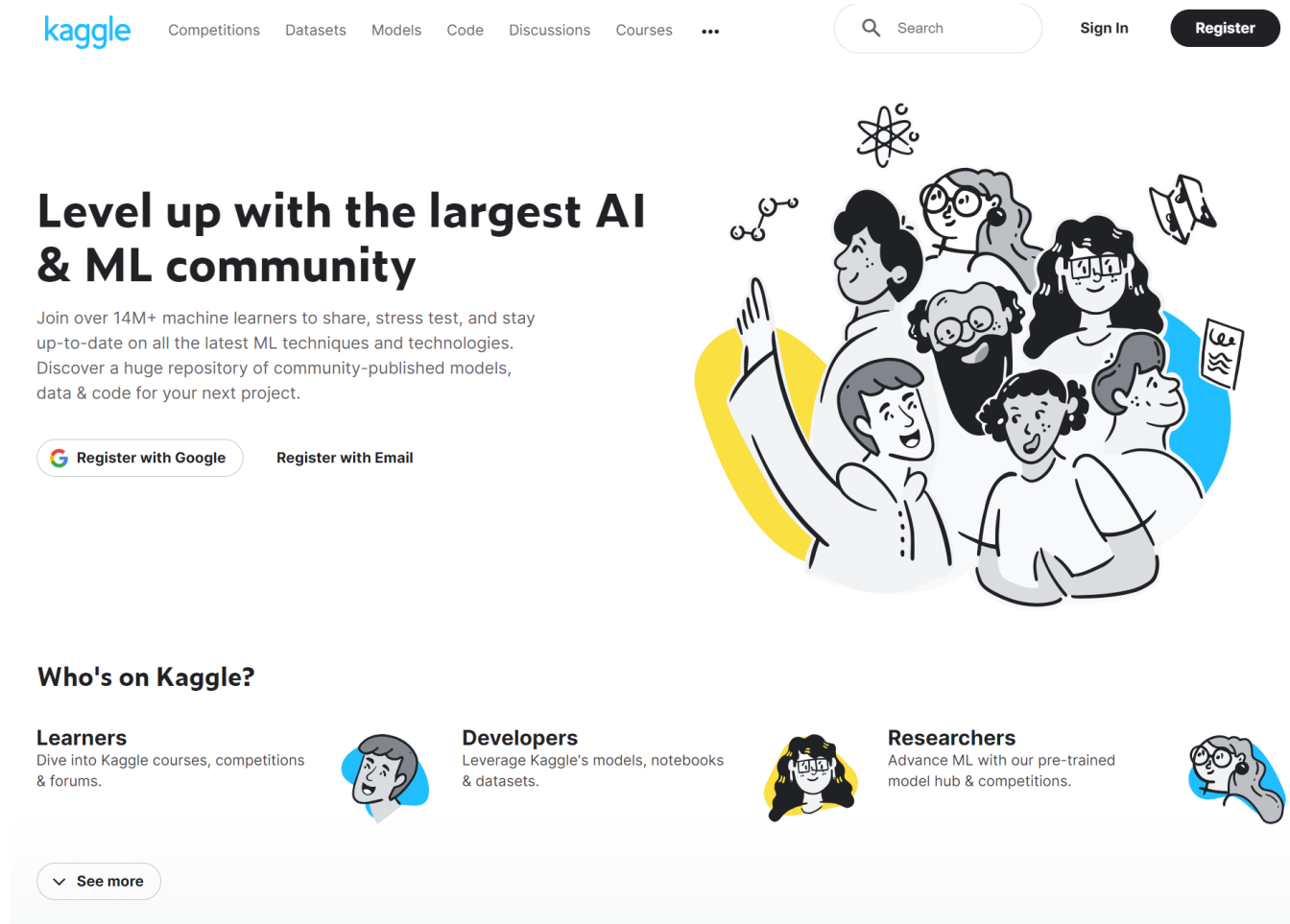


Introduction to AI (lab)

HyungGi Jo
Div. of Electronics
Jeonbuk National University

Kaggle

➤ Machine Learning dataset, model, competition, etc.



The image is a screenshot of the Kaggle homepage. At the top, there is a navigation bar with the Kaggle logo on the left, followed by links for Competitions, Datasets, Models, Code, Discussions, and Courses. On the right side of the navigation bar are a search bar, a 'Sign In' link, and a 'Register' button. Below the navigation bar, the main heading reads 'Level up with the largest AI & ML community'. Underneath this heading is a paragraph: 'Join over 14M+ machine learners to share, stress test, and stay up-to-date on all the latest ML techniques and technologies. Discover a huge repository of community-published models, data & code for your next project.' Below the text are two buttons: 'Register with Google' and 'Register with Email'. To the right of the text is a large, colorful illustration of a diverse group of people, some wearing glasses, with various icons like a brain, a lightbulb, and a document floating around them. Below the illustration, there is a section titled 'Who's on Kaggle?' which features three columns. The first column is for 'Learners' with the description 'Dive into Kaggle courses, competitions & forums.' and a small icon of a person. The second column is for 'Developers' with the description 'Leverage Kaggle's models, notebooks & datasets.' and a small icon of a person. The third column is for 'Researchers' with the description 'Advance ML with our pre-trained model hub & competitions.' and a small icon of a person. At the bottom left of the 'Who's on Kaggle?' section is a button that says 'See more'.

kaggle Competitions Datasets Models Code Discussions Courses ... Search Sign In Register

Level up with the largest AI & ML community

Join over 14M+ machine learners to share, stress test, and stay up-to-date on all the latest ML techniques and technologies. Discover a huge repository of community-published models, data & code for your next project.

Register with Google Register with Email

Who's on Kaggle?

Learners

Dive into Kaggle courses, competitions & forums.

