



첫째마당

딥러닝 시작을 위한 준비 운동

2장 딥러닝의 핵심 미리 보기

- 1** 미지의 일을 예측하는 원리
- 2** 딥러닝 코드 실행해 보기
- 3** 딥러닝 개괄하기
- 4** 이제부터가 진짜 딥러닝?



1 미지의 일을 예측하는 원리



1 미지의 일을 예측하는 원리

- 미지의 일을 예측하는 원리



기존 환자의
데이터를 이용해
새로운 환자의
수술 결과를 예측하는
프로그램을 짜 보세요!



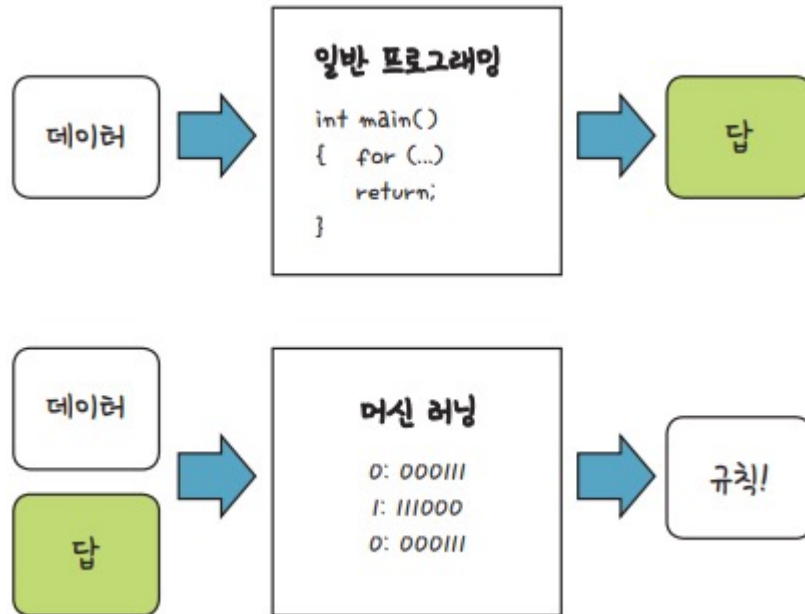
1 미지의 일을 예측하는 원리

● 미지의 일을 예측하는 원리

- 이러한 과제를 받았다고 해 보자
- 기존 프로그래밍 기법으로 이러한 프로그램을 만들려면 쉽지 않음
- 머신 러닝은 이를 매우 쉽게 해결
- 기존에 우리가 했던 프로그래밍이 데이터를 입력해서 답을 구하는 데 초점이 맞추어 있었음
- 머신 러닝은 데이터 안에서 규칙을 발견하고 그 규칙을 새로운 데이터에 적용해서 새로운 결과를 도출하는 데 초점이 맞추어 있기 때문임
- 머신 러닝은 기존 데이터를 이용해 아직 일어나지 않은 미지의 일을 예측하기 위해 만들어진 기법

1 미지의 일을 예측하는 원리

▼ 그림 2-1 | 머신 러닝과 일반 프로그래밍 비교



1 미지의 일을 예측하는 원리

● 미지의 일을 예측하는 원리

- 실제 예를 들어 머신 러닝을 활용하는 방법에 대해 살펴보자
- 중환자를 전문으로 수술하는 어느 병원의 의사가 수많은 환자를 수술해 오던 중 다음과 같은 질문을 던져 보았음
- “혹시 수술하기 전에 수술 후의 생존율을 수치로 예측할 수 있는 방법이 있을까?” 방법이 있음
- 자신이 그동안 집도한 수술 환자의 수술 전 상태와 수술 후 생존율을 정리해 놓은 **데이터**를 머신 러닝 알고리즘에 넣는 것
- 머신 러닝은 **데이터가 가진 패턴과 규칙**을 분석해서 저장해 두자
- 이후 새로운 환자가 오면 저장된 분석 결과와 비교해 생존 가능성을 예측하게 되는 것
- 이것이 바로 머신 러닝이 하는 일

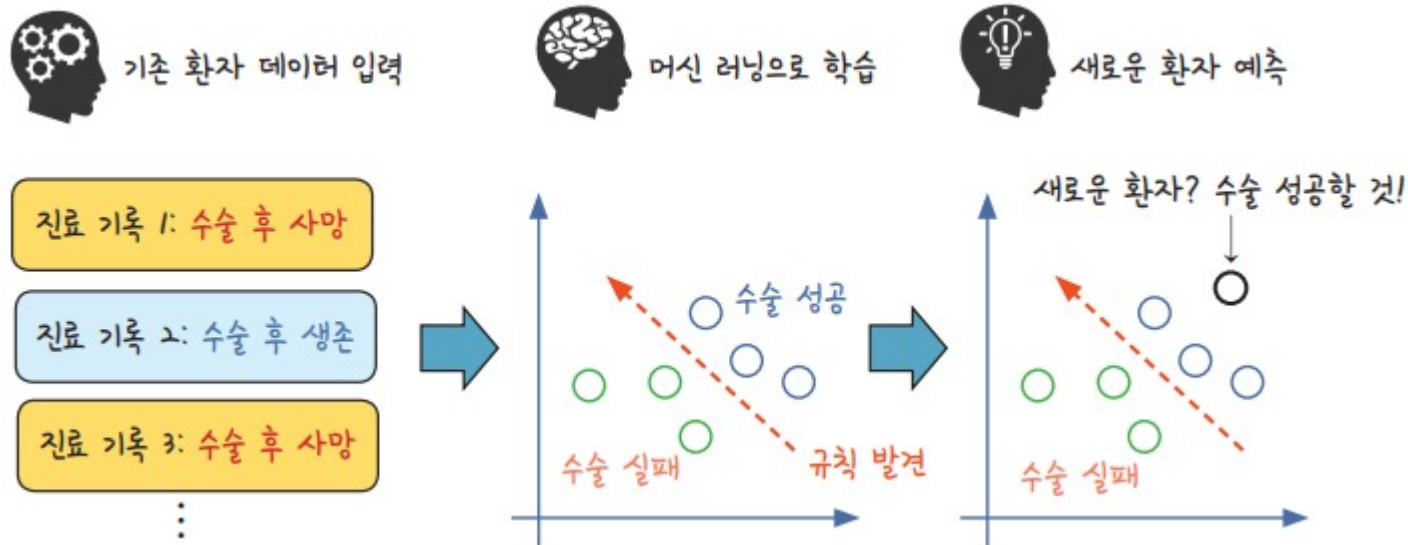
1 미지의 일을 예측하는 원리

● 미지의 일을 예측하는 원리

- 여기서 데이터가 입력되고 패턴이 분석되는 과정을 **학습(training)**이라고 함
- 다시 말해 학습 과정은 깨끗한 좌표 평면에 기존 환자들을 하나씩 배치하는 과정이라고 할 수 있음
- 예를 들어 환자들의 분포를 그래프 위에 펼쳐 놓고 이 분포도 위에 수술 성공과 실패 여부를 구분짓는 경계를 그려 넣음
- 이를 잘 저장해 놓았다가 새로운 환자가 오면 분포도를 다시 꺼냄
- 새 환자가 분포도의 어디쯤 위치하는지 정하고는 아까 그려 둔 경계선을 기준으로 이 환자의 수술 결과를 예측하는 것

