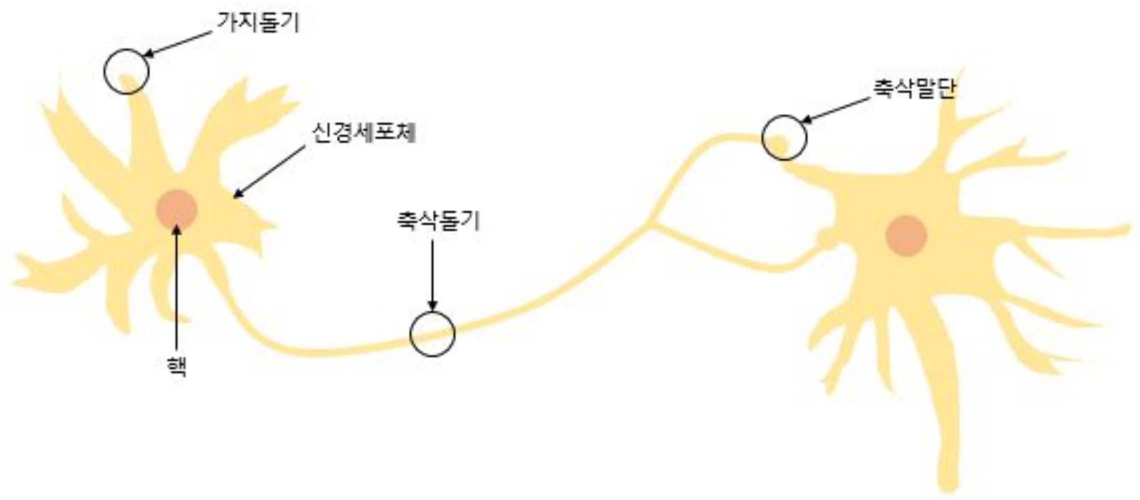
08.딥러닝 개요;
1-5

9조 이제일, 금예은

목차

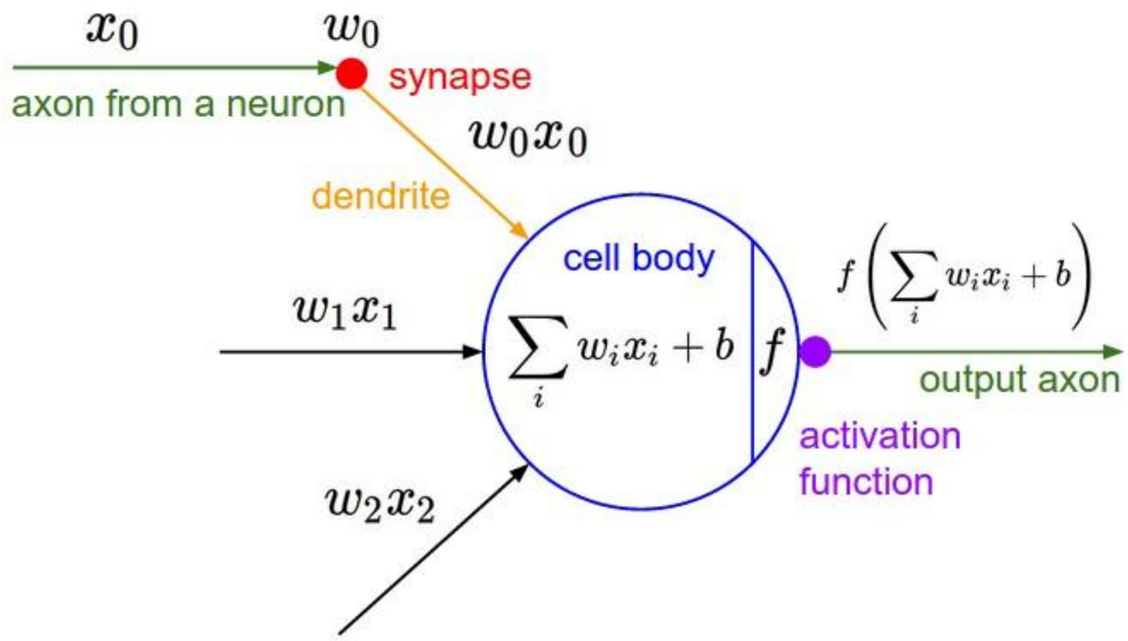
1. 퍼셉트론

1. 퍼셉트론 - 뉴런



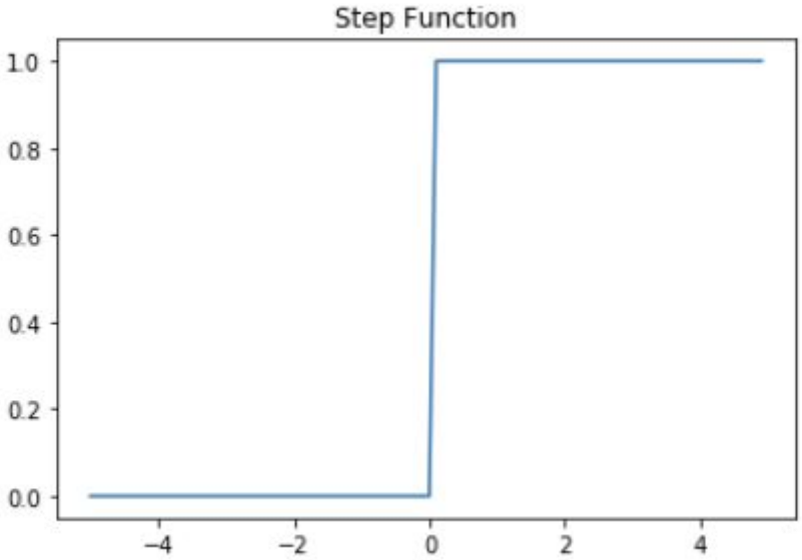
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

1. 퍼셉트론 - 인공신경



https://cs231n.github.io/neural-networks-1/

1. 퍼셉트론 - 활성화 함수



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

1. 퍼셉트론 - 임계치(편향)

$$if \sum_{i}^{n} W_i x_i \geq \boxed{\theta} \rightarrow y = 1$$

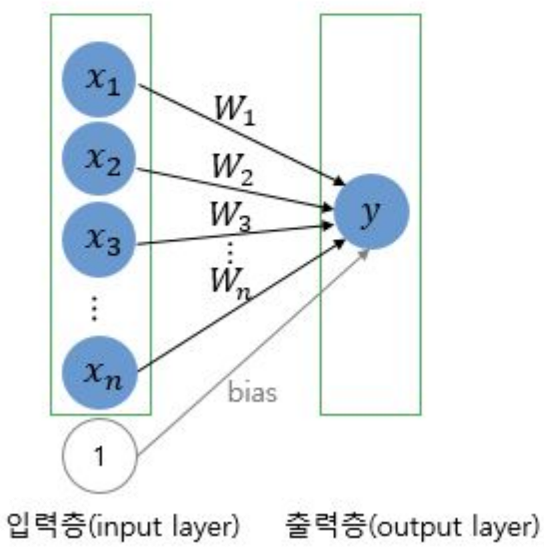
$$if \sum_{i}^{n} W_i x_i < \boxed{\theta} \rightarrow y = 0$$

$$if \sum_{i}^{n} W_i x_i \boxed{+ b} \geq 0 \rightarrow y = 1$$

$$if \sum_{i}^{n} W_i x_i + \boxed{b} < 0 \rightarrow y = 0$$

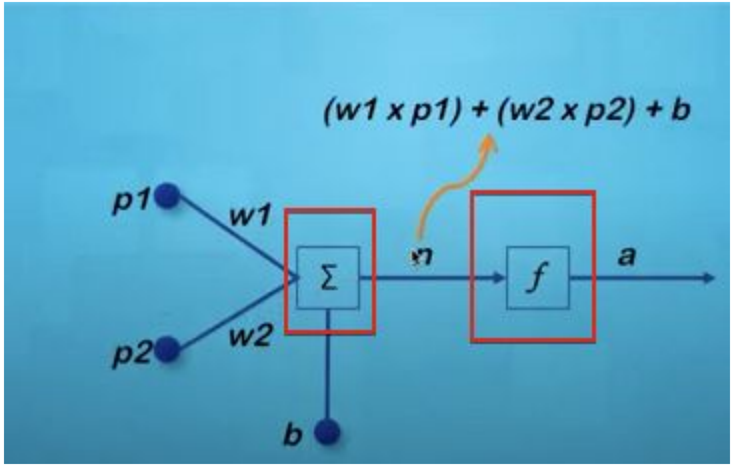
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 단층 퍼셉트론



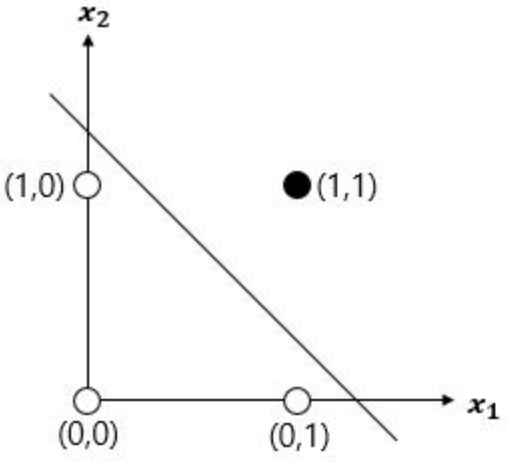
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 단층 퍼셉트론



허민석, 딥러닝이론, 멀티인풋뉴론, https://youtu.be/soZ9u-LsrX0

2. 단층 퍼셉트론 - AND 게이트

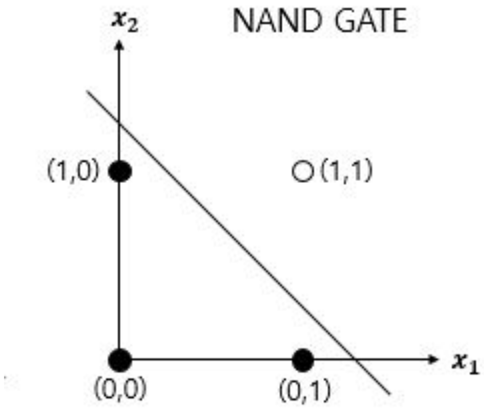


x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

```
def AND_gate(x1, x2):
    w1=0.5
    w2=0.5
    b=-0.7
    result = x1*w1 + x2*w2 + b
    if result <= 0:
        return 0
    else:
        return 1
```

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 단층 퍼셉트론 - NAND 게이트

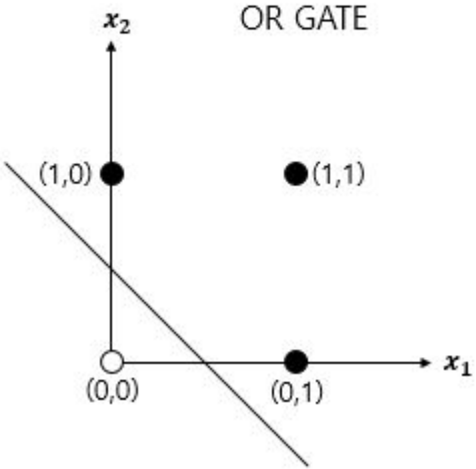


x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

```
def NAND_gate(x1, x2):
    w1=-0.5
    w2=-0.5
    b=0.7
    result = x1*w1 + x2*w2 + b
    if result <= 0:
        return 0
    else:
        return 1
```

