Python 을 활용한 머신러닝 입문

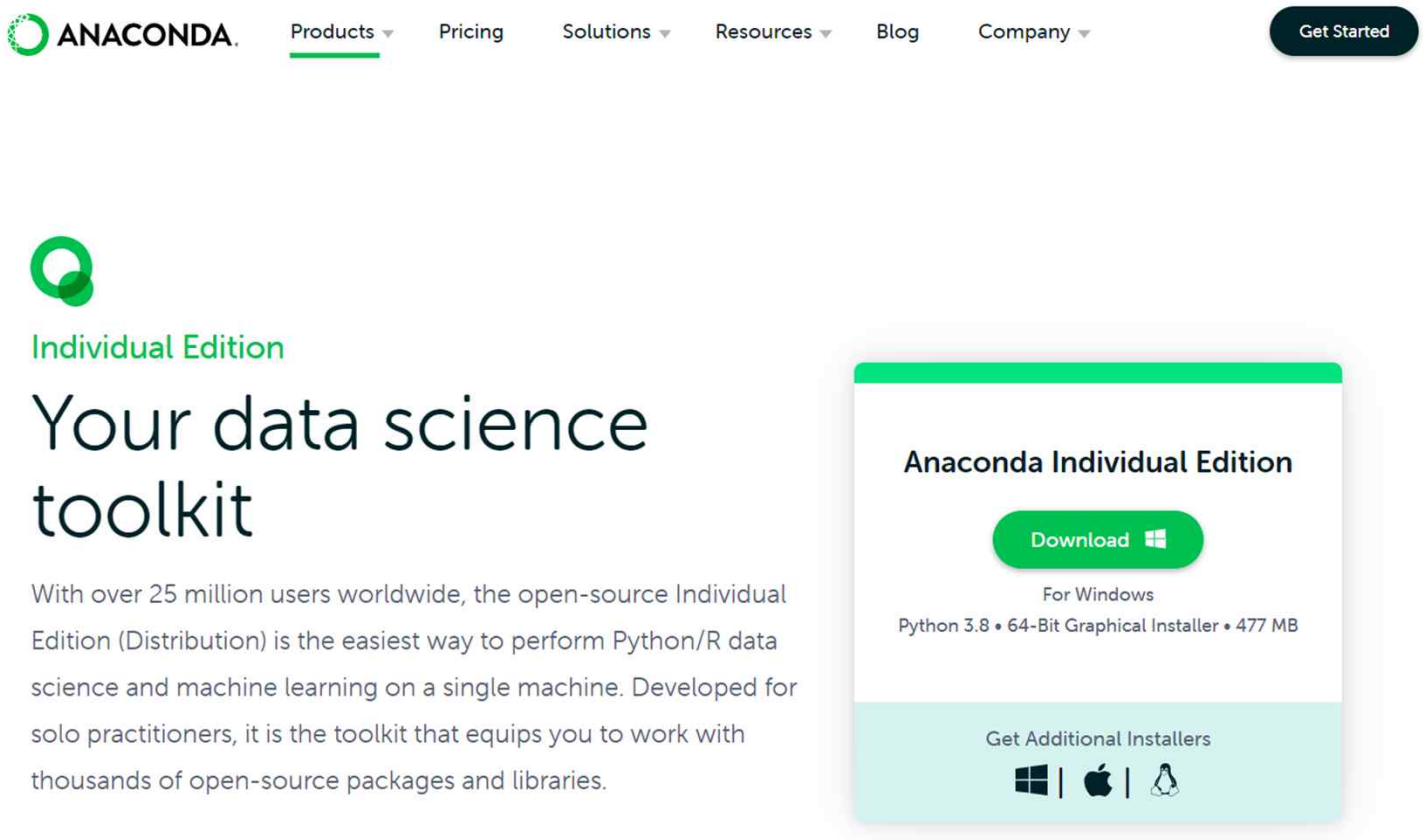
강사: 오영제

교육 환경 준비

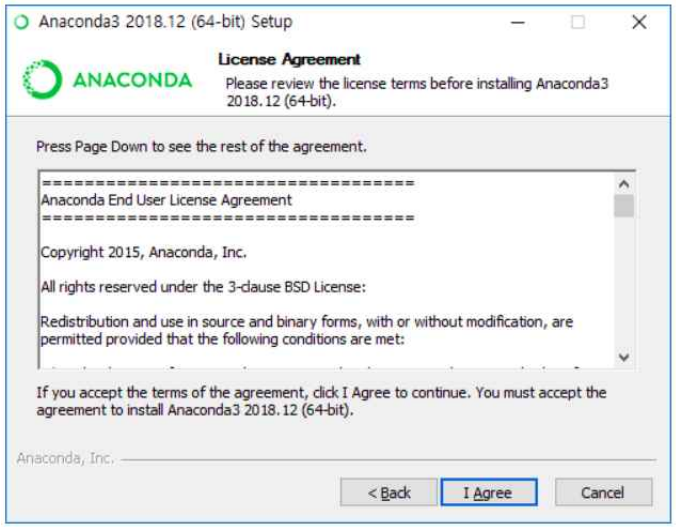
- 인터넷 연결
- Chrome Browser
- Anaconda 설치
- Gmail ID

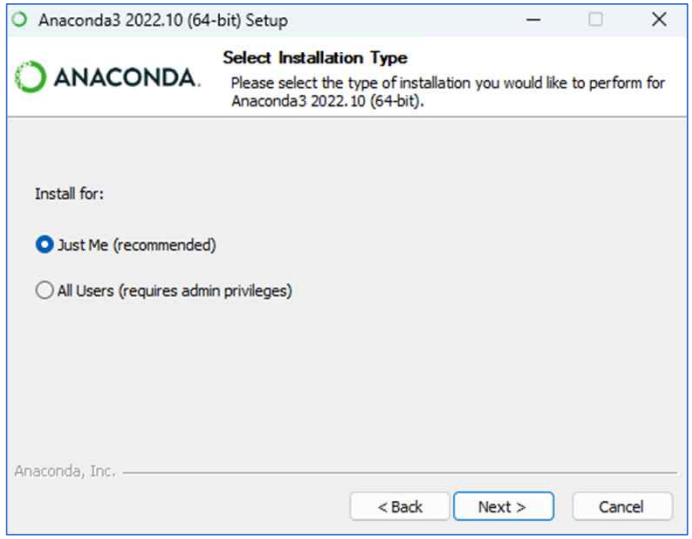
Anaconda 설치

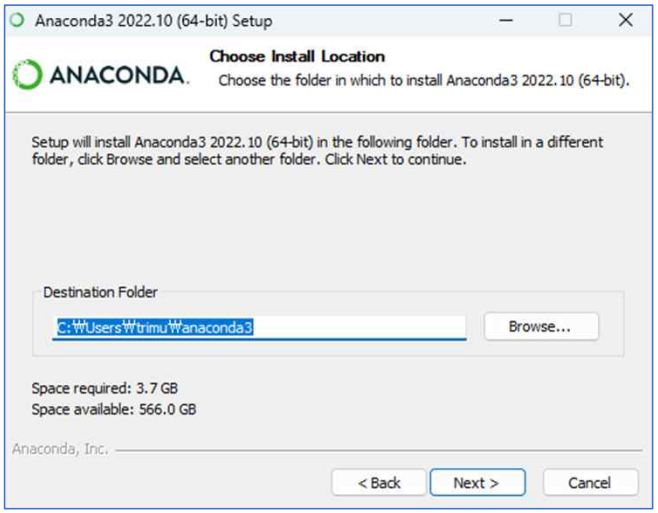
Download from https://www.anaconda.com/products/individual#download-section

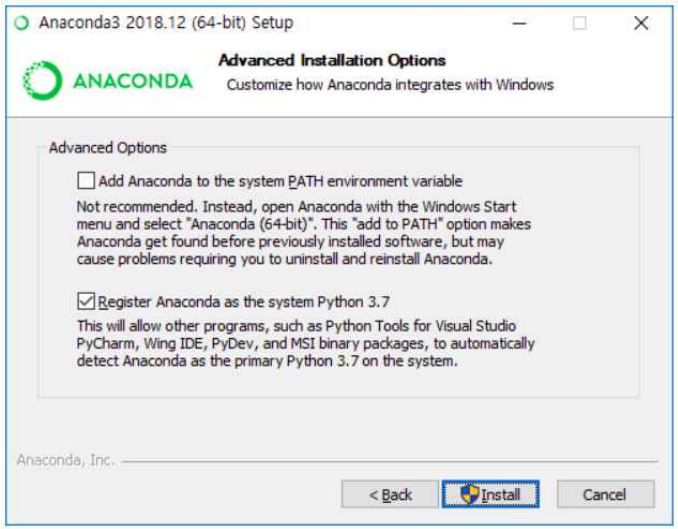


Next 를 눌러 설치 시작





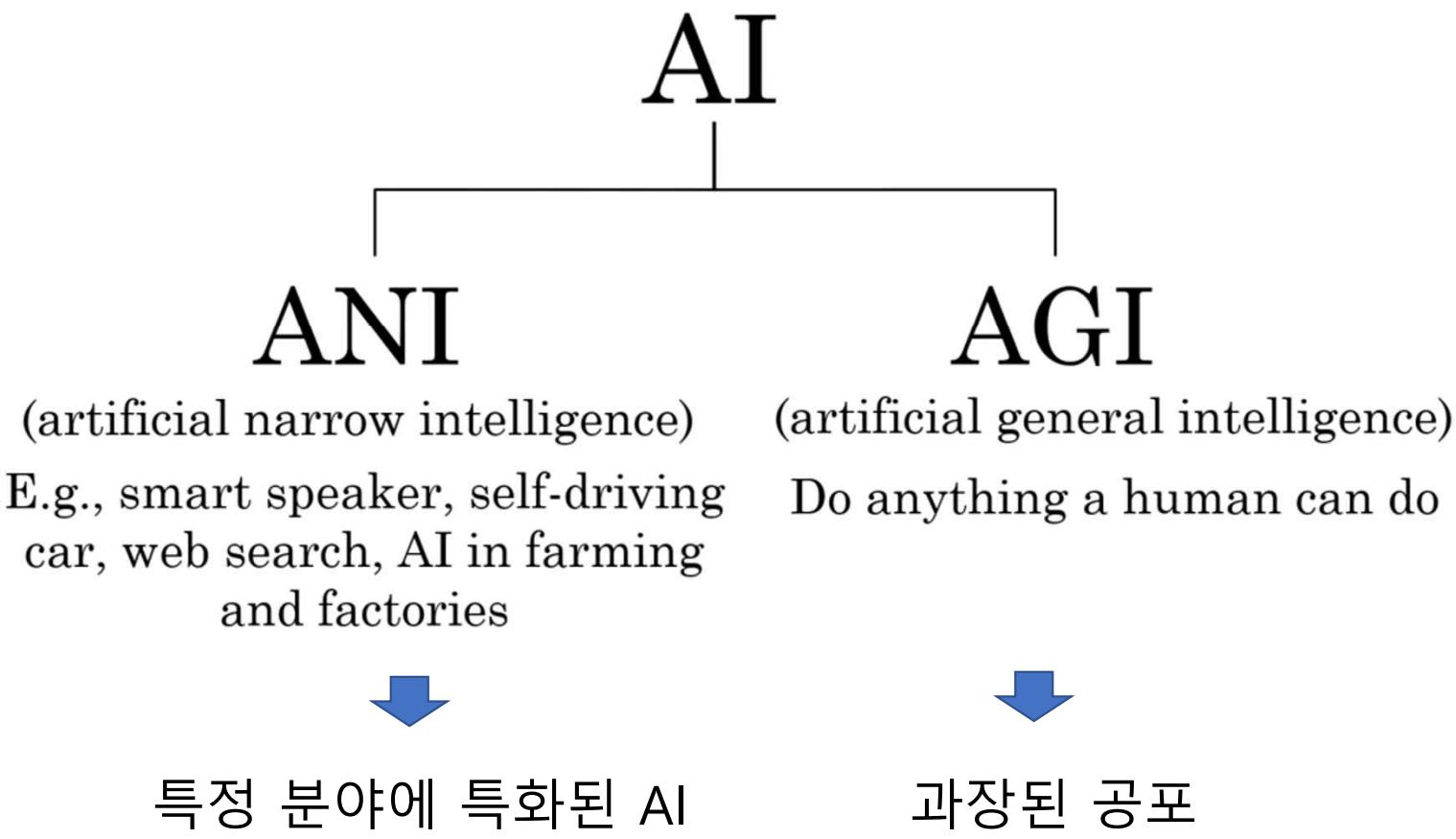




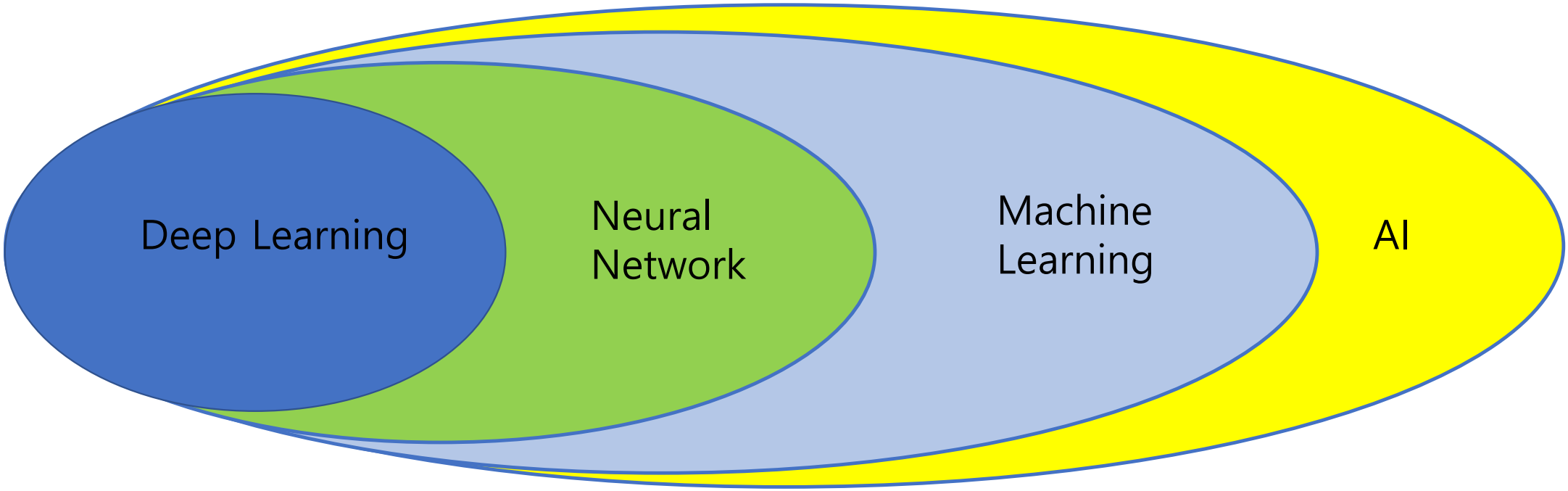
완료

Machine Learning 개요

AI 의 종류


AI
ANI
(artificial narrow intelligence)
E.g., smart speaker, self-driving car, web search, AI in farming and factories
AGI
(artificial general intelligence)
Do anything a human can do
특정 분야에 특화된 AI
과장된 공포

AI vs. Machine Learning vs. Deep Learning



History of Machine Learning

- **탄생 [1950 년대]**

  1958 년 코넬대 심리학자 프랭크 로센블래트가 인간의 뇌신경을 본떠 Perceptron 고안. 신경망 기반 인공지능 연구의 부흥기 시작

- **AI 의 첫번째 암흑기 [1970 년대]**

  Marvin Minsky 가 대규모 신경망은 학습시킬 수 없음을 수학적으로 증명. 인공지능에 대한 대규모 연구 지원 중단

- **중흥기 [1980 년대]**

  산업계에 전문가 시스템(Expert System)이 도입되며 본격 확산

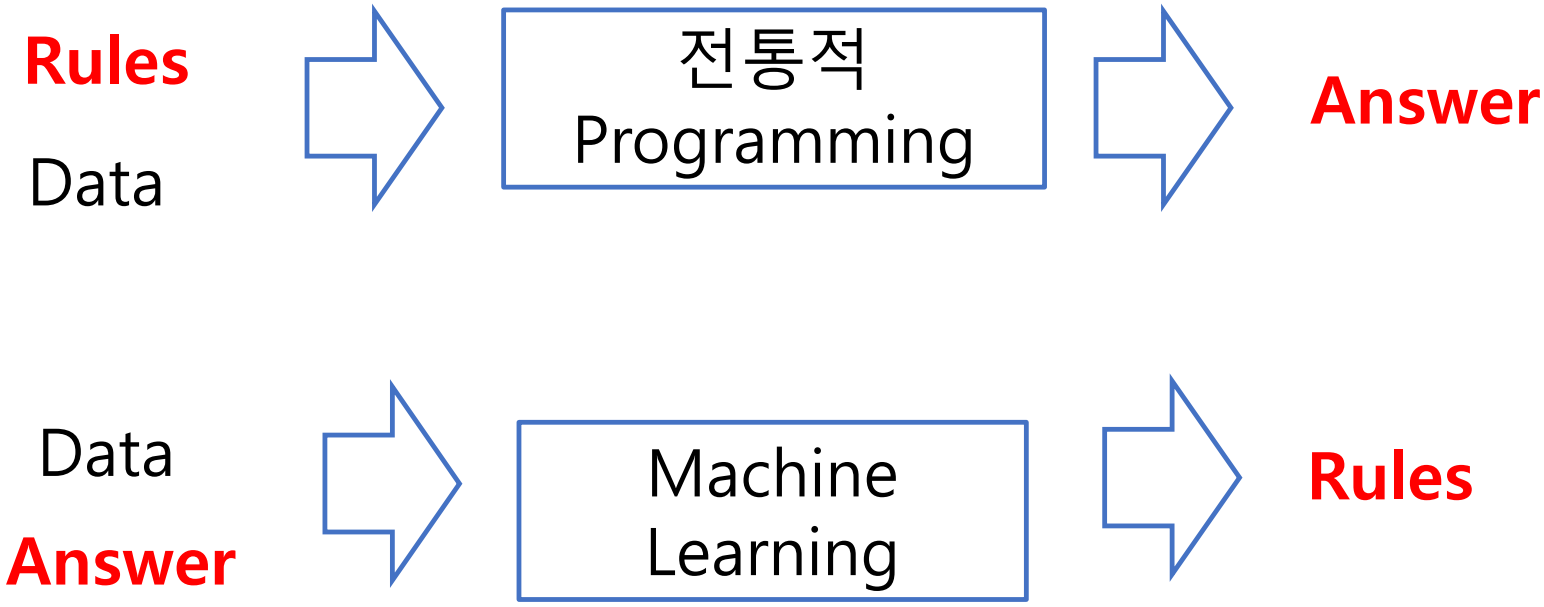
- AI 의 두번째 암흑기 [1987 – 1993]
  - 투자대비 효용성의 한계 노출
  - 슈퍼컴퓨터와 시뮬레이션 분야로 연구방향을 전환
  - Jeffery Hinton (Toronto 대학) back-propagation algorithm 개발
    → Marvin Minsky가 틀렸다는 것을 증명