Developers

Leverage Kaggle's models, notebooks & datasets.

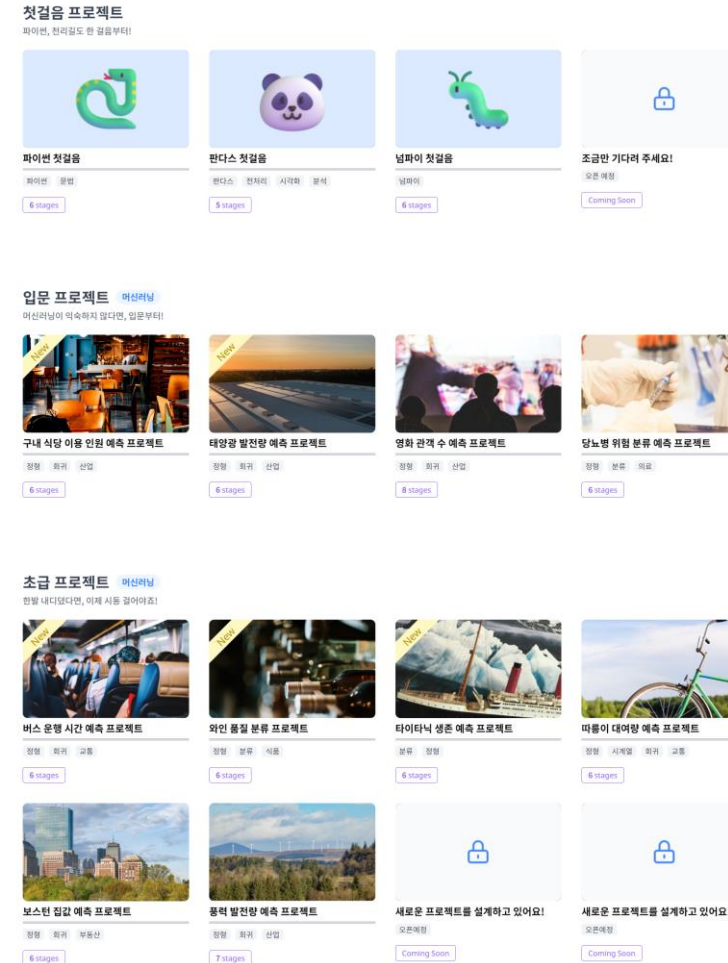
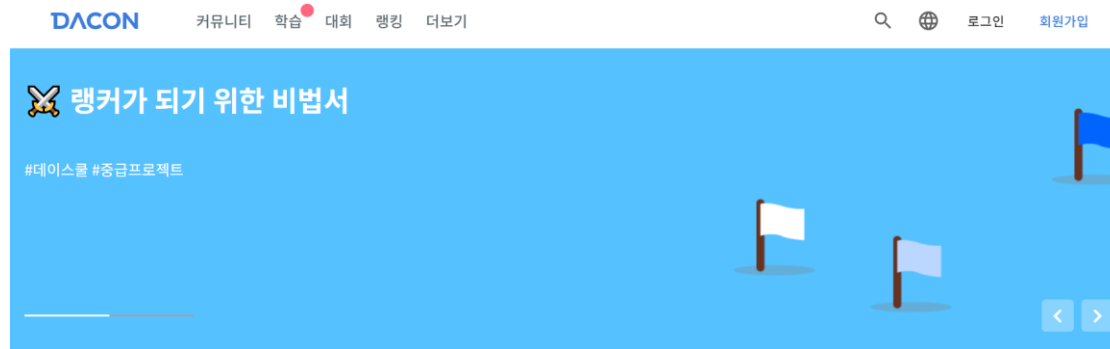
Researchers

Advance ML with our pre-trained model hub & competitions.

