기계학습 활용 (11주차)

2019. 11. 22.

Prof. Seung Ho Lee

KOREATECH
한국기술교육대학교

강의 주제 : 딥러닝 활용(다중 분류)

- 이론
  - 오차 역전파의 이해
  - 기울기 소실 문제
  - 다중 분류

- 실습
  - 다중 분류 (아이리스 품종 예측하기)

지난 강의 요약

- 딥러닝 모델 설계 및 구현
  - 딥러닝 모델의 설계 방법(구조 및 학습 측면)
  - 폐암 생존여부 예측을 위한 딥러닝 모델 구현 및 평가

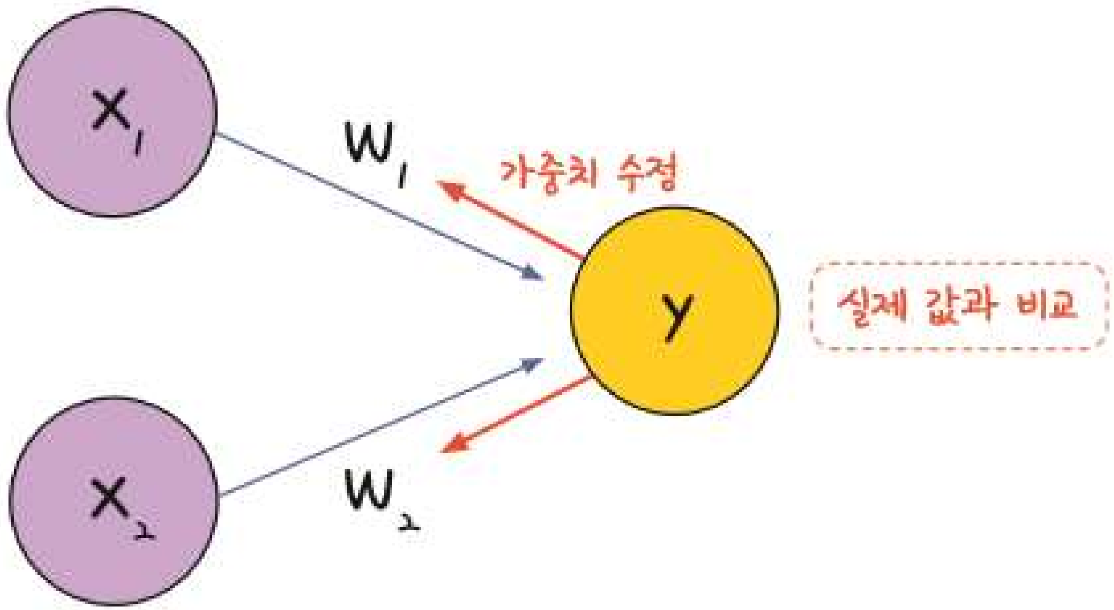
오늘 강의 요약

- 목표
  - 은닉층을 포함하는 신경망의 가중치 학습 원리 이해
    - ✓ 오차 역전파 사용
  - 오차 역전파의 기울기 소실 문제와 극복 방법 이해
  - 다중 분류 관련 개념 이해

퍼셉트론 학습과 오차 역전파

▪ 단일 퍼셉트론 학습

- 앞서 배운 경사 하강법은 입력층과 출력층만 있는 단일 퍼셉트론에 해당함
- 단일 퍼셉트론에서 출력 값을 얻으면 실제 값과 비교하여 오차를 구해 앞 단계에서 정한 가중치 및 바이어스를 조정

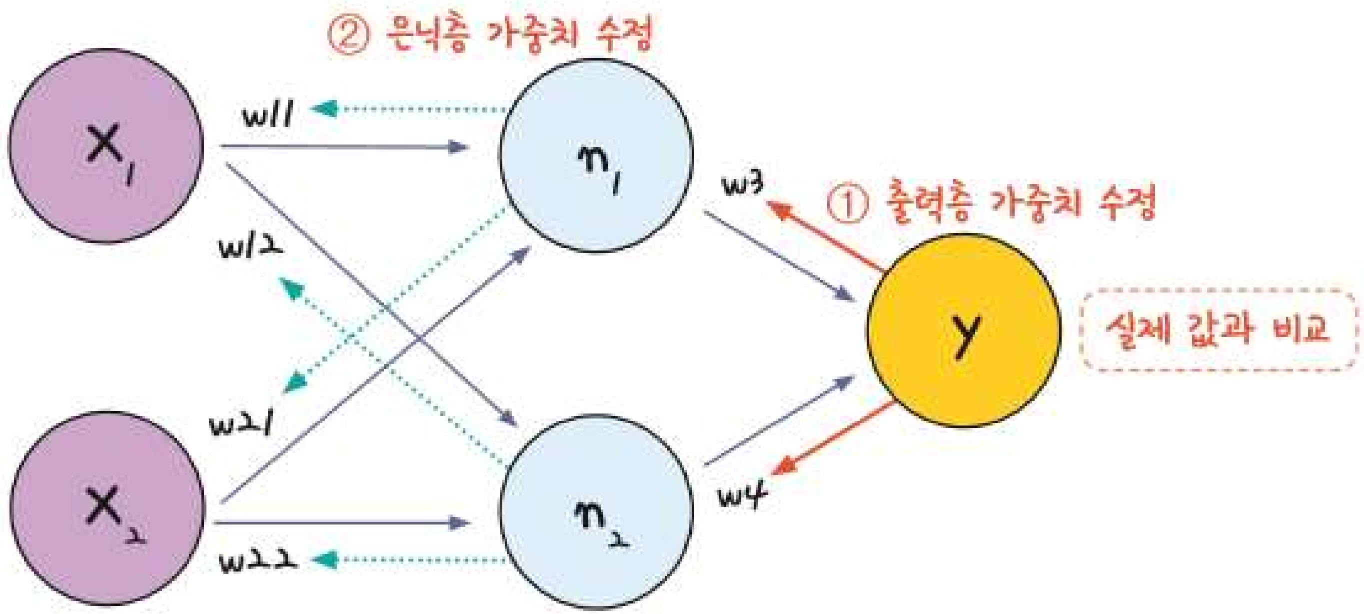


퍼셉트론 학습과 오차 역전파

▪ 다층 퍼셉트론 학습

- 입력층과 출력층 사이에 은닉층이 추가된 경우
- 출력 값에 대한 오차를 구해 이를 토대로 앞 단계의 가중치 및 바이어스를 차례로 거슬러 올라가며 조정해 감
- 최적화의 계산 방향이 출력층에서 시작해 앞으로 거꾸로 진행됨

→ 오차 역전파Back propagation 라고 부름



출처 : 모두의 딥러닝(조태호)