- IBM Deep Blue 가 Garry Kasparov 에 승리 – 1996
  - Expert System

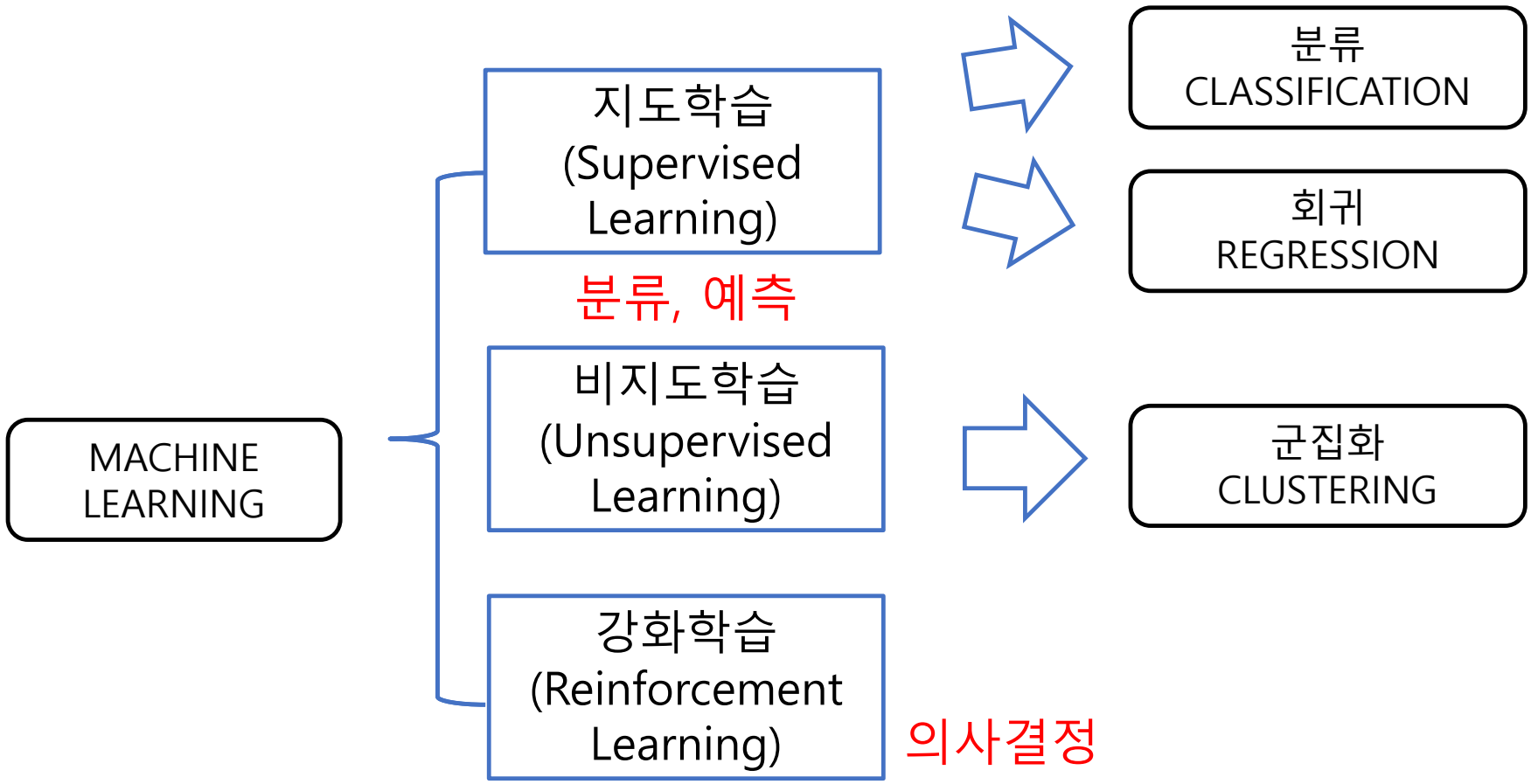
- Google Brain 이 최초로 인간 얼굴 인식 - 2012
- ~ 상업적 대 폭발기

- Alphago 이세돌에 승리 – 2016
  - 강화학습 알고리즘

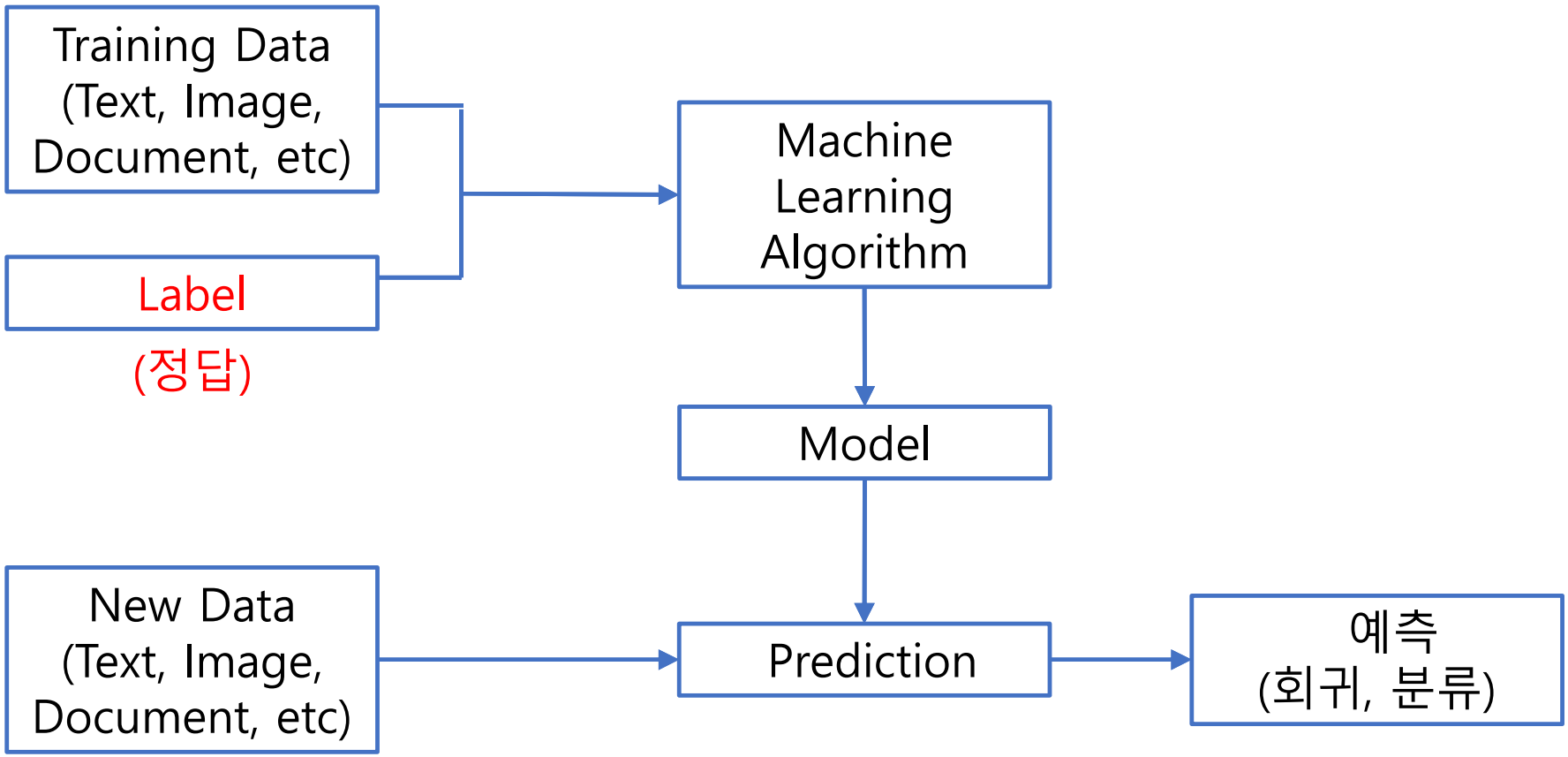
전통적 Programming
vs
Machine Learning



Machine Learning 의 종류



Supervised Learning (지도학습)



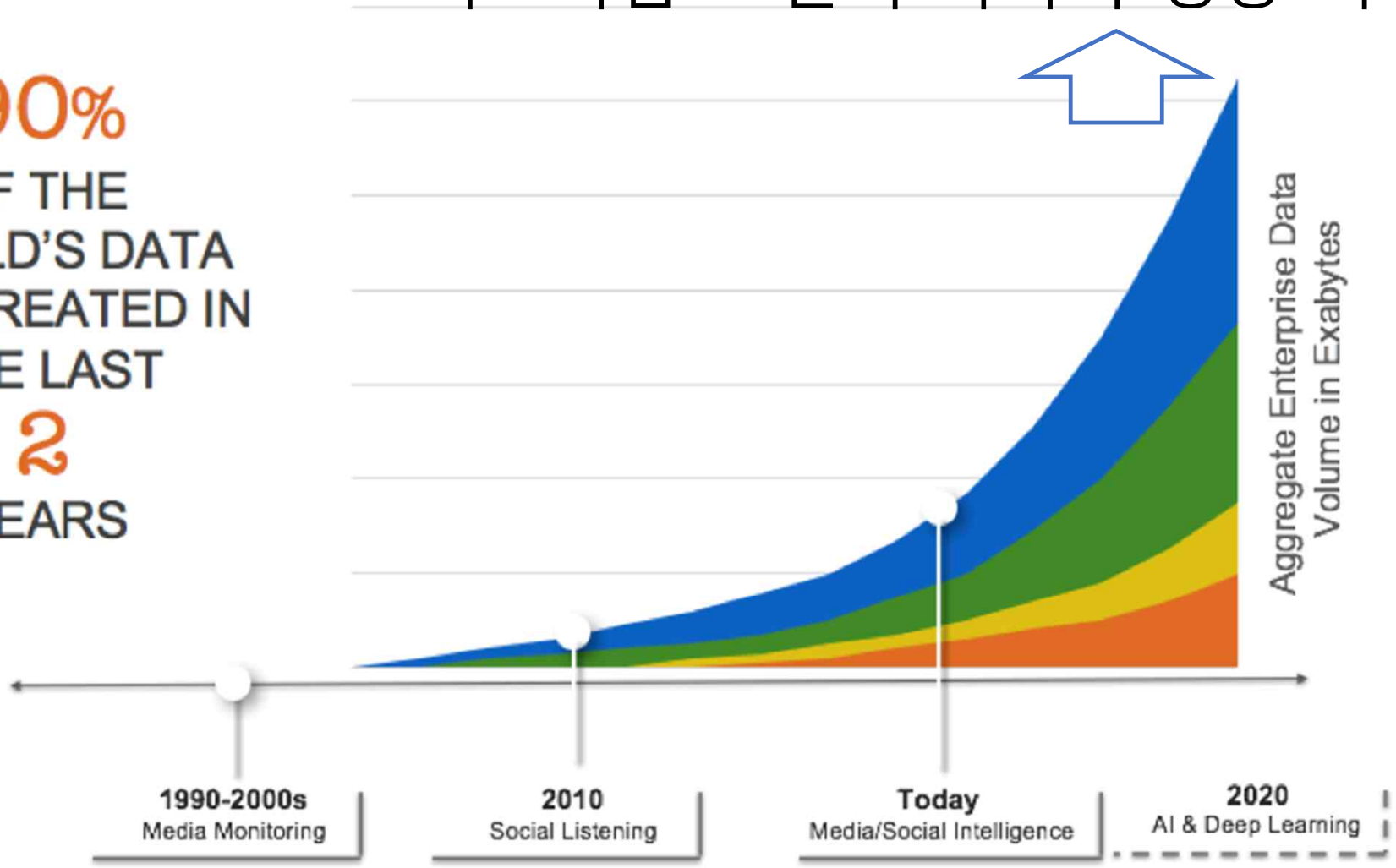
지도학습 모델의 획기적 성공 이유

90%

OF THE WORLD'S DATA WAS CREATED IN THE LAST

2

YEARS

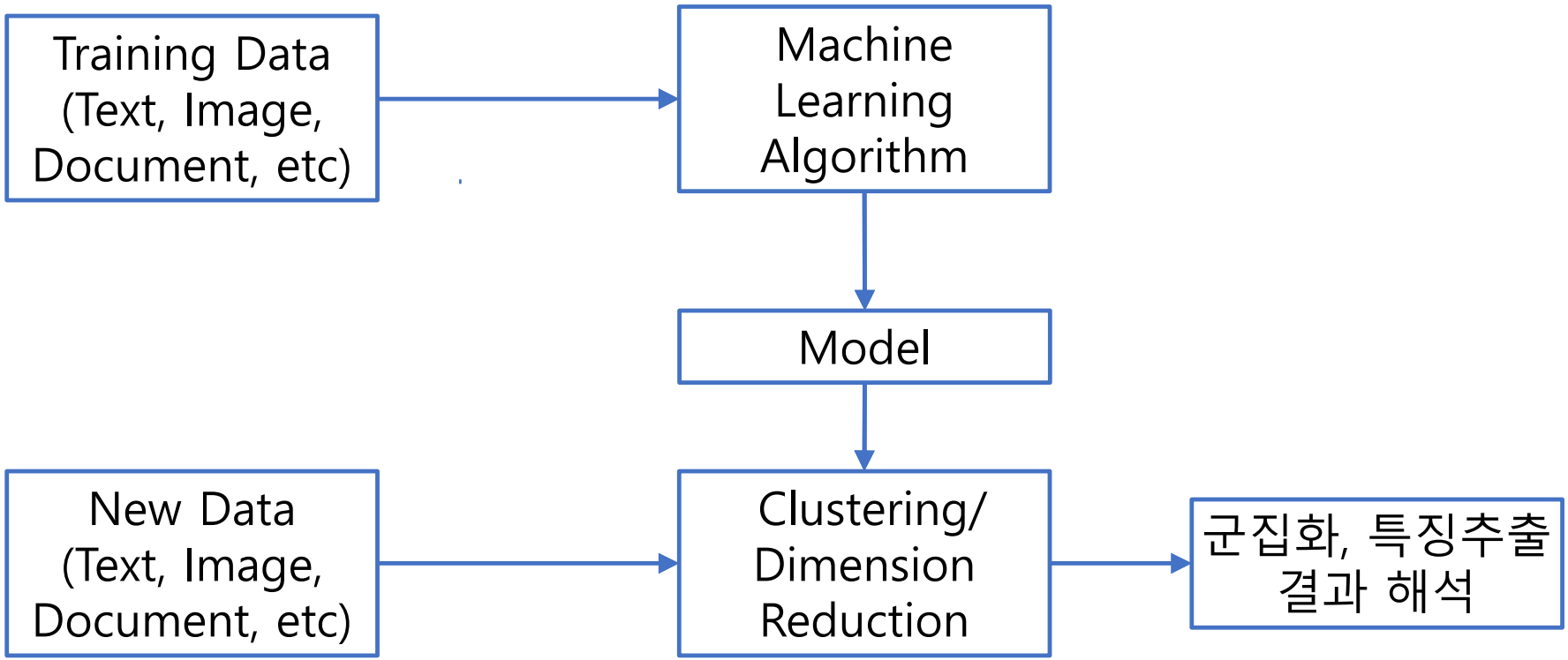


1.2 billion photos and videos are uploaded to Google Photos every day.

