파이썬 딥러닝

이성주

seongjoo@codebasic.io

jupyter Day 4

Logout

Notebook saved | Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

다층 퍼셉트론으로 XOR 문제 해결하기

In [20]:
```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y
```

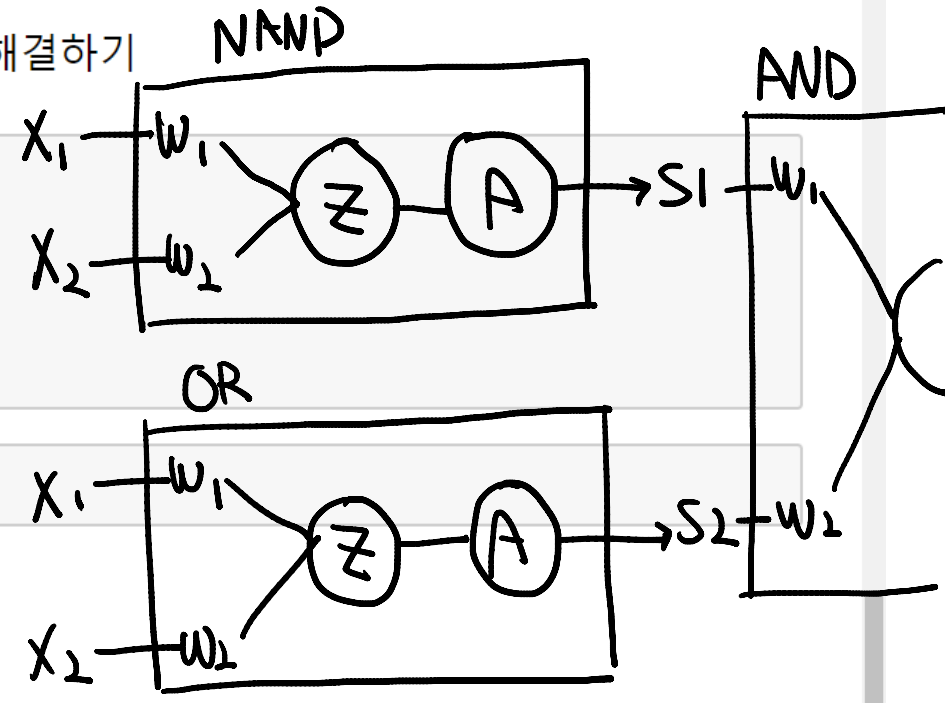
In [21]:
```
test(XOR)
```

```
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 0
```

~ 비선형 문제

⇓

인공신경망

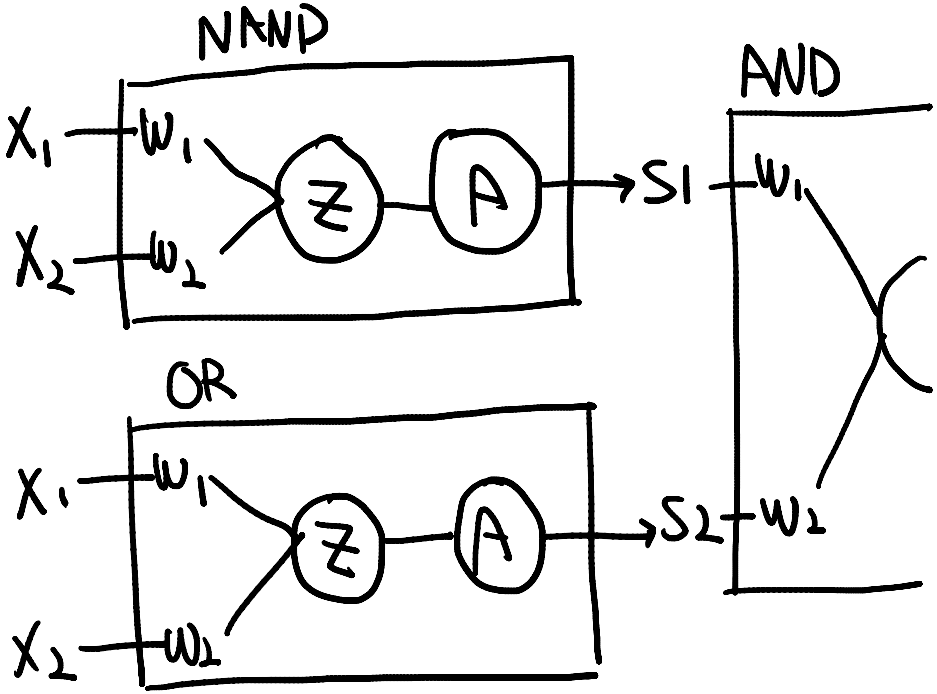


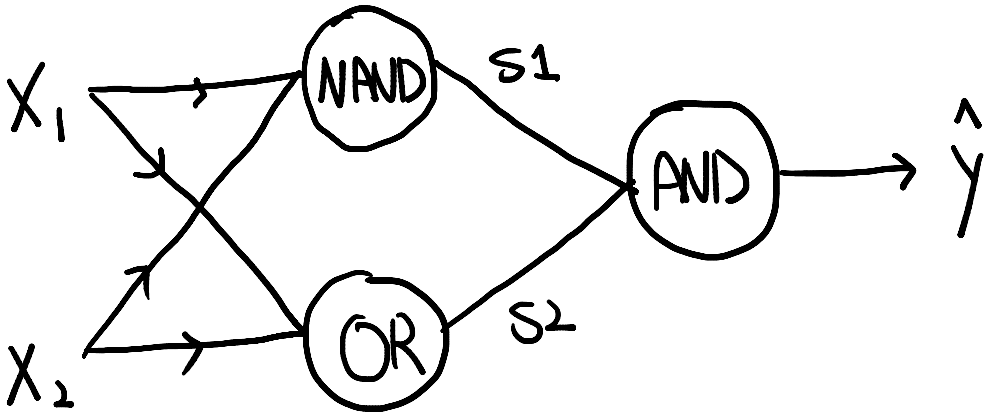
In [ ]:

