06장 2개의 층을 연결합니다 — 다층 신경망

06-1 신경망 알고리즘을 벡터화하여 한 번에 전체 샘플을 사용합니다

스칼라: scalar

스칼라는 하나의 숫자만으로 이루어진 데이터를 의미합니다. 스칼라는 보통 **x** 와 같이 알파벳 소문자로 표기하며 실수(real number)인 숫자 중의 하나이므로 실수 집합 "**R**"의 원소라는 의미에서 다음과 같이 표기한다.

$$x \in \mathbb{R}$$

벡터: vector

벡터는 여러 숫자가 순서대로 모여 있는 것으로, 일반적인 일차원 배열이 벡터입니다.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

하나의 벡터를 이루는 데이터의 개수를 차원(dimension)이라고 합니다. 위에서 예로든 벡터는 4개의 실수로 이루어져 있고, 4차원 벡터입니다. 이 벡터는 다음과 같이 표기할 수 있습니다.

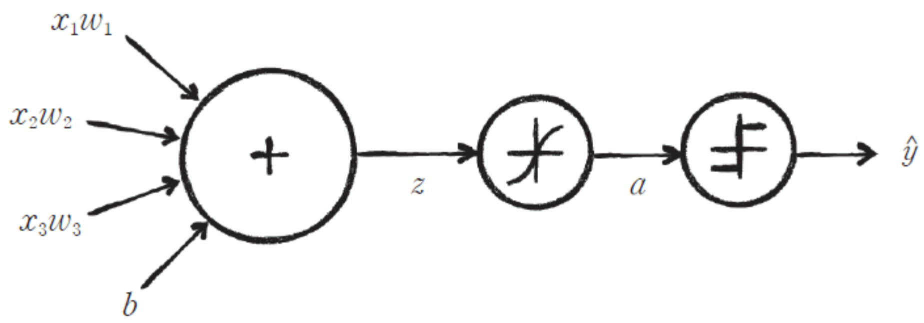
$$x \in \mathbb{R}^4$$

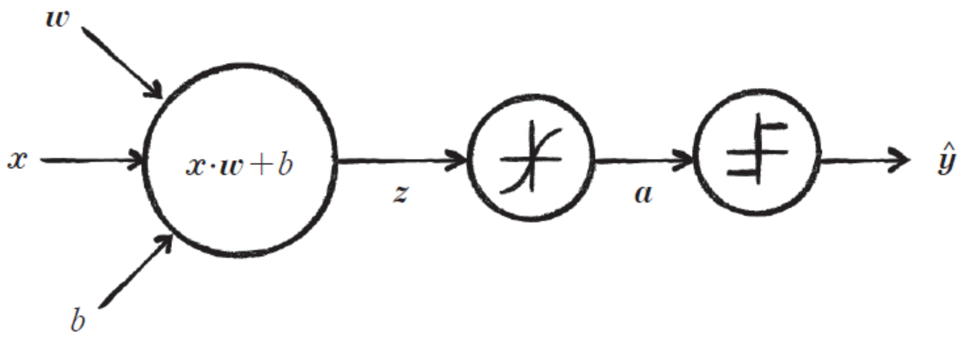
행렬: matrix

행렬은 복수의 차원을 가지는 데이터가 다시 여러 개 있는 경우의 데이터를 합쳐서 표기한 것이다. 일반적으로 2차원 배열이 행렬입니다. 특히 3차원 이상 배열은 텐서(tensor)라고 합니다.

06-1 신경망 알고리즘을 벡터화하여 한 번에 전체 샘플을 사용합니다

점 곱(스칼라 곱)을 도입하면 단일층 신경망의 연산을 간결하게 표현할 수 있음





06-1 신경망 알고리즘을 벡터화하여 한 번에 전체 샘플을 사용합니다

점 곱(스칼라 곱)을 행렬 곱셈으로 표현(샘플 1개)

$$XW = [x_1\ x_2\ x_3]\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = w_1 \times x_1 + w_2 \times x_2 + w_3 \times x_3$$

```
z = np.dot(x, self.w) + self.b
```

06-1 신경망 알고리즘을 벡터화하여 한 번에 전체 샘플을 사용합니다

점 곱(스칼라 곱)을 행렬 곱셈으로 표현(샘플 m개)

m개의 샘플

점 곱

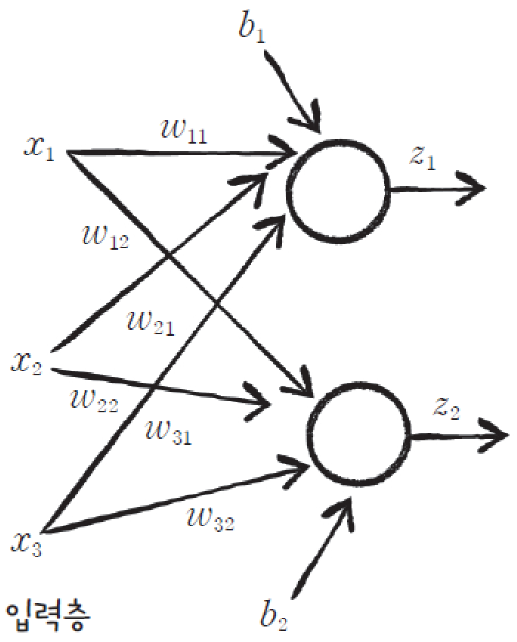
$$XW = \begin{bmatrix} \overbrace{x_1^{(1)} \quad x_2^{(1)} \quad x_3^{(1)}}^{\text{3개의 특성}} \\ x_1^{(2)} \quad x_2^{(2)} \quad x_3^{(2)} \\ \vdots \\ x_1^{(m)} \quad x_2^{(m)} \quad x_3^{(m)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} x_1^{(1)}w_1 + x_2^{(1)}w_2 + x_3^{(1)}w_3 \\ x_1^{(2)}w_1 + x_2^{(2)}w_2 + x_3^{(2)}w_3 \\ \vdots \\ x_1^{(m)}w_1 + x_2^{(m)}w_2 + x_3^{(m)}w_3 \end{bmatrix}$$

```
np.dot(x, w)
```

06-2 2개의 층을 가진 신경망을 구현합니다

★ 층에 있는 뉴런의 개수를 늘려보고, 층의 수도 늘려 본다 ★

하나의 층에 여러 개의 뉴런을 사용하면 한 번에 여러 특성을 뉴런에 전달할 수 있음



06-2 2개의 층을 가진 신경망을 구현합니다

★ 뉴런이 2개이므로 출력도 2개가 된다 (z1, z2) ★

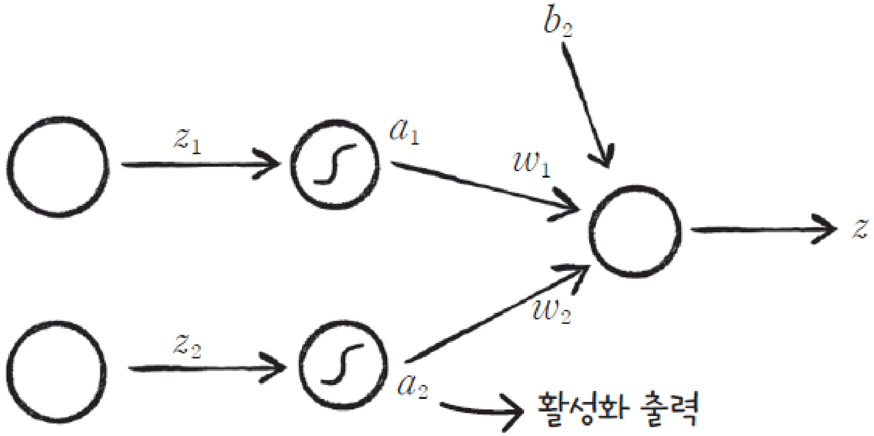
여러 개의 뉴런을 추가한 신경망을 행렬 곱셈으로 표현

$$[x_1\ x_2\ x_3]\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}+[b_1\ b_2]=[z_1\ z_2] \qquad \boldsymbol{XW_1+b_1=Z_1}$$

