AI School 6기 1주차

파이썬 기초1

딥러닝 기초 이론1

텐서플로우 기초

AI School 6기 1주차

파이썬 기초1

파이썬이란?

- 파이썬(Python)은 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어

- 인터프리터 언어: 한 줄씩 소스 코드를 해석해서 그때그때 실행해 결과를 바로 확인할 수 있는 언어

- 구글에서 만든 소프트웨어의 50%이상이 파이썬으로 작성

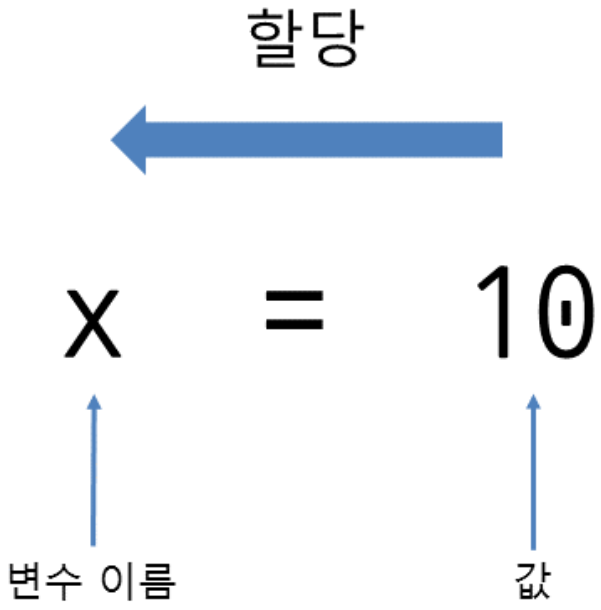
파이썬의 특징

1. 파이썬은 인간다운 언어이다
2. 파이썬은 문법이 쉬워 빠르게 배울 수 있다
3. 파이썬은 무료이지만 강력하다
4. 파이썬은 간결하다
5. 파이썬은 프로그래밍을 즐기게 해준다
6. 파이썬은 개발 속도가 빠르다

파이썬의 자료형 - 숫자

- 자료형이란 프로그래밍을 할 때 쓰이는 숫자, 문자열 등 자료 형태로 사용하는 모든 것을 뜻함
- 변수: 메모리에 값을 저장하기 위해 할당하는 공간
  (할당 후 내부의 값 변경 가능)

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8진수	0o34, 0o25
16진수	0x2A, 0xFF



파이썬의 자료형 - 숫자

- 숫자형(Number)이란 숫자 형태로 이루어진 자료형

```
# int (정수형)

a = 3
print(a)

b = 2
print(b)

c = 0
print(c)

a = -12
print(a)
```


```
# float (실수형)
b = 3.4
print(b)

a = -2.5
print(a)

a = 3.5e-3
print(a)

a = 2e+3
print(a)
```


```
# float -> int

a = 3.7
print(a)

b = int(a)
print(b)

# int ->float
print(float(b))

# type()
print(type(a))
print(type(b))
```

파이썬의 자료형 - 숫자

- 숫자형을 활용하기 위한 연산자들

```
# 사칙연산
a = 3
b = 5
c = a + b
print(a + b)
print(c)
print(a - b)
print(a*b)
print(a/b)

a = 3.0
print(a + b)
```

```
# 제곱, 나머지, 몫
a = 5
b = 2

print(a**b)
print(a%b)
print(a//b)
```

연습문제 - 숫자

- 언어 = 90, 영어 = 60, 수학 = 81
- 위 학생의 평균 성적을 구하는 코드를 작성하세요.
- (average)

.

.

.

.

print(average)

파이썬의 자료형 - 문자열

- 문자열(String)이란 문자, 단어 등으로 구성된 문자들의 집합

```
print("Hello World")
print('Hello World')
print("a")
print('123', end=" ")
print('456')
```

```
c = "Kangmin's paper"
print(c)
d = 'He said "Hi"'
print(d)
c = 'Kangmin₩'s
paper'
print(c)
d = "He said ₩"Hi₩""
print(d)
```

```
# multiline
multiline = "Life is too short₩nYou need
python"
print(multiline)

multiline = """
Life is too short
You need python
"""
print(multiline)

multiline = '''
Life is too short
You need python
'''
print(multiline)
```

파이썬의 자료형 - 문자열

- 문자열 연산
- 문자열 인덱싱
- 문자열 슬라이싱

```
teacher = “Kim's "
title = "AI School"
print(teacher + title)
print("="*30)

print(len(title))

print(title[0])
print(title[-1])
print(title[:2])
print(title[3:])
```

```
odd_even = "홀짝홀짝홀짝"
print(odd_even[::2])
print(odd_even[1::2])
```

AI School

0 123 -1

파이썬의 자료형 - 문자열

- 문자열 관련 함수

```
a = "apple"
print(a.count("p"))
print(a.find("p"))
print(a.index("p"))
print(".".join(a))
a = a.upper()
print(a)
print(a.lower())
```

```
num = "234"
print(num)
num = float(234)
print(num)
num = str(num)
print(num)
```

```
b = " How can I improve my coding skills?
"
print(b)

b = b.strip()
print(b)

b = b.replace("?", "")
print(b)

word_list = b.split(" ")
print(word_list)
```

파이썬의 자료형 – 문자열 포매팅

```
apple_num = 4
orange_num = 2
apple_num_string = "three“

print("I eat %d apples." % apple_num)
print("I eat {0} apples.".format(apple_num))
print("I eat %s apples." % apple_num_string)
print("I eat %d apples and %d oranges." % (apple_num, orange_num))
print("I eat {0} apples and {1} oranges.".format(apple_num, orange_num))
print("I eat {apple_num} apples and {orange_num}
oranges.".format(apple_num=1, orange_num=2))

print("Error is %d%%." % 98)

print(f'I eat {apple_num} apples.')
pi = 3.141592
print("pi = %f" % pi)
print("pi = %0.4f" % pi)
```

연습문제 - 문자열

- Mary's cosmetics 을 출력하세요.
- "dk2jd923i1jdk2jd93jfd92”의 길이를 구하세요.
- t1 = 'python’, t2 = 'java’일 때 문자열 더하기와 곱하기를 이용하여 ”python java python java python java”를 출력 하세요.
- id = "890910-1157963"에서 성별을 나타내는 수를 출력하세요.
- license_plate = "24가 2210”에서 번호판 뒷자리만 출력하세요.
- url = portal.ac.kr 에서 kr만 출력하세요. (split 함수 사용)

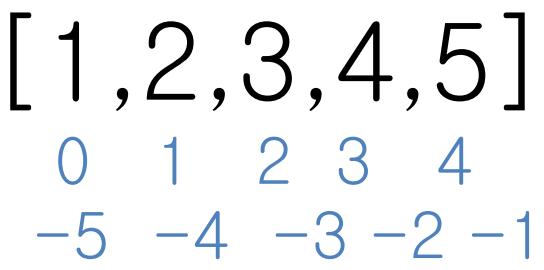
- dha8102@naver.com

파이썬의 자료형 – 리스트

- 리스트 생성
- 리스트 인덱싱과 슬라이싱

```
# 리스트 (List) 사용법
a = [1, 2, 3, 4, 5]
print(a)
print(len(a))

b = ["a", "b", "c", "d"]
print(b)
print(len(b))

c = [1, "a", 2.5]
print(c)
print(len(c))

d = [1, 2, [3, 4, 5]]
print(d)
print(len(d))
```

```
# 리스트 인덱싱과 슬라이싱
print(a[0])
print(a[1])
print(a[-1])
print(a[0:2])
print(a[:2])
print(a[2:])
print(a+b)
print(a*3)
print(len(a))


print(d[2][:2])
```



파이썬의 자료형 – 리스트

- 리스트 관련 함수

```
a = [1, 2, 3, 4, 5]
a[2] = 6
print(a)
print(a.index(4))
del a[2]
print(a)
del a[2:]
print(a)
a.append(2)
print(a)
print(a.count(2))
print(max(a))
print(min(a))
print(sum(a))
```

```
a = [2, 1, 5, 4, 3]
b = sorted(a)
print(a)
print(b)
a.sort()
print(a)

a.reverse()
print(a)
a.insert(0, 6)
print(a)
a.remove(6)
print(a)
print(a.pop())
print(a)
a.extend([1, 0])
print(a)
```

연습문제 - 리스트

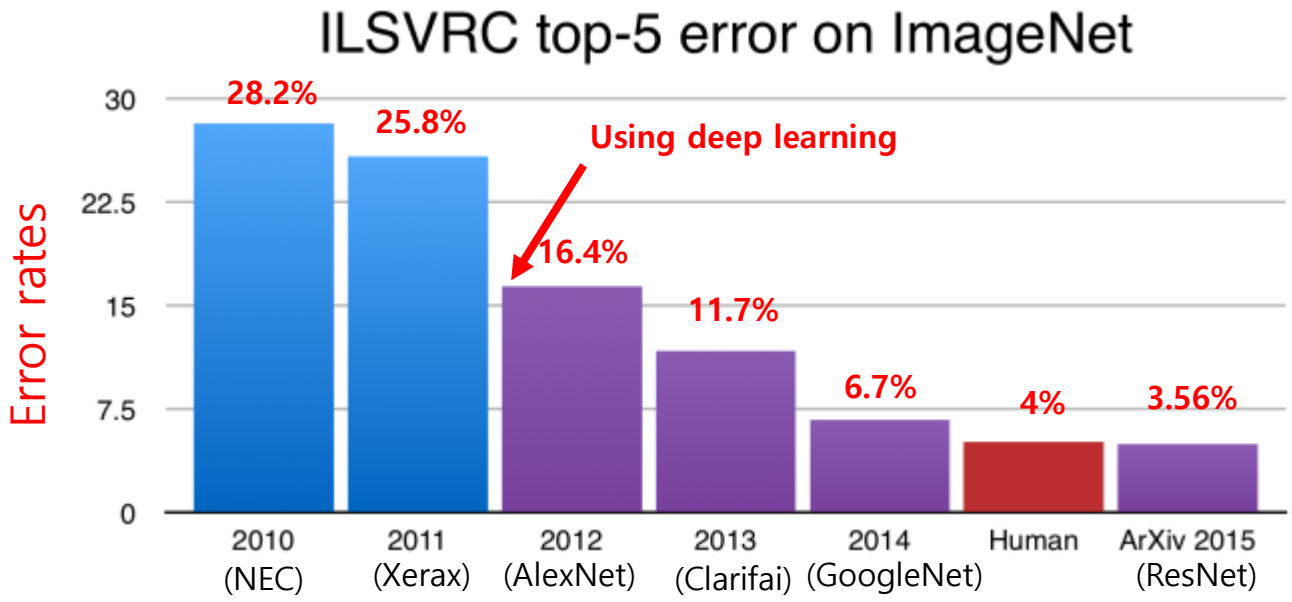
- language1 = ["C", "C++", "JAVA"], language2 = ["Python", "Go", "C#"] 두 리스트의 원소를 모두 갖는 languages를 만드세요.
- nums = [12, 245, 33, 77, 858]의 평균을 구하세요.
- a = ["b", "a", "d", "c"] 리스트를 알파벳 순으로 정렬하세요.

AI School 6기 1주차

딥러닝 기초 이론

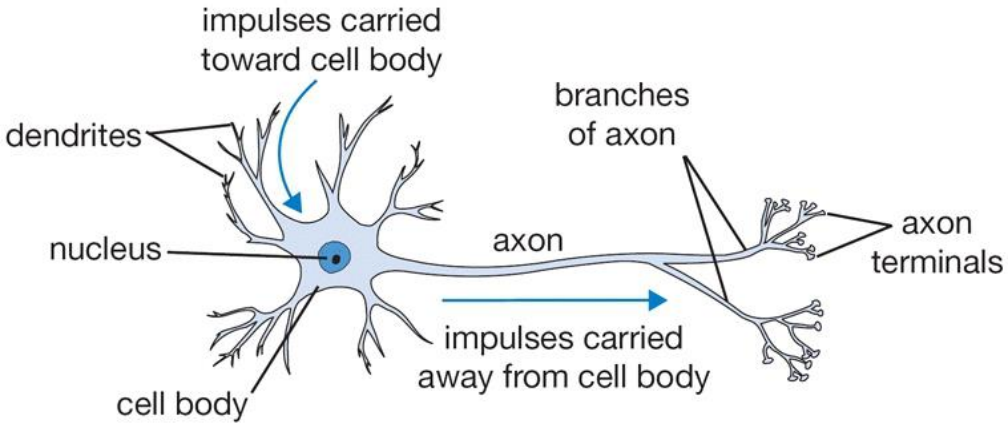
Why deep learning?

