



# Deep learning: Python coding 해보기

---

Sang Yup Lee



# Python coding 해보기

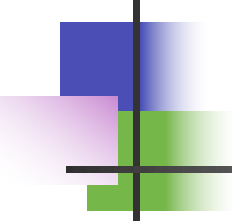
---

- FNN을 이용한 데이터 분석
  - 데이터 분석 예제
    - 회귀문제: Boston housing price 예측
      - Data 설명: <http://lib.stat.cmu.edu/datasets/boston>
    - 분류문제
      - 텍스트: 네이버 영화평 감성분석
      - 이미지
        - Handwritten numbers MNIST
  - 사용 프레임워크: Keras
    - You can install it using tensorflow
      - <https://www.tensorflow.org/install>
      - pip install tensorflow



---

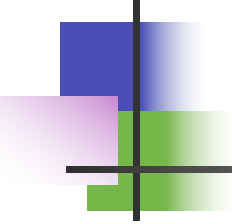
# FNN을 이용한 회귀문제 분석



# FNN을 이용한 회귀문제 분석

---

- Boston housing price 예측
  - Dataset
    - 설명: <https://www.kaggle.com/c/boston-housing>
    - 종속변수: 도시 집값의 median
    - 독립변수: 도시의 특성 관련 13개 변수
      - 예) 범죄율, 연령 등
  - 사용 방법
    - 선형회귀모형
      - from sklearn
        - LinearRegression, Lasso, Ridge
    - 신경망
      - FNN



# FNN을 이용한 회귀문제 분석

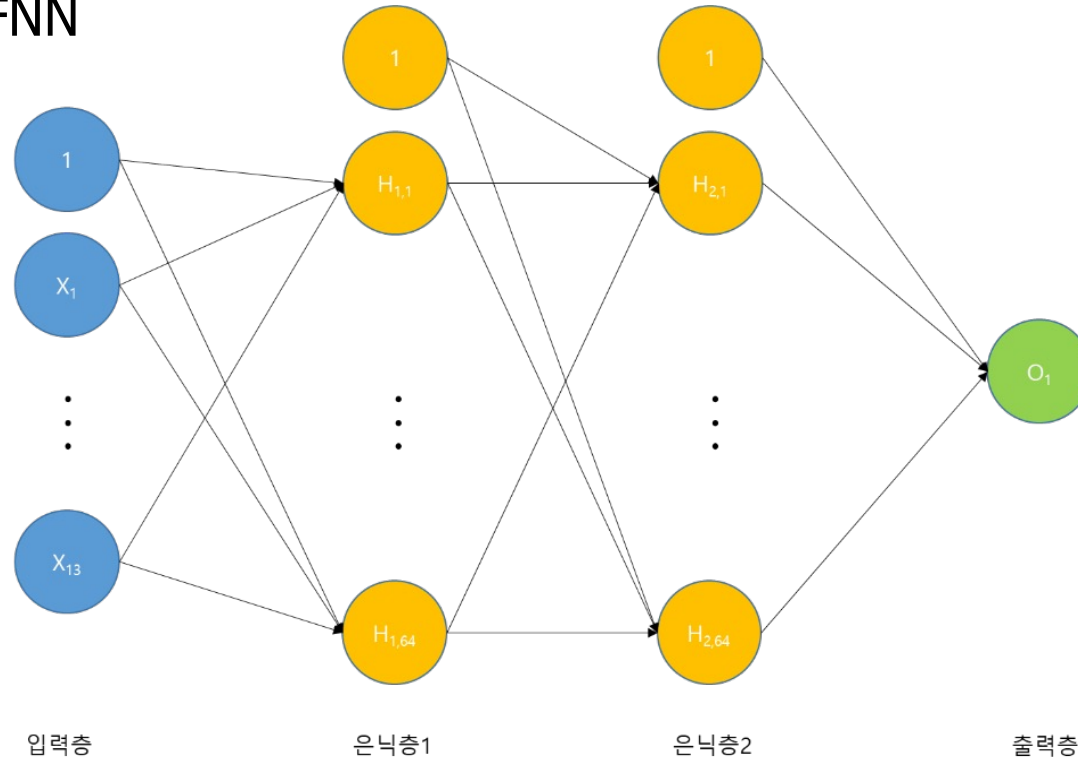
---

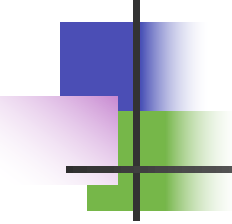
- Boston housing price 예측 (cont'd)
  - FNN
    - The first thing to do?
      - You need to decide the model architecture!
      - For this case, let's assume that you want to use
        - Two hidden layers
        - # of nodes in the 1<sup>st</sup> hidden layer = 64
        - # of nodes in the 2<sup>nd</sup> hidden layer = 64
    - **What about the # of nodes in the input and output layers?**

# FNN을 이용한 회귀문제 분석

- Boston housing price 예측 (cont'd)

- FNN





# FNN을 이용한 회귀문제 분석

---

- 층을 쌓는 법 using Keras
  - Sequential API
    - [https://keras.io/guides/sequential\\_model/](https://keras.io/guides/sequential_model/)
    - Sequential class의 add() 함수를 순차적으로 추가해 주는 방식
  - Functional API
    - [https://keras.io/guides/functional\\_api/](https://keras.io/guides/functional_api/)
    - <https://machinelearningmastery.com/keras-functional-api-deep-learning/>

# FNN을 이용한 회귀문제 분석

- Python code

- FNN\_housing\_example\_SGD.ipynb 참고

첫번째 은닉층에서는 관측치의  
입력 형태를 지정해줘야 함

```
model = models.Sequential()  
# 입력층을 별도로 추가하지 않는다!  
model.add(layers.Dense(64, activation = 'relu', input_shape=(train_data.shape[1],)))  
model.add(layers.Dense(64, activation = 'relu'))  
model.add(layers.Dense(1))
```

우리가 사용하는 은닉층과 출력층을 Keras에서는 Dense layer라고 표현



# FNN을 이용한 회귀문제 분석

## ■ model.summary()

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	896
dense_1 (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 1)	65