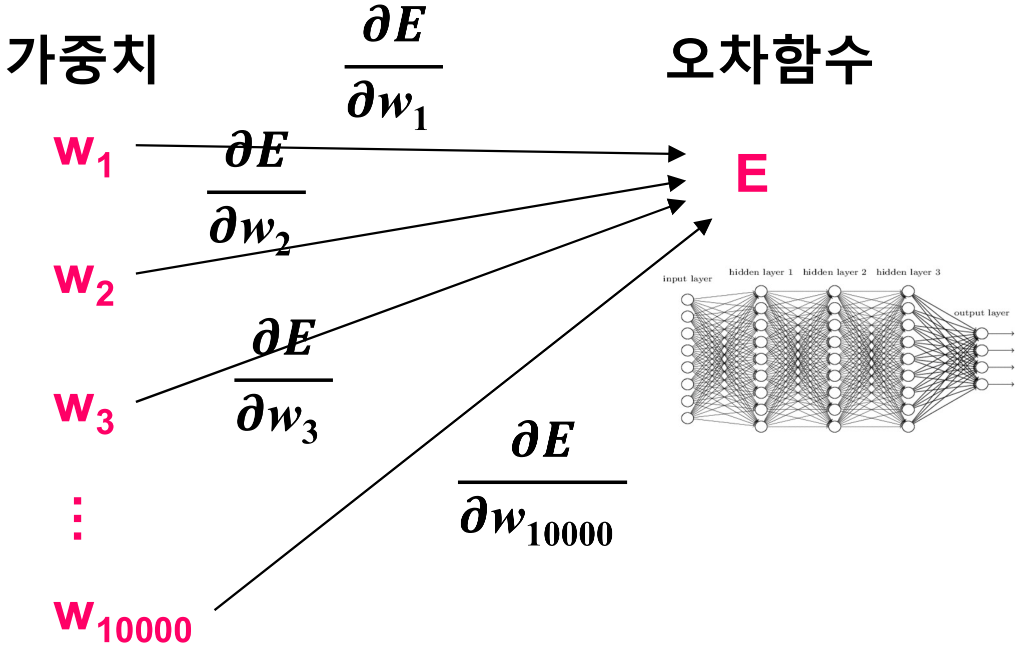
퍼셉트론 학습과 오차 역전파

- ## 경사 하강법과 미분
  - 경사 하강법 : 미분의 개념을 사용하여 가중치를 반복 갱신
  - 은닉층이 많아지면 오차함수의 식이 너무 복잡해져 미분을 계산하기 어려움 → 연쇄법칙으로 해결

$$W_5(t+1) = W_5(t) - \boxed{\frac{\partial E_{total}}{\partial W_5}}$$

갱신 후 가중치 / 현 시점 가중치 / 미분

가중치 갱신 규칙



퍼셉트론 학습과 오차 역전파

▪ [참고] 편미분 이해하기

$$f(x,\ y) = x^2 + xy + a \text{일 때,}$$

$$\frac{\partial f}{\partial x} = 2x + y$$

- 먼저 미분의 성질 $\frac{d}{dx}(x^n) = nx^{n-1}$에 의해 x^2항은 $2x$가 됨
- 그리고 x에 관해 미분하므로 y는 상수로 취급해야 함
- 그러면 미분 공식 $\frac{d}{dx}(ax) = a$(a = 상수)에 해당함
- 따라서 xy가 들어 있는 항은 미분하면 y가 됨
- 또한 $\frac{d}{dx}(a) = 0$에 의해 a항은 미분하면 0이 됨

퍼셉트론 학습과 오차 역전파

- **[참고] 연쇄법칙Chain Rule 이해하기**

$z = (x + y)^2$: z를 x에 대해 미분하는 문제

$$t = (x + y)$$
$$z = t^2$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\cancel{\partial t}} \frac{\cancel{\partial t}}{\partial x}$$

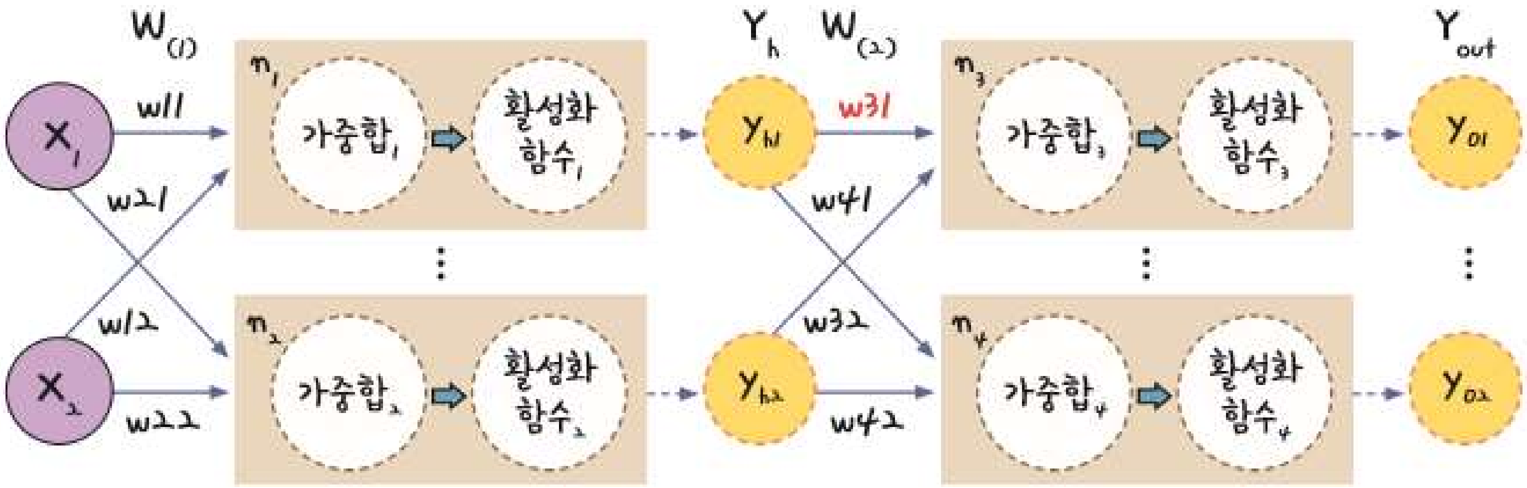
$$\frac{\partial z}{\partial t} = 2t$$
$$\frac{\partial t}{\partial x} = 1$$

$$\frac{\partial z}{\partial x} = 2t \cdot 1 = 2(x + y)$$

오차 역전파로 가중치 갱신하기

- 가중치 w_{31} 갱신하기

$$w_{31}(t+1)=w_{31}(t)-\boxed{\frac{\partial(\text{오차 } Y_{out})}{\partial w_{31}}}$$



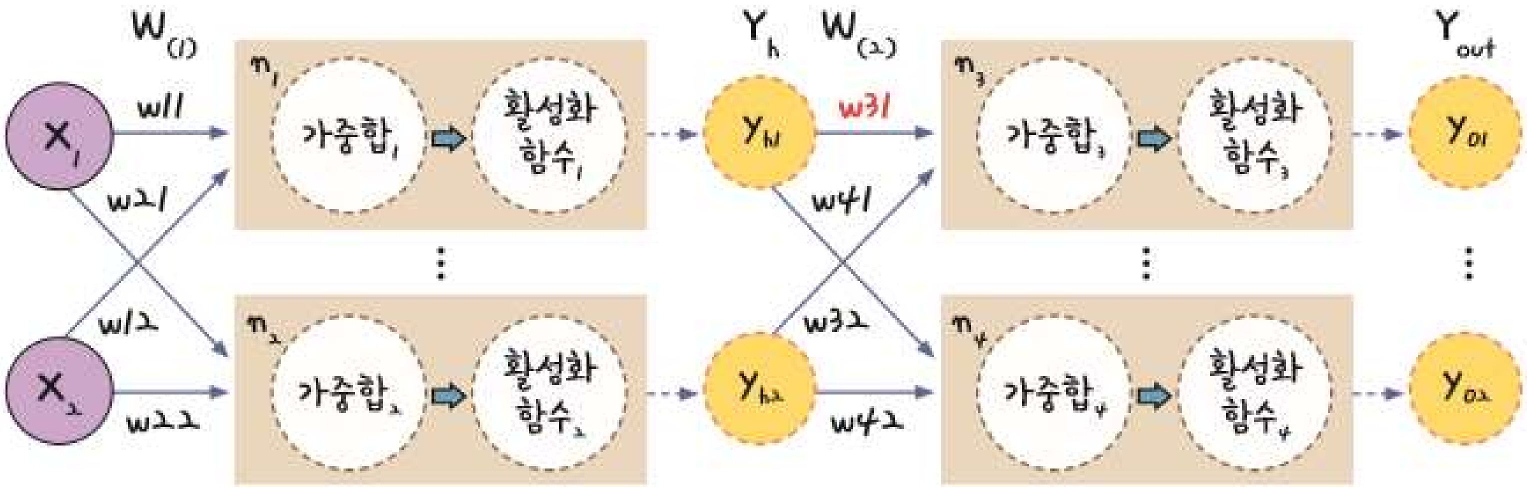
출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