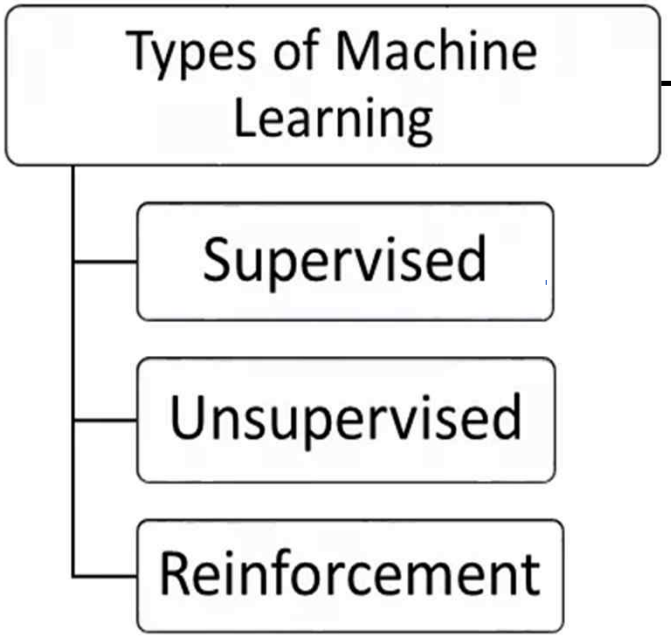
Total size of over 13 PB of photo data.

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

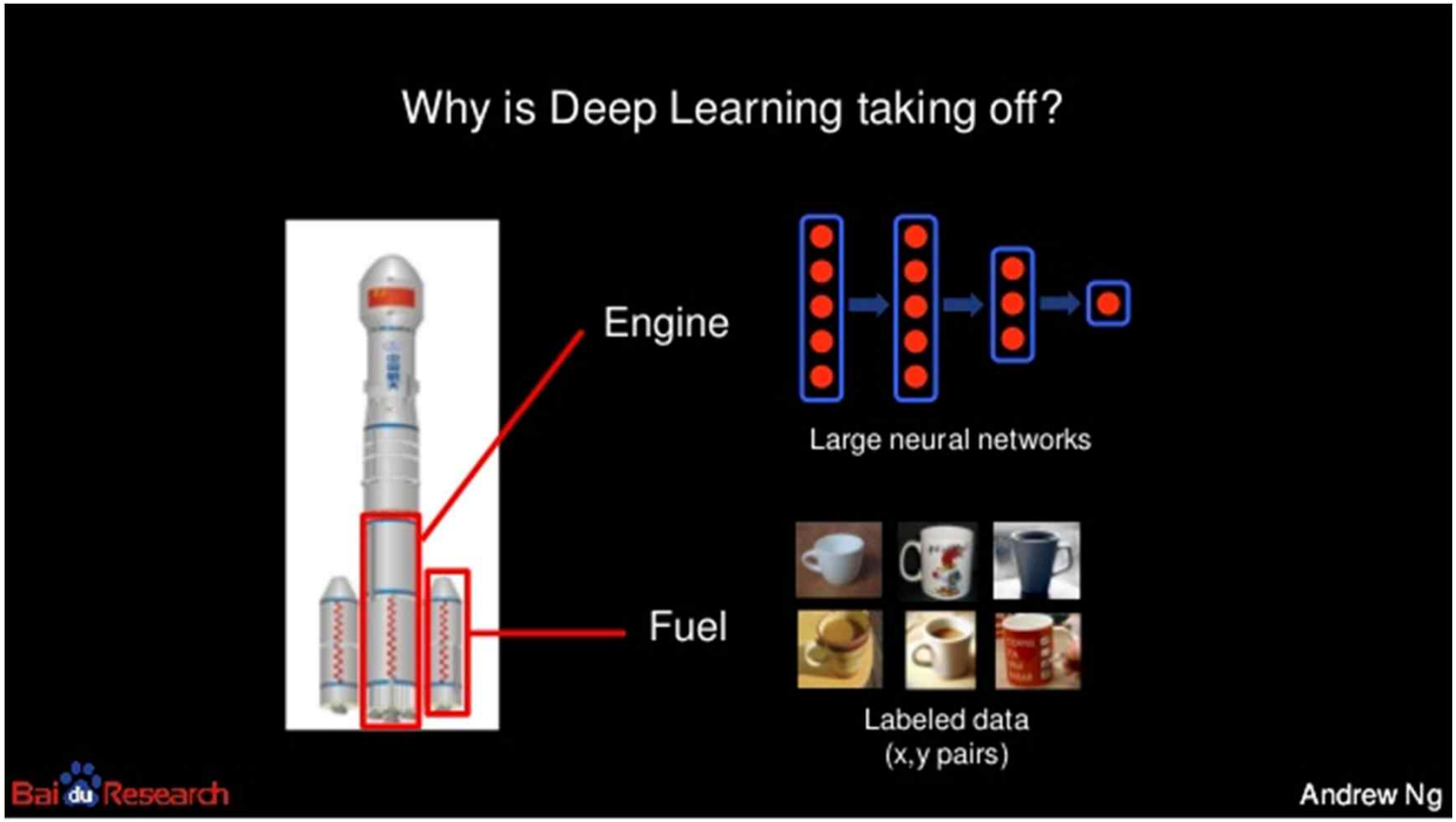
Unsupervised Learning (비지도학습)



Machine Learning 기법의 종류



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

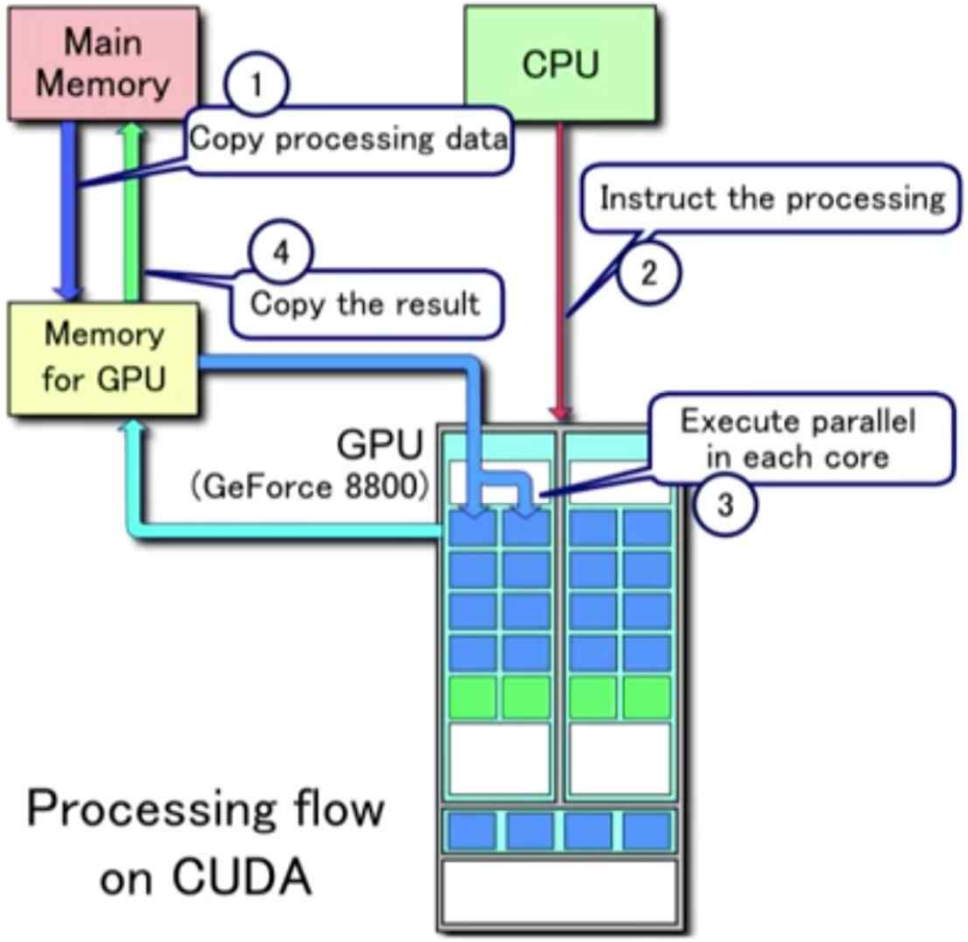


Rocket Engine : NVIDIA + Deep Learning Algorithm

Fuel : Data (25,000 pictures for cat)

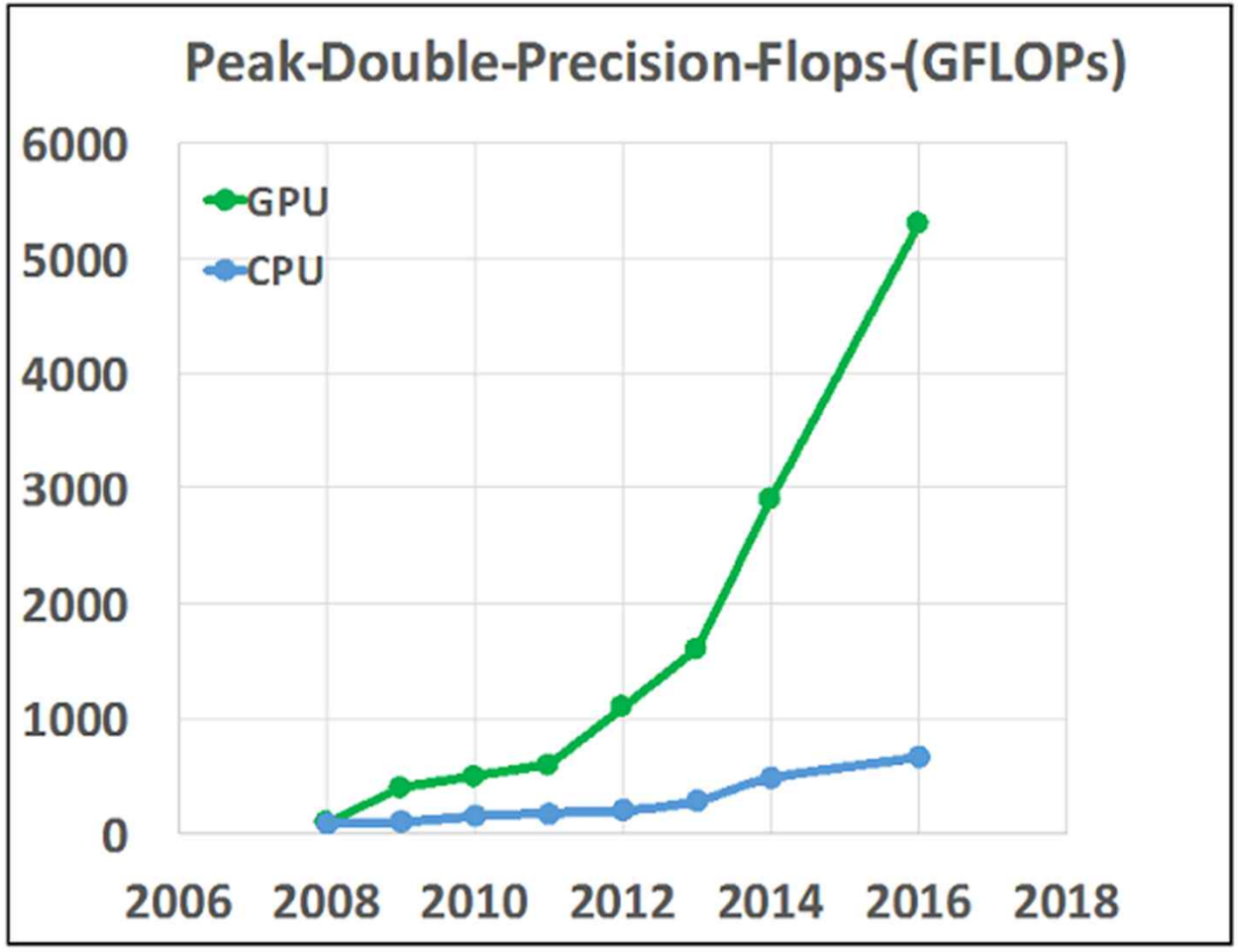
CUDA for GPU

CUDA - Compute Unified Device Architecture



Source: Wikimedia Commons

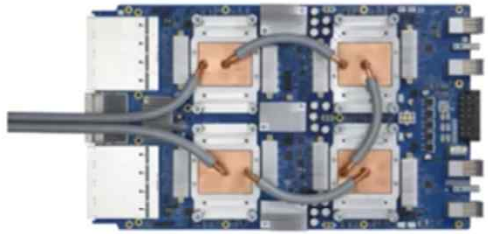
CPU vs GPU Performance



TPU (Tensor Processing Unit) - Custom ASIC by Google

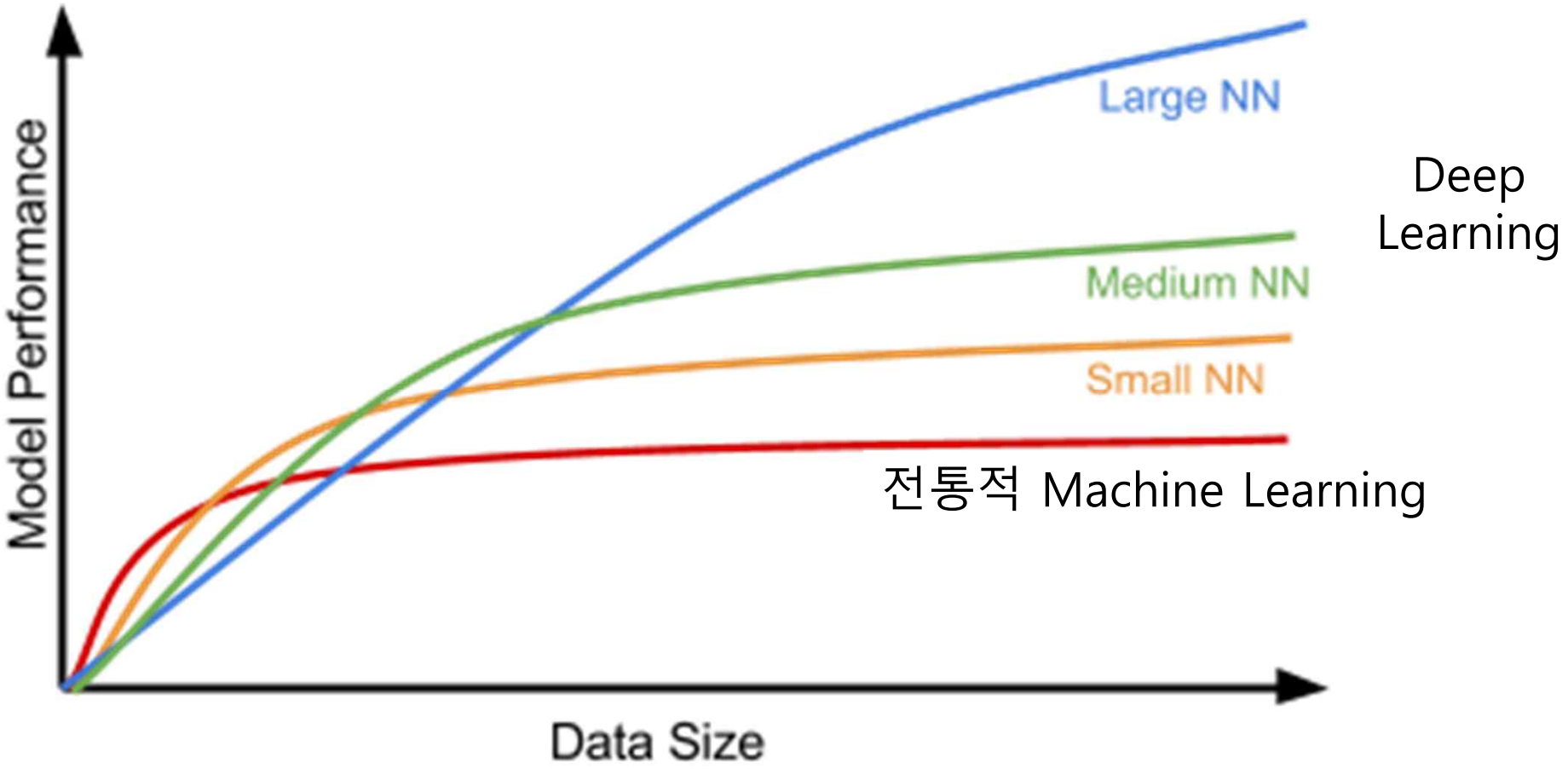
TPU v1
(2015)
92 teraops
First Generation

TPU v2
(2017)
180 teraflops
Available via Google Cloud



TPU v3
(2018)
420 teraflops
Available via Google Cloud

전통적 Machine Learning vs. Deep Neural Network



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)

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

Jupyter Notebook 사용 방법

What is Jupyter Notebook ?