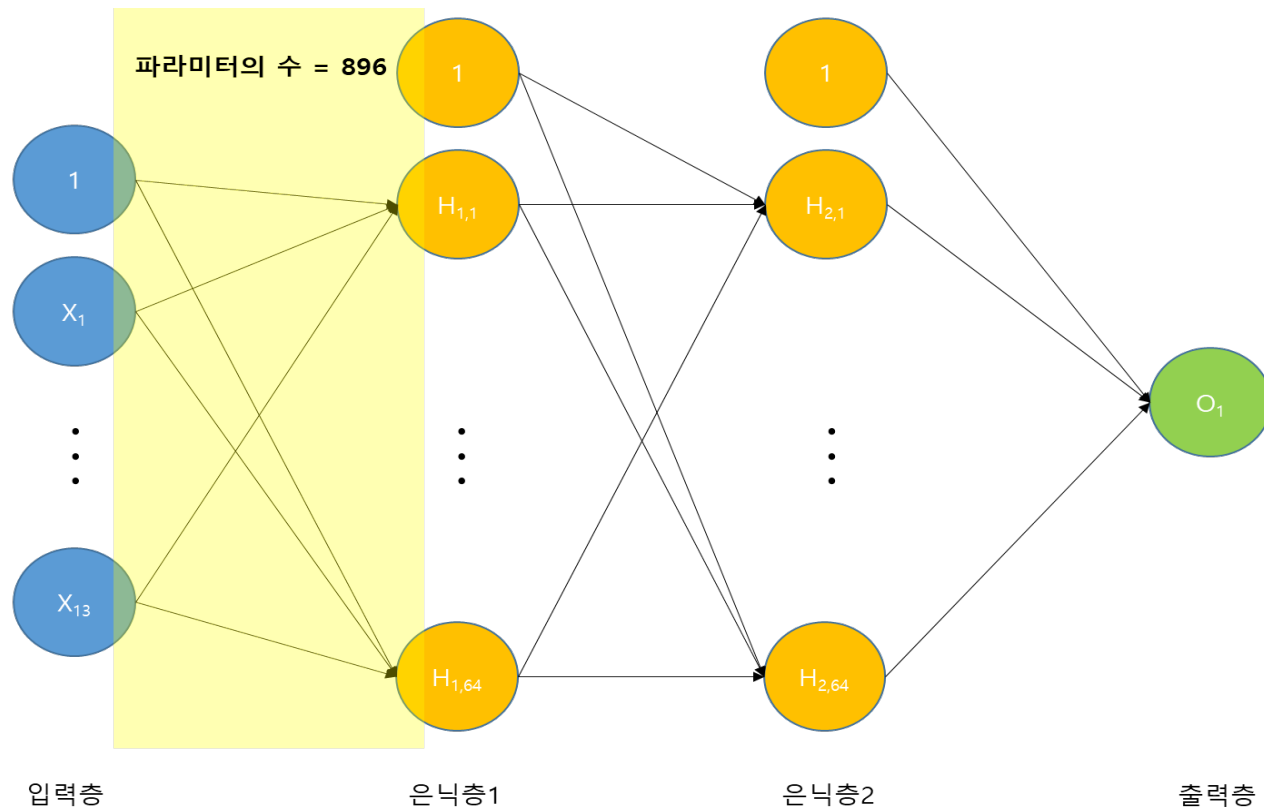
Total params: 5,121

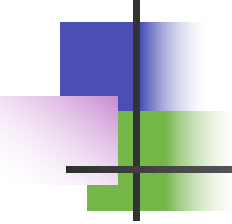
Trainable params: 5,121

Non-trainable params: 0

# FNN을 이용한 회귀문제 분석

- 파라미터수 (예, 896)





# FNN을 이용한 회귀문제 분석

---

- Python coding

- SGD 사용해 보기

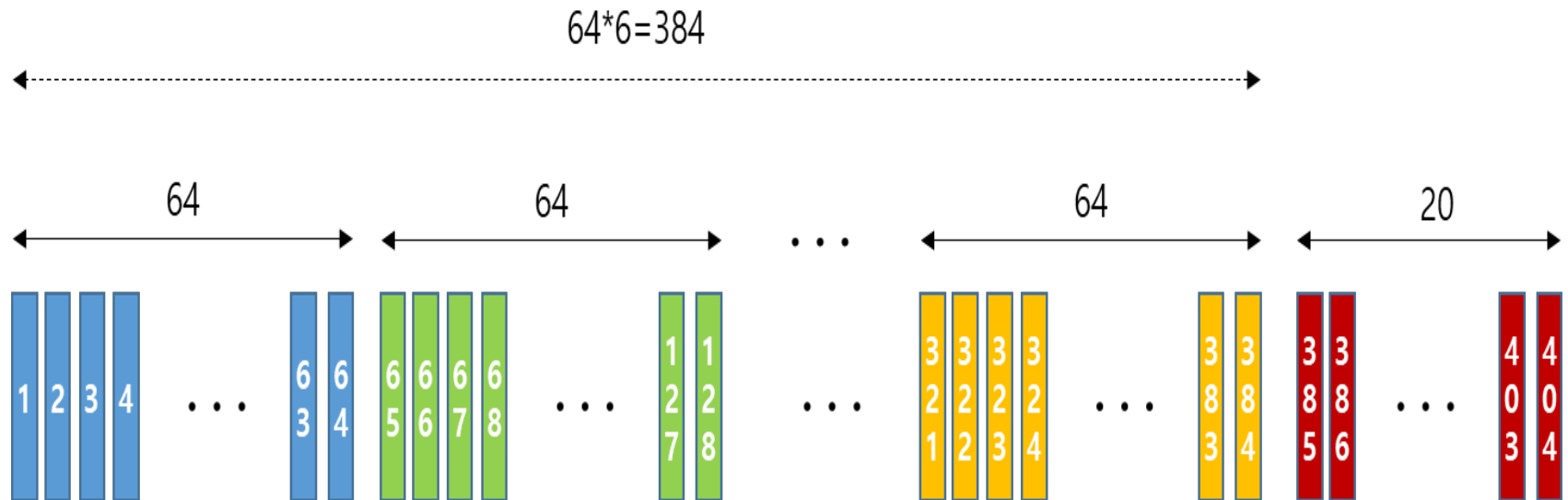
- <https://keras.io/api/optimizers/sgd/>

```
sgd= tf.keras.optimizers.SGD(learning_rate=0.001, momentum=0.9,  
nesterov=True)  
model.compile(optimizer='sgd', loss='mse')  
model.fit(train_data, train_targets, epochs=80, batch_size=64)
```

학습 데이터에 존재하는 관측치의 수 = 404  
그렇다면 하나의 에포크당 업데이트 횟수는?