- 가중치 w_{31} 갱신하기
  - 연쇄법칙을 적용하여 최대한 단순한 미분으로 분해

$$w_{31}(t+1) = w_{31}(t) - \boxed{\frac{\partial(\text{오차 } Y_{out})}{\partial w_{31}}}$$

$$\frac{\partial \text{오차} Y_{\text{out}}}{\partial w_{31}} = \underbrace{\frac{\partial \text{오차} Y_{\text{out}}}{\partial y_{o1}}}_{1} \cdot \underbrace{\frac{\partial y_{o1}}{\partial \text{가중합}_3}}_{2} \cdot \underbrace{\frac{\partial \text{가중합}_3}{\partial w_{31}}}_{3}$$



출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

▪ 가중치 w_{31} 갱신하기

① $\dfrac{\partial 오차 Y_{out}}{\partial y_{o1}}$

$$오차\ Y_{out} = \frac{1}{2}(\underset{정답}{y_{t1}} - \underset{예측}{y_{o1}})^2 + \frac{1}{2}(\underset{정답}{y_{t2}} - \underset{예측}{y_{o2}})^2$$

(오차함수에서 Root는 편의상 생략하였음)

- 이를 y_{o1}에 의해서 편미분하면 y_{o1}과 관계없는 y_{o2} 부분은 상수가 되어 사라짐
- 남은 오차는 $\frac{1}{2}(y_{t1} - y_{o1})^2$
- $\frac{1}{2}(y_{t1} - y_{o1})^2$를 y_{o1}에 대해 편미분하면 $y_{o1} - y_{t1}$이 됨

$$\frac{\partial 오차 Y_{out}}{\partial y_{o1}} = y_{o1} - y_{t1} \quad ①$$

오차 역전파로 가중치 갱신하기

▪ 가중치 w_{31} 갱신하기

$$② \quad \frac{\partial y_{o1}}{\partial 가중합_3}$$

미분 = 가중합이 조금 변했을 때 활성화 함수의 출력값이 변하는 정도

- 여기서 가중합$_3$이 y_{o1}으로 바뀌는 과정에는 활성화 함수$_3$을 거치는 것을 알 수 있음
- 가중합$_3$이 활성화 함수$_3$을 통해 y_{o1}이 됨
- 그러면 가중합$_3$을 y_{o1}에 대해 미분하라는 것은 y_{o1}을 출력한 활성화 함수$_3$을 미분하라는 뜻이 됨

$$\frac{\partial y_{o1}}{\partial 가중합_3} = 활성화\ 함수_3의\ 미분$$

오차 역전파로 가중치 갱신하기

▪ 가중치 w_{31} 갱신하기

② $\dfrac{\partial y_{o1}}{\partial 가중합_3}$

- 활성화 함수로 시그모이드 함수 사용 : $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

- 시그모이드 함수를 미분한 결과 :

$$\frac{d\sigma(x)}{dx} = \boxed{\sigma(x)\cdot(1-\sigma(x))}$$

시그모이드(1-시그모이드)
의 형태를 띔

출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

- [참고] 시그모이드 함수 미분 증명

$$\sigma(x)' = \frac{\delta\{1+e^{-x}\}^{-1}}{\delta x} = -(1+e^{-x})^{-2}-e^{-x} = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$\sigma(x)(1-\sigma(x)) = \frac{1}{1+e^{-x}}(1-\frac{1}{1+e^{-x}}) = \frac{1}{1+e^{-x}}(\frac{e^{-x}}{1+e^{-x}}) = \frac{e^{-x}}{(1+e^{-x})^2}$$

출처 : https://t1.daumcdn.net/cfile/tistory/2229D94C55446E3128

오차 역전파로 가중치 갱신하기

- **가중치 w_{31} 갱신하기**

② $\frac{\partial y_{o1}}{\partial 가중합_3}$

- 여기서 활성화 함수$_3$의 값은 y_{o1}
- 활성화 함수$_3$의 미분 :

$$활성화\ 함수_3의\ 미분 = y_{o1} \cdot (1 - y_{o1})$$

- 이제 주어진 $\frac{\partial y_{o1}}{\partial 가중합_3}$ 식을 정리하면

$$\boxed{\frac{\partial y_{o1}}{\partial 가중합_3} = y_{o1} \cdot (1 - y_{o1}) \cdots ②}$$

출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

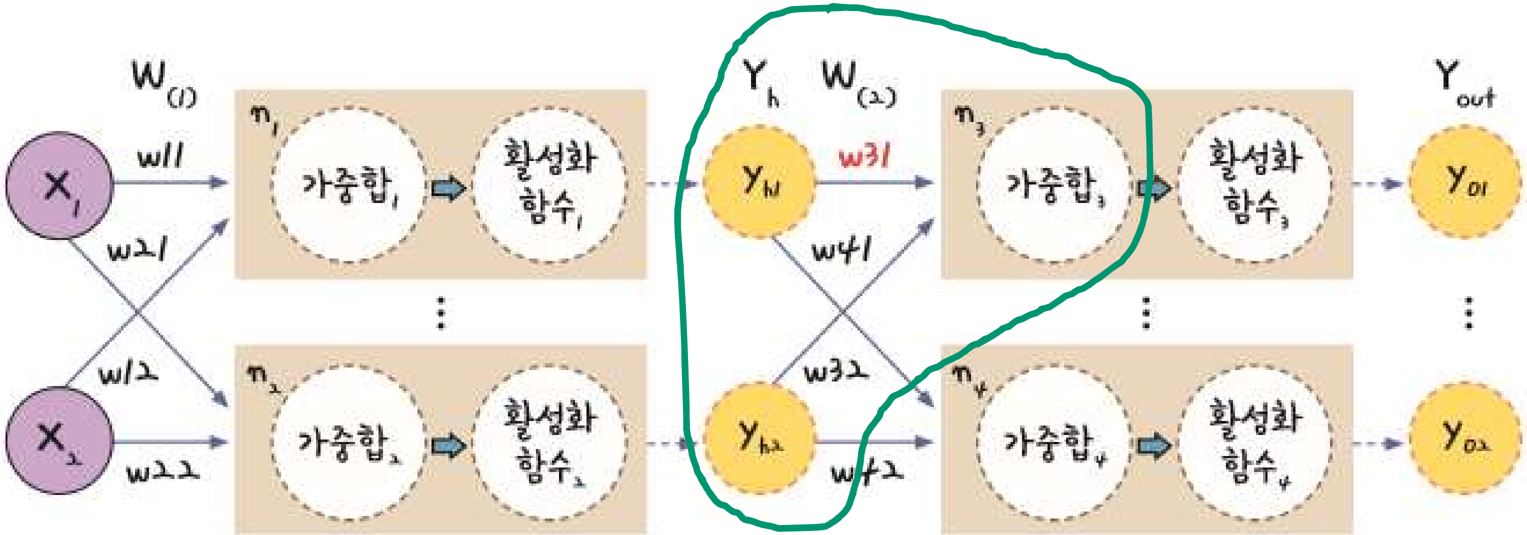
- **가중치 w_{31} 갱신하기**

③ $\dfrac{\partial 가중합_3}{\partial w_{31}}$

$$\frac{\partial 가중합_3}{\partial w_{31}} = y_{h1} \cdots ③$$

$$가중합_3 = w_{31} y_{h1} + w_{41} y_{h2} + 1\,(바이어스)$$

(바이어스는 편의상 1로 설정하는 경우가 많음)