1 미지의 일을 예측하는 원리

▼ 그림 2-2 | 머신 러닝의 학습 및 예측 과정





1 미지의 일을 예측하는 원리

● 미지의 일을 예측하는 원리

- 우리가 지금 배우려는 것이 바로 이러한 학습과 예측의 구체적인 과정
- 머신 러닝의 예측 성공률은 결국 얼마나 정확한 경계선을 긋느냐에 달려 있음
- 더 정확한 선을 긋기 위한 여러 가지 노력이 계속되어 왔고, 그 결과 퍼셉트론(perceptron), 아달라인(adaline), 선형 회귀(linear regression) 등을 지나 오늘날 딥러닝이 탄생



2 딥러닝 코드 실행해 보기



2 딥러닝 코드 실행해 보기

- 딥러닝 코드 실행해 보기

- 백문이 불여일견! 먼저 딥러닝의 코드를 불러와 그 형태를 살펴보고, 예측 결과가 나오는 과정을 미리 살펴보자
- 깃허브에 있는 소스 코드를 내 계정으로 불러와 저장하고 실행하는 연습을 해보자

2 딥러닝 코드 실행해 보기

● 딥러닝 코드 실행해 보기

1. 먼저 웹 브라우저에 다음 주소를 입력해 소스 코드가 저장되어 있는 깃허브에 접속

`https://github.com/taehojo/deeplearning`

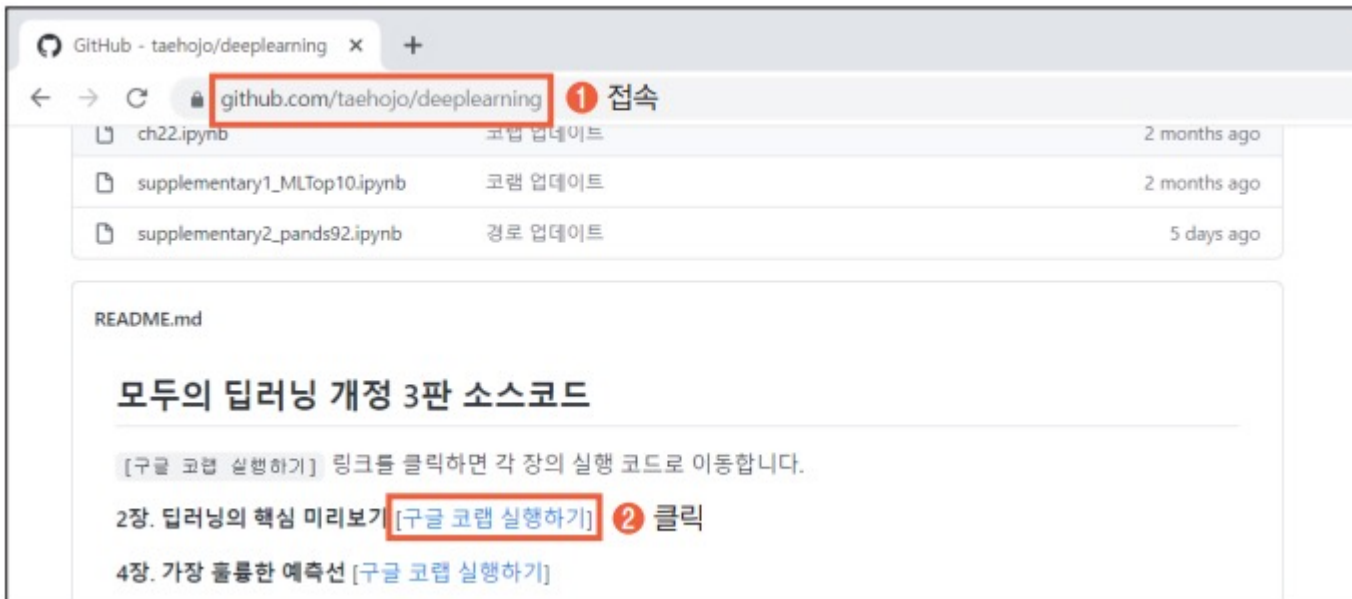
구글 코랩을 바로 실행하려면 깃허브 화면 하단에 있는 2장 딥러닝의 핵심
미리보기

[구글 코랩 실행하기]를 클릭



2 딥러닝 코드 실행해 보기

▼ 그림 2-3 | 깃허브에 접속해 소스 코드 확인하기





2 딥러닝 코드 실행해 보기

- 딥러닝 코드 실행해 보기

- 주피터 노트북으로 코드와 실행 결과를 먼저 확인한 후 구글 코랩으로 이동하려면, 저자 깃허브에 접속하여 그림 2-4의 ❶ 화면 상단 목록에 있는 **ch02.ipynb**를 클릭한 후 열린 주피터 노트북 상단의 ❷ 코랩에서 실행하기 이미지를 클릭



2 딥러닝 코드 실행해 보기

▼ 그림 2-4 | 주피터 노트북에서 코랩 실행하기

The image shows a GitHub repository page for 'taehojo/deeplearning' and a Jupyter Notebook viewer. The GitHub page displays the repository structure with files like 'colab', 'data', '.gitignore', 'ch02.ipynb', and 'ch04.ipynb'. The file 'ch02.ipynb' is highlighted with a red box and a '1' and '클릭' (Click) label. The Jupyter Notebook viewer shows the content of 'ch02.ipynb', which includes a title '2장. 딥러닝 핵심 미리보기' and a section '1. 환경 준비'. The 'colab' link is highlighted with a red box and a '2' and '클릭' (Click) label.

