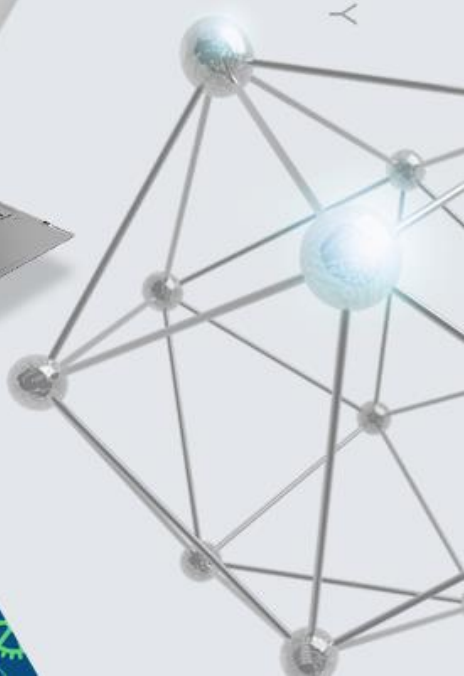
한국기술교육대학교
온라인평생교육원

The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

TECHNOLOGY

딥러닝 입문

데이터셋에 대한 이해와 텐서플로우 활용 방법



The 4th Industrial Revolution is characterized by super connectivity and super intelligence, where various products and services are connected to the network, and artificial intelligence and information communication technologies are used in 3D printing, unmanned transportation, robotics, Of the world's most advanced technologies.

데이터셋에 대한 이해와 텐서플로우 활용 방법

학/습/목/표

1. 데이터셋의 개념에 대해 이해하고, 데이터셋을 목적에 맞게 분류할 수 있다.
2. 딥러닝이 어떻게 학습이 되는지 이해하고, 직접 모델을 구성할 수 있다.
3. 텐서플로우를 활용하여, 직접 모델을 설계하고 학습할 수 있다.

학/습/내/용

1. 데이터셋에 대한 이해
2. 텐서플로우를 활용한 딥러닝의 학습 과정
3. 간단한 딥러닝 모델 실습

1. 데이터셋에 대한 이해

1) 데이터의 중요성

(1) 데이터의 중요성

- 딥러닝은 기계학습의 분야 중 하나

  - 특정 분야의 전문가

- 입력과 결과를 전달해 컴퓨터가 패턴을 찾고 특정 프로그램을 만드는 과정

  - 최대한 많은 데이터를 경험해야만 전문가가 됨

- 2016년 구글 딥마인드 알파고 이후 국내외에서 인공지능 열풍

  - 실험 단계를 넘어 상용화 단계로 일상생활 곳곳에서 활용되고 있음

- 인공지능 적용으로 비생산적인 반복 작업 자동화 등 업무 효율성 증가

- 비즈니스 의사결정 및 예측 등 상당한 효과 기대

1. 데이터셋에 대한 이해

1) 데이터의 중요성

(2) 인공지능을 활용하려면?

- 수많은 데이터를 수집하여 컴퓨터에게 입력, 정답 데이터를 알려주고, 학습시켜야 함

데이터 수집 → 데이터 분석 → 데이터 가공

↓

학습 데이터셋 구성

(3) 인공지능 활용 예시

- 피서철 드론을 활용한 물에 빠진 사람 실시간 탐지

  - 인공지능 학습을 하려면?

    → 물에 빠져 위험한 사람의 이미지 데이터 필요

  - 학습의 정확도를 올리려면?