출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

▪ 가중치 w_{31} 갱신하기

$$\frac{\partial \text{오차}Y_{out}}{\partial w_{31}} = \frac{\partial \text{오차}Y_{out}}{\partial y_{o1}} \cdot \frac{\partial y_{o1}}{\partial \text{가중합}_3} \cdot \frac{\partial \text{가중합}_3}{\partial w_{31}}$$
$$= \boxed{(y_{o1} - y_{t1}) \cdot y_{o1}(1 - y_{o1}) \cdot y_{h1}}$$

가중치 갱신 규칙을 표현한 식:

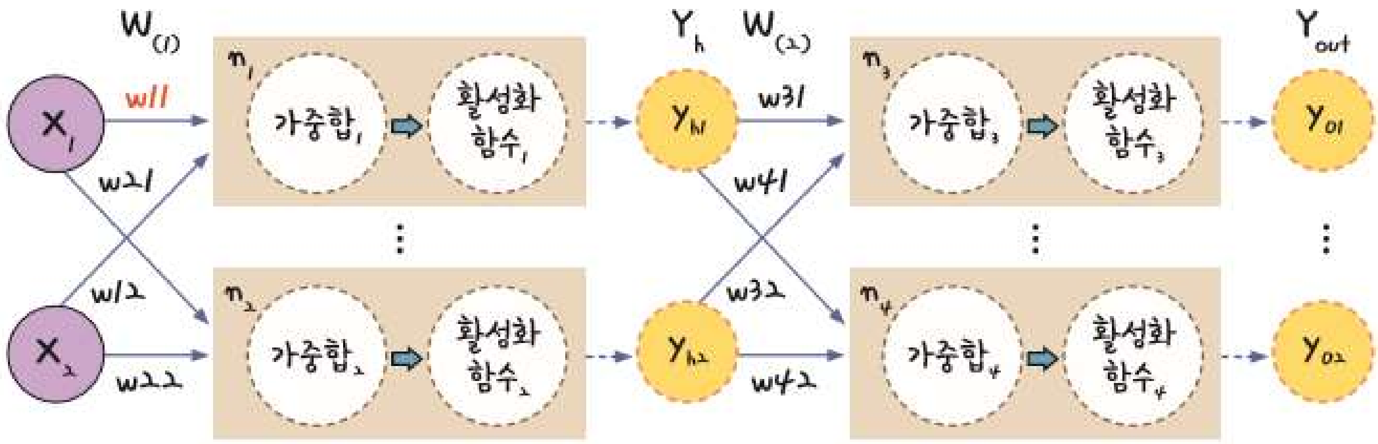
$$w_{31}(t+1) = w_{31}(t) - \boxed{(y_{o1} - y_{t1}) \cdot y_{o1}(1 - y_{o1}) \cdot y_{h1}}$$

출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

- [참고] 가중치 w_{11} 갱신하기

$$w_{11}(t+1) = w_{11}(t) - \boxed{\frac{\partial(\text{오차 } Y_{out})}{\partial w_{11}}}$$



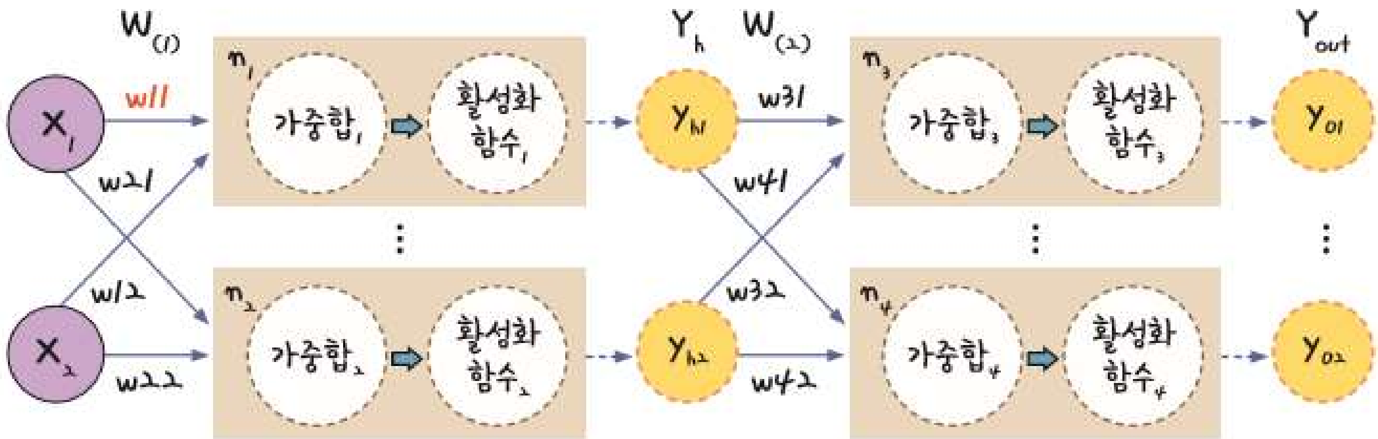
출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

- [참고] 가중치 w_{11} 갱신하기

$$w_{11}(t+1) = w_{11}(t) - \boxed{\frac{\partial(\text{오차 } Y_{out})}{\partial w_{11}}}$$

$$\frac{\partial \text{오차} Y_{out}}{\partial w_{11}} = \underbrace{\frac{\partial \text{오차} Y_{out}}{\partial y_{h1}}}_{1} \cdot \underbrace{\frac{\partial y_{h1}}{\partial \text{가중합}_1}}_{2} \cdot \underbrace{\frac{\partial \text{가중합}_1}{\partial w_{11}}}_{3}$$



출처 : 모두의 딥러닝(조태호)

오차 역전파로 가중치 갱신하기

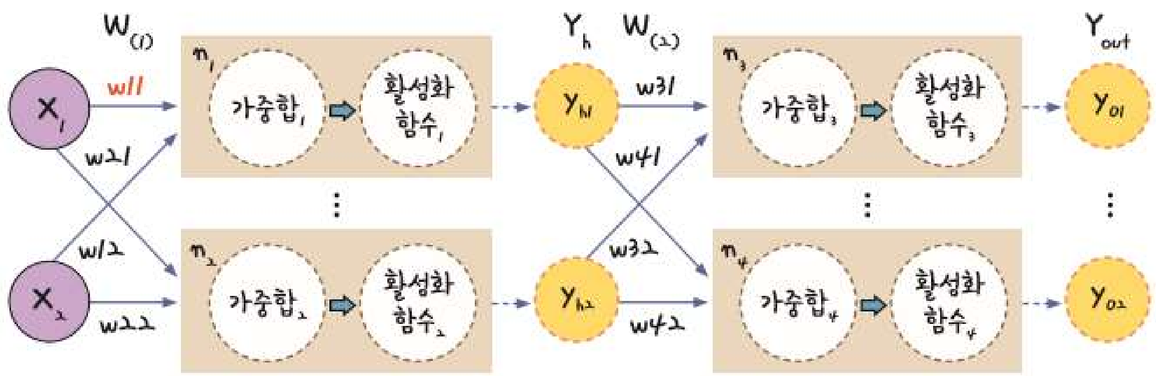
▪ [참고] 가중치 w_{11} 갱신하기

$$\frac{\partial 오차Y_{out}}{\partial w_{11}} = \underbrace{\frac{\partial 오차Y_{out}}{\partial y_{h1}}}_{1} \cdot \underbrace{\frac{\partial y_{h1}}{\partial 가중합_1}}_{2} \cdot \underbrace{\frac{\partial 가중합_1}{\partial w_{11}}}_{3}$$

$$= \left(\frac{\partial(오차\ y_{o1})}{\partial y_{h1}} + \frac{\partial(오차\ y_{o2})}{\partial y_{h1}}\right) \cdot \frac{\partial y_{h1}}{\partial 가중합_1} \cdot \frac{\partial 가중합_1}{\partial w_{11}}$$

$$= \left(\frac{\partial(오차\ y_{o1})}{\partial 가중합_3} \cdot \frac{\partial 가중합_3}{\partial y_{h1}} + \frac{\partial(오차\ y_{o2})}{\partial 가중합_4} \cdot \frac{\partial 가중합_4}{\partial y_{h1}}\right) \cdot \frac{\partial y_{h1}}{\partial 가중합_1} \cdot \frac{\partial 가중합_1}{\partial w_{11}}$$

