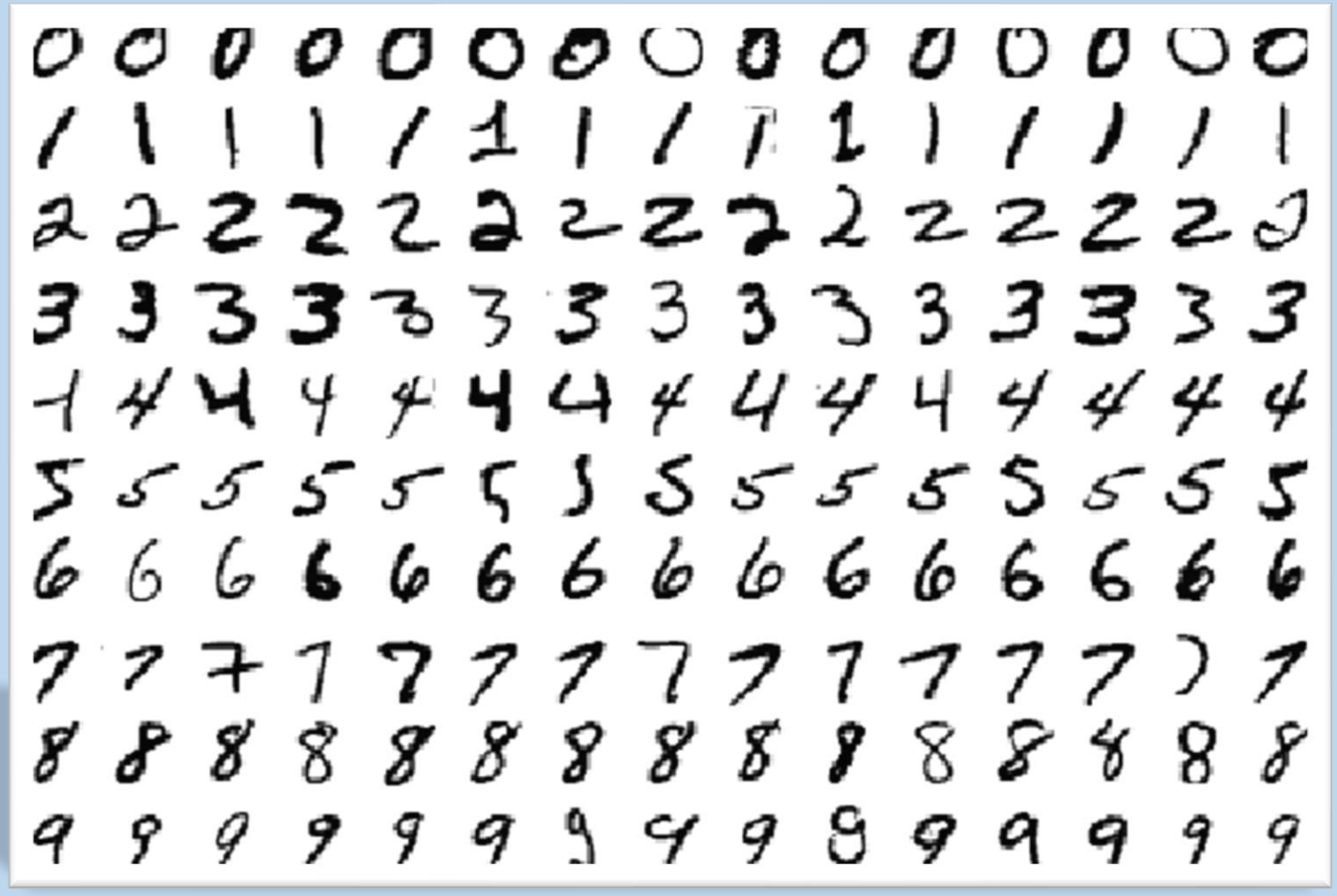
Tensorflow

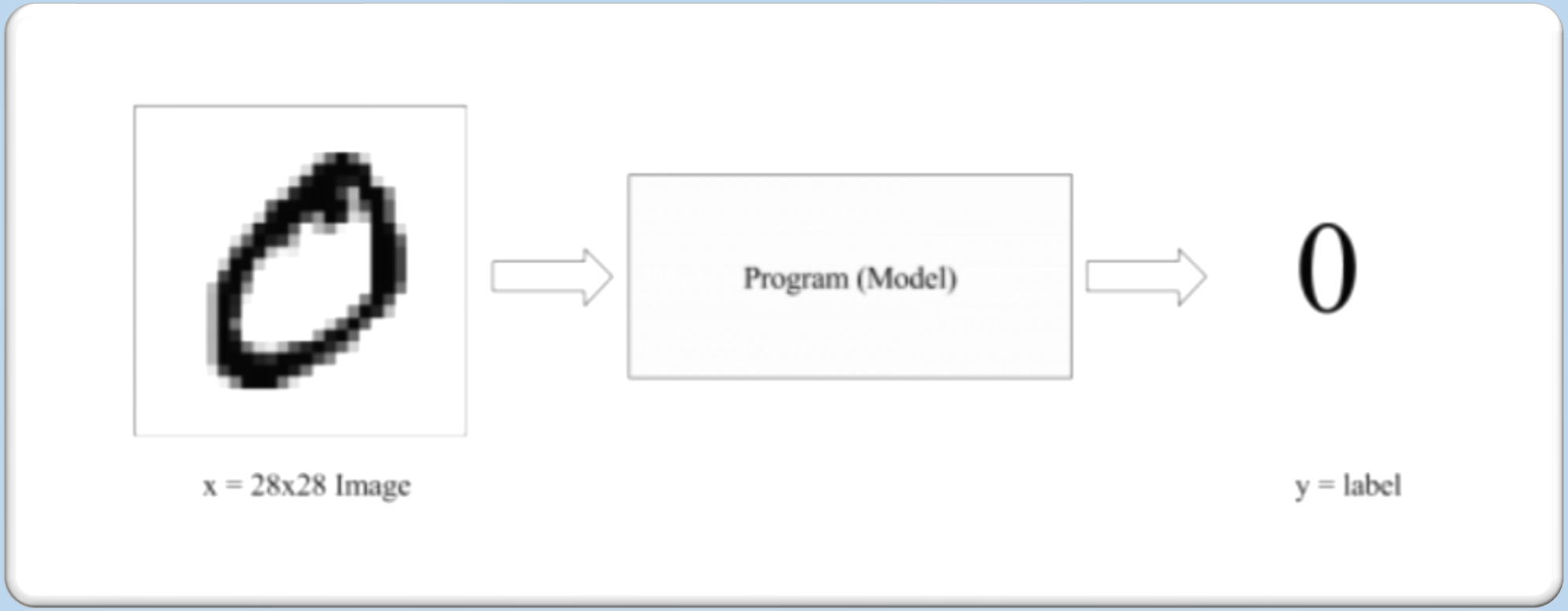
-mnist

contents

Mnist



Mnist



Mnist

≈

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Mnist

≈

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Mnist

활성화 함수(Activation Function)

신경학적으로 볼 때 뉴런발사(Firing of a Neuron)의 과정에 해당한다고 볼 수 있다