# FNN을 이용한 회귀문제 분석

## ■ 에포크 당 업데이트 방식



# FNN을 이용한 회귀문제 분석

- Python coding
  - 다른 optimizer 사용해 보기
    - RMSprop
      - FNN\_housing\_example\_RMSProp.ipynb

$$w_{i,t+1} = w_{i,t} - \frac{\eta}{\sqrt{E[g^2]_{i,t} + \epsilon}} \frac{\partial E}{\partial w_i}$$

$$E[g^2]_{i,t} = \rho E[g^2]_{i,t-1} + (1 - \rho) g_{i,t}^2$$

- Adam
  - FNN\_housing\_example\_Adam.ipynb

$$m_{i,t} = \beta_1 \cdot m_{i,t-1} + (1 - \beta_1) \cdot g_{i,t}$$

$$v_{i,t} = \beta_2 \cdot v_{i,t-1} + (1 - \beta_2) \cdot g_{1,t}^2$$

$$w_{i,t+1} = w_{i,t} - \eta \frac{m_{i,t}}{\sqrt{v_{i,t} + \epsilon}}$$



---

# FNN을 이용한 이미지 분류



# 비정형 데이터

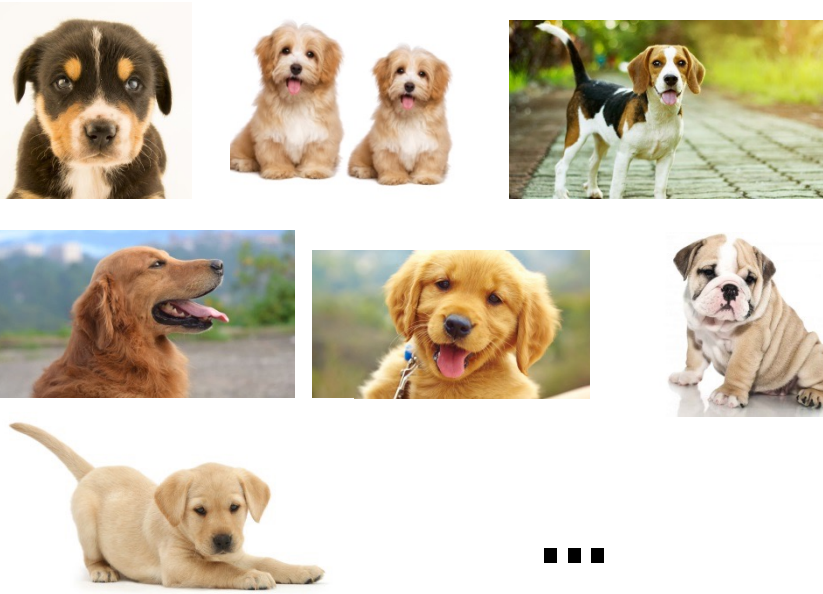
---

- 비정형 데이터
  - 예) 이미지, 텍스트, 오디오, 비디오 등
  - 딥러닝 모형은 비정형 데이터를 분석하는데 적합
  - 사람이 독립변수를 따로 추출하지 않아도, 딥러닝 모형이 알아서 종속변수의 값을 예측하는데 중요한 정보를 추출함

# 비정형 데이터

## ■ Example

Dogs



Cats

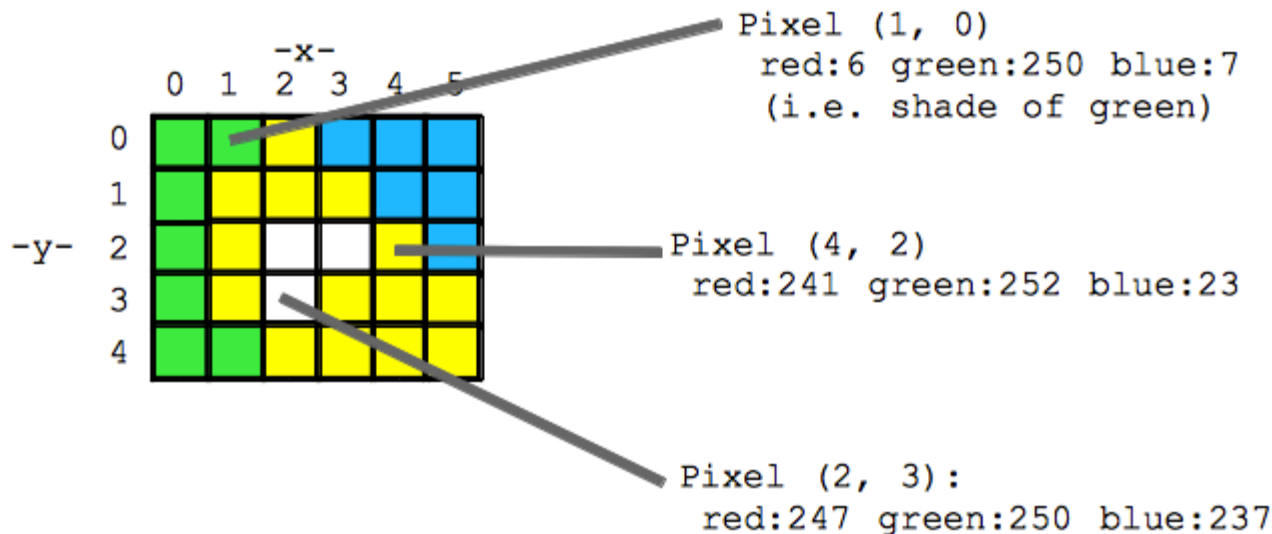




# FNN을 이용한 이미지 분류

## ■ 이미지 데이터

- 하나의 이미지 (예, 사진)은 여러개의 픽셀로 구성
  - $m \times n$  이미지의 경우  $m \times n$  개의 픽셀로 구성된 이미지가 됨
- 칼라 이미지의 경우, 하나의 픽셀이 보통 3개의 색 정보를 지님 (Red, Green, Blue, a.k.a., RGB; 이를 채널이라고 함)