06-2 2개의 층을 가진 신경망을 구현합니다

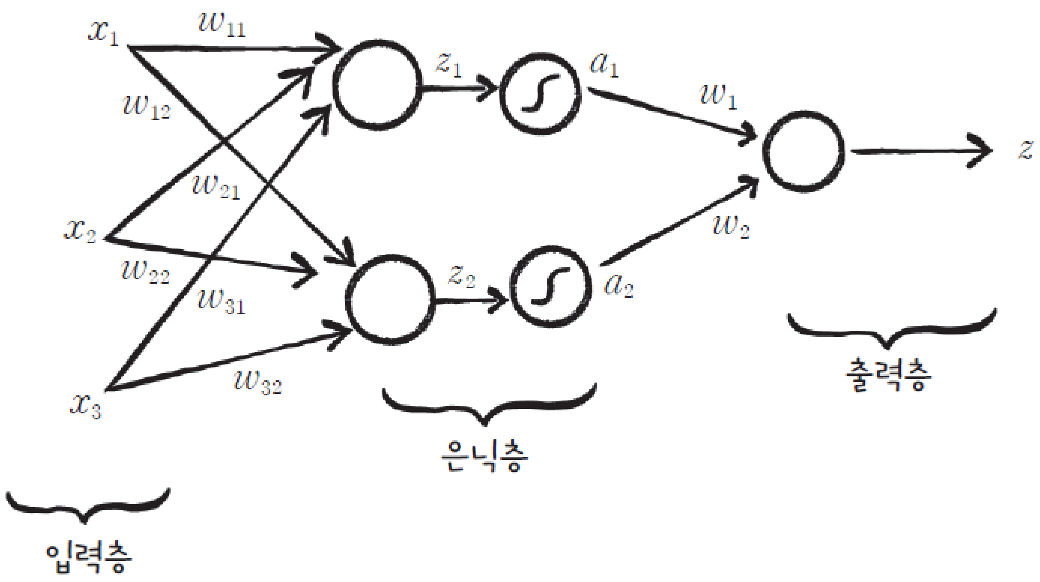
각 뉴런에서 나온 출력을 모아야 이진 분류에 사용할 수 있음

이때, 출력을 모은 값을 z라고 함



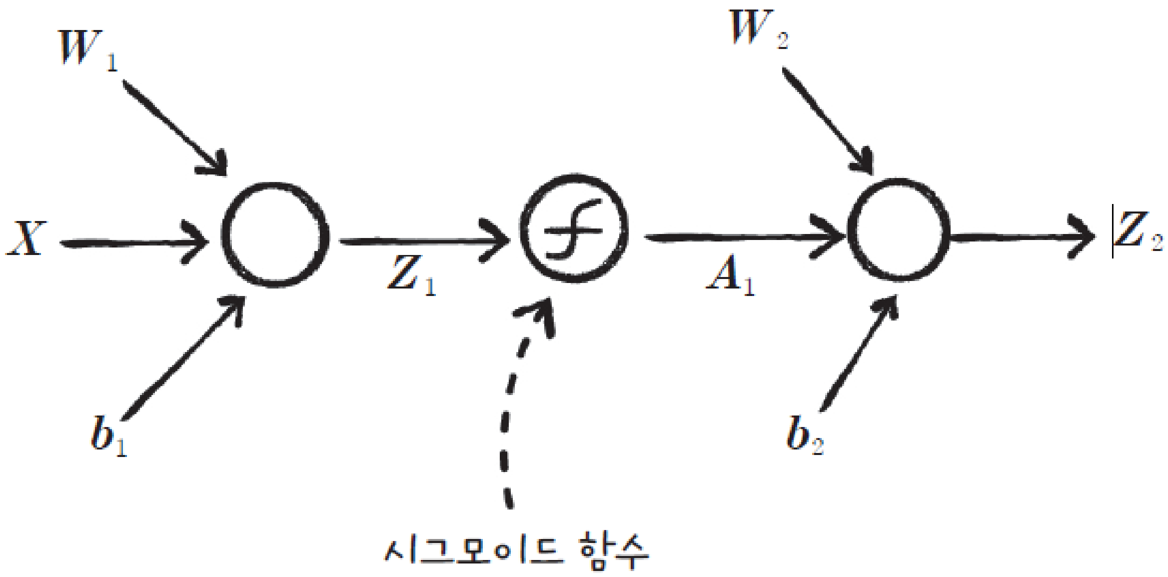
06-2 2개의 층을 가진 신경망을 구현합니다

입력층, 은닉층, 출력층이라는 용어를 도입



06-2 2개의 층을 가진 신경망을 구현합니다

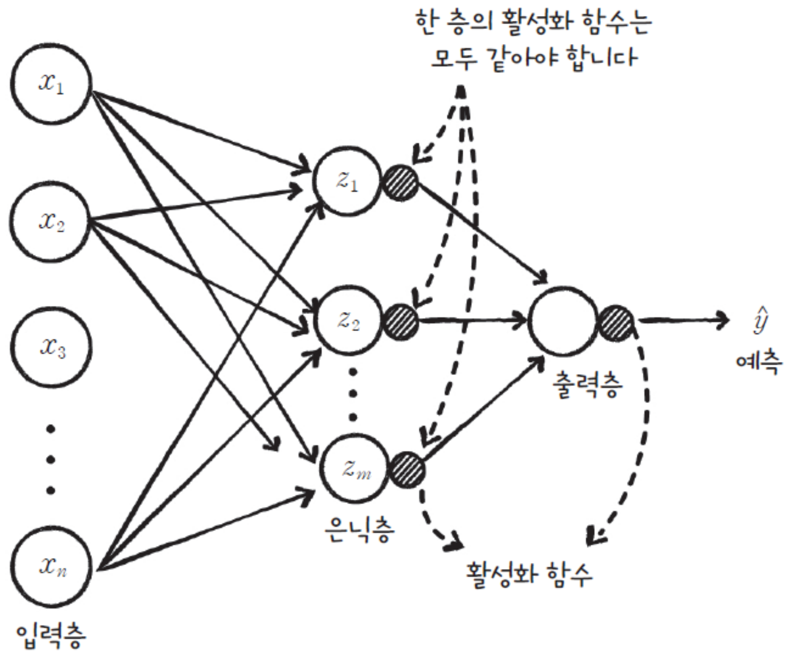
각 층의 입력값들을 행렬로 표기



06-2 2개의 층을 가진 신경망을 구현합니다

다층 신경망으로 확장

한 층의 활성화 함수는 모두 같아야 함에 주의

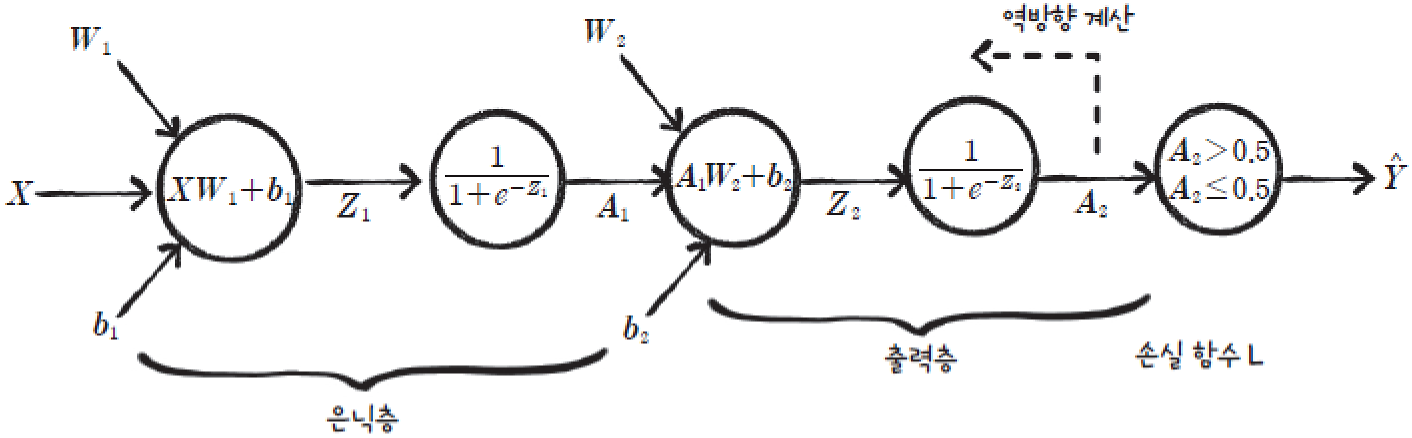


모든 뉴런이 연결되어 있으므로
완전 연결 신경망이라고 부르기도 함
(fully connected neural network)

06-2 2개의 층을 가진 신경망을 구현합니다

다층 신경망에 경사 하강법 적용

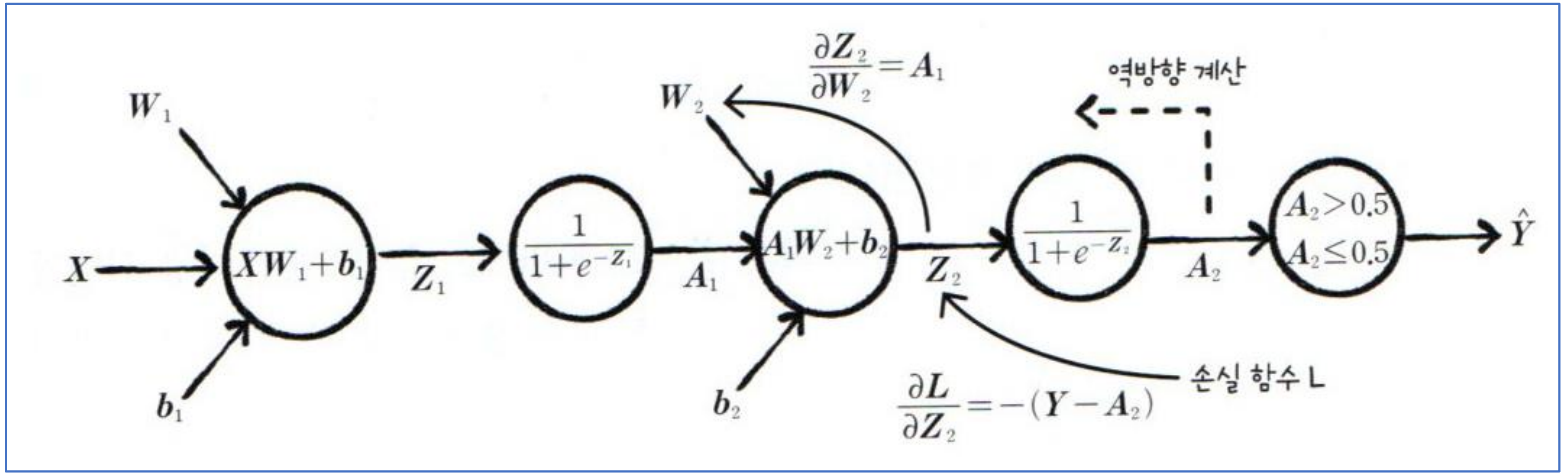
오른쪽 출력층에서 왼쪽 은닉층 방향으로 미분해야 함



06-2 2개의 층을 가진 신경망을 구현합니다

[출력층의 그레이디언트 구하기]

