딥러닝, 머신러닝 기반 데이터 분석 과정

2021

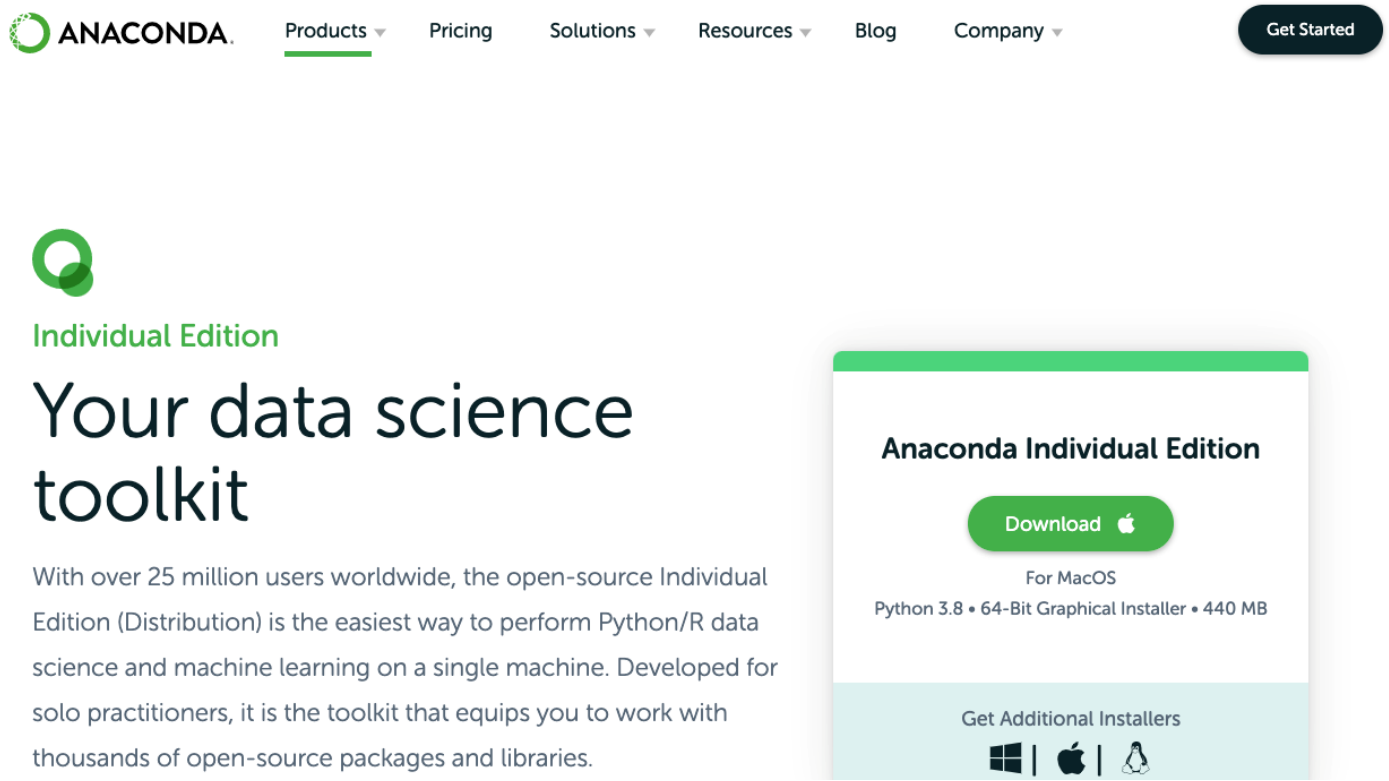
강사: 오영제

교육 환경 준비

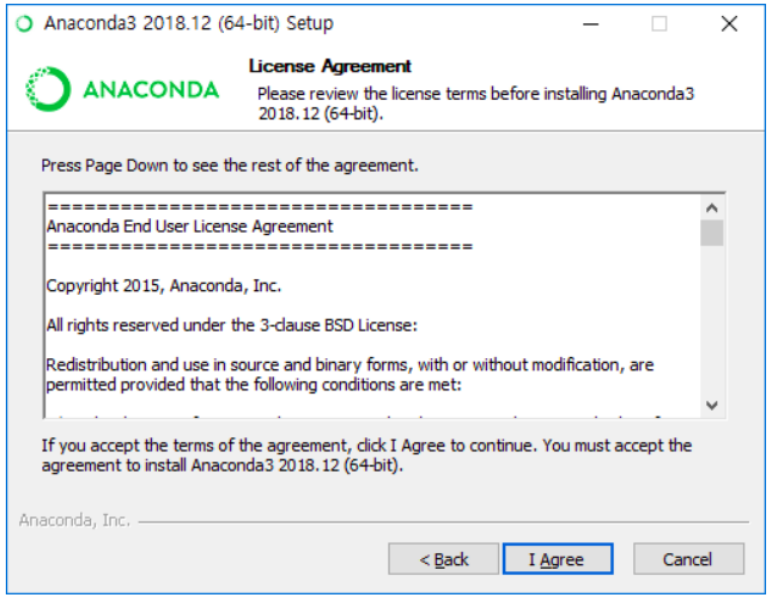
- 인터넷 연결
- Chrome Browser
- Anaconda 설치
- Gmail ID

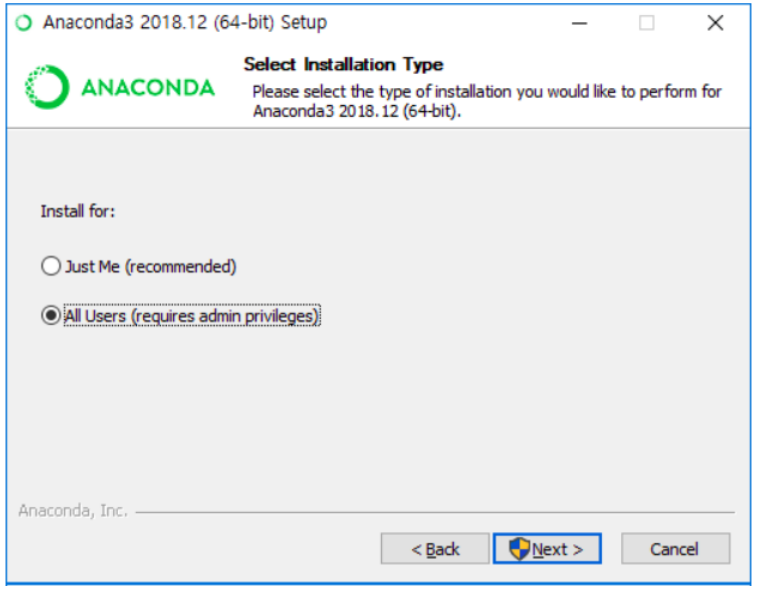
Anaconda 설치

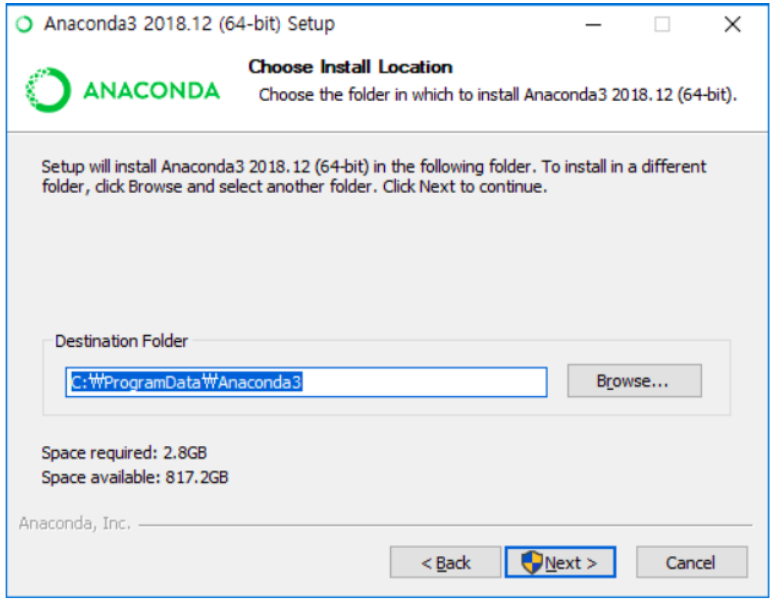
Download from https://www.anaconda.com/distribution/#download-section

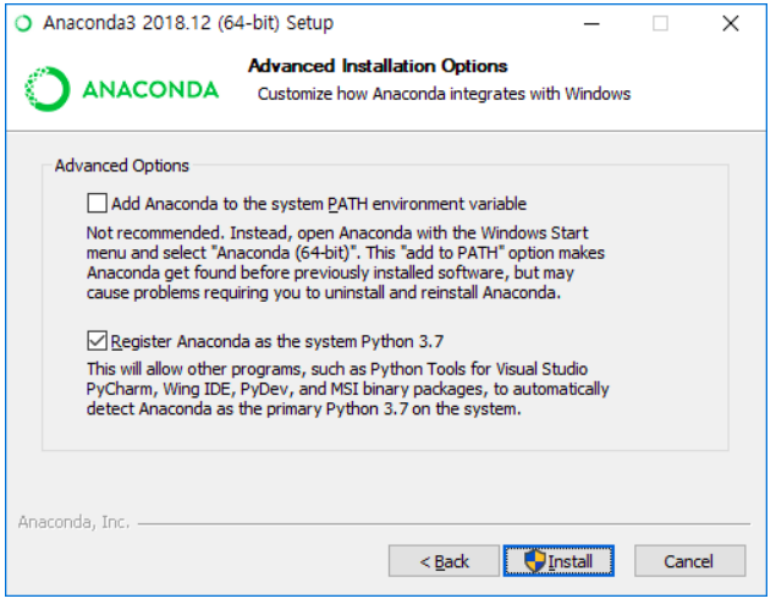


Next 를 눌러 설치









완료

Jupyter Notebook 사용 방법

What is Jupyter Notebook ?

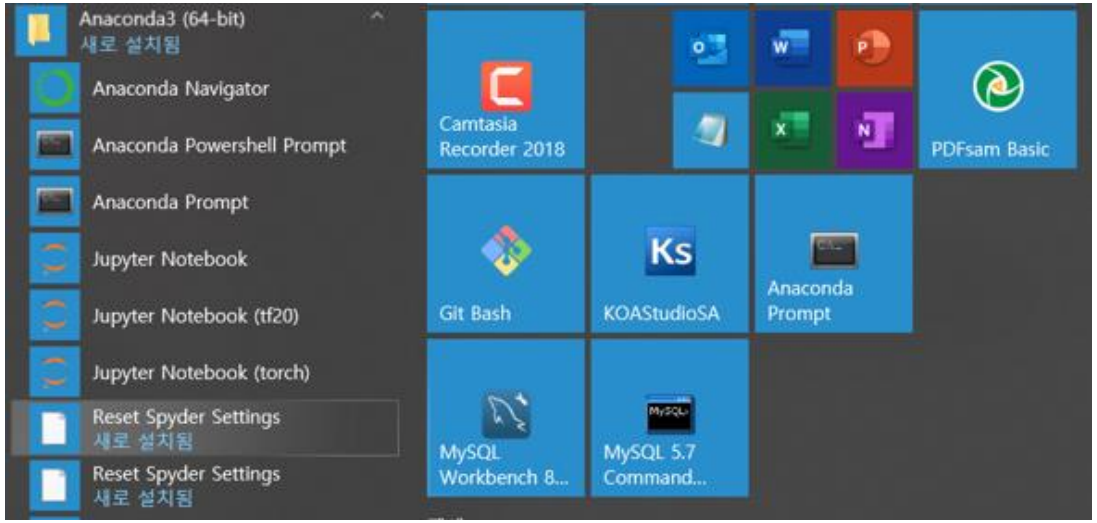
- 주피터 노트북(**Jupyter Notebook**)은 웹 브라우저에서 파이썬 코드를 작성하고 실행해 볼 수 있는 개발도구

- 아나콘다(Anaconda)를 설치하면 **Jupyter Notebook**이 함께 설치

- Notebook 서버 프로그램은 백그라운드에서 실행되는 파이썬 프로그램으로 웹 브라우저를 통해 interactive 하게 Python 명령 실행

Jupyter Notebook 실행 방법

1) Start menu 에서 실행

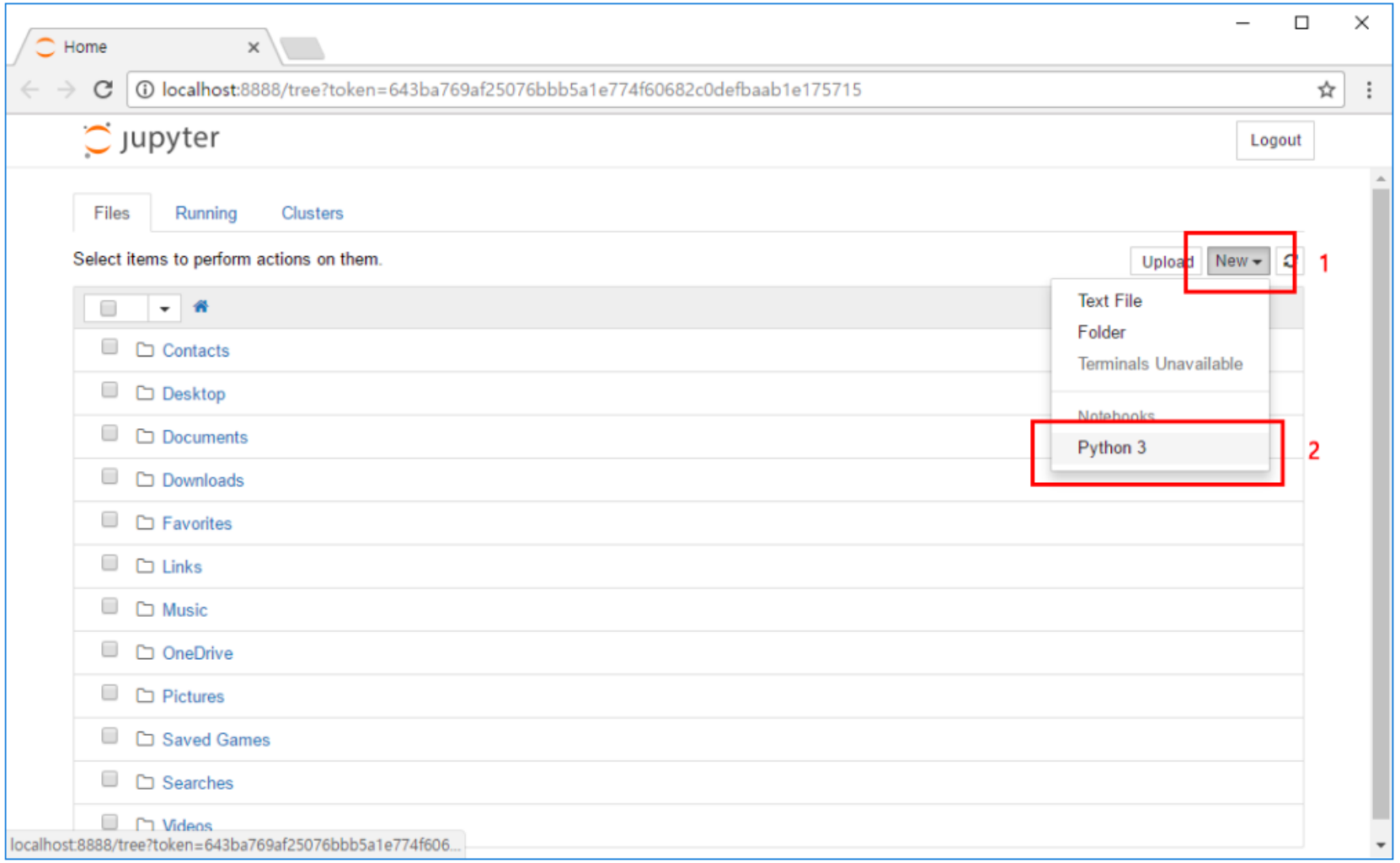


2) 명령 prompt 에서 실행
> jupyter notebook

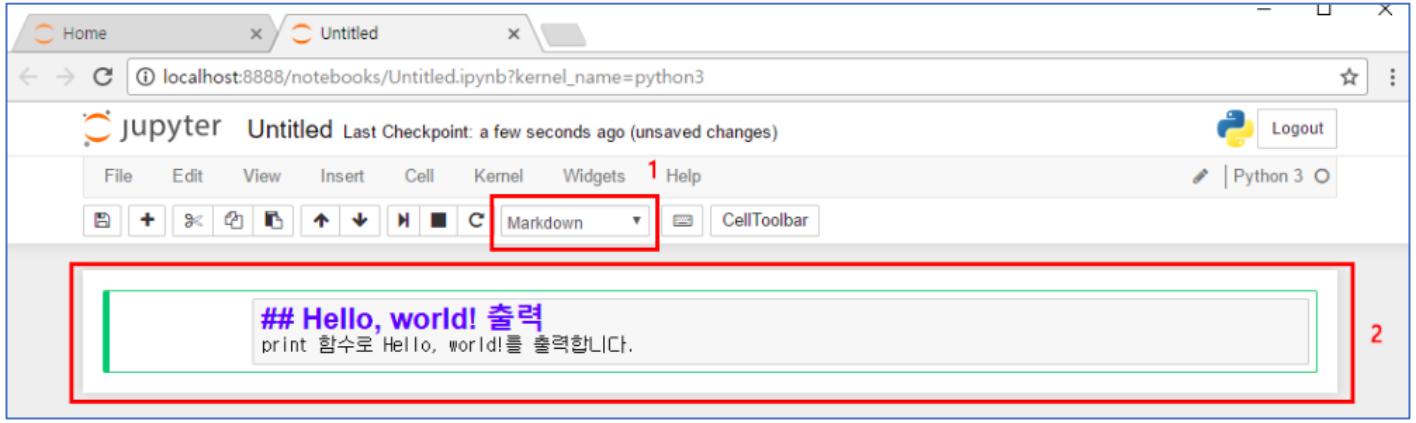
```
(base) C:₩Users₩trimu>jupyter notebook
[I 08:20:31.790 NotebookApp] Serving notebooks from local directory: C:₩Users₩trimu
[I 08:20:31.791 NotebookApp] The Jupyter Notebook is running at:
[I 08:20:31.793 NotebookApp] http://localhost:8888/?token=e1bb3f10db684cd21f9dd3c0e89f713306df271678b97d9d
[I 08:20:31.798 NotebookApp]  or http://127.0.0.1:8888/?token=e1bb3f10db684cd21f9dd3c0e89f713306df271678b97d9d
[I 08:20:31.800 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation
).
[C 08:20:31.994 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/trimu/AppData/Roaming/jupyter/runtime/nbserver-12876-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=e1bb3f10db684cd21f9dd3c0e89f713306df271678b97d9d
     or http://127.0.0.1:8888/?token=e1bb3f10db684cd21f9dd3c0e89f713306df271678b97d9d
```

Jupyter Notebook 사용방법

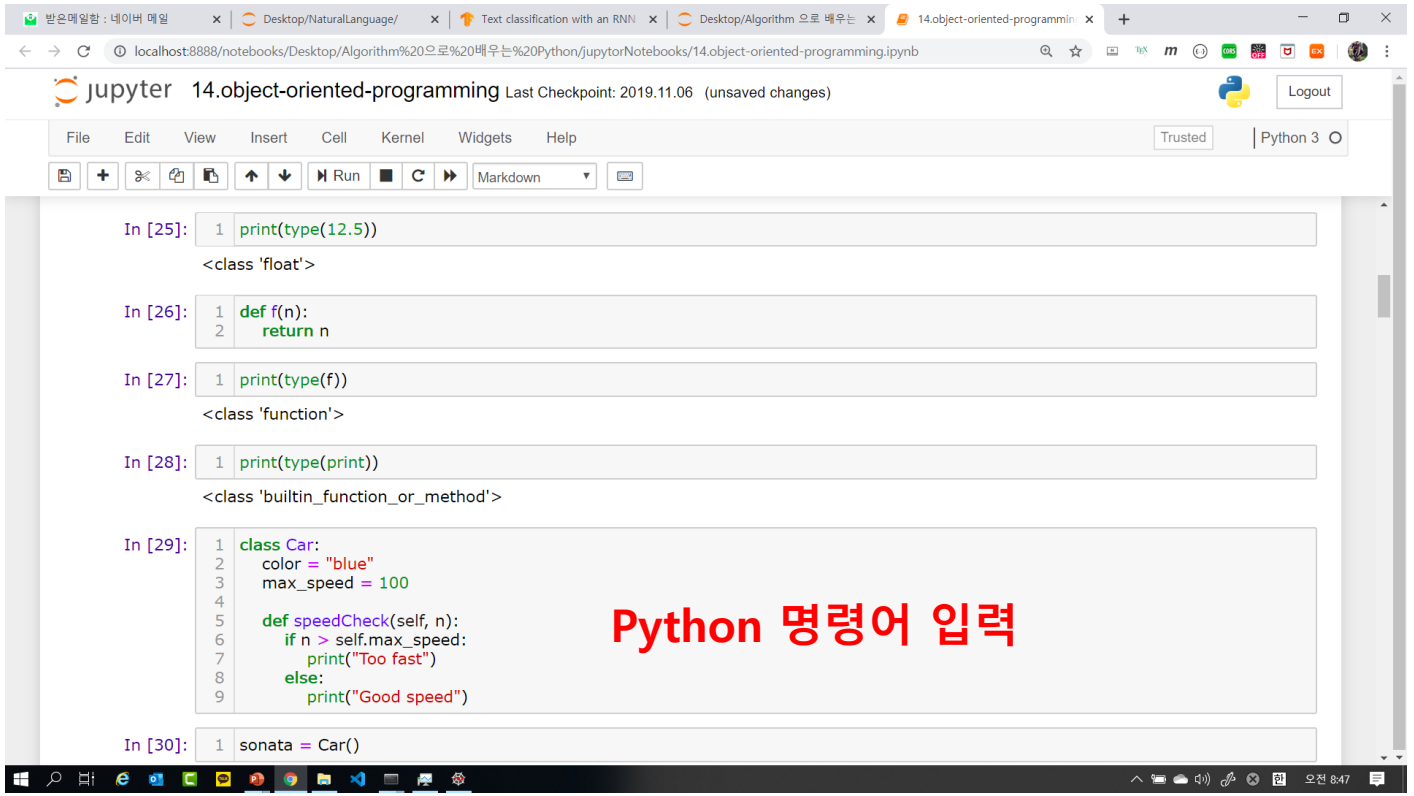
주피터 노트북 초기 화면 에서 New → Python 3 선택 → 새로운 notebook 생성



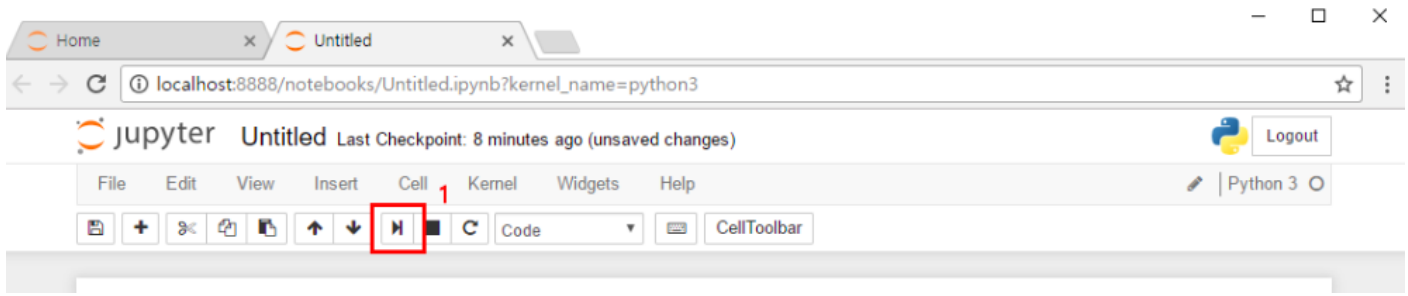
설명 추가



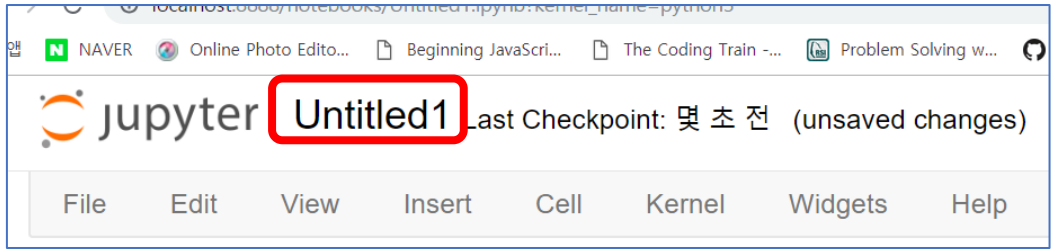
Python Code 입력
및
실행



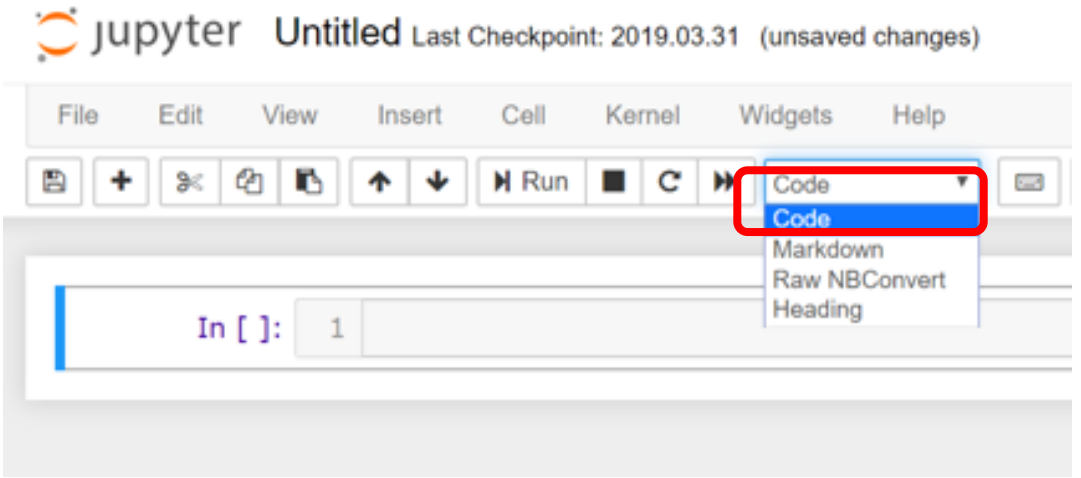
Code 실행



Notebook File Name 바꾸기



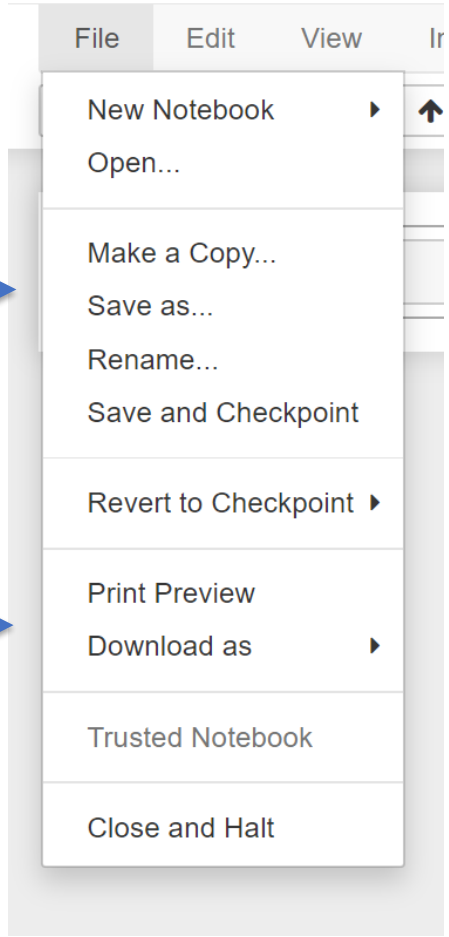
Cell Type 변경



자주 사용하는 menu 들

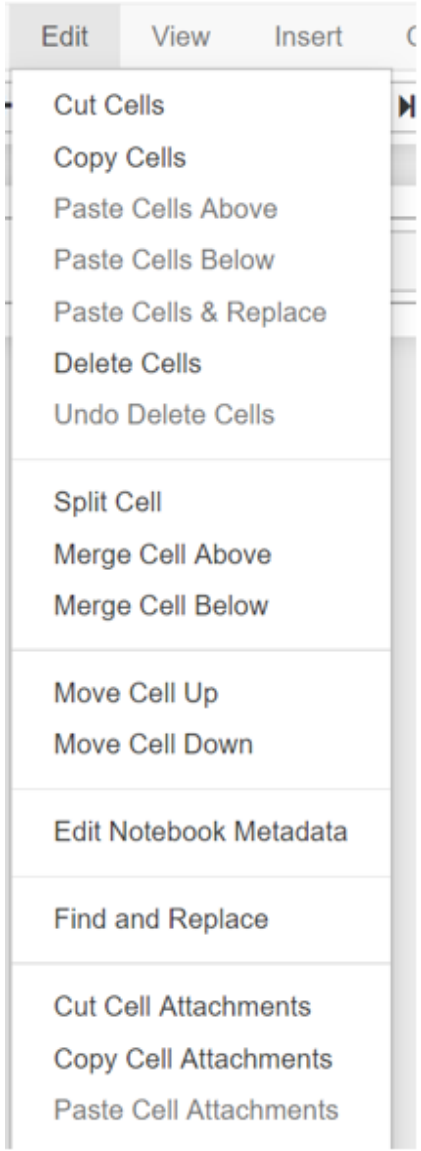
Notebook
저장

Notebook
download

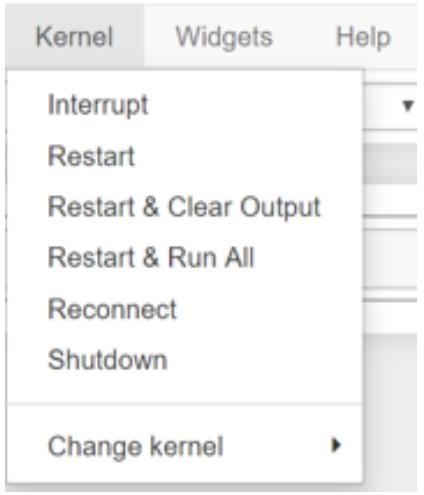


Cell 삭제
취소

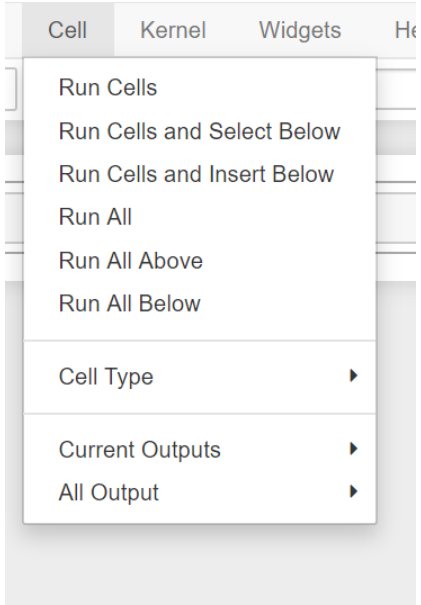
Find/
Replace



Notebook
restart



Cell 실행



Jupyter Notebook 사용방법

- 자주 사용하는 short-cut key

Shift + Enter : cell 실행 + 다음 cell 이동

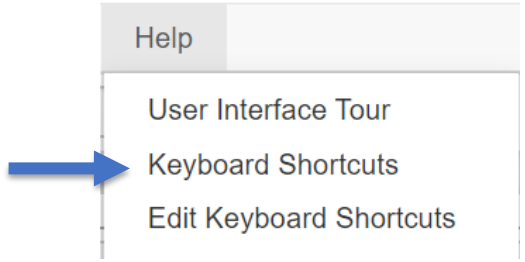
Ctrl + S : save + checkpoint

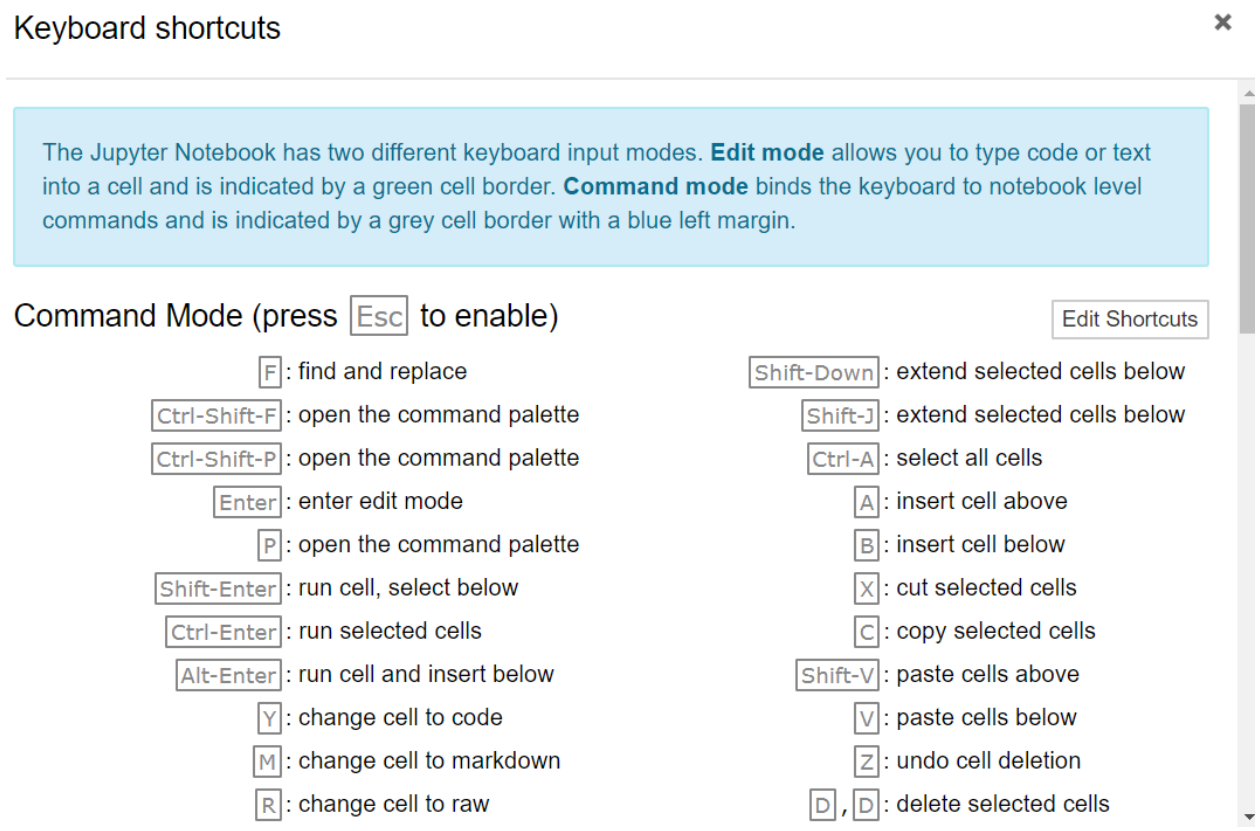
Esc+A : 위쪽에 cell 삽입
Esc+B : 아래쪽에 cell 삽입

Esc+X : cell 삭제

Esc+Z : cell 삭제 취소

Esc+M : cell 을 markdown type 으로 변경

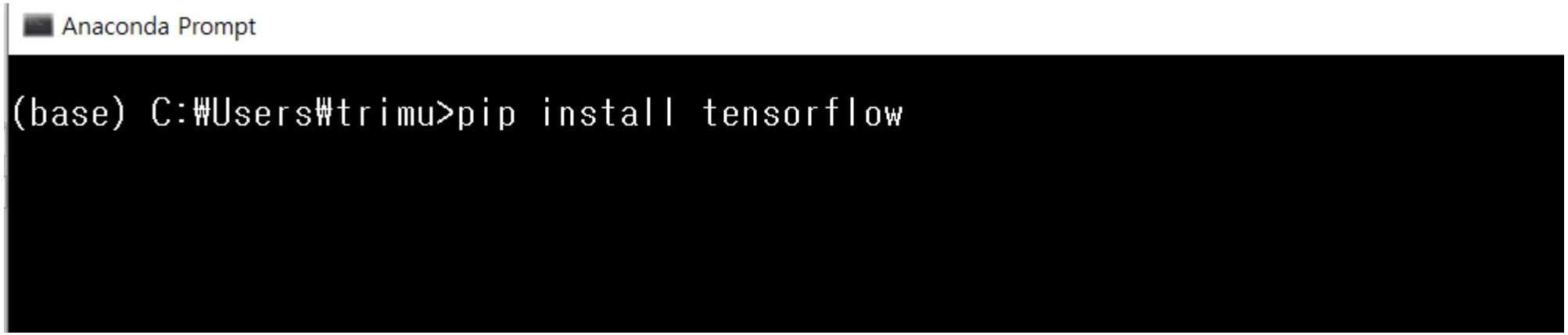


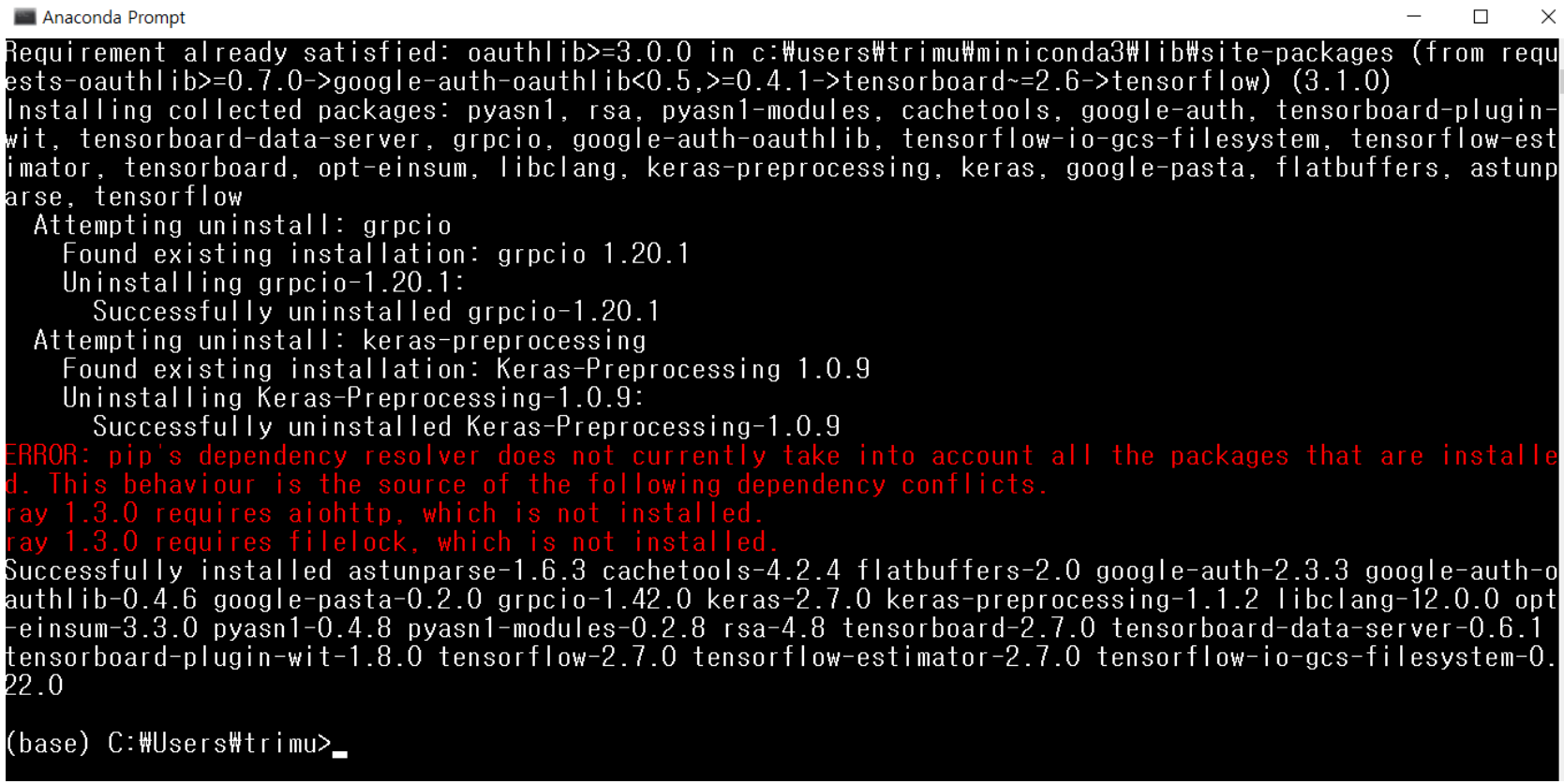


Tensorflow 설치

- Windows 시작 → Anaconda3 → Anaconda Prompt 선택

- pip install tensorflow






```
(base) C:\Users\trimu>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
2021-12-12 11:02:16.101071: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2021-12-12 11:02:16.113268: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
>>>
```

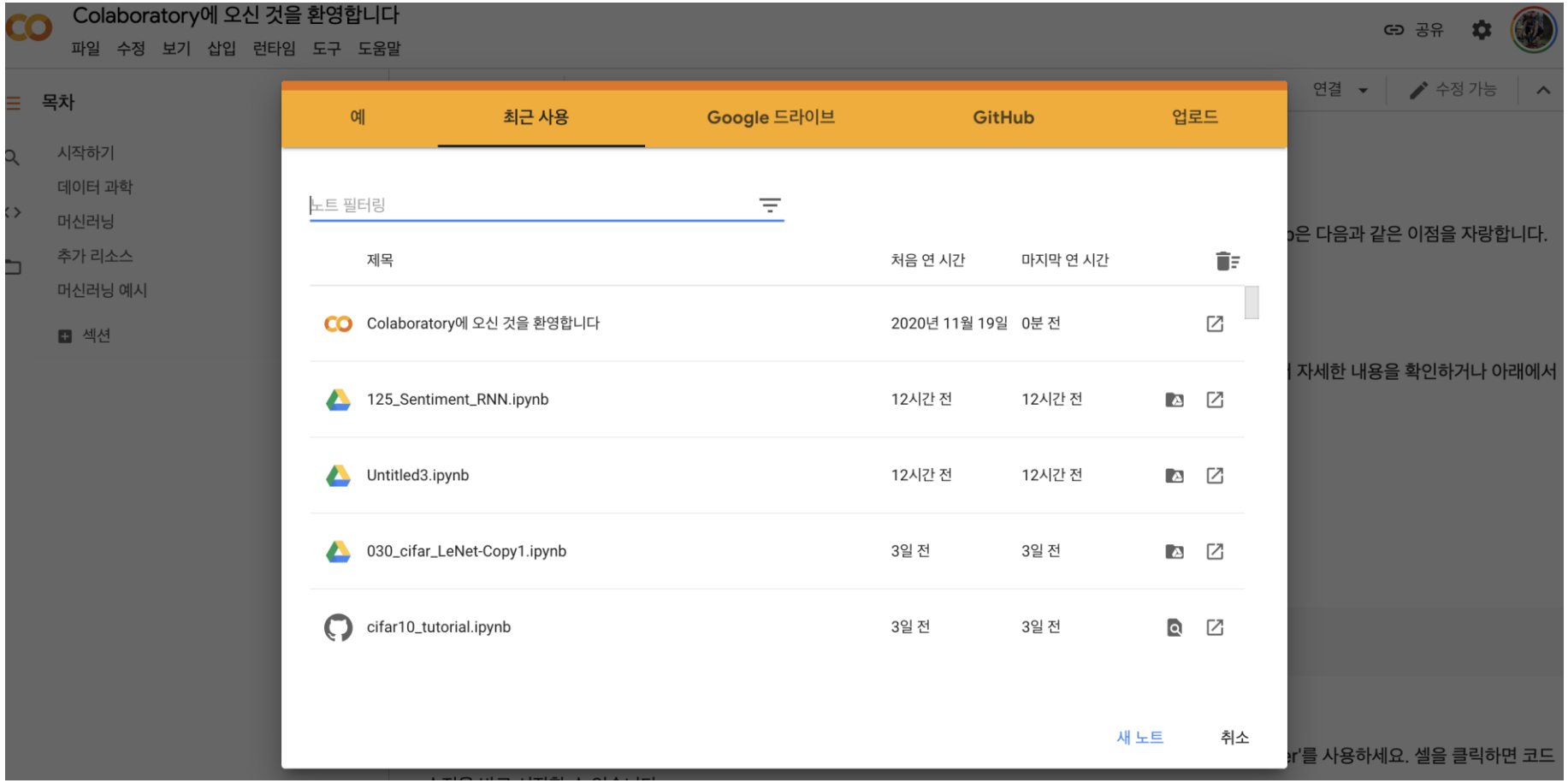
```
>>> tf.version.VERSION
'2.7.0'
>>>
```

Google Colaboratory 소개

- Free GPU 제공
- Google Drive 와 연동
- Jupyter Notebook 환경
- Deep Learning beginner 를 위한 최적의 환경
- 각종 snippet 제공

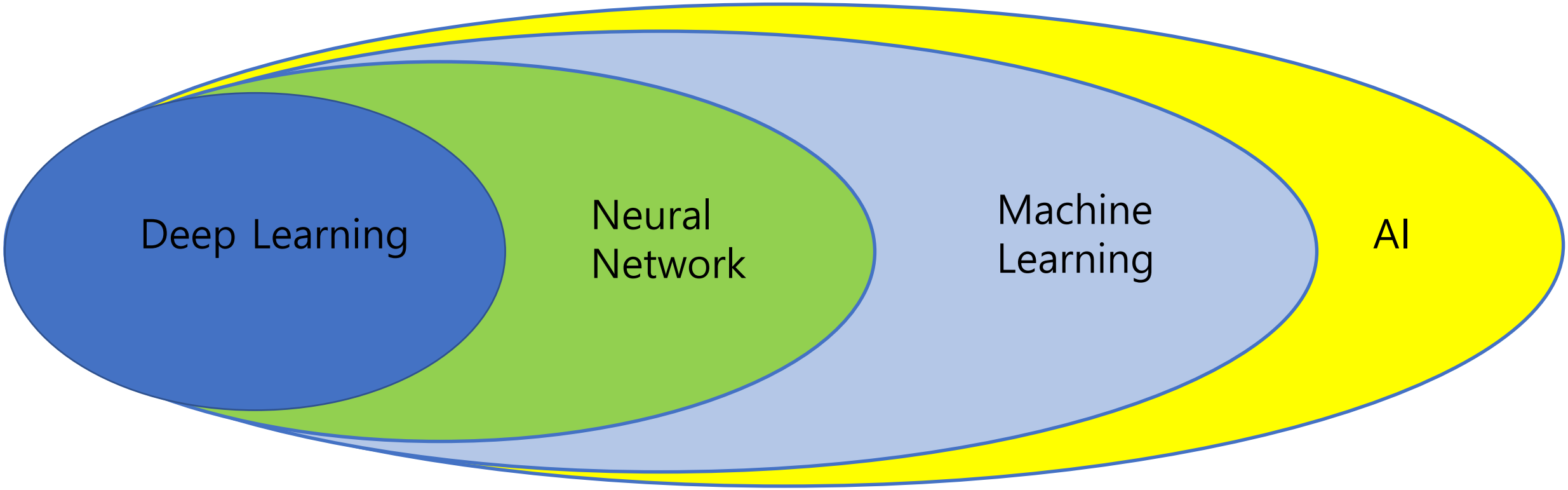
Google Colaboratory 사용하기

https://colab.research.google.com/

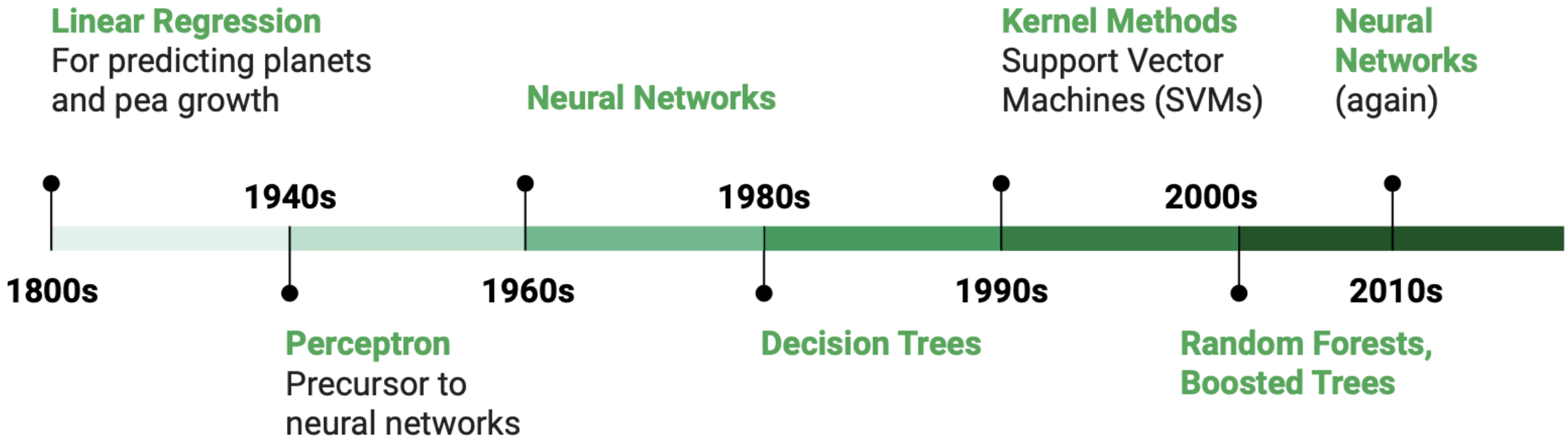


Machine Learning 개요

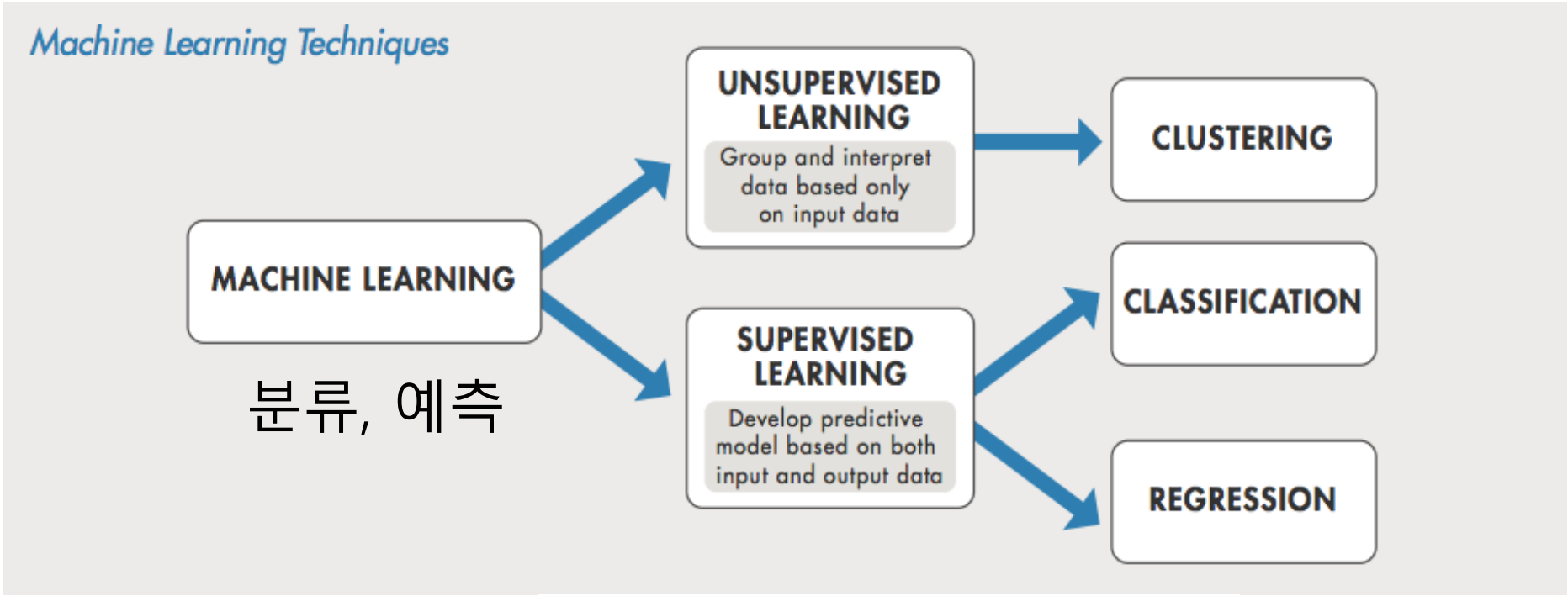
AI vs. Machine Learning vs. Deep Learning



인공지능 model 의 역사

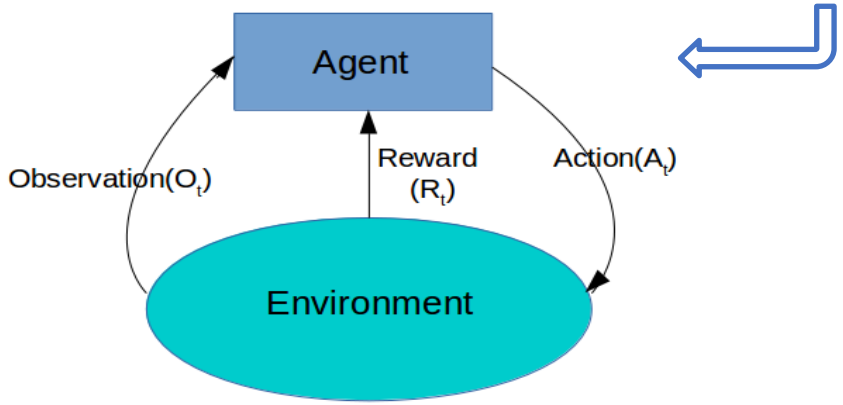


Machine Learning 의 종류



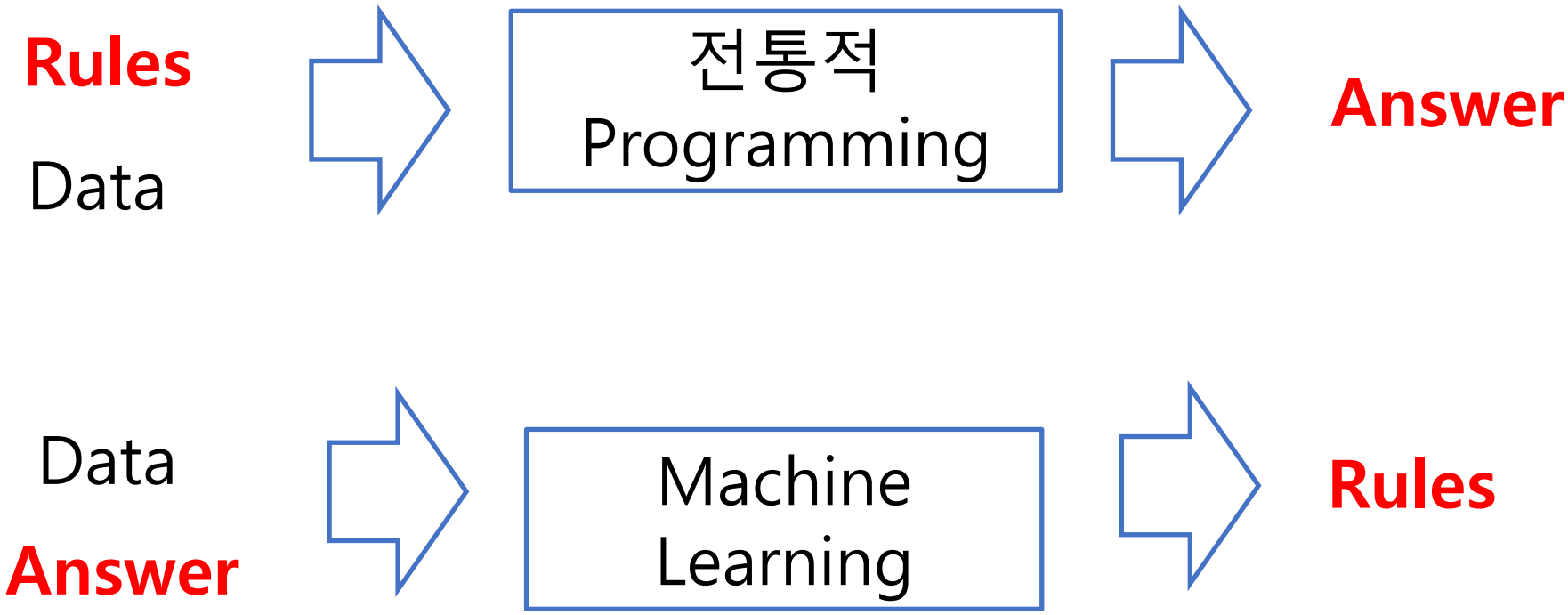
분류, 예측

REINFORCEMENT LEARNING

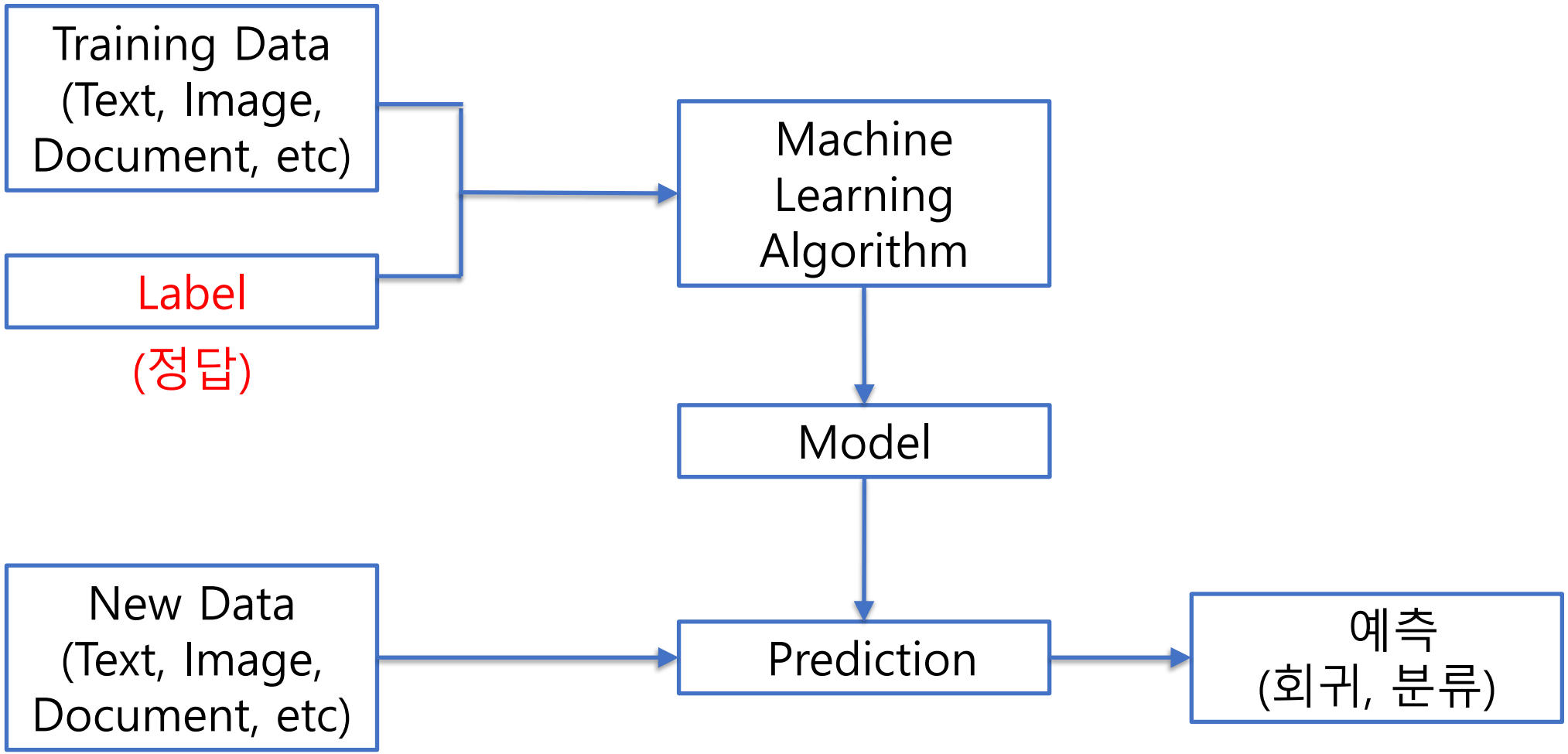


의사결정

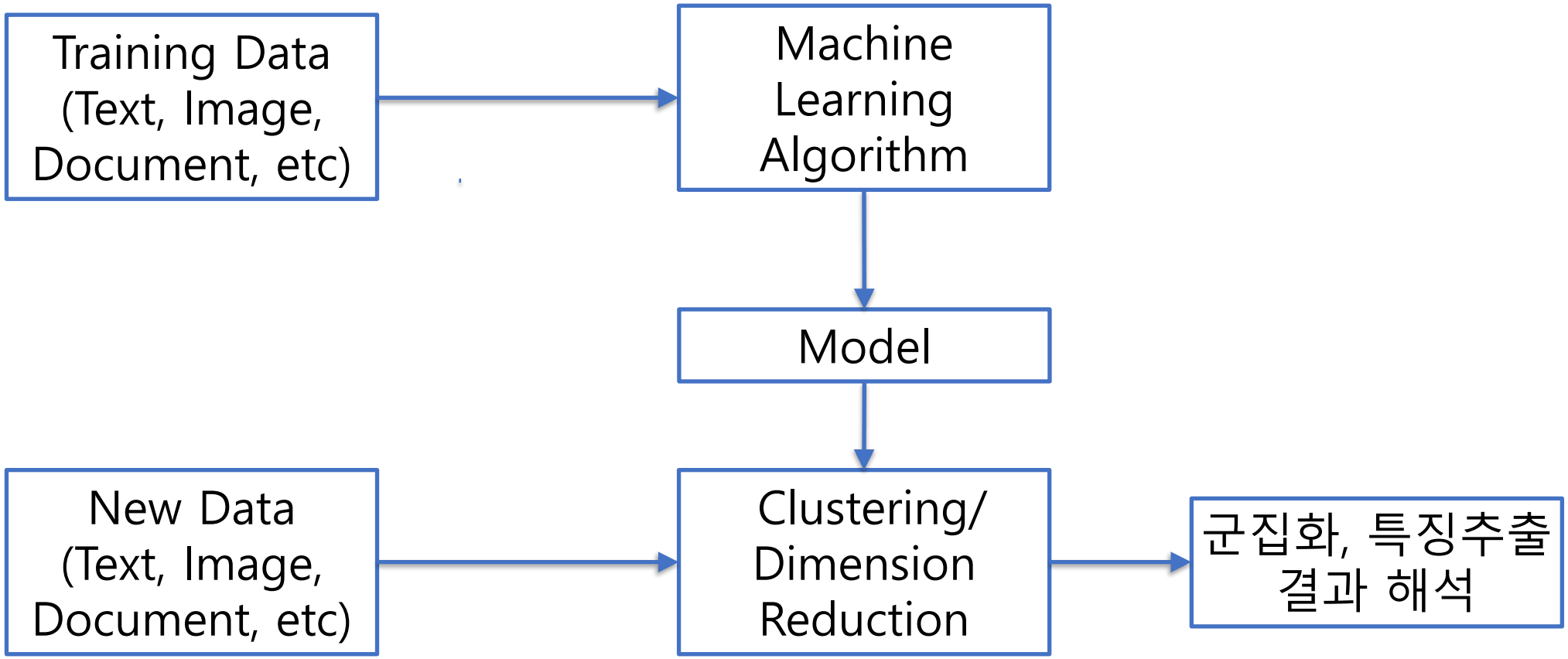
전통적 Programming
vs
Machine Learning



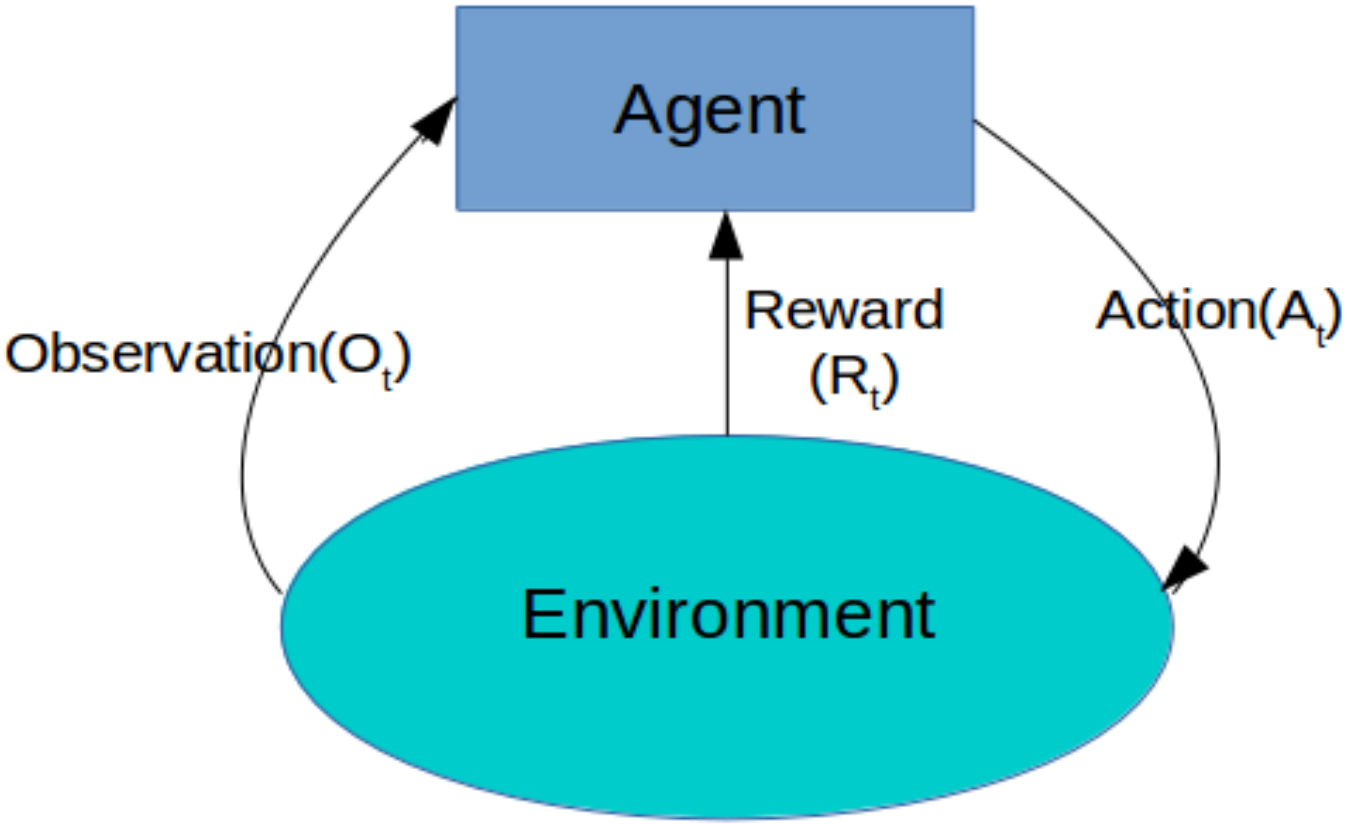
Supervised Learning (지도학습)



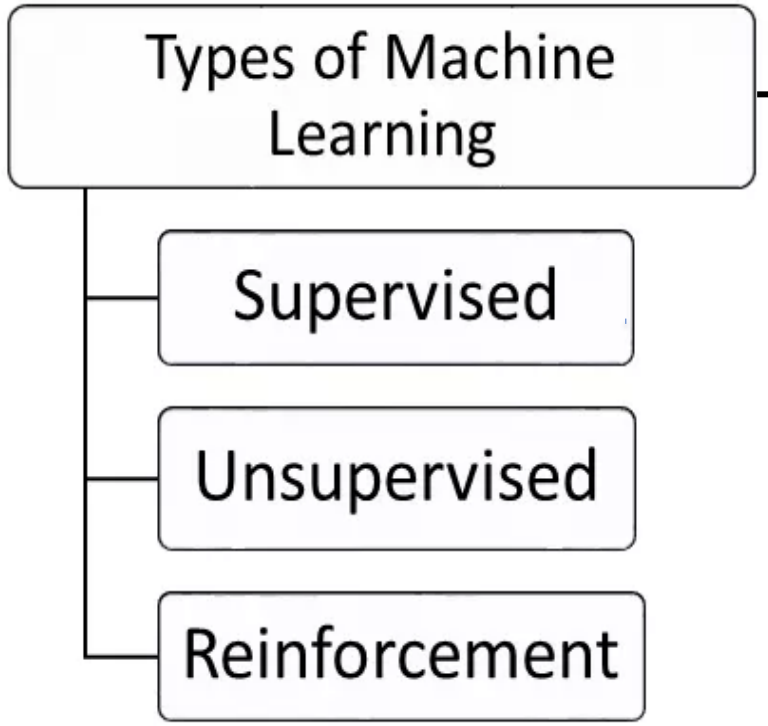
Unsupervised Learning (비지도학습)



Reinforcement Learning (강화학습)



Machine Learning 기법의 종류



전통적 ML	• Linear & Logistic Regression • KNN • Decision Tree • SVM • Ensembles • K-Means Clustering • PCA • 통계 모델
Deep Learning (Neural Network)	• FC • CNN • RNN • Auto Encoder • GAN

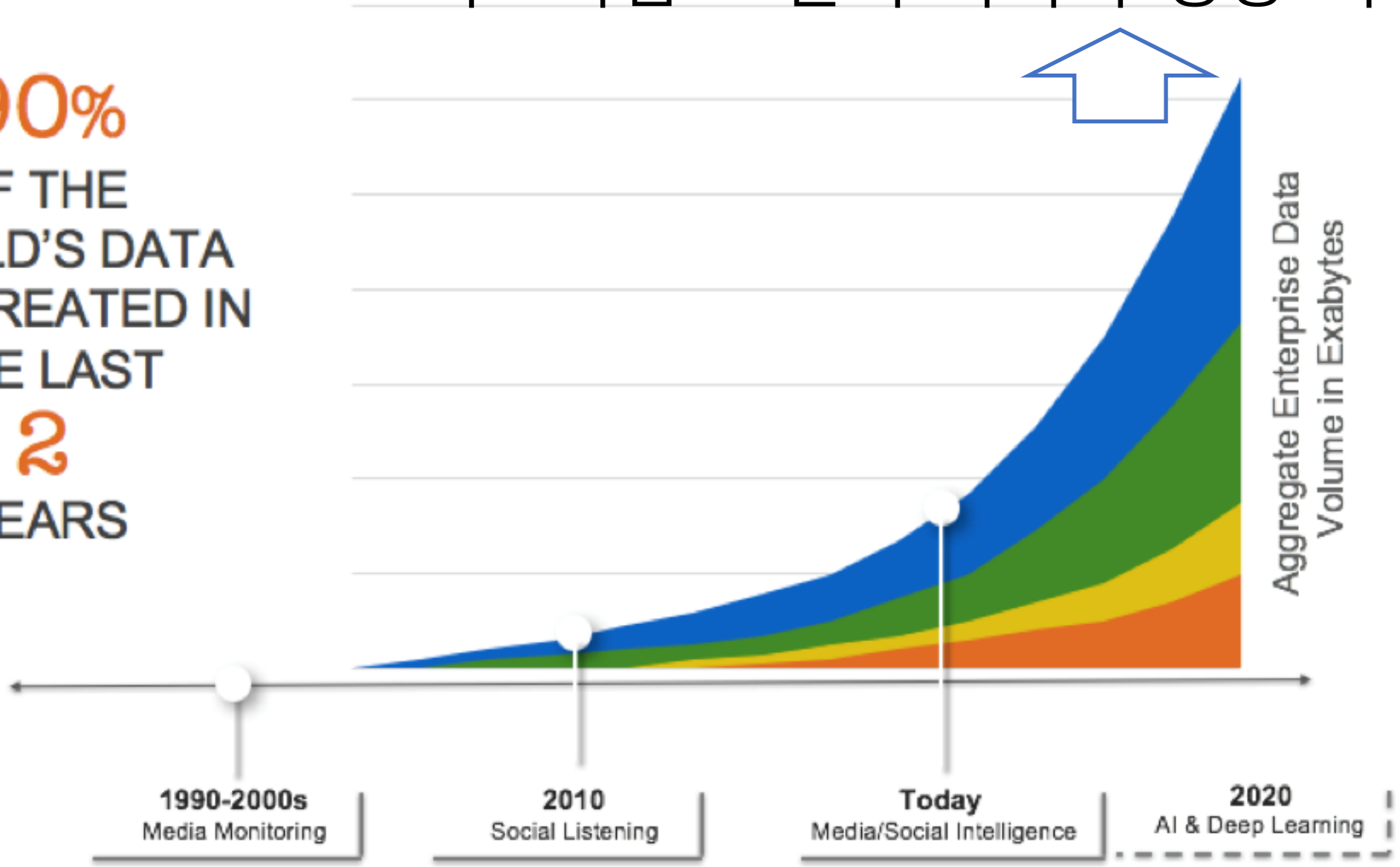
지도학습 모델의 획기적 성공 이유

90%

OF THE WORLD'S DATA WAS CREATED IN THE LAST

2

YEARS



1.2 billion photos and videos are uploaded to Google Photos every day.