즉, 각 픽셀이  
3개의 채널에 대한  
정보를 갖는다

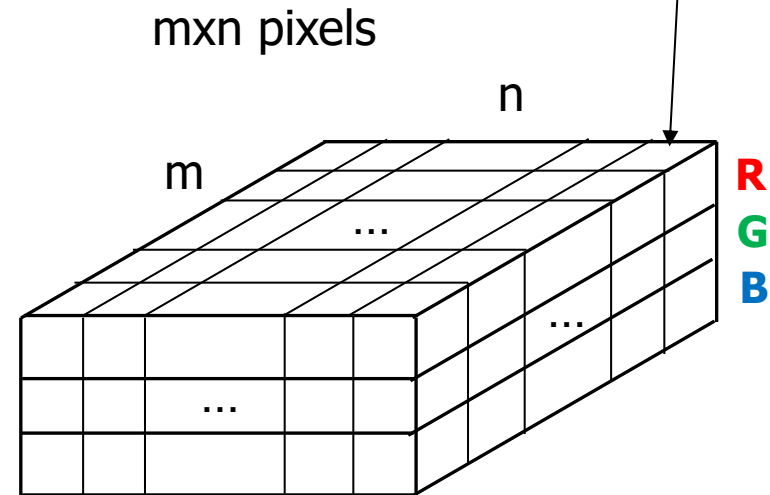
# FNN을 이용한 이미지 분류

- 이미지

- 각 채널 정보 (즉, 색상 정보는) 숫자로 표현 (0 ~ 255)
- 각 픽셀마다 3가지 채널정보를 저장하기 위해 3차원 3차원 형태의 행렬로 표현 (3D array 혹은 tensor 라고 함)

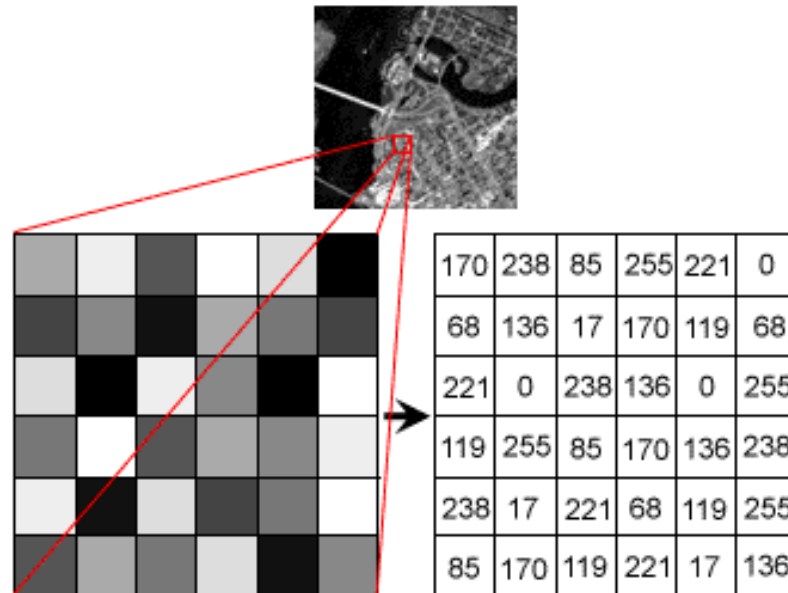
					165	187	209	58	7
					14	125	233	201	98
253	144	120	251	41	147	204			
67	100	32	241	23	165	30			
209	118	124	27	59	201	79			
210	236	105	169	19	218	156			
35	178	199	197	4	14	218			
115	104	34	111	19	196				
32	69	231	203	74					

각 셀이 하나의 색상 정보를 지님  
(각 셀이 하나의 독립변수로 간주)



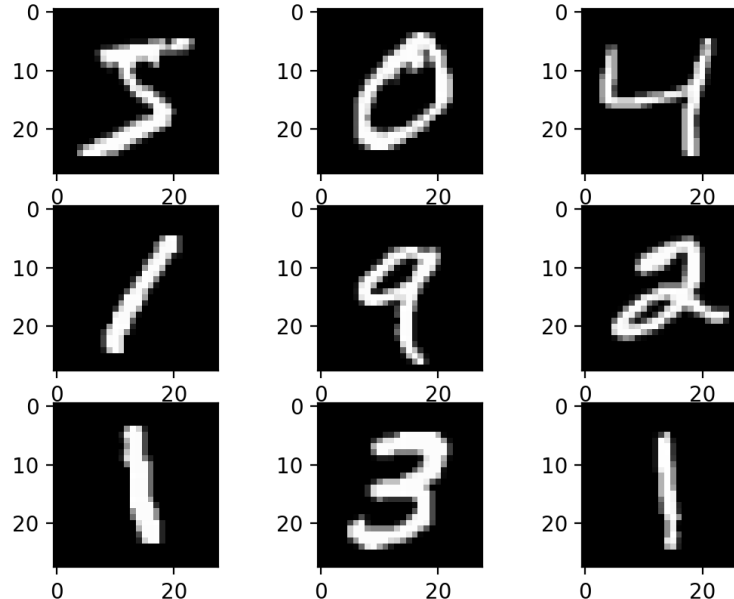
# FNN을 이용한 이미지 분류

- 이미지 데이터
  - 흑백 이미지의 경우는 channel 1개인 이미지라고 생각할 수 있음
  - 각 픽셀은 0 ~ 255 사이의 숫자를 지님 => 255에 가까울수록 흰색