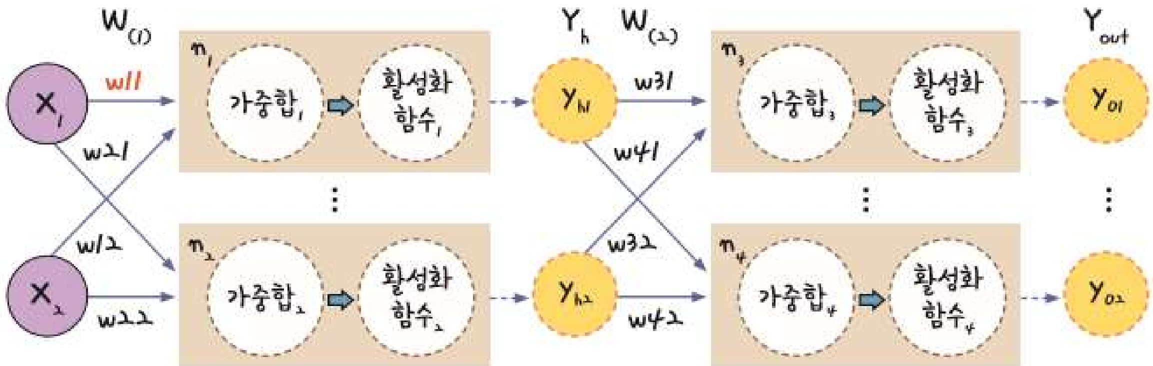
$$= \left(\frac{\partial(오차\ y_{o1})}{\partial y_{o1}} \cdot \frac{\partial y_{o1}}{\partial 가중합_3} \cdot \frac{\partial 가중합_3}{\partial y_{h1}} + \frac{\partial(오차\ y_{o2})}{\partial y_{o2}} \cdot \frac{\partial y_{o2}}{\partial 가중합_4} \cdot \frac{\partial 가중합_4}{\partial y_{h1}}\right) \cdot \frac{\partial y_{h1}}{\partial 가중합_1} \cdot \frac{\partial 가중합_1}{\partial w_{11}}$$



출처 : 모두의 딥러닝(조태호)

기울기 소실 문제

- 오차 역전파를 이용한 가중치 갱신
  - 딥러닝의 가중치들은 연쇄법칙에 의해 미분을 계속 곱하는 과정에서 이루어짐
  - 중간에 있는 미분이 0이거나 매우 작은 값이면 가중치 갱신이 더 이상 진행이 안 되거나 부정확해 짐

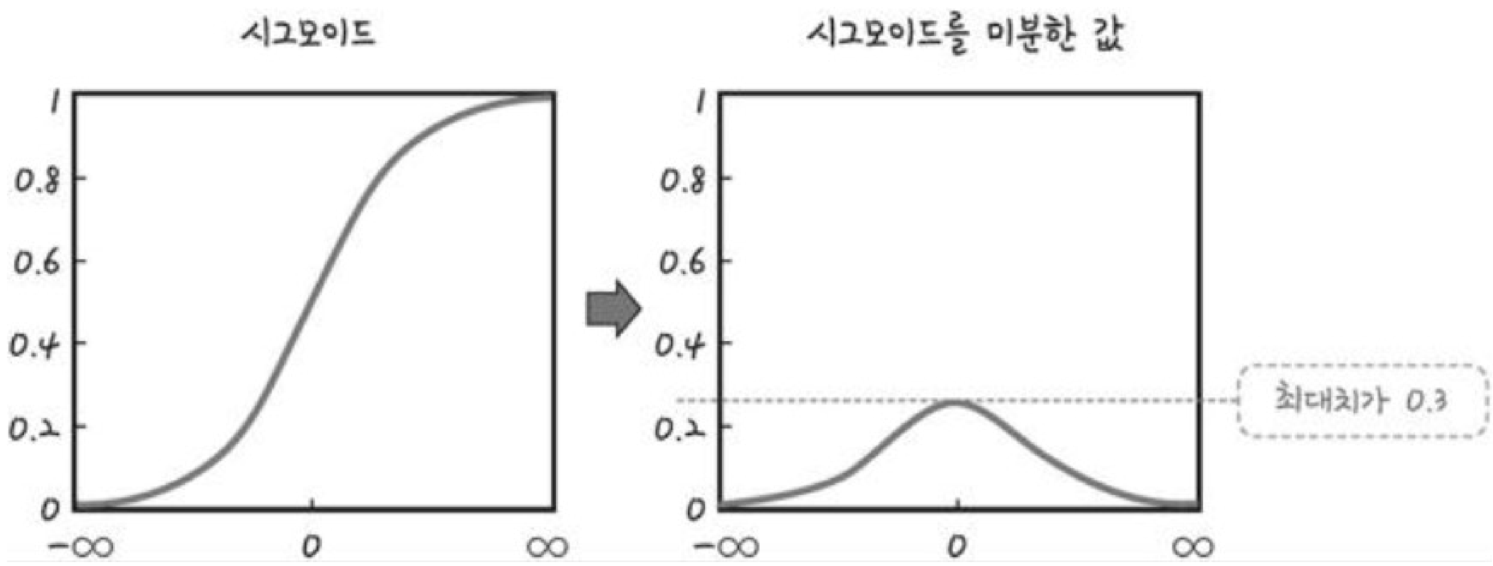


$$\left(\frac{\partial(\text{오차 } y_{o1})}{\partial y_{o1}}\cdot\frac{\partial y_{o1}}{\partial \text{가중합}_3}\cdot\frac{\partial \text{가중합}_3}{\partial y_{h1}}+\frac{\partial(\text{오차 } y_{o2})}{\partial y_{o2}}\cdot\frac{\partial y_{o2}}{\partial \text{가중합}_4}\cdot\frac{\partial \text{가중합}_4}{\partial y_{h1}}\right)\cdot\frac{\partial y_{h1}}{\partial \text{가중합}_1}\cdot\frac{\partial \text{가중합}_1}{\partial w_{11}}$$

출처 : https://ratsgo.github.io/deep%20learning/2017/09/25/gradient/

기울기 소실 문제

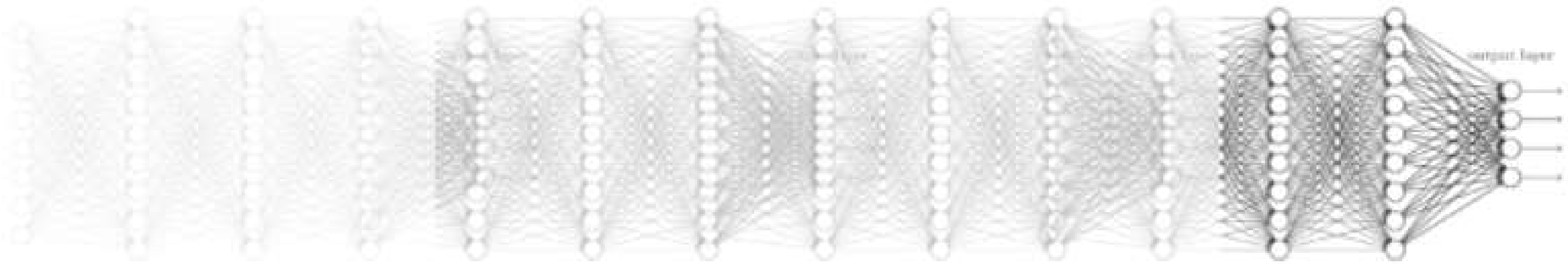
- 시그모이드 함수의 한계
  - 미분 최대치가 0.3
  - 1보다 작으므로 계속 곱하다 보면 0에 가까워짐
  - 앞의 층으로 갈수록 곱해진 미분 값이 작아져 가중치를 수정하기 어려워짐 → 기울기 소실 문제라고 함