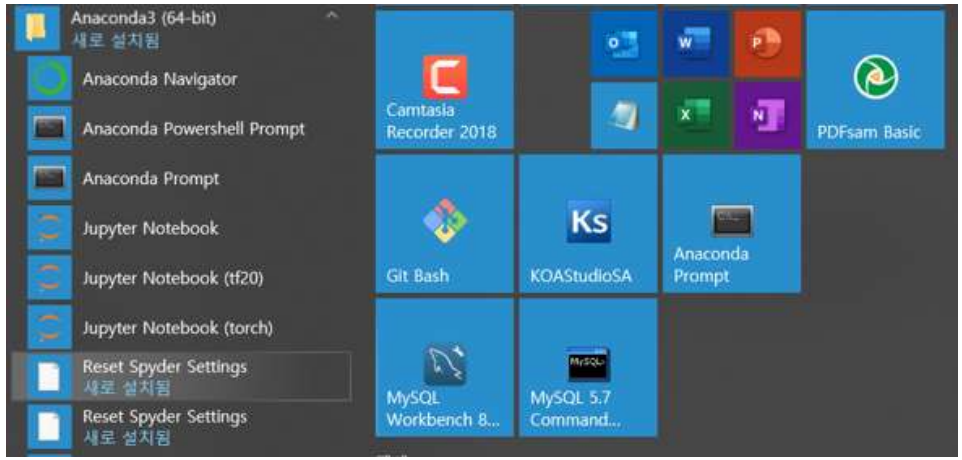
- 주피터 노트북(**Jupyter Notebook**)은 웹 브라우저에서 파이썬 코드를 작성하고 실행해 볼 수 있는 개발도구