    → 노인, 어린이, 물에 빠졌지만 위험하지 않은 사람 등 수많은 데이터 필요

1. 데이터셋에 대한 이해

2) 데이터셋의 종류

(1) 데이터셋의 종류

- 학습을 위한 훈련 데이터셋

  - 문제를 풀고 정답을 알려줘 무엇이 틀린지 학습

- 검증을 위한 검증 데이터셋

  - 학습 중 학습이 잘 되는지 평가

- 테스트를 위한 시험(테스트) 데이터셋

  - 학습 완료 후 학습이 잘 되었는지 평가

(2) 올해 수능을 가장 잘 보기 위한 데이터셋 분류하기

- 현재 보유중인 데이터는 총 6개

  - 모의고사 1회~5회

  - 작년 수능 기출문제 1회

- 모두 훈련 데이터로 사용

  - 테스트 없이 공부만 시켰기에 올해 수능을 잘 볼 수 있을지 장담하지 못함

1. 데이터셋에 대한 이해

2) 데이터셋의 종류

(2) 올해 수능을 가장 잘 보기 위한 데이터셋 분류하기

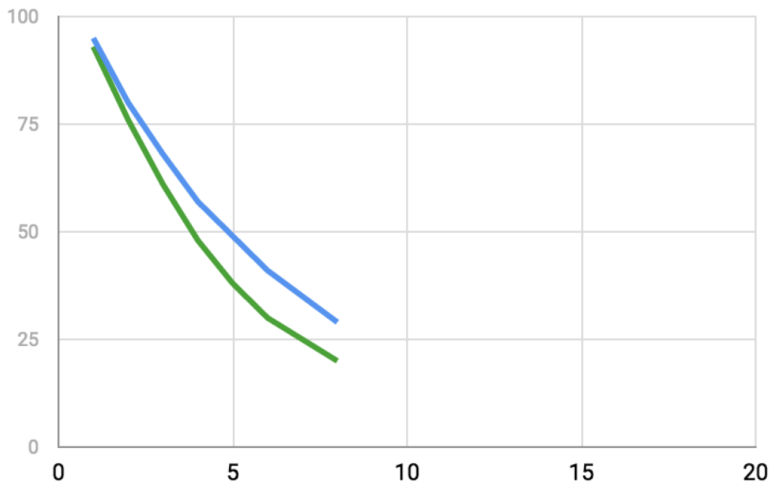
- 모의고사로만 훈련, 작년 수능문제는 공부를 잘 했는지 테스트를 위한 시험셋으로 사용
  - 작년 수능문제를 통해 올해 수능을 어느 정도로 볼지 가늠할 수 있음

- 모의고사를 훈련/검증셋으로 다시 분류, 작년 수능문제는 시험셋으로 사용
  - 검증셋을 활용해 공부를 잘 하고 있는지 중간에 확인하며 학습 가능

1. 데이터셋에 대한 이해

2) 데이터셋의 종류

(1) 훈련 데이터셋 vs 검증 데이터셋

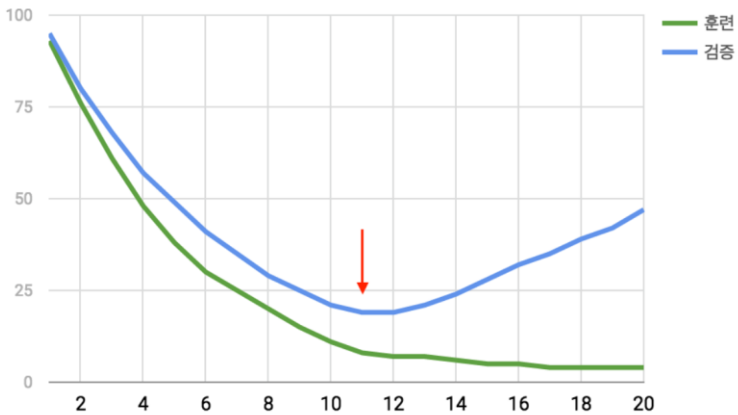


x축: 반복횟수

y축: 틀린 개수

- 언더피팅(underfitting)

  - 더 반복해서 학습시키면 성능이 더 좋아질 수 있음



x축: 반복횟수

y축: 틀린 개수

- 오버피팅(overfitting)