See more

Dacon

➤ 인공지능 빅데이터 대회, 프로젝트, 커뮤니티

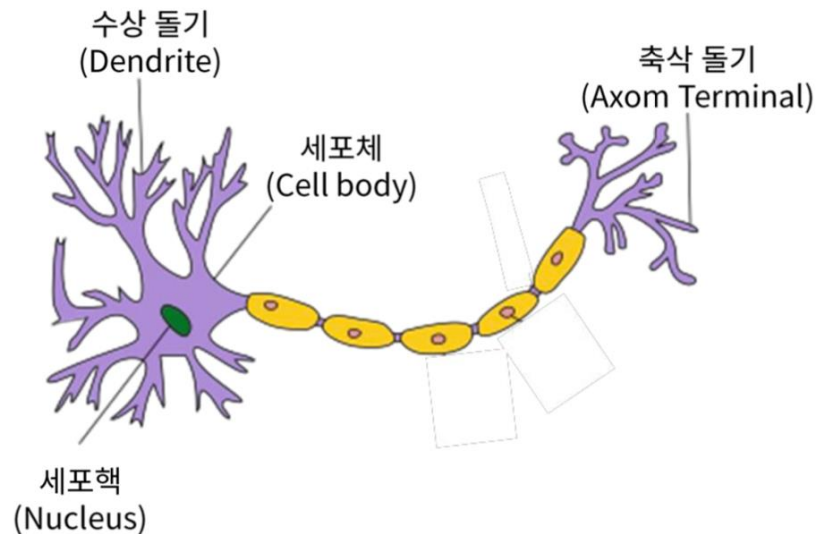


Artificial Neural Network

➤ 인공신경망 – 생물학적 신경체계(기본 단위: 뉴런)를 모방

▪ Biological Neuron

- 수상 돌기로 신호가 입력
- 세포체에서는 입력 받은 신호를 처리하고 Threshold 이상이면 축삭 돌기로 전달

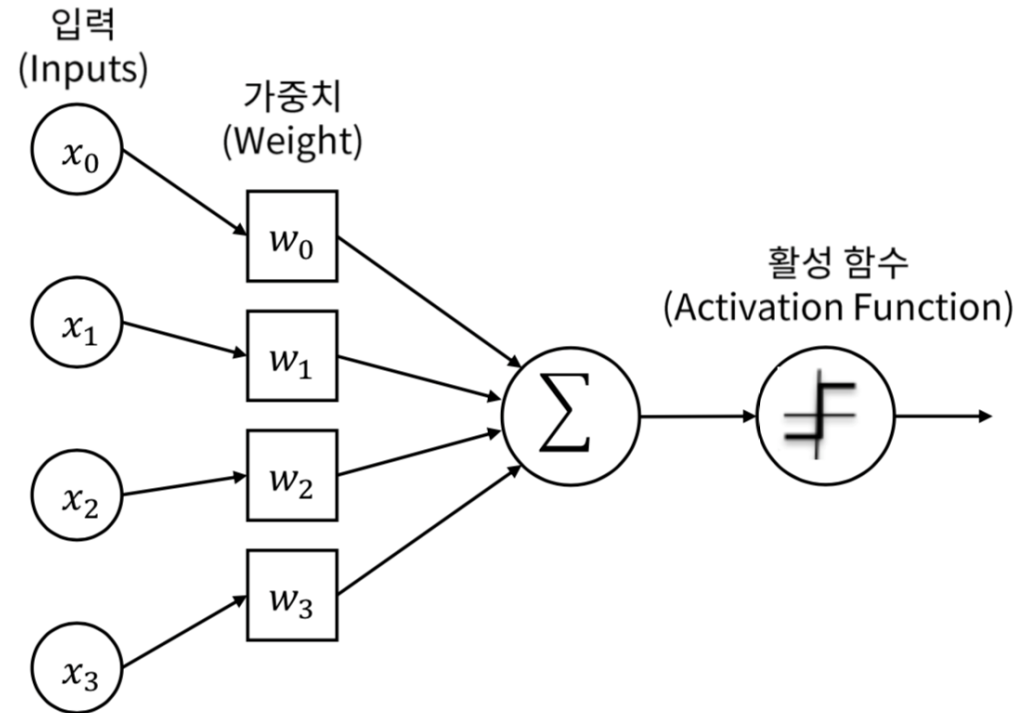


Artificial Neural Network

➤ 인공신경망 – 생물학적 신경체계(기본 단위: 뉴런)를 모방

▪ Artificial Neuron

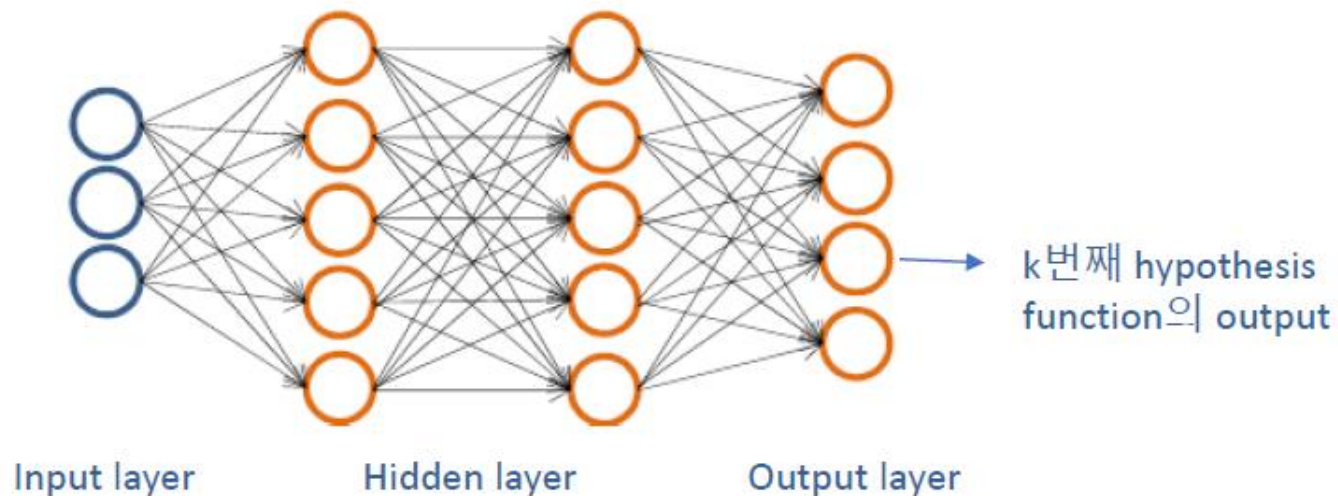
- 여러 뉴런으로부터 신호를 받아 처리하여 하나의 Output 생성 및 여러 뉴런으로 전달



Neural Network

➤ 뉴럴네트워크를 이용한 다중 분류 (multi-class classification)

- Input layer node 수는 feature의 개수
- Output layer node의 수는 Class의 개수와 같음
- Training Samples $\{(x^i, y^i), i = 1, \dots, m\}, y^i \in \mathbf{R}^k$



Neural Network

➤ 뉴럴네트워크를 이용한 다중 분류 (multi-class classification)

▪ Label의 구성

- 원-핫 인코딩(one-hot encoding)
- 분류하는 Class의 수가 4일 때, y^i 는 아래 Vector 중 하나

$$y^{(i)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

▪ e.g. 이미지 분류 (image classification) 문제



$$h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Pedestrian

Car

Motorcycle

Tensorflow 기초

- Neural Network를 학습시키기 위해서는 multi-core processing이 필수
 - 파라미터의 개수
- <https://www.tensorflow.org/?hl=ko>