- 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

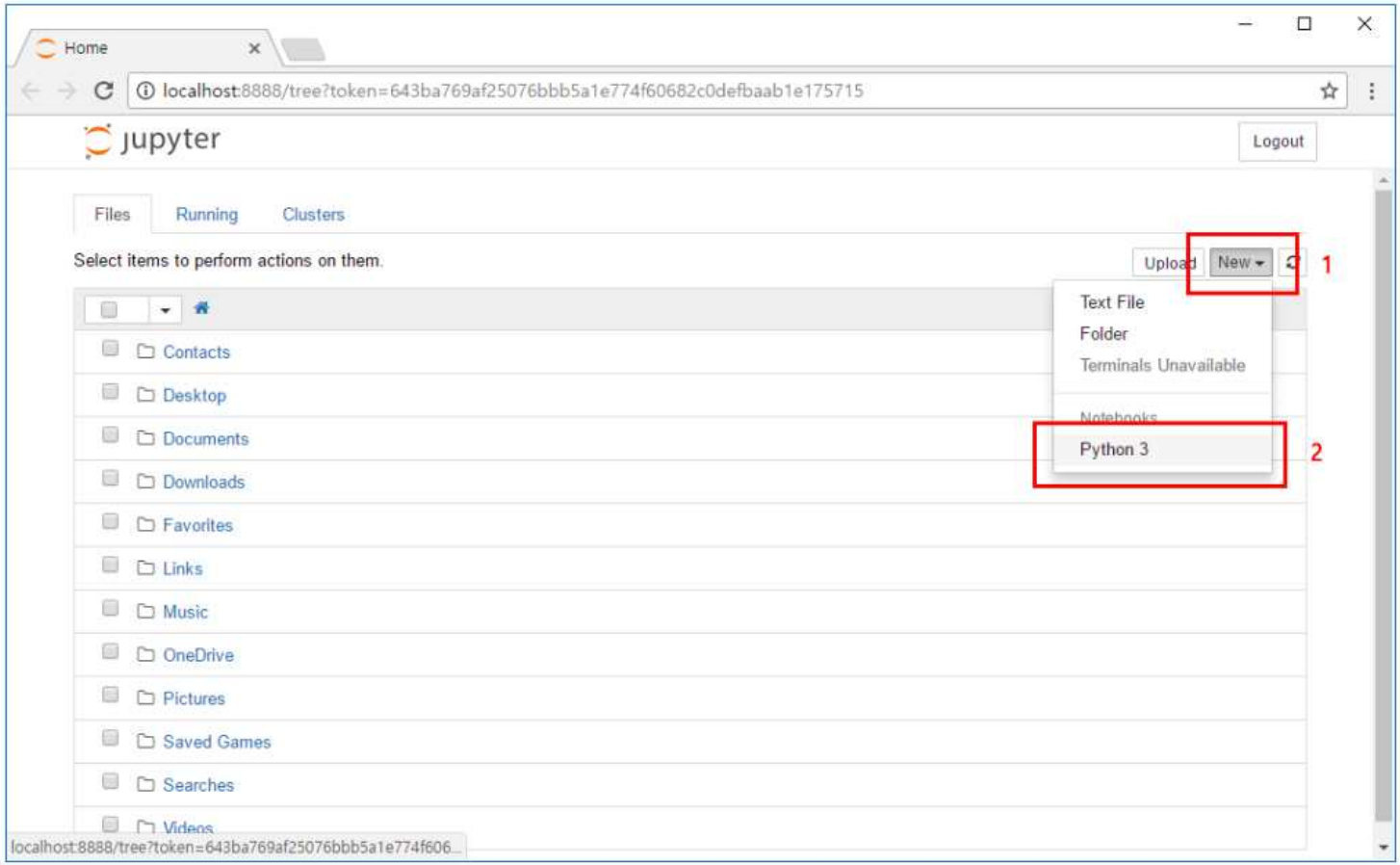


2) 명령 prompt 에서 실행
> jupyter notebook

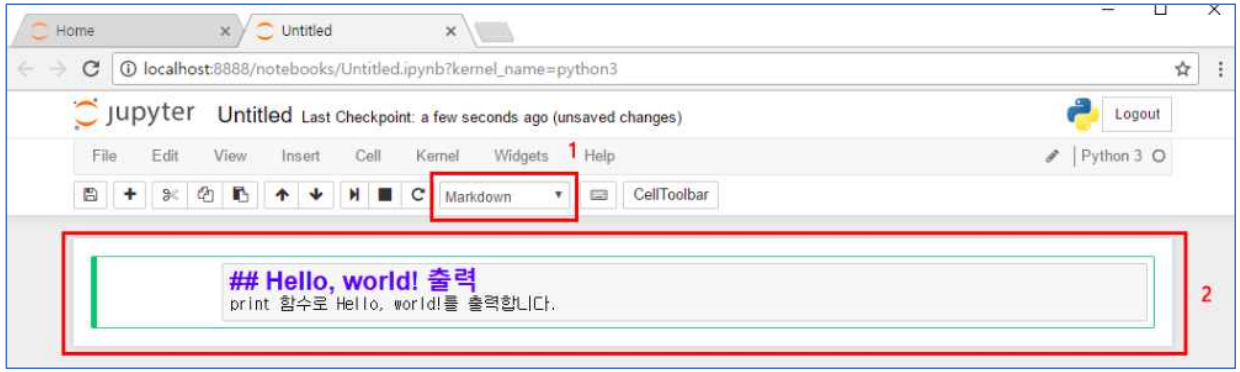
```
(base) C:₩Users₩trimu>jupyter notebook
[I 08:20:31.790 NotebookApp] Serving notebooks from local directory: C:₩Users₩trimu
[I 08:20:31.791 NotebookApp] The Jupyter Notebook is running at:
[I 08:20:31.793 NotebookApp] http://localhost:8888/?token=e1bb3f10db684cd21f9dd3c0e89f713306df271678b97d9d
[I 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

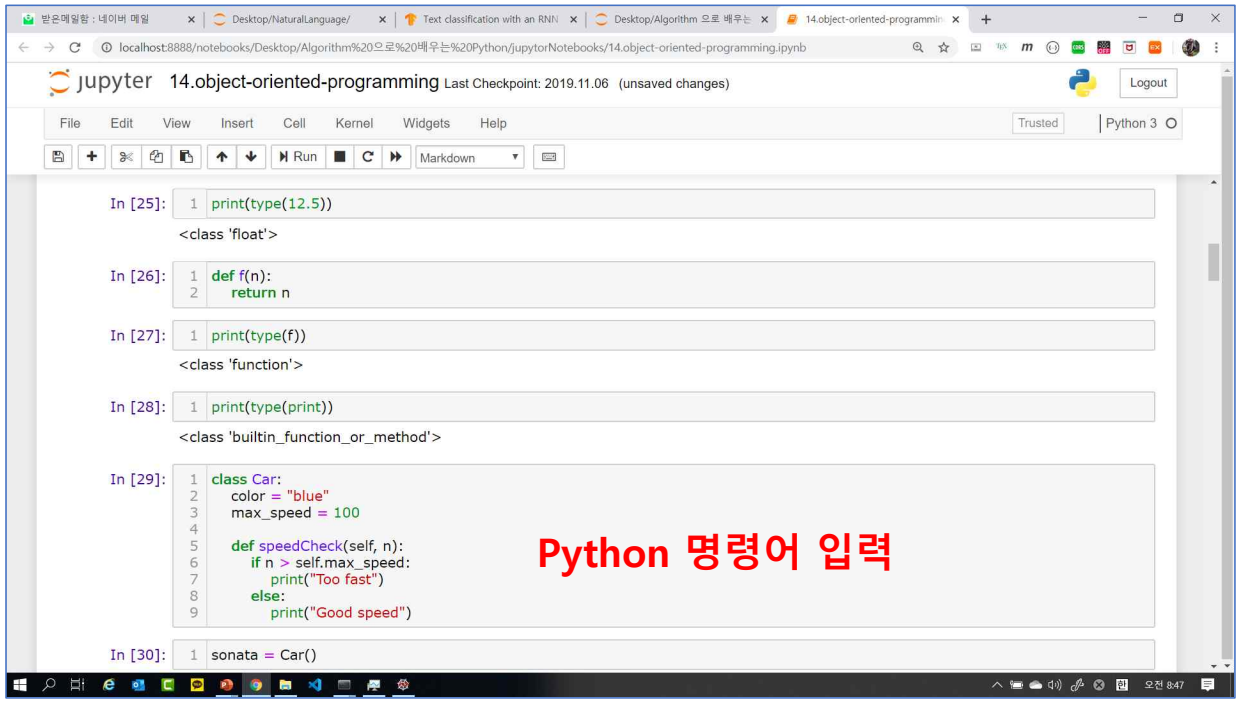
주피터 노트북 초기 화면 에서 New → Python 3 선택 → 새로운 notebook 생성



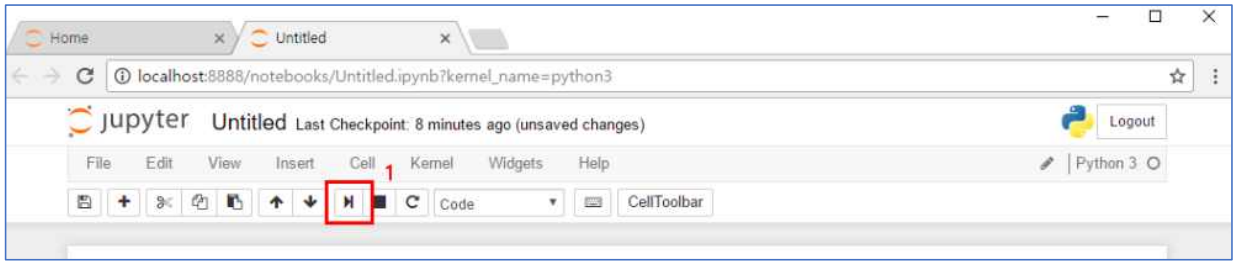
설명 추가



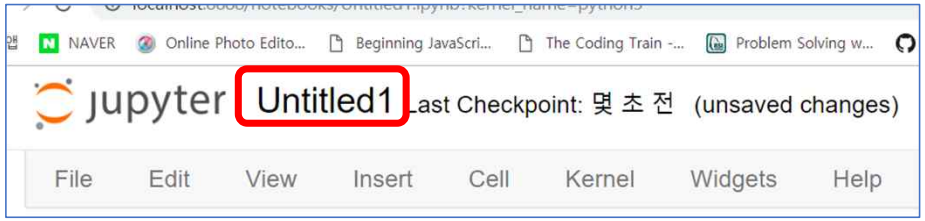
Python Code 입력
및
실행



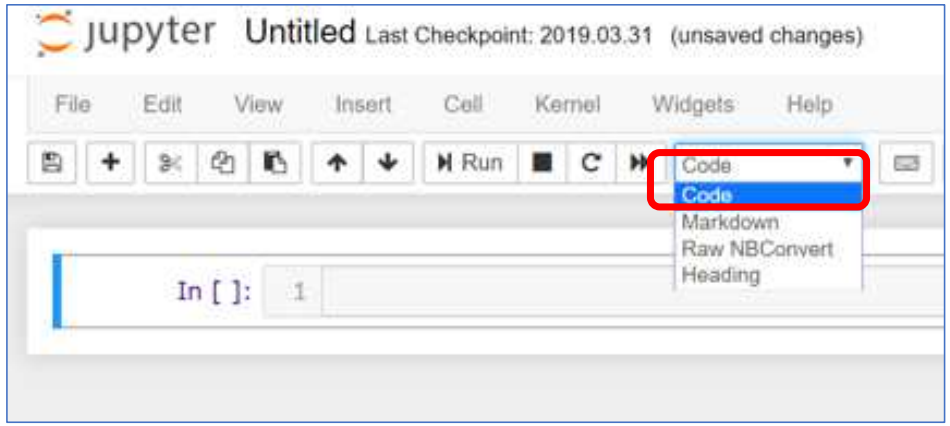
Code 실행



Notebook File Name 바꾸기



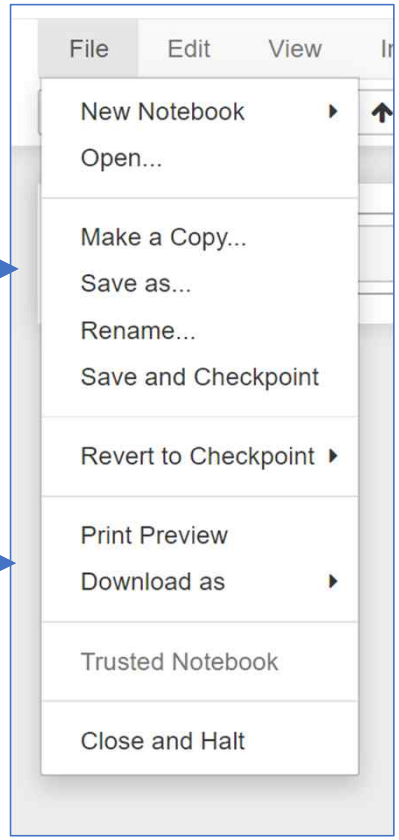
Cell Type 변경



자주 사용하는 menu 들

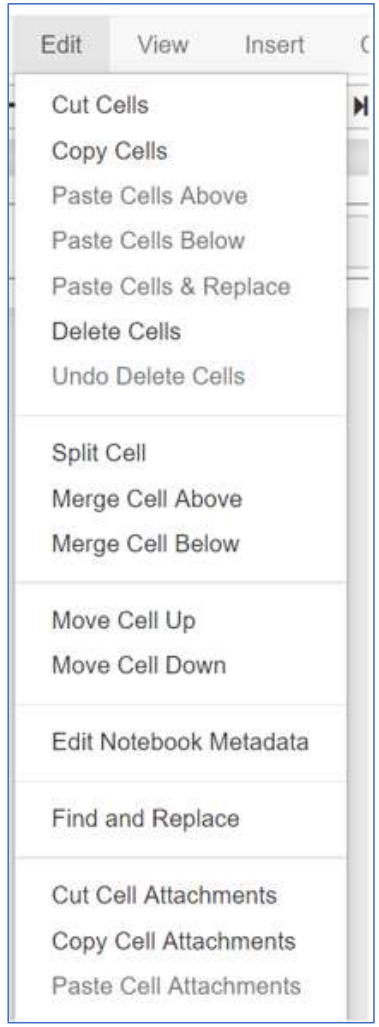
Notebook
저장

Notebook
download

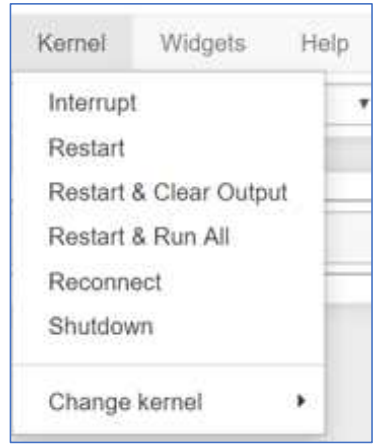


Cell 삭제
취소

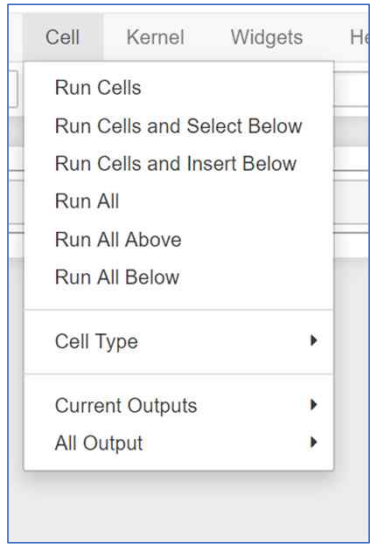
Find/
Replace



Notebook
restart



Cell 실행



Jupyter Notebook 사용방법

- 자주 사용하는 short-cut key

Shift + Enter : cell 실행 + 다음 cell 이동

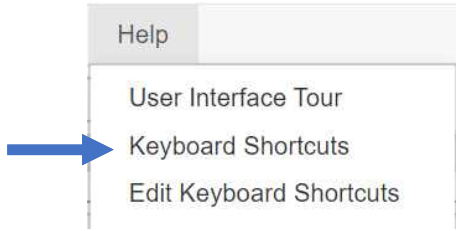
Ctrl + S : save + checkpoint

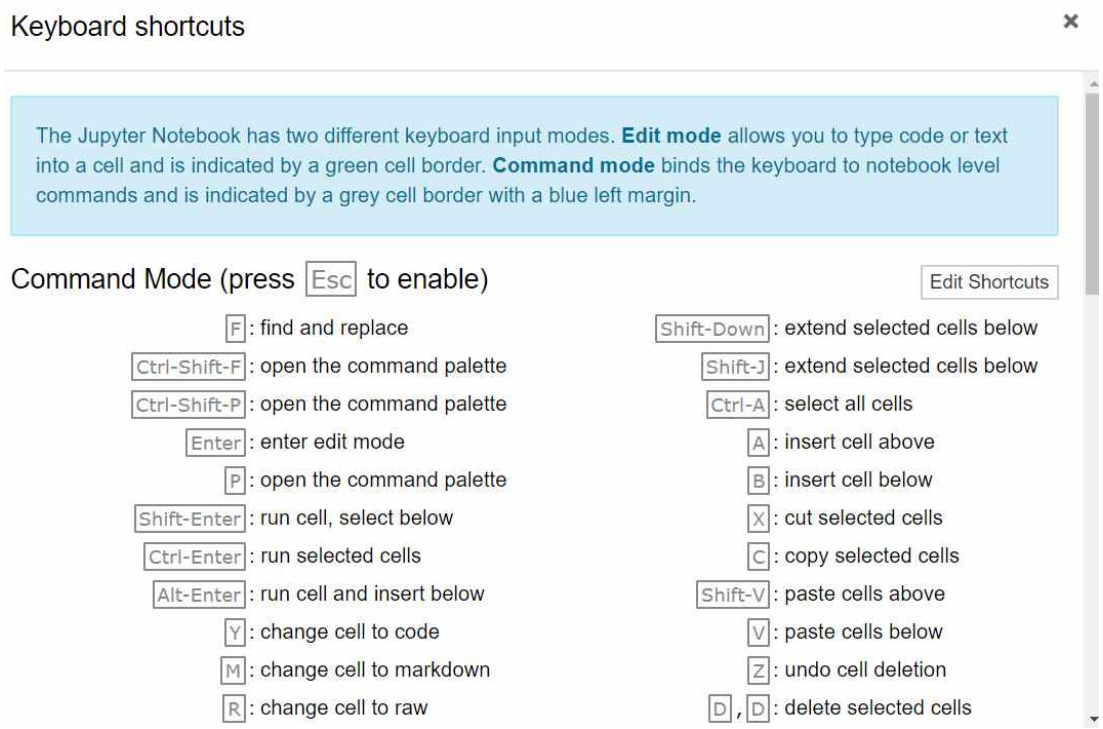
Esc+A : 위쪽에 cell 삽입
Esc+B : 아래쪽에 cell 삽입

Esc+X : cell 삭제

Esc+Z : cell 삭제 취소

Esc+M : cell 을 markdown type 으로 변경

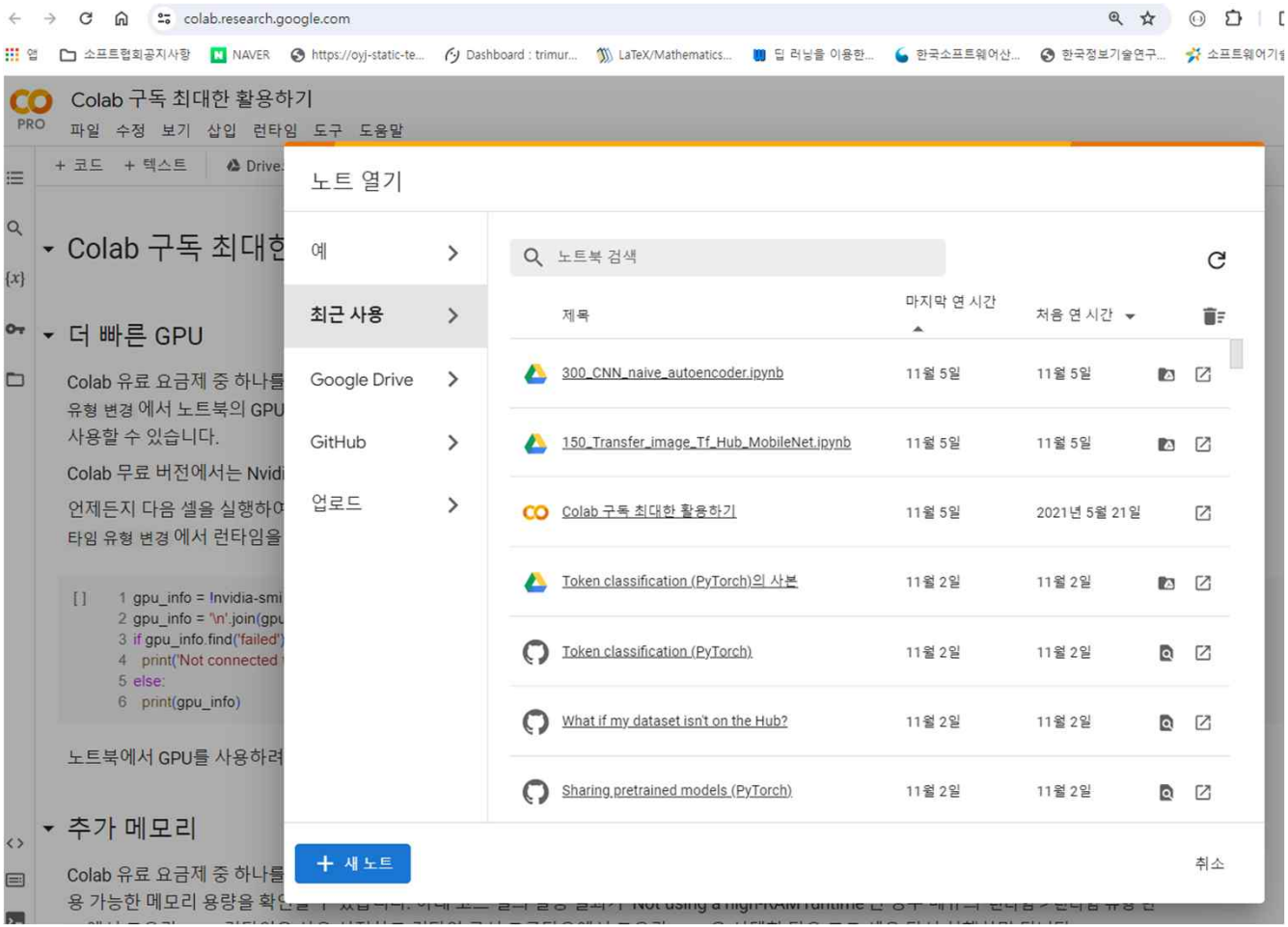




Google Colaboratory 소개

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

Google Colaboratory (https://colab.research.google.com/)



Crash Course

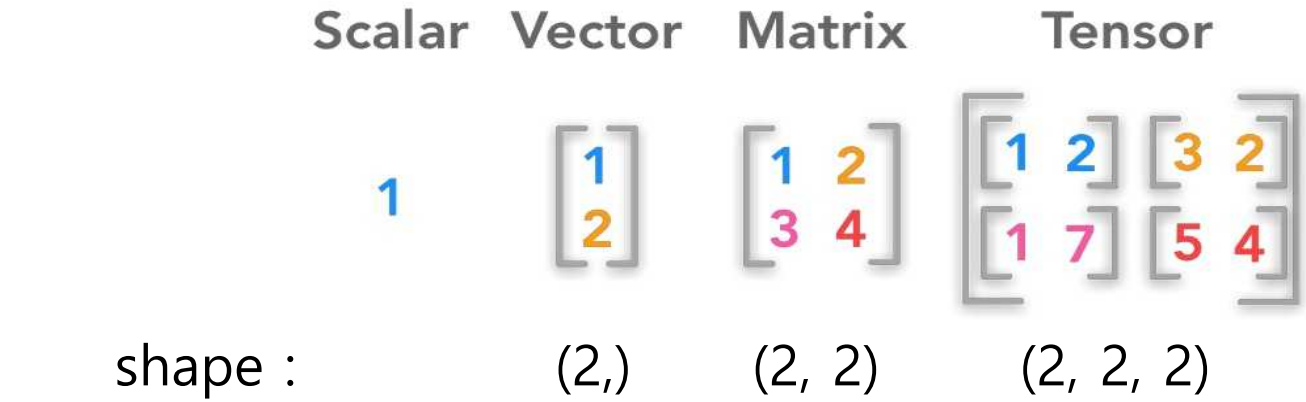
1. Numpy & Linear Algebra
2. Pandas
3. Matplotlib

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 (Numerical Python)

- 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)

Matrix 의 numpy 표현

1D array(Vector)

8	5	3

np.array([8, 5, 3]), shape=(3,)

2D array(Matrix)

axis=0

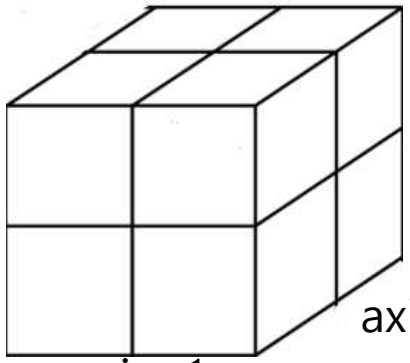
8	5	3
1	2	9

axis=1

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

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

3D array(Tensor)



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

shape=(2, 2, 2)

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

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