W2에 대한 손실함수의 미분

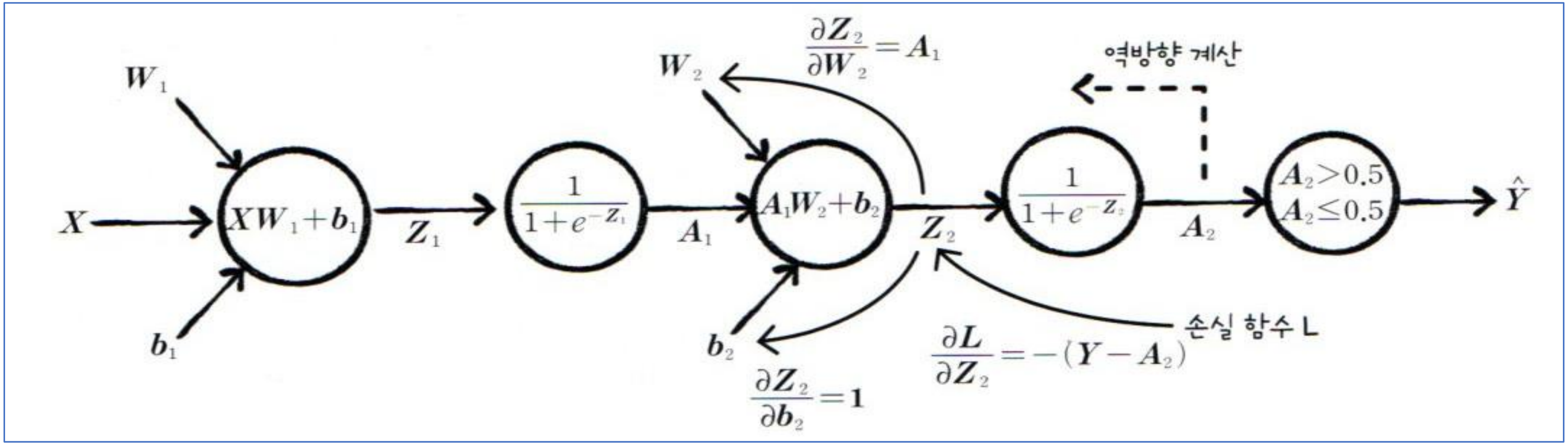


그레이디언트의 총 합

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial Z_2} \bullet \frac{\partial Z_2}{\partial W_2} = A_1^T(-(Y-A_2)) = \underbrace{\begin{bmatrix} -1.37 & \cdots & 2.10 \\ 0.96 & & -0.17 \end{bmatrix}}_{m\text{개}} \begin{bmatrix} 0.7 \\ 0.3 \\ \vdots \\ 0.6 \end{bmatrix} = \begin{bmatrix} -0.12 \\ 0.36 \end{bmatrix}$$

타깃과 예측의 차이

06-2 2개의 층을 가진 신경망을 구현합니다



[출력층의 그레디언트 구하기]

절편에 대한 손실함수의 미분

$$\frac{\partial L}{\partial b_2}=\frac{\partial L}{\partial Z_2}\frac{\partial Z_2}{\partial b_2}=\mathbf{1}^T(-(Y-A_2))=\underbrace{[1 \cdots 1]}_{m\text{개}}\begin{bmatrix}0.7\\0.3\\\vdots\\0.6\end{bmatrix}=0.18$$

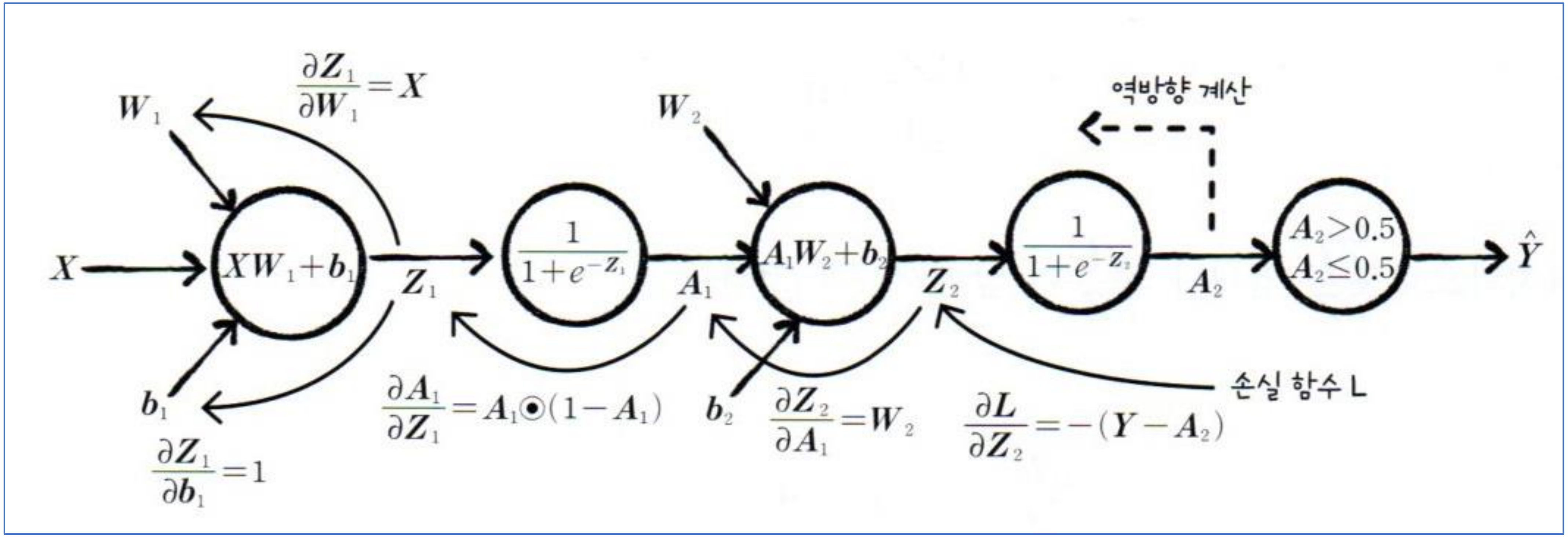
타깃과 예측의 차이

06-2 2개의 층을 가진 신경망을 구현합니다

같은 위치 원소끼리 곱합니다.

$$\begin{bmatrix} 1 & 6 \\ 4 & 2 \end{bmatrix} \odot \begin{bmatrix} 12 & 2 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 12 & 12 \\ 12 & 12 \end{bmatrix}$$

[은닉층의 그레디언트 구하기] W1에 대한 손실함수의 미분



err_to_hidden

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial W_1} = X^T \boxed{(-(Y - A_2)W_2^T \odot A_1 \odot (1 - A_1))}$$

※ 시그모이드 함수의 도함수 = a(1-a)

