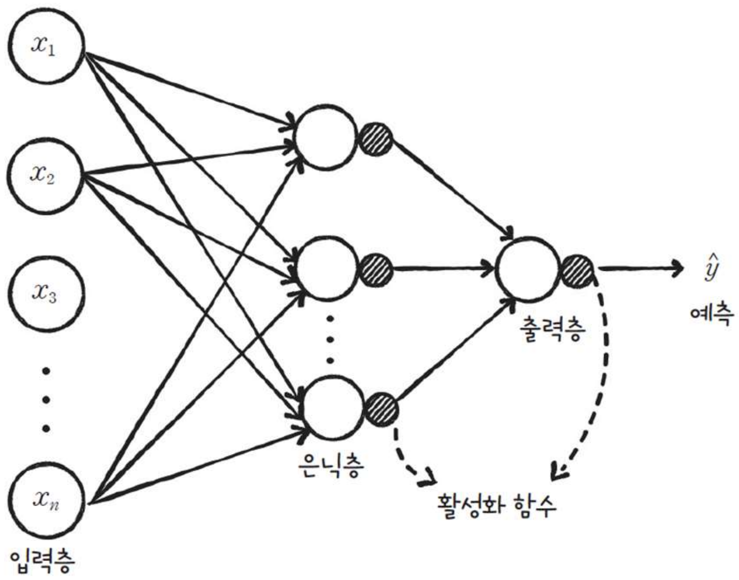
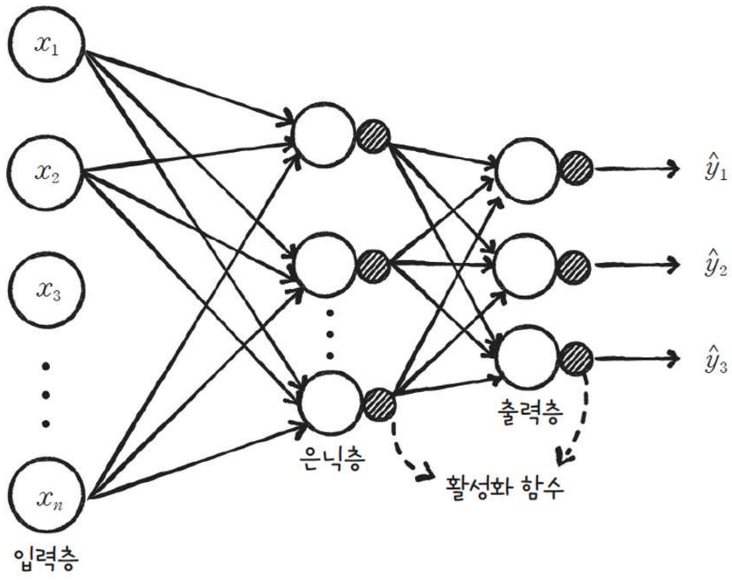
07 여러 개를 분류합니다

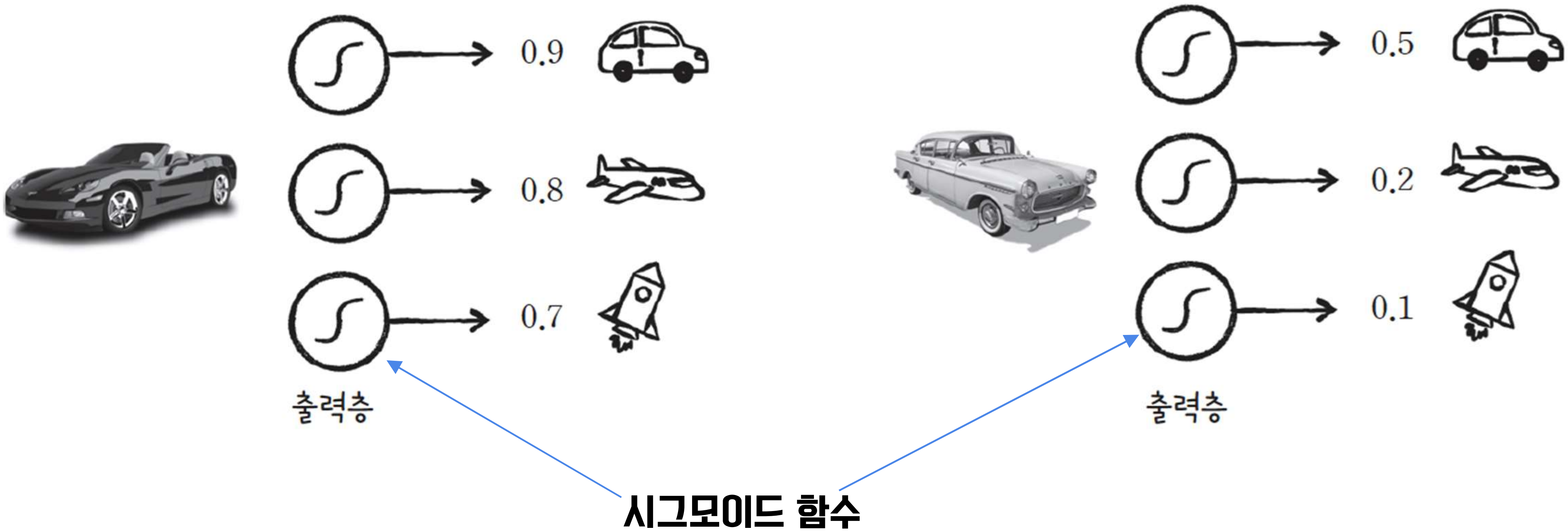
- 다중 분류(multiclass classification)

