인공 신경망


구글 코랩에서 실행하기

패션 MNIST

In [1]:
```
from tensorflow import keras # keras 패키지 import

(train_input, train_target), (test_input, test_target) = keras.datasets.fashion_mnist.load_data() # 패션 MNIST 데이터 다운로드
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

In [4]:
```
print(train_input.shape, train_target.shape) # 사이즈 확인
```

```
(60000, 28, 28) (60000,)
```

In [5]:
```
print(test_input.shape, test_target.shape) # 사이즈 확인
```

```
(10000, 28, 28) (10000,)
```

In [6]:
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1, 10, figsize=(10,10))
for i in range(10): # 10개 이미지 출력
    axs[i].imshow(train_input[i], cmap='gray_r')
    axs[i].axis('off')
plt.show()
```

In [7]:
```
print([train_target[i] for i in range(10)]) # 10개 샘플의 타깃 값 출력
```

```
[9, 0, 0, 3, 0, 2, 7, 2, 5, 5]
```

In [8]:
```
import numpy as np

print(np.unique(train_target, return_counts=True)) # 레이블 당 샘플의 개수 출력
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=uint8), array([6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000]))
```

로지스틱 회귀로 패션 아이템 분류하기

In [9]:
```
train_scaled = train_input / 255.0 # 정규화
train_scaled = train_scaled.reshape(-1, 28*28) # 각 샘플을 1차원 배열로 만듦
```

In [10]:
```
print(train_scaled.shape)
```

```
(60000, 784)
```

In [11]:
```
from sklearn.model_selection import cross_validate
from sklearn.linear_model import SGDClassifier

sc = SGDClassifier(loss='log', max_iter=5, random_state=42)

scores = cross_validate(sc, train_scaled, train_target, n_jobs=-1)
print(np.mean(scores['test_score']))
```

```
0.8196000000000001
```

인공신경망

텐서플로와 케라스

In [12]:
```
import tensorflow as tf
```

In [13]:
```
from tensorflow import keras
```

인공신경망으로 모델 만들기

In [14]:
```
from sklearn.model_selection import train_test_split

train_scaled, val_scaled, train_target, val_target = train_test_split(
    train_scaled, train_target, test_size=0.2, random_state=42)
# 검증 세트 비율을 20%로 하여 훈련 세트와 검증 세트 비율 8 : 2로 나눔
```

In [15]:
```
print(train_scaled.shape, train_target.shape)
```

```
(48000, 784) (48000,)
```

In [16]:
```
print(val_scaled.shape, val_target.shape)
```

```
(12000, 784) (12000,)
```

In [17]:
```
dense = keras.layers.Dense(10, activation='softmax', input_shape=(784,)) # dense layer 생성
# 파라미터로 뉴런 개수와 뉴런의 출력에 적용할 함수, 입력의 크기 전달
```

In [18]:
```
model = keras.Sequential(dense) # 신경망 모델 생성
```

인공신경망으로 패션 아이템 분류하기

In [19]:
```
model.compile(loss='sparse_categorical_crossentropy', metrics='accuracy')
# compile()을 통해서 훈련 저 설정
# loss 함수로는 sparse_categorical_cross entropy, 출력 지표로는 'accuracy'를 설정
```

In [21]:
```
print(train_target[:10])
```

```
[7 3 5 8 6 9 3 3 9 9]
```

In [20]:
```
model.fit(train_scaled, train_target, epochs=5) # epoch=5로 설정하여 5번 반복
```

```
Epoch 1/5
1500/1500 [==============================] - 7s 2ms/step - loss: 0.6081 - accuracy: 0.7942
Epoch 2/5
1500/1500 [==============================] - 4s 2ms/step - loss: 0.4751 - accuracy: 0.8373
Epoch 3/5
1500/1500 [==============================] - 5s 3ms/step - loss: 0.4504 - accuracy: 0.8484
Epoch 4/5
1500/1500 [==============================] - 4s 3ms/step - loss: 0.4375 - accuracy: 0.8518
Epoch 5/5
1500/1500 [==============================] - 3s 2ms/step - loss: 0.4288 - accuracy: 0.8557
<keras.src.callbacks.History at 0x7bd2ce0a3c40>
```

Out[20]:

In [22]:
```
model.evaluate(val_scaled, val_target) # 검증 세트에서 모델 성능 확인
```

```
375/375 [==============================] - 1s 2ms/step - loss: 0.4601 - accuracy: 0.8380
[0.46013012528419495, 0.8379999995231628]
```

Out[22]:

In [ ]:
```

```