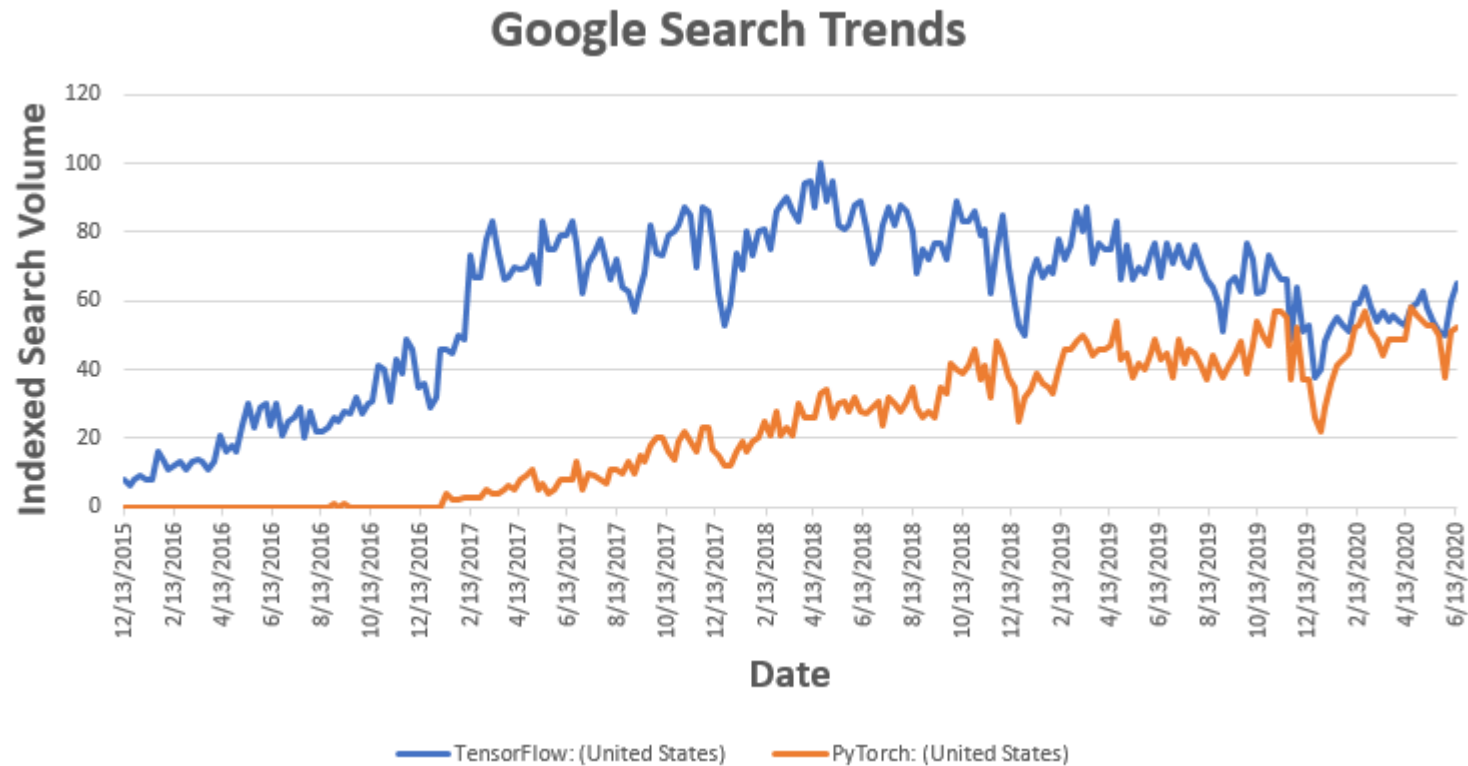


Specifications	Intel® Core™ i9-9960X X-series Processor	NVIDIA GeForce® RTX™ 2080 Ti
Base Clock Frequency	3.1 GHz	1.35 GHz
Cores	16 (32 threads)	4352
Memory Bandwidth	79.47 GB/s	616 GB/s
Floating-Point Calculations	1290 GFLOPS	13400 GFLOPS
Cost	~ \$1700.00	~ \$1100.00

Tensorflow 기초

➤ Tensorflow vs. Pytorch

- Tensorflow : Theano 기반, Google, 산업 친화적 (머신러닝 엔지니어)
- Pytorch: Torch 기반, Facebook, 연구 친화적 (딥러닝 연구자)



Tensorflow 기초

Tensor (텐서)

- Scalar : rank 0 텐서
- Vector : rank 1 텐서
- Matrix : rank 2 텐서
- ...


▼ 텐서플로에서 텐서 만들기


```
[ ] a = np.array([1, 2, 3], dtype=np.int32)
    b = [4, 5, 6]
```

```
t_a = tf.convert_to_tensor(a)
t_b = tf.convert_to_tensor(b)
```

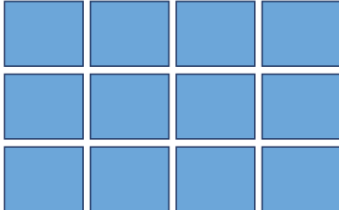
```
print(t_a)
print(t_b)
```

```
tf.Tensor([1 2 3], shape=(3,), dtype=int32)
tf.Tensor([4 5 6], shape=(3,), dtype=int32)
```

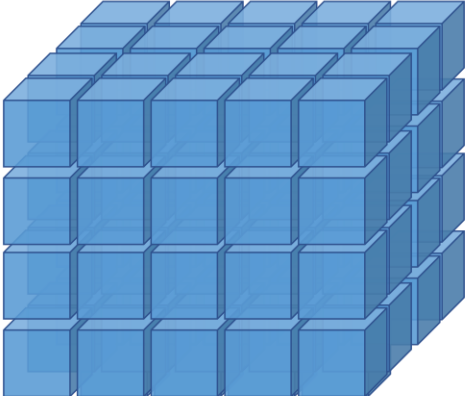
Rank 0: 
(scalar)