07-1 여러 개의 이미지를 분류하는 다층 신경망을 만듭니다



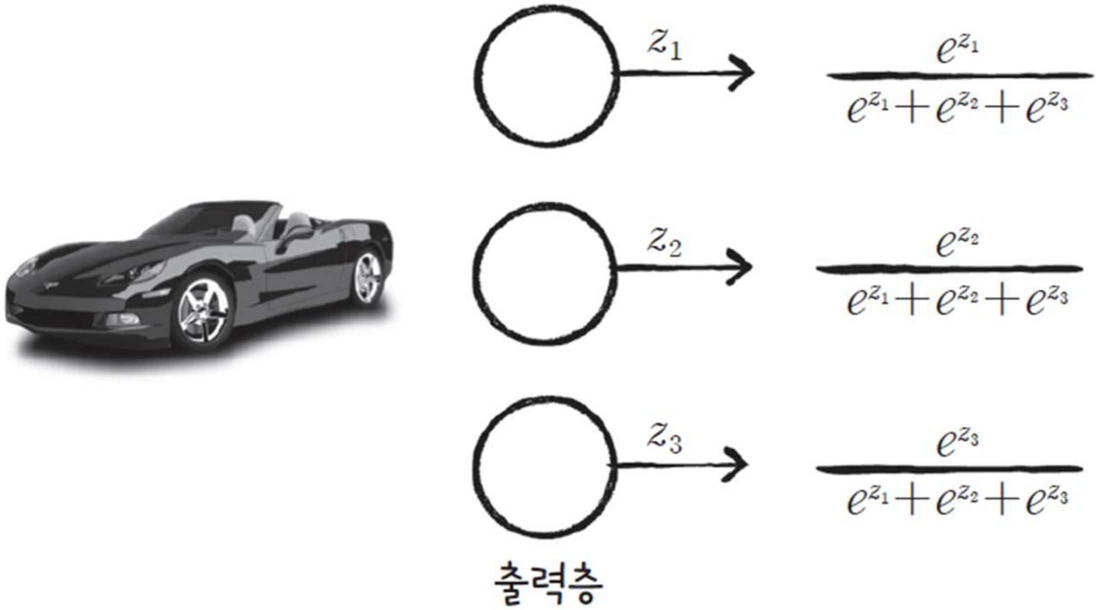


다중 분류의 문제점



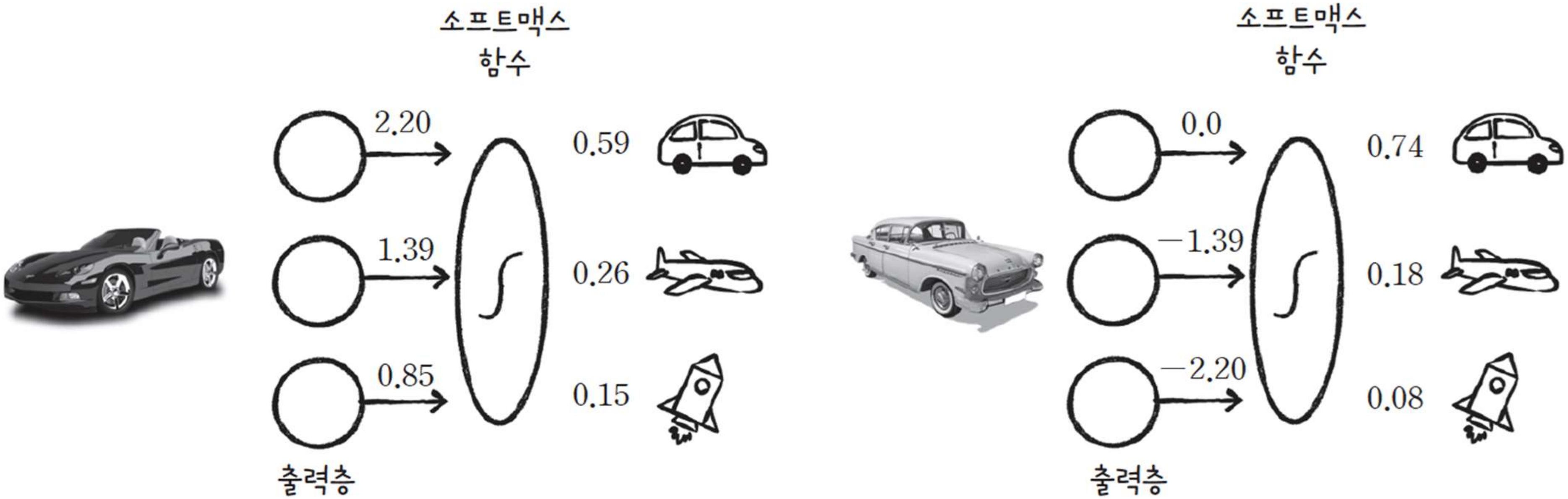
소프트맥스(softmax) 함수

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



출력 정규화

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



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