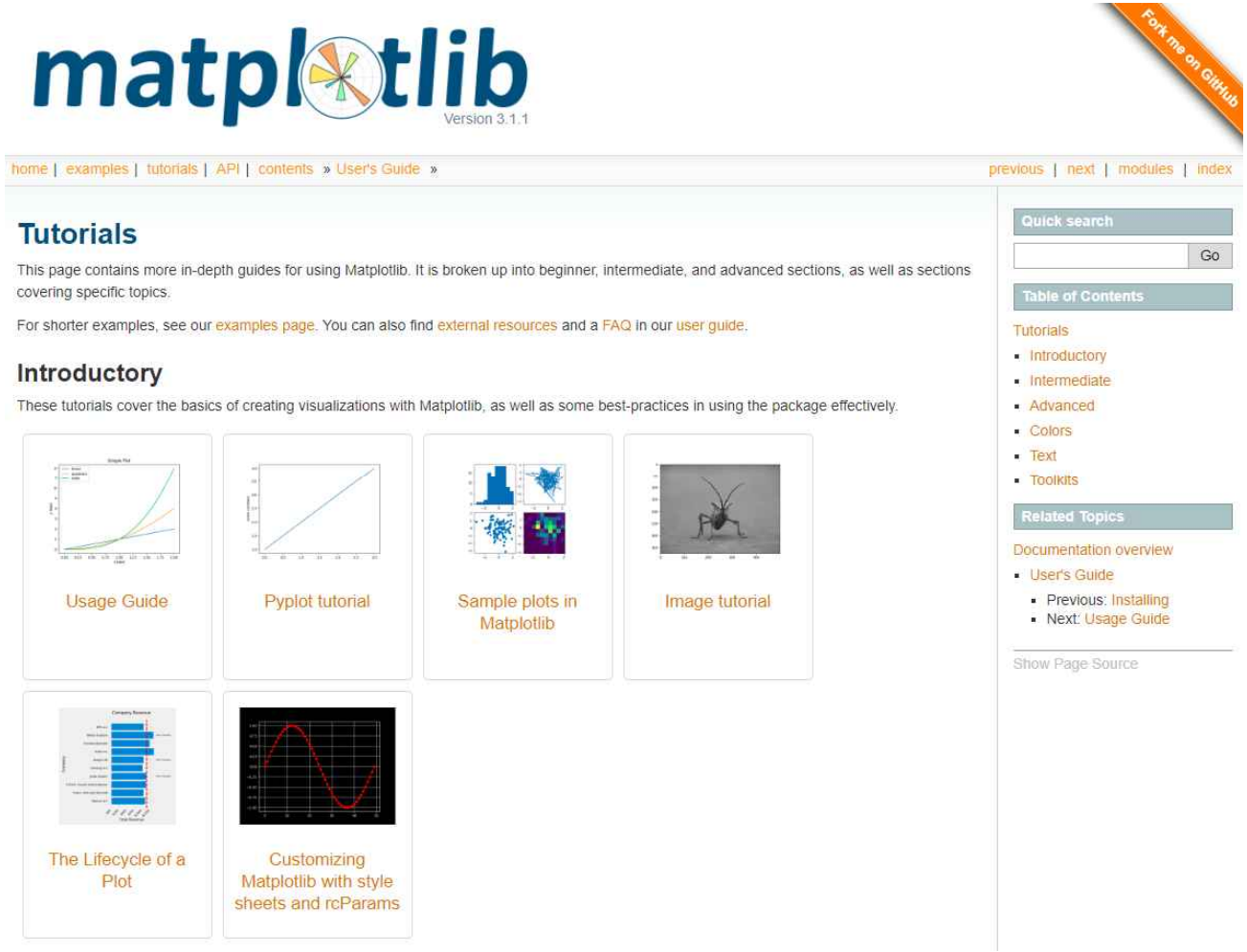
Pandas

- 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

Matplotlib – 대표적 시각화 도구



실습:

- 001. Numpy Array Crash
- 002. Pandas Crash
- 003. Matplotlib Crash

Feature Scaling

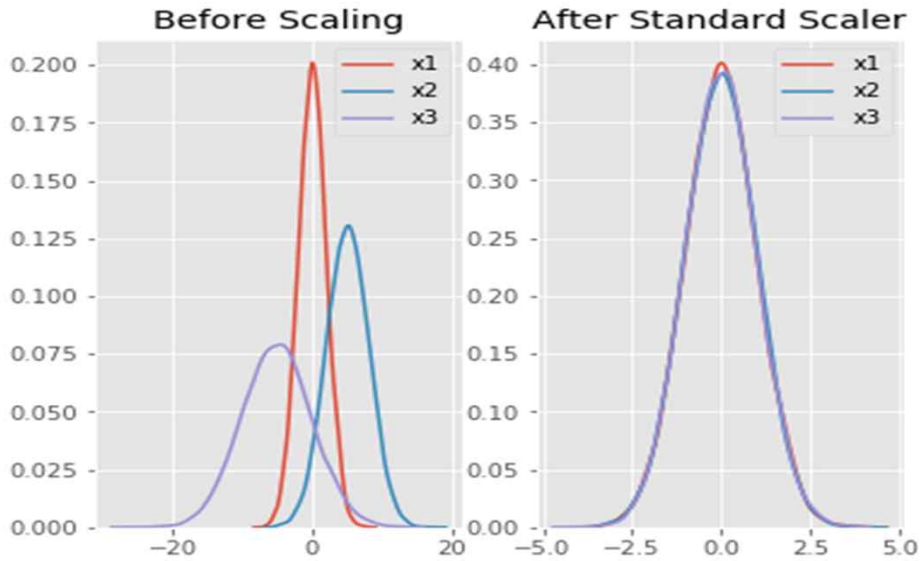
- Raw data 를 전 처리하여 input data 의 구간을 표준화

- Standard Scaling :

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

- Minmax Scaling :

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



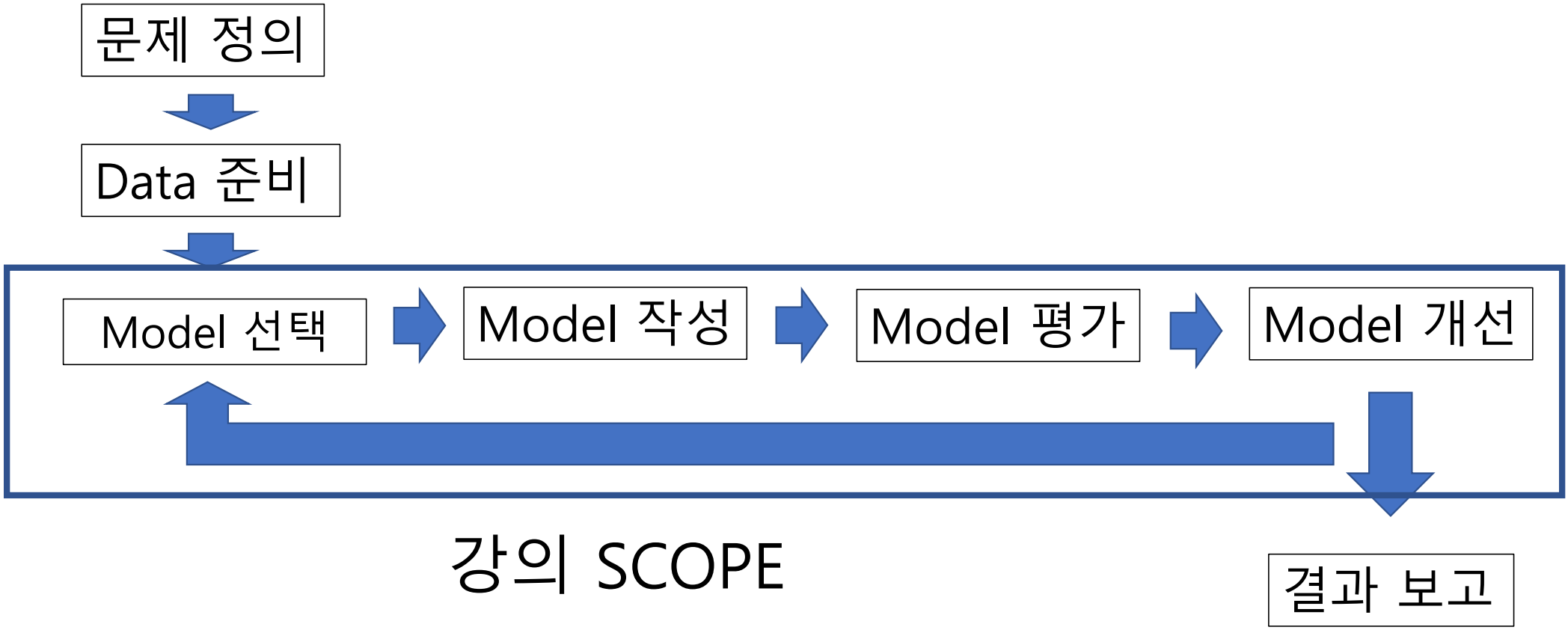
실습: 004. sklearn 을 이용한 feature scaling (normalization)

- Simple Feature Scaling
- Standard Scaling
- MinMax Scaling

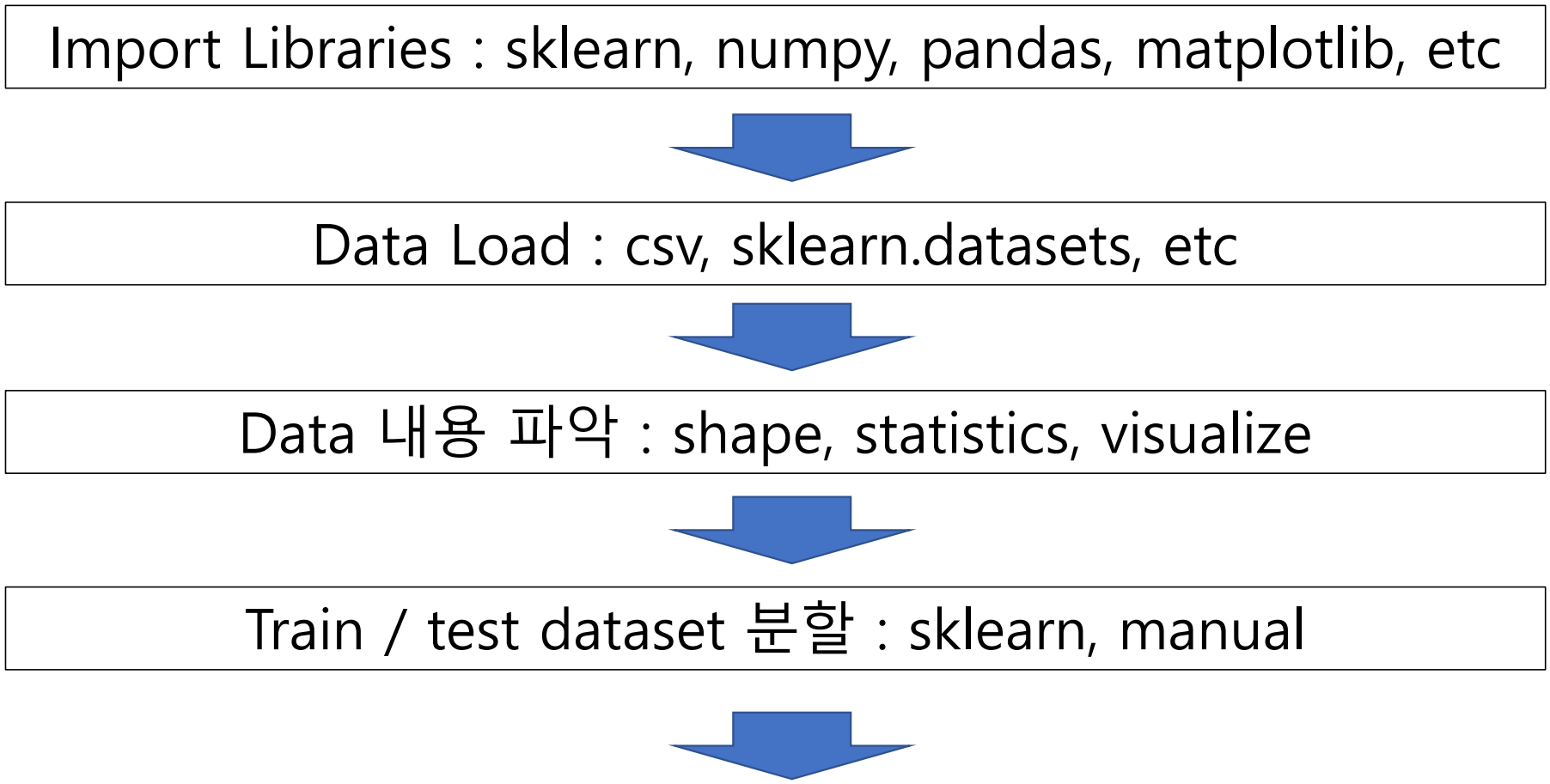
Machine Learning

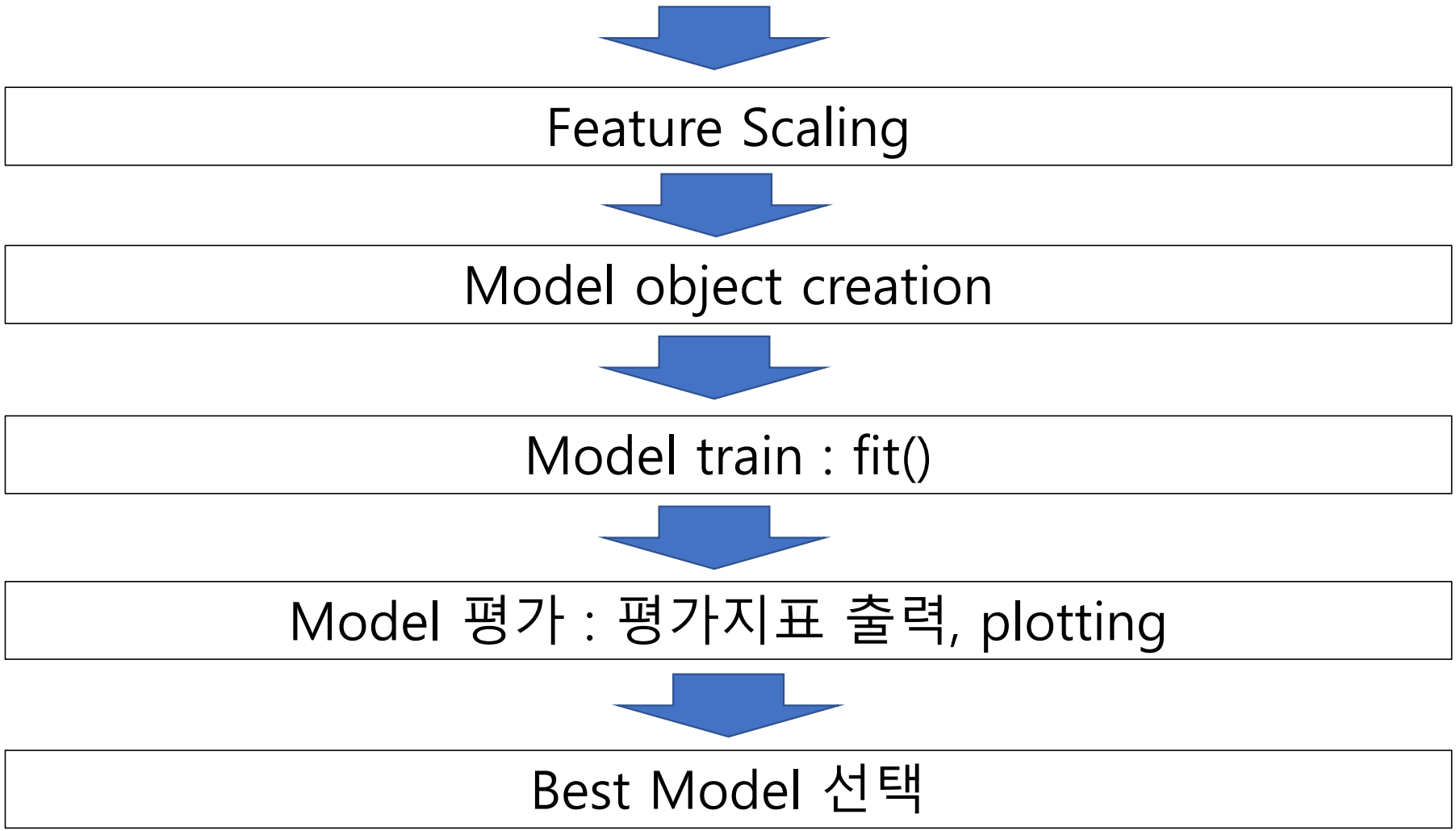
End-to-End Process

Machine Learning Model 작성



Model 작성 순서

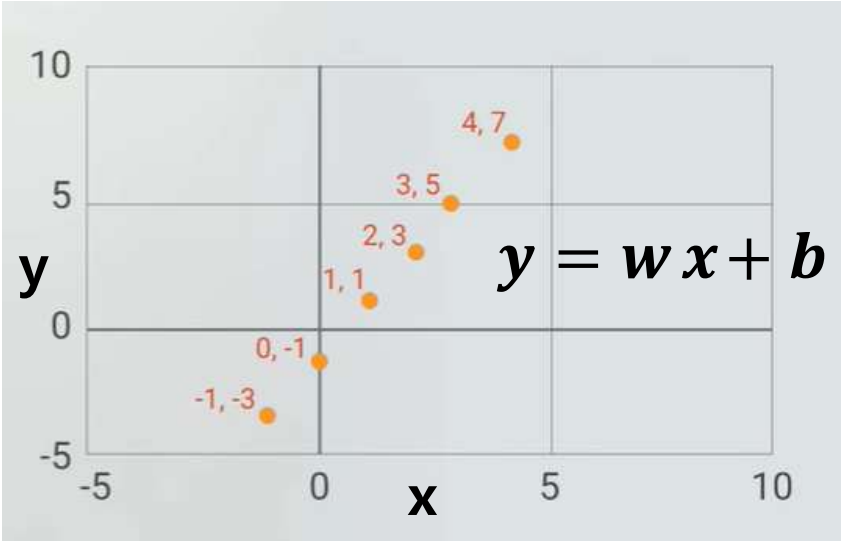

Import Libraries : sklearn, numpy, pandas, matplotlib, etc
Data Load : csv, sklearn.datasets, etc
Data 내용 파악 : shape, statistics, visualize
Train / test dataset 분할 : sklearn, manual


Feature Scaling
Model object creation
Model train : fit()
Model 평가 : 평가지표 출력, plotting
Best Model 선택

Linear Regression

1. Univariate Linear Regression (단변수 선형회귀)

- 한개의 변수로 결과 예측 (ex. 혈압으로 당뇨병 여부 예측)



- x, y 가 주어지고
  w, b 가 미지수

- **w, b 를 infer (추정)**

1. Univariate Linear Regression (단변수선형회귀)

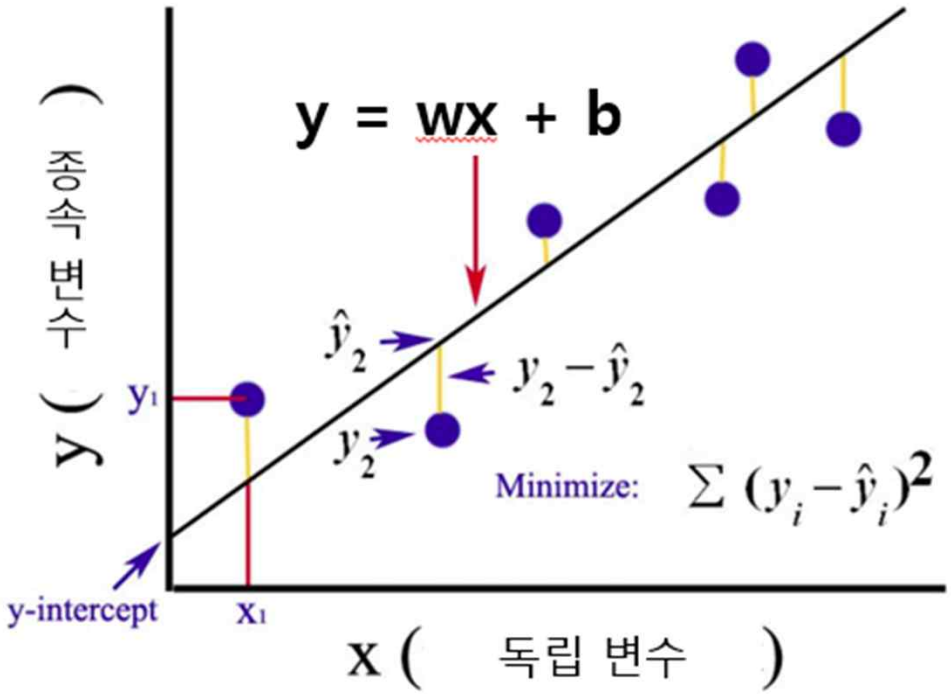
y = wx + b → **Hypothesis (가설)**

- OLS (Ordinary Least Squares, 최소자승법)

$$Minimize \quad \sum_{i=1}^{n} (true \quad - prediction \quad)^2$$

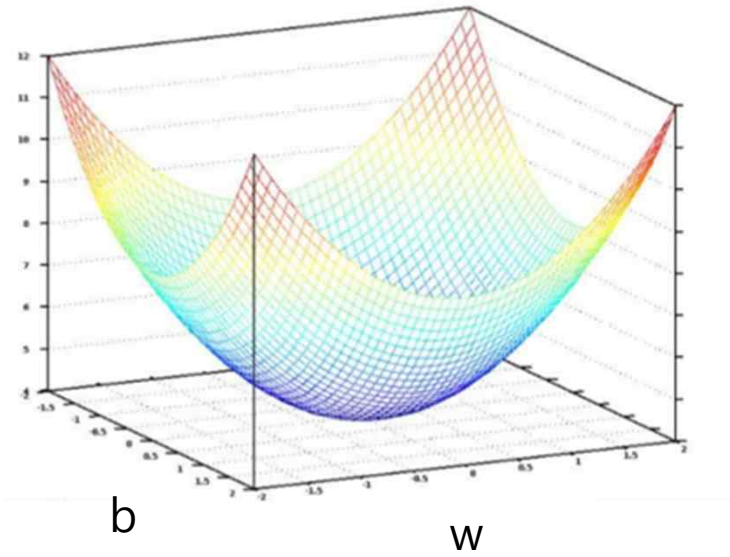
Cost Function (비용함수)

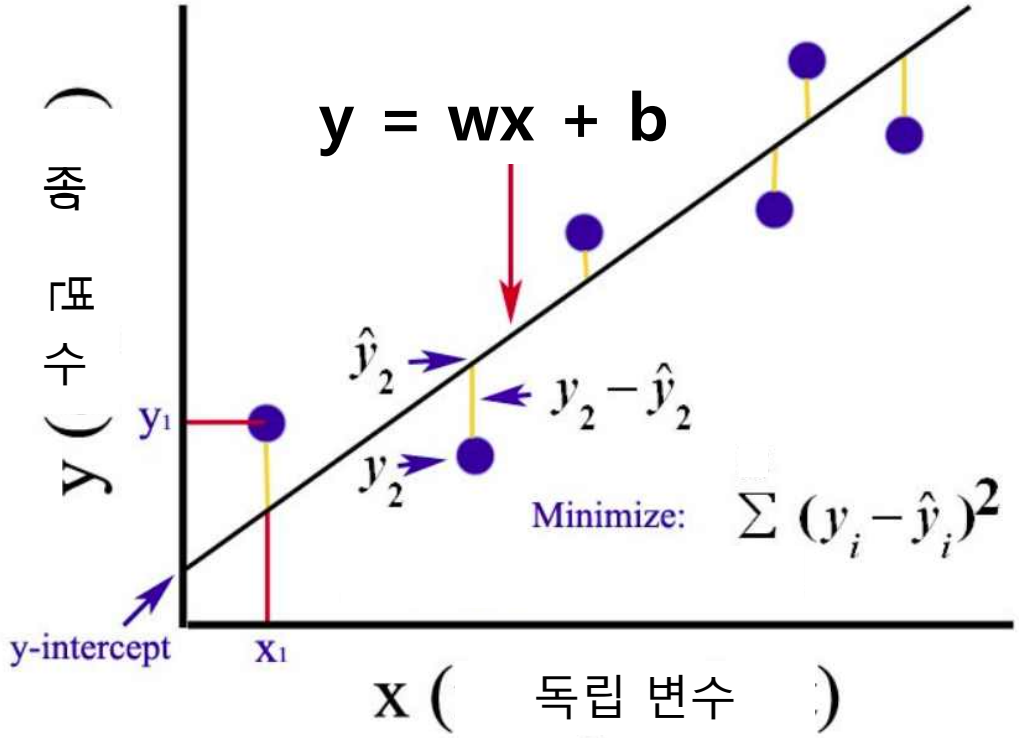
- 가설이 얼마나 틀렸는지 측정



Cost Function - Linear Regression

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





MSE 를 최소화 하는
θ 와 b 를 optimize

Linear Regression 의 Accuracy (정확도) 측정

- 평가기준 2 : R2 score (결정계수)
  sklearn.metrics.r2_score

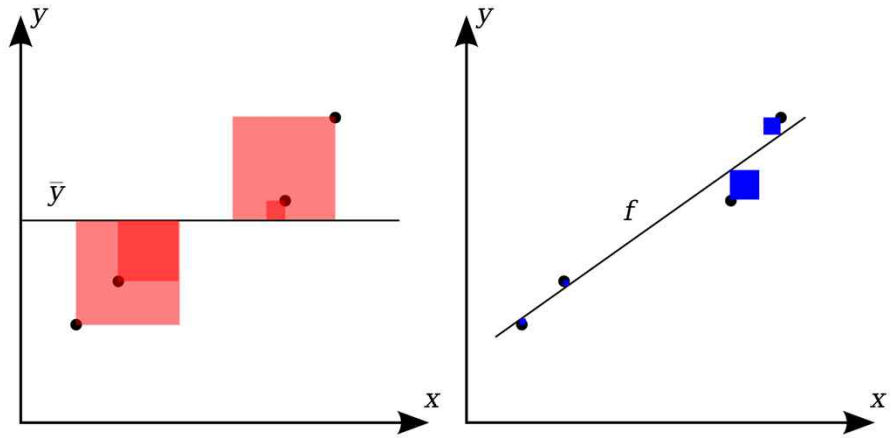
* 결정계수(R2) – 회귀식의 정확도 측정

$$0 \leq R2 \leq 1$$

$$R2 = 1 - SSE / SST$$

$$= 1 - \left(\frac{Sum\ of\ Square\ \ Error}{Sum\ of\ Square\ \ Total}\right)$$

$$= 1 - \left(\frac{\text{예측값에 대한 분산의 합}}{\text{분산의 합}}\right)$$



R2 ≈ 0 R2 ≈ 1

2. Multivariate Linear Regression (다변수선형회귀)

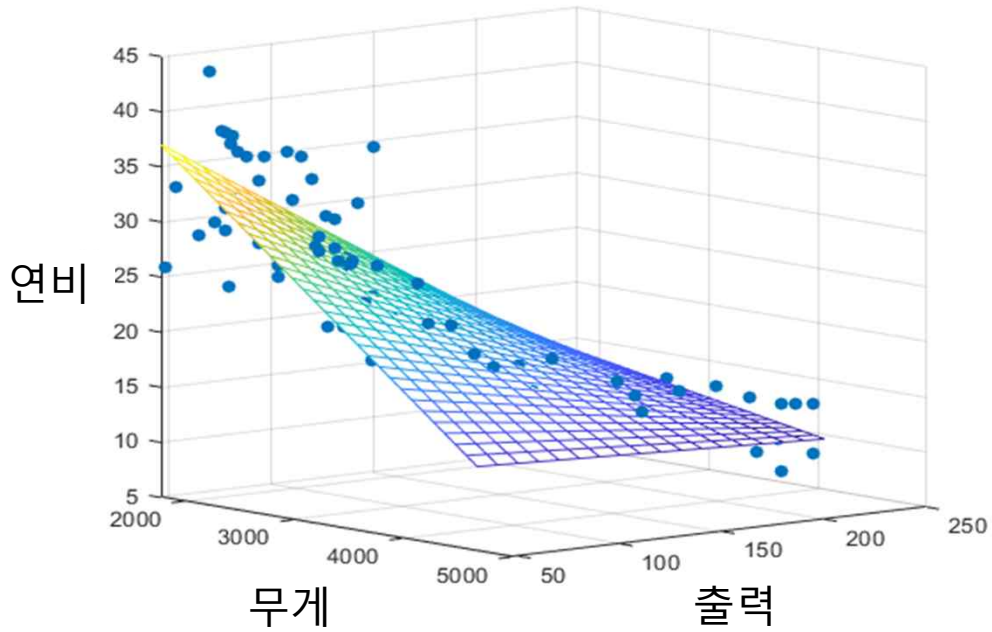
$$\hat{Y} = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 + \ldots + \theta_n X_n$$

$$\hat{Y} = \theta X$$