06-2 2개의 층을 가진 신경망을 구현합니다

[은닉층의 그레디언트 구하기] 절편에 대한 손실함수의 미분

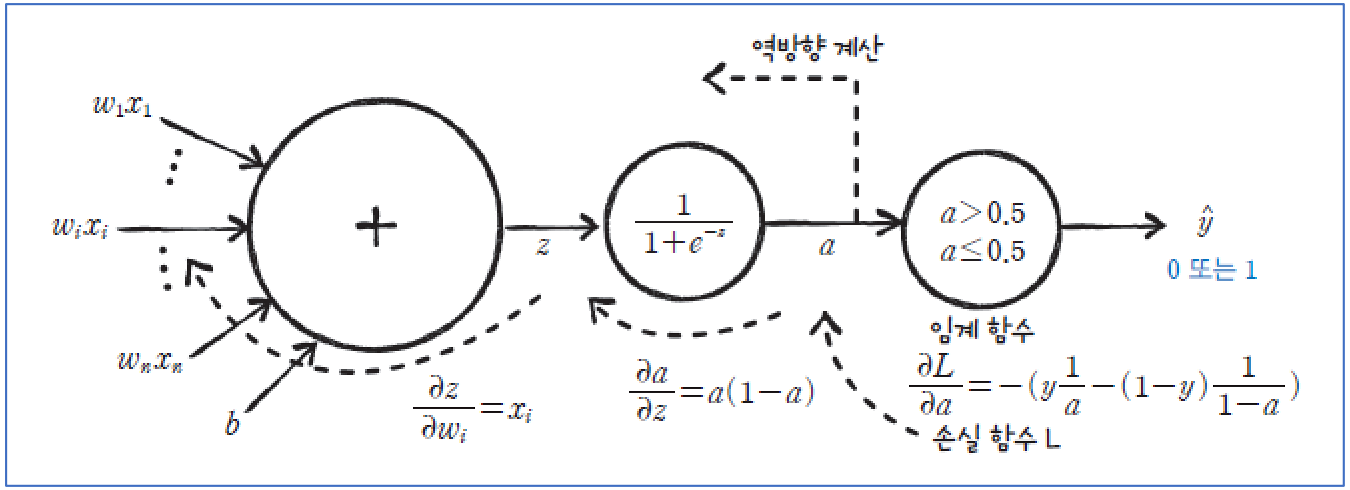
$$\frac{\partial L}{\partial \boldsymbol{b}_1}=\frac{\partial L}{\partial \boldsymbol{Z}_2}\frac{\partial \boldsymbol{Z}_2}{\partial \boldsymbol{A}_1}\frac{\partial \boldsymbol{A}_1}{\partial \boldsymbol{Z}_1}\frac{\partial \boldsymbol{Z}_1}{\partial \boldsymbol{b}_1}=\mathbf{1}^T\overbrace{\boxed{(-(\boldsymbol{Y}-\boldsymbol{A}_2)\boldsymbol{W}_2^T\odot\boldsymbol{A}_1\odot(1-\boldsymbol{A}_1))}}^{\text{err\_to\_hidden}}$$

$$=[1\ \ 1\ \cdots\ 1]\left.\begin{bmatrix}-0.045 & -3.48\\ & \vdots & \\ -0.018 & -1.768\end{bmatrix}\right\}m\text{개}$$

$$=[0.121\ \ -0.034]$$

06-2 2개의 층을 가진 신경망을 구현합니다 [고찰]

로지스틱 손실 함수의 미분 과정 정리하고 역전파 이해하기



<로지스틱 손실함수와 연쇄 법칙>

* 연쇄법칙을 도입하는 이유

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\{-(y\log a + (1-y)\log(1-a))\}$$

그런데 $a = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(wx+b)}}$

따라서

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\{-(y\log\frac{1}{1+e^{-(wx+b)}} + (1-y)\log(1-\frac{1}{1+e^{-(wx+b)}}))\}$$

⇒ 미분이 너무 복잡하다. 따라서 chain rule 이용함.

* Chain Rule 수식적용

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a}\cdot\frac{\partial a}{\partial z}\cdot\frac{\partial z}{\partial w_i}$$

06-2 2개의 층을 가진 신경망을 구현합니다 [고찰]

로지스틱 손실 함수 미분하기

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_i}$$

① ② ③

① 항 풀기

$$\frac{\partial L}{\partial a} = \frac{\partial}{\partial a}\{-(y\log a + (1-y)\log(1-a))\}$$

$$= -\{y \frac{\partial}{\partial a}\log a + (1-y)\frac{\partial}{\partial a}\log(1-a)\}$$

$\therefore \log a \rightarrow \frac{1}{a}$

$$\frac{\partial L}{\partial a} = -y \cdot \frac{1}{a} - (1-y) \cdot \frac{1}{1-a} \cdot (1-a)'$$

$(1-a)' \hookrightarrow -1$

$$= \boxed{-\frac{y}{a} + (1-y) \cdot \frac{1}{(1-a)}}$$

② 항 풀기

$$\frac{\partial a}{\partial z} = \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) = \frac{\partial}{\partial z}(1+e^{-z})^{-1}$$

$\therefore e^{-z} \rightarrow -e^{-z}$

$$\frac{\partial a}{\partial z} = -(1+e^{-z})^{-2} \cdot (1+e^{-z})'$$

$$= -(1+e^{-z})^{-2} \cdot (-e^{-z})$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{1+e^{-z}-1}{1+e^{-z}} = \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= \boxed{a(1-a)}$$

③ 항 풀기

$$\frac{\partial z}{\partial w_i} = \frac{\partial}{\partial w_i}(w_i x_i + b) = x_i$$

즉, $\frac{\partial L}{\partial w_i} = ① \times ② \times ③$

$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) a(1-a) \cdot x_i$$

$$= \{-y(1-a) + a(1-y)\} x_i$$

$$= \{-y + \cancel{ay} + a - \cancel{ay}\} x_i$$

$$= \boxed{-(y-a) x_i}$$

이번에는 절편에 대한 손실함수 변화율을 구한다.

$$\frac{\partial L}{\partial b} = \boxed{\frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z}} \cdot \frac{\partial z}{\partial b}$$

① ②

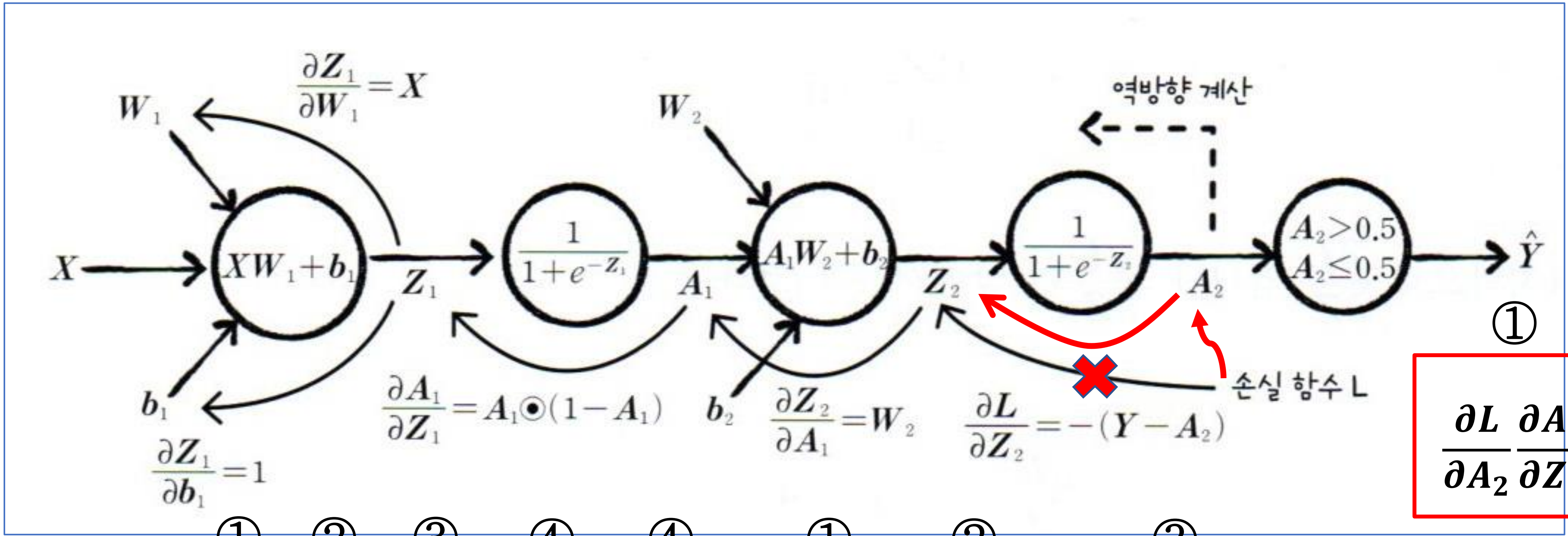
① 항 $\frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} = -(y-a)$ ← 위에서 이미 구했음

② 항 $\frac{\partial z}{\partial b} = \frac{\partial}{\partial b}(wx+b) = 1$

$$\Rightarrow \boxed{\frac{\partial L}{\partial b} = -(y-a) \cdot 1}$$

06-2 2개의 층을 가진 신경망을 구현합니다 [고찰]

[은닉층의 그레디언트 구하기] W1에 대한 손실함수의 미분



①

$$\boxed{\frac{\partial L}{\partial A_2}\frac{\partial A_2}{\partial Z_2}}$$

$$\frac{\partial L}{\partial W_1} = \overset{①}{\boxed{\cancel{\frac{\partial L}{\partial Z_2}}}}\ \overset{②}{\boxed{\frac{\partial Z_2}{\partial A_1}}}\ \overset{③}{\boxed{\frac{\partial A_1}{\partial Z_1}}}\ \overset{④}{\boxed{\frac{\partial Z_1}{\partial W_1}}} = \overset{④}{\boxed{X^T}}\overset{①}{\boxed{(-(Y-A_2)}}\overset{②}{\boxed{W_2^T}}\odot\overset{③}{\boxed{A_1\odot(1-A_1)}})$$

※ 시그모이드 함수의 도함수 = a(1-a)

06-2 2개의 층을 가진 신경망을 구현합니다

DualLayer 클래스 훈련시키기

[데이터]

특성 30개

[SingleLayer]

(단층 뉴런 1개) 가중치 30개 + 절편 1개 = 31개

[DualLayer]

(은닉층 뉴런 10개) 가중치 30*10개 + 절편 10개

(출력층 뉴런 01개) 가중치 10개 + 절편 1개

06-3 미니 배치를 사용하여 모델을 훈련합니다

[특징]

- 배치 경사하강법과 비슷함.