2장. 딥러닝 핵심 미리보기

코랩에서 실행하기 → **colab** 2 클릭

나의 첫 딥러닝: '10장 폐암 수술 환자의 생존율 예측' 코드 미리보기

1. 환경 준비

```
In [1]: from tensorflow.keras.models import Sequential # 텐서플로의 케라스 API에서 필요한 함수들을 불러옵니다.
from tensorflow.keras.layers import Dense # 데이터를 다루는데 필요한 라이브러리를 불러옵니다.
import numpy as np
```


2 딥러닝 코드 실행해 보기

● 딥러닝 코드 실행해 보기

2. 해당 노트북 파일이 구글 코랩을 통해 열림

▼ 그림 2-5 | 구글 코랩으로 열기



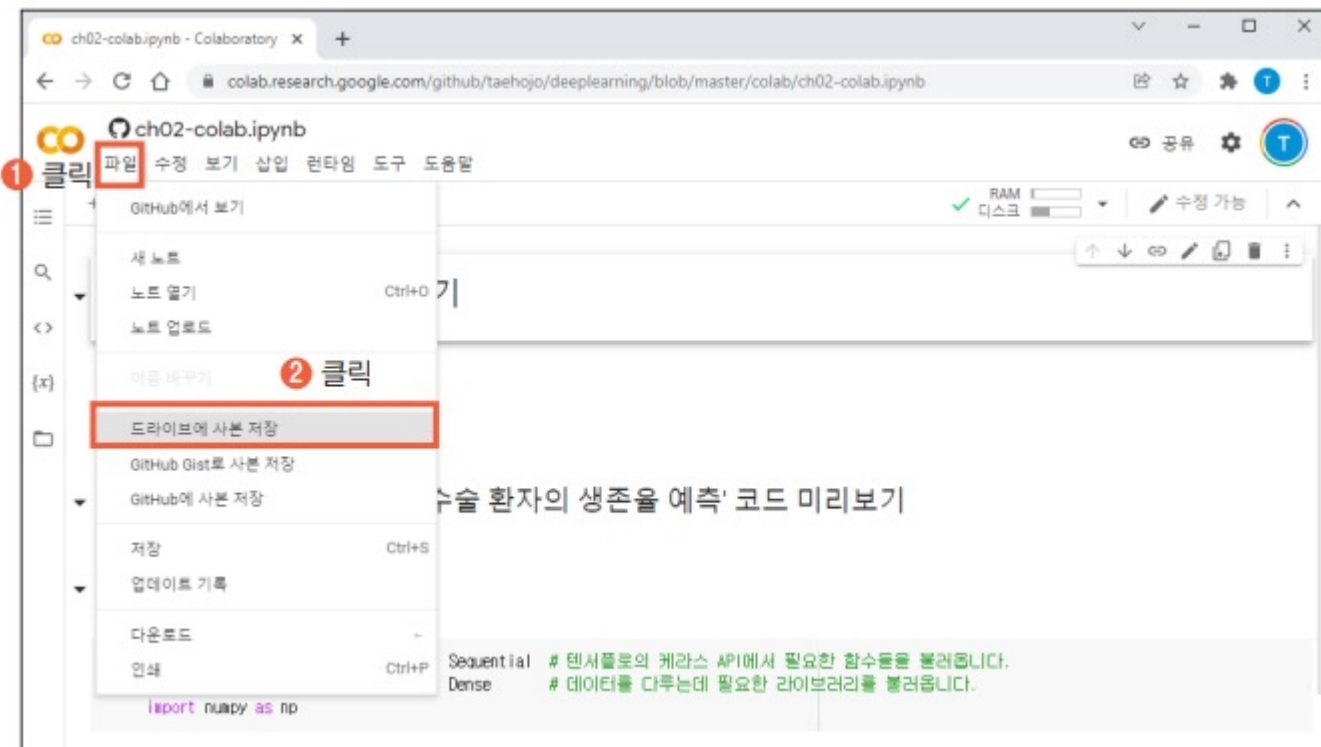
아직은 내 계정에서 오픈한 상태가 아니므로 실행하거나 저장할 수 없음

2 딥러닝 코드 실행해 보기

● 딥러닝 코드 실행해 보기

3. ① 파일 > ② 드라이브에 사본 저장을 선택해 해당 코드의 사본을 내 드라이브에 저장

▼ 그림 2-6 | 나의 구글 계정으로 사본 복사하기



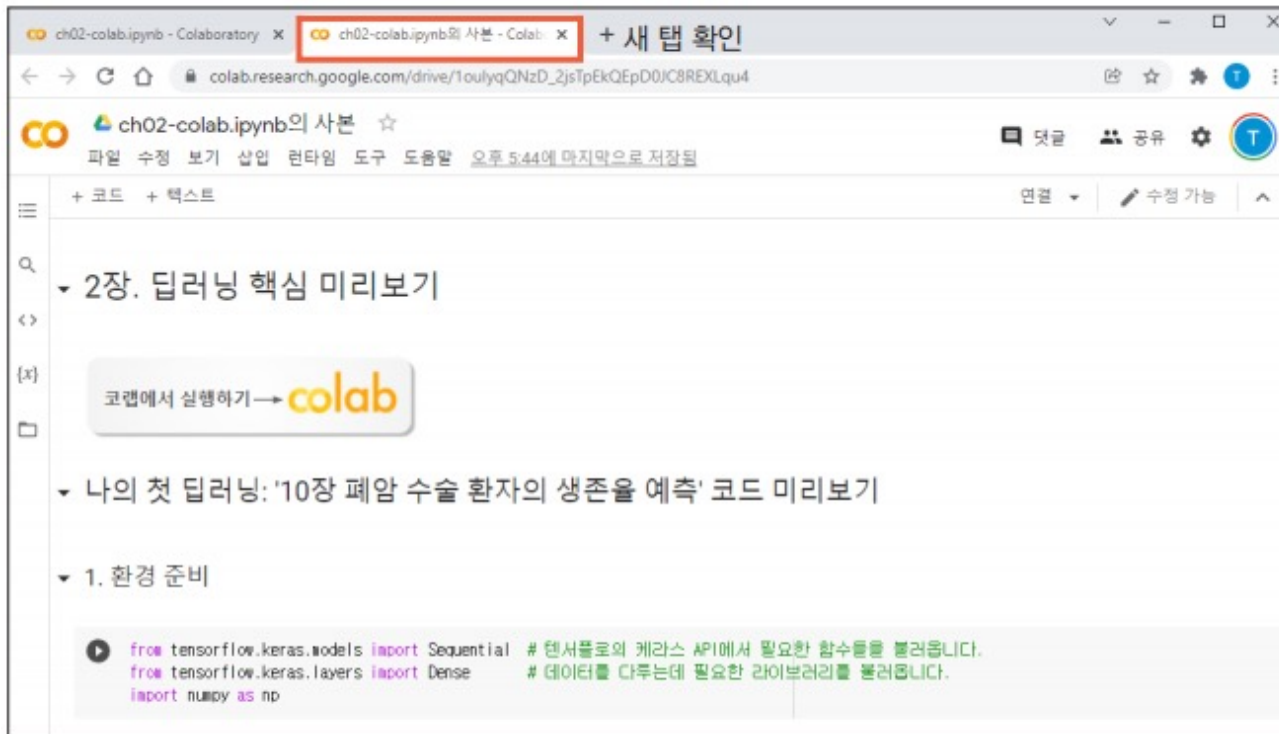
2 딥러닝 코드 실행해 보기

● 딥러닝 코드 실행해 보기

4. 새 탭이 열리며 해당 코드의 사본이 실행되는 것을 확인

이 사본은 나의 구글 계정에서 실행되는 것이므로 이제 코드를 내가 실행하거나 저장할 수 있음

▼ 그림 2-7 | 내 계정으로 사본 복사

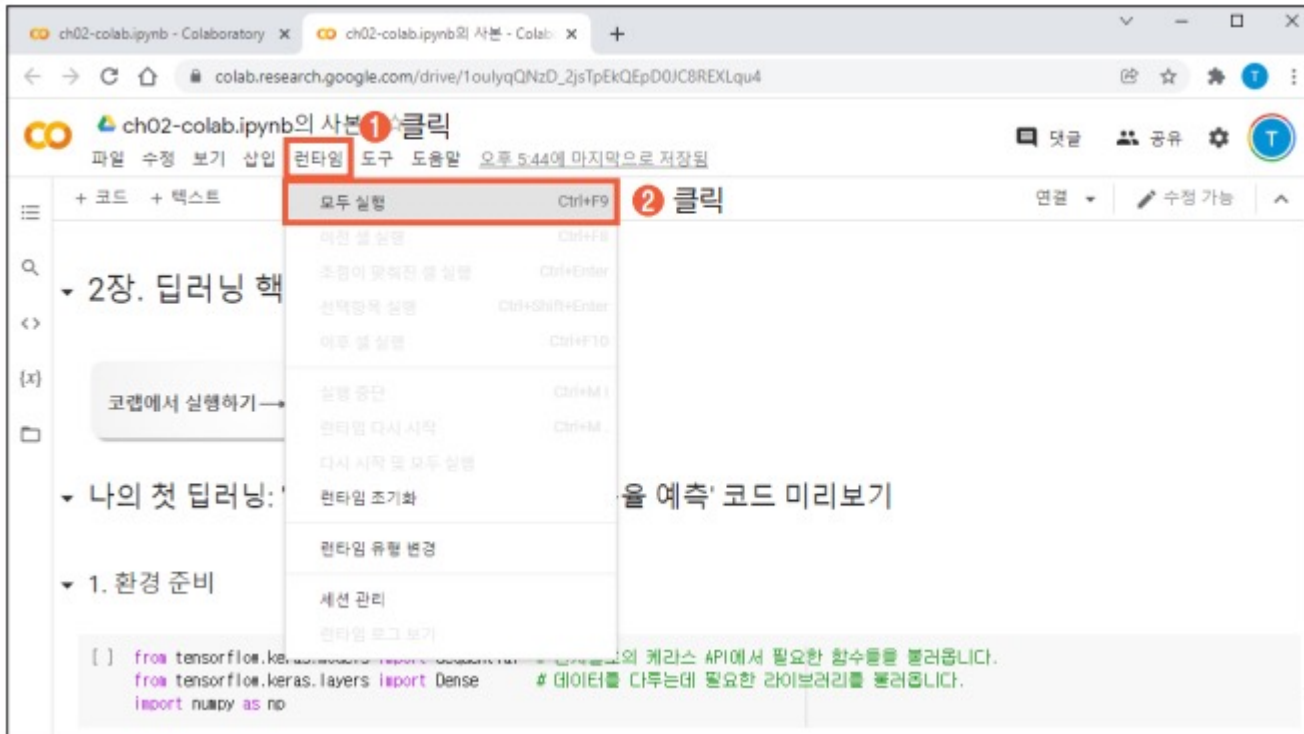


2 딥러닝 코드 실행해 보기

● 딥러닝 코드 실행해 보기

5. 구글 코랩 파일 전체를 한 번에 실행하려면 ❶ 런타임 > ❷ 모두 실행을 선택

▼ 그림 2-8 | 구글 코랩 전체를 한 번에 실행하기





2 딥러닝 코드 실행해 보기



- 딥러닝 코드 실행해 보기