다중 분류를 위한 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

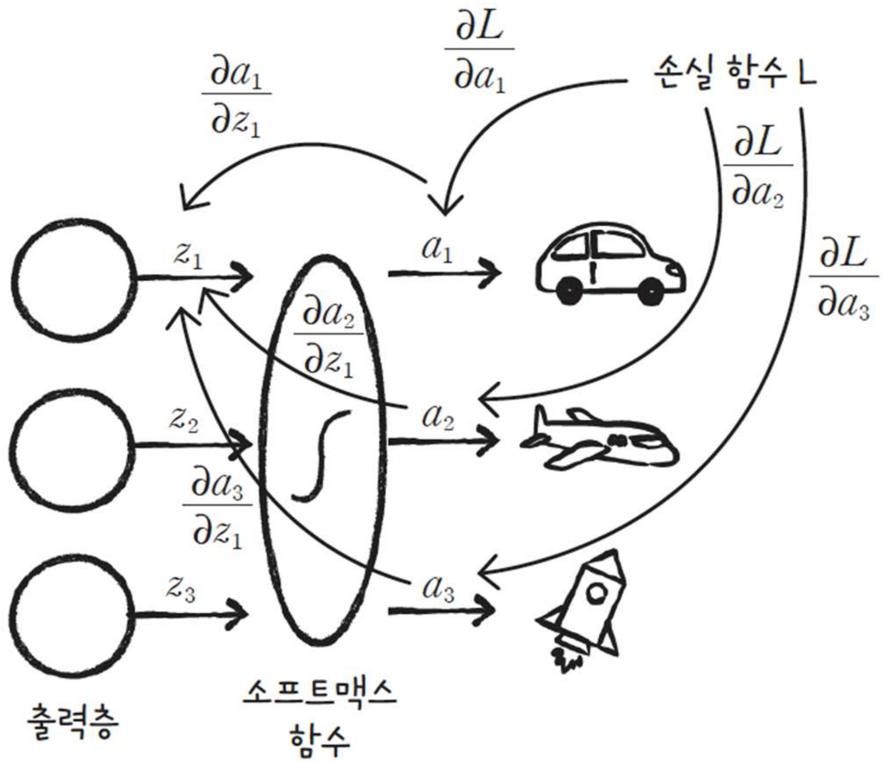
$$L = -(y log(a) + (1-y) log(1-a))$$

$$y = \begin{cases} -log\ a & \text{(양성 클래스인 경우)} \\ -log(1-a) & \text{(음성 클래스인 경우)} \end{cases}$$

크로스 엔트로피 손실 함수의 미분

z_1에 대한 미분 (다변수 함수의 연쇄 법칙)