[정의]

- 배치 경사하강법처럼 에포크마다 전체 데이터를 사용하는 것이 아님.
- 조금씩 나누어서 정방향계산 ➔ 그레이디언트계산 ➔ 가중치업데이트를 수행함.

[고려사항]

- 미니배치의 크기 : 보통 16, 32, 64 등 2의 배수를 사용함.
- 만약 미니배치의 크기가,

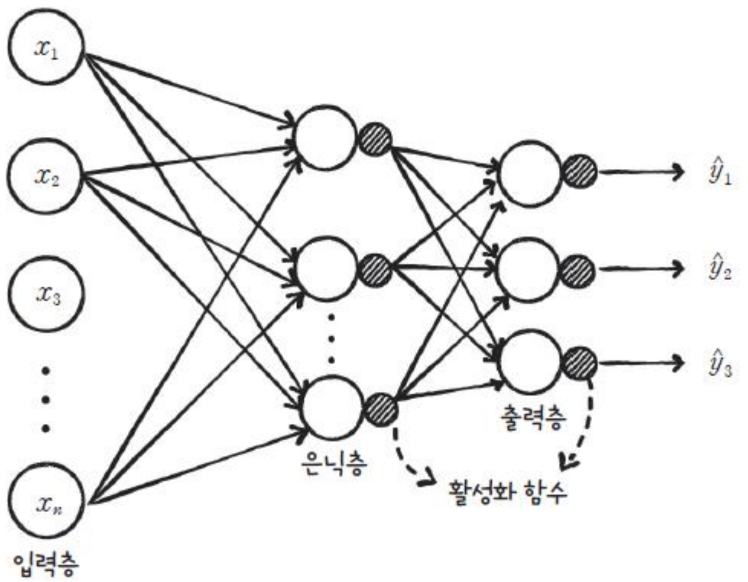
  1이면 ➔ 확률적 경사하강법

  전체데이터 크기 ➔ 배치 경사하강법

07장 여러 개를 분류합니다 — 다중 분류

07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

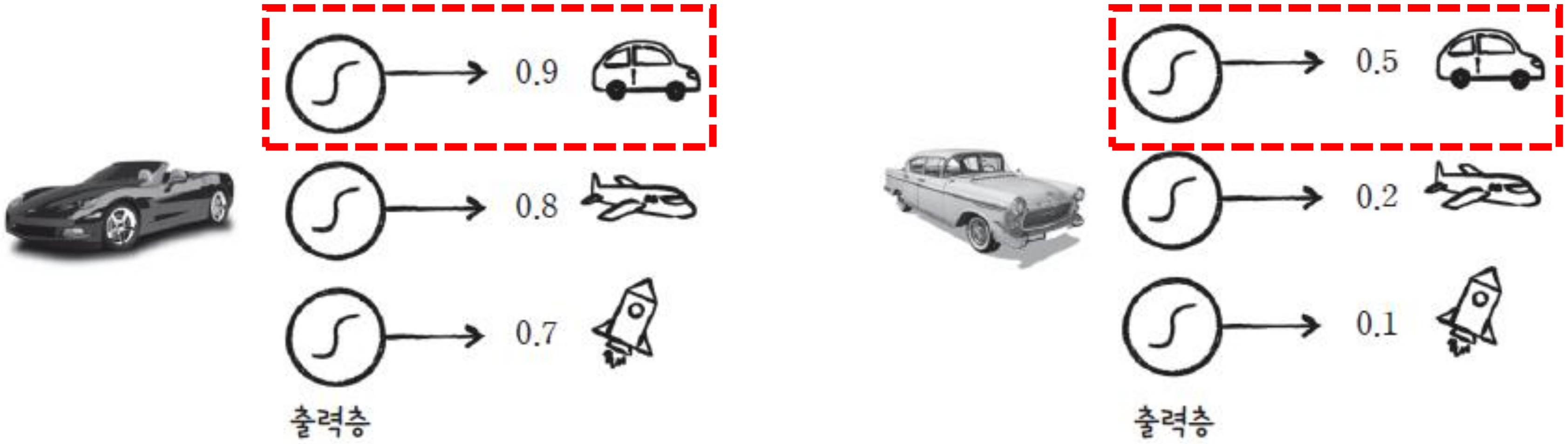
다중 분류 신경망의 구조



07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

소프트맥스 함수를 도입해야 하는 이유

출력의 합(y_1 + y_2 + y_3)을 1로 만들기 위해(확률로 생각하기 위해)



왼쪽과 오른쪽 모델 중 어떤 결과가 더 정확하게 이미지를 분류했는지 알기 어렵다

산술적으로 계산하면, (왼편 0.9) 37%   (오른쪽 0.5) 62%

➔ 활성화 출력의 합이 1이 아니면 비교하기 어렵다

07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

소프트맥스 함수의 정의

$$\frac{e^{z_i}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

z에 대해 정리하면 $z=-ln(\frac{1}{\hat{y}}-1)$이 됩니다. 이 식을 이용하여 앞 그림의 z 값을 계산하면 다음과 같습니다.

$$z_1=-ln(\frac{1}{0.9}-1)=2.20 \quad z_2=-ln(\frac{1}{0.8}-1)=1.39 \quad z_3=-ln(\frac{1}{0.7}-1)=0.85$$

이 값을 소프트맥스 함수에 대입하면 다음과 같은 정규화된 값을 얻을 수 있습니다.

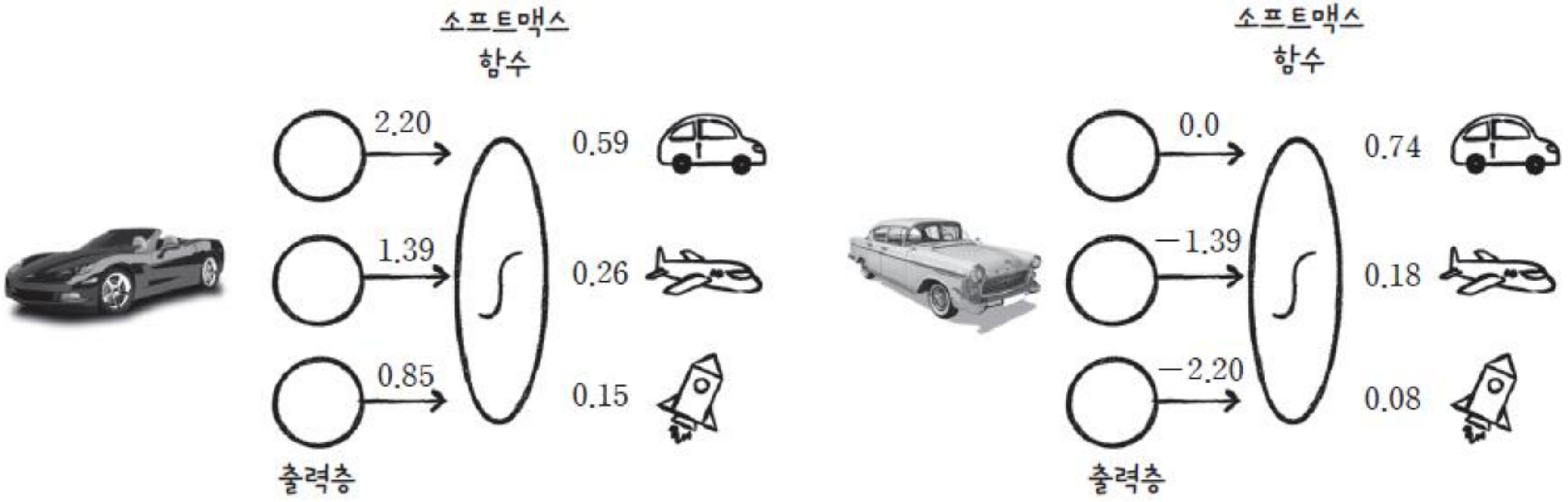
$$\hat{y}_1=\frac{e^{2.20}}{e^{2.20}+e^{1.39}+e^{0.85}}=0.59 \quad \hat{y}_2=\frac{e^{1.39}}{e^{2.20}+e^{1.39}+e^{0.85}}=0.26 \quad \hat{y}_3=\frac{e^{0.85}}{e^{2.20}+e^{1.39}+e^{0.85}}=0.15$$

$$z_1=-ln(\frac{1}{0.5}-1)=0.0 \quad z_2=-ln(\frac{1}{0.2}-1)=-1.39 \quad z_3=-ln(\frac{1}{0.1}-1)=-2.20$$

$$\hat{y}_1=\frac{e^{0.0}}{e^{0.0}+e^{-1.39}+e^{-2.20}}=0.74 \quad \hat{y}_2=\frac{e^{-1.39}}{e^{0.0}+e^{-1.39}+e^{-2.20}}=0.18 \quad \hat{y}_3=\frac{e^{-2.20}}{e^{0.0}+e^{-1.39}+e^{-2.20}}=0.08$$