최종출력 신호를 다음 뉴런으로 보내줄 지 말지 결정하는 역할을 한다

Activation function의 결정이 결과에 크게 영향을 준다

시그모이드(sigmoid) 함수, 소프트맥스(softmax) 함수, relu 함수

사용할 활성화 함수를 선택할 땐 풀고자 하는 문제의 종류를 고려해야 한다.

1. 문제의 결과가 직선을 따른다면?

선형함수를 이용한다

2. 신경망에서는?

비선형함수를 이용한다

Mnist

활성화 함수(Activation Function)

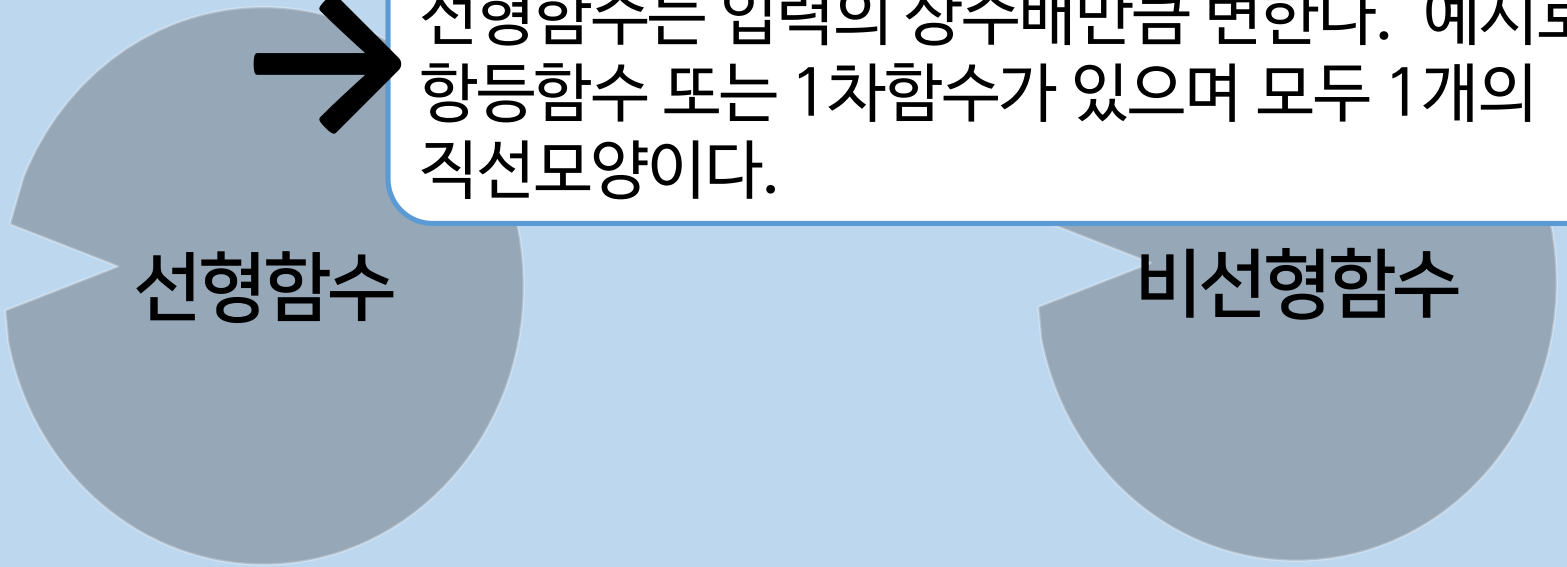
입력 값에 따라 출력 값의 모양이 어떠한 가에 따라 선형과 비 선형으로 나눌 수 있다.