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 단층 퍼셉트론 - OR 게이트

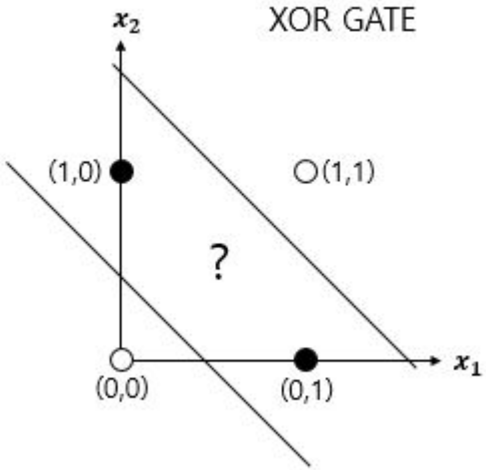


x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

```python
def OR_gate(x1, x2):
    w1=0.6
    w2=0.6
    b=-0.5
    result = x1*w1 + x2*w2 + b
    if result <= 0:
        return 0
    else:
        return 1
```

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

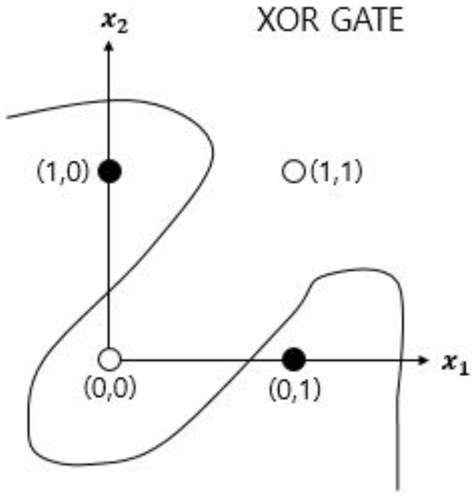
2. 단층 퍼셉트론 - XOR 게이트



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

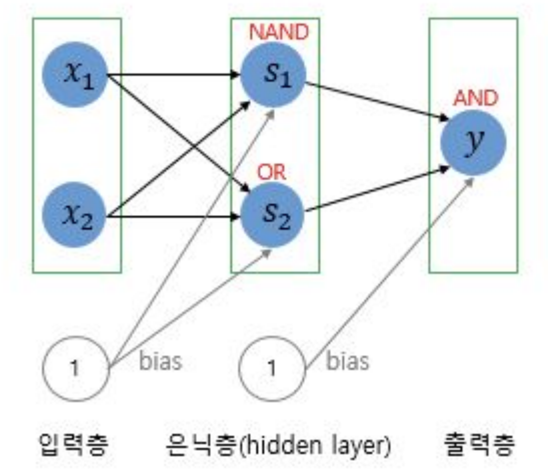
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 단층 퍼셉트론 - XOR 게이트



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

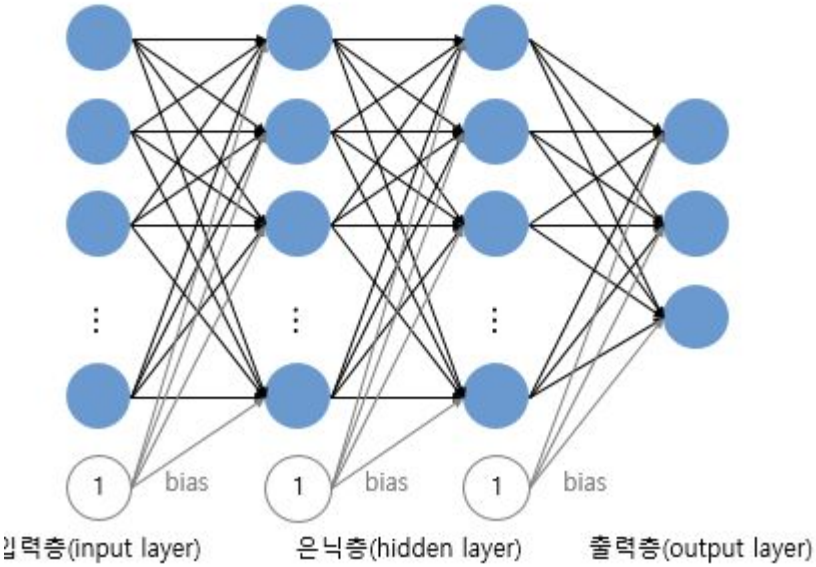
3. 다층 퍼셉트론




```
def XOR_gate(x1,x2):
    s1 =  NAND_gate(x1,x2)
    s2 = OR_gate(x1,x2)
    y = AND_gate(s1,s2)
    return y
```

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

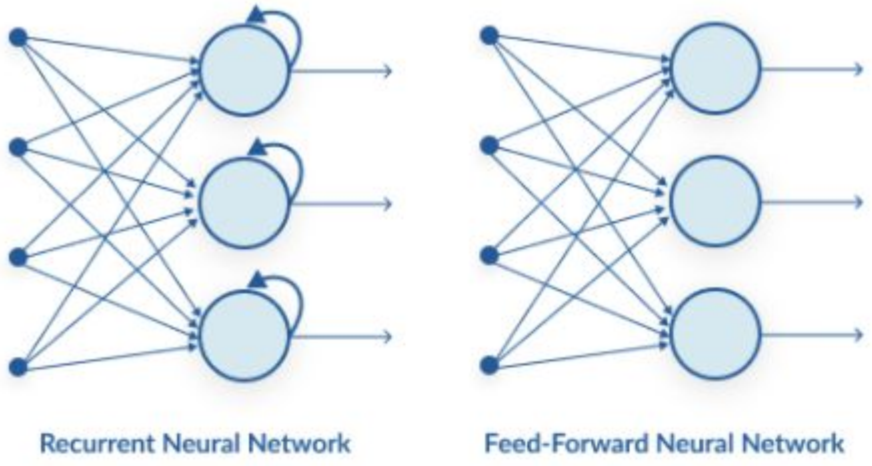
3. 다층 퍼셉트론



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

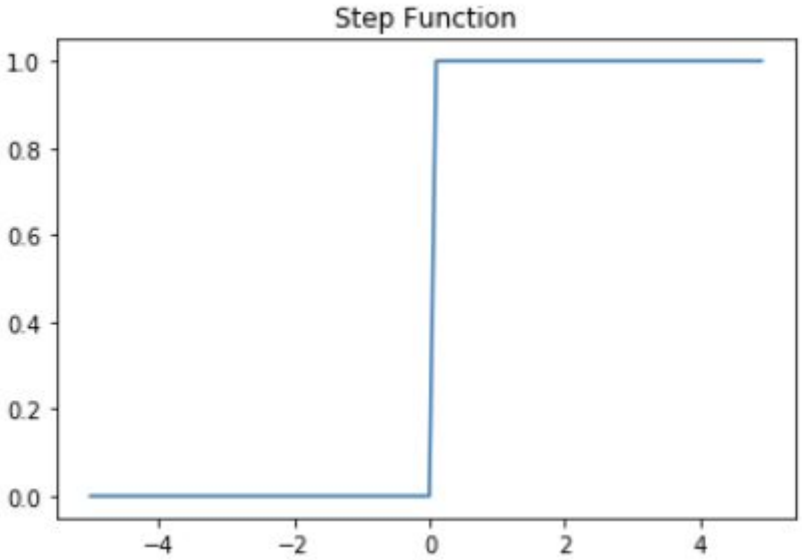
2. 인공신경망

1. 피드 포워드 신경망(순방향 신경망)



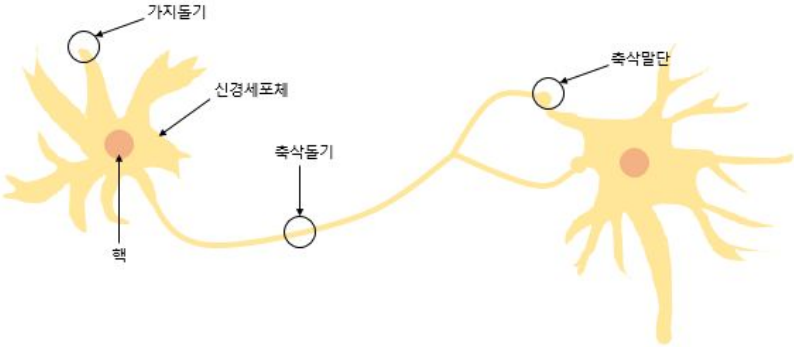
https://docs.paperspace.com/machine-learning/wiki/recurrent-neural-network-rnn