07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

소프트맥스 함수를 도입한 결과



07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

크로스 엔트로피 손실 함수를 도입하여 가중치와 절편을 업데이트

크로스 엔트로피 손실 함수의 '이진 버전'이 로지스틱 손실 함수

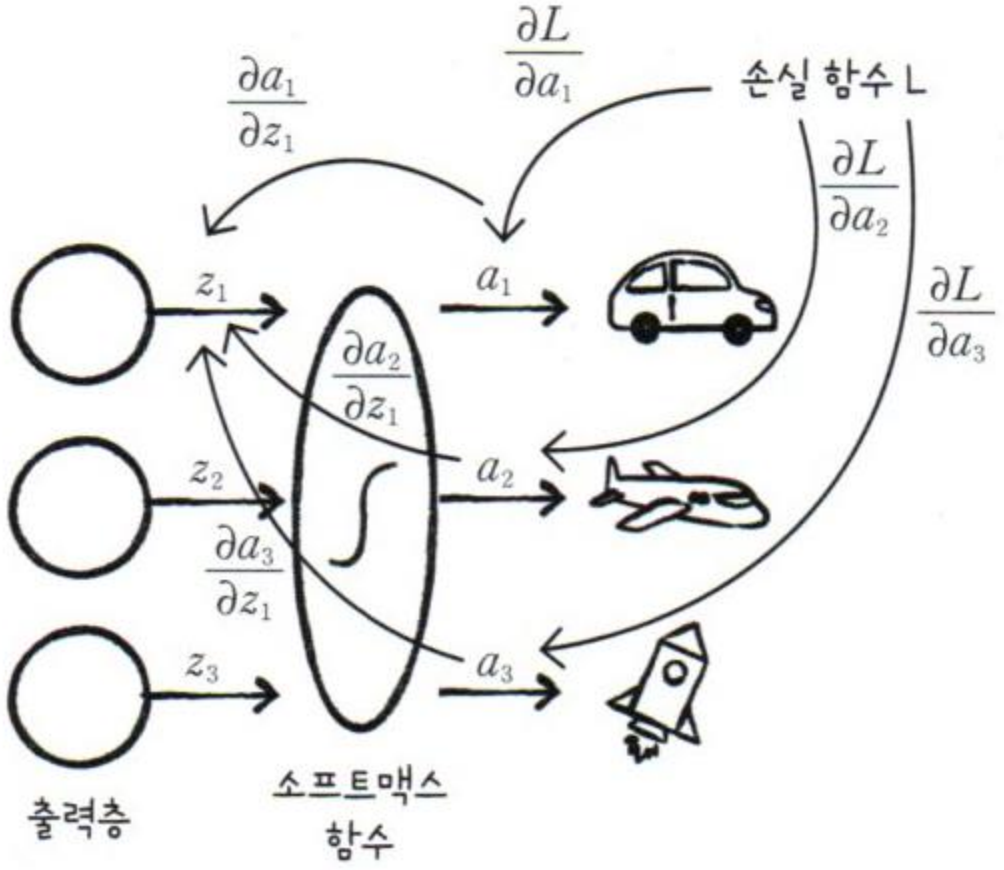
크로스 엔트로피 손실 함수

$$L = -\sum_{c=1}^{c} y_c log(a_c) = -(y_1 log(a_1) + y_2 log(a_2) + \cdots + y_c log(a_c)) = -1 \times log(a_{y=1})$$

로지스틱 손실 함수

$$L = -(y log(a) + (1-y) log(1-a))$$

07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다



크로스 엔트로피 손실 함수 미분

$$\frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1} + \frac{\partial L}{\partial a_3}\frac{\partial a_3}{\partial z_1}$$

Z1에 대한 미분결과

$$\frac{\partial L}{\partial z_1} = -(y_1 - a_1)$$

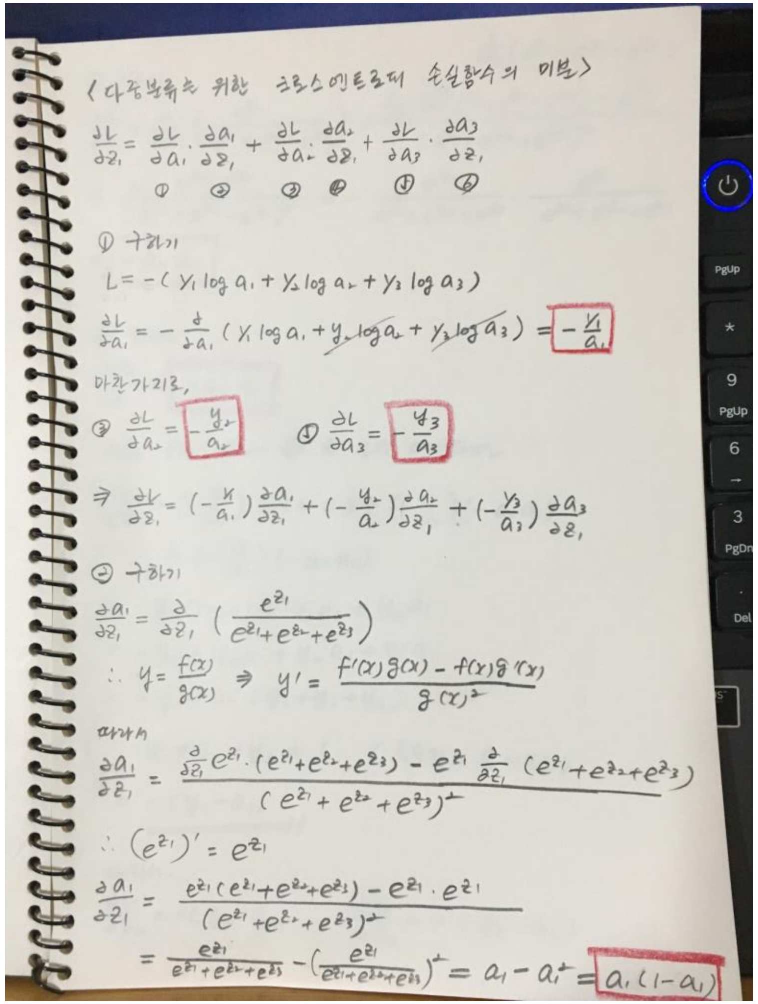
Z2에 대한 미분결과

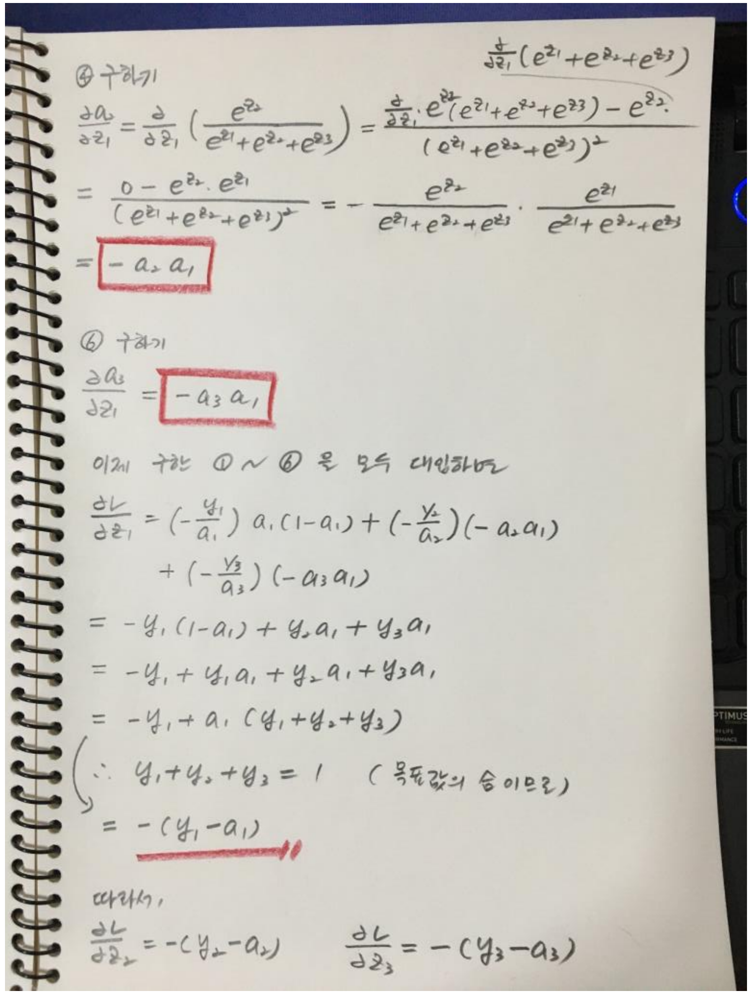
$$-(y_2 - a_2)$$

벡터z에 대해 정리한 결과

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{z}} = -(\boldsymbol{y} - \boldsymbol{a})$$