~ 비선형 문제

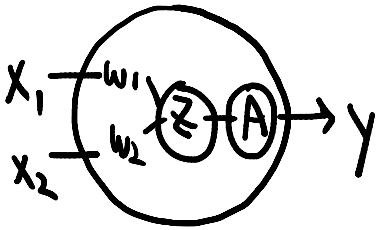
⇓

인공 신경망

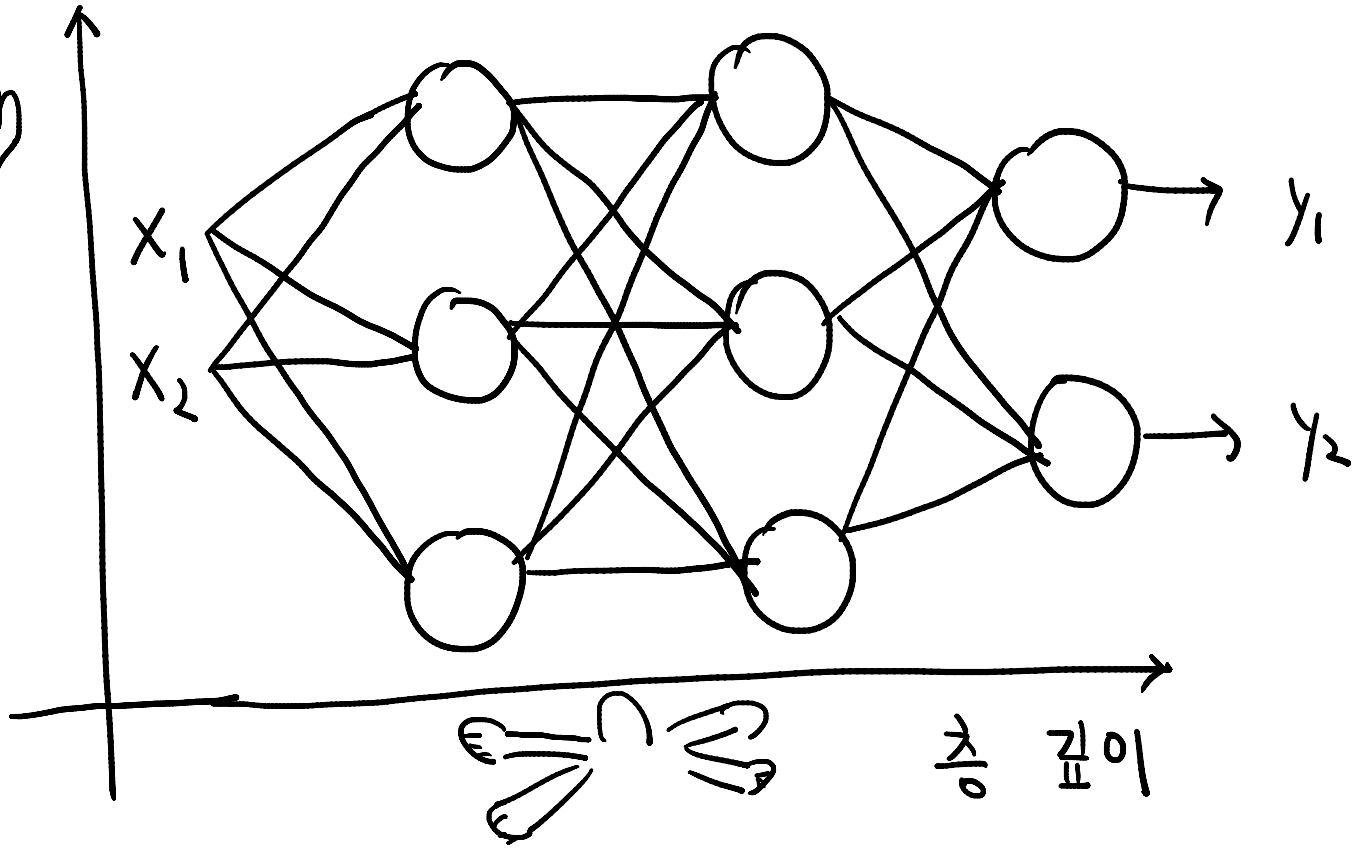

NAND
X_1
X_2
w_1
w_2
Z
A
S1
AND
w_1
w_2
OR
X_1
X_2
w_1
w_2
Z
A
S2


X_1
X_2
NAND
OR
S1
S2
AND
$\hat{y}$

인공신경망



ANN ; Artificial Neural Network



인공 신경망

TRUE

In [26]:
```
def step(x):
    y = np.where(x > 0, 1, -1)
    return y
```

Fake

In [27]:
```
x = np.arange(-5., 5., 0.1)
```

-5.0 -4.9 ... 4.9 5.0

In [28]:
```
y = step(x)
```

In [29]:
```
Series(y, index=x).plot()
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0xace3a20>

jupyter Day 4

Logout

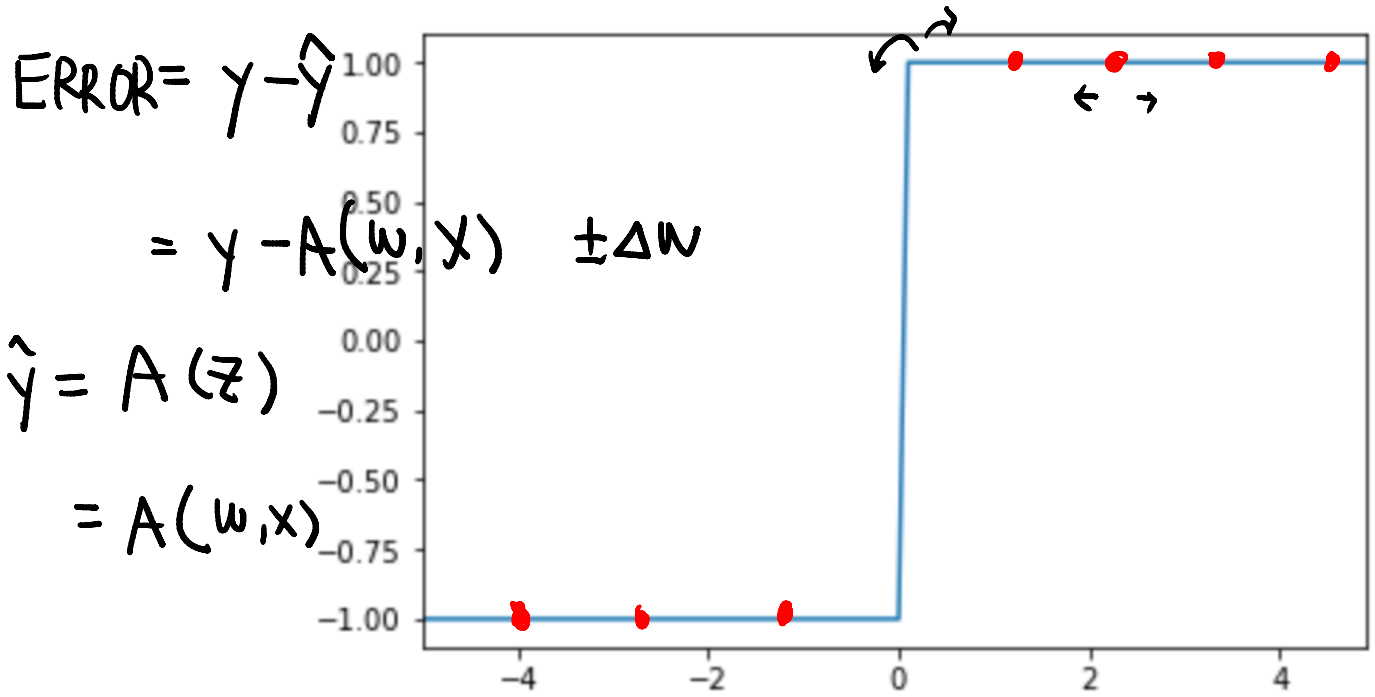
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [29]: Series(y, index=x).plot()

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0xace3a20>



In [30]:
```
def sigmoid(x):
    return 1 / (1+np.exp(-x))
```

jupyter Day 4

Logout

Trusted | Python 3

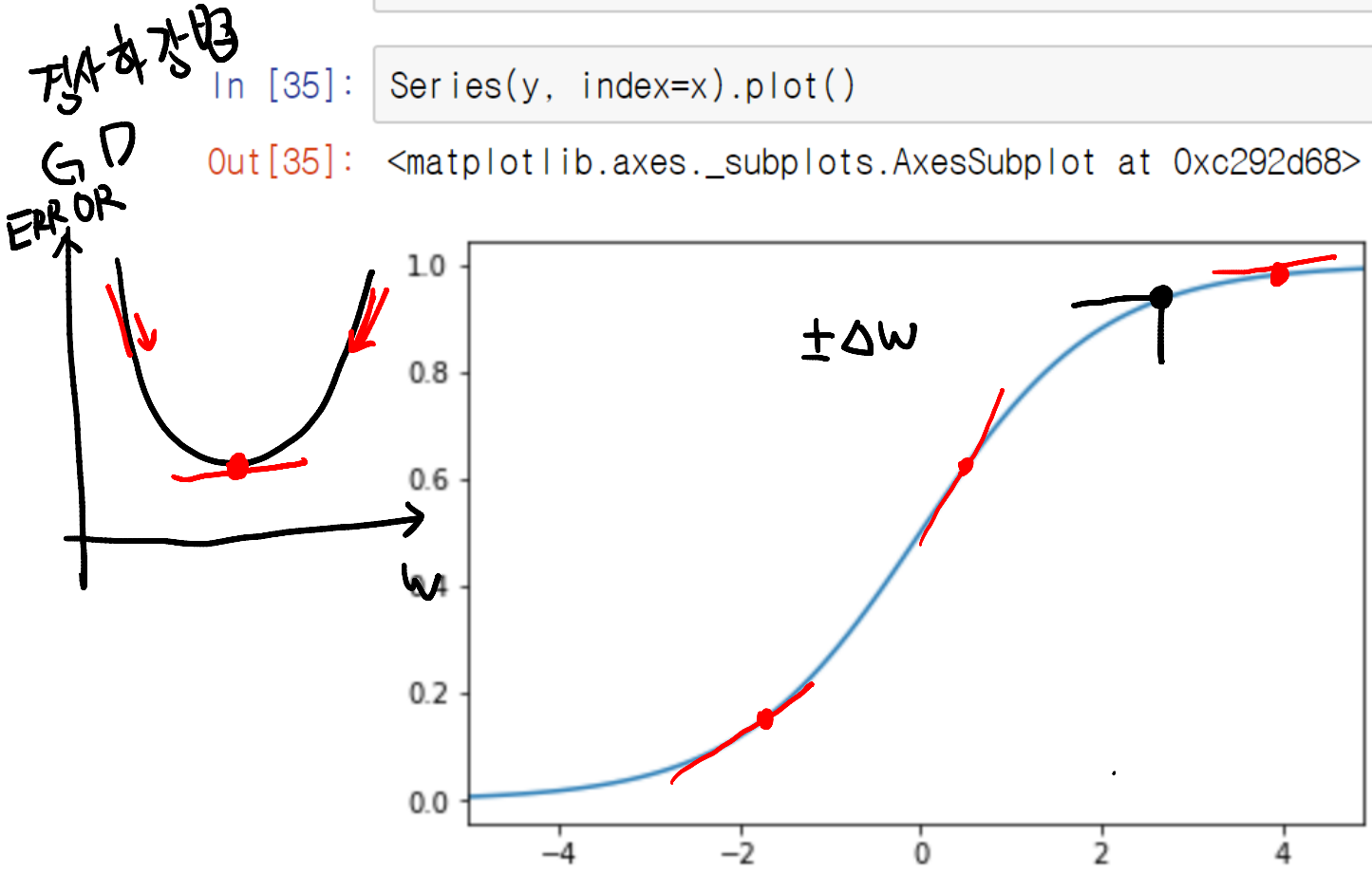
File Edit View Insert Cell Kernel Widgets Help

Code

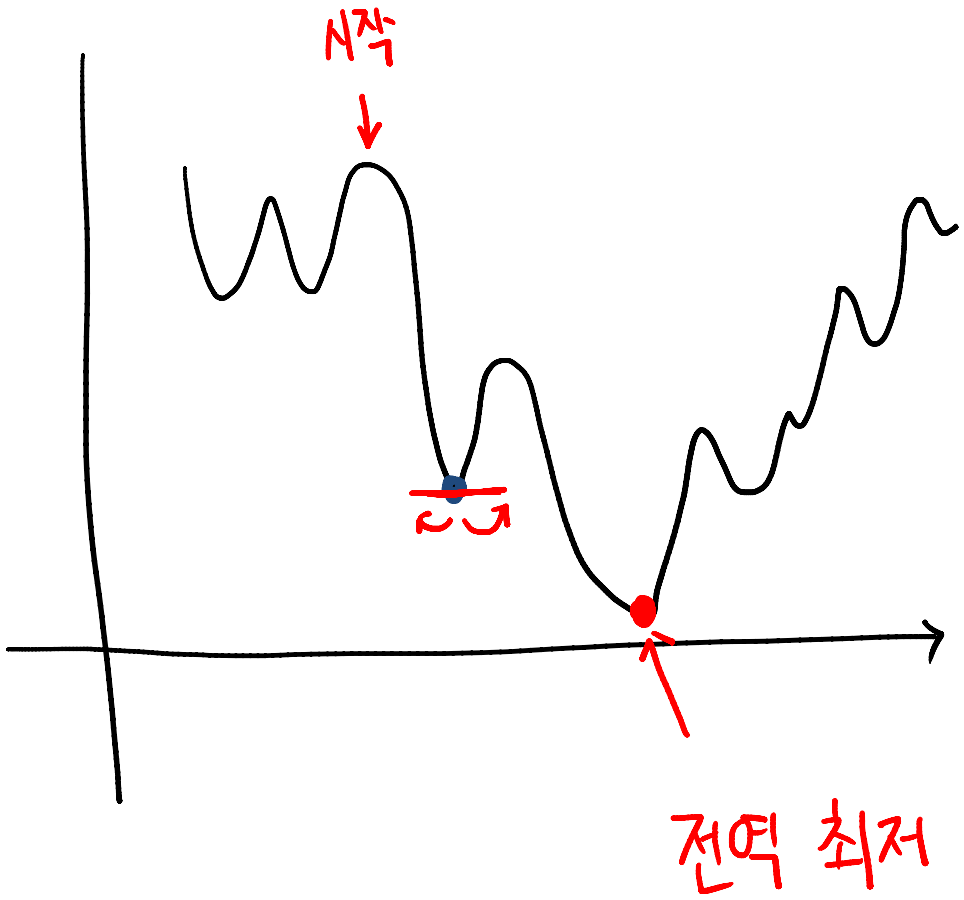
```
In [34]: y = sigmoid(x)
```

```
In [35]: Series(y, index=x).plot()
```

```
Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0xc292d68>
```



LR.predict_proba( )


시작
전역 최저

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

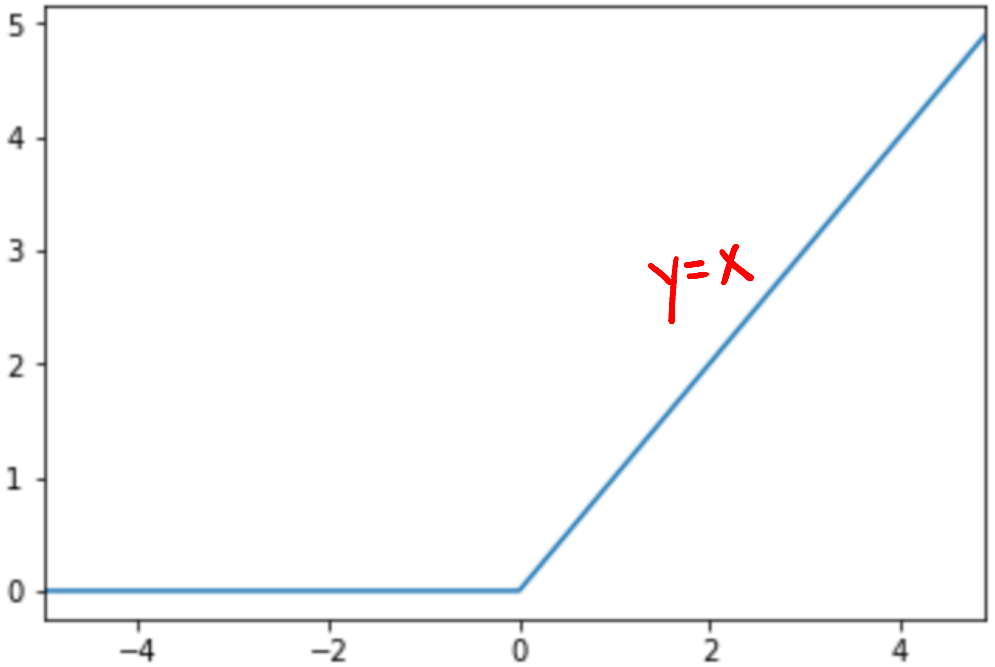
Code

In [36]:
```
def ReLU(x):
    return np.maximum(0, x)
```

In [37]:
```
y = ReLU(x)
```

In [38]:
```
Series(y, index=x).plot()
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0xc2bf5c0>



Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

신경망 순전파

1.0 → x_1

In [48]:
```
x = np.array([1.0, 0.5])
```

1층

0.5 → x_2

In [40]:
```
W1 = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
b1 = np.array([0.1, 0.2, 0.3])
```