Total size of over 13 PB of photo data.

(1PB or 400 hours of video uploaded every minute)

지도 학습 (Supervised Learning)

Input (A)	Output (B)	Application
email	spam? (0/1)	spam filtering
audio	text transcripts	speech recognition
English	Chinese	machine translation
ad, user info	click? (0/1)	online advertising
image, radar info	position of other cars	Self-driving car
image of phone	defect? (0/1)	visual inspection

인간이 1 초 정도 생각하여 할 수 있는 것들은 즉시 혹은 가까운 미래에 자동화

Supervised Learning (지도 학습) 모델에 필요한 Data

user ID	time	price ($)	purchased
4783	Jan 21 08:15.20	7.95	yes
3893	March 3 11:30.15	10.00	yes
8384	June 11 14:15.05	9.50	no
0931	Aug 2 20:30.55	12.90	yes

image	label
	cat
	not cat
	cat
	not cat

Data 를 획득하는 방법

- 수작업으로 일일이 Label 작성


cat


not cat


cat


not cat

- 관측된 결과 활용

user ID	time	price ($)	purchased
4783	Jan 21 08:15.20	7.95	yes
3893	March 3 11:30.15	10.00	yes
8384	June 11 14:15.05	9.50	no
0931	Aug 2 20:30.55	12.90	yes

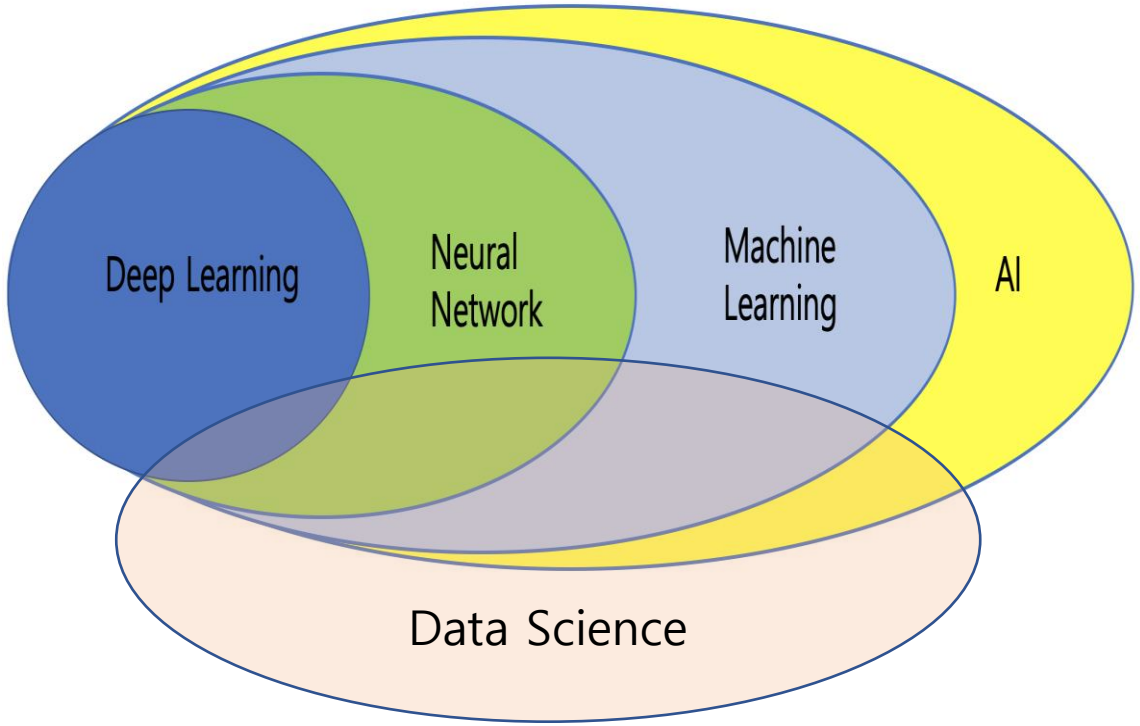
input label

machine	temperature (◦C)	pressure (psi)	machine fault
17987	60	7.65	N
34672	100	25.50	N
08542	140	75.50	Y
98536	165	125.00	Y

- 이미 만들어진 Data 획득 (website, 제휴 기관, etc)

AI 구현의 도구

- Machine Learning & Data Science
- Deep Learning / Neural Network



Machine Learning 과 Data Science 의 차이점

- Machine Learning
  - Data 로부터 Rule 을 학습하는 새로운 S/W 개발 방법
  - 결과물 → software
    ex) 판매가 더 많이 될 부문 예측 (Blackbox model)
- Data Science
  - Data 로부터 새로운 지식과 통찰을 얻는 방법
  - 결과물 → Slide deck
    ex) 판매 동향을 분석하여 특정 부문에 더 광고비를 지출하도록 전략 제시

Machine Learning vs. Data Science project

Manufacturing line

Mix clay → Shape mug → Add glaze → Fire kiln → Final inspection

Machine learning

1. Data Collection

Mug Batch #	Humidity	Temperature in kiln (F)	Duration in kiln (hours)
301	0.002%	1410°	22
302	0.003%	1520°	24
303	0.002%	1420°	22

Data science

1. Data Collection

Machine learning

ok　　ok　　defect

2. Model Train

3. 제품 결함 발견 자동화

인건비 절감 + 품질관리 수준 향상

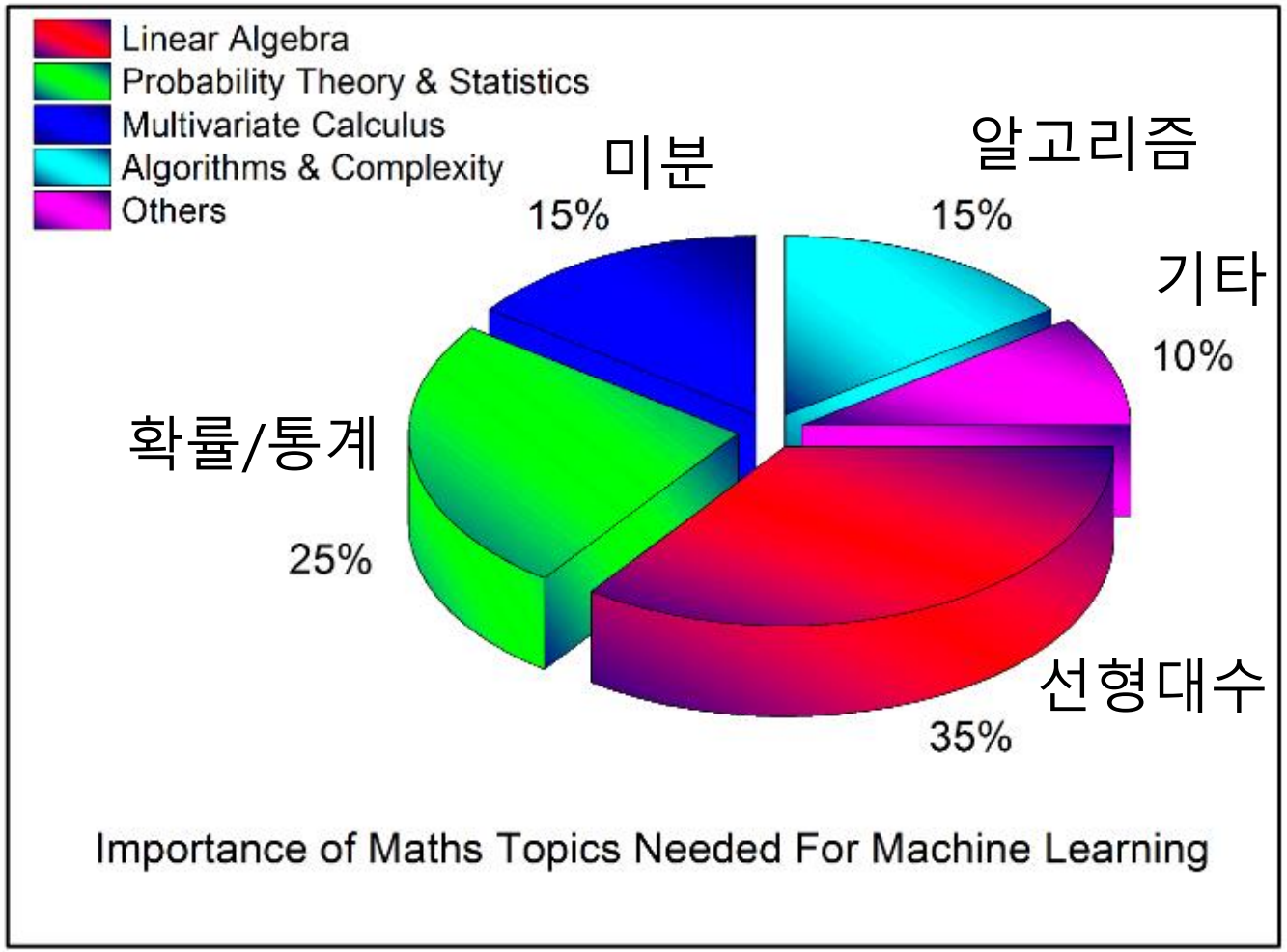
Data science

2. 습도, 가마의 온도 등과 제품의 균열 발생 관계 분석

3. 최적 습도, 온도 제시

생산공정 최적화

Machine Learning 학습에 필요한 지식



Scikit-Learn

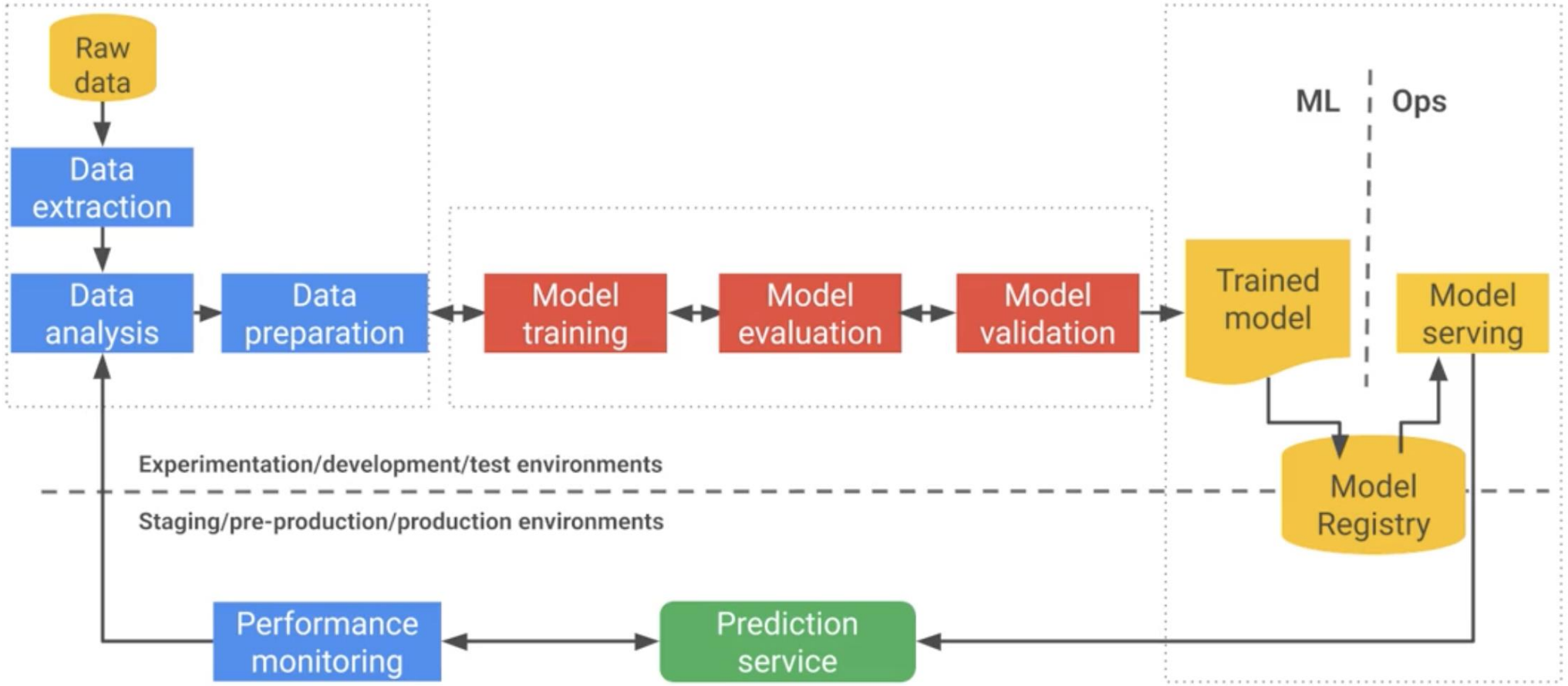
전통적 Machine Learning Tool:

- 벤치마크용 데이터셋 예제
- 데이터 전처리(preprocessing)
- 지도 학습(Supervised learning)
- 비지도 학습(Unsupervised learning)
- 모형 평가 및 선택 (evaluation and selection)

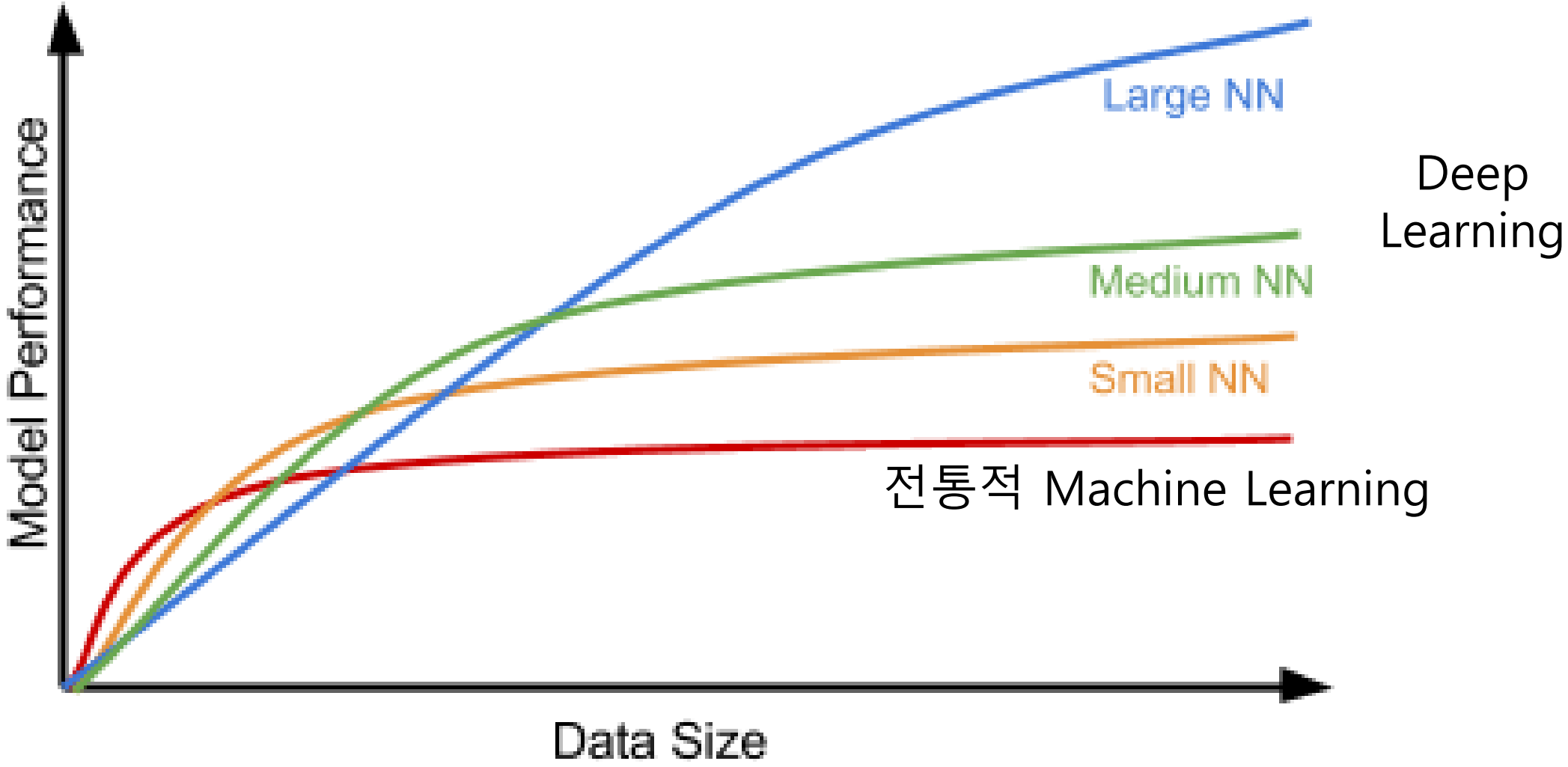
scikit-learn API 의 구조

- 모든 object 의 interface 가 일관성 있게 구성
  - fit() – dataset 의 data 를 기반으로 model parameter 추정
  - transform() – dataset 을 변환
  - fit_transform() – fit() + transform()
  - predict() – 주어진 data 에 대한 예측
  - XXXX_score() – 예측의 quality 측정

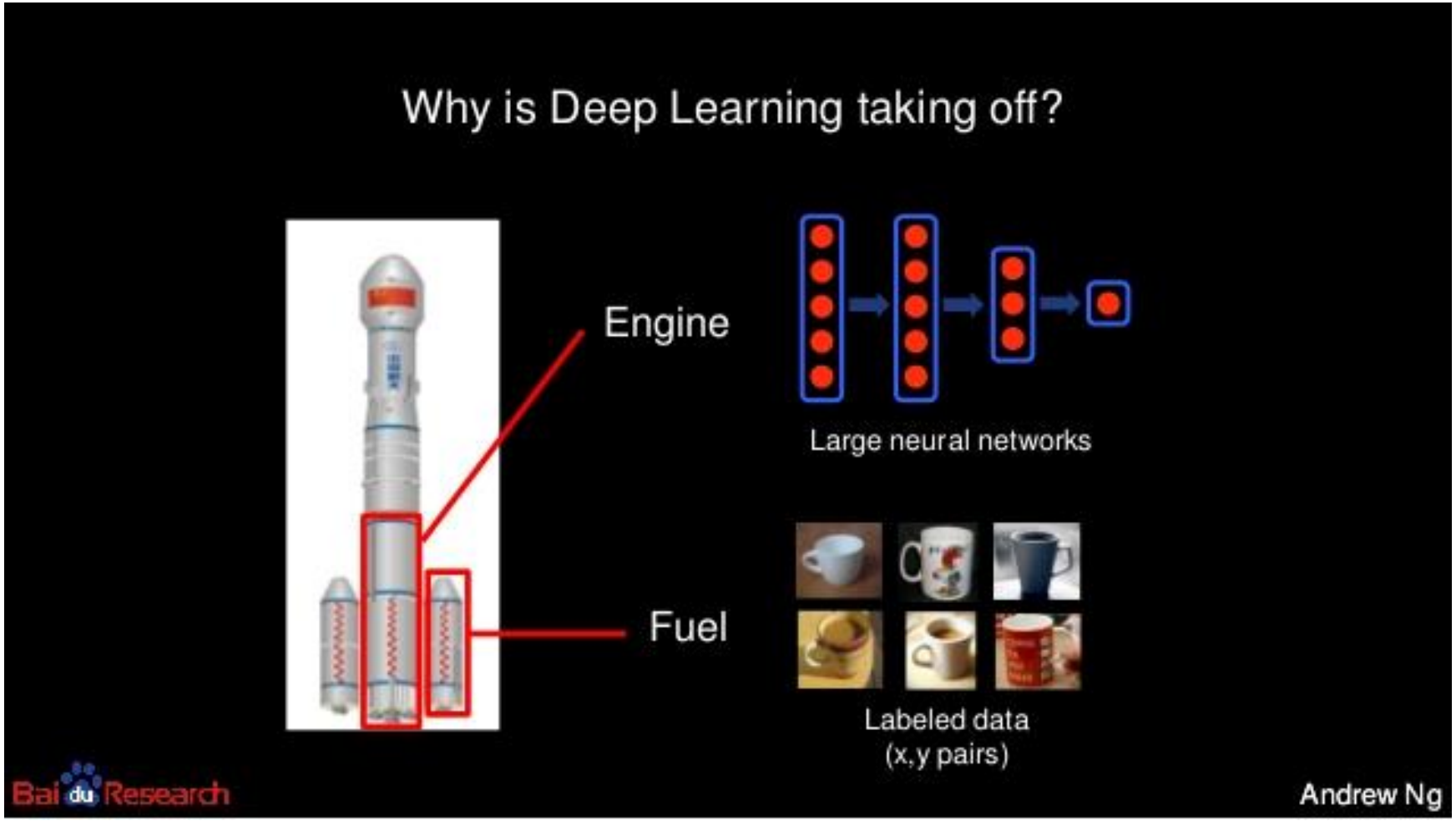
ML Pipeline


Raw data
Data extraction
Data analysis
Data preparation
Model training
Model evaluation
Model validation
Trained model
ML
Ops
Model serving
Experimentation/development/test environments
Staging/pre-production/production environments
Model Registry
Performance monitoring
Prediction service

전통적 Machine Learning vs. Deep Neural Network



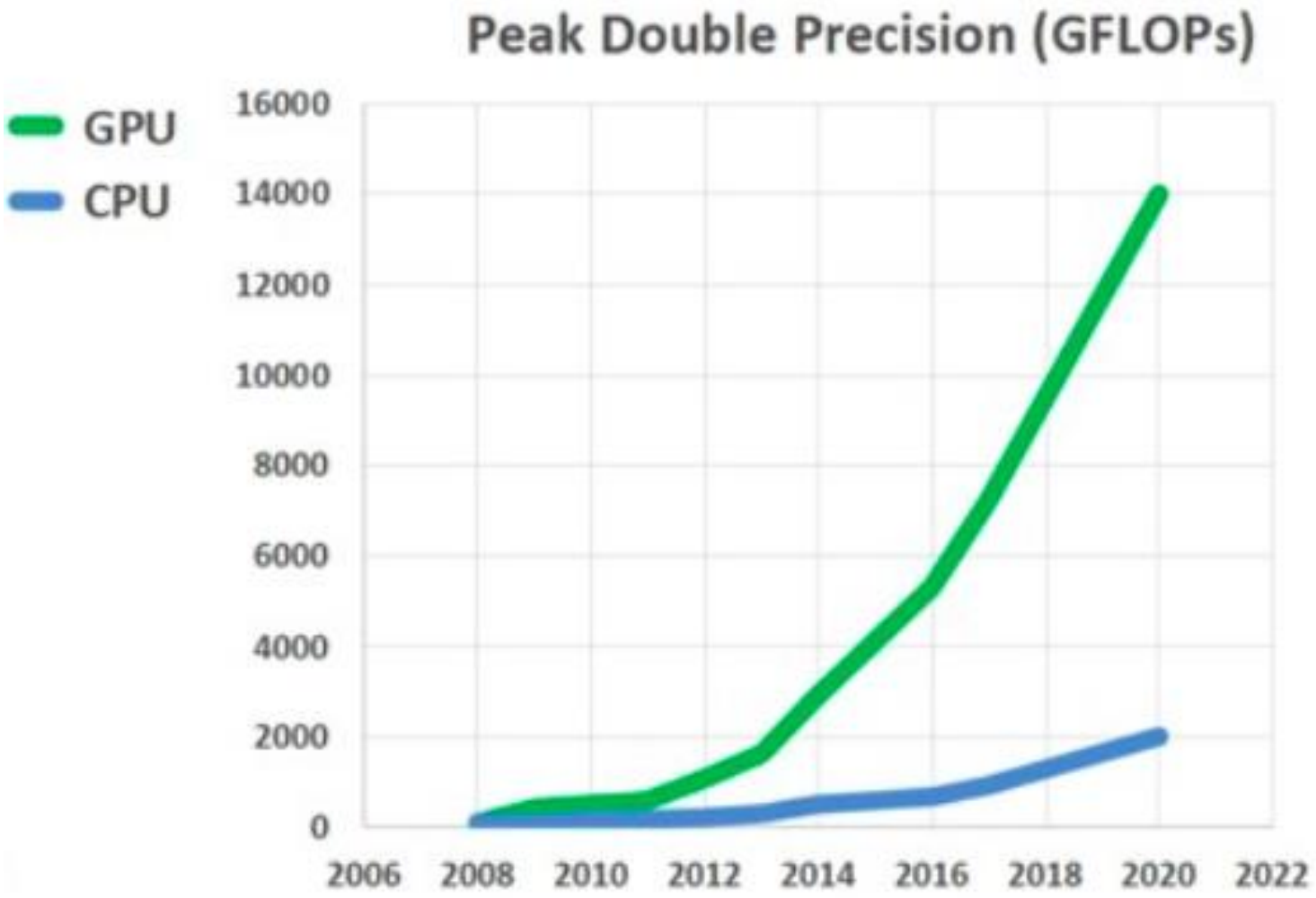
Deep Learning



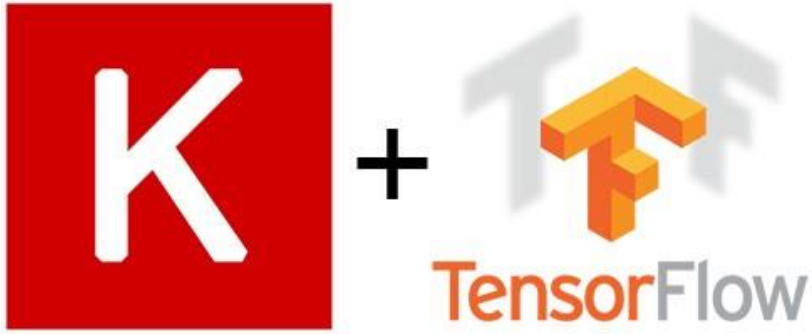
Rocket Engine : NVIDIA + Deep Learning Algorithm

Fuel : Data (25,000 pictures for cat)

CPU vs GPU Performance



Tensorflow + Keras

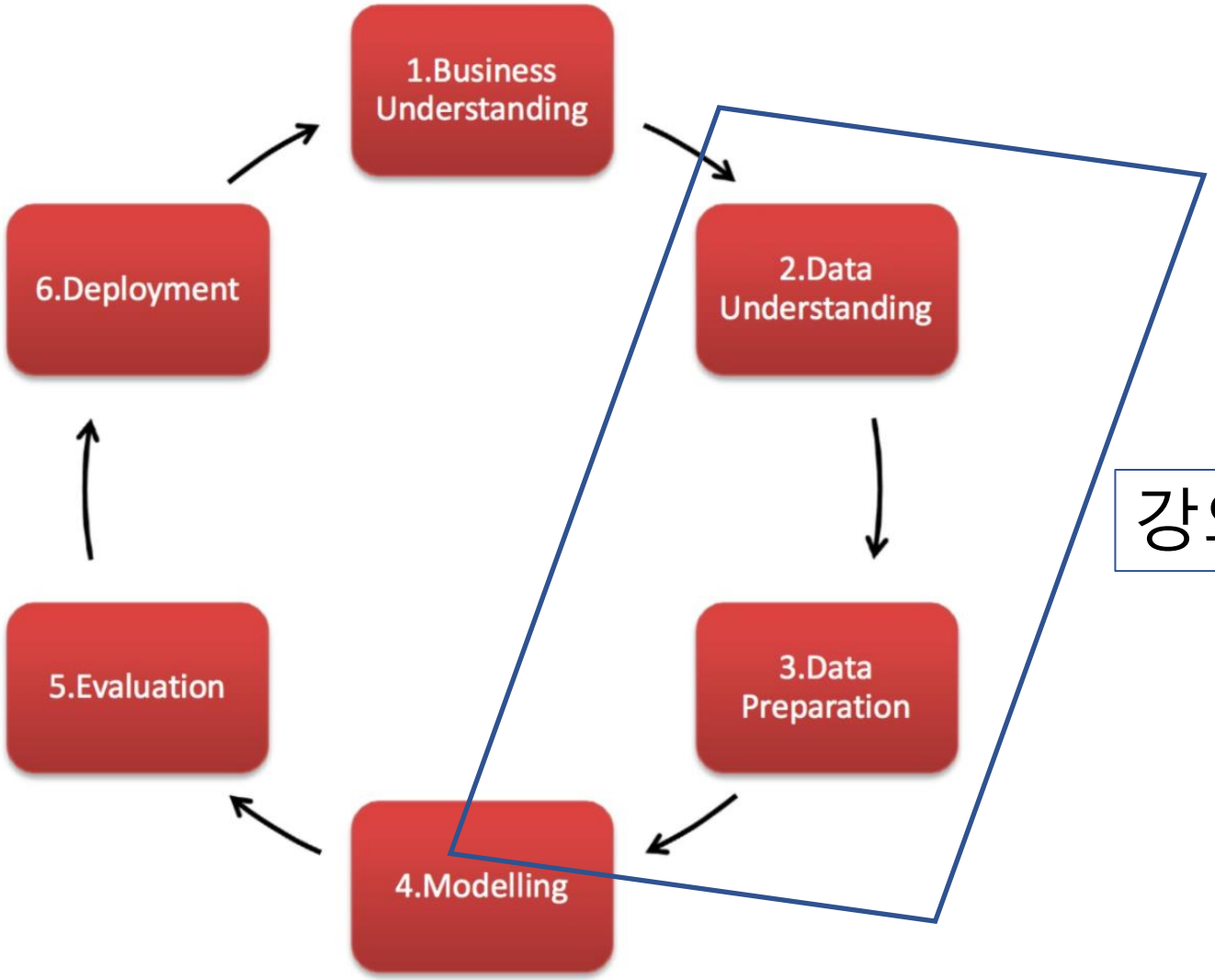


Deep Learning Tool:

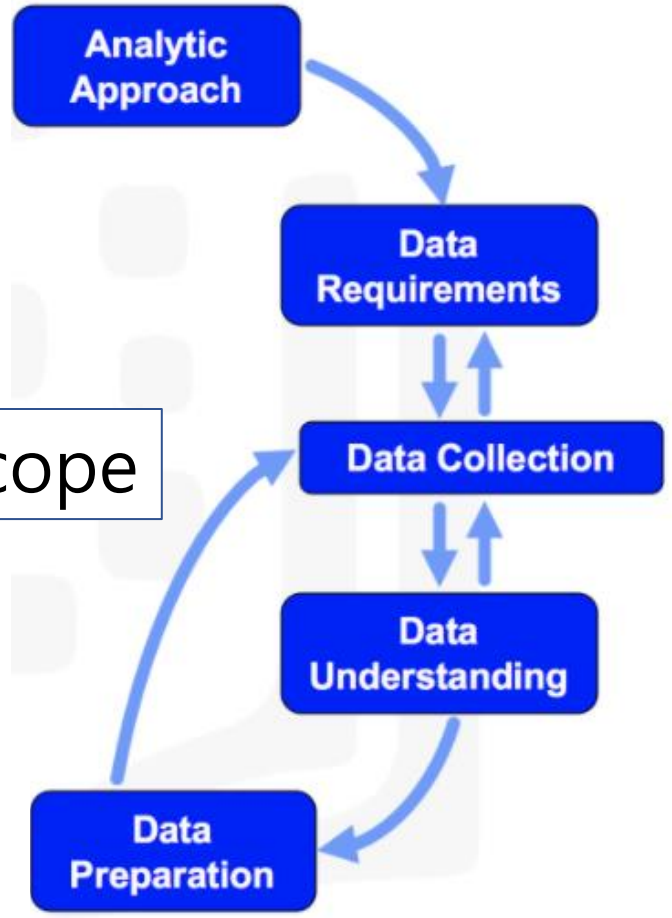
- Pre-train model, Sample Dataset 제공
- Deep Learning 에 필요한 각종 함수 제공
- GPU support
- 각종 language 지원 API 제공

빅데이터 분석

Data 분석 단계

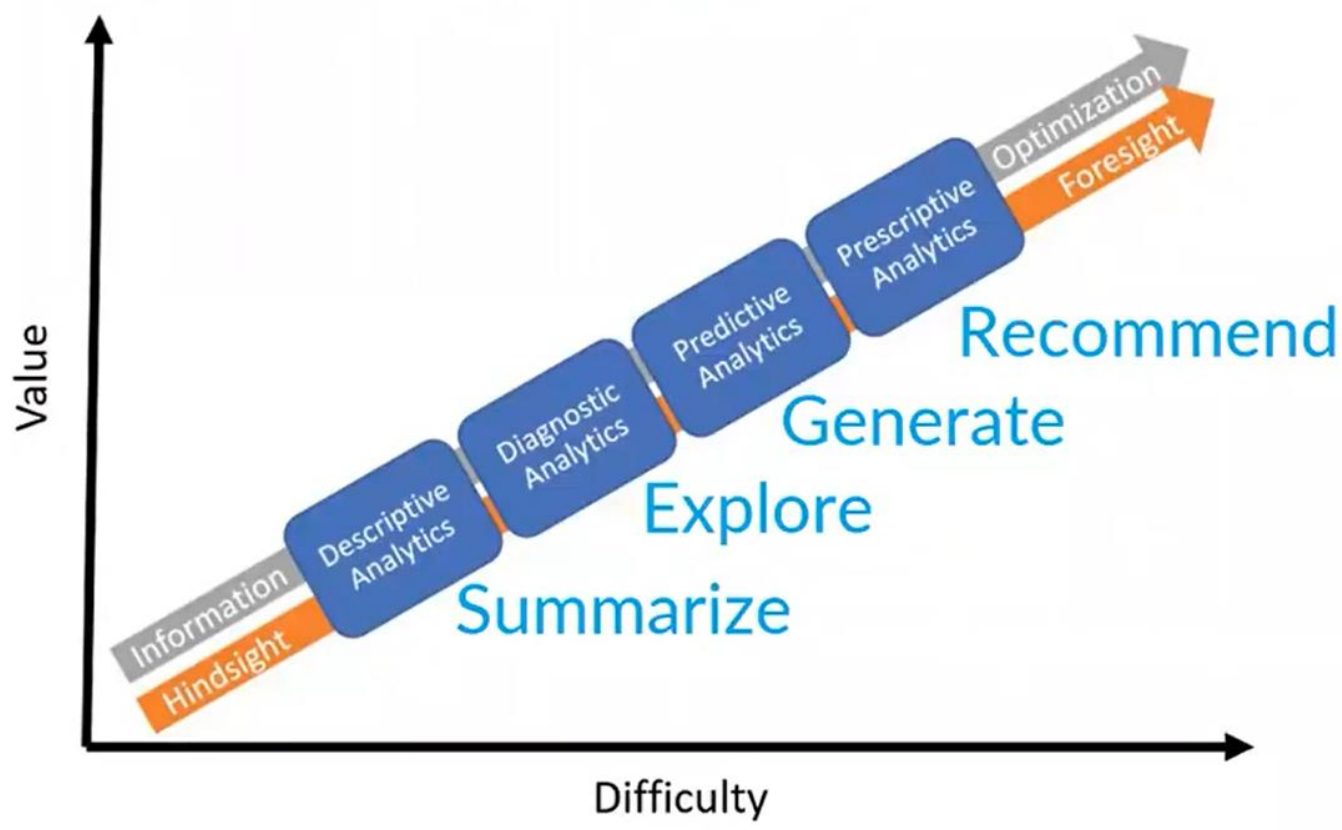


강의 scope



문제 유형에 따른 Data Understanding 방법

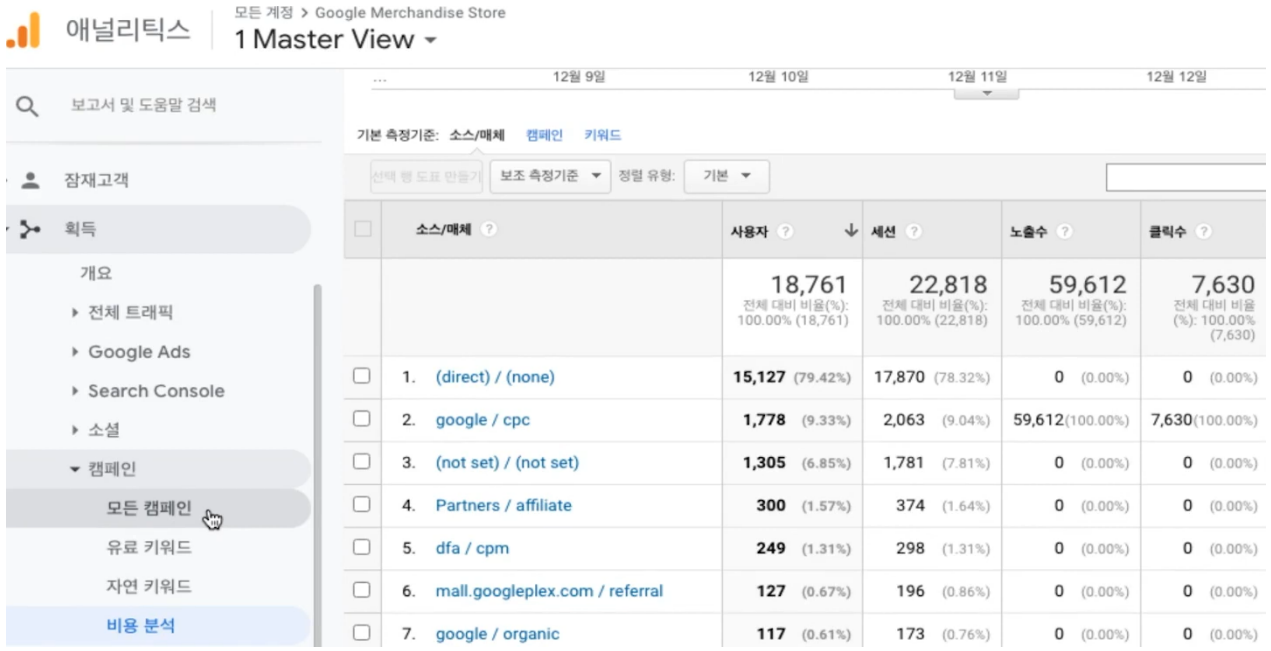
- 기술적 분석 (Descriptive Analysis)
- 진단적 분석 (Diagnostic Analysis)
- 예측 분석 (Predictive Analysis)
- 처방적 분석 (Prescriptive Analysis)



- 기술적 분석 (Descriptive Analysis)

  - score card 나 dashboard 사용
  - 무슨 일이 일어났는지 설명해 준다.
  - 결과 해석 및 어떻게 행동으로 옮길지 판단은 인간이 한다.

  Ex) Google Analytics - web/app 행동 데이터의 측정 및 분석 도구



- 진단적 분석 (Diagnostic Analysis)

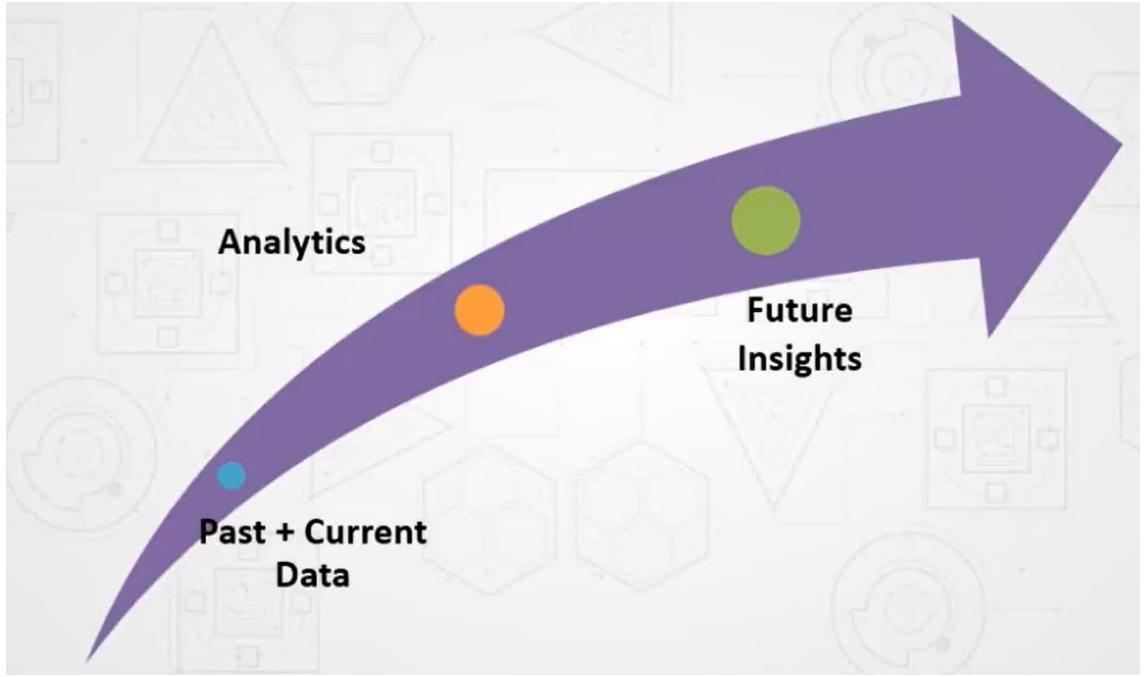
  - 과거에 축적된 data 에 의해 인과 관계를 찾아낸다.
  - 일이 발생한 원인을 밝힌다.
  - 결과 해석 및 어떻게 행동으로 옮길지 판단은 인간이 한다.

  Ex) Typical pattern 인식, Clustering, Matching, etc

- 예측 분석 (Predictive Analysis)

  - 통계학적 모델링을 방법 활용
  - 미래에 어떤 일이 어떤 정도의 확률로 발생할지를 예측

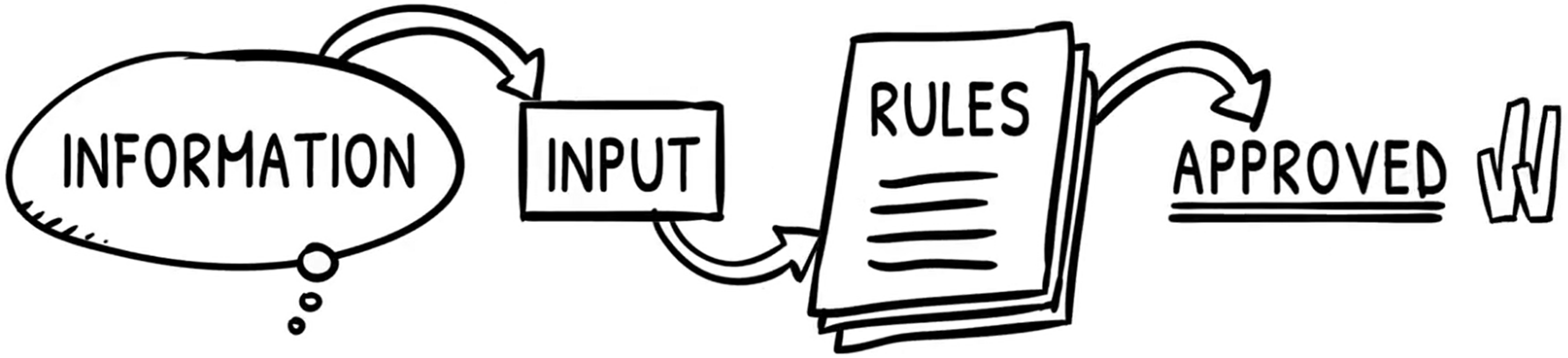
  ex) machine learning models



- 처방적 분석 (Prescriptive Analysis)

  - 예측되는 사태를 위해 무엇을 하면 좋은지 처방
  - 기술적 분석 + 진단적 분석 + 예측 분석을 조합
  - 다음에 취해야 할 최선의 행동을 인간 대신 고찰

  Ex) 금융기관의 자동 대출 승인



Data Pre-processing (Data 전처리)

- 데이터 분석을 위해 원시 데이터를 보다 편리하게 사용할 수 있는 다른 형식으로 변환

- Data cleaning , Data wrangling 이라고도 하며, 복잡하고 지저분한 상태의 데이터를 간단한 분석과 접근을 위해 통합하는 과정

- 데이터를 분석할 수 있는 상태로 조직화하는 필수 과정

EDA(Exploratory Data Analysis)

- 중요한 variable 과 중요하지 않은 variable 구분
- Outlier, missing value, dirty data 파악

  Garbage in garbage out

- Variable 간의 관계 유무 파악

Why Python for EDA ?

- 누구나 쉽게 배울 수 있는 프로그래밍 언어

- Full Featured Language
  - Data 획득, cleaning, database, 통계, 시각화, computing, Machine Learning 등 end-to-end 지원

- Strong Data Science Libraries
  - Pandas, scipy 등

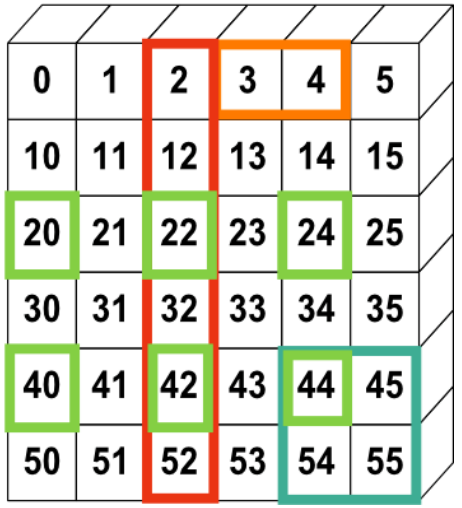
Python Tools for EDA

- Scientific Computing Library
  - Pandas (data structures & tools)
  - Numpy (arrays & matrices)
  - Scipy (과학기술계산 라이브러리)
- Visualization Library
  - Matplotlib (plots & graphs)
  - Seaborn (plots - easy to use)
- Algorithmic Library
  - Scikit-learn (machine learning)
  - Statsmodels (통계 library)

1. Numpy & Linear Algebra
2. Pandas
3. Matplotlib / Seaborn
4. Data Wrangling / Feature Scaling

Numpy


```
>>> a[0,3:5]
array([3,4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```



What is Numpy ?

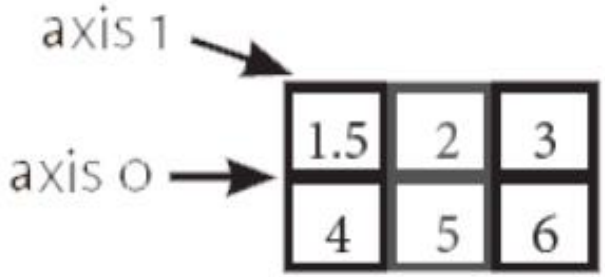
- Numerical Python
- 대용량 데이터 배열을 효율적으로 다룰 수 있도록 설계
- 선형대수, 매트릭스 연산 등의 함수 내장
- numpy 의 속도가 필요한 알고리즘은 C, C++ 로 작성

- numpy array
  - 1 차원 - vector, 2 차원 – matrix, 3 차원 이상 - tensor
  - rank – array 의 dimension (차원)
  - shape – 각 dimension 의 size
  - dtype – tensor 의 data type

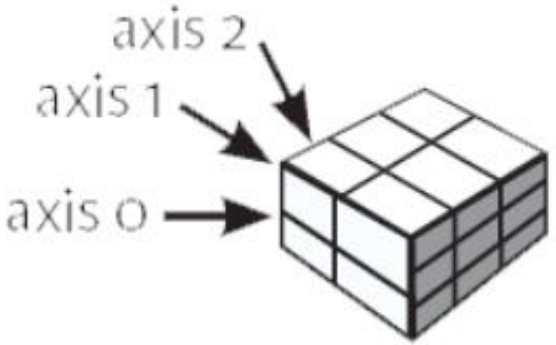
shape : (2,) (2, 2) (2, 2, 2)

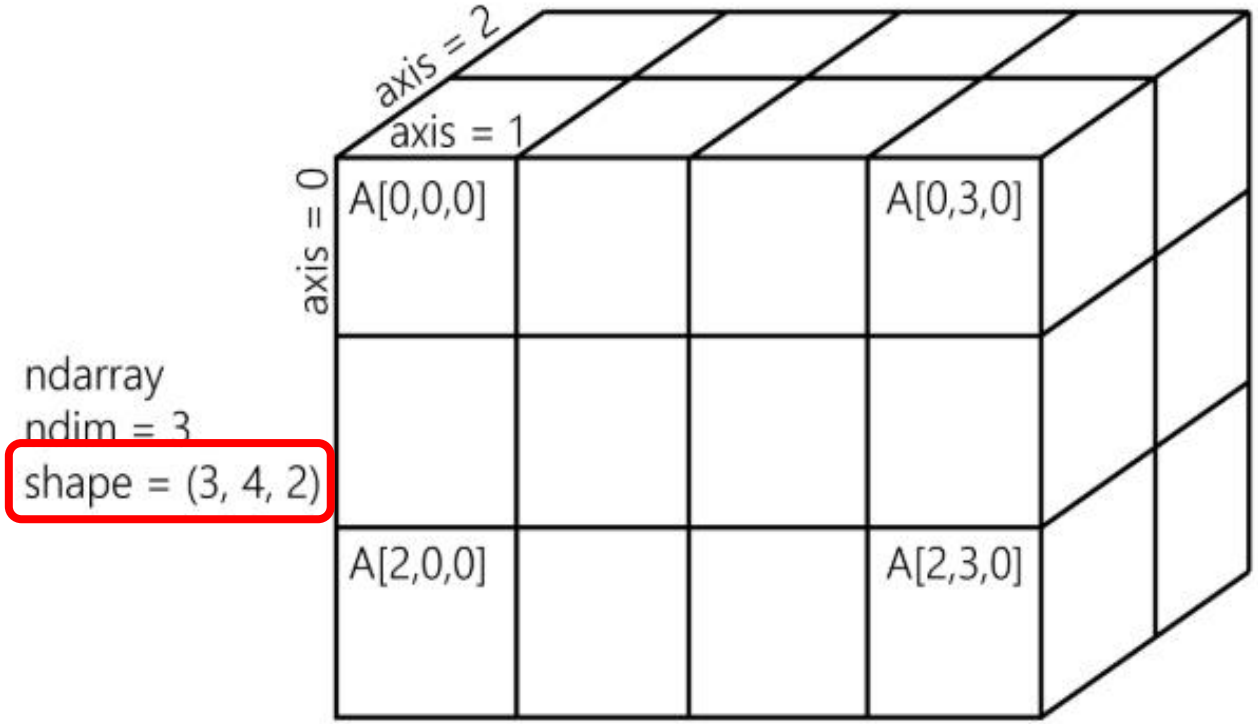
1D array

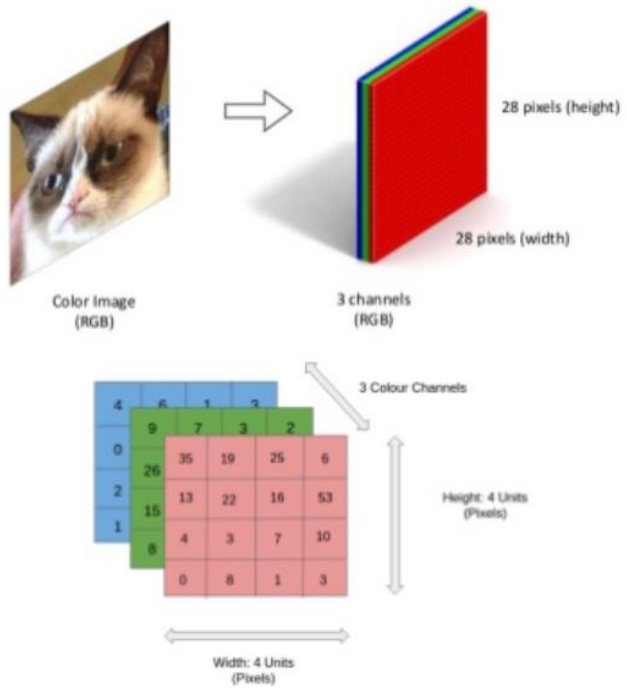
2D array



3D array





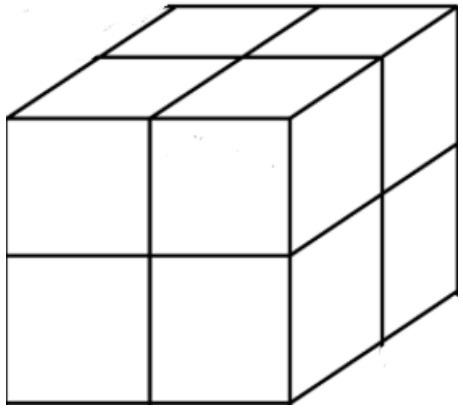


Matrix 의 numpy 표현

8	5	3
1	2	9

np.array([[8, 5, 3], [1, 2, 9]])

```
[[8 5 3]
 [1 2 9]]
```

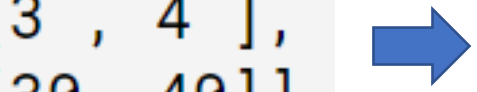


np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

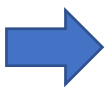
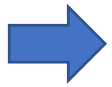
Joining & Slicing

- concatenate

```
[[1 , 2 ],
 [10, 20]]
```
+
```
[[3 , 4 ],
 [30, 40]]
```

```
[[1 , 2 ],
 [10, 20],
 [3 , 4 ],
 [30, 40]]
```
axis=0

```
[[1 , 2 , 3 , 4 ],
 [10, 20, 30, 40]]
```
axis=1

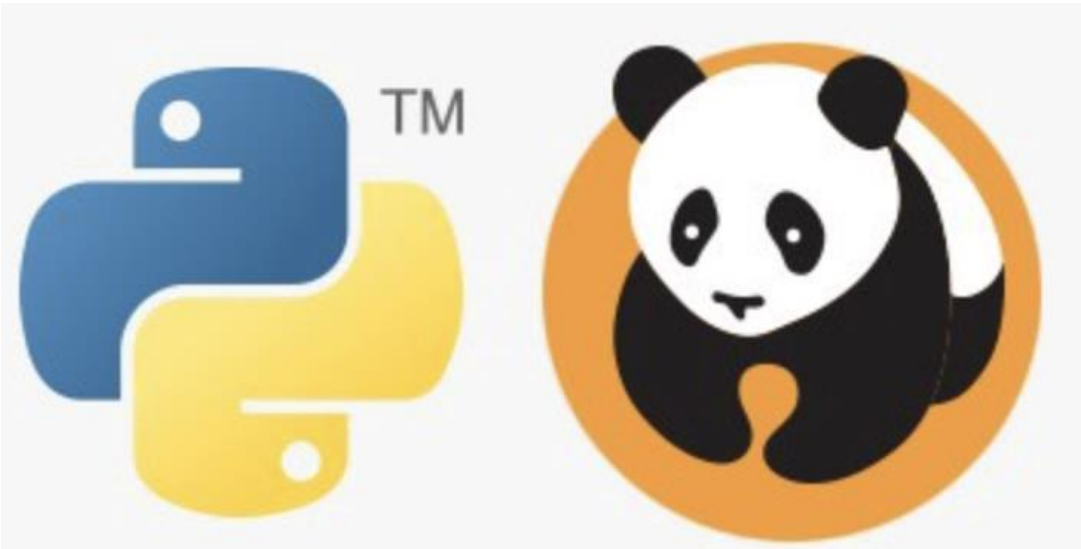
- slicing