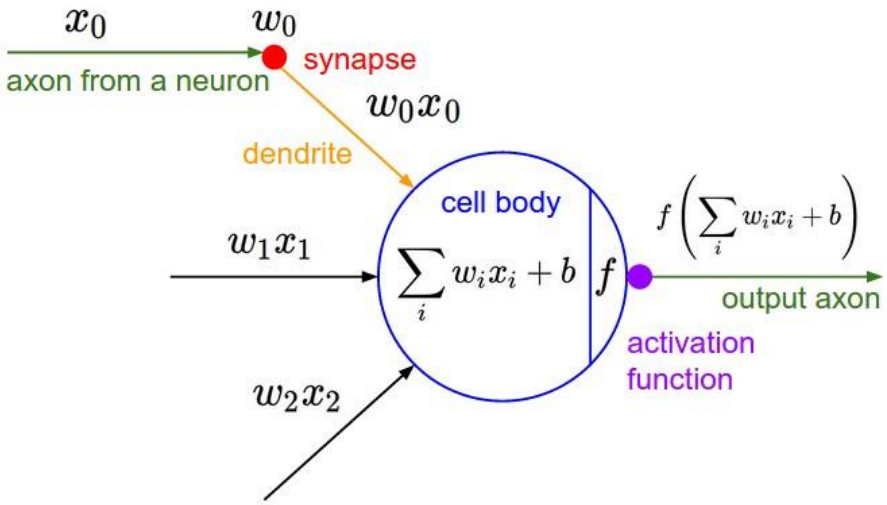
- **Visual Recognition**



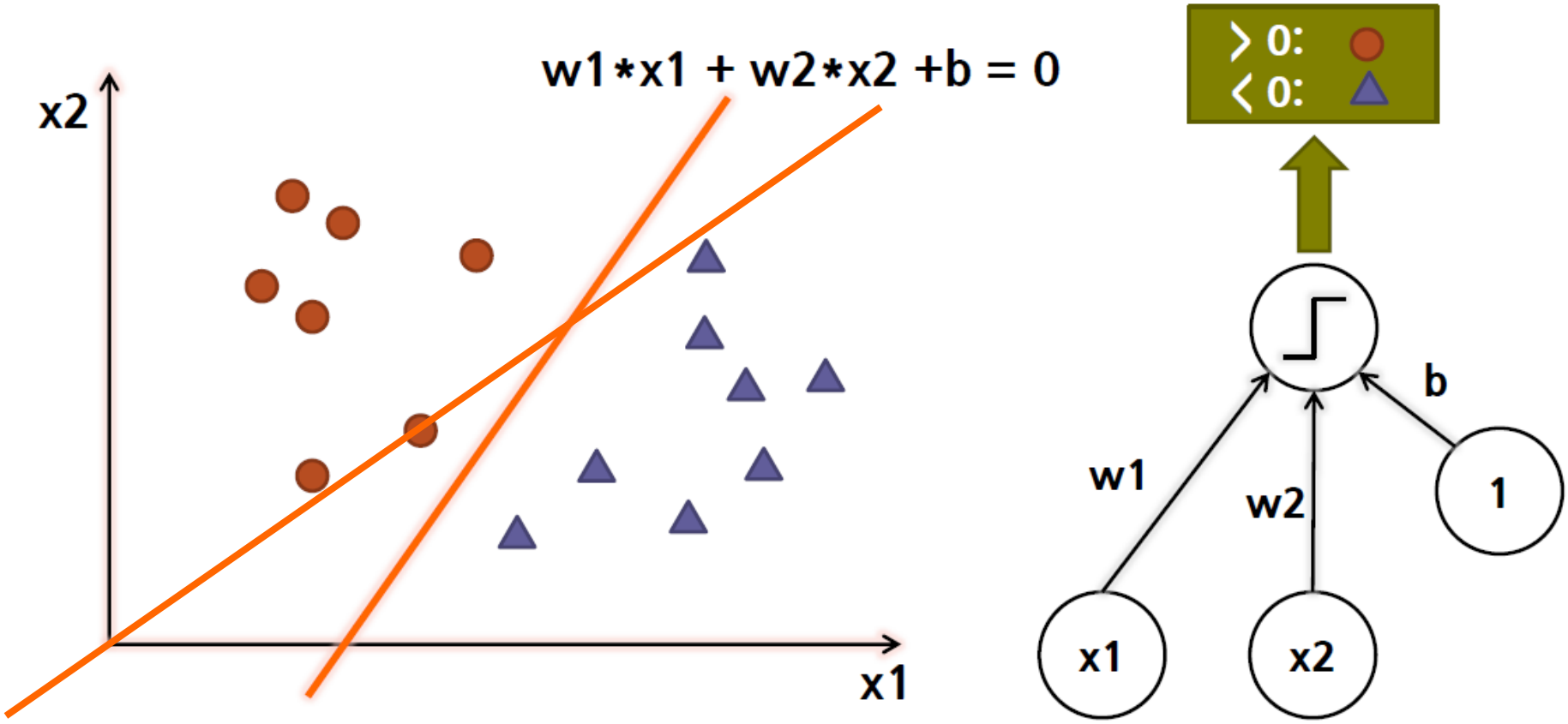
* ILSVRC – ImageNet Large Scale Visual Recognition Challenge

Perceptron

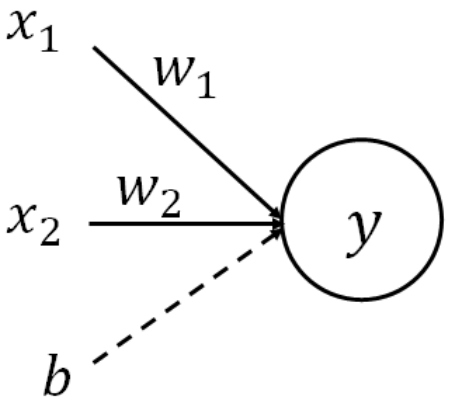

impulses carried
toward cell body
dendrites
branches
of axon
nucleus
axon
axon
terminals
impulses carried
away from cell body
cell body


x_0
w_0
axon from a neuron
synapse
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation
function
$w_2 x_2$

Perceptron (1958~)

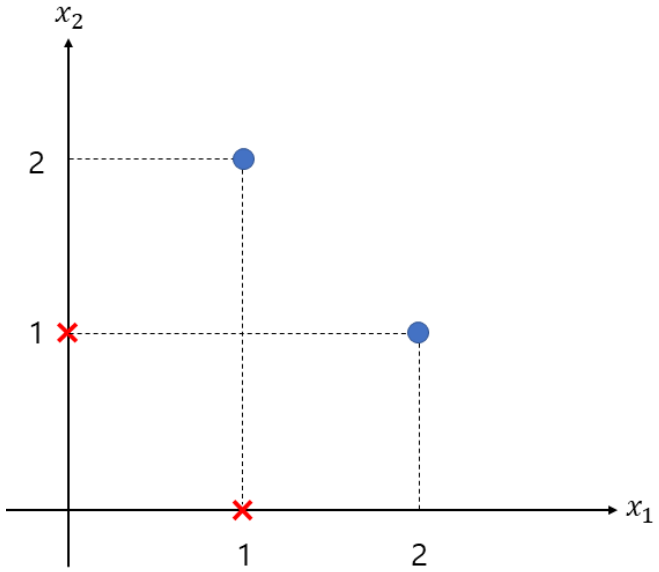


Perceptron (1958~)



$$y = f(w_1 x_1 + w_2 x_2 + b)$$

$$f(x) = \begin{cases} 1 \ (x \geq 0) \\ 0 \ (x < 0) \end{cases}$$

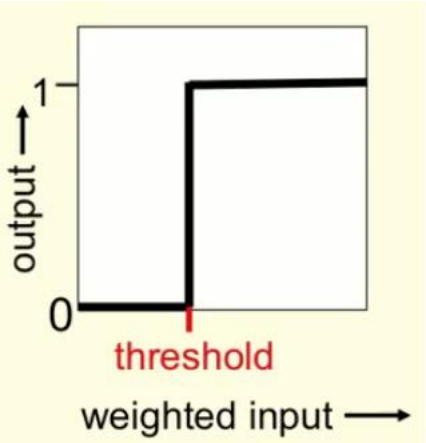


w_1	1
w_2	1
b	-2
f	단위 계단 함수

Activation Function

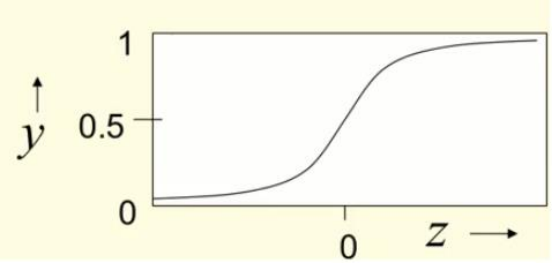
- 입력이 작을 때의 출력은 0에 가깝고, 입력이 커지면 출력이 1에 가까워지는 구조 (ReLU는 입력값을 출력)
- 즉, 입력이 중요하면 큰 값을 출력하고 입력이 중요하지 않으면 작은 값을 출력

Step function



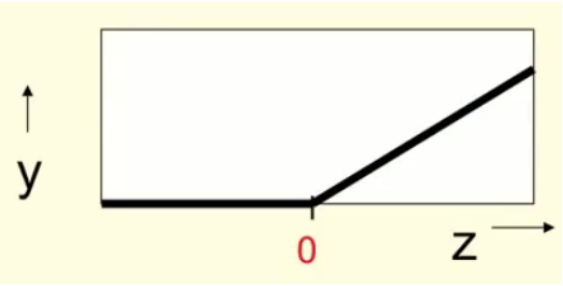
- 퍼셉트론에서 사용
- 미분 값 0

Sigmoid function



- 입력의 절대값이 포화하여 일정값을 가짐
- 그 사이의 값에 대해서는 출력이 서서히 매끄럽게 변함
- 생물의 신경세포가 갖는 성질을 모델링

Rectified linear unit



- 단순하고 계산량 적음
- 학습이 빠름
- 최종 결과도 더 좋은 경우가 많음

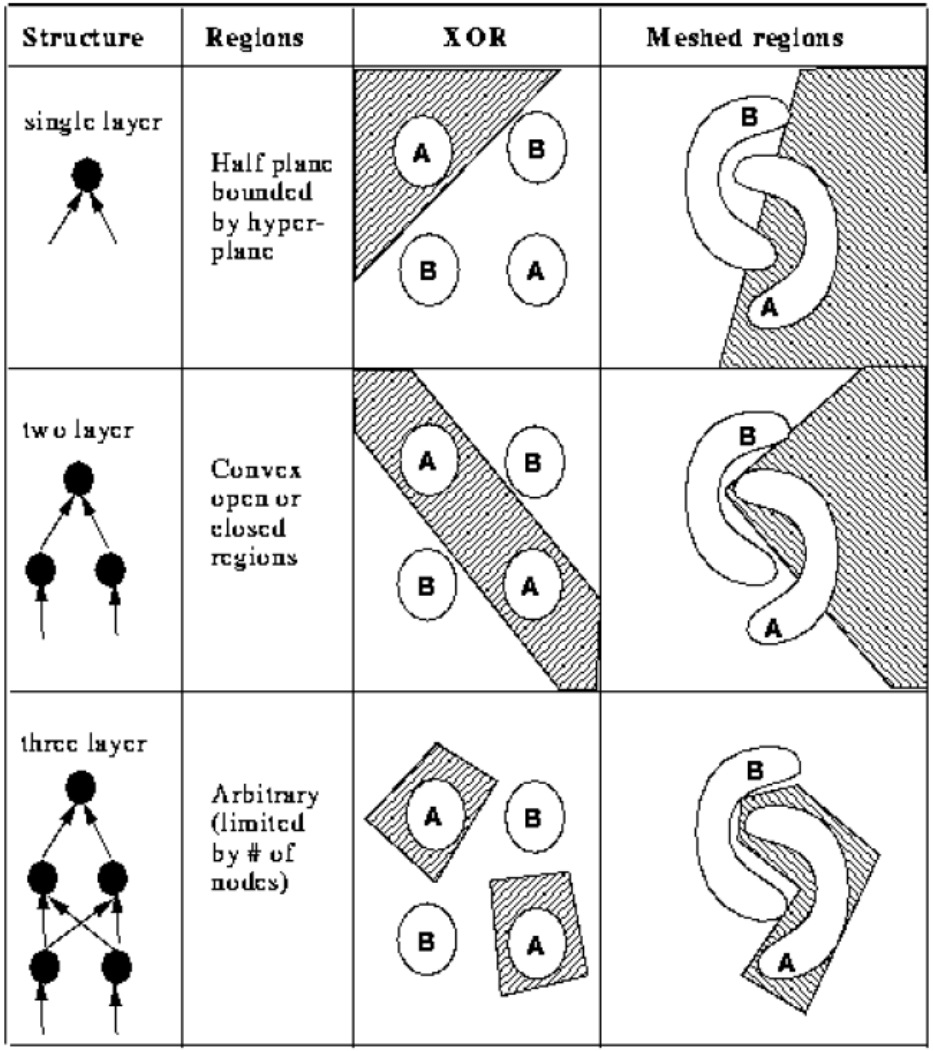
Multiple layer Perceptron(1969)