  - 검증셋은 어느 순간부터 안 좋아지기 시작함

  - 검증 데이터셋을 따로 추가해 학습이 잘되고 있는지 평가

1) 딥러닝의 학습과정

(1) 딥러닝의 학습과정

데이터 준비(입력과 정답) — 훈련, 검증, 시험 데이터셋 준비

↓

모델 준비 — 예: 다층 퍼셉트론 인공신경망

↓

학습 과정 준비 — 학습 방법, 평가 방법을 결정함

↓

학습 및 평가 — 컴퓨터에게 문제를 풀리고, 정답을 알려준 뒤 틀린 문제를 다시 맞춰보고 잘못된 점을 수정

↓

분석 및 활용 — 실전 적용, 시각화 등을 활용하여 학습한 모델 사용

2) 딥러닝의 모델 준비

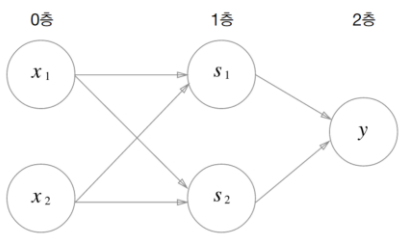
(1) 딥러닝의 모델 준비

- 딥러닝 모델의 종류는 여러 가지가 있음
- MLP(Multi Layer Perceptron, 다층 퍼셉트론) 활용
  - 순차적으로 층을 쌓아나가기 때문에 Sequential 모델 사용
- 하나의 층을 쌓아 연결을 해주는 Dense(densely connected Neural Network)
  - Dense(출력 뉴런 수, input_dim=입력 뉴런 수, activation=활성화 함수)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(20, input_dim=2, activation='sigmoid'))
```

출력 입력

- XOR 예시



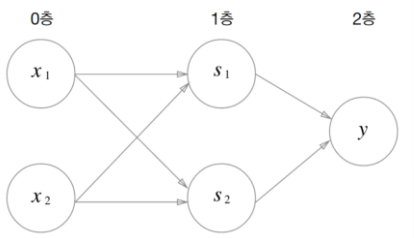
0층 : 입력이 2개,
1층: 0층 출력 2개가 입력
2층: 출력 1개

```
model = Sequential()
model.add(Dense(2, input_dim=2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))
```

2) 딥러닝의 모델 준비

(1) 딥러닝의 모델 준비

- XOR 예시



0층 : 입력이 2개,
1층: 0층 출력 2개가 입력
2층: 출력 1개

```
model = Sequential()
model.add(Dense(2, input_dim=2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))
```

input_dim을 사용하지 않는 이유?
→ 순차적으로 하나씩 층이 추가가 되기 때문에 자동으로 텐서플로우가 인식 해서 적용함

3) 딥러닝의 학습 과정 준비

(1) 딥러닝의 학습 과정 준비

- 완성한 모델을 조립하여 학습하는 방법을 지정

  (정답을 찾아가는 방법)

  - 학습 할 모델의 손실 함수 및 옵티마이저, 평가지표 등을 설정

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(2, input_dim=2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))

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

4) 딥러닝의 학습

(1) 준비된 모델 학습

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(2, input_dim=2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))

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

hist = model.fit(X_train, Y_train, epochs=200, batch_size=10, validation_data=(X_val,Y_val))
```

- 모델 학습을 위한 함수
- 반복횟수
- 한번에 풀 갯수
- 검증 데이터셋

(2) 배치사이즈(Batch_size)

- 한 번에 몇 문제씩 풀고 정답을 맞춰볼까?
  - 학습: 문제를 다 풀고 틀린 문제를 맞춰보며 가중치 갱신을 통해 정답을 더 잘 맞출 수 있도록 모델을 향상시켜 나가는 것

문제지	문제1	문제2	문제3	문제4	문제5	문제6	...	문제100
해답지	정답1	정답2	정답3	정답4	정답5	정답6	...	정답100

- Batch_size 10
  - 10문제를 풀고 또 비교하고 가중치 갱신
- Batch_size 100
  - 100문제를 다 푼 다음에 한번만 정답을 비교해서 가중치 갱신