```
a = np.array([[1,2], [3,4]])
a[1,:]
```

```
[[1 2]
 [3 4]]
```

```
[[3, 4],]
```

실습 – 002. Numpy 와 선형대수

- ndarray
- np.arange()¶
- np.linspace()
- matrix
- Indexing / Slicing
- matrix 간의 연산
- random value

Pandas



Pandas 기초

- Dataframe - R 을 모방하여 구현

- Tabular 형식의 Data 처리 (Excel 대체 가능)

	A	B	C	D	E	F	G	H	I
1	Order Date	OrderID	Salesperson	UK Units	UK Order Amt	USA Units	USA Order Amt	Total Units	Total Order Amt
2	1/01/2011	10392	Fuller			13	1440	13	1440
3	2/01/2011	10397	Gloucester	17	716.72			17	716.72
4	2/01/2011	10771	Bromley	18	344			18	344
5	3/01/2011	10393	Finchley			16	2556.95	16	2556.95
6	3/01/2011	10394	Finchley			10	442	10	442
7	3/01/2011	10395	Gillingham	9	2122.92			9	2122.92
8	6/01/2011	10396	Finchley			7	1903.8	7	1903.8
9	8/01/2011	10399	Callahan			17	1765.6	17	1765.6
10	8/01/2011	10404	Fuller			7	1591.25	7	1591.25
11	9/01/2011	10398	Fuller			11	2505.6	11	2505.6
12	9/01/2011	10403	Coghill	18	855.01			18	855.01
13	10/01/2011	10401	Finchley			7	3868.6	7	3868.6

Pandas basic 기능

- Series
- DataFrame
- Indexing / slicing
- Missing Data 처리

실습 : 004. Pandas Basic

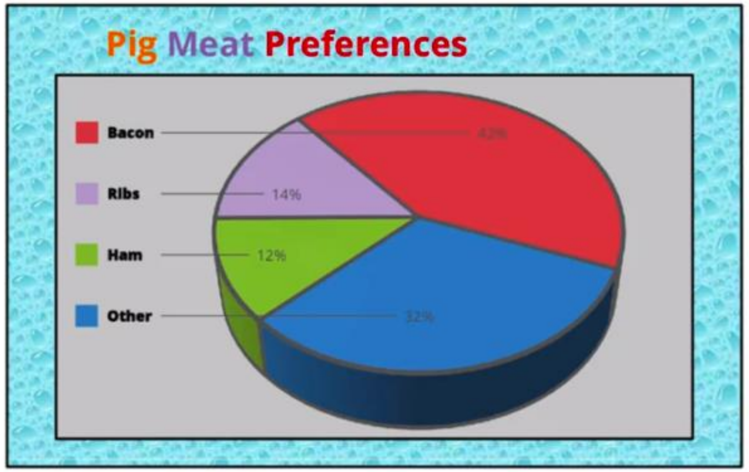
- Series
- DataFrame
- indexing, slicing
- Missing Data 처리 등

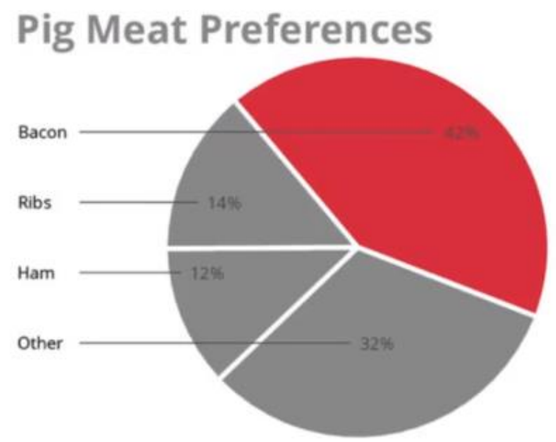
실습 : 005. Pandas Advanced

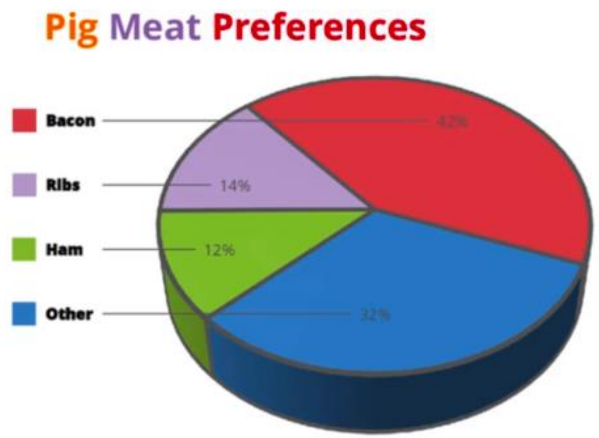
- csv file handling
- apply + lambda
- Boolean indexing
- 복사본 만들기 (shallow copy, deep copy)
- 다중 indexing
- method chaining
- group by

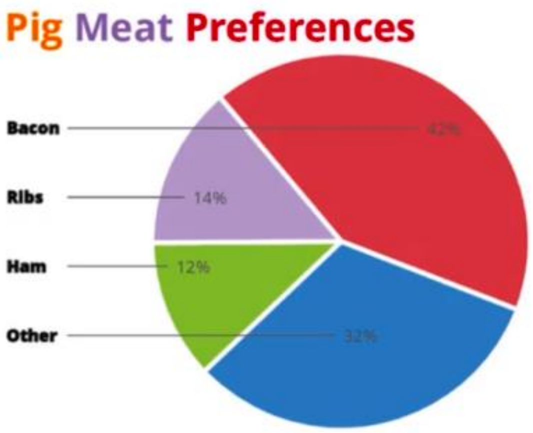
시각화 도구

Best Practice of Visualization

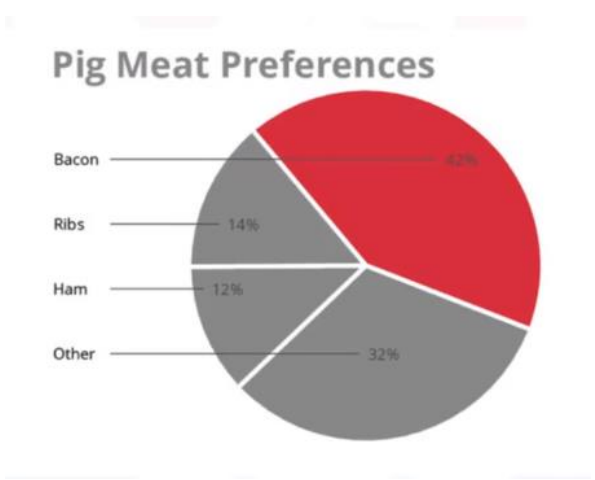




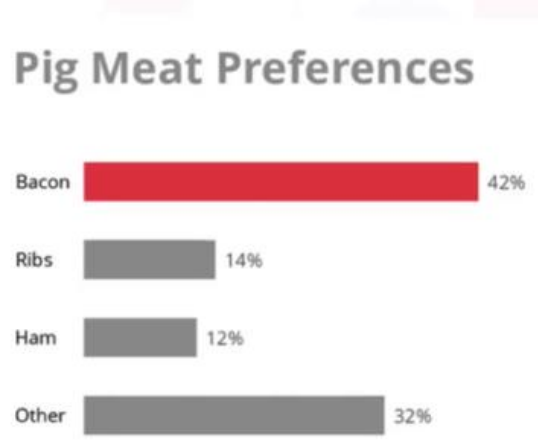


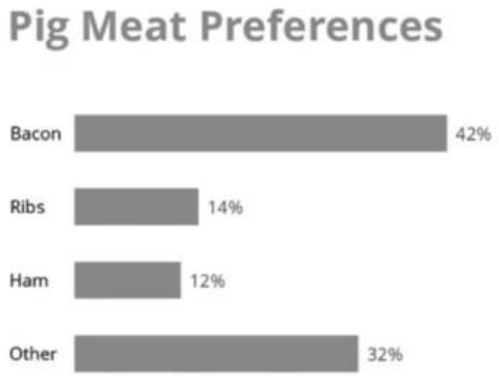


Best Practice of Visualization

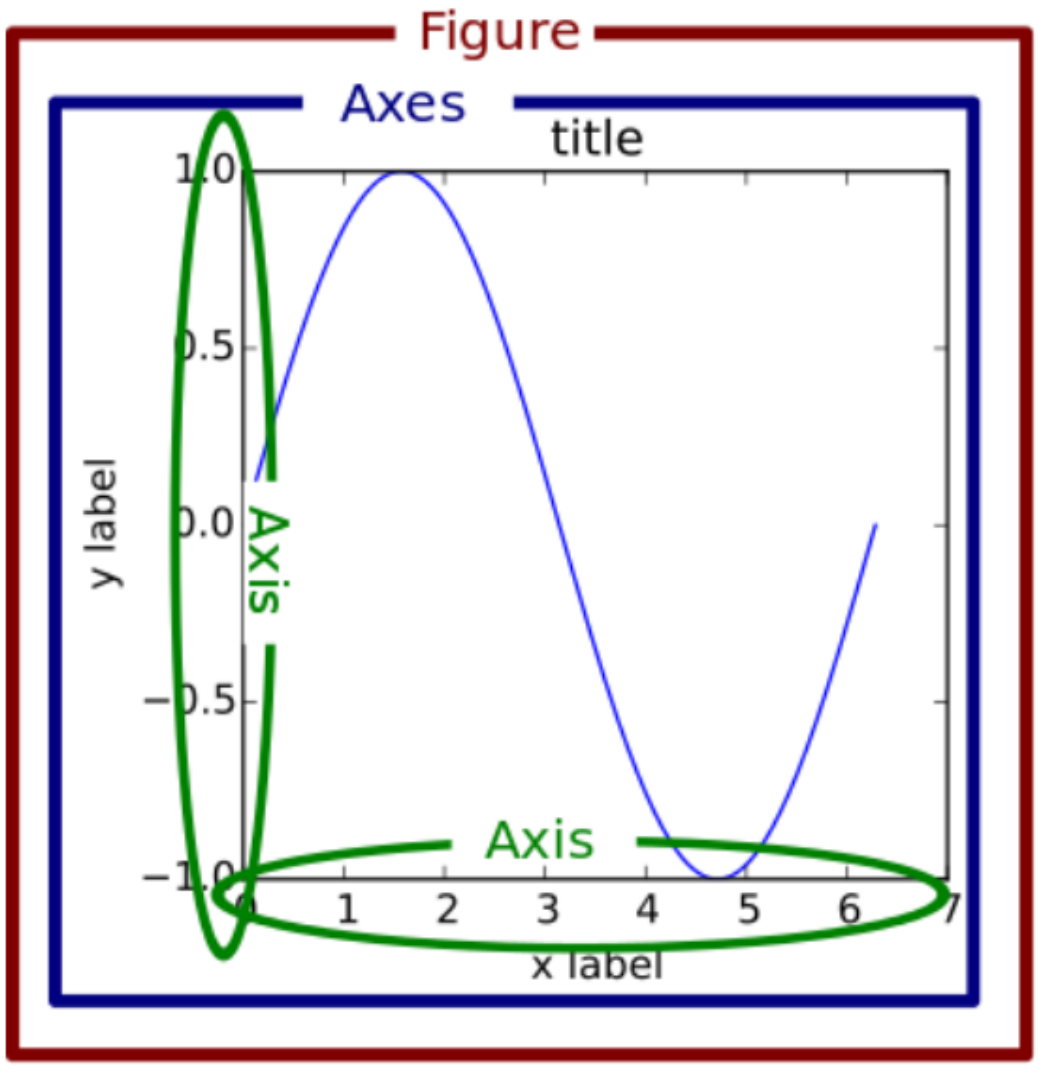


Final



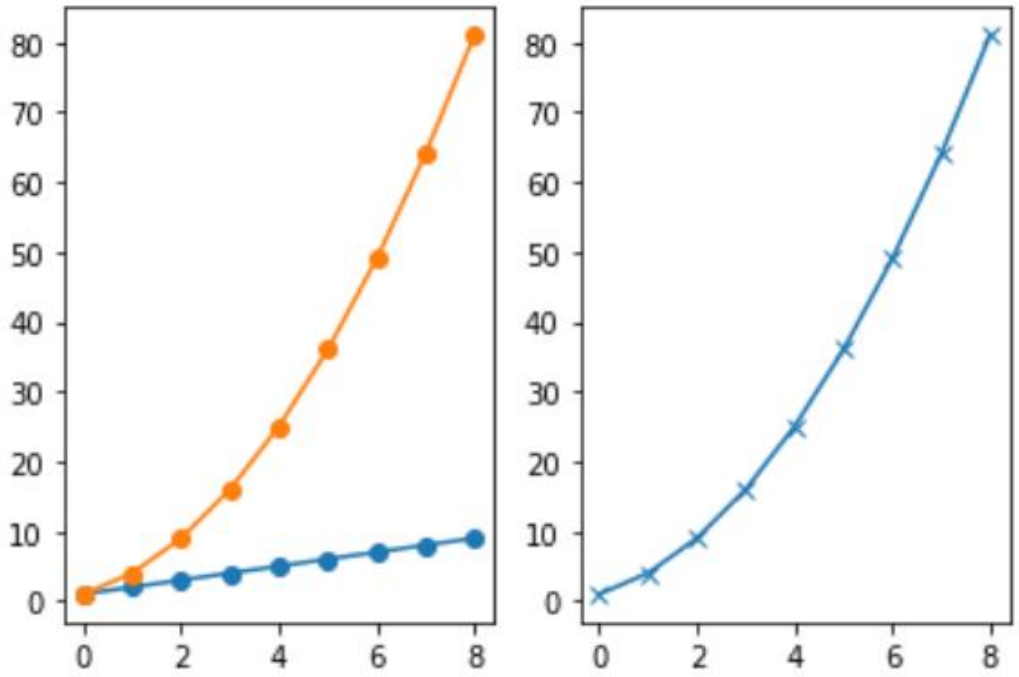


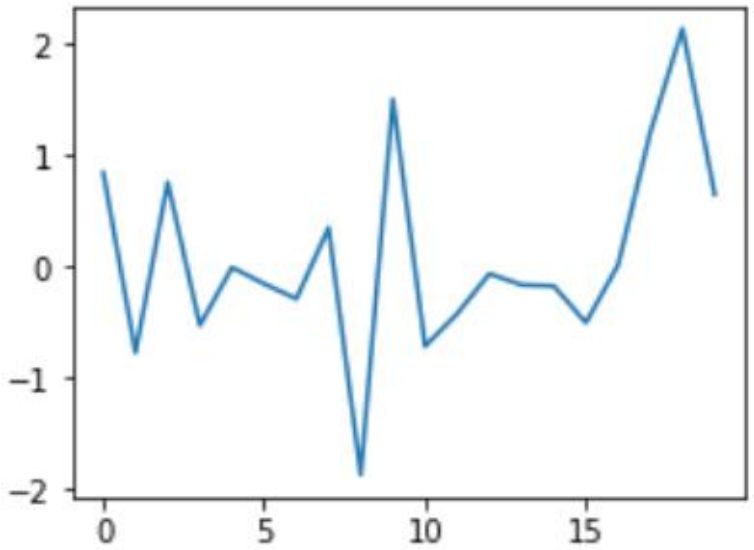
Matplotlib – 대표적 시각화 도구

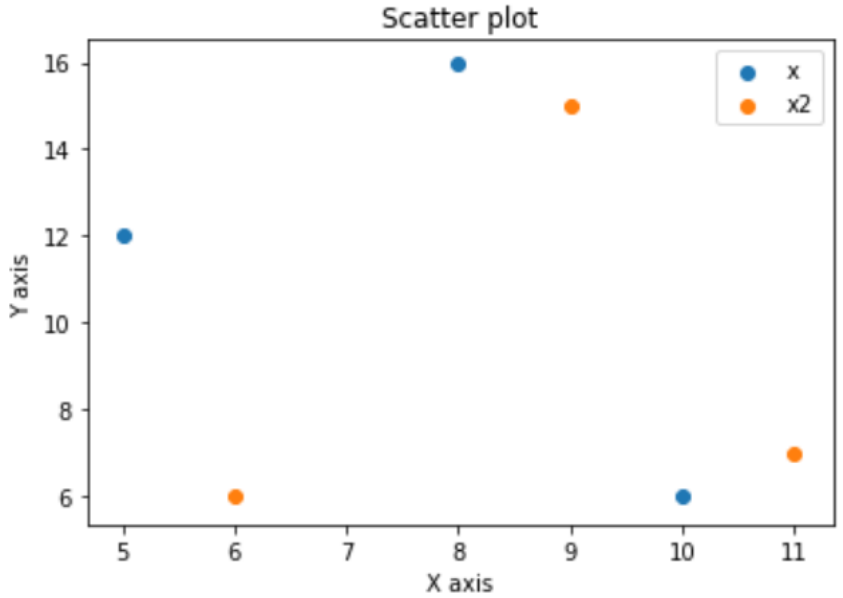


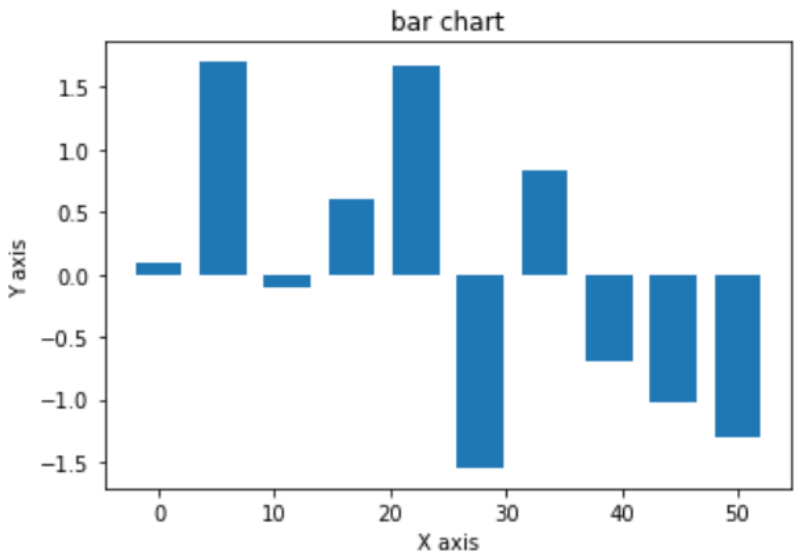
subplot

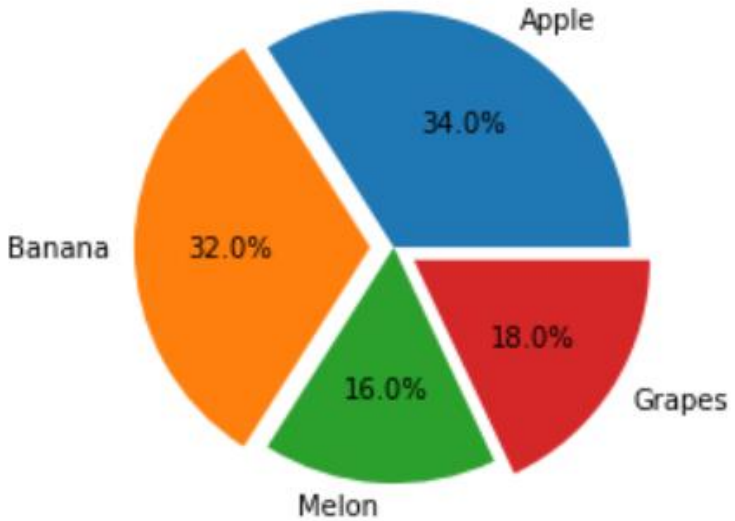
```
linear_data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
exponential_data = linear_data ** 2

plt.subplot(1, 2, 1)
plt.plot(linear_data, '-o')
plt.plot(exponential_data, '-o')

plt.subplot(1, 2, 2)
plt.plot(exponential_data, '-x')
```




2
1
0
−1
−2
0
5
10
15


Scatter plot
x
x2
Y axis
X axis
16
14
12
10
8
6
5
6
7
8
9
10
11


bar chart
Y axis
X axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50


Apple
34.0%
Banana
32.0%
18.0%
16.0%
Grapes
Melon

실습 : 008. Matplotlib Basic

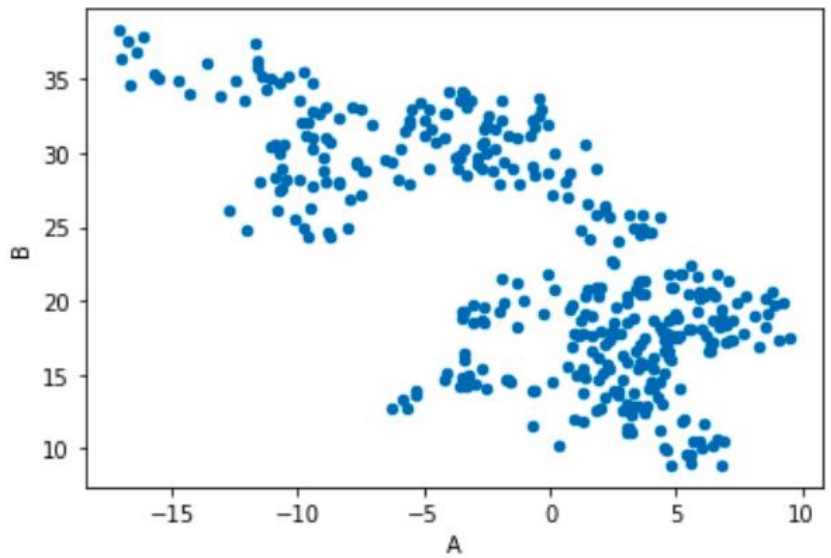
- Matplotlib 기초
- Subplots
- gridspec
- Word Cloud

Pandas Built-in 시각화 기능

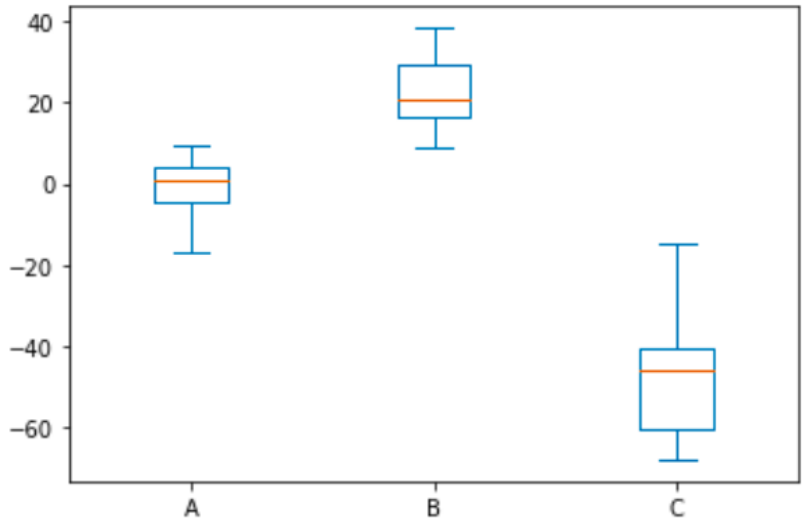
- 플롯을 생성하는 두 가지 방법
  - data_frame['column_name'].plot(kind='plot_type')
  - data_frame['column_name'].plot.plot_type_function()

- `line` (default)
- `bar` : vertical bar plots
- `barh` : horizontal bar plots
- `hist` : histogram
- `box` : boxplot
- `kde` or `density` : density plots
- `area` : area plots
- `pie` : pie plots
- `scatter` : scatter plots
- `hexbin` : hexbin plot

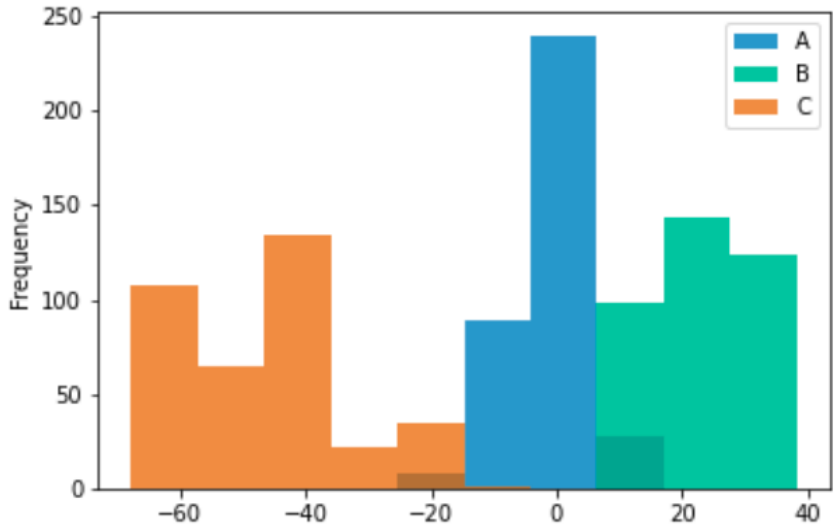
```
df.plot('A', 'B', kind='scatter');
```



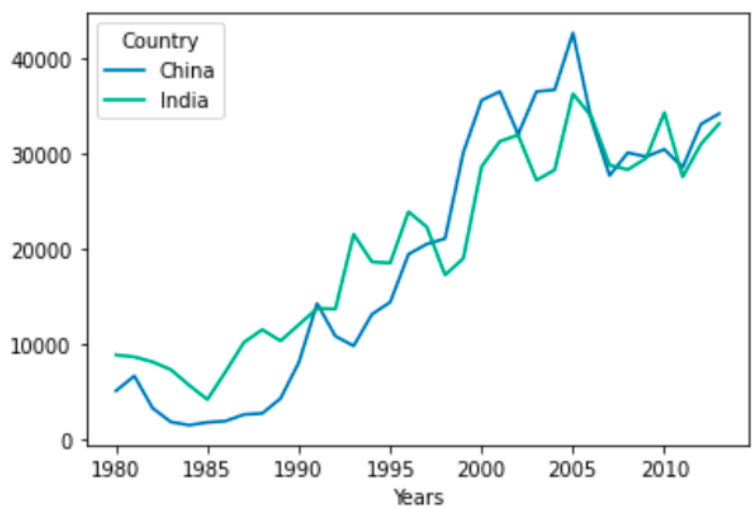

```
df.plot.box();
```

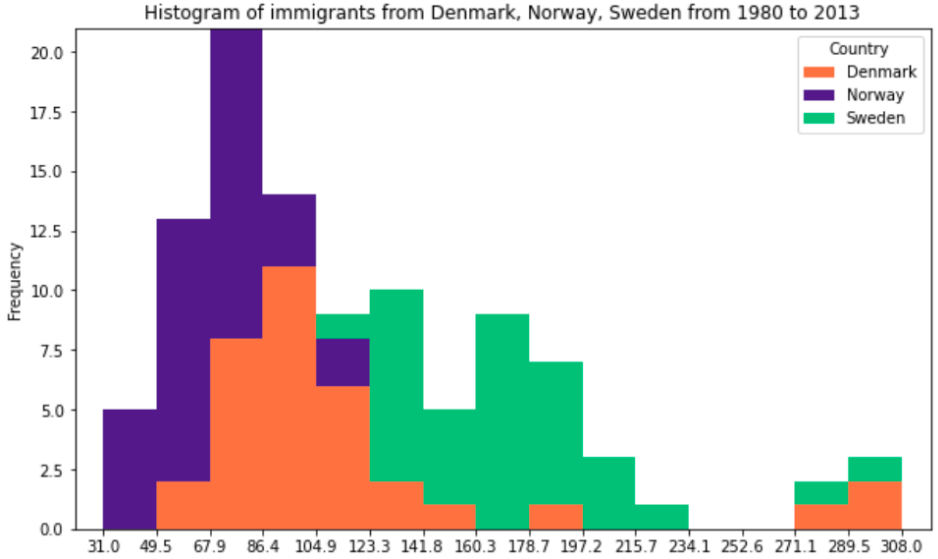


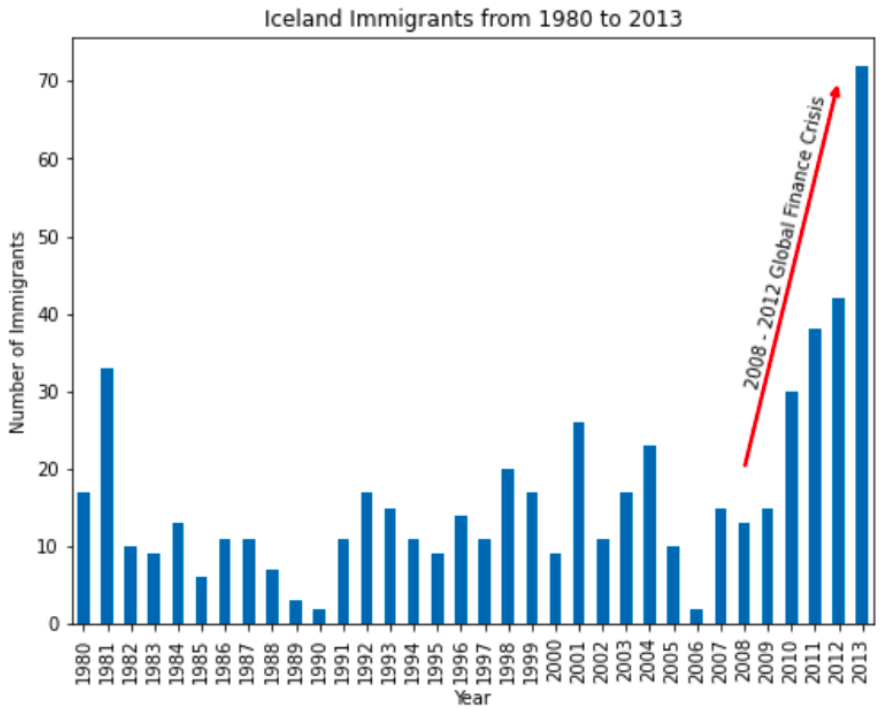