- Multiple boundaries are needed(e.g. XOR problem)

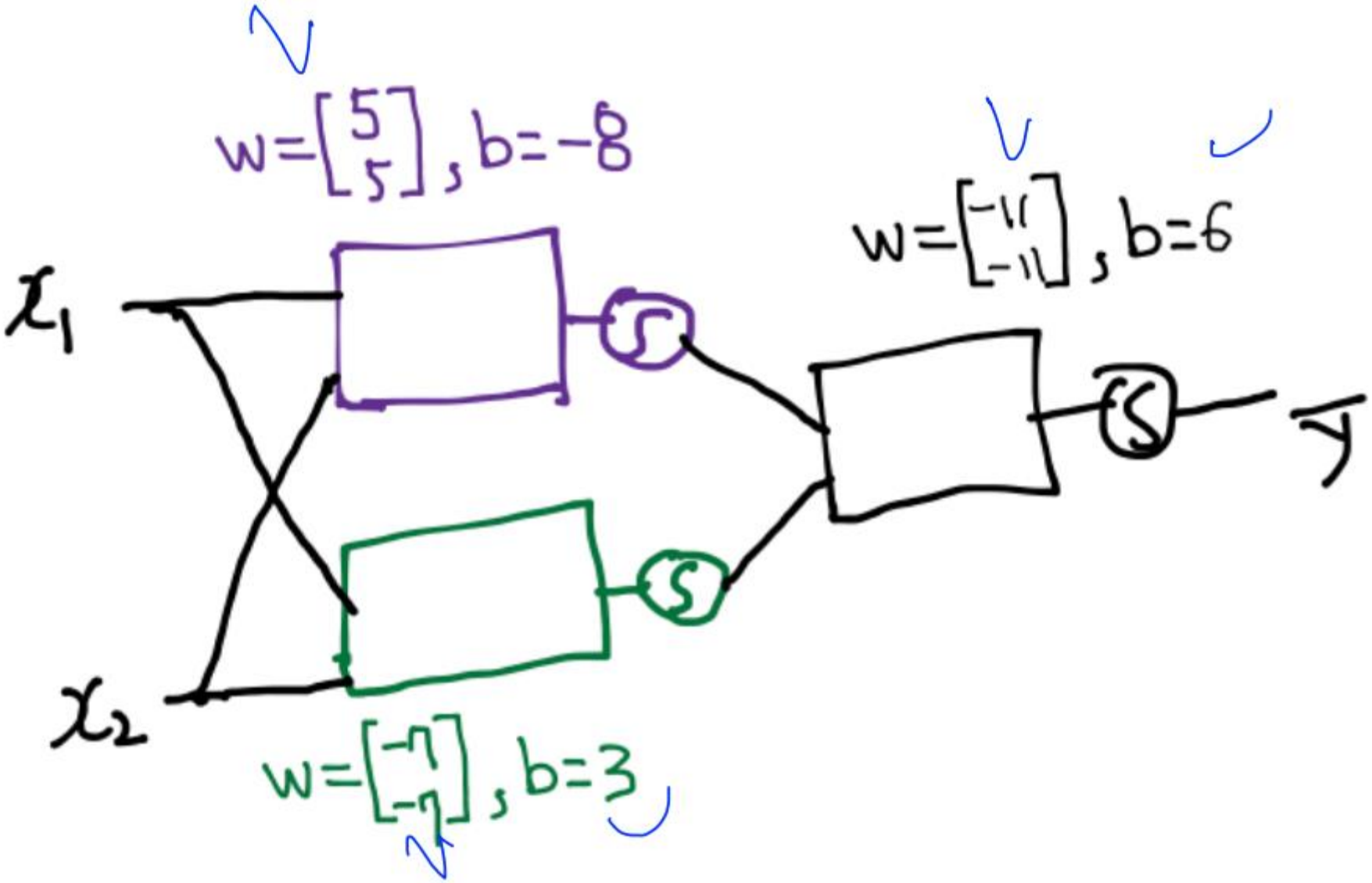
  ➡ Multiple Units

- More complex regions are needed(e.g. Polygons)

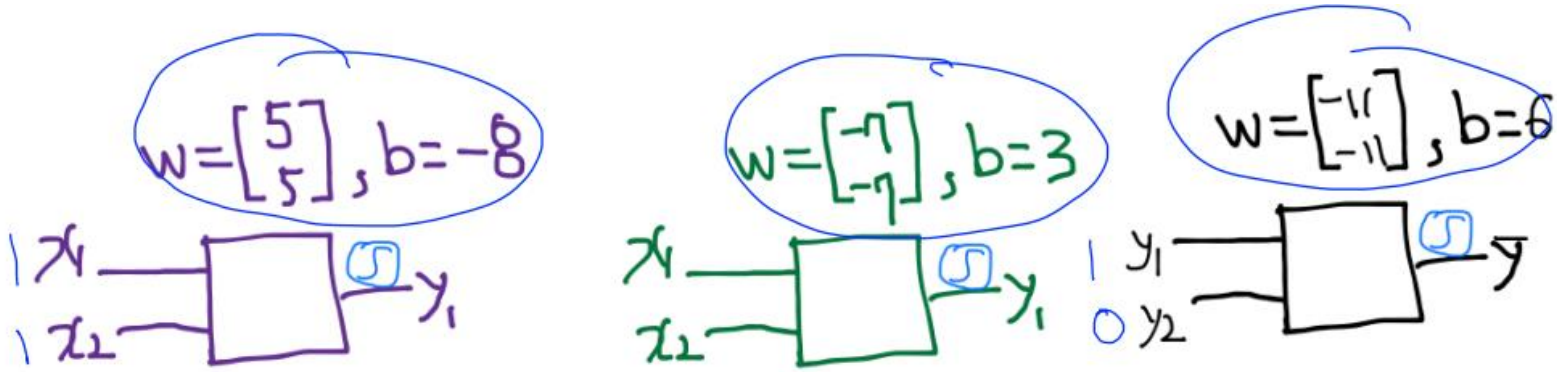
  ➡ Multiple Layers



Multiple layer Perceptron(1969)



Multiple layer Perceptron(1969)



$[1\ 1]\begin{bmatrix}5\\5\end{bmatrix}-8=5+5-8=2$, Sigmoid(2) = 1

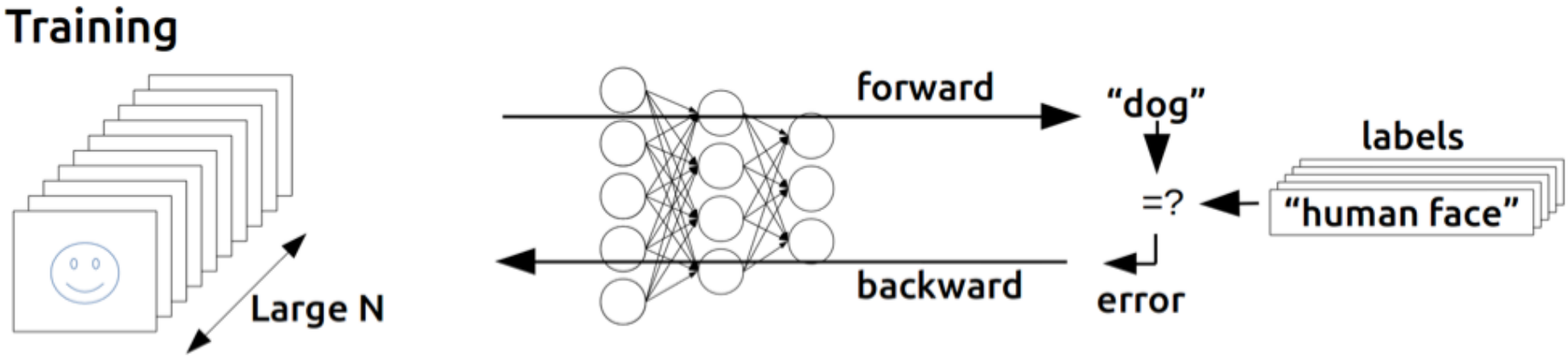
$[1\ 1]\begin{bmatrix}-7\\-7\end{bmatrix}+3=-7+-7+3=-11$, Sigmoid(-11) = 0

$[1\ 0]\begin{bmatrix}-11\\-11\end{bmatrix}+6=-11+0+6=-5$

Sigmoid(-5) = 0

x_1	x_2	y_1	y_2	y	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0

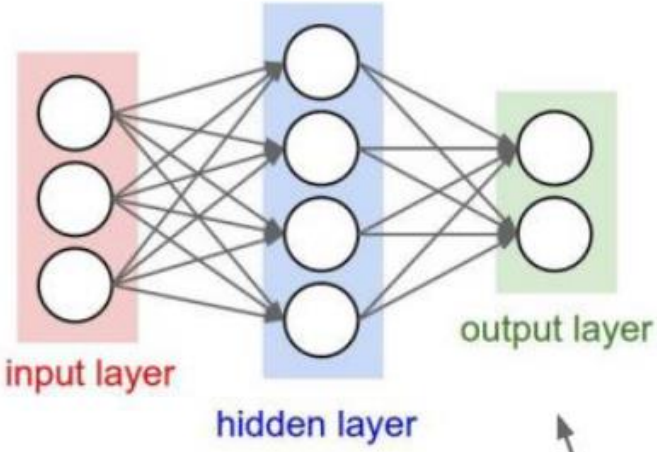
Multiple layer Perceptron(1969)

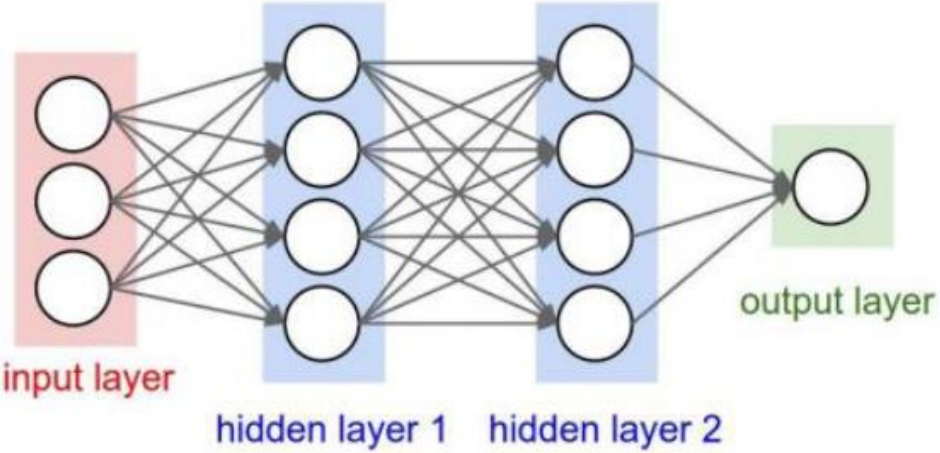


Multiple layer Perceptron(1986~)

of hidden layers <=1
→ Shallow neural network.