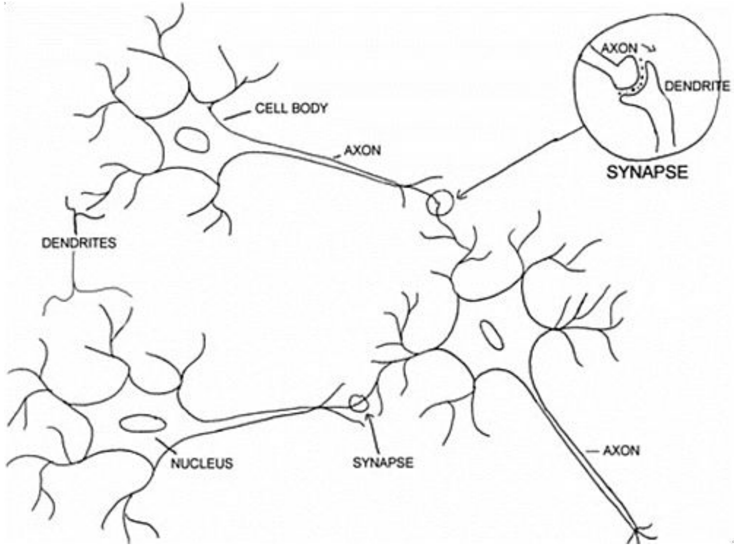
2. 활성화 함수



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

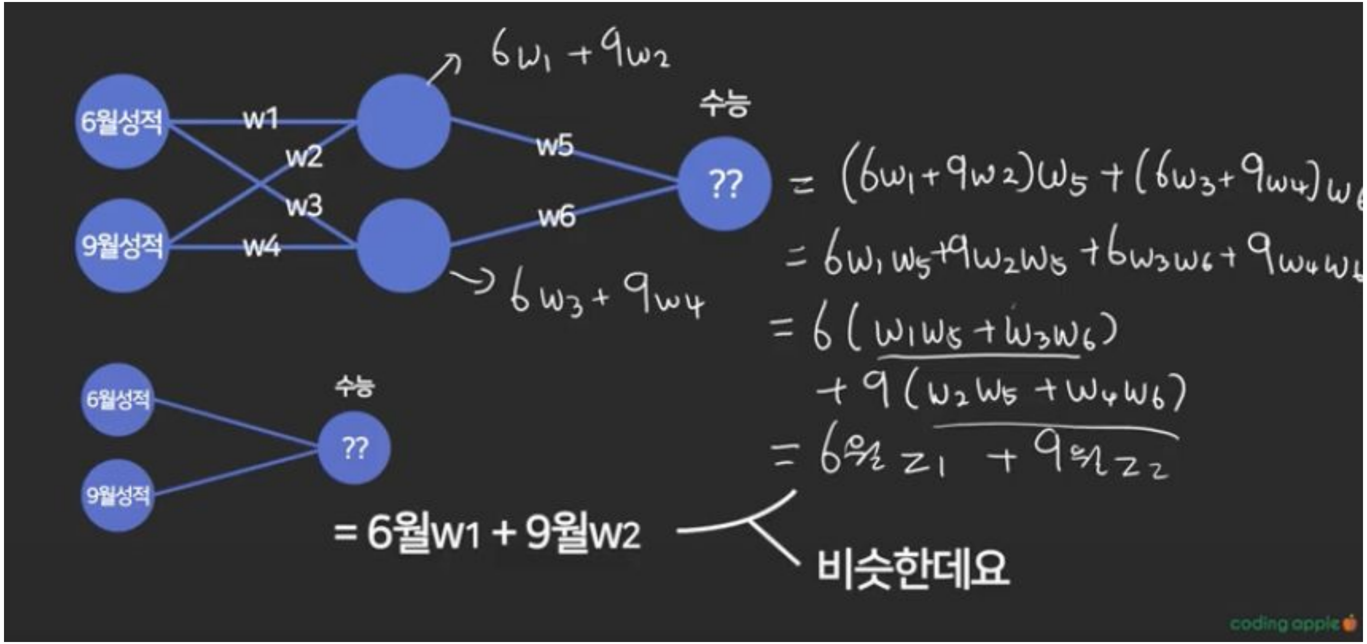
2. 활성화 함수



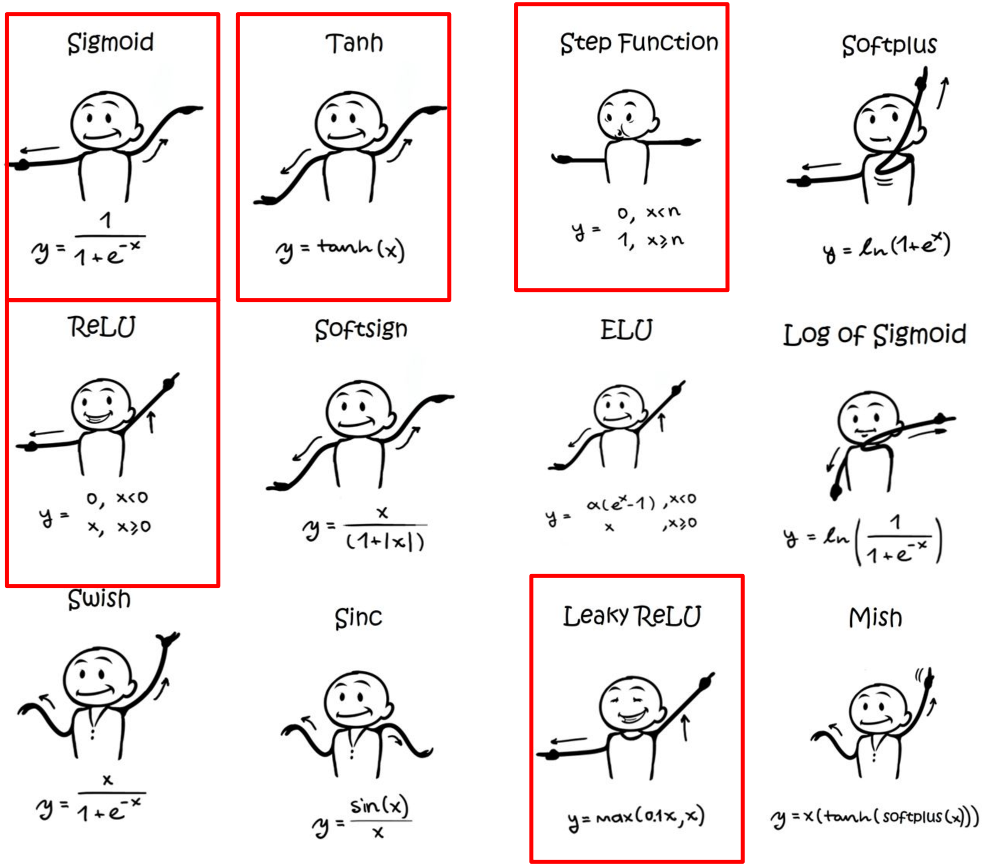


딥러닝을 이용한 자연어 처리 입문, 네이버 블로그 **dic1224** 느린서재 퍼셉트론, 허민석 딥러닝
https://youtu.be/ShaqWZx3Wzc

2. 활성화 함수 - 특징 : 비선형함수

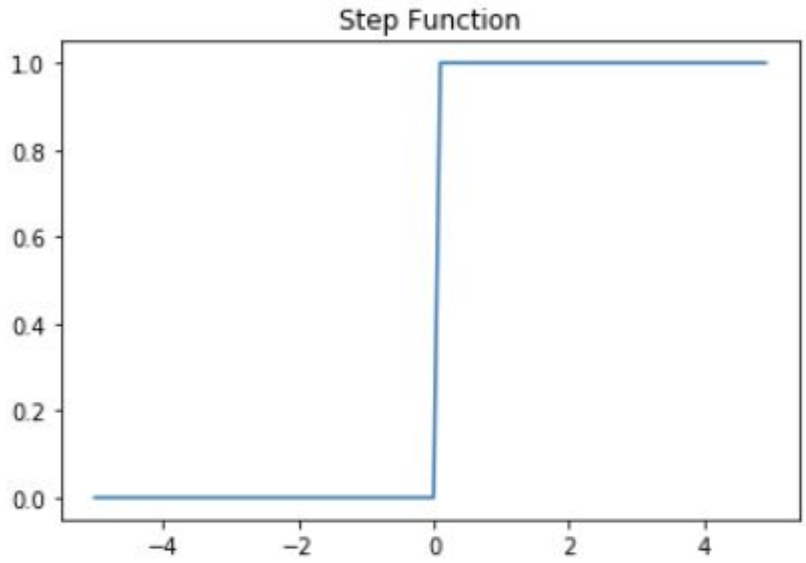


코딩애플, 활성화 함수, https://www.youtube.com/watch?v=rpHuwa-dbbI



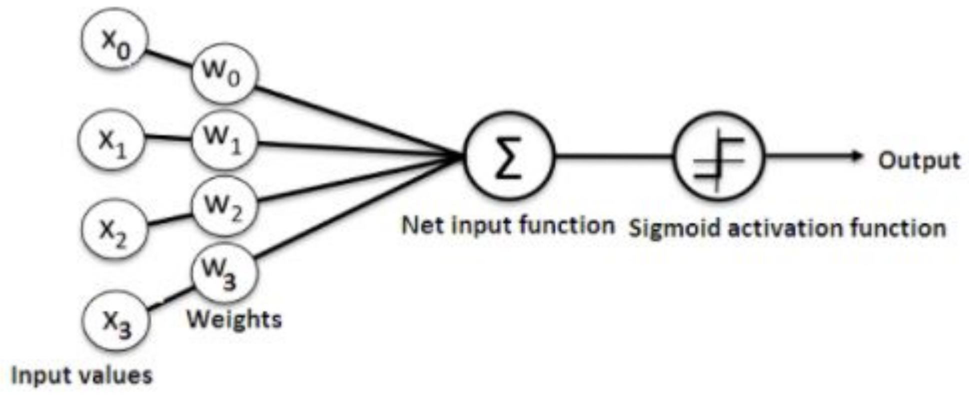
Dance moves of DL activation function - sefik

2. 활성화 함수 - 계단함수



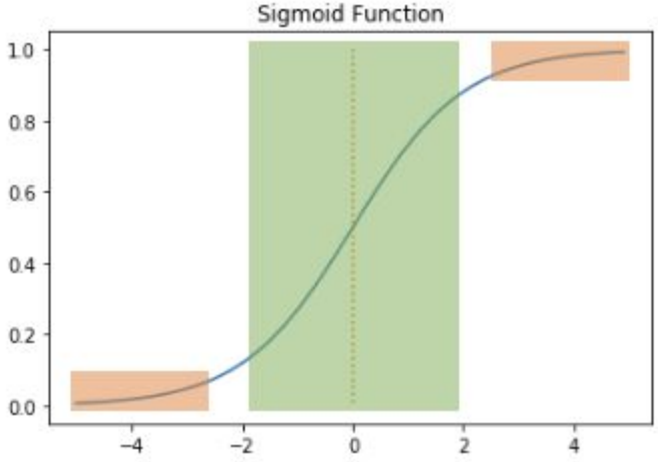
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 활성화 함수 - 시그모이드 함수



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

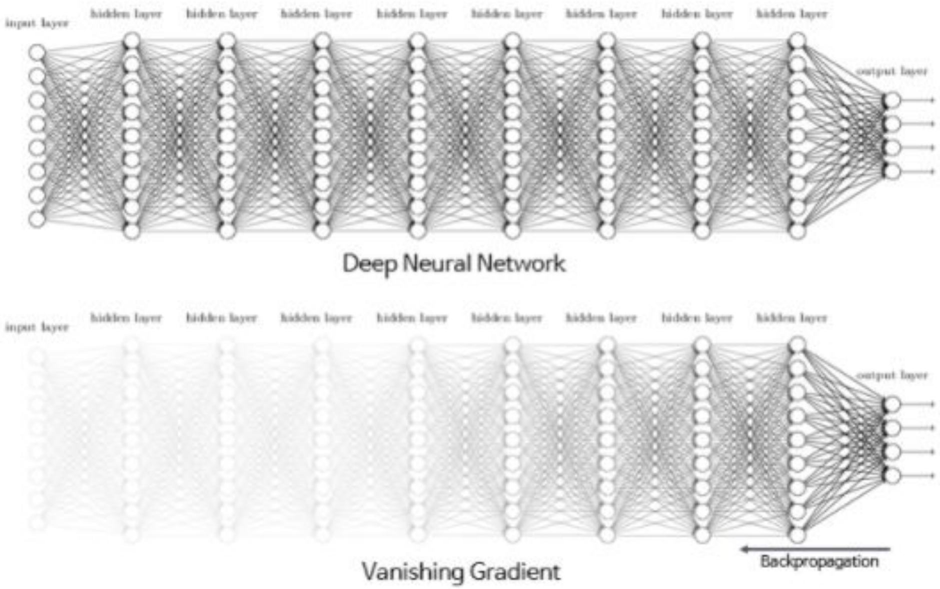
2. 활성화 함수 - 시그모이드 함수



$$\frac{1}{1+e^{-x}}$$

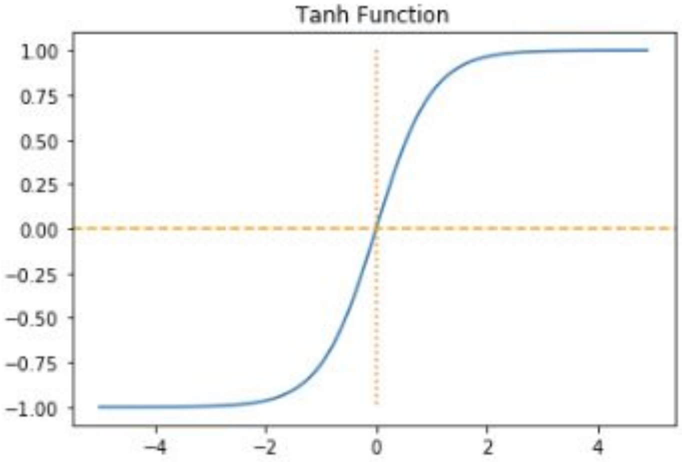
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 활성화 함수 - 시그모이드 함수



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 활성화 함수 - 하이퍼볼릭탄젠트 함수

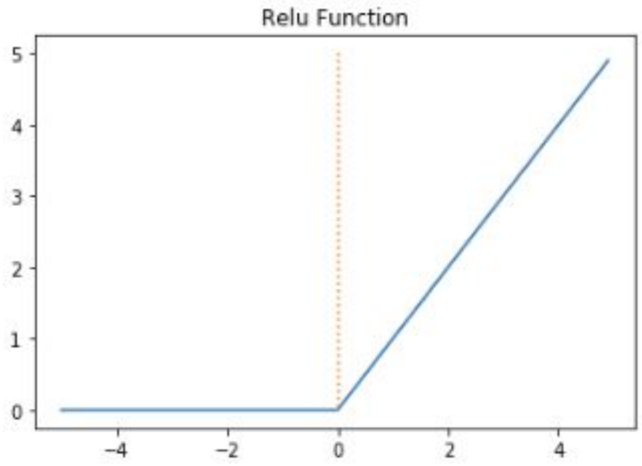


tanh

$$\tanh(x)$$

딥러닝을 이용한 자연어 처리 입문 https://wikidocs.net/24987, ,미디엄@shrutijadon10104776, activation-functions-for-deep-learning,