```
df.plot.hist(alpha=0.7);
```

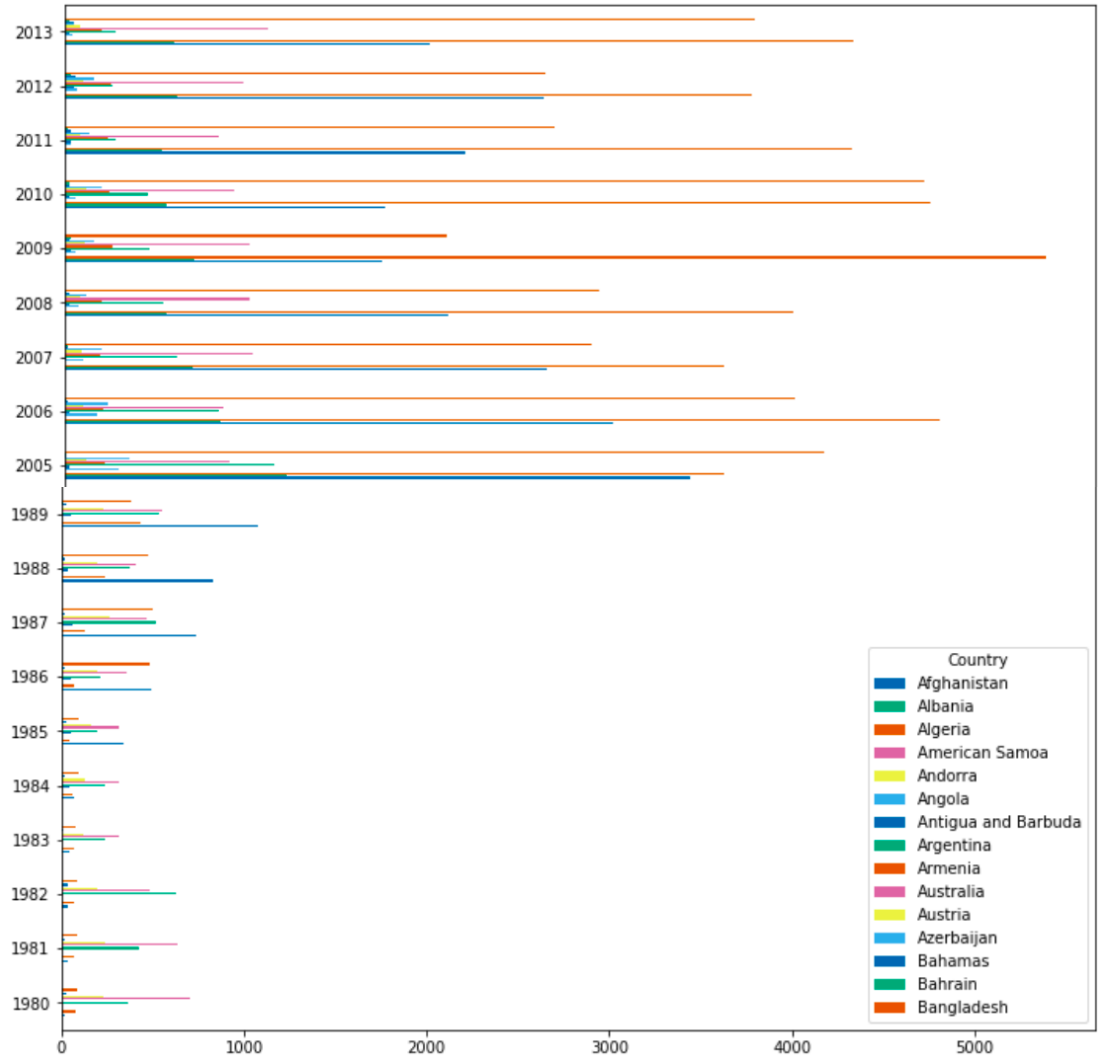



```
df_ci.plot(kind='line')
plt.xticks(df_ci.index[::5])   # 5 년 단위 xtick
plt.xlabel('Years')
plt.show()
```




Histogram of immigrants from Denmark, Norway, Sweden from 1980 to 2013
Country
Denmark
Norway
Sweden
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
31.0
49.5
67.9
86.4
104.9
123.3
141.8
160.3
178.7
197.2
215.7
234.1
252.6
271.1
289.5
308.0


Iceland Immigrants from 1980 to 2013
Number of Immigrants
Year
2008 - 2012 Global Finance Crisis
0
10
20
30
40
50
60
70
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013


Country
Afghanistan
Albania
Algeria
American Samoa
Andorra
Angola
Antigua and Barbuda
Argentina
Armenia
Australia
Austria
Azerbaijan
Bahamas
Bahrain
Bangladesh
2013
2012
2011
2010
2009
2008
2007
2006
2005
1989
1988
1987
1986
1985
1984
1983
1982
1981
1980
0
1000
2000
3000
4000
5000

실습 : 009. Pandas 내장 시각화

- DataFrame/Series plot
- Canada Immigration Dataset 분석 및 시각화
- Waffle Chart

Remind of Statistics

Random Variables

Discrete

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Continuous

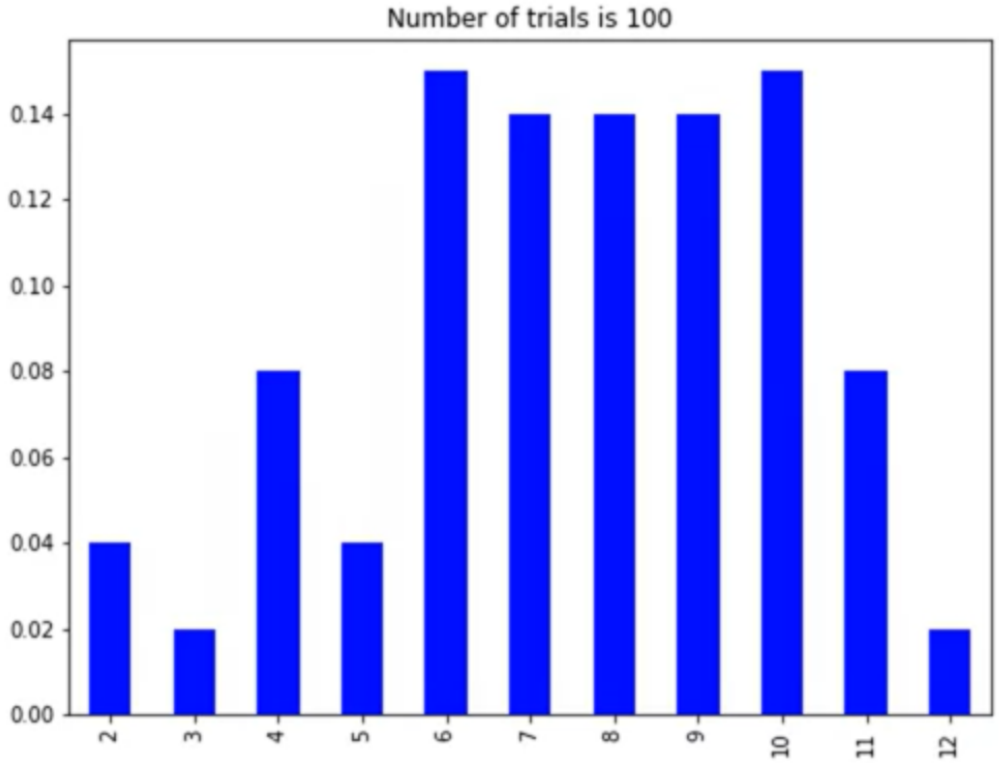
1.5% -4.558%

-8.8756% 13.48%

...

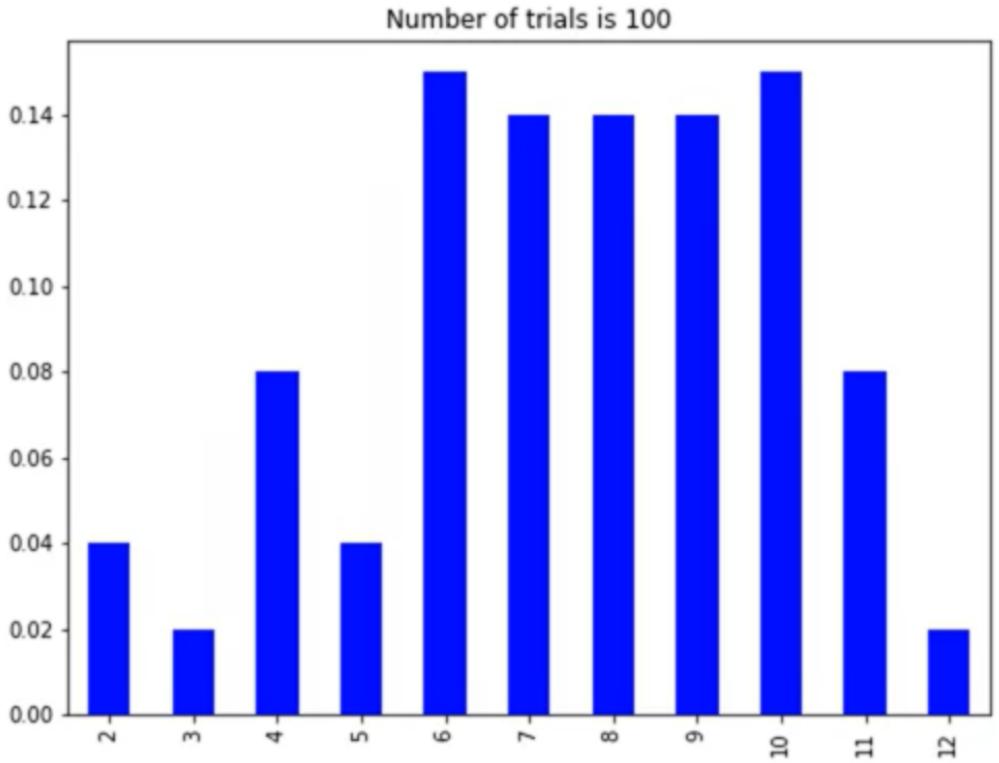
Frequency (빈도수)

주사위 2 개의
합이 나온 횟수

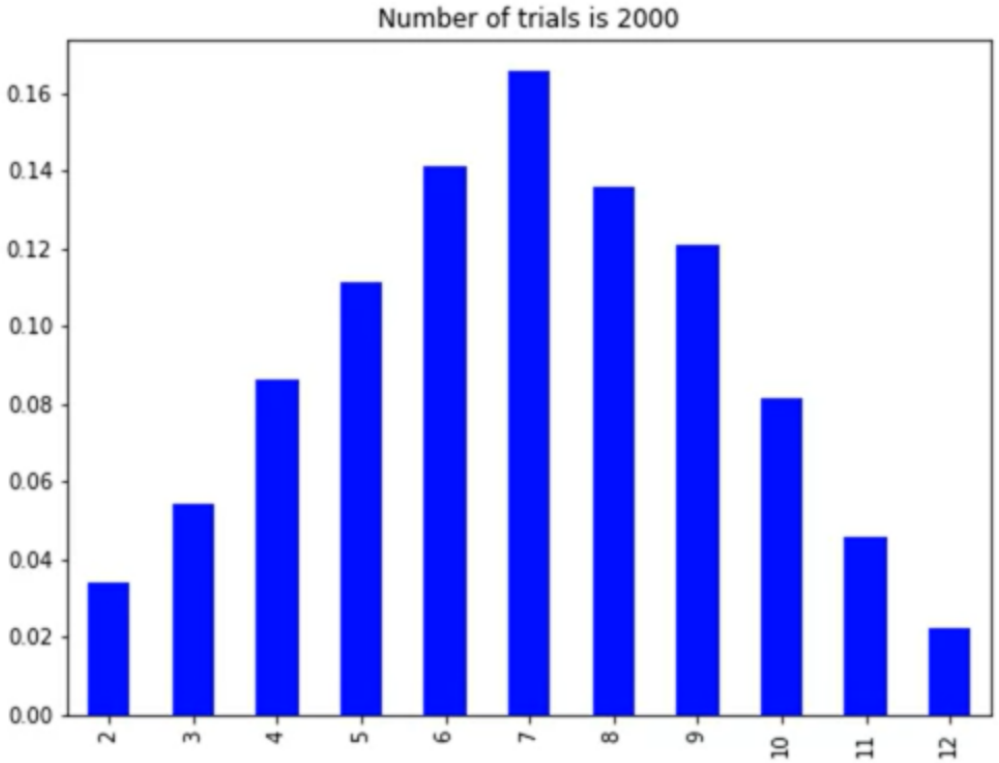


주사위의 합

상대 빈도
= 빈도수 / 시행 횟수



100 회



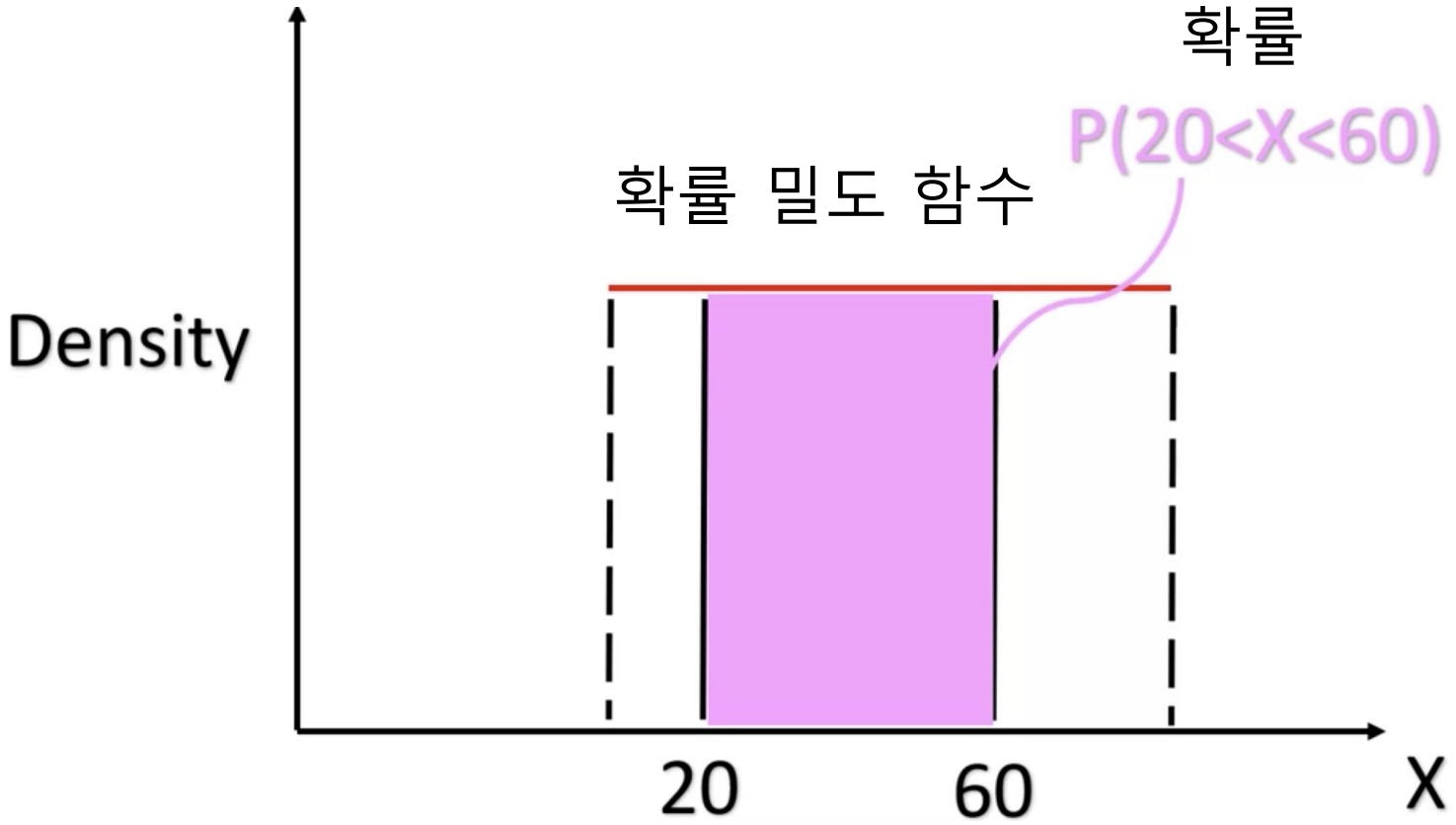
2000 회

이산 확률 변수의 분포

X=	2	3	4	5	6	7	8	9	10	11	12
P(X)	$1\times\left(\frac{1}{6}\right)^2$	$2\times\left(\frac{1}{6}\right)^2$	$3\times\left(\frac{1}{6}\right)^2$	$4\times\left(\frac{1}{6}\right)^2$	$5\times\left(\frac{1}{6}\right)^2$	$6\times\left(\frac{1}{6}\right)^2$	$5\times\left(\frac{1}{6}\right)^2$	$4\times\left(\frac{1}{6}\right)^2$	$3\times\left(\frac{1}{6}\right)^2$	$2\times\left(\frac{1}{6}\right)^2$	$1\times\left(\frac{1}{6}\right)^2$
	(1, 1)	(1, 2) (2, 1)	(1, 3) (3, 1) (2, 2)							(6, 5) (5, 6)	(6, 6)

- Mean = Expectation = $\sum_i p_i x_i$
- Variance = $\sum_i (x_i - Mean)^2 p_i$

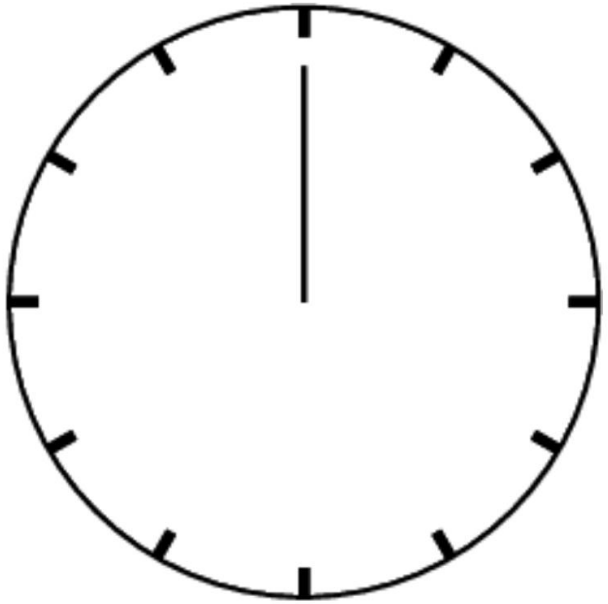
연속 확률 변수의 분포



Distribution of continuous random variable

- 시계 바늘이 정각 12시(각도 0도)를 가리킬 확률 = 0

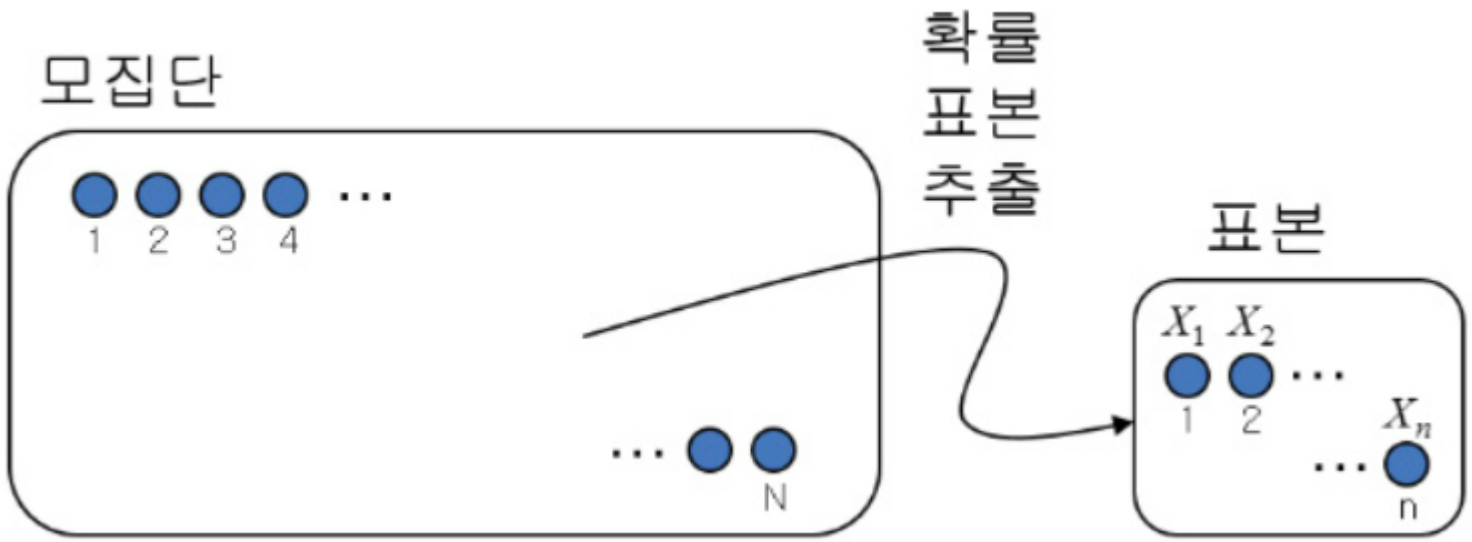
  → 0~360 도 사이에 실수가 무한대 경우 있으므로
  각 경우의 확률은 0



- 시계 바늘이 12시와 1시 사이에 있을 확률 1/12
- 시계 바늘이 1시와 3시 사이에 있을 확률 2/12 = 1/6
- 시계 바늘이 6시와 9시 사이에 있을 확률 3/12 = 1/4

모집단(Population) 과 표본집단 (Sample)

- 모집단 : 알고 싶은 대상 전체
- 표본집단 : 측정 또는 관찰한 결과들의 집합



모수(Parameter) 와 통계량(Statistics)

- 모수 – 모집단의 특성을 수치로 나타낸 것
  - 모평균 (μ), 모분산 (σ^2)
- 통계량 – 표본의 특성을 수치로 나타낸 것
  - 표본평균 ($\bar{X}$), 표본분산(S^2)

자유도 (Degrees of Freedom)

- 계산 중 자유롭게 바꿀 수 있는 값의 갯수

$$n = 3, \bar{x} = 100$$

x_1 x_2 x_3

90 80 Not Free!

$$\frac{90 + 80 + x_3}{3} = 100$$

$$x_3 = 130$$

x1, x2는 any number 를 선택 가능

평균이 100 이 나오게 하려면 x3 는 130 이어야 함

정규분포 (Normal Distribution)

- Gaussian Distribution
- μ 와 σ^2 에 따른 확률밀도함수

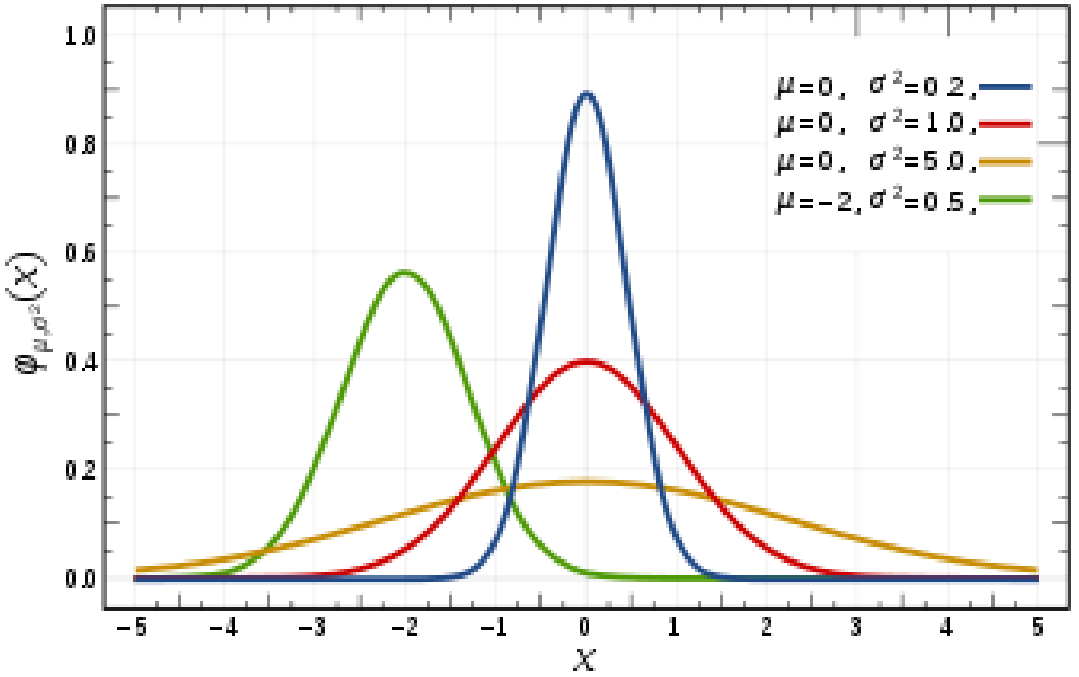
  $X \sim N(\mu, \sigma^2)$

- μ : 분포의 중심 ($-\infty < \mu < \infty$)
  (평균, 중앙값, 최빈값)

- σ^2 : 퍼져있는 정도 (분산)
  (σ : 표준편차)

$$mean = \sum_i x_i p_i$$

$$variance = \sum_i (x_i - mean)^2 p_i$$



표준정규분포 (Standard Normal Distribution)

- $\mu = 0$ 와 $\sigma^2 = 1$ 인 경우 0 을 중심으로 대칭을 이루므로 계산 편리

- Z 으로 표시 : $Z \sim N(0, 1)$

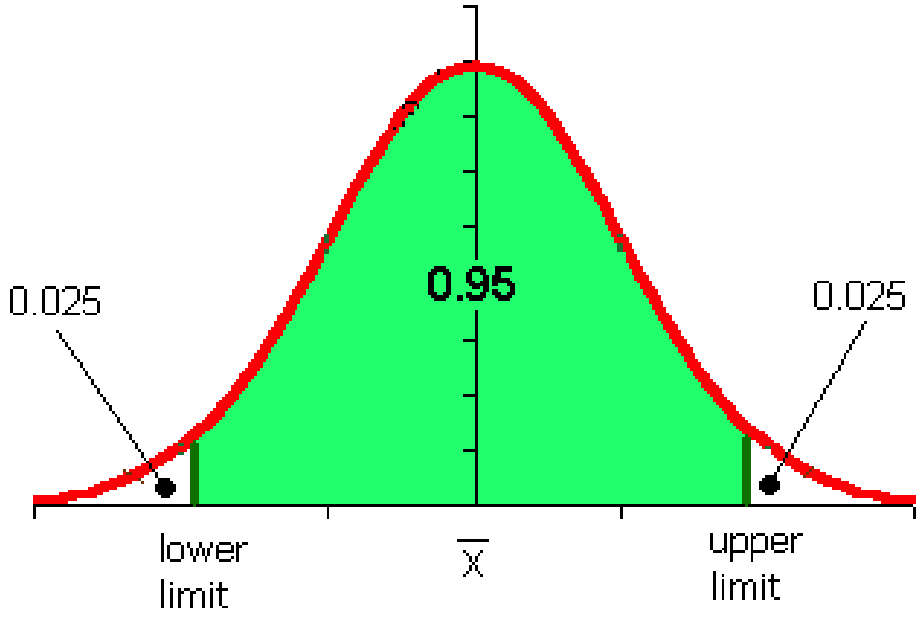
- 표준정규분포표를 이용하여 확률 계산

신뢰구간 (confidence interval)

- 모평균을 포함할 확률이 xx % 가 되는 구간

  ("interval_left" 와 "interval_right" 사이에 average stock return 이 위치할 확률이 xx %)

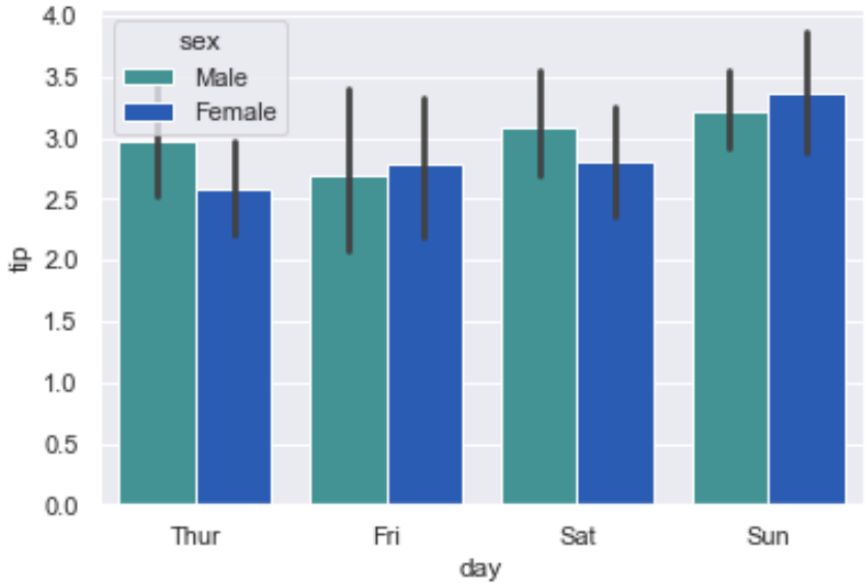
  - z_left = norm.ppf(0.025)
  - z_right = norm.ppf(0.975)
  - interval_left = sample_mean + z_left * sample_std
  - interval_right = sample_mean + z_right * sample_std

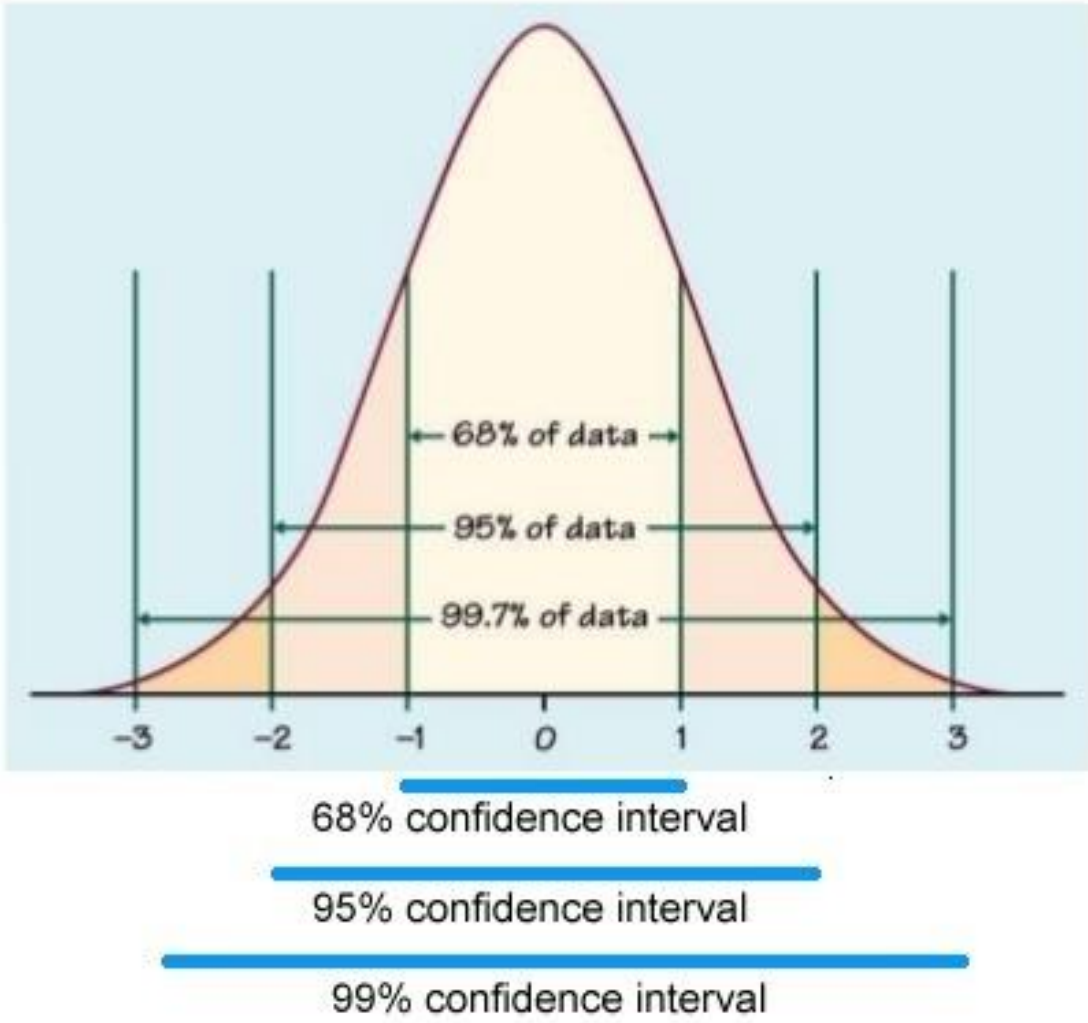


신뢰구간 (confidence interval)

- Empirical Rule

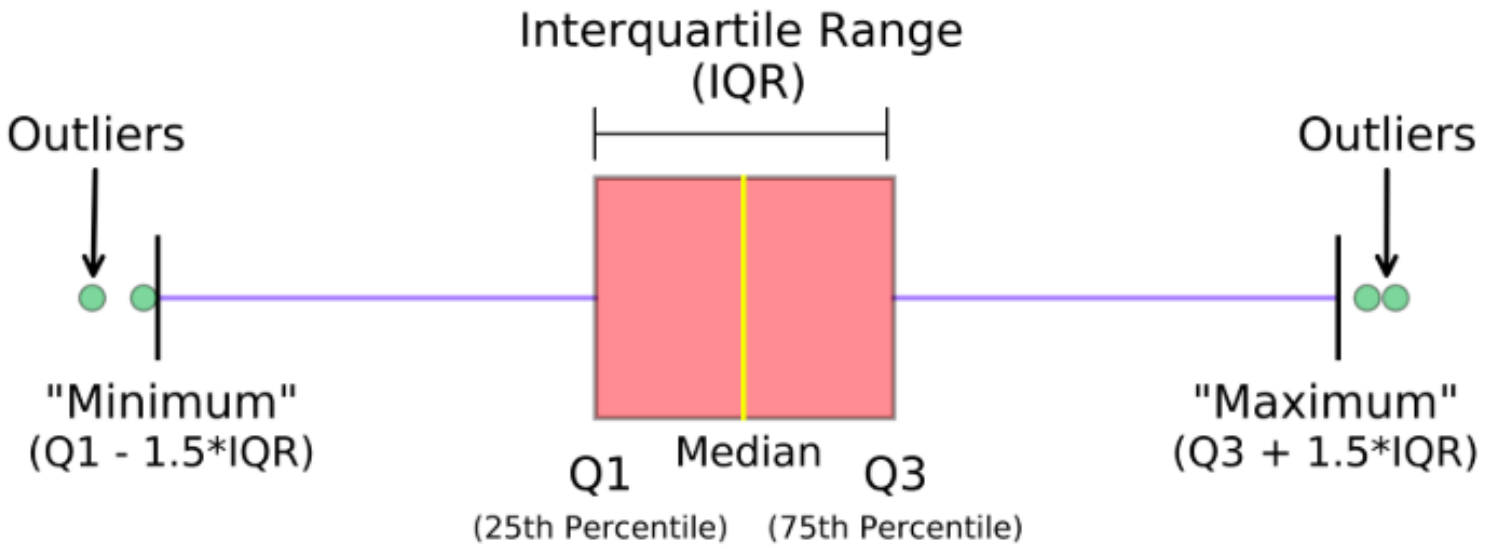
  1 σ - 68%, 1.96 σ - 95%, 3 σ - 99.7%

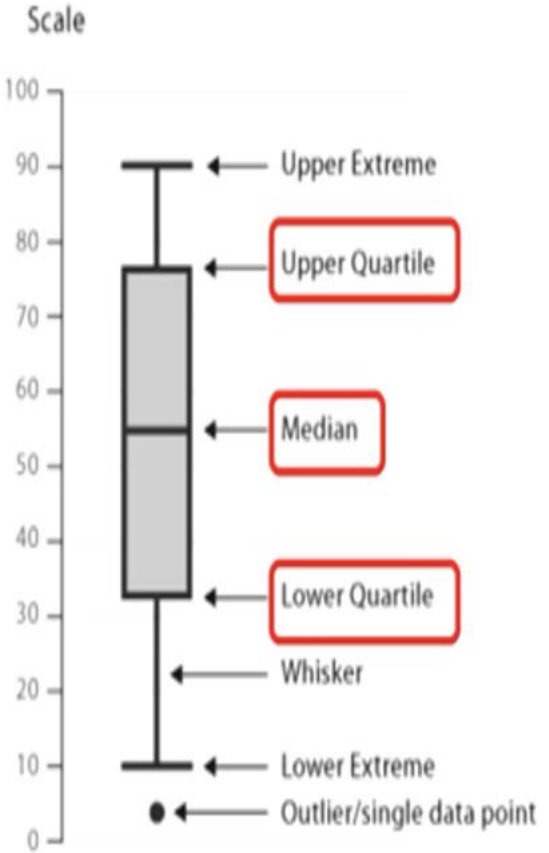




Boxplot

- 4 분위수





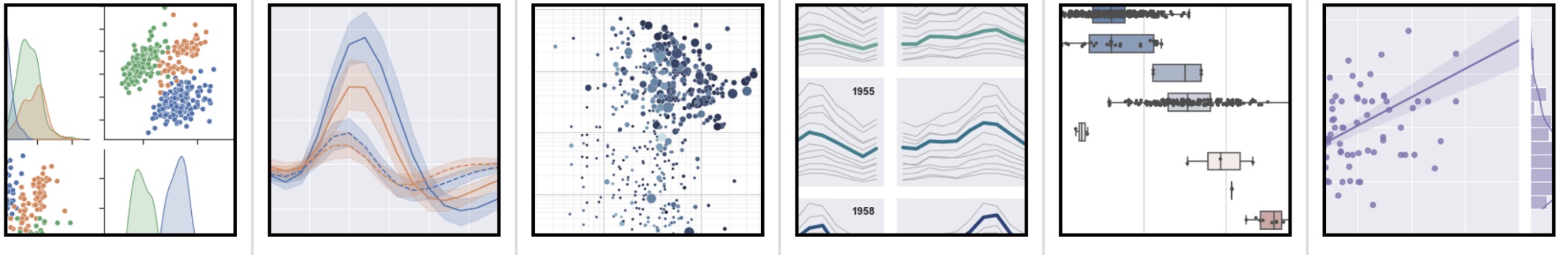
실습 – 010. Pandas를 활용한 통계 기초

- 이산/연속 확률 변수
- 주사위 게임 모방을 통한 빈도 및 분포 simulation
- 평균, 분산, 확률 분포 시각화
- 신뢰 구간

Seaborn – 시각화 Package

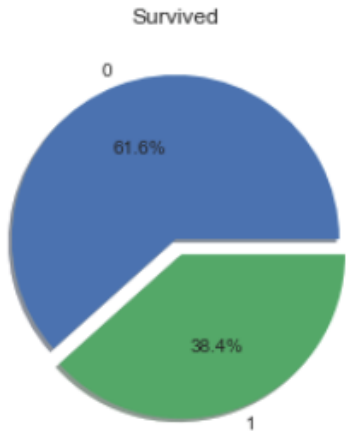
- Seaborn은 Matplotlib을 기반으로 다양한 색상 테마와 통계용 차트 등의 기능을 추가한 시각화 패키지

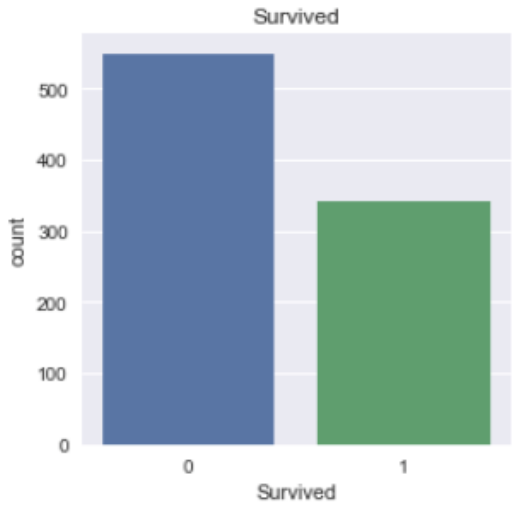
- 기본적인 시각화 기능은 Matplotlib 패키지에 의존하며 통계 기능은 Statsmodels 패키지에 의존

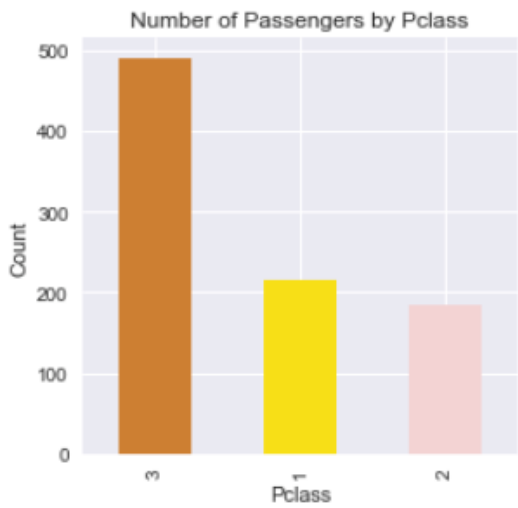


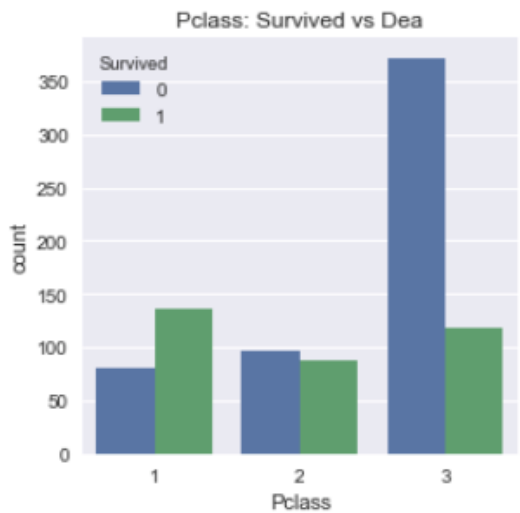
Seaborn 기본 기능

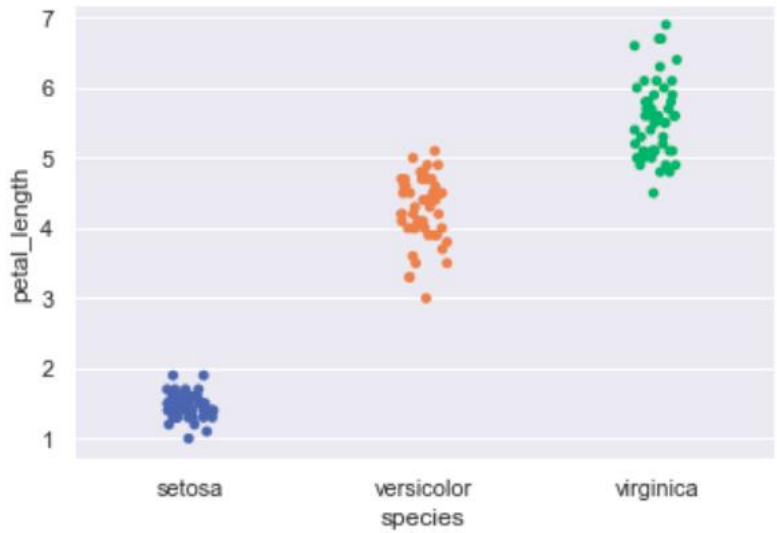
- Matplotlib 과 Seaborn 의 미적인 측면 비교
- 통계적 추론 (Statistical Estimation) 과 관련된 plot 들
- 변수 한 개 (Univariate) 의 분포를 그리는 기능
- 변수 두 개(bivariate) 분포를 그리는 기능
- Categorical Data plotting

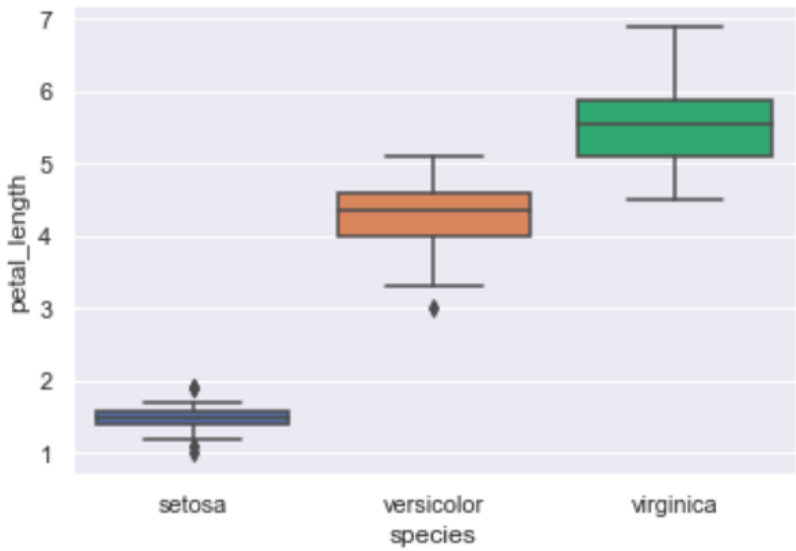

Survived
0
61.6%
38.4%
1

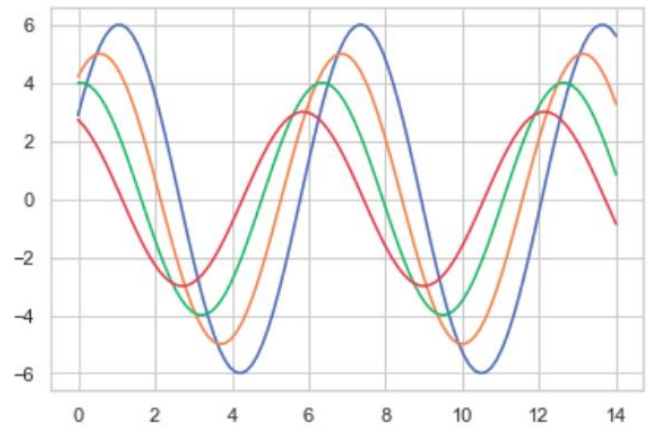

Survived
count
Survived

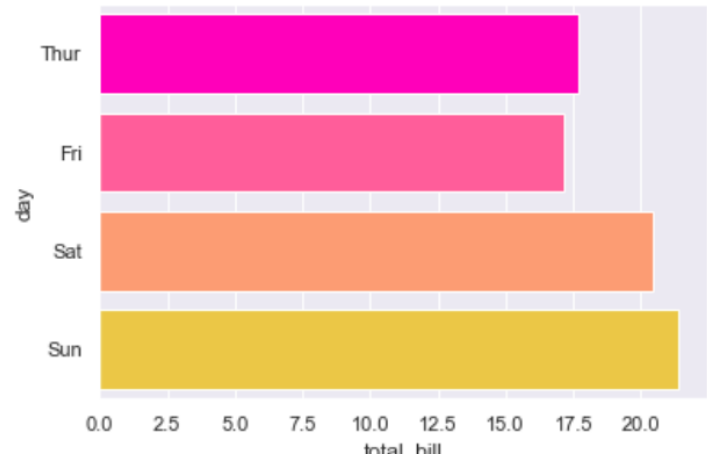

Number of Passengers by Pclass
Count
Pclass

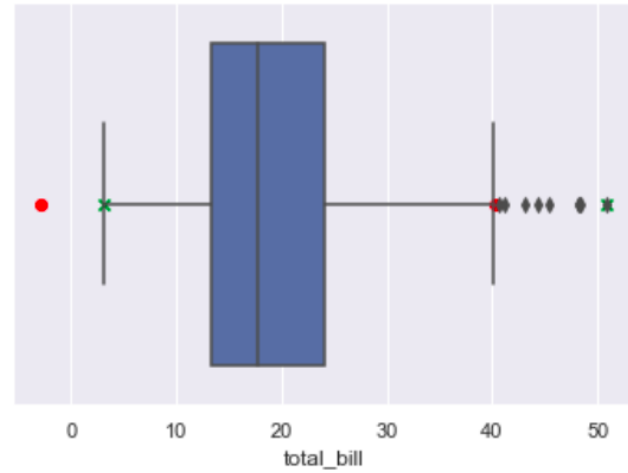

Pclass: Survived vs Dea
Survived
0
1
count
Pclass

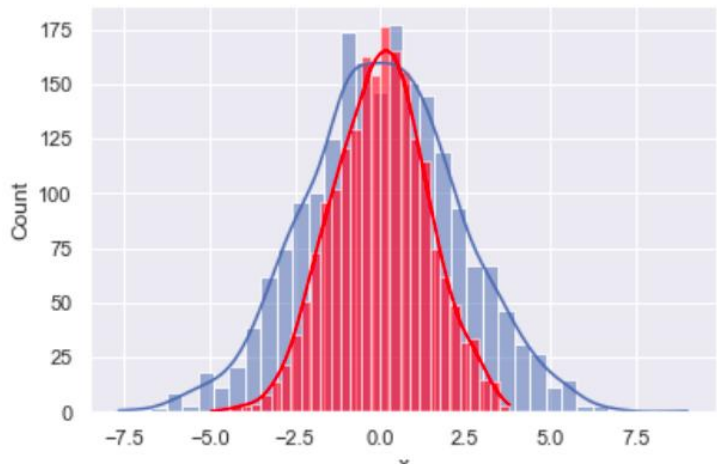

petal_length
setosa
versicolor
virginica
species

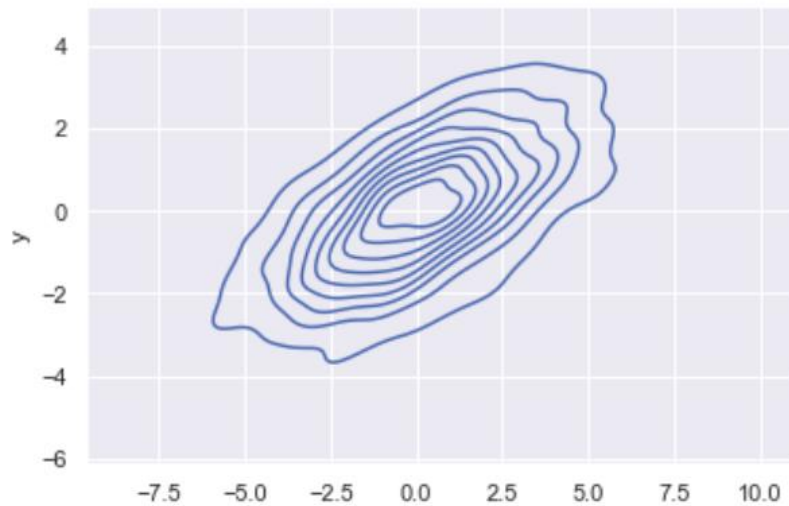

petal_length
setosa
versicolor
virginica
species

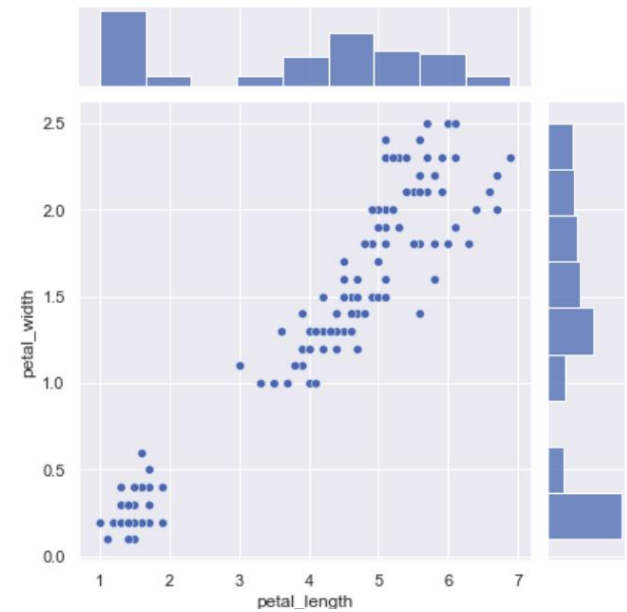

6
4
2
0
−2
−4
−6
0
2
4
6
8
10
12
14


Thur
Fri
Sat
Sun
day
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
total_bill


0
10
20
30
40
50
total_bill


175
150
125
100
75
50
25
0
Count
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5


4
2
0
−2
−4
−6
y
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


2.5
2.0
1.5
1.0
0.5
0.0
petal_width
1
2
3
4
5
6
7
petal_length

실습 – 012. Seaborn Introduction

- Matplotlib 과 Seaborn 의 미적인 측면 비교
- 통계적 추론과 관련된 plot 들
- 단변수 / 다변수 시각화
- Categorical Data Plotting

Pandas 를 이용한
EDA (Exploratory Data Analysis)

탐색적 데이터 분석 (EDA)

- Dataset 을 분석하고 특징을 파악
  - 기본적인 통계 분석
  - 시각화

- Descriptive Statistics (기술 통계)– overall statistics
  - df.head(), df.shape, df.info()

- Univariate Statistics (단변량 통계) – single feature
  - df.describe(), hist(df['feature']), df['feature'].value_counts()

- Multivariate Statistics (다변량 통계) – 하나 이상의 feature
  - df.plot.scatter(feature1, feature2), df[['feature1', 'feature2']].corr()

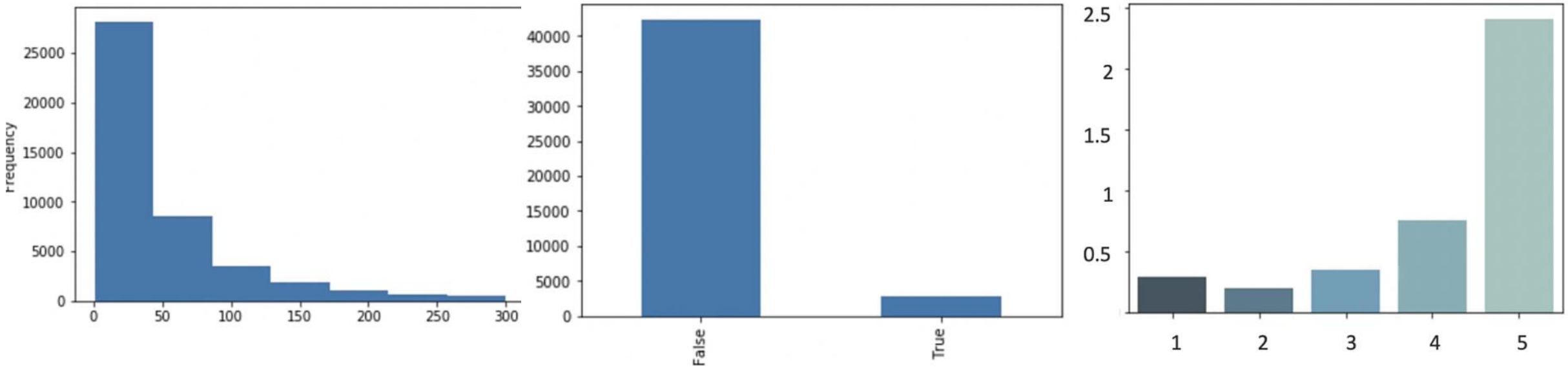
- 모델링 이전에 Data 특성 파악 목적
- 데이터의 분포나 변수간 관계를 파악하기 위해 히스토그램, 산점도, 상관관계표 등 다양한 시각화 방법 동원
- 지루하고 따분한 과정 - 각각의 변수가 어떤 의미인지 one-by-one 파악
- 각 변수가 명목형/수치형 인지 이해
- 데이터로부터 좋은 insight 를 얻는 출발점

Univariate Statistics : Histogram

연속된 숫자

Binary Class

Multiple Class

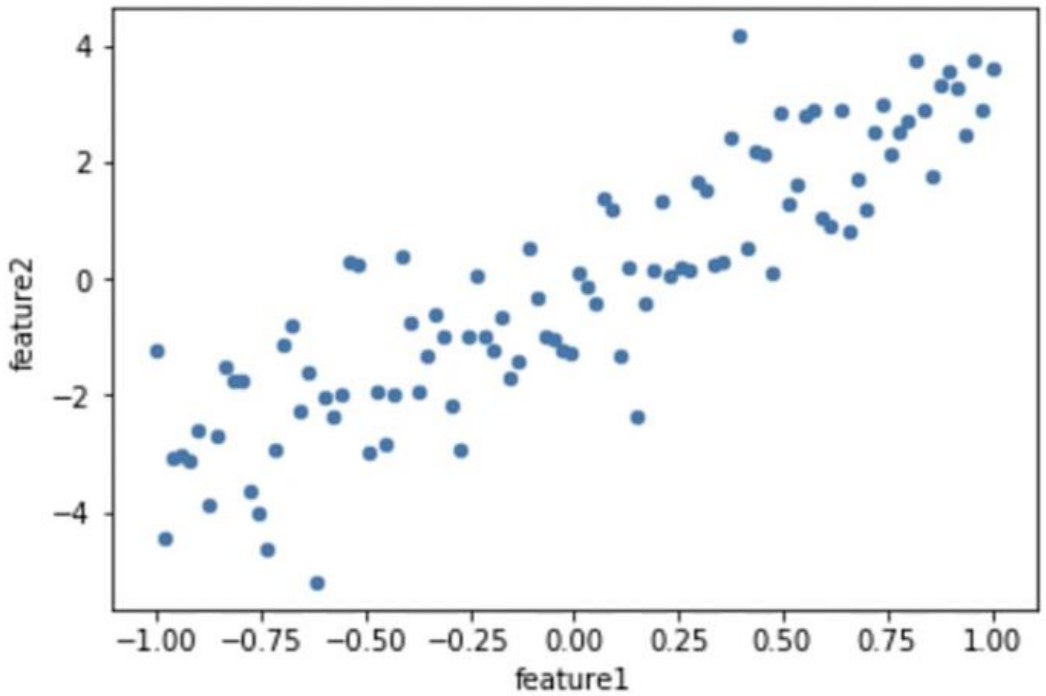


skewed data

unbalanced data

Correlation – 두 feature 간의 상관 관계

Scatterplot



correlation matrix

	feature1	feature2
feature1	1	0.882106
feature2	0.882106	1

1 : perfect positive correlation
-1 : perfect negative correlation
0 : no relationship

실습 – 070. 자동차 가격 분석 project

- 자동차 가격에 영향을 미치는 주요 특징 분석
- 시각화 도구를 사용하여 개별 feature의 패턴 분석
- 연속 수치 변수 (Continuous numerical variables) 분석
- 범주형 변수 분석
- 기술적 통계 분석(Descriptive Statistical Analysis)
- Grouping
- Pearson Correlation

실습 – 071. 맥도날드 메뉴 영양 분석 EDA

- 어떤 food item 이 가장 sodium 함량이 높은지 분석
- 단백질과 지방과의 상관 관계 탐색

Feature Scaling

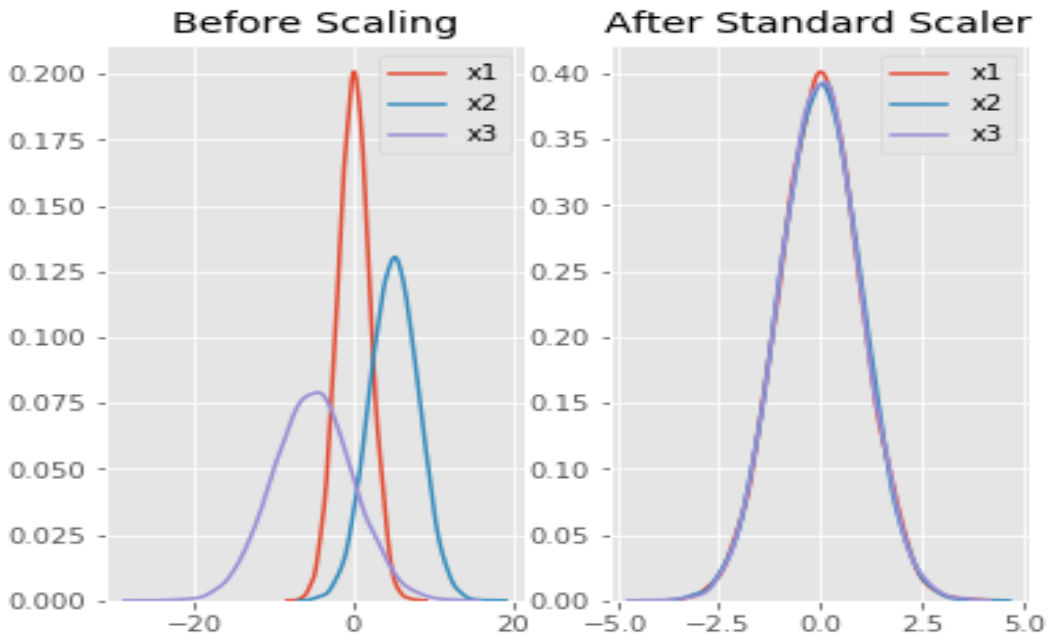
- Raw data 를 전 처리하여 input data 의 구간을 표준화

- Standard Scaling :

  $z = (x - \mu) / s$ (μ : 평균, s : 표준편차)

- Minmax Scaling :

$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$



실습: 072. feature scaling (normalization)

- Simple Feature Scaling
- Standard Scaling
- MinMax Scaling

Feature Engineering

Good Feature 의 조건

- Target 과의 높은 관련성
- prediction 시점에 알 수 있음 (ex. sales data 는 익월에 집계)
- numeric
- 충분한 example 수
- 인간 전문가의 domain 지식 활용 가능

머신러닝을 위한 Feature Engineering

- ✓ Missing Values 처리
- ✓ Data Formatting
- ✓ 편향(skewed data) 처리
- ✓ Data Normalization
- ✓ Binning
- ✓ categorical 변수의 수치화

Data Formatting

City
NY
New York
N.Y
N.Y

City
New York
New York
New York
New York

Data Normalization

age	income
20	100000
30	20000
40	500000

age	income
0.2	0.2
0.3	0.04
0.4	1

category 변수의 수치화

- categorical feature : not numerical → 숫자로 변환 (ordinal, nominal)

- ordinal category (순서/크기가 있는 feature)

  → 숫자로 크기(순서) 표시
  ex) L > M > S → 3, 2, 1

- nominal category (순서/크기가 없는 feature)

  → one-hot encoding
  ex) color 의 숫자 표시

Binning

price
13495
16500
18920
41315
5151
6295
...

price	price-binned
13495	Low
16500	Low
18920	Medium
41315	High
5151	Low
6295	Low
...	...

Categorical 변수의 수치화

fuel
gas
diesel
gas
gas

gas	diesel
1	0
0	1
1	0
1	0

실습: 075. Titanic 호 data 를 이용한 Feature Engineering 과 Modeling

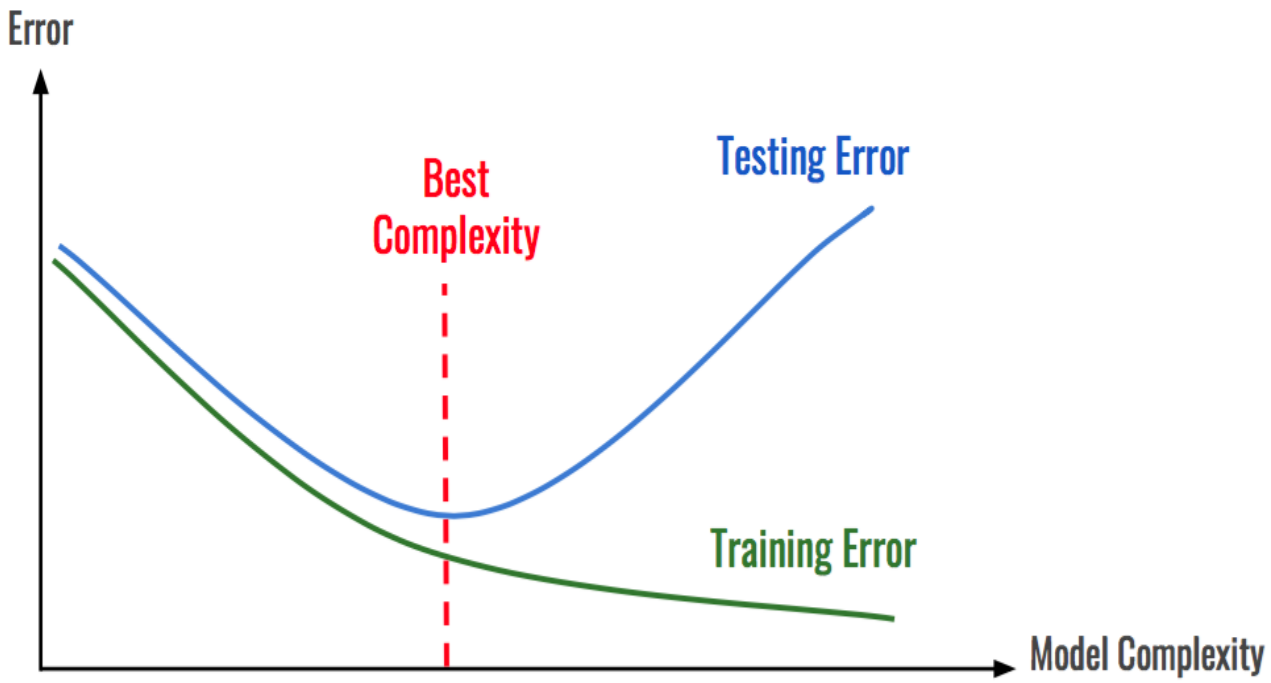
- Survival 예측 model 작성
- Feature Engineering
- Deep Neural Network – binary classification

Training/Test/Evaluation

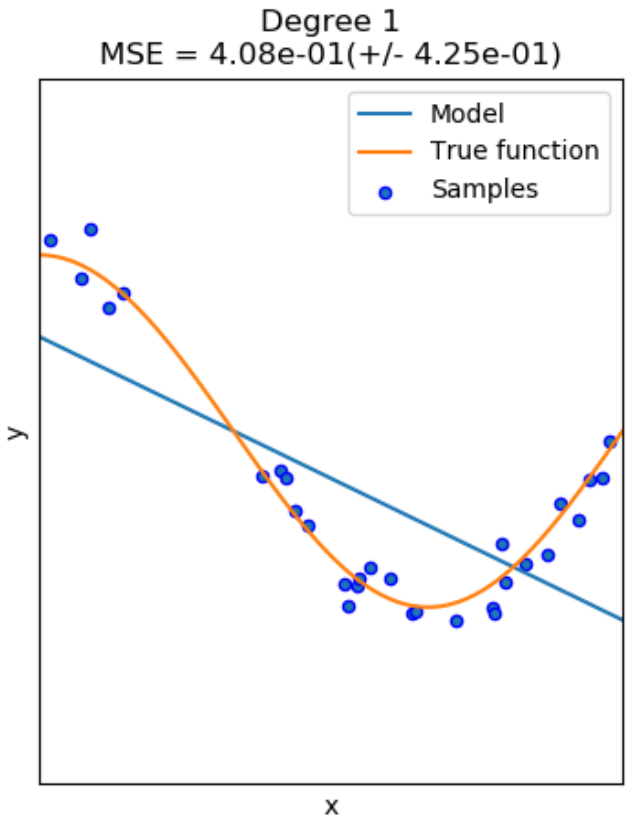
Overfitting(과적합) 과 Underfitting(과소적합)

- Training Data 에 비해 Test Data 의 ERROR 율이 높게 나타나는 경우 이를 과적합 (Overfitting) 이라고 한다.

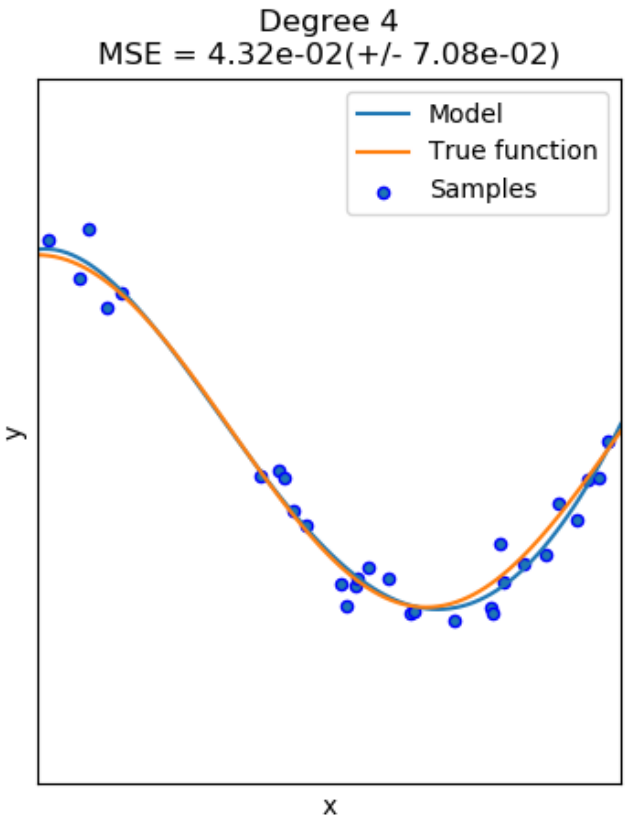
- 반대로 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못하는 경우 과소적합 (Underfitting) 이라고 한다.



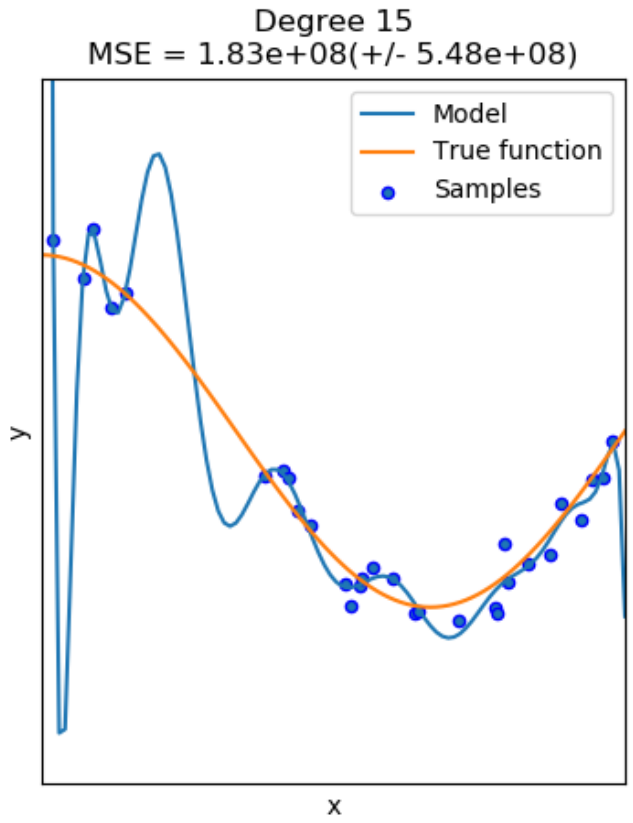
Underfitting
(과소적합)



모델이 너무 단순
(data 의 중요 부분을 놓침)
High Bias Model



Overfitting
(과대적합)



모델이 (데이터에 비해)너무 복잡
(실제와 무관한 noise 까지 학습)
High Variance Model

Bias-Variance Trade-off

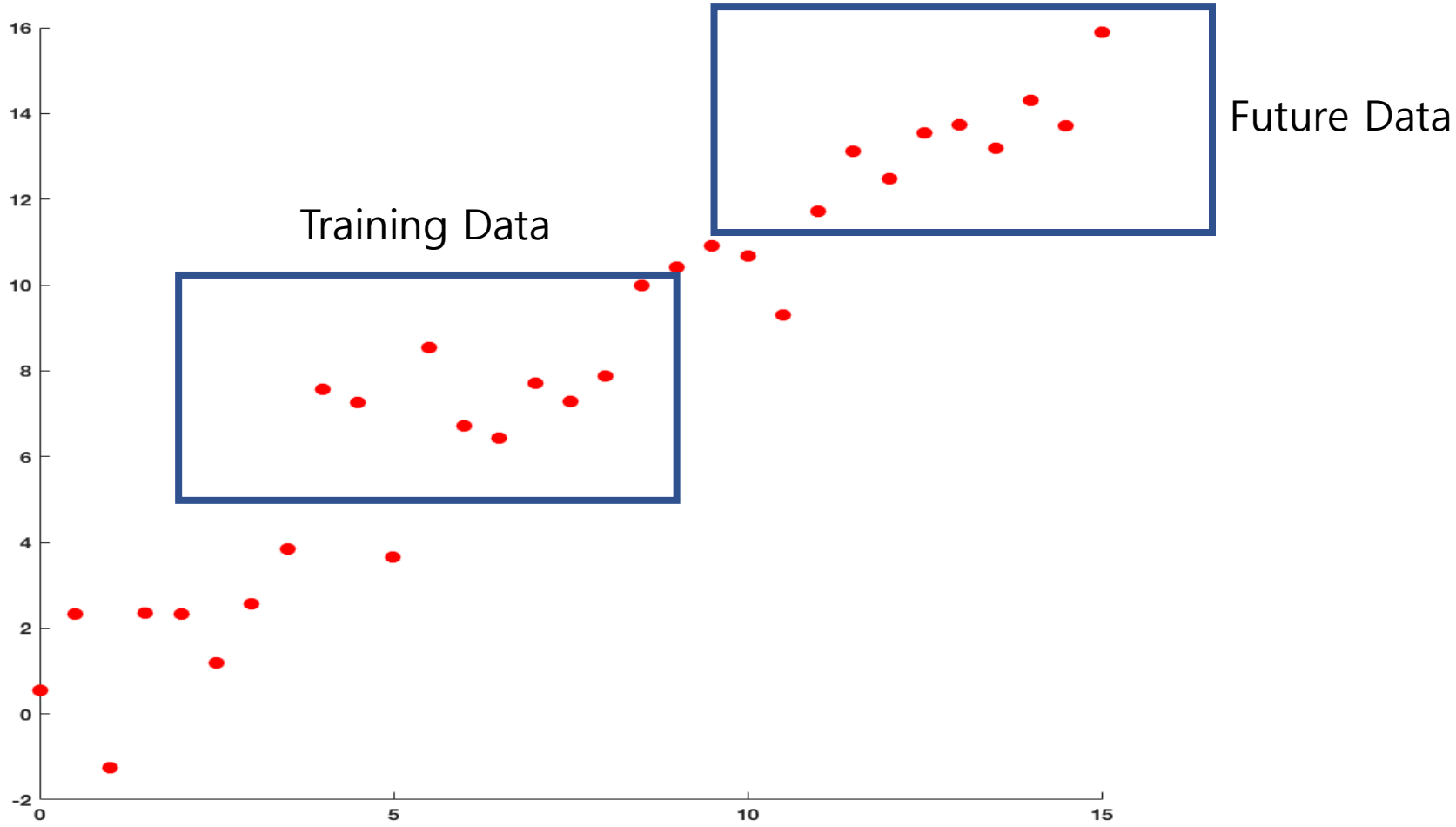
- ML의 세가지 Error Source

  1. 학습 Data 와 실제 data 분포의 차이에 의한 error → Variance
  2. Approximation Model 과 True Function 의 차이에 의한 error → Bias
  3. Noise 에 의한 error → 제거할 수 없음

- Variance 를 줄이려면 Dataset 의 크기를 늘이고,
  Bias 를 줄이려면 모델의 Complexity 를 올린다.

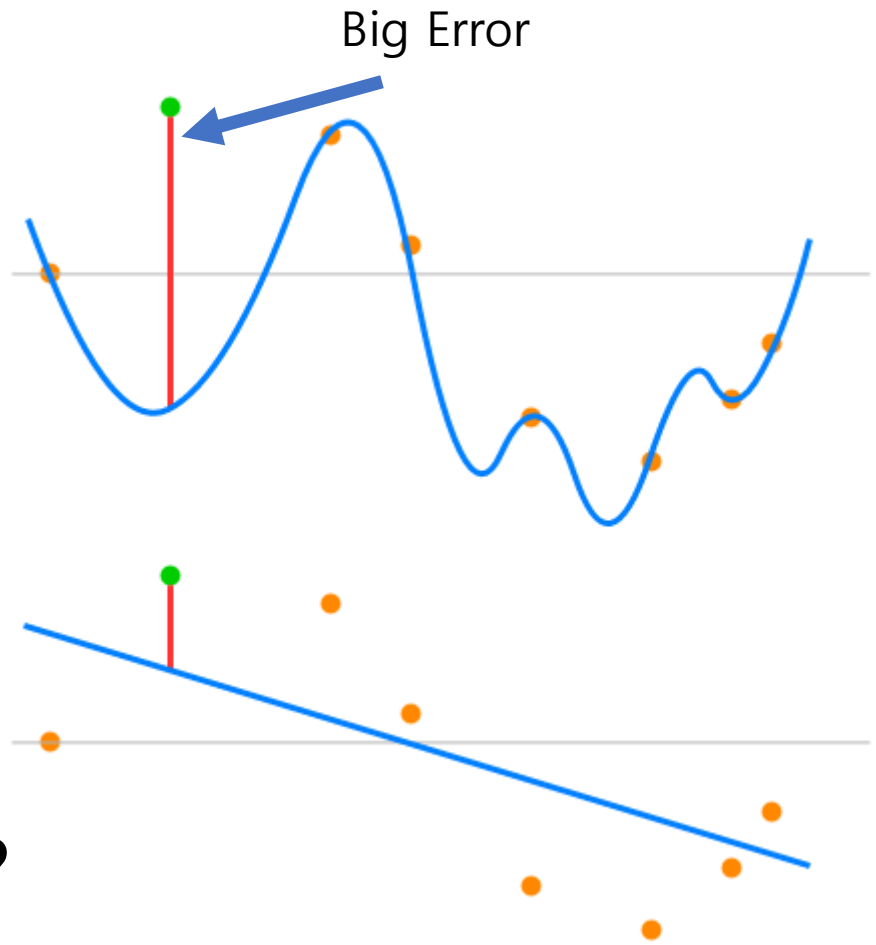
  → Bias-Variance Dilemma (Bias-Variance Trade-off)

Variance – Training Data vs. Future Data



Bias – True & Inference Function 차이

- True function 은 sine 함수
- Model 1 – polynomial regression
- Model 2 – Linear regression



Which one is better model ?

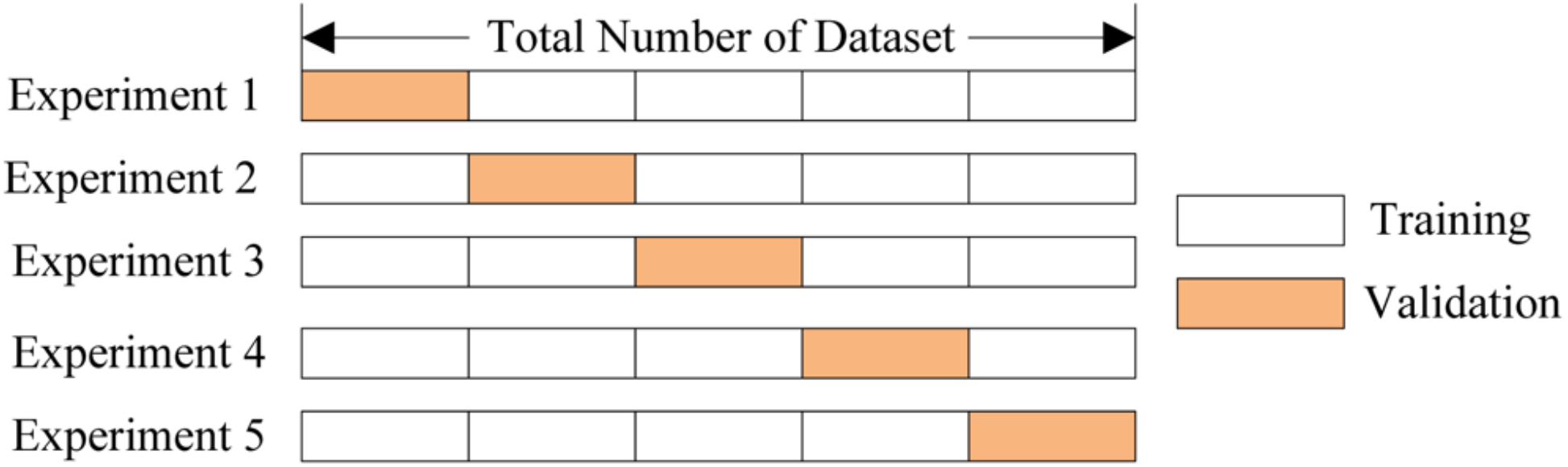
- Variance – infinite data sampling 으로 해결 가능
- Bias – true function 을 알면 해결 가능
- **BUT,** 현실에서는 infinite data sampling 도 할 수 없고 true function 도 알 수 없으므로 간접적 방법을 사용
  1. Cross Validation
  2. Precision / Recall / F1-Score

Training & Testing & Cross-Validation Set

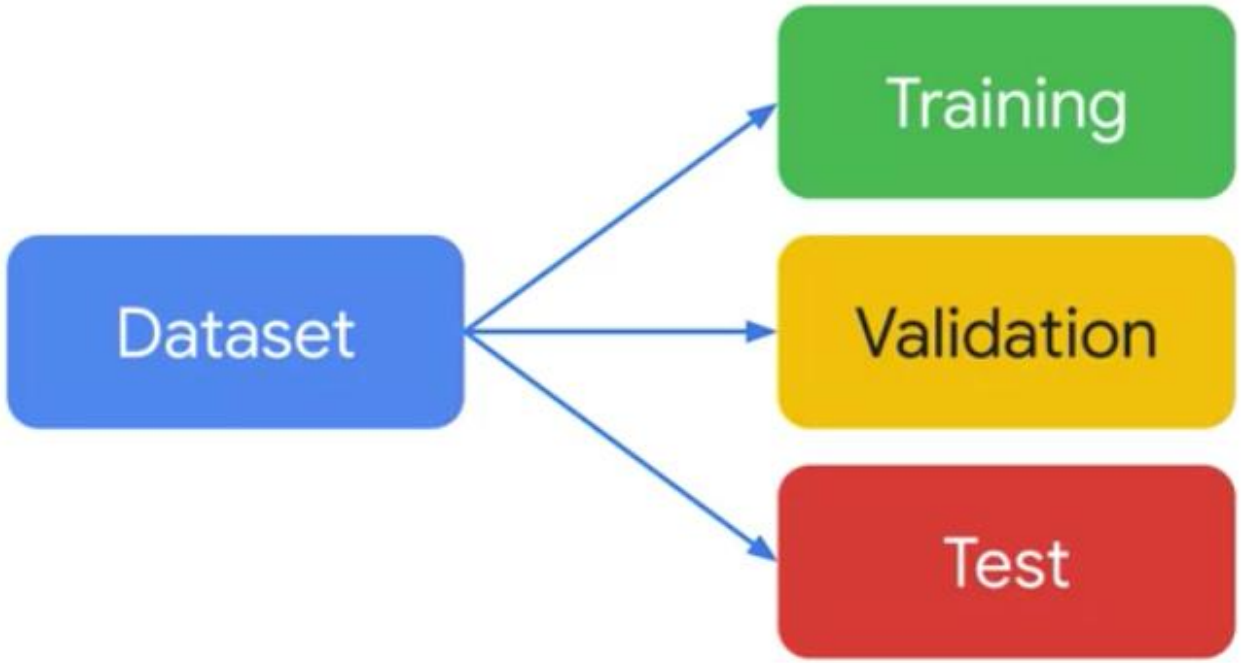
- Training Set
  - parameter 를 inference 하는 procedure 에 사용하는 data
  - 보지 못한 Data 의 분포가 Training set 과 상이할 경우 문제
  - Training set 내에서 cross-validation set 을 구성 (Data 가 불충분한 경우)
- Testing Set
  - 학습한 Machine Learning Model 을 Test 하기 위해 사용하는 data
  - 미래의 instance 인 것처럼 간주
- Training set 과 Testing set 은 섞이면 안되고 동일한 분포를 유지해야 함.

Cross Validation (교차검증)