# of hidden layers >=2
→ deep neural network





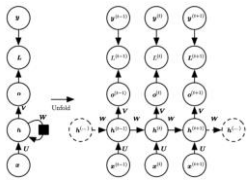
"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

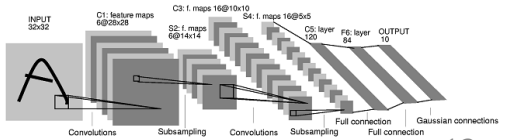
"Fully-connected" layers

word2vec

RNN

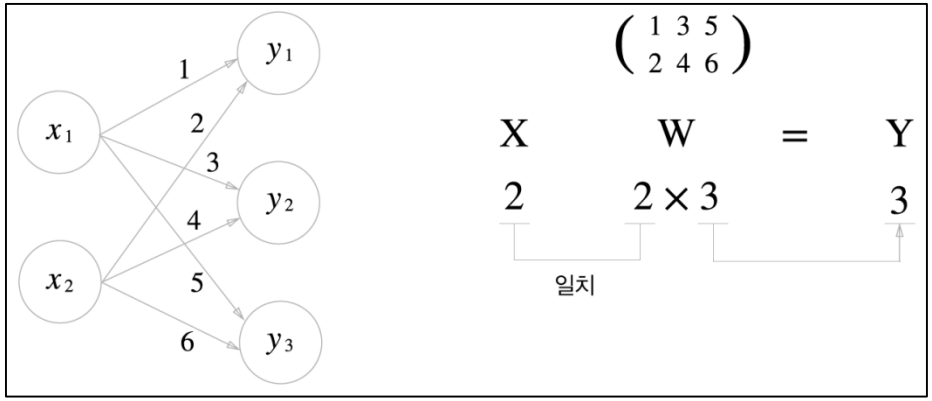


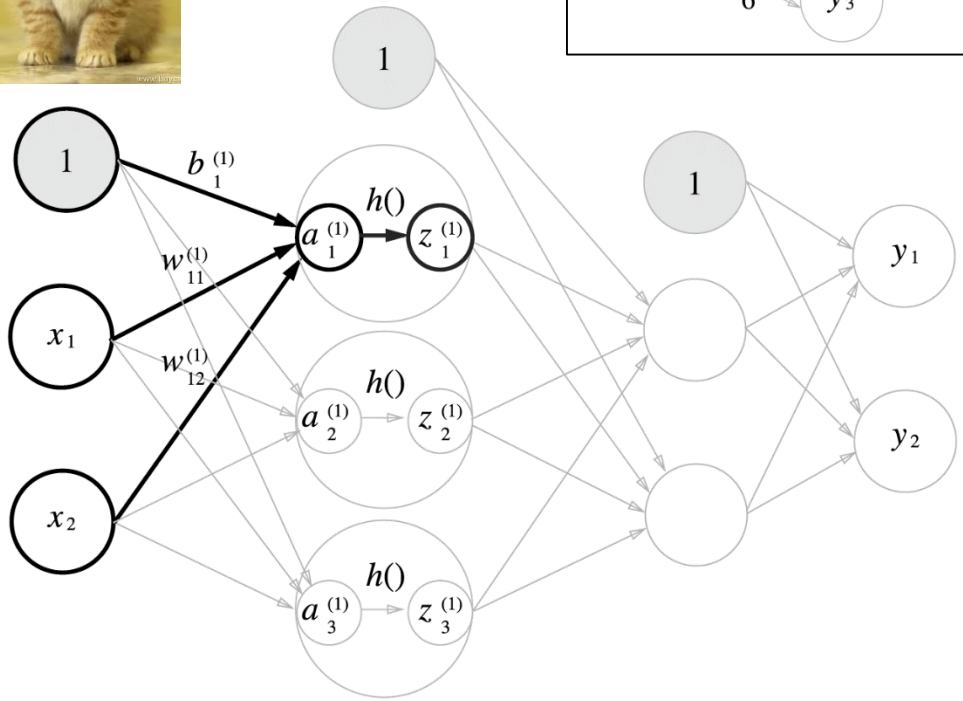
CNN



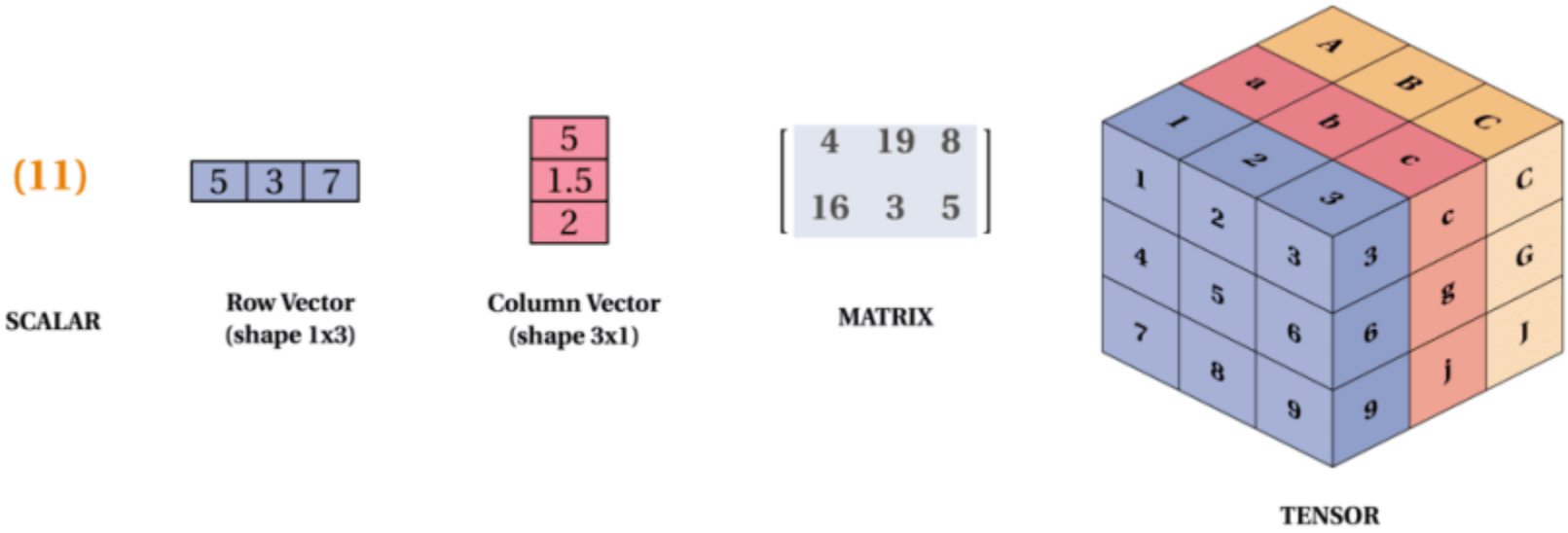
Feedforward

problem : $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$





Scalar, Vector, Matrix, Tensor



Scalar Vector Matrix Tensor

1

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

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

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$$

Vector

Scalar

24

Vector

$$\begin{bmatrix} 2 & -8 & 7 \end{bmatrix}$$

row

or column

$$\begin{bmatrix} -6 \\ -4 \\ 27 \end{bmatrix}$$

Matrix

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

row(s) × column(s)

Sum

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad \vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\vec{u} + \vec{v} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\vec{u} + \vec{v} = \begin{bmatrix} u_1+v_1 \\ u_2+v_2 \\ u_3+v_3 \end{bmatrix}$$

Dot

$$\mathbf{a} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}$$

$$\mathbf{a}.\mathbf{b} = x_1x_2 + y_1y_2 + z_1z_2$$

$$\mathbf{a} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 5 \\ 7 \end{pmatrix}$$

$$\mathbf{a}.\mathbf{b} = 2(3) + 4(5) + 6(7)$$

$$= 68$$

Norm

$$x = [x_1, x_2, \ldots, x_n]$$

Define the p-Norm: $|x|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$

L1-Norm is $|x|_1 = \sum_{i=1}^{n} |x_i|$

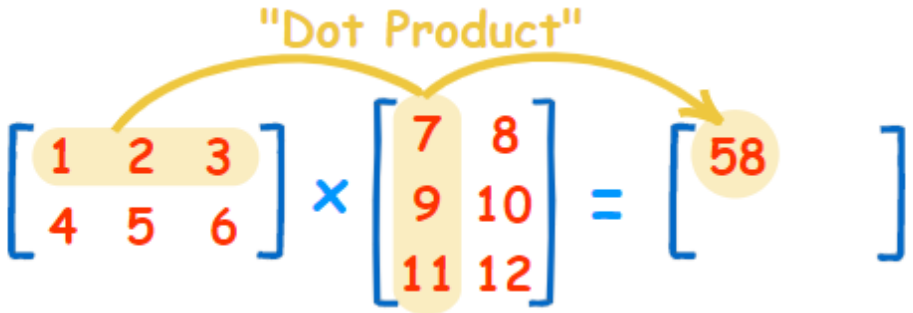
L2-Norm is $|x|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$

Matrix

Sum

$$\begin{pmatrix} 5 & 2 \\ 4 & 9 \\ 10 & -3 \end{pmatrix} + \begin{pmatrix} -11 & 0 \\ 7 & 1 \\ -6 & -8 \end{pmatrix} = \begin{pmatrix} 5+(-11) & 2+0 \\ 4+7 & 9+1 \\ 10+(-6) & -3+(-8) \end{pmatrix} = \begin{pmatrix} -6 & 2 \\ 11 & 10 \\ 4 & -11 \end{pmatrix}$$

Dot

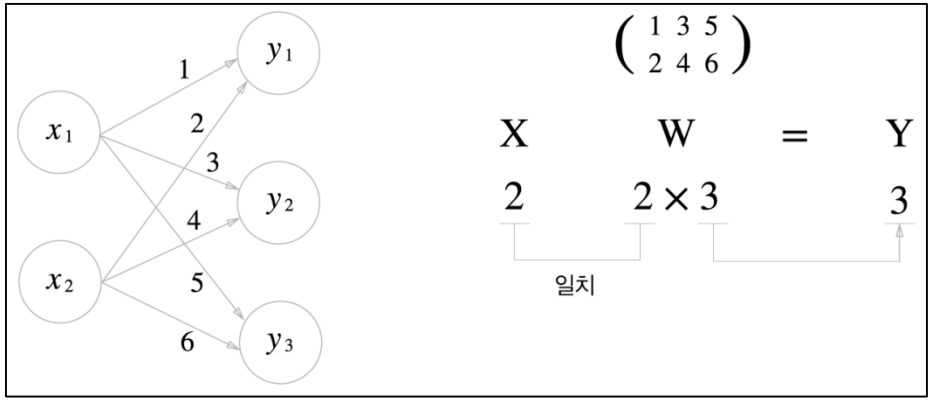


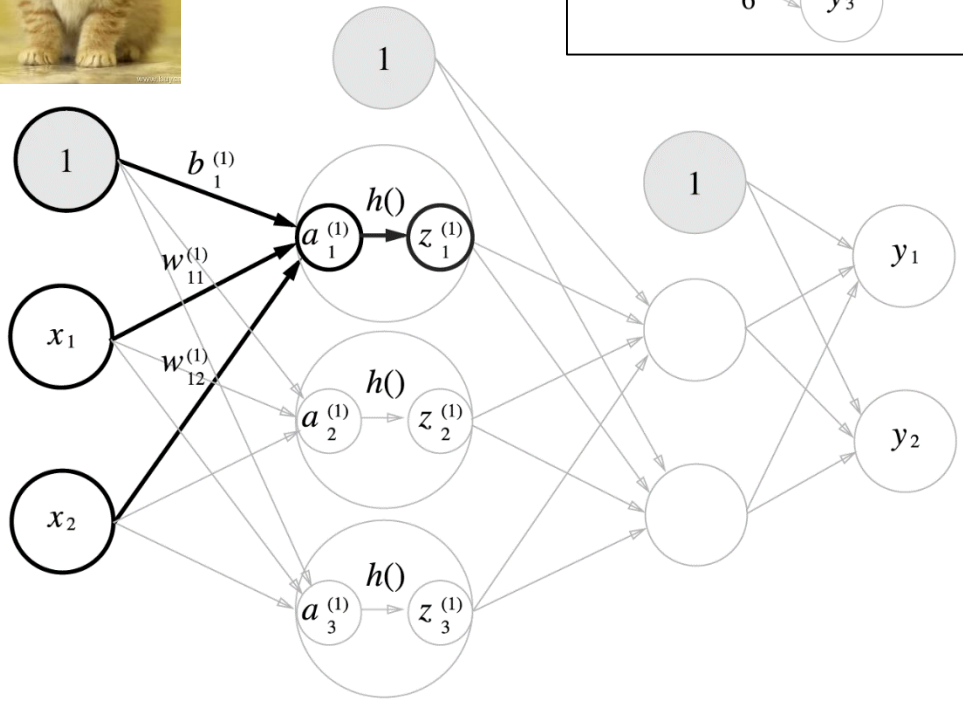
Element wise multiplication

$$\underset{G}{\begin{bmatrix} 3 & 5 & 7 \\ 4 & 9 & 8 \end{bmatrix}} \circ \underset{H}{\begin{bmatrix} 1 & 6 & 3 \\ 0 & 2 & 9 \end{bmatrix}} = \underset{N}{\begin{bmatrix} 3\times1 & 5\times6 & 7\times3 \\ 4\times0 & 9\times2 & 8\times9 \end{bmatrix}}$$