2. 활성화 함수 - 렐루 함수

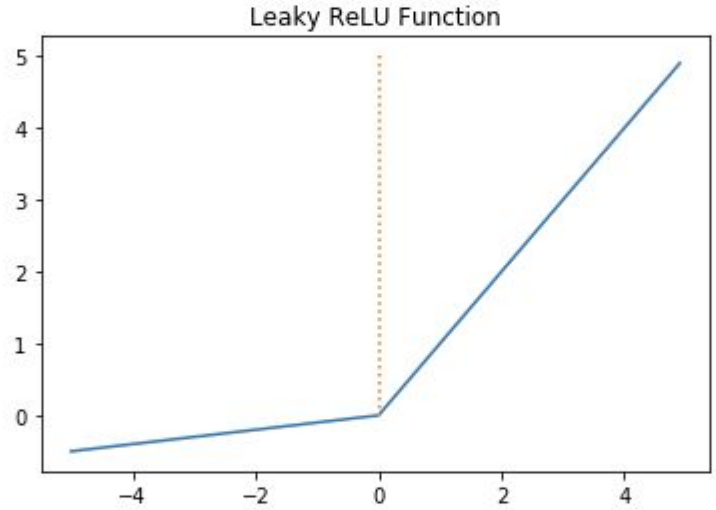


ReLU

$\max(0, x)$

딥러닝을 이용한 자연어 처리 입문 https://wikidocs.net/24987, ,미디엄@shrutijadon10104776, activation-functions-for-deep-learning,

2. 활성화 함수 - 리키 렐루 함수

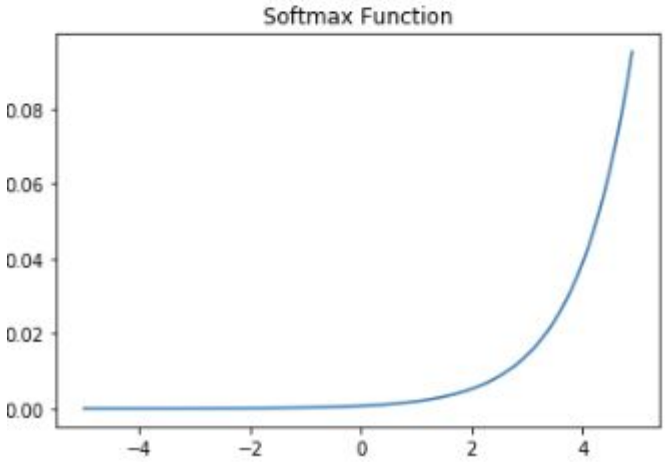


Leaky ReLU

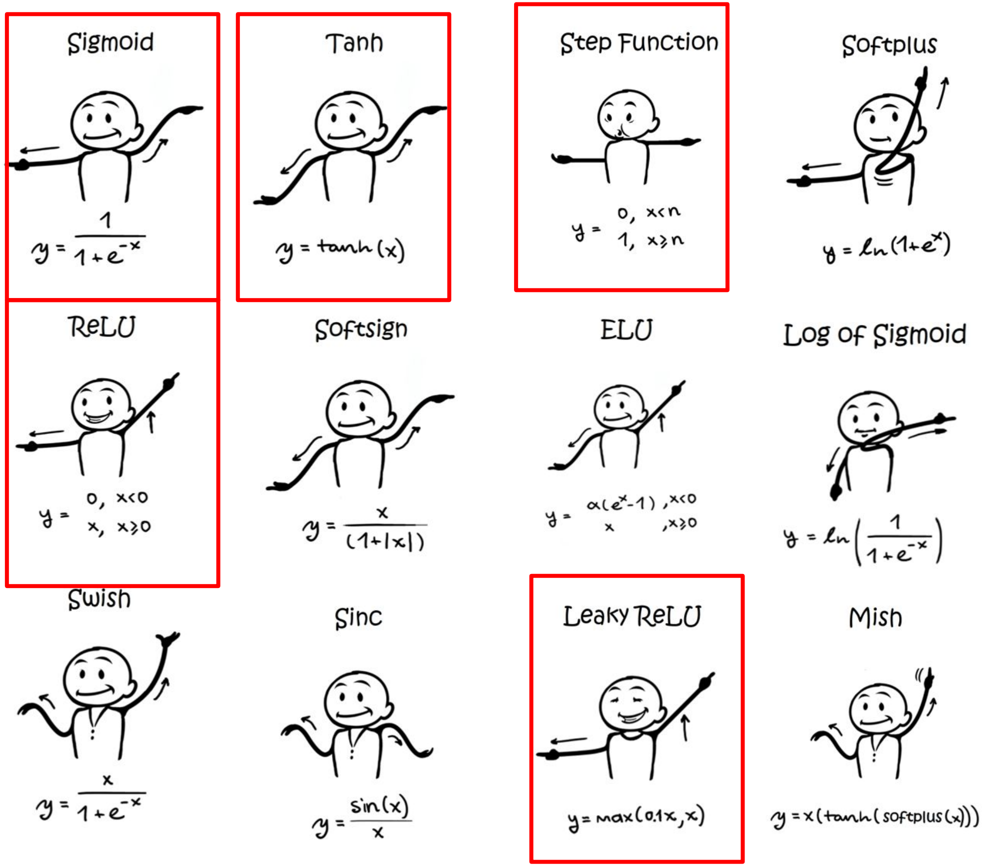
$$\max(0.1x, x)$$

딥러닝을 이용한 자연어 처리 입문 https://wikidocs.net/24987, ,미디엄@shrutijadon10104776, activation-functions-for-deep-learning,

2. 활성화 함수 - 소프트맥스 함수

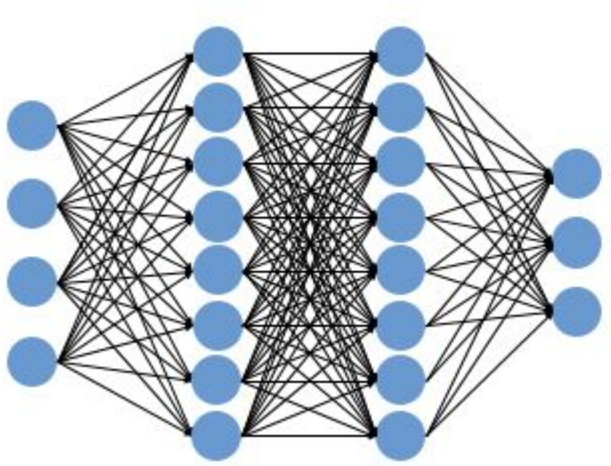


딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**



Dance moves of DL activation function - sefik

3. 행렬의 곱셈을 이용한 순전파



입력층 : 4개의 입력과 8개의 출력

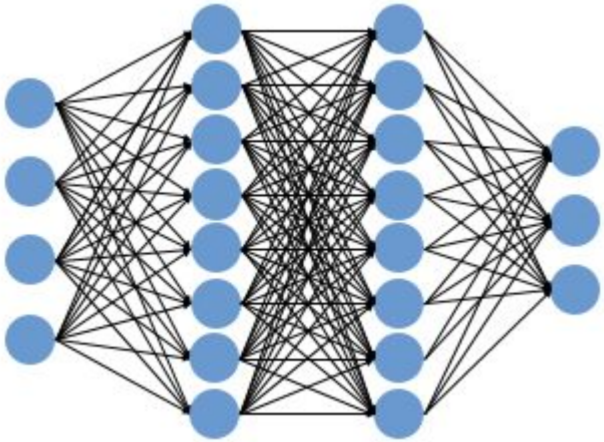
은닉층1 : 8개의 입력과 8개의 출력

은닉층2 : 8개의 입력과 3개의 출력

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

출력층 : 3개의 입력과 3개의

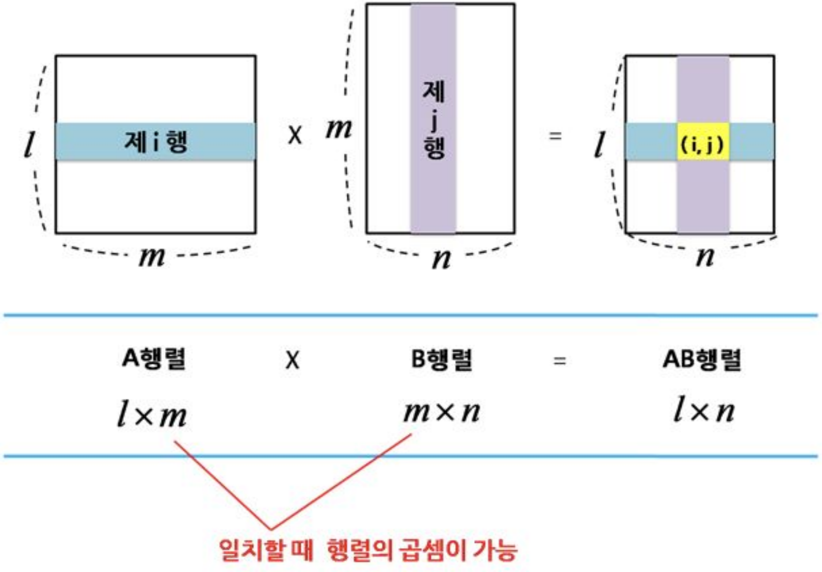
3. 행렬의 곱셈을 이용한 순전파



```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential() # 층을 추가할 준비
model.add(Dense(8, input_dim=4, init='uniform', activation='relu'))
# 입력층(4)과 다음 은닉층(8) 그리고 은닉층의 활성화 함수는 relu
model.add(Dense(8, activation='relu')) # 은닉층(8)의 활성화 함수는 relu
model.add(Dense(3, activation='softmax')) # 출력층(3)의 활성화 함수는 softmax
```

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

3. 행렬의 곱셈을 이용한 순전파



https://j1w2k3.tistory.com/575

3. 행렬의 곱셈을 이용한 순전파

입력층 : 4개의 입력과 8개의 출력

$$\boxed{X_{1\times 4}} \times W_{4\times 8} + B_{1\times 8} = \boxed{Y_{1\times 8}}$$

은닉층1 : 8개의 입력과 8개의 출력

은닉층2 : 8개의 입력과 3개의 출력

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

출력층 : 3개의 입력과 3개의

3. 행렬의 곱셈을 이용한 순전파

입력층 : 4개의 입력과 8개의 출력

은닉층1 : 8개의 입력과 8개의 출력

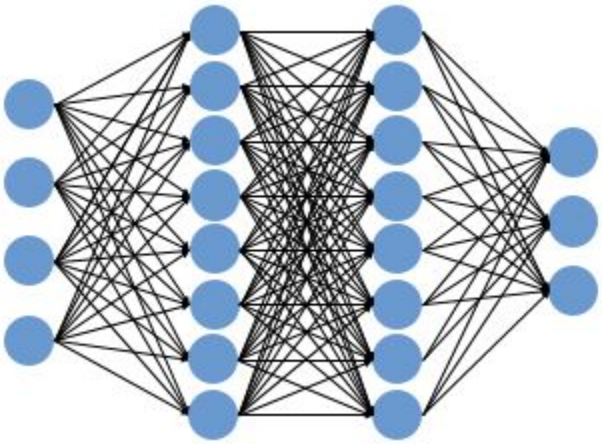
은닉층2 : 8개의 입력과 3개의 출력

$$X_{1\times 8} \times W_{8\times 8} + B_{1\times 8} = Y_{1\times 8}$$
$$X_{1\times 8} \times W_{8\times 3} + B_{1\times 3} = Y_{1\times 3}$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