Mnist

활성화 함수(Activation Function)

입력 값에 따라 출력 값의 모양이 어떠한 가에 따라 선형과 비 선형으로 나눌 수 있다.

선형함수는 입력의 상수배만큼 변한다. 예시로는 항등함수 또는 1차함수가 있으며 모두 1개의 직선모양이다.



Mnist

활성화 함수(Activation Function)

입력 값에 따라 출력 값의 모양이 어떠한 가에 따라 선형과 비 선형으로 나눌 수 있다.

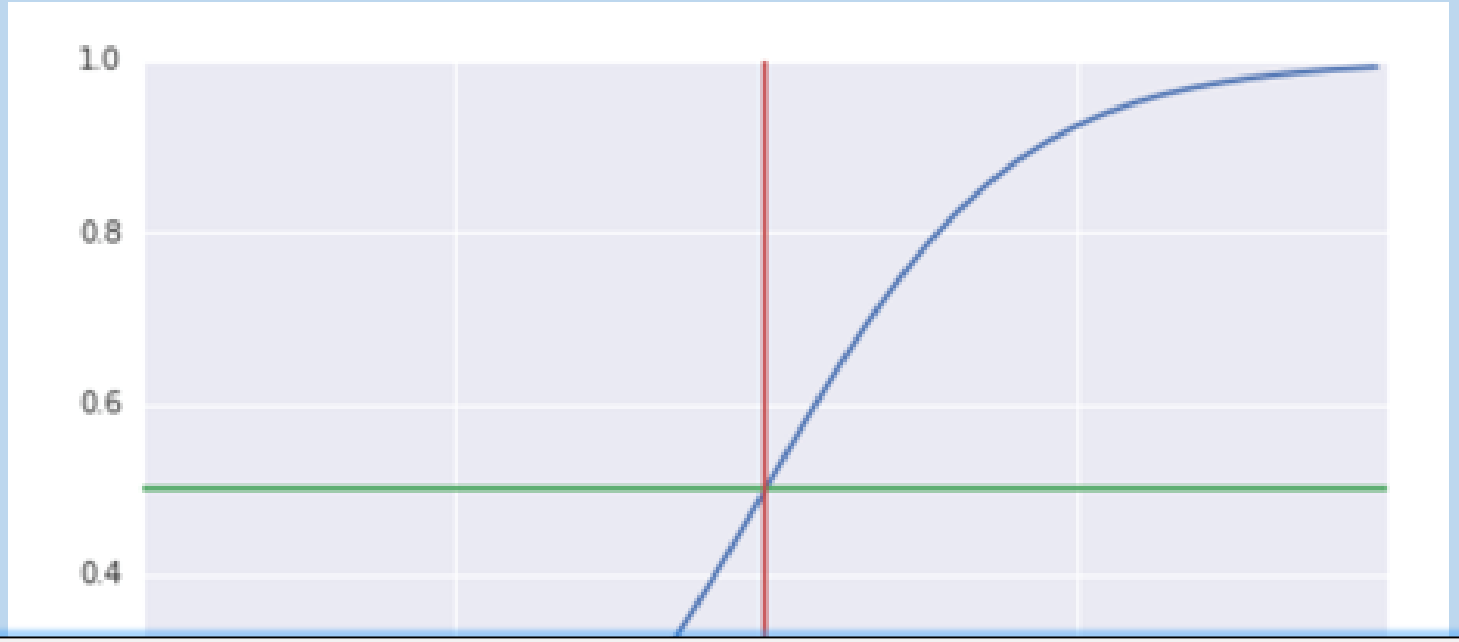
선형함수는 입력의 상수배만큼 변한다. 예시로는 항등함수 또는 1차함수가 있으며 모두 1개의 직선모양이다.