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

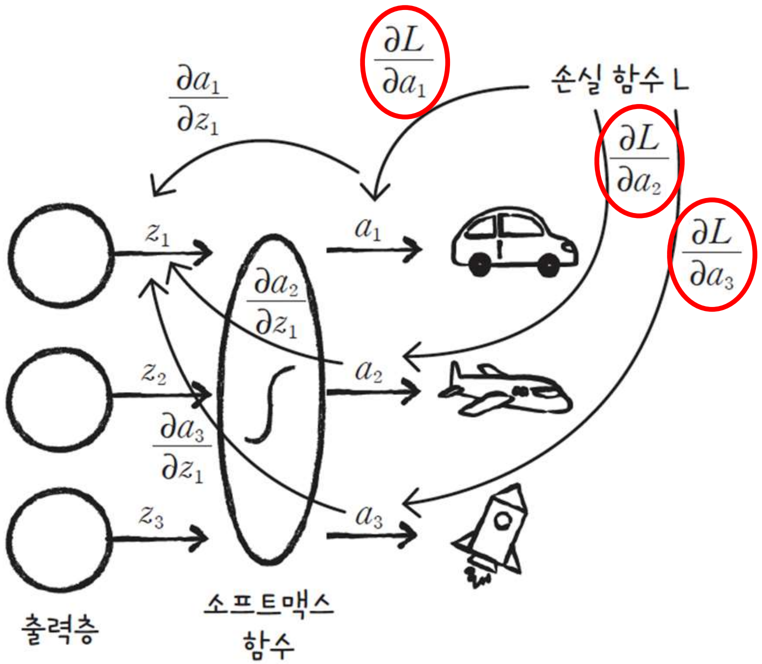


$\partial L / \partial a$

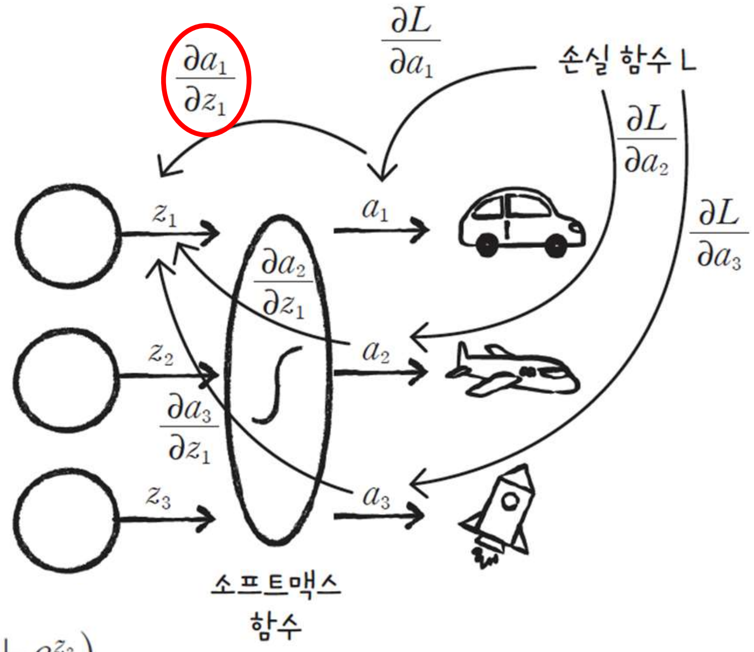
$$L = -(y_1 log \ (a_1) + y_2 log(a_2) + y_3 log(a_3))$$

$$\frac{\partial L}{\partial a_1} = -\frac{\partial}{\partial a_1}(y_1 log a_1 + y_2 log a_2 + y_3 log a_3) = -\frac{y_1}{a_1}$$

$$\frac{\partial L}{\partial a_2} = -\frac{y_2}{a_2} \qquad \frac{\partial L}{\partial a_3} = -\frac{y_3}{a_3}$$



$\partial a_1 / \partial z_1$



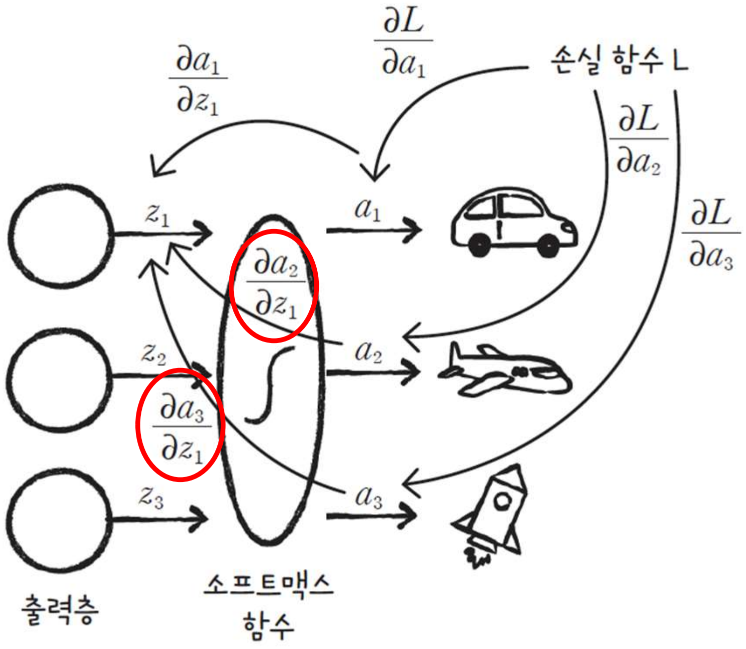
$$a_1 = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

$$\frac{\partial a_1}{\partial z_1} = \frac{\partial}{\partial z_1}\left(\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}\right) = \frac{(e^{z_1}+e^{z_2}+e^{z_3})\frac{\partial}{\partial z_1}e^{z_1} - e^{z_1}\frac{\partial}{\partial z_1}(e^{z_1}+e^{z_2}+e^{z_3})}{(e^{z_1}+e^{z_2}+e^{z_3})^2}$$

$$= \frac{e^{z_1}(e^{z_1}+e^{z_2}+e^{z_3}) - e^{z_1}e^{z_1}}{(e^{z_1}+e^{z_2}+e^{z_3})^2}$$

$$= \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}} - \left(\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}\right)^2 = a_1 - {a_1}^2 = a_1(1-a_1)$$

$\partial a_2/\partial z_1$, $\partial a_3/\partial z_1$



$$\frac{\partial a_2}{\partial z_1} = \frac{\partial}{\partial z_1}\left(\frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}\right) = \frac{(e^{z_1}+e^{z_2}+e^{z_3})\frac{\partial}{\partial z_1}e^{z_2} - e^{z_2}\frac{\partial}{\partial z_1}(e^{z_1}+e^{z_2}+e^{z_3})}{(e^{z_1}+e^{z_2}+e^{z_3})^2}$$

$$= \frac{O - e^{z_2}e^{z_1}}{(e^{z_1}+e^{z_2}+e^{z_3})^2} = -a_2a_1$$

$$\frac{\partial a_3}{\partial z_1} = \frac{\partial}{\partial z_1}\left(\frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}\right) = \frac{(e^{z_1}+e^{z_2}+e^{z_3})\frac{\partial}{\partial z_1}e^{z_3} - e^{z_3}\frac{\partial}{\partial z_1}(e^{z_1}+e^{z_2}+e^{z_3})}{(e^{z_1}+e^{z_2}+e^{z_3})^2}$$