출처 :
https://thebook.io/006958/part03/ch09/01-01/

기울기 소실 문제

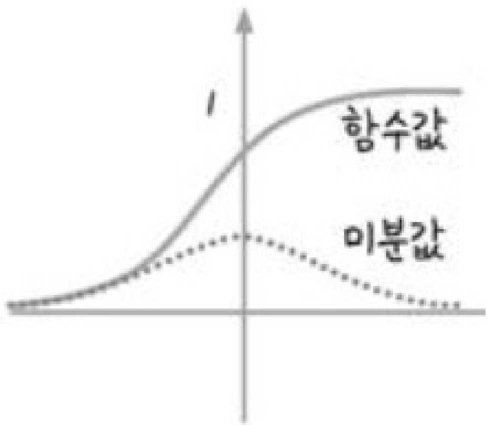
- **기울기 소실 문제 비유**



출처 :
https://ratsgo.github.io/deep%20learning/2017/09/25/gradient/

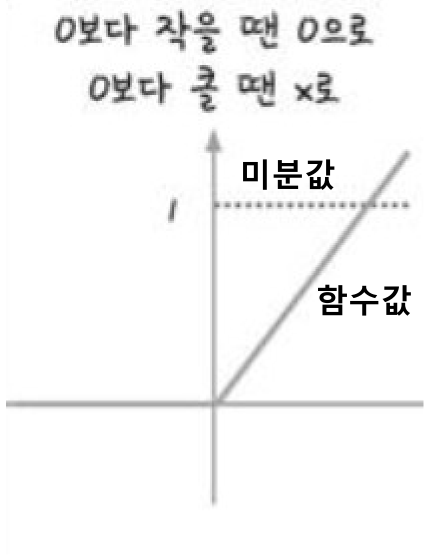
렐루 함수

- 렐루ReLU 함수
  - 입력이 0보다 클 때 미분하면 무조건 1



시그모이드 :

$$f(x) = \frac{1}{1+e^{-x}}$$



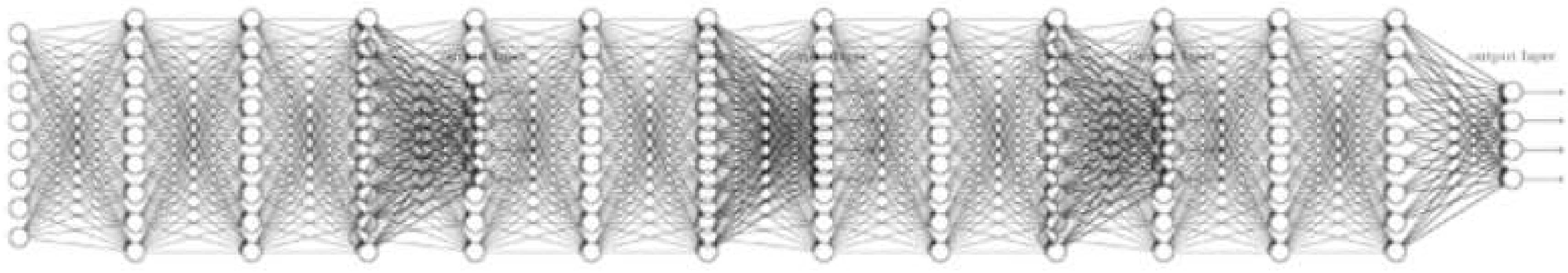
렐루 :

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

출처 : 모두의 딥러닝(조태호)