sklearn.linear_model.LinearRegression()

θ = coef_

θ_0 = intercept_



실습: 010. 당뇨병 data 를 이용한 선형회귀

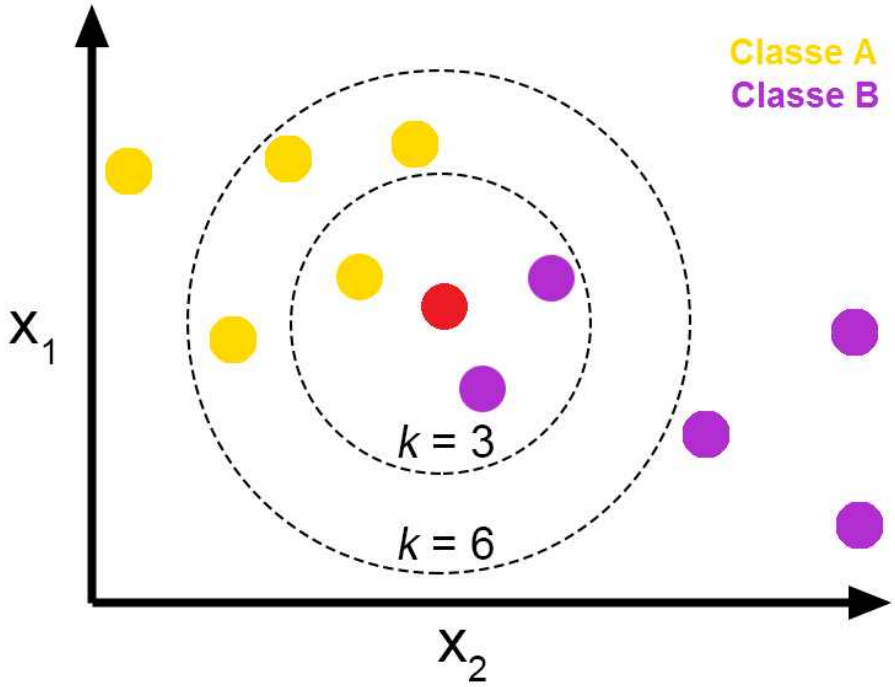
- Dataset : sklearn.datasets.load_diabetes()
- Feature : 나이, 성별, 체질량지수, 혈압, 6가지 혈청 수치 → scaling 되어 있음
- Target : 1년 뒤 측정한 당뇨병의 진행률
- Model : linear_model.LinearRegression()

Classification

K-Nearest Neighbors

KNN (K-Nearest Neighbors, K 최근접 이웃)

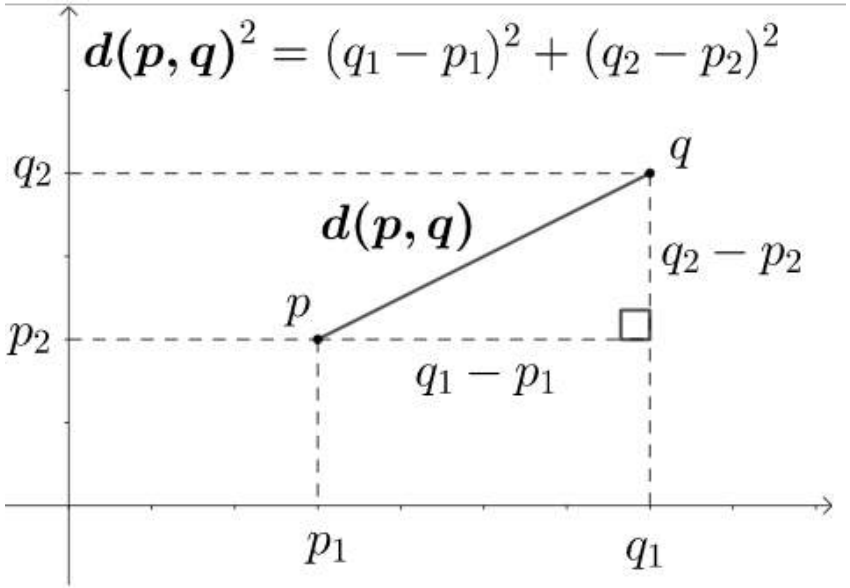
- 다른 observation (관측치, X data) 과의 유사성에 따라 분류 (classify)
- 서로 가까이 있는 data 들을 "이웃" (neighbor) 이라고 부른다.
- 가까이 있는 이웃의 label 들 중 가장 많은 것을 unknown case 의 prediction 으로 응답한다.
- 장점 : simple and easy to implement
- 단점 : dataset 이 커지면 slow.
  outlier/missing value 의 영향이 크다.



KNN 알고리즘

1. K 값을 선택한다.

2. Unknown case 와 모든 data point 간의 거리를 계산한다.



$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

3. Training dataset 에서 unknown data point 와 가장 가까이 있는 K 개의 관측치(observation) 을 선택한다.

4. K 개의 nearest neighbors 의 label 중 가장 많은 것을 unknown data point 의 **class** 로 **분류**한다 (→ classification)

   K 개의 nearest neighbors 의 label value 의 평균을 predicted **value** 로 **계산** 한다 (→ regression)

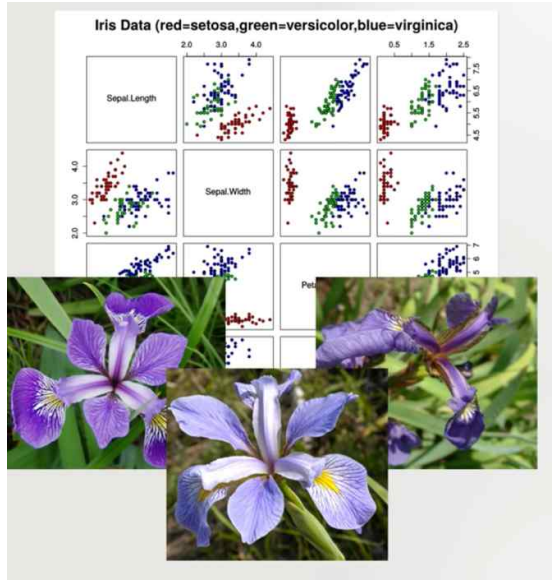
실습 : 020. KNN(K-Nearest Neighbors, 최근접 이웃)

1. sklearn 에서 제공하는 iris (붓꽃) 분류 dataset 사용 :

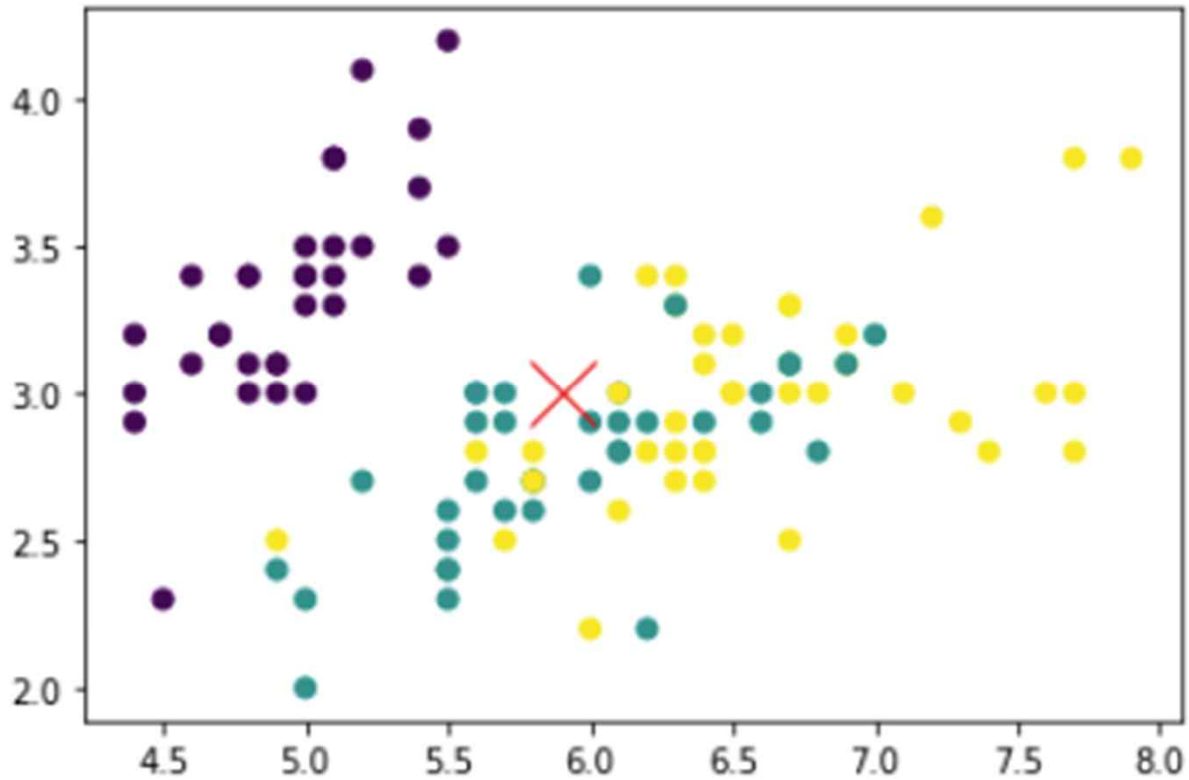
   꽃잎의 각 부분의 너비와 길이 등을 측정한 데이터이며 150개의 레코드로 구성

2. Data 의 내용 :

   Sepal Length : 꽃받침 길이
   Sepal Width : 꽃받침 너비
   Petal Length : 꽃잎 길이
   Petal Width : 꽃잎 너비



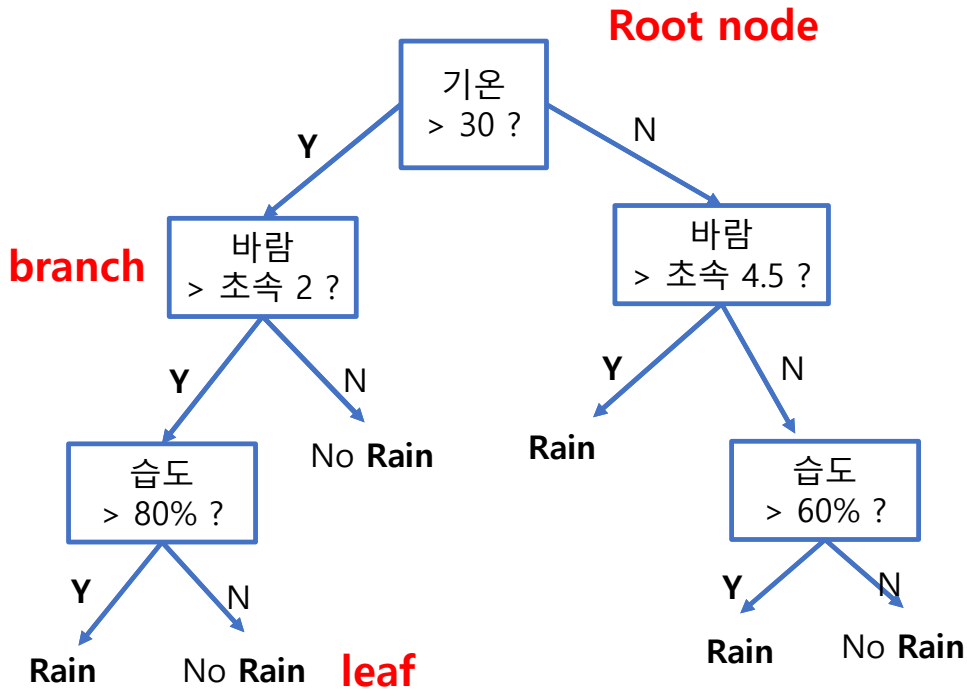
3. 시각화



Decision Tree

Decision Tree (결정나무)

- 모든 가능한 결정 경로(Decision Path)를 tree 형태로 구성
- 각 node 는 test 를 의미
- 각 branch 는 test 의 결과에 해당
- 각 leaf node 는 classification 에 해당
- 장점 : **white-box model**
  data preprocessing 불필요
- 단점 : **overfitting 되기 쉽다.**
  훈련 데이터의 작은 변화에도 매우 민감



Decision Tree 알고리즘의 종류

1. ID3 – 기본적 알고리즘. 정보이득(Information Gain) 을 이용한 트리 구성

2. CART (Classification and Regression Tree)

    – Gini 불순도에 기반한 트리 구성

3. C4.5, C5.0 – ID3 개선

4. 기타 – CHAID, MARS

엔트로피 (Entropy), 정보 이득(information gain)

- 엔트로피(Entropy) - 주어진 데이터 집합의 혼잡도. 즉, 우리가 가지고 있지 않은 정보의 양을 의미.

- 주어진 데이터 집합에서 서로 다른 종류의 레코드들이 섞여 있으면 엔트로피가 높고, 같은 종류의 레코드들이 섞여 있으면 엔트로피가 낮음.

- 우리가 시스템에 대해 알게 될수록 시스템의 엔트로피는 감소. 예를 들어 데이터의 통계를 알게 되면 엔트로피는 감소. 이것을 정보 이득(information gain)이라고 한다.

- 엔트로피 값은 0에서 1사이의 값. 가장 혼합도가 높은 상태의 값이 1이고, 반대는 0

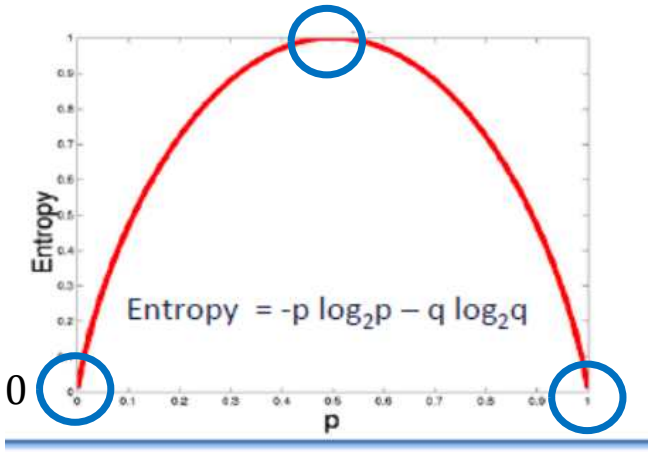
