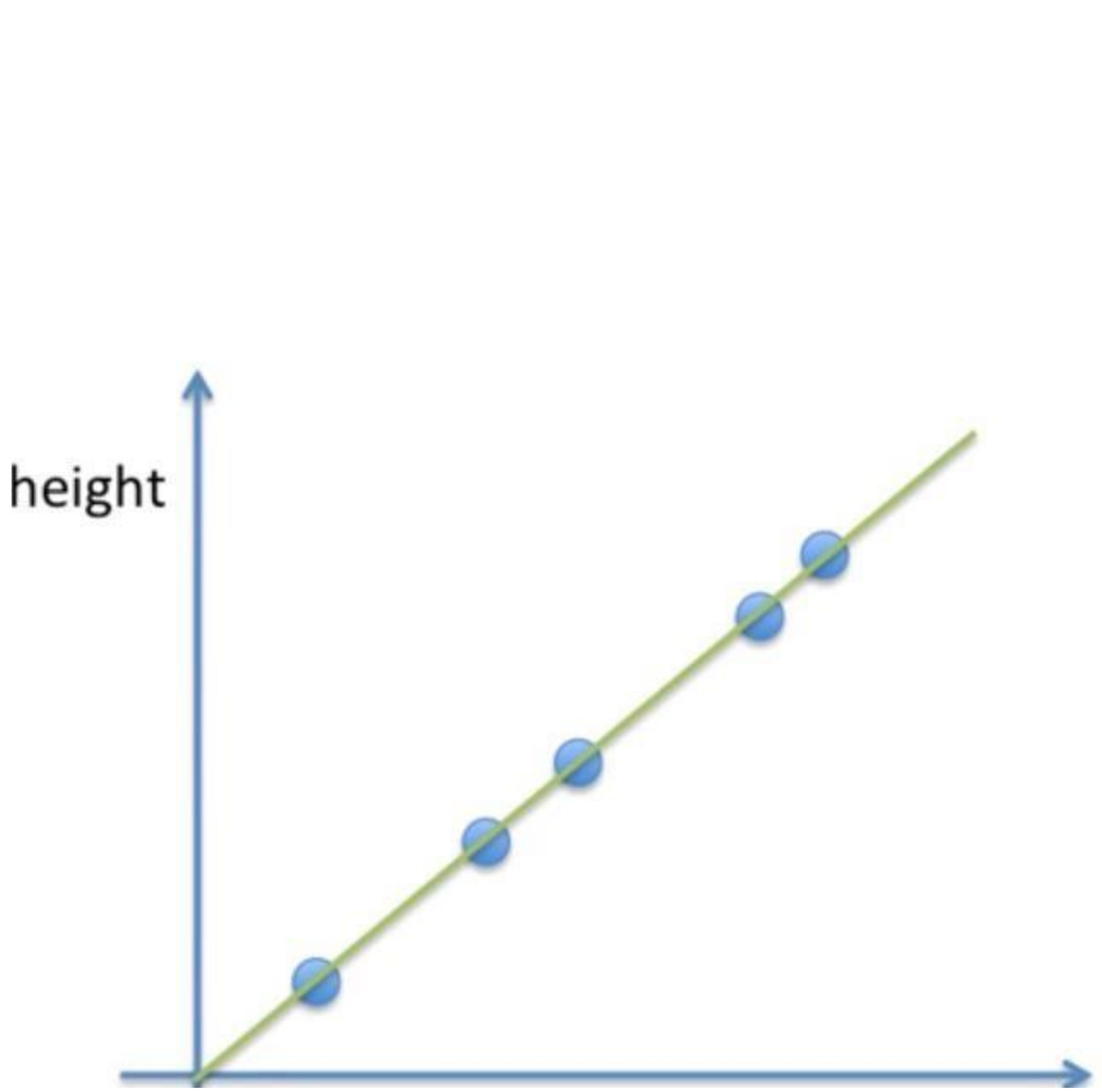
Square Error = $(h(x) - y)^2$

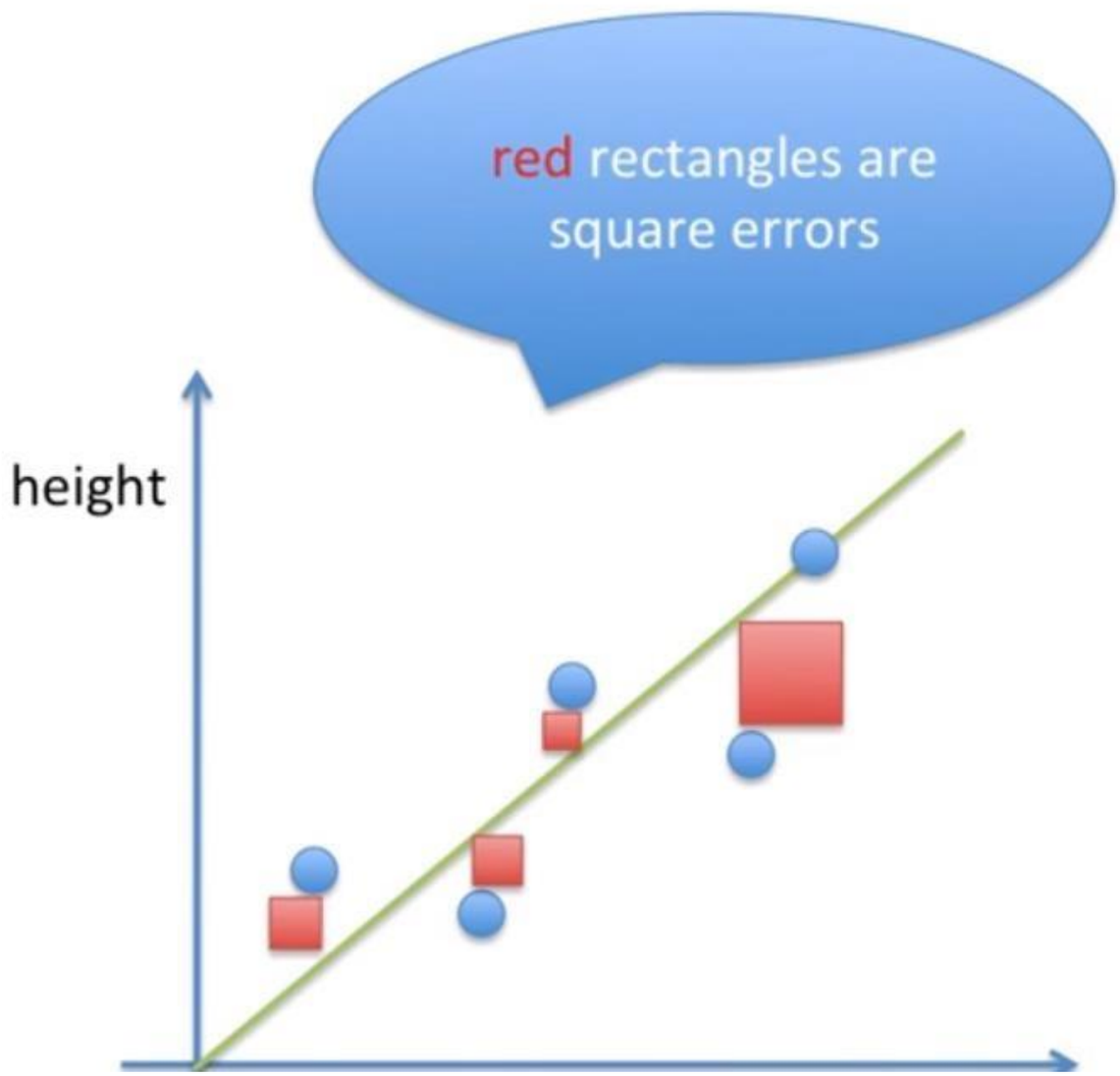
Mean Square Error = $1/n \sum (h(x) - y)^2$





Square Error– (difference between prediction and real value)^2

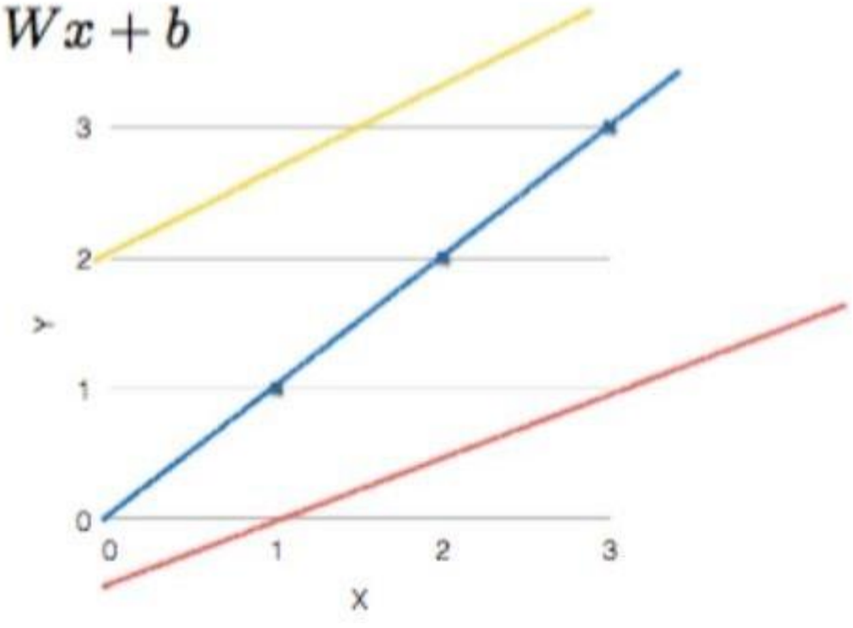




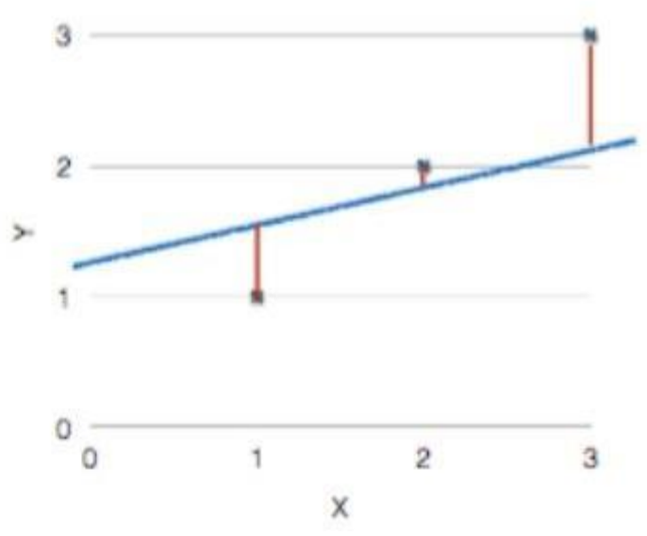
좋은 가설 ?

(Linear) Hypothesis

$H(x) = Wx + b$



Which hypothesis is better?



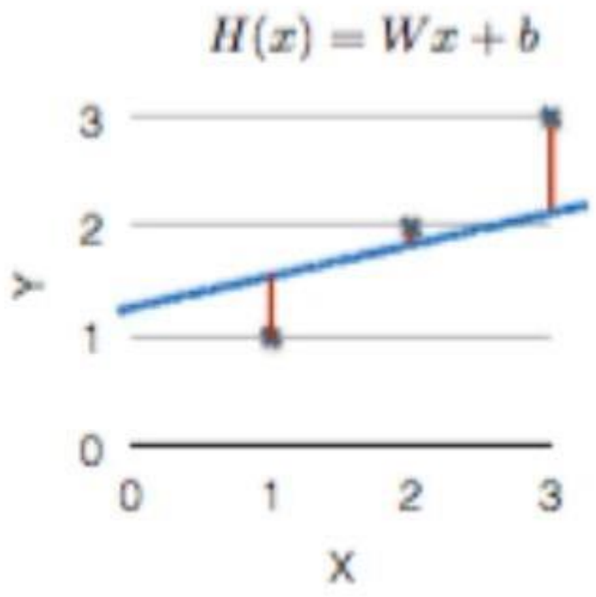
직선으로부터 점까지의 거리 계산

Cost function

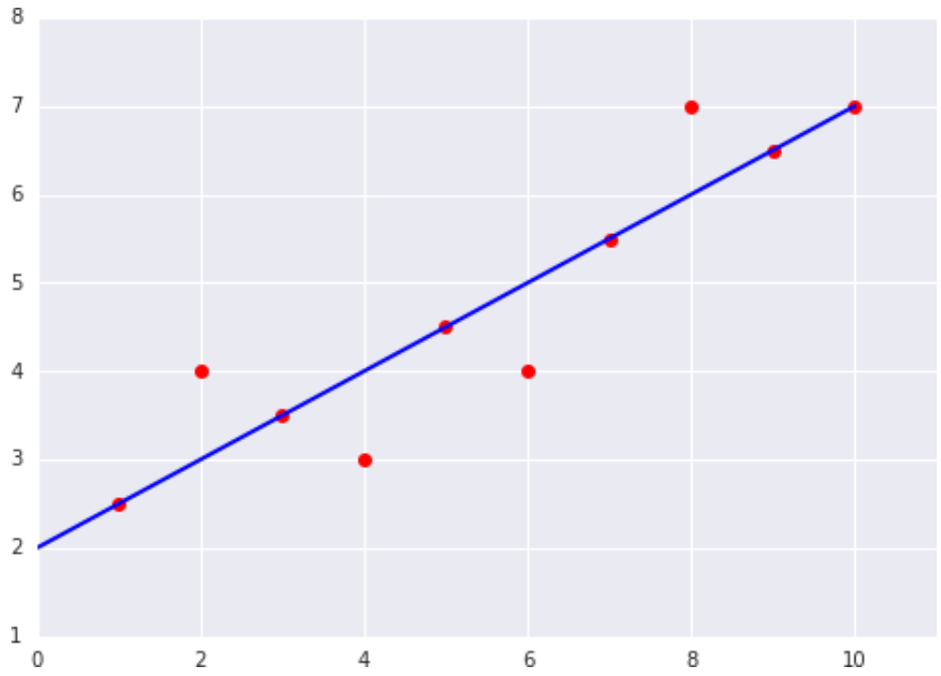
- How fit the line to our (training) data

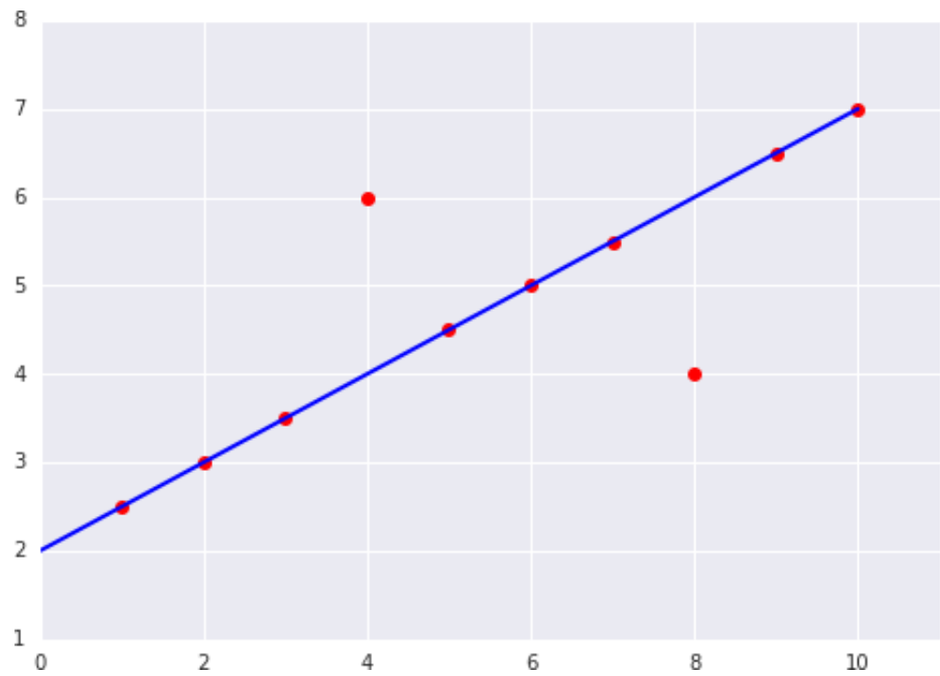
$$\frac{(H(x^{(1)})-y^{(1)})^2+(H(x^{(2)})-y^{(2)})^2+(H(x^{(3)})-y^{(3)})^2}{3}$$

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



Least mean Square Error(LMS)





선에 위치한 8개의 점에 발생하는 총 손실은 0입니다. 그러나, 선에서 벗어난 점은 2개밖에 안 되지만, 2개의 점 모두 선에서 벗어난 정도가 왼쪽 그림의 이상점에 비해 *2배* 더 큽니다. 제곱 손실값의 경우 이러한 차이가 증폭되므로 오프셋 2의 경우 오프셋 1보다 4배 더 큰 손실이 발생합니다.

$$MSE = \frac{0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2}{10} = 0.8$$

COST 수식

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

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

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

DEEP LEARNING

GRADIENT DESCENT ALGORITHM

HOW TO HIDE BIAS?

Hypothesis and Cost

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

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

Simplified hypothesis

$$H(x) = Wx$$

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

COST FUNCTION GRAPH

What *cost(W)* looks like?

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

x	Y
1	1
2	2
3	3

- W=1, cost(W)=0

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

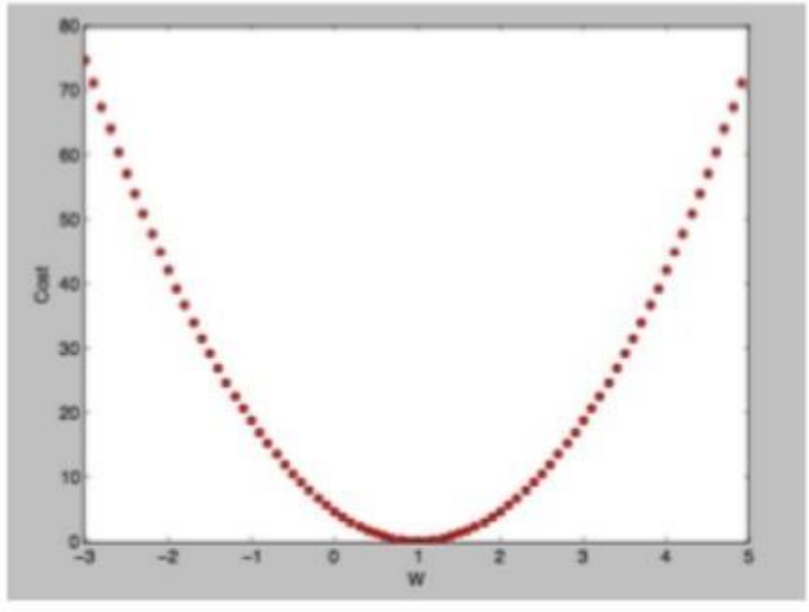
- W=0, cost(W)=4.67

  $\frac{1}{3}((0*1-1)^2 + (0*2-2)^2 + (0*3-3)^2)$

- W=2, cost(W)=?

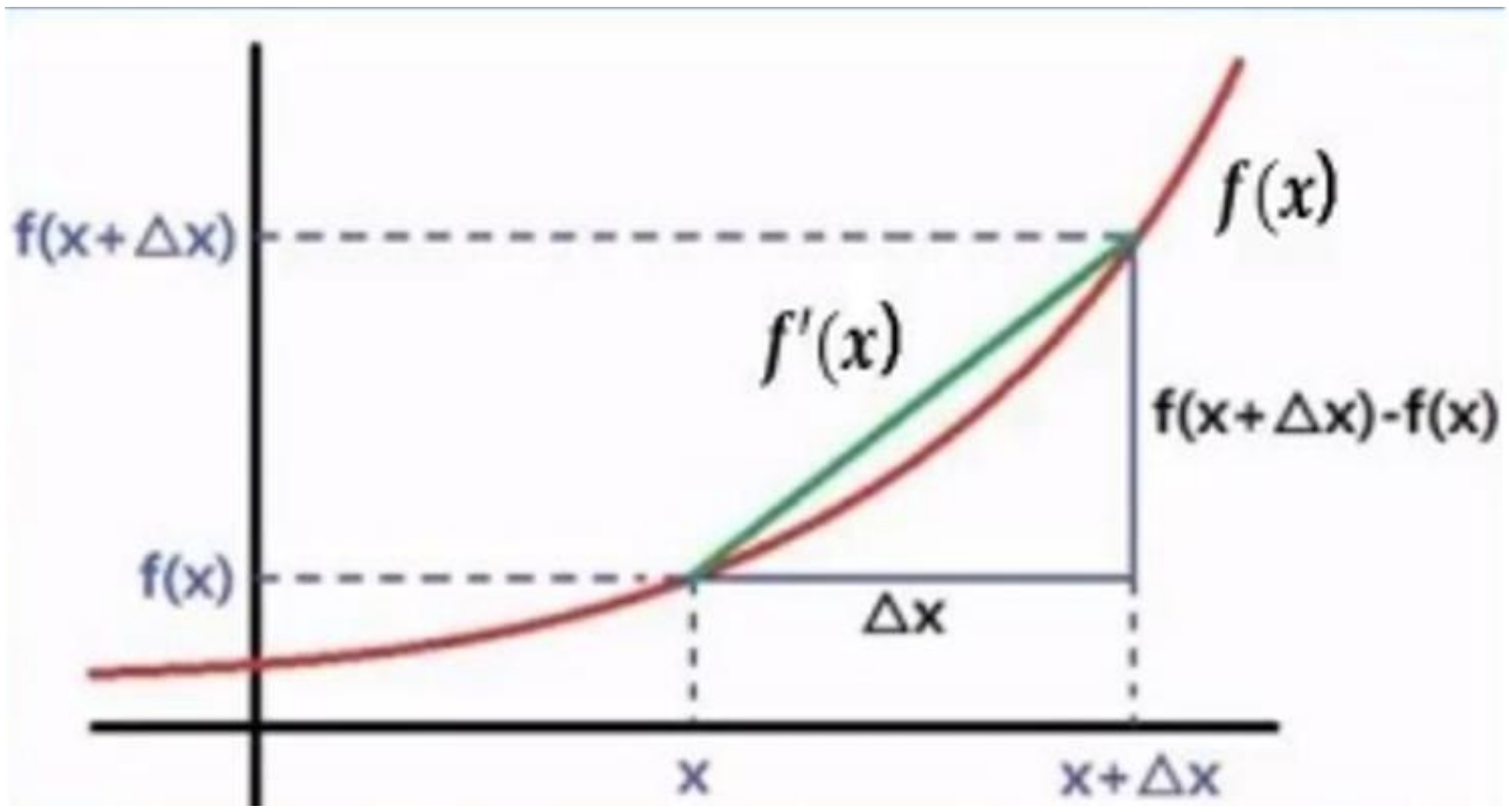
How to minimize cost?

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



미분

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$



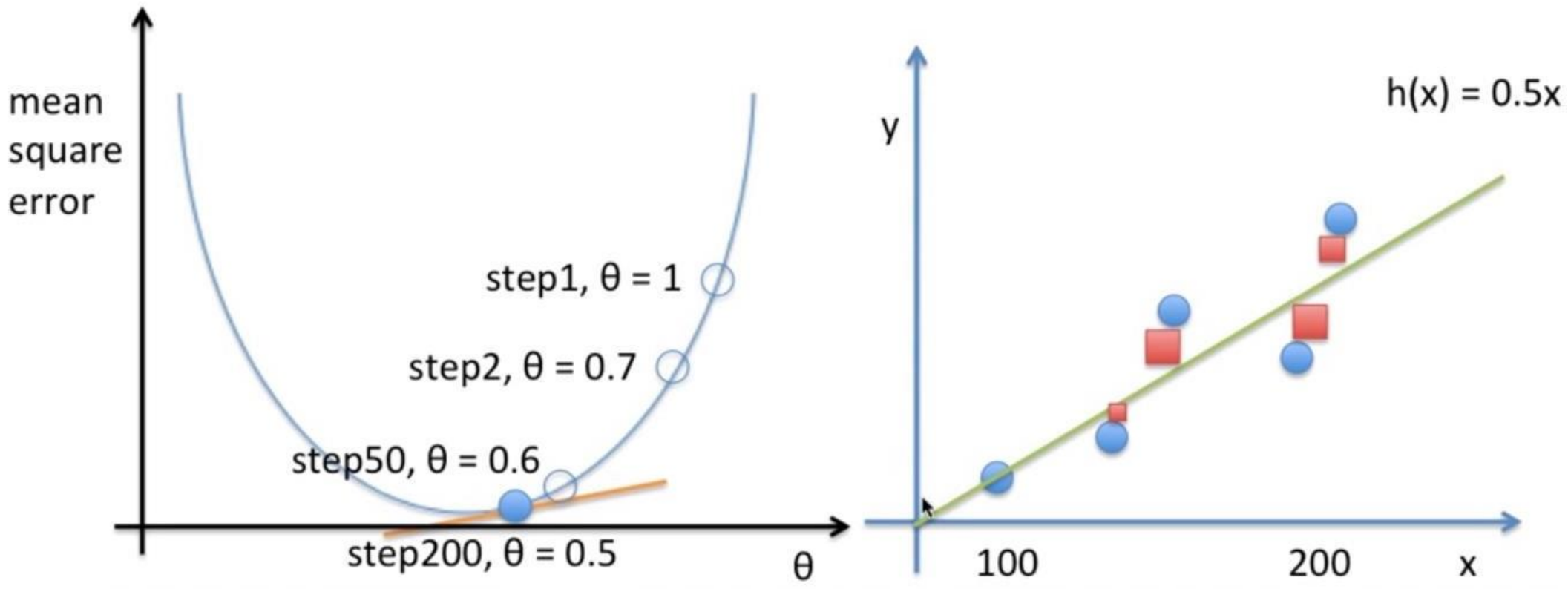
합성함수의 미분

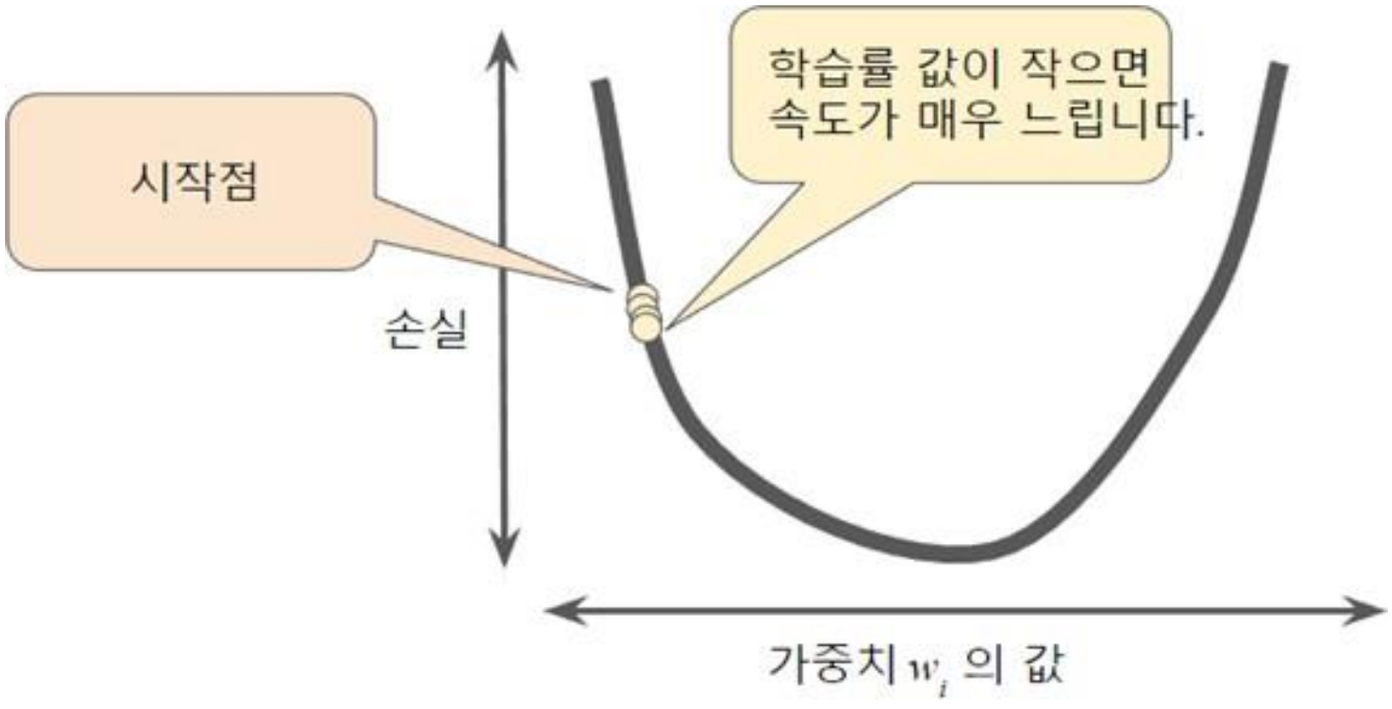
- f(g(x))
- f(t), t=g(x)

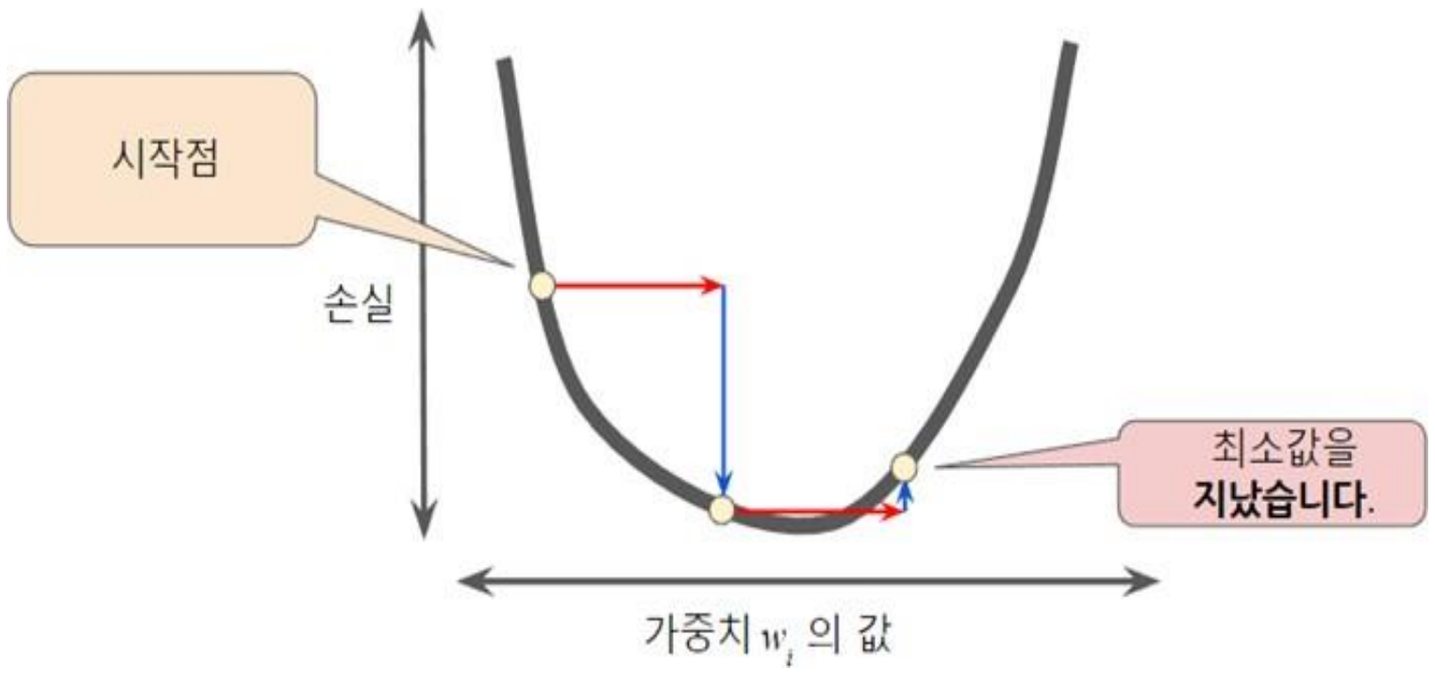
- $f(x) = (x+2)^2$
- Let $t = x+2$
- $\frac{\partial f}{\partial t} = \frac{\partial}{\partial t} t^2 = 2t$
- $\frac{\partial t}{\partial x} = \frac{\partial}{\partial x}(x+2) = 1$
- $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial t}\frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x+2)$

Repeat until converge:

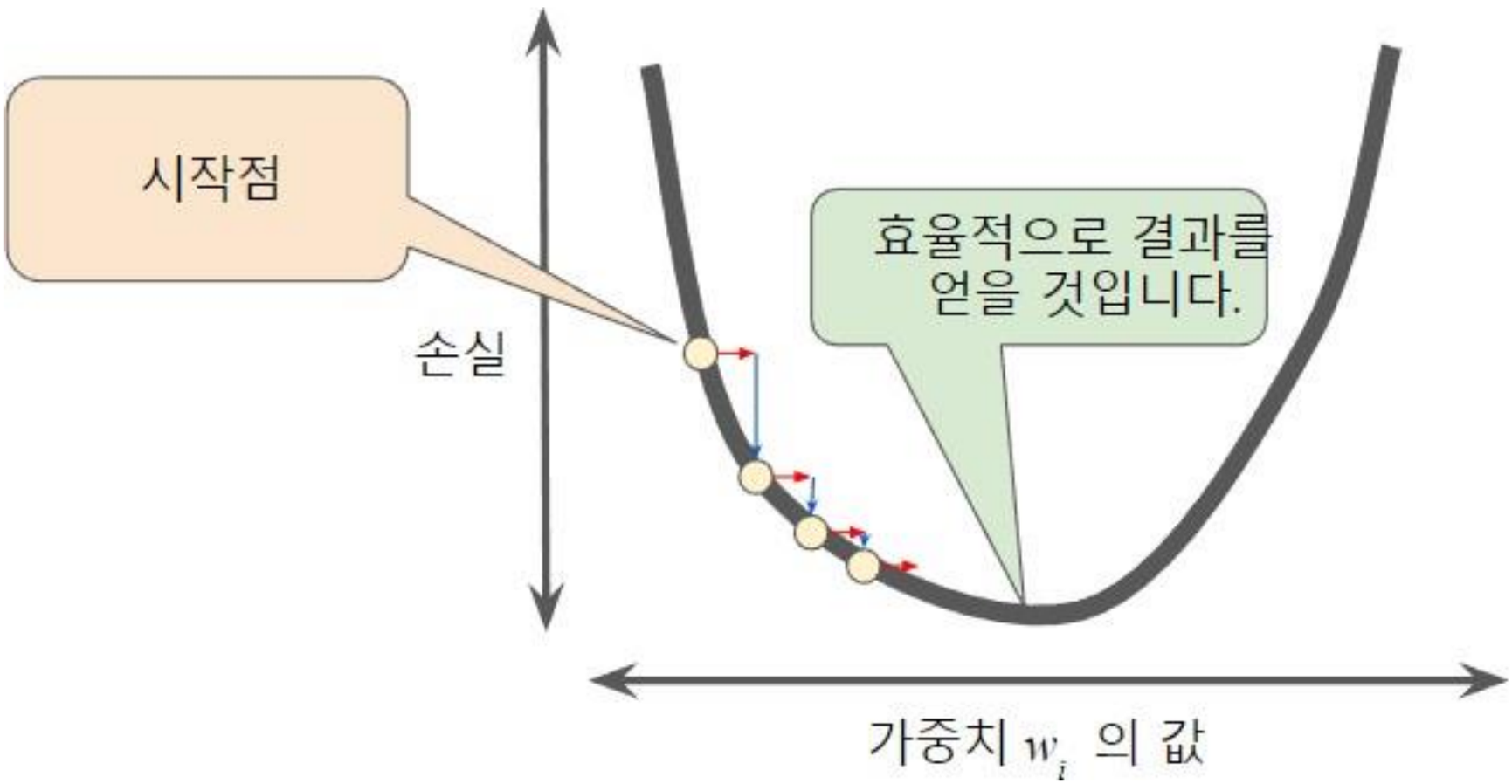
$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} \text{(Mean Square Error)}$$




학습률 값이 작으면
속도가 매우 느립니다.
시작점
손실
가중치 w_i 의 값


시작점
손실
최소값을
지났습니다.
가중치 w_i 의 값

실무에서는 모델 학습의 성공을 위해 최적 또는 최적에 근접한 학습률을 반드시 구할 필요는 없다.
경사하강법이 효과적으로 수렴할 정도로 크지만 발산할 정도로 크지는 않은 적당한 학습률을 구하는 것이 목표.



FORMAL DEFINITION

Formal definition

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

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

Formal definition

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

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

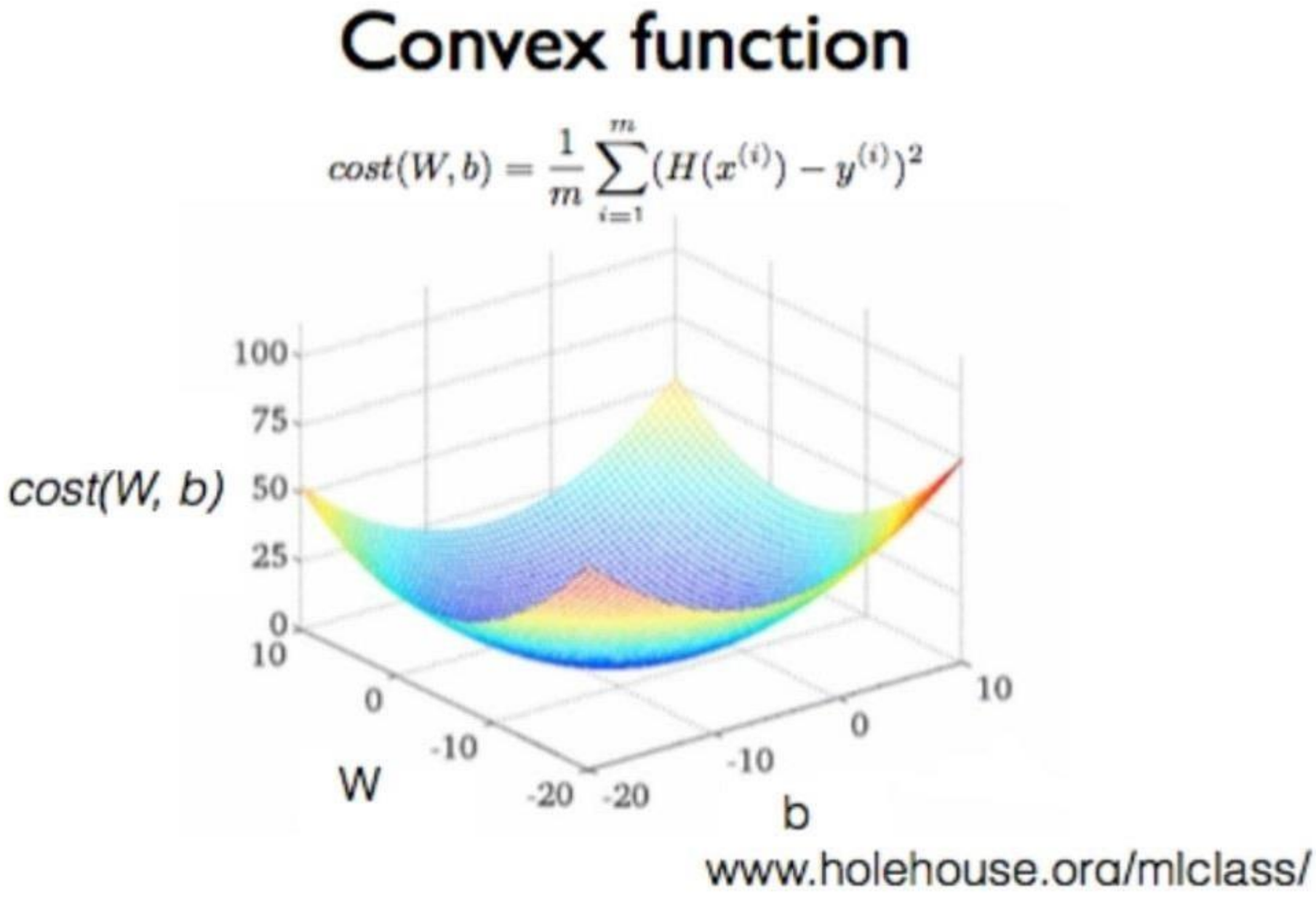
Formal definition

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

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

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

CONVEX FUNCTION



DEEP LEARNING

MULTI-VARIABLE LIN EAR REGRESSION

MULTI FEATURES

one-variable
one-feature

x (hours)	y (score)
10	90
9	80
3	50
2	60
11	40

multi-variable/feature

x1 (hours)	x2 (attendance)	y (score)
10	5	90
9	5	80
3	2	50
2	4	60
11	1	40

HYPOTHESIS AND COST FUNCTION

Hypothesis

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

$$H(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

Cost function

$$H(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

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

MULTI-FEATURES

Predicting exam score:

regression using three inputs (x1, x2, x3)

multi-variable/feature

x_1 (quiz 1)	x_2 (quiz 2)	x_3 (midterm 1)	Y (final)
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Test Scores for General Psychology

HYPOTHESIS USING MATRIX (1)

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (2)

$$H(x_1, x_2, x_3) = x_1w_1 + x_2w_2 + x_3w_3$$

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Test Scores for General Psychology

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1w_1 + x_2w_2 + x_3w_3 \end{pmatrix}$$

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (3)

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Hypothesis using matrix

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

$$H(X) = XW$$

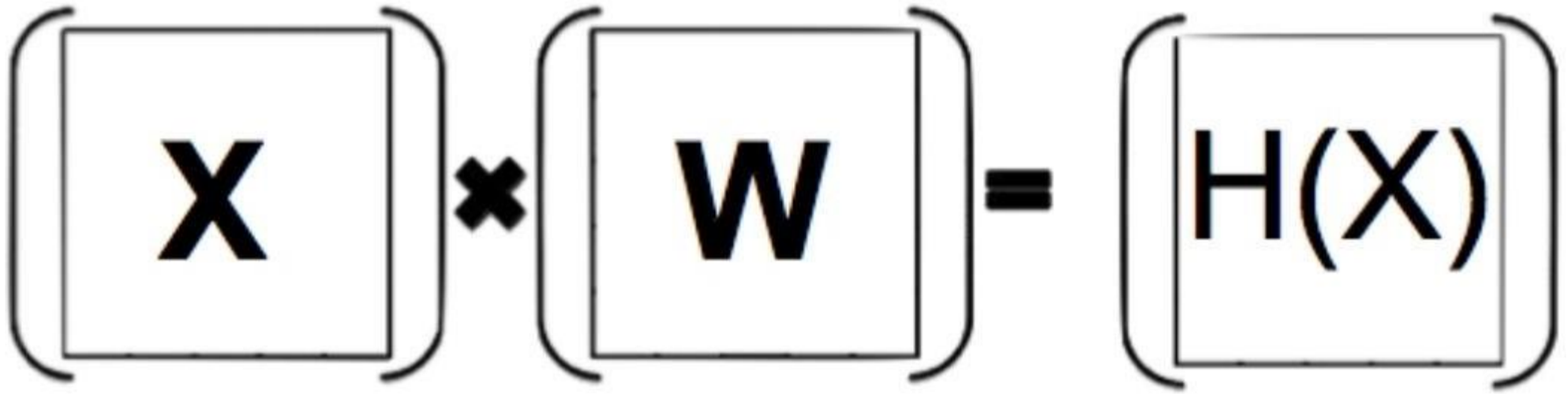
HYPOTHESIS USING MATRIX (4)

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

[5, 3] [3, 1] [5, 1]

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (5)



$$H(X) = XW$$

HYPOTHESIS USING MATRIX (6)

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

`[n, 3]` `[3, 1]` `[n, 1]`

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (7)

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

[n, 3] [?, ?] [n, 2]

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (8)

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

[n, 3] [3, 2] [n, 2]

$$H(X) = XW$$

HYPOTHESIS USING MATRIX (9)

- Lecture (theory):

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

- Implementation (TensorFlow)

$$H(X) = XW$$

DEEP LEARNING

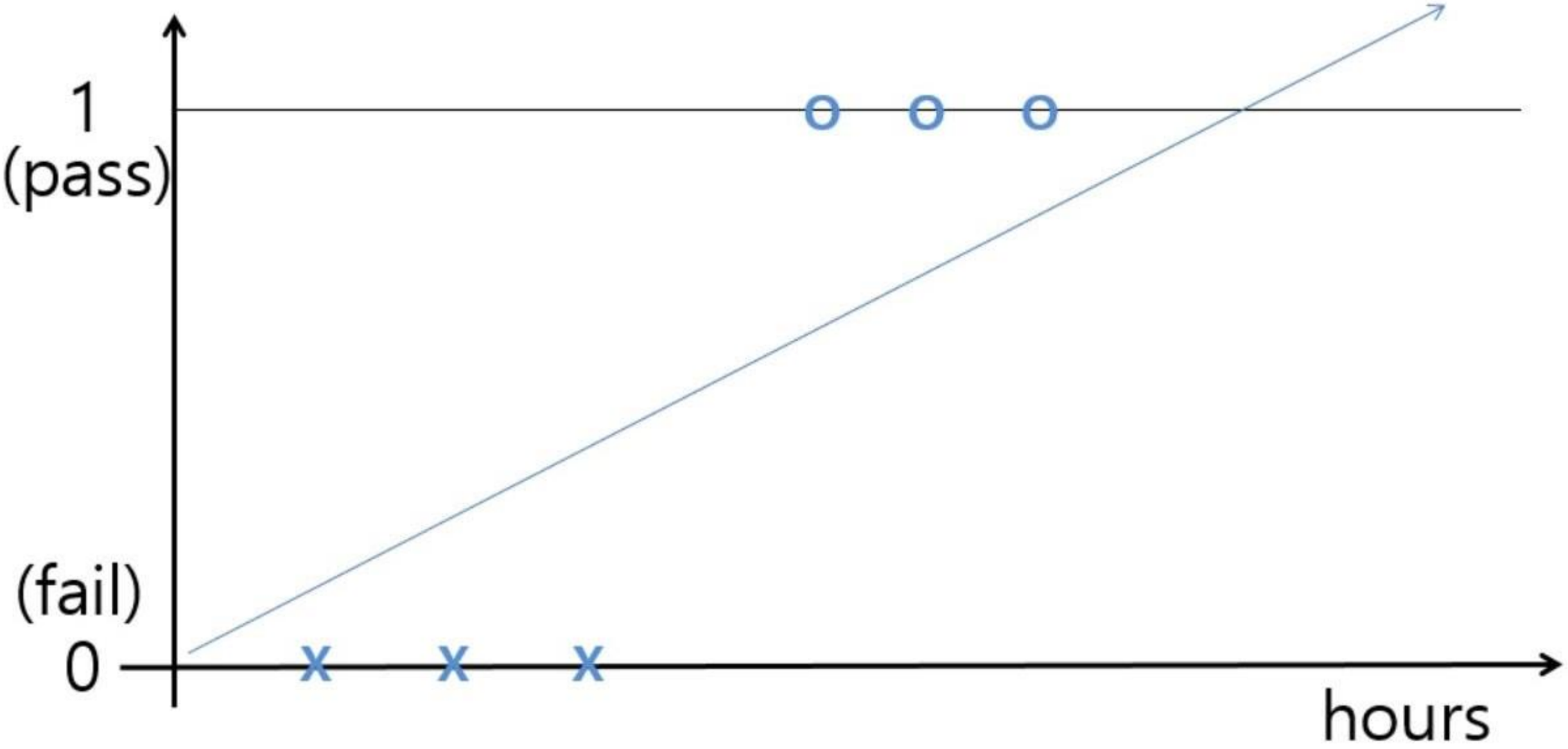
LOGISTIC CLASSIFICATION

CLASSIFICATION AND ENCODING

- Spam Detection: Spam or Ham
- Facebook feed: show or hide
- Credit Card Fraudulent Transaction detection: legitimate/fraud

- Spam Detection: Spam (1) or Ham (0)
- Facebook feed: show(1) or hide(0)
- Credit Card Fraudulent Transaction detection: legitimate(0) or fraud (1)

LINEAR REGRESSION?



공부한 시간	합격여부
1	0
2	0
3	0
4	1
5	1
6	1

PROBLEMS

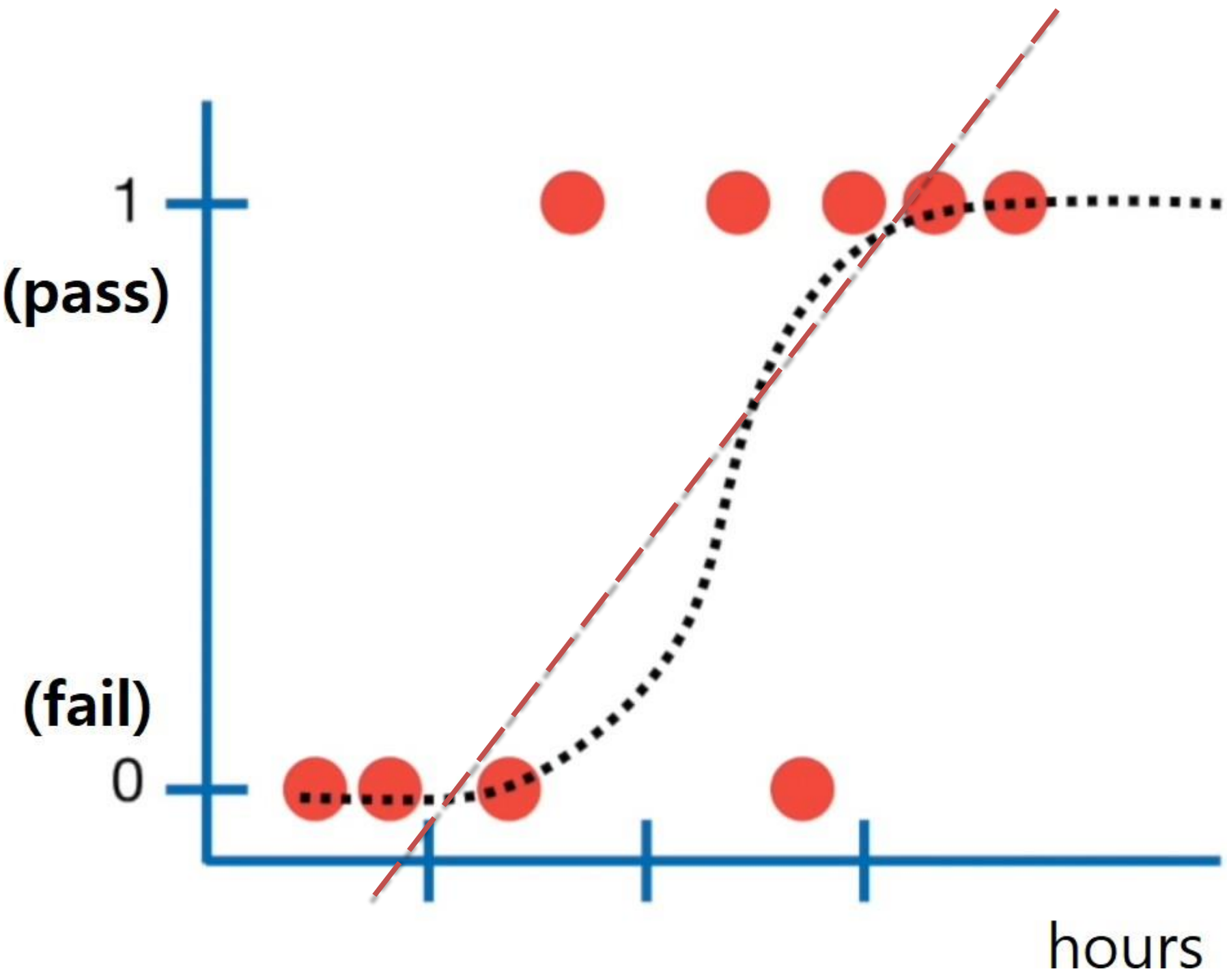
Linear regression

- We know Y is 0 or 1

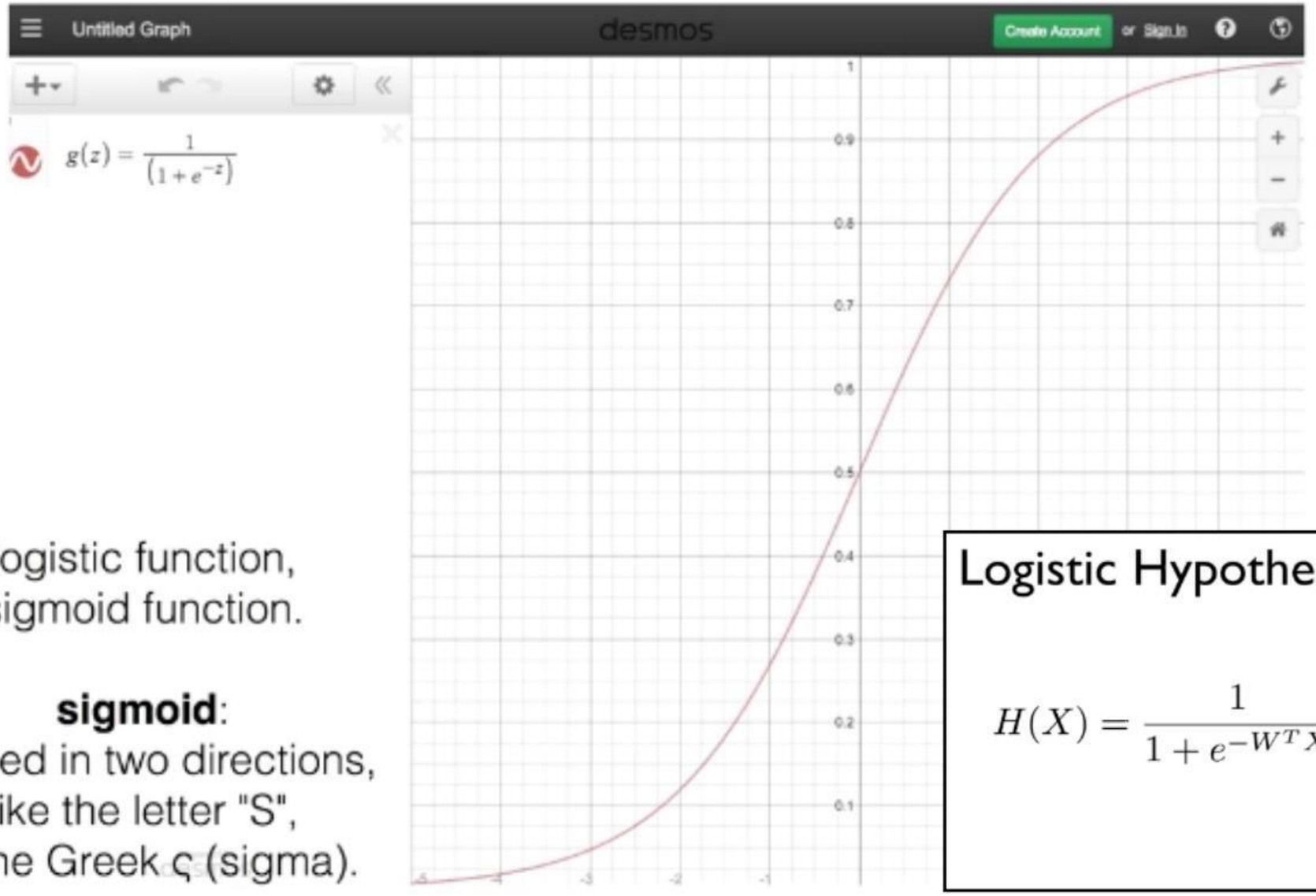
$$H(x) = Wx + b$$

- Hypothesis can give values large than 1 or less than 0

LINEAR REGRESSION?