- 코드별로 하나씩 실행하려면 각 코드창 앞의 실행() 버튼을 클릭하면 됨



2 딥러닝 코드 실행해 보기

● 딥러닝 코드 실행해 보기

- 코드창 맨 앞에  ,  아이콘이 차례로 나타나면서 코드가 실행되면 성공
- 실행을 마치면 다음과 같이 ❶ 각 창별 실행 시간이 나타나고 ❷ 실행 결과가 표시



2 딥러닝 코드 실행해 보기

▼ 그림 2-9 | 구글 코랩 실행 결과

1 실행 시간

▼ 나의 첫 딥러닝: '10장 폐암 수술 환자의 생존을 예측' 코드 미리보기

▼ 1. 환경 준비

```
[1] from tensorflow.keras.models import Sequential # 텐서플로의 케라스 API에서 필요한 함수들을 불러옵니다.
from tensorflow.keras.layers import Dense # 데이터를 다루는데 필요한 라이브러리를 불러옵니다.
import numpy as np
```

▼ 2. 데이터 준비

```
[2] !git clone https://github.com/taehojo/data.git # 깃허브에 준비된 데이터를 가져옵니다.

Data_set = np.loadtxt("./data/ThoracicSurgery3.csv", delimiter=",") # 수술 환자 데이터를 불러옵니다.
X = Data_set[:,0:16] # 환자의 진찰 기록을 X로 지정합니다.
y = Data_set[:,16] # 수술 후 사망/생존 여부를 y로 지정합니다.

fatal: destination path 'data' already exists and is not an empty directory.
```

▼ 3. 구조 결정

```
[3] model = Sequential() # 딥러닝 모델의 구조를 결정합니다.
model.add(Dense(30, input_dim=16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

▼ 4. 모델 실행