# FNN을 이용한 이미지 분류

- MNIST dataset (Handwritten numbers)
  - Example images

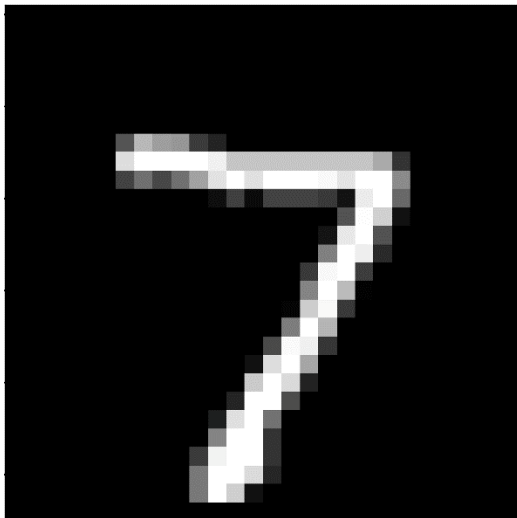


# FNN을 이용한 이미지 분류

## ■ 이미지

28

28



- 각 셀은 하나의 독립변수를 의미
- 각 셀은 0 ~ 255 값을 가짐
- 255에 가까울수록 흰색
- $28 \times 28 = 784$
- 이러한 정보를 입력층에 입력하기 위해서는  $784 \times 1$ 의 형태로 만들어줘야 함 => reshape() 함수를 이용
- 독립변수의 수가 784인 분류의 문제를 푼다고 생각

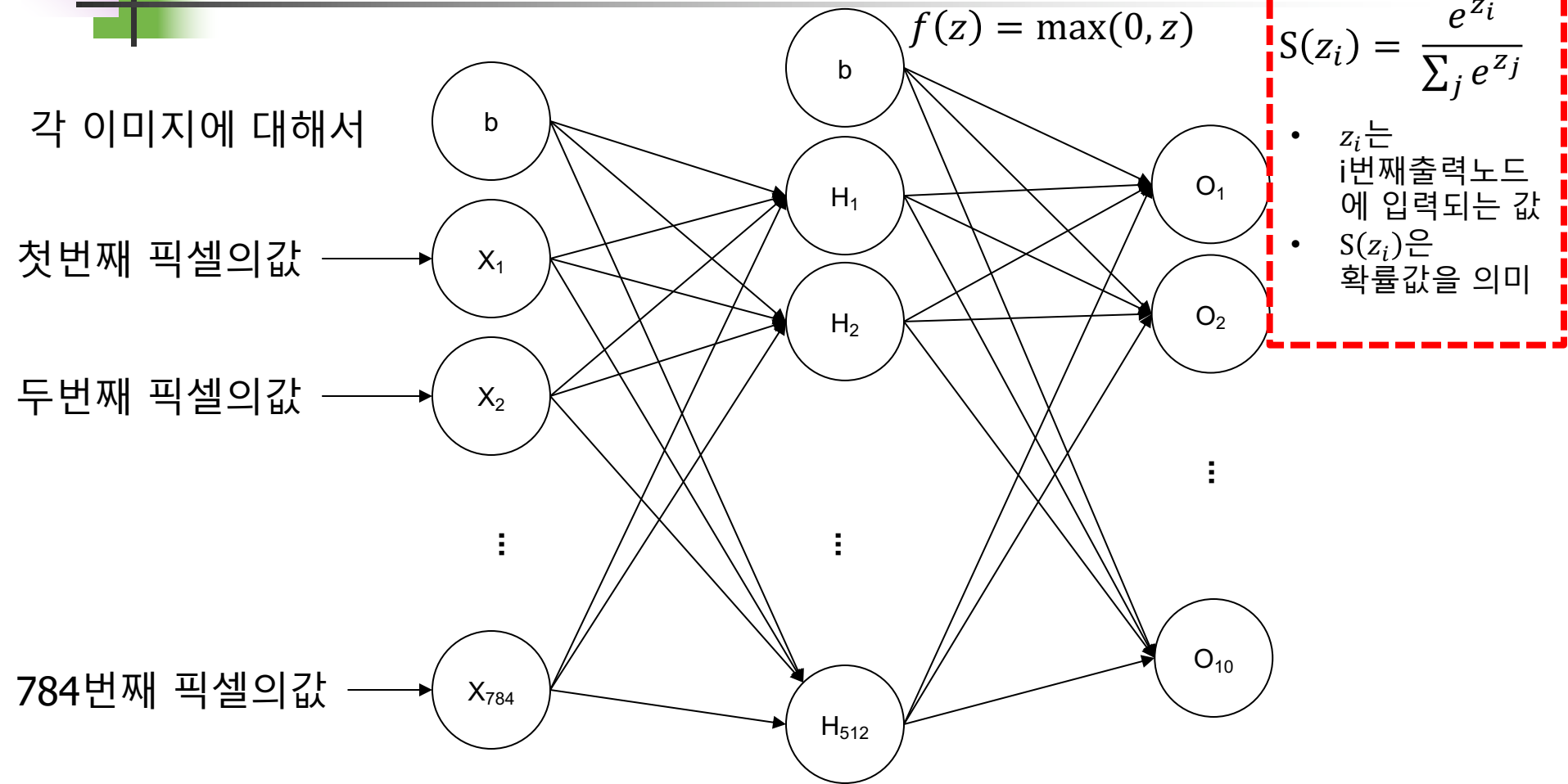


# FNN을 이용한 이미지 분류

---

- 신경망의 구조를 결정하자!!
  - # of input nodes
  - # of output nodes
  - What values are returned from the output nodes?
  - # of hidden layer
  - # of hidden nodes on each HL
  - Example
    - # of hidden layer = 1
    - # of hidden nodes on the HL = 512

# FNN을 이용한 이미지 분류



# FNN을 이용한 이미지 분류

- Reshaping

- $28*28 \rightarrow 784*1$  의 형태로 변환
- 예)  $3*3 \rightarrow 9*1$

1	1	0
4	2	1
0	2	1

Reshaping

1
1
0
4
2
1
0
2
1





# FNN을 이용한 이미지 분류

---

- Python code
  - “FNN\_MNIST\_example.ipynb”
  - What is the structure of the neural network?
  - # of output nodes?
  - What values are returned from the output nodes?