SIGMOID



logistic function,
sigmoid function.

sigmoid:
Curved in two directions,
like the letter "S",
or the Greek ς (sigma).

Logistic Hypothesis

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

COST FUNCTION

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

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

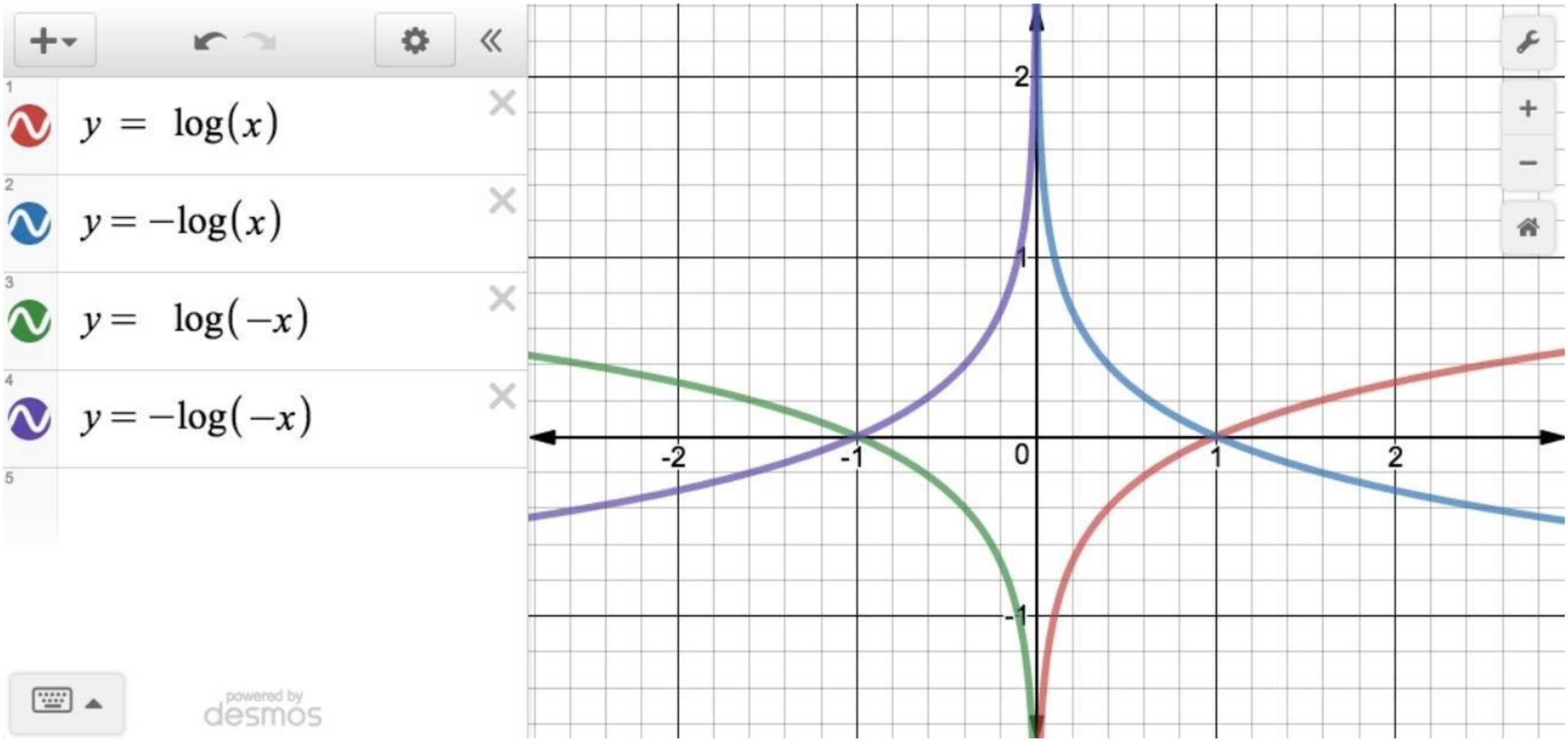
$$H(X) = \frac{1}{1+e^{-W^T X}}$$

NEW COST FUNCTION FOR LOGISTIC

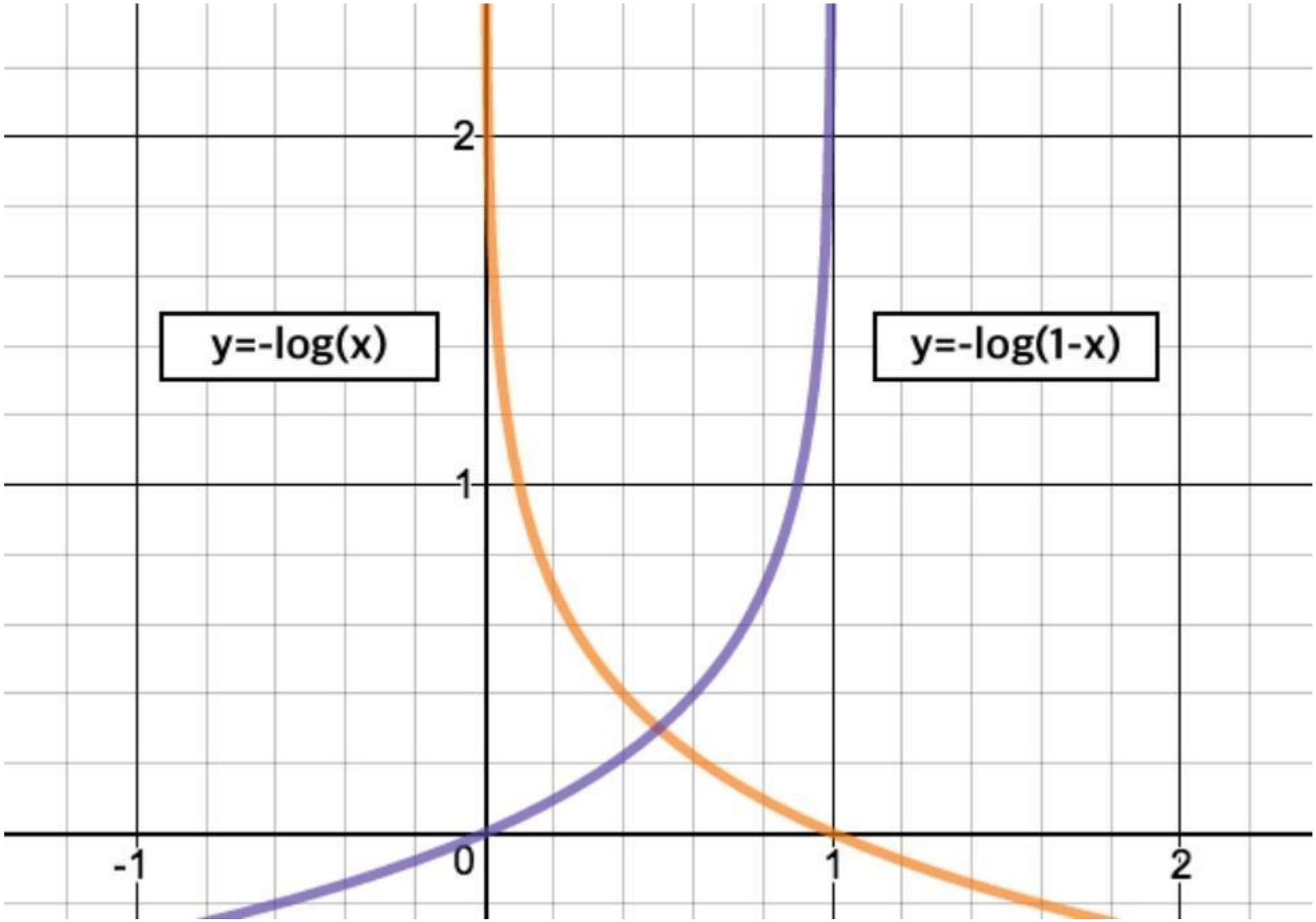
$$cost(W) = \frac{1}{m}\sum \quad c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

LOG FUNCTION



LOG FUNCTIONS WE NEED TO KNOW



UNDERSTANDING COST FUNCTION

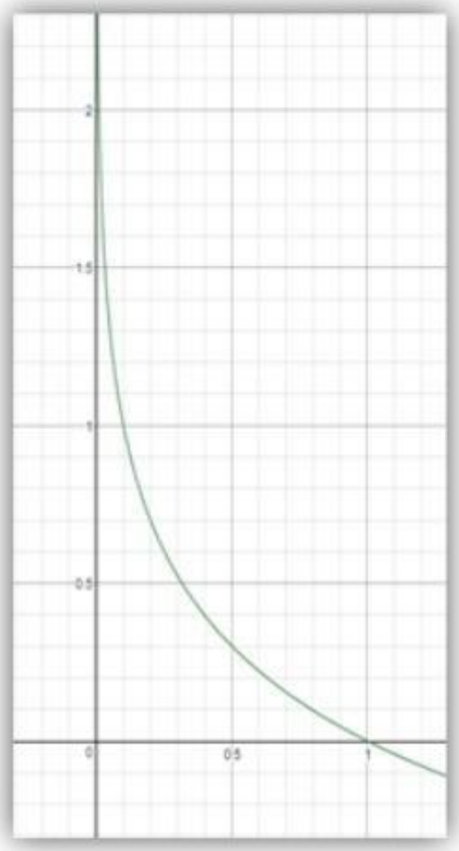
$$C(H(x),y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1-H(x)) & : y = 0 \end{cases}$$

$y = 1$

$H(x) = 1$, $cost(1) = 0$

$H(x) = 0$, $cost(0) = \infty$

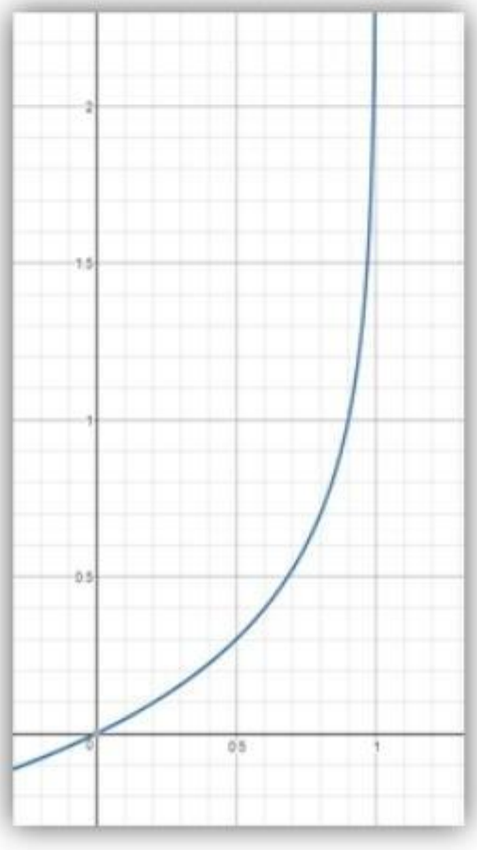
$g(z) = -\log(z)$



$y = 0$

$H(x) = 0$, $cost(0) = 0$

$H(x) = 1$, $cost(1) = \infty$



$g(z) = -\log(1-z)$

COST FUNCTION

$$cost(W) = \frac{1}{m}\sum \quad c\,(H(x), y)$$

$$c\,(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

$$c\,(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

GRADIENT DESCENT ALGORITHM

$$cost(W) = -\frac{1}{m}\sum ylog(H(x)) + (1-y)log(1-H(x))$$

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

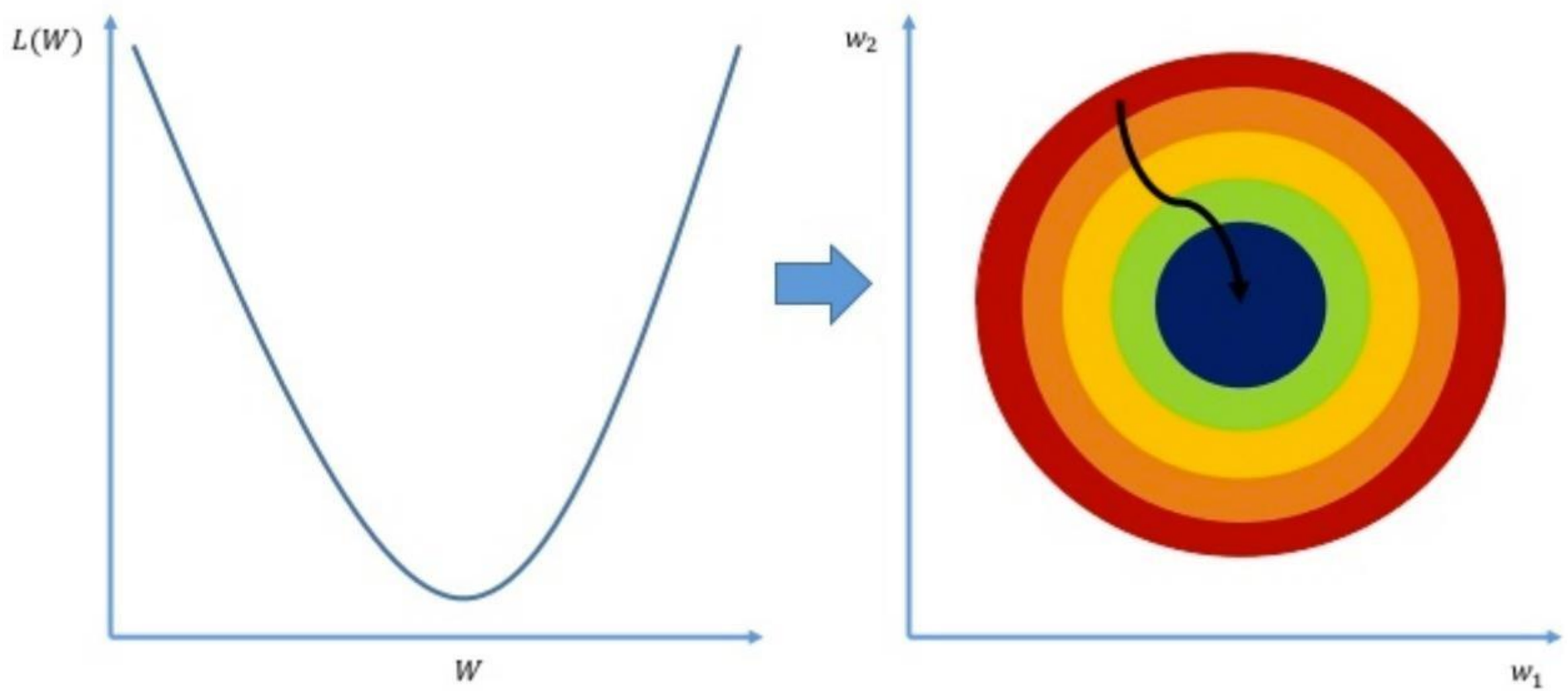
```
# cost function
cost =-tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))

# Minimize
a = tf.Variable(0.1) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)
```

DEEP LEARNING

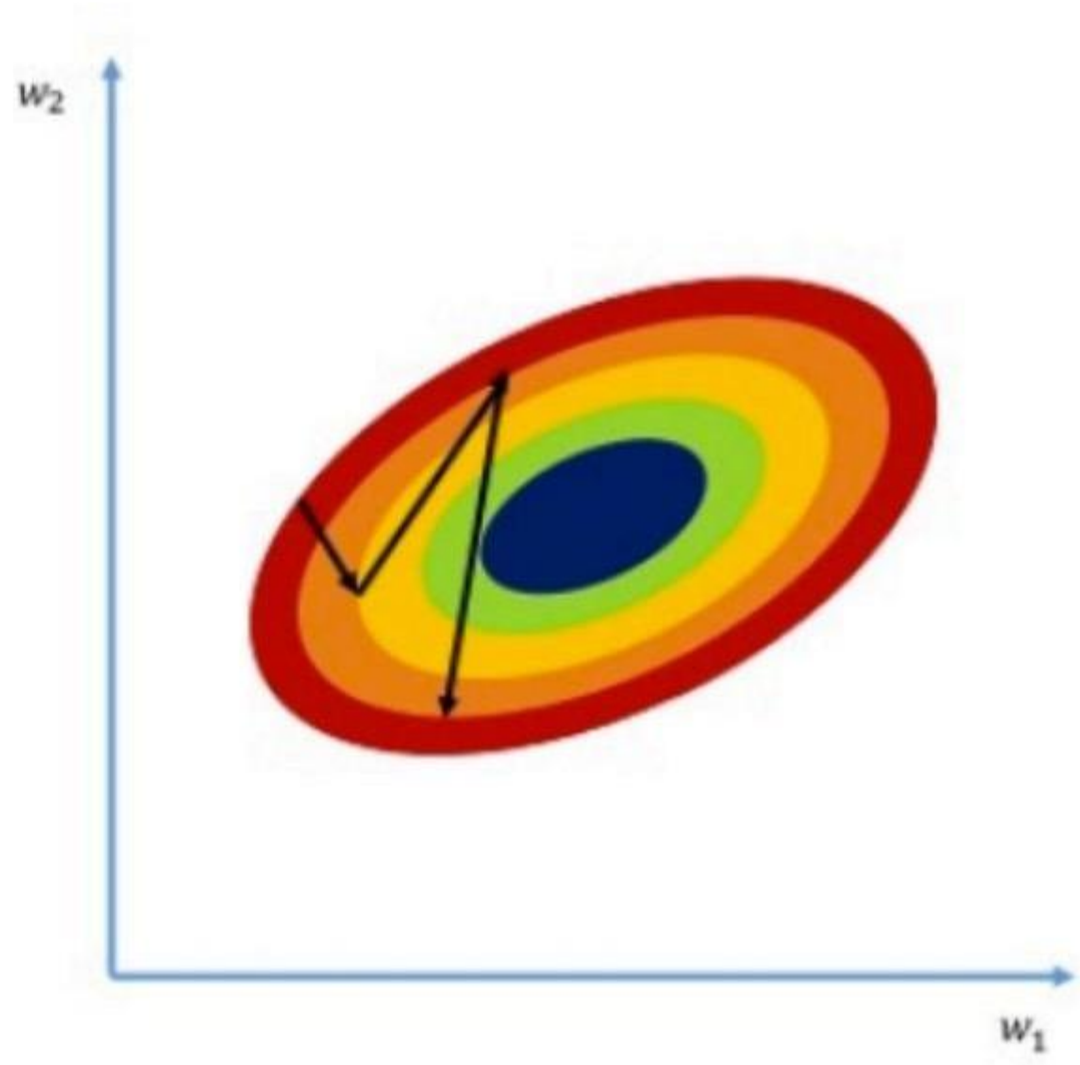
APPLICATION AND TIPS

DATA PREPROCESSING FOR GRADIENT DESCENT

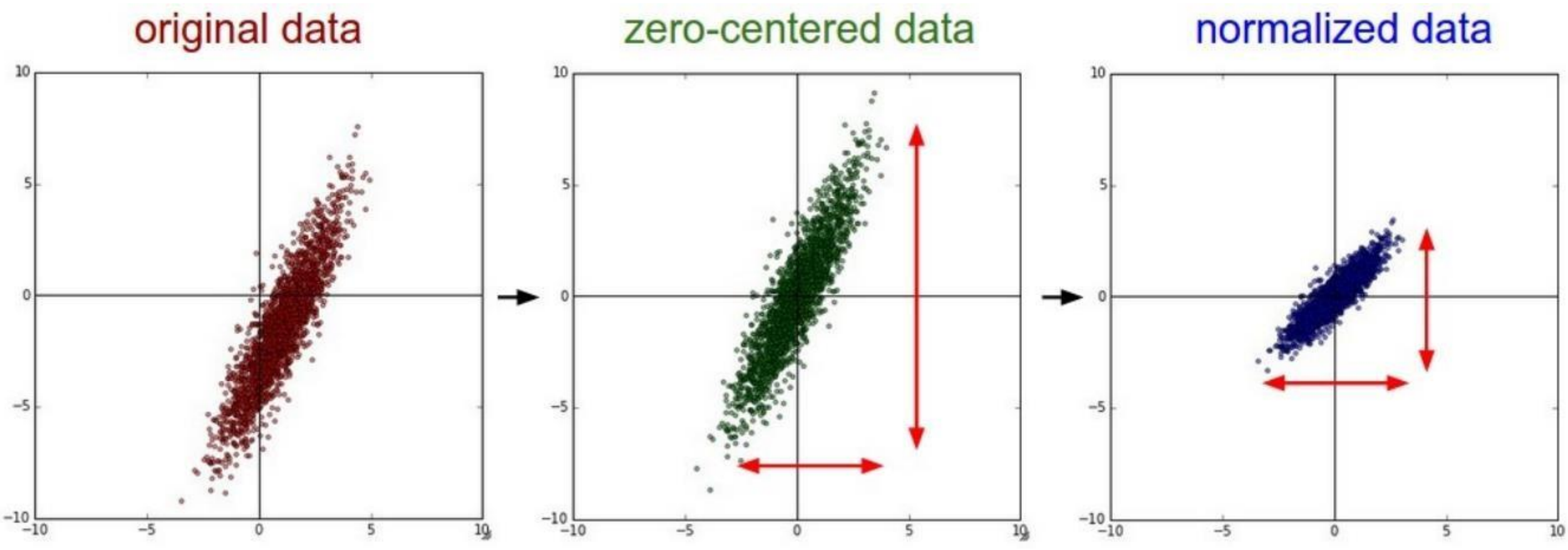


DATA PREPROCESSING FOR GRADIENT DESCENT

x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C



PREPROCESSING FOR GRADIENT DESCENT



NORMALIZATION AND STANDARDIZATION

Normalization은 "해당 속성(변수)값-최소값/최대값-최소값"으로 0~1사이의 값으로 나타내는 척도법이고

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization은 특정한 분포(ex. 정규분포)들의 평균과 분산 혹은 표준편차를 이용해 "속성값-평균/표준편차"로 해당 분포에서의 이 속성값이 평균으로부터의 위치를 표준편차 단위로 옮겨서 다시 나타낸 것이다.

$$x_{new} = \frac{x - \mu}{\sigma}$$

http://gentlej90.tistory.com/26

NORMALIZATION AND STANDARDIZATION

Normalization

수식 : (요소값 - 최소값) / (최대값 - 최소값)

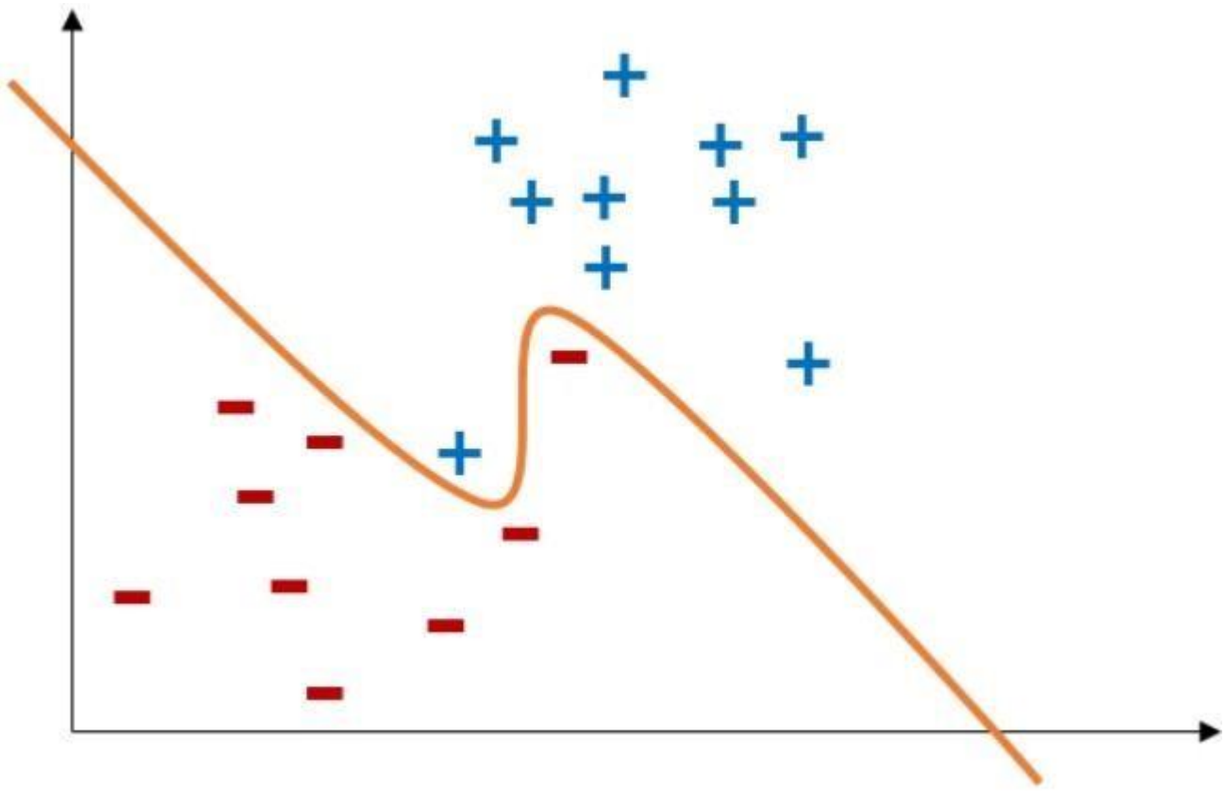
설명 : 전체 구간을 0~100 으로 설정하여 데이터를 관찰하는 방법으로 ,
특정 데이터의 위치를 확인할 수 있게 해줌

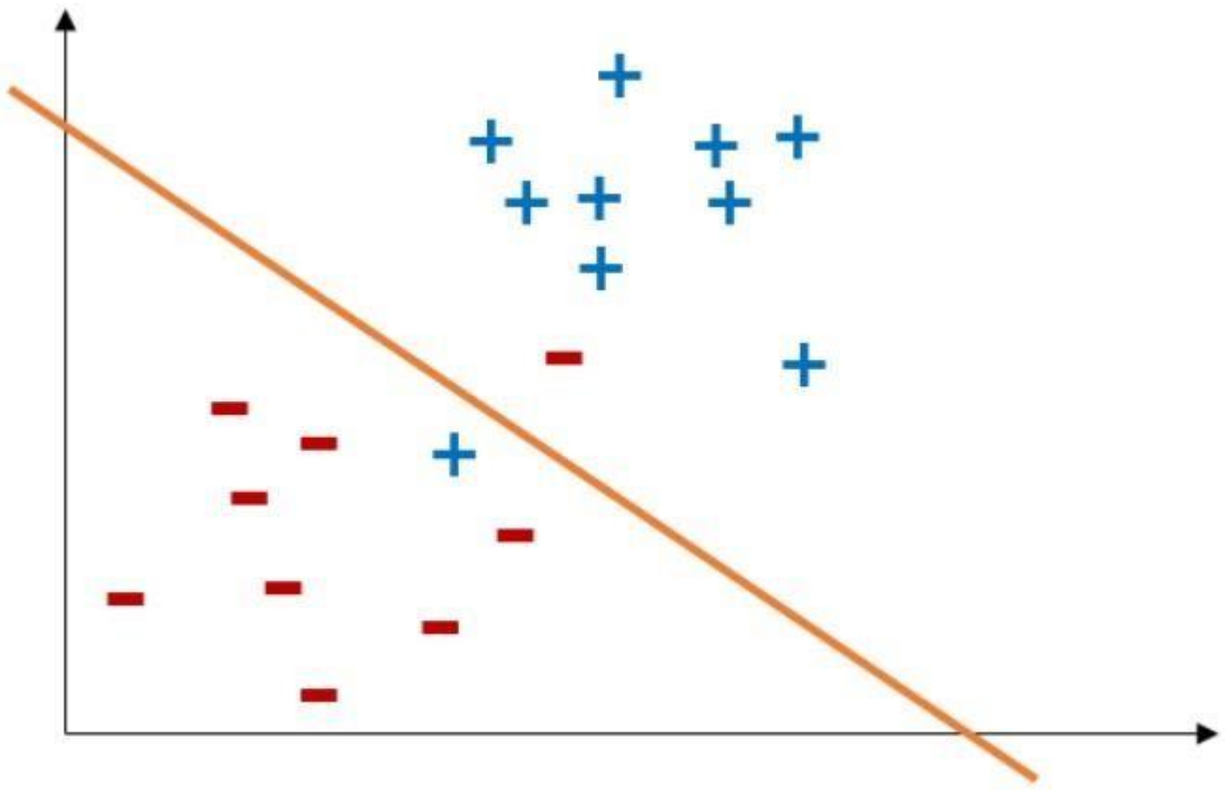
Standardization

수식 : (요소값 - 평균) / 표준편차

설명 : 평균까지의 거리로 , 2 개 이상의 대상이 단위가 다를 때 ,
대상 데이터를 같은 기준으로 볼 수 있게 해줌

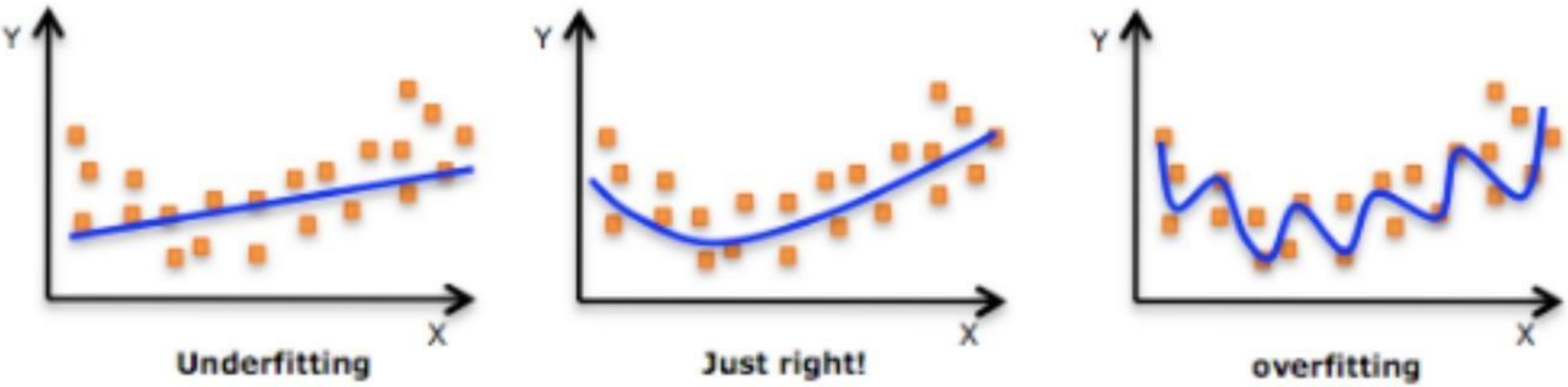
OVERFITTING





- Our model is very good with training data set (with memorization)
- Not good at test dataset or in real use

UNDERFITTING VS. OVERFITTING



SOLUTIONS FOR OVERFITTING

- More training data
- Reduce the number of features
- Regularization
- Early stopping
- Dropout

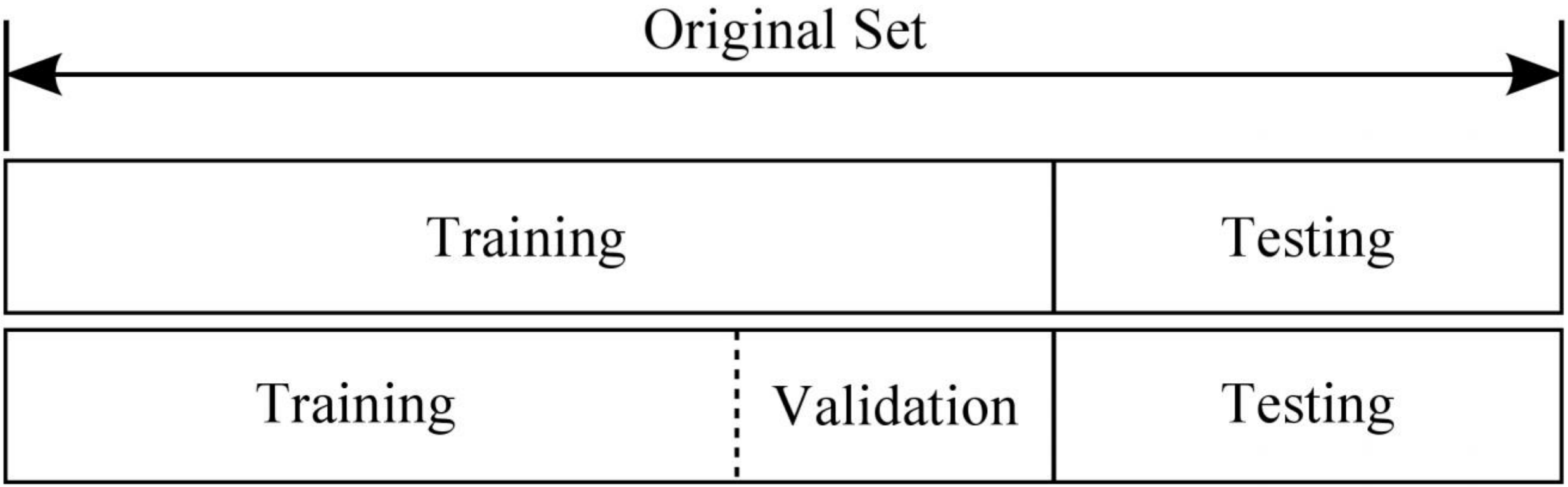
TRAINING, VALIDATION AND TEST SETS

매개변수 학습

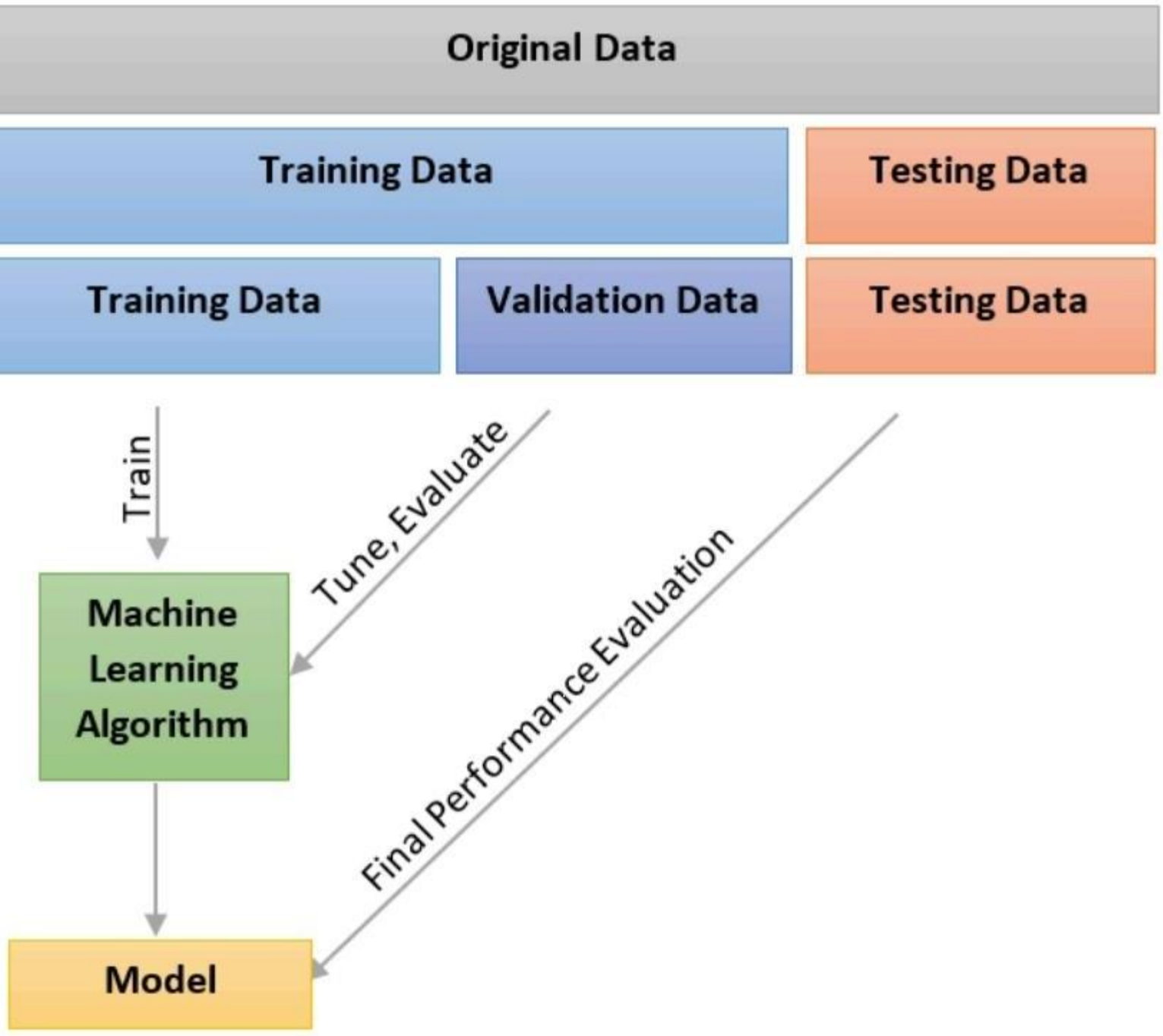
하이퍼파라미터 성능 평가

모델 성능 평가

TRAINING, VALIDATION AND TEST SETS



TRAINING, VALIDATION AND TEST SETS

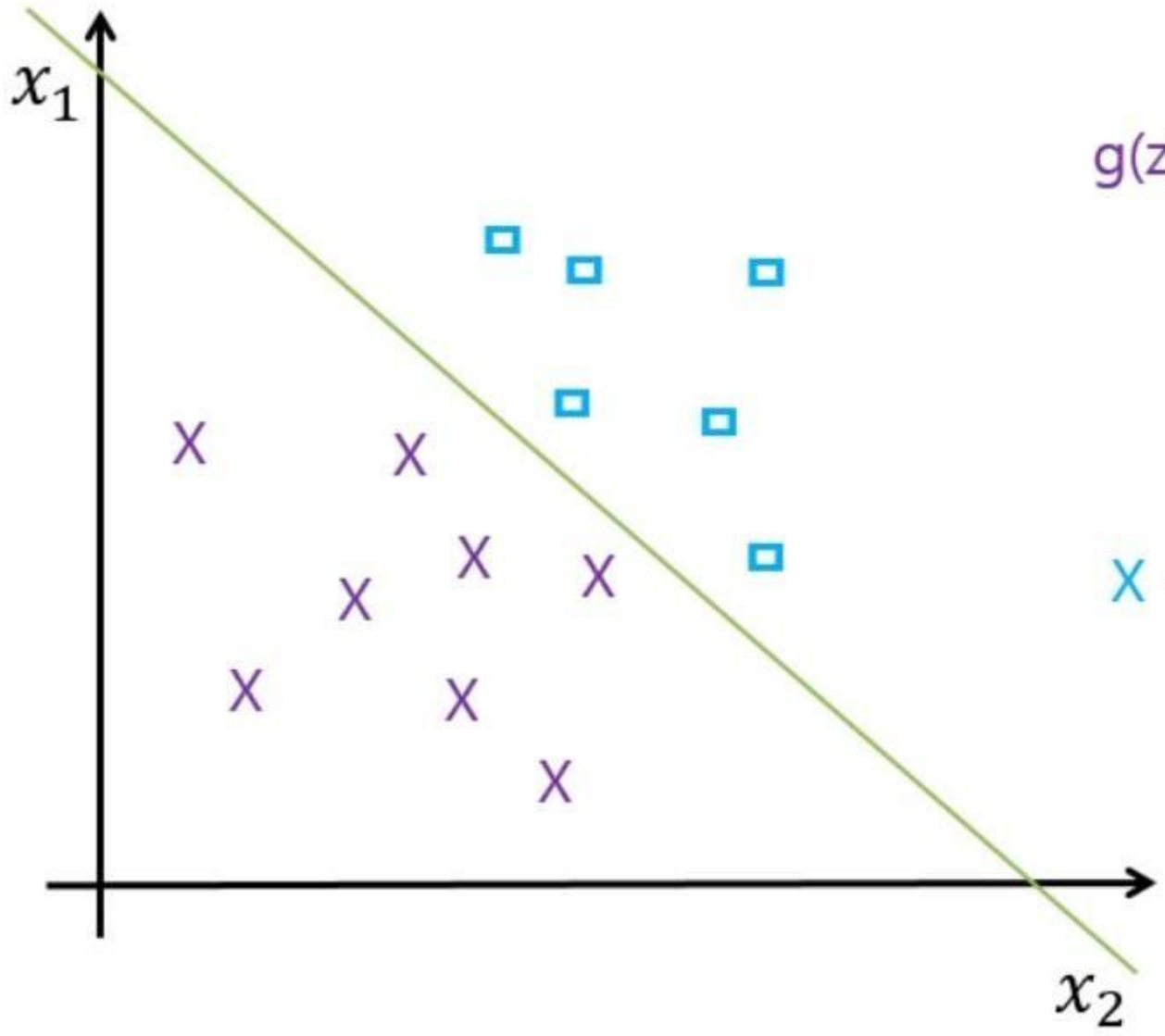


DEEP LEARNING

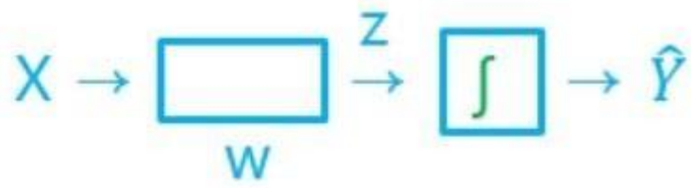
SOFTMAX

CROSS-ENTROPY

LOGISTIC REGRESSION

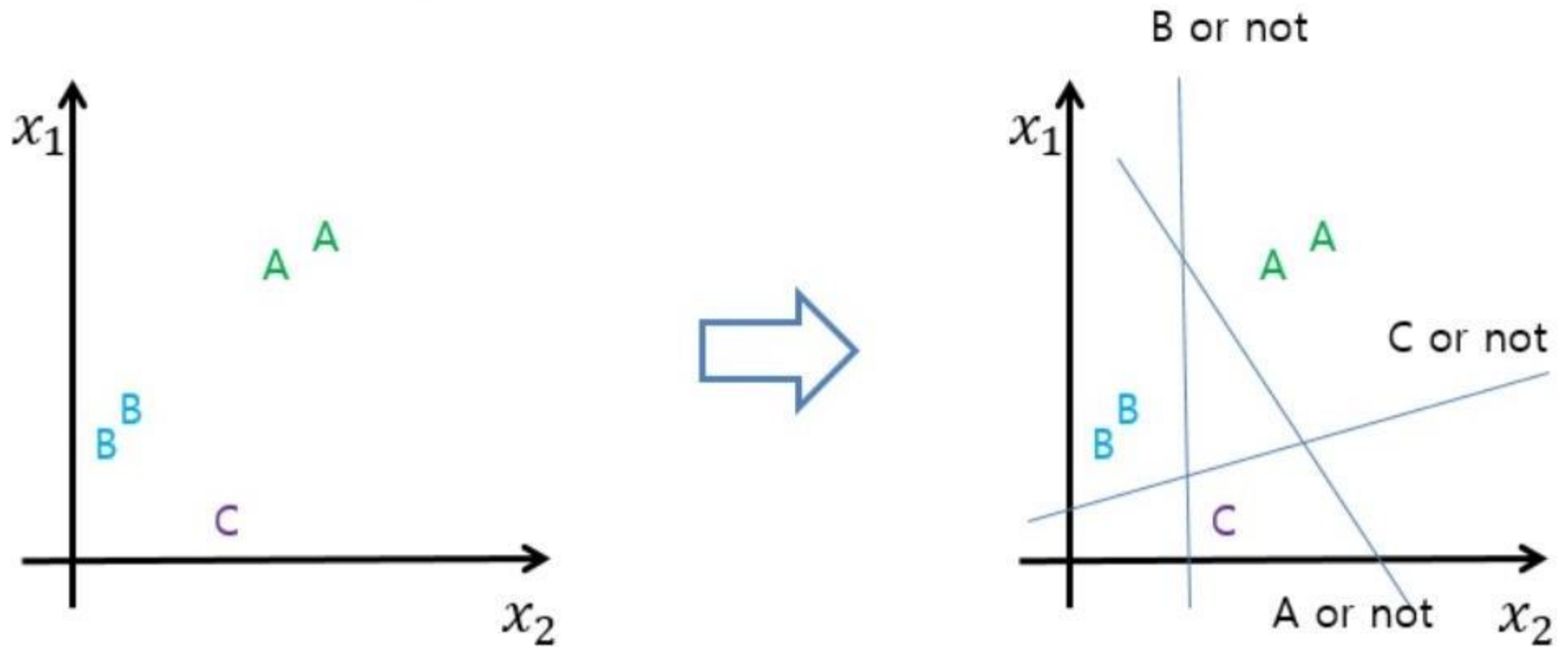


$$g(z)=\frac{1}{1+e^{-z}}$$

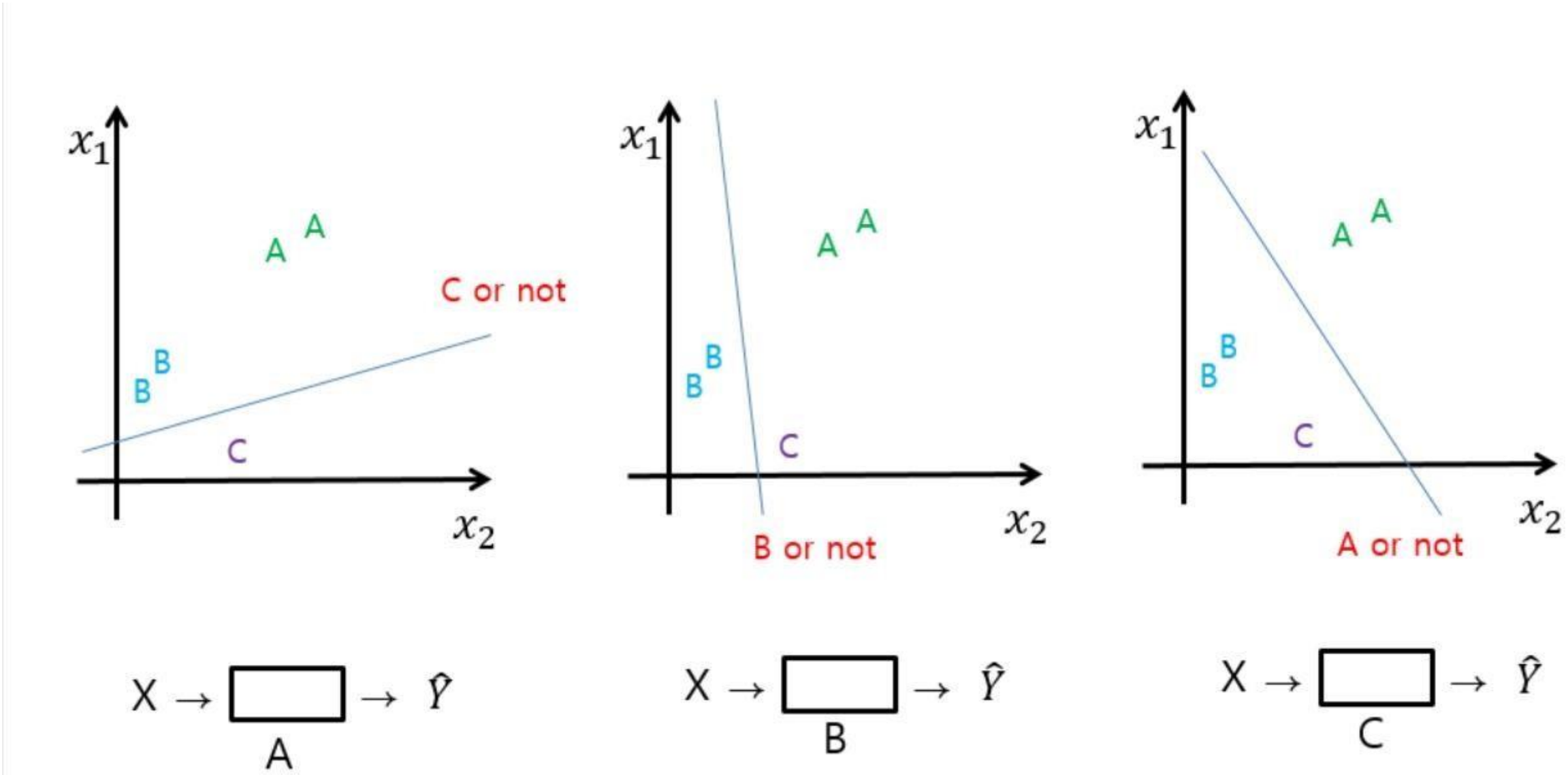


MULTINOMIAL CLASSIFICATION

x1 (hours)	x2 (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C

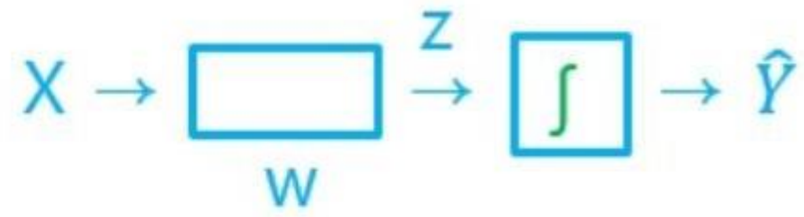


MULTINOMIAL CLASSIFICATION

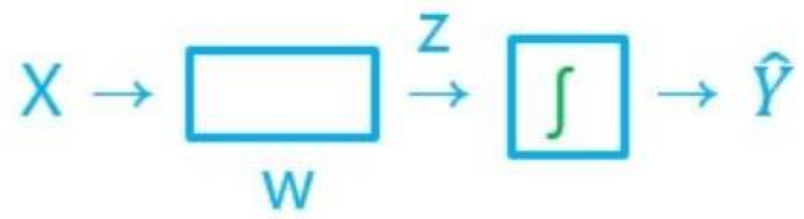


MULTINOMIAL CLASSIFICATION

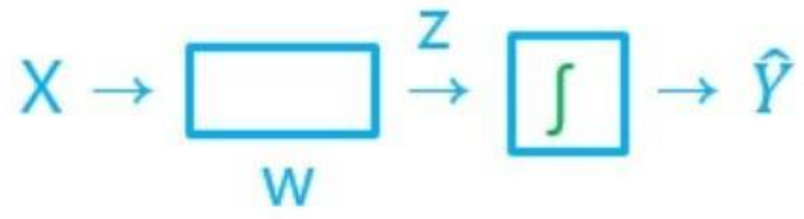
$$[w_1 \quad w_2 \quad w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$



$$[w_1 \quad w_2 \quad w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$



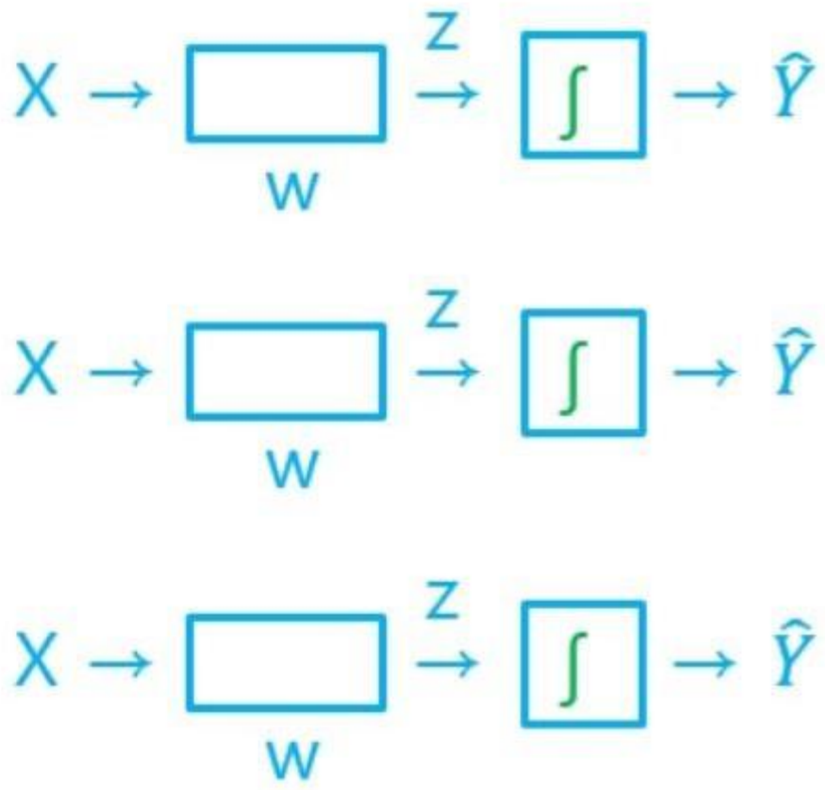
$$[w_1 \quad w_2 \quad w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$