출력층 : 3개의 입력과 3개의

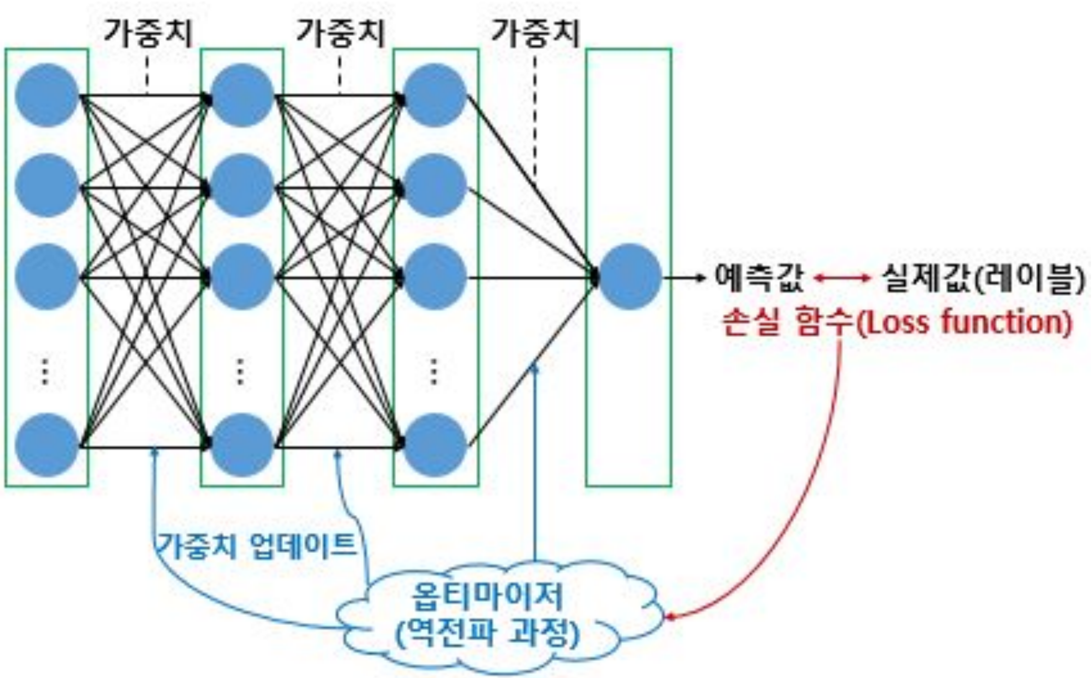
3. 행렬의 곱셈을 이용한 순전파



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

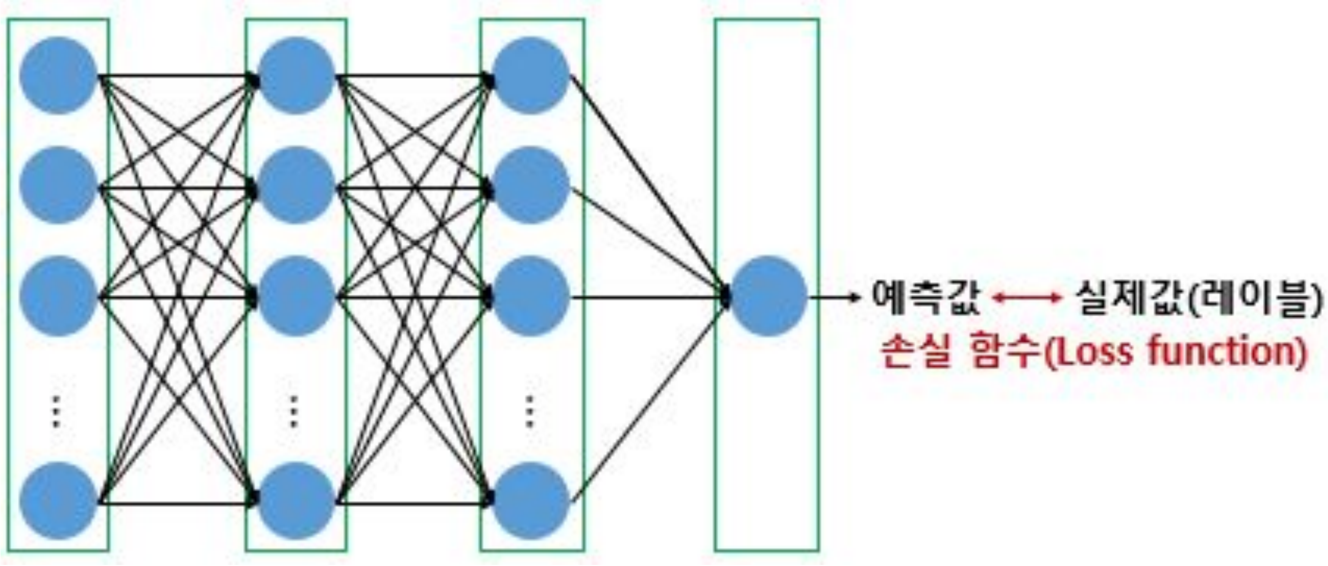
3. 딥러닝의 학습 방법

1. 순전파



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 손실함수



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 손실함수 - MSE

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \widehat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

2. 손실함수 - 크로스 엔트로피

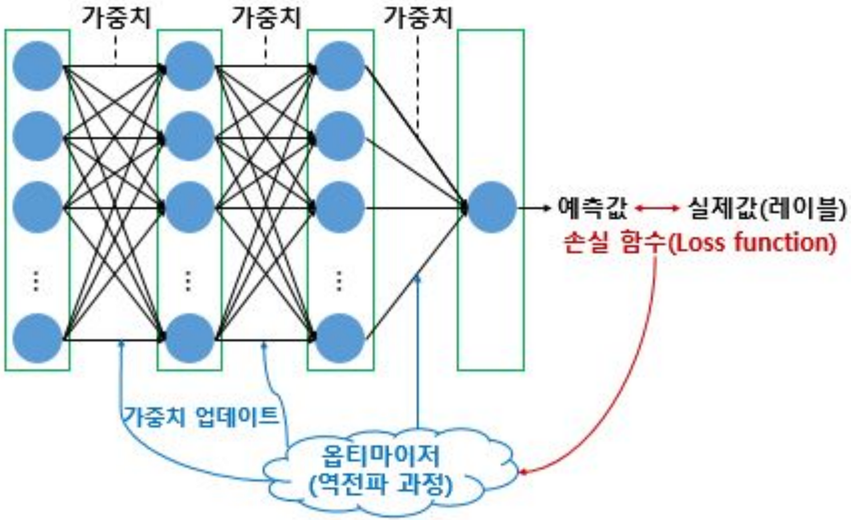
$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i$$

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['acc'])
```

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['acc'])
```

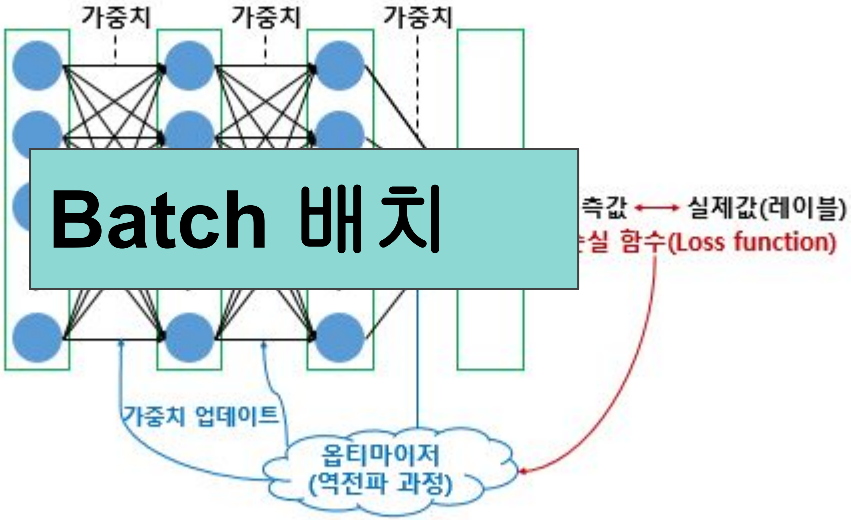
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

3. 옵티마이저



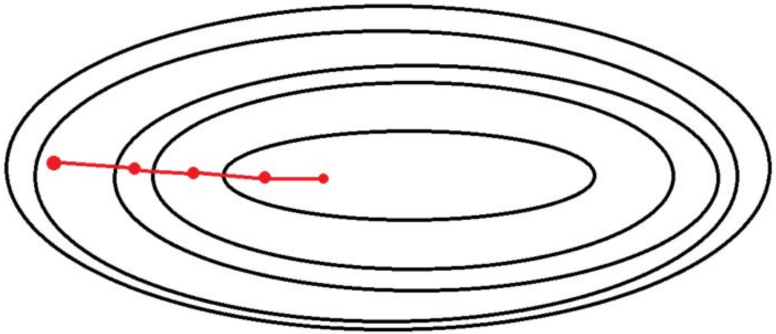
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

3. 옵티마이저



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

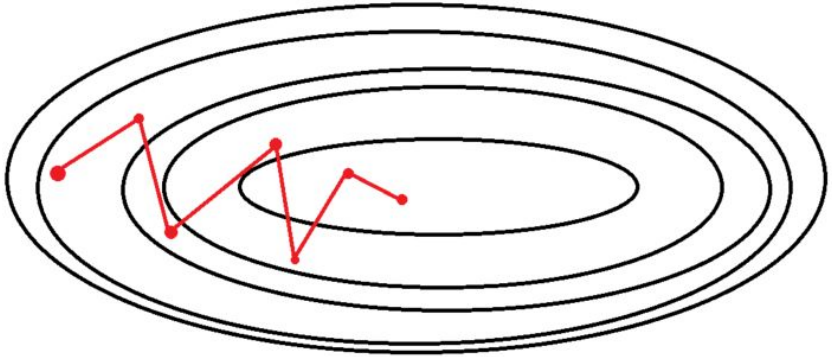
3. 옵티마이저 - 배치 경사 하강법



```
model.fit(X_train, y_train,
          batch_size=len(train_X))
```

컴퓨터와 수학 그리고 몽상 조금 https://skyil.tistory.com/68

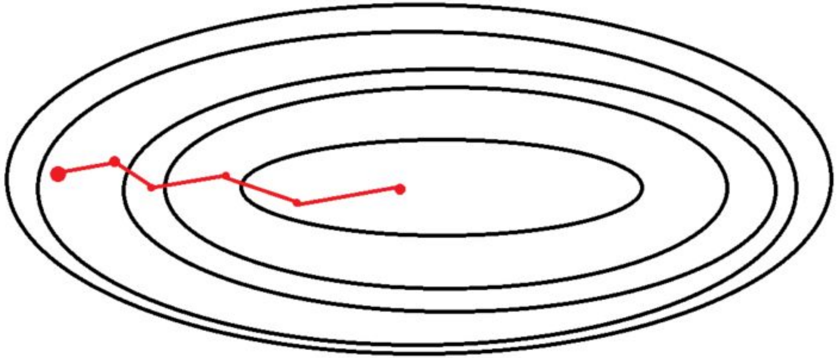
3. 옵티마이저 - 확률적 경사 하강법



```
model.fit(X_train, y_train,
          batch_size=1)
```