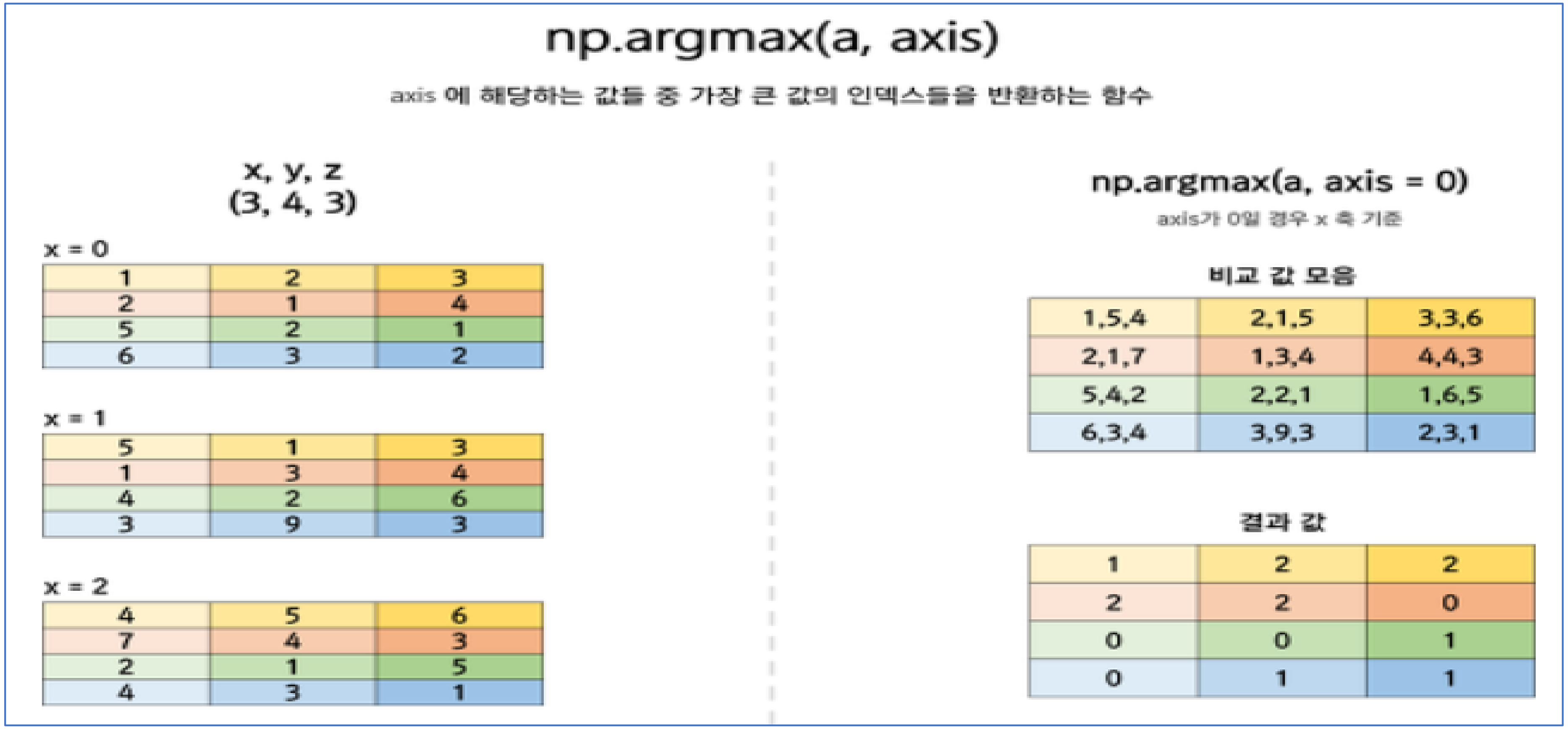
07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다





07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

MinibatchNetwork 클래스를 확장하여 다중 분류 신경망 구현



07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

패션 MNIST 의류 이미지를 다중 분류 신경망으로 분류

패션 MNIST는 텐서플로에 포함되어 있음(텐서플로 최신 버전 설치 필요)

```
!pip install tensorflow_gpu==2.0.0-rc1
```

코랩에 텐서플로 최신 버전 설치

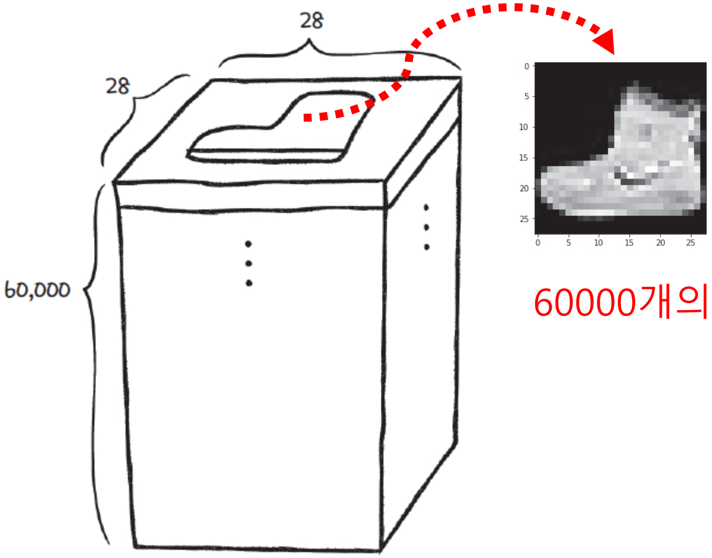
<텐서플로 2.0>

-딥러닝모델을 더 쉽게 만들 수 있도록 개선됨

-케라스(Keras)가 핵심 파이썬 API가 됨.

07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

의류 데이터 준비하기

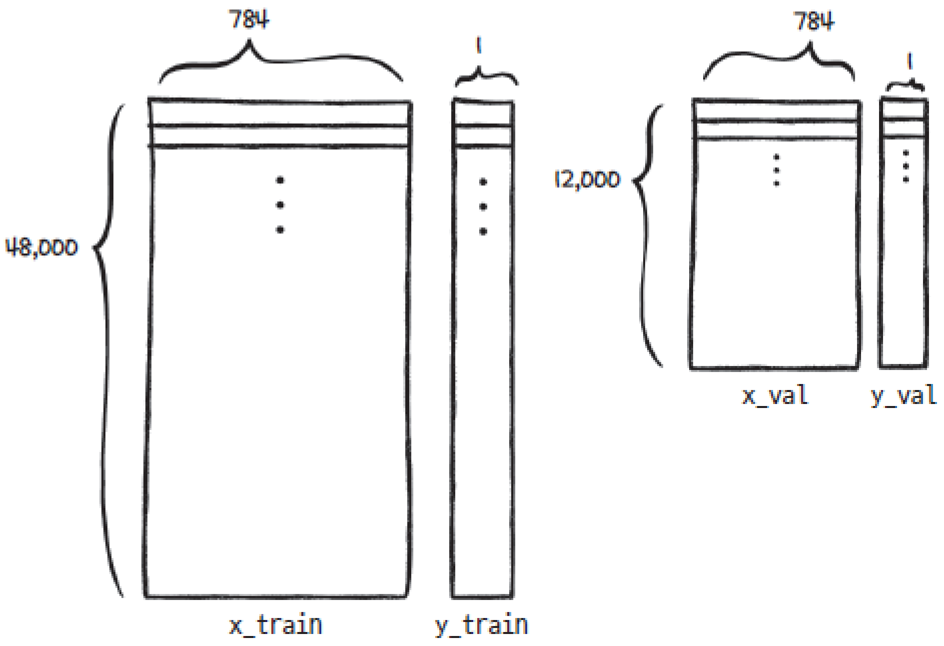


60000개의 흑백 이미지로 구성되어 있음

07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

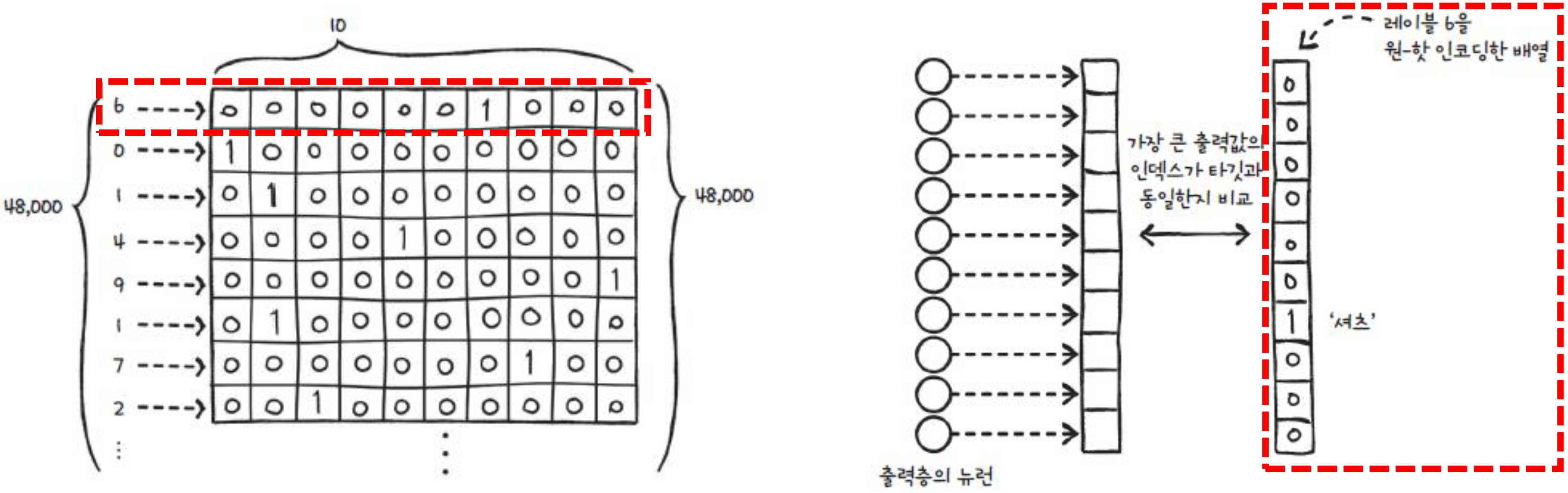
MultiClassNetwork는 1차원 배열을 입력으로 기대함

즉, 패션 MNIST 데이터는 그대로 사용할 수 없음(2차원에서 1차원 데이터로 변환)