MULTINOMIAL CLASSIFICATION

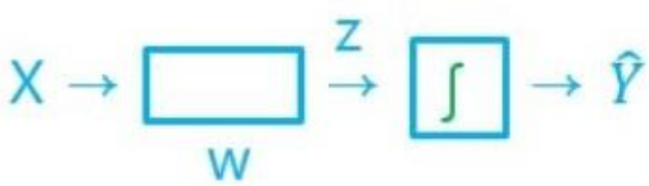
$$[w_1 \;\; w_2 \;\; w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

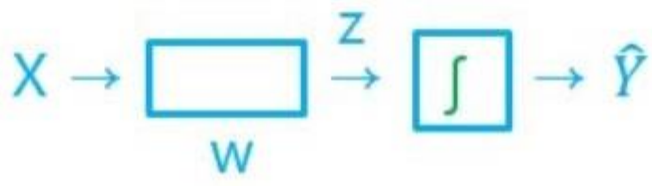
$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = ?$$

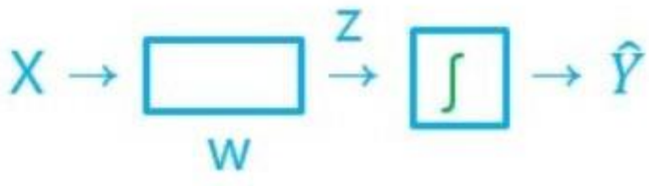


MULTINOMIAL CLASSIFICATION

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \hat{Y}_A \\ \hat{Y}_B \\ \hat{Y}_C \end{bmatrix}$$



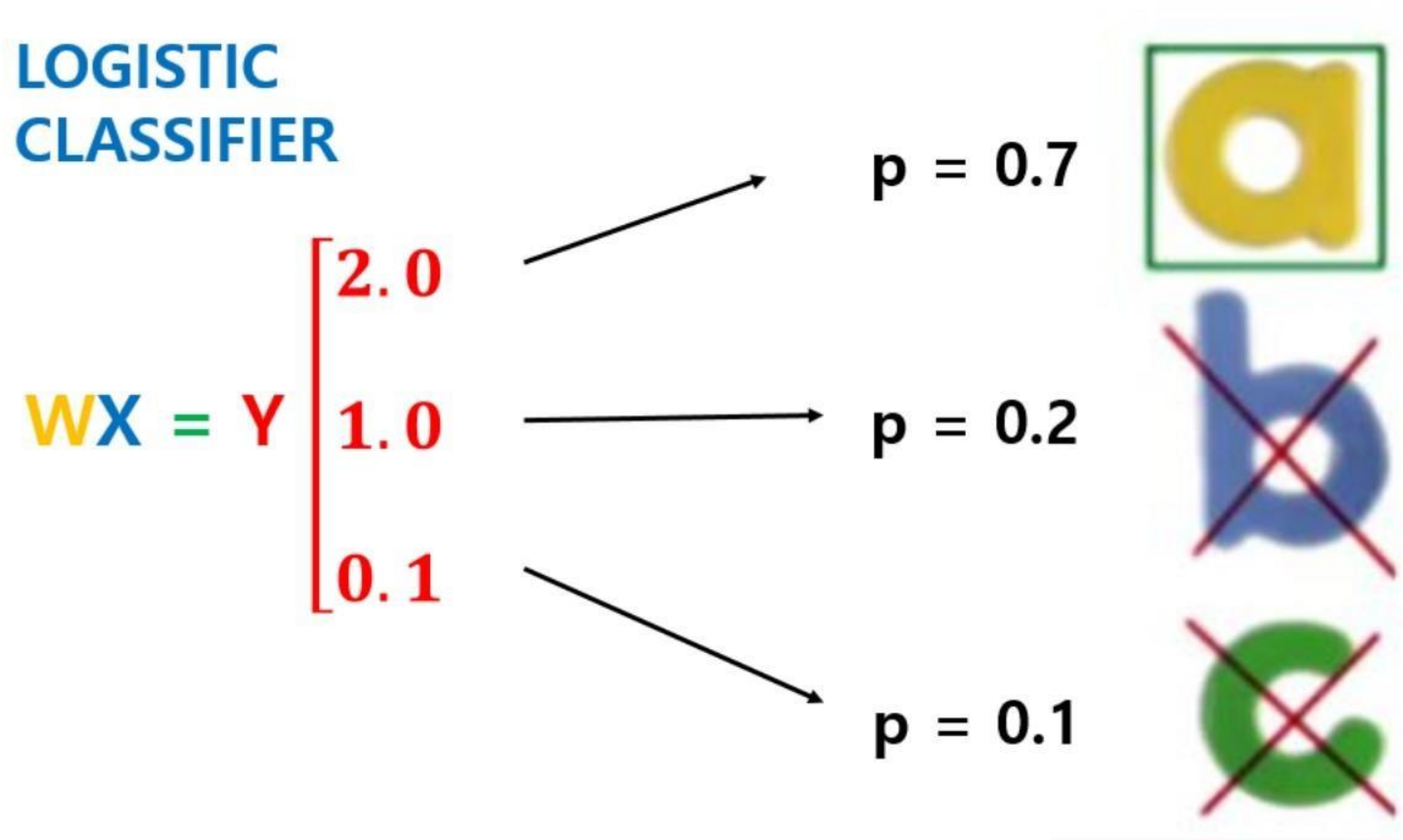




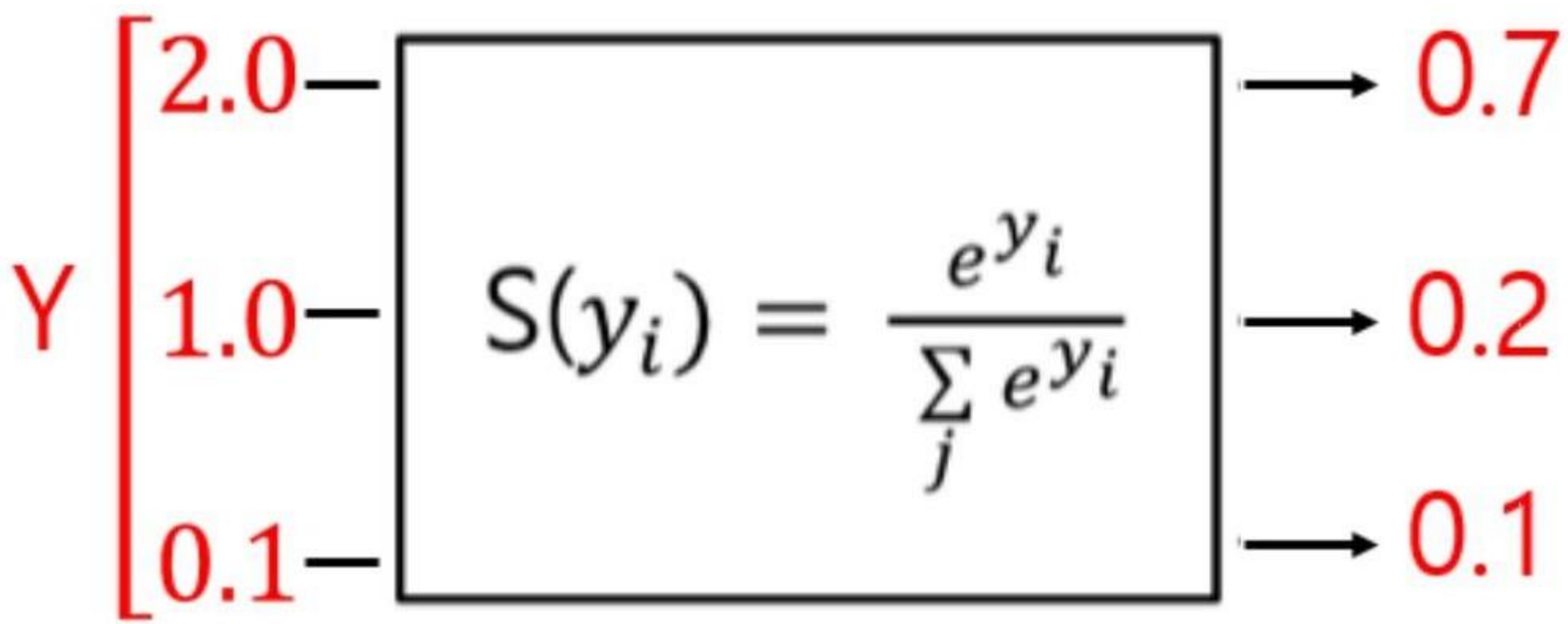
WHERE IS SIGMOID?

$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \hat{Y}_A \\ \hat{Y}_B \\ \hat{Y}_C \end{bmatrix} \begin{bmatrix} \mathbf{2.0} \\ \mathbf{1.0} \\ \mathbf{0.1} \end{bmatrix}$$

SIGMOID?



SOFTMAX

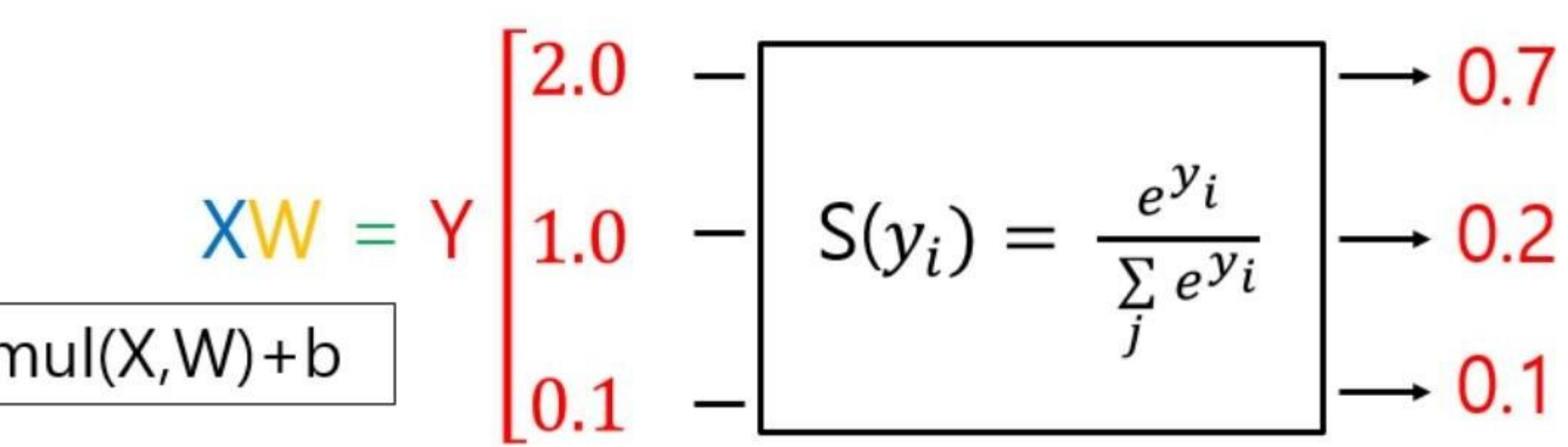


SOFTMAX

hypothesis = tf.nn.**softmax**(tf.matmul(X,W)+b)

XW = Y

tf.matmul(X,W)+b



SCORES ⟶ PROBABILITIES

SOFTMAX_CROSS_ENTROPY_WITH_LOGI

```
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)
```

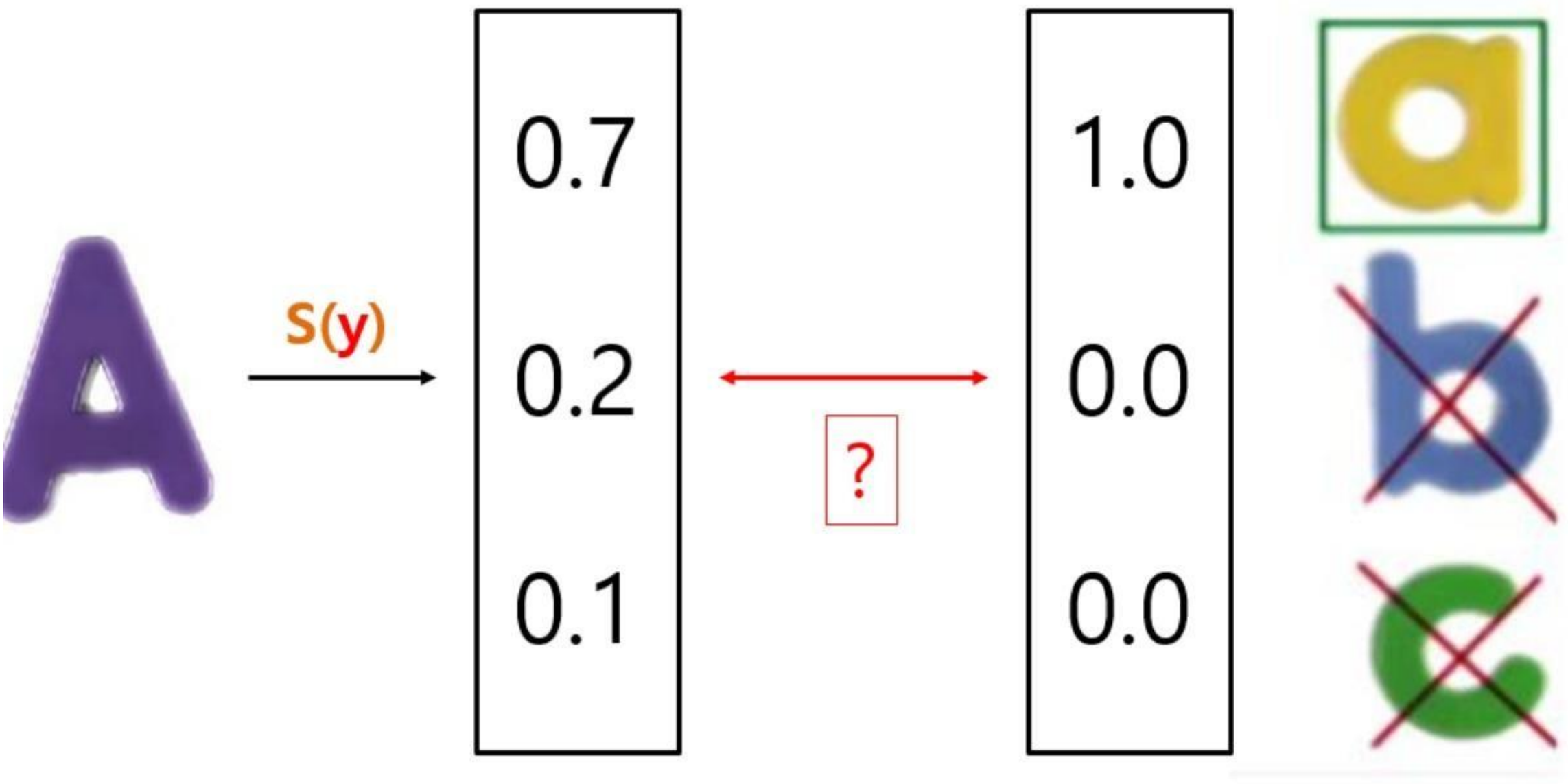
1

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

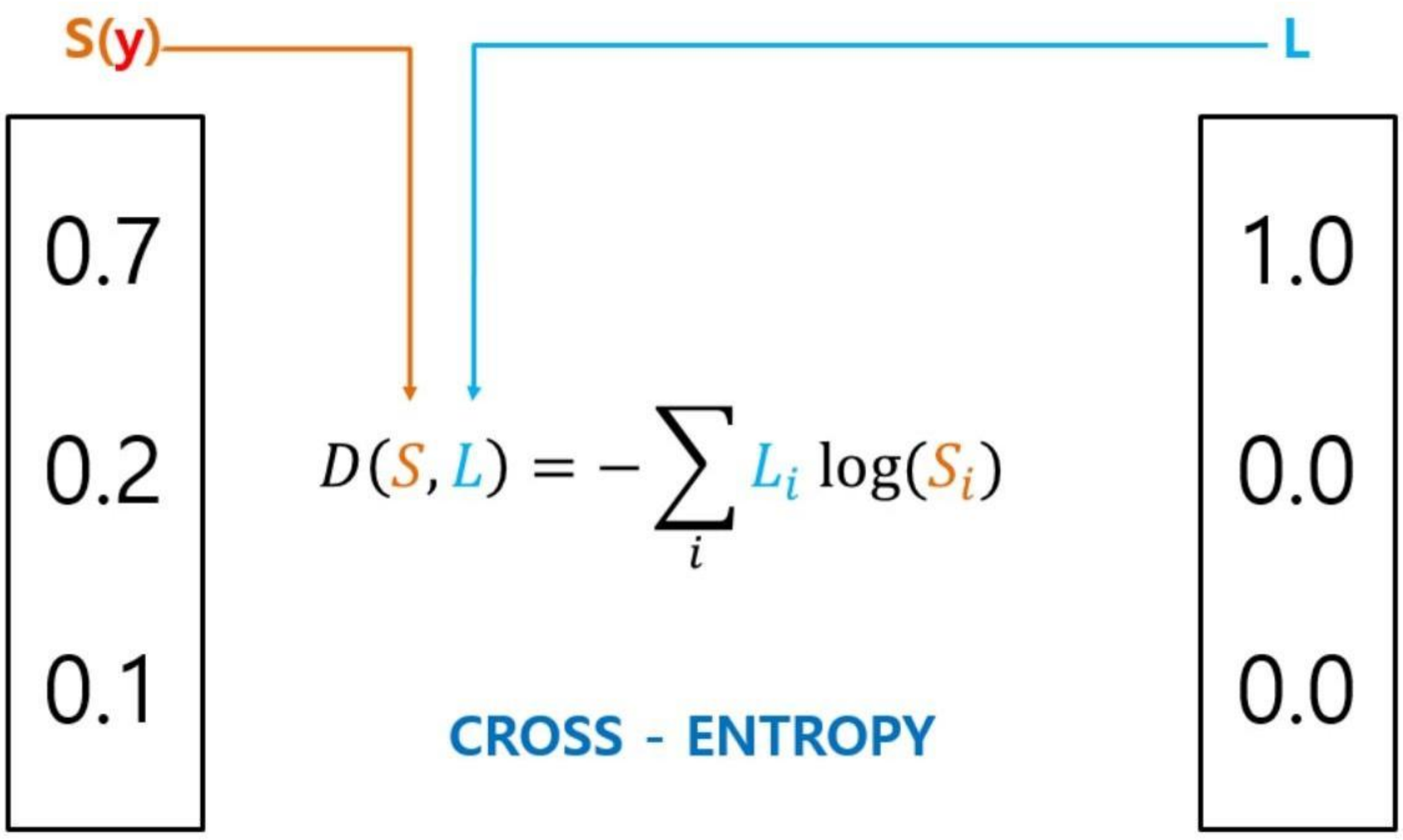
2

```
# Cross entropy cost/loss
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,
                                                 labels=Y_one_hot)
cost = tf.reduce_mean(cost_i)
```

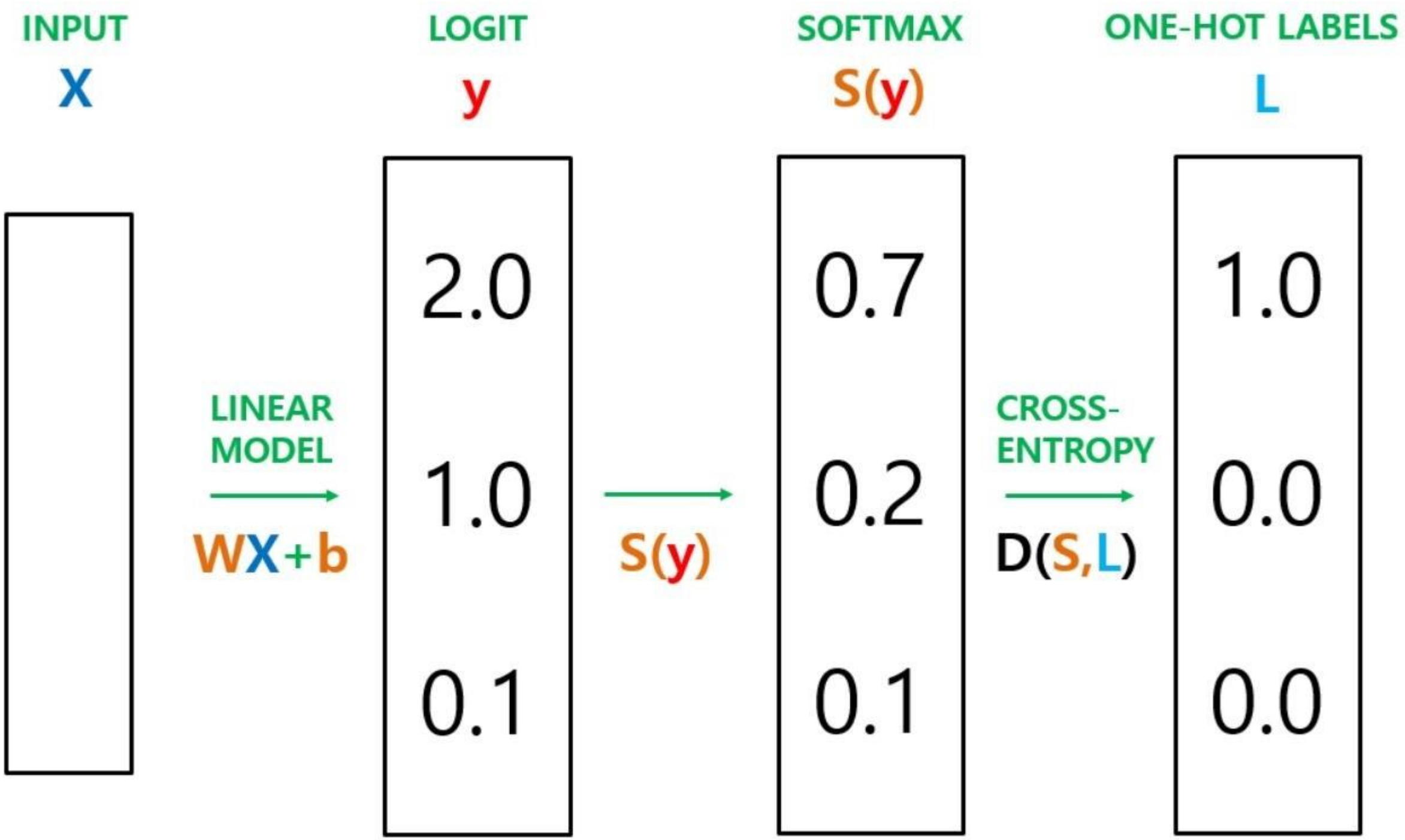
ONE-HOT ENCODING



COST FUNCTION

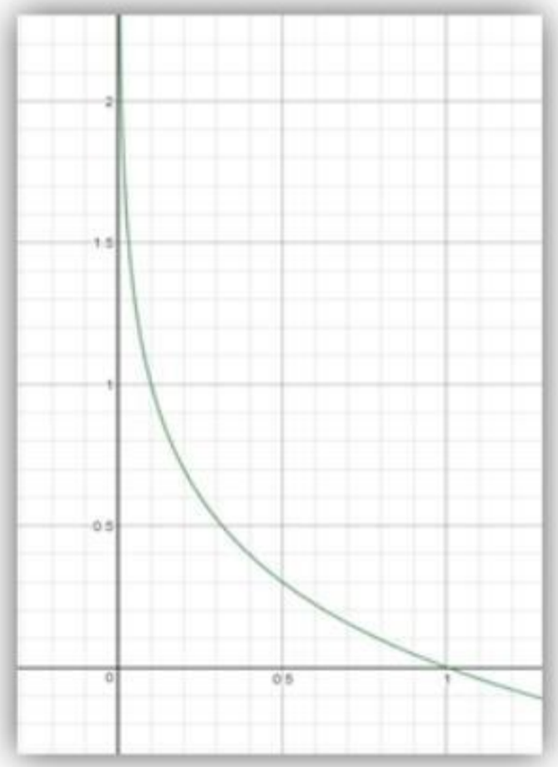


COST FUNCTION



CROSS-ENTROPY COST FUNCTION

$$-\sum_i L_i \log(S_i) \rightarrow -\sum_i L_i \log(\hat{y}_i) \rightarrow \sum_i L_i * -\log(\hat{y}_i)$$



$$L = \begin{matrix}A\\B\end{matrix}\begin{bmatrix}0\\1\end{bmatrix} = \text{B}$$

$$\hat{Y} = \begin{matrix}A\\B\end{matrix}\begin{bmatrix}0\\1\end{bmatrix} = \text{B}(0), \quad \begin{bmatrix}0\\1\end{bmatrix} * -\log\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} * \begin{bmatrix}\infty\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} = 0$$

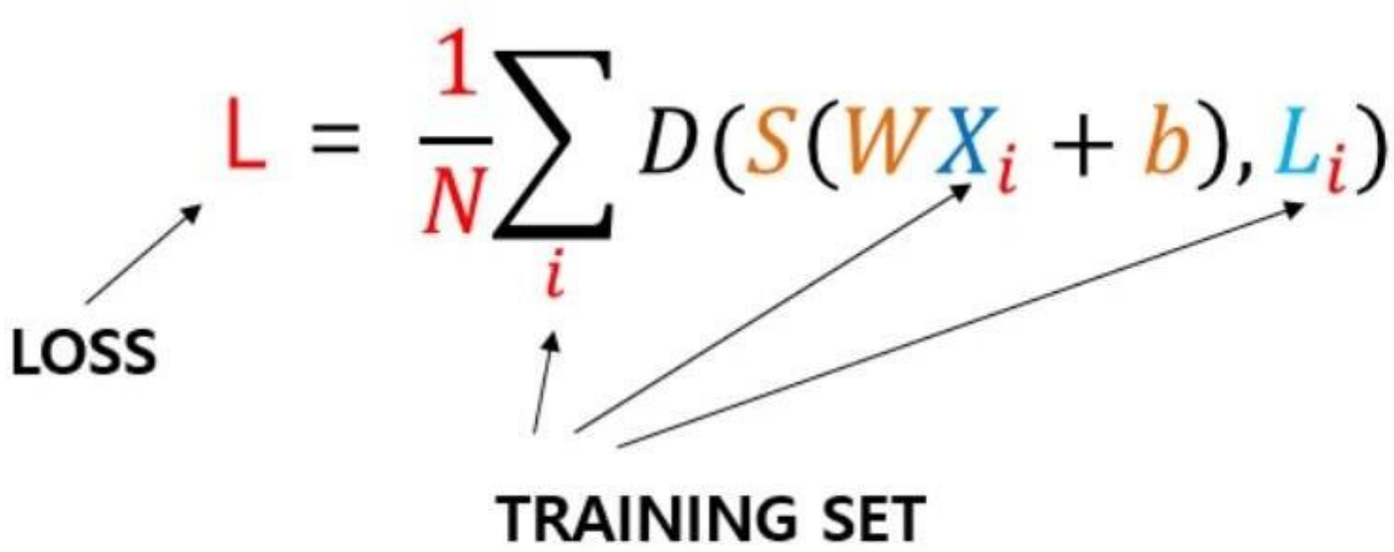
$$\hat{Y} = \begin{matrix}A\\B\end{matrix}\begin{bmatrix}1\\0\end{bmatrix} = \text{A}(\text{X}), \quad \begin{bmatrix}0\\1\end{bmatrix} * -\log\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} * \begin{bmatrix}0\\\infty\end{bmatrix} = \begin{bmatrix}0\\\infty\end{bmatrix} = \infty$$

LOGISTIC COST VS. CROSS ENTROPY

$$C(H(x), y) = -ylog\big(H(x)\big) - (1-y)\log(1-H(x))$$

$$D(S, L) = -\sum_i L_i \log(S_i)$$

COST FUNCTION : CROSS ENTROPY

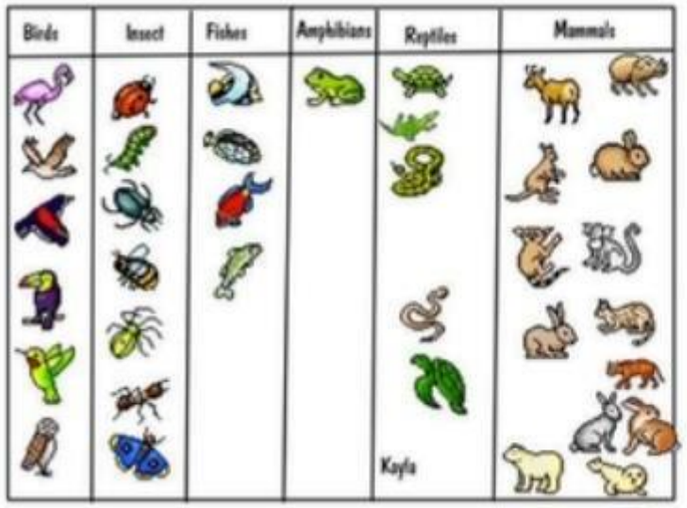



```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

ANIMAL CLASSIFICATION

with *softmax_cross_entropy_with_logits*



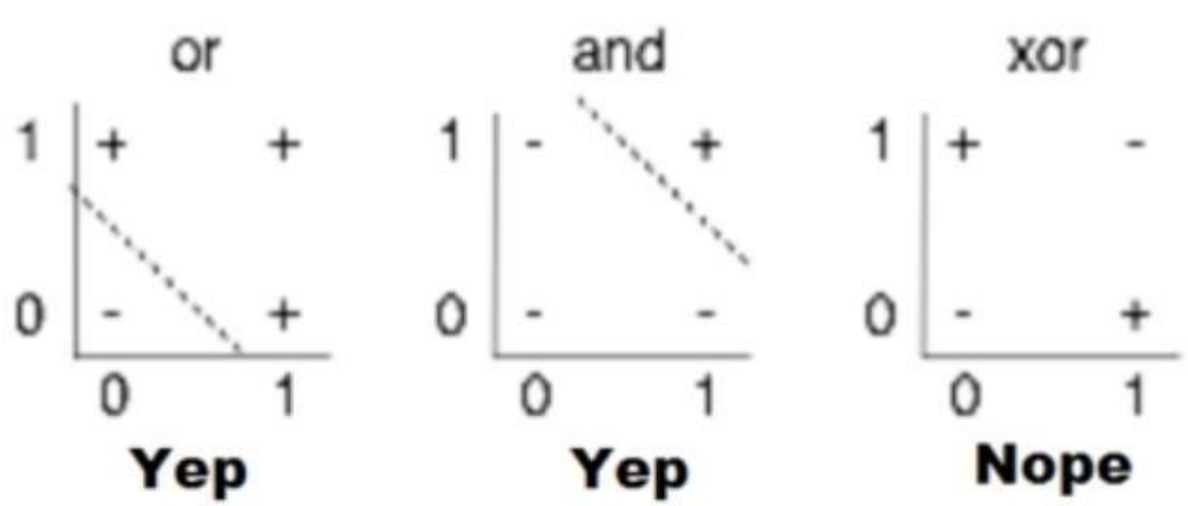
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	1
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	6
0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	1
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0

DEEP LEARNING

NEURAL NETWORKS HISTORY

XOR PROBLEM

(Simple) XOR problem: linearly separable?



$$W = \begin{pmatrix} -11 \\ -11 \end{pmatrix}$$

$$B = 6$$

x1	x2	y1	y2	y3		
0	0	0	1	0		
0	1	0	0	1		
1	0	0	0	1		
1	1	1	0	0		

$$W = \begin{pmatrix} 5 \\ 5 \end{pmatrix} \qquad B = -8$$

$$W = \begin{pmatrix} -7 \\ -7 \end{pmatrix} \qquad B = 3$$

$$\begin{pmatrix} 0 & 0 \end{pmatrix} \times \begin{pmatrix} -11 \\ -11 \end{pmatrix} + 6 = \begin{pmatrix} \\ \end{pmatrix}$$

$$W = \begin{pmatrix} -11 \\ -11 \end{pmatrix}$$

$$B = 6$$

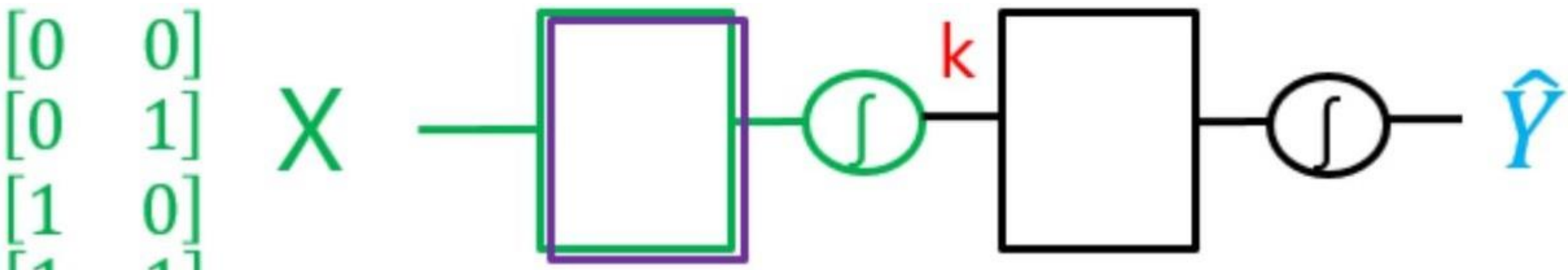
x1	x2	y1	y2	y3		
0	0	0	1	0		
0	1	0	0	1		
1	0	0	0	1		
1	1	1	0	0		

$$W = \begin{pmatrix} 5 & -7 \\ 5 & -7 \end{pmatrix}$$

$$B = -8 \qquad B = 3$$

$$\begin{pmatrix} 0 & 0 \end{pmatrix} \times \begin{pmatrix} -11 \\ -11 \end{pmatrix} + 6 = \begin{pmatrix} \\ \end{pmatrix}$$

NEURAL NETWORK FOR XOR



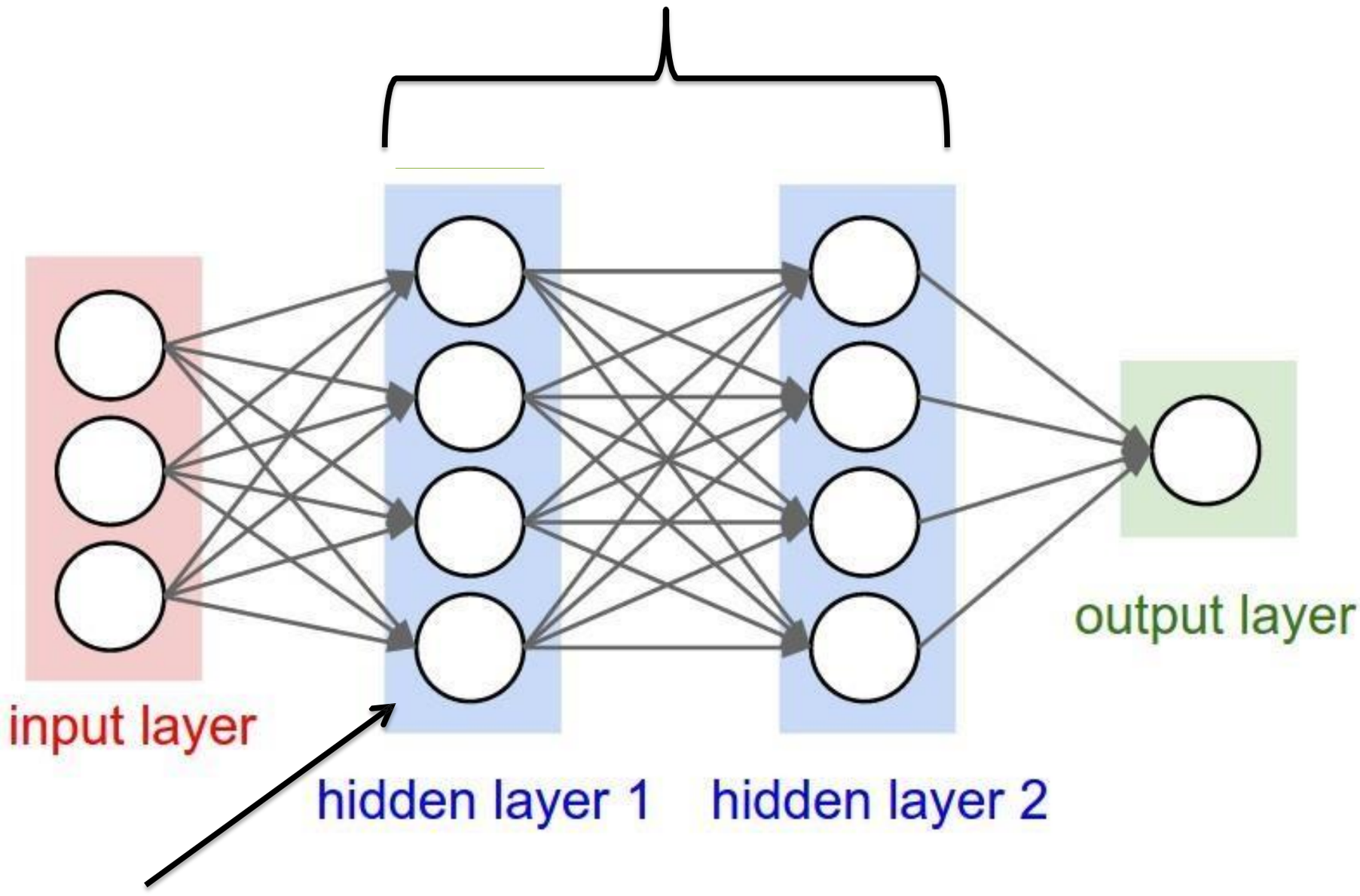
$$W_1 = \begin{bmatrix} 5 & -7 \\ 5 & -7 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} -11 \\ -11 \end{bmatrix}$$

$$B_1 = [-8 \quad 3]$$

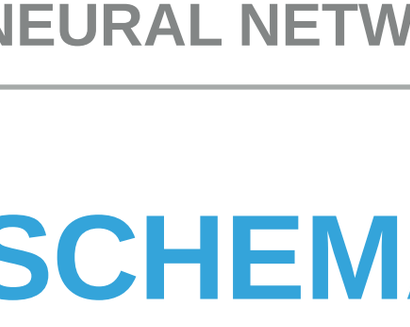
$$B_2 = 6$$

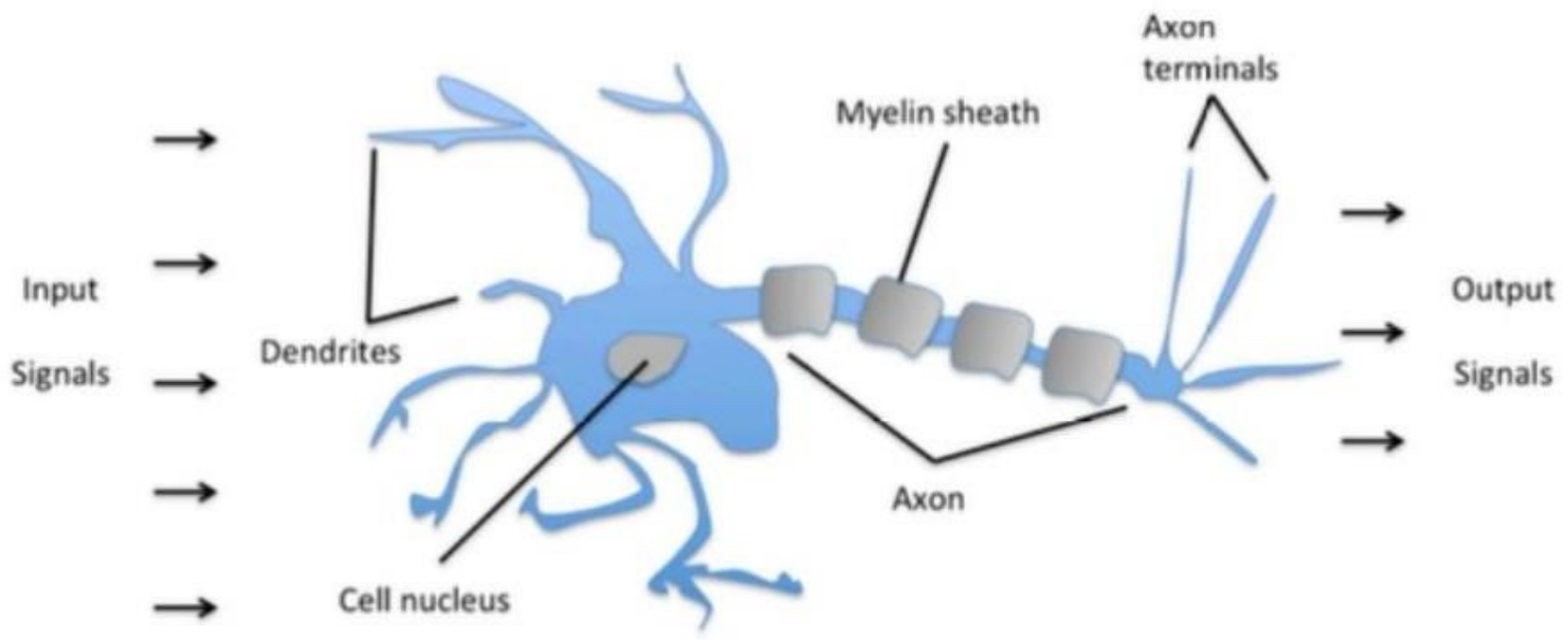
Deep



Wide

SCHEMATIC OF A BIOLOGICAL NEURON

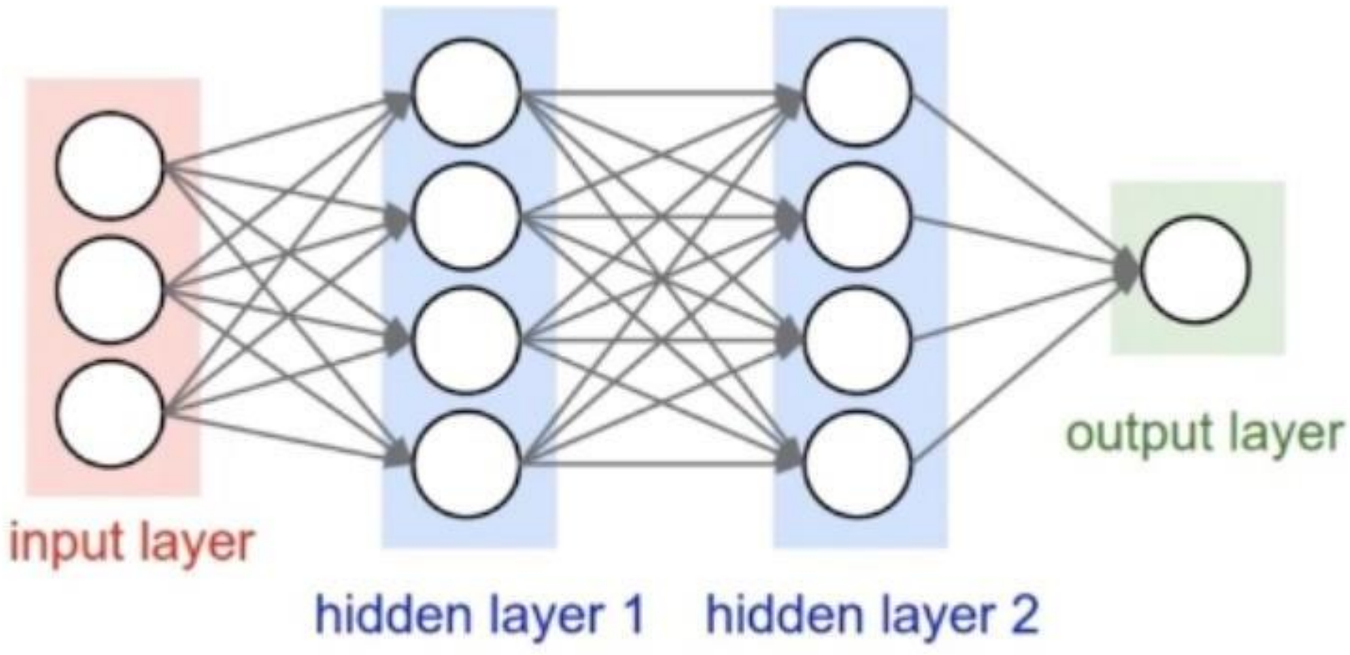




Schematic of a biological neuron.

MARVIN MINSKY, 1969

"No one on earth had found a viable way to train*"

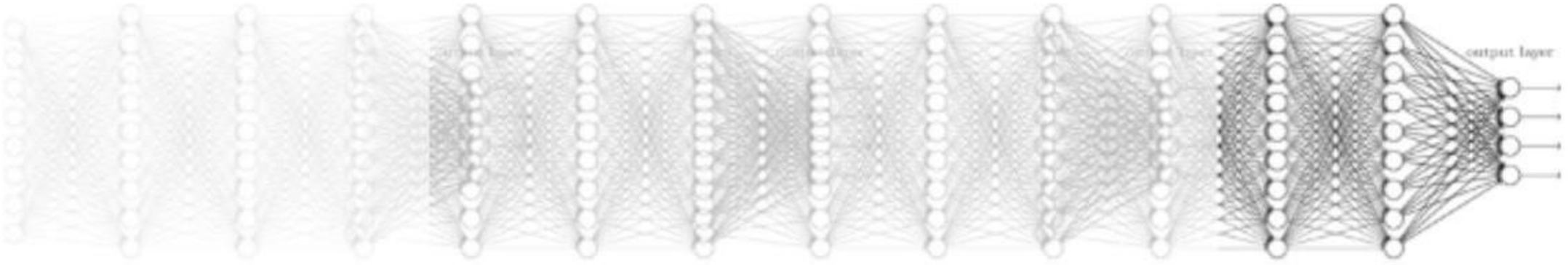


*Marvin Minsky, 1969

http://cs231n.github.io/convolutional-networks/

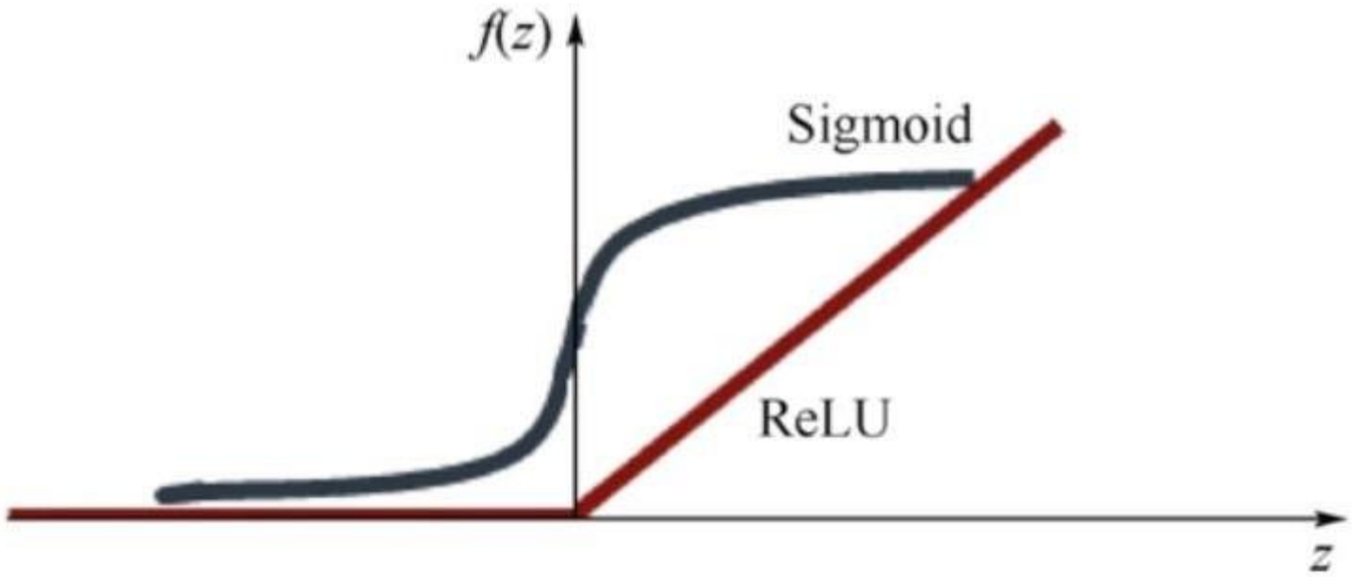
VANISHING GRADIENT

Vanishing gradient (NN winter2: 1986-2006)



SIGMOID VS. RELU

Sigmoid!



ReLU: Rectified Linear Unit

```
L1 = tf.sigmoid(tf.matmul(X, W1) + b1)
```

```
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
```

RELU

ReLu

```
# Our hypothesis
with tf.name_scope("layer1") as scope:
    L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
with tf.name_scope("layer2") as scope:
    L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)
with tf.name_scope("layer3") as scope:
    L3 = tf.nn.relu(tf.matmul(L2, W3) + b3)
with tf.name_scope("layer4") as scope:
    L4 = tf.nn.relu(tf.matmul(L3, W4) + b4)
with tf.name_scope("layer5") as scope:
    L5 = tf.nn.relu(tf.matmul(L4, W5) + b5)
with tf.name_scope("layer6") as scope:
    L6 = tf.nn.relu(tf.matmul(L5, W6) + b6)
with tf.name_scope("layer7") as scope:
    L7 = tf.nn.relu(tf.matmul(L6, W7) + b7)
with tf.name_scope("layer8") as scope:
    L8 = tf.nn.relu(tf.matmul(L7, W8) + b8)
with tf.name_scope("layer9") as scope:
    L9 = tf.nn.relu(tf.matmul(L8, W9) + b9)
with tf.name_scope("layer10") as scope:
    L10 = tf.nn.relu(tf.matmul(L9, W10) + b10)

with tf.name_scope("last") as scope:
    hypothesis = tf.sigmoid(tf.matmul(L10, W11) + b11)
```

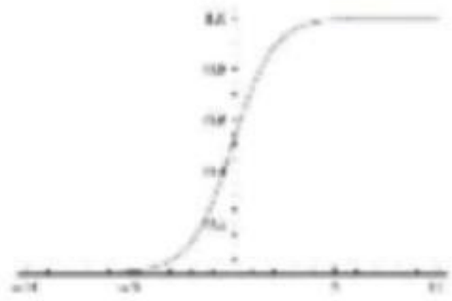
Works very well