$$= \frac{O - e^{z_3}e^{z_1}}{(e^{z_1}+e^{z_2}+e^{z_3})^2} = -a_3a_1$$

$\partial L / \partial z_1$

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

$$\begin{aligned}\frac{\partial L}{\partial z_1} &= \left(-\frac{y_1}{a_1}\right)\frac{\partial a_1}{\partial z_1} + \left(-\frac{y_2}{a_2}\right)\frac{\partial a_2}{\partial z_1} + \left(-\frac{y_3}{a_3}\right)\frac{\partial a_3}{\partial z_1} \\ &= \left(-\frac{y_1}{a_1}\right)a_1(1-a_1) + \left(-\frac{y_2}{a_2}\right)(-a_2 a_1) + \left(-\frac{y_3}{a_3}\right)(-a_3 a_1) \\ &= -y_1(1-a_1) + y_2 a_1 + y_3 a_1 = -y_1 + (y_1 + y_2 + y_3)a_1 = -(y_1 - a_1)\end{aligned}$$

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

다중 분류 신경망을 구현합니다

소프트맥스 함수 구현

```
def sigmoid(self, z):
    a = 1 / (1 + np.exp(-z))     # 시그모이드 계산
    return a

def softmax(self, z):
    # 소프트맥스 함수
    exp_z = np.exp(z)
    return exp_z / np.sum(exp_z, axis=1).reshape(-1, 1)
```

$$\begin{bmatrix} & \vdots & \\ 0.9, & 0.8, & 0.7 \\ 0.5, & 0.2, & 0.1 \\ & \vdots & \end{bmatrix} \xrightarrow[\text{np.exp(z)}]{} \begin{bmatrix} & \vdots & \\ e^{0.9}, & e^{0.8}, & e^{0.7} \\ e^{0.5}, & e^{0.2}, & e^{0.1} \\ & \vdots & \end{bmatrix}$$

$$\begin{bmatrix} & \vdots & \\ 2.45 & 2.22 & 2.01 \\ 1.64 & 1.22 & 1.10 \\ & \vdots & \end{bmatrix} \xrightarrow[\text{np.sum(exp\_z, axis=1)}]{} [\cdots,\ 6.69,\ 3.97,\ \cdots] \xrightarrow[\text{reshape}(-1,\ 1)]{} \begin{bmatrix} \vdots \\ 6.69 \\ 3.97 \\ \vdots \end{bmatrix}$$

init_weights 메서드 수정

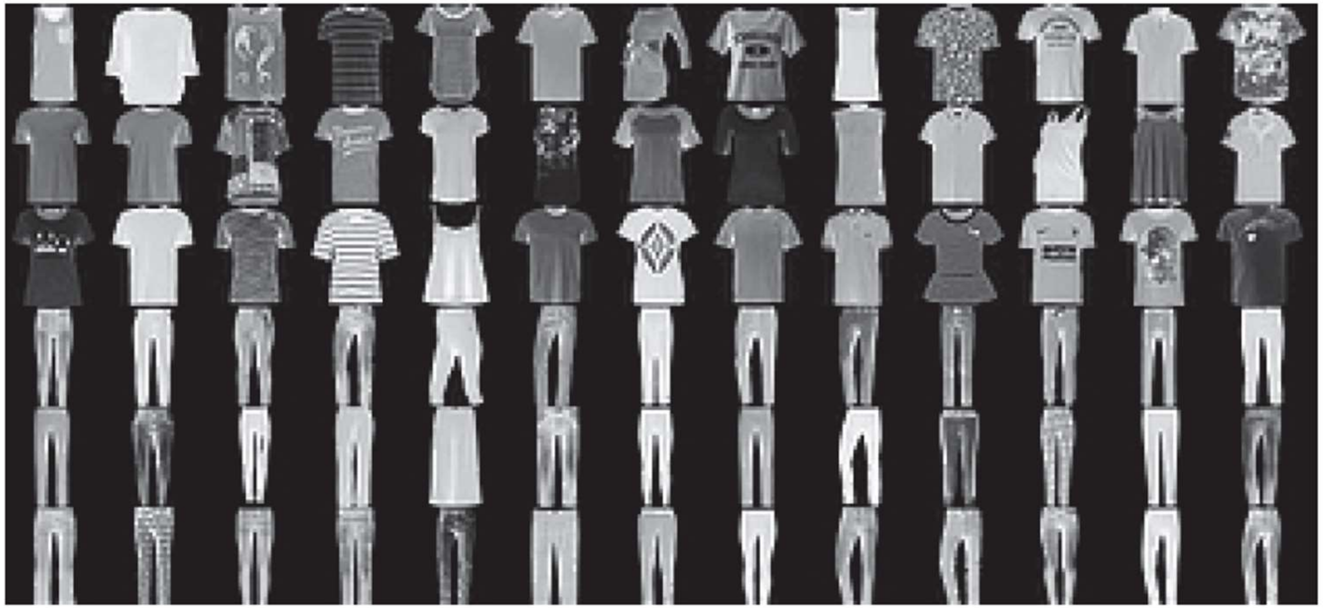
```
def init_weights(self, n_features, n_classes):
    ...
    self.w2 = np.random.normal(0, 1, (self.units, n_classes)) # (은닉층의 크기, 클래스 개수)
    self.b2 = np.zeros(n_classes)
```

update_val_loss() 메서드 수정

```
def update_val_loss(self, x_val, y_val):
    ...
    a = self.softmax(z)                     # 활성화 함수를 적용합니다.
    ...
    # 크로스 엔트로피 손실과 규제 손실을 더하여 리스트에 추가합니다.
    val_loss = np.sum(-y_val*np.log(a))
    ...
```