Rank 1: 
(vector)

Rank 2: (matrix)



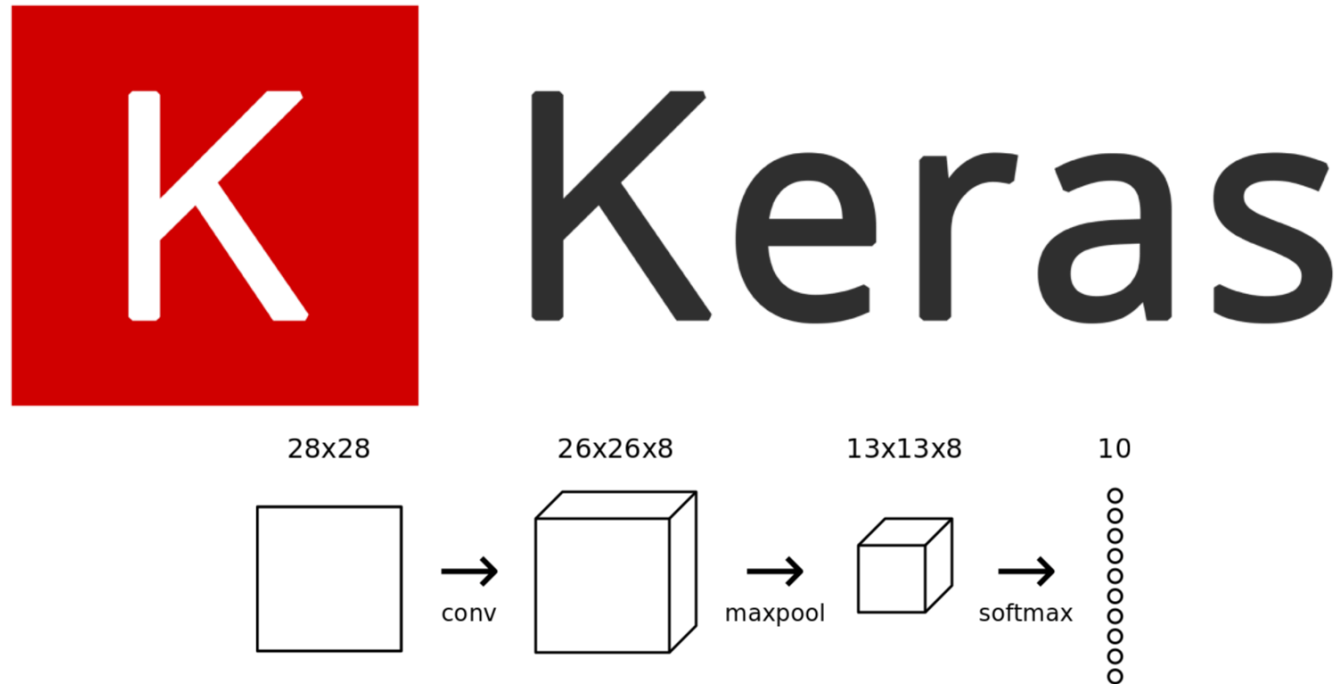
Rank 3:



Tensorflow 기초

➤ Tensorflow(Keras)를 사용한 MNIST

- Keras <https://keras.io/ko/>
- 파이썬 딥러닝 라이브러리 (Tensorflow를 사용한 API)



Tensorflow 기초

시작하기: 30초 케라스

케라스의 주요 데이터 구조는 **model**,로 레이어를 조직하는 방식입니다. 가장 간단한 종류의 모델인 **Sequential** 모델은 레이어를 선형적으로 쌓습니다. 보다 복잡한 구조를 만드려면, **Keras functional API**를 사용하여 레이어로 임의의 그래프를 구축하면 됩니다.

Sequential 모델입니다:

```
from keras.models import Sequential

model = Sequential()
```

.add()를 통해 레이어를 간단하게 쌓을 수 있습니다:

```
from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

모델이 마음에 드신다면, **.compile()**로 학습 과정을 조정하십시오:

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

필요한 경우 한 걸음 나아가 옵티마이저를 조정할 수 있습니다. 케라스의 주요 철학중 하나는 사용자가 필요한 작업에 대해서는 완전한 제어권을 가질 수 있도록 하되 간결성을 유지하는 것입니다 (제어권의 궁극적인 형태로는 소스코드의 간편한 확장성이 있습니다).

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(lr=0.01, momentum=0.9, nesterov=True))
```

배치의 형태로 트레이닝 데이터에 대한 반복작업을 수행할 수 있습니다:

```
# x_train and y_train are Numpy arrays --just like in the Scikit-Learn API.
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

혹은, 모델에 배치를 수동으로 전달할 수도 있습니다:

```
model.train_on_batch(x_batch, y_batch)
```

코드 한 줄로 모델의 성능을 평가해 보십시오:

```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

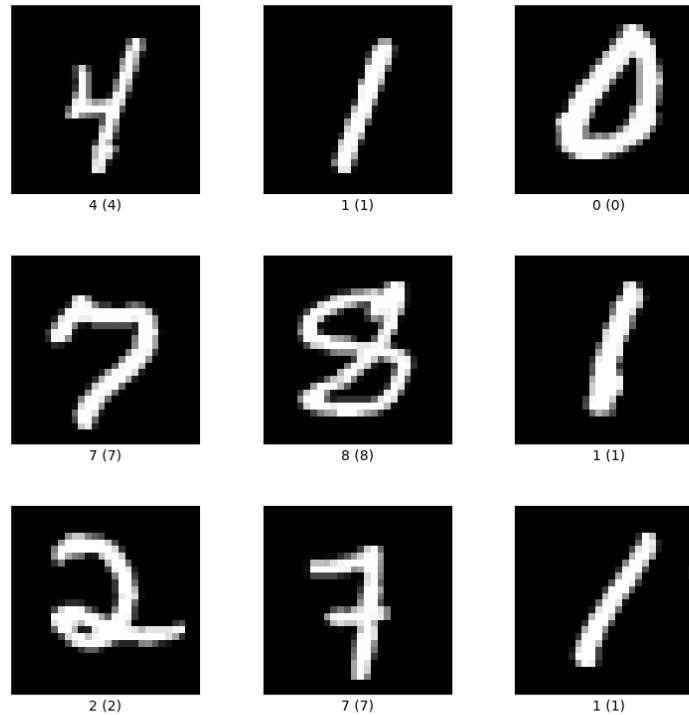
혹은 새로운 데이터에 대해서 예측 결과를 생성해 보십시오:

```
classes = model.predict(x_test, batch_size=128)
```

질문에 대답하는 시스템, 이미지 분류 모델, 신경망 튜링 기계나 그 외의 다른 모델도 이처럼 빠르게 만들 수 있습니다. 딥러닝의 기본이 되는 아이디어가 간단하데 그 실행이 복잡할 이유가 어디 있겠습니까?

Tensorflow 기초

- <https://www.tensorflow.org/tutorials/quickstart/beginner?hl=ko>
- Keras를 사용한 MNIST 데이터셋 Image Classification 수행



Regression

➤ Boston Housing 보스턴 집값 예측

- 1978년, 주택 가격에 가장 큰 영향을 미치는 요인이 깨끗한 공기라는 연구결과 발표
- 주택 가격 변동에 영향을 미치는 여러 요인을 모아서 환경과 주택 가격의 변동을 보여주는 데이터셋 구축
- 머신러닝 Regression 문제에 널리 알려진 유명 데이터셋이 됨

Most expensive U.S. rental markets

For a two-bedroom apartment

Market	Annual change	Median rent
San Francisco, CA	12%	\$3,930
New York, NY	27%	\$3,400
Boston, MA	26%	\$3,150
Miami, FL	24%	\$3,100
Washington, DC	15%	\$3,010
Los Angeles, CA	9%	\$2,940
San Diego, CA	21%	\$2,900
San Jose, CA	8%	\$2,870
Fort Lauderdale, FL	28%	\$2,810
Oakland, CA	10%	\$2,770

+ Show 20 more

Source: Zumper. As of Jan. 2022.

make it

Regression

➤ Boston Housing 보스턴 집값 예측

- 데이터 불러오기
- 총 샘플의 개수 506개, Column의 수 14개 : 13개 속성과 1개의 Target value

```
import pandas as pd
df = pd.read_csv("housing.csv", delim_whitespace=True, header=None)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

Regression

➤ Boston Housing 보스턴 집값 예측

x

506 행 13 열

CRIM per capita crime rate by town

ZN proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS proportion of non-retail business acres per town

CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

NOX nitric oxides concentration (parts per 10 million)

RM average number of rooms per dwelling

AGE proportion of owner-occupied units built prior to 1940

DIS weighted distances to five Boston employment centres

RAD index of accessibility to radial highways

TAX full-value property-tax rate per \$10,000

PTRATIO pupil-teacher ratio by town

B $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town

LSTAT % lower status of the population

y

506 행 1 열

target (MEDV) Median value of owner-occupied homes in \$1000's

BostonHousing 데이터 설명

[01]	CRIM	자치시(town) 별 1인당 범죄율
[02]	ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03]	INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04]	CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05]	NOX	10ppm 당 농축 일산화질소
[06]	RM	주택 1가구당 평균 방의 개수
[07]	AGE	1940년 이전에 건축된 소유주택의 비율
[08]	DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09]	RAD	방사형 도로까지의 접근성 지수
[10]	TAX	10,000 달러 당 재산세율
[11]	PTRATIO	자치시(town)별 학생/교사 비율
[12]	B	$1000(B_k - 0.63)^2$, 여기서 B_k 는 자치시별 흑인의 비율을 말함.
[13]	LSTAT	모집단의 하위계층의 비율(%)
[14]	MEDV	본인 소유의 주택가격(중앙값) (단위: \$1,000)

이미지 출처: http://dator.co.kr/?vid=ctg258&mid=textyle&document_srl=1721307

Regression

- Boston Housing 보스턴 집값 예측
 - 데이터 읽고 텐서로 변환, target value : 13번째 column

1. 데이터 준비

```
import pandas as pd
df = pd.read_csv("housing.csv", delim_whitespace=True, header=None)
```

✓ 0.3s

df

✓ 0.6s

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

```
X = df.drop(columns=[13]).values
y = df[13].values
```

✓ 0.4s

```
X_tensor = torch.tensor(X).float()
y_tensor = torch.tensor(y[:, np.newaxis]).float()
```

✓ 0.4s

Regression

➤ Boston Housing 보스턴 집값 예측

2. 뉴럴네트워크 모델 생성

```
model = nn.Sequential(  
    nn.Linear(13, 16),  
    nn.Tanh(),  
    nn.Linear(16, 8),  
    nn.Tanh(),  
    nn.Linear(8, 1)  
)
```

✓ 0.2s

3. 손실함수 및 최적화 기법 정의

```
loss_func = nn.MSELoss()  
optimizer = optim.Adam(model.parameters(), lr=0.0001)
```

✓ 0.5s

4. 학습 iterations