4) 딥러닝의 학습

(2) 배치사이즈(Batch_size)

큰 것	VS	작은 것
• 학습 속도 빠름 → 가중치 갱신을 자주 하지 않기 때문 • 많은 메모리, 컴퓨팅 성능 필요 → 한 번에 많은 문제를 풀고 문제에 대한 정답들을 기억 및 학습 해야함		• 학습 속도 느림 → 가중치 갱신을 자주 하기 때문 • 많은 메모리, 컴퓨팅 성능 불필요

5) 딥러닝의 평가

(1) 딥러닝의 평가

- 학습 완료된 모델을 시험셋에 적용하여 모델 성능 평가

Loss(손실)	Accuracy(정확도)
낮을 수록 모델의 성능이 좋음	높을 수록 모델의 성능이 좋음

```
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=1)

print('')
print('loss : ' + str(loss_and_metrics[0]))
print('accuracy : ' + str(loss_and_metrics[1]))
```

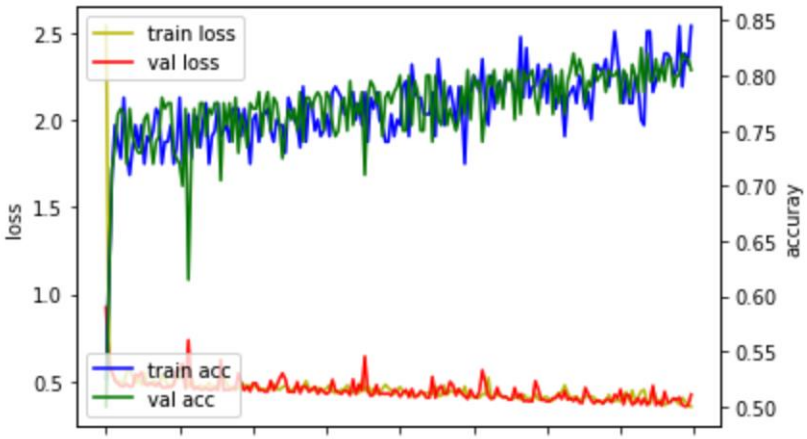
5) 딥러닝의 평가

(2) 딥러닝의 분석 및 활용

- 분석 및 활용 방법의 대표적 방법은 그래프로 시각화하여 활용하는 것

  - 단순한 숫자로는 알기 힘든 정보들을 한눈에 파악 할 수 있음

```
import matplotlib.pyplot as plt
fig, loss_ax = plt.subplots()
acc_ax = loss_ax.twinx()

loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')

acc_ax.plot(hist.history['accuracy'], 'b', label='train acc')
acc_ax.plot(hist.history['val_accuracy'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuray')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```



Accuracy(정확도) : 우상향일수록 좋음

Loss(손실) : 우하향일수록 좋음

3. 간단한 딥러닝 모델 실습

1) 귤과 오렌지 분류 실습

(1) 분류를 위한 속성 값

- 크기
  - 귤 : 1~10
  - 오렌지 : 7~20
- 무게
  - 귤 : 50~100
  - 오렌지 : 80~130

(2) 모듈 호출 및 데이터 준비

```
import tensorflow as tf
import tensorflow.keras.utils as utils
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
import numpy as np
import matplotlib.pyplot as plt
import random

x_data = []
for i in range(100):
    x_data.append([random.randint(7, 20),random.randint(80, 130)]) #오렌지 [크기, 무게]
    x_data.append([random.randint(1, 10),random.randint(50, 100)]) #귤 [크기, 무게]
y_data = []
for i in range(100):
    y_data.append(1)  #오렌지
    y_data.append(0)  #귤
```

3. 간단한 딥러닝 모델 실습

1) 귤과 오렌지 분류 실습

(2) 모듈 호출 및 데이터 준비

- Reshape

  - 하나의 리스트에 통으로 200개의 데이터를 입력 했기 때문에, 각각 분리하기 위해 사용