의류 이미지를 분류합니다

패션 MNIST 데이터셋



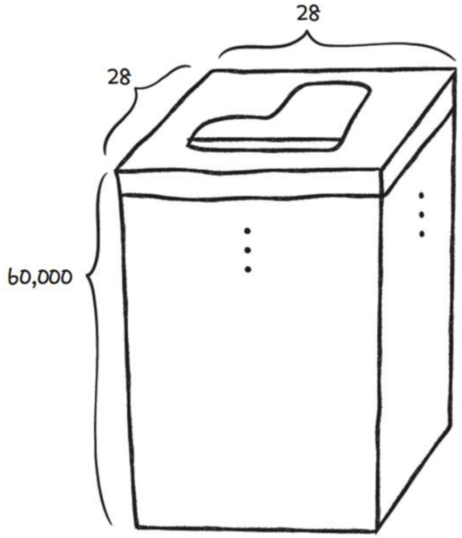
텐서플로 2.0 설치

현재 코랩에 설치된 텐서플로는 2.3.0입니다. 책에서처럼 수동으로 최신 버전을 설치할 필요가 없습니다.

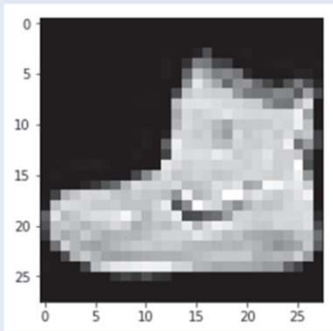
데이터 준비

```
(x_train_all, y_train_all), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data( )
```

```
print(x_train_all.shape, y_train_all.shape)
(60000, 28, 28) (60000,)
```




```
import matplotlib.pyplot as plt
plt.imshow(x_train_all[0], cmap='gray')
plt.show( )
```



타깃 확인

```
print(y_train_all[:10])
[9 0 0 3 0 2 7 2 5 5]
```

```
class_names = ['티셔츠/윗도리', '바지', '스웨터', '드레스', '코트',
               '샌들', '셔츠', '스니커즈', '가방', '앵클부츠']
```

```
print(class_names[y_train_all[0]])
앵클부츠
```

```
np.bincount(y_train_all)
array([6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000])
```

훈련 세트와 검증 세트 준비

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train_all, y_train_all,
stratify=y_train_all, test_size=0.2, random_state=42)
```

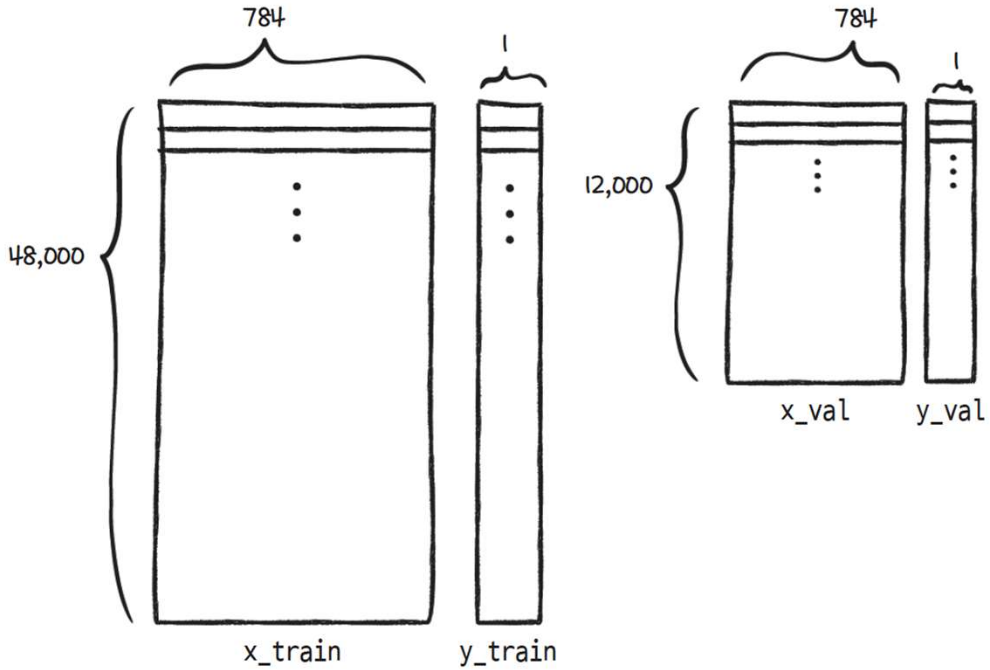
```
np.bincount(y_train)
array([4800, 4800, 4800, 4800, 4800, 4800, 4800, 4800, 4800, 4800])
np.bincount(y_val)
array([1200, 1200, 1200, 1200, 1200, 1200, 1200, 1200, 1200, 1200])
```