컴퓨터와 수학 그리고 몽상 조금 https://skyil.tistory.com/68

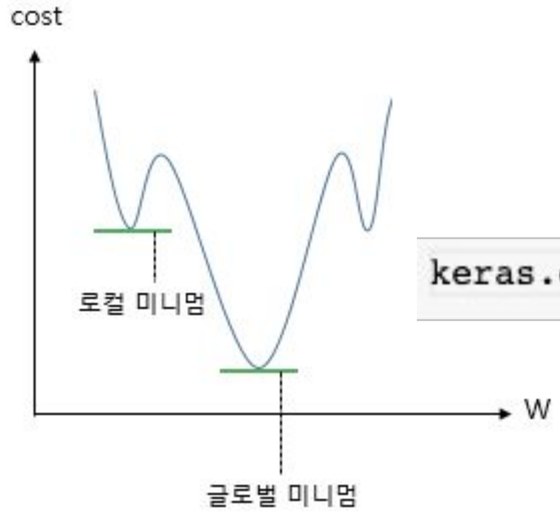
3. 옵티마이저 - 미니 배치 경사 하강법

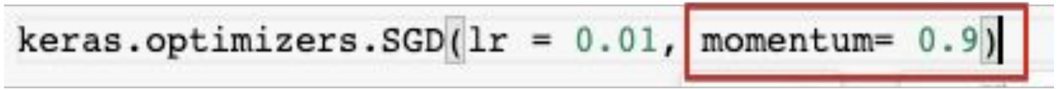


```
model.fit(X_train, y_train,
          batch_size=32)
```

컴퓨터와 수학 그리고 몽상 조금 https://skyil.tistory.com/68

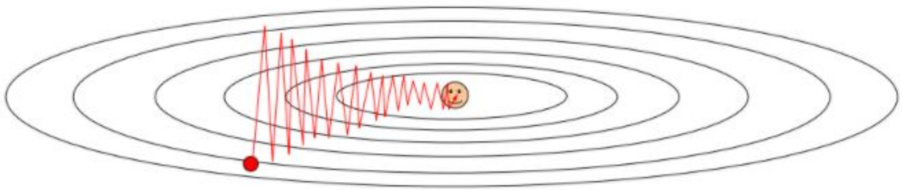
3. 옵티마이저 - 모멘텀





딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

3. 옵티마이저 - 아다그라드(Adagrad)



```
keras.optimizers.Adagrad(lr=0.01, epsilon=1e-6)
```

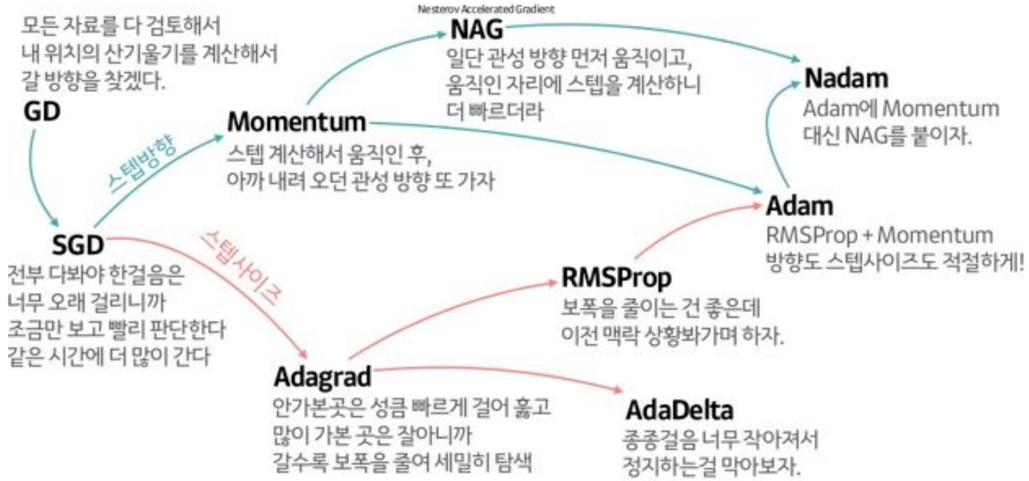
https://velog.io/@reversesky/Optimizer의-종류와-간단한-정리

3. 옵티마이저 - RMSProp, Adam

```
keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=1e-06)
```

```
keras.optimizers.Adam(lr=0.001,
                      beta_1=0.9,
                      beta_2=0.999,
                      epsilon=None,
                      decay=0.0,
                      amsgrad=False)
```

3. 옵티마이저

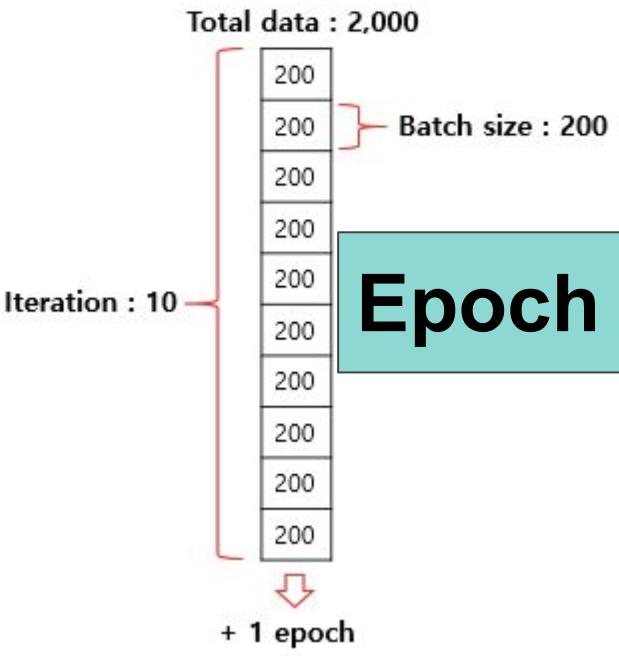


https://blog.naver.com/another0430/222063836606

4. 에포크, 배치크기 그리고 이터레이션

https://theround-uponline.com/student-life/2020/02/13/test-tips/

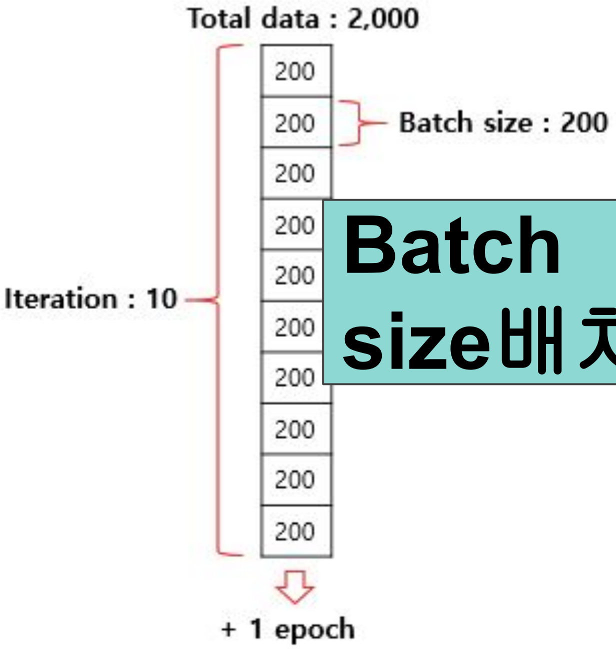
4. 에포크, 배치크기 그리고 이터레이션



Epoch 에포크

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

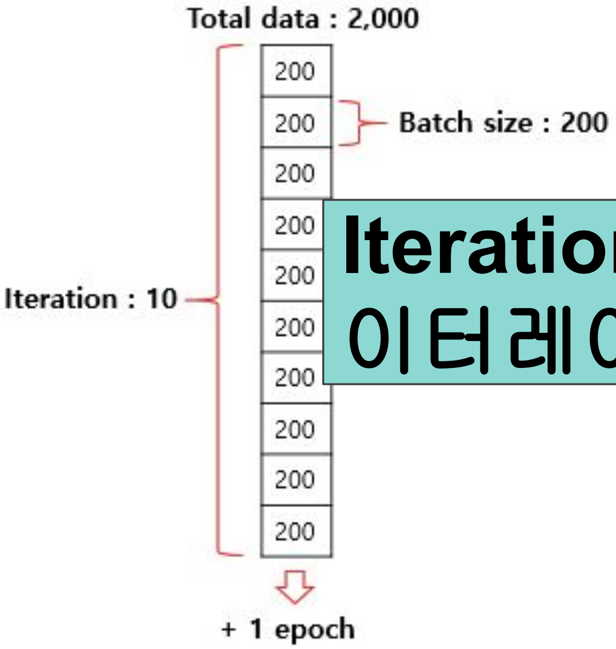
4. 에포크, 배치크기 그리고 이터레이션



Batch size배치크기

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

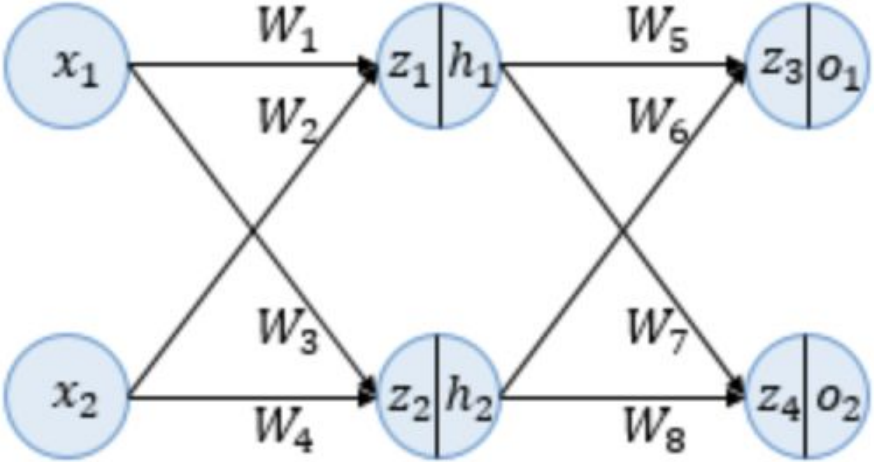
4. 에포크, 배치크기 그리고 이터레이션



Iteration
이터레이션

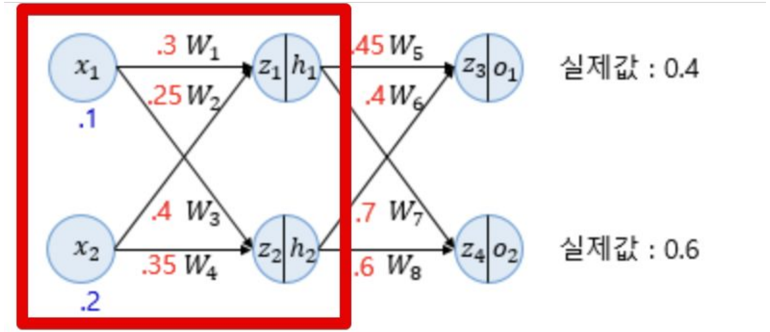
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

5. 순전파



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

5. 순전파



$$z1 = 0.08$$

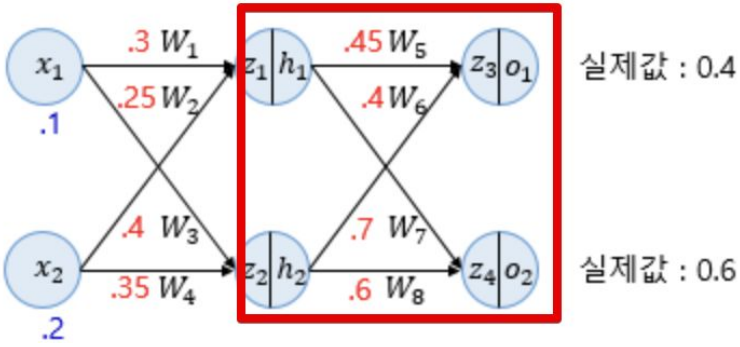
$$z2 = 0.11$$

$$h1 = \text{sigmoid}(z1) = 0.51999$$

$$h2 = \text{sigmoid}(z2) = 0.52747$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