```
X_train = np.array([x_data])
X_train = X_train.reshape(200,2)

Y_train = np.array(y_data)
Y_train = Y_train.reshape(200,)
```

```
print(X_train)
```

```
[[ 19 120]
 [  9  97]
 [ 16 118]
 [  4  50]
 [ 19  84]
 [  9  72]
 [ 12  88]
 [  3  81]
```

```
print(Y_train)
```

```
[1 0 1 0 1 0 1 0 1 0 1 0
 0 1 0 1 0 1 0 1 0 1 0 1
 1 0 1 0 1 0 1 0 1 0 1 0
 0 1 0 1 0 1 0 1 0 1 0 1
```

```
X_train = np.array([x_data])
print(X_train.shape)
X_train = X_train.reshape(200,2)
print(X_train.shape)
```

```
(1, 200, 2)
(200, 2)
```

3. 간단한 딥러닝 모델 실습

1) 귤과 오렌지 분류 실습

(3) 모델 구성 및 학습 과정

- 3개의 층을 쌓음
  - 처음 입력 : 2개(크기, 무게)
  - 최종 출력 : 1개(귤 or 오렌지)

```
model = Sequential()
model.add(Dense(20, input_dim=2, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

(4) 모델 학습

- 200번 반복, 배치 사이즈는 10,
  검증 데이터셋은 훈련 데이터셋을 그대로 활용

```
hist = model.fit(X_train, Y_train, epochs=200, batch_size=10,
                 validation_data=(X_train,Y_train))
```

```
Epoch 1/200
20/20 [==============================] - 0s 5ms/step - loss: 2.5395 - accuracy: 0.5250 - val_loss: 0.9257 - val_accur
acy: 0.5000
Epoch 2/200
20/20 [==============================] - 0s 2ms/step - loss: 0.6525 - accuracy: 0.6200 - val_loss: 0.5708 - val_accur
acy: 0.6150
Epoch 3/200
20/20 [==============================] - 0s 2ms/step - loss: 0.5401 - accuracy: 0.7100 - val_loss: 0.5586 - val_accur
acy: 0.7150
```

3. 간단한 딥러닝 모델 실습

1) 귤과 오렌지 분류 실습

(5) 모델 평가 및 분석

- 테스트 데이터 생성 및 모델 평가

```
x_data = []
for i in range(10):
    x_data.append([random.randint(7, 20),random.randint(80, 130)])
    x_data.append([random.randint(1, 10),random.randint(50, 100)])
y_data = []
for i in range(10):
    y_data.append(1)
    y_data.append(0)

X_test = np.array([x_data])
X_test = X_test.reshape(20,2)

Y_test = np.array(y_data)
Y_test = Y_test.reshape(20,)
```

```
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=1)

print('')
print('loss : ' + str(loss_and_metrics[0]))
print('accuracy : ' + str(loss_and_metrics[1]))
```

```
20/20 [==============================] - 0s 629us/step - loss: 0.2269 - accuracy: 1.0000

loss : 0.2269195020198822
accuracy : 1.0
```