Feedforward

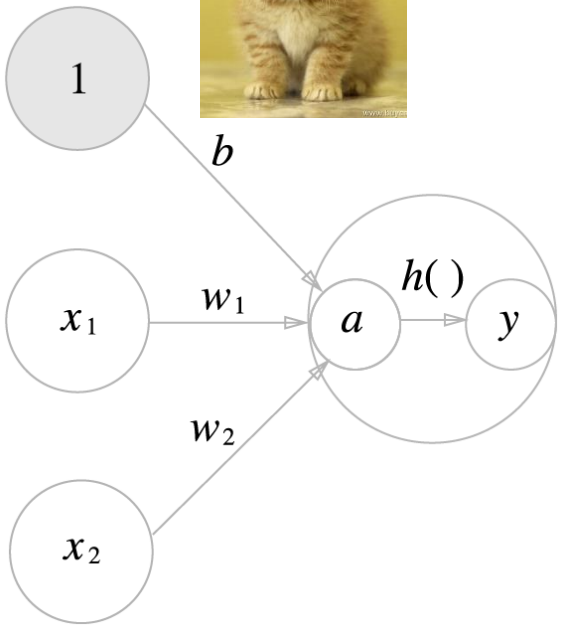
problem : $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$





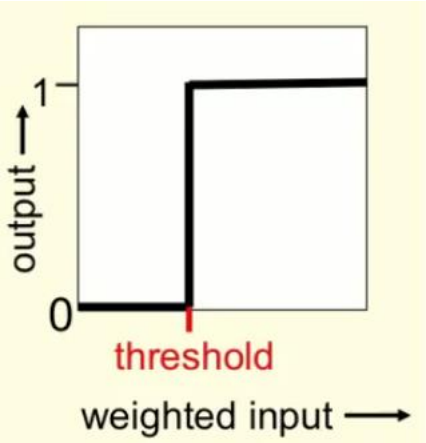
Activation Function

$$a = b + w_1 x_1 + w_2 x_2$$

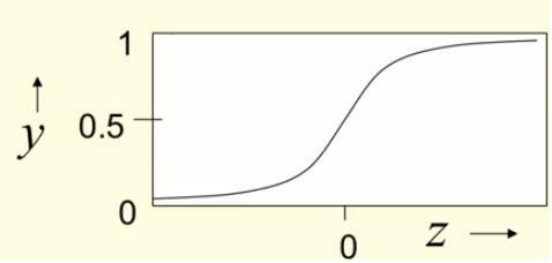


$$y = h(a)$$

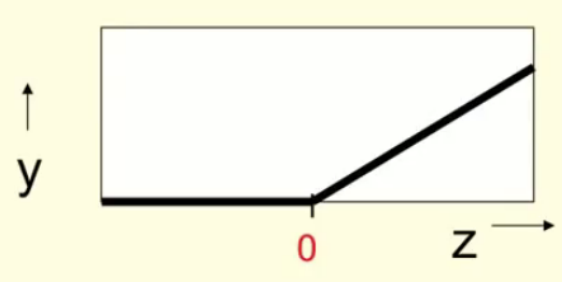
Step function



Sigmoid function

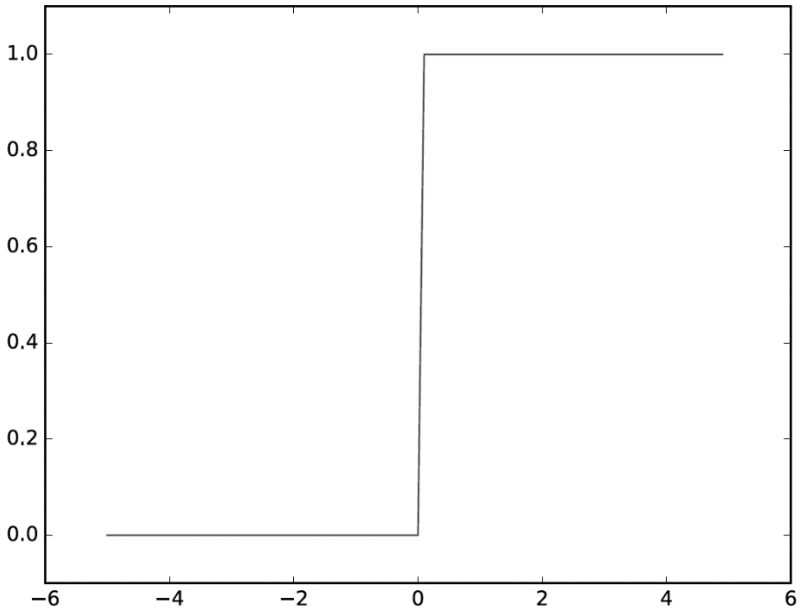


Rectified linear unit



Step Function

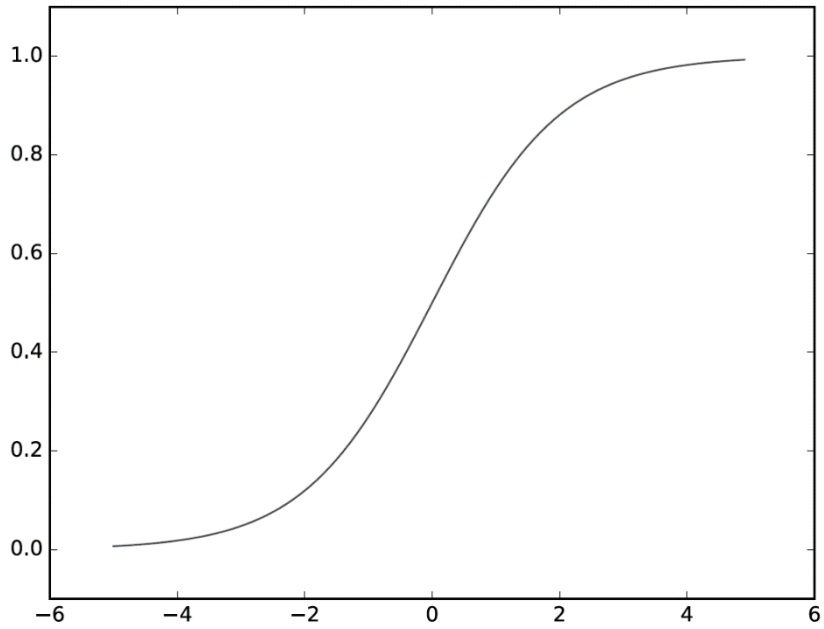
$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$




```
import numpy as np
import matplotlib.pylab as plt


def step_function(x):
    return np.array(x > 0, dtype=np.int)

X = np.arange(-5.0, 5.0, 0.1)
Y = step_function(X)
plt.plot(X, Y)
plt.ylim(-0.1, 1.1)
plt.show()
```

Sigmoid Function

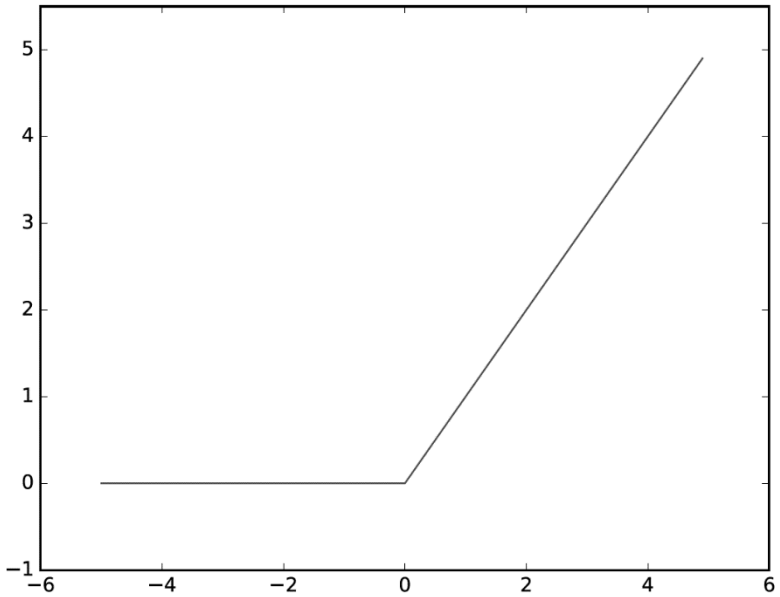
$$h(x) = \frac{1}{1 + \exp(-x)}$$




```
import numpy as np
import matplotlib.pylab as plt


def sigmoid(x):
    return 1 / (1 + np.exp(-x))

x = np.array([-1.0, 1.0, 2.0, -6.0, 6.0])
print(sigmoid(x))

X = np.arange(-5.0, 5.0, 0.1)
Y = sigmoid(X)
plt.plot(X, Y)
plt.ylim(-0.1, 1.1)
plt.show()
```

ReLU Function

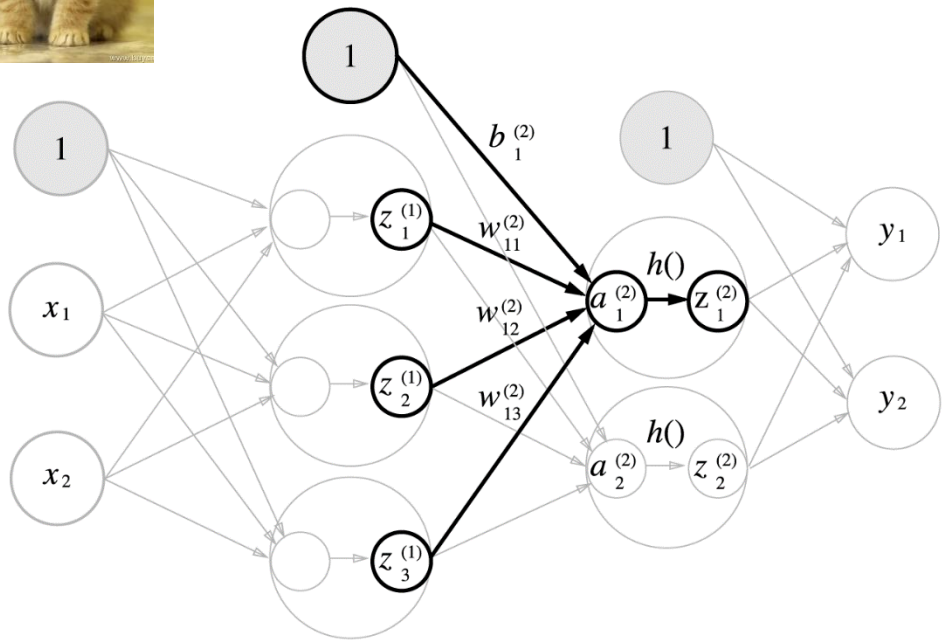
$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$




```
import numpy as np
import matplotlib.pylab as plt


def relu(x):
    return np.maximum(0, x)

x = np.arange(-5.0, 5.0, 0.1)
y = relu(x)
plt.plot(x, y)
plt.ylim(-1.0, 5.5)
plt.show()
```