```
[4] model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy']) # 딥러닝 모델을 실행합니다.
history=model.fit(X, y, epochs=5, batch_size=16)
```

```
Epoch 1/5
30/30 [=====] - 1s 2ms/step - loss: 1.5790 - accuracy: 0.4957
Epoch 2/5
30/30 [=====] - 0s 2ms/step - loss: 0.4954 - accuracy: 0.8511
Epoch 3/5
30/30 [=====] - 0s 2ms/step - loss: 0.4470 - accuracy: 0.8511
Epoch 4/5
30/30 [=====] - 0s 2ms/step - loss: 0.4369 - accuracy: 0.8511
Epoch 5/5
30/30 [=====] - 0s 2ms/step - loss: 0.4368 - accuracy: 0.8511
```

2 실행 결과

2 딥러닝 코드 실행해 보기

- 딥러닝 코드 실행해 보기

- 실행 결과는 매번 실행할 때마다 미세하게 달라짐
- 이것은 첫 가중치를 랜덤하게 정하고 실행을 반복하며, 조금씩 가중치를 수정해 가는 딥러닝의 특성 때문임
- 딥러닝의 동작 원리에 대해서 앞으로 차차 배워 나갈 것



3 딤러닝 개괄하기

3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 지금 불러와 실행한 코드는 10장에서 상세히 다루게 될 폐암 수술 환자의 수술 1년 후 생존율을 예측한 모델
- 먼저 코드를 개괄적으로 살펴보며 딥러닝을 프로그래밍하는 과정에 대한 감을 잡아 보자
- 단 몇 줄로 이루어진 간략한 코드는 다음과 같이 크게 네 부분으로 나뉘어 있음

1. 환경 준비

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
```

딥러닝을 구동하거나 데이터를 다루는 데 필요한 라이브러리들을 불러옵니다.

2. 데이터 준비

```
!git clone https://github.com/taehojo/data.git

Data_set = np.loadtxt("./data/ThoracicSurgery3.csv",
delimeter=",")
X = Data_set[:,0:16]
y = Data_set[:,16]
```

준비된 수술 환자 정보 데이터를 나의 구글 코랩 계정에 저장합니다. 해당 파일을 불러와 환자 상태의 기록에 해당하는 부분을 X로, 수술 1년 후 사망/생존 여부를 y로 지정합니다.



3 딥러닝 개괄하기

● 딥러닝 개괄하기

3. 구조 결정

```
model = Sequential()  
model.add(Dense(30, input_dim=16, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

딥러닝 모델의 구조를 결정합니다.
여기에 설정된 대로 딥러닝을 수행
합니다.

4. 모델 실행

```
model.compile(loss='binary_crossentropy',  
optimizer='adam', metrics=['accuracy'])  
history = model.fit(X, y, epochs=5, batch_size=16)
```

딥러닝 모델을 실행합니다. 앞서 설
정된 구조대로 실행하고 결과를 출
력합니다.

3 딥러닝 개괄하기

● 딥러닝 개괄하기

1. 환경 준비 딥러닝을 구동하는 데 필요한 라이브러리 호출

- 이 책의 모든 코드는 파이썬으로 되어 있음
- 파이썬은 초보자부터 전문가까지 모두에게 애용되는 프로그래밍 언어로, 특히 다양한 플랫폼에서 데이터를 분석하고 딥러닝, 머신 러닝을 구현하는 데 사용
- 파이썬은 풍부한 라이브러리를 가지고 있다는 것이 장점인데, 라이브러리란 특정한 기능을 담은 작은 프로그램들(module, API)을 모아 놓은 것을 의미
- 목적에 따라 라이브러리를 불러오면 다양한 작업을 간단히 진행할 수 있음
- 라이브러리를 불러올 때 사용하는 명령어가 import



3 딥러닝 개괄하기

- 딥러닝 개괄하기
 - 코드의 처음이 다음과 같이 시작

```
from tensorflow.keras.models import Sequential .... ❶  
from tensorflow.keras.layers import Dense .... ❷  
import numpy as np .... ❸
```



3 딥러닝 개괄하기

- 딥러닝 개괄하기

- 라이브러리에 포함된 모듈이 너무 많을 때, 그중 지금 필요한 일부 모듈만 다음과 같이 불러올 수 있음

```
from (라이브러리명) import (함수명)
```



3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 예를 들어 ❶ **from** tensorflow.keras.models **import** Sequential은 텐서플로(tensorflow)의 케라스(keras)라는 API에 있는 모델(model) 클래스로부터 Sequential() 함수를 불러오라는 의미
- 마찬가지로 ❷ **from** tensorflow.keras.layers **import** Dense는 케라스 API의 레이어(layers) 클래스에서 Dense()라는 함수를 불러오라는 의미
- 불러온 라이브러리명이 길거나 같은 이름이 이미 있을 경우 다음과 같이 짧게 줄일

```
import (라이브러리명) as (새로운 이름)
```

- 예를 들어 ❸ **import** numpy **as** np 명령은 아나콘다에 이미 포함되어 있는 넘파이(numpy) 라이브러리를 np라는 짧은 이름으로 불러와 사용할 수 있게 해 줌



3 딥러닝 개괄하기

● 딥러닝 개괄하기

2. 데이터 준비 데이터를 불러와 사용할 수 있도록 준비

- 이제 데이터를 불러와 구글 코랩에서 사용할 수 있도록 준비할 차례
- 데이터는 직접 업로드하는 방법과 깃허브에서 불러오는 방법이 있음
- 우리는 이 책을 위해 깃허브에 준비된 데이터를 내 계정으로 불러오도록 하겠음
- 데이터를 가져오기 위해 실행하는 코드는 다음과 같음