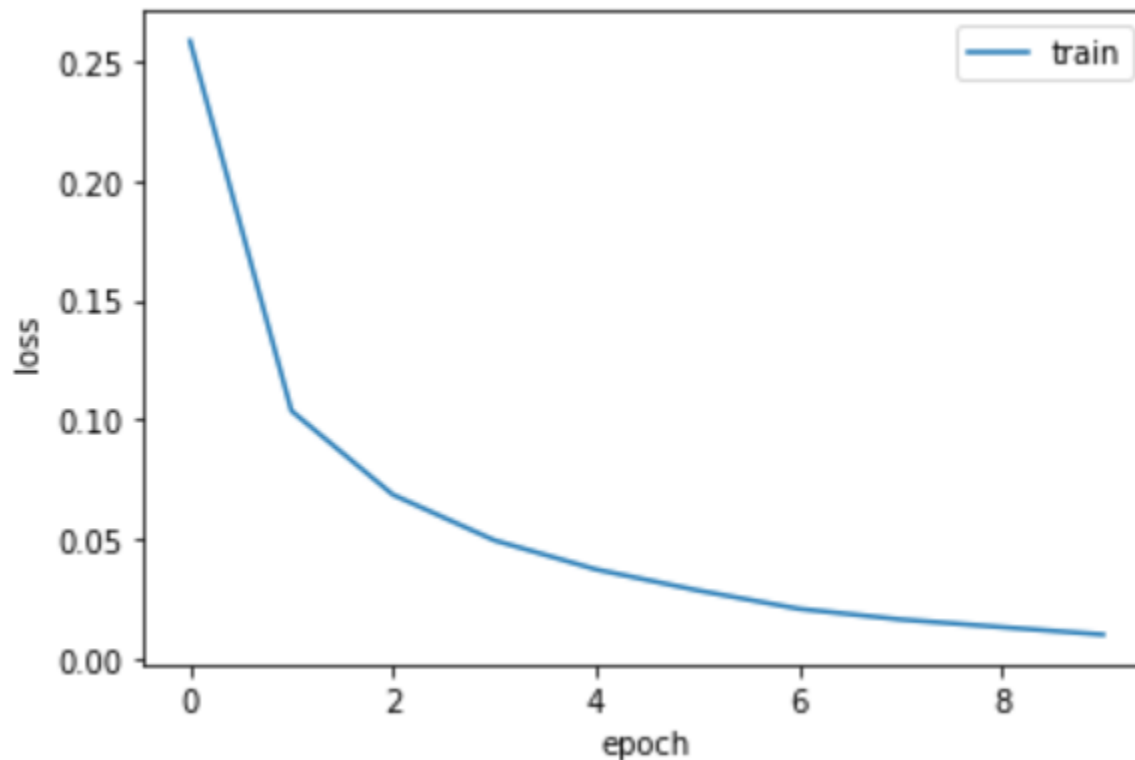
# FNN을 이용한 이미지 분류

---

- 학습 과정 check
  - 즉, 학습 과정에서 비용함수의 값과 accuracy가 어떻게 달라지는지 확인하기
  - Good to check out an overfitting problem
  - 방법1
    - `model.fit()` 함수의 결과를 이용
    - See "FNN\_MNIST\_example.ipynb"