```
x_train = x_train / 255
x_val = x_val / 255
```

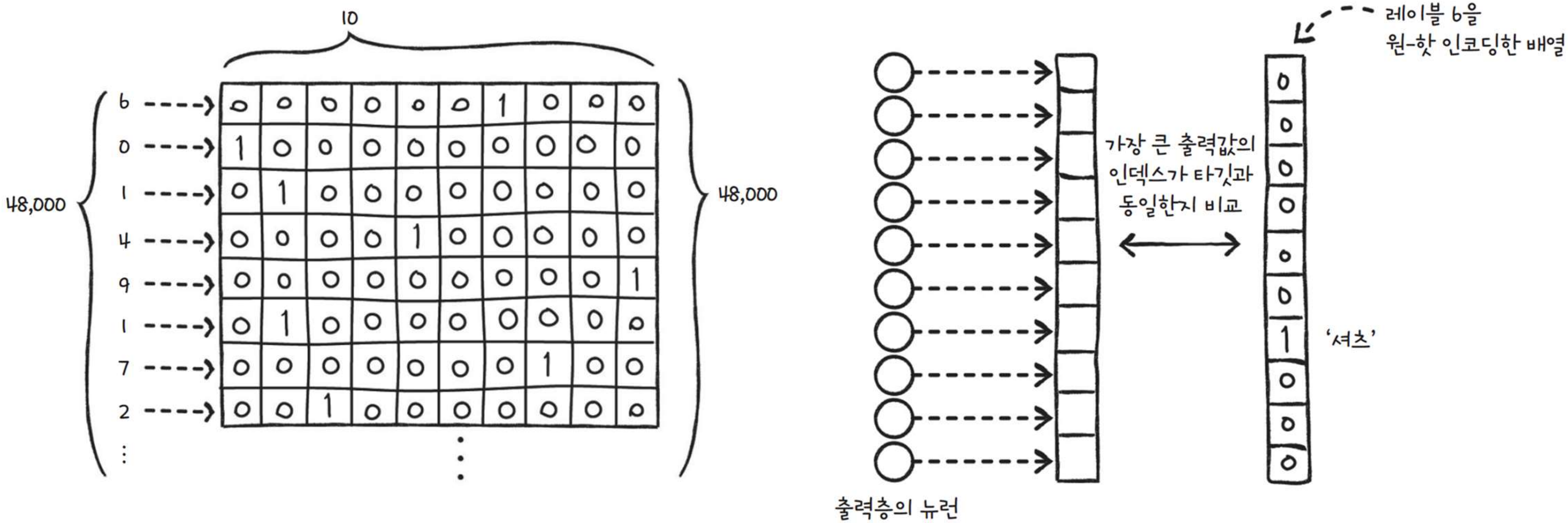
훈련 세트와 검증 세트 차원 변경

```
x_train = x_train.reshape(-1, 784)
x_val = x_val.reshape(-1, 784)
```

```
print(x_train.shape, x_val.shape)
(48000, 784) (12000, 784)
```



타깃을 원-핫 인코딩으로 바꾸기



to_categorical 함수로 원-핫 인코딩하기

```
tf.keras.utils.to_categorical([0, 1, 3])
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 0., 1.]], dtype=float32)
```

```
y_train_encoded = tf.keras.utils.to_categorical(y_train)
y_val_encoded = tf.keras.utils.to_categorical(y_val)
```

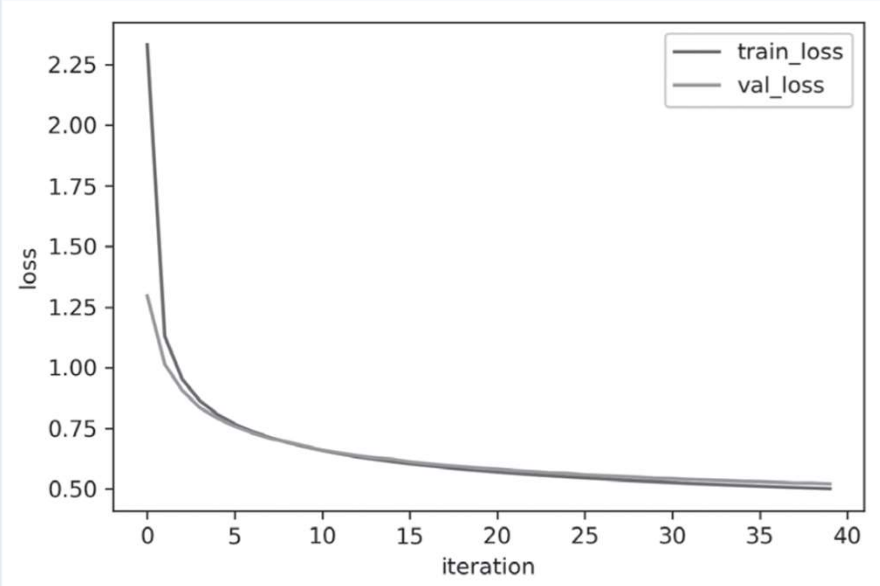
```
print(y_train_encoded.shape, y_val_encoded.shape)
(48000, 10) (12000, 10)
```