```
!git clone https://github.com/taehojo/data.git
```

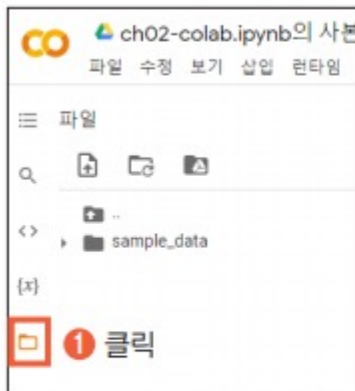
- data라는 폴더가 새로 생기는 것을 확인할 수 있음
- ❶ 폴더 모양의 아이콘을 클릭한 후 ❷ data 폴더를 클릭하면 ❸ 준비된 데이터를 확인할 수 있음



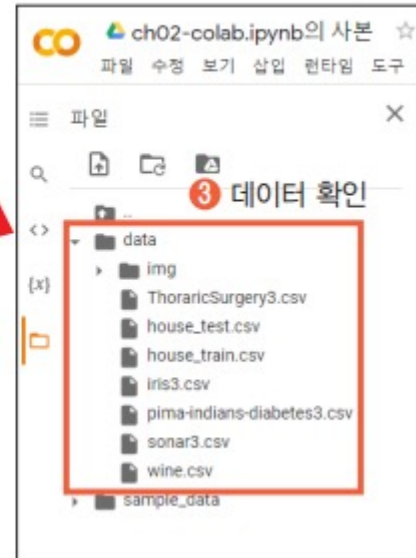
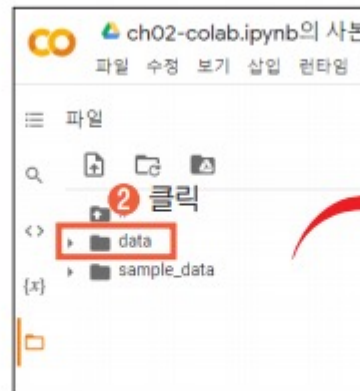
3 딥러닝 개괄하기

▼ 그림 2-10 | 깃허브에서 데이터 가져오기

```
!git clone https://github.com/taehojo/data.git
```





실행 후
→

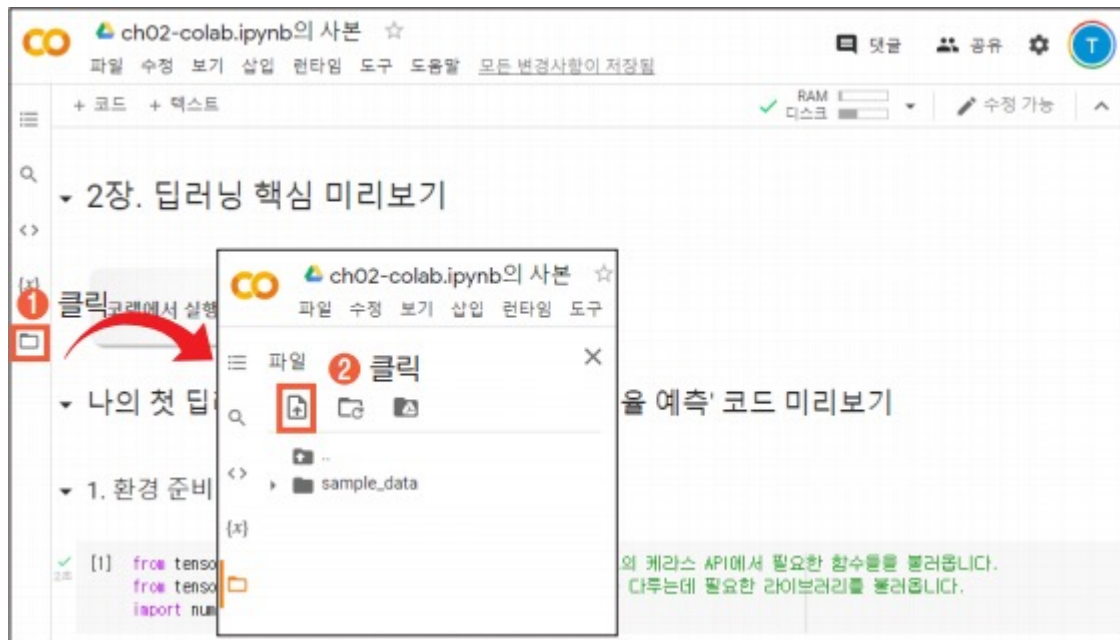


3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 자신이 가지고 있는 파일을 직접 업로드하려면 좌측 하단의 ❶  을 클릭해 파일 관련 메뉴를 열면 됨
- 업로드 아이콘 ❷  을 클릭하면 데이터를 업로드할 수 있음

▼ 그림 2-11 | 파일 업로드하기



3 딥러닝 개괄하기

● 딥러닝 개괄하기

- data 폴더 안에 있는 데이터들은 **./data/데이터명** 형식으로 불러올 수 있음
- 넘파이 라이브러리를 이용해 data 폴더에 있는 csv 파일을 불러오는 부분은 다음과 같음

```
data_set = np.loadtxt("./data/ThoraricSurgery3.csv", delimiter=",")
```

- 폴란드의 브로츠와프 의과대학에서 2013년 공개한 폐암 수술 환자의 수술 전 진단 데이터와 수술 후 생존 결과를 기록한 실제 의료 기록 데이터
- 입력(속성값 17개): 17가지의 환자 상태 (종양의 유형, 폐활량, 호흡 곤란 여부, 고통 정도, 기침, 흡연, 천식 여부 등)
- 출력(판정결과 1개): 수술 후 생존 결과(1=수술 후 생존했음/ 0= 수술 후 사망했음)

3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 넘파이 라이브러리의 loadtxt() 함수를 사용해 'ThoraricSurgery3.csv'라는 외부 데이터셋을 불러왔음
- 머신 러닝에서 알고리즘이나 좋은 컴퓨터 환경만큼 중요한 것이 바로 좋은 데이터를 준비하는 일
- 데이터를 면밀히 관찰하고 효율적으로 다루는 연습을 하는 것이 중요
- 우선은 지금 불러온 ThoraricSurgery3.csv 파일에 관해 좀 더 살펴보자
- ① 먼저 그림 2-12와 같이 data 폴더의 ThoraricSurgery3.csv 파일을 더블클릭
- ② 웹 브라우저 우측에 새로운 공간이 생기며 해당 데이터를 미리 볼 수 있음

3 딥러닝 개괄하기

▼ 그림 2-13 | 폐암 수술 환자의 의료 기록과 1년 후 사망 여부 데이터

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2.88	2.16	1	0	0	0	1	1	3	0	0	0	1	0	60	0
2	2	3.4	1.88	0	0	0	0	0	0	1	0	0	0	1	0	51	0
3	2	2.76	2.08	1	0	0	0	1	0	0	0	0	0	1	0	59	0
4	2	3.68	3.04	0	0	0	0	0	0	0	0	0	0	0	0	54	0
5	2	2.44	0.96	2	0	1	0	1	1	0	0	0	0	1	0	73	1
...
470	2	4.72	3.56	0	0	0	0	0	0	1	0	0	0	1	0	51	0

↑
샘플 수
(환자 수: 470명)

↑
속성
(의료 기록: 16가지)

↑
클래스
(사망: 0/생존: 1)



3 딤러닝 개괄하기

● 딤러닝 개괄하기

- 가로줄 한 행이 한 사람의 환자로부터 기록된 정보를 의미
- 총 470행이므로 환자 470명에 대한 정보
- 한 행에는 17개의 숫자가 들어 있음
- 이는 환자마다 17개의 정보를 순서에 맞추어 정리했다는 의미
- 앞의 정보 16개는 종양의 유형, 폐활량, 호흡 곤란 여부, 고통 정도, 기침, 흡연, 천식 여부 등 16가지 환자 상태를 조사해서 기록해 놓은 것
- 마지막 17번째 정보는 수술 1년 후의 생존 결과
- 1은 수술 후 생존했음을, 0은 수술 후 사망했음을 의미



3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 이번 프로젝트의 목적은 1번째 항목부터 16번째 항목까지 이용해서 17번째 항목, 즉 수술 1년 후의 생존 또는 사망을 맞추는 것
- 1번째 항목부터 16번째 항목까지 **속성(attribute)**이라 하고, 정답에 해당하는 17번째 항목을 **클래스(class)**라고 함
- 클래스는 앞서 이야기한 '이름표'에 해당
- 딥러닝을 위해서는 속성과 클래스를 서로 다른 데이터셋으로 지정해 주어야 함



3 딥러닝 개괄하기

- 딥러닝 개괄하기

- 먼저 속성으로 이루어진 데이터셋을 X라는 이름으로 만들어 줌