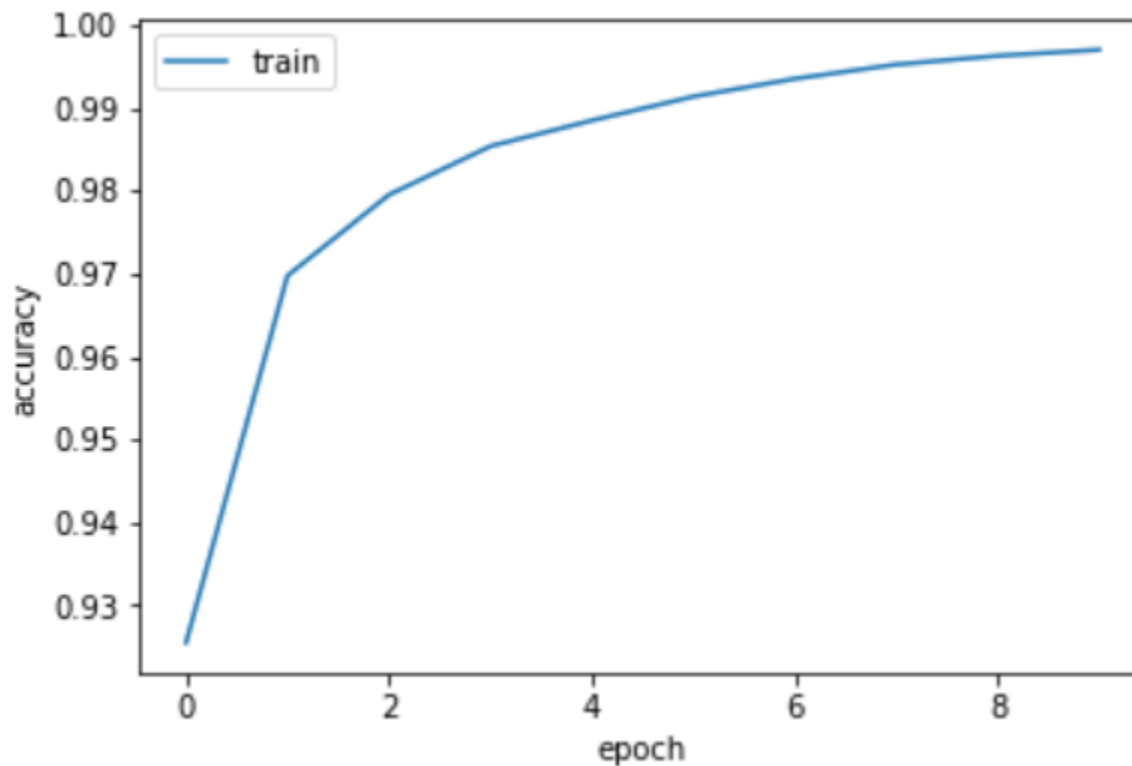
# FNN을 이용한 이미지 분류

## ■ 비용함수의 값



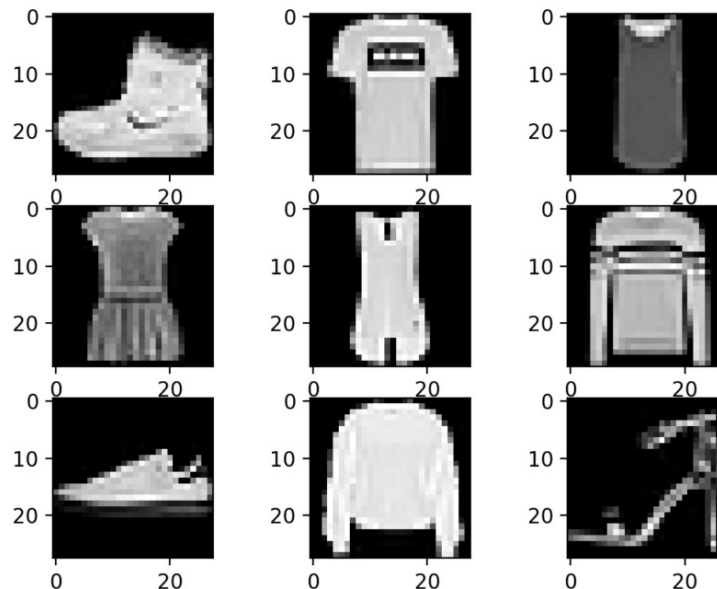
# FNN을 이용한 이미지 분류

## ■ Accuracy



# FNN을 이용한 이미지 분류

- Example 2
  - Fashion MNIST dataset



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot



# FNN을 이용한 이미지 분류

---

- Example 2 (cont'd)
  - Python code
    - See “FNN\_fashion\_MNIST\_example.ipynb”



---

# **FNN을 이용한 SENTIMENT ANALYSIS**

# FNN을 이용한 Sentiment analysis

- FNN을 이용한 텍스트 분석의 예
  - 감성분석 등의 텍스트 분석에는 일반적으로 FNN 보다는 CNN이나 RNN 등의 다른 딥러닝 알고리즘이 더 많이 사용
  - 전처리가 끝난 데이터가 있다고 가정하는 경우, 첫번째로 해야하는 것 => 신경망의 구조 결정
    - 1) 입력층과 출력층에 존재하는 노드의 수 결정
    - 2) 은닉층의수와 은닉노드의 수 결정



# FNN을 이용한 Sentiment analysis

- 출력노드의 수 결정
  - 출력노드의 수 = 종속변수가 취할 수 있는 값의 수 = 2
- 입력노드의 수 결정
  - vocabulary 에 있는 단어들 중에서 최종 분석에 사용하는 단어의 수로 결정
  - 만약, vocabulary에 있는 단어들이 10000개인데 그 중에서 빈도수를 기준으로 상위 1000개만 사용해서 감성분석을 수행한다고 하는 경우에는 입력 노드의 수가 1000개가 됨  
=> 우리가 해야하는 것은 각 문서를 선택된 1000개의 단어들을 이용해서 표현하고 그렇게 표현된 정보를 입력층에 전달하여 감성분석 수행
    - 즉, 각 문서에 사용된 단어들 중에서 상위 1000개에 해당하는 단어들만을 사용해서 각 문서를 다시 표현

# FNN을 이용한 Sentiment analysis

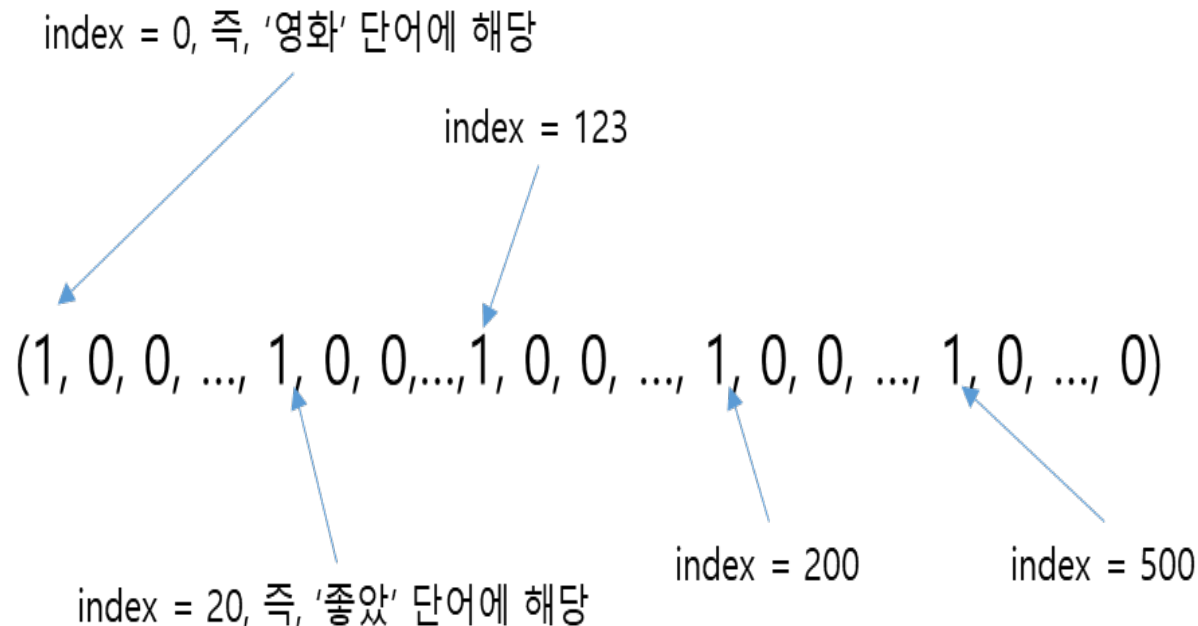
- 상위 K개의 단어들을 사용하여 각 문서 (즉, 영화평)을 표현
  - K=1000
- 예)
  - 영화평 = “어제 본 영화는 줄거리는 좋았지만, 주인공인 아무개의 연기는 엉망이었다”
  - 전처리후 아래와 같은 리스트로 표현했다고 가정
    - 문서1 = ['어제', '본', '영화', '줄거리', '좋았', '주인공', '아무개', '연기', '엉망']
    - 이중에서 상위 1000개 안에 포함되는 단어들이 '영화', '줄거리', '좋았', '주인공', '엉망' 인 경우, 해당 단어들만을 사용해서 해당 문서 다시 표현
      - 문서1 = ['영화', '줄거리', '좋았', '주인공', '엉망']
- 그 다음?
  - 각 문서를 숫자 정보로 변환
  - FNN에서는 일반적으로 one-hot encoding 방법을 사용