In [52]:
```
z1 = np.dot(x, W1) + b1
```

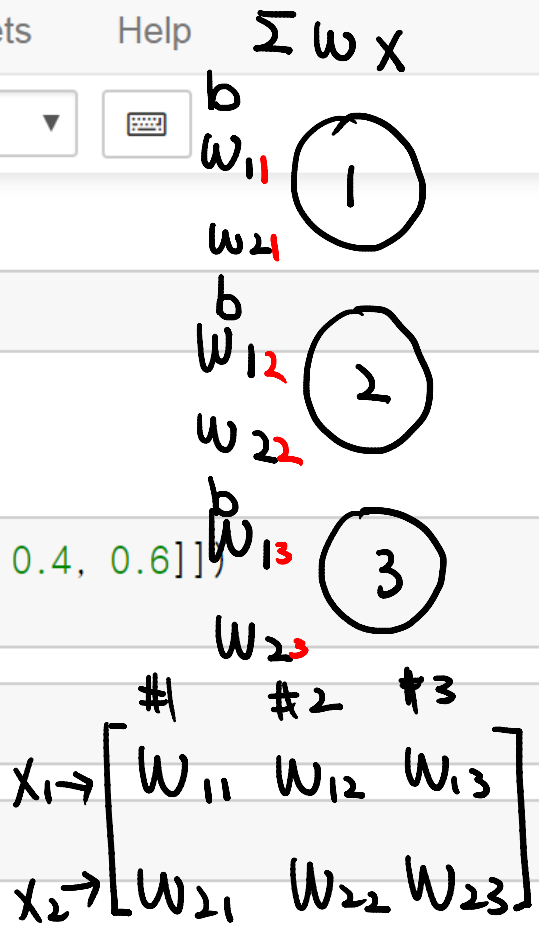
In [55]:
```
z1
```

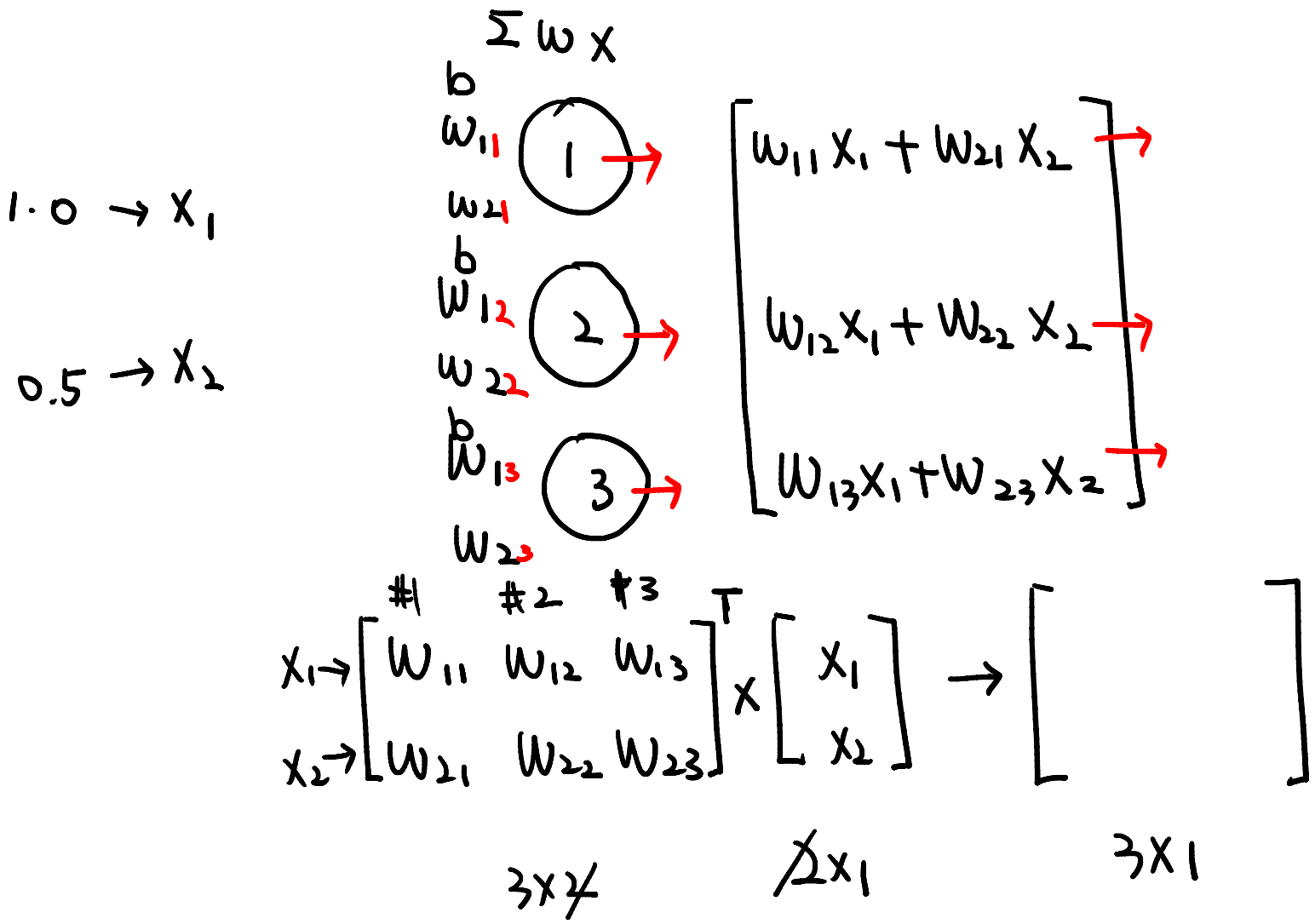
Out[55]:
```
array([ 0.3,  0.7,  1.1])
```

In [53]:
```
a1 = sigmoid(z1)
```

In [54]:
```
a1
```

Out[54]:
```
array([ 0.57444252,  0.66818777,  0.75026011])
```




Σ w x
b
w11
w21
1
1.0 → x1
b
w12
w22
2
0.5 → x2
b
w13
w23
3
w11 x1 + w21 x2
w12 x1 + w22 x2
w13 x1 + w23 x2
#1 #2 #3
T
x1 →
x2 →
x
→
3x2
2x1
3x1

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

신경망 순전파

In [48]:
```
x = np.array([1.0, 0.5])
```

1x2

1층

In [40]:
```
W1 = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
b1 = np.array([0.1, 0.2, 0.3])
```

2x3

In [52]:
```
z1 = np.dot(x, W1) + b1
```

(1x2) x (2x3) → (1x3)

In [55]:
```
z1
```

(1, 3)

Out[55]:
```
array([ 0.3,  0.7,  1.1])
```

In [53]:
```
a1 = sigmoid(z1)
```

In [54]:
```
a1
```

Out[54]:
```
array([ 0.57444252,  0.66818777,  0.75026011])
```

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

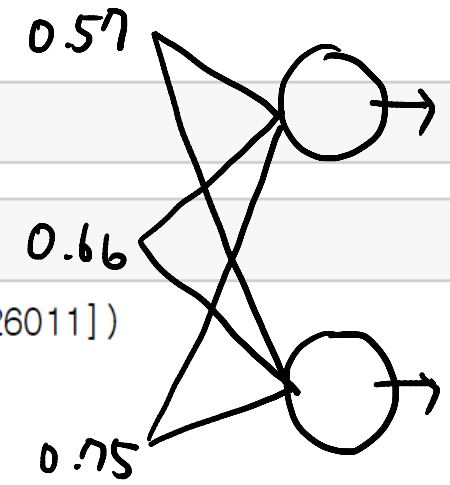
Code

Out[55]: array([0.3, 0.7, 1.1])

In [53]:
```
a1 = sigmoid(z1)
```

In [54]:
```
a1
```

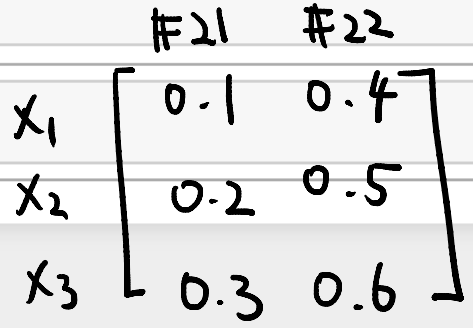
Out[54]: array([0.57444252, 0.66818777, 0.75026011])



2층

In [56]:
```
W2 = np.array([[0.1, 0.4], [0.2, 0.5], [0.3, 0.6]])
b2 = np.array([0.1, 0.2])
```

In [ ]:



Trusted | Python 3

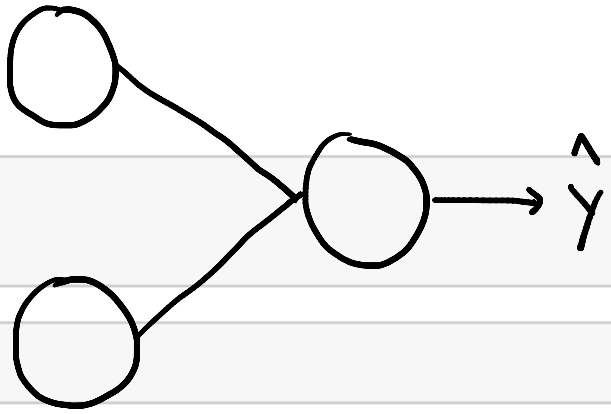
File Edit View Insert Cell Kernel Widgets Help

Code

2층 출력

Out[64]: (2,)

출력



In [70]:
```
W3 = np.array([0.1, 0.3])
b3 = 0.1
```

In [71]:
```
z3 = np.dot(a2, W3) + b3
```

In [72]:
```
a3 = sigmoid(z3)
```

In [73]:
```
y = a3
```

회귀

In [74]:
```
y
```

Out[74]: 0.59722796177405213

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

```
b3 = 0.1
```

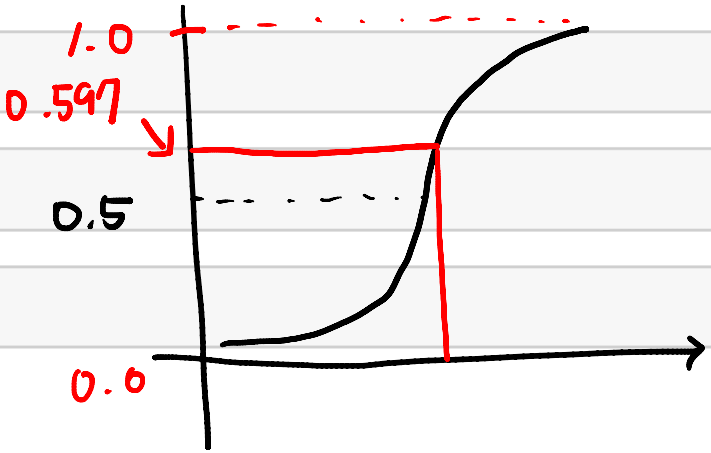
In [71]:
```
z3 = np.dot(a2, W3) + b3
```

In [72]:
```
a3 = sigmoid(z3)
```

In [73]:
```
y = a3
```

In [74]:
```
y
```

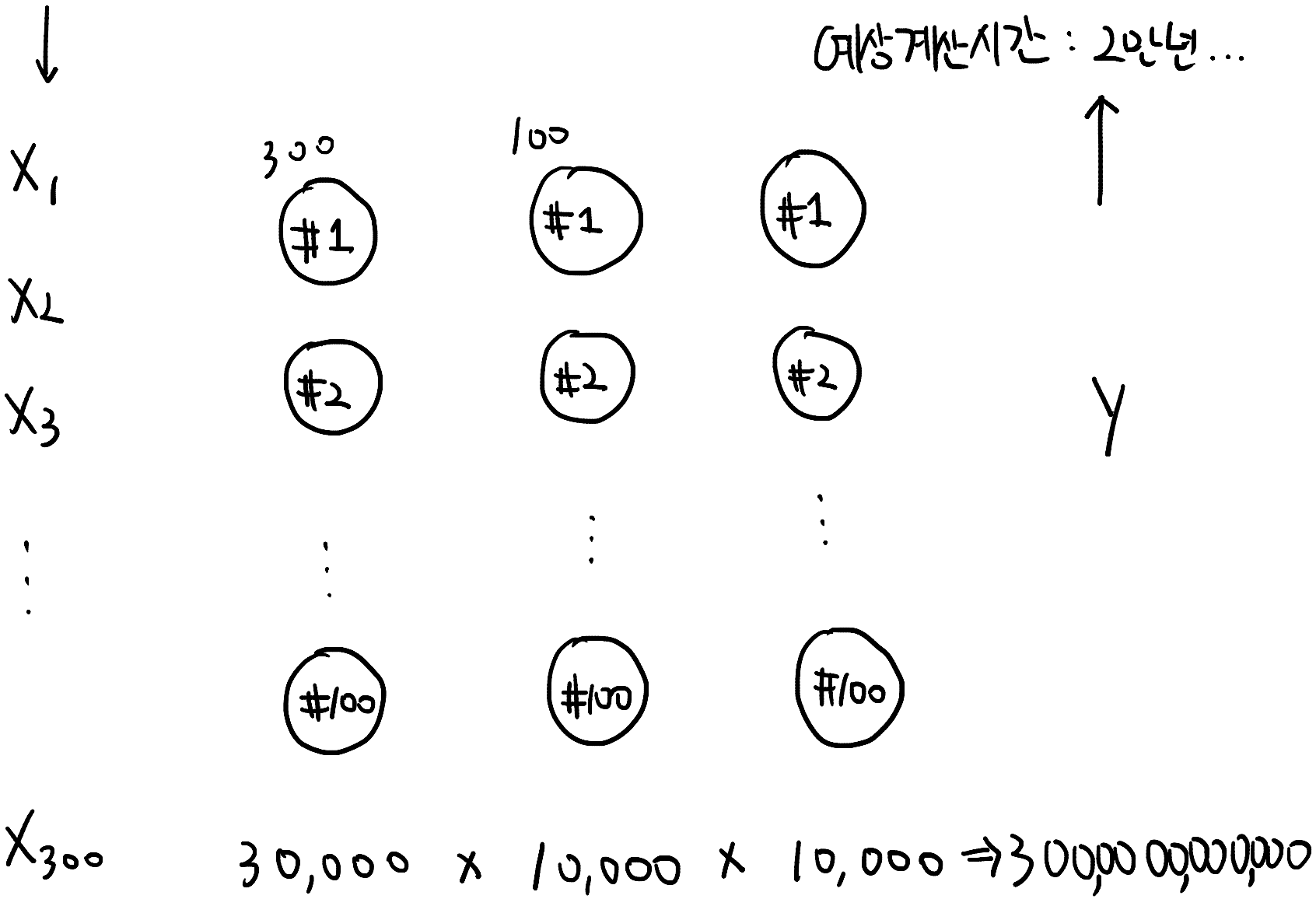
Out[74]: 0.59722796177405213



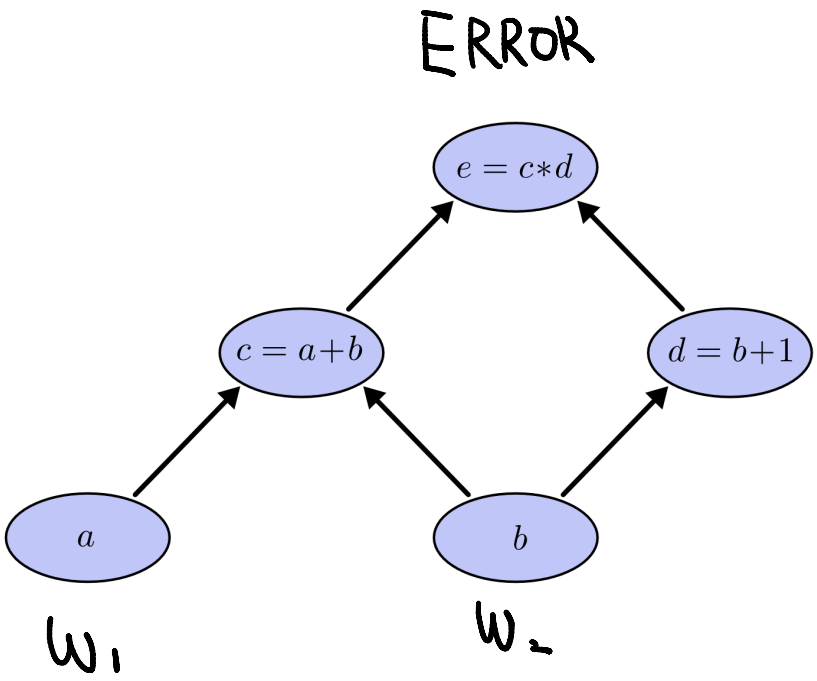
In [75]:
```
1 if y > 0.5 else 0
```

Out[75]: 1

In [ ]:


예상계산시간 : 2만년 ...
X1
X2
X3
X300
300
100
#1
#2
#100
Y
30,000 x 10,000 x 10,000 ⇒ 3000,0000,0000

오차 역전파 (Backpropagation)



c = a + b

d = b + 1

e = c * d

$$\frac{\partial E}{\partial a} \quad \frac{\partial E}{\partial b}$$

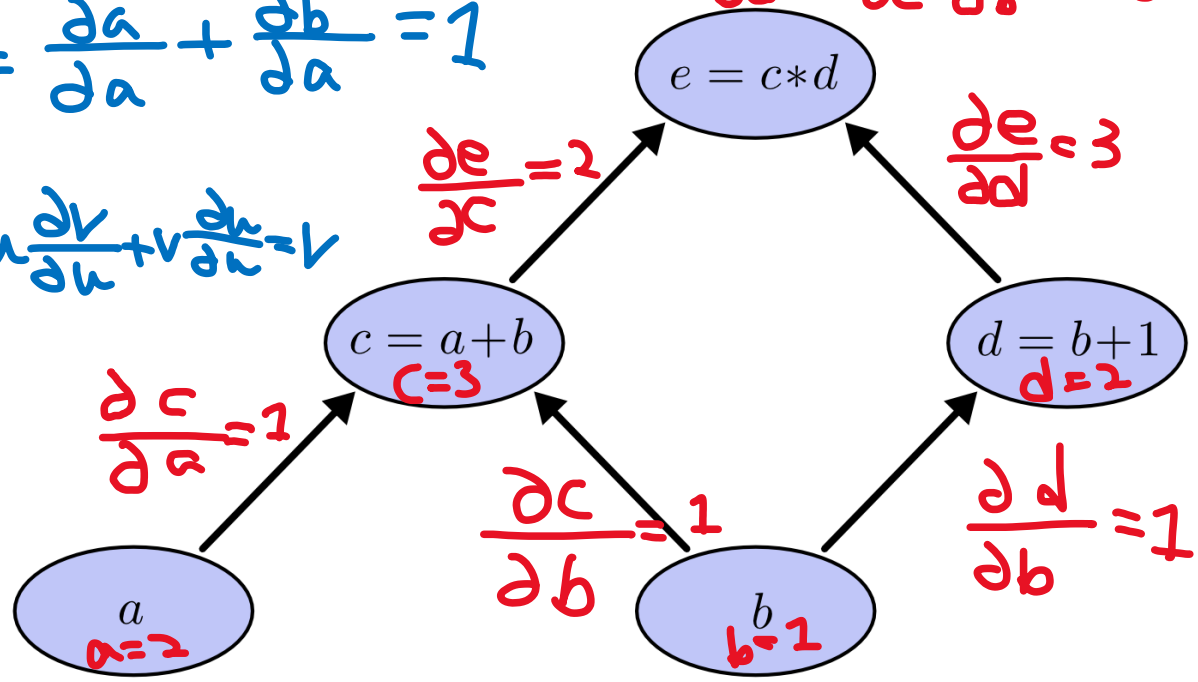
변화량 계산

$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a} = 1$

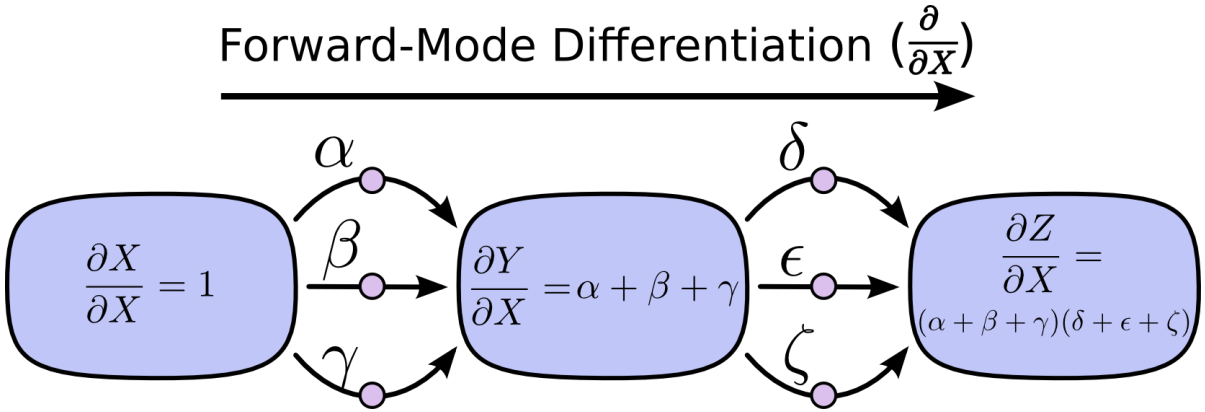
$\frac{\partial}{\partial u}(uv) = u\frac{\partial v}{\partial u} + v\frac{\partial u}{\partial u} = v$

$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c}\frac{\partial c}{\partial a} = 1 \times 2 = 2$

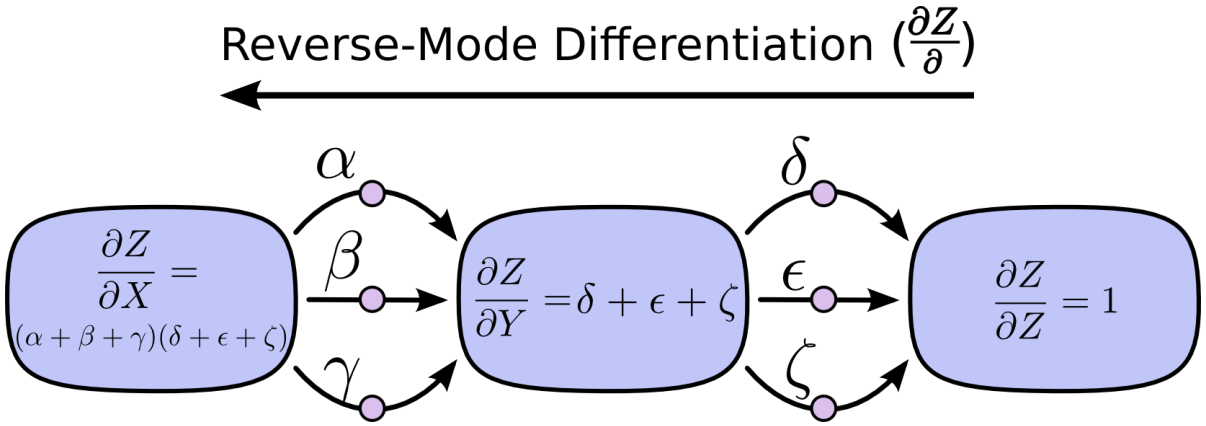
$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c}\frac{\partial c}{\partial b} + \frac{\partial e}{\partial d}\frac{\partial d}{\partial b} = 2\times1 + 3\times1 = 2+3 = 5$



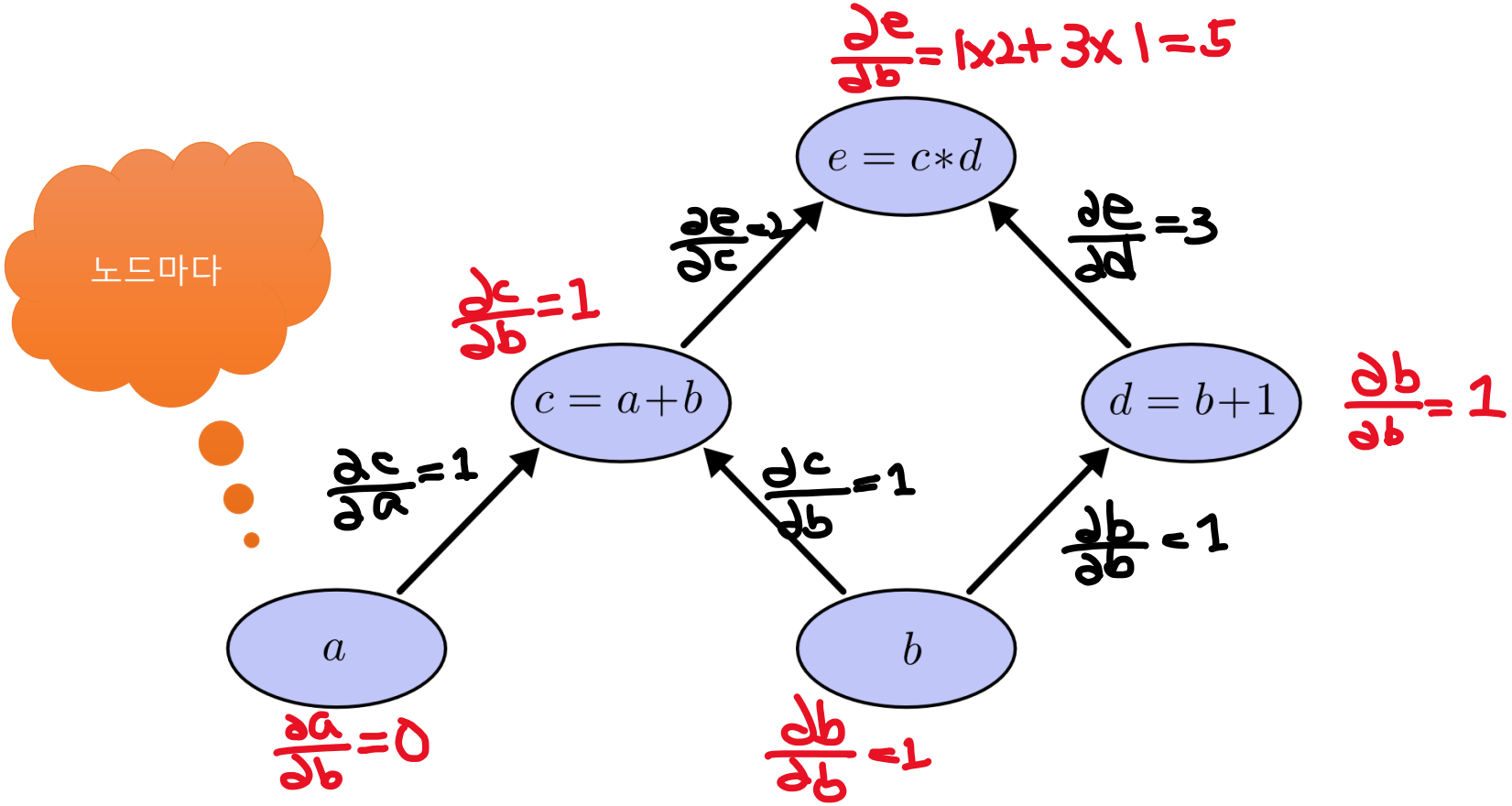
전향 미분


Forward-Mode Differentiation ($\frac{\partial}{\partial X}$)
$\frac{\partial X}{\partial X} = 1$
α
β
γ
$\frac{\partial Y}{\partial X} = \alpha + \beta + \gamma$
δ
ϵ
ζ
$\frac{\partial Z}{\partial X} =$
$(\alpha + \beta + \gamma)(\delta + \epsilon + \zeta)$