5. 순전파



$$z3 = 0.44498$$

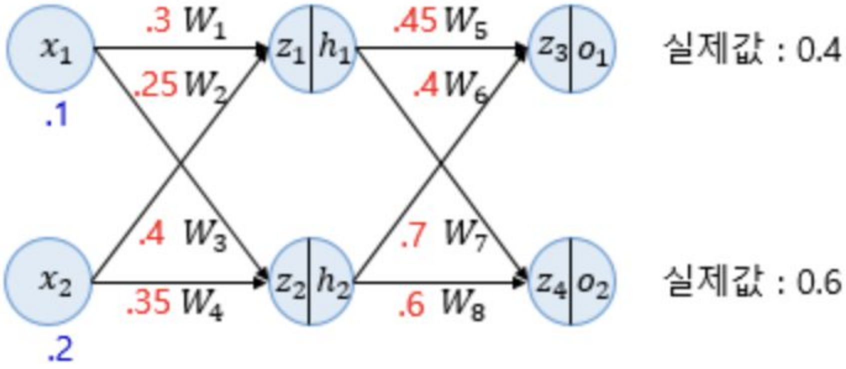
$$z4 = 0.68047$$

$$o1 = \text{sigmoid}(z3) = 0.60944$$

$$o2 = \text{sigmoid}(z4) = 0.68047$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

5. 순전파



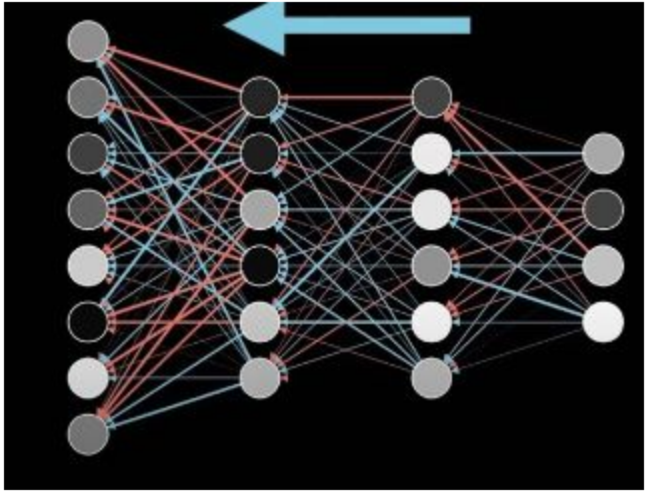
$$E_{o1} = \frac{1}{2}(target_{o1} - output_{o1})^2 = 0.02193381$$

$$E_{o2} = \frac{1}{2}(target_{o2} - output_{o2})^2 = 0.00203809$$

$$E_{total} = E_{o1} + E_{o2} = 0.02397190$$

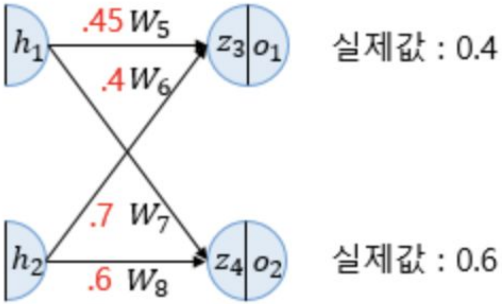
딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

6. 역전파



테디노트, 오차역전파, https://youtu.be/1Q_etC_GHHk

6. 역전파 1단계



$$\frac{\partial E_{total}}{\partial W_5} = \frac{\partial E_{total}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_5}$$

6. 역전파 1단계

$$\frac{\partial E_{total}}{\partial W_5} = \boxed{\frac{\partial E_{total}}{\partial o_1}} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_5}$$

$$\frac{\partial E_{total}}{\partial o_1} = 2 \times \frac{1}{2}(target_{o1} - output_{o1})^{2-1} \times (-1) + 0$$

$$\frac{\partial E_{total}}{\partial o_1} = 0.20944600$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

6. 역전파 1단계

$$\frac{\partial E_{total}}{\partial W_5} = \frac{\partial E_{total}}{\partial o_1} \times \boxed{\frac{\partial o_1}{\partial z_3}} \times \frac{\partial z_3}{\partial W_5}$$

$$\frac{\partial o_1}{\partial z_3} = o_1 \times (1 - o_1) = 0.23802157$$

6. 역전파 1단계

$$\frac{\partial E_{total}}{\partial W_5} = \frac{\partial E_{total}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \boxed{\frac{\partial z_3}{\partial W_5}}$$

$$\frac{\partial z_3}{\partial W_5} = h_1 = 0.51998934$$

6. 역전파 1단계

$$\frac{\partial E_{total}}{\partial W_5} = \frac{\partial E_{total}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_5} = 0.02592286$$

$$W_5^+ = W_5 - \alpha \frac{\partial E_{total}}{\partial W_5} = 0.45 - \boxed{0.5} \times 0.02592286 = 0.43703857$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

6. 역전파 1단계

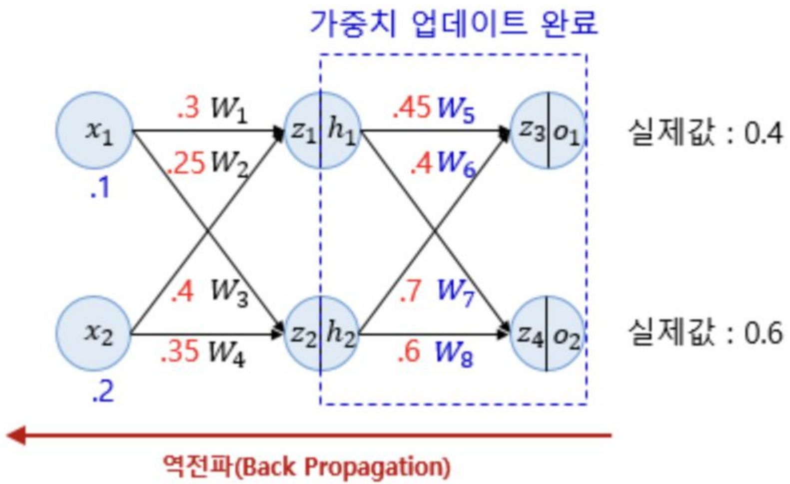
$$\frac{\partial E_{total}}{\partial W_6} = \frac{\partial E_{total}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_6} \rightarrow \boxed{W_6^+} = 0.38685205$$

$$\frac{\partial E_{total}}{\partial W_7} = \frac{\partial E_{total}}{\partial o_2} \times \frac{\partial o_2}{\partial z_4} \times \frac{\partial z_4}{\partial W_7} \rightarrow \boxed{W_7^+} = 0.69629578$$

$$\frac{\partial E_{total}}{\partial W_8} = \frac{\partial E_{total}}{\partial o_2} \times \frac{\partial o_2}{\partial z_4} \times \frac{\partial z_4}{\partial W_8} \rightarrow \boxed{W_8^+} = 0.59624247$$

딥러닝을 이용한 자연어 처리 입문 https://wikidocs.net/24987

6. 역전파 2단계



딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

6. 역전파 2단계

$$\boxed{\frac{\partial E_{total}}{\partial W_1}} = \frac{\partial E_{total}}{\partial h_1} \times \frac{\partial h_1}{\partial z_1} \times \frac{\partial z_1}{\partial W_1} = 0.00080888$$

$$W_1^+ = W_1 - \alpha \frac{\partial E_{total}}{\partial W_1} = 0.1 - \boxed{0.5} \times 0.00080888 = 0.29959556$$

6. 역전파 2단계

$$\frac{\partial E_{total}}{\partial W_2} = \frac{\partial E_{total}}{\partial h_1} \times \frac{\partial h_1}{\partial z_1} \times \frac{\partial z_1}{\partial W_2} - W_2^+ = 0.24919112$$

$$\frac{\partial E_{total}}{\partial W_3} = \frac{\partial E_{total}}{\partial h_2} \times \frac{\partial h_2}{\partial z_2} \times \frac{\partial z_2}{\partial W_3} - W_3^+ = 0.39964496$$

$$\frac{\partial E_{total}}{\partial W_4} = \frac{\partial E_{total}}{\partial h_2} \times \frac{\partial h_2}{\partial z_2} \times \frac{\partial z_2}{\partial W_4} - W_4^+ = 0.34928991$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

5. 역전파 - 결과확인

$$w7 = 0.7$$

$$W_7^+ = 0.69629578$$

$$w8 = 0.6$$

$$W_8^+ = 0.59624247$$

$$\boxed{E_{total} = E_{o1} + E_{o2} = 0.02397190}$$

$$\boxed{E_{total} = E_{o1} + E_{o2} = 0.02323634}$$

딥러닝을 이용한 자연어 처리 입문 **https://wikidocs.net/24987**

5. 역전파 - 영상추천

유튜브 테디노트 “오차역전파"의 개념을 쉽게 알아봅시다

8. 딥 러닝(Deep Learning)

9조 이제일, 금예은

Part 4, 과적합(Overfitting)을 막는 방법들

0. 과적합(Overfitting) 이란?

과적합(Overfitting)

: 학습 데이터를 과하게 잘 학습하여 훈련 데이터의 노이즈까지 학습해, 새로운 테스트 데이터에 대한 정확도가 낮아 제대로 동작하지 않는 것.

1. 데이터 양을 늘리기

Data 양↓ → 특정 패턴 / 노이즈까지 학습될 수도,, → Overfitting 발생 확률↑

>> data 양 늘려 일반적인 패턴을 학습하도록 하여 Overfitting 방지 가능

▶ 데이터 증식 / 데이터 증강(Data Augmentation)

: 보유하고 있는 데이터의 양이 적을 경우, 의도적으로 데이터 양을 늘리는 방법

Image data의 경우, rotation / noise 추가 등의 방법을 이용해 데이터 증식

2. 모델의 복잡도 줄이기

신경망의 복잡도는 은닉층(hidden layer)의 수나 매개변수의 수 등으로 결정

Overfitting 발생 시, 복잡도를 줄이면 개선 가능

* 모델의 수용력(capacity): 모델에 있는 매개변수의 수

3. 가중치 규제(Regularization) 적용하기

	L1 규제(L1 노름)	L2 규제(L2 노름)
비용 함수		
적용할 상황	어떤 특성이 모델에 영향을 주고 있는지 정확히 판단하고자 할 때 유용	특정 특성이 모델에 영향을 주는지 판단할 때 외에는 L2 규제가 잘 동작
		가중치 감쇠(weight decay)라고도 부름

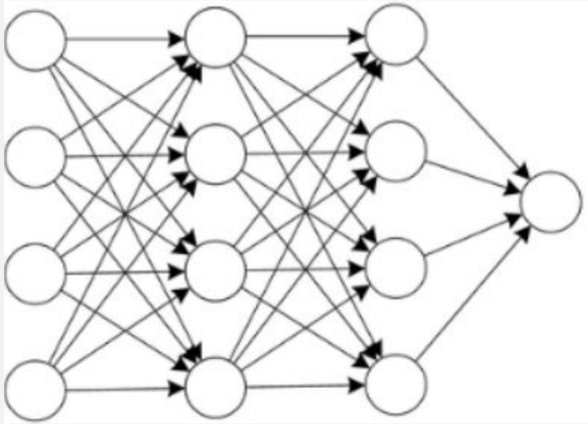
λ : 규제의 강도를 정하는 하이퍼파라미터
(λ가 크다면 규제를 위해 추가된 항을 작게 유지하는 것을 우선)

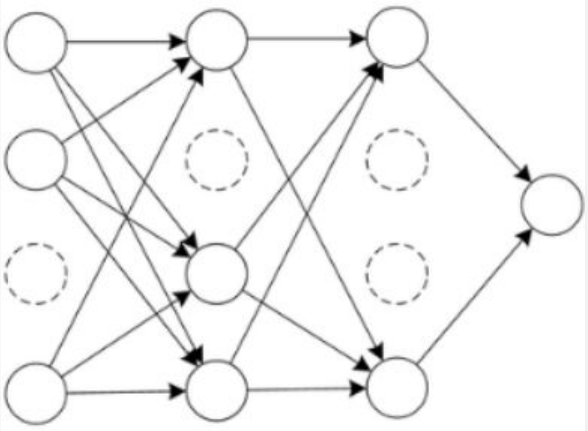
4. 드롭아웃 (Dropout)

드롭아웃(Dropout): 학습 과정에서 신경망 일부를 사용하지 않는 방법

- 신경망 학습 시에만 사용 (예측 시 사용X)
- 학습 시, 신경망이 특정 뉴런/조합에 의존하는 것 방지해 Overfitting 방지

```
model = Sequential()
model.add(Dense(256, input_shape=(max_words,), activation='relu'))
model.add(Dropout(0.5)) # 드롭아웃 추가. 비율은 50%
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5)) # 드롭아웃 추가. 비율은 50%
model.add(Dense(num_classes, activation='softmax'))
```





Part 5

,

기울기 소실(Gradient Vanishing) 과 폭주(Exploding)

0. 기울기 소실과 폭주란?