# FNN을 이용한 Sentiment analysis

- 각 문서를 숫자로 표현하기 (one-hot encoding)
  - 상위 1000개 단어들에 인덱스 번호를 부여
    - 즉, 0 ~ 999
  - 각 문서를 단어 정보가 아닌 각 단어들의 인덱스 번호를 가지고 표현
    - 만약 '영화'의 인덱스가 0, '줄거리'=123, '좋았' = 20, '주인공' = 500, '영망'=200의 인덱스를 갖는다고 하면
      - 문서1 = [0, 123, 20, 500, 200]
    - 이러한 인덱스 정보를 사용해서 one-hot encoding 수행
    - 각 문서는 원소의 수가 1000인 벡터로 변환
    - 해당 단어가 위치하는 자리의 원소값만을 1로하고 나머지 원소의 값은 0을 갖는 벡터로 표현

# FNN을 이용한 Sentiment analysis

## ■ 각 문서를 숫자로 표현하기





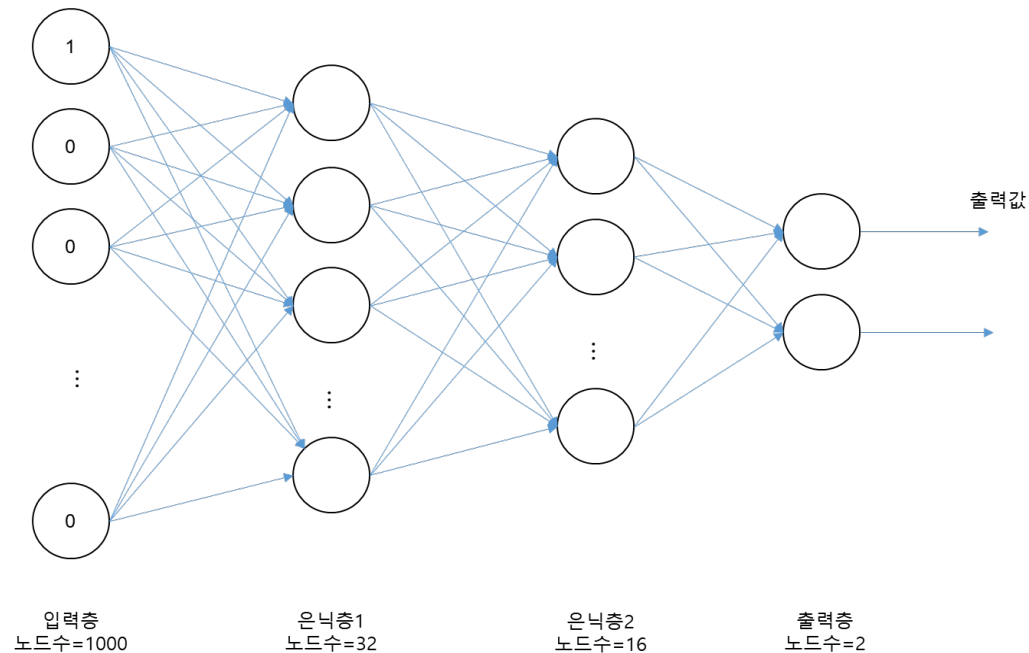
# FNN을 이용한 Sentiment analysis

---

- 신경망을 이용한 텍스트 분석에서의 문서 표현 (문서의 벡터화)
  - 전통적인 기계학습 (예, 로지스틱회귀모형)의 경우 Bag of Words 모형과 n-gram 사용
  - 하지만, 신경망에서는 사용하지 않는다!!
    - 필터 등을 사용해서 연속된 단어들 (혹은 문자들)의 순서를 파악해서 그 안에 숨겨진 내용을 추출할 수 있기 때문

# FNN을 이용한 Sentiment analysis

- 신경망의 구조 결정하기 (cont'd)
  - 은닉층의 수 & 은닉노드의 수



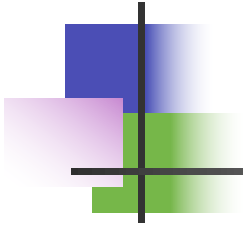
# FNN을 이용한 Sentiment analysis

- 출력노드에서 출력되는 값
  - 활성화함수: Softmax 함수
    - $S(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$ 
      - $z_i$ 는  $i$  번째 출력 노드에 입력되는 입력값
  - 출력노드의 수 = 2인 경우
    - 첫번째 노드에서 출력되는 값:  $S(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}}$ 
      - $P(y=0)$ 을 의미
    - 두번째 노드에서 출력되는 값:  $S(z_2) = \frac{e^{z_2}}{e^{z_1} + e^{z_2}}$ 
      - $P(y=1)$ 을 의미

# FNN을 이용한 Sentiment analysis

- 비용함수
  - 교차엔트로피
    - $-\sum_{i=1}^N \{y_i \log(P(y_i = 1)) + (1 - y_i) \log(P(y_i = 0))\}$
    - 즉,  $-\sum_{i=1}^N \{y_i \log S_i(z_2) + (1 - y_i) \log S_i(z_1)\}$
- 파이썬 코드
  - FNN\_text\_sentiment.ipynb 참고





# Q & A