```
196000 [2.6226094e-06, array([[  2.59195826e-06],
       [  9.99999642e-01],
       [  9.99994874e-01],
       [  2.43454133e-06]], dtype=float32)]
198000 [2.607708e-06, array([[  2.55822852e-06],
       [  9.99999642e-01],
       [  9.99994874e-01],
       [  2.40260101e-06]], dtype=float32)]
[array([[  2.52509381e-06],
       [  9.99999642e-01],
       [  9.99994874e-01],
       [  2.37124709e-06]], dtype=float32), array([[ 0.],
       [ 1.],
       [ 1.],
       [ 0.]], dtype=float32)]
Accuracy: 1.0
```

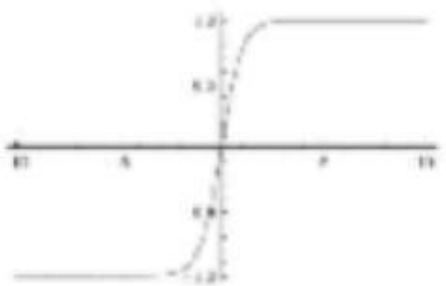
ACTIVATION FUNCTIONS

Sigmoid

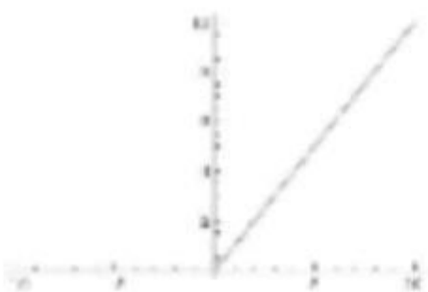
$$\sigma(x) = 1/(1+e^{-x})$$



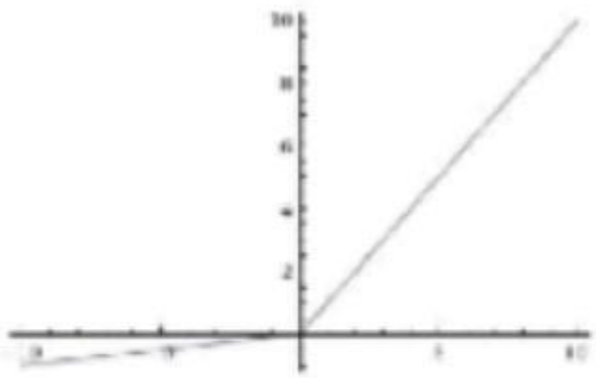
tanh tanh(x)



ReLU max(0,x)

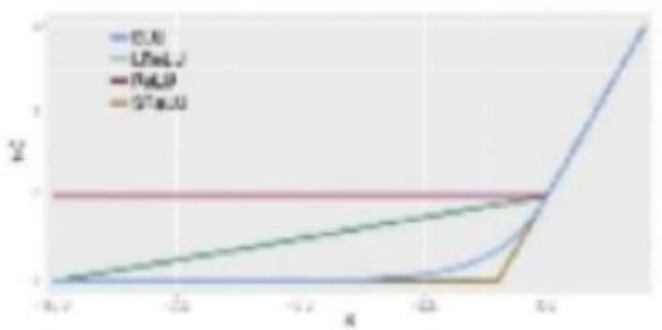


Leaky ReLU
max(0.1x, x)

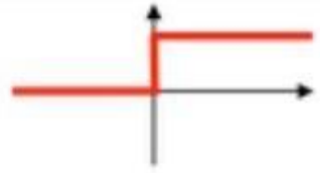
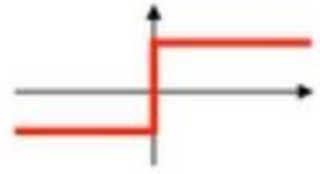
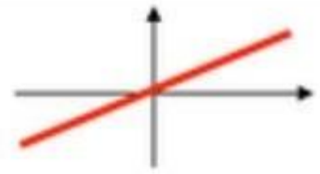
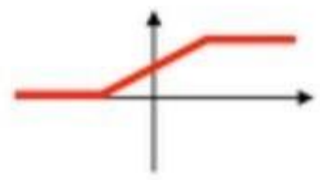
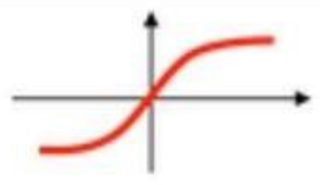


Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$



ACTIVATION FUNCTIONS

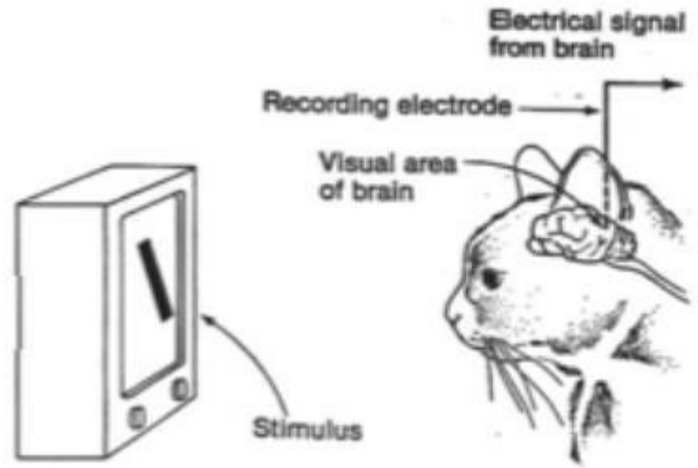
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

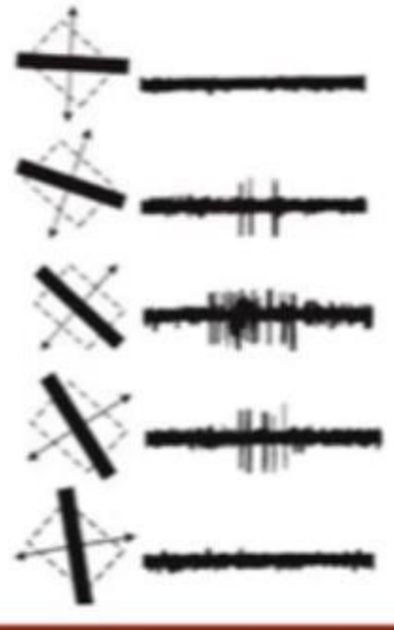
DEEP LEARNING

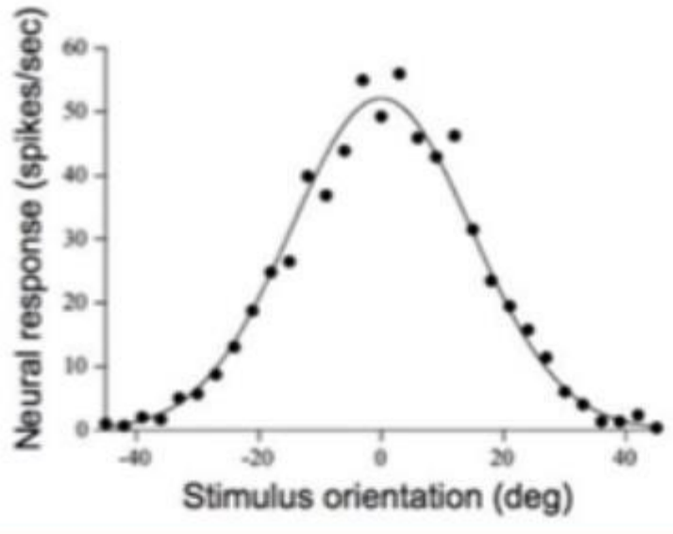
CONVOLUTIONAL NEURAL NETWORK

A BIT OF HISTORY

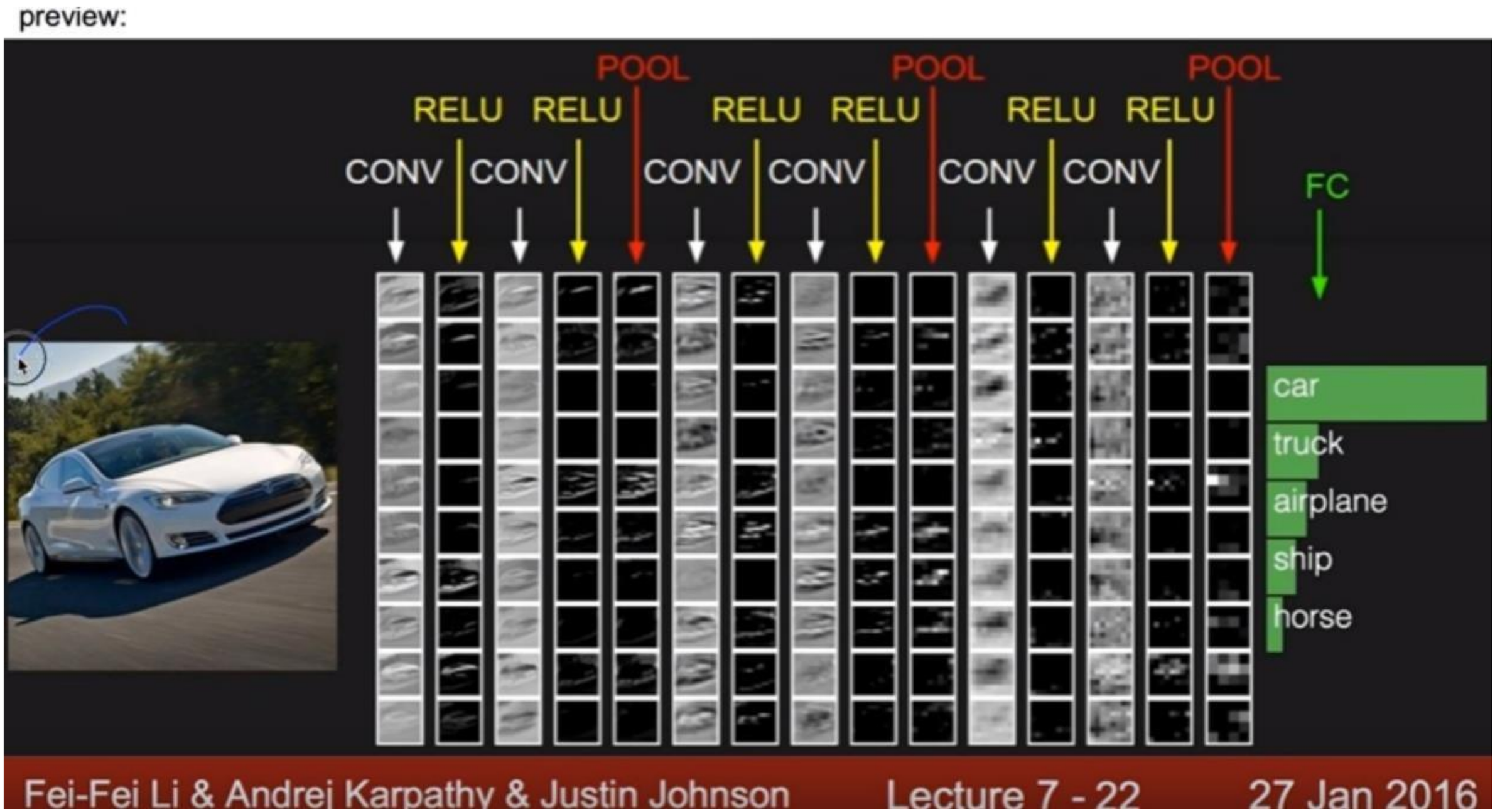
Hubel & Wiesel,
1959
RECEPTIVE FIELDS OF SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX
1962
RECEPTIVE FIELDS, BINOCULAR INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX
1968...



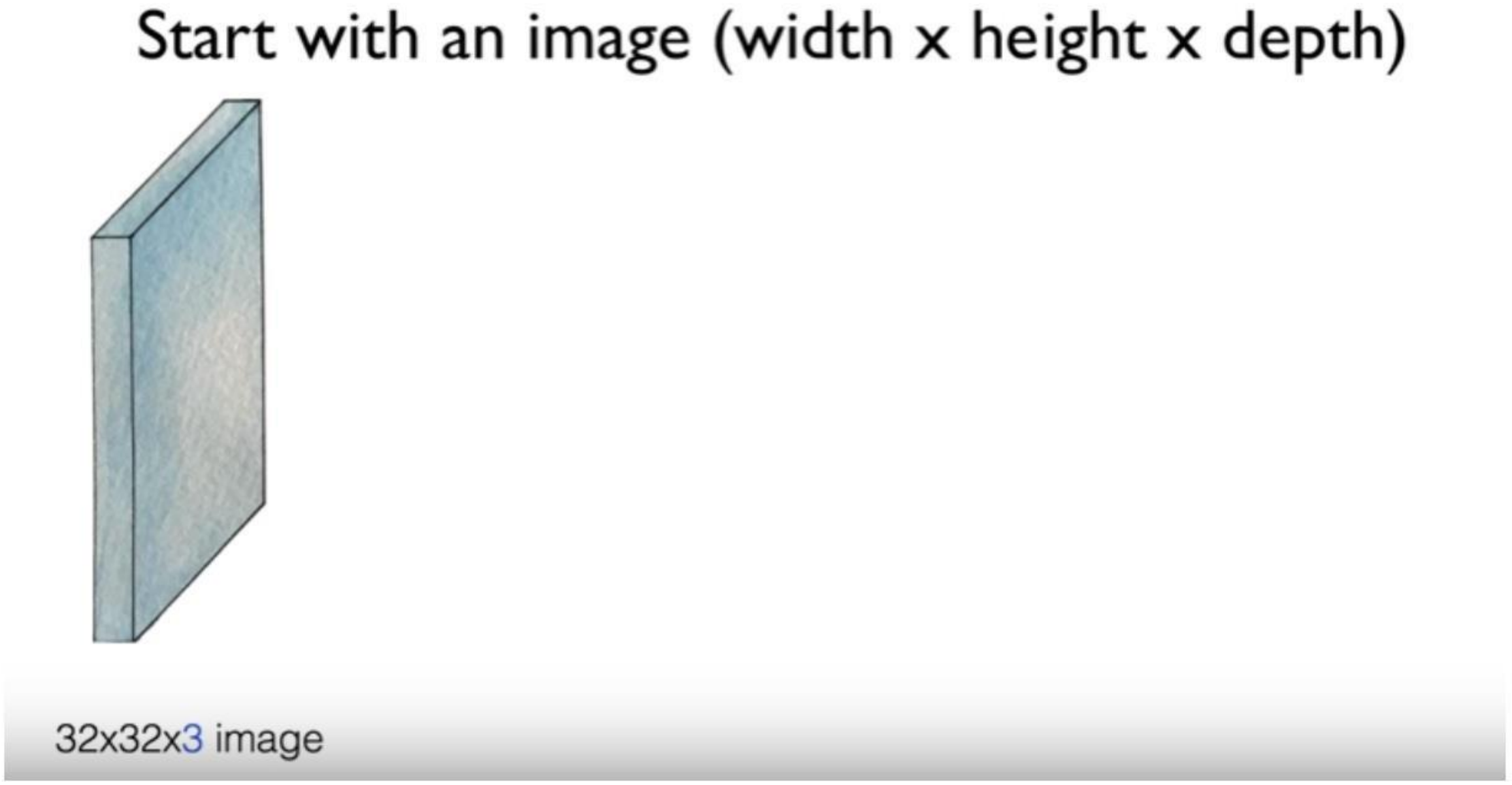




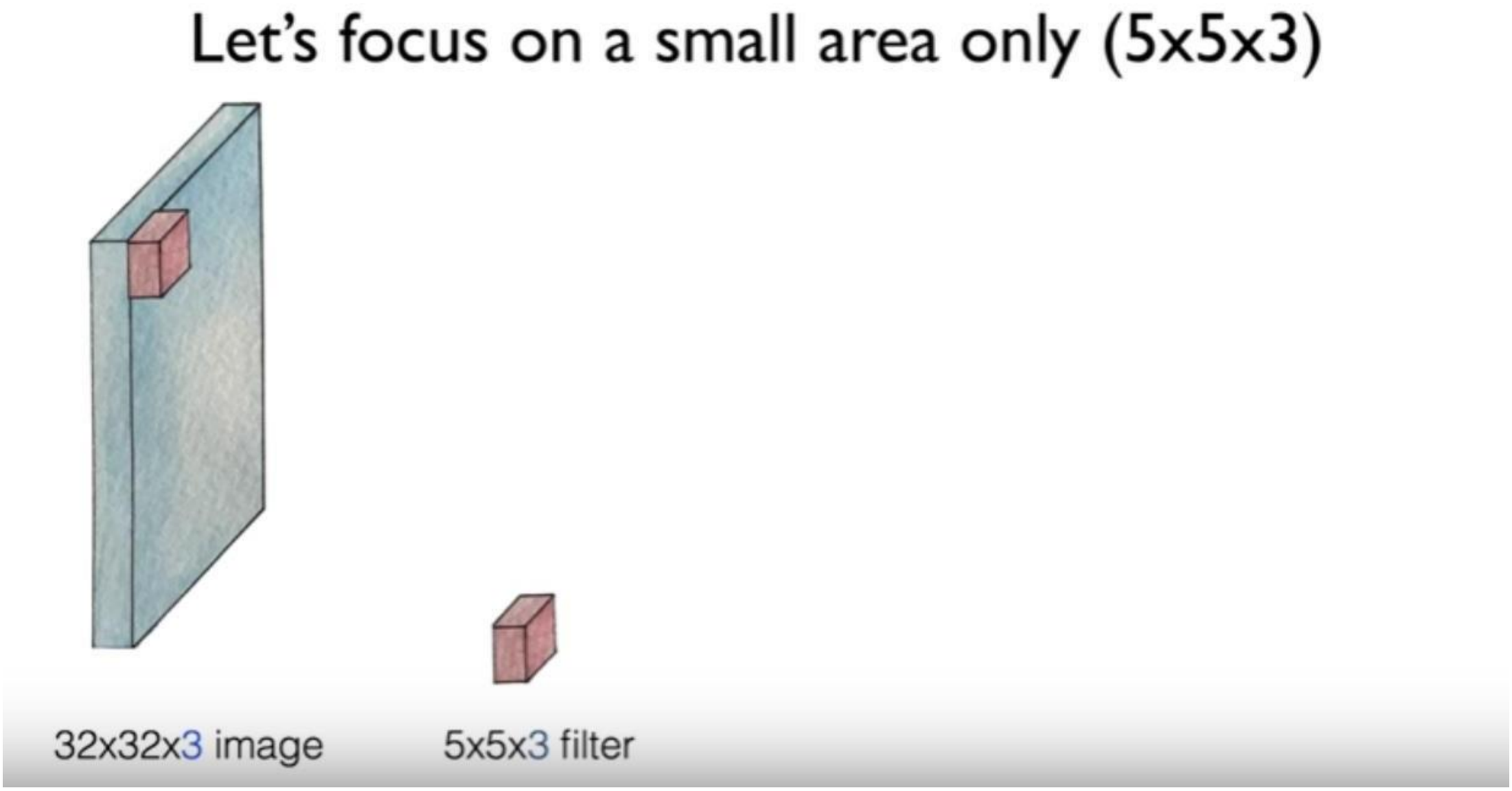
CONVOLUTIONAL NEURAL NETWORK



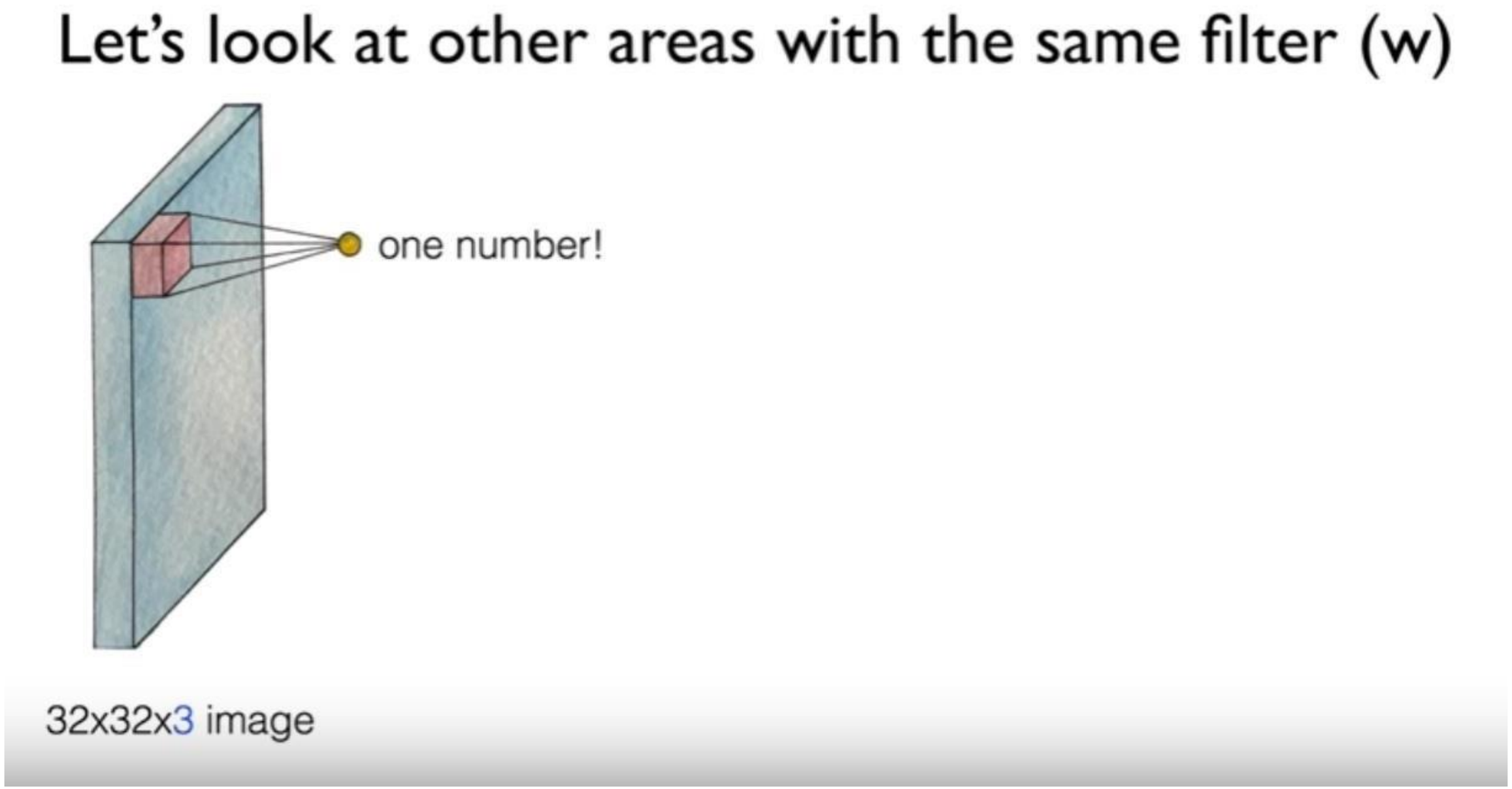
CNN FILTER



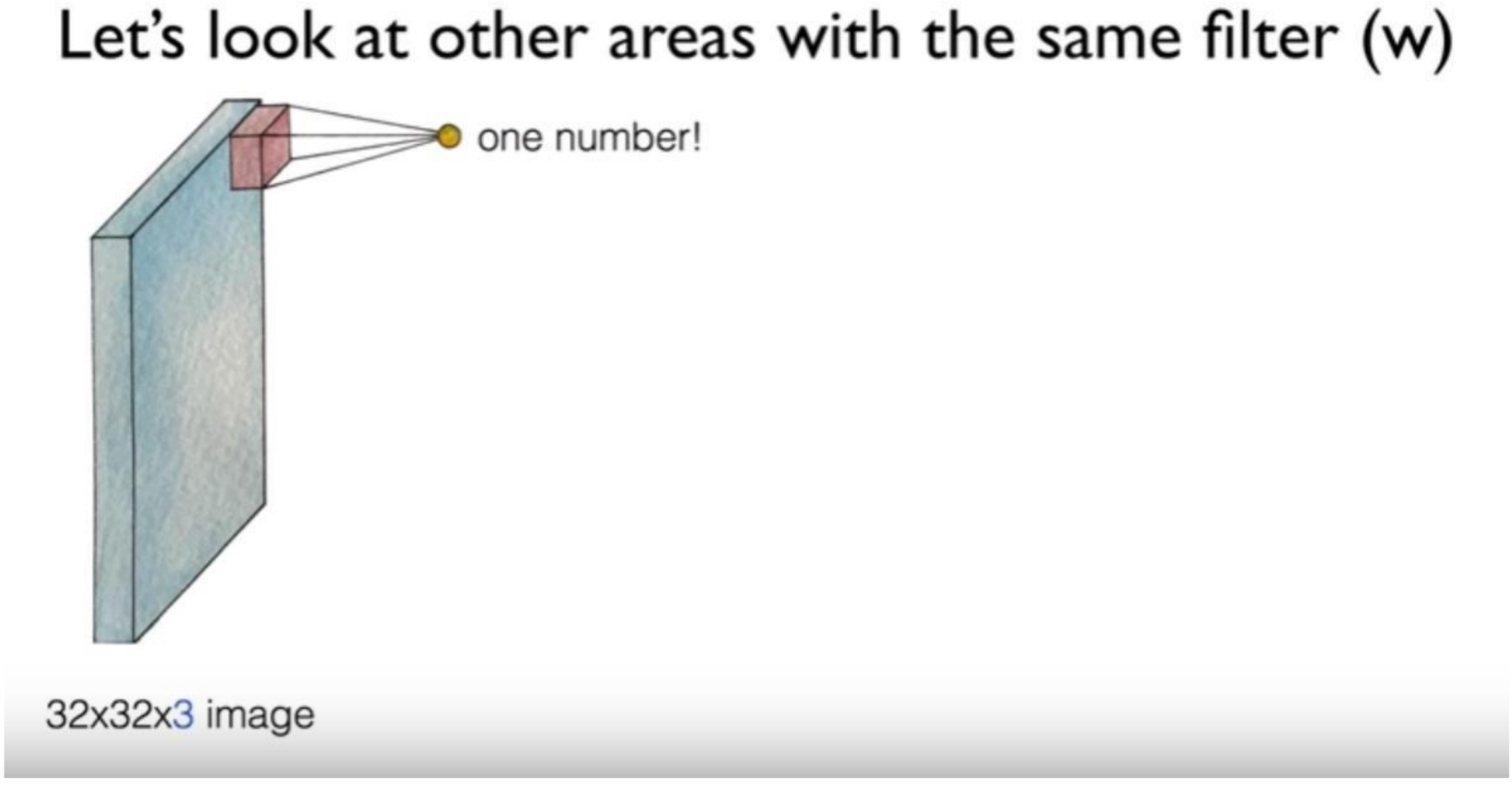
CNN FILTER


Let's focus on a small area only (5x5x3)
32x32x3 image
5x5x3 filter

CNN FILTER

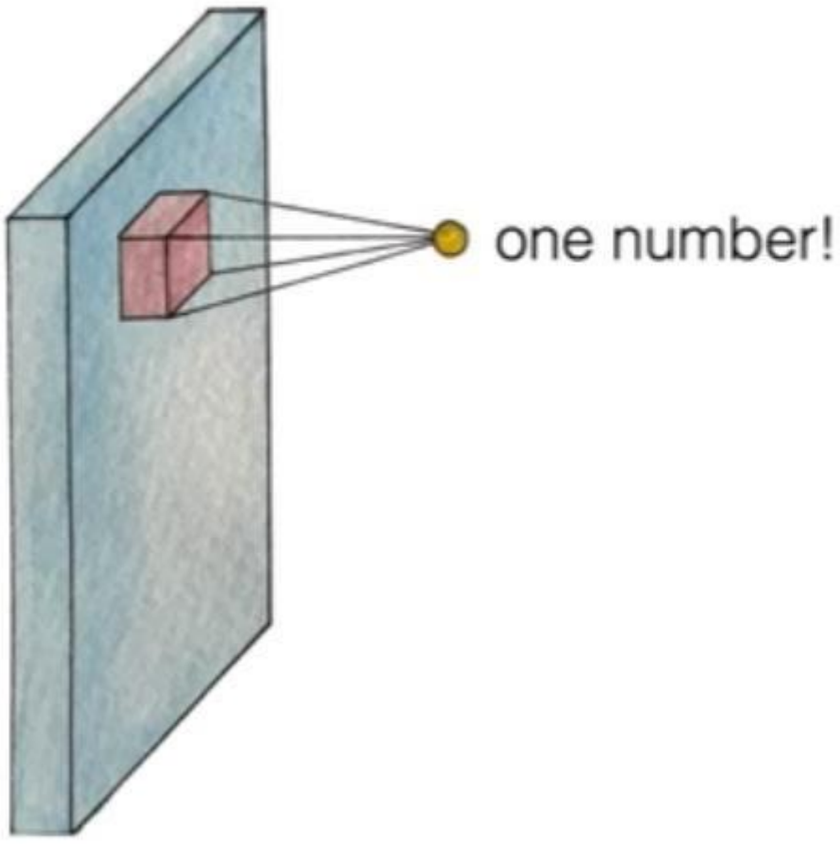

Let's look at other areas with the same filter (w)
one number!
32x32x3 image

CNN FILTER


Let's look at other areas with the same filter (w)
one number!
32x32x3 image

CNN FILTER

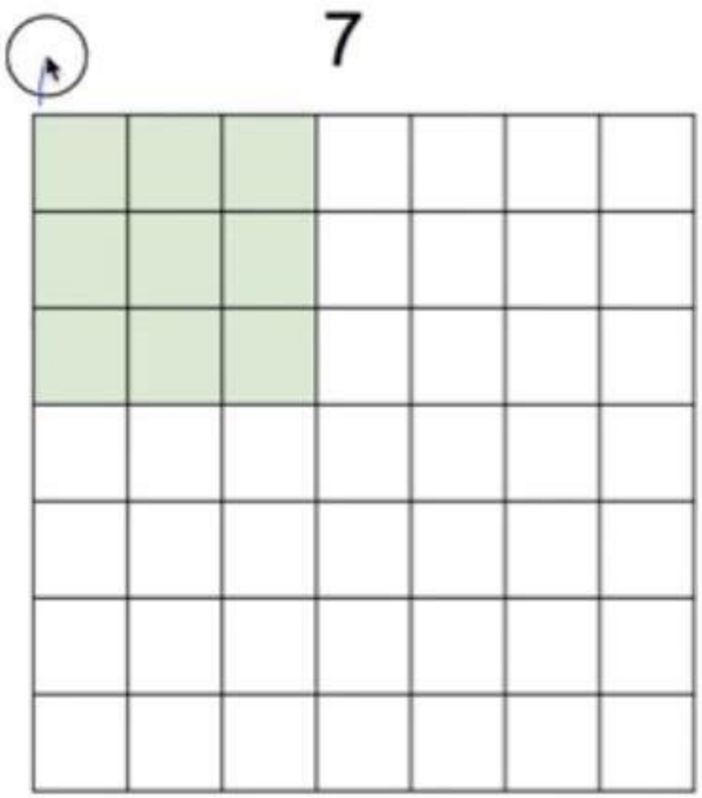
Let's look at other areas with the same filter (w)



How many numbers
can we get?

MOVING FILTER

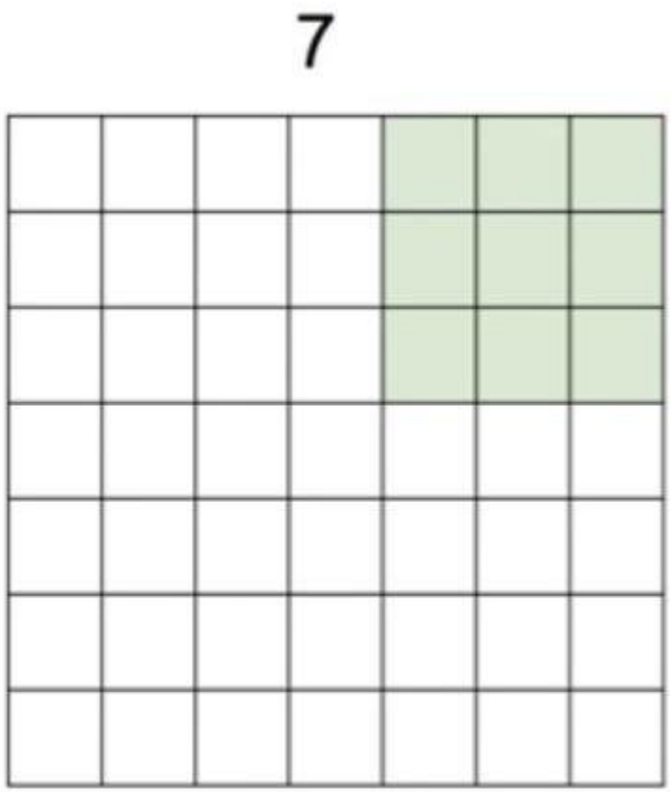
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

MOVING FILTER

A closer look at spatial dimensions:

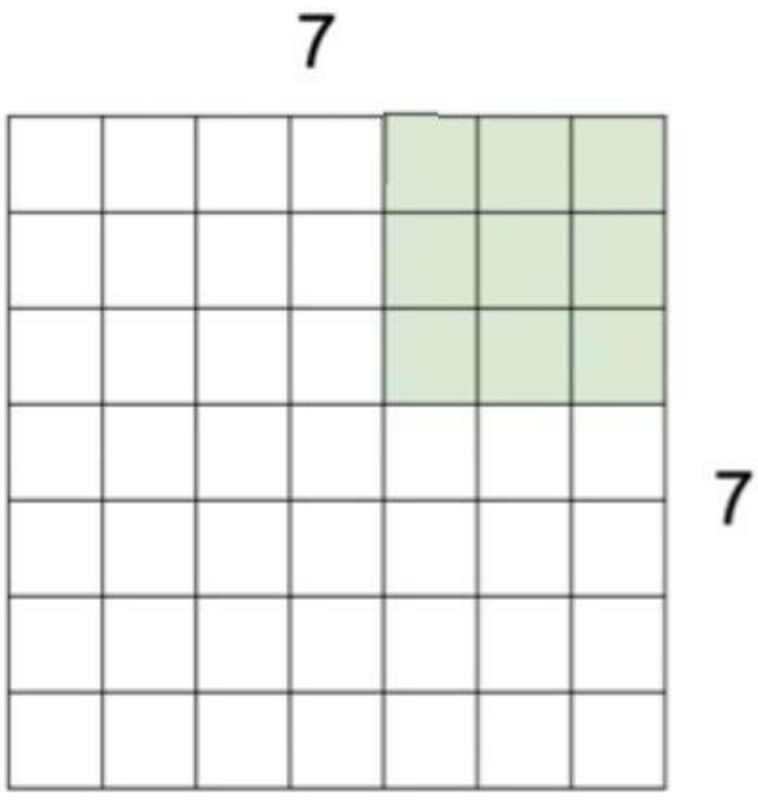


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

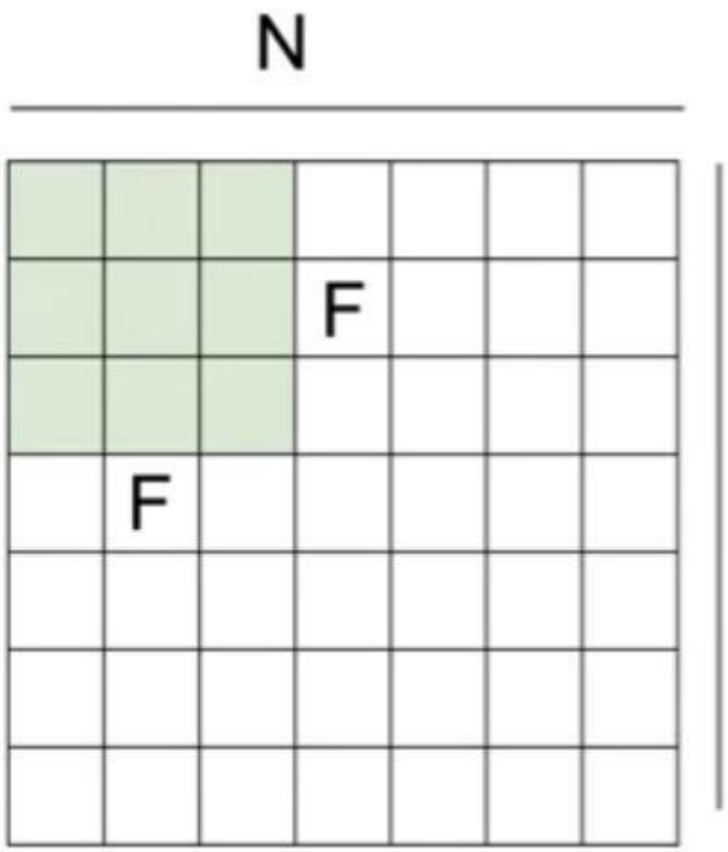
MOVING FILTER

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

MOVING FILTER

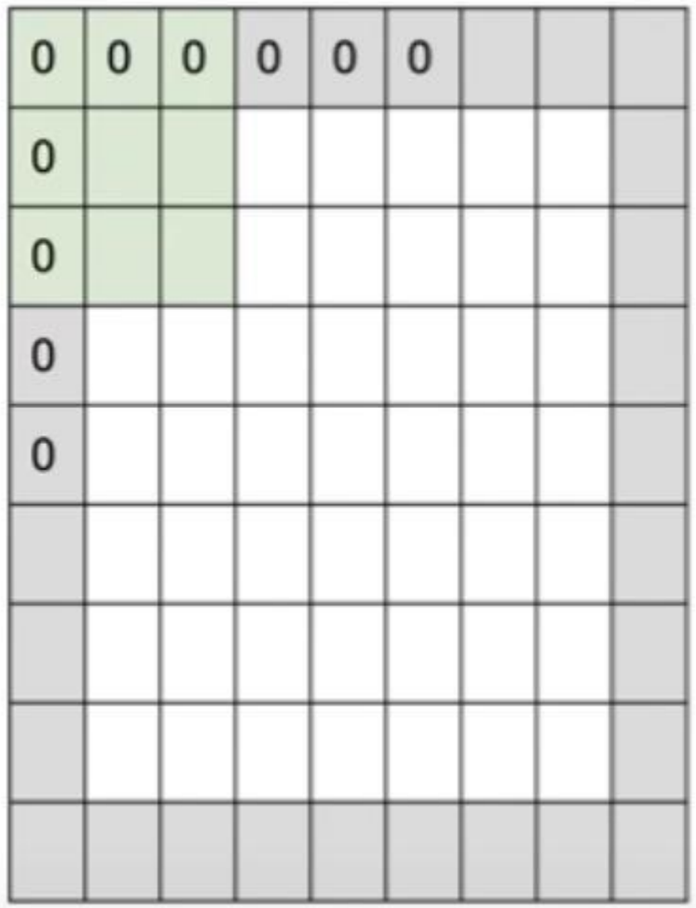


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

ZERO PADDINGS

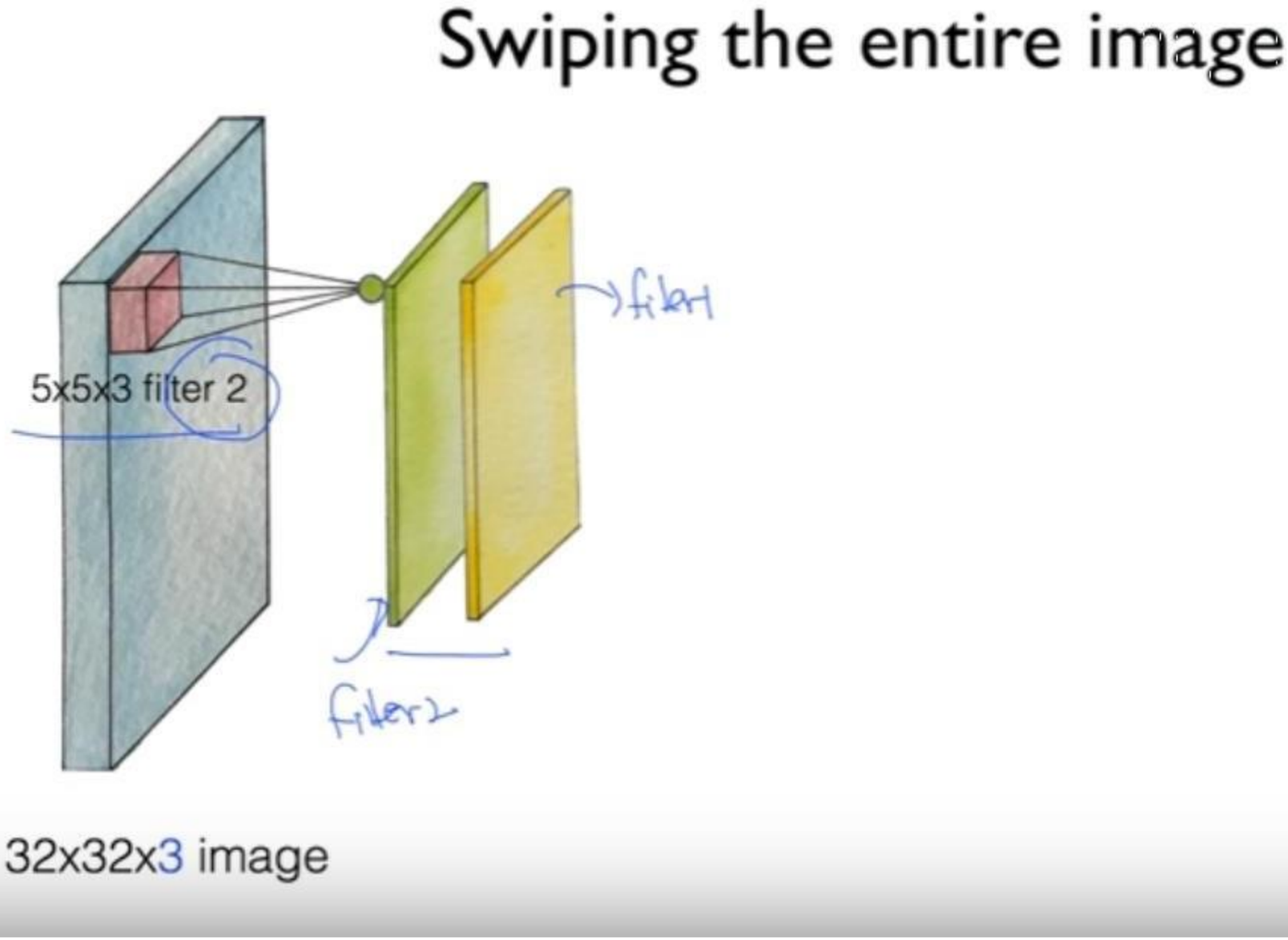
In practice: Common to zero pad the border



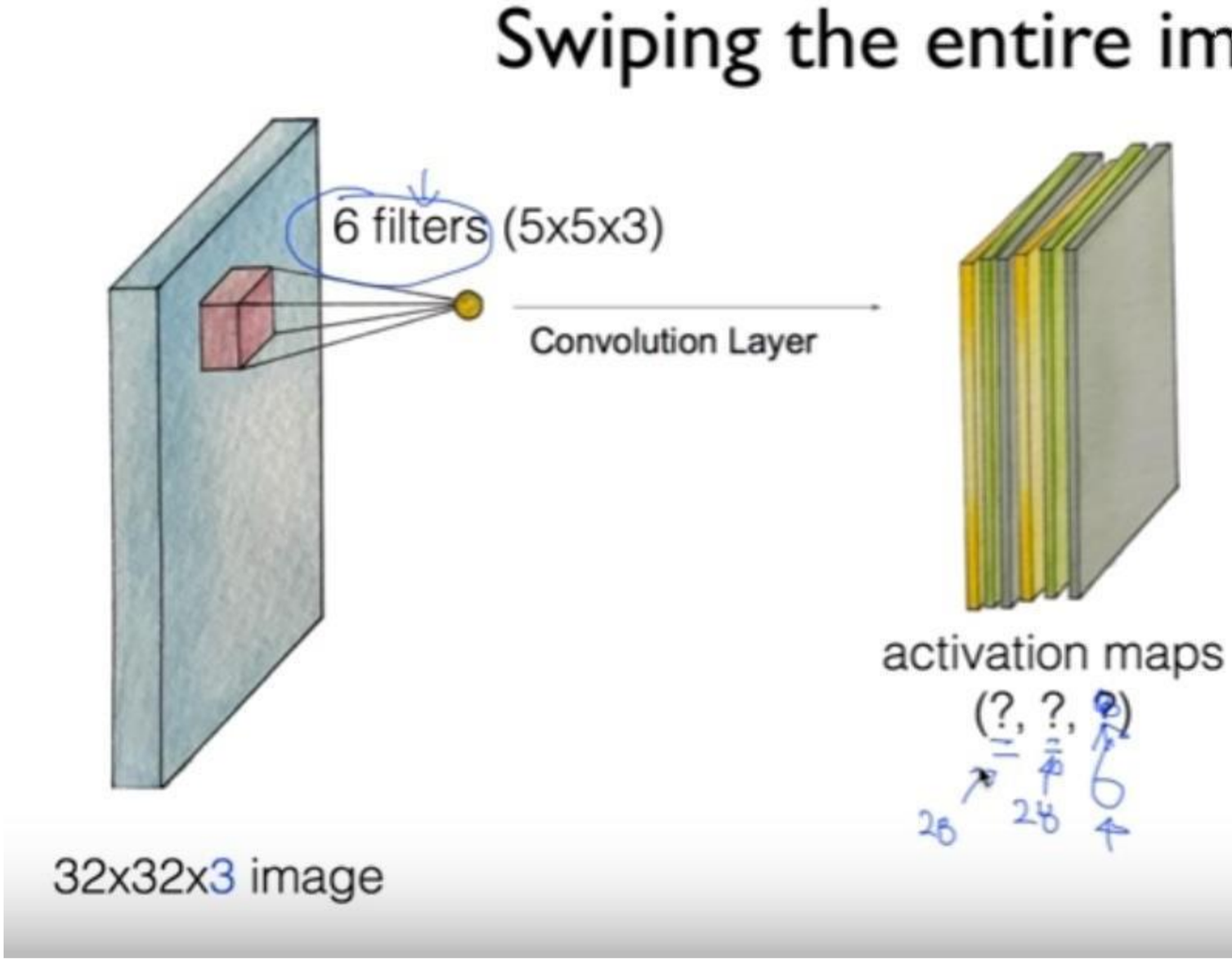
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

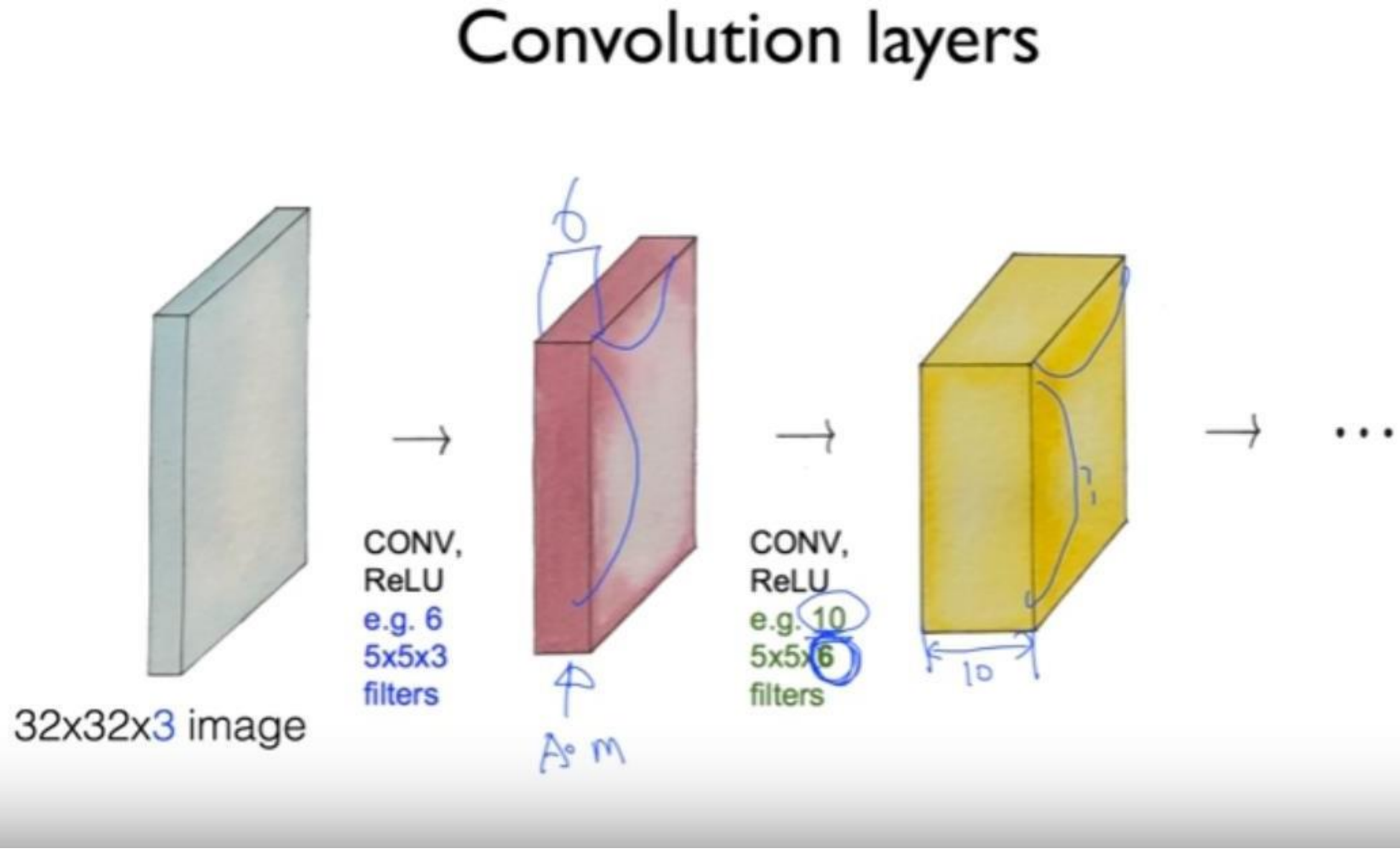
CONVOLUTIONAL LAYERS



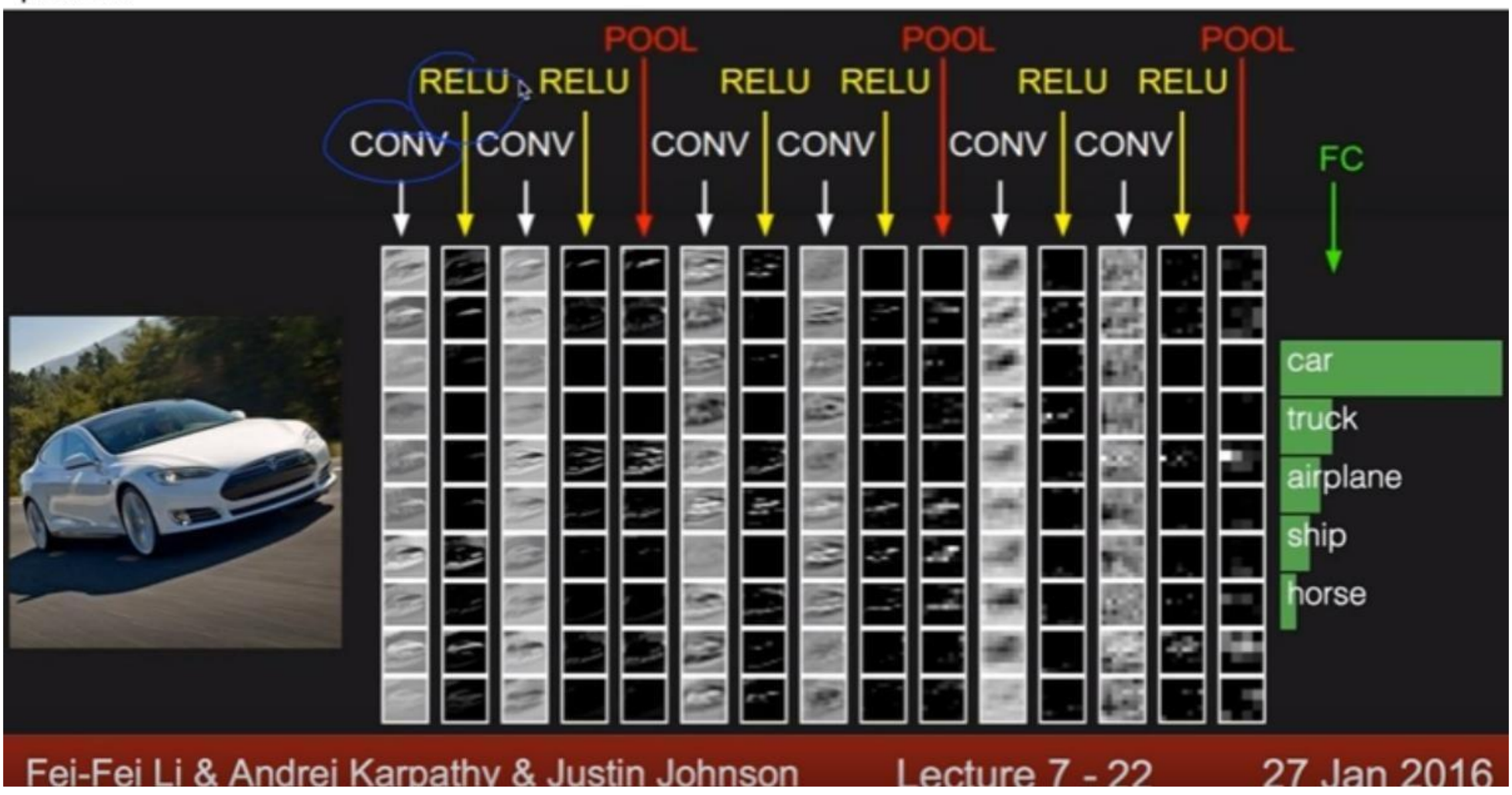
CONVOLUTIONAL LAYERS



CONVOLUTIONAL LAYERS



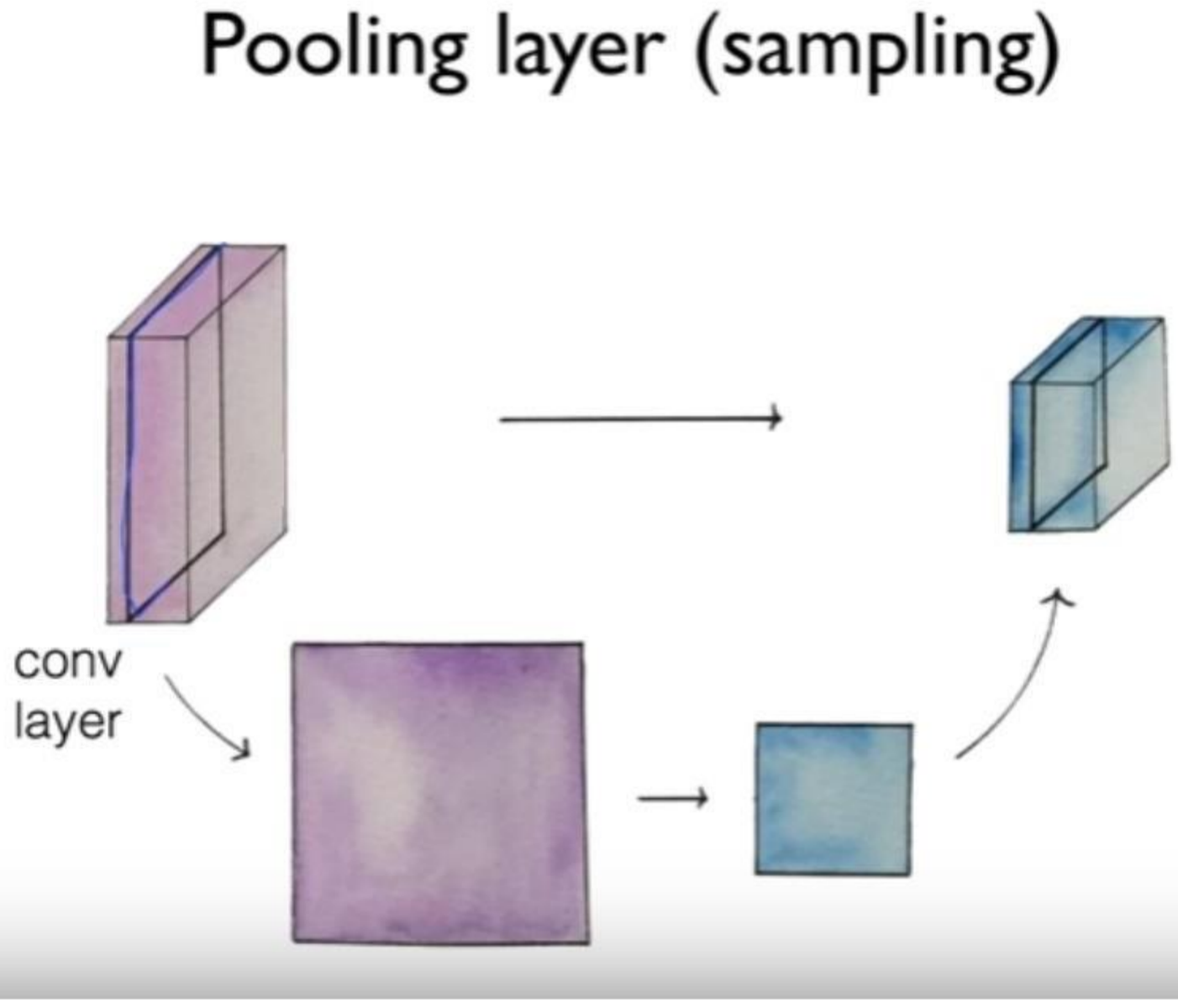
POOLING LAYER



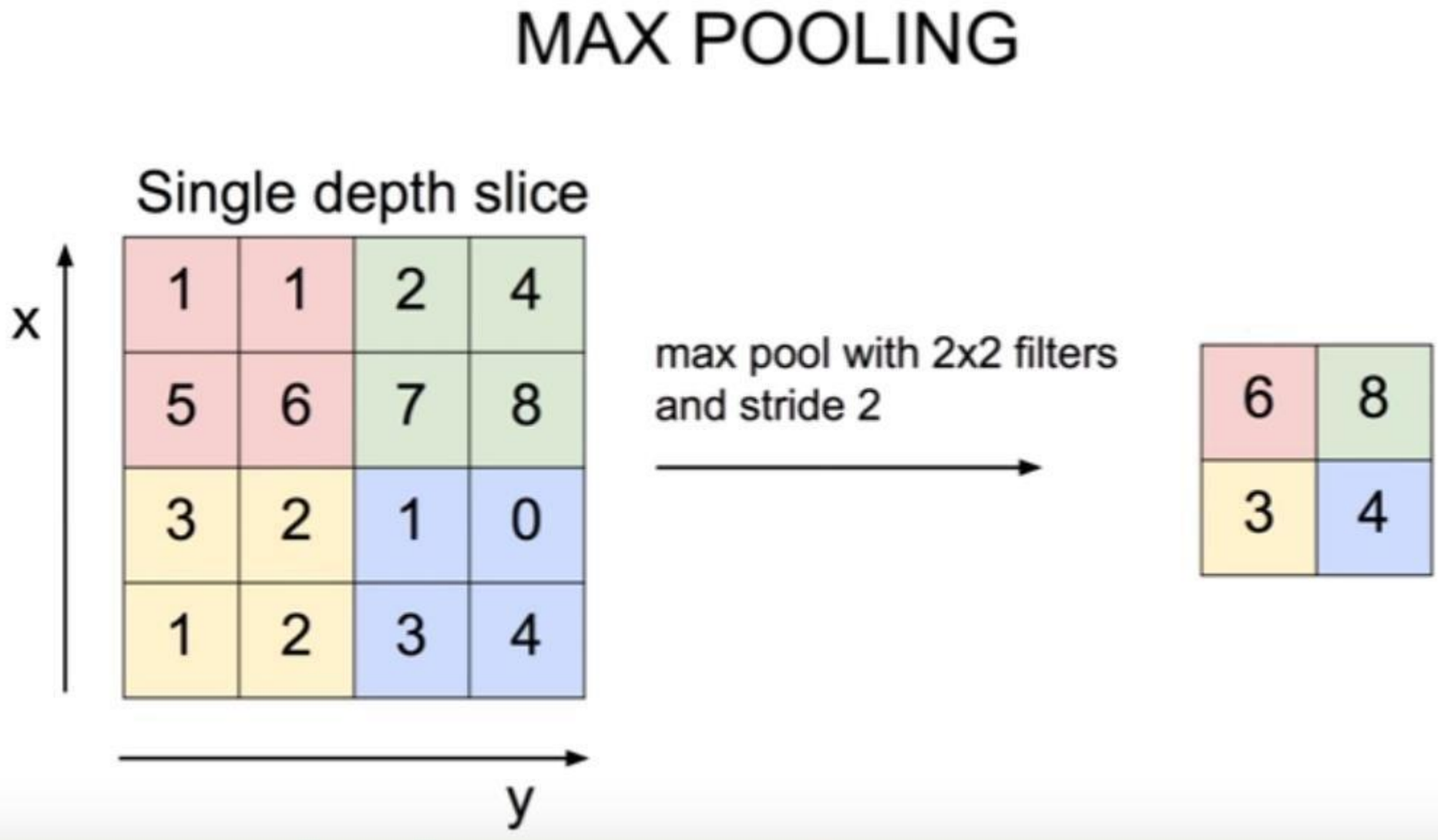
Input_size =28*28

Ouput_size =
[(input-filter_size)/stride]+1
=

POOLING LAYER

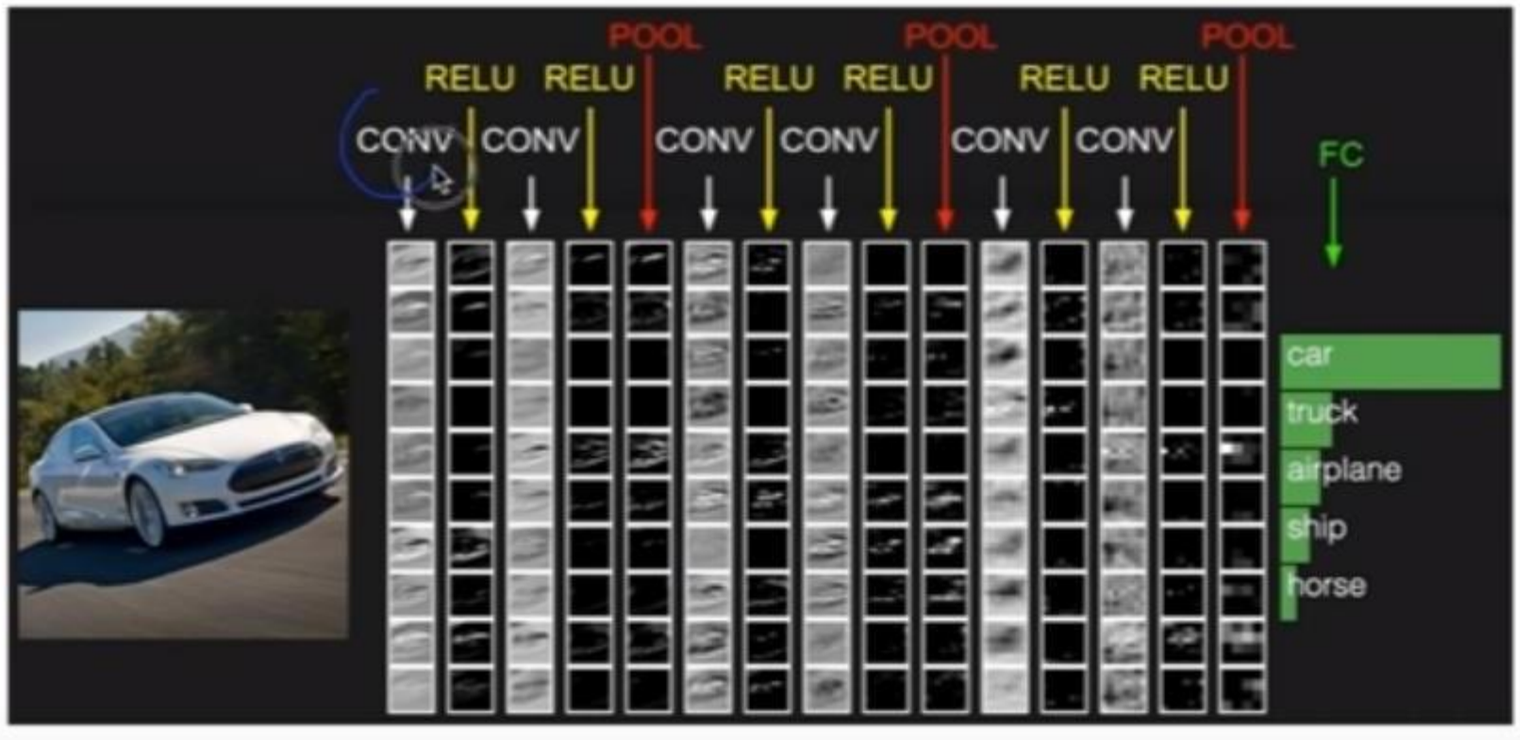


POOLING LAYER



FULLY CONNECTED LAYER

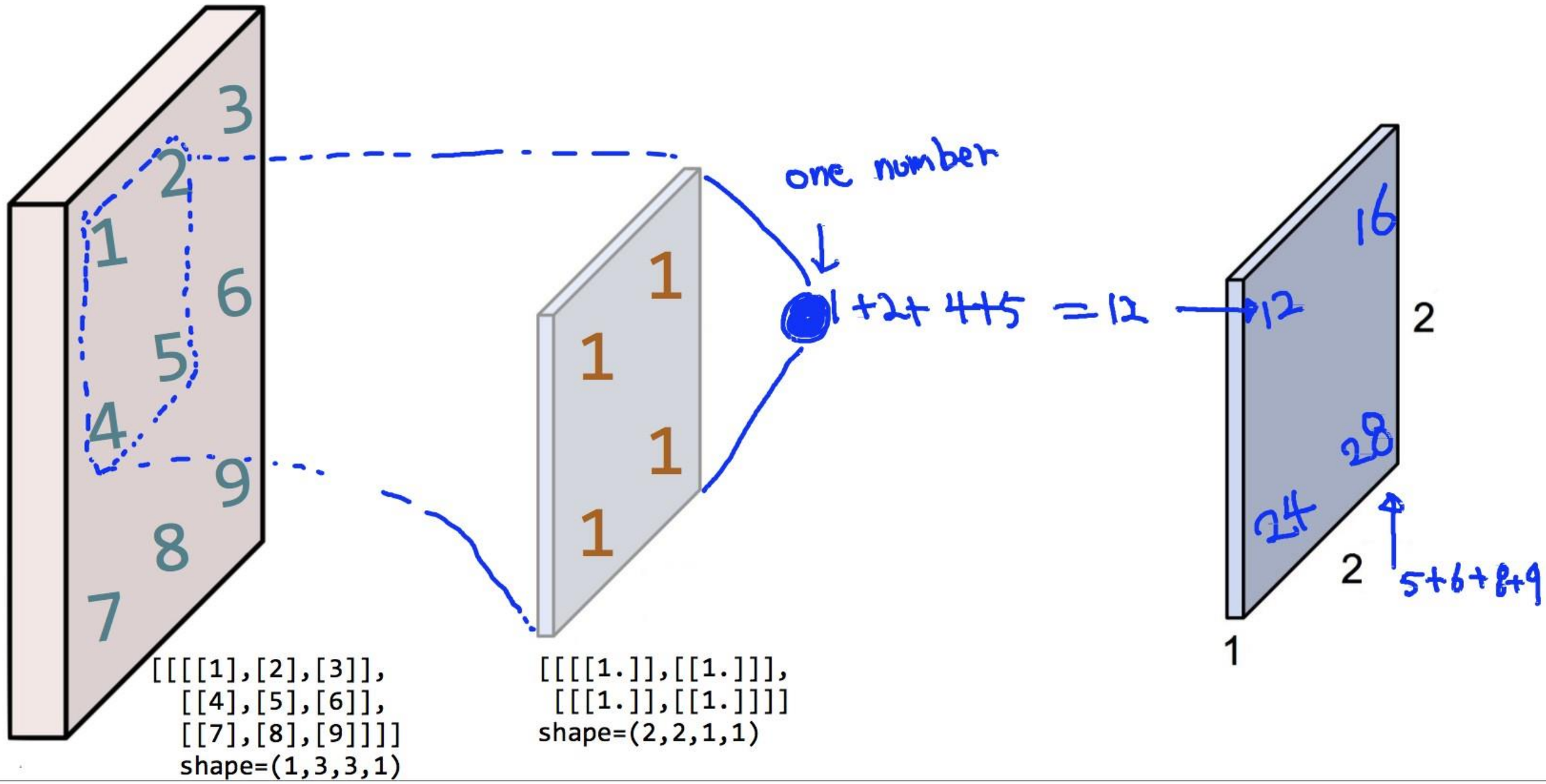
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

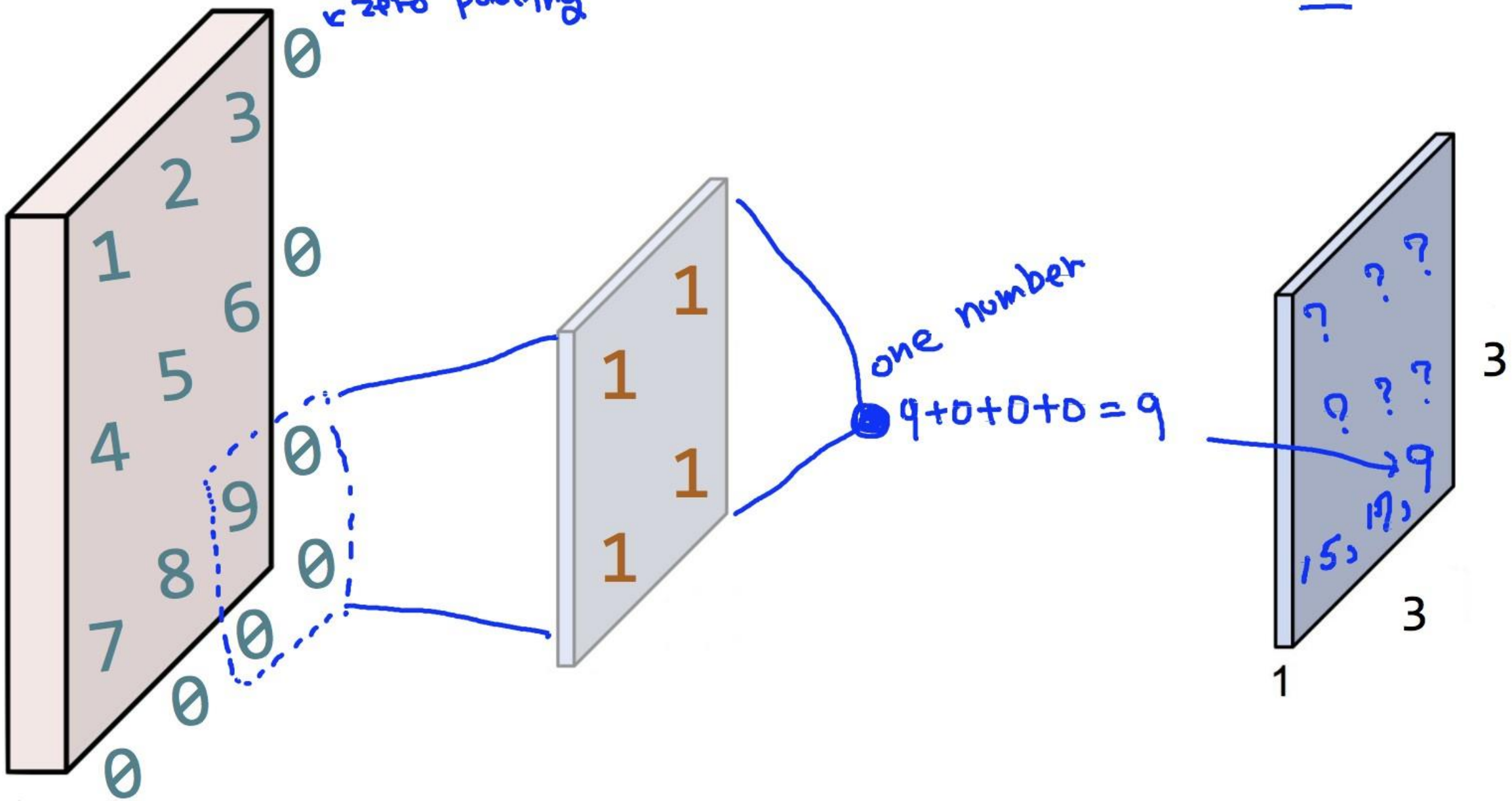
Simple convolution layer

Image: 1,3,3,1 image, Filter: 2,2,1,1, Stride: 1x1, Padding: VALID



Simple convolution layer

Image: 1,3,3,1 image, Filter: 2,2,1,1, Stride: 1x1, Padding: SAME