선형함수

비선형함수

반대로 비선형은 2개 이상의 직선 또는 곡선의 모양을 가지고 있다. **비선형에는 Sigmoid, ReLU 등이 있다.**

시그모이드 함수

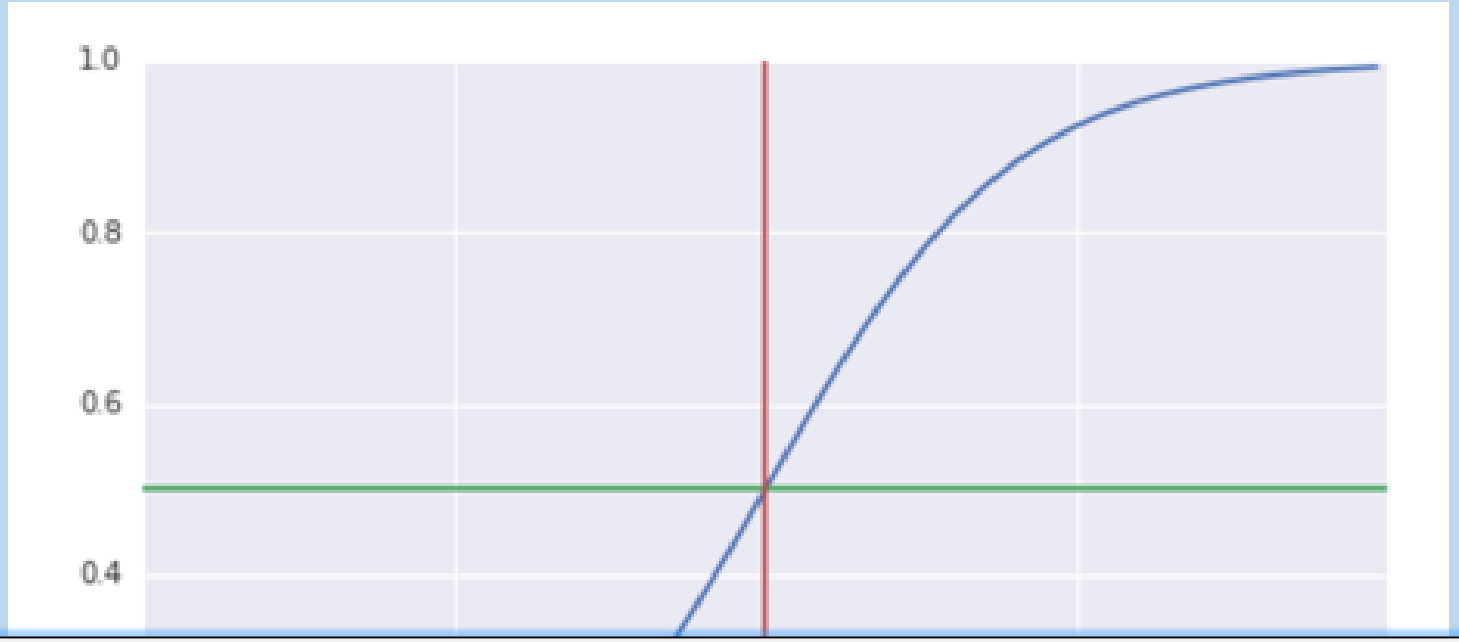


이항 분류 분석에 적절한 함수로 시그모이드(sigmoid)함수가 있다.