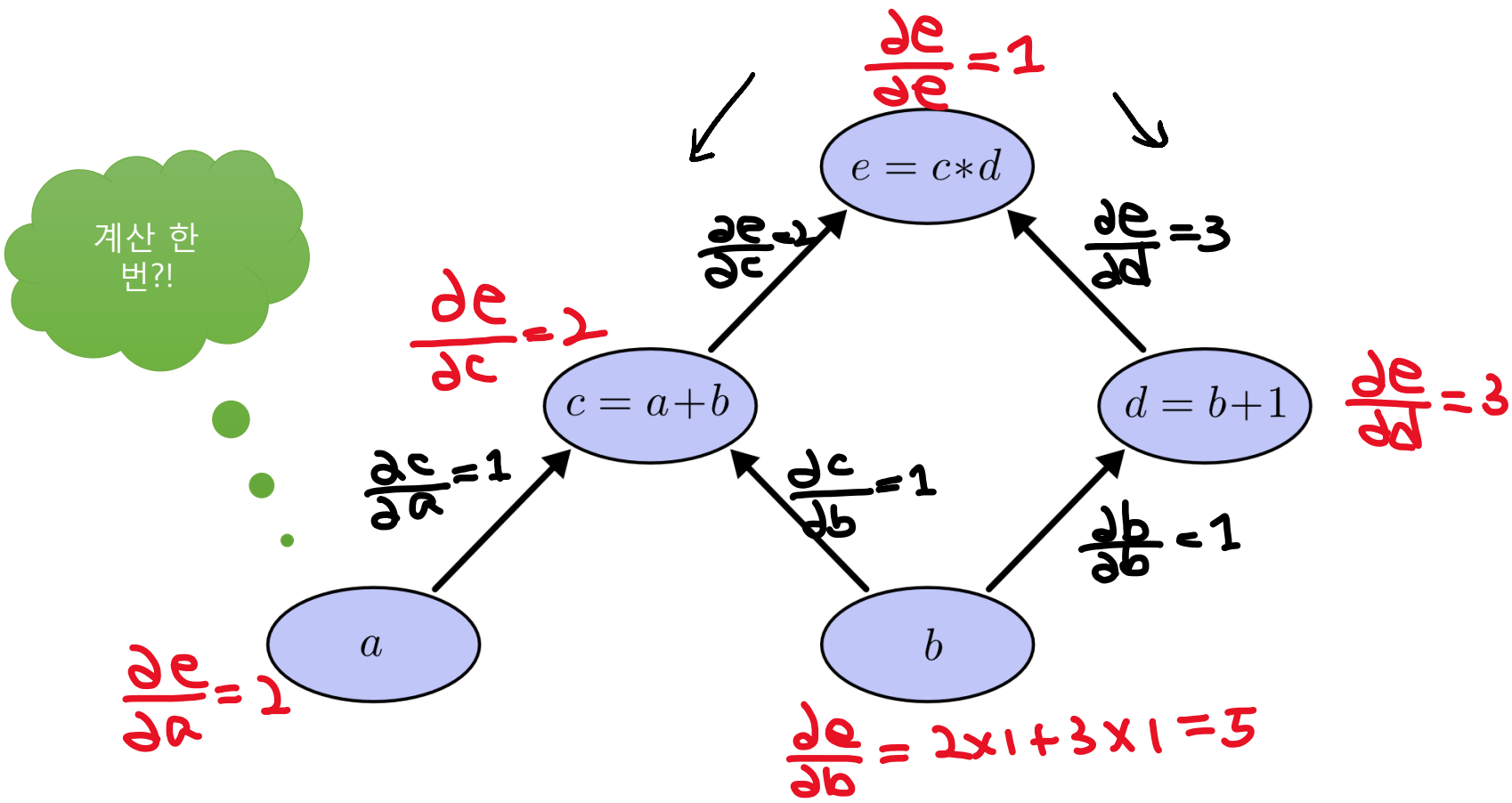
후향 미분


Reverse-Mode Differentiation ($\frac{\partial Z}{\partial}$)
$\frac{\partial Z}{\partial X} = (\alpha+\beta+\gamma)(\delta+\epsilon+\zeta)$
$\frac{\partial Z}{\partial Y} = \delta+\epsilon+\zeta$
$\frac{\partial Z}{\partial Z} = 1$
α
β
γ
δ
ϵ
ζ

변화량 계산: 전향

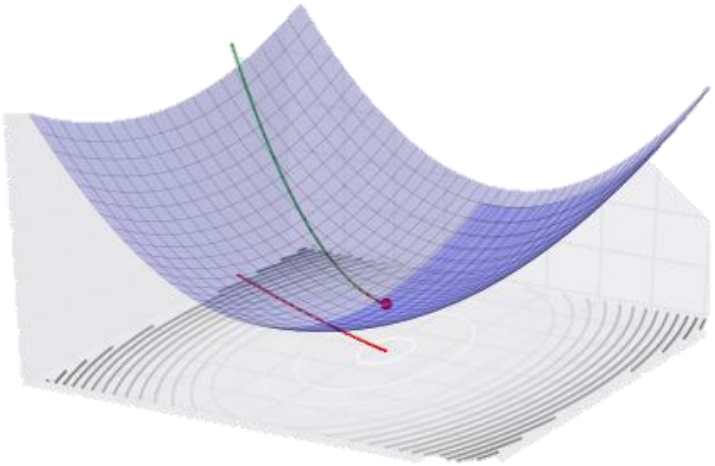

노드마다
$e = c*d$
$c = a+b$
$d = b+1$
a
b
$\frac{\partial e}{\partial b} = 1\times 2 + 3\times 1 = 5$
$\frac{\partial e}{\partial c} = 2$
$\frac{\partial e}{\partial d} = 3$
$\frac{\partial c}{\partial b} = 1$
$\frac{\partial d}{\partial b} = 1$
$\frac{\partial c}{\partial a} = 1$
$\frac{\partial c}{\partial b} = 1$
$\frac{\partial b}{\partial b} = 1$
$\frac{\partial a}{\partial b} = 0$
$\frac{\partial b}{\partial b} = 1$

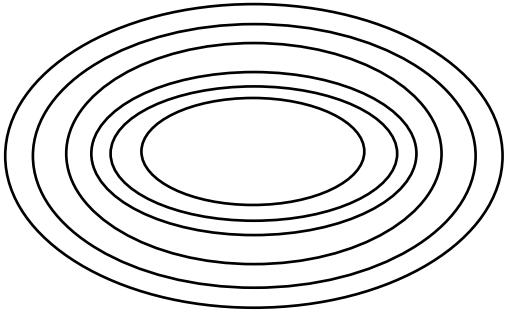
변화량 계산: 후향


계산 한 번?!
$e = c*d$
$c = a+b$
$d = b+1$
a
b
$\frac{\partial e}{\partial e} = 1$
$\frac{\partial e}{\partial c} = 2$
$\frac{\partial e}{\partial d} = 3$
$\frac{\partial e}{\partial c} = 2$
$\frac{\partial e}{\partial d} = 3$
$\frac{\partial c}{\partial a} = 1$
$\frac{\partial c}{\partial b} = 1$
$\frac{\partial d}{\partial b} = 1$
$\frac{\partial e}{\partial a} = 2$
$\frac{\partial e}{\partial b} = 2 \times 1 + 3 \times 1 = 5$

최적화

ERROR ≈ Loss 손실함수





Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [76]:
```
def 평균제곱오차(y, y_pred):
    return np.mean((y-y_pred)**2)
```