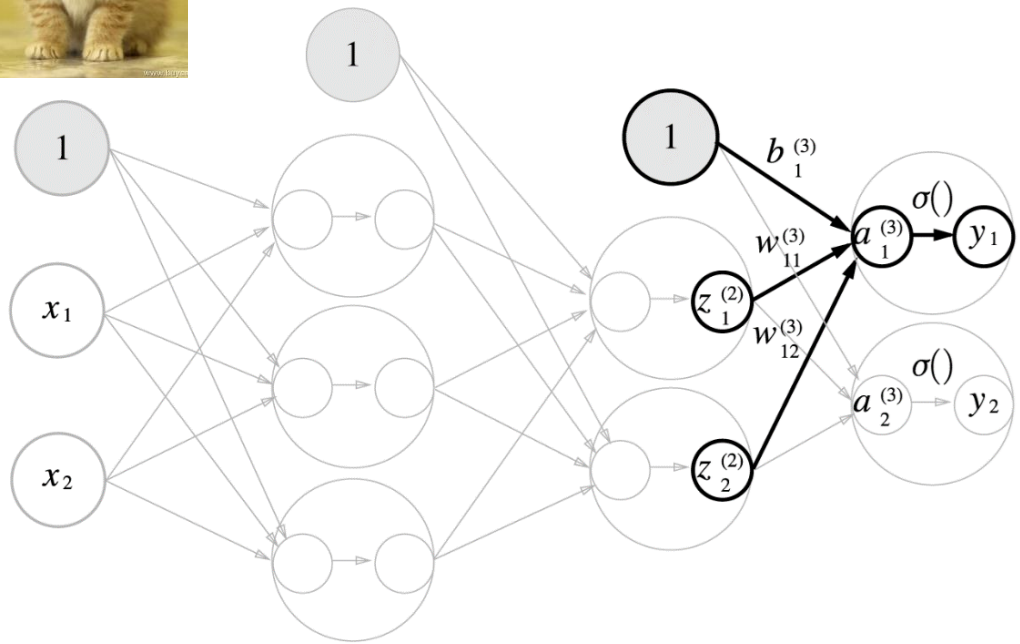
Feedforward

problem : $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$



Feedforward

problem : $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$



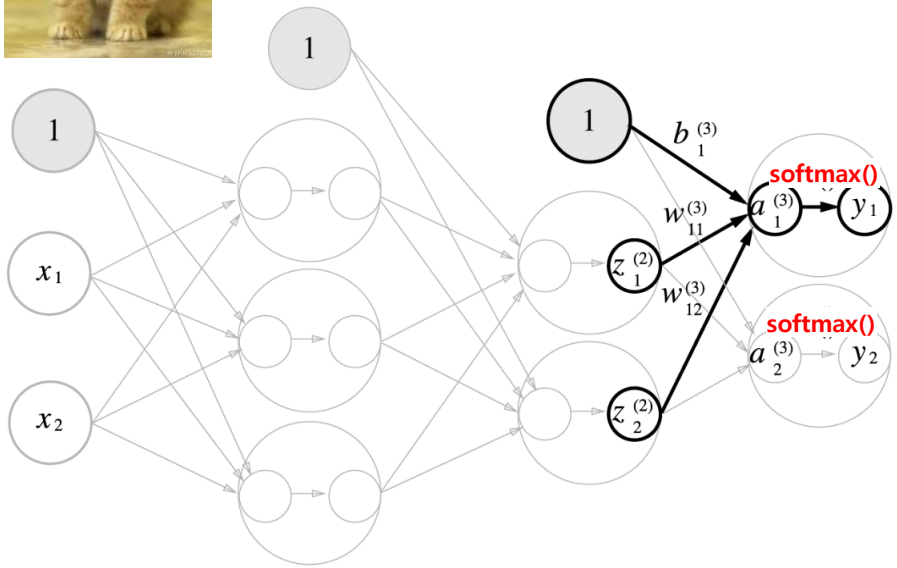
Cat: 1.5

Dog: 3.7

Softmax Function

- 입력의 지수함수에 모든 입력의 지수 함수의 합으로 나누어 줌
- 출력을 확률값으로 나타냄 (출력의 총합이 1)
- 분류문제에 사용

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$



Cat: 1.5 $\frac{e^{1.5}}{e^{1.5}+e^{3.7}} = 0.1$

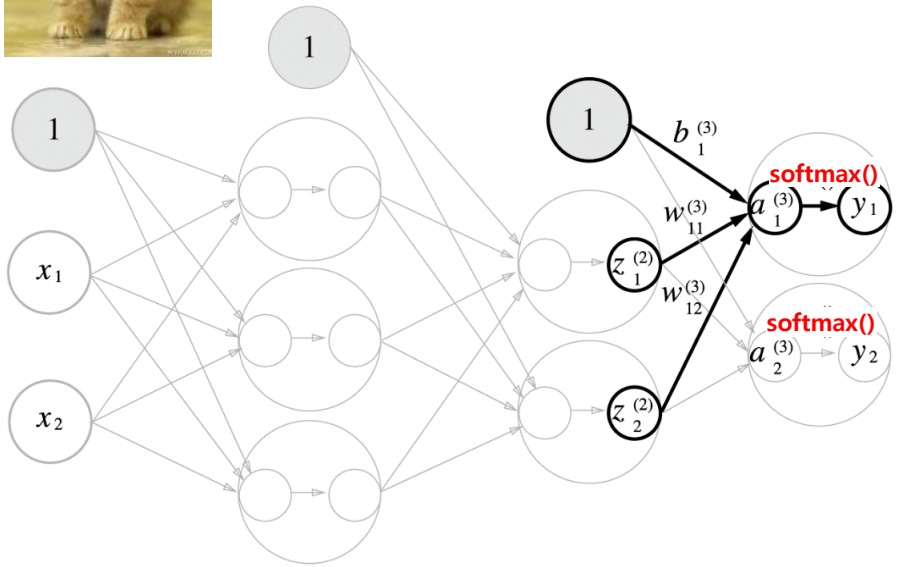
Dog: 3.7 $\frac{e^{3.7}}{e^{1.5}+e^{3.7}} = 0.9$

* $\frac{1.5}{1.5+3.7} + \frac{3.7}{1.5+3.7} = 1$ <-이렇게는 안되나요?

Softmax Function

- 입력의 지수함수에 모든 입력의 지수 함수의 합으로 나누어 줌
- 출력을 확률값으로 나타냄 (출력의 총합이 1)
- 분류문제에 사용

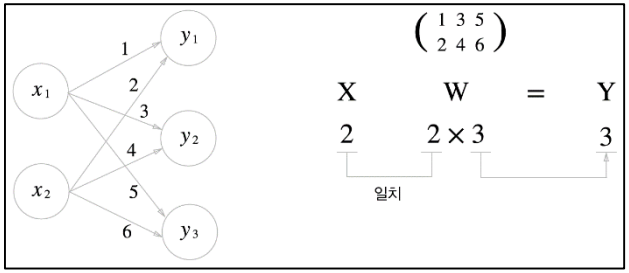
```
def softmax(a):
    c = np.max(a)
    exp_a = np.exp(a-c)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
print(softmax(a))
print(np.sum(softmax(a)))
```



Cat: 1.5 $\frac{e^{1.5}}{e^{1.5}+e^{3.7}} = 0.1$

Dog: 3.7 $\frac{e^{3.7}}{e^{1.5}+e^{3.7}} = 0.9$

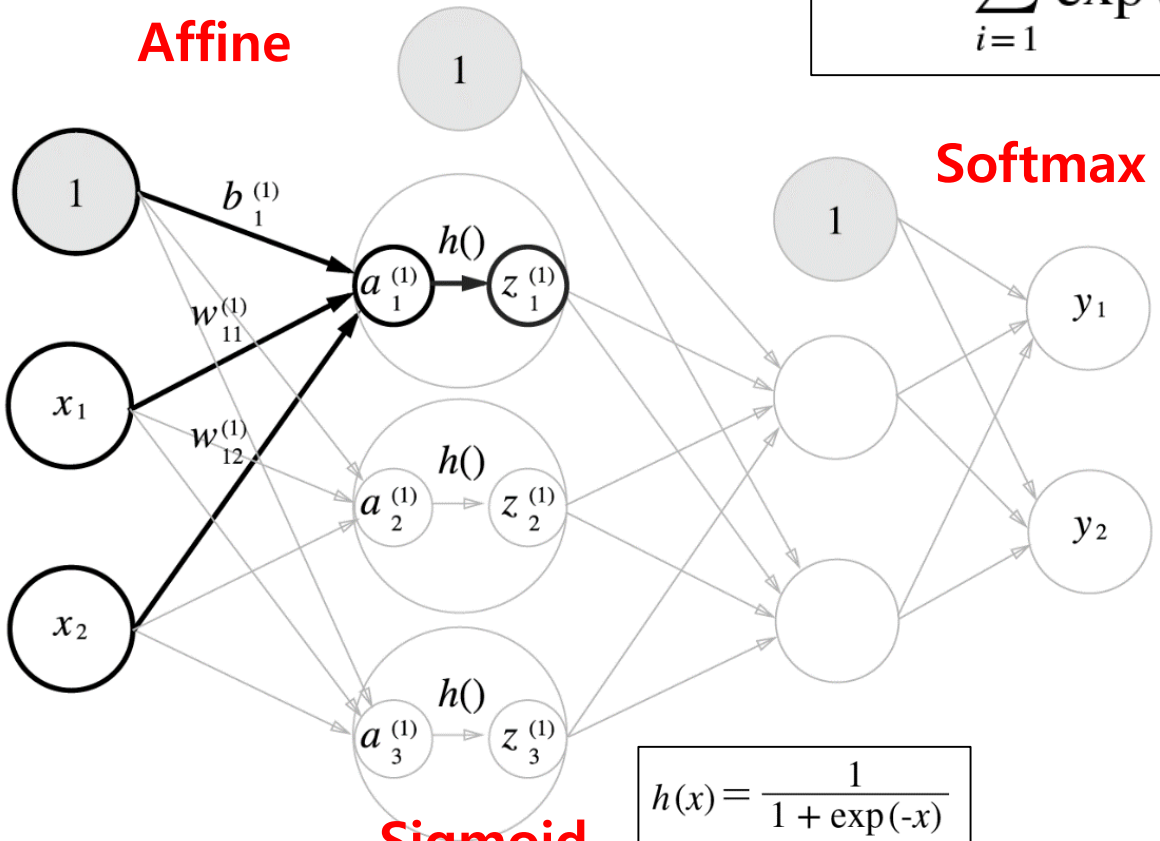
Feedforward



Affine

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

Softmax



Loss function

$$E = -\sum_k t_k \log y_k$$

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

Sigmoid

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

ReLU

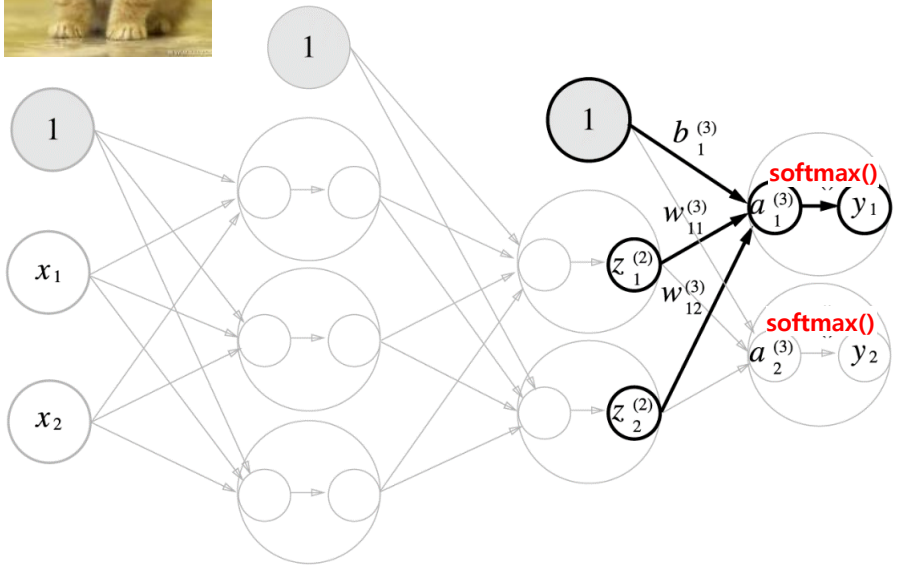
Loss Function (Error Function)

- Mean squared error (평균 제곱 오차)

$$E = \frac{1}{2}\sum_{k}(y_k - t_k)^2$$

```
def mean_squared_error(y, t):
    return 0.5 * np.sum((y-t)**2)
```

y = [**0.1**, 0.9]
t = [**1.0**, 0.0]