- 훈련세트를 여러 개의 sub-set 으로 나누고 각 모델을 이 sub-set 의 조합으로 훈련시키고 나머지 부분으로 검증
- Data 의 수가 적은 경우 사용



Dataset Split – 3 splits



Parameter 학습용

Hyper-parameter Tuning 용

Performance Metrics 용

Generalization

- No Free Lunch Theorem
  - 모든 문제를 하나의 model 로 해결할 수 없다. 즉, 하나의 문제에 잘 맞는 모델이 다른 문제에도 잘 맞는 것은 아님. 따라서, 다양한 model 를 try 하여 가장 잘 맞는 모델을 선정.

- Speed, accuracy, complexity 의 trade-off 로 model 과 알고리즘 선택
  - 모든 data 를 구할 수 없고, True Function 을 아는 것은 불가능하므로 machine learning 은 단지 확률과 통계에 기반하여 approximate 하는 것

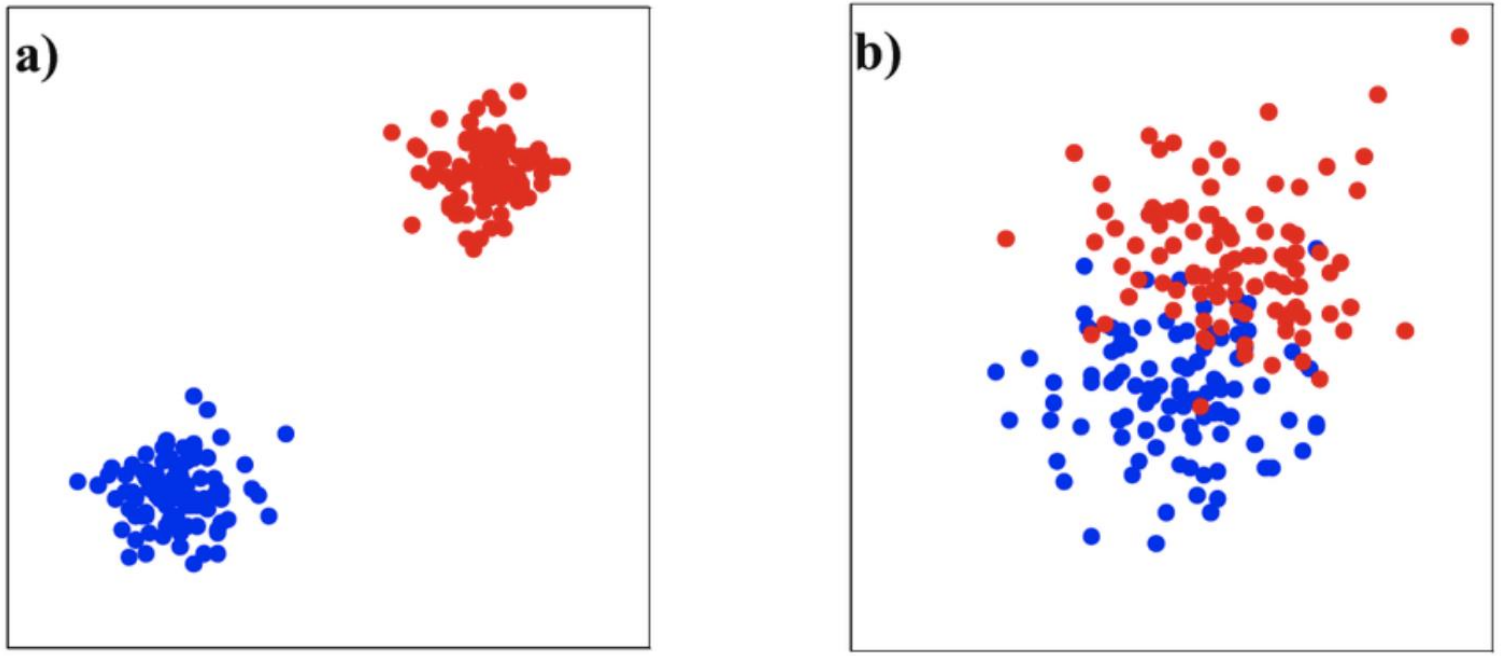
- 간단한 model ➔ 복잡한 model 순으로 Try
  - Bias-variance Trade-off 원칙을 잊지 말 것

Intra-/Inter-class variability(variance)

- Deep Learning은 feature를 스스로 찾지만 많은 데이터 필요. 하지만 데이터의 양만 많다고 좋은 data는 아니다.

- Intra-/Inter-class variability 는 training data 의 quality 의미
  - Intra-class variability – class 내부의 분산
  - Inter-class variability – class 간의 분산

- class 간의 분산은 크고, class 내부의 분산은 작은 data 가 좋은 data 이다.



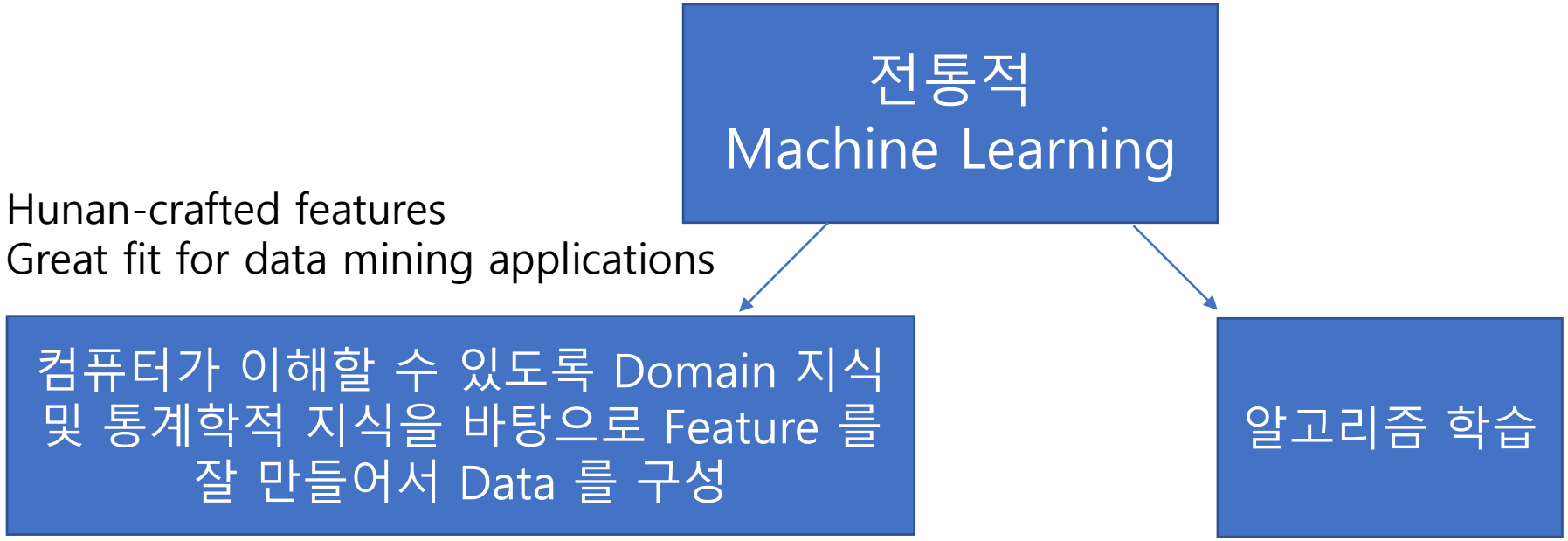
a) class 내 분산은 작고, class 간 분산은 크다 → good data

b) class 내 분산은 크고,, class 간 분산은 작다 → 결정 경계 불분명

Neural Network
and
Deep Learning

전통적 Machine Learning 의 학습

-. Hunan-crafted features
-. Great fit for data mining applications



80~90 % 의 비중

Domain 전문지식을 가진 석,박사급 인재 필요

10~20% 의 비중

각 feature 의 weight 를 optimize

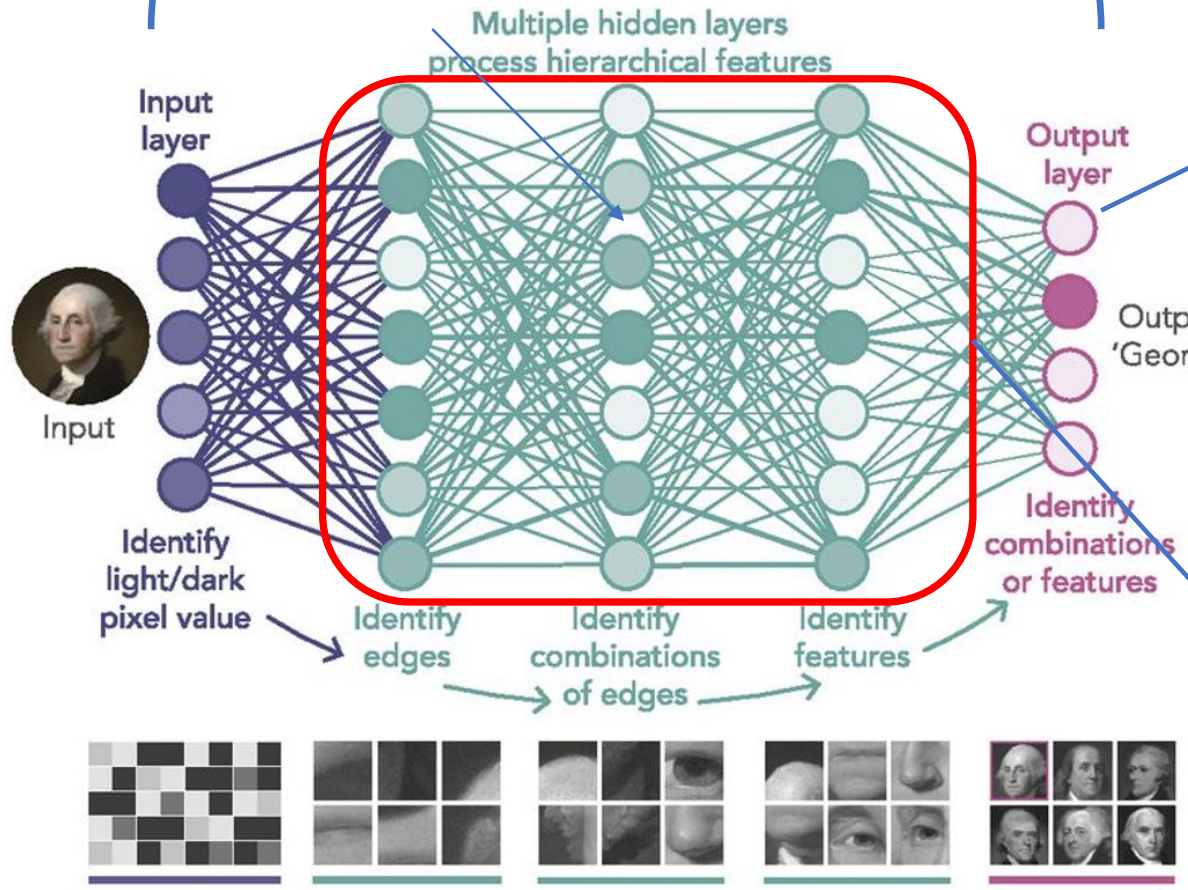
Deep Learning 의 학습

- 중요한 Feature 를 스스로 구분하여weight 를 부여
  - 사람이 manually 정해준 feature 는 over-specified, incomplete 위험성 있고 작성에 많은 시간 소요
- 여러 층에 걸친 내부 parameter 를 스스로 학습
  - 적용하기 쉽고 빠르다.
- Raw data 를 거의 그대로 사용 – computer vision, 언어처리 등
  (ex, image, sound, characters, words)
- Unsupervised, supervised learning 모두 가능
- 이미지 인식, 대화/언어 문제에 탁월한 성능

Deep Learning

- 인간의 두뇌의 신경망을 모방
- 실제 작동 방식은 인간 두뇌와 전혀 다름

Artificial Neural Network



neuron

예측 결과

Big Mathematical Equation

Artificial Neuron (Perceptron)

구성요소:

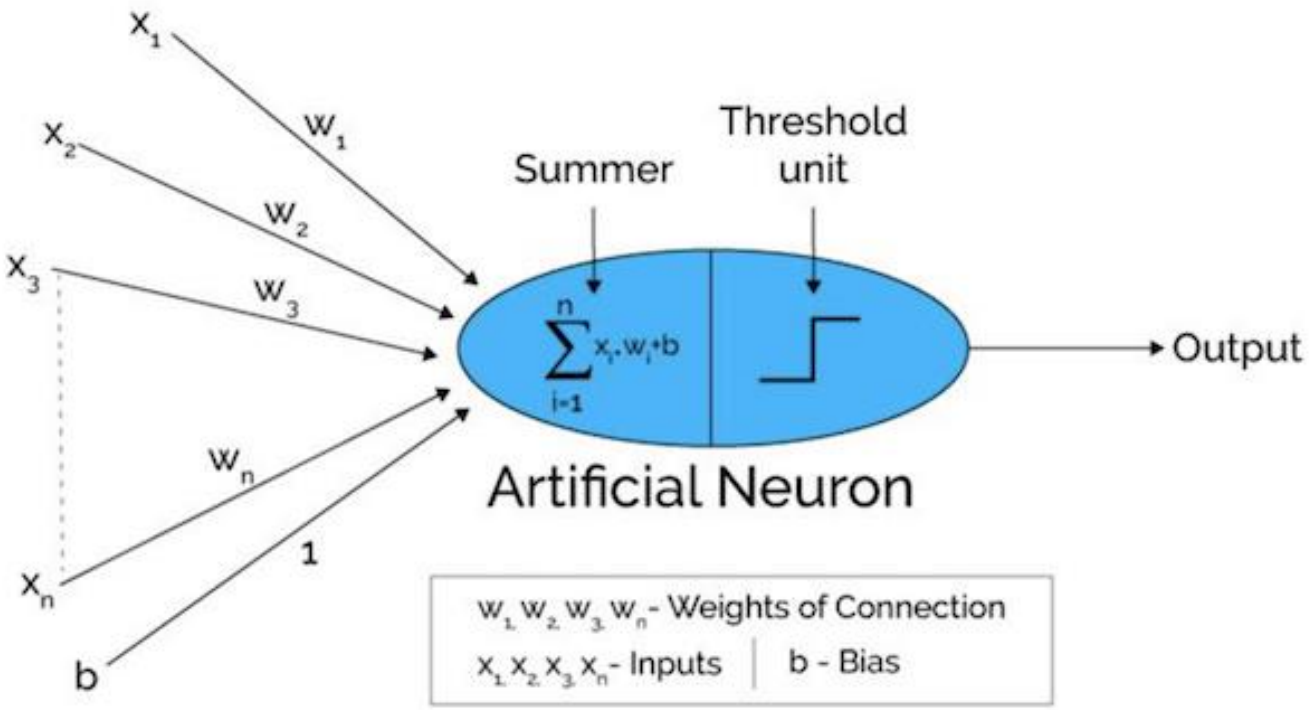
Pre-Activation :

$a(x) = b + \sum_i w_i x_i = b + w^T X$

Activation :

$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$

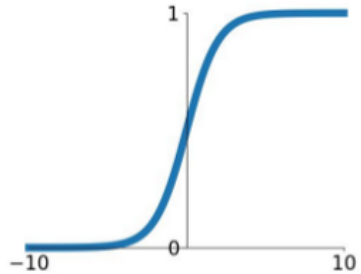
w : connection weights
b : bias
g : activation function



Activation Functions

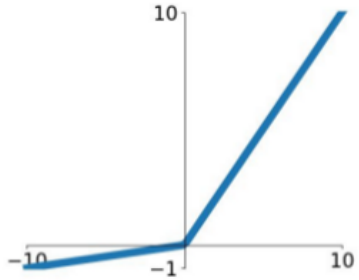
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



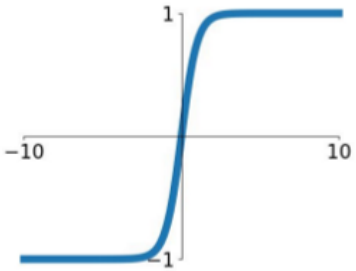
Leaky ReLU

$$\max(0.1x, x)$$



tanh

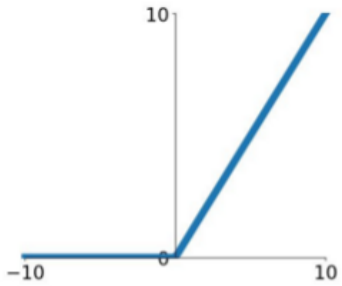
$$\tanh(x)$$



$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

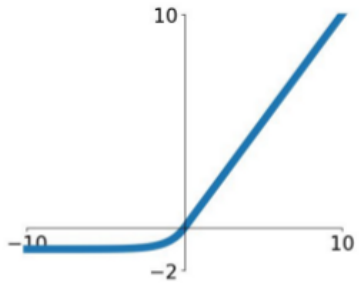
ReLU

$$\max(0, x)$$



ELU

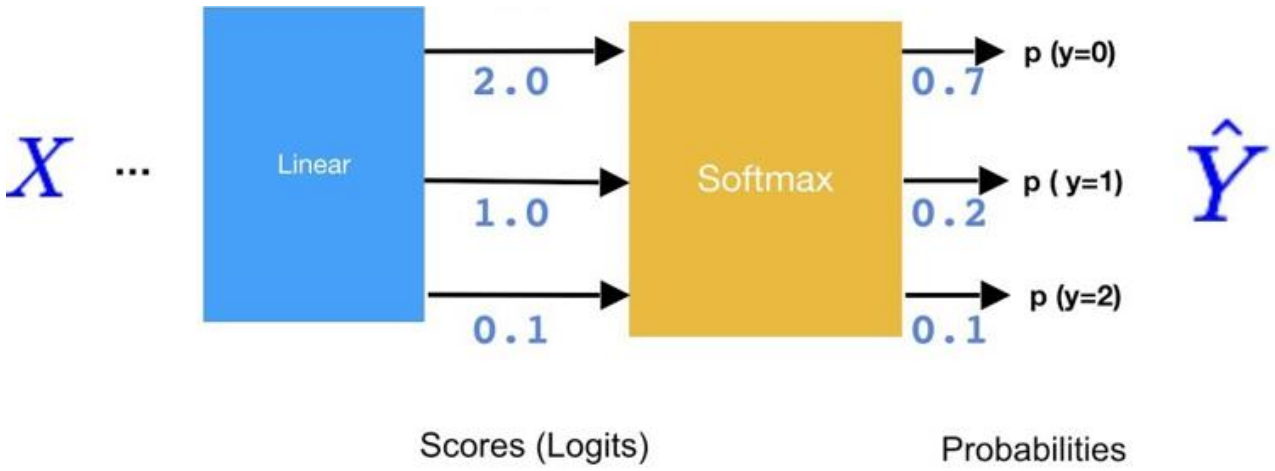
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



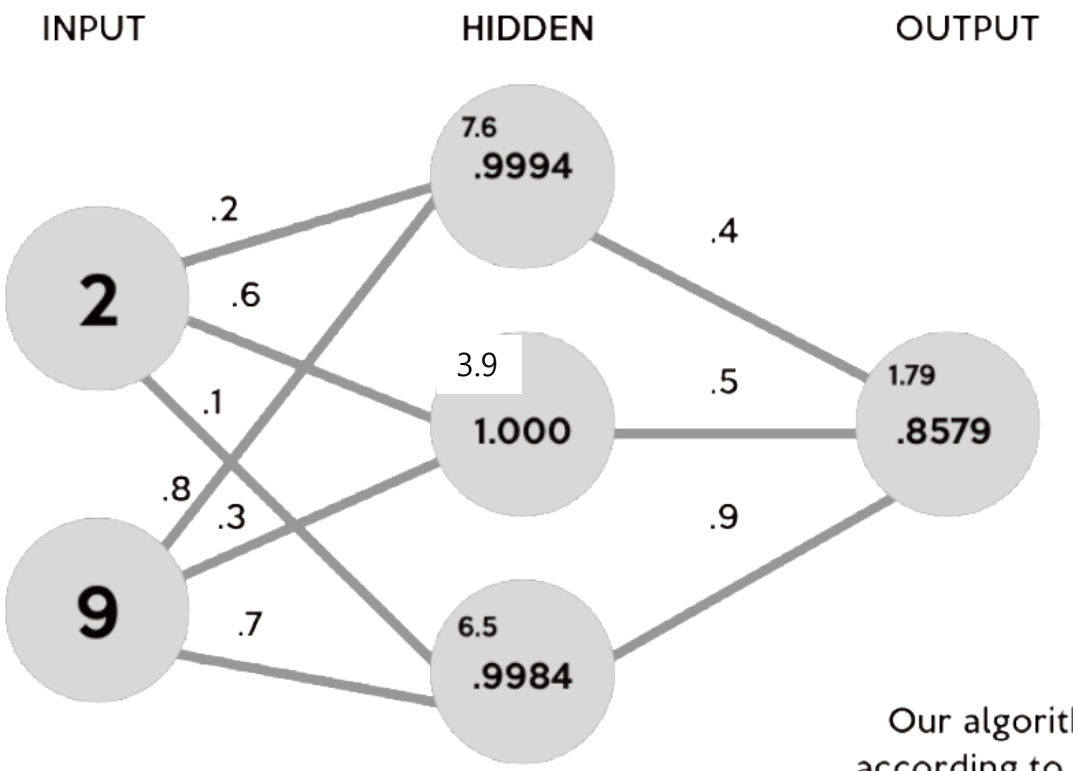
Softmax

- 출력값의 class 분류를 위하여 출력값에 대해 **정규화 → 확률 분포 출력**

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



Neural Network 의 작동 원리

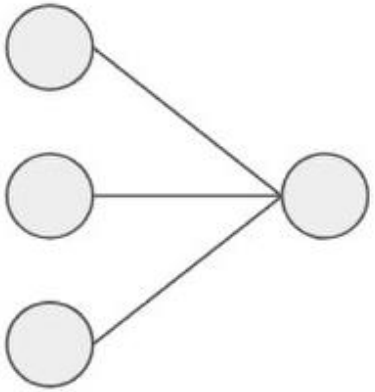


Our algorithm, according to the untrained (random) weights, produced .85 as our test score result.

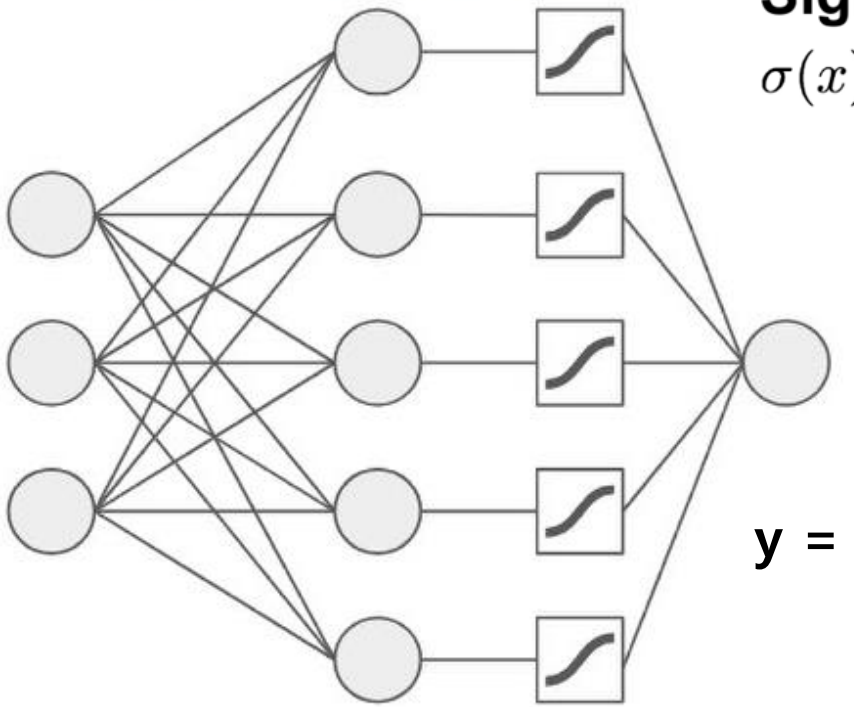
Regression (Linear / non-Linear)

A. Linear-regression model

$$y = Wx + b$$

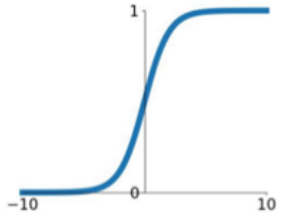


B. Two-layer neural network with nonlinear internal activation



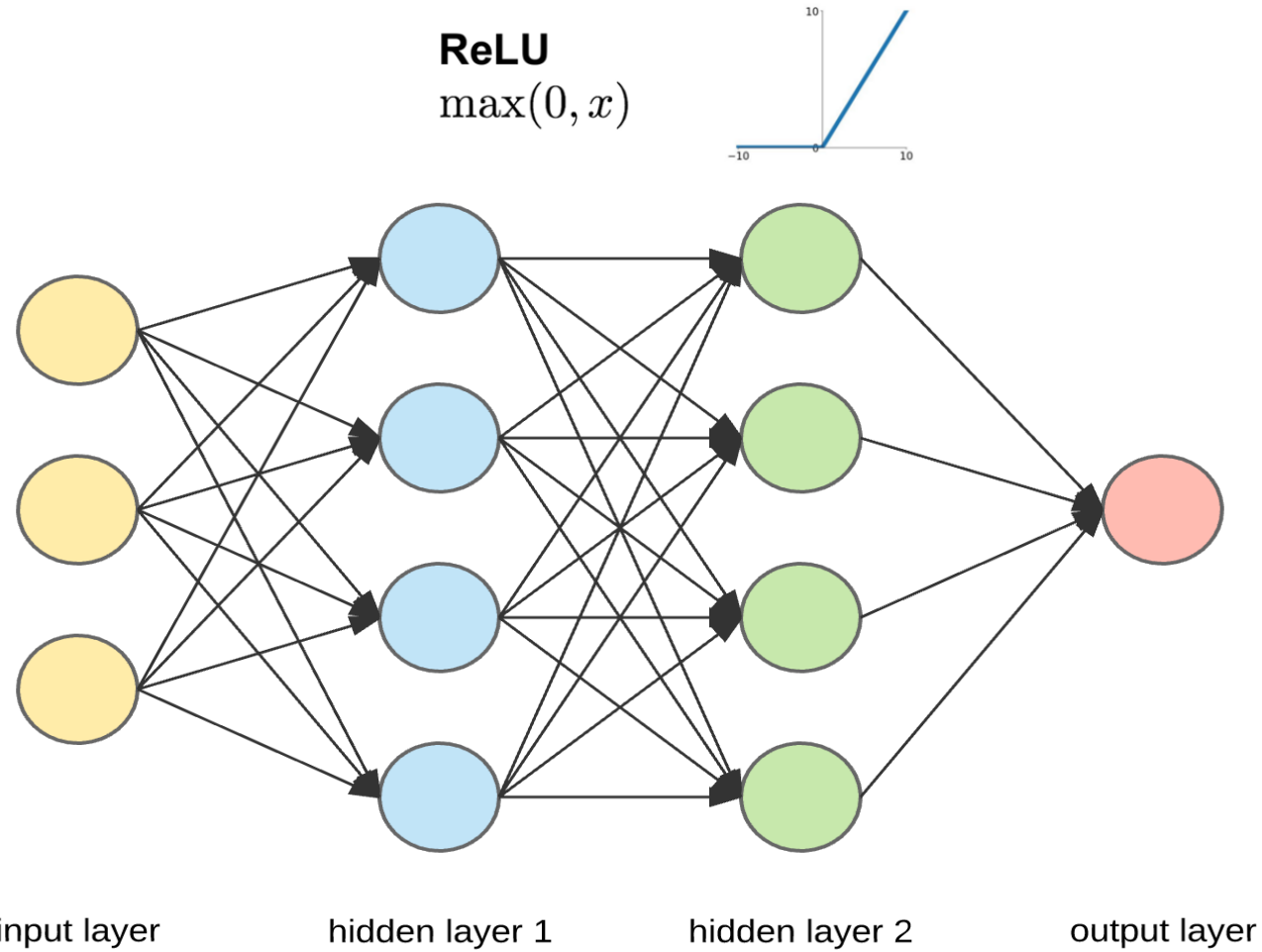
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

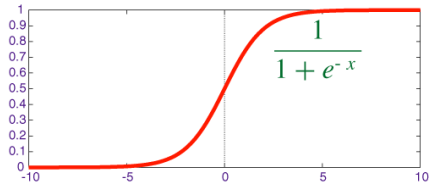


$$y = Wx + b$$

Binary Classification (Sigmoid)

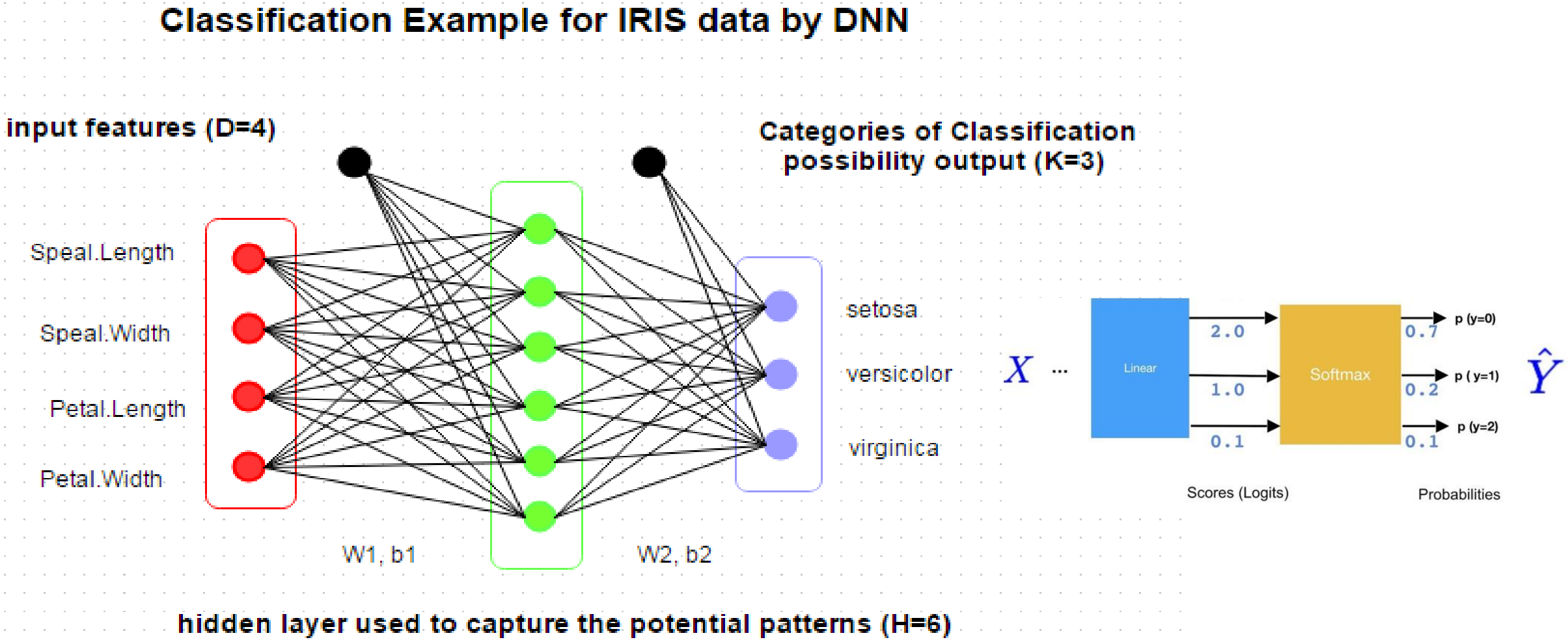


true



false

Multi-Class Classification (Softmax)

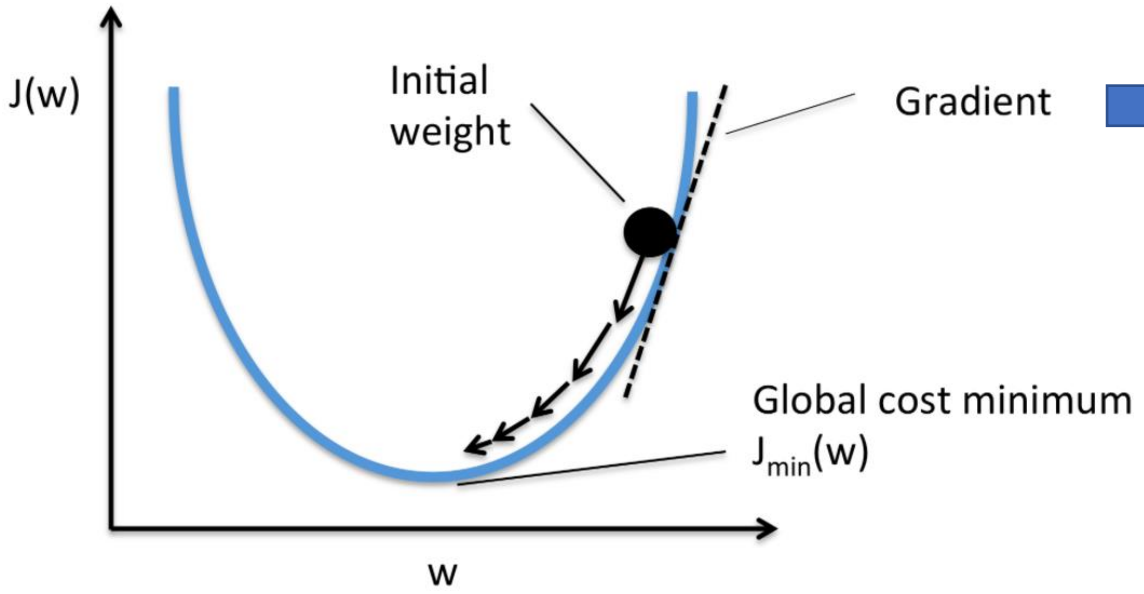


Neural Network
훈련 원리

Deep Learning 학습 방법의 핵심

- Gradient Descent (경사하강법)
  - 목적 – 실제값과 예측값의 차이를 최소화 하는 parameter(θ) 발견
  - 방법 – 손실함수를 정의하여 손실함수의 값이 0 으로 수렴하도록 parameter(θ) 조절
- Backpropagation (오차역전파)
  - 손실함수를 최소화 하는 방향으로 신경망 전체의 parameter 가 update 되도록 하는 기법
- 손실함수
  - 비용함수(cost function), 목적함수(object function) 등으로도 불림
  - 경사하강법이 가능하도록 미분 가능한 함수를 정의

Gradient Descent (경사하강법) Optimization



방향 – Gradient
(derivative of Cost Function)

이동 속도 – Learning Rate

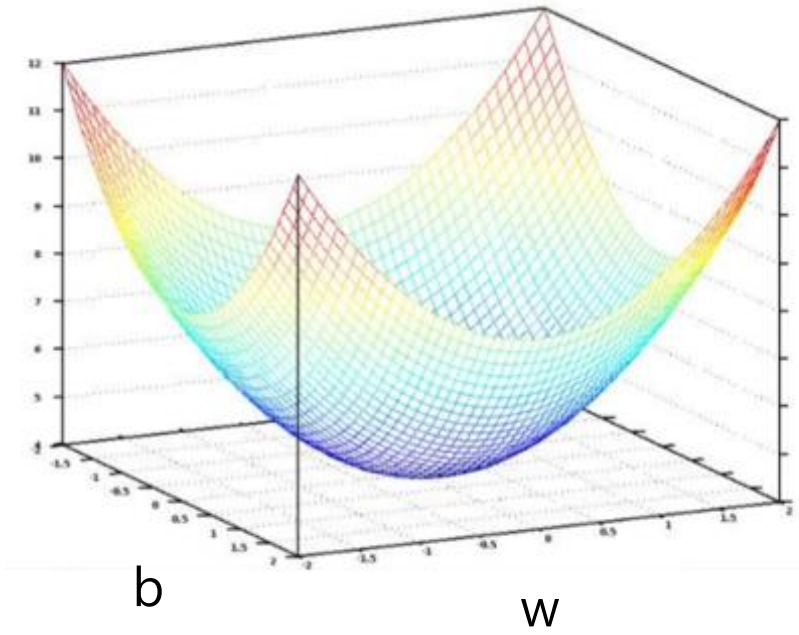
New W = old W – (Learning Rate) * (Gradient)

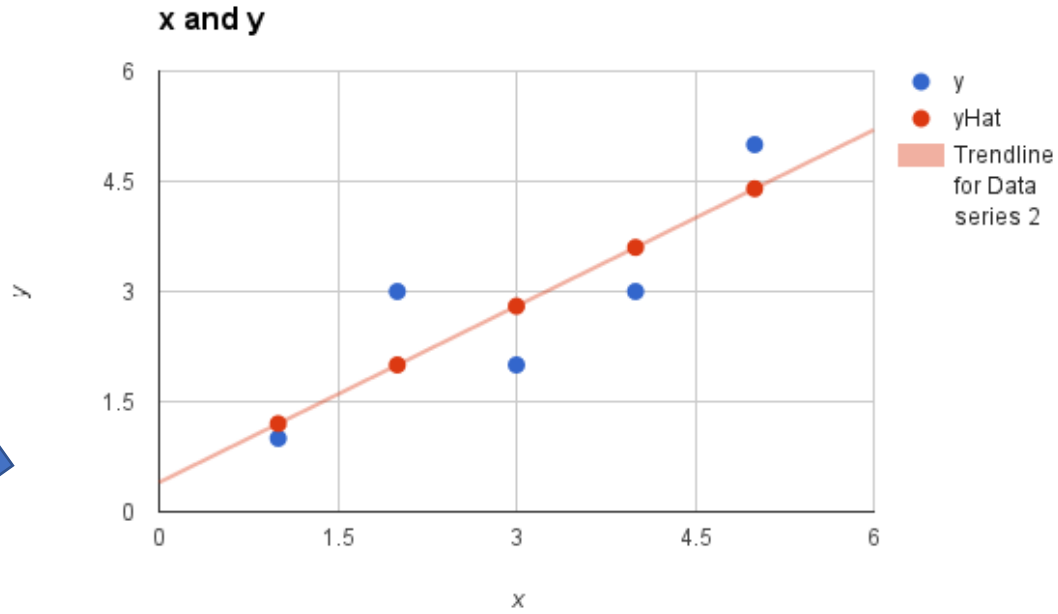
대표적 손실 함수

- Linear Regression (선형회귀) – MSE (Mean Squared Error)
- Binary Classification (이진분류) – Binary-Cross-Entropy
- Multi-Class Classification (다중분류) – Categorical-Cross-Entropy

Cost Function (손실함수) - Linear Regression

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$





y = Wx + b

MSE(Mean Squared Error) 를 최소화 하는 W 와 b 를 optimize

Gradient Descent for Logistic Regression

- Hypothesis (model) : $\sigma(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$

$$\theta^T = [\theta_0, \theta_1, \ldots., \theta_n]$$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ x_n \end{bmatrix} \qquad y = \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ y_n \end{bmatrix}$$

- Cost Function : $J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log(\hat{y}^{(i)}) + \left(1 - y^{(i)}\right) \log(1 - \hat{y}^{(i)}))$

if y = 1 : $J(\theta) = -\log(\hat{y}^{(i)})$

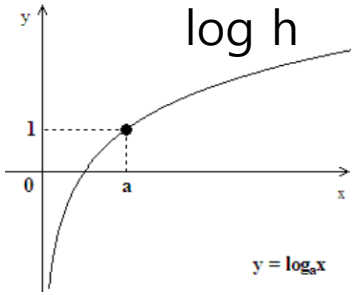
y = 0 : $J(\theta) = -\log(1 - \hat{y}^{(i)})$

Cost Function - Logistic Regression (Binary Cross-entropy)

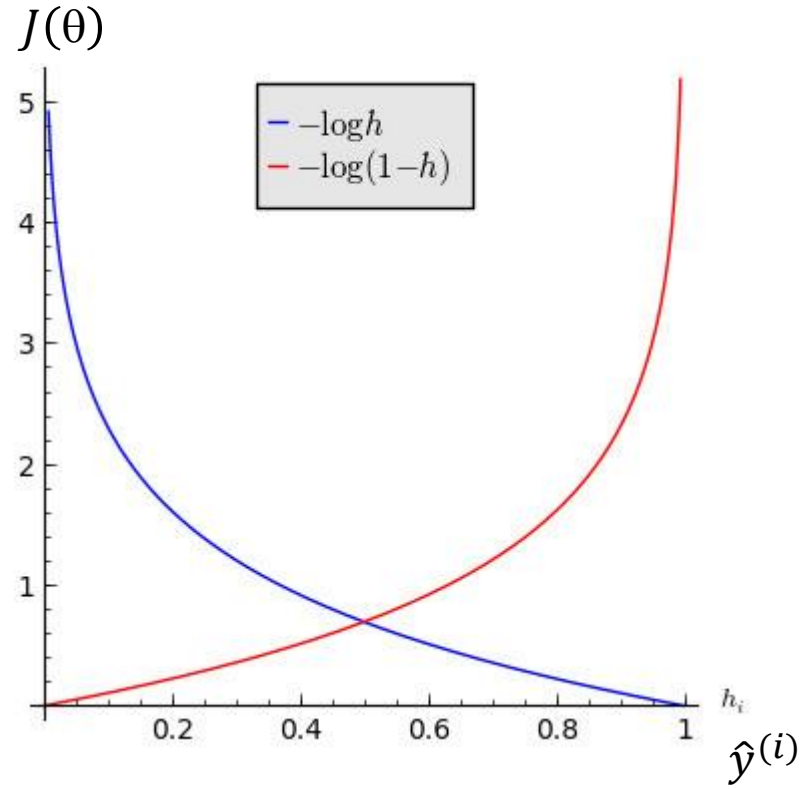
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}log(\hat{y}^{(i)}) + \left(1-y^{(i)}\right)log(1-\hat{y}^{(i)})\right)$$

If $y^{(i)} = 1 : J(\theta) = -log\,(\hat{y}^{(i)})$
where $h_\theta\,(x^{(i)})$ should be close to 1

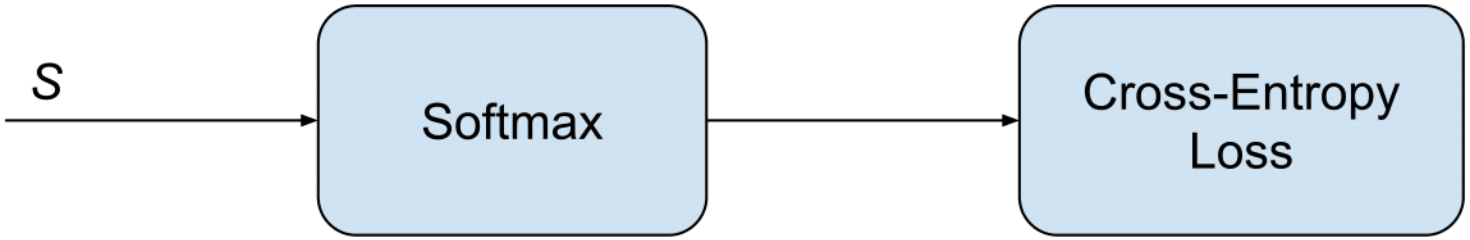
If $y^{(i)} = 0 : J(\theta) = -log(1-\hat{y}^{(i)})$
where $h_\theta\,(x^{(i)})$ should be close to 0



Log Loss



Cost Function - Categorical Crossentroy (Softmax Loss)



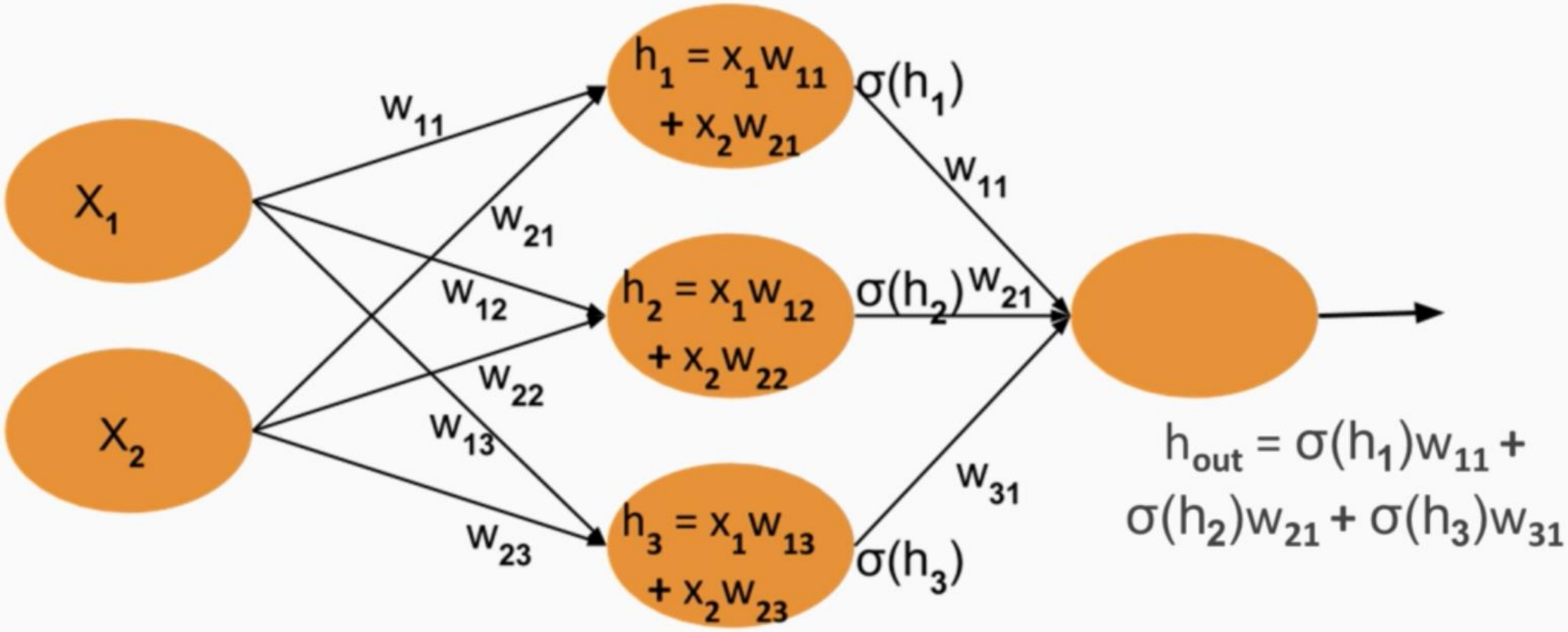
$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

t_i : 0 이 아닌 target (one-hot encoded 되어 있으므로 multi-class 중 오직 1 개만 1)

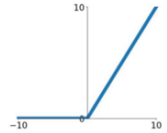
C : multi-classes

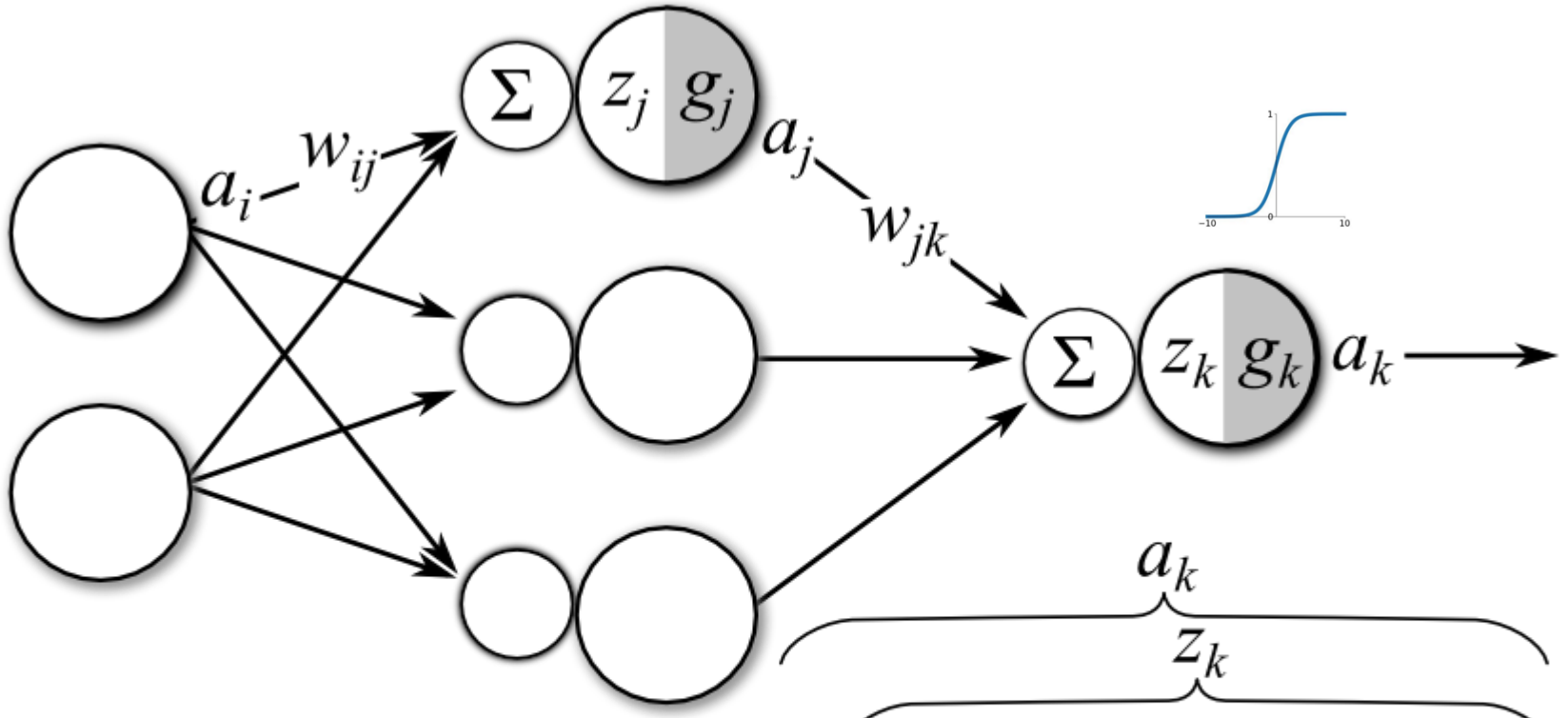
Index	0	1	2	3	4	5	6	7	8	9
True Label	0	0	0	0	0	0	0	**1**	0	0
Prediction	0.1	0.01	0.01	0.01	0.20	0.01	0.01	**0.60**	0.03	0.02

Forward Propagation (순방향전파)



Forward Propagation





$$a_k = g_k(b_k + \Sigma_j g_j(\underbrace{b_j + \Sigma_i a_i w_{ij}}_{z_j})w_{jk})$$

a_k = $g_k(\underbrace{b_k + \Sigma_j g_j(\underbrace{b_j + \Sigma_i a_i w_{ij}}_{z_j})w_{jk}}_{z_k})$

$a_k = g_k(\underbrace{b_k + \Sigma_j g_j(\underbrace{b_j + \Sigma_i a_i w_{ij}}_{z_j})w_{jk}}_{z_k})$

$a_k = g_k(b_k + \Sigma_j g_j(\underbrace{b_j + \Sigma_i a_i w_{ij}}_{z_j})w_{jk})$

$a_k = g_k(b_k + \Sigma_j g_j(b_j + \Sigma_i a_i w_{ij})w_{jk})$

Backward Propagation 기초 공식

- 기본 함수의 도함수 (derivative) : $\frac{dx^2}{dx} = 2x$, $\frac{de^x}{dx} = e^x$, $\frac{dln(x)}{dx} = \frac{1}{x}$

- Chain Rule :

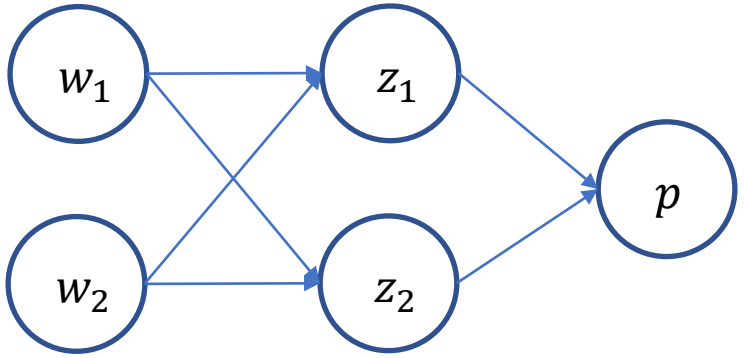
$p = f(z_1, z_2)$: 합성함수

$z_1 = f(w_1, w_2)$

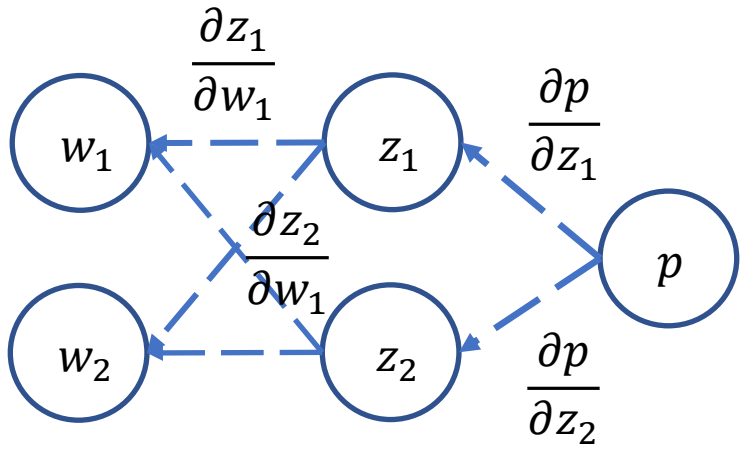
$z_2 = f(w_1, w_2)$

$$\frac{\partial p}{\partial w_1} = \frac{\partial p}{\partial z_1}\frac{\partial z_1}{\partial w_1} + \frac{\partial p}{\partial z_2}\frac{\partial z_2}{\partial w_1}$$

$$\frac{\partial p}{\partial w_2} = \frac{\partial p}{\partial z_1}\frac{\partial z_1}{\partial w_2} + \frac{\partial p}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$



forward



backward

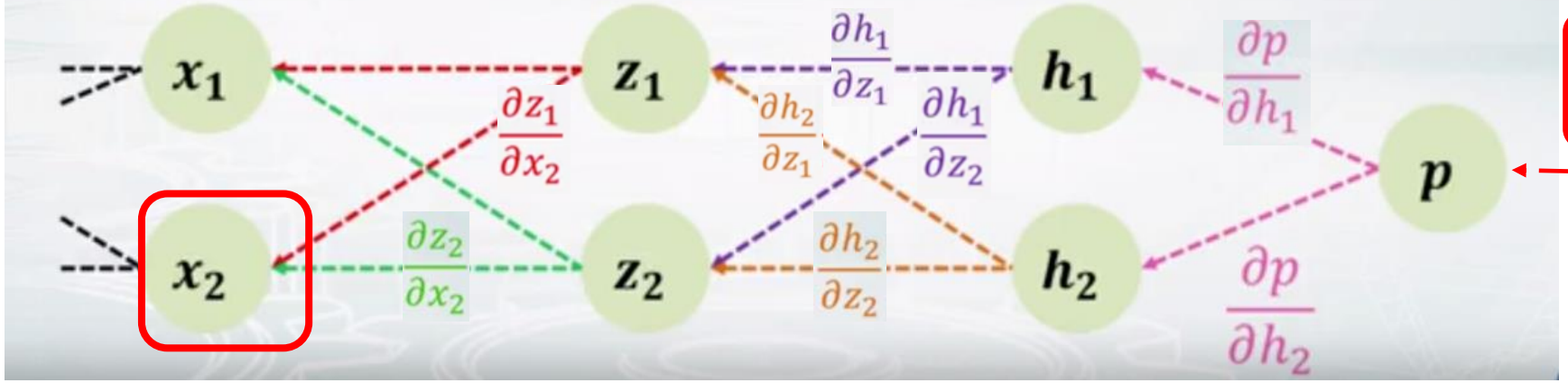
Backpropagation (Chain Rule 적용)

3: $\frac{\partial p}{\partial h_1}$ $\frac{\partial p}{\partial h_2}$ We will need these for GD

2: $\frac{\partial p}{\partial z_1} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_1} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_1}$ $\frac{\partial p}{\partial z_2} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_2} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_2}$

$\frac{\partial p}{\partial x_1} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_2}\frac{\partial z_2}{\partial x_1} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_2}\frac{\partial z_2}{\partial x_1}$

1: $\frac{\partial p}{\partial x_2} = \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial x_2} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_1}\frac{\partial z_1}{\partial x_2} + \frac{\partial p}{\partial h_1}\frac{\partial h_1}{\partial z_2}\frac{\partial z_2}{\partial x_2} + \frac{\partial p}{\partial h_2}\frac{\partial h_2}{\partial z_2}\frac{\partial z_2}{\partial x_2}$



$$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial h_1}$$

$$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial h_2}$$

$$\frac{\partial L}{\partial p} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial h_1} + \frac{\partial L}{\partial p}\frac{\partial p}{\partial h_2}$$

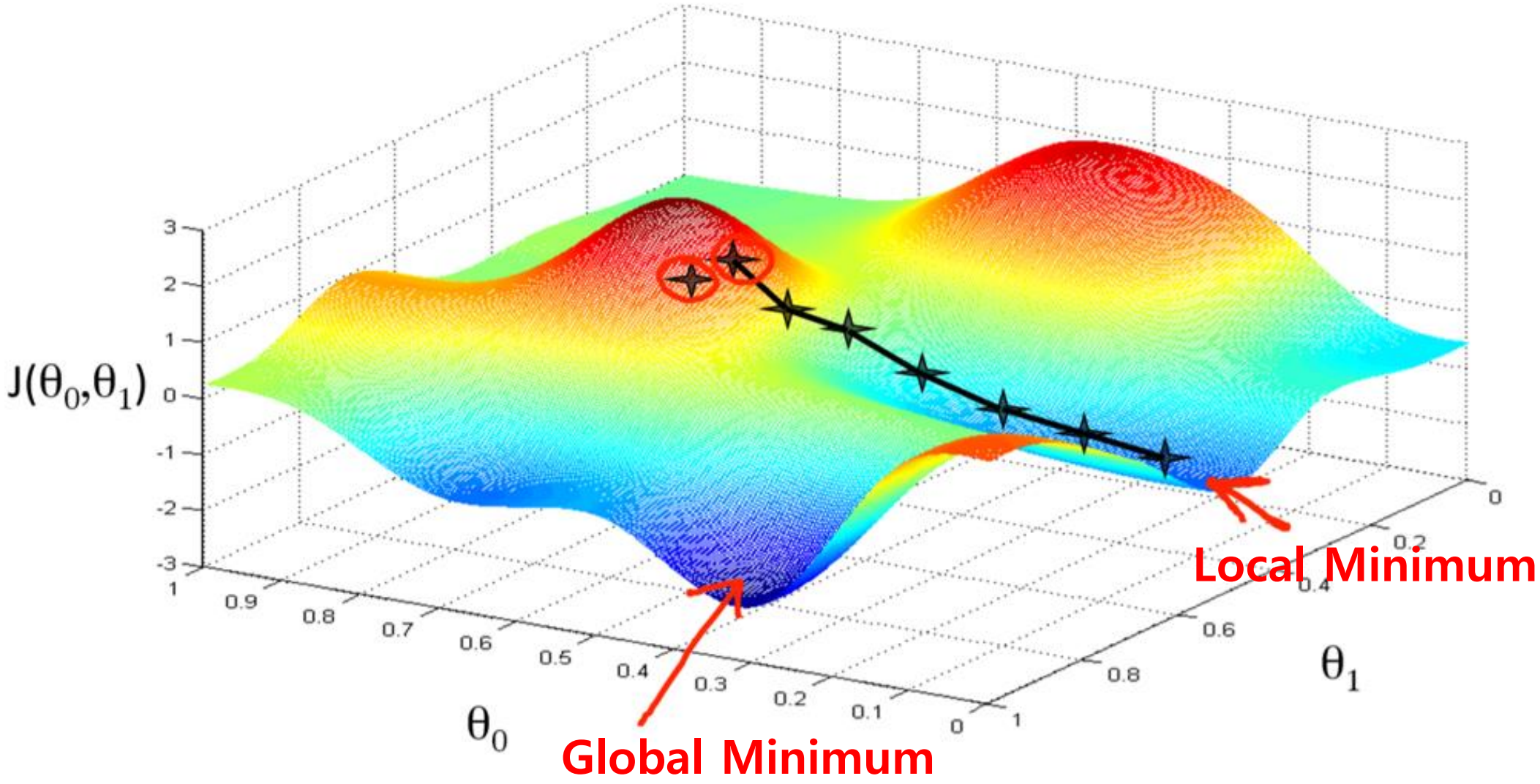
* 같은 색으로 표시된 부분은 한번 계산하면 여러 번 reuse

$$h_2 = \sigma(w_1 * x_1 + w_2 * x_2)$$

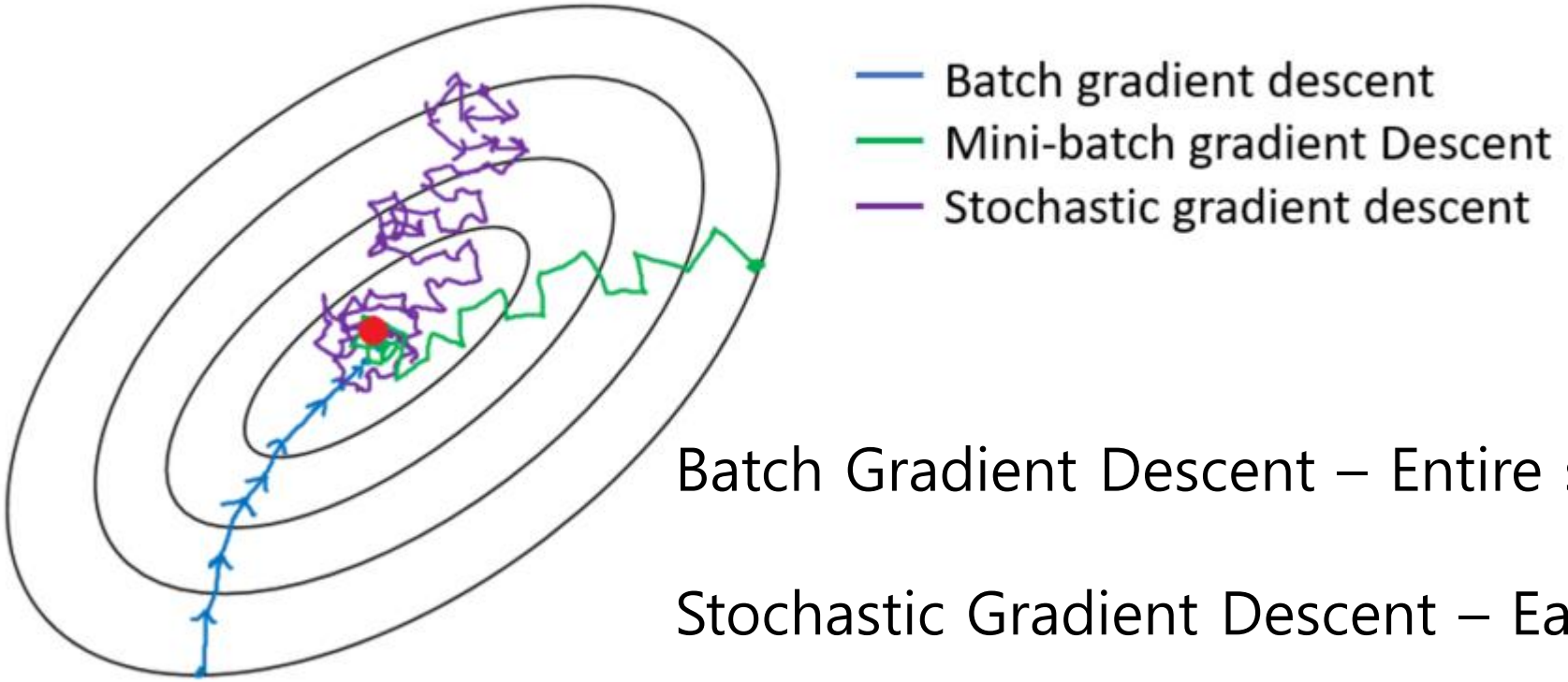
Backpropagation 요약

- 각각의 input data 에 대하여,
- 각 layer 별로 forward pass output 값을 계산
- Output layer 의 cost function 값을 계산
- Backpropagation 을 통해 cost function 의 derivative 를 전단계의 layer 로 전달
- Error term 의 값에 따라 각 layer 의 weight 를 update

Global Minima / Local Minima



Stochastic Gradient Descent (확률적 경사하강법)



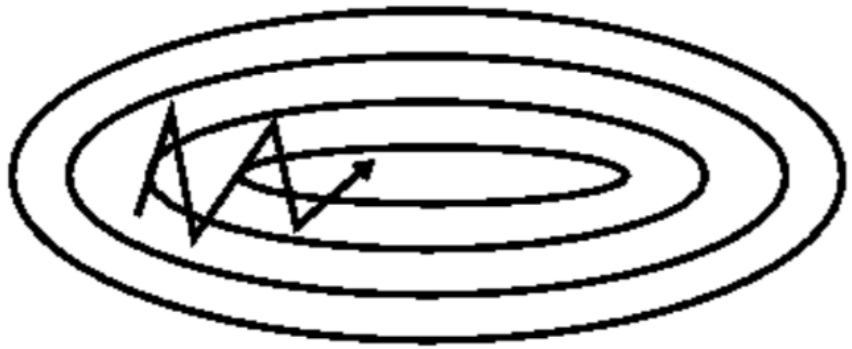
Batch Gradient Descent – Entire samples

Stochastic Gradient Descent – Each sample (size 1)

Mini-batch Gradient Descent – small size of samples

Momentum : 방향성을 유지하며 가속

(a) SGD without momentum



(b) SGD with momentum

Global minimum 에 빨리 도달하기 위해 vertically 는 변화가 적고 horizontally 는 변화가 크도록 parameter 조절

Optimizers

- Stochastic Gradient Descent Optimizer
- RMSProp Optimzer
- Adagrad Optimizer
- Adam Optimizer, etc

http://ruder.io/content/images/2016/09/contours_evaluation_optimizers.gif

http://ruder.io/content/images/2016/09/saddle_point_evaluation_optimizers.gif

epoch

- 정의 – 전체 dataset 이 neural network 을 통해 한번 처리된 것
- Epoch 은 model 의 training 시에 hyperparameter 로 횟수 지정
- 하나의 epoch 은 한번에 처리하기 어려운 size 이므로 여러 개의 batch 로 나누어 처리
- Parameter training 을 위해서는 여러 번 epoch 을 반복해야 한다.
- One epoch 내에서의 iteration 횟수는 total sample size / batch size
- Ex) 1 epoch = 4 iterations = 2000 training example / 500 batches

Hyper-parameters

- α - Learning Rate
- β - momentum term
- # of layers
- Dropout rate
- # of epochs
- Batch size

Hyper-parameter 값 결정 ?

- 정해진 RULE 이 없음
- 유사한 model 참조
- 경험에 의한 guessing
- Grid search – computationally expensive
- Hyper-parameter Tuning 도구 사용

Keras Tuner

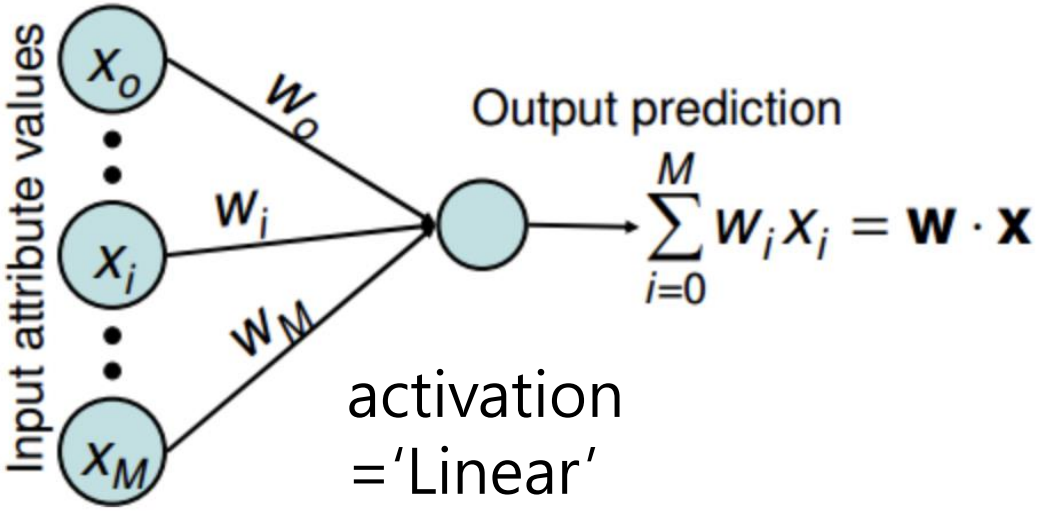
- RandomSearch, Hyperband, BayesianOptimization 및 Sklearn 의 4 가지 algorithm 제공