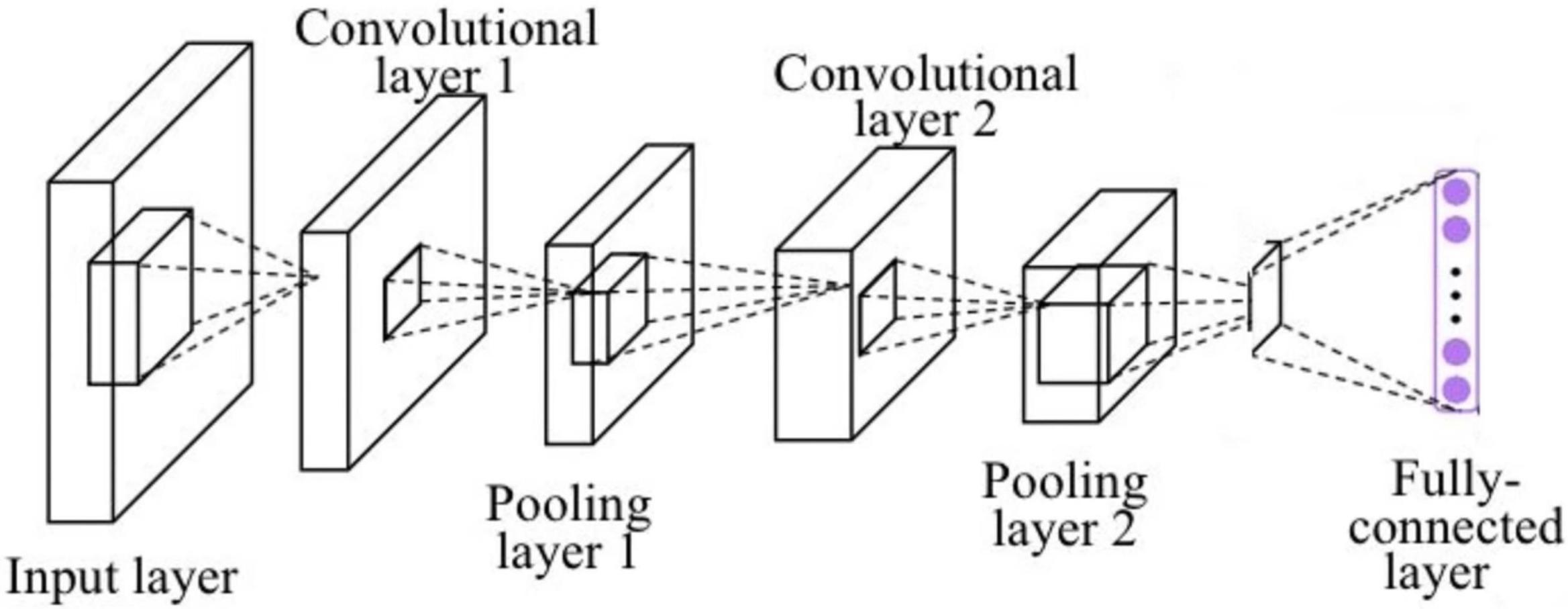
4	3	0
2	1	0
0	0	0

4	3	0
2	1	0
0	0	0

4	3	0
2	1	0
0	0	0

4	3	0
2	1	0
0	0	0

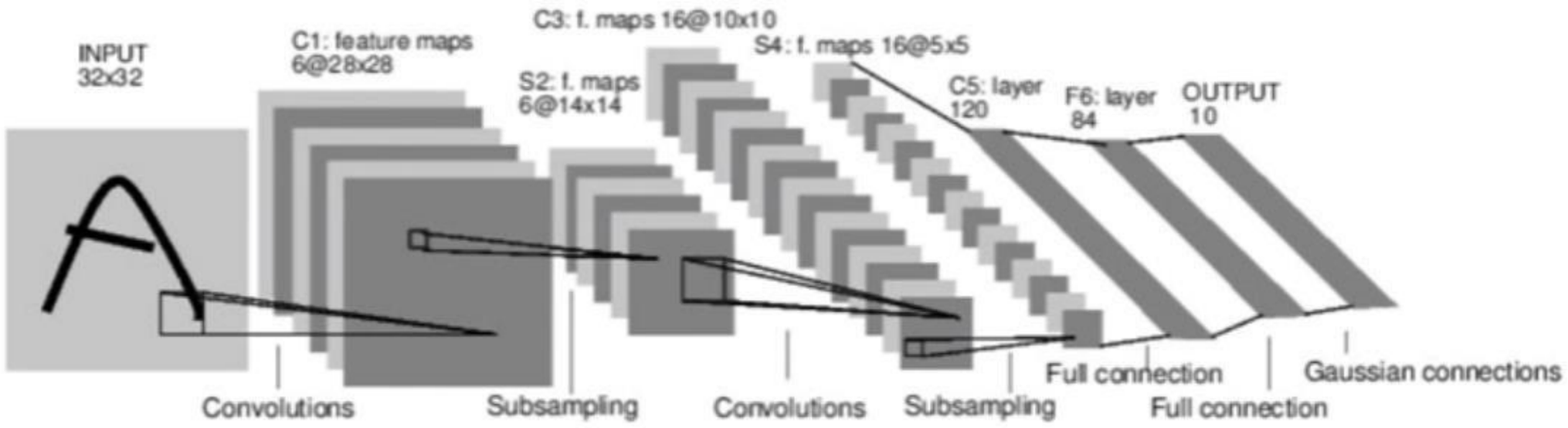
Simple CNN


Convolutional layer 1
Convolutional layer 2
Pooling layer 1
Pooling layer 2
Fully-connected layer
Input layer

CASE STUDY : LENET-5

Case Study: LeNet-5

[LeCun et al., 1998]

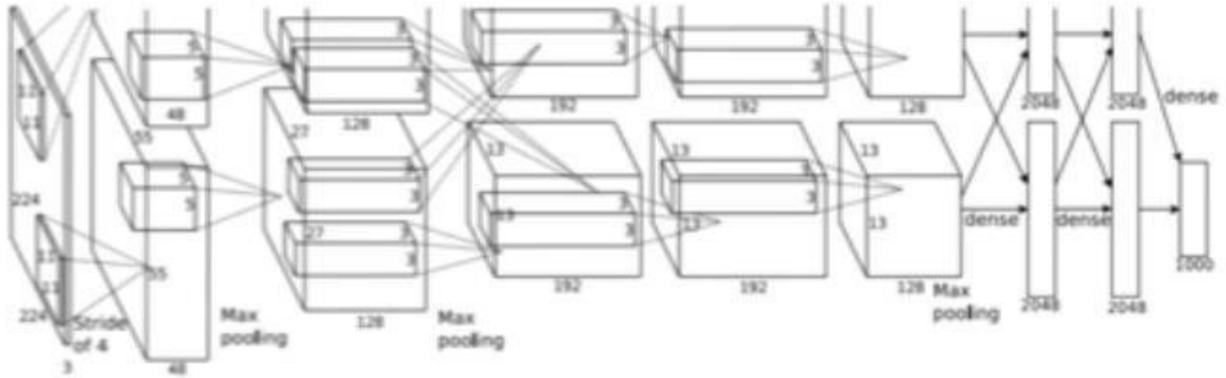


Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

CASE STUDY : ALEXNET

Case Study: AlexNet

[Krizhevsky et al. 2012]



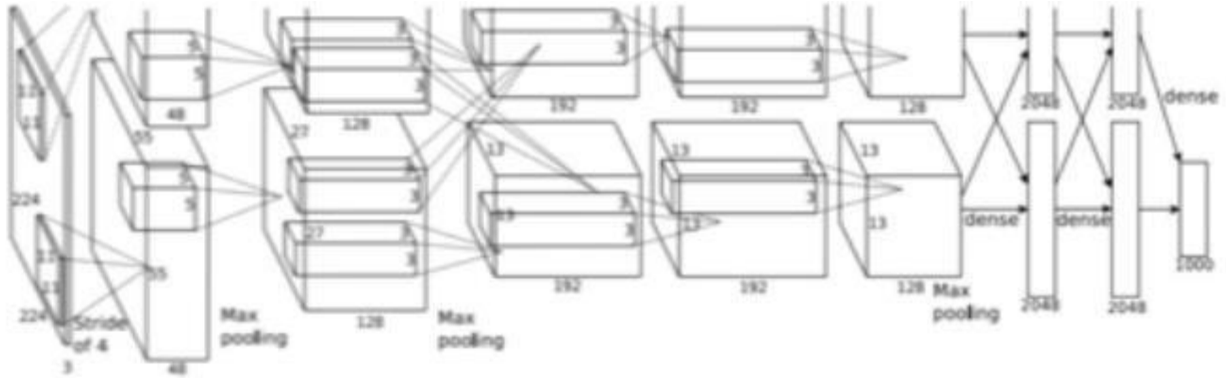
Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

CASE STUDY : ALEXNET

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!

딥러닝 기본 미니프로젝트

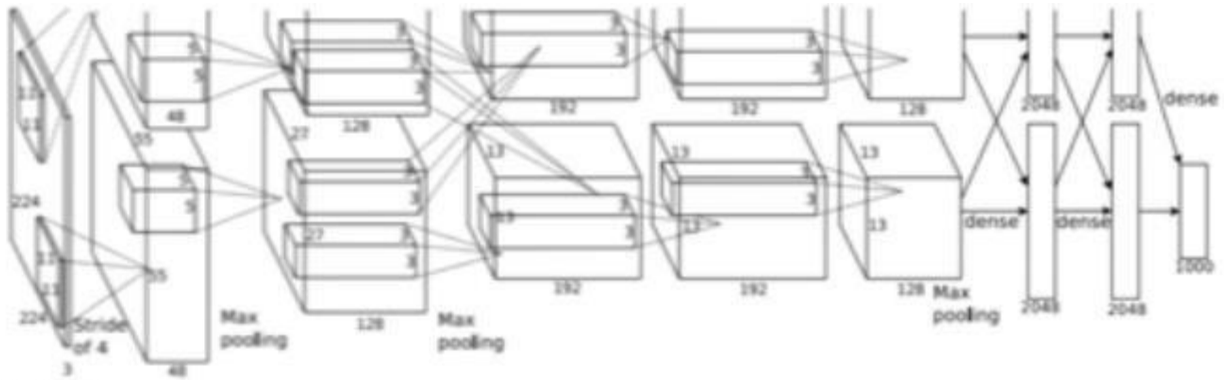
1.주제 정하기
2.데이터 수집(data.go.kr, 또는 크롤링 등)
3.학습
4.대시보드
- 데이터입력
- 예측값
- Heat map, confusion 등 그래프

16:30 분 발표 팀
단위 점수 부여

CASE STUDY : ALEXNET

Case Study: AlexNet

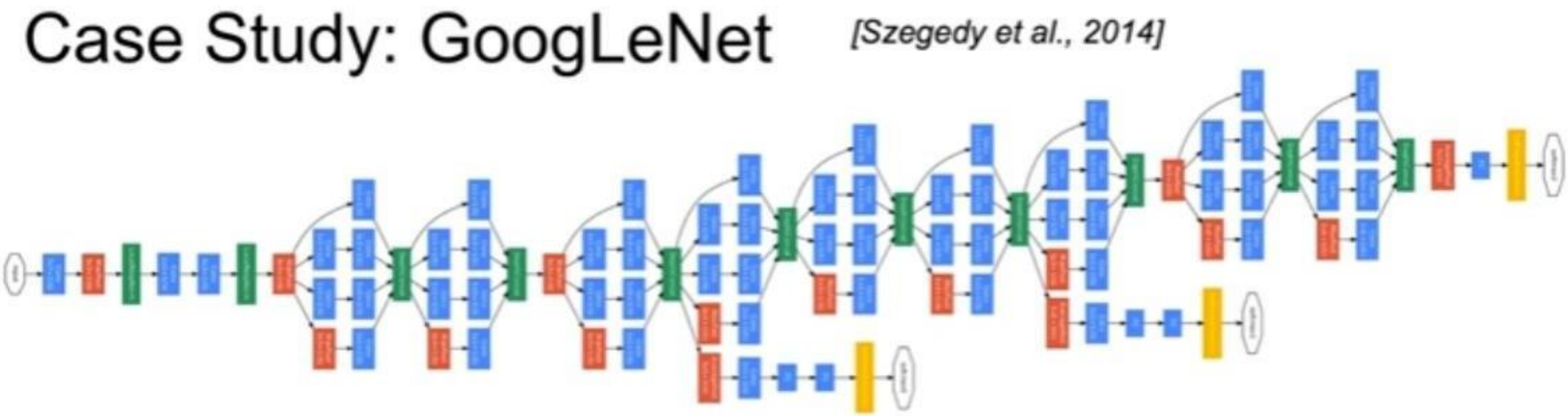
[Krizhevsky et al. 2012]

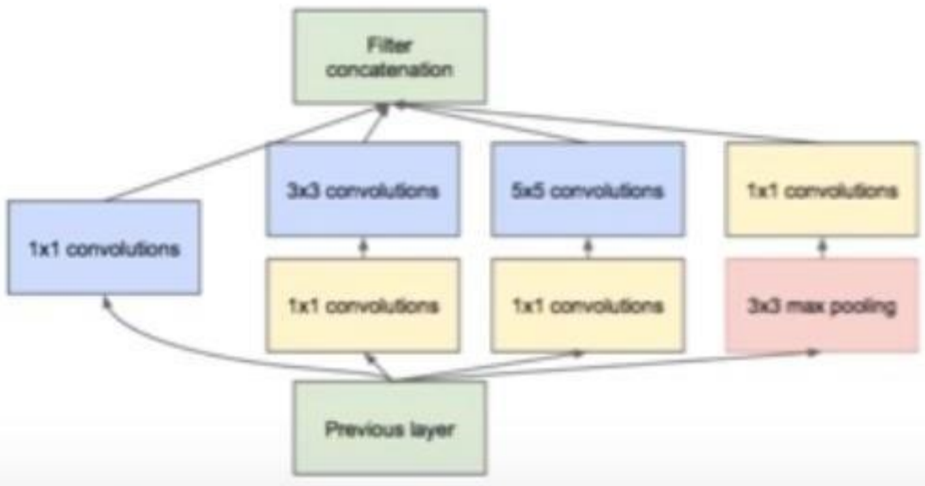


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

CASE STUDY : GOOGLENET





Inception module

ILSVRC 2014 winner (6.7% top 5 error)

CASE STUDY : RESNET

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)

Research

MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: ***"Ultra-deep"*** (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

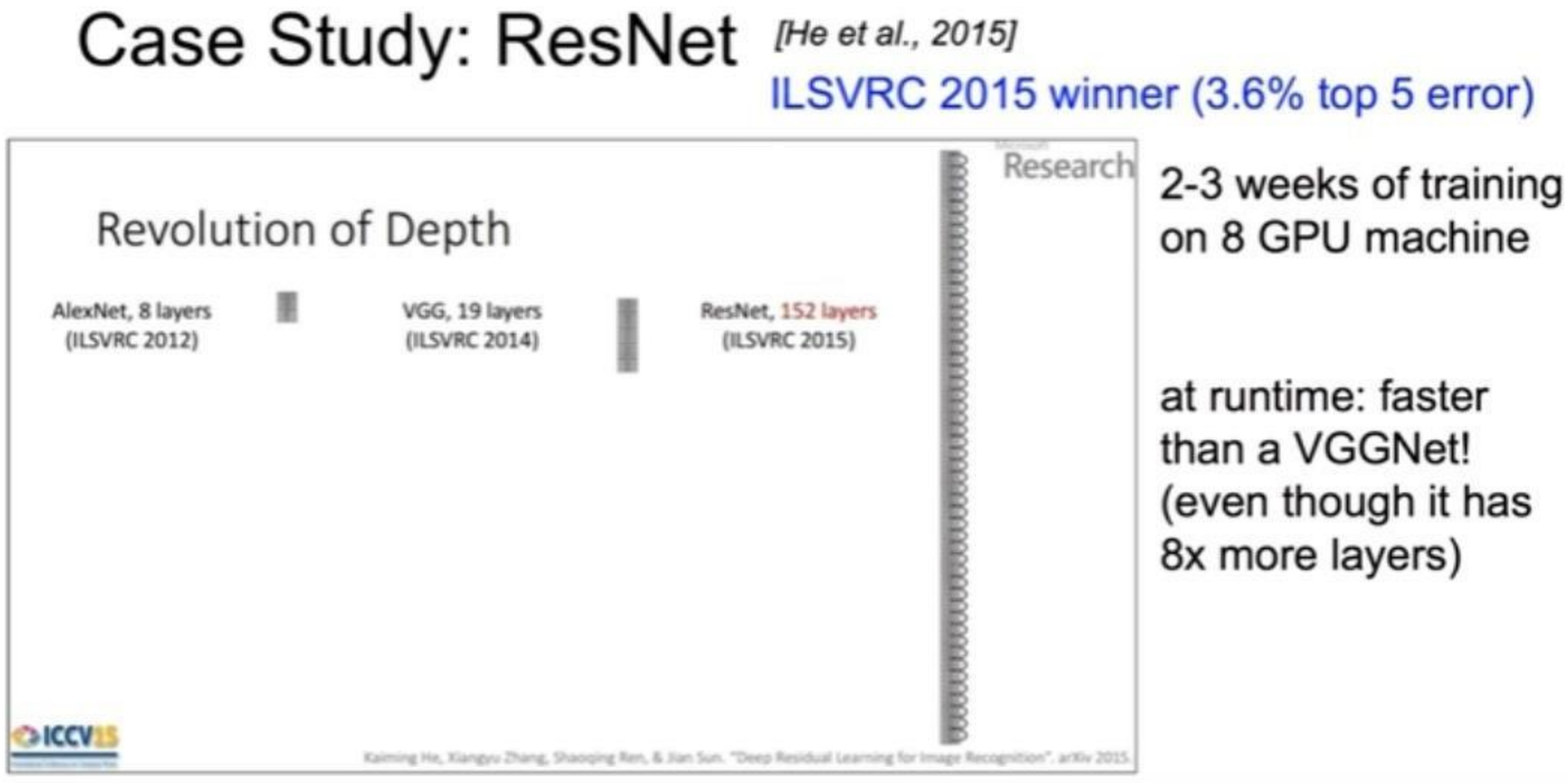
*improvements are relative numbers

ICCV15

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

CASE STUDY : RESNET

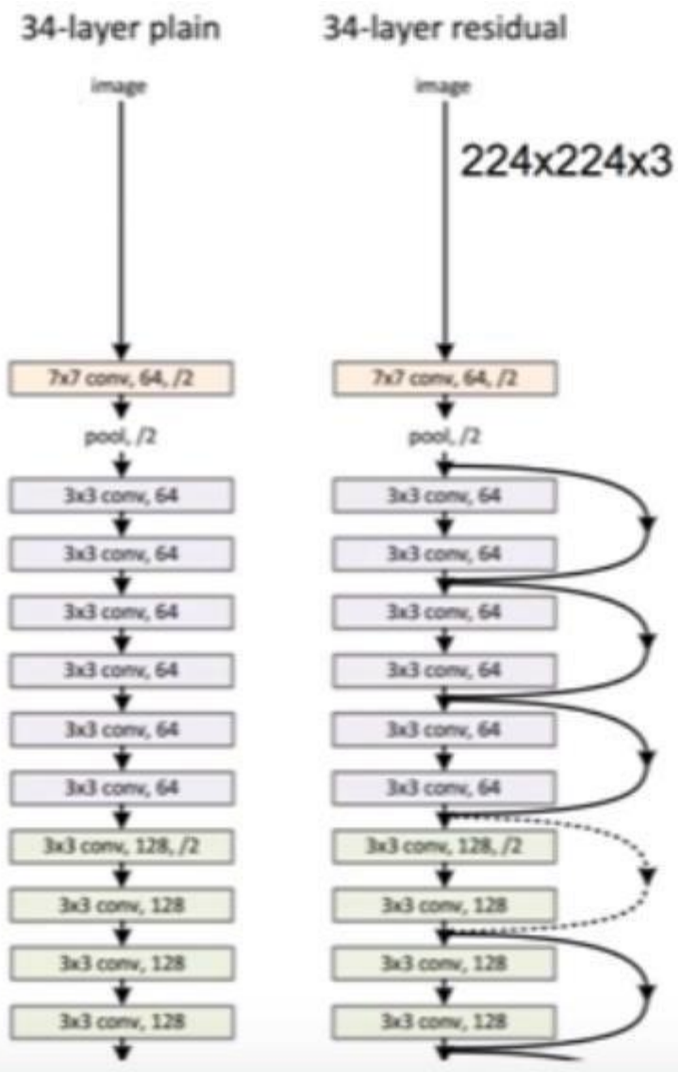


(slide from Kaiming He's recent presentation)

CASE STUDY : RESNET

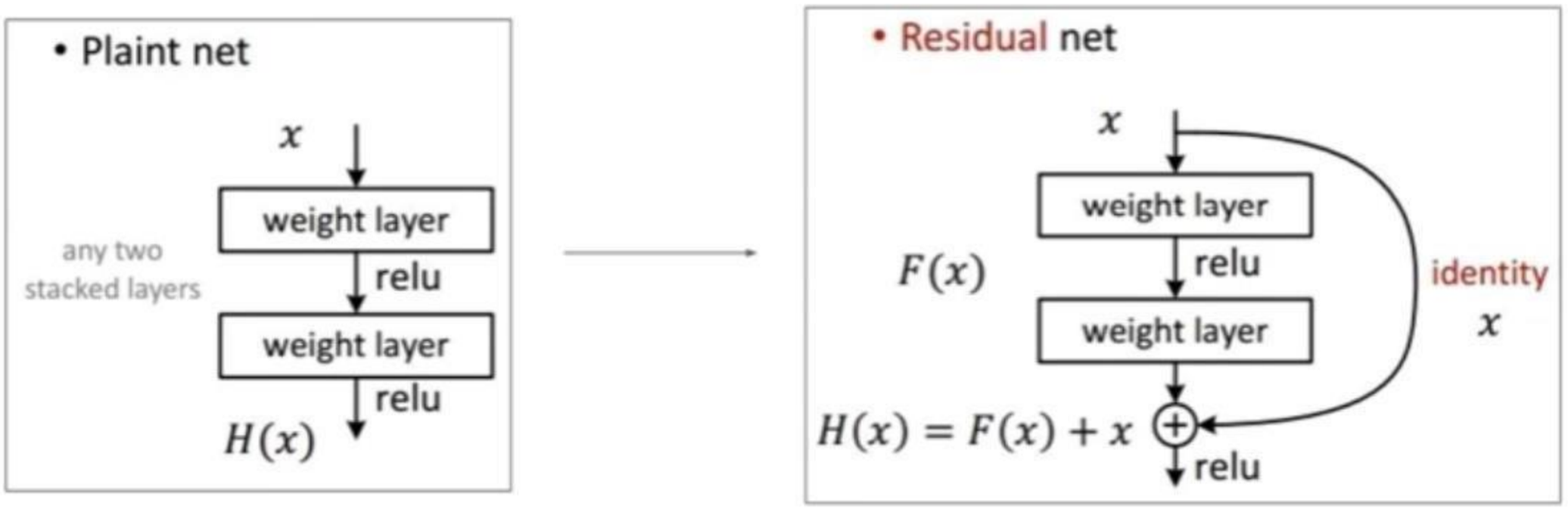
Case Study:
ResNet

[He et al., 2015]



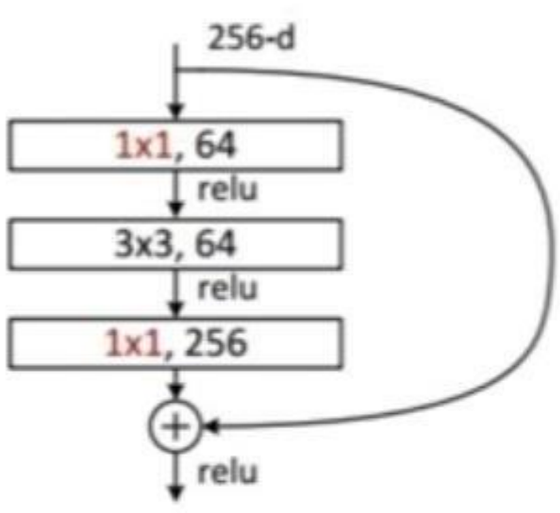
CASE STUDY : RESNET

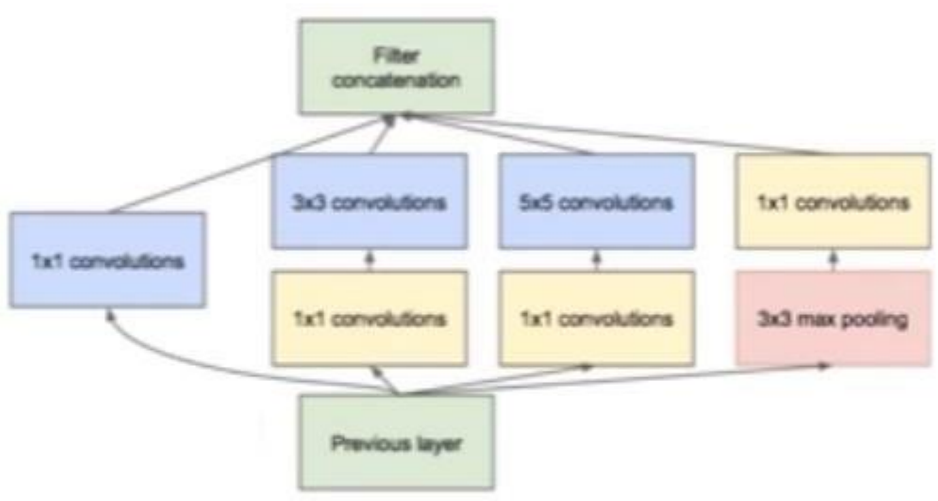
Case Study: ResNet *[He et al., 2015]*



CASE STUDY : RESNET

Case Study: ResNet *[He et al., 2015]*





CASE STUDY : SENTENCE CLASSIFICATIO

Convolutional Neural Networks for Sentence Classification

[Yoon Kim, 2014]

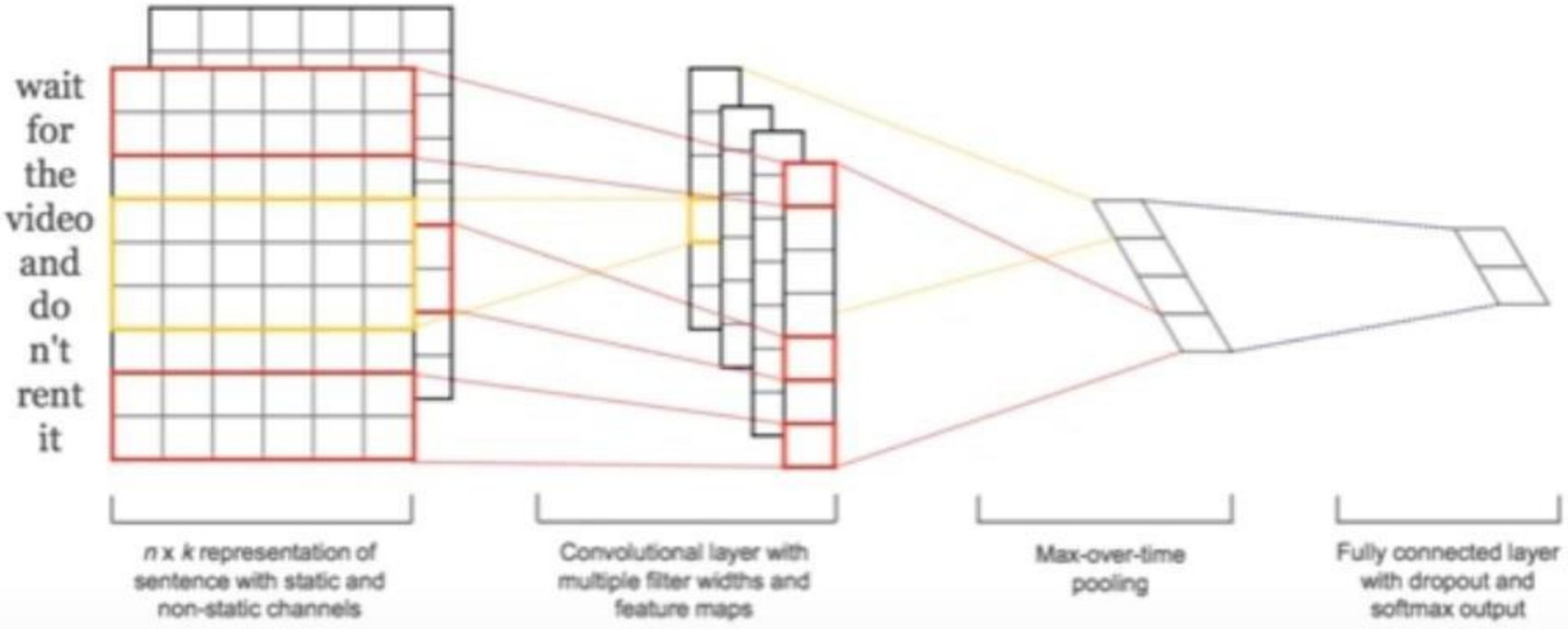


Figure 1: Model architecture with two channels for an example sentence.

➢ 학습 데이터 부족

✓ CNN 모델의 품질을 높이기 위해서는, 즉 임의의 데이터에 대해서 정확도는 높이고 오버피팅은 줄이기 위해서는 기본적으로 많은 양의 데이터를 이용하여 학습하여야 함

✓ 그러나 많은 학습 데이터를 확보하려면 많은 비용과 시간이 소요되기 때문에 현실적으로 쉽지 않은데, 이러한 데이터가 부족한 어려움을 해결하기 위해 등장한 것이 전이학습임

➢ 전이 학습 개념 및 필요성

✓ 전이 학습(Transfer Learning)이란 아주 큰 데이터 셋, 즉 21,841 부류에 대해서 총 1419만 7122장의 이미지로 구성되어 있는 ImageNet 데이터를 사용해서 학습된 모델의 가중치를 가져와서, 우리가 해결하려는 문제에 맞게 보정해서 사용하는 것을 의미함

✓ 이때 큰 데이터 셋을 사용해서 훈련된 모델을 사전 학습 모델(pre-trained model)이라고 함

※ ImageNet 데이터의 이미지 크기는 평균적으로 469x387 이며, 이러한 2만개 이상의 부류 가운데 1000 부류만 뽑아서 데이터를 구성하고, 정확도를 높이는 대회가 바로 ImageNet Challenge 임

풍부한 데이터(ImageNet)

데이터 부족 해결

부족한 사용자 데이터

사전 학습 모델
(pre-trained model)

전이 학습
(Transfer Learning)

새롭게
학습되는 모델

➢ 사전 학습된 특징 추출기

- 특징 추출기(feature extractor)는 컨볼루션 층과 풀링 층의 조합으로 구성되어 있으며 ImageNet 데이터에 대해 이미 학습되어 있음

※ 특징 추출기의 출력 데이터를 bottleneck 또는 feature vector 등으로 지칭함

➢ 사전 학습된 분류기

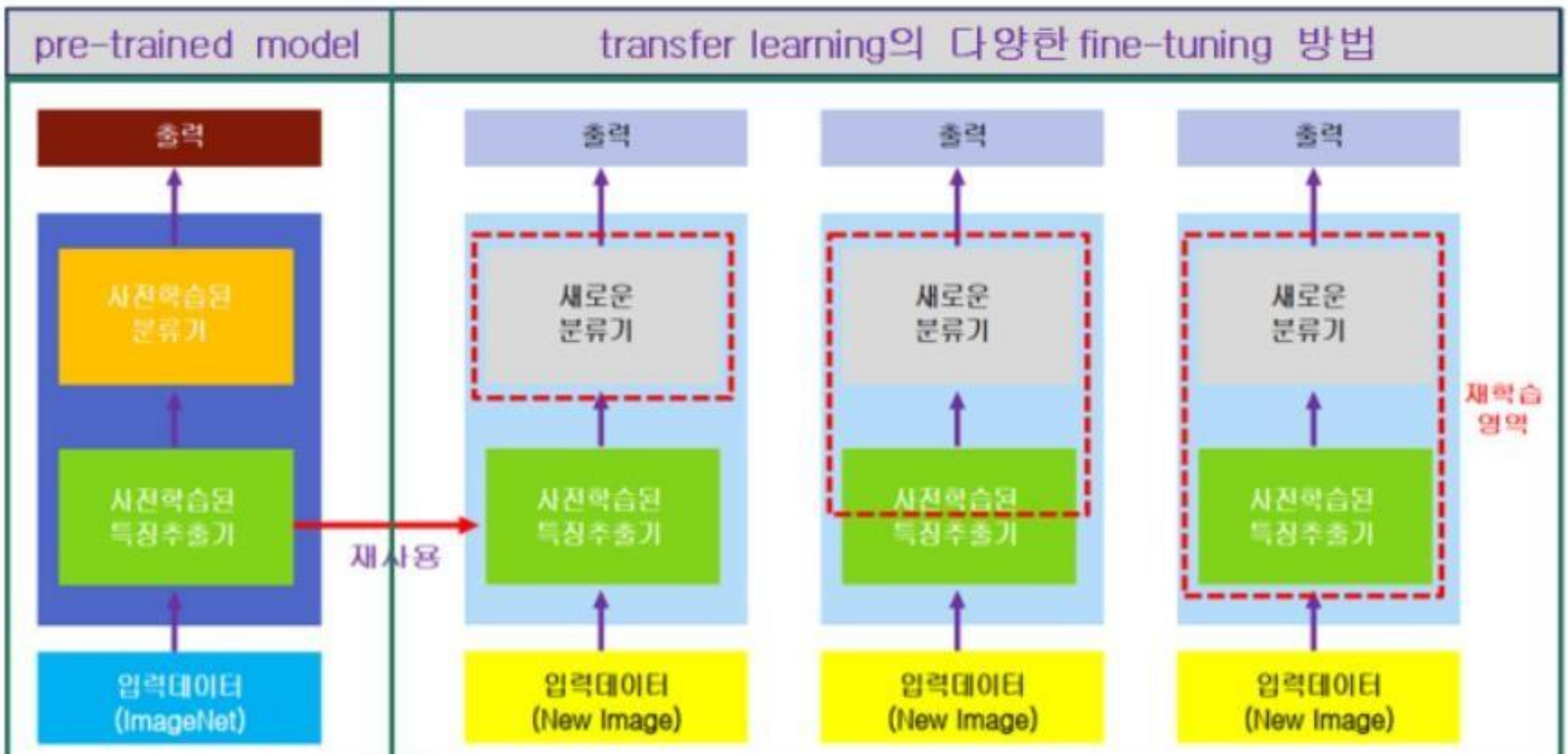
- 분류기(classifier)는 완전 연결 층으로 구성되며 추출된 특징을 입력으로 받아 최종적으로 주어진 이미지에 대한 클래스(정답)을 카테고리 형태로 분류하는 역할 수행

※ 오버피팅을 줄이기 위해 출력 층 이전의 Dense layer 사이에는 Dropout , BatchNormalization layer 등을 add 할수 있음

파인 튜닝 (fine-tuning)

- 사전 학습 모델의 가중치를 미세하게 조정하는 기법이며, 새롭게 분류하려는 데이터의 종류와 전체 개수를 미리 분석한 후에, 그것을 바탕으로 사전 학습 모델 가중치 일부만을 재학습 시키거나 또는 모든 가중치를 처음부터 다시 학습시킬 수 있음

  - ✓ 파인 튜닝 진행 시 많은 연산량이 필요하므로 일반적으로 CPU보다는 GPU를 많이 사용함



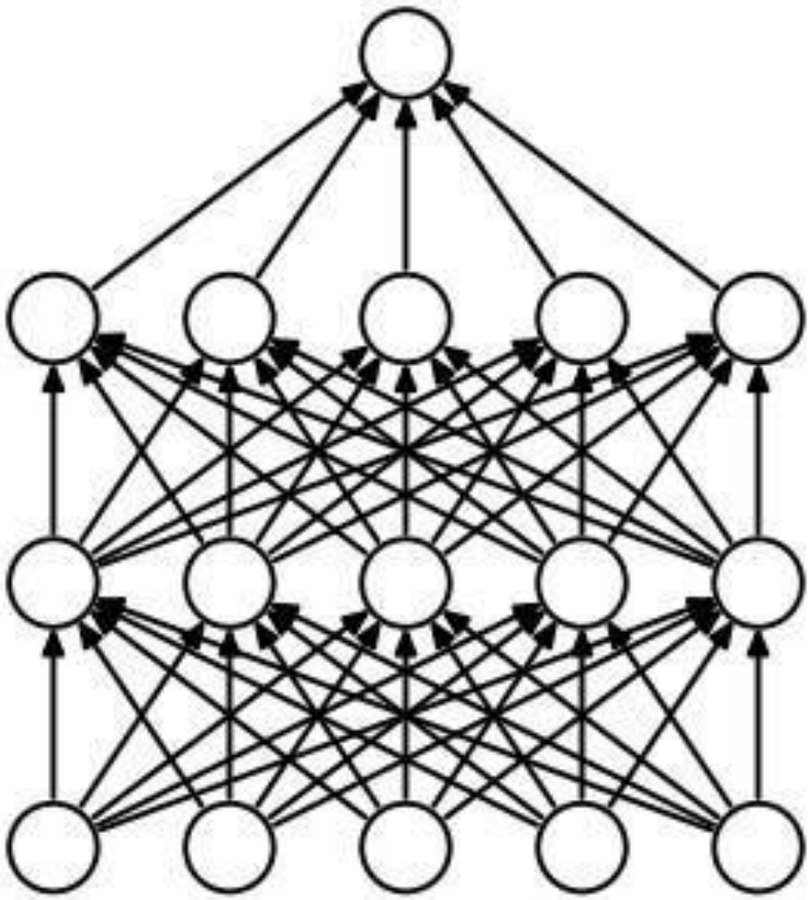
인공신경망 모델을 효율적으로 학습시키기 위한 개선 방법

- 배치 정규화(Batch Normalization)
- 드랍아웃(Dropout)
- 앙상블(Model Ensemble)
- L1, L2 규제
- Early stopping