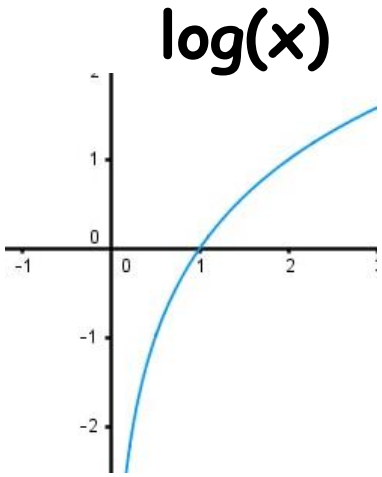


error

Cat: 0.1 $(0.1 - 1)^2$

Dog: 0.9 $(0.9 - 0)^2$

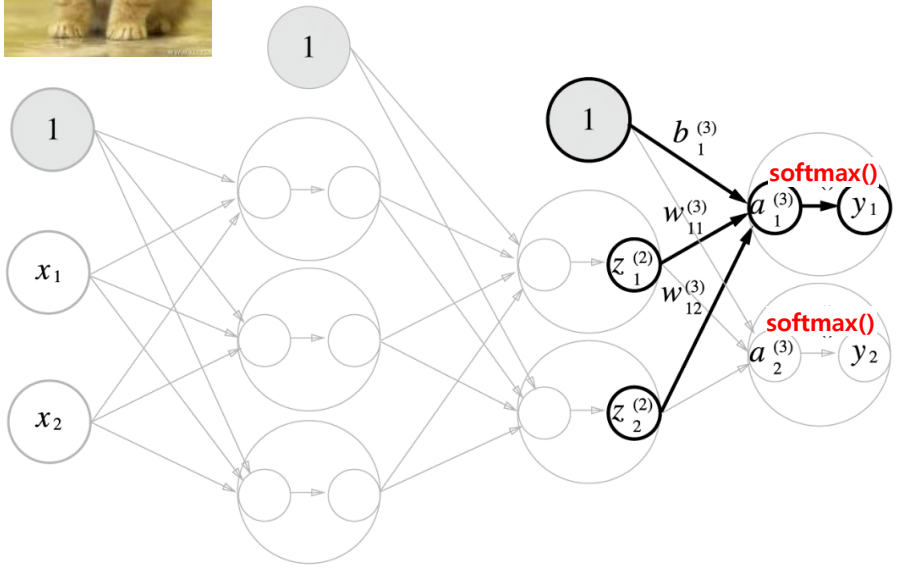
Loss Function (Error Function)



- Cross entropy error (교차 엔트로피 오차)
- 정답일 때의 출력이 전체 값을 결정

$$E = -\sum_{k} t_k \log y_k$$

y = [**0.1**, 0.9]
t = [**1.0**, 0.0]

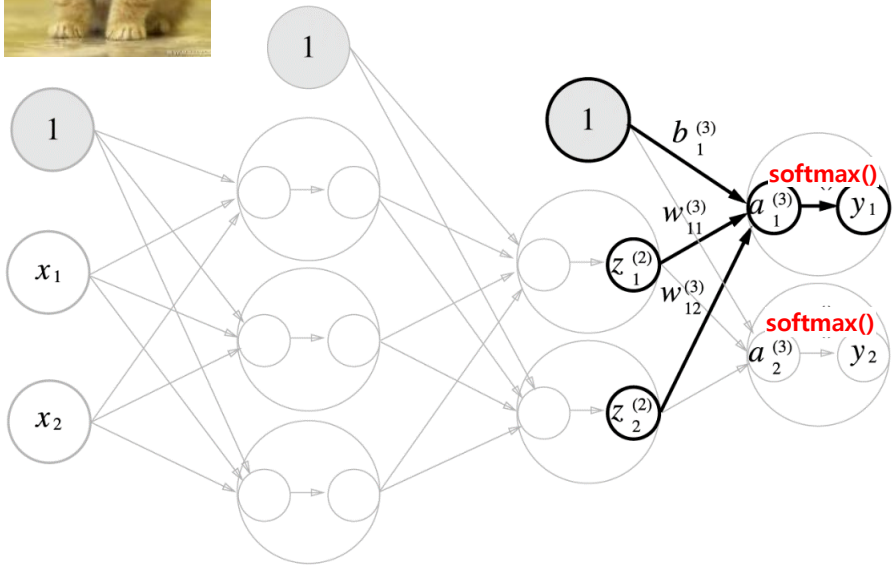


error

Cat: 0.1 $1 * \log 0.1$

Dog: 0.9 $0 * \log 0.9$

Calculate Loss!



Cat: ?

Dog: ?

error

?

?

z_1	z_2	w_{11}	w_{12}	w_{21}	w_{22}	b_1	b_2	activation
0.5	0.1	0.2	0.3	0.1	0.4	0.7	0.2	softmax

Calculate Loss!

z_1	z_2	w_{11}	w_{12}	w_{21}	w_{22}	b_1	b_2	activation
0.5	0.1	0.2	0.3	0.1	0.4	0.7	0.2	softmax

truth = [1.0 0.0]

z = [0.5 0.1] w = $\begin{bmatrix} 0.2 & 0.1 \\ 0.3 & 0.4 \end{bmatrix}$ b = [0.7 0.2]

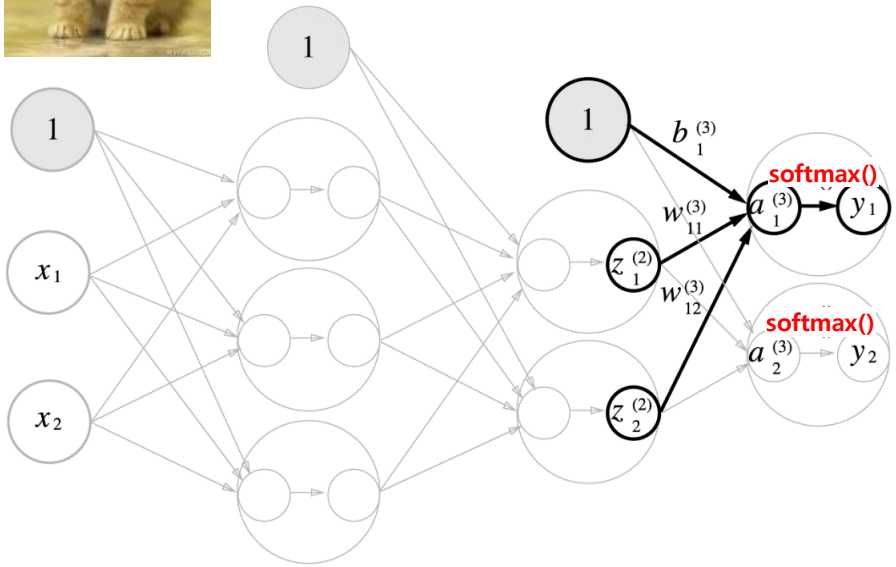
z · w = [0.13 0.09]

z · w+b = [0.83 0.29]

softmax(z · w+b) = [0.63 0.37]

crossentropy(softmax(z · w+b)) = 0.2+0.0 = 0.2

Homework - Calculate Loss!



Cat: ?

Dog: ?

error

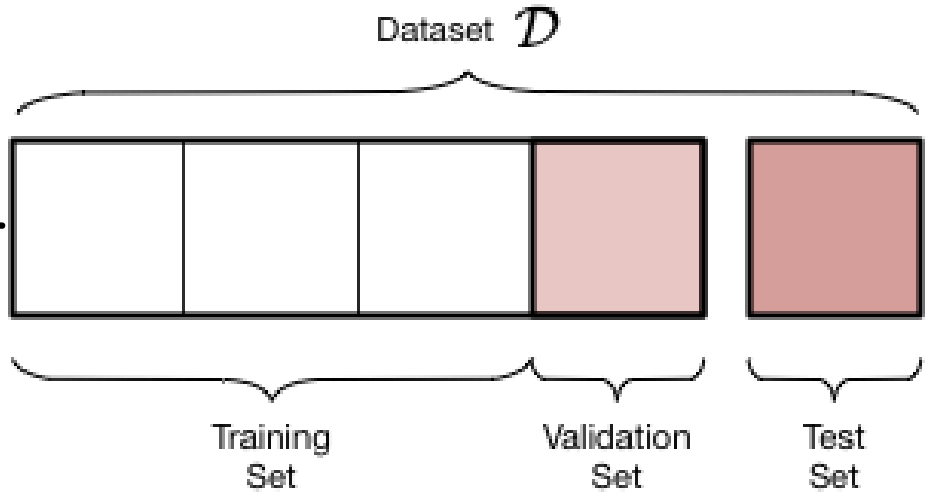
?

?

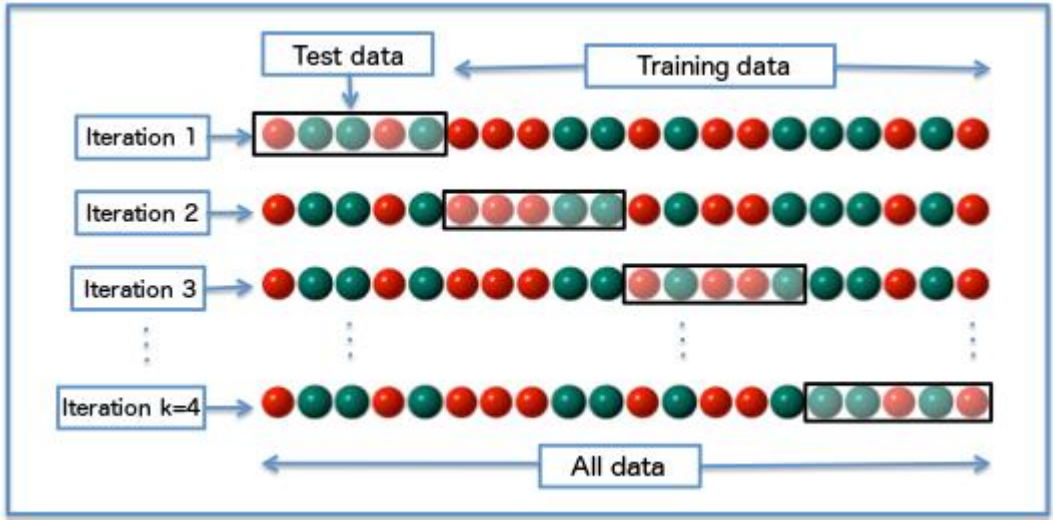
z_1	z_2	w_{11}	w_{12}	w_{21}	w_{22}	b_1	b_2	activation
0.2	0.4	0.1	0.2	0.9	0.2	0.1	0.9	softmax