S 자 형태의 그래프 모양으로 중심축 (x=0)을 중심으로 좌측은 0으로 수렴하고

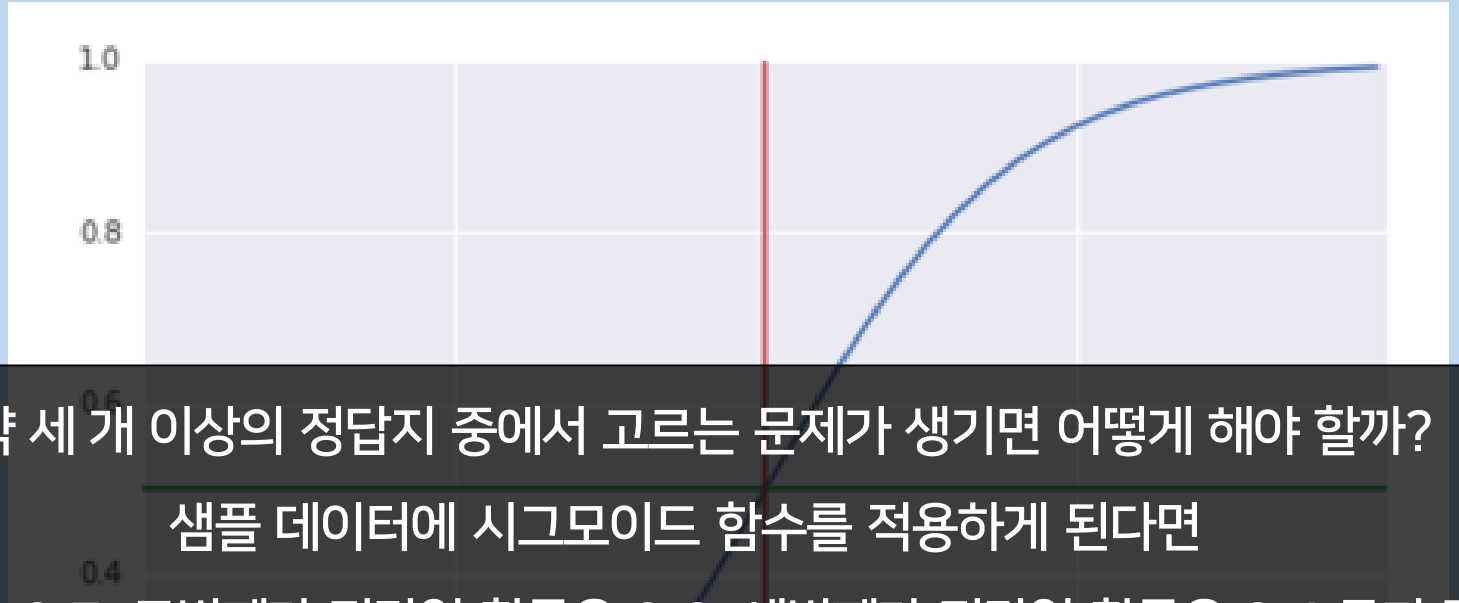
우측은 1로 수렴한다.

시그모이드 함수



예를 들어 0이 정상 메일, 1이 스팸 메일이라고 정의해놓는다면 시그모이드 함수의 0과 1사이의 출력 값을 스팸 메일일 확률로 해석할 수 있다.
확률값이 0.5를 넘으면 1에 더 가까우니까 스팸 메일로 판단하면 되고, 그 반대라면 정상 메일로 판단하면 된다!

시그모이드 함수



만약 세 개 이상의 정답지 중에서 고르는 문제가 생기면 어떻게 해야 할까?

샘플 데이터에 시그모이드 함수를 적용하게 된다면

첫번째가 정답일 확률은 0.7, 두번째가 정답일 확률은 0.6, 세번째가 정답일 확률은 0.4 등과 같은 출력을 얻었다.

그런데 이 전체 확률의 합계가 1이 되도록 하여 전체 정답지에 걸친 확률로 바꿀 순 없을까?

예를 들어 0이 정상 메일, 1이 스팸 메일이라고 정의했을 때 시그모이드 함수의 0과 1사이의 출력 값을 스팸 메일일 확률로 해석할 수 있다.

확률값이 0.5를 넘으면 1에 더 가까우니까 스팸 메일로 판단하면 되고, 그 반대라면 정상 메일로 판단하면 된다!