```
print(y_train[0], y_train_encoded[0])
6 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```

MutliClassNetwork으로 다중 분류 신경망 훈련하기

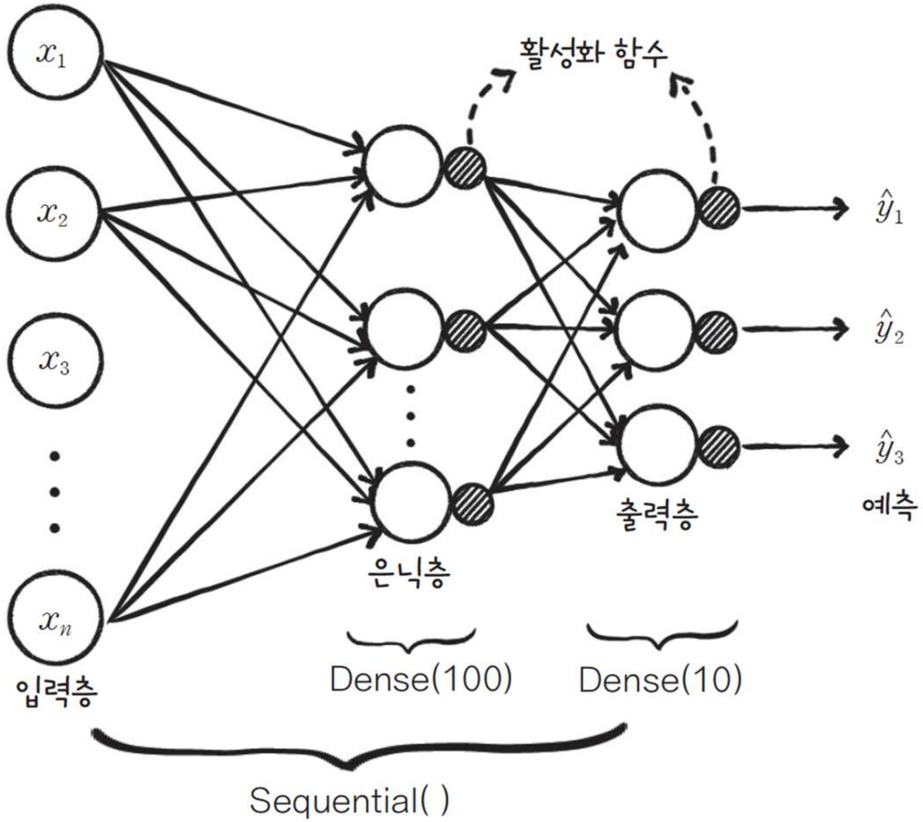
```
fc = MultiClassNetwork(units=100, batch_size=256)
fc.fit(x_train, y_train_encoded,
       x_val=x_val, y_val=y_val_encoded, epochs=40)
........................................
```

```
plt.plot(fc.losses)
plt.plot(fc.val_losses)
plt.ylabel('loss')
plt.xlabel('iteration')
plt.legend(['train_loss', 'val_loss'])
plt.show( )
```




```
fc.score(x_val, y_val_encoded)
0.8150833333333334
```

07-2 텐서플로와 케라스를 사용하여 신경망을 만듭니다



Sequential 모델 훈련하기

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
model = Sequential( )
```

```
model.add(Dense(100, activation='sigmoid', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

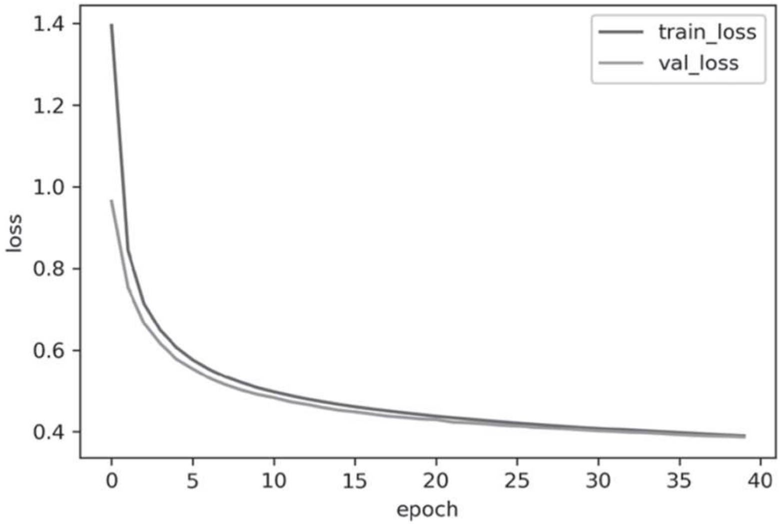
```
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
history = model.fit(x_train, y_train_encoded, epochs=40,
                    validation_data=(x_val, y_val_encoded))
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 2s 45us/sample - loss: 1.3944 - accu-
racy: 0.6396 - val_loss: 0.9643 - val_accuracy: 0.7212
```

손실과 정확도 그리기

```
print(history.history.keys( ))
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```




```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```