Training, Test, Validation (development) set

Training, Test, Validation set



cross validation

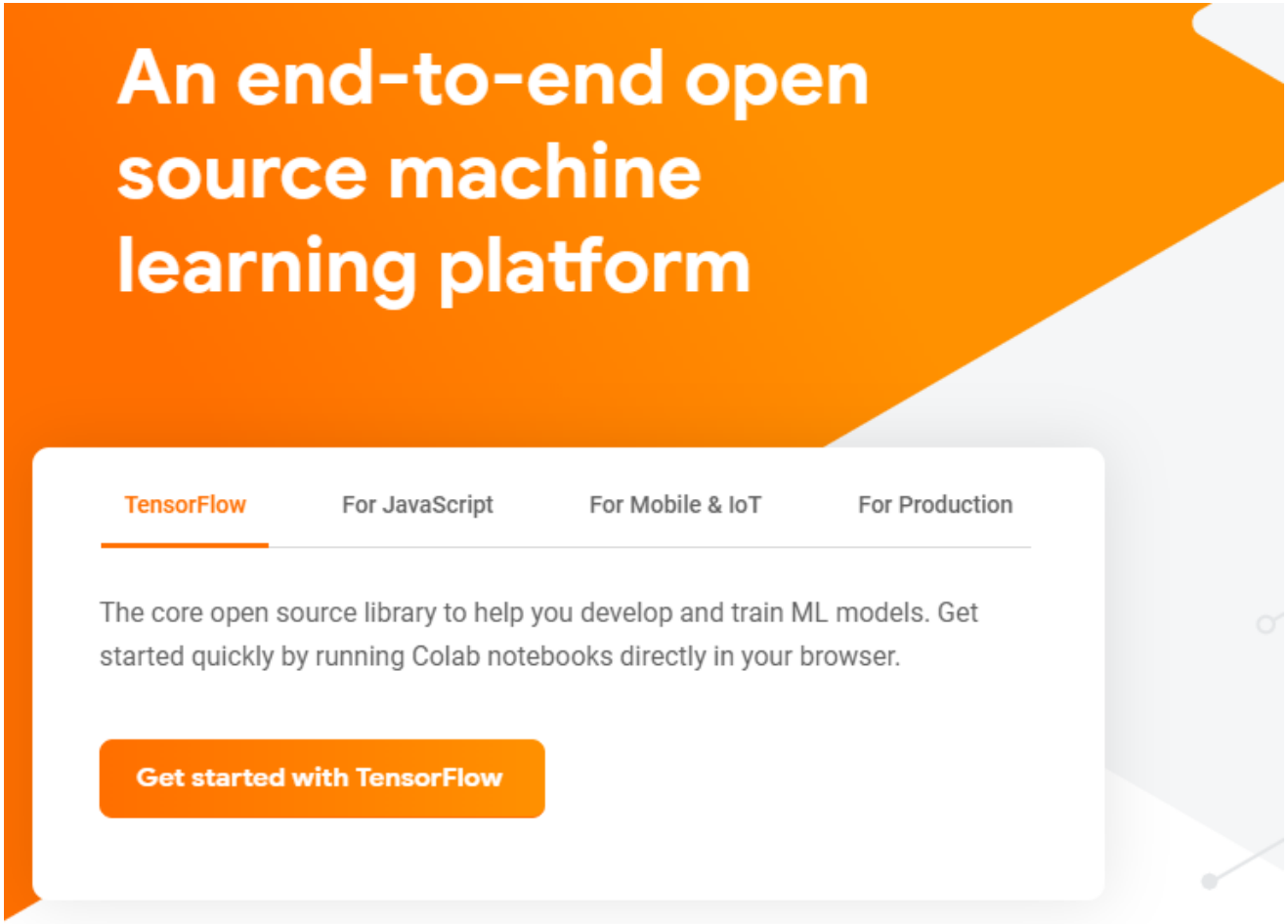


AI School 6기 1주차

텐서플로우 기초

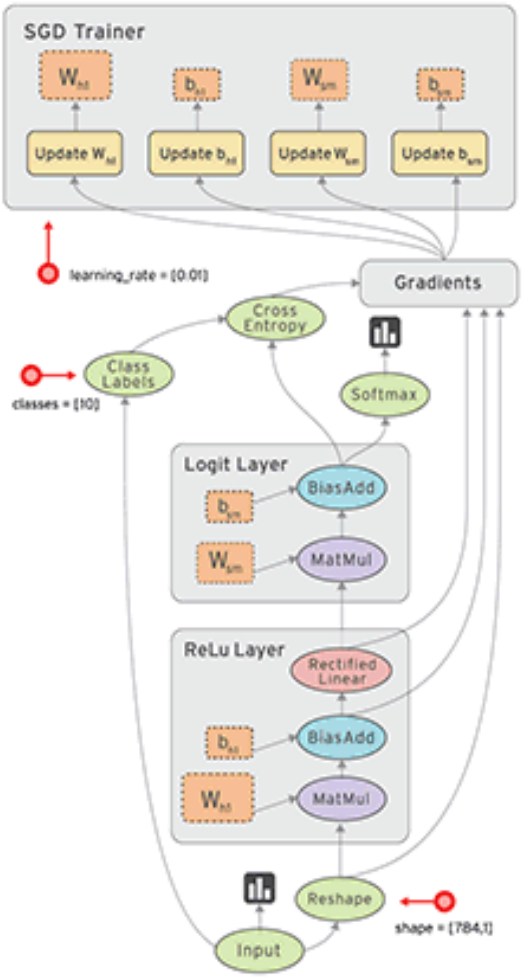
TensorFlow

- 오픈소스 머신러닝 플랫폼



Dataflow graph

- the **nodes** represent units of computation, and the **edges** represent the data consumed or produced by a computation (multidimensional data arrays)



Hello TensorFlow!

```
import tensorflow as tf

hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

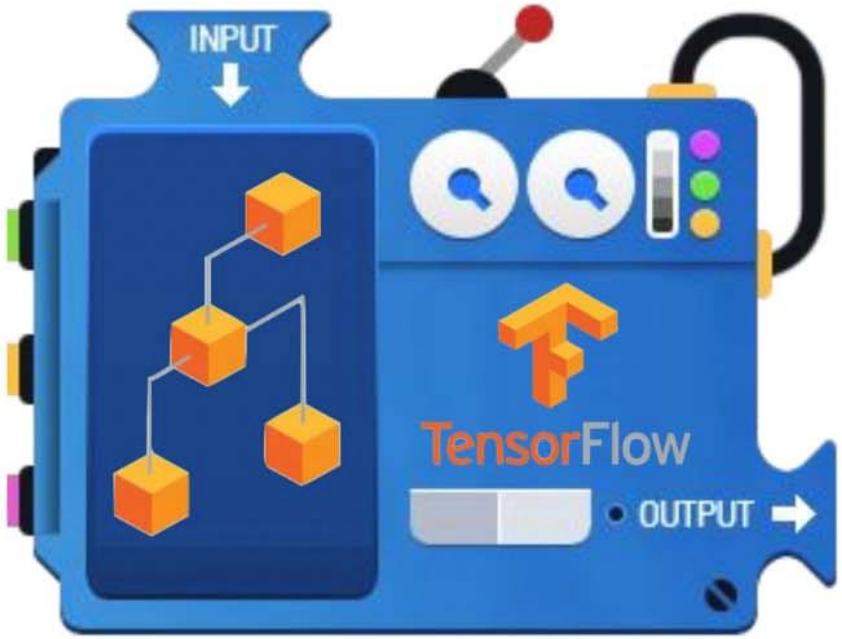
Computational Graph

```
import tensorflow as tf

node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)

print("node1:", node1, "node2:", node2)
print("node3: ", node3)

sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

TensorFlow process

```
sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

2 feed data and run graph (operation)
sess.run (op)



1 Build graph using TensorFlow operations

3 update variables in the graph (and return values)

```
node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0)
node3 = tf.add(node1, node2)
```

Placeholder

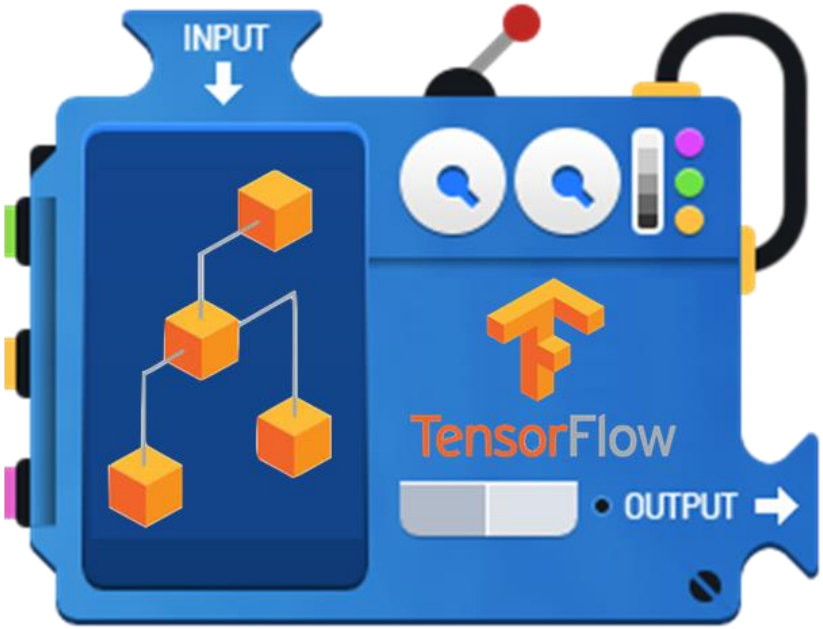
- Placeholder: 프로그램 실행 중에 값을 변경할 수 있는 가역변수

```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
adder_node = a + b

sess = tf.Session()

print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
print(sess.run(adder_node, feed_dict={a: [1,3], b: [2, 4]}))
```

```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
y = tf.add(a, b)

sess = tf.Session()

print(sess.run(y, feed_dict={a: 3, b: 4.5}))
```

TensorFlow process

2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})



1 Build graph using TensorFlow operations

3 update variables in the graph (and return values)

Tensor rank, Shapes, and Types

```
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)	`....`

Tensor rank, Shapes, and Types

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

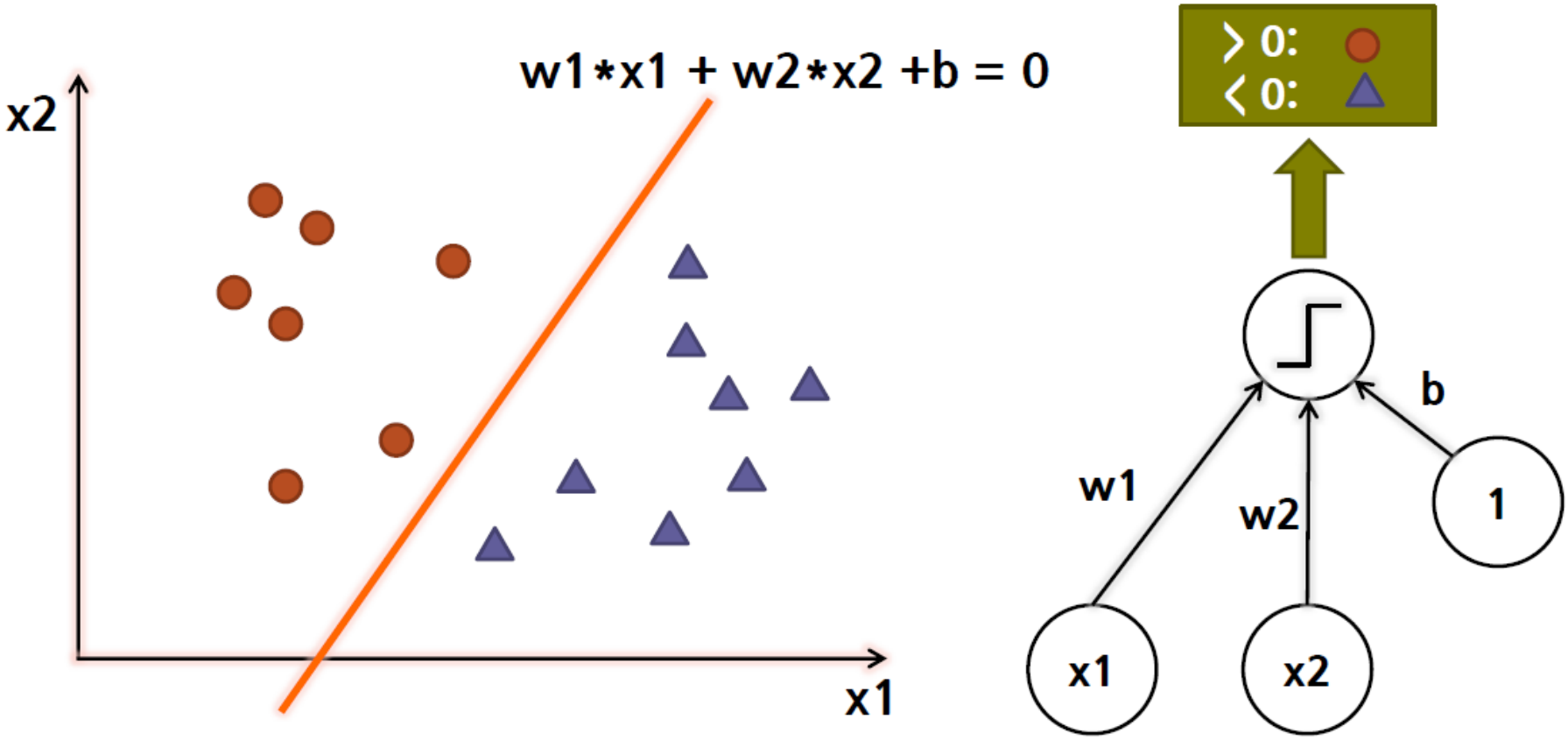
Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

Tensor rank, Shapes, and Types

```
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.

Perceptron (1958~)



Perceptron (1958~)

```
import tensorflow as tf

x_data = [[1, 2]]

X = tf.placeholder(tf.float32, shape=[None, 2])

W = tf.Variable(tf.random_normal([2, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = tf.sigmoid(tf.matmul(X, W) + b)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    prediction = sess.run(hypothesis, feed_dict={X: x_data})
    print(prediction)
```

ReLU?

Q&A

과제 송부 메일 : dha8102@naver.com