```
X = Data_set[:,0:16]
```



3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 파이썬은 숫자를 1부터 세지 않고 0부터 셈
- 범위를 정할 경우 콜론(:) 앞의 숫자는 범위의 맨 처음을 의미하고, 콜론(:) 뒤의 숫자는 이 숫자가 가리키는 위치 '바로 앞'이 범위의 마지막이라는 의미
- 쉼표(,)를 기준으로 앞은 행(샘플), 뒤는 열(속성)의 범위가 입력
- 예를 들어[:,0:16]은 모든 행의 1번째 열부터 16번째 열까지 가져오라는 의미



3 딥러닝 개괄하기

- 딥러닝 개괄하기

- 다음으로 17번째 줄에 위치한 클래스를 따로 모아 데이터셋 y로 지정

```
y = Data_set[:,16]
```

- 보통 집합은 대문자로, 원소는 소문자로 표시
- X에는 여러 개의 속성이 담기기 때문에 대문자 X로, y는 클래스 하나의 원소만 담기기 때문에 소문자 y로 썼음



3 딥러닝 개괄하기

● 딥러닝 개괄하기

3. 구조 결정 어떤 딥러닝 구조를 만들 것인가

- 앞서 우리는 딥러닝을 실행시키기 위해 텐서플로를 불러왔음
- 텐서플로는 구글에서 만든 딥러닝 전용 라이브러리
- 텐서플로를 이용하면 여러 가지 알고리즘을 활용해 다양한 딥러닝 작업을 할 수 있지만, 사용법이 쉽지 않다는 단점이 있음

▼ 그림 2-14 | 텐서플로(<https://www.tensorflow.org>)





3 딥러닝 개괄하기

- 딥러닝 개괄하기

- 이를 해결해 주기 위해 개발된 것이 케라스(Keras)

▼ 그림 2-15 | 케라스(<https://keras.io>)

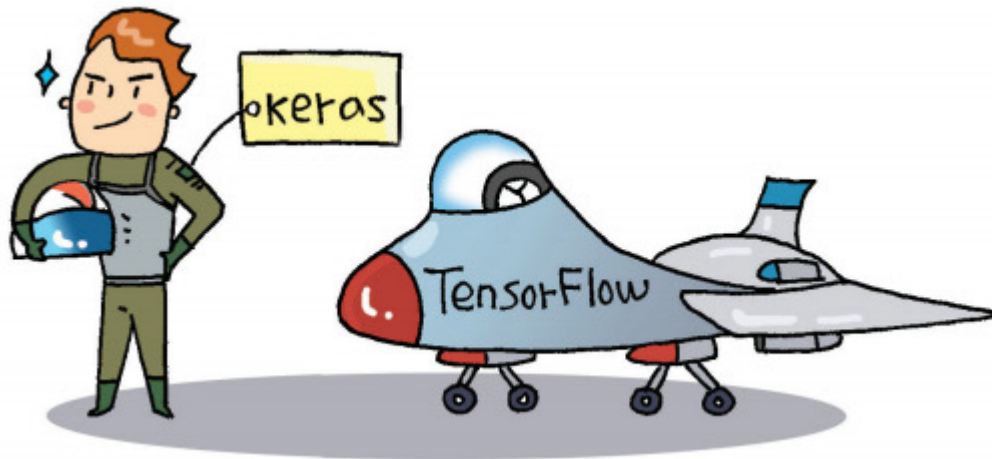


3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 텐서플로가 목적지까지 이동시켜 주는 비행기라면 케라스는 조종사에 해당
- 케라스를 활용하면 딥러닝의 거의 모든 작업을 쉽게 처리할 수 있음

▼ 그림 2-16 | 텐서플로와 케라스의 관계





3 딥러닝 개괄하기

- 딥러닝 개괄하기

- 불러온 예제에서 케라스를 어떻게 활용했는지 알아보자