- RandomSearch – hyperparameter 의 조합을 random 하게 검색
- Hyperband - 작은 epoch 동안 많은 수의 모델을 훈련하고 그 중 최고 성능의 모델 절반만 다음 라운드로 전달 (토너먼트 방식)
- BayesianOptimization – 이전 결과를 기록하여 확률 모델을 작성하여 search
- Scikit-learn - Scikit-learn model 들에 대한 cross-validated hyperparameter search

Open Source Libraries for Deep Learning

- **Scikit-Learn** – 2007, Python Library based on Matplotlib, NumPy, SciPy
- Theano - 2007, Open Source Python Library
- **Tensorflow** – 2015, Google. Open Source Machine Learning Framework
- **Keras** – 2015, Open Source Python Library
  (working on top of Tensorflow, Theano, CNTK)
- Microsoft Cognitive Tool – 2016, CNTK
- Caffe – 2017, Berkeley AI Research
- **Pytorch** – 2016, Facebook
- H2O – 2011, Open Source Big Data platform on Apache Hadoop

Basic Neural Network

NN 을 이용한 Linear Regression (선형 회귀)

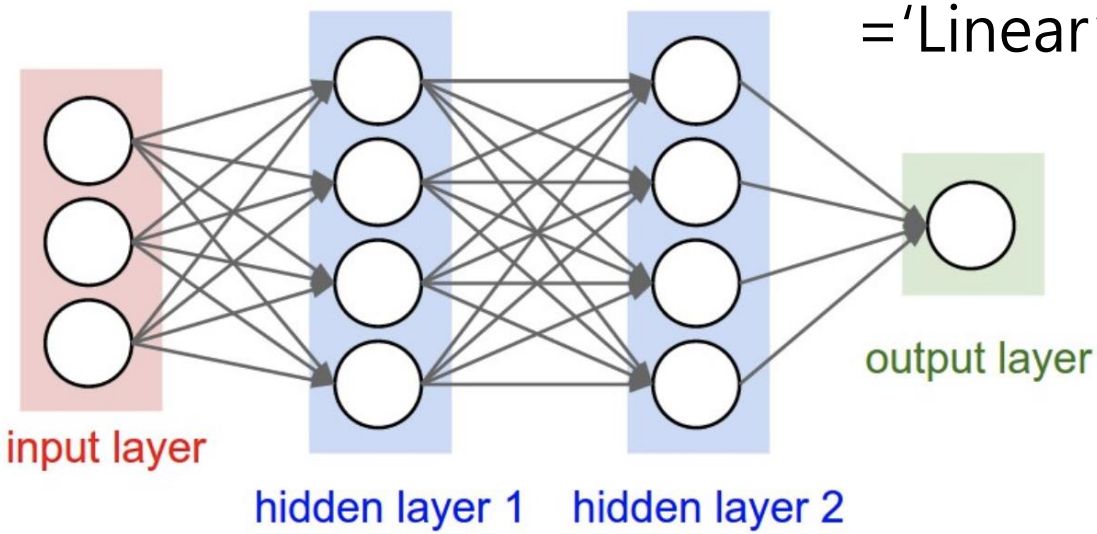


activation
='Linear'

One-Layer

Multiple-Layers

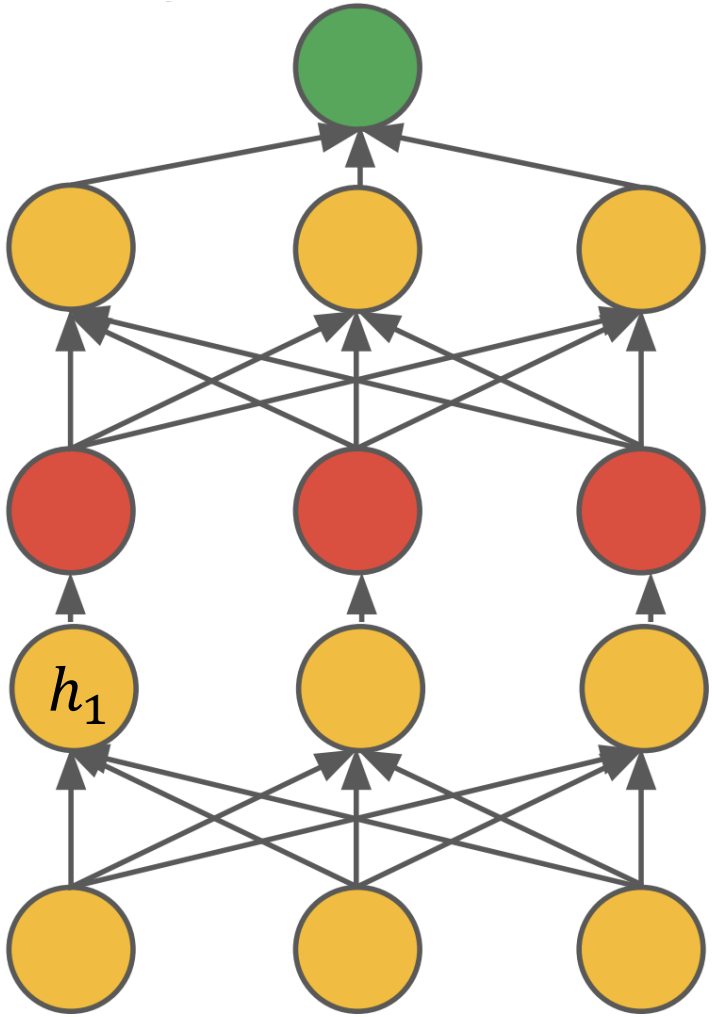
activation
='Linear'



Non-Linear Complexity 추가 방법

$\max(0, h_1)$

$\tanh(h_1)$



- Activation Function
- ReLU, tanh, Sigmoid, etc

실습 : 200. Boston 주택가격 Regression

1. Boston House Price Dataset
   - sklearn.datasets.load_boston 이용

2. 보스턴 시의 주택 가격에 대한 데이터
   - 주택의 여러가진 요건들과 주택의 가격 정보가 포함.
   - 주택의 가격에 영향을 미치는 요소를 이용하여 회귀분석

3. 13 개의 독립변수와 1 개의 종속변수 (주택가격 중앙값) 으로 구성

- Feature 설명

CRIM 자치시(town) 별 1인당 범죄율,

ZN 25,000 평방피트를 초과하는 거주지역의 비율

INDUS 비소매상업지역이 점유하고 있는 토지의 비율

CHAS 찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)

NOX 10ppm 당 농축 일산화질소

RM 주택 1가구당 평균 방의 개수

AGE 1940년 이전에 건축된 소유주택의 비율

DIS 5개의 보스턴 직업센터까지의 접근성 지수

RAD 방사형 도로까지의 접근성 지수

TAX 10,000 달러 당 재산세율

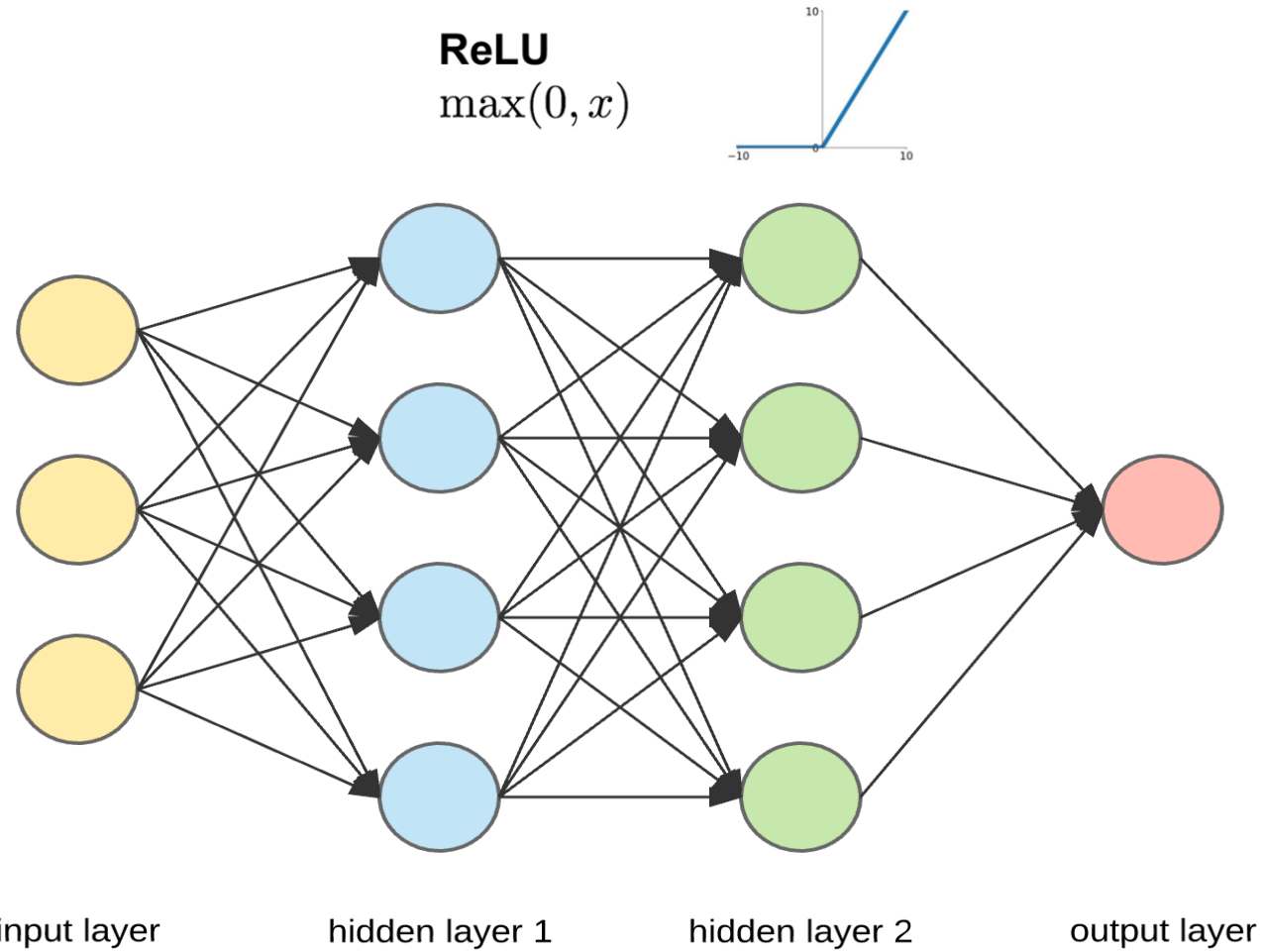
PTRATIO 자치시(town)별 학생/교사 비율

B 1000(Bk-0.63)^2, 여기서 Bk는 자치시별 흑인의 비율을 말함

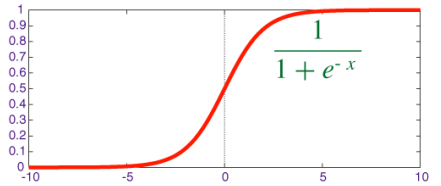
LSTAT 모집단의 하위계층의 비율(%)

MEDV 본인 소유의 주택가격(중앙값) (단위: $1,000)

Binary Classification (Sigmoid)



true



false

Sigmoid 함수

- S curve 형성
- Logistic 함수

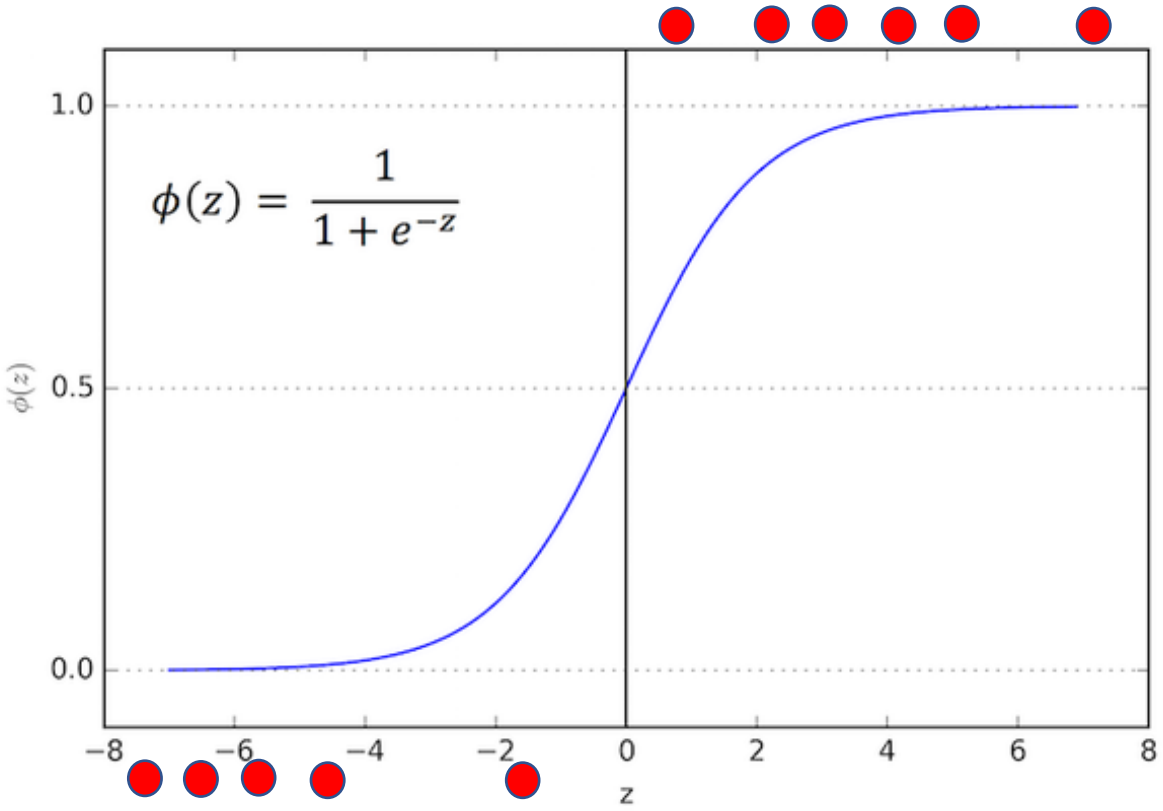
logit

$$f(z) = \frac{1}{1+e^{-z}} \quad (z = \theta X)$$

$\hat{y}$ = 0 if f(z) < 0.5

1 if f(z) ≥ 0.5

- [0, 1] 로 bound 되어 있음
- 미분가능
- 0.5 부근에서 급격히 변화



실습 : 210. Binary Classification

1. Malware Detection - binary classification
2. Neural Network Model 구성 및 Compile
3. Model Train
4. Performance Evaluation

편향된 Data (Biased Data) 의 Model Performance 측정

Confusion Matrix (혼동 행렬)

		True condition	
	Total population	**Condition positive**	**Condition negative**
Predicted Condition	**Predicted condition positive**	**True positive** TP	**False positive** FP
	Predicted condition negative	**False negative** FN	**True negative** TN

- Classification (분류) 성능의 정확성 측정

  TP – 1 을 1 로 제대로 분류, FP – 0 을 1 로 잘못 분류
  FN – 1 을 0 로 잘못 분류, TN – 0 을 0 으로 제대로 분류

- Classification rate (Accuracy) = (TP + TN) / (TP+TN+FP+FN)
  ➔ 단순 정확성. 전체 데이터 중에서, 제대로 분류된 데이터의 비율

Confusion Matrix 를 이용한 분류 모델 성능 평가

- Precision = TP / (TP + FP)
  - 정밀성. **Positive로 예측한 내용 중에, 실제 Positive**의 비율
    → Model 이 sample 을 True 로 분류했을 때 얼마나 자주 맞추었는가 ?
  - positive 분류의 정확성 측정 (1 에 가까울 수록 좋음)

		True condition	
	Total population	**Condition positive**	**Condition negative**
Predicted Condition	**Predicted condition positive**	**True positive**	**False positive**
	Predicted condition negative	**False negative**	**True negative**

ex) 포르노 영상 검출기 – 포르노로 분류했을 때 실제 포르노인 비율
security check 영상 탐지기 – 통과 승인된 사람 중 실제 직원 비율

Confusion Matrix 를 이용한 분류 모델 성능 평가

- Recall (Sensitivity/ True positive Rate) = TP / (TP + FN)
  - 민감도. **전체 Positive 데이타 중에서 Positive로 분류**한 비율 (1 에 가까울 수록 좋음)
  - Positive case 를 놓치고 싶지 않은 경우의 성능 측정

		True condition	
	Total population	**Condition positive**	**Condition negative**
Predicted Condition	**Predicted condition positive**	**True positive**	**False positive**
	Predicted condition negative	**False negative**	**True negative**

Type I error

Type II error

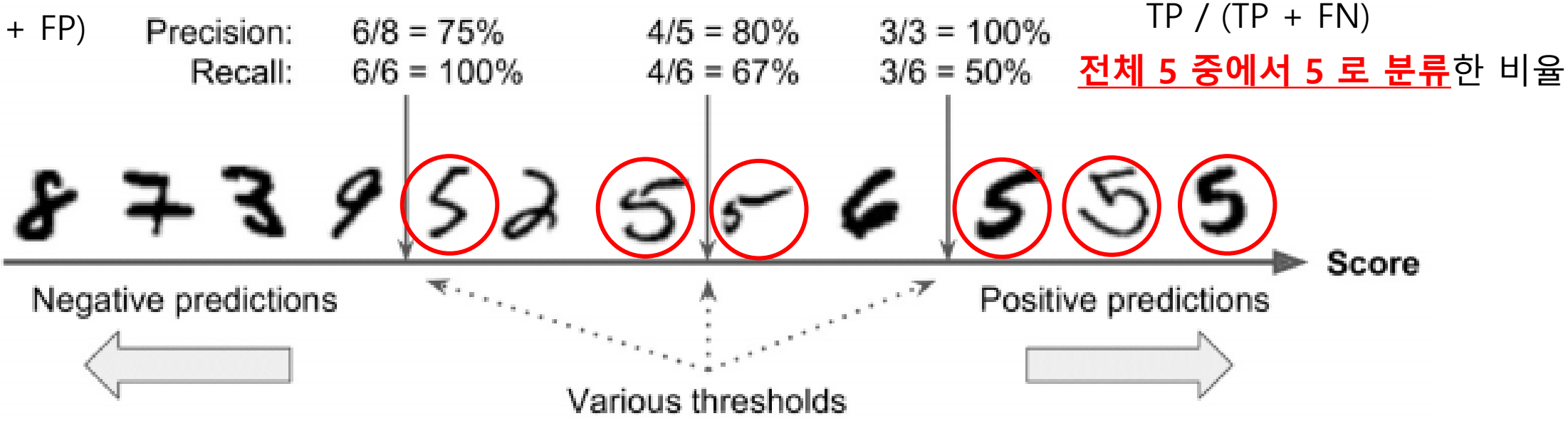
ex) 포르노 영상 검출기 – 전체 포르노 중 포르노로 분류된 비율
security check 영상 탐지기 – 전체 직원 중에서 통과로 분류된 비율

Precision/Recall Trade-off

- 숫자 중 5 를 검출

5로 예측한 내용 중에, 실제 5의 비율

TP / (TP + FP)



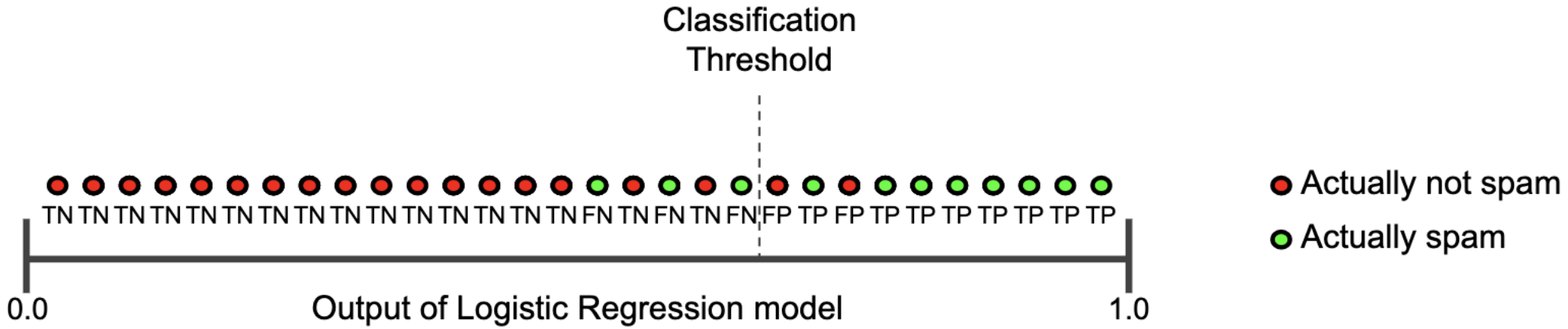
TP / (TP + FN)

전체 5 중에서 5 로 분류한 비율

5 가 검출에서 누락되지 않기 원하는 경우

5 가 정확히 검출되기 원하는 경우

Precision and Recall Trade-off

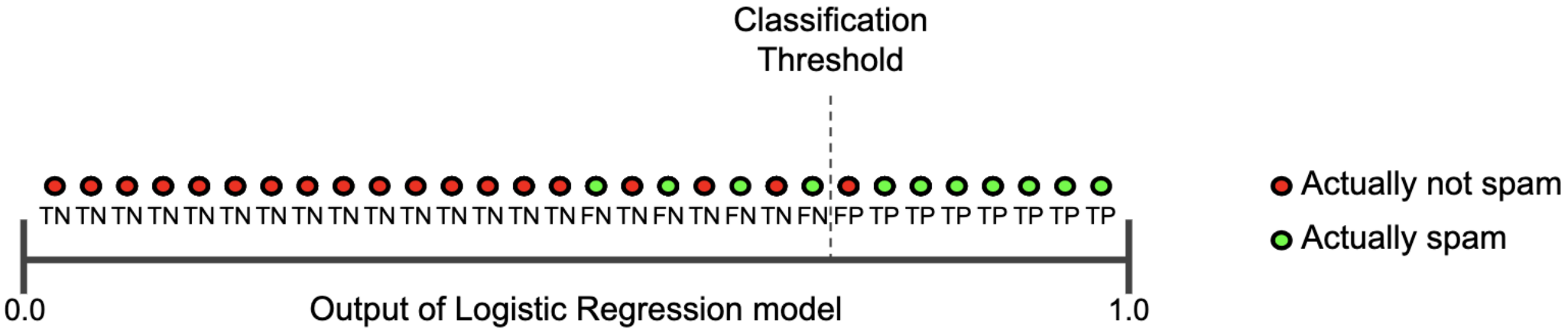


True Positives (TP): 8	False Positives (FP): 2
False Negatives (FN): 3	True Negatives (TN): 17

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{8}{8 + 2} = 0.8$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{8}{8 + 3} = 0.73$$

Precision and Recall Trade-off

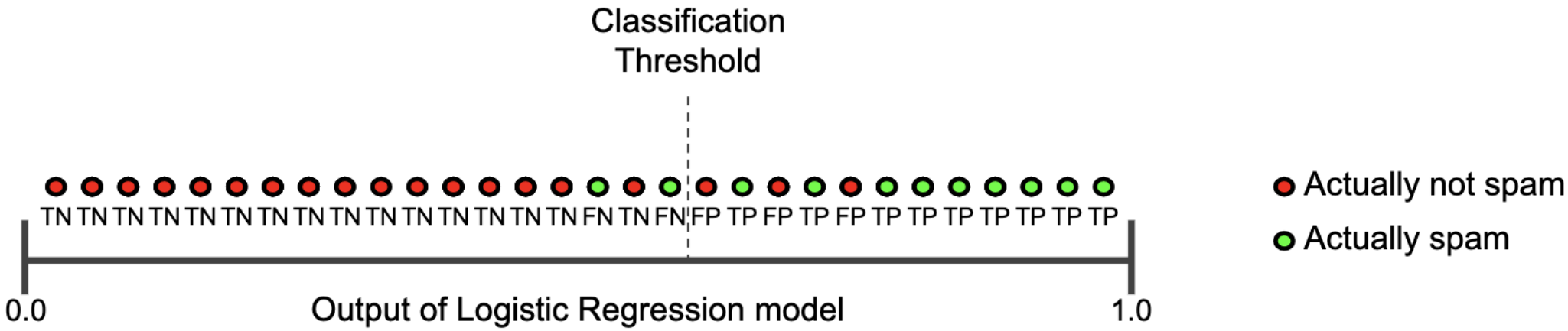


True Positives (TP): 7	False Positives (FP): 1
False Negatives (FN): 4	True Negatives (TN): 18

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{7}{7 + 1} = 0.88$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{7}{7 + 4} = 0.64$$

Precision and Recall Trade-off



True Positives (TP): 9	False Positives (FP): 3
False Negatives (FN): 2	True Negatives (TN): 16

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{9}{9 + 3} = 0.75$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{9}{9 + 2} = 0.82$$

Precision / Recall 활용 방법

- Confidence 수준을 올리고 싶으면 Precision 을 높이고 Recall 을 낮추도록 Threshold 조정. 너무 많은 case 를 놓치고 싶지 않은 경우 Recall 을 높이고, Precision 을 낮춘다.

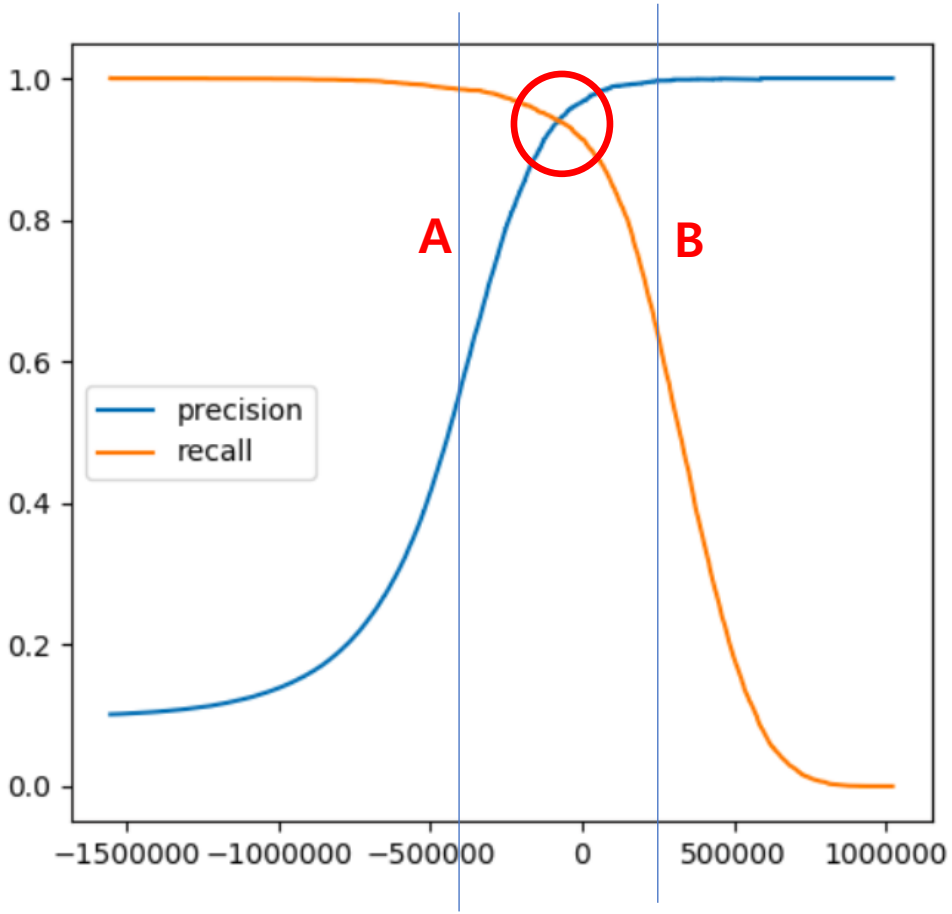
- 전체적 성능 측정에 활용 (조화평균)

$$\text{F1-Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F1-Score = 0 ---> Poor (P=0 or R=0)
F1-Score = 1 ---> Perfect (P=1 and R=1)

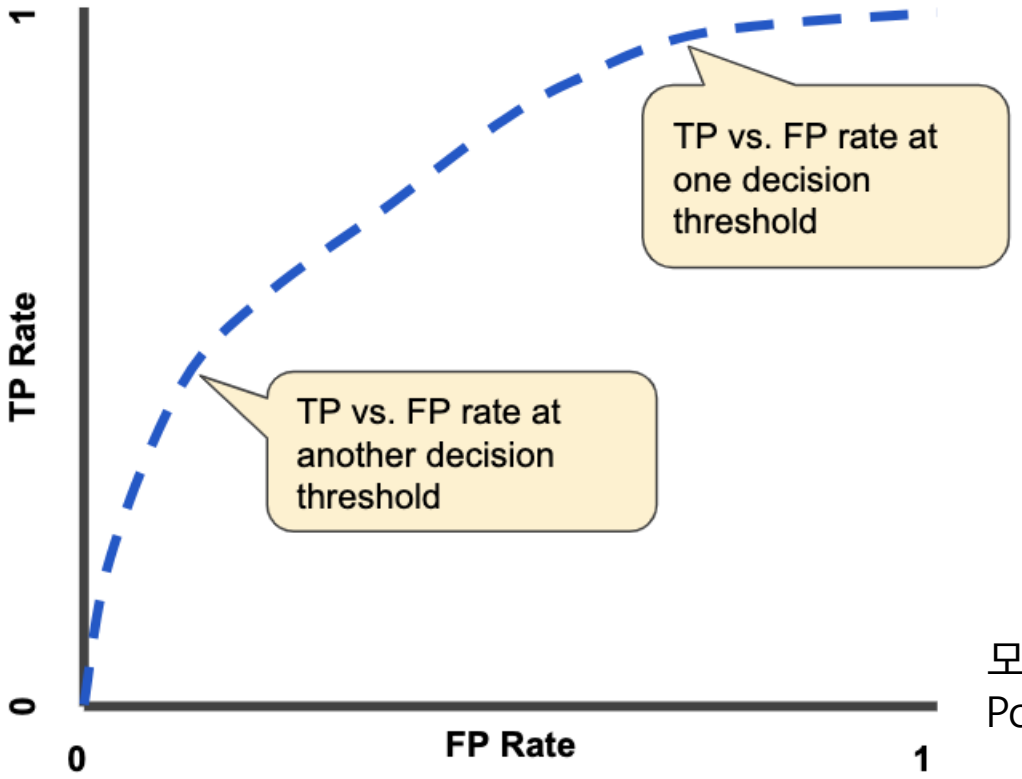
A – High Recall / Low Precision

B – High Precision / Low Recall

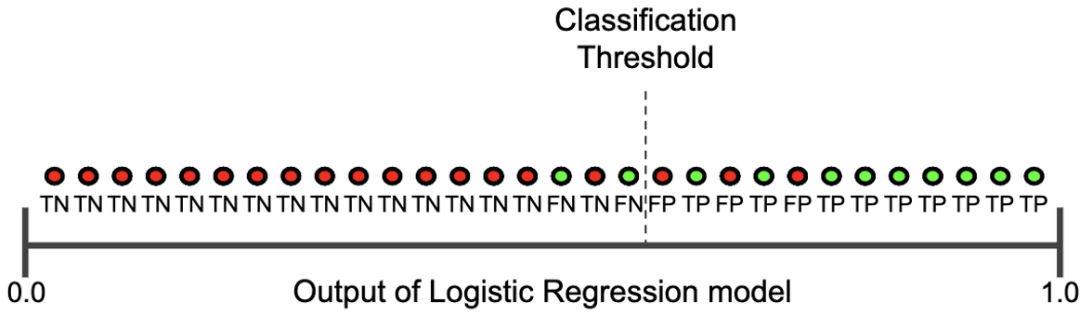


다양한 threshold 에서의 TP vs. FP

모든 Positive 를
Positive 로 정확히 분류



Negative 를 Positive 로 잘못 분류한 비율

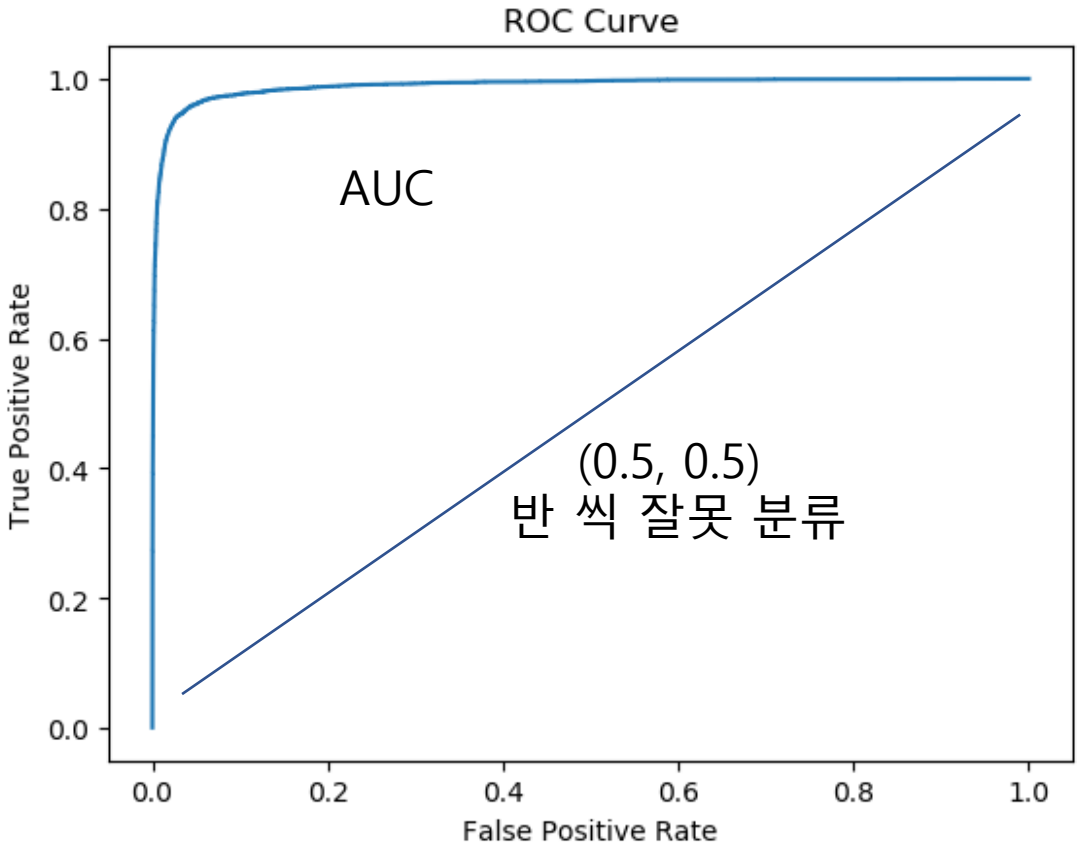


모든 Negative 가
Positive 로 잘못 분류

$$TPR = \frac{TP}{TP + FN} \quad = \text{Recall}$$

$$FPR = \frac{FP}{FP + TN}$$

ROC Curve (수신자 조작 특성 곡선)



- ROC_AUC (Area Under the Receiver Operating Characteristic Curve) 라고도 함
- roc_auc_score 를 이용하여 **분류기(classifier) 간의 성능 비교**를 할 수 있다.

* ROC (Receiver Operating Characteristic) curve 는 radar 상의 적기 탐지를 위해 개발되었던 분석 기법

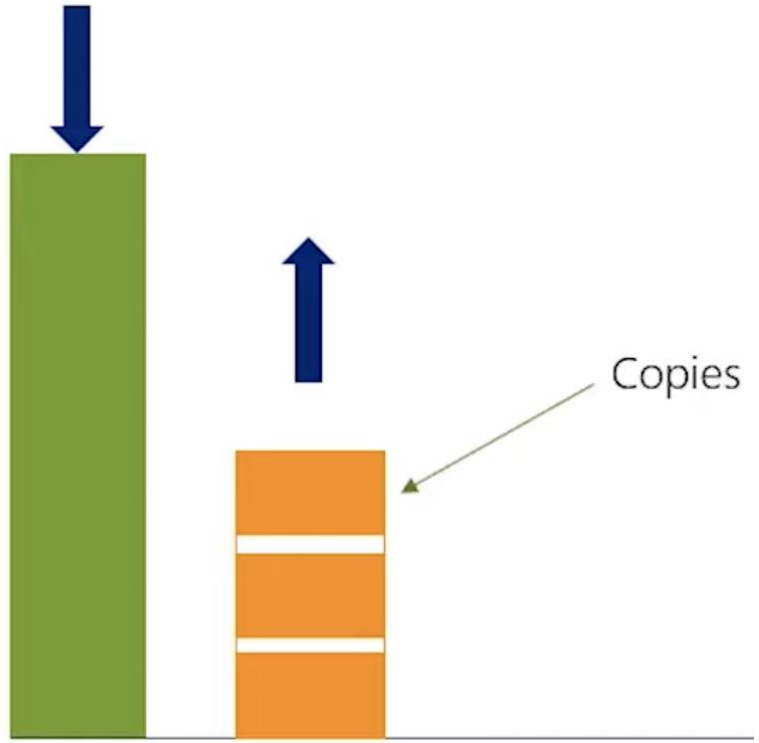
Highly Imbalanced Dataset

Imbalanced Class 의 문제 해결 방법

- Down-sampling
  - 주로 많이 나타나는 class 의 sample 개수 축소
- Up-sampling
  - minor class 의 sample size 증가
- Data Generation
  - 유사한 data 생성 (ex. Image Augmentation)
- Sample Weights 조정
  - Cost Function 계산 시 small class 에 높은 가중치 부여

Unbalanced Data 의 sampling method

- Random over-sampling (ROS)
  - fraud sample 을 여러번 Training set 에 추가
- Random under-sampling (RUS)
  - 정상 sample 을 random 하게 제거
- Both
  - over-sampling + under-sampling



실습 : 220. Highly Imbalanced Fraud Data 분류

- 신용카드 회사의 이상 거래 검출 문제
- Class Weight 조정, Oversampling
- ROC-AUC 를 이용한 모델 성능 비교
- Confusion Matrix 를 이용한 Precision, Recall 비교

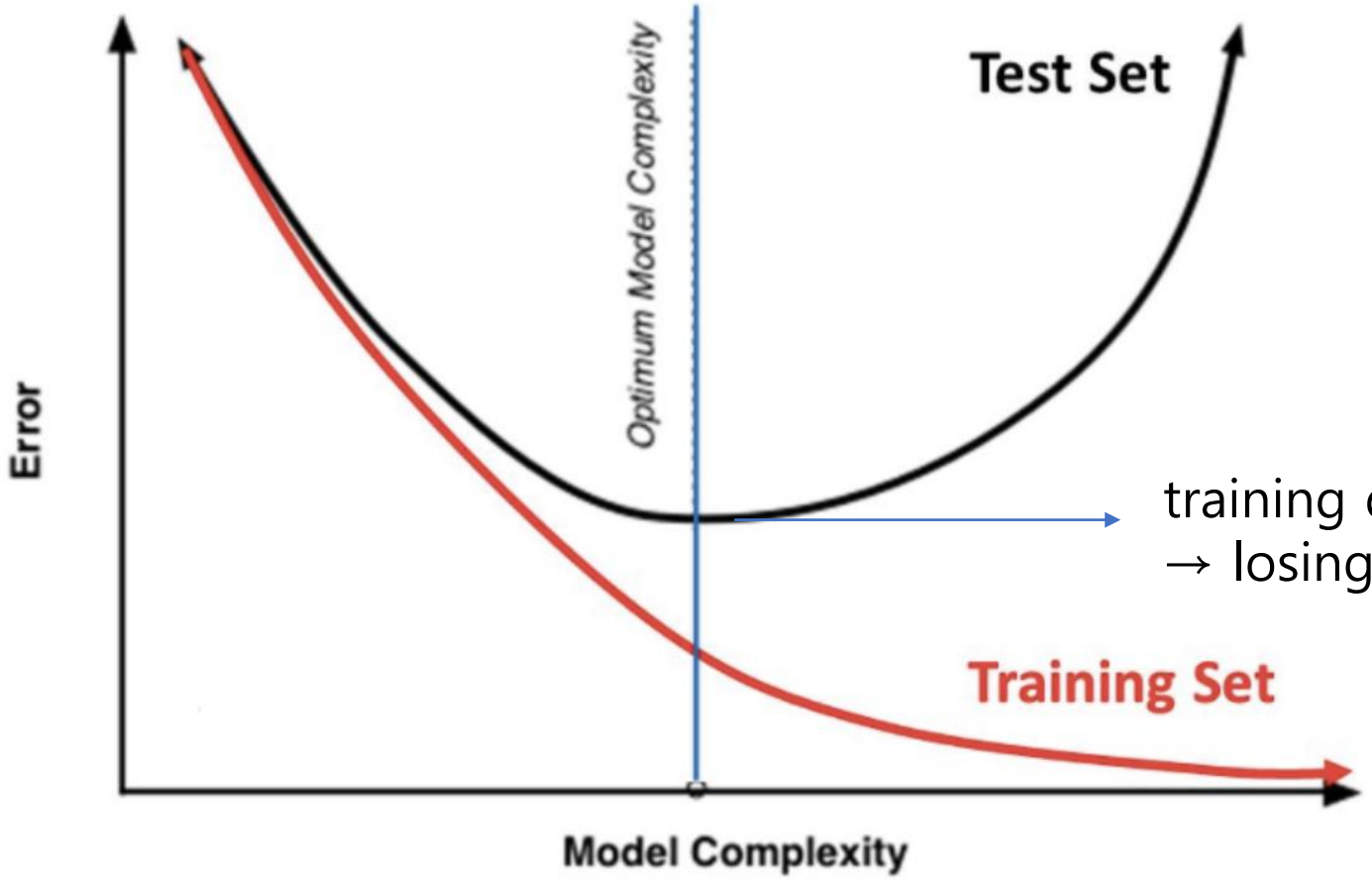
Overfitting 방지 기법

Regularization

Neural Network 의 Overfitting / Underfitting

- Overfitting
  - model 이 data 에 비해 too complex
  - hidden Layer 에 neuron 이 너무 많을 때 발생
- Underfitting
  - model 이 data 의 complexity 를 capture 하지 못할 때 발생
  - hidden layer 에 neuron 이 너무 적을 때 발생
- Solution
  - validation data 사용 → optimum 한 neuron 숫자 결정
  - Get more data
  - Dropout 과 같은 regularization 방법 사용

What is Overfitting ?



training dataset 을 기억하기 시작
→ losing generalization

Overfitting 방지 기법 (Regularization)

- Linear Regression (선형회귀)

$$J(W) = MSE_{train}(W) + \lambda\Omega(W)$$

손실함수 Train loss

λ : regularization 강도 (hyperparameter)
Ω : regularizer

- 대표적 regularizer

1. L1 regularization : $\Omega(W) = ||W||_1$ (lasso regularization)
2. L2 regularization : $\Omega(W) = ||W||_2$ (ridge regularization)
3. Elastic net regularization : L1 + L2

Overfitting 방지 기법 (Regularization)

- Logitstic Regression (이진 분류)

$$J(\theta) = -\frac{1}{\mathrm{m}}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] + \lambda\frac{1}{2m}\sum_{i=0}^{m}\theta_j^2$$

regularizer

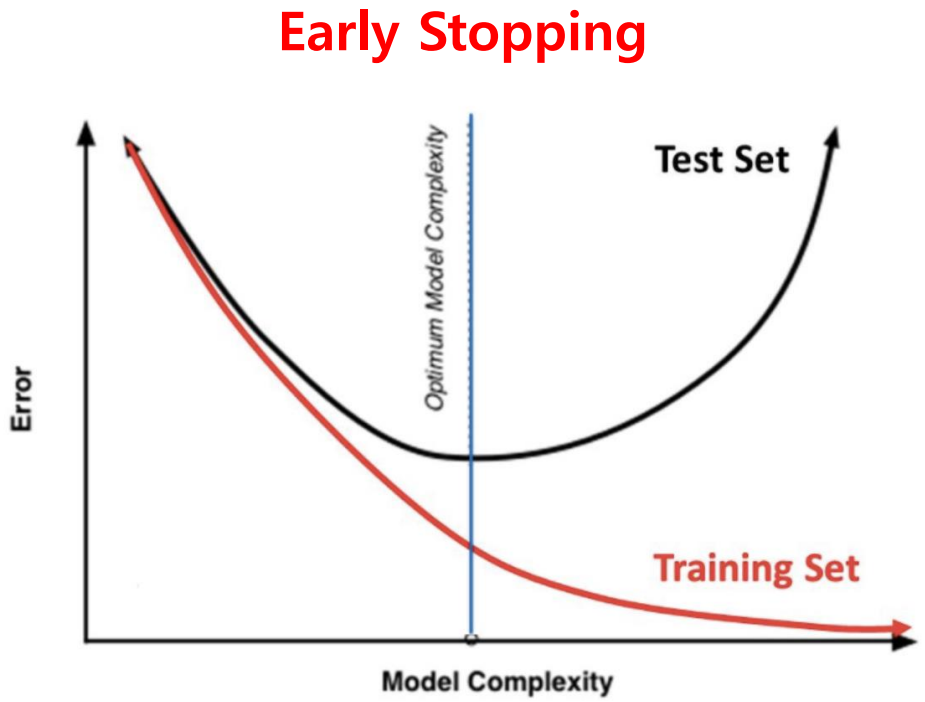
Binary cross-entropy 함수

λ : regularization 강도

요점 – weight(θ) 를 의도적으로 더해주면 optimization 은 더해준 만큼의 weight(θ) 영향을 줄여주는 방향으로 진행

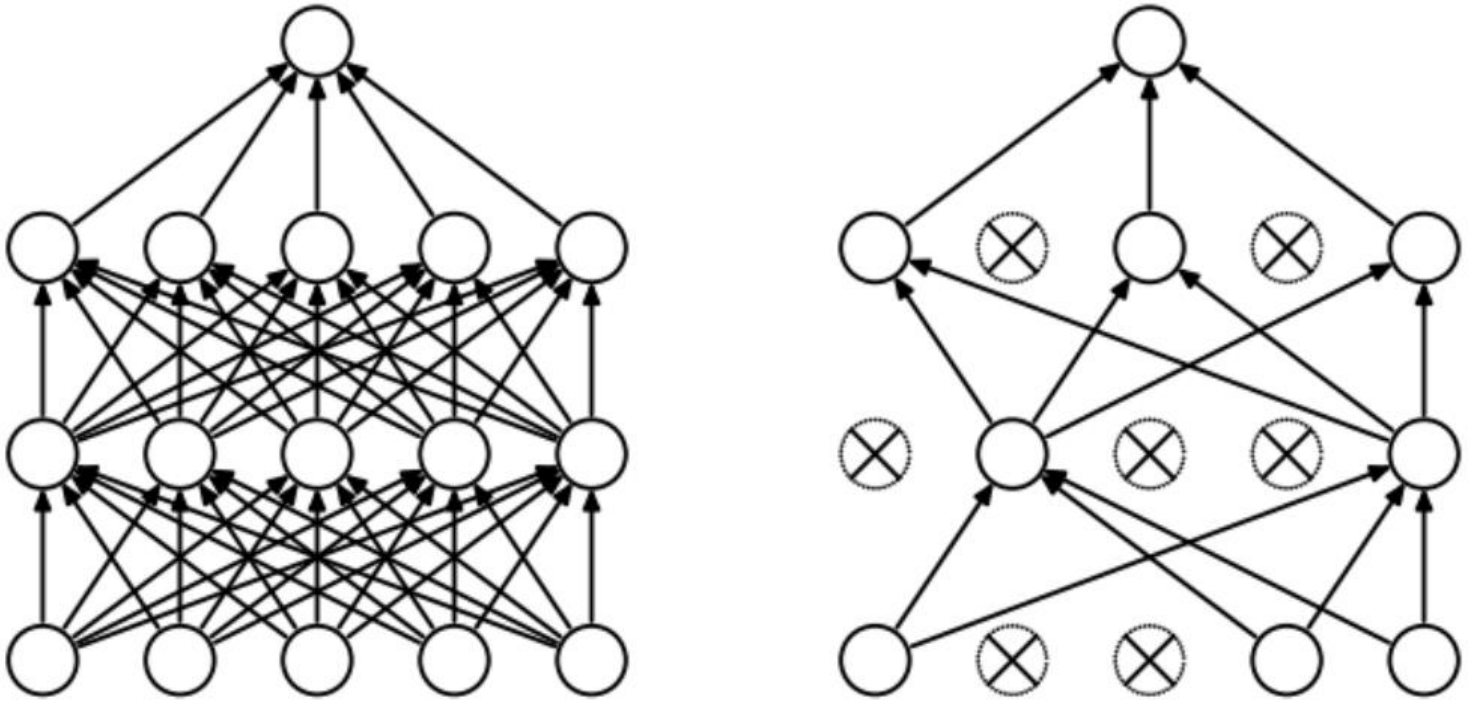
Early Stopping

- L2 regularization 과 비슷한 효과이면서 computationally cheaper
- 두가지를 함께 사용하면 optimal hyper-parameter를 정하기 어려운 경우 도움이 된다.



Dropout regularization

Random 한 drop out 을 통한 과적합 방지 (특정 feature 의존 방지)



(a) Standard Neural Net

(b) After applying dropout.

CNN (Convolutional Neural Network)

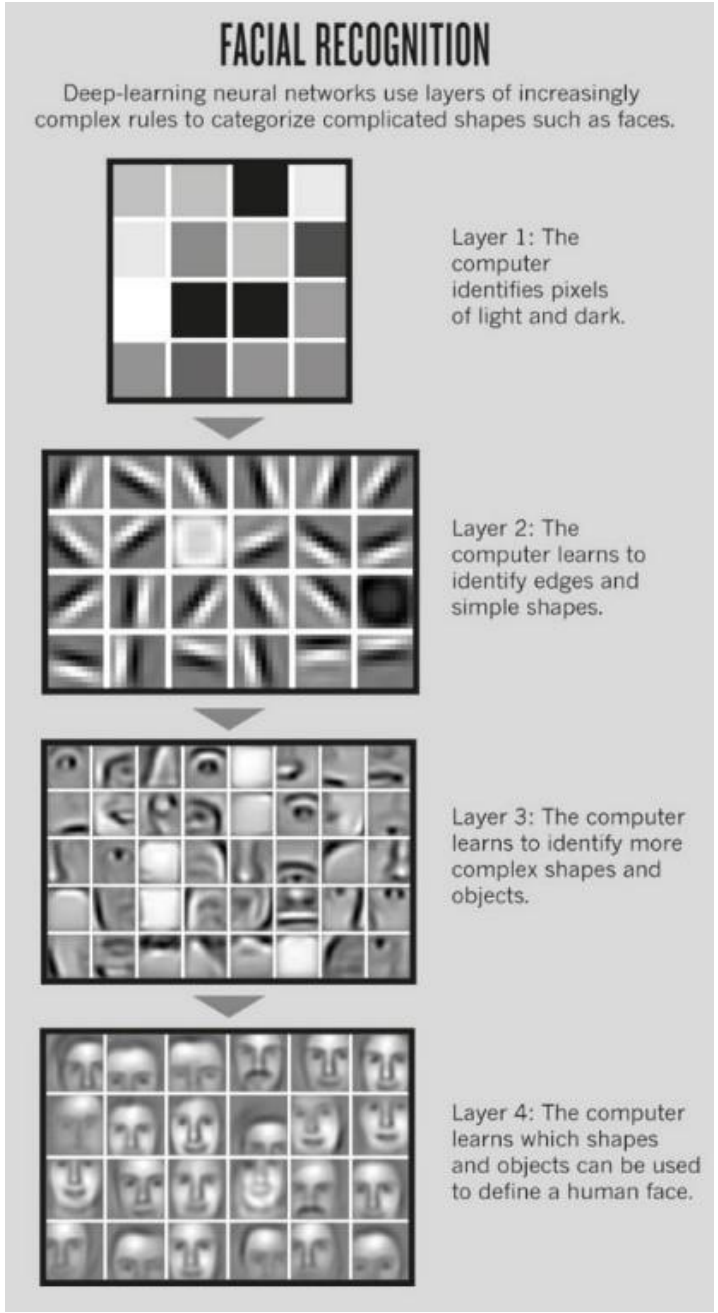
Multiple Levels of Abstraction

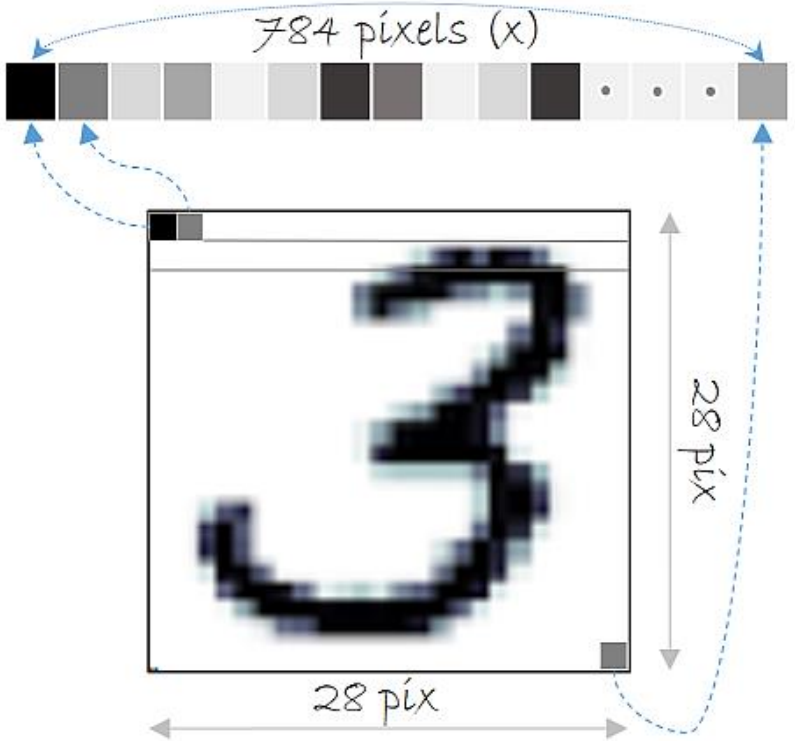
Pixel 정보 인식

Edge/Simple Shape 특성 학습

Complex Shape 특성 학습

얼굴인식에 필요한 특성 학습





input image 처리

60000 x **28 X 28** → reshape(60000, **784**)

Image Data 의
공간적, 지역적 특성 상실

If 1 mega pixel → 1,000 x 1,000 x 3 = 3 million features !!!

→ **300 만 차원 x Layer 수 x 각 Layer 의 Neuron 수**

→ **계산량 급증**

→ Image Data 를 처리하기 위한 특별한 구조의 Neural Network 필요

Image Data의 large input features

8 Megapixel resolution

3264 (w) x 2448 (h) x 3 (RGB) =

23,970,816 per image*

* ML training time impacted

CNN 의 특별한 Layers

- **Convolutional Layer (합성곱층)**

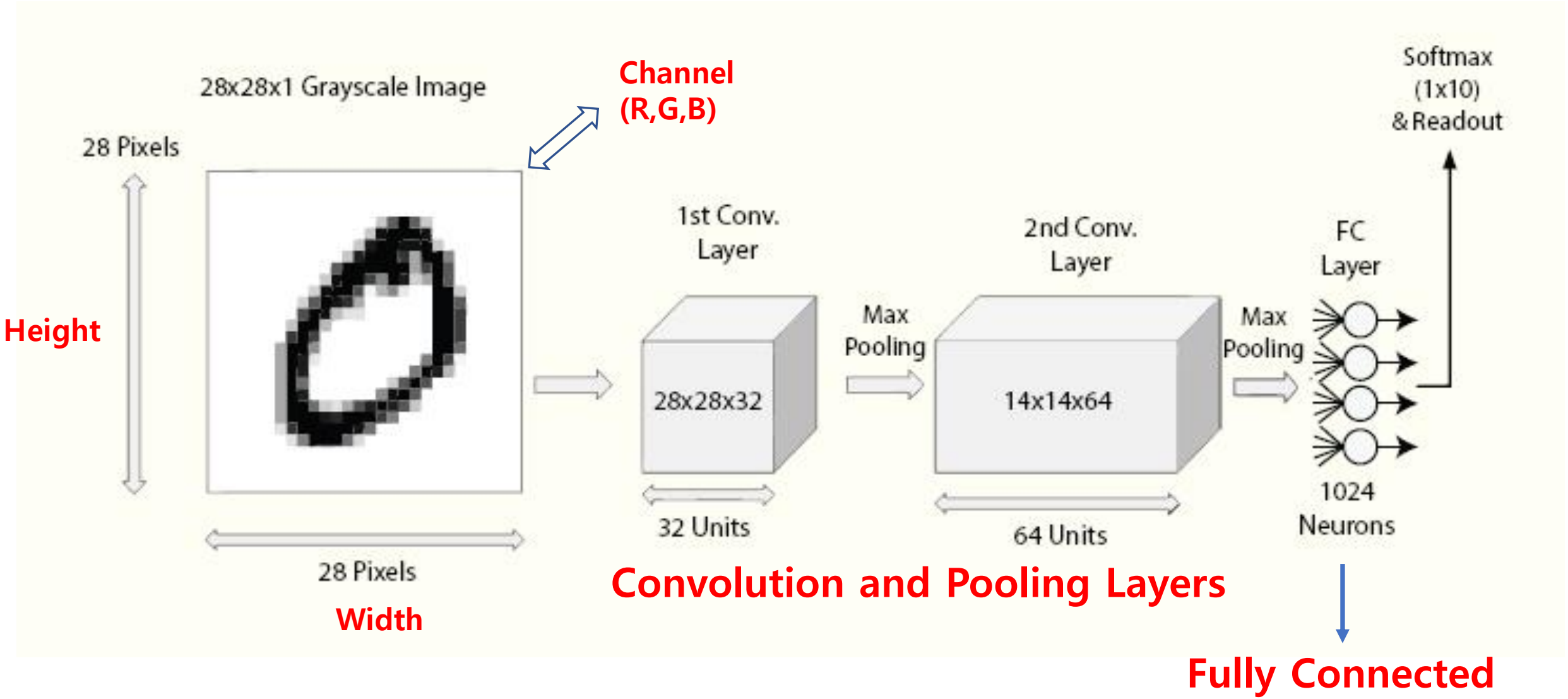
  - Image 정보의 공간적 지역 특성 보존
  - Filter (Kernel) 을 이용한 이미지 특성 추출

- **Pooling Layer (풀링층)**

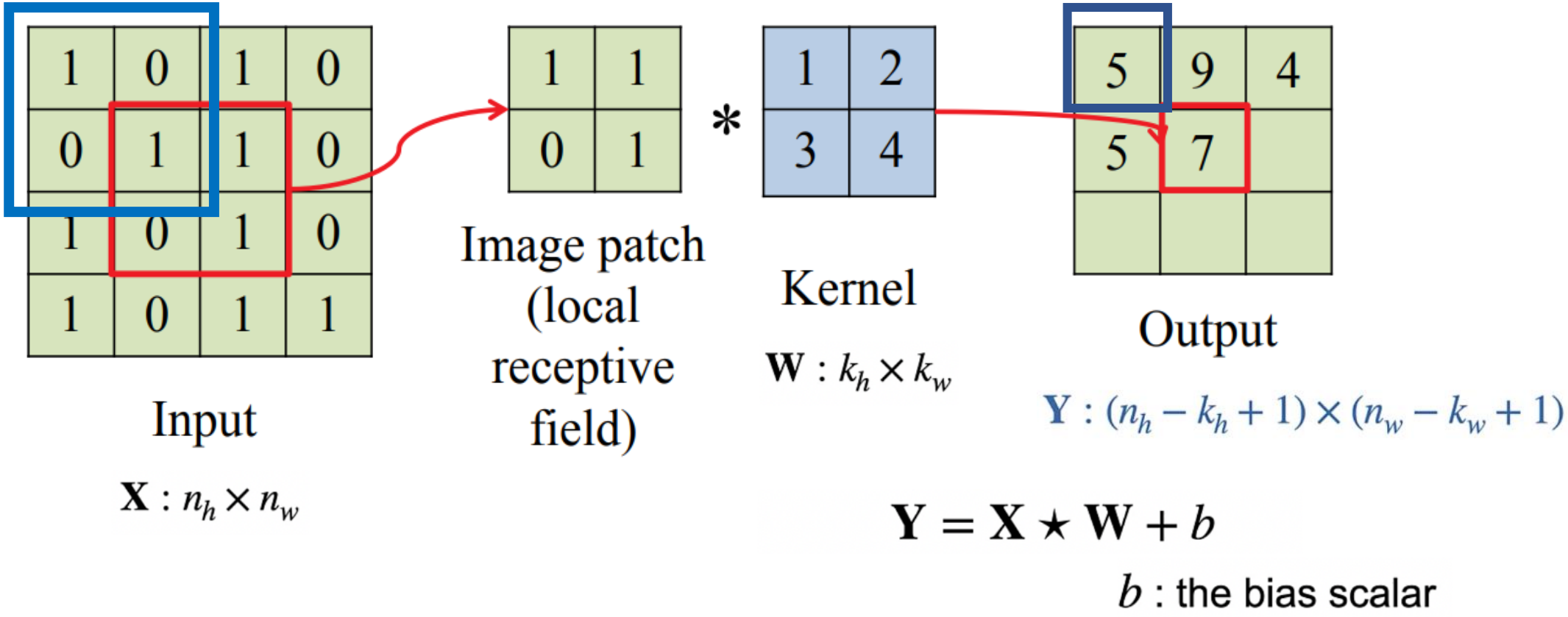
  - Image data 의 정보 손실 없는 압축

  ➔ 계산량 및 메모리 사용량 축소, 파라미터의 수 감소 (과적합 방지)

2. Dimensions of Layers



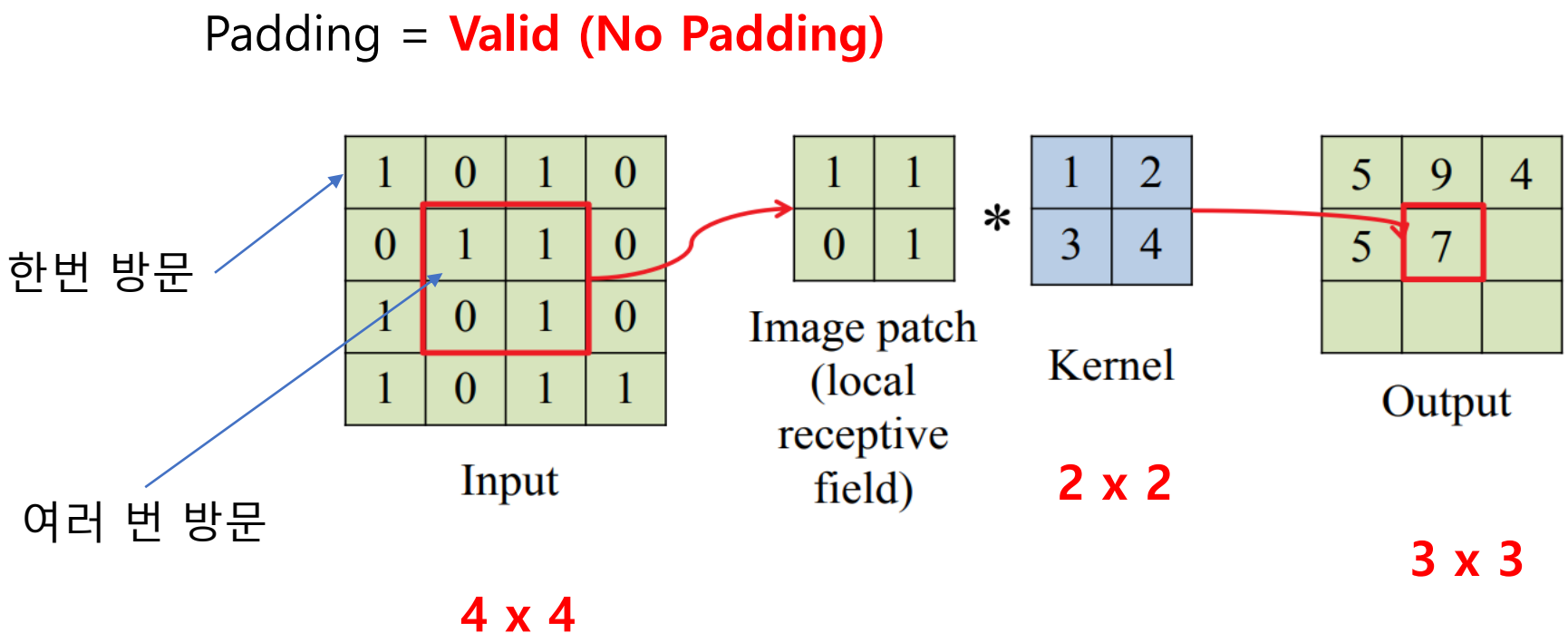
How Convolution works ?



Kernel_size=(2, 2), stride=(1, 1), No padding

Padding

- 주변 pixel 과 중앙 pixel의 방문 횟수 차이 조정



- Input size = output size (➔ Input 의 주위에 0 pixel padding)
- Padding = “same” convolution

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

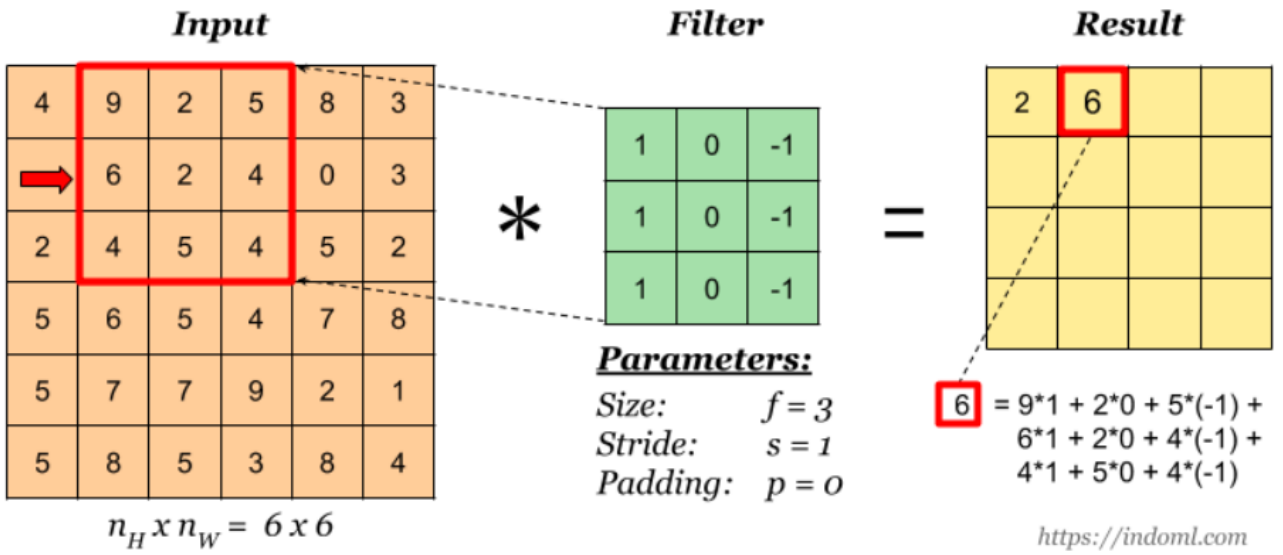
Kernel

0	-1	0
-1	5	-1
0	-1	0

114	328	-26	470	158
53	266	-61	-30	344
403	116	-47	295	244
108	-135	256	-128	344

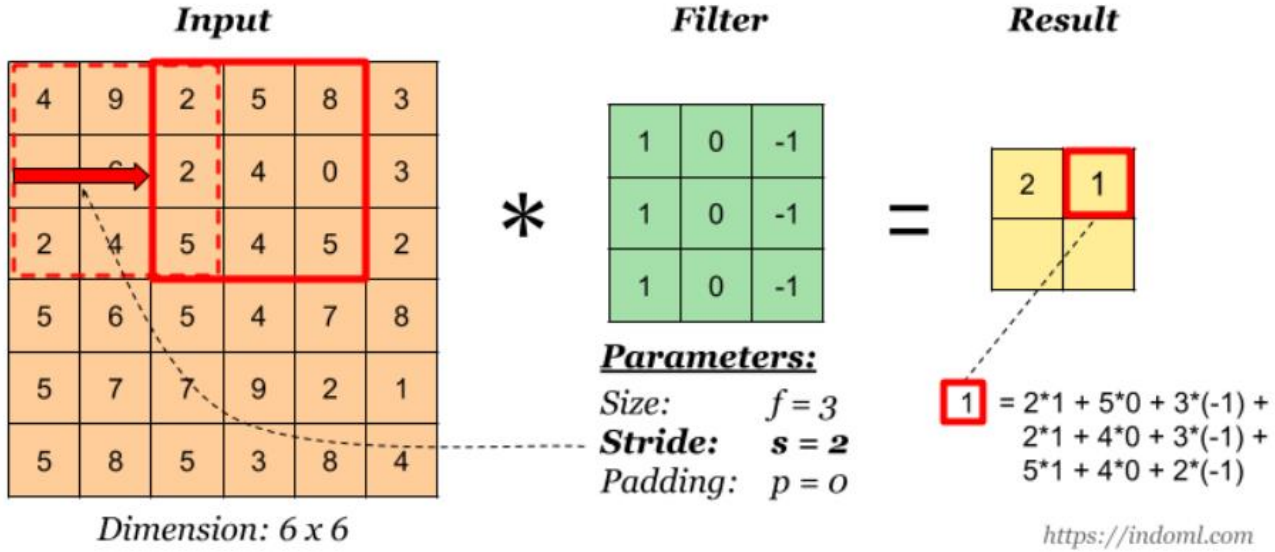
Striding

Stride = 1



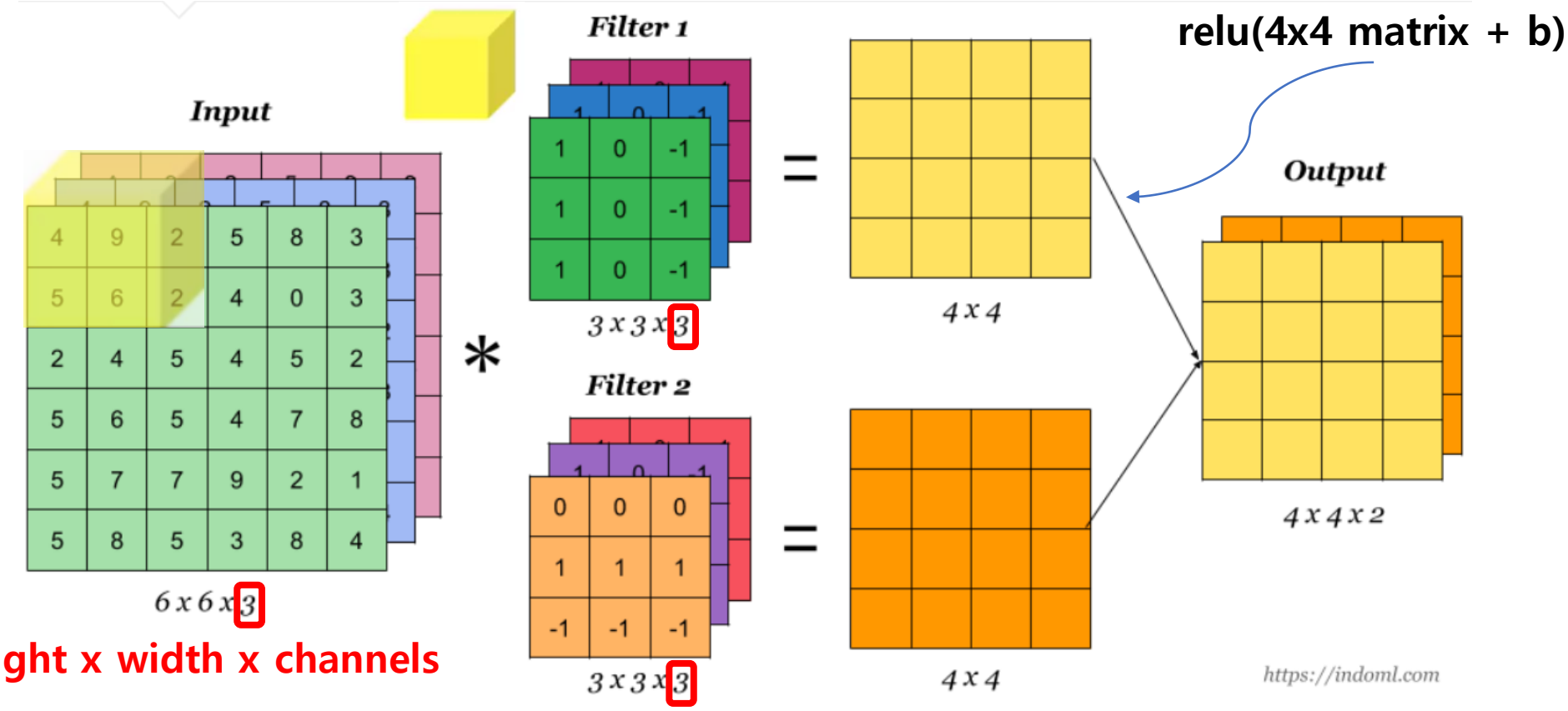
4 x 4

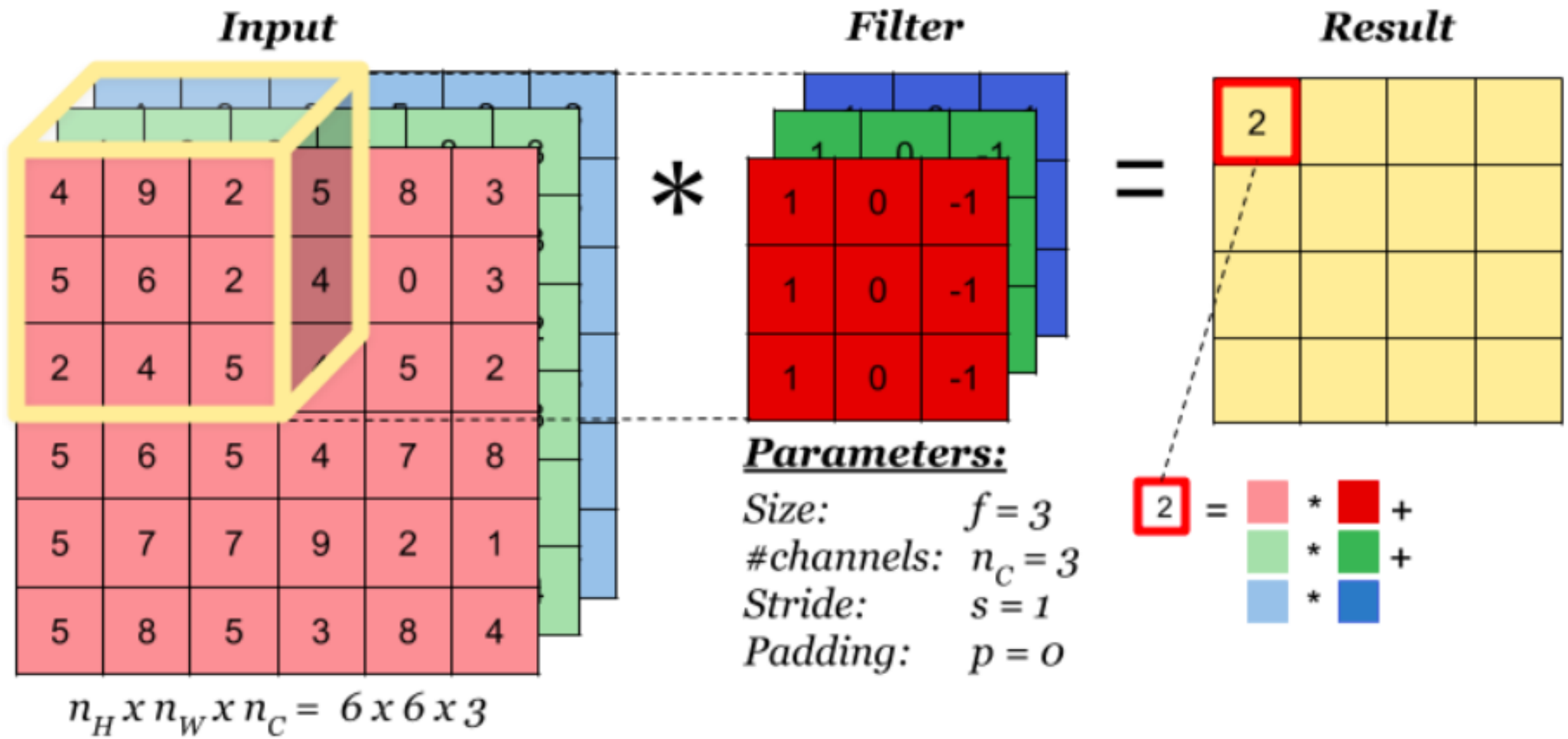
Stride = 2



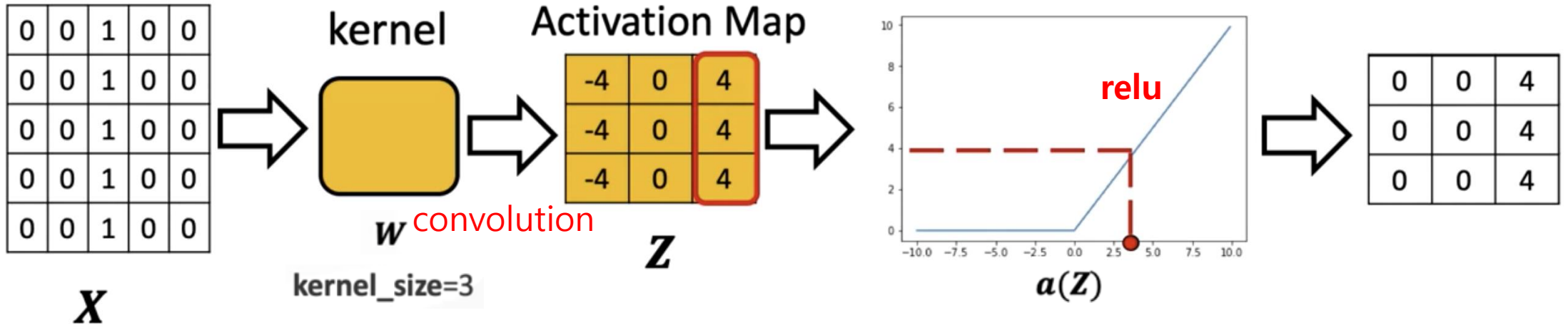
2 x 2

Convolutions over Volumes (RGB images)




Input
Filter
Result
4 9 2 5 8 3
5 6 2 4 0 3
2 4 5 5 2
5 6 5 4 7 8
5 7 7 9 2 1
5 8 5 3 8 4
1 0 -1
1 0 -1
1 0 -1
2
Parameters:
Size: f = 3
#channels: n_C = 3
Stride: s = 1
Padding: p = 0
n_H x n_W x n_C = 6 x 6 x 3

Activation Function (relu) 적용

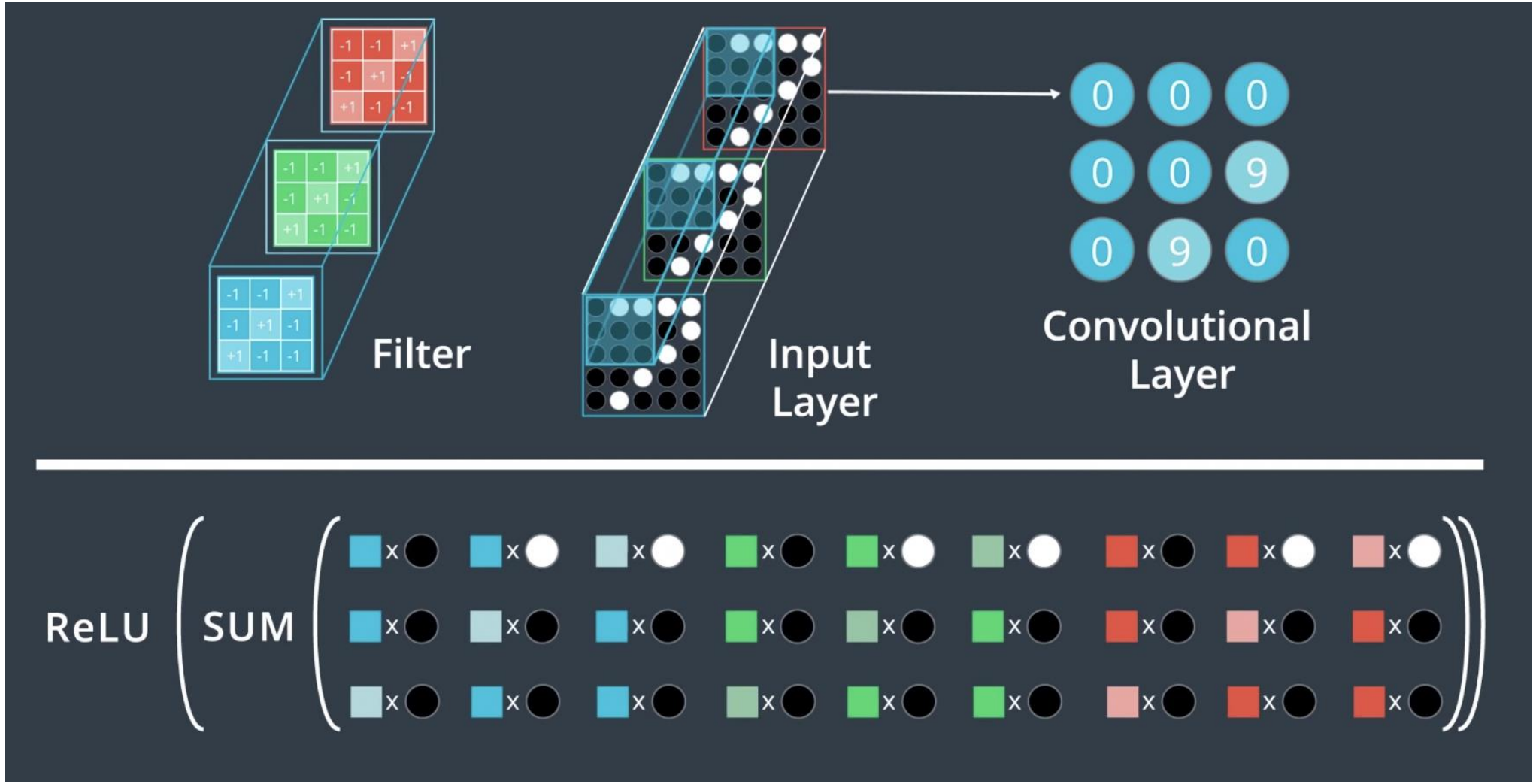