```
for i in range(3000):  
    optimizer.zero_grad()  
  
    output = model(X_tensor)  
    loss = loss_func(output, y_tensor)  
  
    loss.backward()  
    optimizer.step()  
  
    if i%1000 == 0:  
        print(loss.item())
```

✓ 1.5s

```
427.4305419921875  
395.08233642578125  
365.03802490234375
```

Regression

➤ Boston Housing 보스턴 집값 예측

5. 모델 평가

```
output = model(X_tensor)
err = torch.mean(torch.abs(output - y_tensor))
print(err)
```

✓ 0.4s

```
tensor(15.9079, grad_fn=<MeanBackward0>)
```

6. 모델 저장 및 불러오기

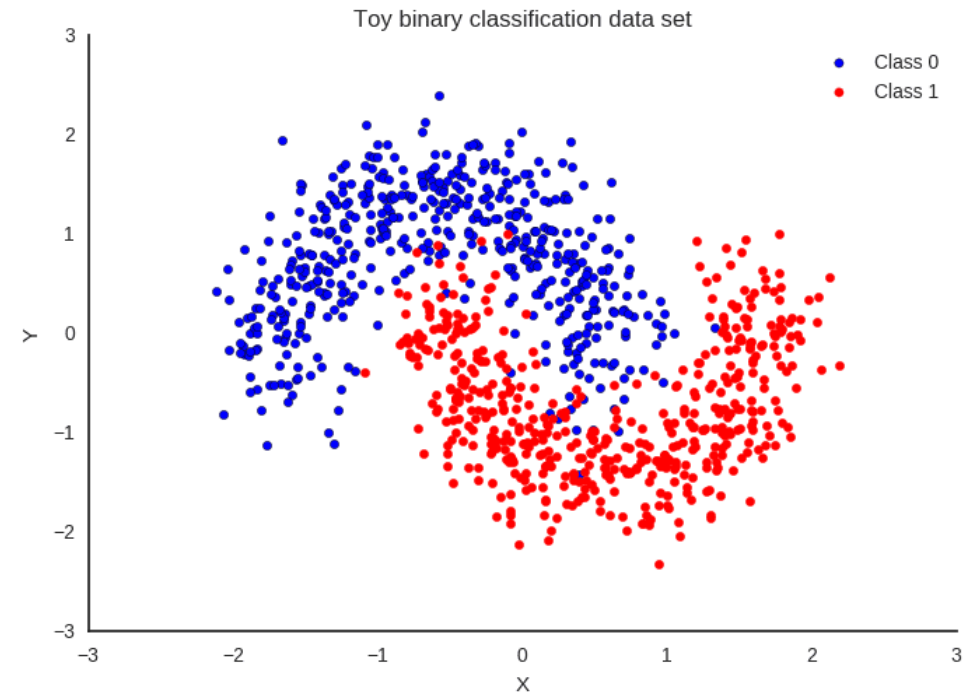
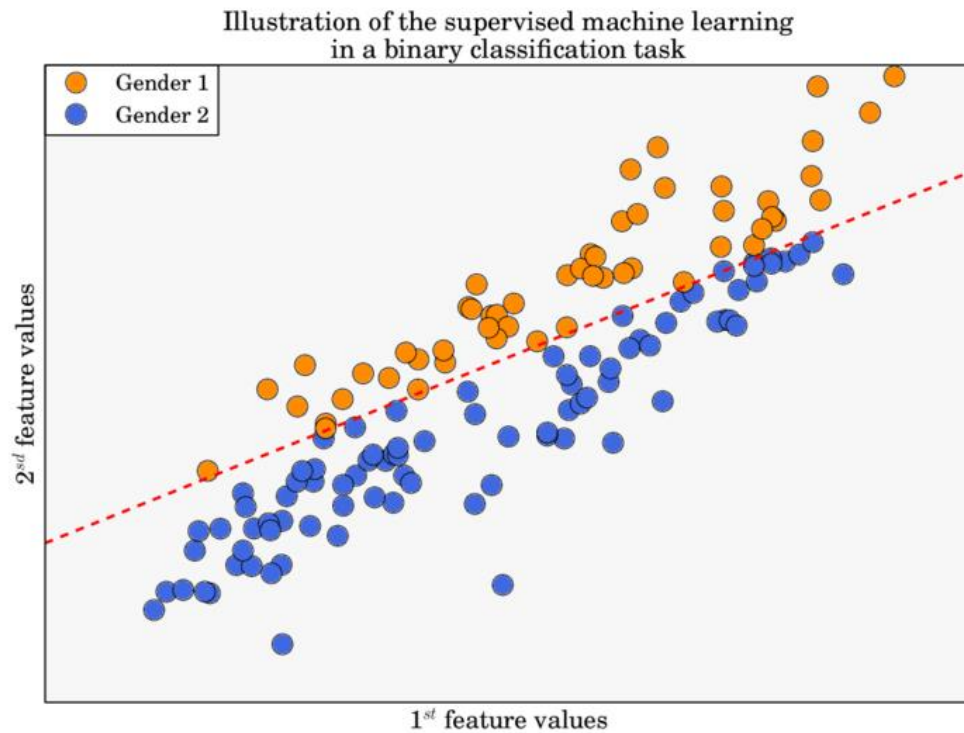
```
torch.save(model, "model.pth")
```

✓ 0.7s

NN: Binary Classification

➤ Binary Classification

- 입력 데이터를 0(False) 또는 1(True)로 분류하는 문제



NN: Binary Classification

➤ Neural Network를 이용한 이진 분류

- 피마 인디언 데이터셋
- <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

UCI MACHINE LEARNING · UPDATED 6 YEARS AGO

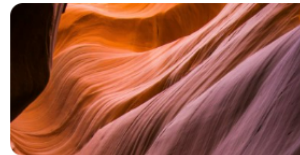
3120

New Notebook

Download (9 kB)

Pima Indians Diabetes Database

Predict the onset of diabetes based on diagnostic measures



Data Code (2111) Discussion (40) Metadata

About Dataset

Context

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Content

The datasets consists of several medical predictor variables and one target variable, `Outcome`. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Acknowledgements

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261--265). IEEE Computer Society Press.

Usability

8.82

License

CC0: Public Domain

Expected update frequency

Not specified



NN: Binary Classification

- 피마 인디언 데이터 분석
- 당뇨병이 유전, 환경에 얼마나 영향이 있는지 확인 가능