기울기 소실(Gradient Vanishing)

: 신경망에서 가중치를 학습시키는 과정에서 곡선의 기울기가 0이 되는 것

기울기 폭주(Gradient Exploding)

: 신경망에서 가중치를 학습시킬 때, 가중치가 발산하는 것

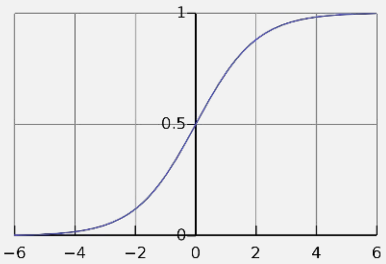
1. ReLU와 ReLU의 변형들

Hidden layer에서 Sigmoid function 사용 → 입력 절대값 크면 기울기가 0에 가까워짐

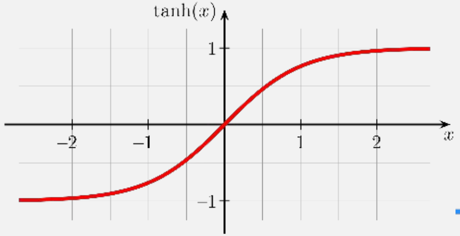
>> 기울기 소실 문제 발생할 수도

Hidden layer의 활성 함수로 sigmoid function이나 tanh function 사용 X

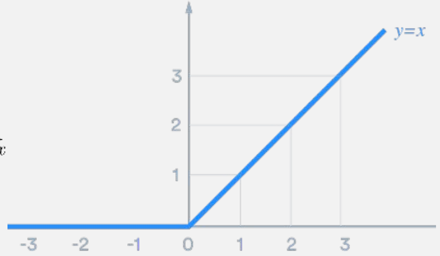
→ ReLU나 Leaky ReLU와 같은 ReLU의 변형을 사용(입력에 대한 기울기가 0에 수렴하지 X)



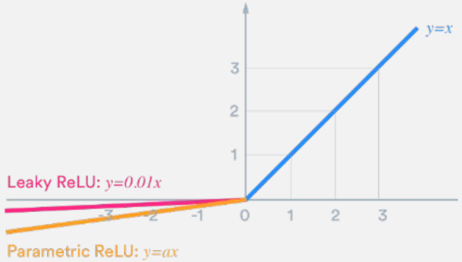
Sigmoid Function



tanh Function



ReLU Function



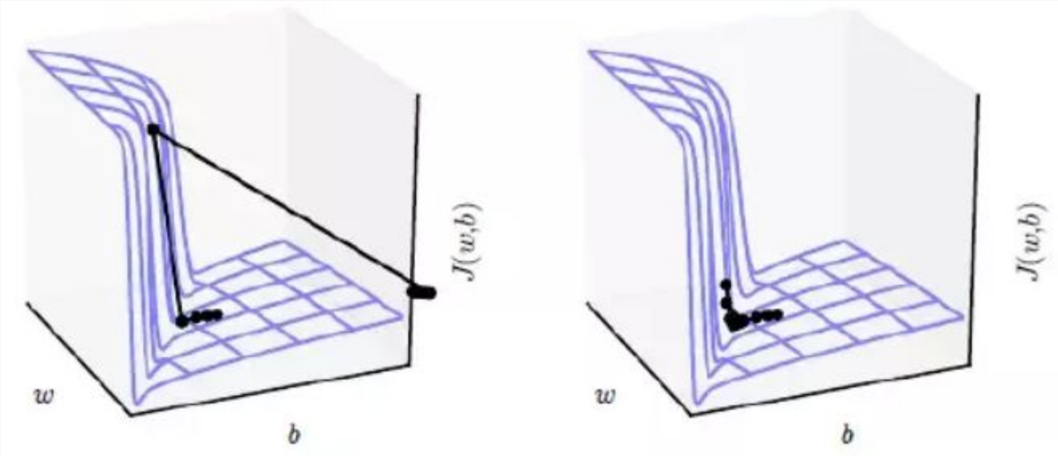
Leaky ReLU Function

2. 그래디언트 클리핑(Gradient Clipping)

```
from tensorflow.keras import optimizers
Adam = optimizers.Adam(lr=0.0001, clipnorm=1.)
```

그래디언트 클리핑(Gradient Clipping)

: 기울기 폭주를 막기 위해 임계 값을 넘지 않도록 기울기 값을 자르는 것



3. 가중치 초기화(Weight Initialization)

1) 세이비어 초기화(Xavier Initialization) / 글로럿 초기화(Glorot Initialization)

n_{in} : 이전 층의 뉴런 개수
n_{out} : 다음 층의 뉴런 개수

① 균등 분포(Uniform Distribution)로 가중치 초기화할 때

$$W \sim Uniform(-\sqrt{\frac{6}{n_{in}+n_{out}}}, +\sqrt{\frac{6}{n_{in}+n_{out}}})$$

② 정규 분포(Normal Distribution)로 가중치 초기화할 때

평균이 0이고, 표준편차 $\sigma = \sqrt{\frac{2}{n_{in}+n_{out}}}$를 만족하도록 초기화

이는 sigmoid function이나 tanh function과 같이 S자형 활성화 함수와 함께 사용하면 성능이 좋음

3. 가중치 초기화(Weight Initialization)

n_{in} : 이전 층의 뉴런 개수

2) He 초기화(He Initialization)

① 균등 분포(Uniform Distribution)로 가중치 초기화할 때

$$W \sim Uniform(-\sqrt{\frac{6}{n_{in}}}, +\sqrt{\frac{6}{n_{in}}})$$

② 정규 분포(Normal Distribution)로 가중치 초기화할 때

평균이 0이고, 표준편차 $\sigma = \sqrt{\frac{2}{n_{in}}}$를 만족하도록 초기화

ReLU 계열의 함수를 사용할 경우, He 초기화 방법이 효율적

가중치 초기화할 경우, ReLU + He 초기화 방법이 보편적

4. 배치 정규화(Batch Normalization)

1) 내부 공변량 변화(Internal Covariate Shift)

: 학습 과정에서 신경망 층 사이에서 발생하는 입력 데이터 분포 변화

참고 논문: **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**

4. 배치 정규화(Batch Normalization)

2) 배치 정규화(Batch Normalization)

: 각 layer의 활성화 함수의 출력값 분포가 골고루 분포되도록 강제하는 것

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$: 미니 배치에 대한 평균 계산

$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu_B)^2$: 미니 배치에 대한 분산 계산

$\hat{x}^{(i)} \leftarrow \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$: 정규화

$y^{(i)} \leftarrow \gamma \hat{x}^{(i)} + \beta = BN_{\gamma,\beta}(x^{(i)})$: 스케일 조정과 시프트를 통한 선형 연산

γ : 스케일 조정을 위한 매개변수
β : 시프트를 위한 매개변수

4. 배치 정규화(Batch Normalization)

▶ 장점

- Sigmoid 함수나 tanh 함수를 사용하더라도 기울기 소실 문제가 크게 개선
- 가중치 초기화에 비해 비교적 덜 민감
- 훨씬 큰 학습률 사용 가능 → 학습 속도 개선
- 미니 배치에 대한 정규화로 인해 드롭아웃과 비슷한 효과를 내 과적합을 방지

4. 배치 정규화(Batch Normalization)

3) 배치 정규화의 한계

① 미니 배치 크기에 의존적

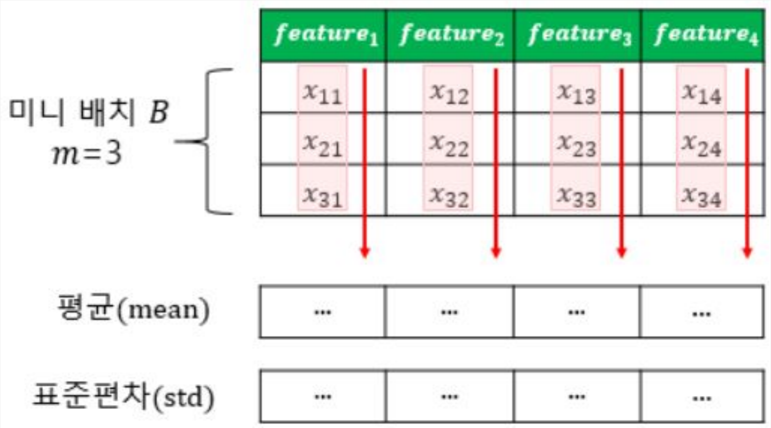
배치 크기가 작다면 배치 정규화가 제대로 동작하지 않을 수 있음

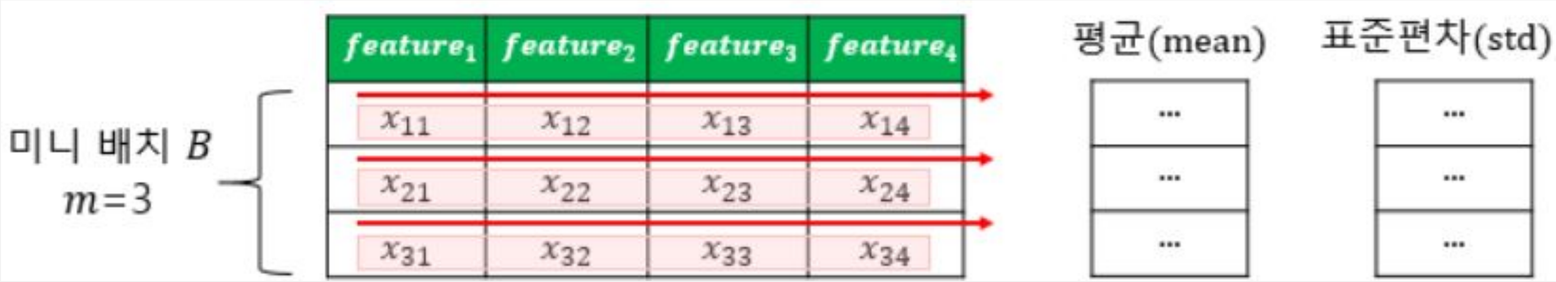
어느 정도 크기가 되는 배치에 배치 정규화를 적용하는 것이 좋음

② RNN에 적용하기 어려움

RNN: 각 시점마다 다른 값을 갖는 특성 → 배치 정규화 적용이 어려움

5. 층 정규화(Layer Normalization)





감사합니
다