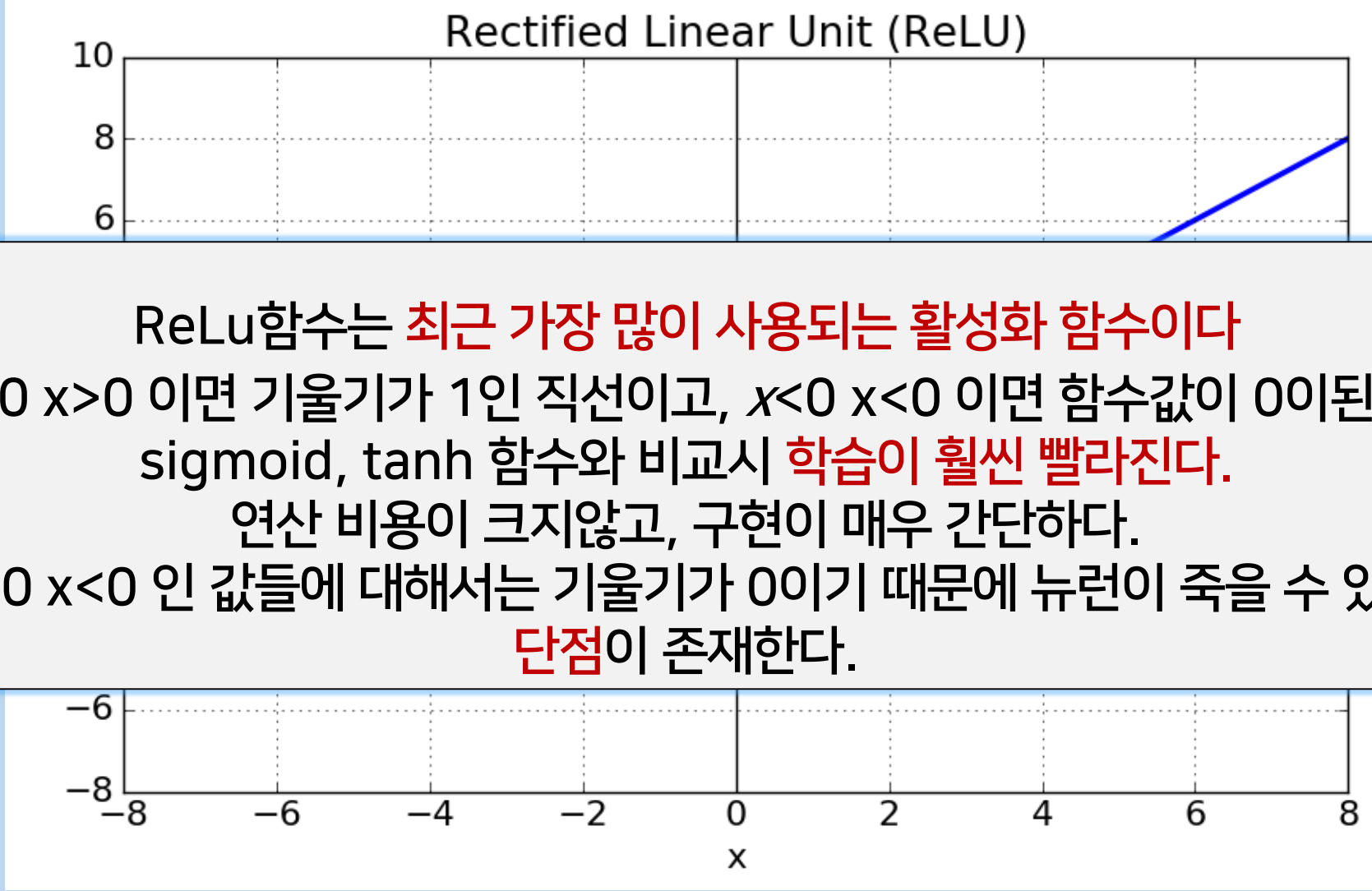
소프트 맥스 함수

소프트맥스 함수는 분류해야하는 정답지(클래스)의 총 개수를 k라고 할 때, k차원의 벡터를 입력받아 각 클래스에 대한 확률을 추정다

쉽게 말해서, 0~1사이의 값으로 모두 정규화하며

출력 값들의 총합은 항상 1이 되는 특성을 가진 함수이다.

Relu 함수



ReLu함수는 최근 가장 많이 사용되는 활성화 함수이다

$x>0$ x>0 이면 기울기가 1인 직선이고, $x<0$ x<0 이면 함수값이 0이된다.

sigmoid, tanh 함수와 비교시 학습이 훨씬 빨라진다.

연산 비용이 크지않고, 구현이 매우 간단하다.

$x<0$ x<0 인 값들에 대해서는 기울기가 0이기 때문에 뉴런이 죽을 수 있는 단점이 존재한다.

Mnist

이제 이론은 됐다! MNIST 문자 인식 프로그램을 만들어보자!

:0~9중에 하나를 쓴 손글씨 이미지를 보고, 어떤 숫자인지 알아맞추는 프로그램

(제공 받은 트레이닝데이터로 부터 원-핫 벡터를 만드는것)

: 데이터를 어떻게 인식하는지 궁금한데 창에 뜨게 해보자!

Mnist

```python
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import cv2
import sys



(x_train ,y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

x_train = x_train.astype('float32')/255.0
x_test = x_test.astype('float32')/255.0


x_train = x_train.reshape(60000, 28*28)
x_test = x_test.reshape(10000, 28*28)
```

필요한 모듈 import 하기

Mnist

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import cv2
import sys



(x_train ,y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

x_train = x_train.astype('float32')/255.0
x_test = x_test.astype('float32')/255.0


x_train = x_train.reshape(60000, 28*28)
x_test = x_test.reshape(10000, 28*28)
```

필요한 모듈 import 하기

mnist 자료 가져요기

Mnist

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import cv2
import sys



(x_train ,y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

x_train = x_train.astype('float32')/255.0
x_test = x_test.astype('float32')/255.0


x_train = x_train.reshape(60000, 28*28)
x_test = x_test.reshape(10000, 28*28)
```

필요한 모듈 import 하기

mnist 자료 가져요기

그 데이터들을
one hot 인코딩으로 바꾸어준다!

Mnist

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import cv2
import sys
```

필요한 모듈 import 하기

```
(x_train ,y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

mnist 자료 가져요기

```
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

그 데이터들을
one hot 인코딩으로 바꾸어준다!