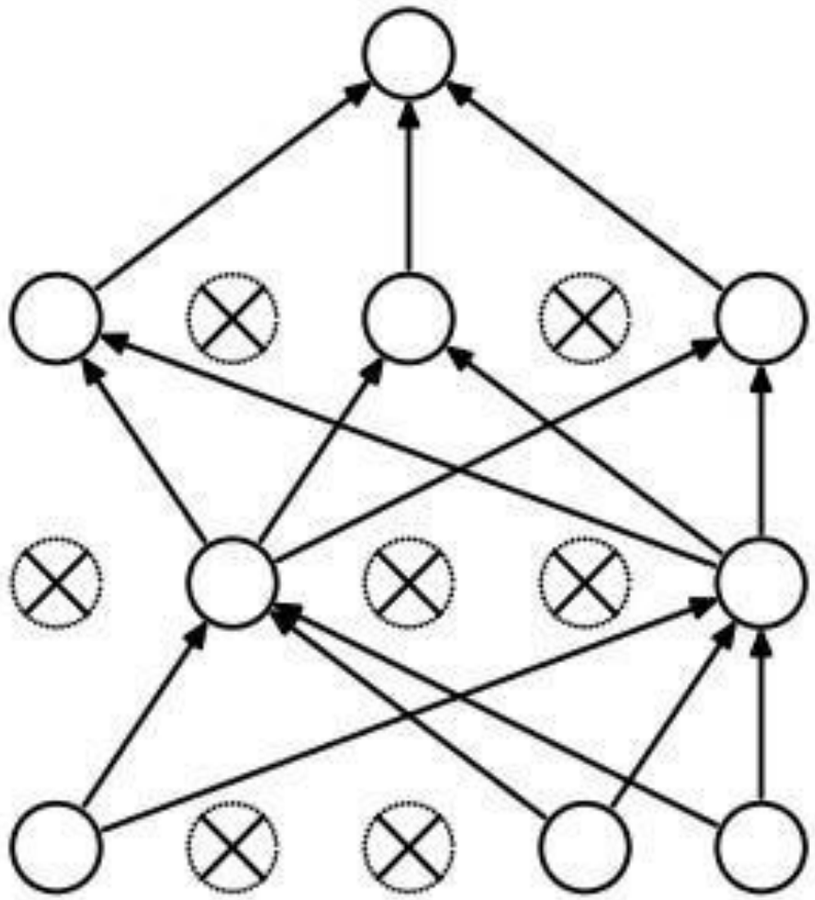
배치정규화

배치 정규화는 인공싞경망에 입력값을 평균 0, 분산 1로 정규화(normalize)해 네트워크의 학습이 잘 일어나도록 돕는 방식

Dropout



(a) Standard Neural Net



(b) After applying dropout.

L1, L2 규제

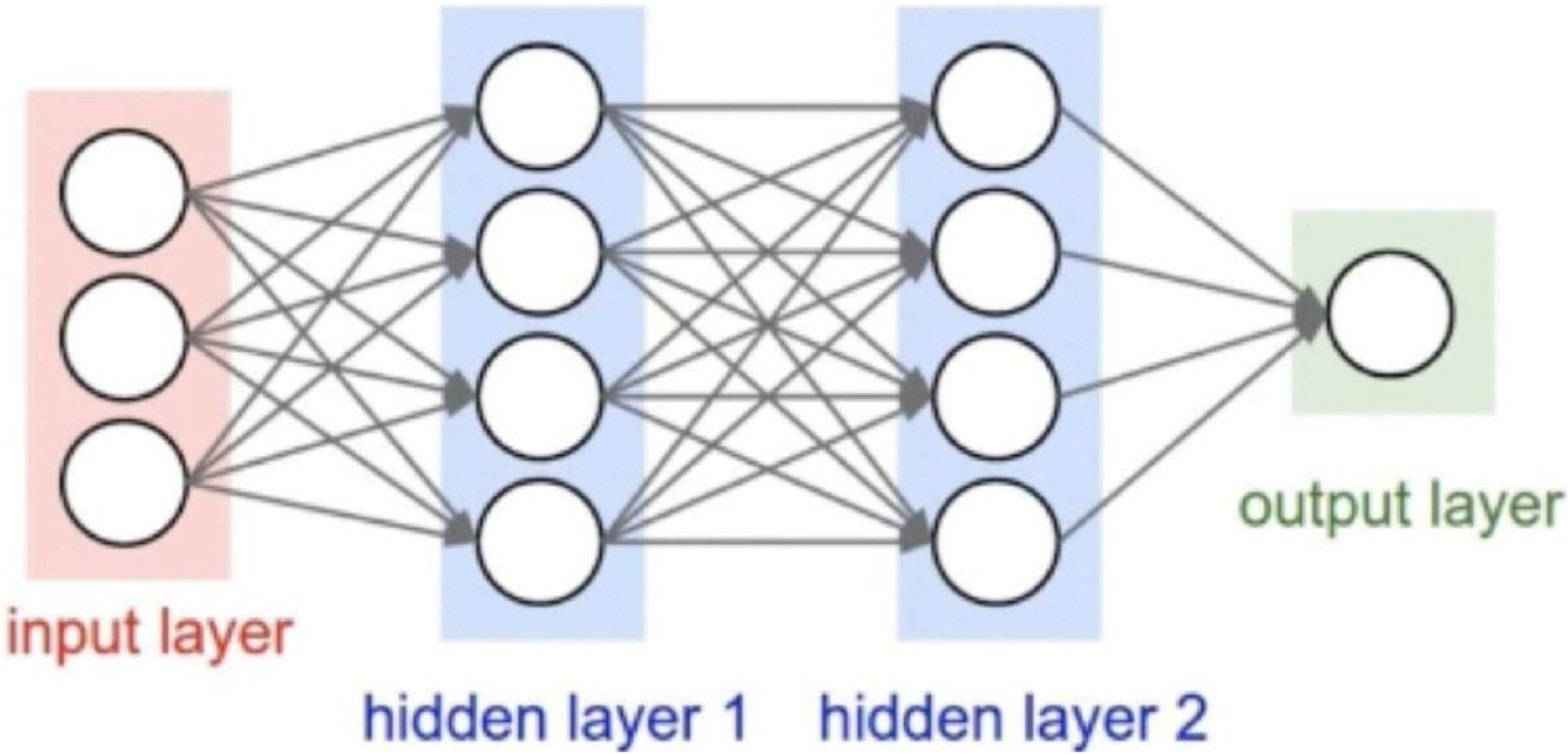
$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2+\alpha\sum_{j=1}^{m}\left|w_j\right| \qquad MSE + \alpha\cdot L_1\text{-}norm$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2+\lambda\sum_{j=1}^{m}w_j^2 \qquad MSE+\lambda\sum_{j=1}^{m}w_j^2$$

DEEP LEARNING

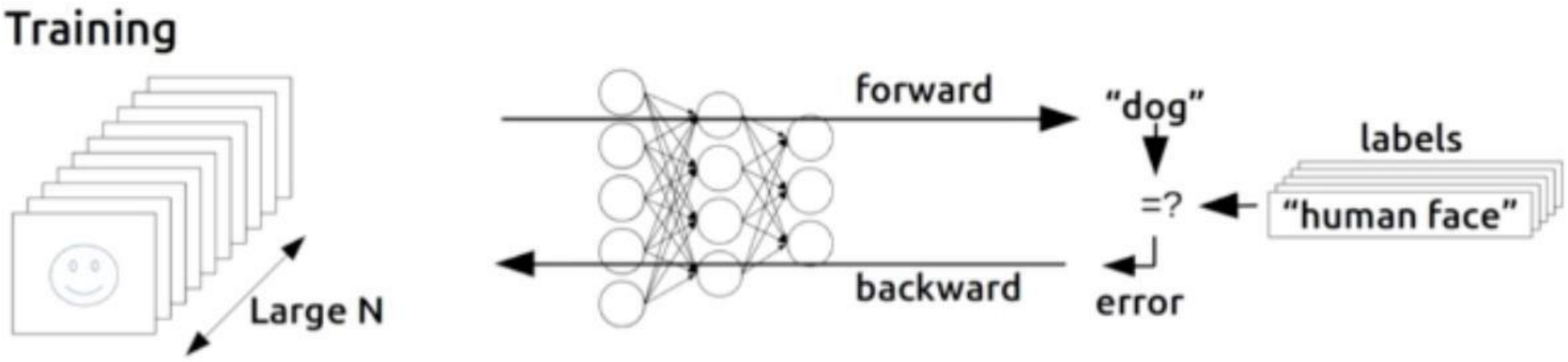
BACK PROPAGATION

DERIVATION

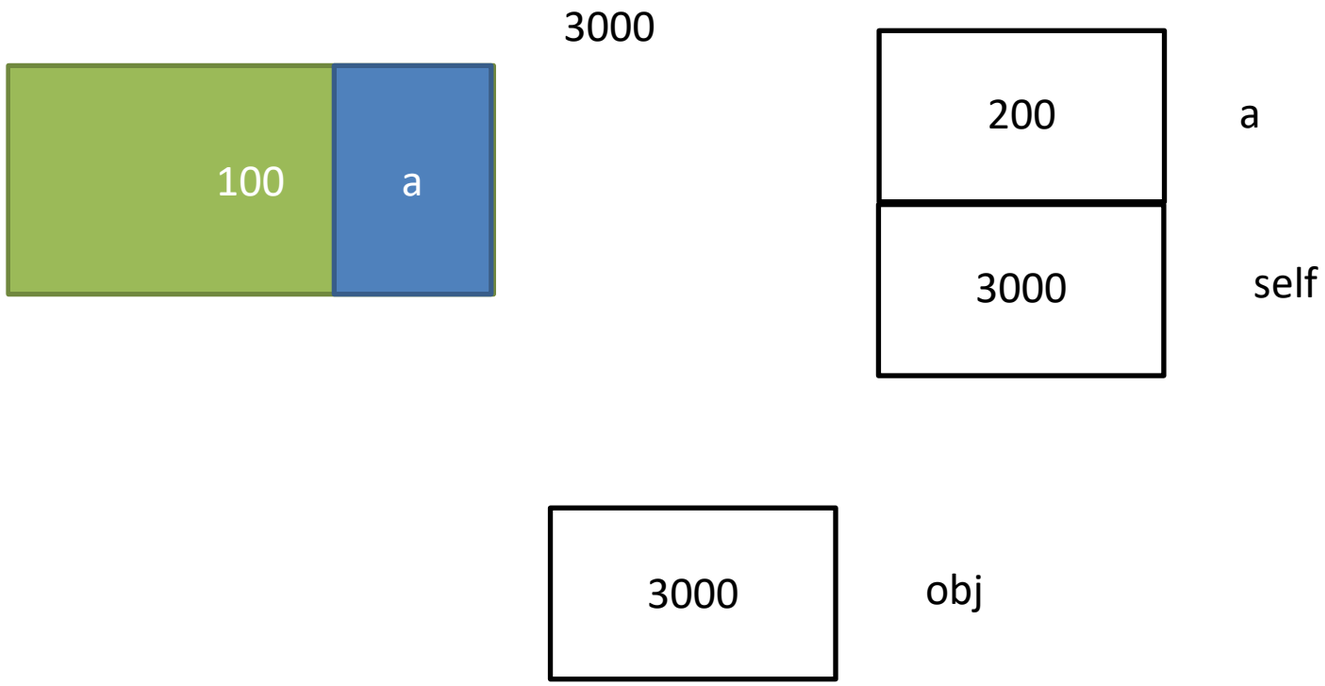


BACKPROPAGATION

Backpropagation
(1974, 1982 by Paul Werbos, 1986 by Hinton)



https://devblogs.nvidia.com/parallelforall/inference-next-step-gpu-accelerated-deep-learning/


3000
100
a
200
a
3000
self
3000
obj

국소적 계산

1) 국소적 계산은 계산 전체에서 어떤 일이 벌어지든 상관없이 자신과 관련된 정보만으로 결과를 출력.

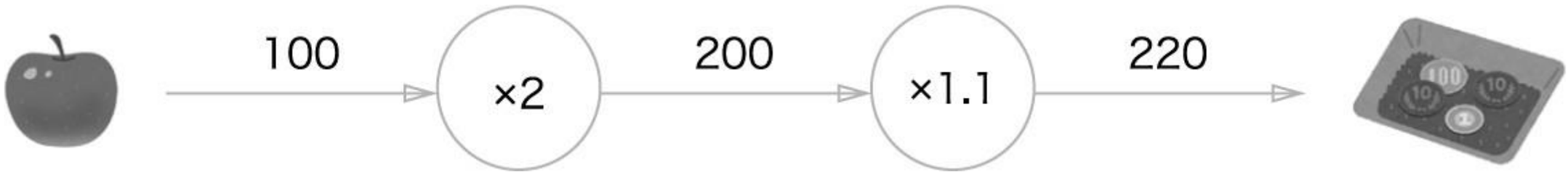
2) *매우 복잡한 수식도 결국 수많은 연산들의 합과 곱이며, 이 수많은 연산들을 하나씩만 놓고 보면 사실 별거없다.* 라는게 국소적 계산

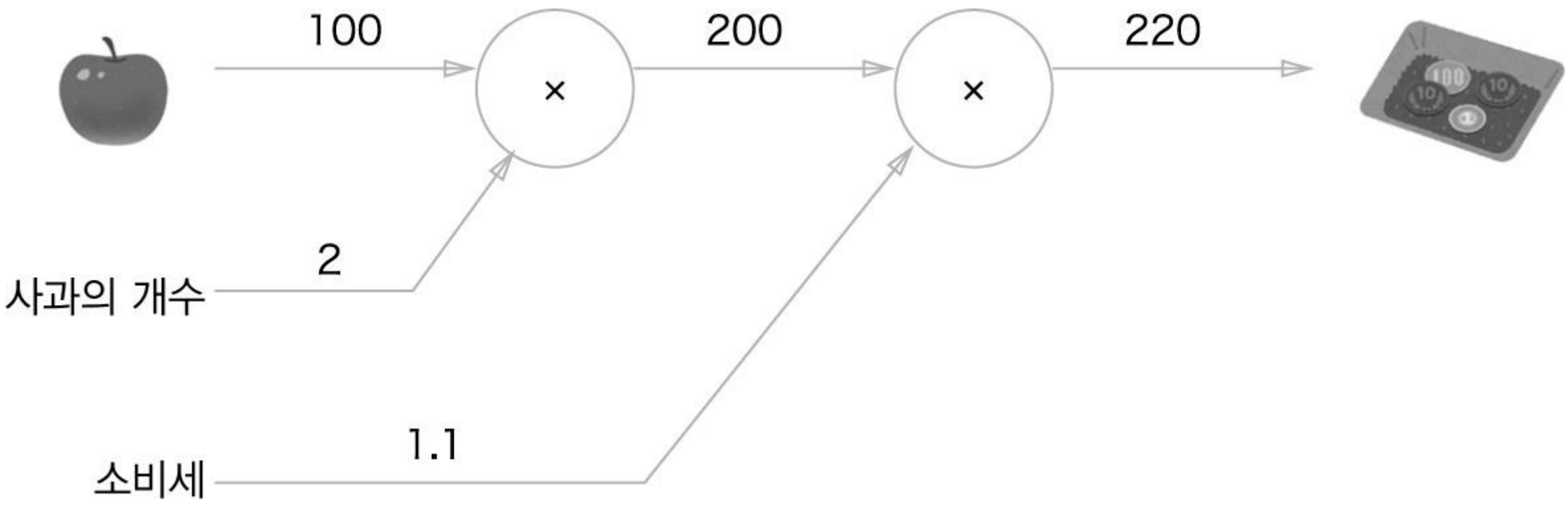
3) 국소적 계산은 연산 하나하나들의 결과를 보관할 수 있다.

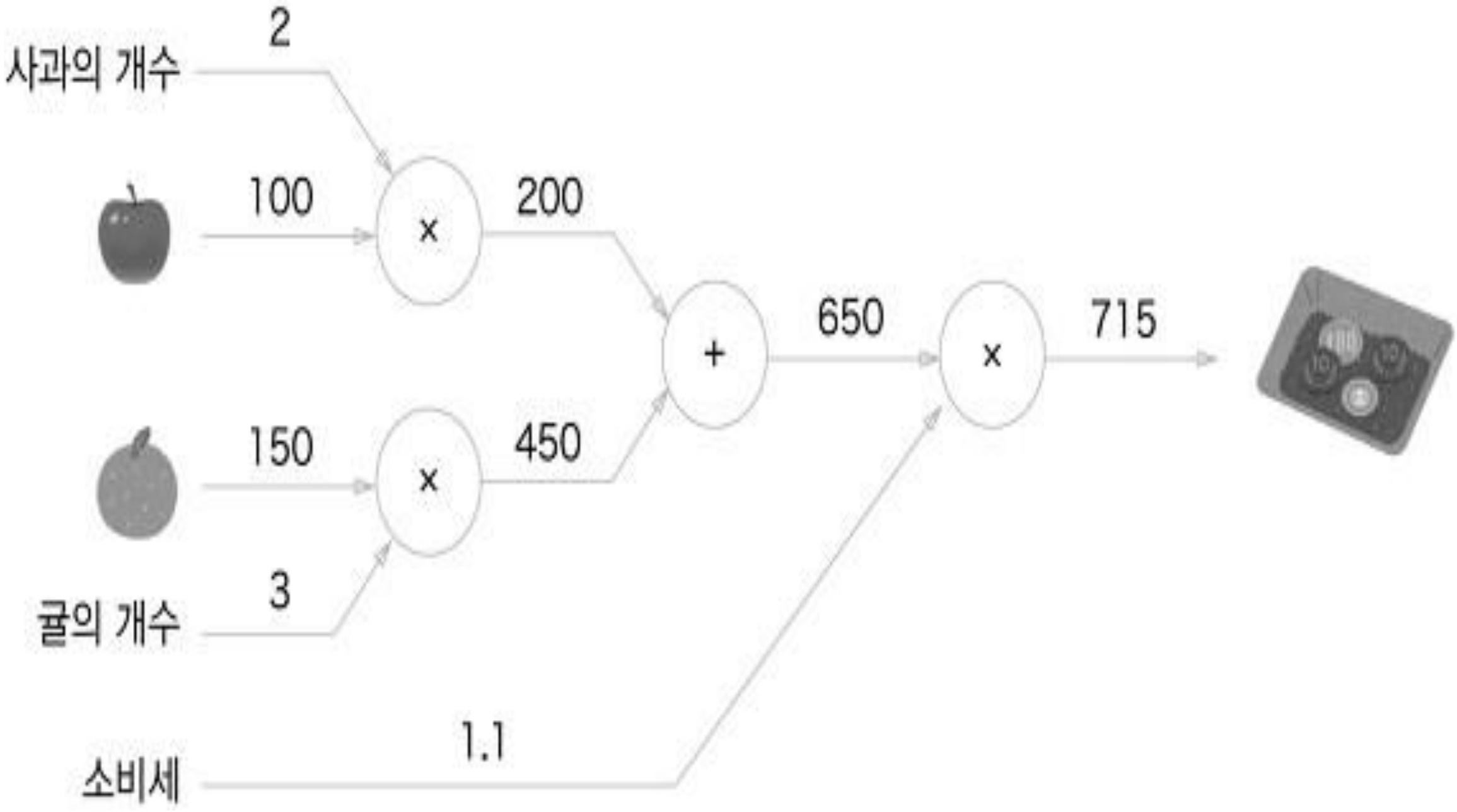
4) 국소적 계산의 효과 덕에 우리는 효율적으로 역전파를 구할 수 있다.

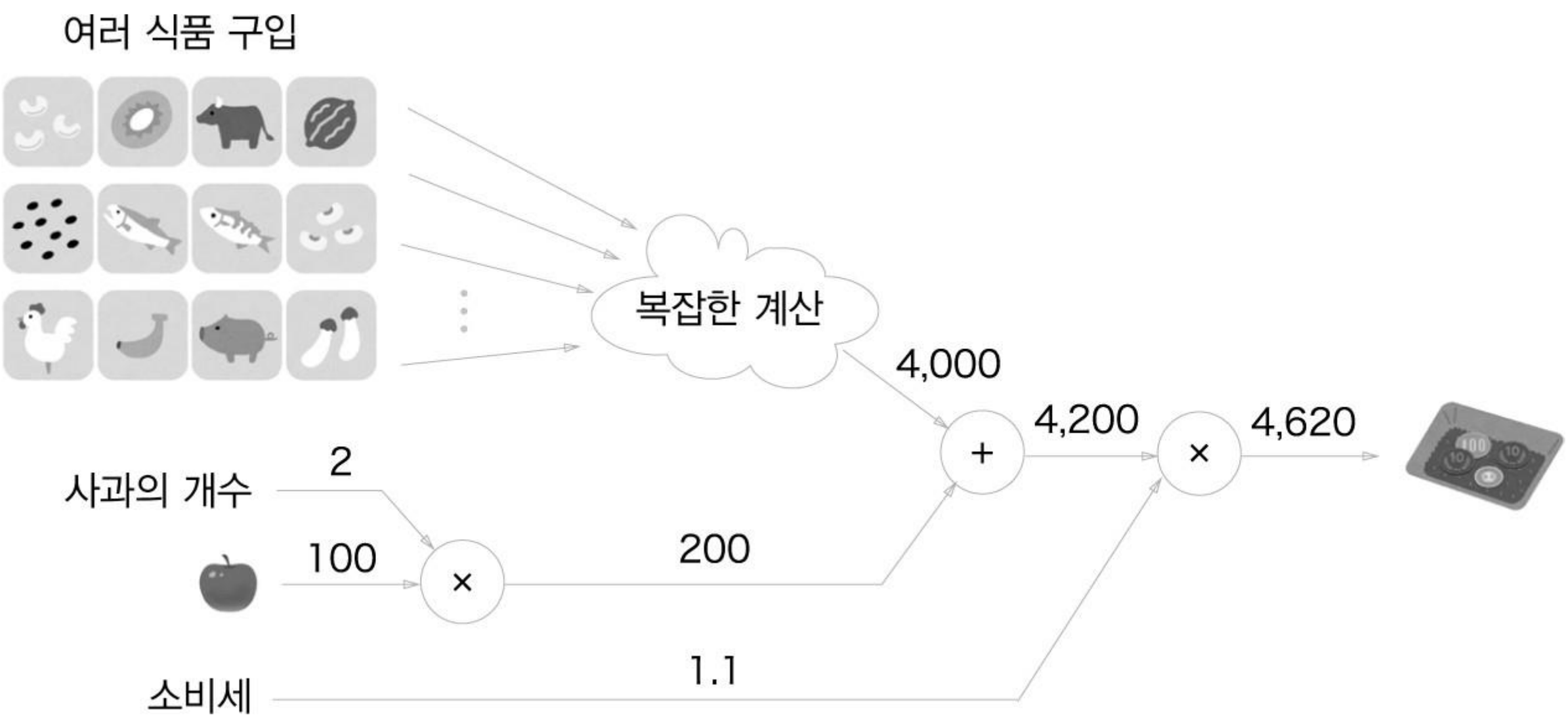
5) 역전파: 위의 계산 그래프를 볼 때 전행 방향(왼쪽에서 오른쪽)의 반대 방향(오른쪽에서 왼쪽)을 의미하며 순전파(→)의 역방향(←)으로 미분 값을 전달 한다.

계산 그래프(computational graph)란 계산 과정을 그래프로 나타낸 것입니다.
그래프는 노드(node)와 에지(edge)로 표현됩니다. 에지는 노드 사이의 직선






사과의 개수
2
100
x
200
+
650
x
715
150
x
450
귤의 개수
3
소비세
1.1

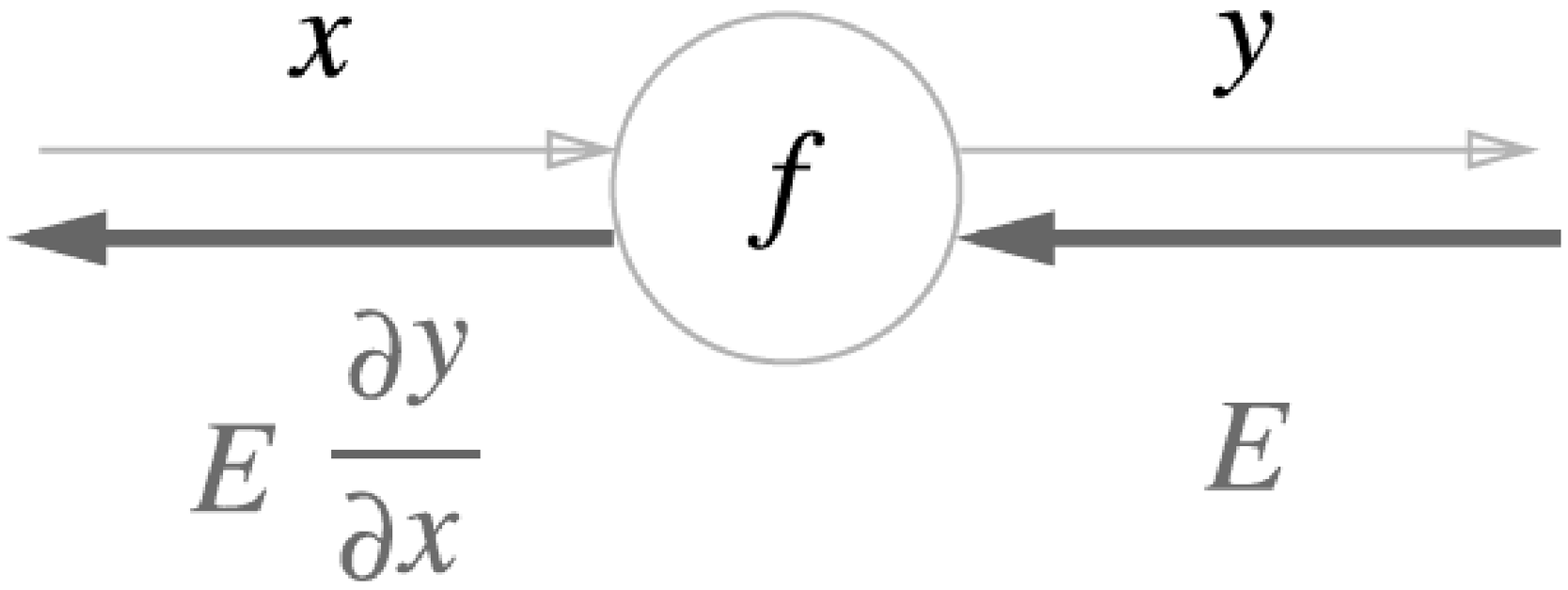

여러 식품 구입
복잡한 계산
4,000
4,200
4,620
+
×
×
사과의 개수
2
100
200
소비세
1.1

계산 그래프 장점

- 국소적 계산: 아무리 복잡한 계산이더라도 각 노드에서는 단순한 계산에 집중할 수 있다

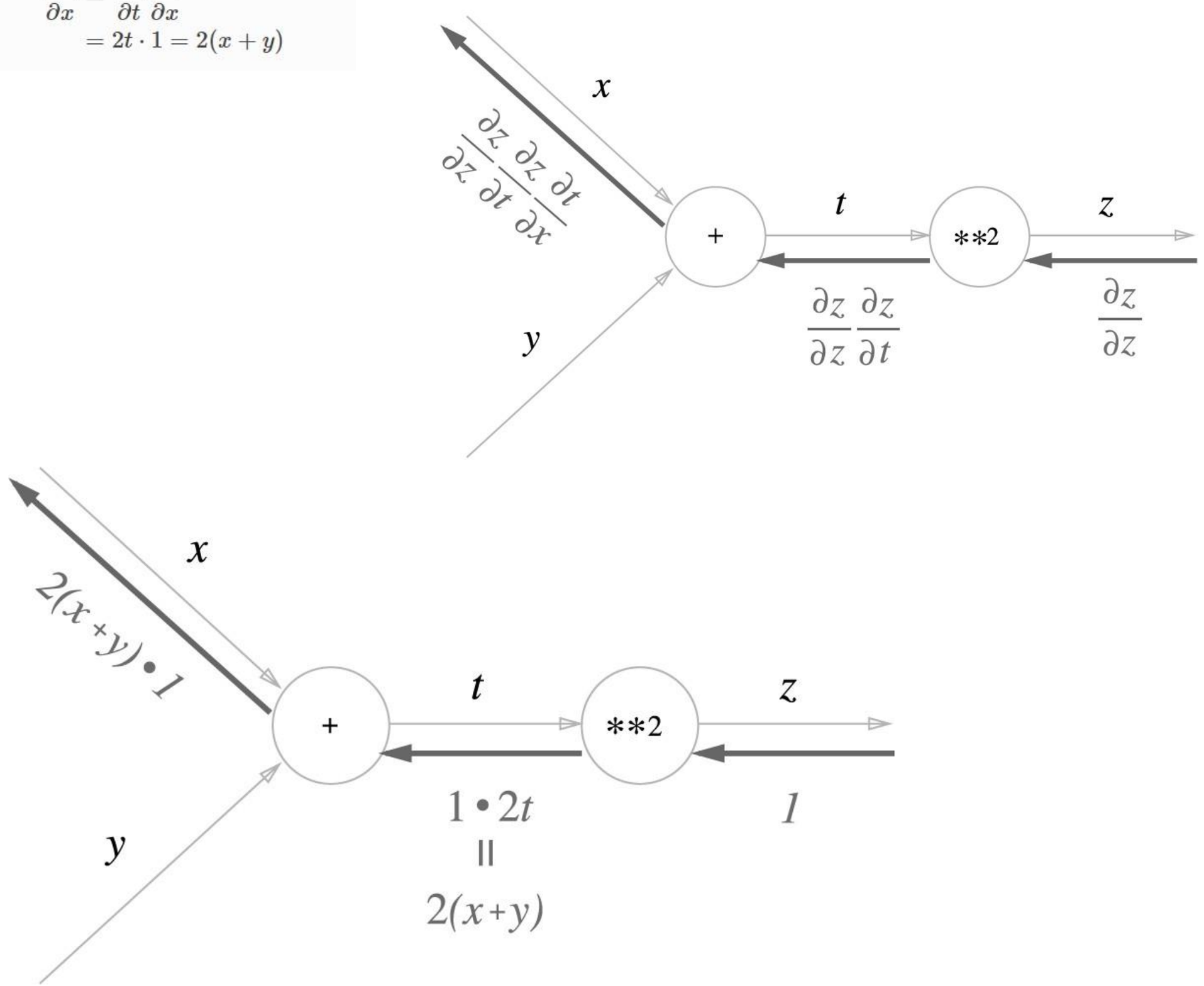
- 중갂 계산 결과를 저장할 수 있다

-역전파를 이용해 미분을 효율적으로 계산할 수 있다


x
y
f
$E \dfrac{\partial y}{\partial x}$
E

함수 $z = (x + y)^2$ 대해서 예를 들어 봅니다.

$$
\begin{aligned}
z &= t^2 \\
t &= x + y \\
\frac{\partial z}{\partial x} &= \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} \\
&= 2t \cdot 1 = 2(x + y)
\end{aligned}
$$

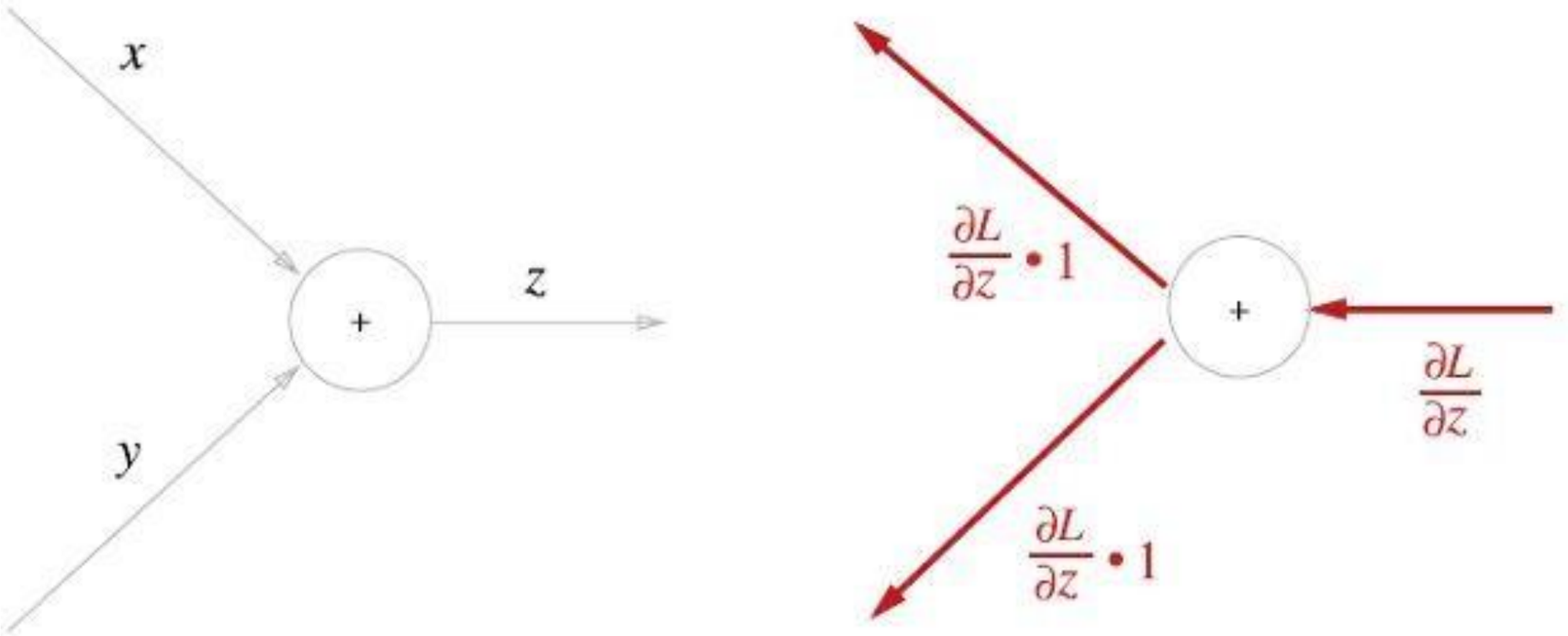


일반 연산(덧셈, 곳셈) 노드의 역젂파

$$z = x + y$$

$$\frac{\partial z}{\partial x} = 1$$

$$\frac{\partial z}{\partial y} = 1$$

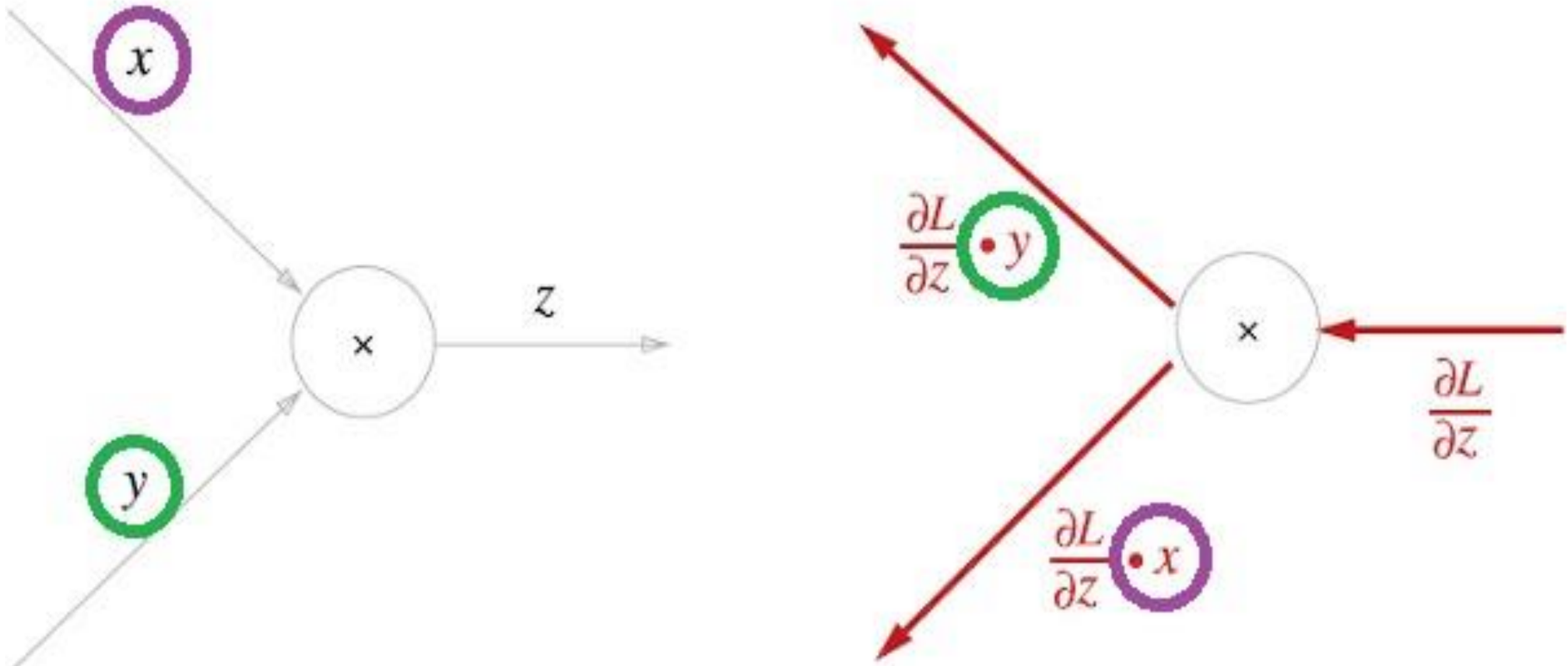


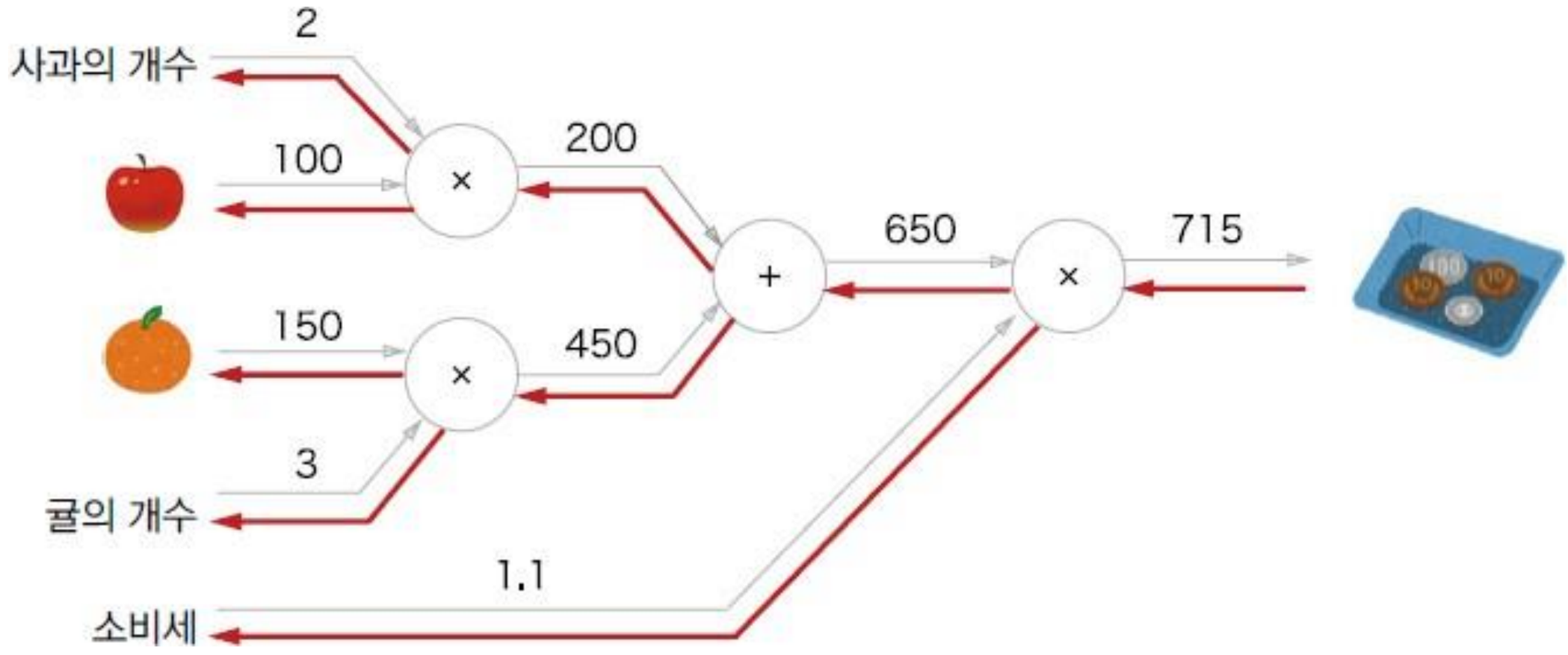
덧셈 노드의 역전파는 입력 값을 그대로 흘려보낸다.

$$z = xy$$

$$\frac{\partial z}{\partial x} = y$$

$$\frac{\partial z}{\partial y} = x$$




2
사과의 개수
100
×
200
150
×
450
3
귤의 개수
+
650
×
715
1.1
소비세

CHAIN RULE

① forward($w = -2, x = 5, b = 3$)
② backward

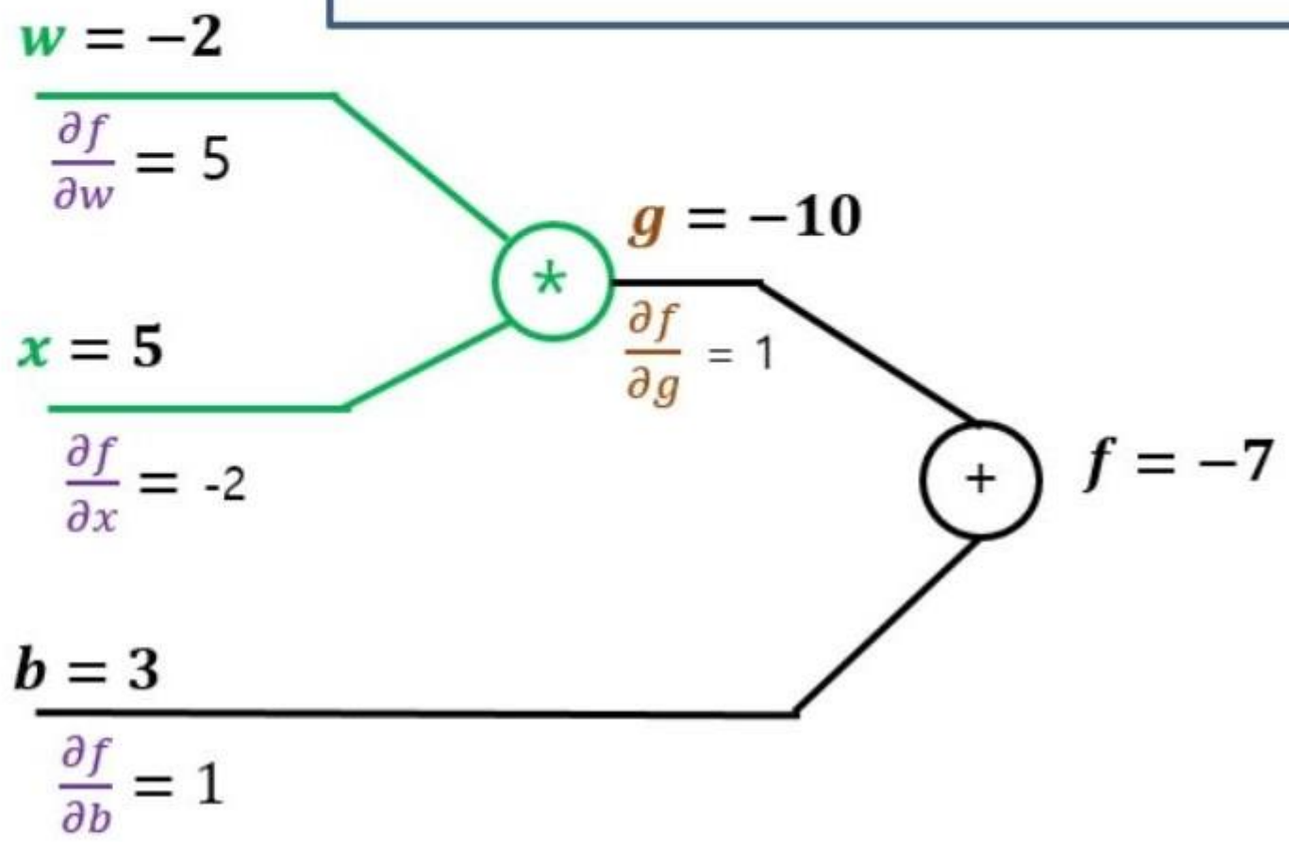
$$f = wx + b$$

$g = wx$, $f = g + b$

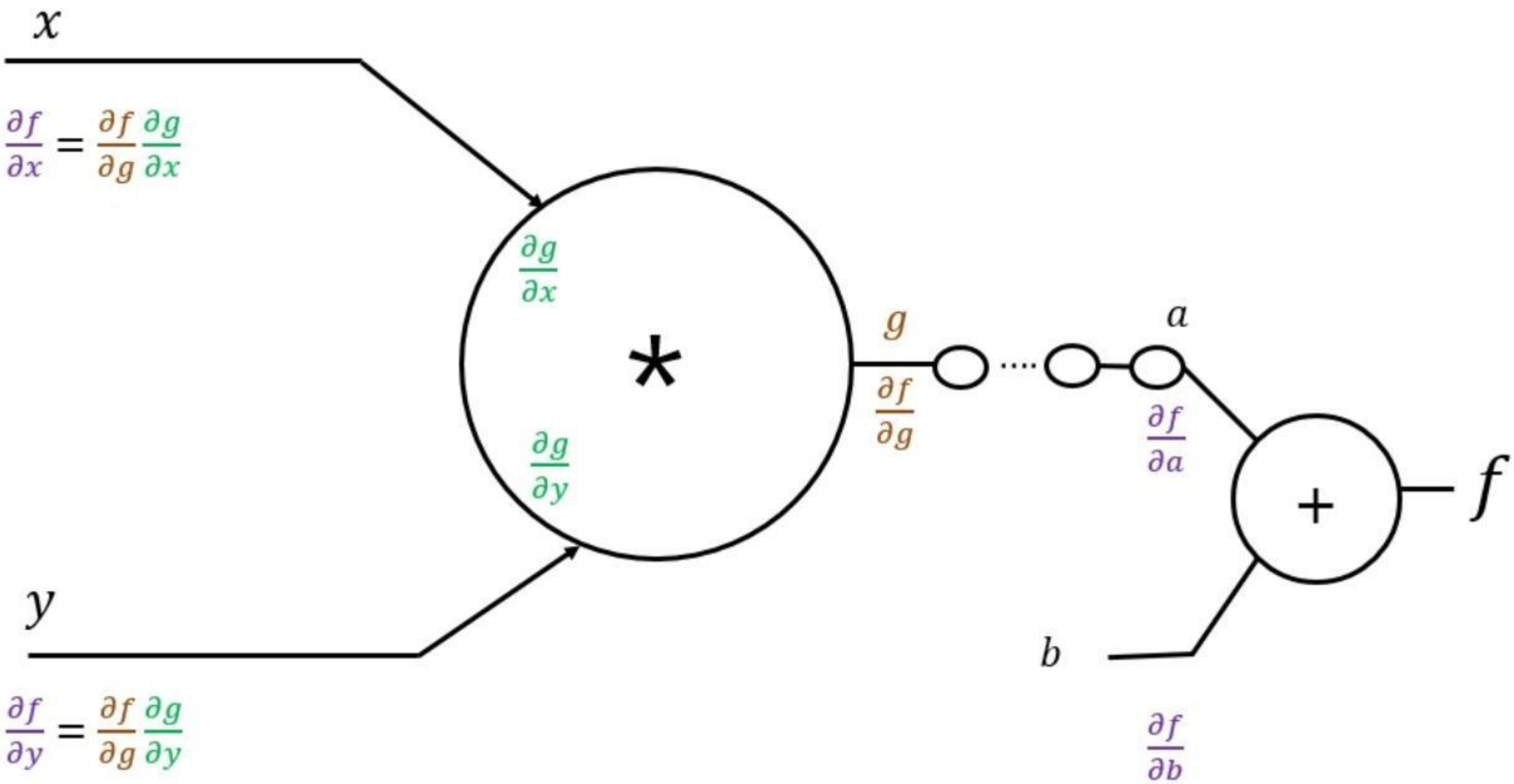
$\frac{\partial g}{\partial w} = x, \frac{\partial g}{\partial x} = w$ $\quad$ $\frac{\partial f}{\partial g} = 1, \frac{\partial f}{\partial b} = 1$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = 1 * w = -2$$

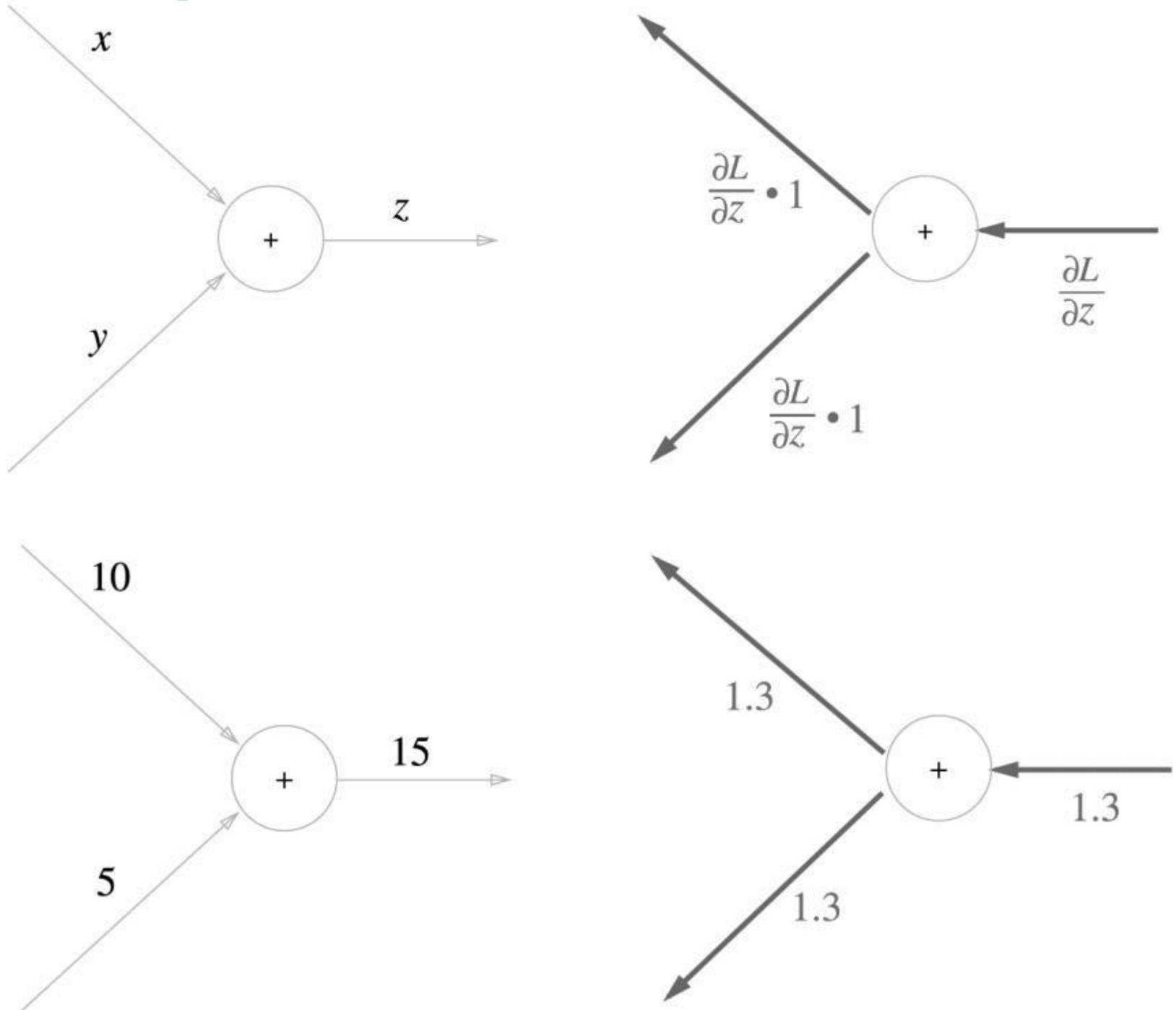
$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = 1 * x = 5$$



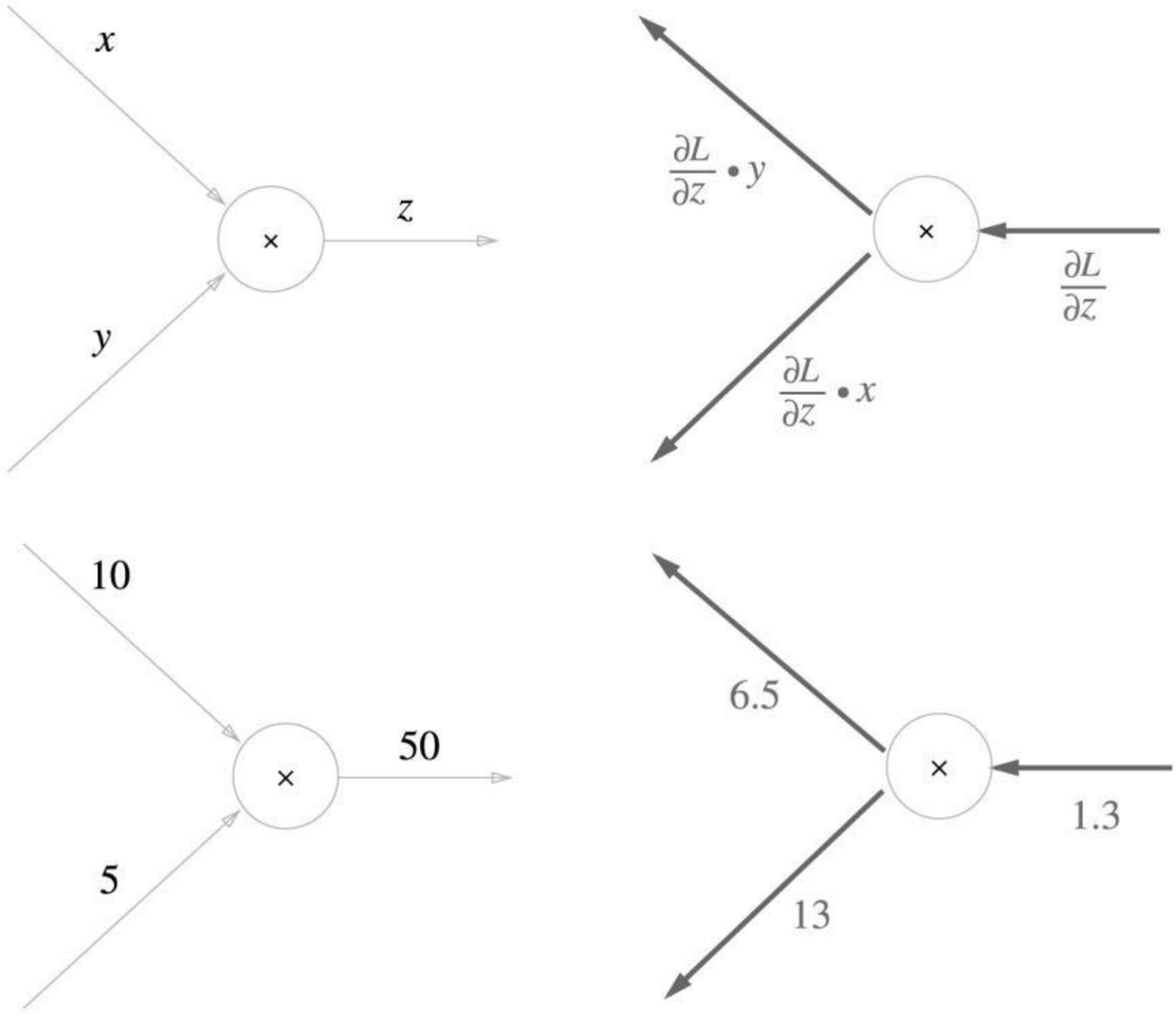
CHAIN RULE


x
$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$
$\frac{\partial g}{\partial x}$
*
$\frac{\partial g}{\partial y}$
g
$\frac{\partial f}{\partial g}$
a
$\frac{\partial f}{\partial a}$
+
f
y
b
$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial y}$
$\frac{\partial f}{\partial b}$

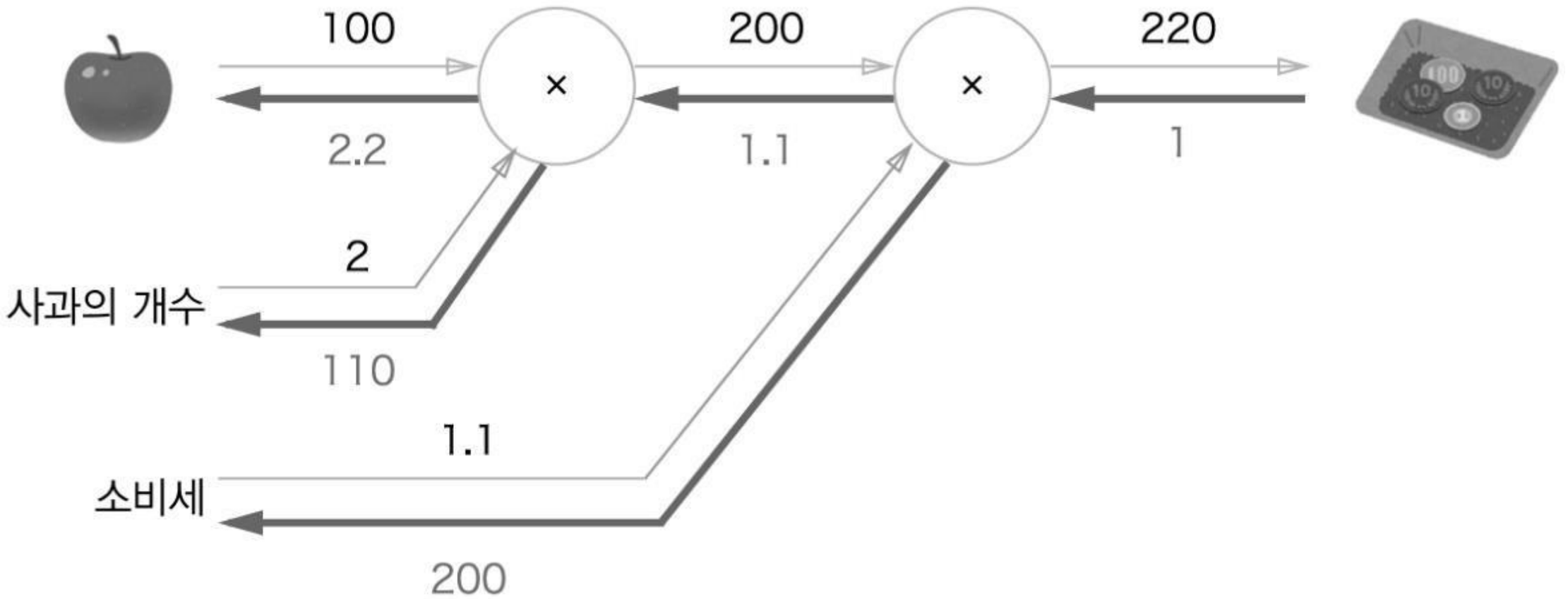
계산 그래프 (덧셈)



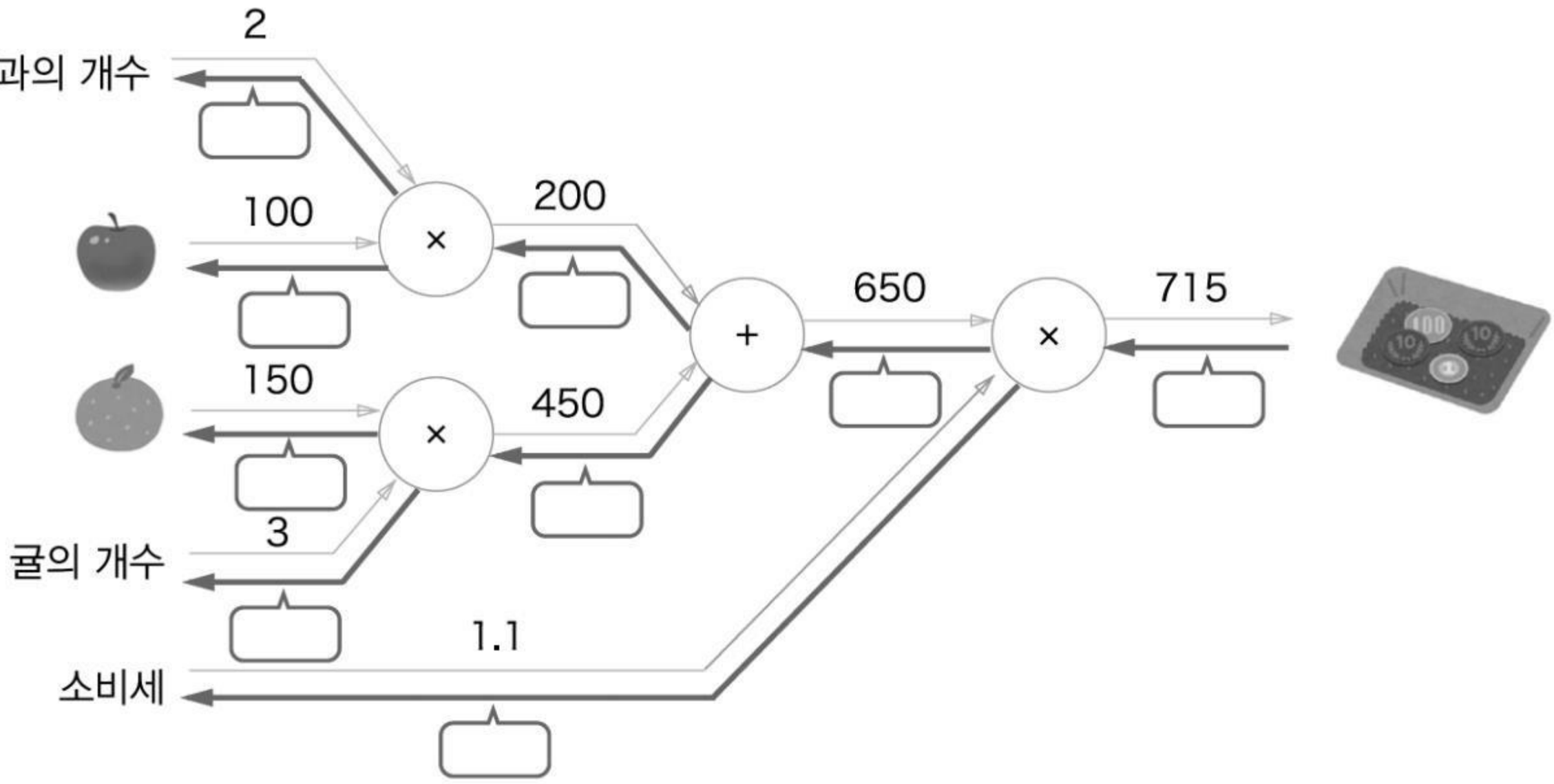
계산 그래프 (곱셈)


x
y
×
z
$\frac{\partial L}{\partial z} \bullet y$
$\frac{\partial L}{\partial z} \bullet x$
$\frac{\partial L}{\partial z}$
10
5
50
6.5
13
1.3

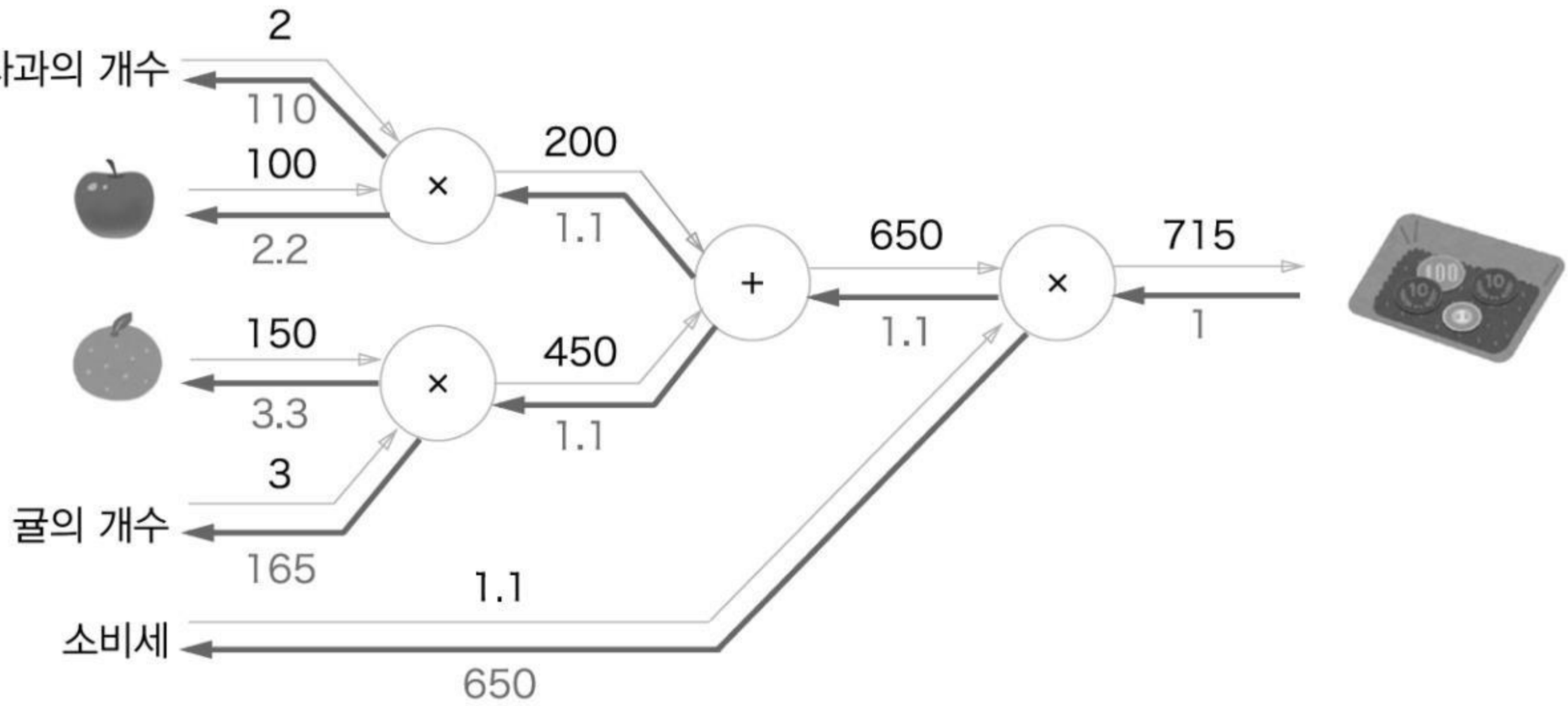
사과 그래프



과일 그래프 (퀴즈)

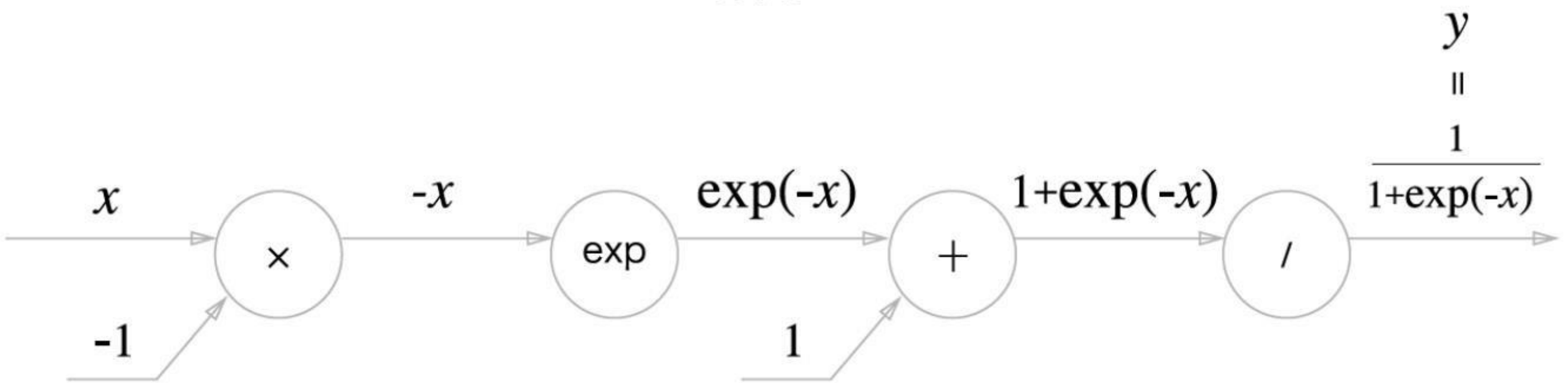