07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

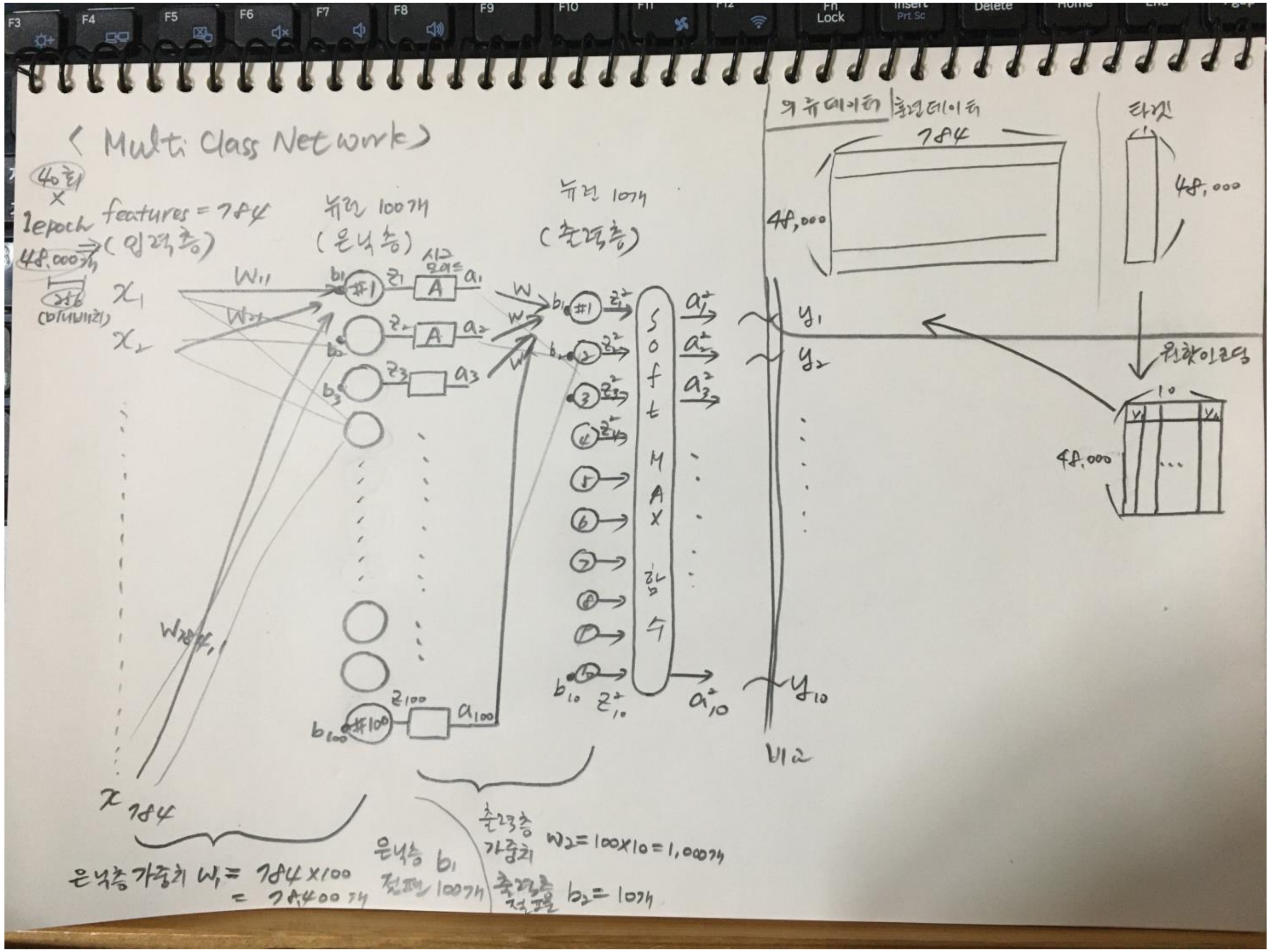
10개의 뉴런으로 구성된 신경망을 위해
타깃 데이터는 다시 원-핫 인코딩으로 변환해야 함



07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다

다중
분류
신경망
훈련

(실습)



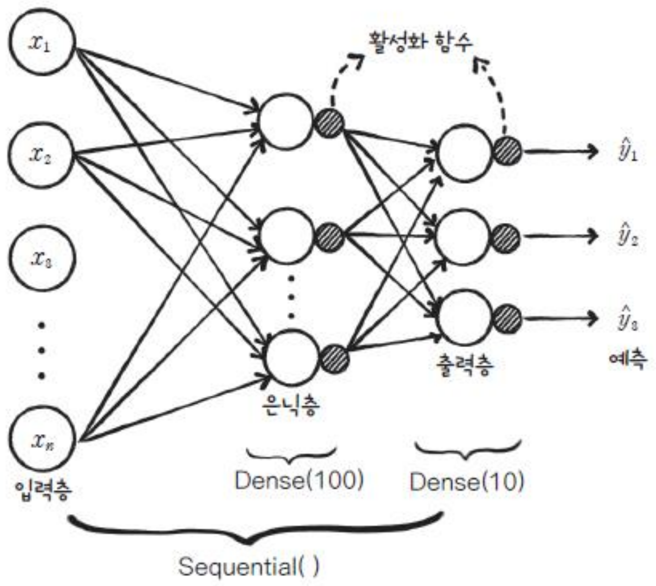
07-2 텐서플로와 케라스를 사용하여 신경망을 만듭니다

1. 케라스 필요성
   : 높은 성능을 위해 전문 라이브러리 도입이 필요함
2. 케라스 정의
   : 딥러닝패키지(텐서플로, 씨아노 등)를 편리하게 사용하기 위해, 만들어진 Wrapper 패키지
3. 케라스 특장점
   1) 텐서플로를 좀 더 쉽게 사용할 수 있다.
   2) 직관적인 신경망 구현이 가능하다.

```
# 신경망 모델을 만듭니다.
model = tf.keras.models.Sequential()
# 완전 연결층을 추가합니다.
model.add(tf.keras.layers.Dense(1))
# 옵티마이저와 손실 함수를 지정합니다.
model.compile(optimizer='sgd', loss='mse')
# 훈련 데이터를 사용하여 에포크 횟수만큼 훈련합니다.
model.fit(x_train, y_train, epochs=10)
```

07-2 텐서플로와 케라스를 사용하여 신경망을 만듭니다

직관적으로 신경망을 구현한다는 뜻?
직접 신경망과 층을 구현하지 않고 Sequential, Dense를 사용



(클래스 정의)
1) Sequential : 인공신경망 모델을 만들기 위한 클래스
2) Dense : 완전연결층을 만들기 위한 클래스

(구축방법)
1) 은닉층과 출력층을 Dense 객체로 구성
2) 각각의 객체를 Sequential 객체에 추가

(용어)
1) 유닛(unit) = 뉴런
2) 커널 = 가중치

07-2 텐서플로와 케라스를 사용하여 신경망을 만듭니다

[Sequential 사용 방법 I]

add() 메소드를 사용하여 Sequential 클래스에 층을 추가함

```
dense = Dense(...)
model.add(dense)
```

```
model = Sequential( )
model.add(Dense(...))
model.add(Dense(...))
```

[Sequential 사용 방법 II]

Compile() 메소드를 사용하여

최적화알고리즘과 손실함수를 지정함

```
model.compile(optimizer='sgd', loss='categorical_crossentropy')
```

[손실함수를 지정하는 매개변수 loss]

제곱오차 손실함수 : mse

로지스틱 회귀 손실함수 : binary_crossentropy

다중분류 손실함수 : categorical_crossentropy

07-2 텐서플로와 케라스를 사용하여 신경망을 만듭니다

케라스로 모델 훈련하고 예측하기 (실습)

```
model = Sequential( )
model.add(Dense(...))
model.add(Dense(...))
model.compile(optimizer='...', loss='...')
model.fit(X, y, epochs=...)
model.predict(X)
model.evalute(X, y)
```