? 0-9

정답

In [77]:
```
y = np.array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

0 1 2 ... 9

R 0 0 1

G 0 1 0

B 1 0 0

y - ŷ = 0

y - ŷ

2 2 =0

2 - 3 =-1

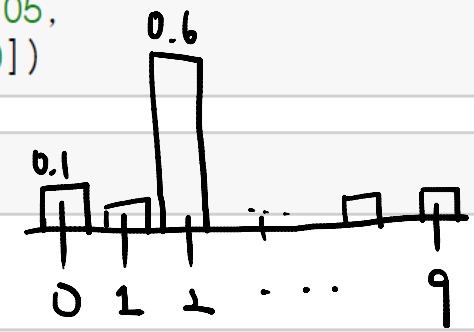
2 - 9 =-7

In [78]:
```
len(y)
```

Out[78]: 10

In [79]:
```
y_pred1 = np.array([
    0.1, 0.05, 0.6, 0.0, 0.05,
    0.1, 0.0, 0.1, 0.0, 0.0])
```

One-Hot



In [80]:
```
len(y_pred1)
```

Out[80]: 10

In [81]:
```
y_pred1.sum()
```

Out[81]: 1.0

In [ ]:

Trusted | Python 3

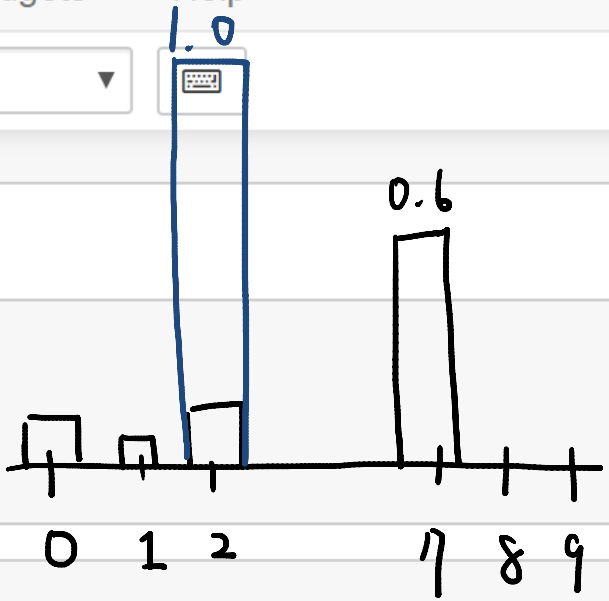
File Edit View Insert Cell Kernel Widgets Help

Code

In [81]:
```
y_pred1.sum()
```

Out[81]: 1.0

In [82]:
```
y_pred2 = np.array([
    0.1, 0.05, 0.1, 0.0, 0.05,
    0.1, 0.0, 0.6, 0.0, 0.0
])
```



In [83]:
```
len(y_pred2)
```

Out[83]: 10

In [84]:
```
y_pred2.sum()
```

Out[84]: 1.0

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [84]:
```
y_pred2.sum()
```

Out[84]: 1.0

In [86]:
```
loss = 평균제곱오차
```

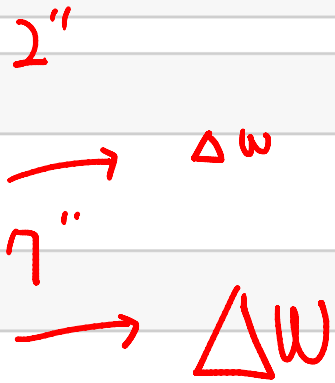
In [87]:
```
loss(y, y_pred1)
```

Out[87]: 0.01950000000000007

In [88]:
```
loss(y, y_pred2)
```

Out[88]: 0.11950000000000001

In [ ]:



Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

Out[88]: 0.11950000000000001

In [89]:
```
def cross_entropy_error(y, y_pred):
    delta = 1e-7 # log(0) -->-무한대 방지
    return -np.sum(y*np.log(y_pred+delta))
```

10^{-7}

In [90]:
```
loss = cross_entropy_error
```

In [91]:
```
loss(y, y_pred1)
```

Δw

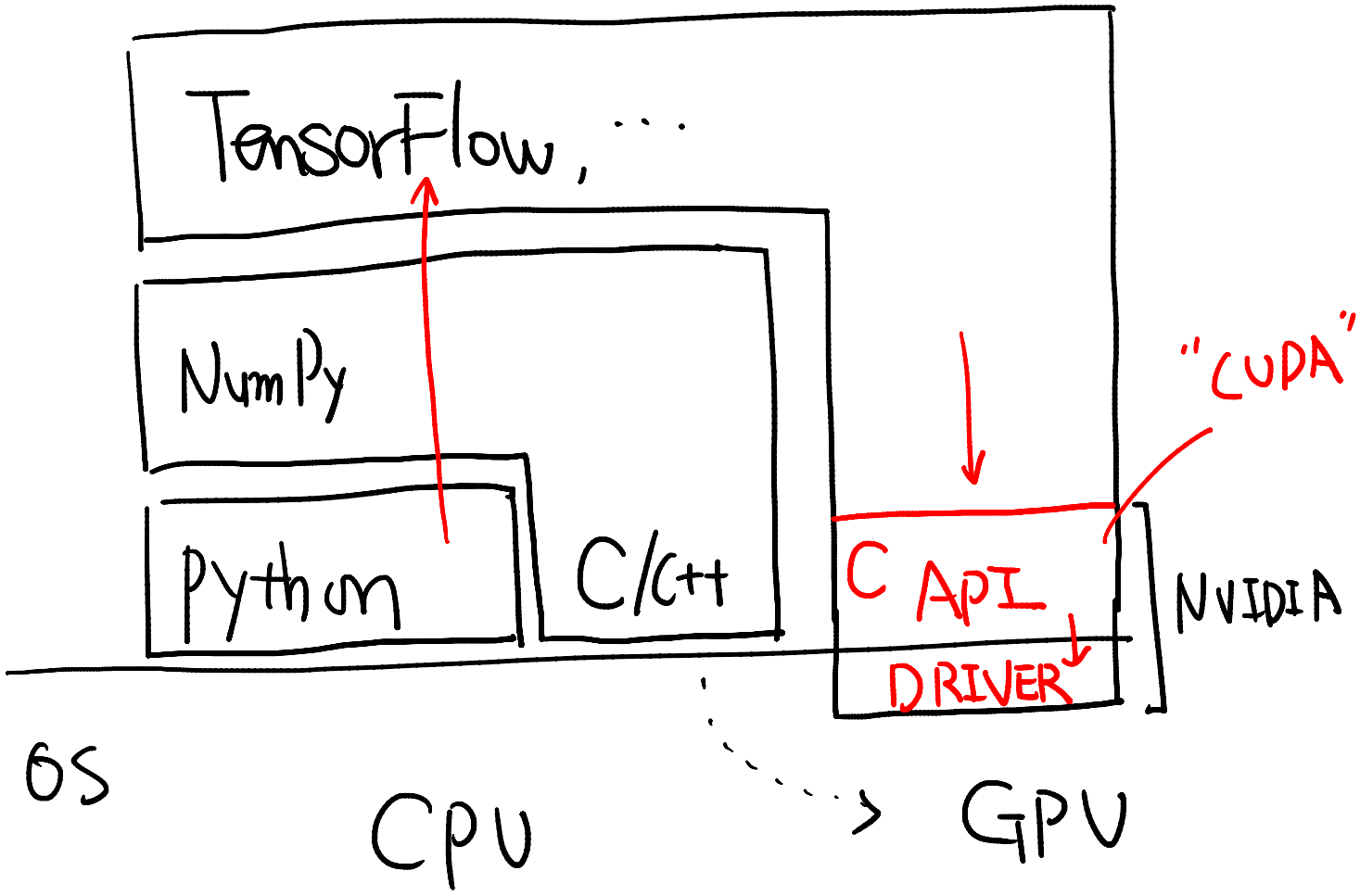
Out[91]: 0.51082545709933802

In [92]:
```
loss(y, y_pred2)
```

ΔW

Out[92]: 2.3025840929945458

In [ ]:


TensorFlow, ...
NumPy
Python
C/C++
C API
"CUDA"
NVIDIA
DRIVER
OS
CPU
GPU

PyPI - the Python Package Index : Python Package Index - Chrome

Python Software Foundation [US] | https://pypi.python.org/pypi

search

» Package Index

~라이브러리 "App Store"

PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **126358** packages here. To contact the PyPI admins, please use the Support or Bug reports links.

PACKAGE INDEX
- Browse packages
- List trove classifiers
- RSS (latest 40 updates)
- RSS (newest 40 packages)
- Terms of Service
- PyPI Tutorial
- PyPI Security
- PyPI Support
- PyPI Bug Reports
- PyPI Discussion
- PyPI Developer Info

ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
COMMUNITY
FOUNDATION
CORE DEVELOPMENT

Not Logged In
- Login
- Register
- Lost Login?
- Login with OpenID
- Login with Google

Status
Nothing to report

Get Packages

To use a package from this index either "pip install *package*" (get pip) or download, unpack and "python setup.py install" it.

Package Authors

Submit packages with "python setup.py upload". You can also use twine! The index hosts package docs. You must register. Testing? Use testpypi.

Infrastructure

To interoperate with the index use the JSON, XML-RPC or HTTP interfaces. Use local mirroring or caching to make installation more robust.

Updated	Package	Description
2018-01-11	ts3ekkomanage 0.0.3.dev7	ts3ekkomanage
2018-01-11	angel_web 1.0.9	The admin web dashboard of a distribute job scheduler named 'angel'
2018-01-11	jstquote 0.0.2	A command line interface for collecting and organizing

tensorflow-1.5.0rc0-cp36-cp36m-macosx_10_11_x86_64.whl (md5)	Python Wheel	cp36	2018-01-04	39MB
tensorflow-1.5.0rc0-cp36-cp36m-manylinux1_x86_64.whl (md5)	Python Wheel	cp36	2018-01-04	41MB
tensorflow-1.5.0rc0-cp36-cp36m-win_amd64.whl (md5)	Python Wheel	cp36	2018-01-04	29MB

Author: Google Inc.
Home Page: https://www.tensorflow.org/
Keywords: tensorflow tensor machine learning
License: Apache 2.0
Categories

- Development Status :: 4 - Beta
- Intended Audience :: Developers
- Intended Audience :: Education
- Intended Audience :: Science/Research
- License :: OSI Approved :: Apache Software License
- Programming Language :: Python :: 2
- Programming Language :: Python :: 2.7
- Programming Language :: Python :: 3
- Programming Language :: Python :: 3.4
- Programming Language :: Python :: 3.5
- Programming Language :: Python :: 3.6
- Topic :: Scientific/Engineering
- Topic :: Scientific/Engineering :: Artificial Intelligence
- Topic :: Scientific/Engineering :: Mathematics
- Topic :: Software Development
- Topic :: Software Development :: Libraries
- Topic :: Software Development :: Libraries :: Python Modules

Requires Distributions

- wheel
- tensorflow-tensorboard
- six (>=1.10.0)
- protobuf (>=3.4.0)
- numpy (>=1.12.1)

This guide explains how to install TensorFlow on Windows.

Determine which TensorFlow to install

You must choose one of the following types of TensorFlow to install:

- **TensorFlow with CPU support only**. If your system does not have a NVIDIA® GPU, you must install this version. Note that this version of TensorFlow is typically much easier to install (typically, in 5 or 10 minutes), so even if you have an NVIDIA GPU, we recommend installing this version first.
- **TensorFlow with GPU support**. TensorFlow programs typically run significantly faster on a GPU than on a CPU. Therefore, if your system has a NVIDIA® GPU meeting the prerequisites shown below and you need to run performance-critical applications, you should ultimately install this version.

Requirements to run TensorFlow with GPU support

If you are installing TensorFlow with GPU support using one of the mechanisms described in this guide, then the following NVIDIA software must be installed on your system:

- CUDA® Toolkit 8.0. For details, see NVIDIA's documentation Ensure that you append the relevant Cuda pathnames to the `%PATH%` environment variable as described in the NVIDIA documentation.
- The NVIDIA drivers associated with CUDA Toolkit 8.0.
- cuDNN v6.0. For details, see NVIDIA's documentation. Note that cuDNN is typically installed in a different location from the other CUDA DLLs. Ensure that you add the directory where you installed the cuDNN DLL to your `%PATH%` environment variable.
- GPU card with CUDA Compute Capability 3.0 or higher. See NVIDIA documentation for a list of supported GPU cards.

If you have a different version of one of the preceding packages, please change to the specified versions. In particular, the cuDNN version must match exactly: TensorFlow will not load if it cannot find `cuDNN64_6.dll`. To use a different version of cuDNN, you must build from source.

Determine how to install TensorFlow

You must pick the mechanism by which you install TensorFlow. The supported choices are as follows:

- "native" pip
- Anaconda

Native pip installs TensorFlow directly on your system without going through a virtual environment. Since a native pip installation is not walled-off in a separate container, the pip installation might interfere with other Python-based installations on your system. However, if you understand pip and your Python environment, a native pip installation often entails only a single command! Furthermore, if you install with native pip, users can run TensorFlow programs from any directory on the system.

In Anaconda, you may use conda to create a virtual environment. However, within Anaconda, we recommend installing TensorFlow with the `pip install` command, not with the `conda install` command.

NOTE: The conda package is community supported, not officially supported. That is, the TensorFlow team neither tests nor maintains this conda package. Use that package at your own risk.

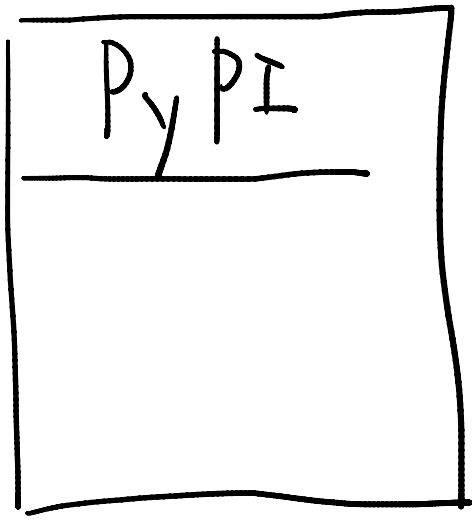
Installing with native pip

If one of the following versions of Python is not installed on your machine, install it now:

- Python 3.5.x 64-bit from python.org
- Python 3.6.x 64-bit from python.org

-TensorFlow supports Python 3.5.x and 3.6.x on Windows. Note that Python 3 comes with the pip3 package manager, which is the program you'll use to install TensorFlow.

To install TensorFlow, start a terminal. Then issue the appropriate `pip3 install` command in that terminal. To install the CPU-only version of TensorFlow, enter the following command:



$ pip install

$ conda install

Installing with native pip

$ pip install tensorflow

If one of the following versions of Python is not installed on your machine, install it now:

- Python 3.5.x 64-bit from python.org
- Python 3.6.x 64-bit from python.org

-TensorFlow supports Python 3.5.x and 3.6.x on Windows. Note that Python 3 comes with the pip3 package manager, which is the program you'll use to install TensorFlow.

To install TensorFlow, start a terminal. Then issue the appropriate `pip3 install` command in that terminal. To install the CPU-only version of TensorFlow, enter the following command:

```
C:\> pip3 install --upgrade tensorflow
```

To install the GPU version of TensorFlow, enter the following command:

```
C:\> pip3 install --upgrade tensorflow-gpu
```

Installing with Anaconda

The Anaconda installation is community supported, not officially supported.

Take the following steps to install TensorFlow in an Anaconda environment:

1. Follow the instructions on the Anaconda download site to download and install Anaconda.
2. Create a conda environment named `tensorflow` by invoking the following command:

```
C:> conda create -n tensorflow python=3.5
```

3. Activate the conda environment by issuing the following command:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

```
    return -np.sum(y*np.log(y_pred+delta))
```

In [90]:
```
loss = cross_entropy_error
```

In [91]:
```
loss(y, y_pred1)
```

Out[91]: 0.51082545709933802

In [92]:
```
loss(y, y_pred2)
```

Out[92]: 2.3025840929945458

$pip install tensorflow

In [93]:
```
import tensorflow
```

In [ ]:

Notebook saved | Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

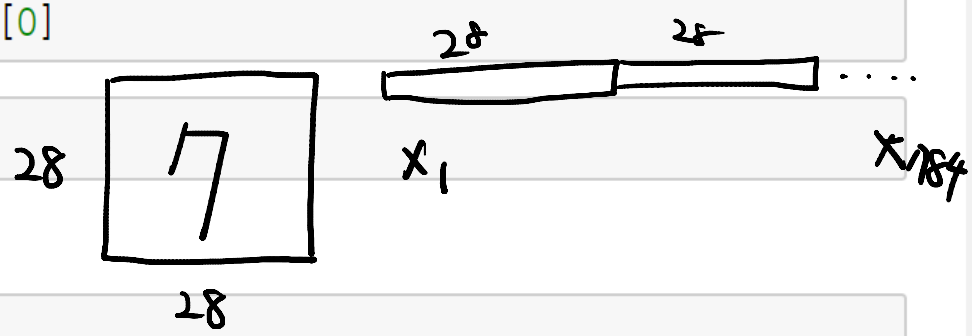
In [125]:
```
mnist = input_data.read_data_sets('mnist/', one_hot=True)
```

...

In [130]:
```
x = mnist.train.images[0]
```

In [132]:
```
type(x)
```

Out[132]: numpy.ndarray



In [131]:
```
x.shape
```

Out[131]: (784,)

In [137]:
```
x = np.reshape(x, (28, 28))
```

In [138]:
```
x.shape
```

Out[138]: (28, 28)

In [133]:
```
from scipy.misc import imsave
```

Trusted | Python 3

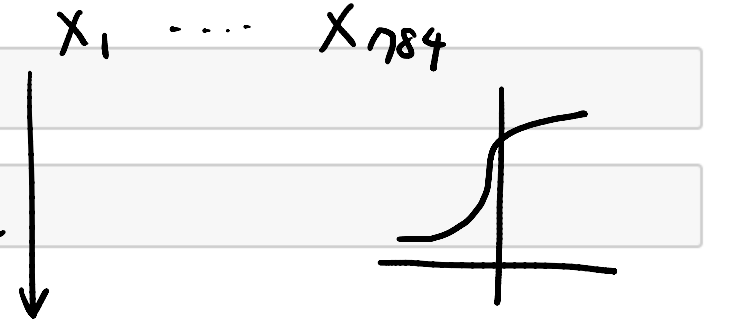
File Edit View Insert Cell Kernel Widgets Help

Code

Out[138]: (28, 28)

In [133]:
```
from scipy.misc import imsave
```

In [139]:
```
imsave('mnist_sample.png', x)
```

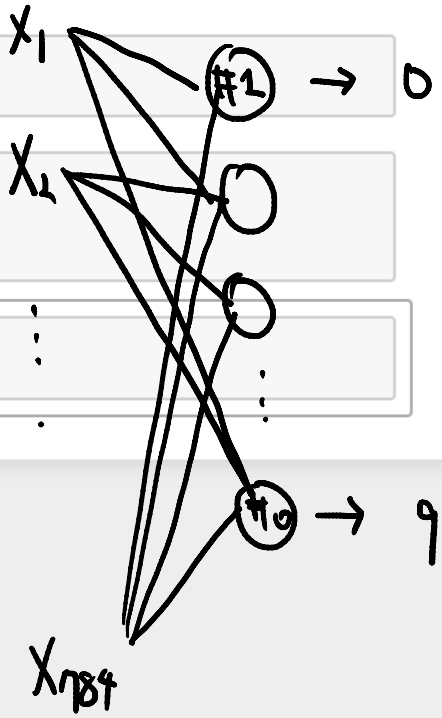


신경망 모델 구성

In [140]:
```
x = tf.placeholder(tf.float32, [None, 784])
```

In [141]:
```
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