```
image=torch.zeros(1,1,5,5)
```

channel

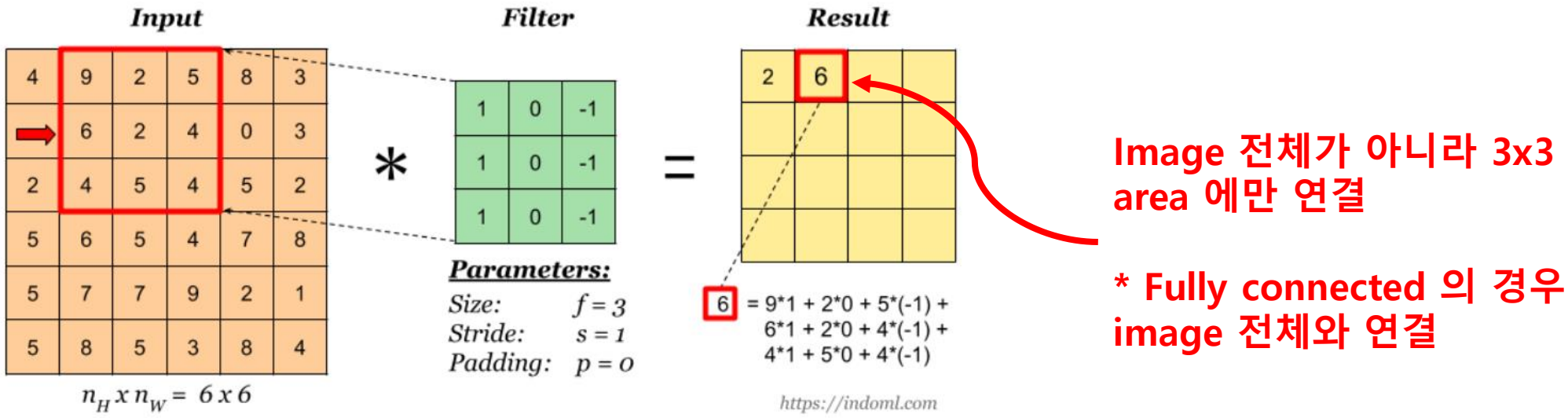
$$Z = W * X + b$$

Convolution + Activation Function



Convolution Layer 의 2 가지 특성

1. **Locality**
   - kernel size 만큼의 작은 구역 (patch) 의 인접한 pixel 들에 대한 correlation 관계를 비선형 필터를 적용하여 추출
   - 이러한 필터를 여러 개 적용하면 다양한 local 특징을 추출 가능



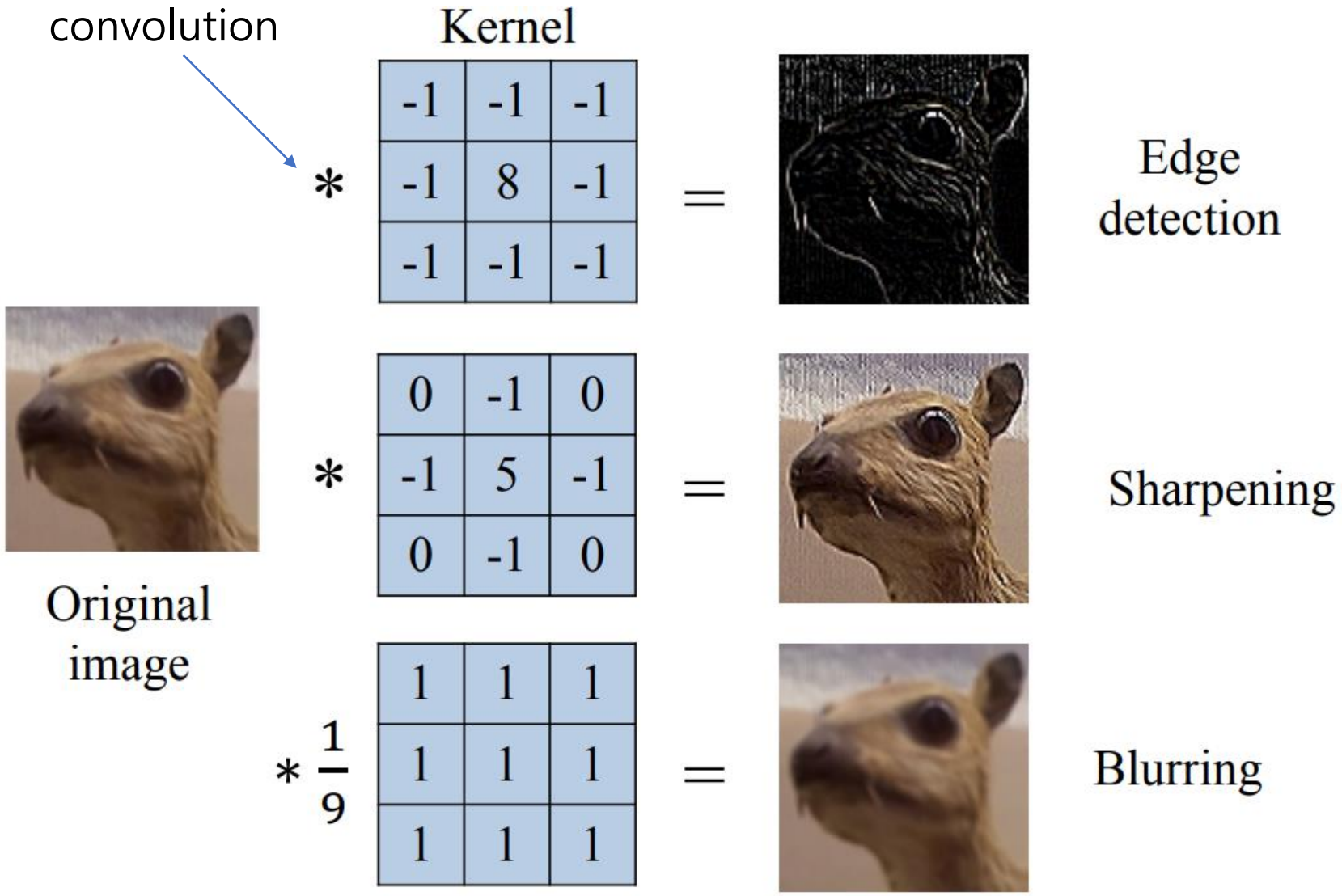
2. **Parameter Sharing**

- input 상의 모든 patch 들은 동일한 kernel 을 적용하여 next layer 의 output 을 출력한다.

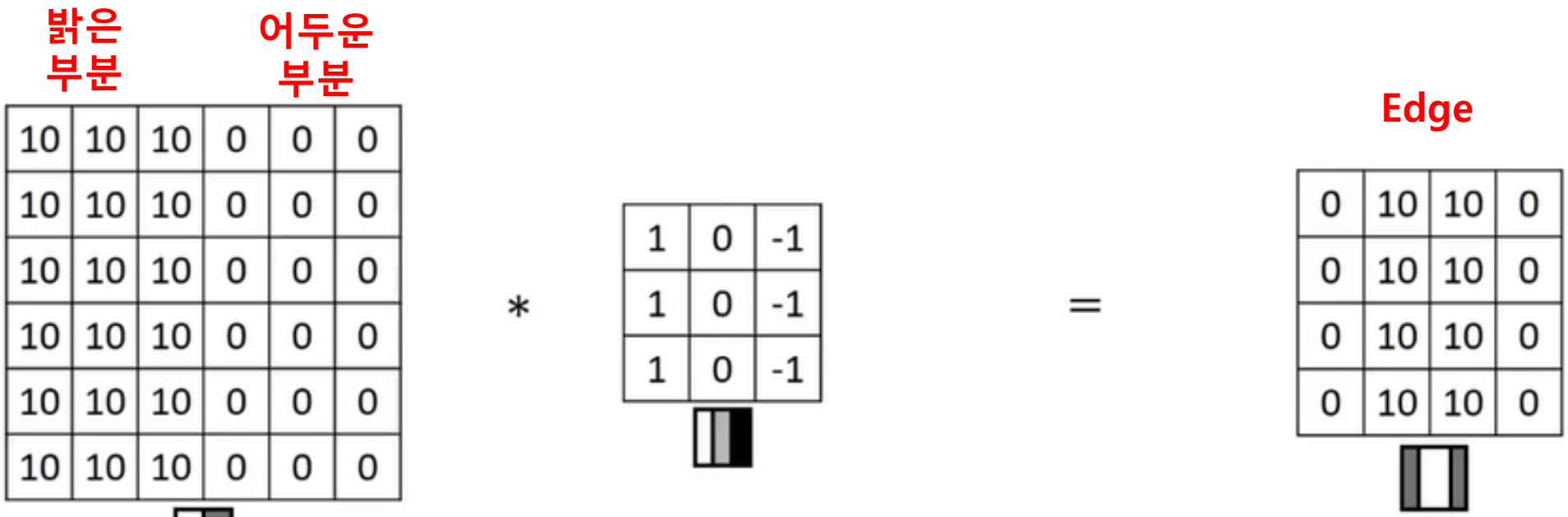
  ex) vertical edge detector – image 전체에 동일한 kernel 적용

- Fully connected layer 를 image data 에 사용할 경우에 비해 parameter 의 수를 획기적으로 줄임

Kernel(Filter) 의 특성 추출 예

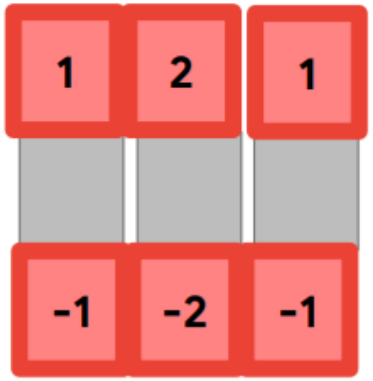


Ex) Edge Detector

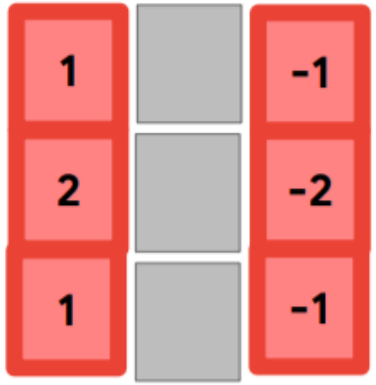


Backpropagation 에 의한 자동 학습

여러장의 kernel 을 조합한 특성 추출

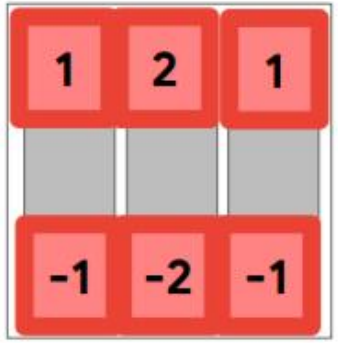


Detects
horizontal edges

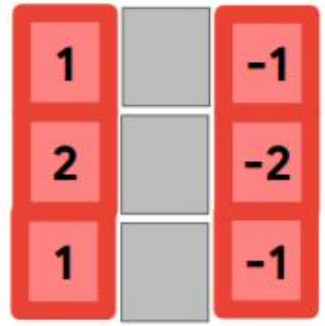


Detects
vertical edges

Kernel 1

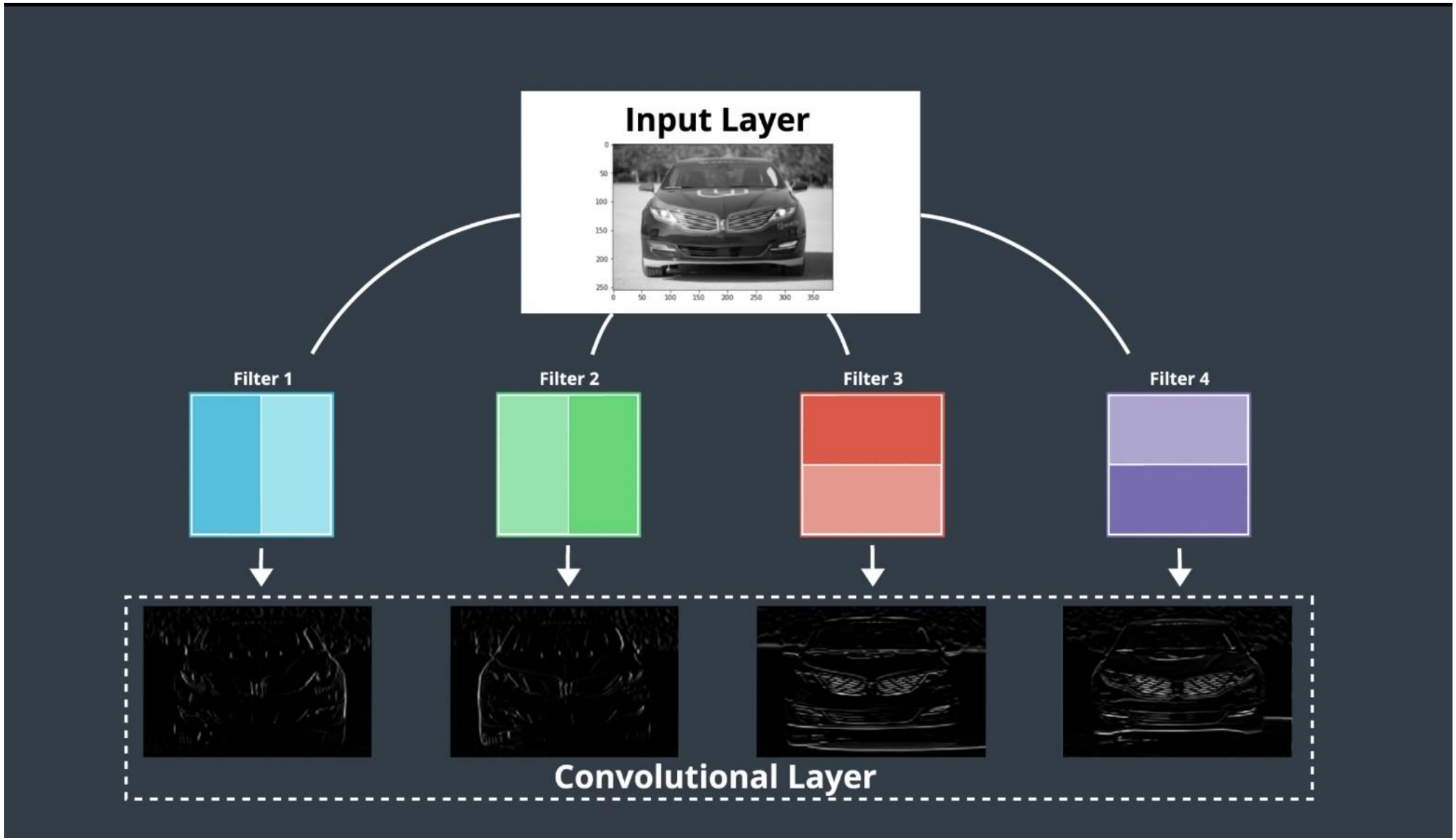


Kernel 2



Sum of the two filter outputs

=


Input Layer
Filter 1
Filter 2
Filter 3
Filter 4
Convolutional Layer

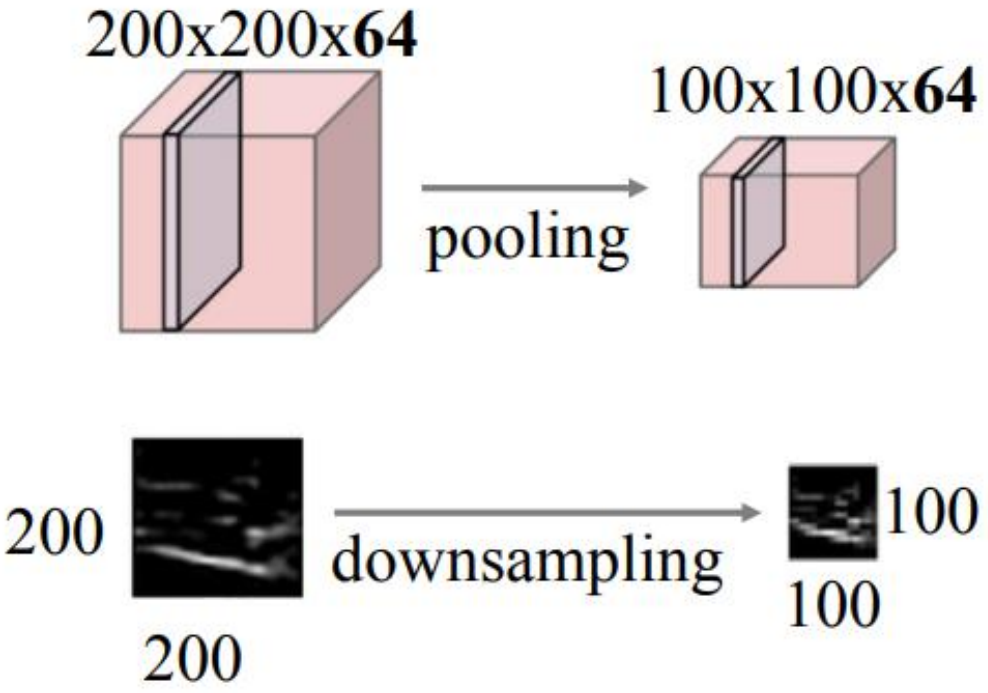
- CNN 이전에는 edge detection filter 를 computer vision 전문가들이 모두 manually 만들어 줌

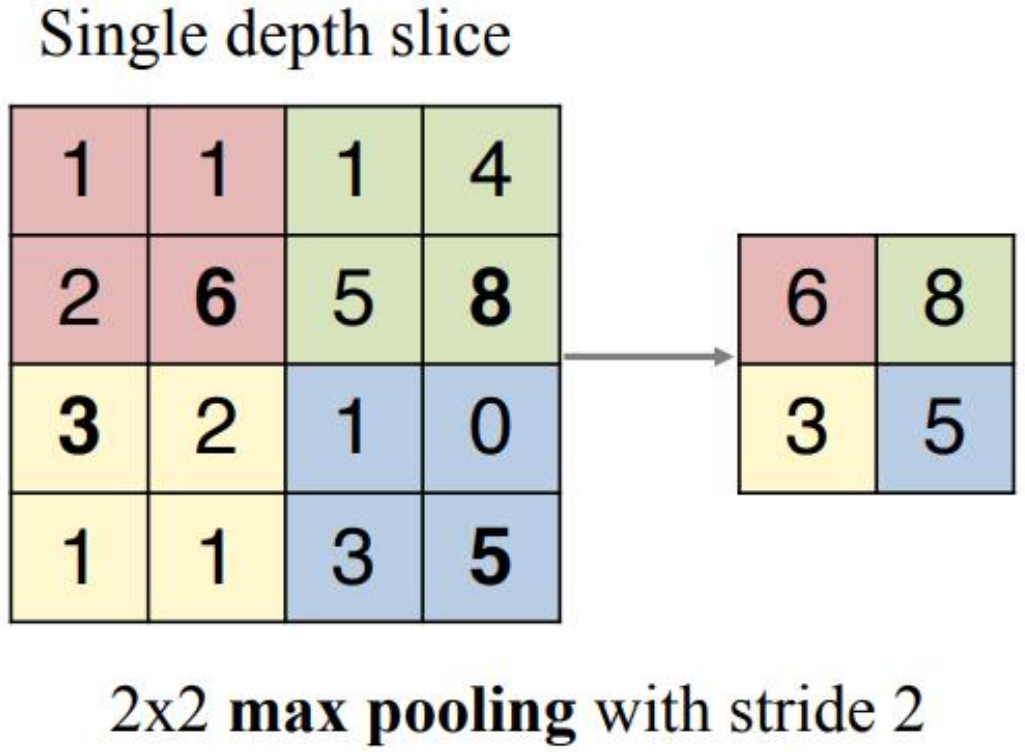
  ex) 수직 / 수평 / 45 도 / 명암 구분 filter 등

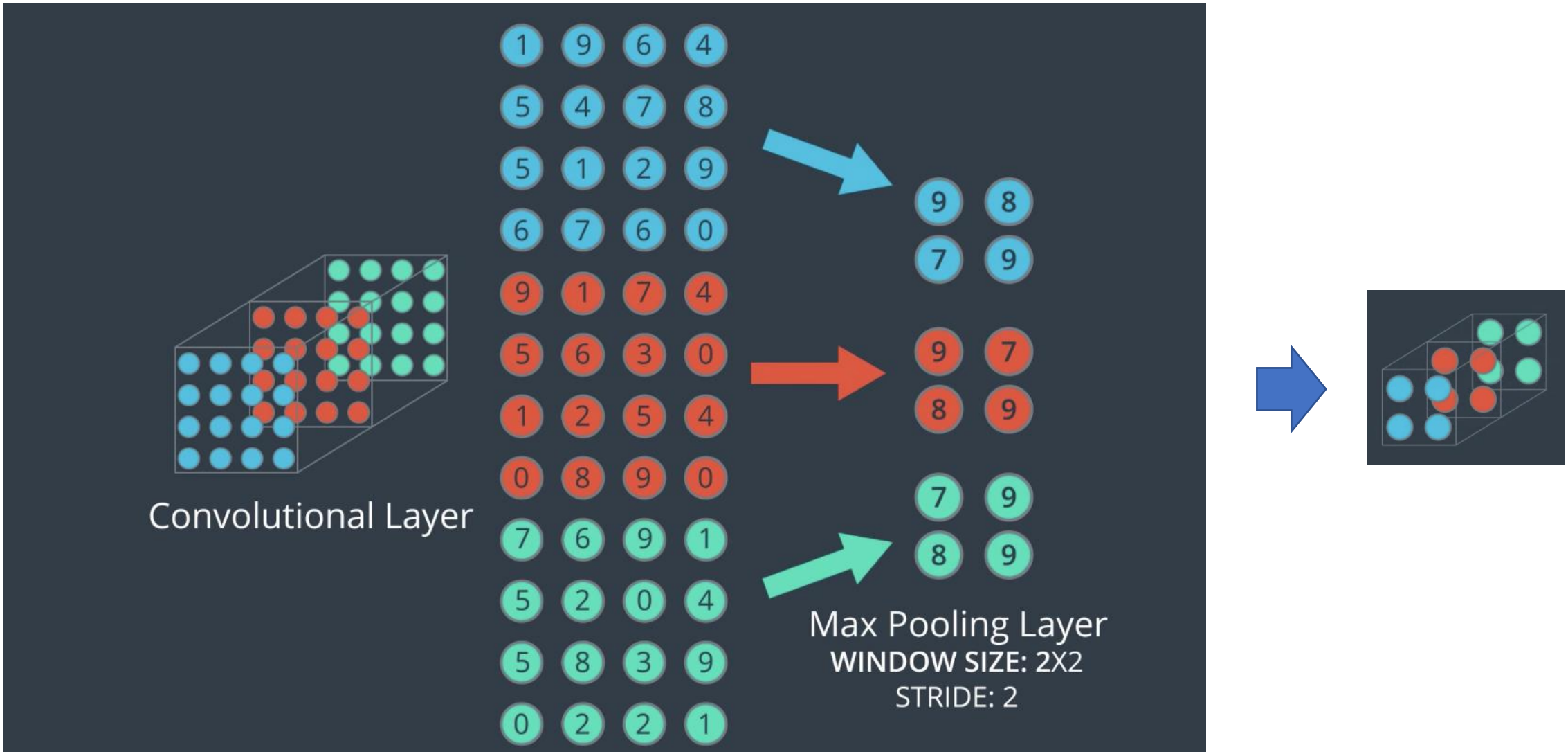
  ➔ Neural Network 은 훨씬 더 다양한 특성의 filter 들을 back-propagation 을 이용하여 자동으로 학습하고 스스로 만들어 냄

How Pooling works ?

- Pooling 의 뉴런은 가중치가 없음
- 최대, 평균을 이용한 이미지 subsampling (부표본 작성)






Convolutional Layer
1 9 6 4
5 4 7 8
5 1 2 9
6 7 6 0
9 1 7 4
5 6 3 0
1 2 5 4
0 8 9 0
7 6 9 1
5 2 0 4
5 8 3 9
0 2 2 1
9 8
7 9
9 7
8 9
7 9
8 9
Max Pooling Layer
WINDOW SIZE: 2X2
STRIDE: 2

Pooling Layer 의 2 가지 특성

1. Positional Invariance

   - 특정 pixel 의 정확한 position 에 less sensitive
   - 여러 번의 pooling 을 거치면 넓은 영역에 걸쳐 같은 효과 발생 (See Wider !)

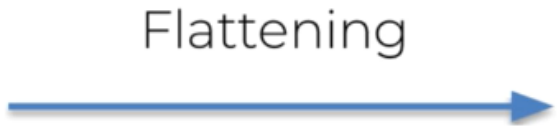
2. Size 축소

   - 계산량을 크게 줄임
   - 과적합 방지

What is Flattening ?

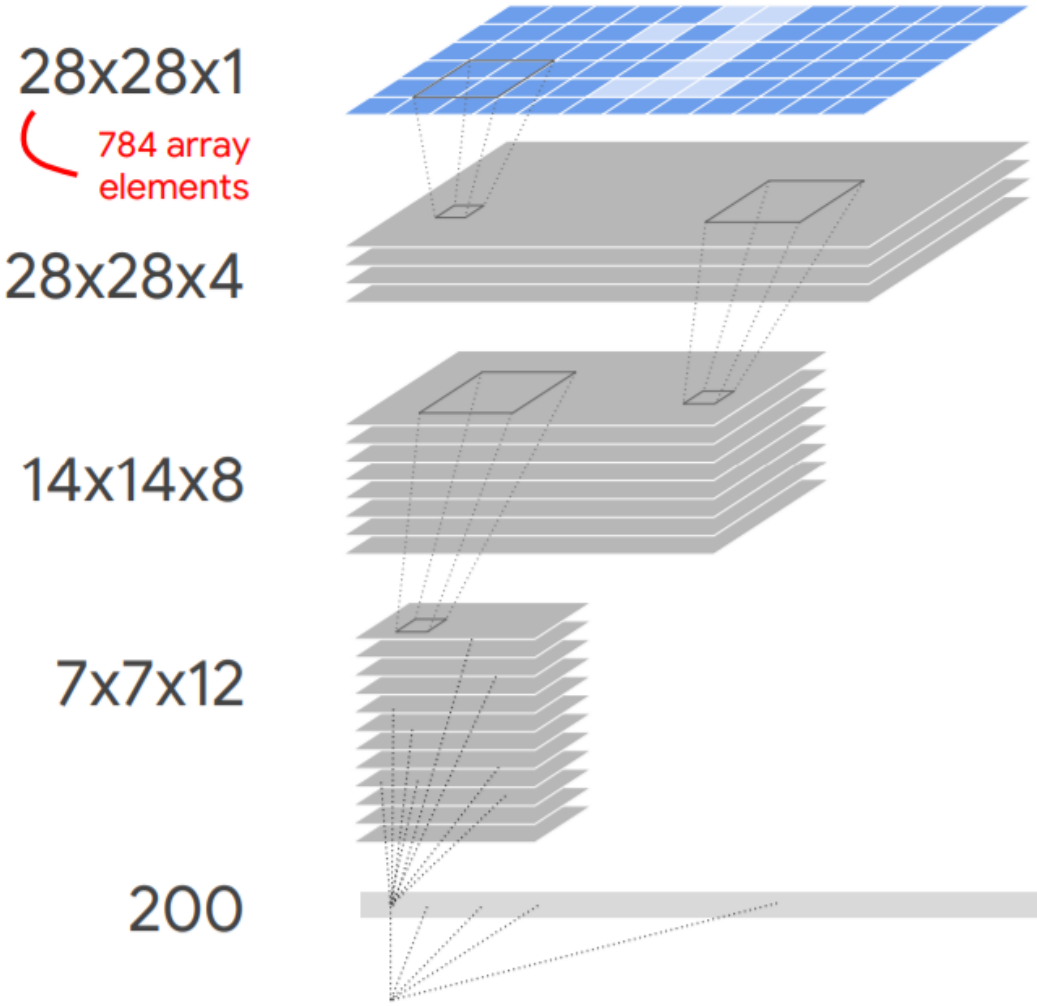
1	1	0
4	2	1
0	2	1

Pooled Feature Map



1
1
0
4
2
1
0
2
1

Image Size 를 축소하면서 Filter 수를 늘인다



convolutional layer, 4 channels deep
W1[5, 5, 1, 4] stride 1

convolutional layer, 8 channels deep
W2[3, 3, 4, 8] stride 2

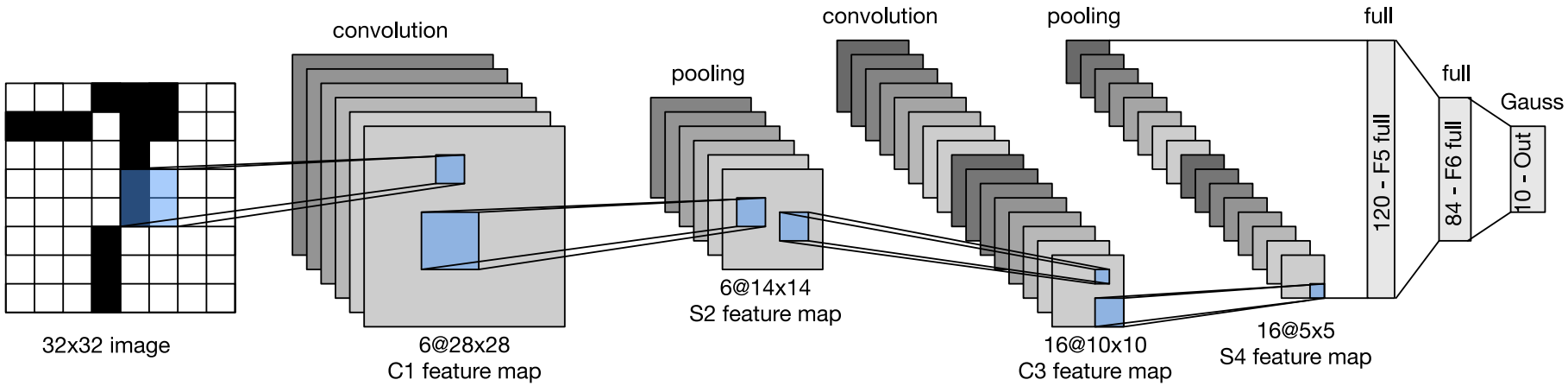
convolutional layer, 12 channels deep
W3[3, 3, 8, 12] stride 2

fully connected layer W4[7x7x12, 200]

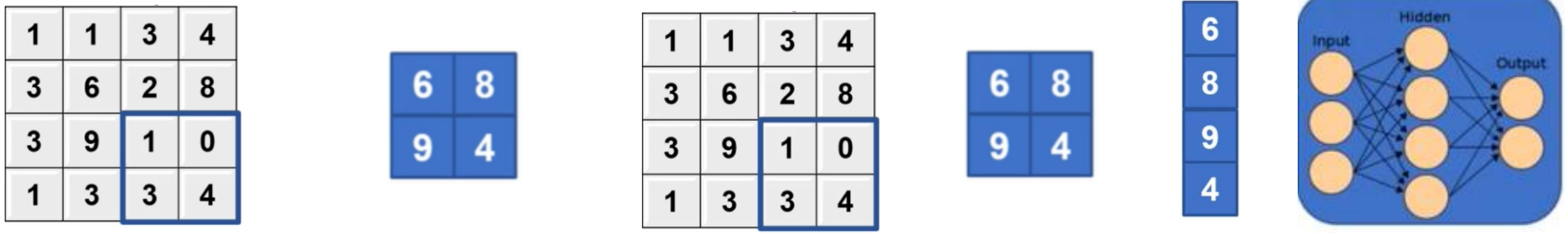
softmax readout layer W5[200, 10]

CNN (Convolutional Neural Network, 합성곱 신경망)

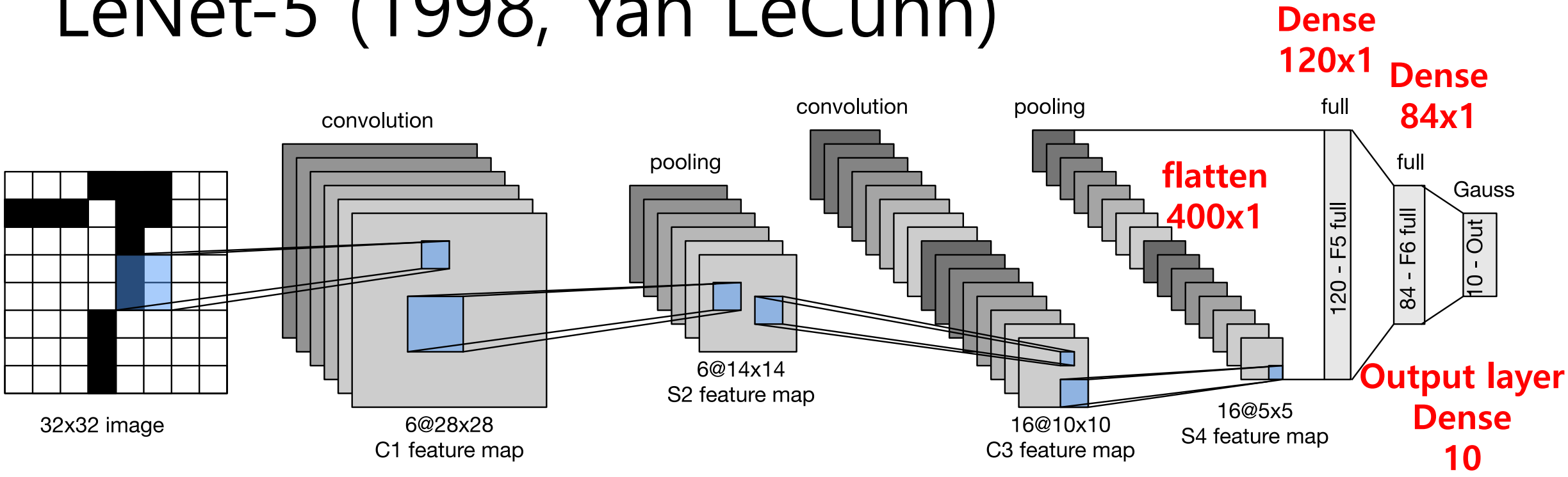
- LeNet : 5 개층 Yan Le Cunn - 1998



입력 **합성곱** **풀링** **합성곱** **풀링** **완전연결**



LeNet-5 (1998, Yan LeCunn)



Dense 120x1

Dense 84x1

Output layer Dense 10

Input layer
32 x 32
Gray Scale Image

1st layer

Convolution
6 filters
(5 x 5, stride 1)
28 x 28 x 6

Average pooling
Filter
(2 x2, stride 2)

14 x 14 x 6

2nd layer

Convolution
16 filters
(5 x 5, stride 1)
10 x 10 x 16

Average pooling
Filter
(2 x2, stride 2)

5 x 5 x 16

실습 : 240. Mnist set 을 이용한 손글씨 인식

1. Keras 를 이용한 Le Net 구축
2. Input reshaping and scaling
3. One-hot encoding
4. Neural Network Model 구성 및 Compile
5. Model Train
6. Performance Evaluation

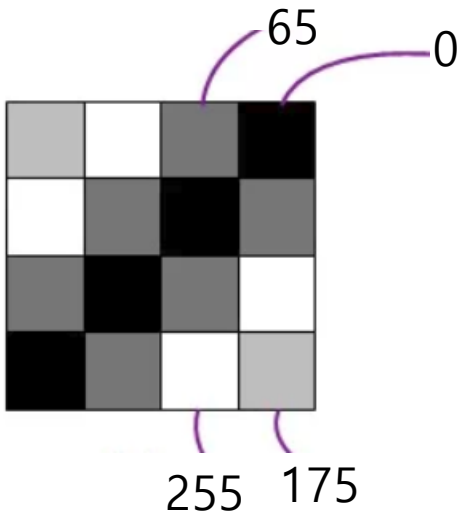
Mnist Dataset 소개

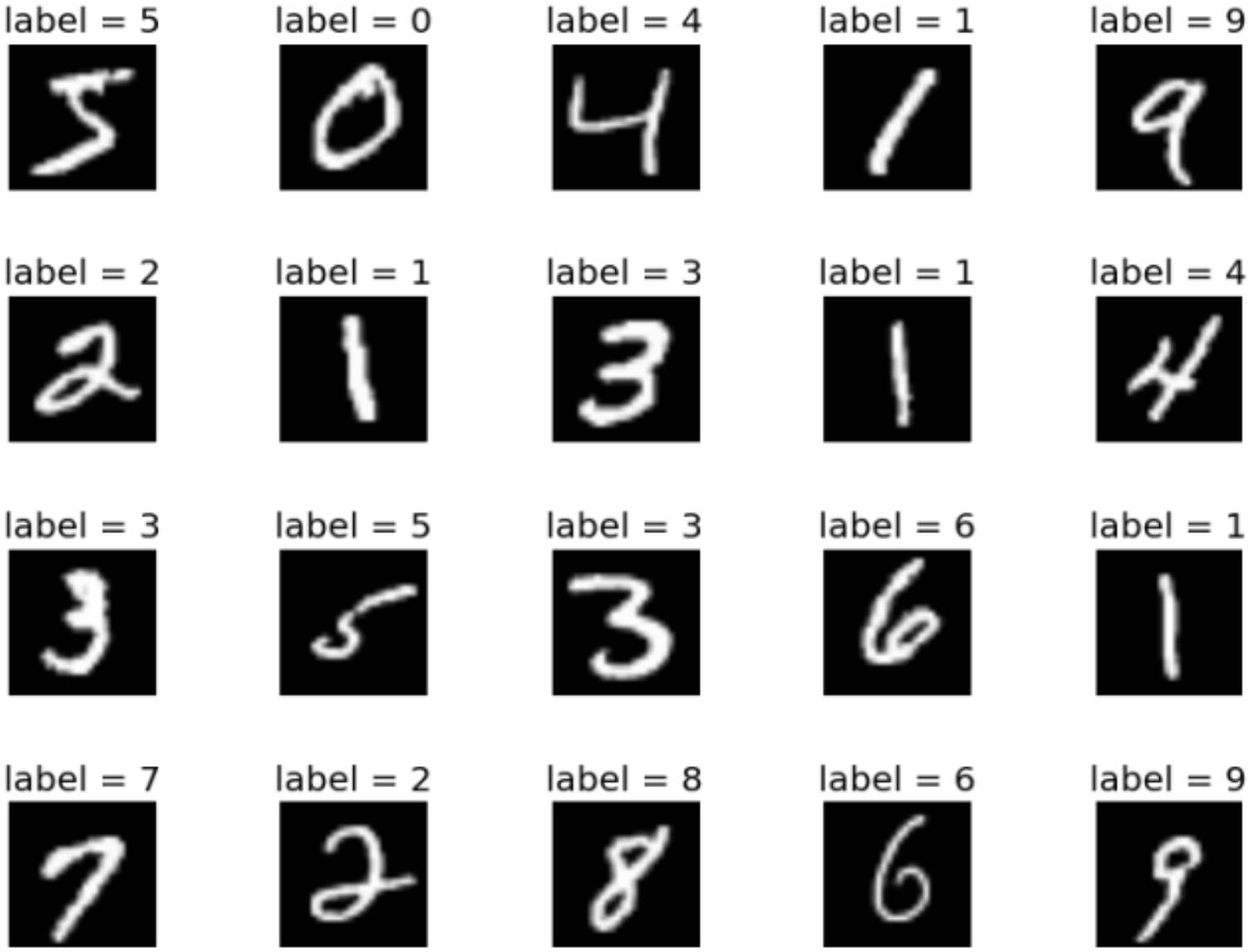
- Mixed National Institute of Standards and Technology

- 0 ~ 9 의 10 개 숫자 손글씨 image dataset
- 28 x 28 pixel 의 gray scale image
- 각 image 마다 0 to 9 의 label 로 쌍을 이루고 있음
- Train set 60,000 / Test set 10,000
- Machine Learning 의 Hello World 에 해당

Pixel 의 구성

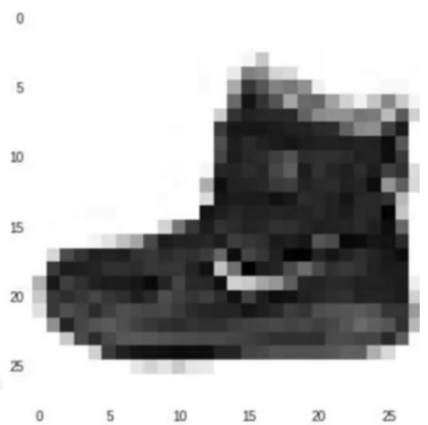
0 – black
255 - white

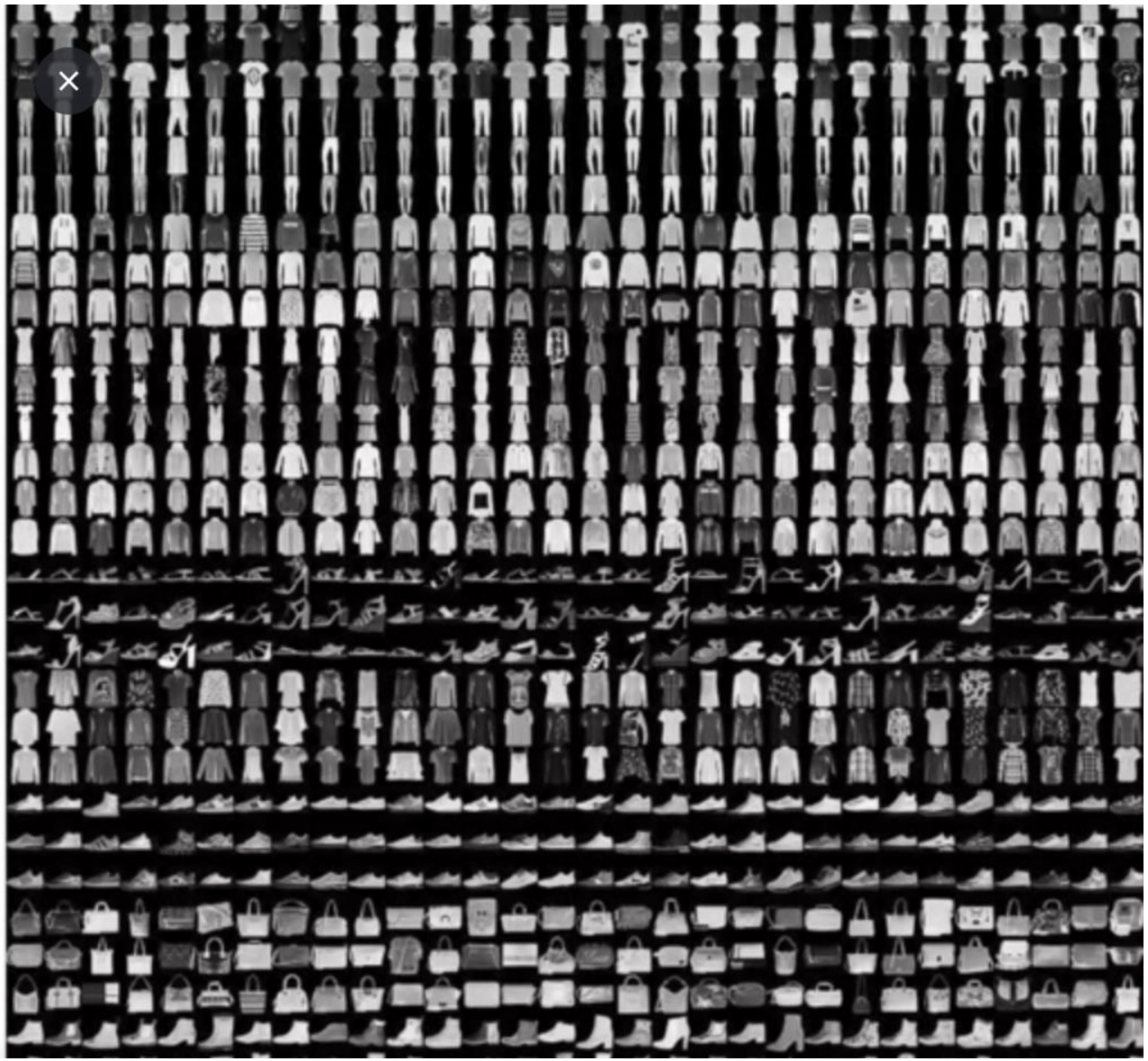




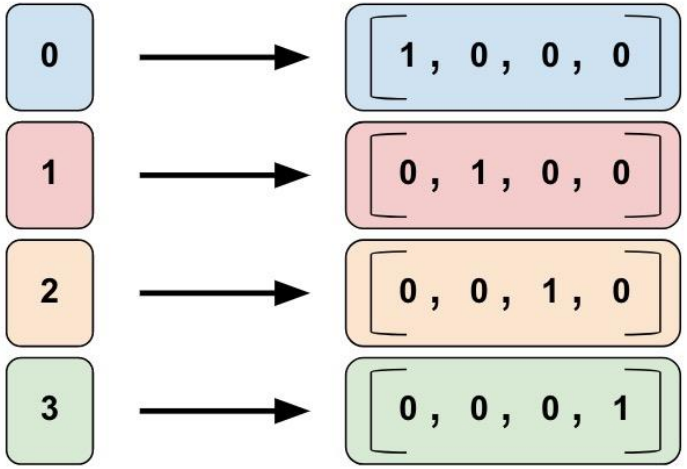
Fashion MNIST

- 70k Images
- 10 Categories
- Images are 28x28
- Can train a neural net!

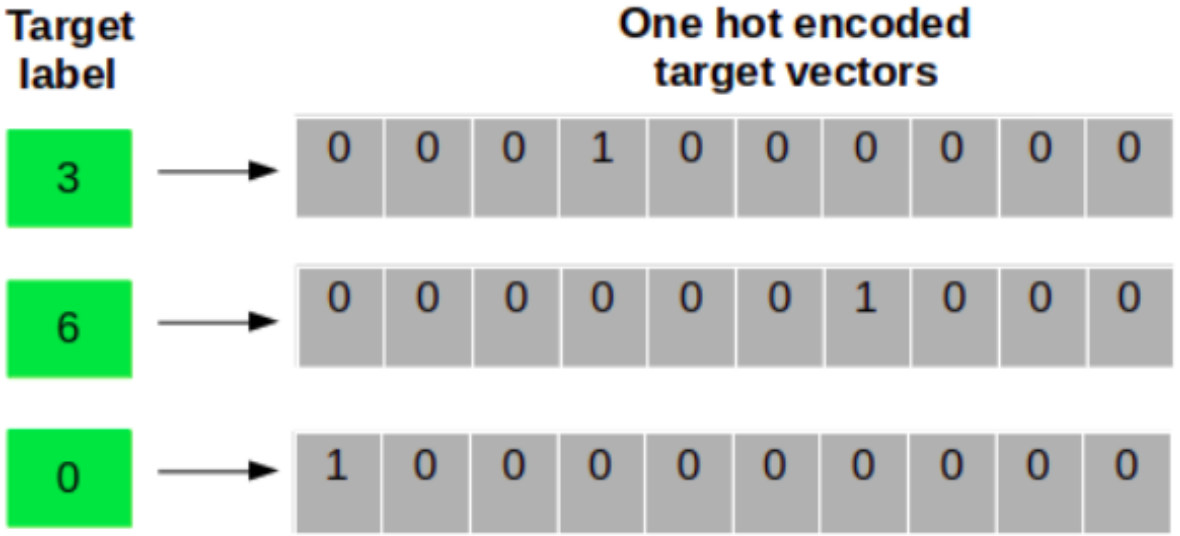




One-Hot encoding



color	color_red	color_blue	color_green
red	1	0	0
green	0	0	1
blue	0	1	0
red	1	0	0

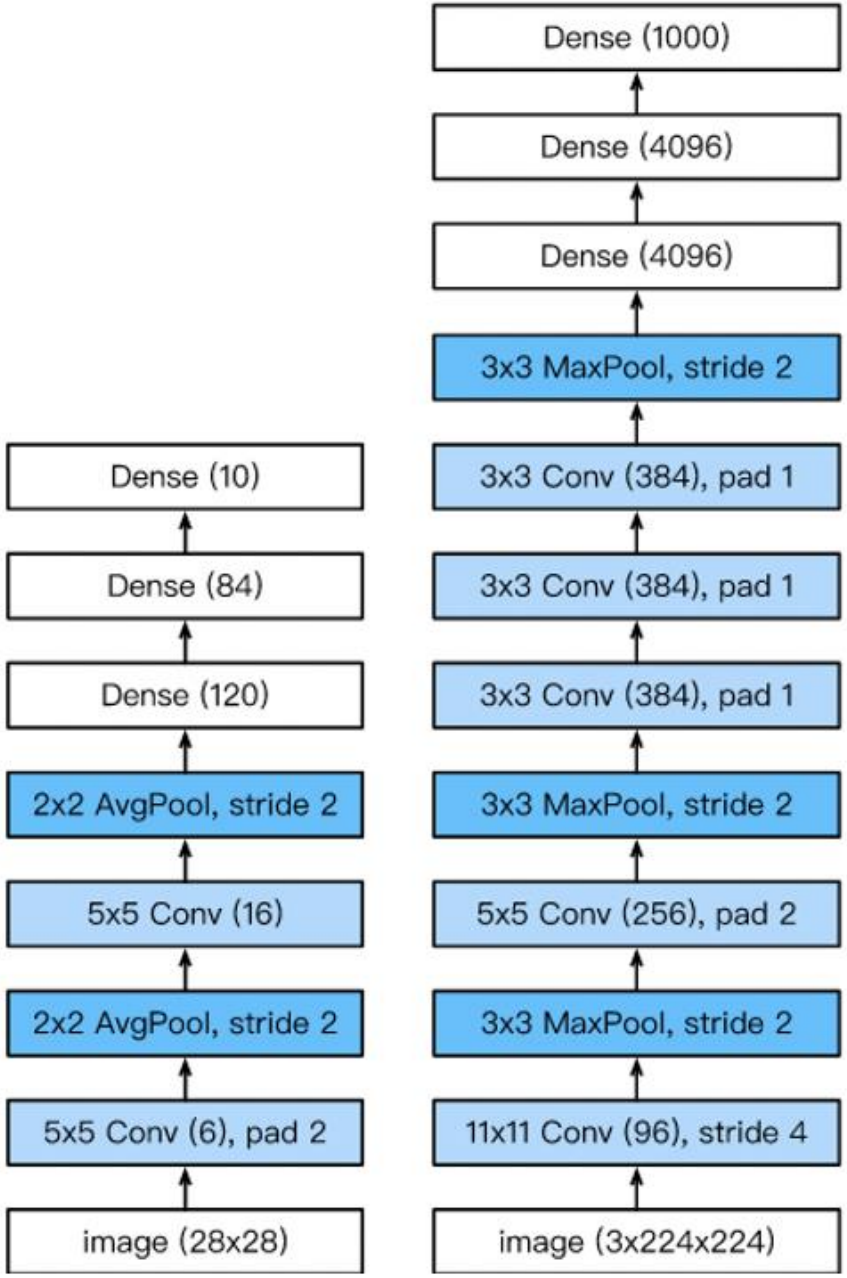


Famous CNN models

- Alex Net – 2012 년 ILSVRC(ImageNet Large Scale Visual Recognition Competition)대회 우승
- GoogleLeNet(Inception Net) – 2014 년 ILSVRC 대회 우승
- ResNet – 2015 년 ILSVRC 대회 우승 (152 개 층)
- MobileNet – mobile device 용 pre-trained model (ImageNet 20,000 개 classes)
- VGG-16 : Keras built-in pre-trained model (2014 년, 16 layers)

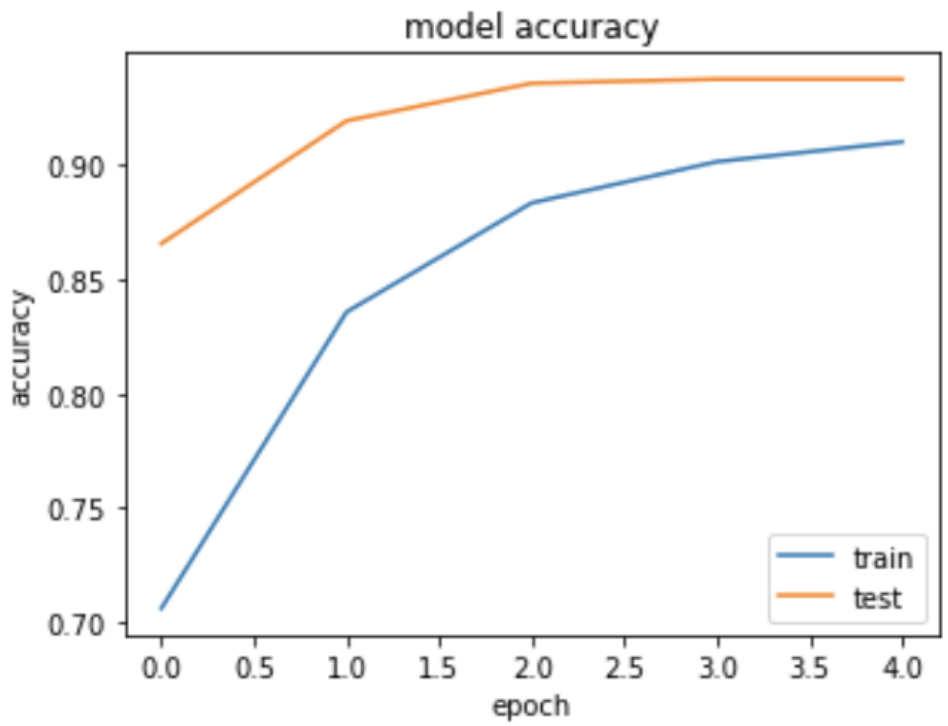
AlexNet vs. LeNet

- 최초의 현대적 CNN
- Larger pooling size
- Larger kernel size and stride
- more channels
- Max Pooling
- Dropout
- ReLU
- Data Augmentation

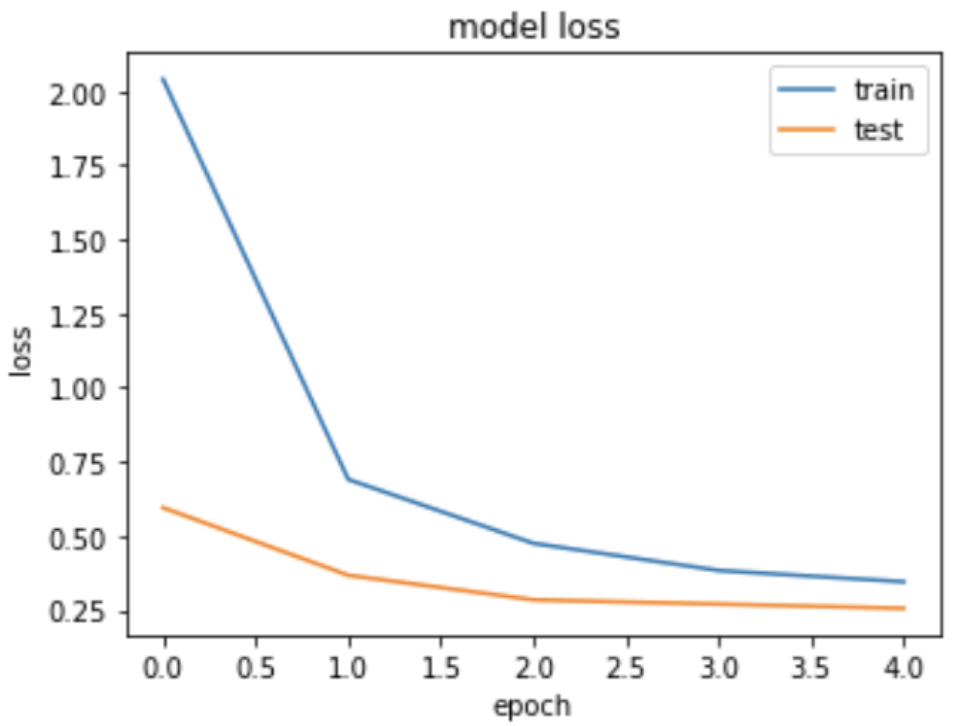


Epoch vs, accuracy 및 loss 변화 시각화

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

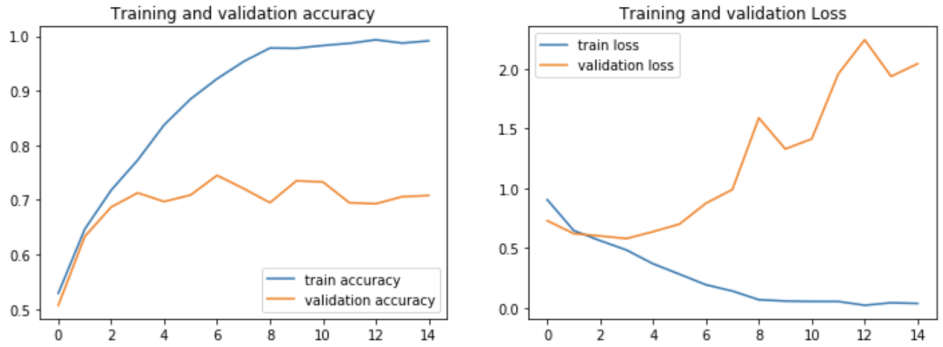



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```



NN 의 Overfitting(과적합) 과 Underfitting(과소적합)

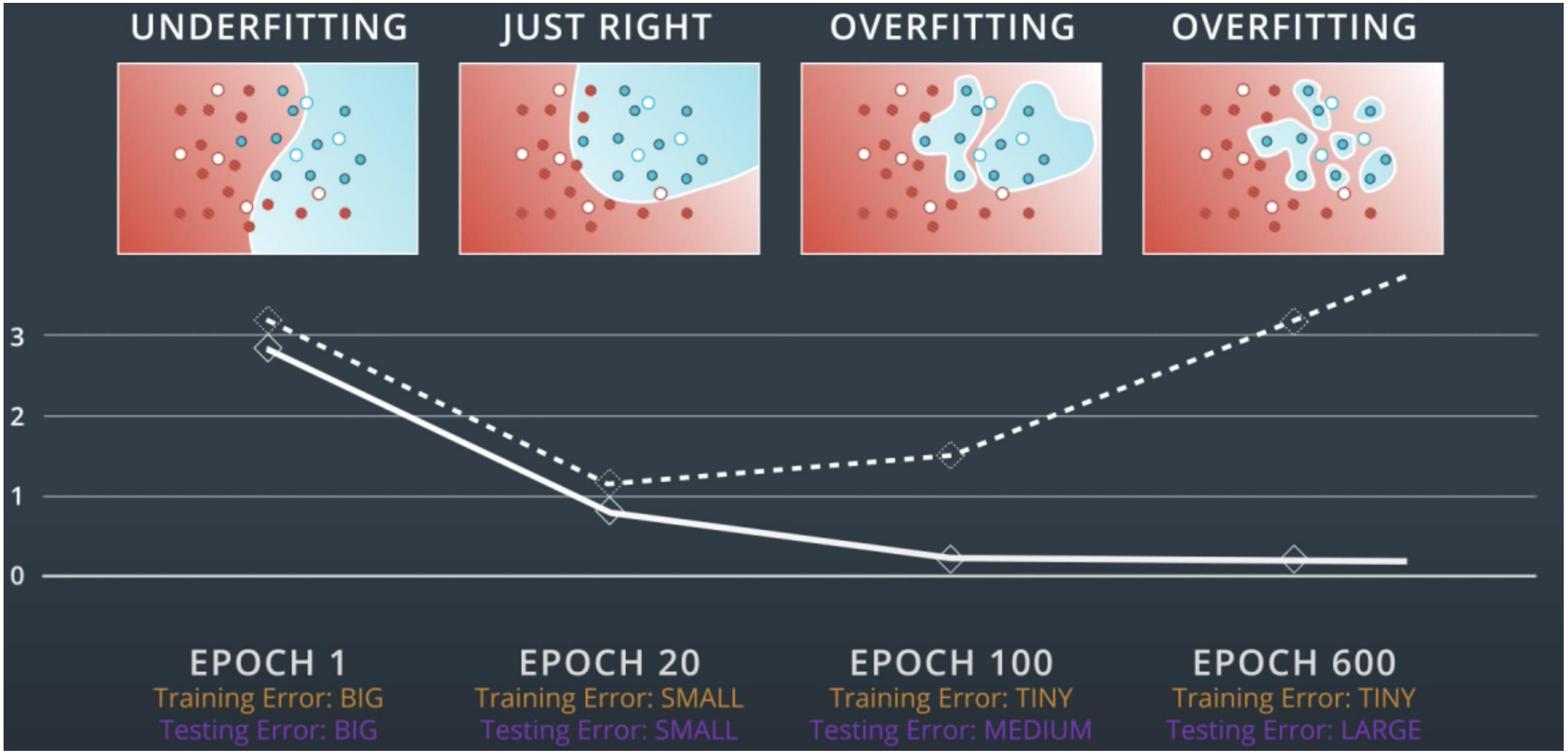
- Training Data 에 비해 Test Data 의 ERROR 율이 높게 나타나는 경우 이를 과적합 (Overfitting) 이라고 한다.
  - Data 특성에 비해 model 이 너무 복잡 (hidden layer 의 neuron 이 너무 많음)



- 반대로 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못하는 경우 과소적합 (Underfitting) 이라고 한다.
  - Model 이 너무 단순 (Hidden layer 의 neuron 이 너무 적음)

Overfitting

White – Test set



실습 : 250. Deeper CNN 을 이용한 CIFAR-10 분류

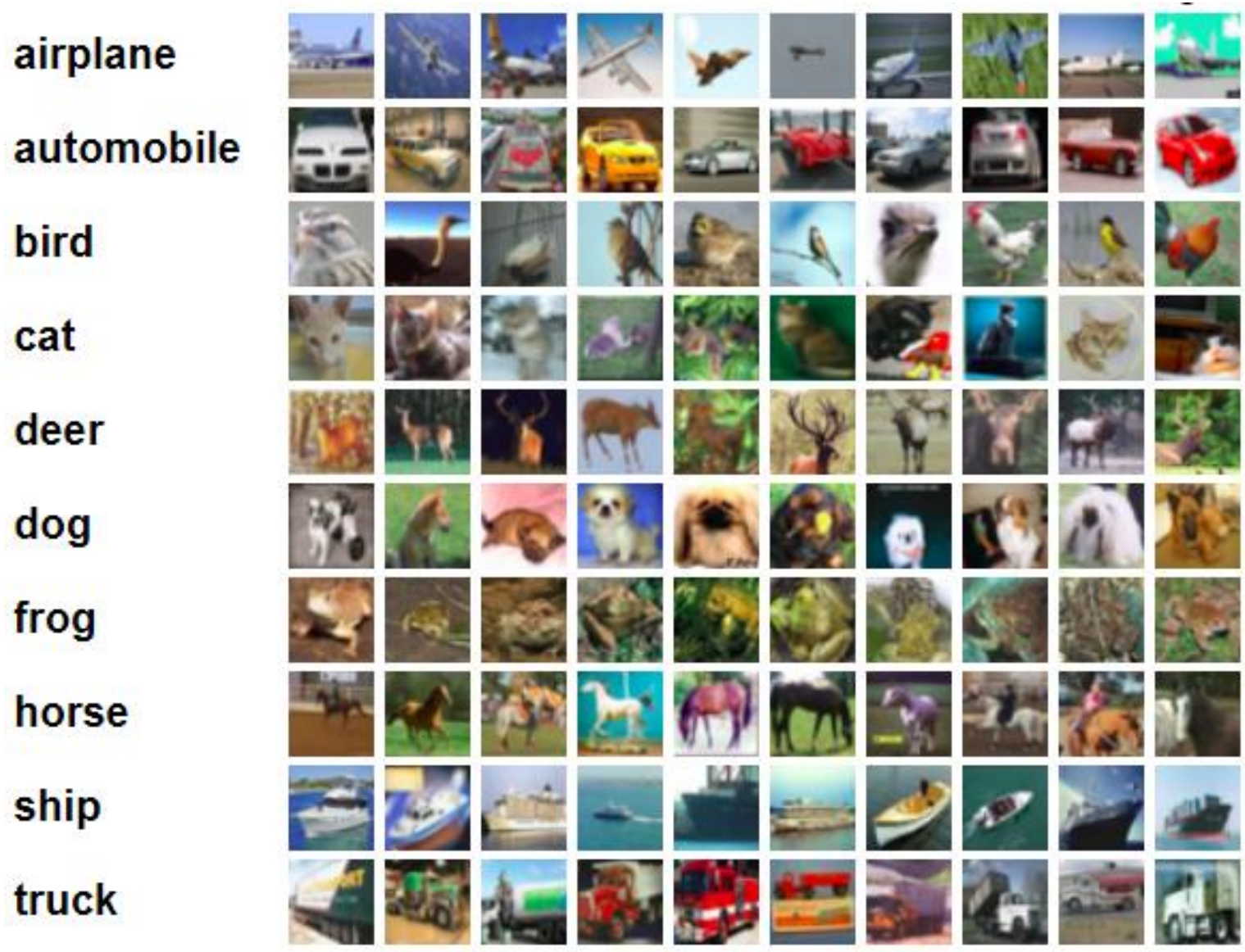
1. CIFAR-10 dataset 은 32x32 color image 를 가진 10개의 class (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)

2. 각 class 별 6,000 개씩 total 60,000 개 image

3. Image 가 blur 하여 난이도 높음
   (최근 성적 : https://en.wikipedia.org/wiki/CIFAR-10)

4. Google Colab GPU 환경 이용

CIFAR-10 image dataset



Neural Network Architecture

Input layer
32 x 32 x 3

Conv2D
16 filters
(3, 3)
Padding=same
Activation : relu

MaxPooling2D
(2, 2)
Padding=same

Dropout : 0.2

Conv2D
32 filters
(3, 3)
Padding=same
Activation : relu

MaxPooling2D
(2, 2)
Padding=same

Dropout : 0.2

Conv2D
64 filters
(3, 3)
Padding=same
Activation : relu

MaxPooling2D
(2, 2)
Padding=same

Dropout : 0.2

Flatten

Dense
256
Activation : relu

Dropout : 0.5

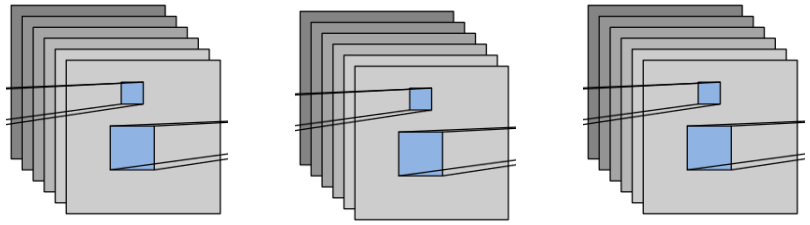
Output

Dense
10
Activation : softmax

Transfer Learning

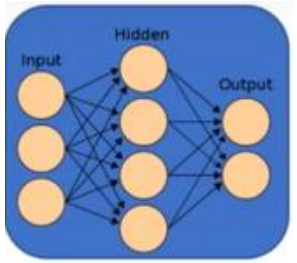
Large Dataset
(ex. ImageNeT)

General Features



Convolutional Layers

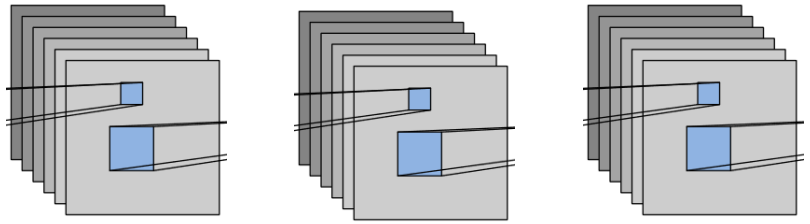
Specific Classification



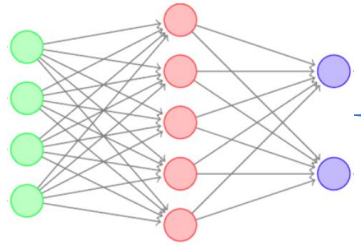
코끼리
사자
하마
호랑이
....

Weight transfer

Small
Dataset

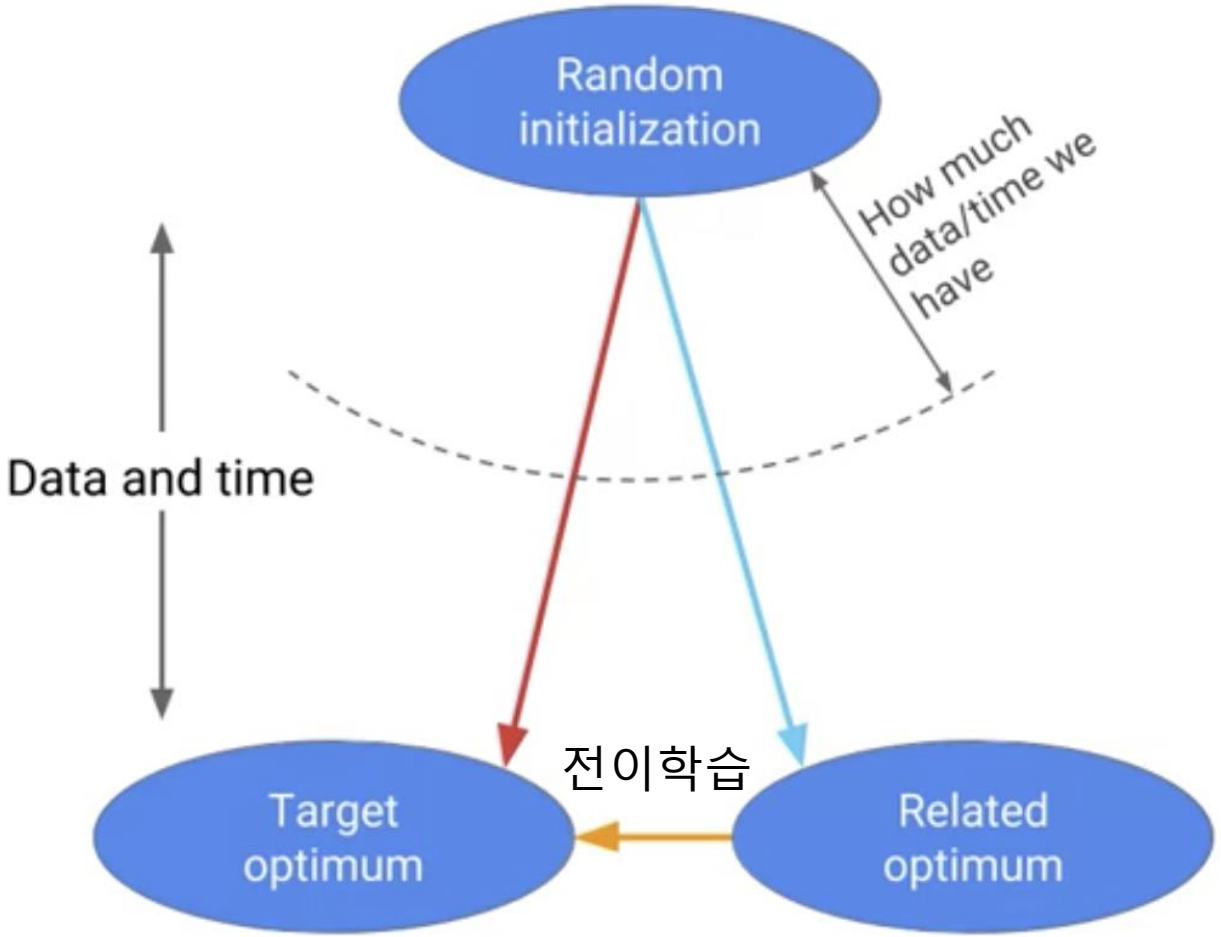


Custom Layers



개
고양이

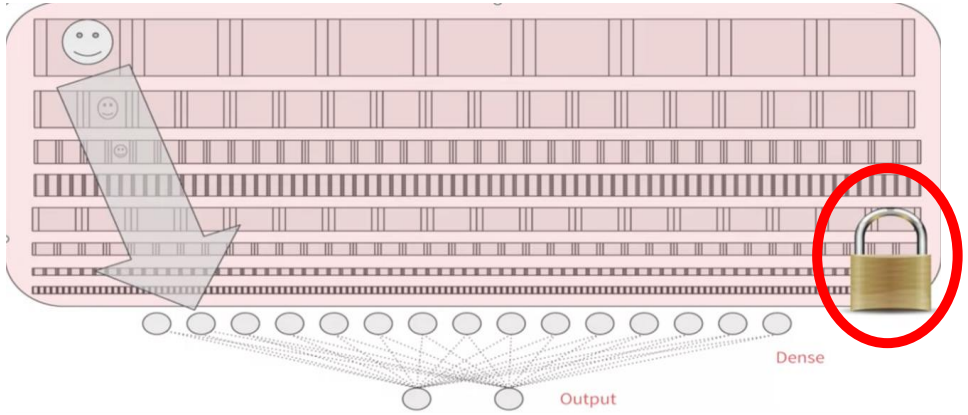
전이 학습 = 지름길


Random initialization
How much data/time we have
Data and time
전이학습
Target optimum
Related optimum

Transfer Learning Strategy

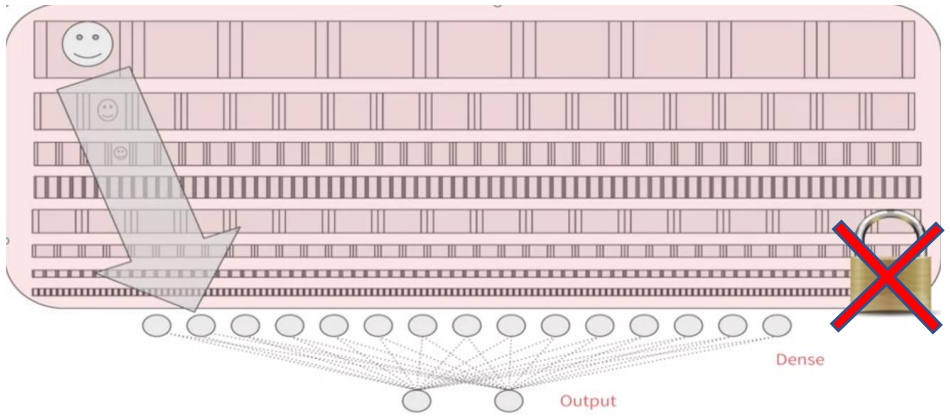
- Strategy 1

CNN layer 는 freeze 하고,
추가한 완전연결층만 새로이 train

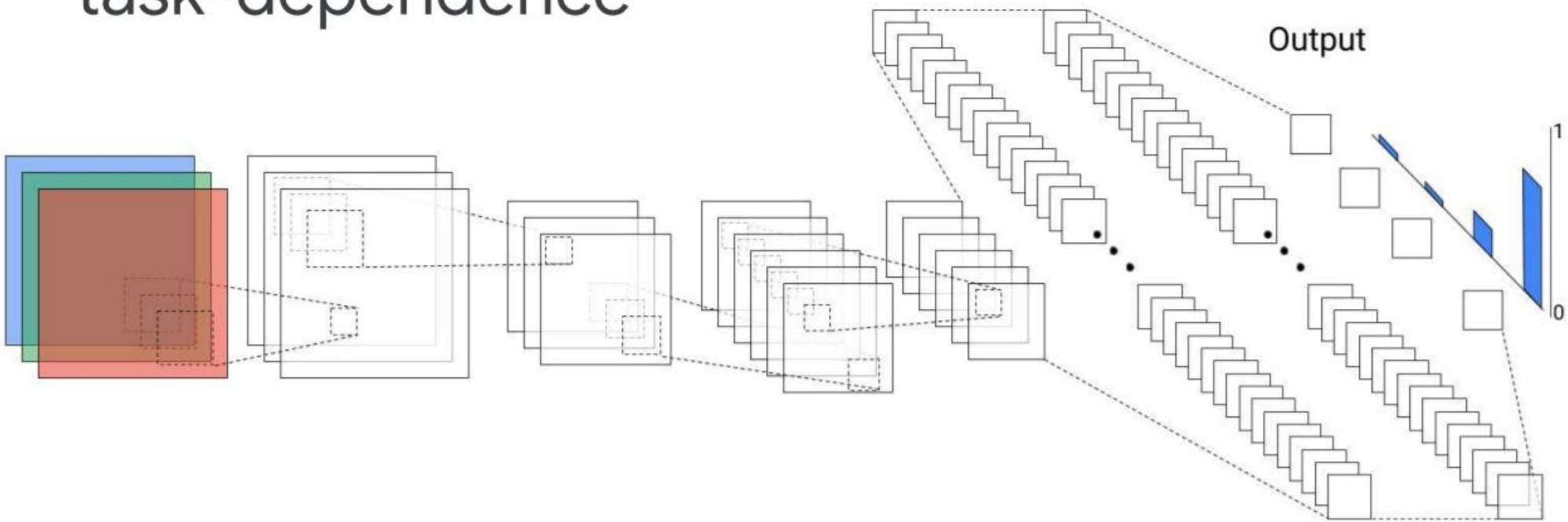


- Strategy 2

전체 Layer 를 매우 작은 learning rate 로 re-train



CNNs and the spectrum of task-dependence

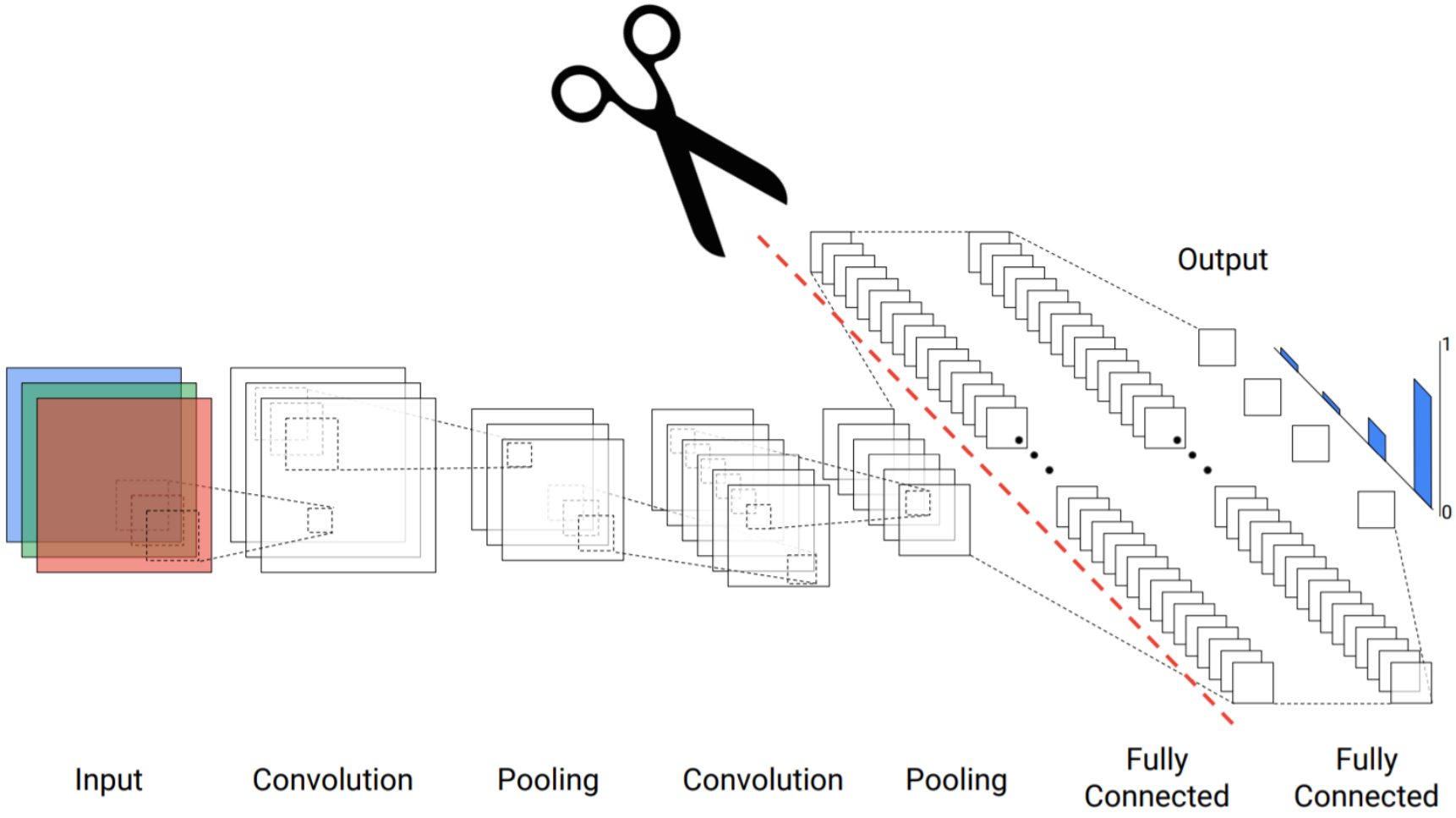


Input | Convolution | Pooling | Convolution | Pooling | Fully Connected | Fully Connected

Most task-general

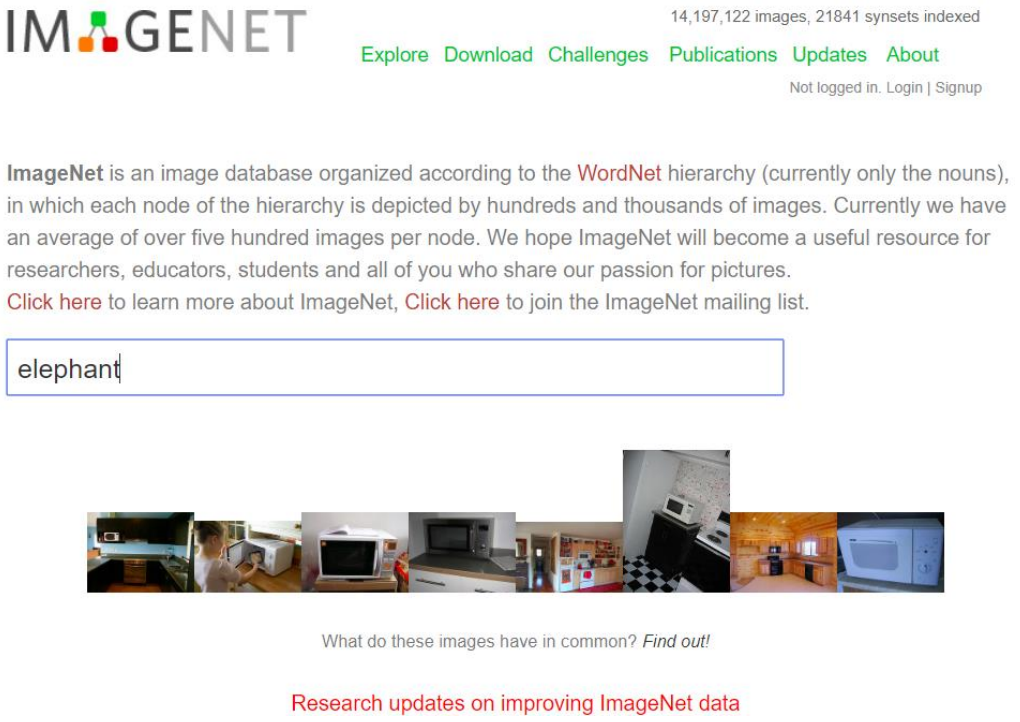
Most task-dependent

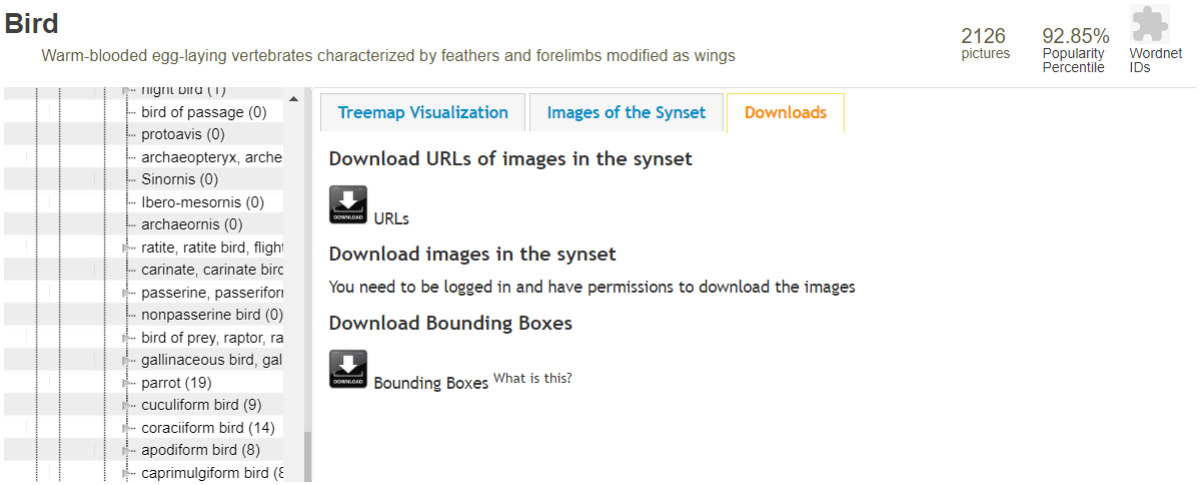
Where to cut ?


Output
1
0
Input
Convolution
Pooling
Convolution
Pooling
Fully Connected
Fully Connected

ImageNet

- Open Source Image Repository http://www.image-net.org/
- 1000 classes over 1.5 MM images





Transfer Learning 고려사항

- **목적에 맞는 dataset 선택**

  ex) Cat & Dog 구분 → ImageNet 에 포함

  Cancer cell 구분 → ImageNet 에 없음

- **보유 Data 의 Volume 고려**

  1. 모든 weight 새로이 training (Large Data 보유)
  2. Weight 의 일부만 training
  3. 마지막 layer 만 Fine-tuning (Small Data 보유)

Repositories of Pre-trained Models

- Keras Applications : https://keras.io/api/applications/
- Caffe Model Zoo : http://caffe.berkeleyvision.org/model_zoo.html
- Tensorflow Hub : https://www.tensorflow.org/hub
- Pytorch : https://github.com/Cadene/pretrained-models.pytorch

실습 : 260. TensorFlow Hub 을 이용한 전이 학습

1. pre-trained model (MobileNet_V2) 을 feature extractor로 이용하여 꽃 image 에 특화된 image 분류 model build

2. Model 구성

3. 전이 학습 MODEL 을 Flower 분류에 적합한 model 로 Retrain