 - Pregnancies: 임신 횟수
 - Glucose: 포도당 부하 검사 수치
 - BloodPressure: 혈압(mm Hg)
 - SkinThickness: 팔 삼두근 뒤쪽의 피하지방 측정값(mm)
 - Insulin: 혈청 인슐린(mu U/ml)
 - BMI: 체질량지수(체중(kg)/키(m))^2
 - DiabetesPedigreeFunction: 당뇨 내력 가중치 값
 - Age: 나이
 - Outcome: 클래스 결정 값(0 또는 1)

속성						클래스	
	정보 1	정보 2	정보 3	...	정보 8	당뇨병 여부	
샘플	1번째 인디언	6	148	72	...	50	1
	2번째 인디언	1	85	66	...	31	0
	3번째 인디언	8	183	64	...	32	1

	768번째 인디언	1	93	70	...	23	0

NN: Binary Classification

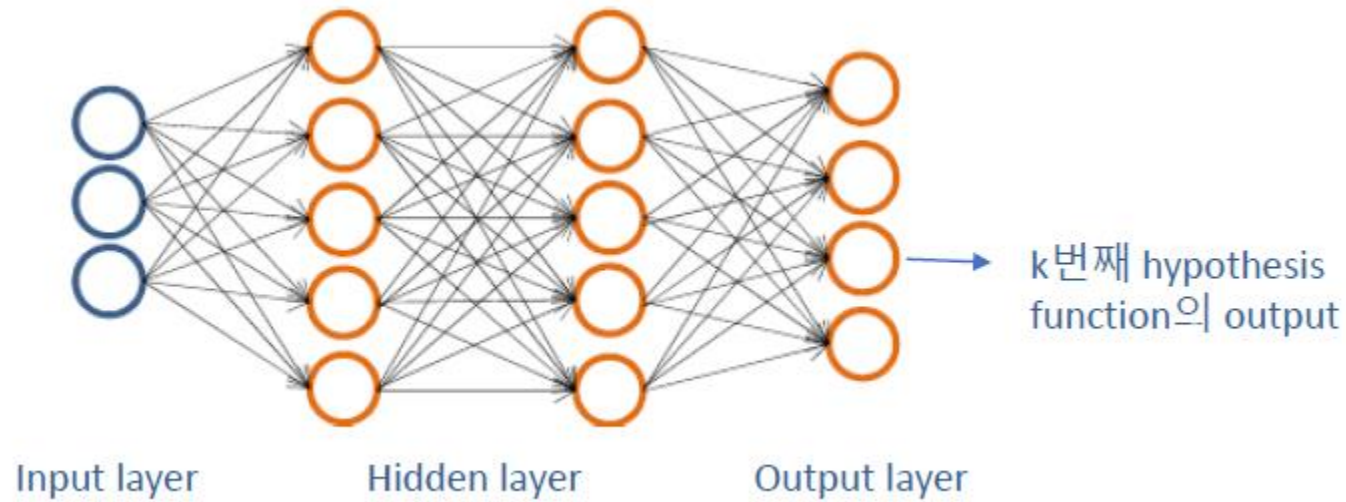
➤ 당뇨병 Classification (True or False)

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

NN: Multiclass Classification

- 다중분류를 위한 Neural Network 구현 (Iris dataset)
- Review
 - Output Node 의 수 : class의 수
 - Output Layer에서 활성화함수 `nn.Sigmoid()` 삭제
 - Binary classification의 `nn.BCELoss()` → `nn.CrossEntropyLoss()` 사용



NN: Multiclass Classification

- 다중분류를 위한 Neural Network 구현 (Iris dataset)
 - 붓꽃은 꽃잎의 모양과 길이에 따라 품종이 나뉨 (3가지)



Iris-virginica



Iris-setosa



Iris-versicolor

NN: Multiclass Classification

➤ 다중분류를 위한 Neural Network 구현 (Iris dataset)

➤ 데이터 분석

- 샘플 수: 150
- 속성 수: 4
 - 꽃받침 길이 (Sepal Length)
 - 꽃받침 너비 (Sepal Width)
 - 꽃잎 길이 (Petal Length)
 - 꽃잎 너비 (Petal Width)
- Class
 - Iris-setosa
 - Iris-versicolor
 - Iris-virginica

속성					클래스	
샘플	정보 1	정보 2	정보 3	정보 4	품종	
	1번째 아이리스	5.1	3.5	4.0	0.2	Iris-setosa
	2번째 아이리스	4.9	3.0	1.4	0.2	Iris-setosa
	3번째 아이리스	4.7	3.2	1.3	0.3	Iris-setosa

	150번째 아이리스	5.9	3.0	5.1	1.8	Iris-virginica

NN: Multiclass Classification

➤ 다중분류를 위한 뉴럴네트워크 구현 (CIFAR-10)

- 6만장의 32 x 32 컬러 이미지 10개 category – Training 5만장, Test 1만장
- <https://www.cs.toronto.edu/~kriz/cifar.html>
- <https://www.kaggle.com/competitions/cifar-10/data>