렐루 함수

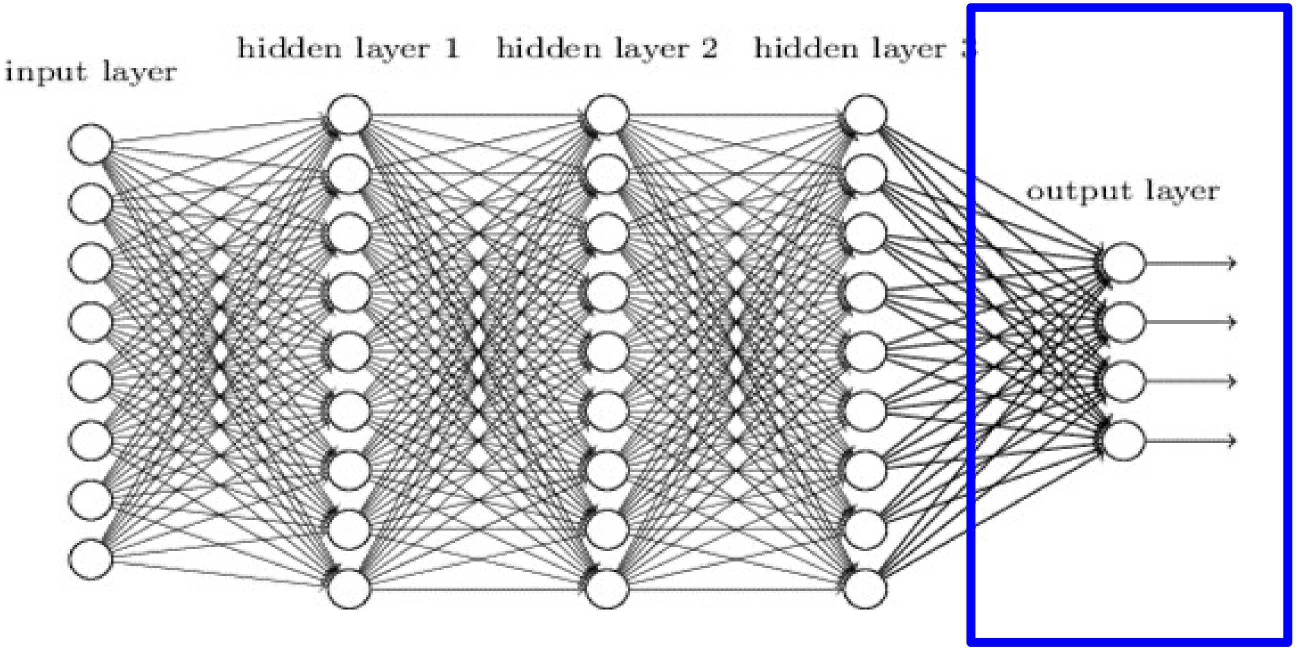
- 렐루 함수로 기울기 소실 문제 극복



출처 : https://ratsgo.github.io/deep%20learning/2017/09/25/gradient/

다중 분류를 위한 딥러닝 구조

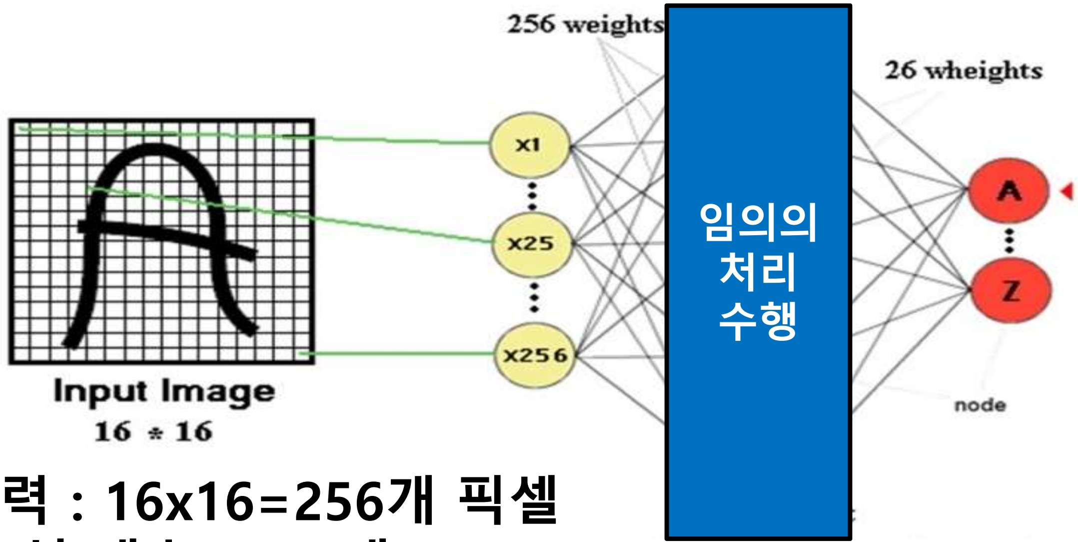
- 출력층 노드 개수 설정
  - 회귀 : 1개
  - 이진 분류(클래스 2개) : 1개
  - 다중 분류(클래스 3개 이상) : 클래스가 N개이면 노드도 N개



출처 : https://ratsgo.github.io/deep%20learning/2017/09/25/gradient/

다중 분류를 위한 딥러닝 구조

- 예시



입력 : 16x16=256개 픽셀
(속성 개수 : 256개)

출력 : 26개(=알파벳 개수)의
확률값(0~1)

A : 0.9
B : 0.001
C : 0.002
D : 0.002
…
Z : 0.001

활성화 함수 : 소프트맥스

- 소프트맥스Softmax
  - 다중 분류의 경우 출력층에 사용하는 활성화 함수
  - 이진 분류의 경우 출력층에 시그모이드 함수를 많이 사용

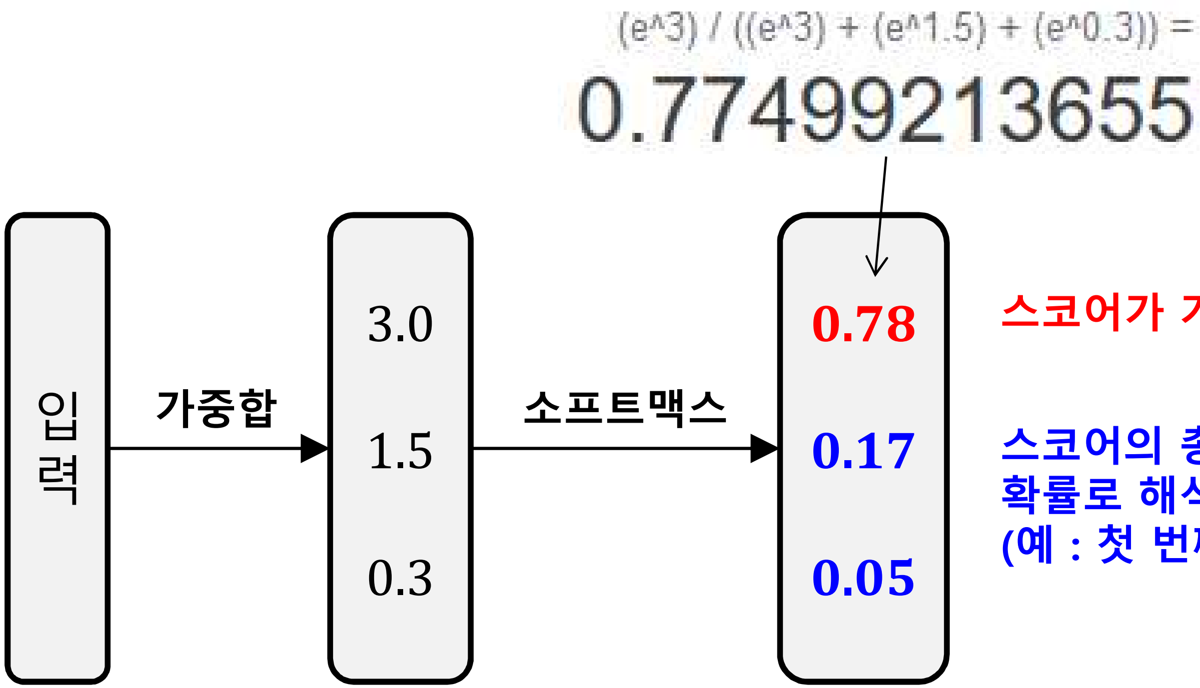
가중합의 출력값

$$\boxed{\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}}$$

출처 : https://thebook.io/img/006958/151.jpg

활성화 함수 : 소프트맥스

- 소프트맥스의 특성
  - 소프트맥스를 거친 활성화 함수 값의 총합이 항상 1
  - 활성화 함수 적용 전과 후의 대소 관계가 바뀌지 않음



스코어가 가장 높은 클래스로 예측

스코어의 총 합이 1이므로
확률로 해석할 수 있음
(예 : 첫 번째 클래스일 확률이 78%다!)