```
model = tf.keras.Sequential([
  MobileNet_feature_extractor_layer,
  tf.keras.layers.Dense(flowers_data.num_classes, activation='softmax')
])
```

TensorFlow Hub

TensorFlow Hub, 학습된 머신러닝 모델의 저장소

TensorFlow Hub는 어디서나 미세 조정 및 배포 가능한 학습된 머신러닝 모델의 저장소입니다. 몇 줄의 코드만으로 BERT 및 Faster R-CNN과 같은 학습된 모델을 재사용할 수 있습니다.

튜토리얼 보기
TensorFlow Hub 사용 방법과 작동 방법을 알아보세요.

튜토리얼 보기
튜토리얼에서는 TensorFlow Hub를 사용하는 엔드 투 엔드 예를 보여줍니다.

모델 보기
사용 사례에 맞는 학습된 TF, TFLite 및 TF.js 모델을 찾아보세요.

```
!pip install --upgrade tensorflow_hub

import tensorflow_hub as hub

model = hub.KerasLayer("https://tfhub.dev/google/nnlm-en-dim128/2")
embeddings = model(["The rain in Spain.", "falls",
                    "mainly", "In the plain!"])

print(embeddings.shape)  #(4,128)
```

MobileNet 소개

- ImageNet 의 수백만장 image 를 이용하여 trained
- 1000 개의 class 로 image 구분

class name	probability	imagenet class id
Egyptian cat	0.478	285
tabby, tabby cat	0.300	281
tiger cat	0.167	282
remote control, remote	0.016	761
Siamese cat, Siamese	0.008	284

Deep Learning

Sequence Model

What is sequence data ?

- Speech recognition : 파동의 연속 → 단어의 연속으로 변환
- Music generation : 연속된 음표 출력
- Sentiment classification : Text → 평점, 부정/긍정 판단
- DNA 분석 : 염기서열 → 질병유무, 단백질 종류 등
- 자동 번역 : 한국어 → 영어
- Video activity recognition : 연속된 장면 → 행동 판단
- Financial Data : 시계열자료 → 주가, 환율 예측 등

Problem of Standard Neural Network

- 입력의 길이와 출력의 길이가 고정되어 있음

  → 따라서 standard NN 에서는 maximum input length 정하고
  초과하면 truncate, 모자라면 padding 을 해 주어야 한다.

- 입력 데이터의 순서를 무시 (모든 observation 은 독립적임)

RNN (Recurrent Neural Network)

RNN (Recurrent Neural Network)

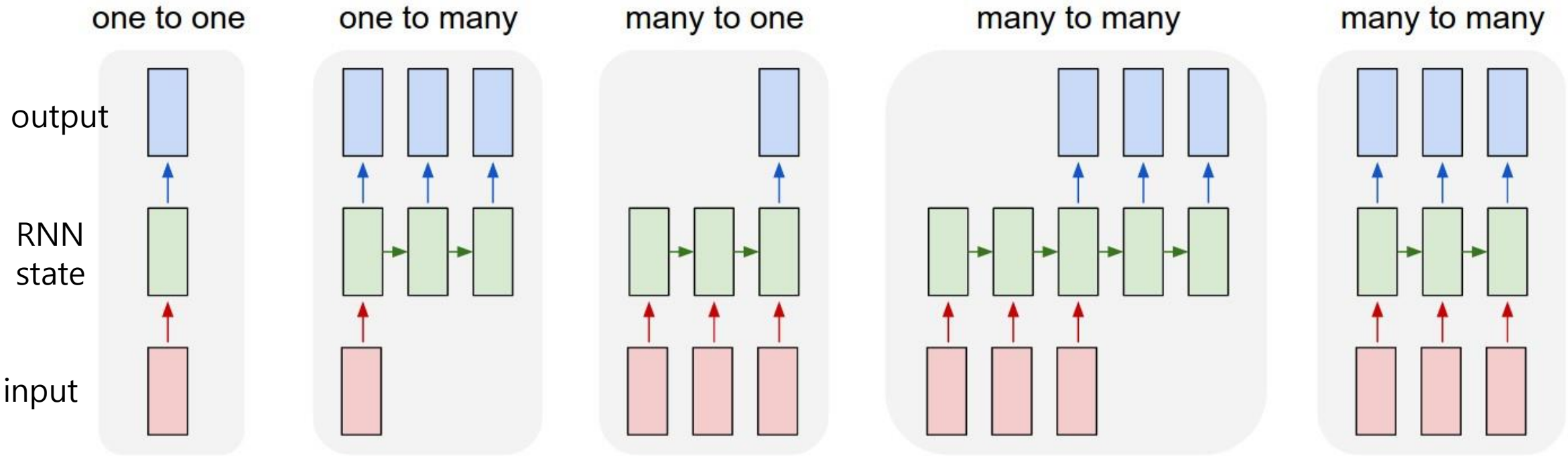
- 시퀀스 데이터에 특화

- '기억' 능력을 갖고 있음
  * 네트워크의 기억 - 지금까지의 입력 데이터를 요약한 정보
  (새로운 입력이 들어올때 마다 네트워크는 자신의 기억을 조금씩 수정)

- 입력을 모두 처리하고 난 후 네트워크에게 남겨진 기억은 시퀀스 전체를 요약하는 정보
  (사람의 시퀀스 정보 처리 방식과 비슷, 기억을 바탕으로 새로운 단어 이해)

- 이 과정은 새로운 단어마다 계속해서 반복 → Recurrent (순환적)

Different Types of RNN



예) 이미지 분류

- 이미지로 부터 문장 생성 (Image Captioning)
- 작곡, 작시

- 감성 분석 (sentiment Analysis → positive/negative)
- Spam detection

- 기계 번역 (영어 → 한국어 문장)
- Chatbot
- Question Answering

- video 각 frame 에 label 생성
- 품사 tagging
- 개체명 인식
- Character 단위 문장 생성 등

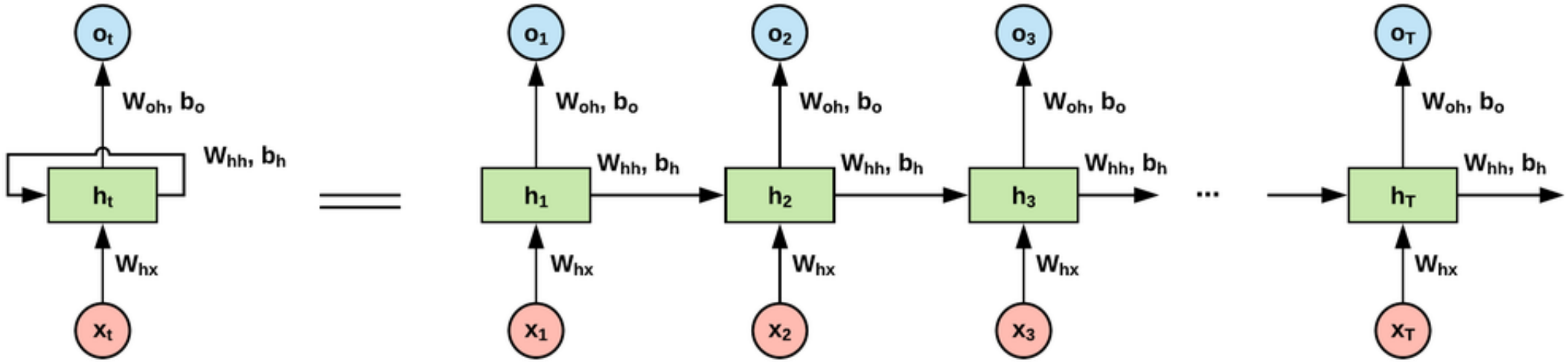
RNN (Unfold 표시)

Internal State :

$$\boldsymbol{h_t = \tanh(W_h h_{t-1} + W_x x_t)}$$

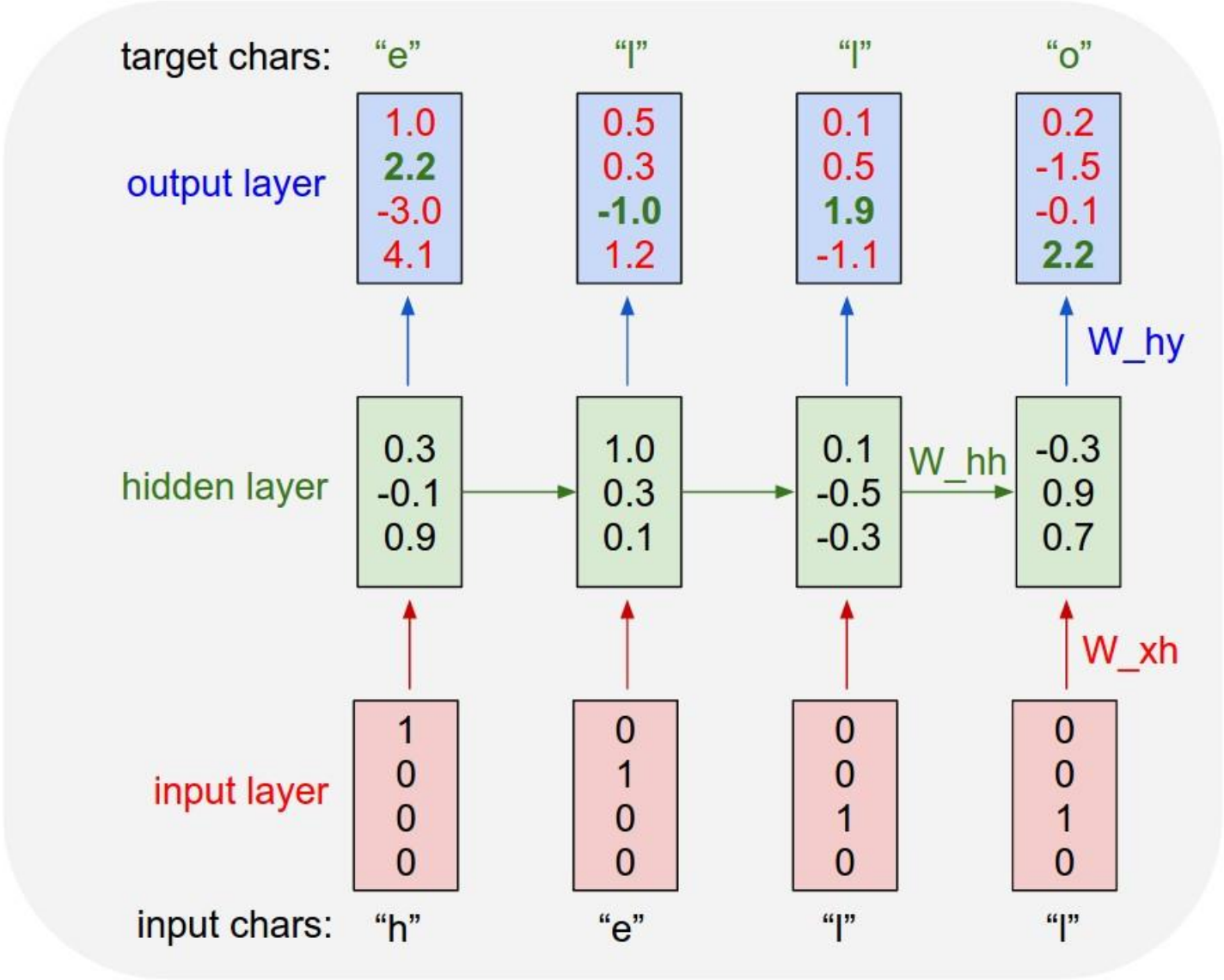
Output :

$$\boldsymbol{O_t = softmax(W_o h_t)}$$



- RNN 을 순서대로 펼쳐 놓으면 weight 를 공유하는 매우 deep 한 neural network 이 된다.
- BPTT (Backpropagation Through Time) 으로 parameter 학습

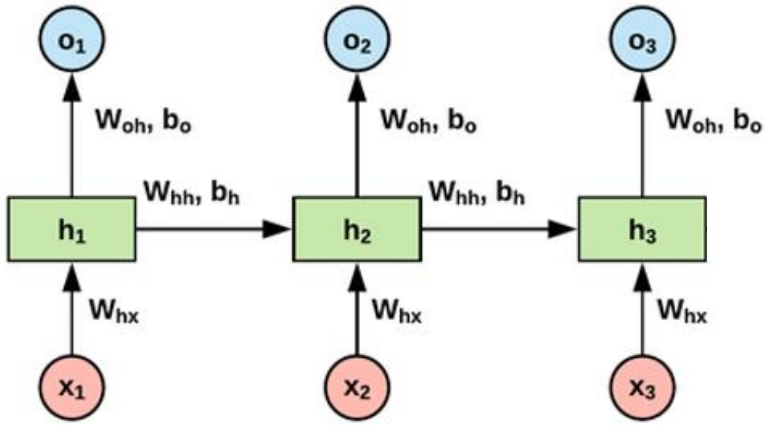
How RNN is trained ?



- "h", "e", "l", "o" 4 개 문자만 있다고 가정
- 각 문자를 One-hot encoding
- "h" 를 시작 문자로 주면 "hello" 가 출력 되도록 훈련
- 각 time step 의 target character 는 "hello" 내의 next character
- Gradient descent 와 backpropagation 을 통해 output layer 의 target score 가 증가되도록 함 (green color)
- "l" 다음의 character 는 현재의 "l" 만으로 판단할 수 없음 (history 필요)

LSTM (Long Short-Term Memory)

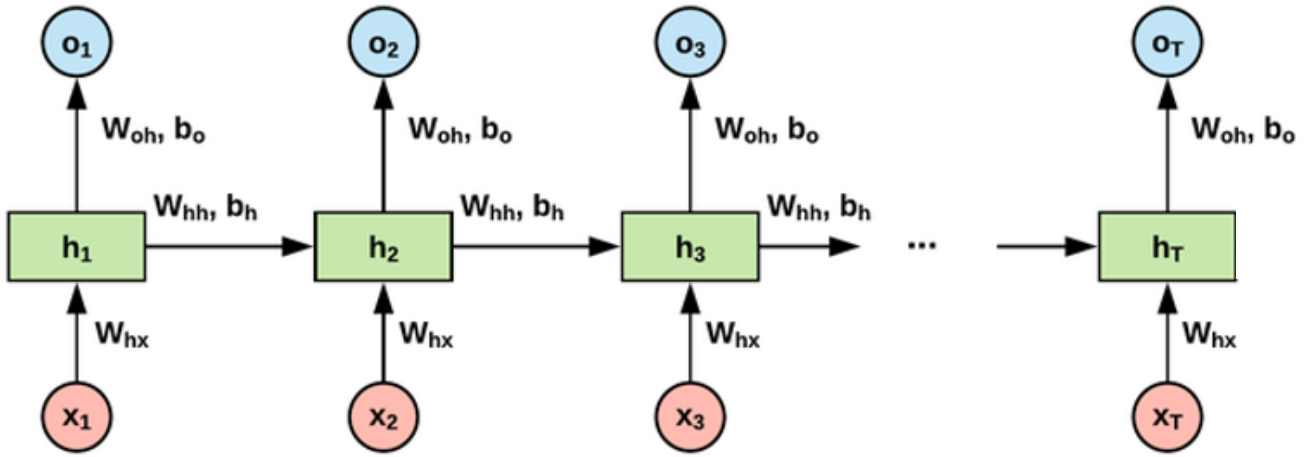
green



Tree color is

Simple RNN 으로 기억 가능

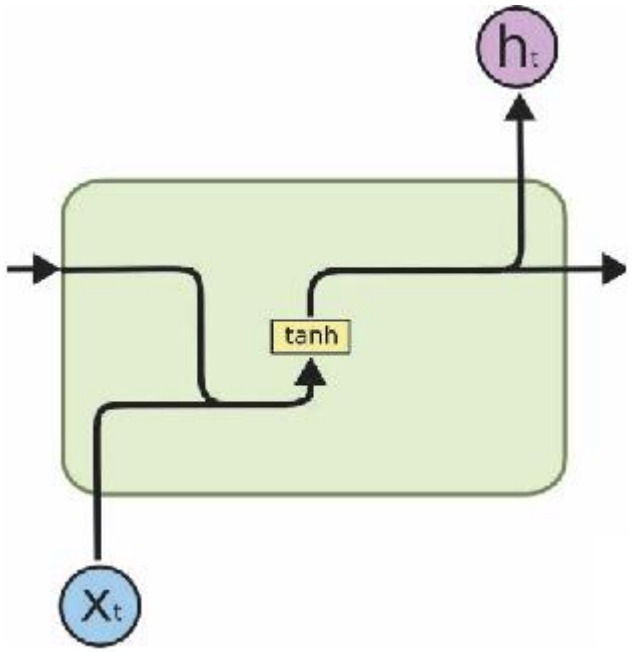
English



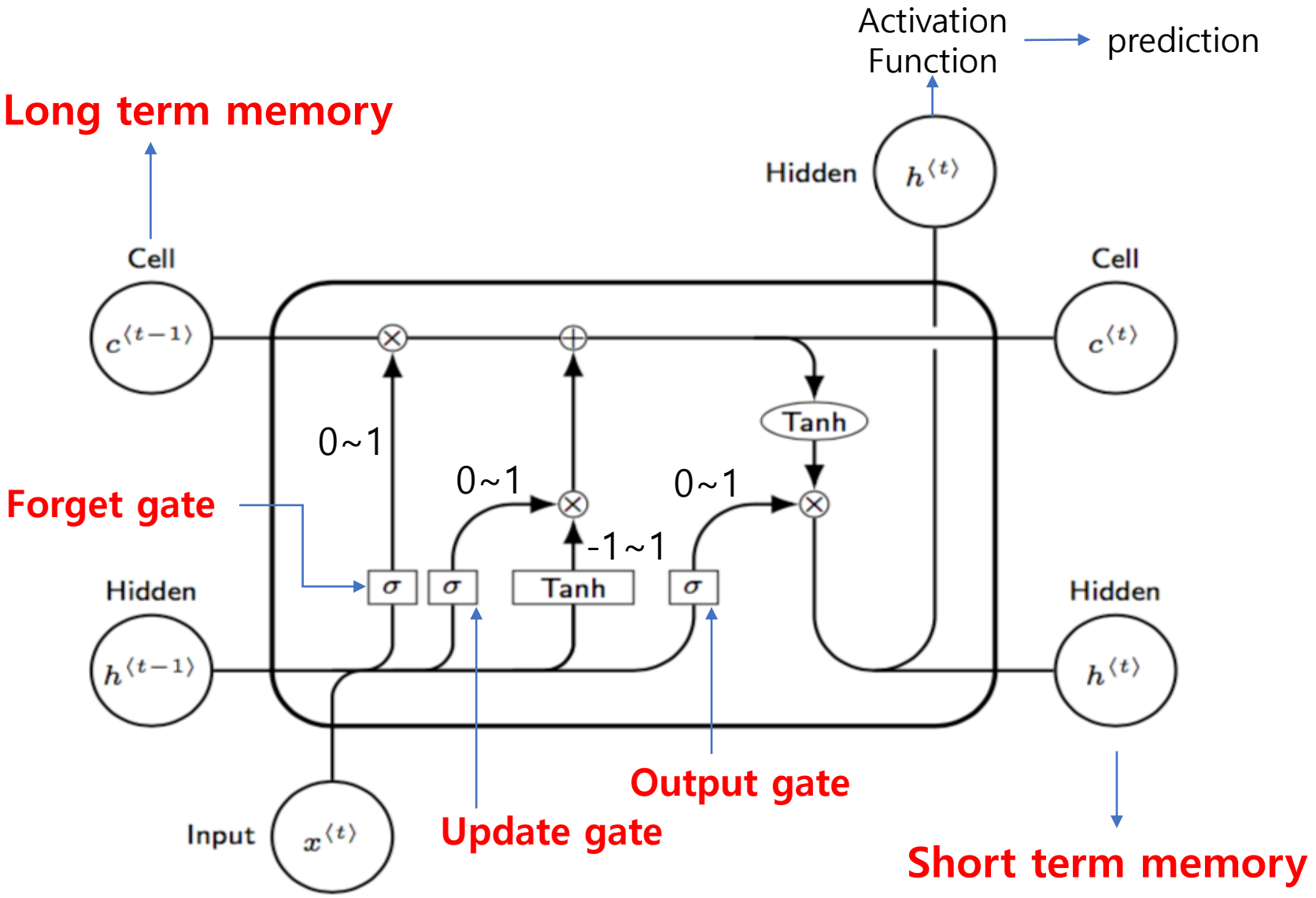
He was born in **England** and moved to Asia and his mother language is

Long-Term Memory 필요

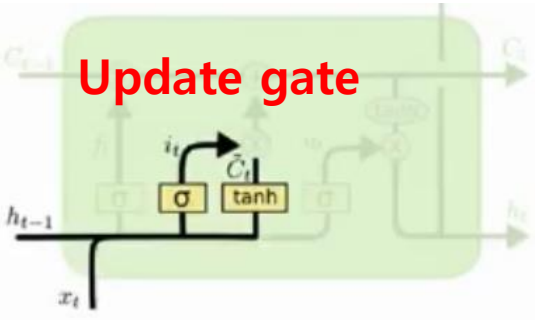
SimpleRNN



LSTM (Long Short-Term Memory)

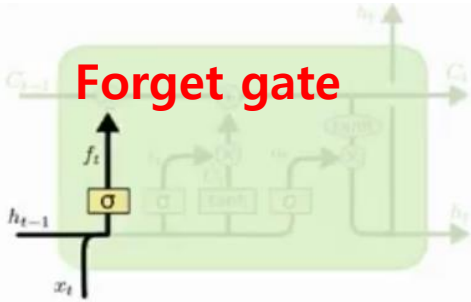


LSTM 내부 구조



- **Input** – 이전 step 의 hidden + new data

  $\tilde{C}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$ ➔ 새로운 cell status 후보

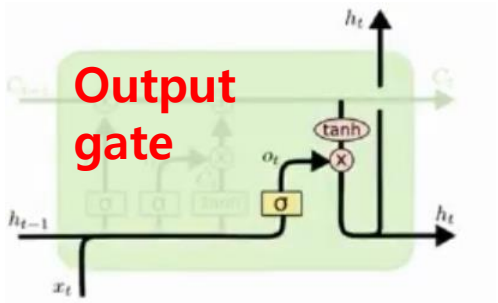


- **Update gate** – input 을 어느정도 받아들일지 결정 (0-무시, 1-전체)

  $\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$

- **Forget gate** – 이전 cell state 를 어느정도 기억할지 결정 (0-forget, 1-전체 기억)

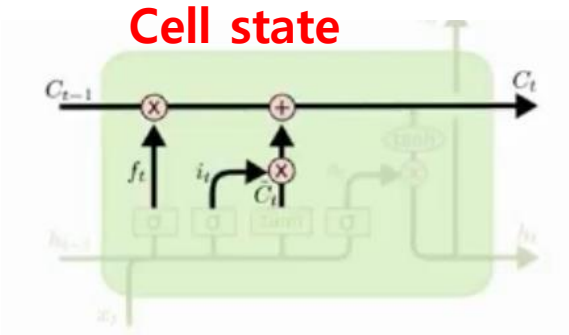
  $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$



- **Output gate** – input 을 어느정도 다음 step 으로 보낼지 결정

  $\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

- $C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + \Gamma_f * C^{<t-1>}$



- $a^{<t>} = \Gamma_o * \tanh C^{<t>}$

실습: 270. Simple LSTM

- input 은 0 ~ 99 까지의 연속된 숫자
- target 은 (1 ~ 101) * 2

- 연속된 5 개의 숫자를 보고 다음 숫자를 알아맞추도록 LSTM 을 이용한 model 작성
  - ex) [[5], [6], [7], [8], [9]] → [20]
    [[35], [36], [37], [38], [39]]. → [80]

실습: 280. Multistep Time Series

- 가정의 Energy 사용량 예측
- EDA
- Downsampling for daily total
- LSTM Multivariate Multi-step Input and Single-step output model 작성

자연어 처리(NLP)의 RNN 적용

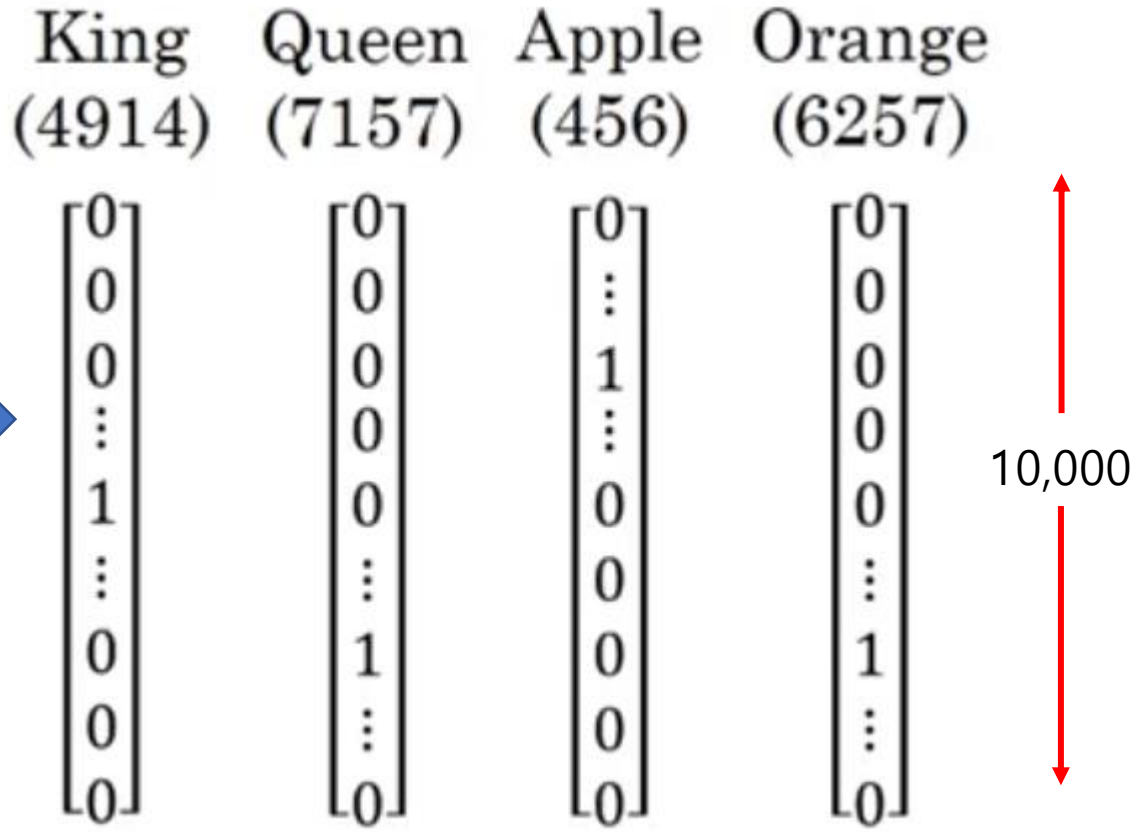
단어의 Vector 표현

- One-Hot-Encoding

Vocaburary 사전

a -1
aaron – 2
.
apple – 456
.
king – 4914
.
orange – 6257
.
queen – 7157
.
zebra – 9,999
<UNK> - 10,000

One-Hot encoding



Word Embedding

- 단순 One-Hot-Encoding 의 문제점 – 단어 간의 유사도가 표시되지 않음
- 수자화된 단어의 나열 → sentiment 추출
- 연관성 있는 단어들을 군집화하여 multi-dimension 공간에 vector 로 표시
- 예를 들어, 호감(positive), 비호감(negative) 두가지 label 에 따라 관련 단어들을 두개의 category 로 군집화

  ex) IMDB : boring, bad, unfunny → negative
  funny, good, interesting → positive

Word Embedding (Feature 화 표시)

dimension	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
성별	-1	1	-0.95	0.97	0.00	0.01
귀족	0.01	0.02	0.93	0.95	-0.01	0.00
나이	0.03	0.02	0.7	0.69	0.03	-0.02
음식	0.04	0.01	0.02	0.01	0.95	0.97

Man (5931) 의 4 dimension vector 표시

King (4914) 의 4 dimension vector 표시

Embedding matrix (example)

학습된 featrues

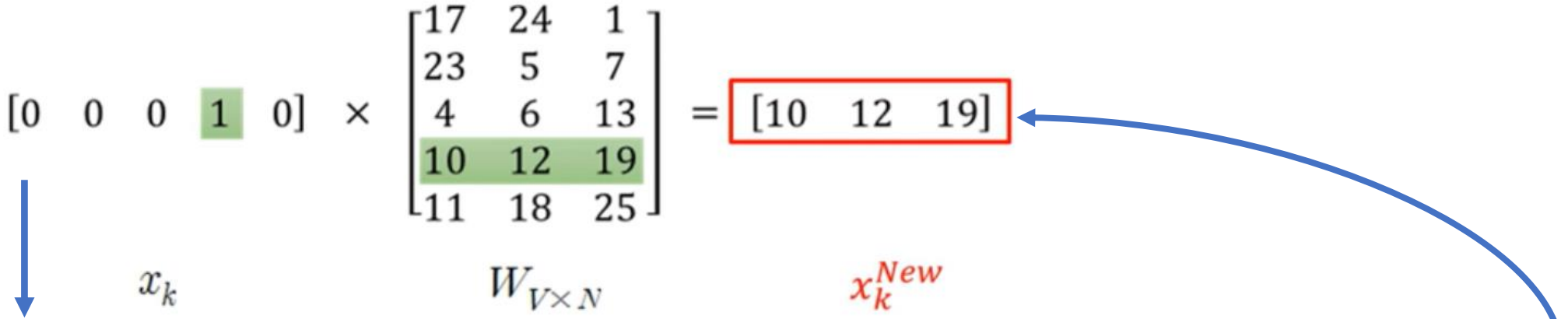
words

	0	1	2	3	4	5	6	7	8	9	...	290	291	292	
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	...	-0.283050	0.270240	-0.654800	0.10
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	...	0.464470	0.481400	-0.829200	0.35
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	...	-0.015404	0.392890	-0.034826	-0.72
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	...	-0.285540	0.104670	0.126310	0.12
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	...	-0.107030	-0.279480	-0.186200	-0.54
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	...	-0.232860	-0.139740	-0.681080	-0.37
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	...	0.048760	0.351680	-0.786260	-0.36
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	...	-0.667050	0.279110	0.500970	-0.27
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	...	0.304240	0.413440	-0.540730	-0.03
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	...	0.647710	0.373820	0.019931	-0.03
toast	0.130740	-0.193730	0.253270	0.090102	-0.272580	-0.030571	0.096945	-0.115060	0.484000	0.848380	...	0.142080	0.481910	0.045167	0.05
today	-0.156570	0.594890	-0.031445	-0.077586	0.278630	-0.509210	-0.066350	-0.081890	-0.047986	2.803600	...	-0.326580	-0.413380	0.367910	-0.26
blue	0.129450	0.036518	0.032298	-0.060034	0.399840	-0.103020	-0.507880	0.076630	-0.422920	0.815730	...	-0.501280	0.169010	0.548250	-0.31
green	-0.072368	0.233200	0.137260	-0.156630	0.248440	0.349870	-0.241700	-0.091426	-0.530150	1.341300	...	-0.405170	0.243570	0.437300	-0.46
kings	0.259230	-0.854690	0.360010	-0.642000	0.568530	-0.321420	0.173250	0.133030	-0.089720	1.528600	...	-0.470090	0.063743	-0.545210	-0.19
dog	-0.057120	0.052685	0.003026	-0.048517	0.007043	0.041856	-0.024704	-0.039783	0.009614	0.308416	...	0.003257	-0.036864	-0.043878	0.00
sausages	-0.174290	-0.064869	-0.046976	0.287420	-0.128150	0.647630	0.056315	-0.240440	-0.025094	0.502220	...	0.302240	0.195470	-0.653980	-0.29
lazy	-0.353320	-0.299710	-0.176230	-0.321940	-0.385640	0.586110	0.411160	-0.418680	0.073093	1.486500	...	0.402310	-0.038554	-0.288670	-0.24
love	0.139490	0.534530	-0.252470	-0.125650	0.048748	0.152440	0.199060	-0.065970	0.128830	2.055900	...	-0.124380	0.178440	-0.099469	0.00
quick	-0.445630	0.191510	-0.249210	0.465900	0.161950	0.212780	-0.046480	0.021170	0.417660	1.686900	...	-0.329460	0.421860	-0.039543	0.15

20 rows × 300 columns

Embedding Layer 를 이용한 vector 표현

- Projection Matrix



k 번째 단어 (One-Hot-Encoding)

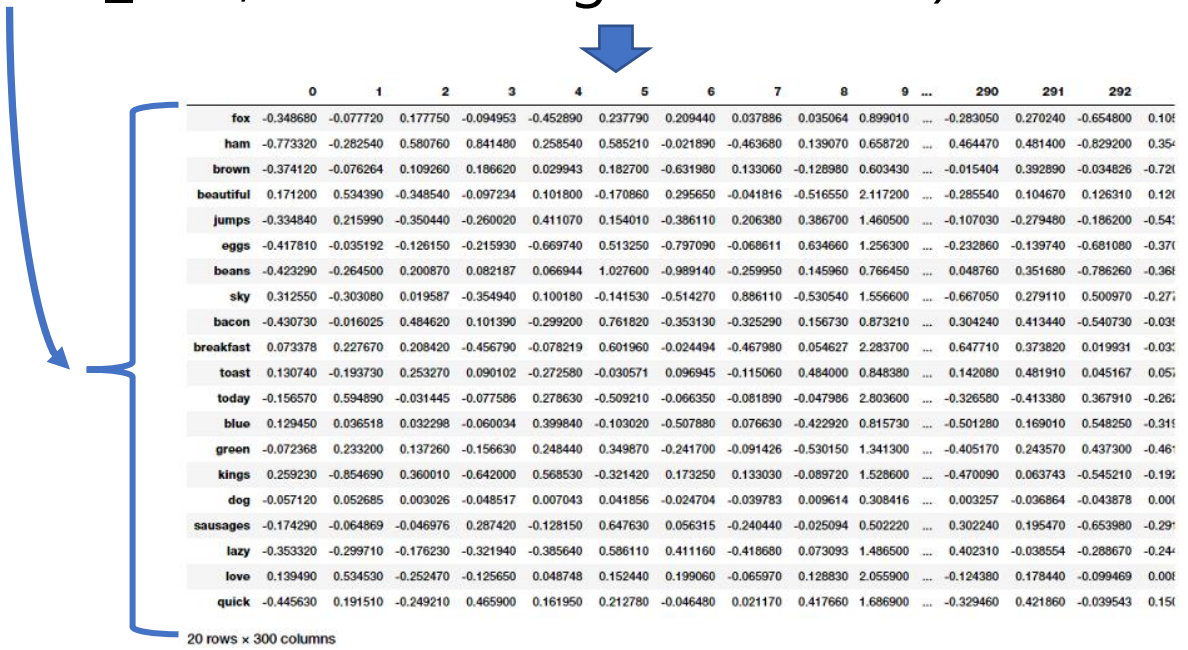
- **One-Hot-Vector 에 Projection Matrix(Embedding Layer) 를 곱해 새로운 vector 생성 ➔** 계산의 간편성

- Projection Matrix 의 k 번째 row 가 k 번째 단어에 대응하는 **weight** 임

Neural Network 의 word embedding Layer

- 입력 data 에 대해 numpy matrix 연산을 manually 하는 것은 비효율적이므로, tf.keras.layers.Embedding() 이용

- Tensorflow NLP model 의 1st layer 는 Embedding layer 로 시작
  - tf.keras.layers.Embedding(vocab_size, embedding dimension)
  - One-hot-encoding
  - indexing

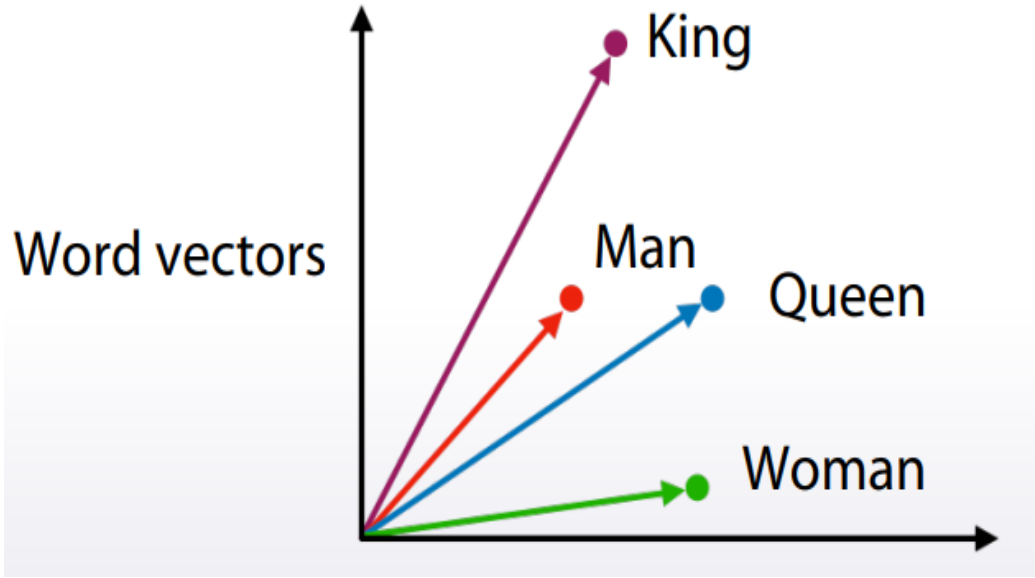


	0	1	2	3	4	5	6	7	8	9	...	290	291	292	
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	...	-0.283050	0.270240	-0.654800	0.10
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	...	0.464470	0.481400	-0.829200	0.35
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	...	-0.015404	0.392890	-0.034826	-0.72
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	...	-0.285540	0.104670	0.126310	0.12
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	...	-0.107030	-0.279480	-0.186200	-0.54
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	...	-0.232860	-0.139740	-0.681080	-0.37
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	...	0.048760	0.351680	-0.786260	-0.36
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	...	-0.667050	0.279110	0.500970	-0.27
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	...	0.304240	0.413440	-0.540730	-0.03
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	...	0.647710	0.373820	0.019931	-0.03
toast	0.130740	-0.193730	0.253270	0.090102	-0.272580	-0.030571	0.096945	-0.115060	0.484000	0.848380	...	0.142080	0.481910	0.045167	0.05
today	-0.156570	0.594890	-0.031445	-0.077586	0.278630	-0.509210	-0.066350	-0.081890	-0.047986	2.803600	...	-0.326580	-0.413380	0.367910	-0.26
blue	0.129450	0.036518	0.032298	-0.060034	0.399840	-0.103020	-0.507880	0.076630	-0.422920	0.815730	...	-0.501280	0.169010	0.548250	-0.31
green	-0.072368	0.233200	0.137260	-0.156630	0.248440	0.349870	-0.241700	-0.091426	-0.530150	1.341300	...	-0.405170	0.243570	0.437300	-0.46
kings	0.259230	-0.854690	0.360010	-0.642000	0.568530	-0.321420	0.173250	0.133030	-0.089720	1.528600	...	-0.470090	0.063743	-0.545210	-0.19
dog	-0.057120	0.052685	0.003026	-0.048517	0.007043	0.041856	-0.024704	-0.039783	0.009614	0.308416	...	0.003257	-0.036864	-0.043878	0.00
sausages	-0.174290	-0.064869	-0.046976	0.287420	-0.128150	0.647630	0.056315	-0.240440	-0.025094	0.502220	...	0.302240	0.195470	-0.653980	-0.29
lazy	-0.353320	-0.299710	-0.176230	-0.321940	-0.385640	0.586110	0.411160	-0.418680	0.073093	1.486500	...	0.402310	-0.038554	-0.288670	-0.24
love	0.139490	0.534530	-0.252470	-0.125650	0.048748	0.152440	0.199060	-0.065970	0.128830	2.055900	...	-0.124380	0.178440	-0.099469	0.00
quick	-0.445630	0.191510	-0.249210	0.465900	0.161950	0.212780	-0.046480	0.021170	0.417660	1.686900	...	-0.329460	0.421860	-0.039543	0.15

20 rows × 300 columns

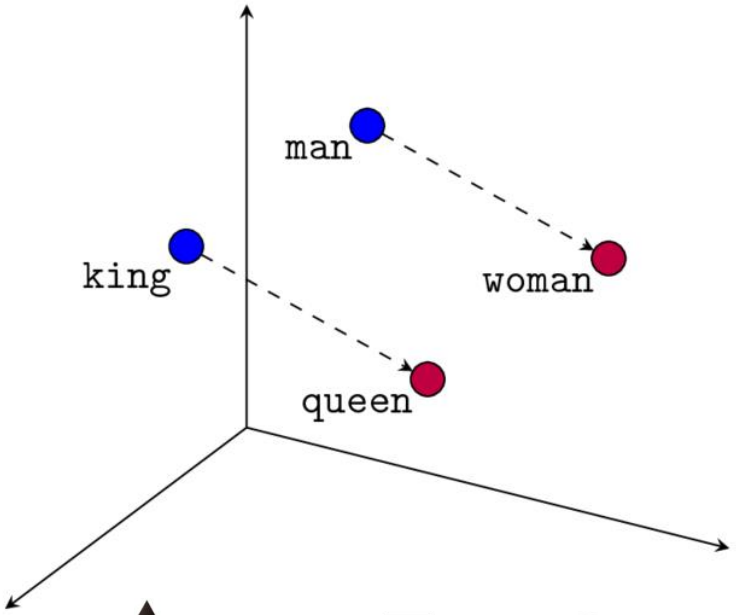
Word2Vec

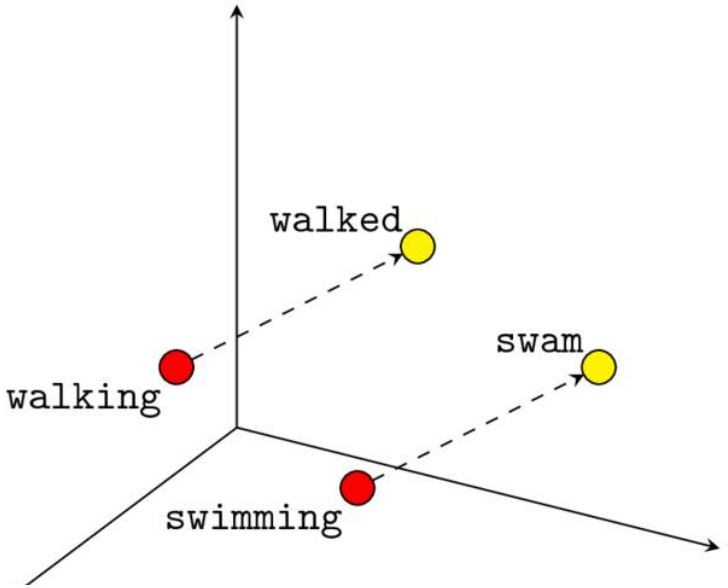
- 2013년 구글에서 개발
- 단어를 vector 화 (Word Embedding) 하여 단어의 의미 표현
- 매우 큰 Corpus (ex, 10억, 100 억 단어) 에서 word embedding 자동 학습



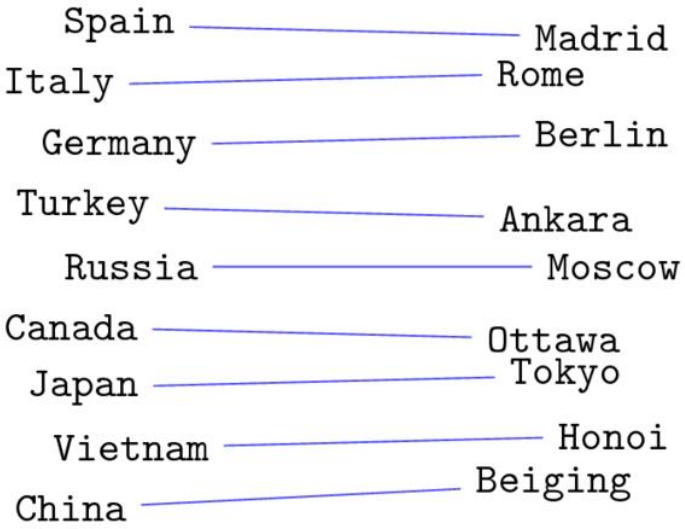
Word2Vec Vectorized (Word Embedding)

king – queen ≈ man - woman

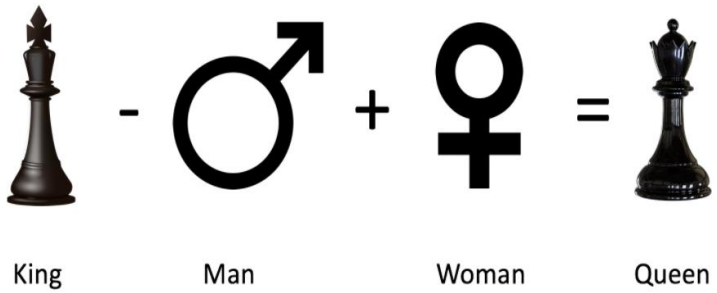




시제



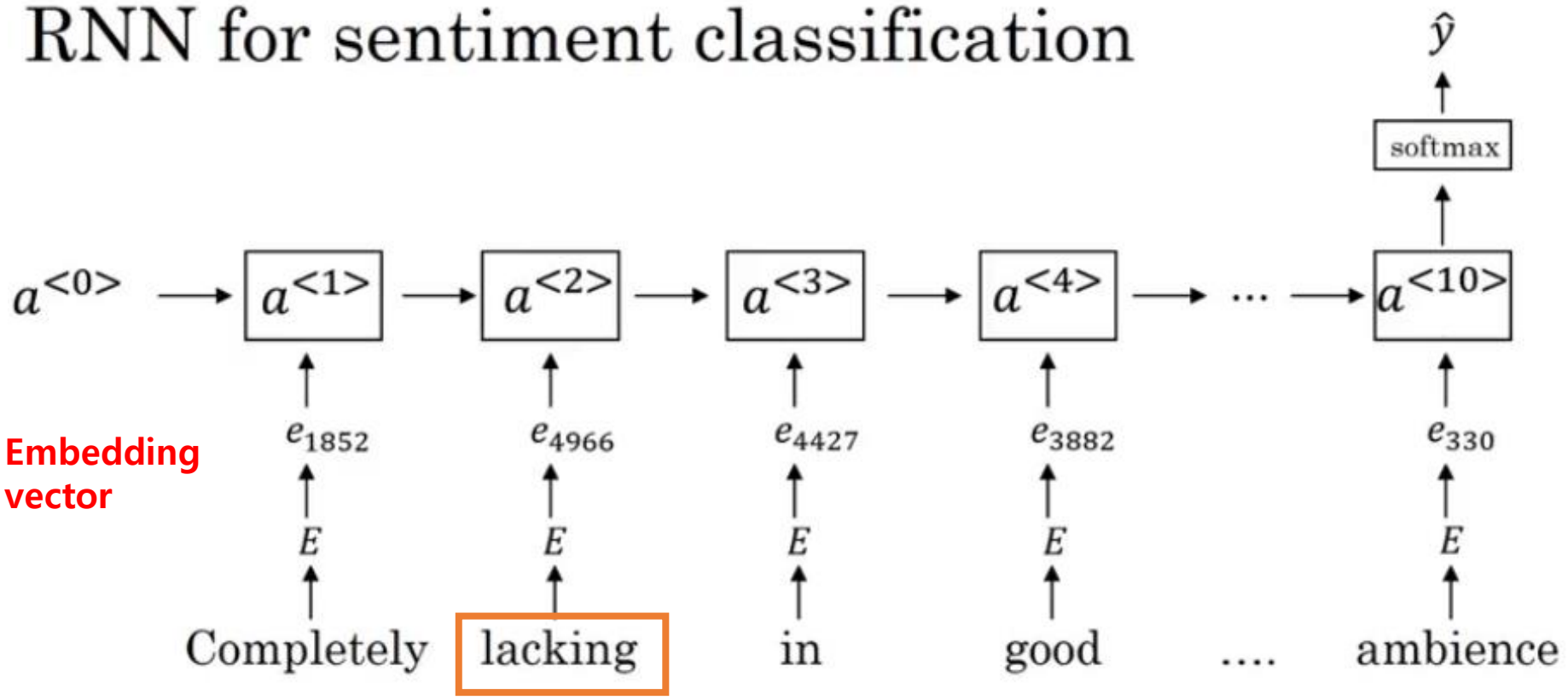
국가-수도



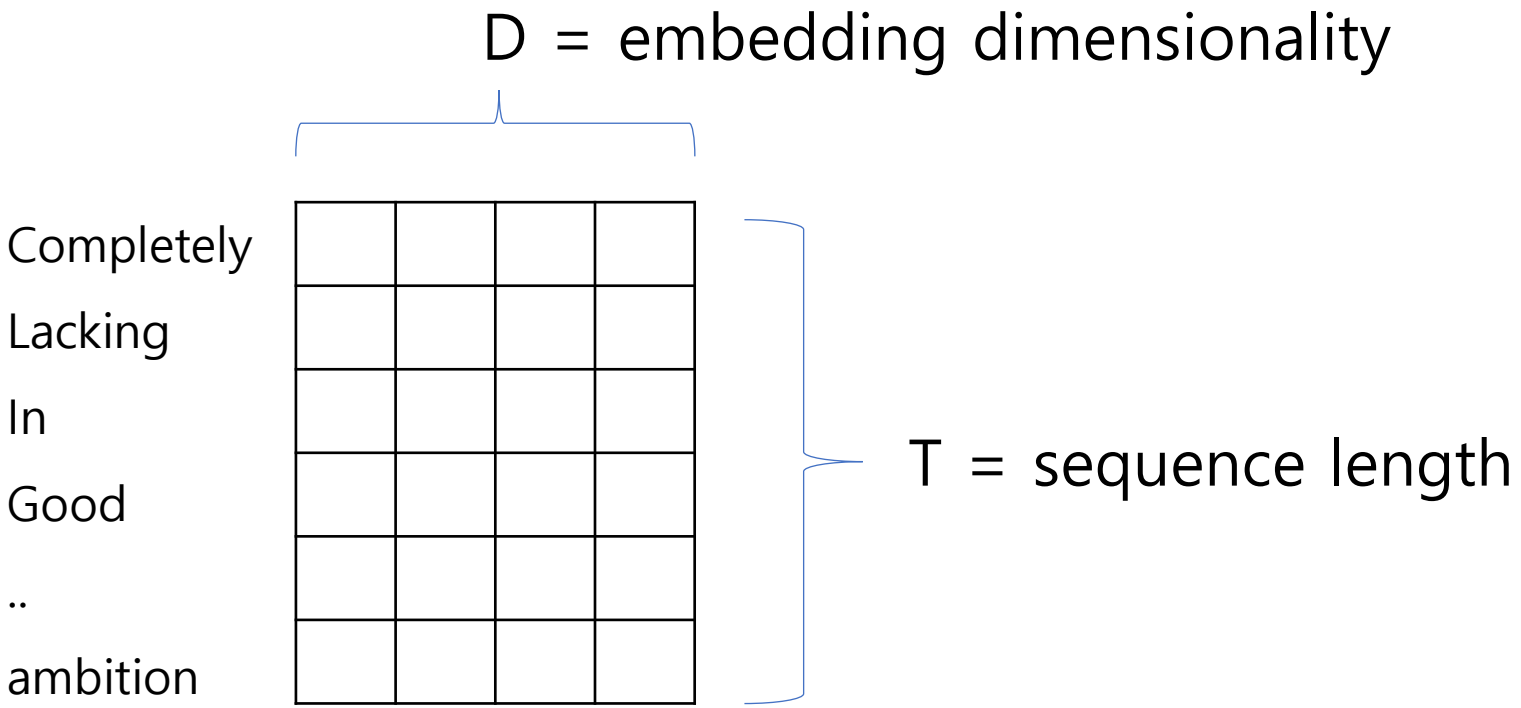
https://ronxin.github.io/wevi/

감성분석 (Sentiment Analysis)

Traditional NLP : 단순히 단어의 출현 빈도를 count 하여 positive / negative 분류
➔ 단어의 순서 무시 (Bag of Words)



RNN input (T x D)



실습: 290. keras word encoding

- keras 제공 pad_sequences method
- sentence 의 token 화

실습 : 300-네이버 영화 관람평 분류

- 150,000 개의 training set, 50,000 개의 testing set 으로 구성
- Label : 0, 1 → binary classification
- Keras basic Tokenizer 이용
- LSTM 을 이용한 Many-to-One type 의 RNN 으로 구현

Capstone Project 1

Keras Neural Network Model 을 Flask Web 으로 서비스

What is Flask ?

1. Microframework for Python

2. Framework 는 web 개발을 지원하는 software 임

3. Flask는 서버 프레임워크에 필요한 핵심 기능만 제공

4. pip install Flask 혹은 conda install –c anaconda flask 로 설치

ML Model deploy using Flask webserver

- ML model 배포 및 서비스는 일반적으로 대용량 클라우드 사용
- 초기 시스템을 개발할 때는 Flask와 같은 개발 시스템을 사용하는 것이 일반적
- 훈련된 신경망 model 을 .h5 파일로 저장하고 web server 에 배포

Project 구현 순서

1. Jupyter Notebook 에서 model 생성

2. Flask web application 작성

3. json data로 입력 및 출력 확인
   - 방법 1 – Python Code 로 request
   - 방법 2 – Web Brower REST API 이용
     - VSCODE의 Thunder Client 확장 프로그램 설치

CapstoneProject_1 :
Flask and Deep Learning Web Services

- 자동차 연비를 예측하는 regression model 작성
  - model.save("./dnn/mpg_model.h5")
  - pickle.dump(sc, open("./dnn/standard_scaler.pkl", "wb"))

- Flask Web application 작성
  - model = load_model("./dnn/mpg_model.h5")
  - sc = pickle.load(open("./dnn/standard_scaler.pkl", "rb"))

- 온라인 query를 통한 연비 예측 서비스 제공

Hello World

```
from werkzeug.wrappers import Request, Response
from flask import Flask
import sys
from werkzeug.serving import run_simple

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == '__main__':
    run_simple('localhost', 9000, app)
```

Flask app

```python
app = Flask(__name__)

# 입력 data validation
EXPECTED = {
    "cylinders": {"min": 3, "max": 8},
    "displacement": {"min": 68.0, "max": 455.0},
    "horsepower": {"min": 46.0, "max": 230.0},
    "weight": {"min": 1613, "max": 5140},
    "acceleration": {"min": 8.0, "max": 24.8},
    "year": {"min": 70, "max": 82},
    "origin": {"min": 1, "max": 3}
}

# model load
model = load_model(os.path.join("./dnn/", "mpg_model.h5"))


@app.route("/api/mpg/", methods=['POST'])
def calc_mpg():
    content = request.json
    errors = []
    #input validity check
    for name in content:
        if name in EXPECTED:
            value = content[name]
            if value < EXPECTED[name]['min'] or value > EXPECTED[name]['max']:
                error.append(f"입력값 범위 초과 - {name}: {value}")
        else:
            error.append(f"Unexpected field - {name}")
    # missing input check
    for name in EXPECTED:
        if name not in content:
            errors.append(f"missing input - {name}")

    # prediction
    if len(errors) < 1:
        x = np.zeros((1, 7))
        x[0, 0] = content['cylinders']
        x[0, 1] = content['displacement']
        x[0, 2] = content['horsepower']
        x[0, 3] = content['weight']
        x[0, 4] = content['acceleration']
        x[0, 5] = content['year']
        x[0, 6] = content['origin']

        pred = model.predict(x)
        mpg = float(pred[0])
        response = {"id": str(uuid.uuid4()), "mpg": mpg, "errors": errors}
    else:
        response = {"id": str(uuid.uuid4()), "errors": errors}

    print(response)

    return jsonify(response)

if __name__ == '__main__':
    run_simple('localhost', 9000, app)
```

방법 1) 다른 notebook에서 query 수행

jupyter Untitled Last Checkpoint: 한 시간 전 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [4]:

```
import requests

json = {
  "cylinders": 8,
  "displacement": 300,
  "horsepower": 78,
  "weight": 3500,
  "acceleration": 20,
  "year": 76,
  "origin": 1
}

r = requests.post("http://localhost:9000/api/mpg",json=json)
if r.status_code == 200:
    print("Success: {}".format(r.text))
else: print("Failure: {}".format(r.text))

```

```
Success: {"errors":[],"id":"551ae306-04a8-403f-b6f3-4b5d829c4aac","mpg":16.958744049072266}
```

방법 2) Web Brower REST API 이용

POST http://localhost:9000/api/mpg Send

Query Headers 2 Auth Body 1 Tests

Json Xml Text Form Form-encode Graphql Binar

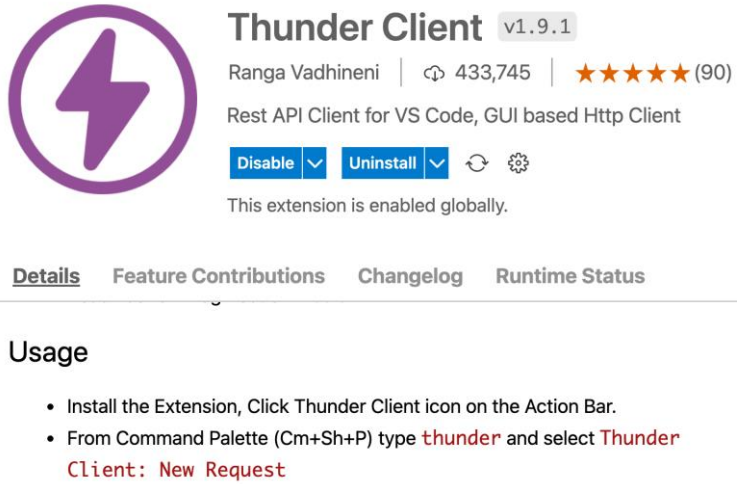
Json Content

```
{
  "cylinders": 8,
  "displacement": 300,
  "horsepower": 78,
  "weight": 3500,
  "acceleration": 20,
  "year": 76,
  "origin": 1
}
```

Status: 200 OK Size: 83 Bytes Time: 106 ms

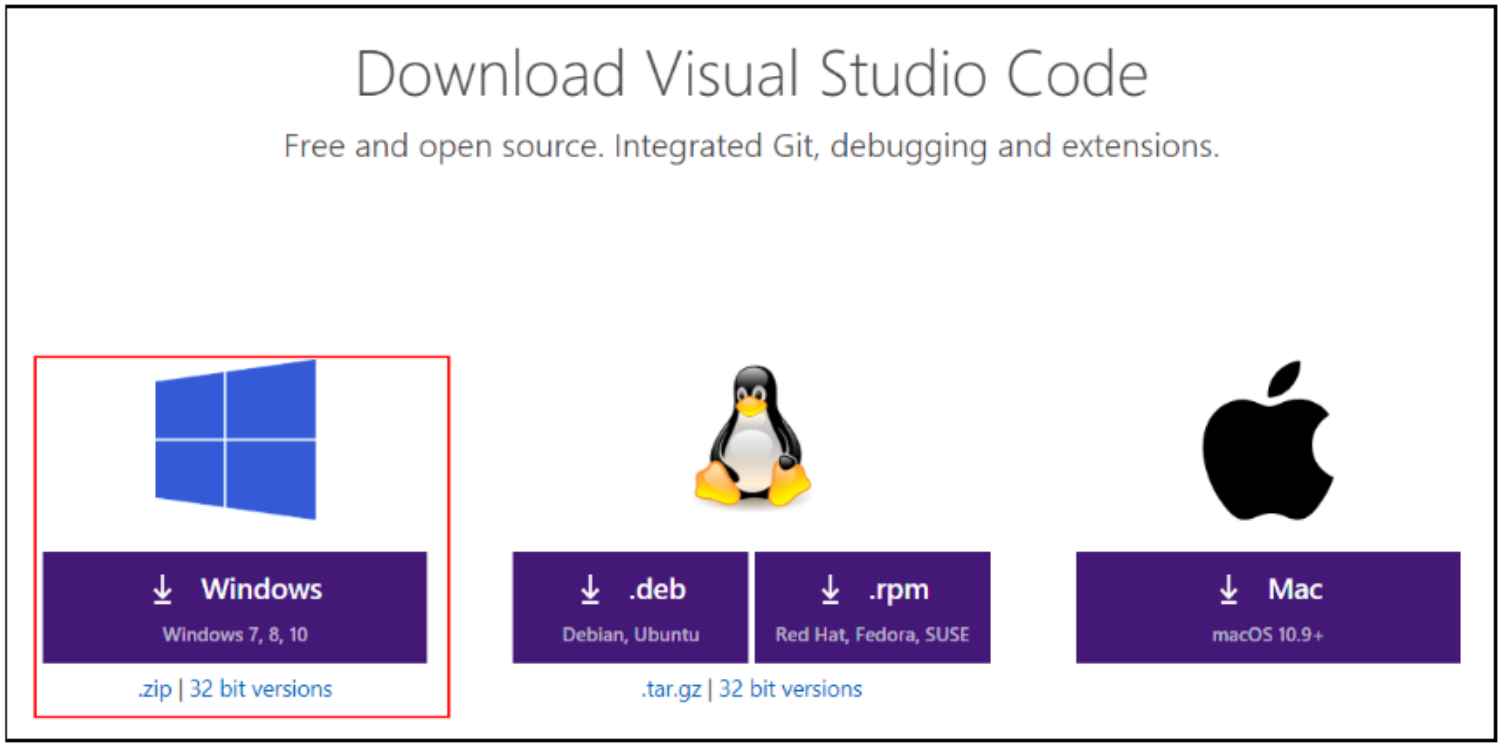
Response Headers 4 Cookies Test Results { }

```
{
  "errors": [],
  "id": "14615961-6706-43e5-9a7c-e7bc3ac8d6d3",
  "mpg": 16.958744049072266
}
```



Visual Studio Code 설치

- https://code.visualstudio.com/download



VSCODE Thunder Client extension 설치

Thunder Client v1.9.1

Ranga Vadhineni | 365,563 | ★★★★★ (85)

Rest API Client for VS Code, GUI based Http Client

사용 안 함 | 제거

이 확장은 전역적으로 사용하도록 설정되었습니다.

세부 정보 | 기능 기여도 | 변경 로그 | 런타임 상태

- Website - www.thunderclient.io
- Follow Twitter for updates - twitter.com/thunder_client
- Support: github.com/rangav/thunder-client-support

Story behind Thunder Client

- Read Launch Blog Post on Medium

Usage

- Install the Extension, Click Thunder Client icon on the Action Bar.
- From Command Palette (Cm+Sh+P) type `thunder` and select `Thunder Client: New Request`

범주

Programming Languages

리소스

마켓플레이스
저장소
라이선스

추가 정보

실습: flask web 을 이용한 model 배포

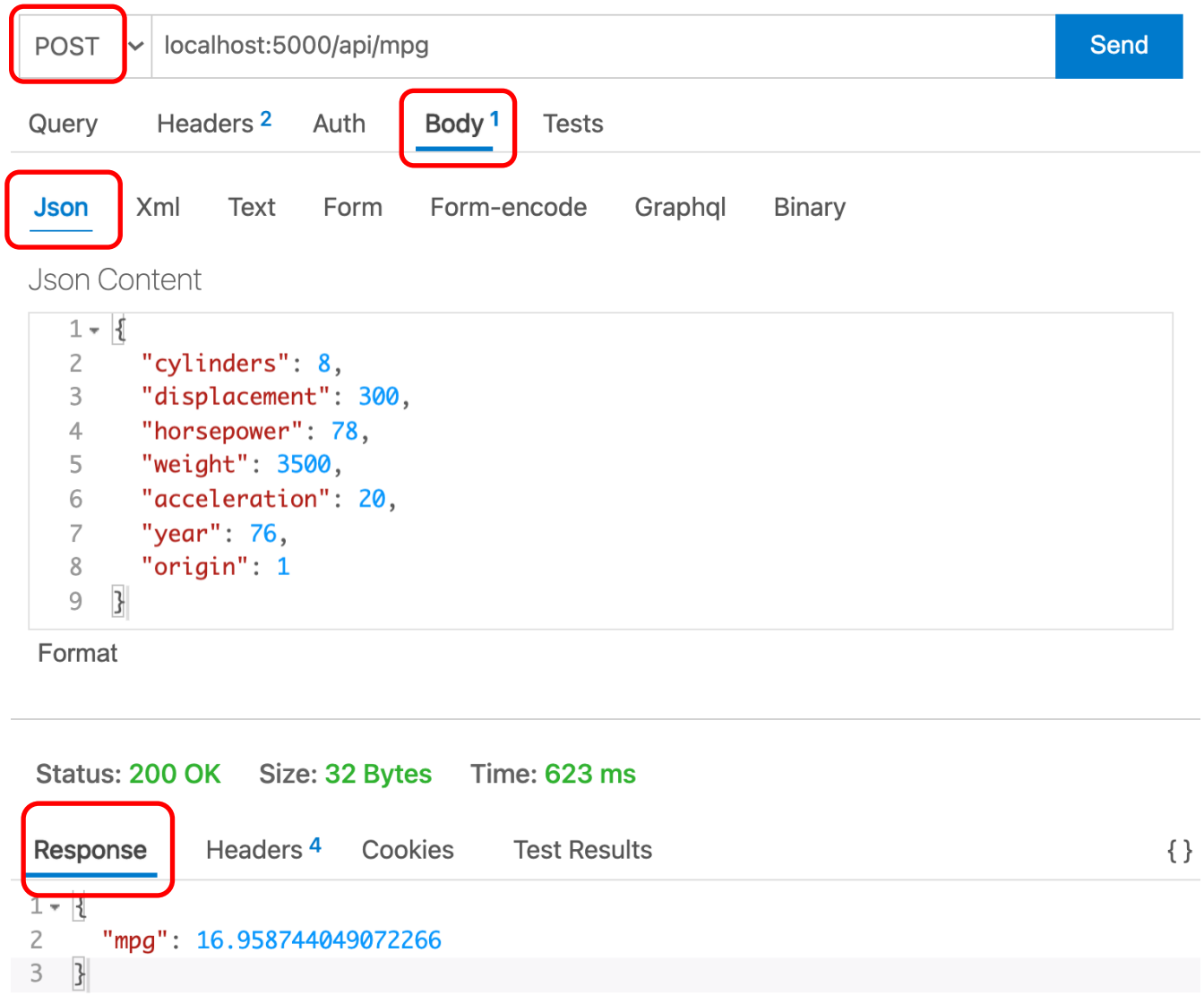
- 901_app.py
  - Flask app template
- 902_mpg_server.py
  - 연비 예측 모델 flask web app – POST method
- 903_mpg_post.py
  - json data 를 이용한 direct query example
- 904_image_server.py
  - image file upload 및 mobilenet 을 이용한 분류

901_app.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return '<h1>Hello Puppy!</h1>'

@app.route('/information')
def info():
    return '<h1>Puppies are cute!</h1>'

if __name__ == '__main__':
    app.run(host='localhost', debug=True)
```

902_mpg_server.py

Thunder Client




```
from tensorflow.keras.models import load_model
import numpy as np

app = Flask(__name__)

# Load neural network model
model = load_model(os.path.join("./dnn/", "mpg_model.h5"))

@app.route('/api/mpg', methods=['POST'])
def calc_mpg():
    content = request.json
    print("******")
    print(content)

    # Predict
    x = np.zeros((1, 7))

    x[0, 0] = content['cylinders']
    x[0, 1] = content['displacement']
    x[0, 2] = content['horsepower']
    x[0, 3] = content['weight']
    x[0, 4] = content['acceleration']
    x[0, 5] = content['year']
    x[0, 6] = content['origin']

    pred = model.predict(x)
    mpg = float(pred[0])
    response = {"mpg": mpg}

    return jsonify(response)

if __name__ == '__main__':
    app.run(host='localhost', debug=True)
```

903_mpg_server.py

```
import requests

json = {
  "cylinders": 8,
  "displacement": 300,
  "horsepower": 78,
  "weight": 3500,
  "acceleration": 20,
  "year": 76,
  "origin": 1
}

r = requests.post("http://localhost:5000/api/mpg",json=json)
if r.status_code == 200:
    print("Success: {}".format(r.text))
else:
    print("Failure: {}".format(r.text))
```

```
Success: {
  "mpg": 16.958744049072266
}
```

Capstone Project 2

Image 분류

CapstoneProject_2_flask_image_app.py

- Pre-trained VGG16 model 을 이용한 Image 분류
- Flask Webserver start
- 온라인 query를 통한 임의의 Image 분류 서비스 구현

127.0.0.1:5000/predict

앱 소프트협회공지사항 NAVER https:/

Upload new File

파일 선택 선택된 파일 없음 Upload

열기

OneDrive - Personal > DeepLearning_ML_based_DataAnalysis > images

구성 새 폴더

OneDrive - Personal
.ipynb_checkpoints
20200208-29ML강의동영상
AlgorithmPython_old
AlgorithmPython-2021
AlgorithmPython-master
Case_based_AI
DeepLearning_ML_based_DataAnalysis


banana.jpg

bird.png

coffee.jpg

puffin.jpg

puffin2.jpg

127.0.0.1:5000/predict

앱 소프트협회공지사항 NAVER https://oyj-static-t


```
{
  "predictions": [
    {
      "label": "king_penguin",
      "probability": 0.8935797810554504
    },
    {
      "label": "albatross",
      "probability": 0.02783546969294548
    },
    {
      "label": "oystercatcher",
      "probability": 0.024877114221453667
    },
    {
      "label": "toucan",
      "probability": 0.01047253143042326
    },
    {
      "label": "magpie",
      "probability": 0.007376980502158403
    }
  ],
  "success": true
}
```