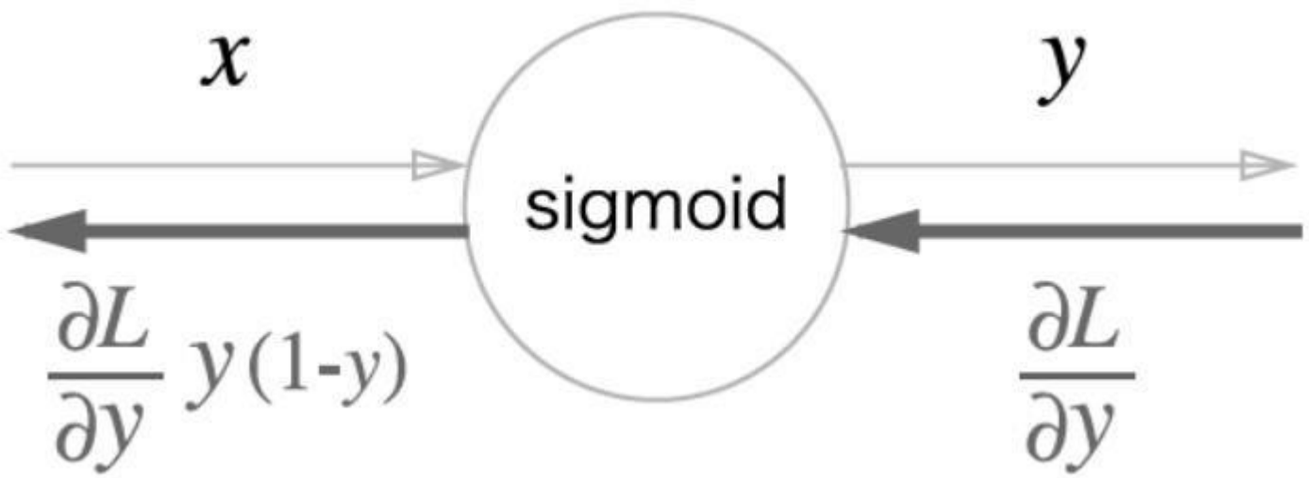
과일 그래프


2
사과의 개수
110
100
200
×
1.1
2.2
650
715
+
×
150
450
×
3.3
1.1
1.1
3
귤의 개수
165
1.1
소비세
650

SIGMOID

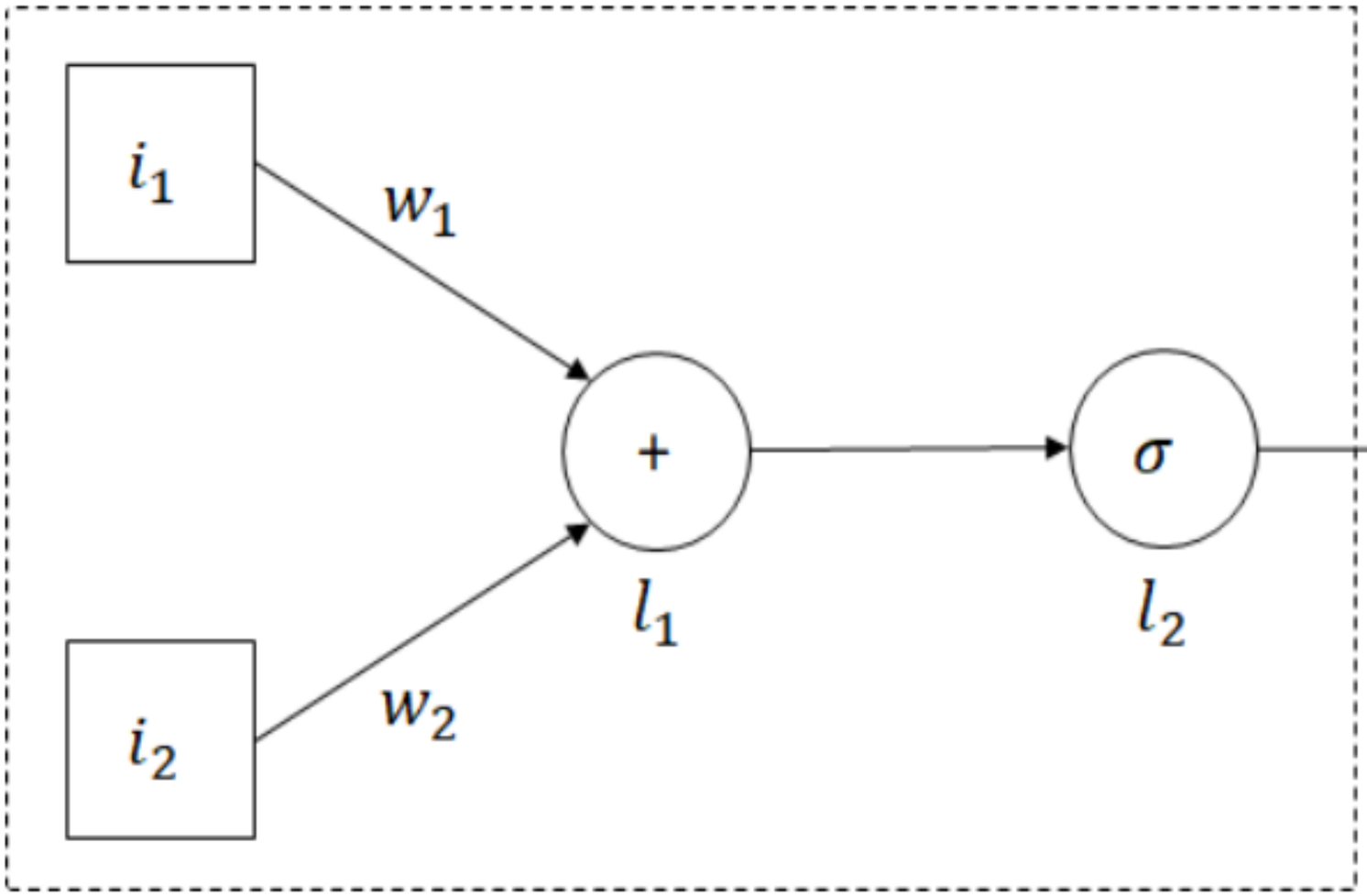
$$g(z)=\frac{1}{1+e^{-z}}$$

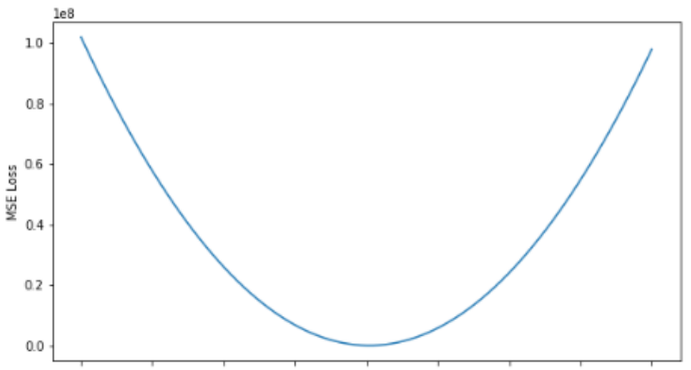




BACKPROPAGATION

로지스틱 example

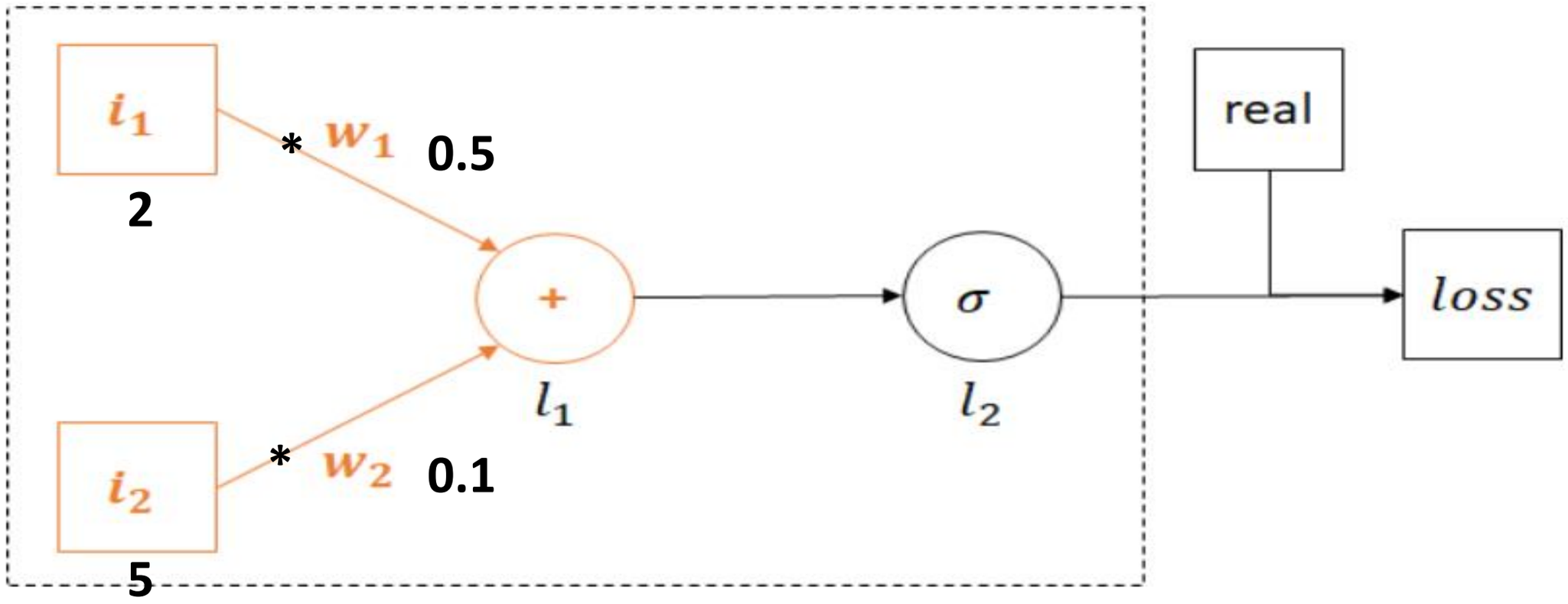




$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

Forward Pass

먼저 forward pass를 진행해 보자!



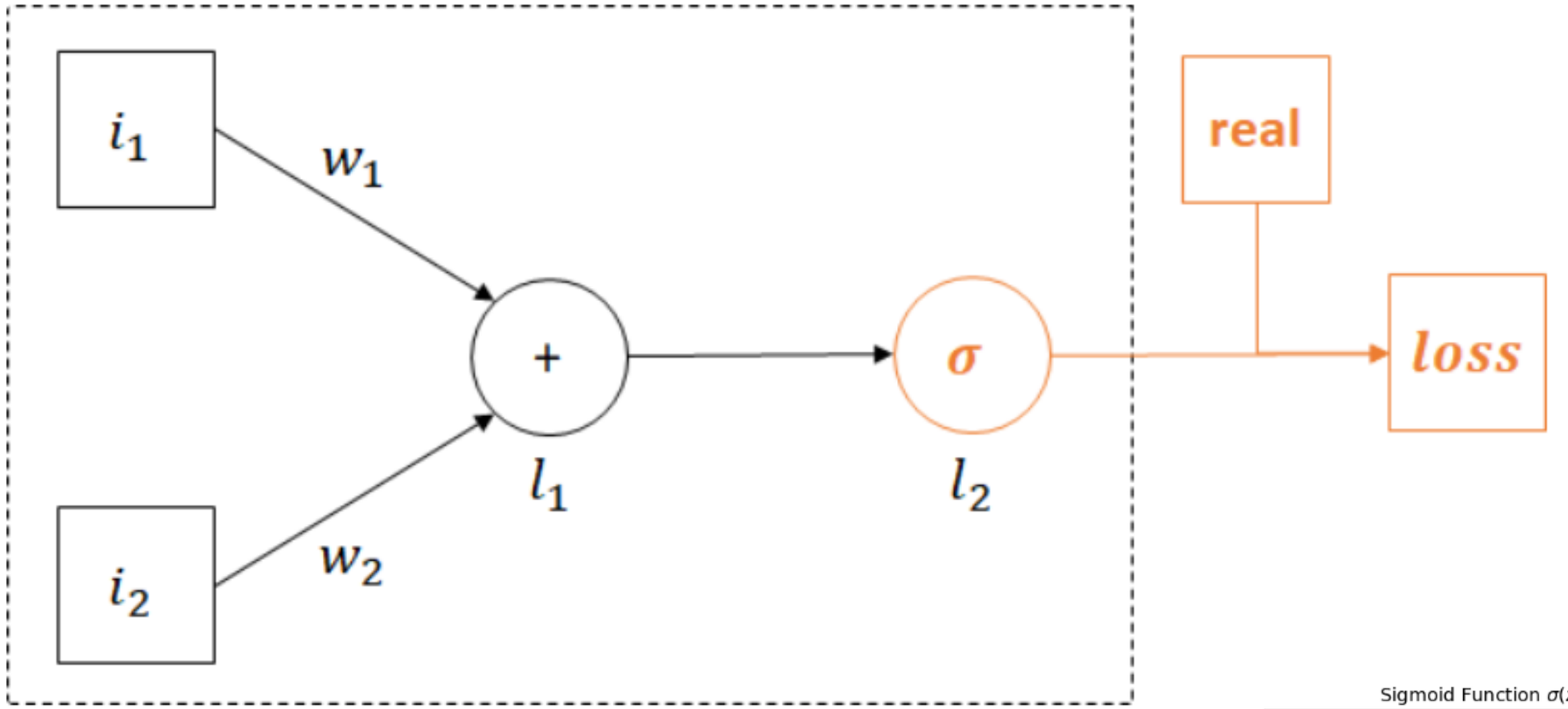
i_1, i_2에 각각 2, 5가 들어왔고, w_1, w_2가 각각 0.5, 0.1로 설정되어있다고 한다면

l_1는 $2 * 0.5 + 5 * 0.1 = 1.5$가 된다.

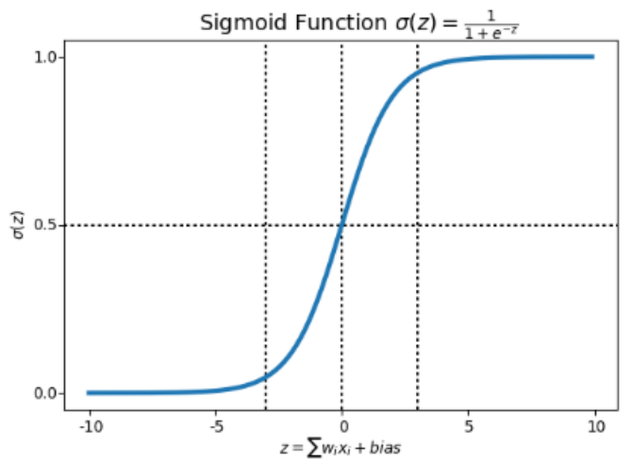
이를 matrix 형태로 나타내면

$[2 \quad 5] \begin{bmatrix} 0.5 \\ 0.1 \end{bmatrix} = 1.5$ 로 표현할 수 있다.

출처:https://yonghyuc.wordpress.com/2019/09/10/backpropagation-weight-update/



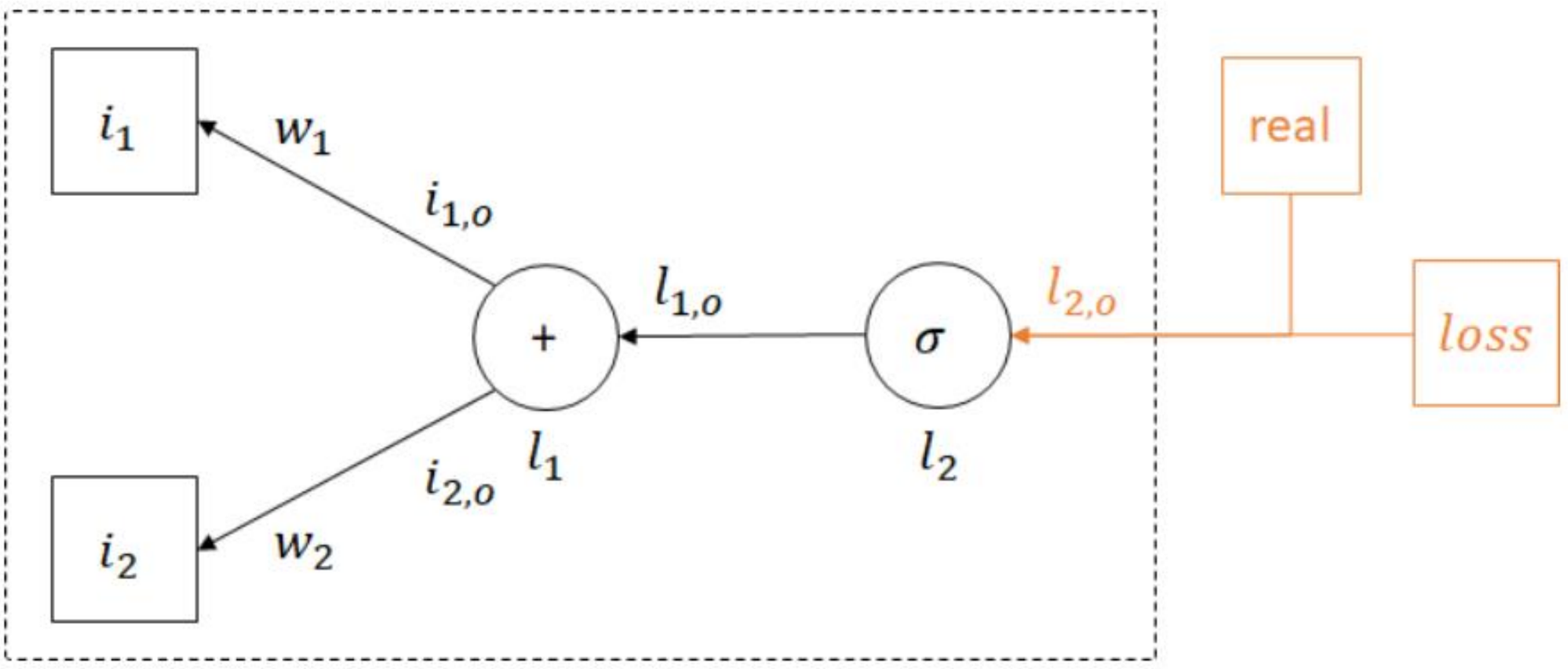
$\sigma(1.5) = 0.818$이 나온다.



실제 값이 0 이라면, loss는 $\frac{1}{1}(0 - 0.818)^2 = 0.67$이 된다.

Backward Pass (Back-propagation)

이제 backward pass를 통해 우리의 weight (w_1, w_2)를 update해보자.

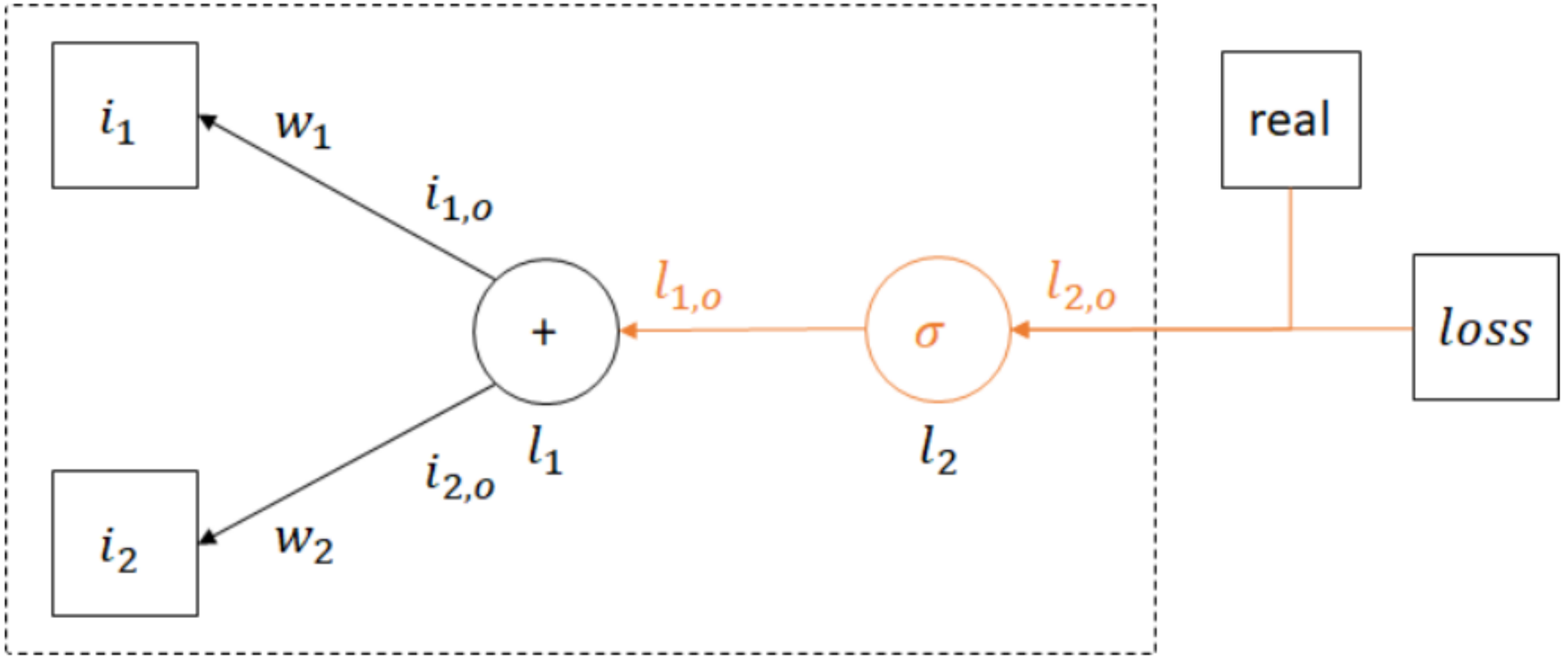


이제 첫번째로 $\frac{\partial loss}{\partial l_{2,o}}$를 구해보자.

MSE수식인 $\frac{1}{n}\Sigma(y-\tilde{y})^2$를 미분하면 $-\frac{2}{n}(y-\tilde{y})$가 되고

결국 $\frac{\partial loss}{\partial l_{2,o}} = -\frac{2}{1}(0-0.818) = 1.636$ 가 된다.

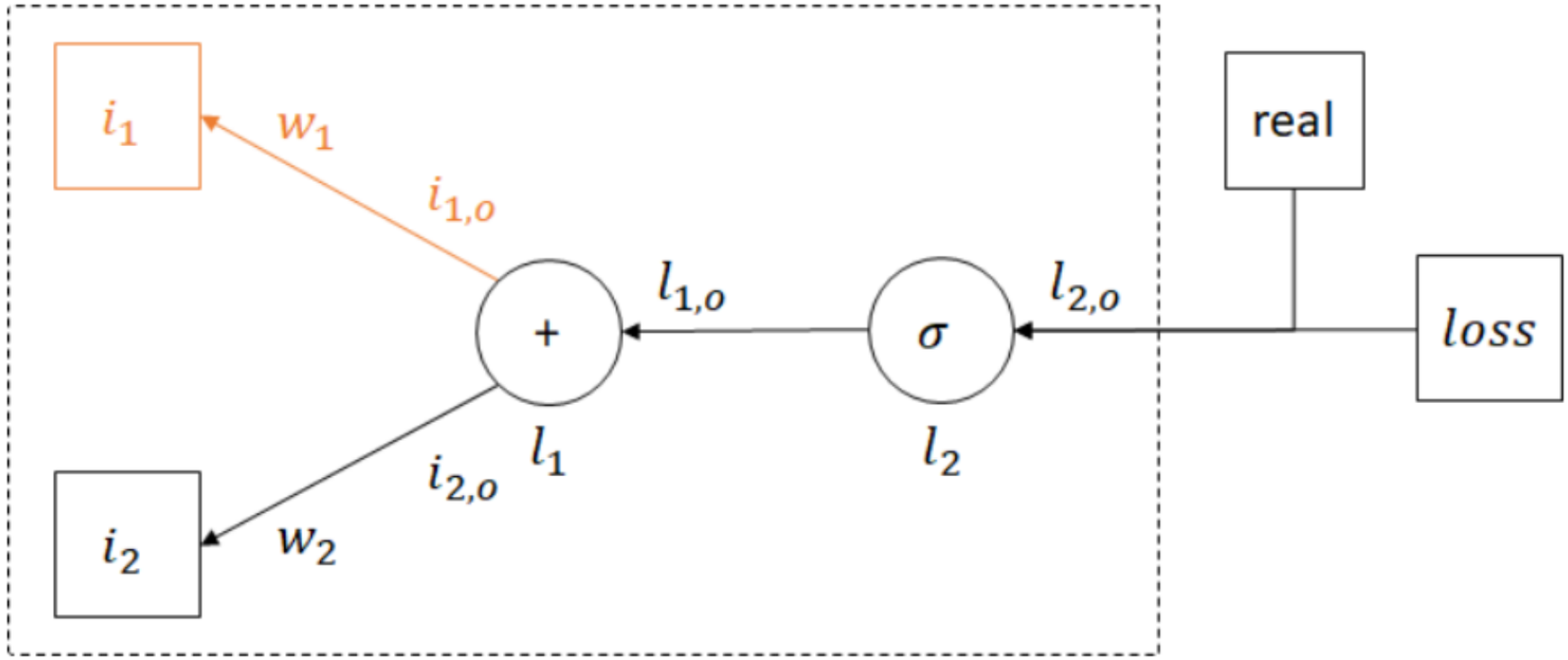
이제부터는 chain rule에 의해 연속적으로 이루어 지게 된다.



Sigmoid의 미분 값은 $\sigma'(x) = \sigma(x)(1 - \sigma(x))$이고

chain rule에 의해 $\frac{\partial loss}{\partial l_{1,o}} = \frac{\partial loss}{\partial l_{2,o}} * \frac{\partial l_{2,o}}{\partial l_{1,o}}$가 되므로

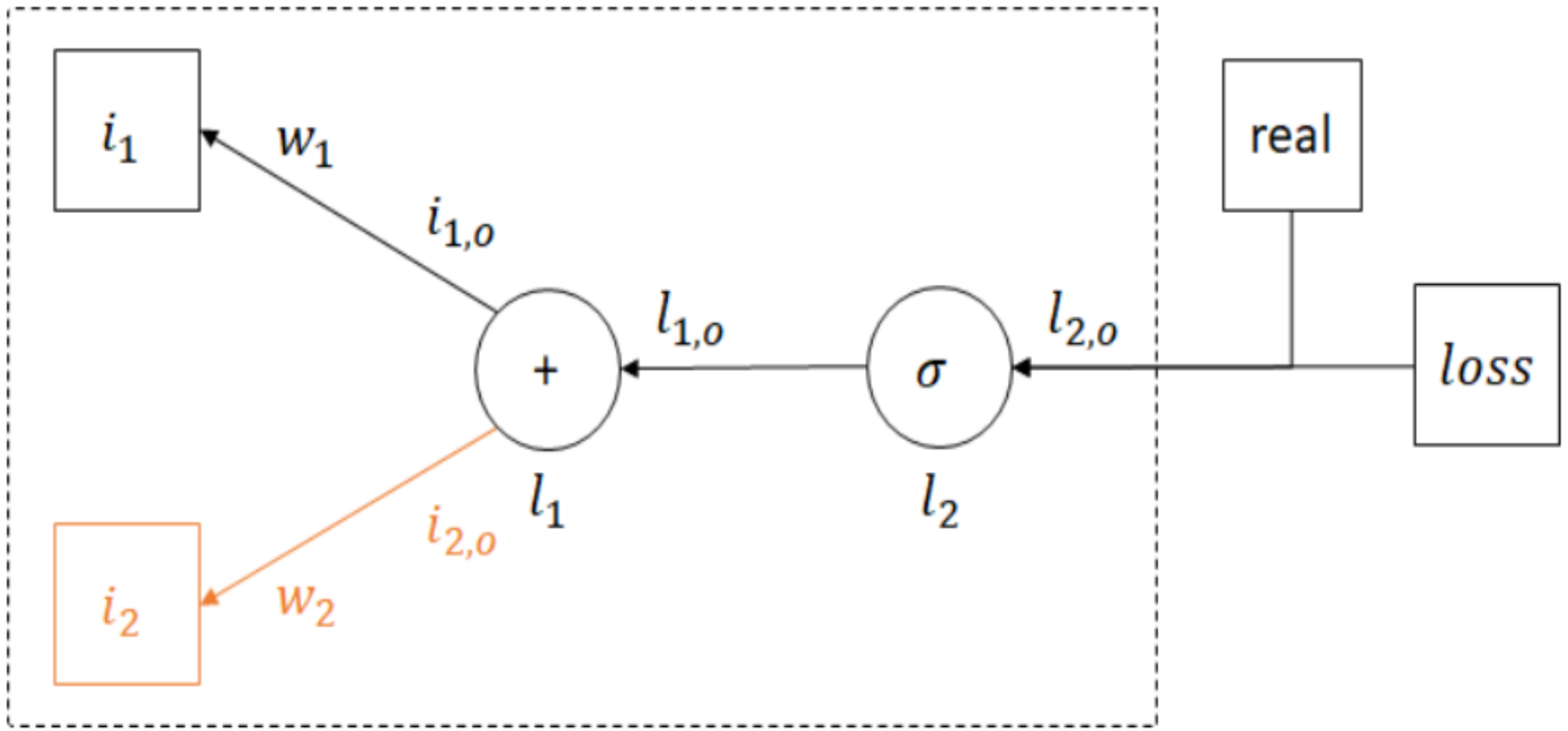
$\frac{\partial loss}{\partial l_{1,o}} = 1.636 * (0.818) * (1 - 0.818) = 0.244$가 된다.



또한 w_1와 i_1의 관계를 살펴보면

$i_{2,0} = w_1 * i_1$이므로 w_1의 gradient는 결국 i_1이 된다고 할 수 있다.

따라서 $\frac{\partial loss}{\partial w_1} = \frac{\partial loss}{\partial i_{1,o}} * \frac{\partial i_{1,o}}{\partial w_1} = 0.244 * 2 = 0.488$가 된다.



마찬가지로 $\frac{\partial loss}{\partial w_2} = \frac{\partial loss}{\partial i_{2,o}} * \frac{\partial i_{2,o}}{\partial w_2} = 0.244 * 5 = 1.22$가 된다.

이제 optimizing이 가능한 w_1과 w_2를 update하면 되는데

기본적으로 앞에서 구한 gradient에 learning rate를 곱한 값으로 weight를 update한다.

(여기서는 learning rate를 1로 설정한다)

$$w_1' = w_1 + (-\frac{\partial loss}{\partial w_1} * lr) = 0.5 + (-0.488 * 1) = 0.0122$$

이제 w_1는 0.5에서 0.136으로 update된다.

w_2 도 동일하게 계산하면 0.1에서 -1.12로 update된다.

산 내려오는 작은 오솔길 잘찾기(Optimizer)의 발달 계보

모든 자료를 다 검토해서
내 위치의 산기울기를 계산해서
갈 방향을 찾겠다.

GD

스텝방향

Momentum
스텝 계산해서 움직인 후,
아까 내려 오던 관성 방향 또 가자

Nesterov Accelerated Gradient

NAG
일단 관성 방향 먼저 움직이고,
움직인 자리에 스텝을 계산하니
더 빠르더라

Nadam
Adam에 Momentum
대신 NAG를 붙이자.

Adam
RMSProp + Momentum
방향도 스텝사이즈도 적절하게!

SGD
전부 다봐야 한걸음은
너무 오래 걸리니까
조금만 보고 빨리 판단한다
같은 시간에 더 많이 간다

스텝사이즈

RMSProp
보폭을 줄이는 건 좋은데
이전 맥락 상황봐가며 하자.

Adagrad
안가본곳은 성큼 빠르게 걸어 훑고
많이 가본 곳은 잘아니까
갈수록 보폭을 줄여 세밀히 탐색

AdaDelta
종종걸음 너무 작아져서
정지하는걸 막아보자.

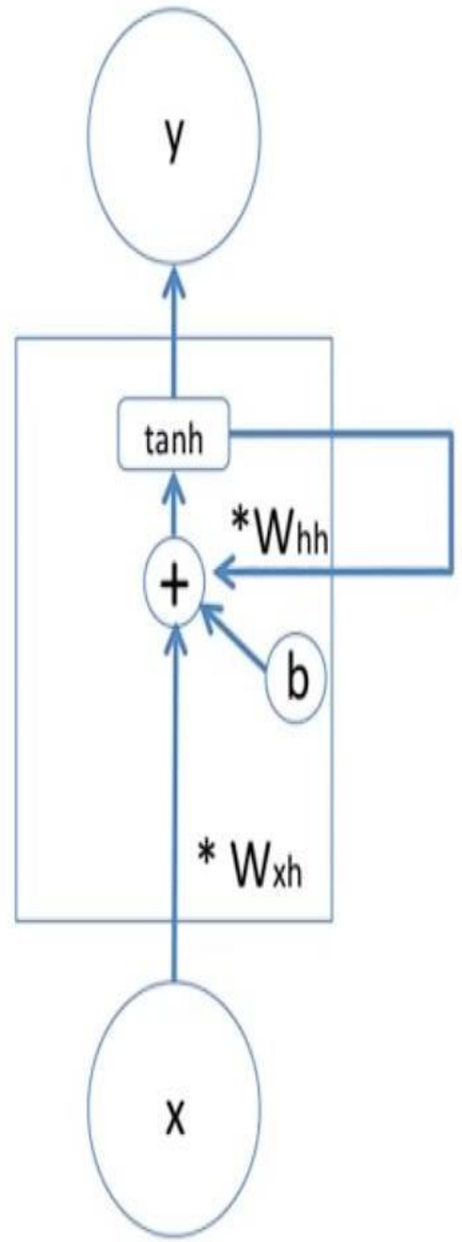
RNN

Recurrent Neural Network (RNN)

순환신경망(Recurrent Neural Network)

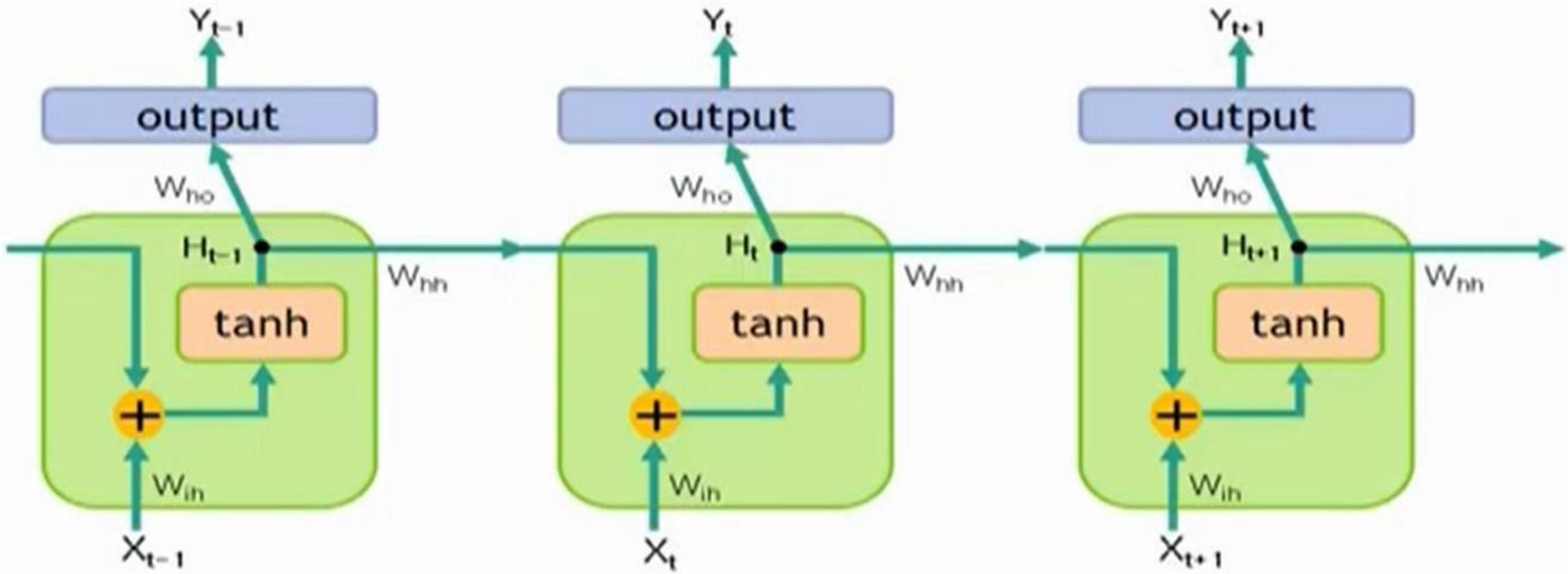
- 여러개의 데이터가 순서대로 입력되었을때 앞서 입력받은 데이터를 잠시 기억해 놓는 방법
- 모든 입력 값에 이 작업이 순서대로 이뤄지며 같은 층을 맴도는 것으로 보여 순환 신경망이라 부른다

simplify of model diagram



$$h_t = \tanh(W_{xh} * x_t + W_{hh} * h_{t-1} + b)$$

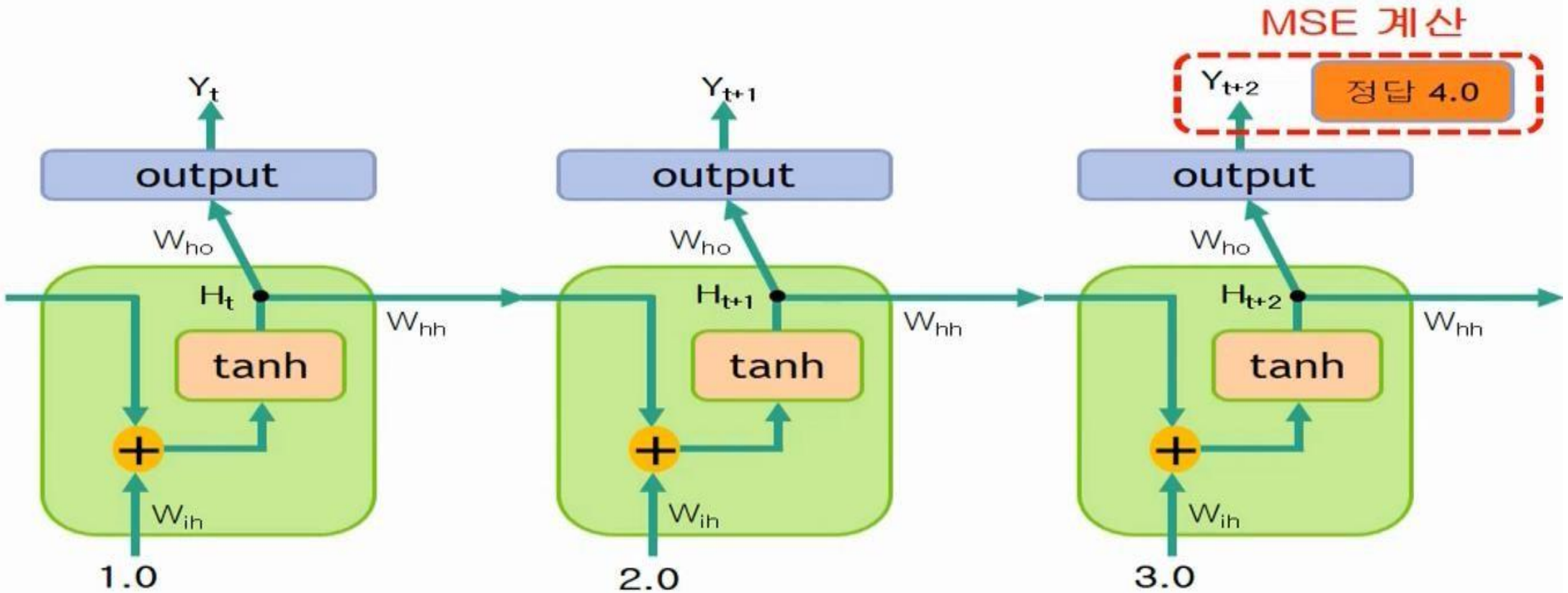
$$h_t = \tanh(W_{xh} * x_t + W_{hh} * h_{t-1} + b)$$



✓ X_{t-1}, X_t, X_{t+1} 은 입력 데이터를 나타내고 H_{t-1}, H_t, H_{t+1} 등은 은닉층 개념의 SimpleRNN 레이어 출력 값을, 그리고 Y_{t-1}, Y_t, Y_{t+1} 등은 출력층의 출력 값을 나타냄

✓ 학습 대상의 가중치는 ① 입력층과 은닉층 사이의 가중치 W_{ih} ② 시간 t 에서의 은닉층과 시간 t+1 에서의 은닉층 간의 가중치 W_{hh} ③ 은닉층과 출력층 사이의 가중치 W_{ho}

✓ 시간 t 에서 은닉층 SimpleRNN 레이어 출력 $H_t = \tanh(X_t W_{ih} + H_{t-1} W_{hh})$