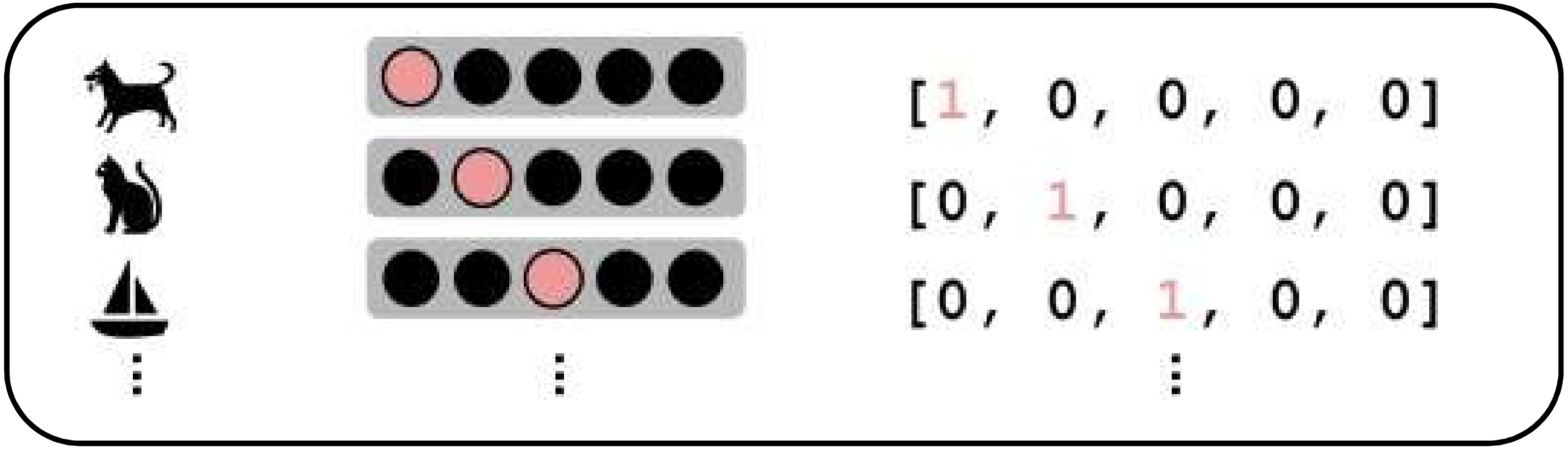
0.78+0.17+0.05=1

KOREATECH 한국기술교육대학교

출처 : https://thebook.io/img/006958/151.jpg

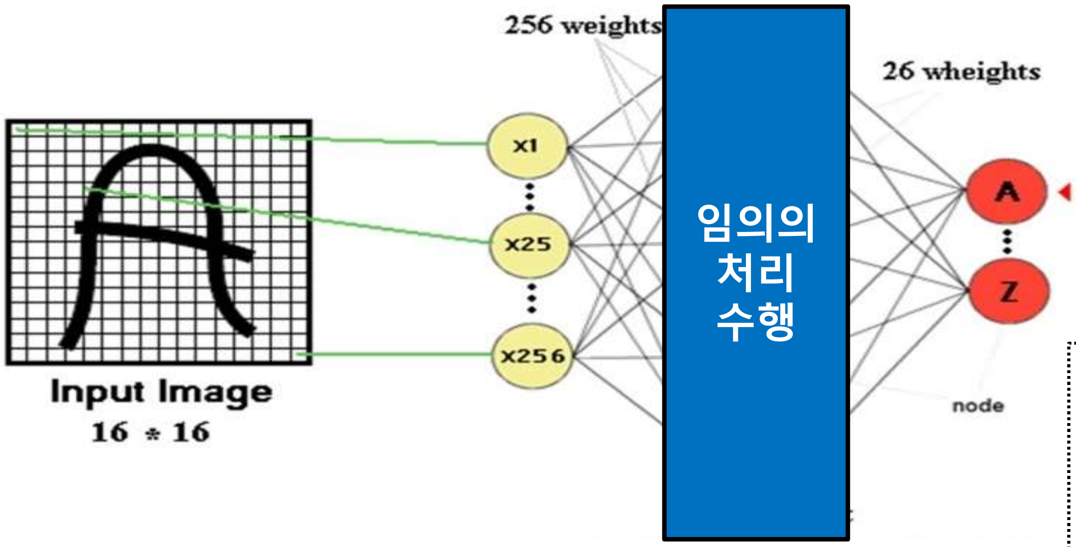
원 핫 인코딩

- 원 핫 인코딩One-hot Encoding
  - 한 개의 요소만 1이고 나머지는 0으로 표현하는 방식



원 핫 인코딩

- **원 핫 인코딩**One-hot Encoding
  - 딥러닝 학습 시 정답을 표현할 때 사용



오차함수

A : 0.9
B : 0.001
C : 0.002
D : 0.002
…
Z : 0.001

예측값

A : 1
B : 0
C : 0
D : 0
…
Z : 0

정답(원 핫 인코딩)

아이리스 품종 예측

▪ 아이리스 품종 예측

- 아이리스는 꽃잎의 모양과 길이에 따라 여러 품종으로 나누어짐

- 목표 : 딥러닝으로 세 가지의 아이리스 종류를 구분하기

Iris-virginica

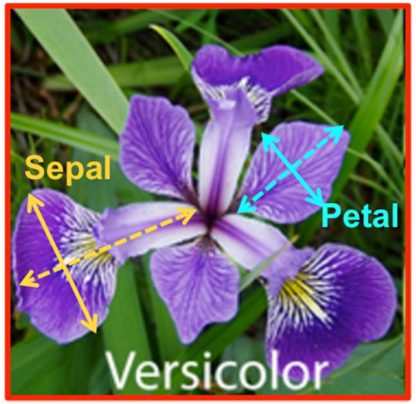
Iris-setosa

Iris-versicolor

출처 : 모두의 딥러닝(조태호)

아이리스 품종 예측

▪ 데이터 들여다보기



- 샘플 수: 150
- 속성 수: 4
  - 정보 1: 꽃받침 길이 (sepal length, 단위: cm)
  - 정보 2: 꽃받침 넓이 (sepal width, 단위: cm)
  - 정보 3: 꽃잎 길이 (petal length, 단위: cm)
  - 정보 4: 꽃잎 넓이 (petal width, 단위: cm)
- 클래스: Iris-setosa, Iris-versicolor, Iris-virginica

속성 / 클래스 / 샘플

	정보 1	정보 2	정보 3	정보 4	품종
1번째 아이리스	5.1	3.5	4.0	0.2	Iris-setosa
2번째 아이리스	4.9	3.0	1.4	0.2	Iris-setosa
3번째 아이리스	4.7	3.2	1.3	0.3	Iris-setosa
...	...	...	...	...	...
150번째 아이리스	5.9	3.0	5.1	1.8	Iris-virginica

출처 : 모두의 딥러닝(조태호)

아이리스 품종 예측

- Iris-setosa, Iris-virginica 등 데이터 안에 문자열이 포함되어 있음
- 이럴 땐 numpy 보다는 panda로 데이터를 불러와 X와 Y의 값을 구분하는 것이 편리함

```
# 필요한 라이브러리
from keras.models import Sequential
from keras.layers.core import Dense
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# 데이터 불러오기
df = pd.read_csv('../dataset/iris.csv', names = ["sepal_length",
"sepal_width", "petal_length", "petal_width", "species"])
```

아이리스 품종 예측

- 속성과 클래스를 각각 변수 X와 Y_obj에 담기

속성 / 클래스 / 샘플

	정보 1	정보 2	정보 3	정보 4	품종
1번째 아이리스	5.1	3.5	4.0	0.2	Iris-setosa
2번째 아이리스	4.9	3.0	1.4	0.2	Iris-setosa
3번째 아이리스	4.7	3.2	1.3	0.3	Iris-setosa
...	...	...	...	...	...
150번째 아이리스	5.9	3.0	5.1	1.8	Iris-virginica

```
dataset = df.values
X = dataset[:,0:4] # 속성
Y_obj = dataset[:,4] # 클래스
```

아이리스 품종 예측

- Y_obj는 숫자가 아니라 문자열
- 문자열을 숫자로 바꿔주는 작업 수행 필요
  - ✓ 이를 가능하게 하는 함수가 sklearn(싸이킷런) 라이브러리의 LabelEncoder()

```
e = LabelEncoder()
e.fit(Y_obj)
Y = e.transform(Y_obj)
```

- 이렇게 하면 ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']가 [1, 2, 3]으로 바뀜

아이리스 품종 예측

- 원 핫 인코딩을 적용하여 출력층의 각 노드가 0 또는 1의 정답을 가질 수 있도록 함

```
Y_encoded = np_utils.to_categorical(Y)
```

- 이렇게 하면 [1,2,3]가 다시 [[1, 0, 0], [0, 1, 0],[0, 0, 1]]로 바뀜
- 이처럼 여러 개의 Y 값을 0과 1로만 이루어진 형태로 바꿔 주는 기법을 원-핫 인코딩(one-hot-encoding)이라고 함

아이리스 품종 예측

- **딥러닝 모델 설계**

```
model = Sequential()
model.add(Dense(16, input_dim=4, activation='relu')) # 입력층과 은닉층
model.add(Dense(3, activation='softmax')) # 출력층
```

- 지난 주 코드와 달라진 점!
  - → 최종 출력 값이 3개 중 하나여야 하므로 출력층의 노드 수를 3으로 설정
  - → 또한 활성화 함수로 소프트맥스를 사용

출처 : 모두의 딥러닝(조태호)

아이리스 품종 예측

- 딥러닝 모델 추가 설정 및 학습 실행

```
# 모델 컴파일
model.compile(loss='mean_squared_error',
              optimizer='sgd',
              metrics=['accuracy'])

# 모델 학습 실행
model.fit(X, Y_encoded, epochs=50, batch_size=1)

# 결과 출력
print("\n Accuracy: %.4f" % (model.evaluate(X, Y_encoded)[1]))
```


출처 : 모두의 딥러닝(조태호)