```
model = Sequential() ..... ❶  
model.add(Dense(30, input_dim=16, activation='relu')) ..... ❷  
model.add(Dense(1, activation='sigmoid')) ..... ❸
```



3 딥러닝 개괄하기

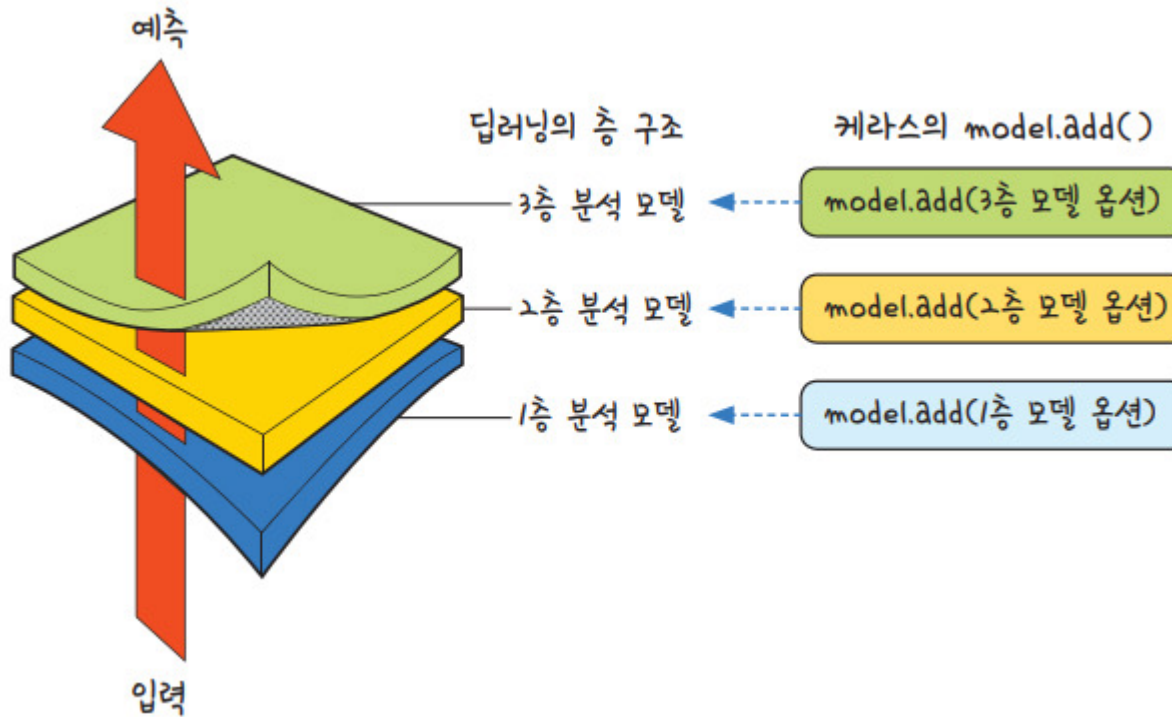
● 딥러닝 개괄하기

- ❶ 먼저 앞서 불러온 Sequential() 함수를 model로 선언
- 앞으로 상세히 다루겠지만, 딥러닝은 그림 2-17과 같이 여러 층이 쌓여 있는 구조
- 준비된 데이터가 입력되는 입력층에 이어 첫 번째 작업을 진행하는 1층, 두 번째 작업을 하는 2층... 이런 식으로 출력 결과가 나오는 출력층까지 여러 개의 층이 각자 자신이 맡은 일을 하면서 앞뒤로 정보를 주고받음
- 케라스의 Sequential() 함수는 딥러닝의 한 층 한 층을 ❷ model.add()라는 함수를 사용해 간단히 추가시켜 줌
- 여기서는 ❷ 와 ❸ , 두 개의 층을 쌓았음
- Model.add() 함수를 한 줄 추가하는 것으로 필요한 만큼 내부의 층을 만들 수 있음



3 딥러닝 개괄하기

▼ 그림 2-17 | 딥러닝의 층 구조와 케라스





3 딥러닝 개괄하기

- 딥러닝 개괄하기

- 각 `model.add()` 함수 안에는 케라스 API의 `layers` 클래스에서 불러온 `Dense()` 함수가 포함되어 있음
- `Dense`는 '밀집한, 뻑뻑한'이란 뜻으로, 여기서는 각 층의 입력과 출력을 촘촘하게 모두 연결하라는 것



3 딥러닝 개괄하기

● 딥러닝 개괄하기

- 이제 두 가지를 더 알면 됨
- 첫째, 좋은 딥러닝 모델을 만들려면 몇 개의 층으로 쌓아 올려야 하는가?
- 둘째, Dense 함수 안에 있는 숫자와 설정의 의미는 무엇이며, 어떻게 정해야 하는가?
- 딥러닝을 설계한다는 것은 결국 몇 개의 층을 어떻게 쌓을지, Dense 외에 어떤 층을 사용할지, 내부의 변수들을 어떻게 정해야 하는지 등에 대해 고민하는 것
- 대개 어떤 데이터를 가지고 무엇을 할 것인지에 따라 **딥러닝의 설계**가 결정
- 각 설정과 변수의 의미를 알고 이것을 자유롭게 구성할 수 있는지가 딥러닝을 잘 다루는지 여부를 결정하는 것
- 이 책에서 배울 내용도 결국 이것
- Dense() 함수의 내부에 쓰인 각 설정의 의미들은 책의 진도가 나감에 따라 앞으로 하나씩 배우게 될 것



3 딥러닝 개괄하기

- 딥러닝 개괄하기

4. 모델 실행 만든 딥러닝을 실행시키고 결과 확인

- 만들어 놓은 모델을 실행시키는 부분

```
model.compile(loss=binary_crossentropy, optimizer='adam',  
metrics=['accuracy']) ..... ❶  
history = model.fit(X, y, epochs=5, batch_size=16) ..... ❷
```



3 딥러닝 개괄하기

● 딥러닝 개괄하기

- **model.compile()** 함수는 앞서 만든 model의 설정을 그대로 실행하라는 의미
- 함수 내부에 **loss, optimizer, metrics** 등 키워드들이 들어 있음
- 이것은 앞 단계에서 만들어진 딥러닝 구조를 어떤 방식으로 구동시키고 어떻게 마무리할 것인지와 관련된 옵션들인데, 둘째 마당과 셋째 마당에서 자세히 배울 것
- 딥러닝은 여러 층이 쌓여 만들어진다는 설명을 이미 한 바 있음
- 딥러닝의 기본 방식은 이 층들을 한 번만 통과하는 것이 아니라 위아래로 여러 차례 오가며 최적의 모델을 찾는 것
- 몇 번을 오갈 것인지, 그리고 한 번 오갈 때 몇 개의 데이터를 사용할 것인지를 정하는 함수가 **model.fit()** 함수



4 이제부터가 진짜 딤러닝?

4 이제부터가 진짜 딥러닝?

● 이제부터가 진짜 딥러닝?

- 지금까지 딥러닝을 위한 작업 환경을 만들고, 딥러닝 모델을 실행해 보면서 학습 목표를 파악했음
- 딥러닝을 위한 학습에는 단순한 **파이썬 프로그래밍**뿐 아니라 **선형 회귀, 로지스틱 회귀 등 기초 통계학 개념**들도 필요함
- 이러한 설명에는 필연적으로 수학 개념이 따라오게 되어 있음
- 예전에 배웠지만 잠시 잊고 지냈던 분들을 위해 '**3장. 딥러닝을 위한 기초 수학**'을 다음 장에 준비했음
- 물론, 수학에 자신이 있다면 둘째 마당으로 직행해도 됨
- 만일 예전에 배웠던 것들을 한 번 더 확인하고 싶다면 다음 장에 이어지는 딥러닝을 위한 기초 수학 편을 통해 필요한 개념들을 정리하고 넘어가길 권함