```
tf.keras.layers.SimpleRNN(units=10, activation='tanh', input_shape=(3, 1))
```

일반 신경망의 은닉층 노드 수와 같은 개념. 즉 units=10 의미는 SimpleRNN 레이어 내의 노드 개수는 10개이며 노드 1개당 활성화 함수 1개를 가지고 있음

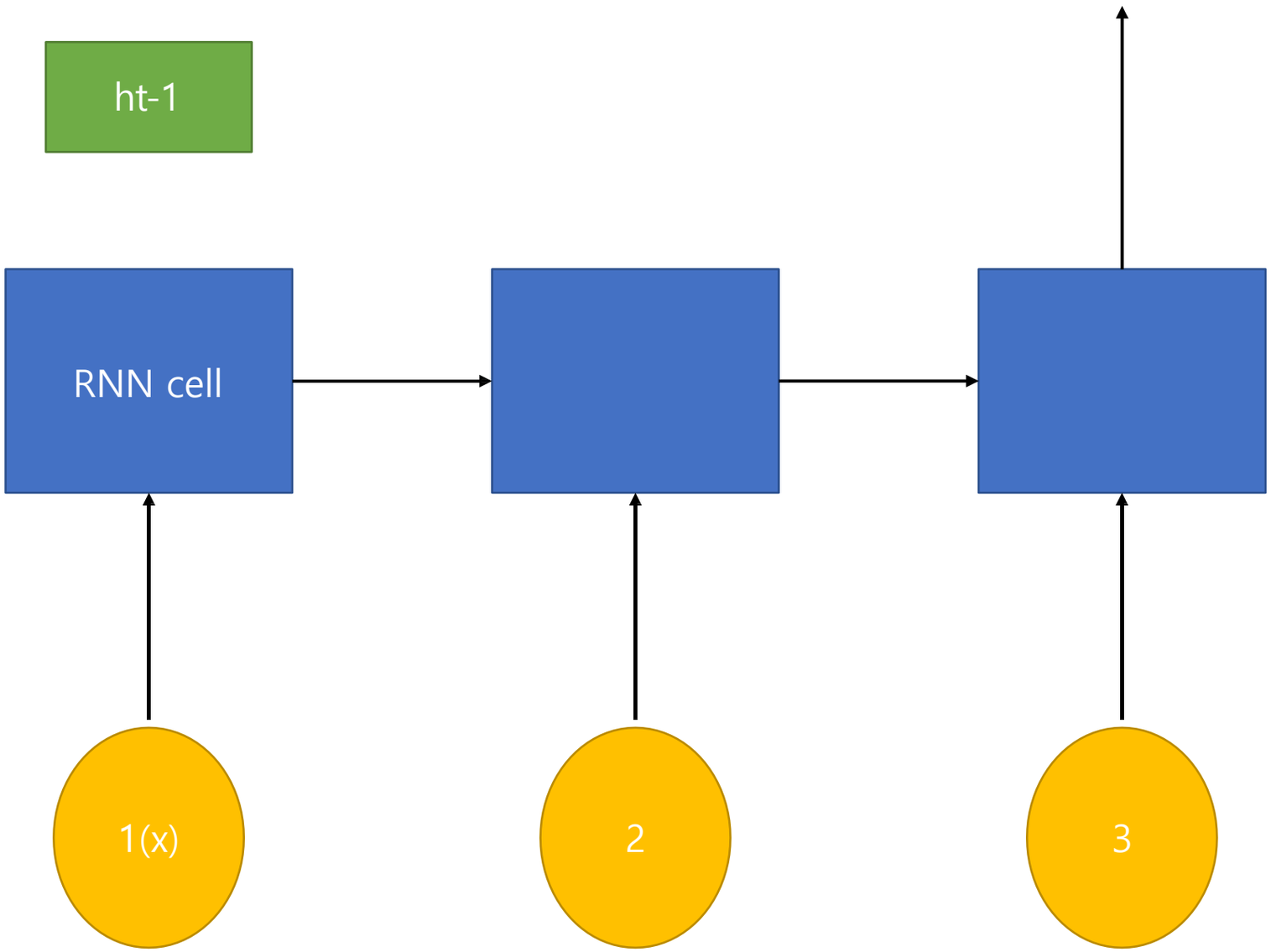
'relu' 같은 다른 활성화 함수를 사용 할 수도 있음

3개의 time-step 데이터를 이용해서 정답을 만든다는 의미이며 window size=3 과 같음

RNN 레이어로 한번에 1개의 데이터가 들어간다는 의미

4, 3, 1
(batch size, time, feature size)

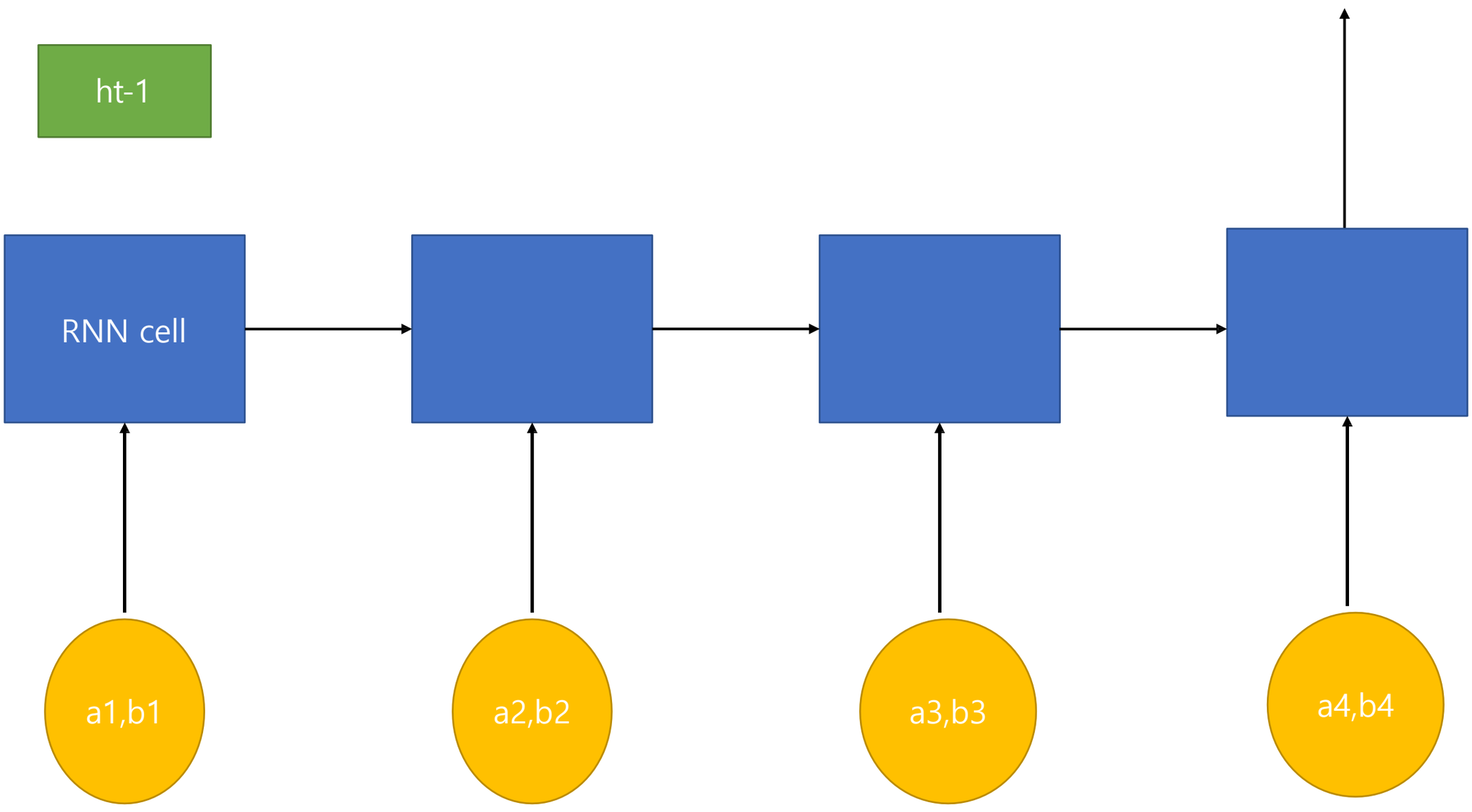
ht = relu(Wxh*Xt + Whh*ht-1 + b)



$$h_t = \tanh(W_{xh} * x_t + W_{hh} * h_{t-1} + b)$$

4, 3, 1
(batch size, time, feature size)

ht = relu(Wxh*Xt + Whh*ht-1 + b)



$$h_t = \tanh(W_{xh} * x_t + W_{hh} * h_{t-1} + b)$$

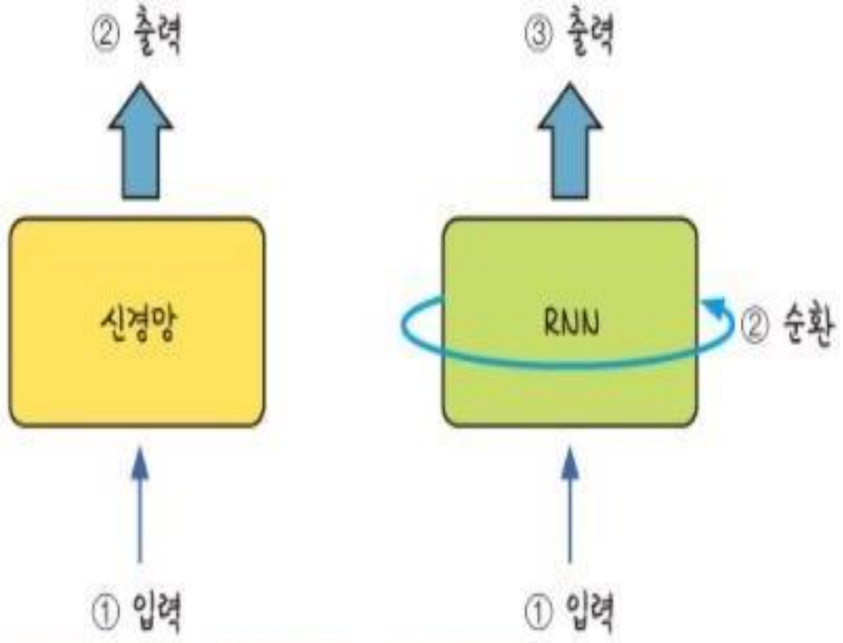


그림 17-1 일반 신경망과 순환 신경망의 차이

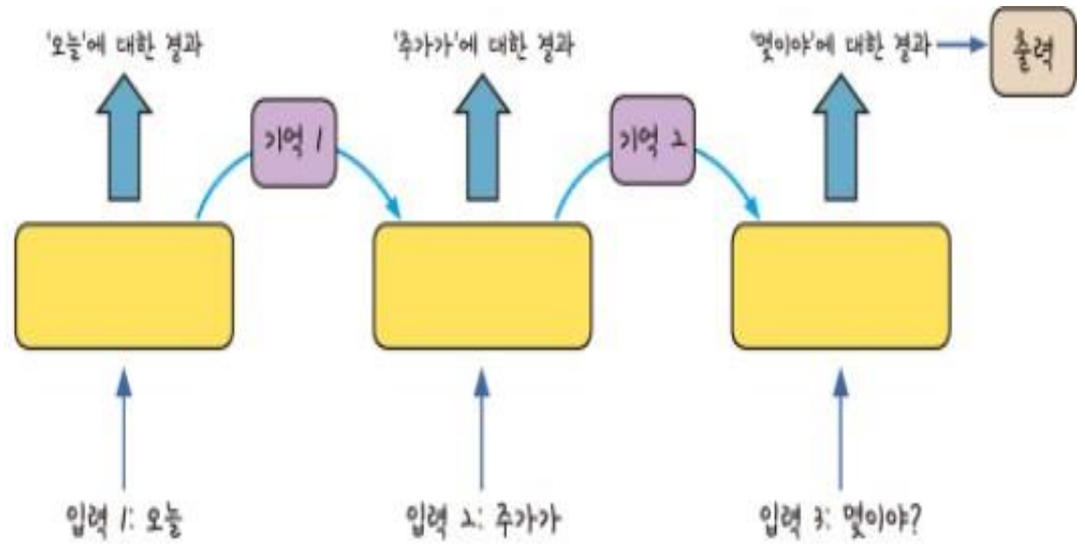


그림 17-2 "오늘 주가가 몇이야?"를 RNN이 처리하는 방식

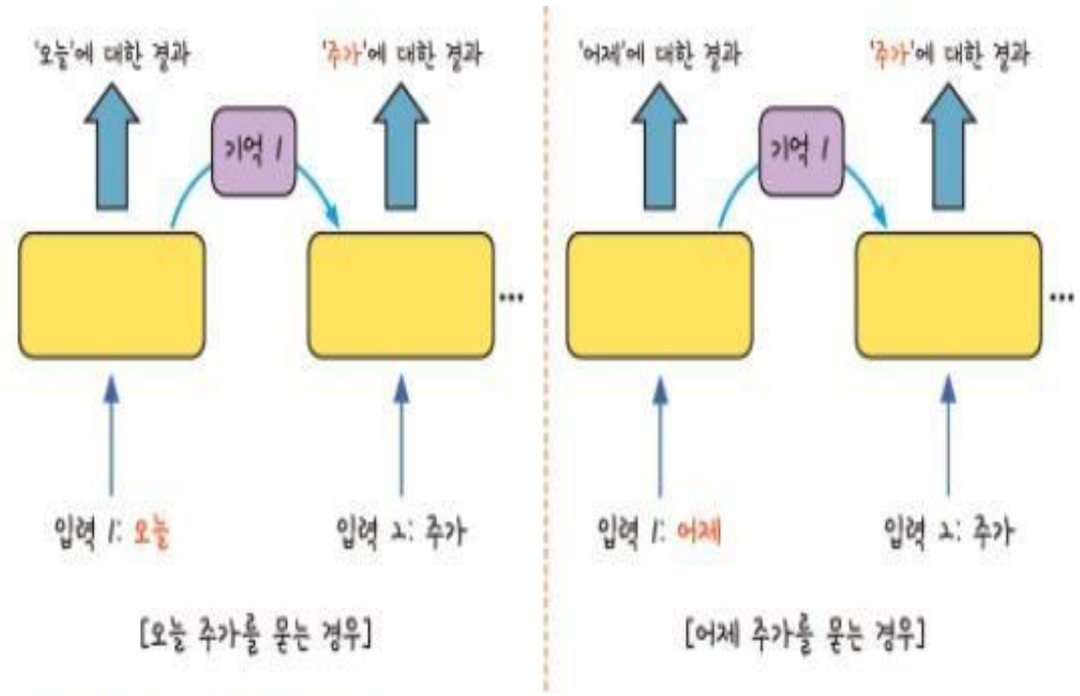
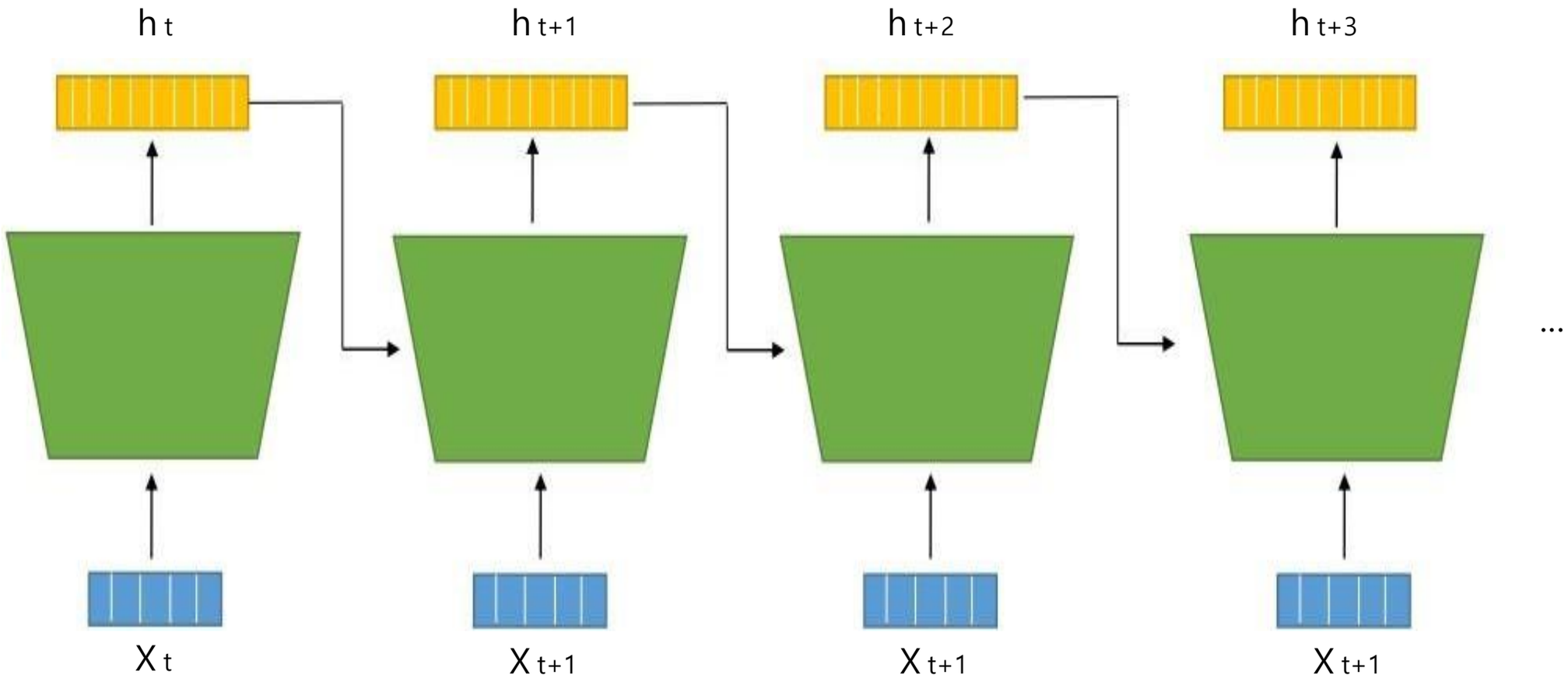


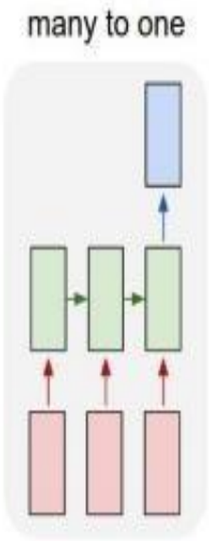
그림 17-3 RNN을 사용하는 이유

Recurrent Neural Network

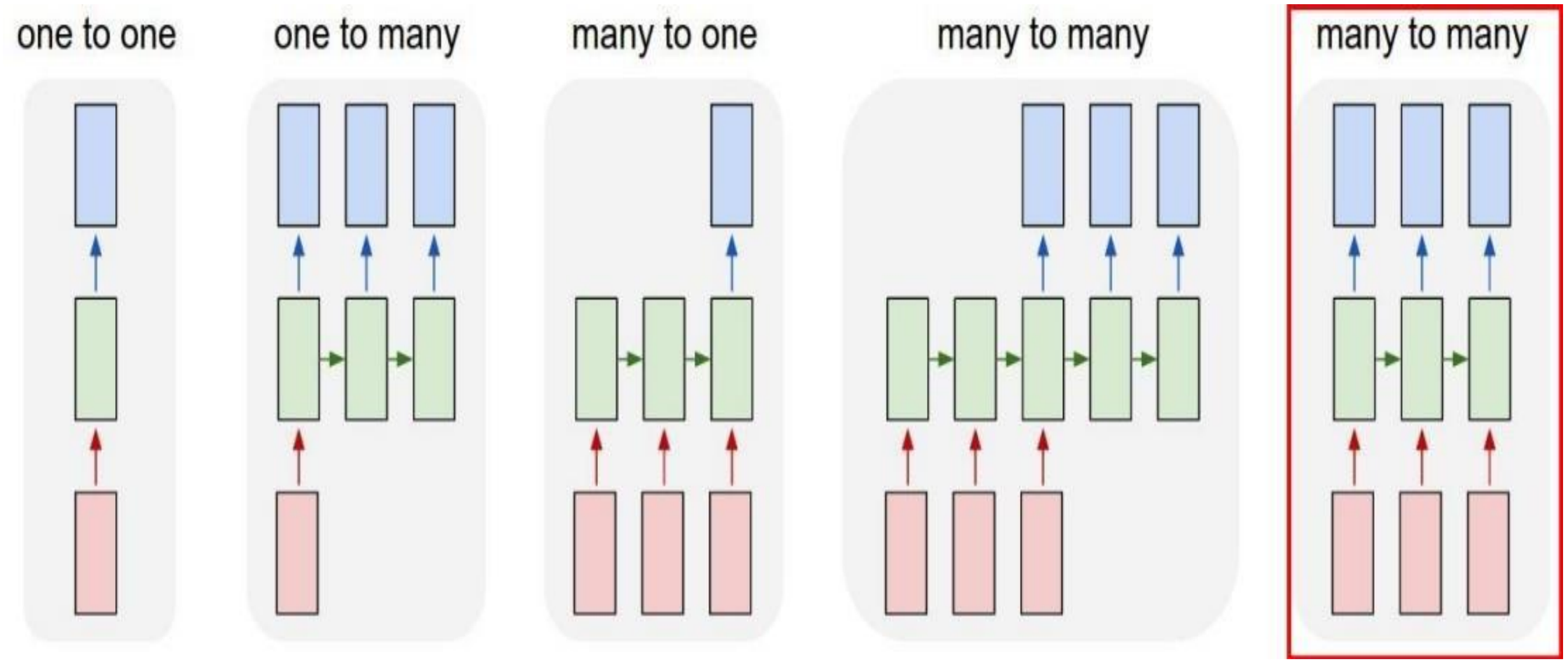
Process both new inputs and model output of previous input!



Recurrent Neural Network

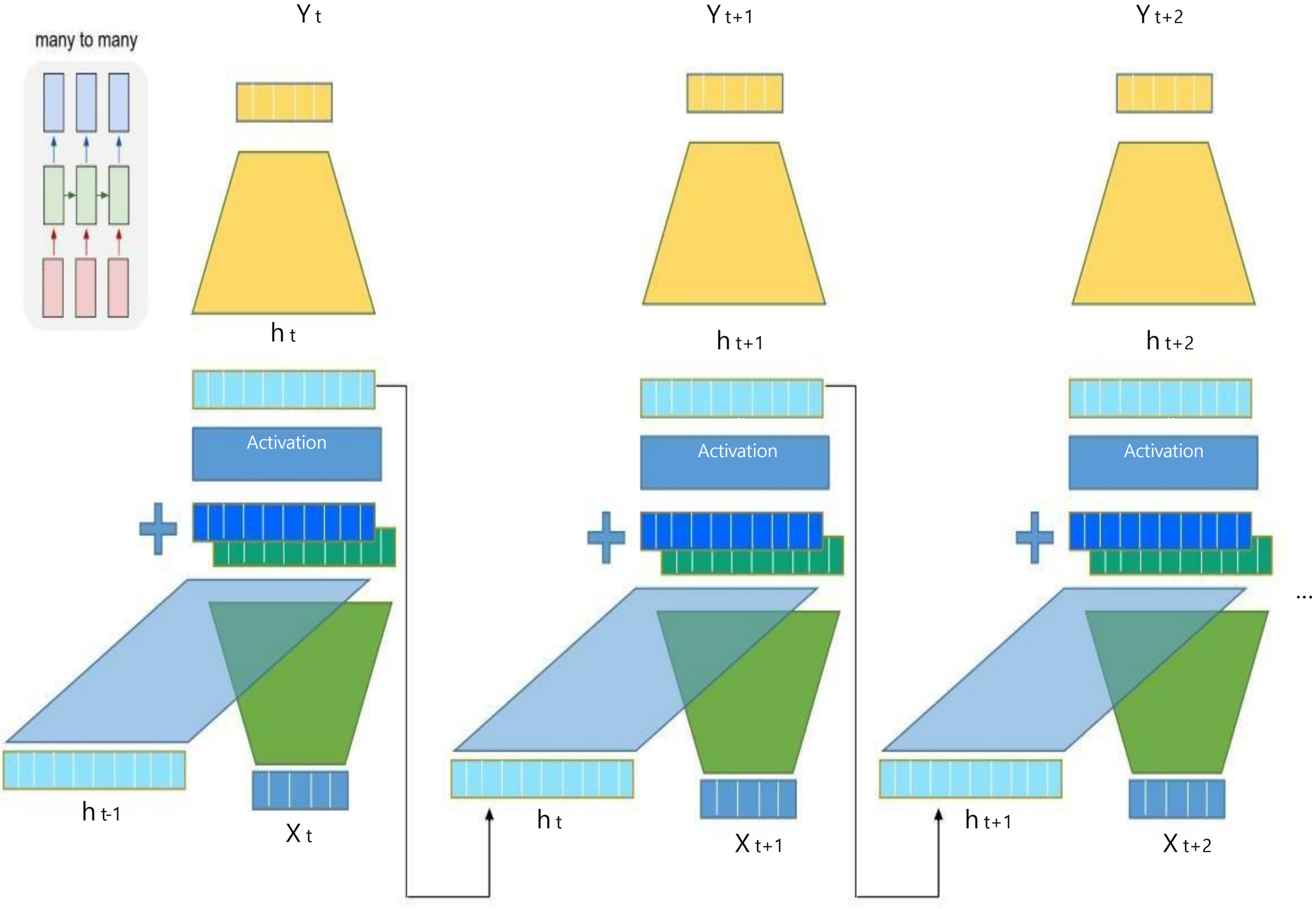


Types of Task Dealing with Sequential Data

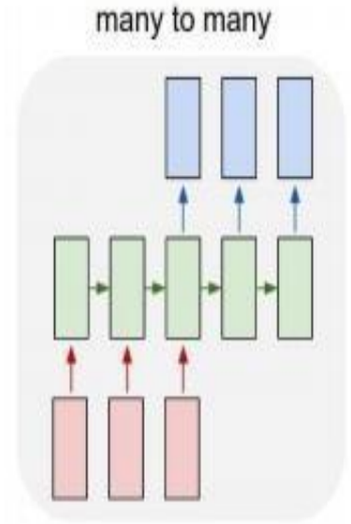


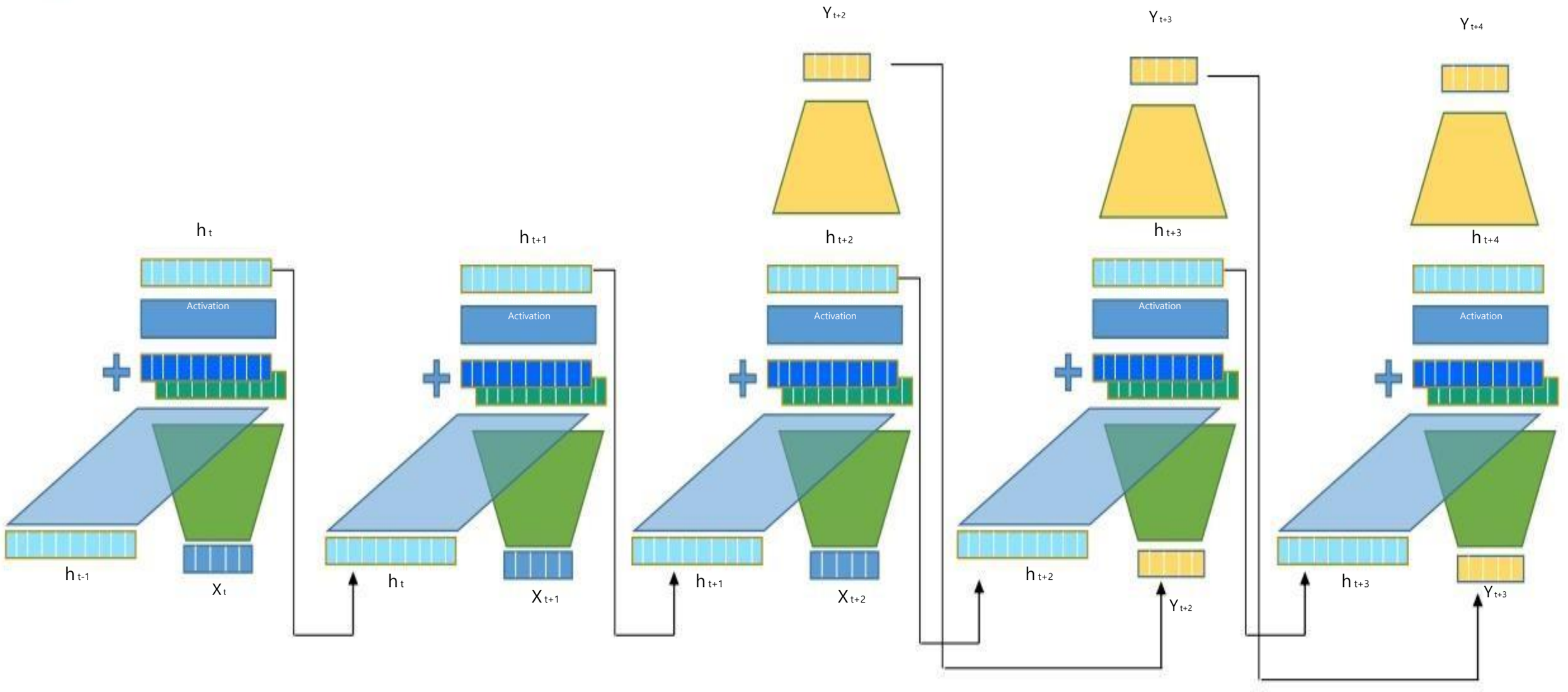
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Network



Recurrent Neural Network





Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

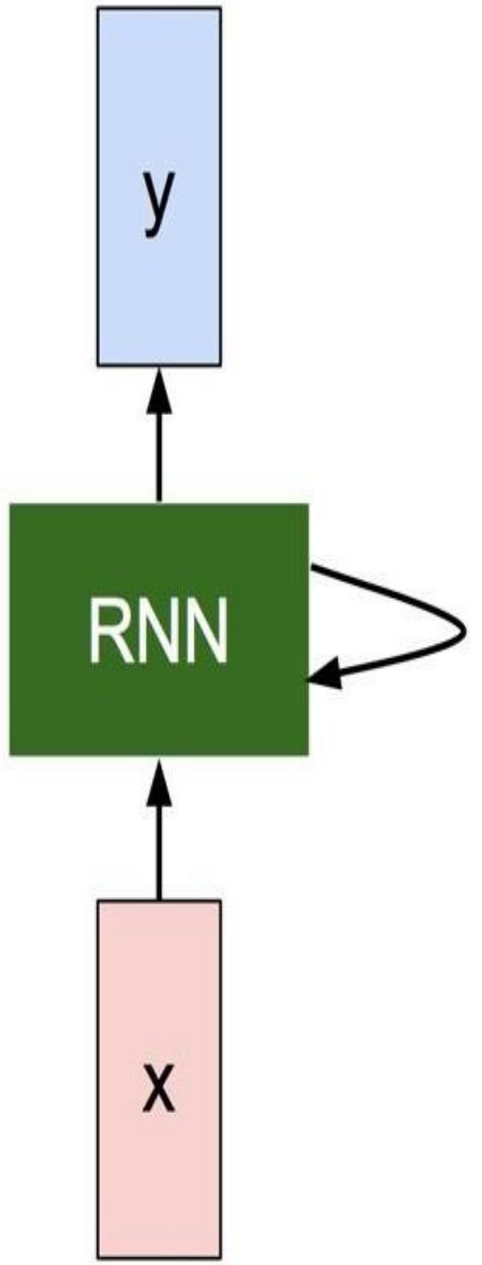
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

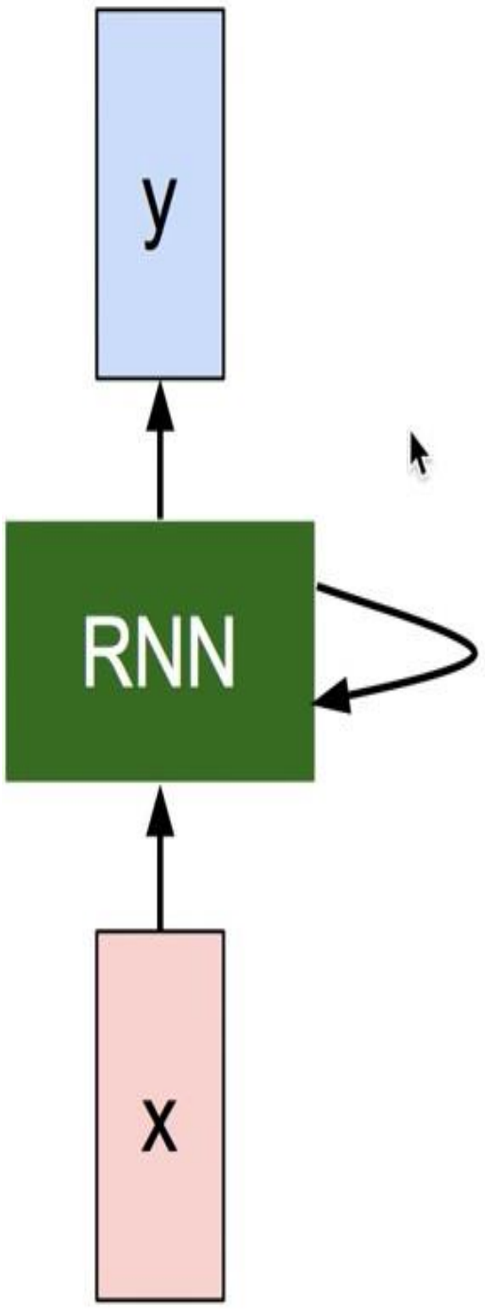
old state

input vector at some time step



(Vanilla) Recurrent Neural Network

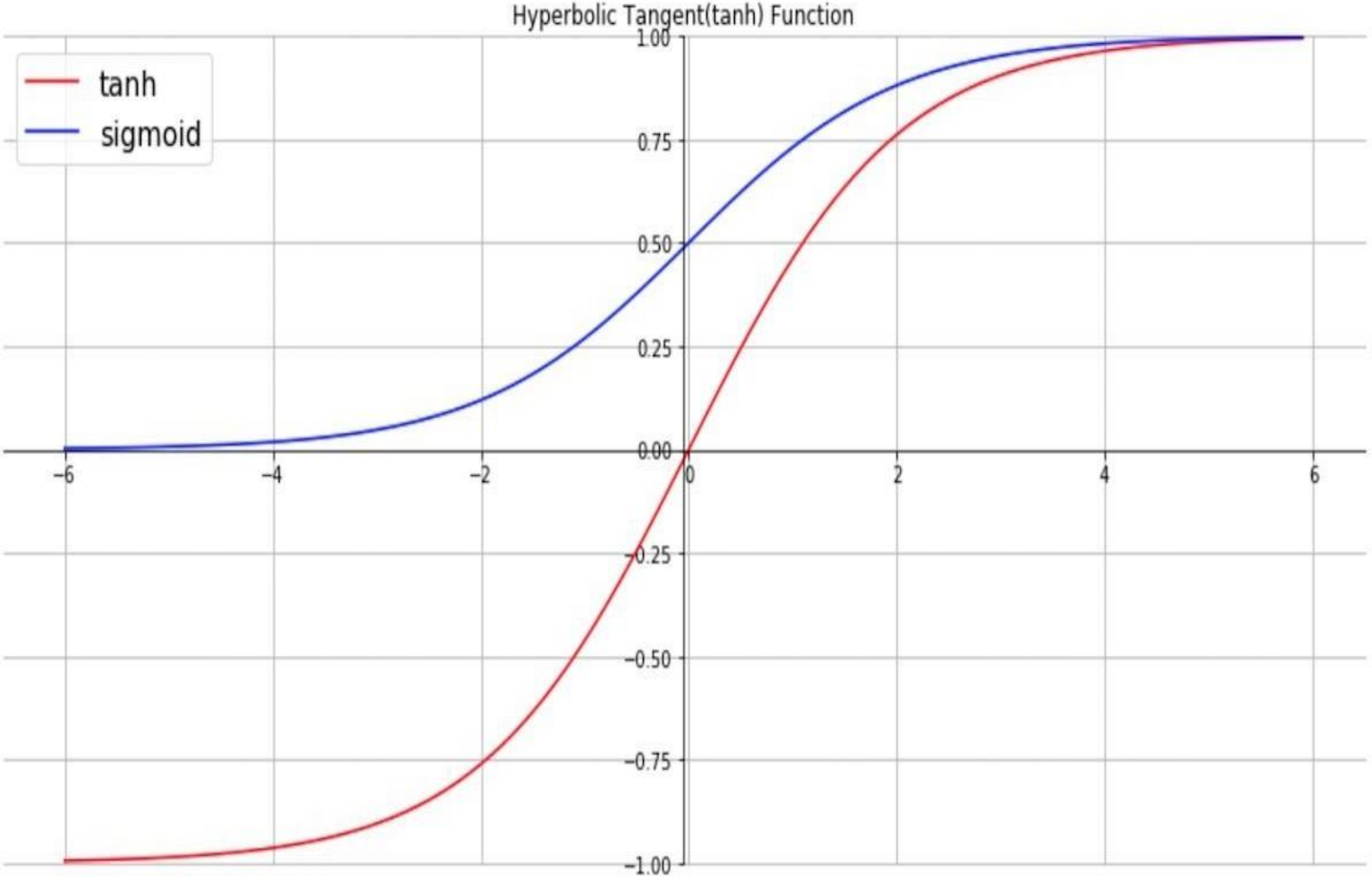
The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

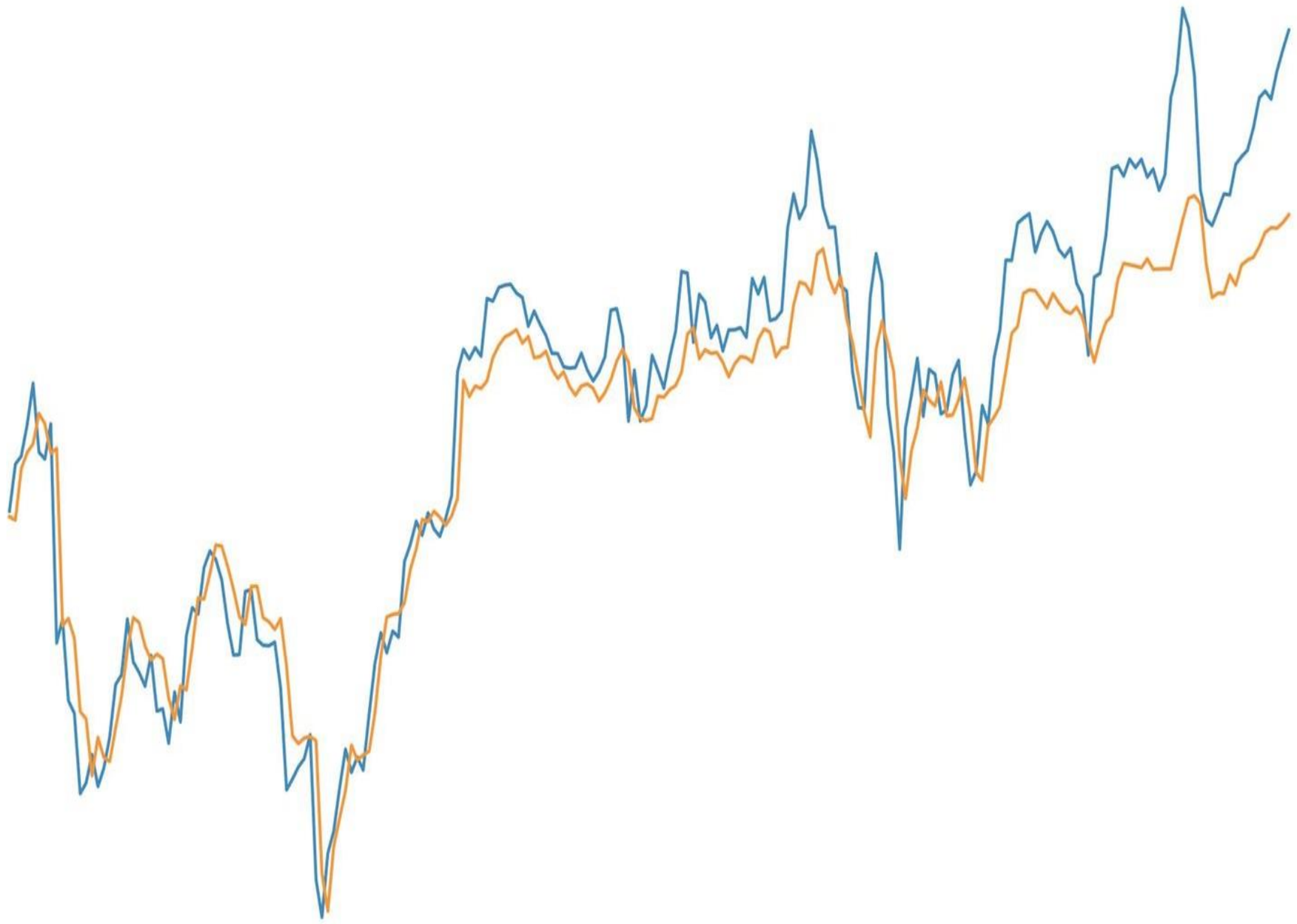
$$y_t = W_{hy}h_t$$


Hyperbolic Tangent(tanh) Function
tanh
sigmoid
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−6
−4
−2
0
2
4
6

LSTM, GRU

- 여러개의 데이터가 순서대로 입력되♘을때 앞서 입력받은 데이터를 잠시 기억해 놓는 방법
- 기억된 데이터의 중요성을 판단하여 별도의 가중치를 줘서 다음 데이터로 넘긴다.
- 모든 입력 값에 이 작업이 순서대로 이뤄지며 같은 층을 맴도는 것으로 보여 순홖 싞경망이라 부른다

Time series data

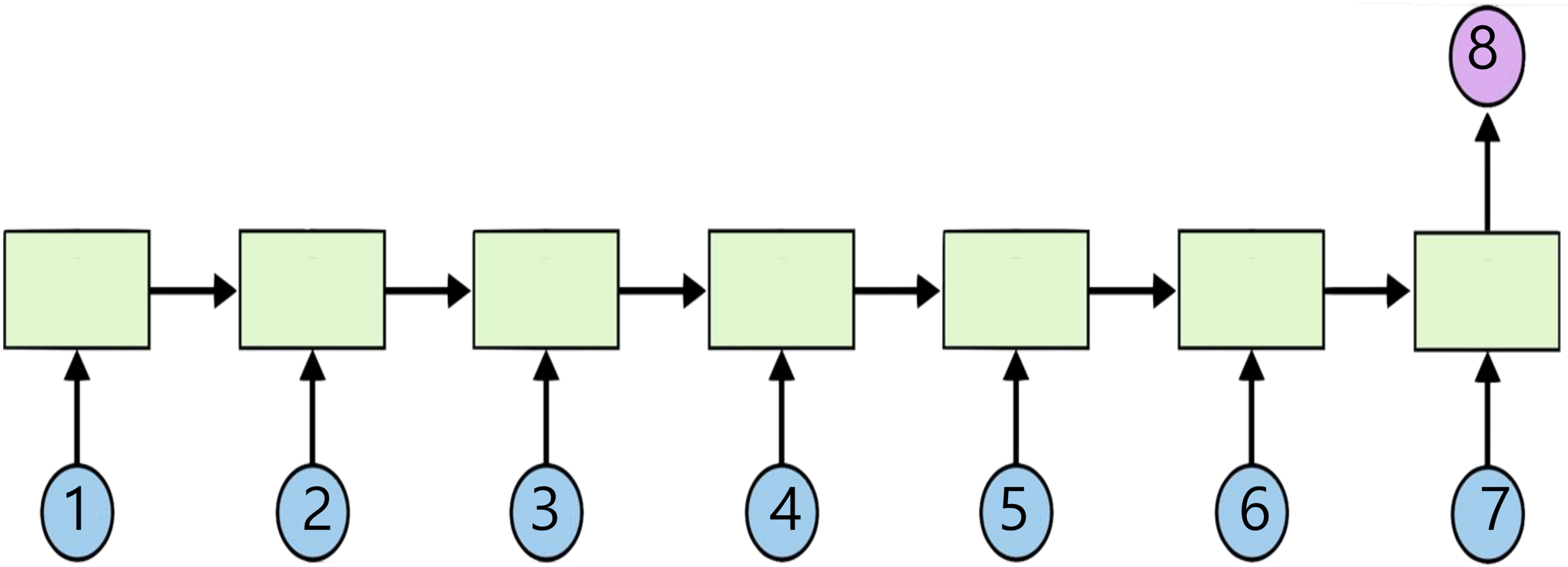


Time series data

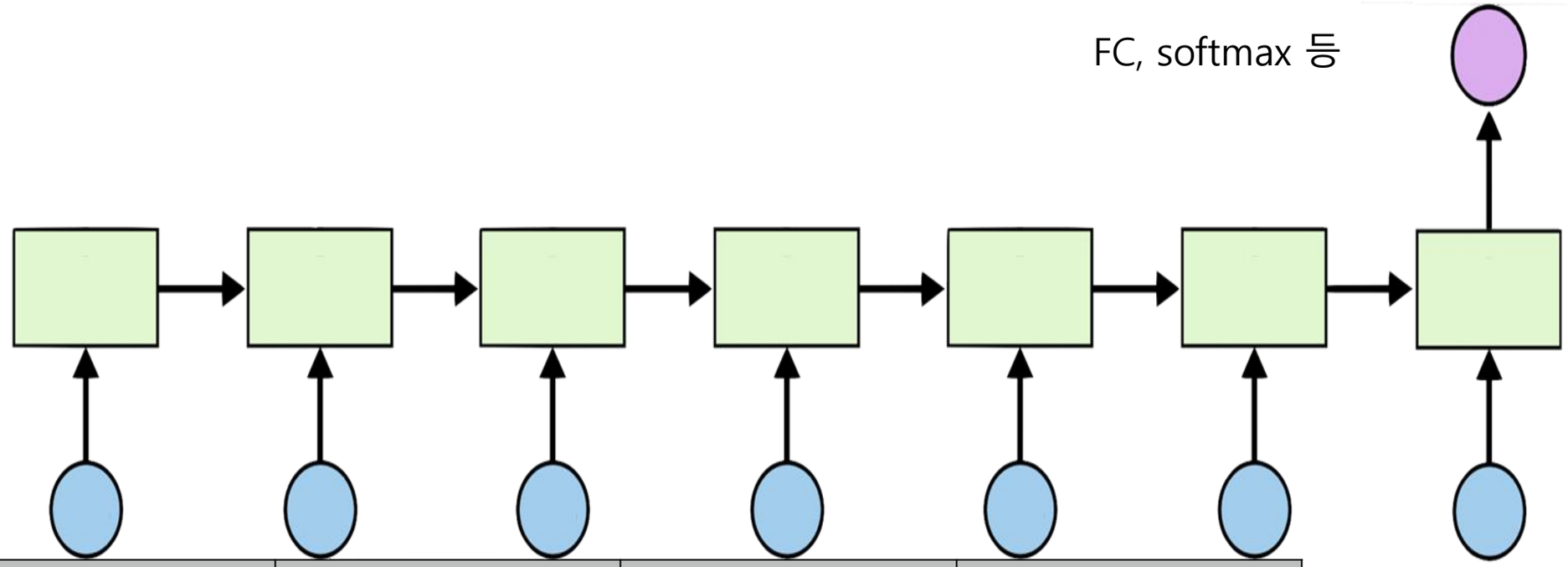
Open	High	Low	Volume	Close
828.659973	833.450012	828.349976	1247700	**831.659973**
823.02002	828.070007	821.655029	1597800	**828.070007**
819.929993	824.400024	818.97998	1281700	**824.159973**
819.359985	823	818.469971	1304000	**818.97998**
819	823	816	1053600	**820.450012**
816	820.958984	815.48999	1198100	**819.23999**
811.700012	815.25	809.780029	1129100	**813.669983**
809.51001	810.659973	804.539978	989700	**809.559998**
807	811.840027	803.190002	1155300	**808.380005**

'data-02-stock_daily.csv'

Many to one



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-12-5-rnn_stock_prediction.py



Open	High	Low	Volume	Close
828.659973	833.450012	828.349976	1247700	**831.659973**
823.02002	828.070007	821.655029	1597800	**828.070007**
819.929993	824.400024	818.97998	1281700	**824.159973**
819.359985	823	818.469971	1304000	**818.97998**
819	823	816	1053600	**820.450012**