In [ ]:



Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [133]:
```
from scipy.misc import imsave
```

In [139]:
```
imsave('mnist_sample.png', x)
```

신경망 모델 구성

In [140]:
```
x = tf.placeholder(tf.float32, [None, 784])
```

In [141]:
```
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

In [142]:
```
y = tf.nn.softmax(tf.matmul(x, W)+b)
```

z → A

y=z (1960)

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [144]:
```
def softmax(a):
    c = np.max(a)
    ea = np.exp(a-c)
    return ea / np.sum(ea)
```

In [ ]:
```
a = np.array([0.3, 2.9, 4.0])
```

In [148]:
```
a.sum()
```

Out[148]: 7.1999999999999993

In [145]:
```
y = softmax(a)
```

In [146]:
```
y
```

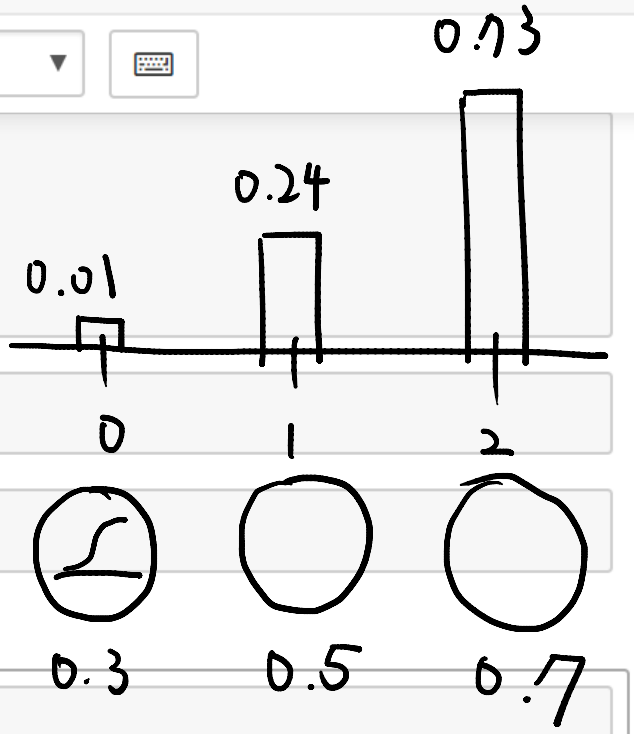
Out[146]: array([0.01821127, 0.24519181, 0.73659691])

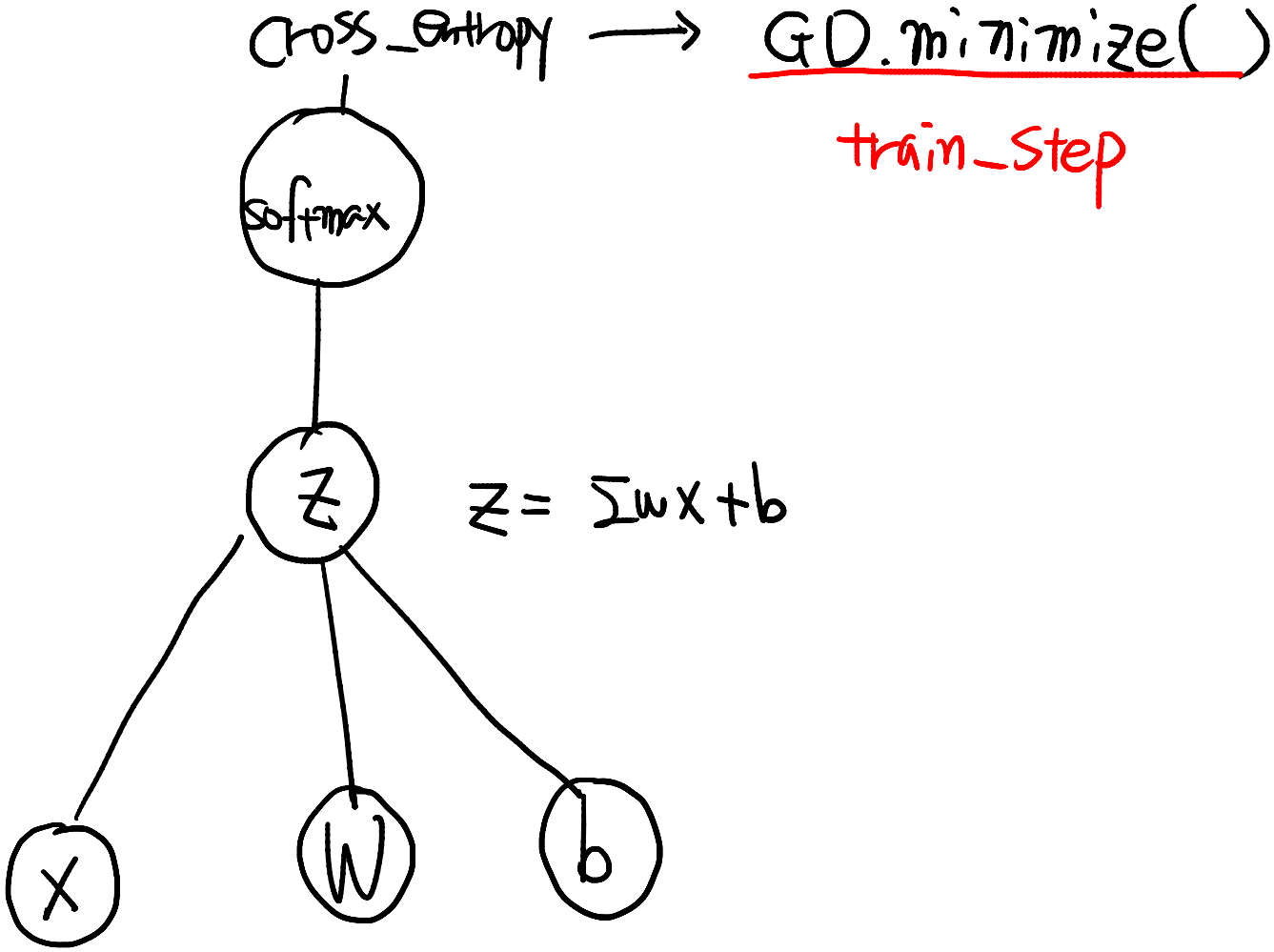
In [147]:
```
y.sum()
```

Out[147]: 1.0

In [ ]:
```

```




Cross_entropy
GD.minimize( )
train_step
softmax
z
z = Σwx+b
x
W
b

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

계산그래프 실행

In [159]:
```
세션 = tf.InteractiveSession()
```

In [160]:
```
tf.global_variables_initializer().run()
```

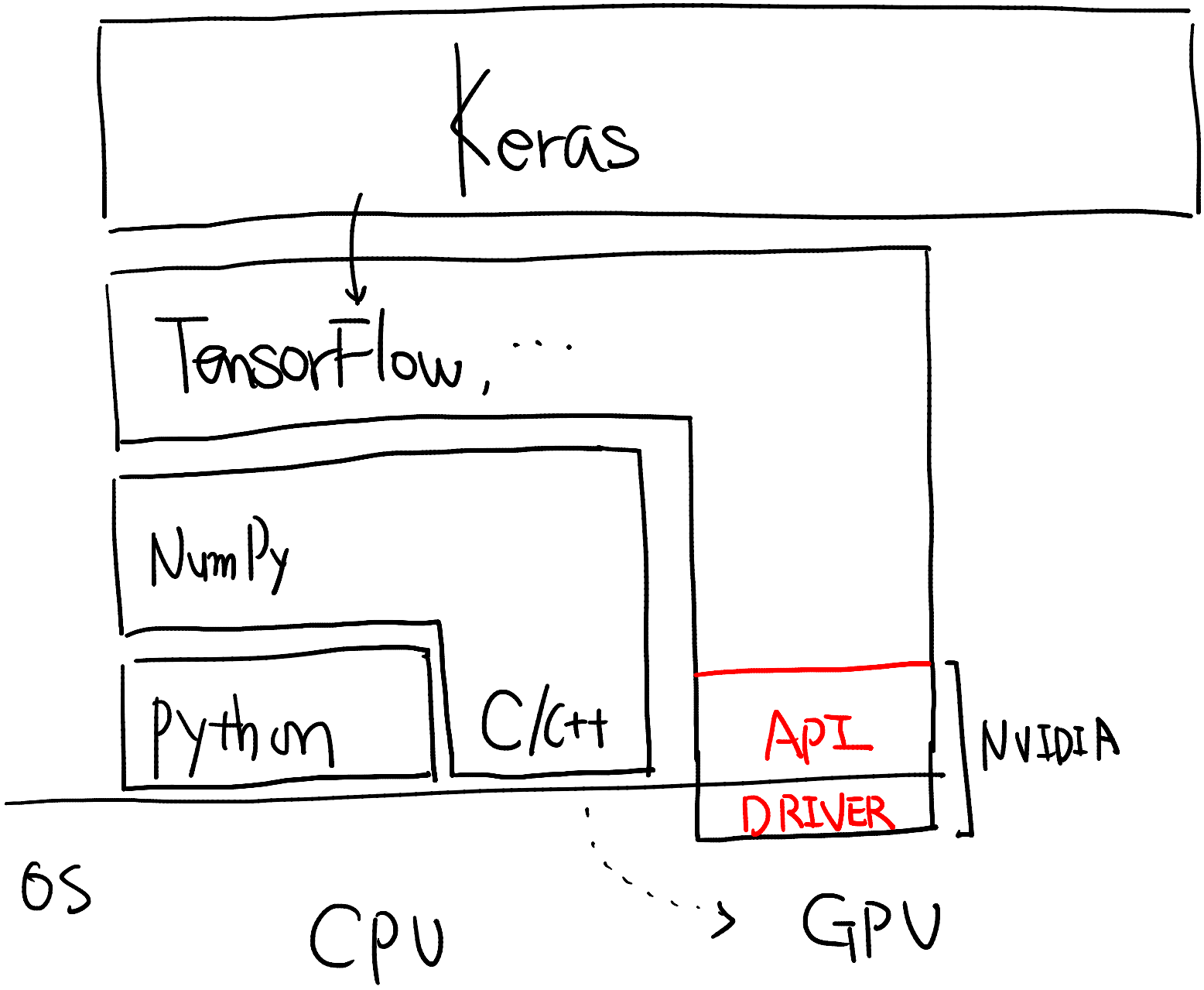
1000번 훈련 수행

In [161]:
```
for _ in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    세션.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

"무작위 100개"

x train y train

In [ ]:


Keras
TensorFlow, ...
NumPy
Python
C/C++
API
DRIVER
NVIDIA
OS
CPU
GPU

```
    정확도,
    feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

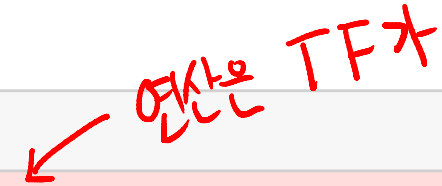
0.916

Keras

numpy 또는 tensorflow로 인공 신경망을 직접 만드는 것은 꽤나 힘든 일입니다.

In [165]:
```
import keras
```

```
Using TensorFlow backend.
```



In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

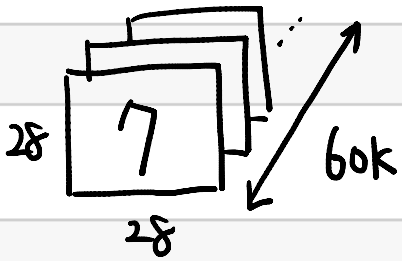
In [166]:
```
from keras.datasets import mnist
```

In [167]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 6s 0us/step
```