```
x_train = x_train.astype('float32')/255.0
x_test = x_test.astype('float32')/255.0
```

Numpy에서 Float형으로 바꿔주기

```
x_train = x_train.reshape(60000, 28*28)
x_test = x_test.reshape(10000, 28*28)
```

Mnist

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import cv2
import sys
```

필요한 모듈 import 하기

```
(x_train ,y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

mnist 자료 가져요기

```
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

그 데이터들을
one hot 인코딩으로 바꾸어준다!

```
x_train = x_train.astype('float32')/255.0
x_test = x_test.astype('float32')/255.0
```

Numpy에서 Float형으로 바꿔주기

```
x_train = x_train.reshape(60000, 28*28)
x_test = x_test.reshape(10000, 28*28)
```

뉴런에 넣으려면 한줄로 되야 하는데 mnist 2차원 배열로 이루어짐
그래서 바꿔준다

Mnist

첫번째 레이어에는 무조건 input shape를 진행해줘야한다.
(x데이터의 shape을 넣어준다.)

층을 쌓아준다

```
#뉴런 만들기
model = keras.Sequential()
model.add(keras.layers.Dense(256, activation="relu", input_shape=(784,)))
model.add(keras.layers.Dense(128, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
optimizer_SGD= keras.optimizers.SGD(lr=0.1)
model.compile(optimizer=optimizer_SGD, loss="categorical_crossentropy", metrics=['accuracy'])
model.summary()

#모델 학습

model.fit(x_train, y_train,
          batch_size=256,
          epochs=20,
          validation_data=(x_test, y_test))


print("--  result --")
result = model.evaluate(x_test, y_test)
print("%s: %.2f%%" %(model.metrics_names[1], result[1]*100))
```

Mnist

첫번째 레이어에는 무조건 input shape를 진행해줘야한다.
(x데이터의 shape을 넣어준다.)

```
#뉴런 만들기                          층을 쌓아준다
model = keras.Sequential()
model.add(keras.layers.Dense(256, activation="relu", input_shape=(784,)))
model.add(keras.layers.Dense(128, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))  Out put은 softmax를 이용해서!
optimizer_SGD= keras.optimizers.SGD(lr=0.1)
model.compile(optimizer=optimizer_SGD, loss="categorical_crossentropy", metrics=['accuracy'])
model.summary()

#모델 학습

model.fit(x_train, y_train,
          batch_size=256,
          epochs=20,
          validation_data=(x_test, y_test))


print("--  result --")
result = model.evaluate(x_test, y_test)
print("%s: %.2f%%" %(model.metrics_names[1], result[1]*100))
```

Mnist

```
#뉴런 만들기
model = keras.Sequential()
model.add(keras.layers.Dense(256, activation="relu", input_shape=(784,)))
model.add(keras.layers.Dense(128, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
optimizer_SGD= keras.optimizers.SGD(lr=0.1)
model.compile(optimizer=optimizer_SGD, loss="categorical_crossentropy", metrics=['accuracy'])
model.summary()

#모델 학습

model.fit(x_train, y_train,
          batch_size=256,
          epochs=20,
          validation_data=(x_test, y_test))


print("--  result --")
result = model.evaluate(x_test, y_test)
print("%s: %.2f%%" %(model.metrics_names[1], result[1]*100))
```

경사하강법 사용 => SGD

Loss => 비용함수 설정!
바이너리 파일이 아니라 클래스 파일이므로 저것을 사용
바이너리 파일? 0또는 1일경우 (binary_crossenropy사용)

Mnist

```
#뉴런 만들기
model = keras.Sequential()
model.add(keras.layers.Dense(256, activation="relu", input_shape=(784,)))
model.add(keras.layers.Dense(128, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
optimizer_SGD= keras.optimizers.SGD(lr=0.1)
model.compile(optimizer=optimizer_SGD, loss="categorical_crossentropy", metrics=['accuracy'])
model.summary()

#모델 학습

model.fit(x_train, y_train,
          batch_size=256,
          epochs=20,
          validation_data=(x_test, y_test))


print("--  result --")
result = model.evaluate(x_test, y_test)
print("%s: %.2f%%" %(model.metrics_names[1], result[1]*100))
```

경사하강법 사용 => SGD

샘플이 되게 많은데 너무 많으면 렉걸리니까 256개만 사용해보자!