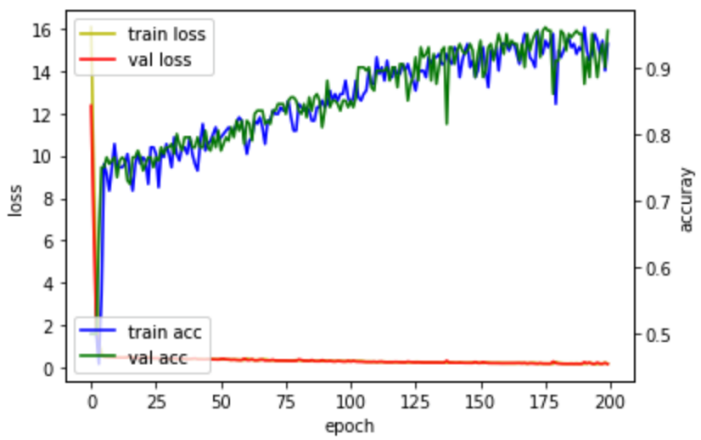
3. 간단한 딥러닝 모델 실습

1) 귤과 오렌지 분류 실습

(5) 모델 평가 및 분석

- 그래프를 활용한 학습 결과 분석

```
fig, loss_ax = plt.subplots()
acc_ax = loss_ax.twinx()

loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')

acc_ax.plot(hist.history['accuracy'], 'b', label='train acc')
acc_ax.plot(hist.history['val_accuracy'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuray')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```



1) 귤과 오렌지 분류 실습

(6) 전체 코드

```
import tensorflow as tf
import tensorflow.keras.utils as utils
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline

x_data = []
for i in range(100):
    x_data.append([random.randint(7, 20),random.randint(80, 130)]) #오렌지 [크기, 무게]
    x_data.append([random.randint(1, 10),random.randint(50, 100)]) #귤 [크기, 무게]
y_data = []
for i in range(100):
    y_data.append(1)  #오렌지
    y_data.append(0)  #귤

x_data = []
for i in range(10):
    x_data.append([random.randint(7, 20),random.randint(80, 130)])
    x_data.append([random.randint(1, 10),random.randint(50, 100)])
y_data = []
for i in range(10):
    y_data.append(1)
    y_data.append(0)

X_test = np.array([x_data])
X_test = X_test.reshape(20,2)

Y_test = np.array(y_data)
Y_test = Y_test.reshape(20,)
```

```
model = Sequential()
model.add(Dense(20, input_dim=2, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

hist = model.fit(X_train, Y_train, epochs=200,
                 batch_size=10, validation_data=(X_train,Y_train))

loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=1)

print('')
print('loss : ' + str(loss_and_metrics[0]))
print('accuracy : ' + str(loss_and_metrics[1]))

fig, loss_ax = plt.subplots()
acc_ax = loss_ax.twinx()

loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')

acc_ax.plot(hist.history['accuracy'], 'b', label='train acc')
acc_ax.plot(hist.history['val_accuracy'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuray')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```

Summary

1. 데이터셋에 대한 이해

- 인공지능은 결국 처음에는 데이터를 활용하여 학습을 해야하기 때문에 데이터가 아주 중요함
- 데이터셋은 크게 세 가지로 훈련 데이터셋, 검증 데이터셋, 시험 데이터셋으로 나뉨
- 훈련 데이터셋은 학습을 위한 데이터로 정답을 알려줘 틀린 것을 제대로 맞춰가기 위해 활용됨
- 검증 데이터셋은 학습과정 중에 학습이 제대로 되고 있는지 평가하기 위해 활용함
- 시험 데이터셋은 학습 완료 후 모델의 성능 평가를 위해 활용함

2. 텐서플로우를 활용한 딥러닝의 학습 과정

- 텐서플로우 2.x 버전부터는 케라스를 활용하여 쉽게 딥러닝 모델을 만들고 학습 시킬 수 있음
- 모델 구성은 Sequential의 Dense 를 활용하여 다층 퍼셉트론을 쉽게 구현할 수 있음
- 모델 구성이 끝나면 fit 함수로 간단하게 반복 횟수, 배치사이즈 등을 설정하고 학습 시킬 수 있으며, 학습이 종료된 후에는 시험 셋을 활용하여 evaluate 함수로 학습된 모델을 평가할 수 있음
- 학습 후 그래프 등 모델을 시각화하여 분석할 수 있음

3. 간단한 딥러닝 모델 실습

- 귤과 오렌지 분류 실습

  ① 모듈 호출 및 데이터 준비

  ② 모델 구성 및 학습 과정 준비

  ③ 모델 학습

  ④ 모델 평가 및 분석

  ⑤ 그래프를 활용한 학습 결과 분석