In [168]:
```
X_train.shape
```

Out[168]:
```
(60000, 28, 28)
```



In [170]:
```
y_train[:10]
```

Out[170]:
```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)
```

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 6s 0us/step
```

In [168]:
```
X_train.shape
```

Out[168]:
```
(60000, 28, 28)
```

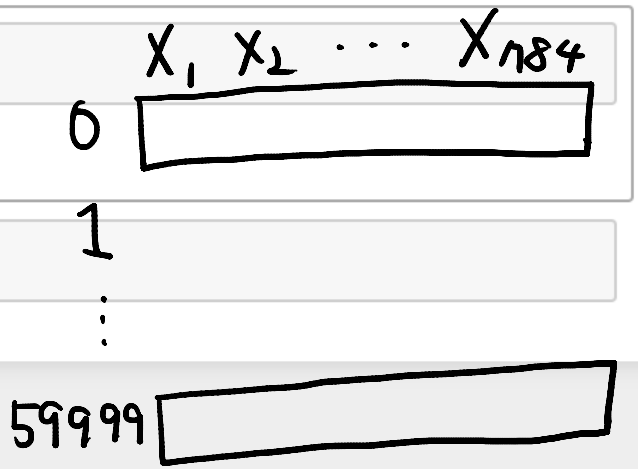
In [170]:
```
y_train[:10]
```

Out[170]:
```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)
```

In [171]:
```
X_train = X_train.reshape(60000, 784)
```

Out[171]:
```
(60000, 784)
```

In [ ]:



Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

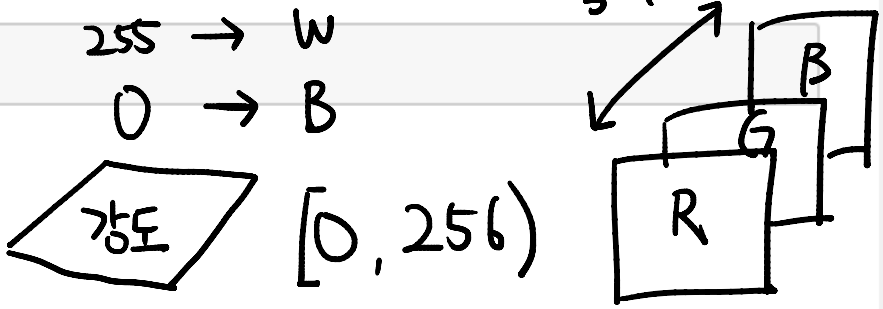
Code

In [172]:
```
X_train = X_train.reshape(60000, 784)
```

In [174]:
```
X_test = X_test.reshape(10000, 28*28)
```

In [175]:
```
X_test.shape
```

Out[175]: (10000, 784)



자료형이 중요합니다.

In [176]:
```
X_train.dtype
```

Out[176]: dtype('uint8')

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [170]:
```
y_train[:10]
```

Out[170]:
```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)
```

One-Hot Encoding

"0" 1 0 0 · · · 0

In [182]:
```
from keras.utils import np_utils
```

In [183]:
```
np_utils.to_categorical(y_train)
```

Out[183]:
```
array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 1.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  1.,  0.]])
```

"0"

"8"

In [ ]:

Permission denied: Documents/pydeep/Day 4.ipynb | Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [188]:
```
model = Sequential()
```

층 추가

In [189]:
```
from keras.layers.core import Dense, Activation
```

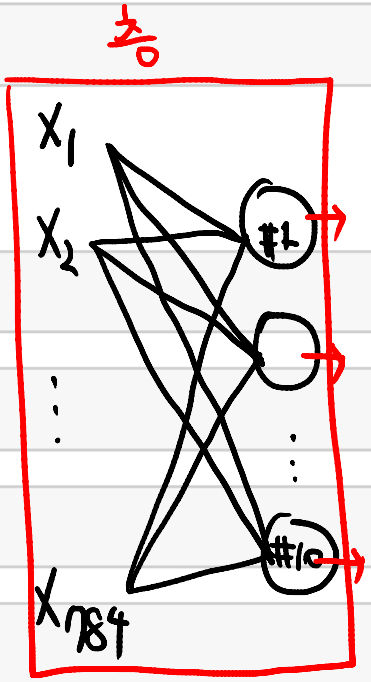
In [190]:
```
layer1 = Dense(10, input_shape=(28*28,))
```

출력

In [192]:
```
model.add(layer1)
```

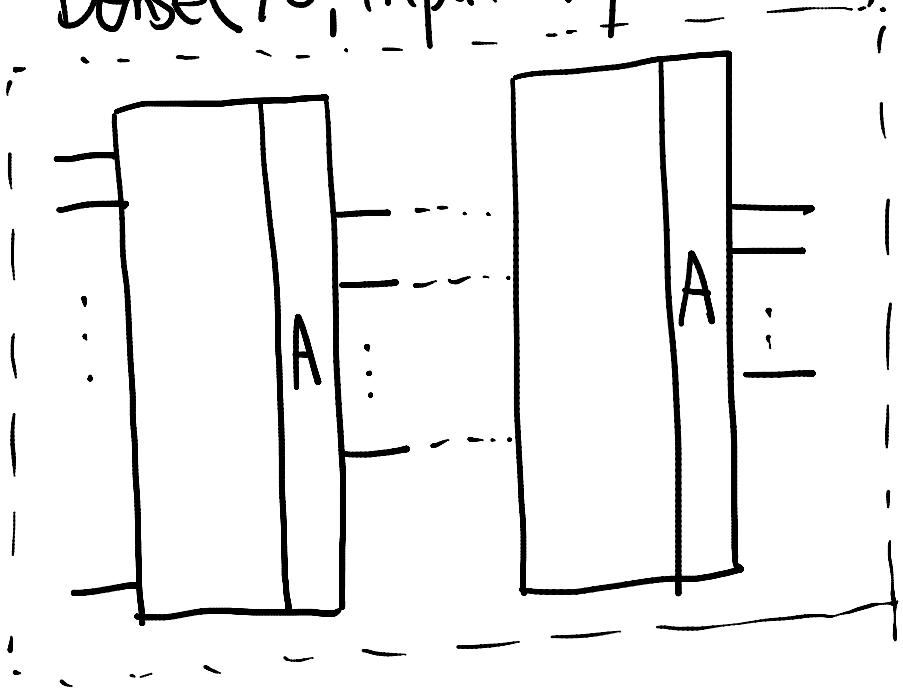
In [193]:
```
model.add(Activation('softmax'))
```

In [ ]:



model = Sequential( )

→ Dense(10, input_shape=(784,), activation='softmax')



Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [194]:
```
model.compile(
    loss='categorical_crossentropy',
    optimizer='sgd',
    metrics=['accuracy']
)
```

훈련

In [*]:
```
history = model.fit(
    X_train, Y_train, batch_size=128, epochs=200,
    validation_split=0.2
)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/200
48000/48000 [==============================] - 1s 14us/step - loss:
1.3958 - acc: 0.6675 - val_loss: 0.8939 - val_acc: 0.8288
Epoch 2/200
48000/48000 [==============================] - 1s 11us/step - loss:
0.7898 - acc: 0.8303 - val_loss: 0.6547 - val_acc: 0.8603
Epoch 3/200
48000/48000 [==============================] - 1s 12us/step - loss:
0.6404 - acc: 0.8523 - val_loss: 0.5596 - val_acc: 0.8703
```

Trusted | Python 3

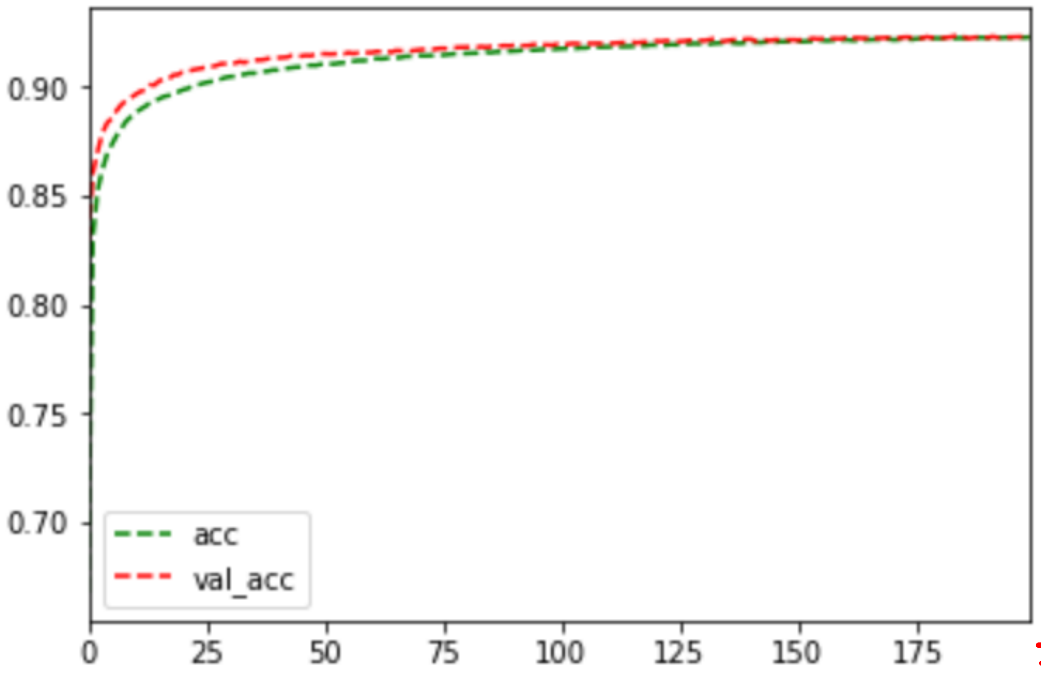
File Edit View Insert Cell Kernel Widgets Help

Code

In [198]:
```
train_results = DataFrame(history.history)
```

In [201]:
```
train_results[['acc', 'val_acc']].plot(style=['g--', 'r--'])
```

Out[201]: <matplotlib.axes._subplots.AxesSubplot at 0xc30be80>



#훈련

In [ ]:

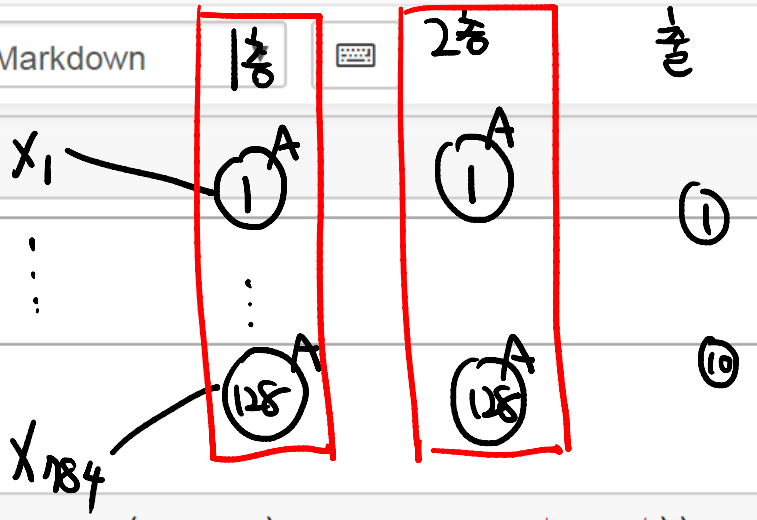
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Markdown

In [205]:
```
model = Sequential()
```

은닉층 (Hidden Layer)



1층

In [209]:
```
model.add(Dense(128, input_shape=(28*28,), activation='relu'))
```

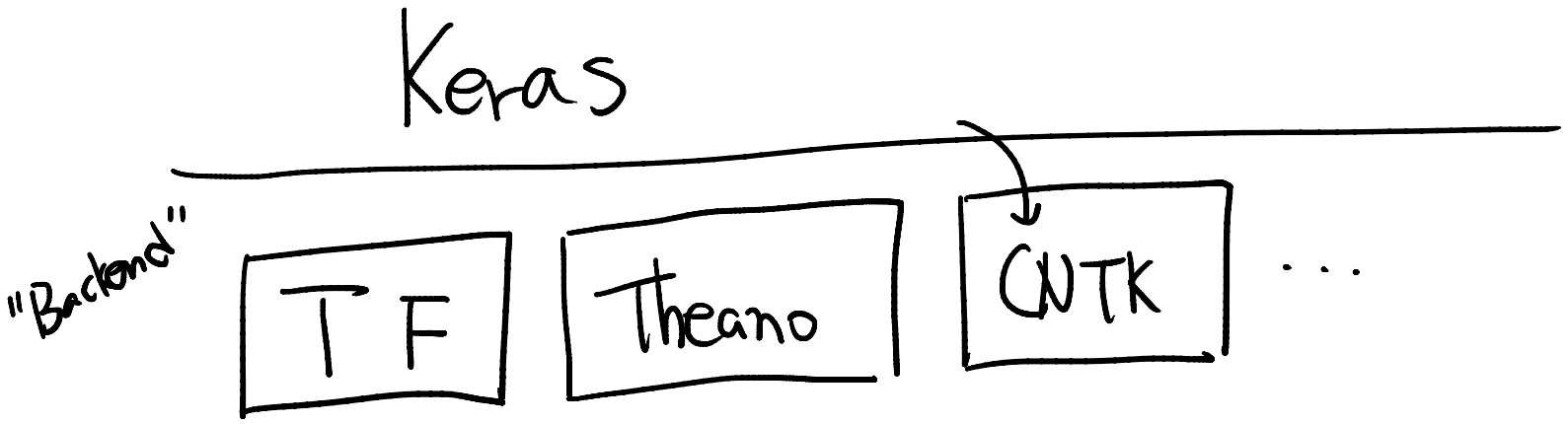
2층

In [210]:
```
model.add(Dense(128, activation='relu'))
```

출력층

In [211]:
```
model.add(Dense(10, activation='softmax'))
```

In [ ]:



.keras/keras.json

```
{
    "floatx": "float32",
    "epsilon": 1e-07,
    "backend": "tensorflow",
    "image_data_format": "channels_last"
}
```

backend 설정