결과 값 출력하기

Mnist

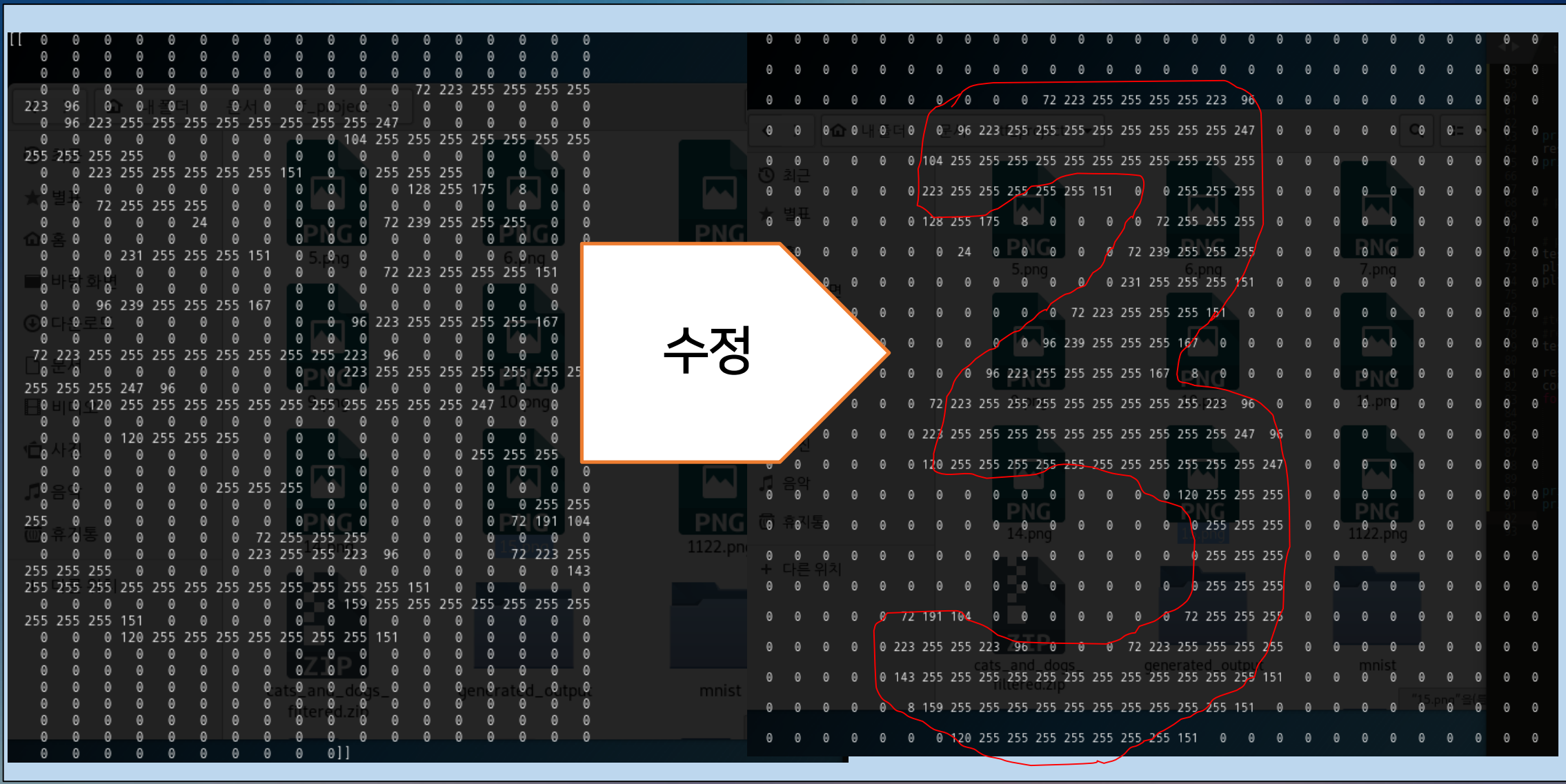
```
# pick up picture
test = cv2.imread(sys.argv[1], cv2.IMREAD_GRAYSCALE)
plt.imshow(test)
plt.show()

test = 255-test # 반전해준다
res = np.reshape(test, (1, np.product(test.shape)))
count = 1
for i in res:
    for j in i:
        print("%4s" % j, end="") #
        if count%28 == 0:
            print()
            print()
        count += 1


print("답 뭐게: %s" % (model.predict_classes(res)))
```

이미지를 Grayscale로 읽어들인다.

사진 가져오기!

이거 안해주면 오류나서
만들었다. 인식을 이상하게한다.

Mnist



Mnist

```
# pick up picture
test = cv2.imread(sys.argv[1], cv2.IMREAD_GRAYSCALE)
plt.imshow(test)
plt.show()

test = 255-test # 반전해준다
res = np.reshape(test, (1, np.product(test.shape)))
count = 1
for i in res:
    for j in i:
        print("%4s" % j, end="") #
        if count%28 == 0:
            print()
            print()
        count += 1


print("답 뭐게: %s" % (model.predict_classes(res)))
```

이미지를 Grayscale로 읽어들인다.

사진 가져오기!

이거 안해주면 오류나서
만들었다. 인식을 이상하게한다.

Thank you