- Decision Tree 에서는 엔트로피가 높은 상태에서 낮은 상태가 되도록 데이터를 특정 조건을 찾아 나무 모양으로 구분해 나감.

$$Entropy = -\sum_{i=1}^{m} p_i \log_2(p_i), \quad p_i = \frac{freq(C_i, S)}{|S|}$$

(S: 주어진 데이터들의 집합, C: 레코드(클래스) 값들의 집합, freq(Ci,S): S에서 Ci에 속하는 레코드의 수, |S|: 주어진 데이터들의 집합의 데이터 개수)

p = 0.5, Entropy = 1

$0 * \log_2 0 = 0$



$1 * \log_2 1 = 0$

Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

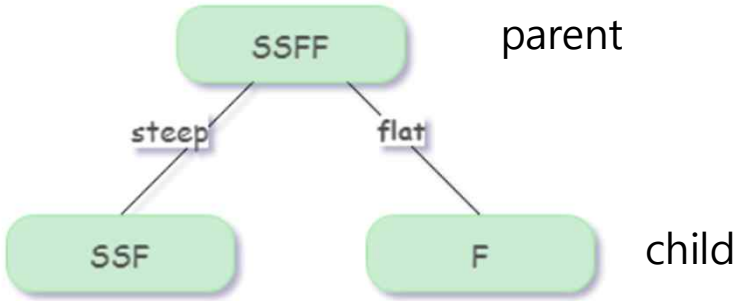
$$\log_2 0.5 = \log_2 \frac{1}{2} = -\log_2 2 = -1$$
$$\log_a 1 = 0$$
$$\log_a a = 1$$

$$\log a^{MN} = \log a^M + \log a^N$$
$$\log a^{M/N} = \log a^M - \log a^N$$
$$\log a^{M^p} = p \log a^M$$

- Information Gain = Entropy(Parent) – (weight) * Entropy(Child)

ex)

Feature			Label
경사도	노면상태	속도제한	속도
steep	bumpy	Yes	slow
steep	smooth	Yes	slow
flat	bumpy	No	fast
steep	smooth	No	fast



- "경사도"의 information gain 계산:

Entropy(Parent) = - {0.5 log2(0.5) + 0.5 log2(0.5)} = 1

Entropy(Child/steep) = - {0.667 log2(0.667) + 0.334 log2(0.334)} = 0.918
Entropy(Child/flat) = - {0 + 1 log2(1)} = 0
Entropy(가중평균) = ¾*0.918 + ¼*0 = 0.688

- Information Gain = 1 – 0.688 = 0.312

Decision Tree 알고리즘 (ID3)

1. Initial open node 를 생성하고 모든 instance 를 open node 에 넣는다.

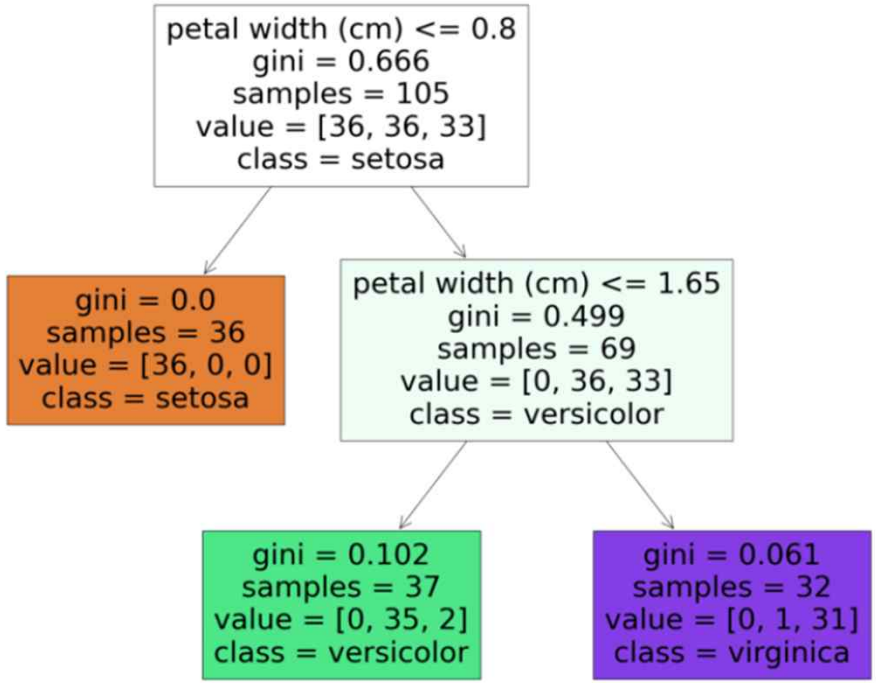
2. Open node 가 없어질 때까지 loop
   - 분할할 open node 선택
   - information gain 이 최대인 attribute(feature) 선택
   - 선택된 attribute 의 class (Y, N) 별로 instance sort
   - sort 된 item 으로 새로운 branch 생성
   - sort 된 item 이 모두 하나의 class 인 경우 leaf node close

실습 : 030. Decision Tree 작성 및 시각화

1. KNN 에서 사용하였던 Iris data 사용

2. Tree 의 max_depth = 2,
   None 으로 변경하여 test/비교

3. visualization

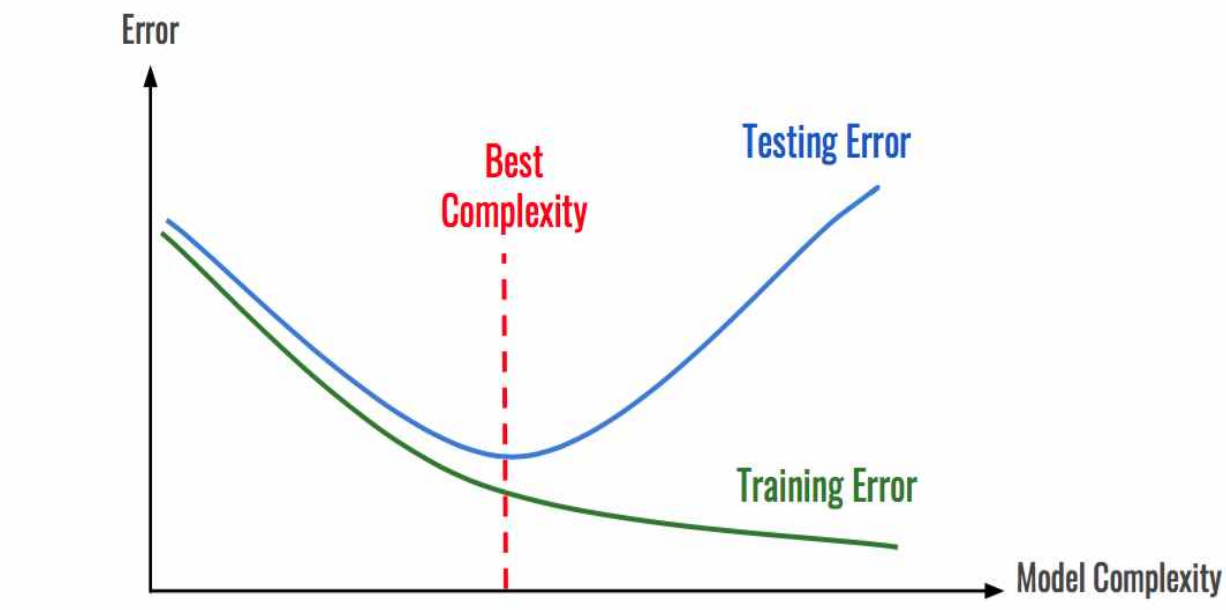


Training/Test/Evaluation

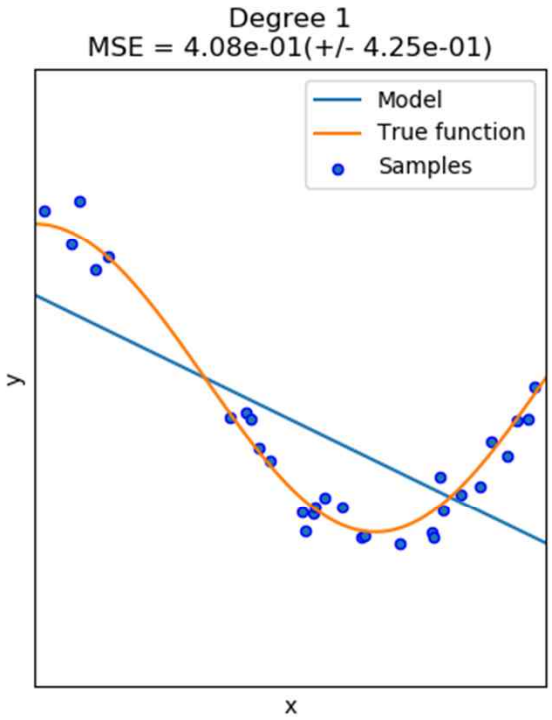
Overfitting(과적합) 과 Underfitting(과소적합)

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

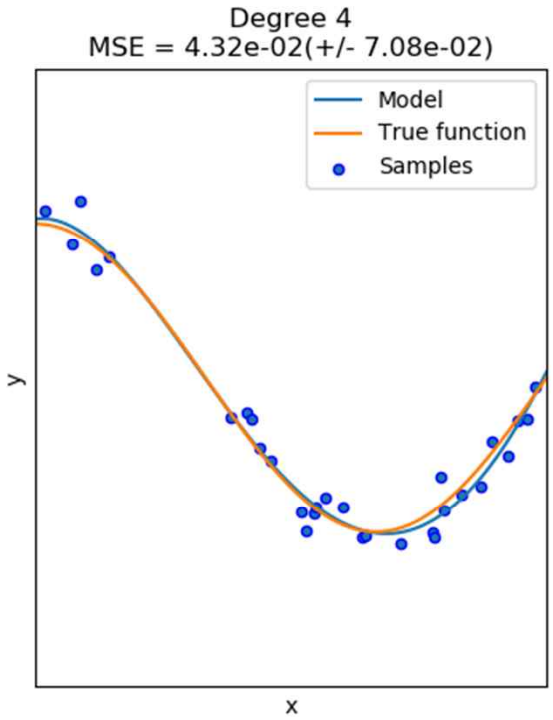
- 반대로 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못하는 경우 과소적합 (Underfitting) 이라고 한다.



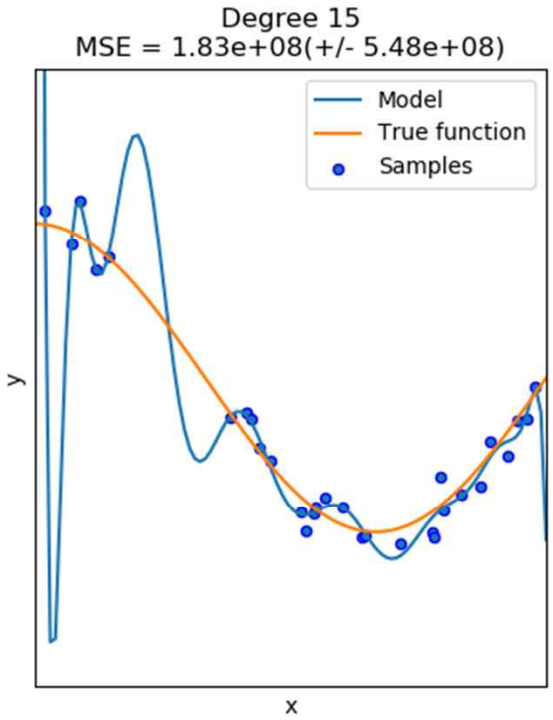
Underfitting
(과소적합)



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

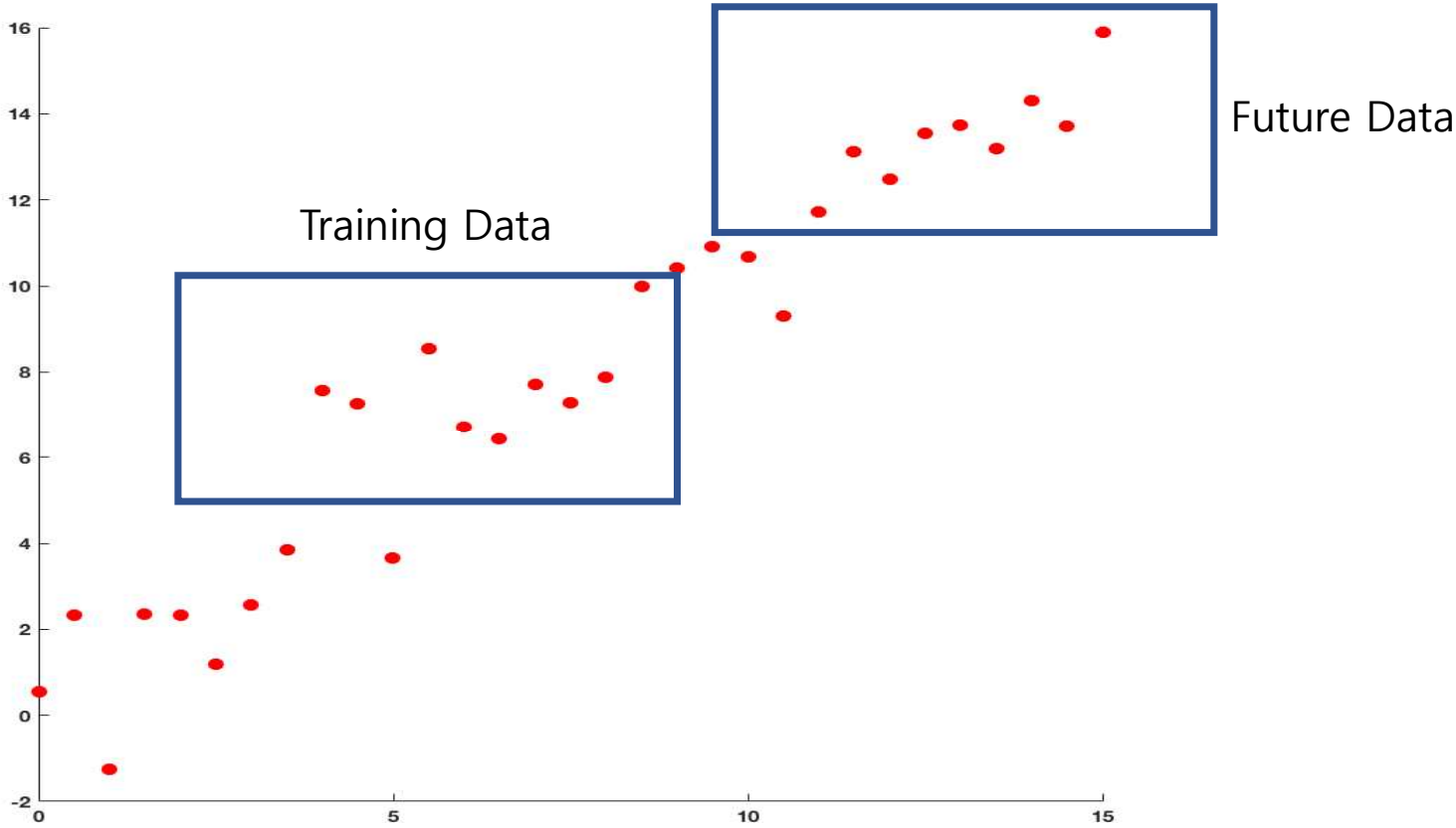
- ML의 세가지 Error Source

  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.

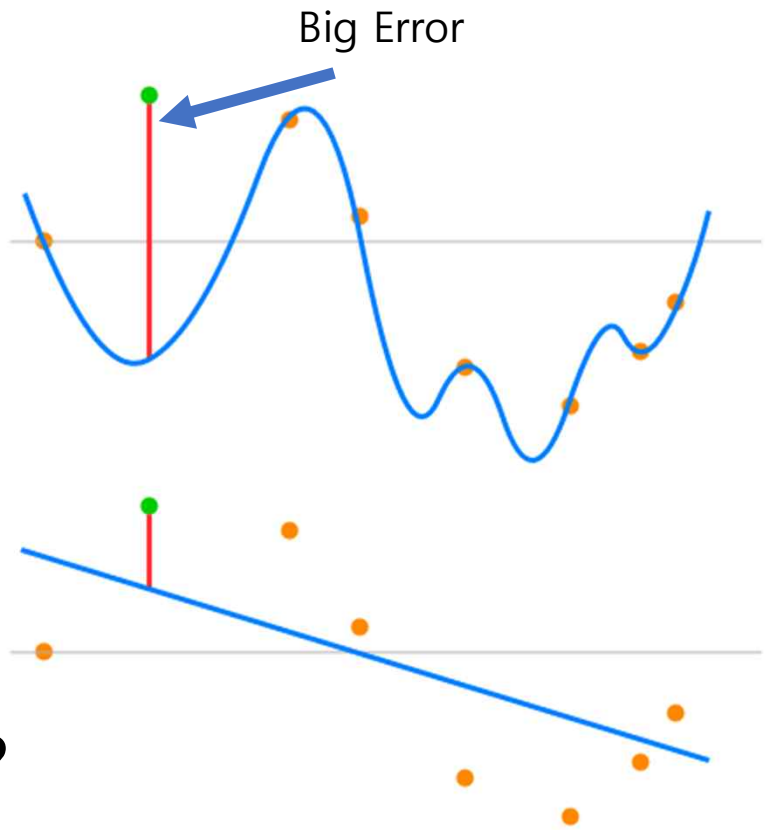
  → Bias-Variance Dilemma (Bias-Variance Trade-off)

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 ?

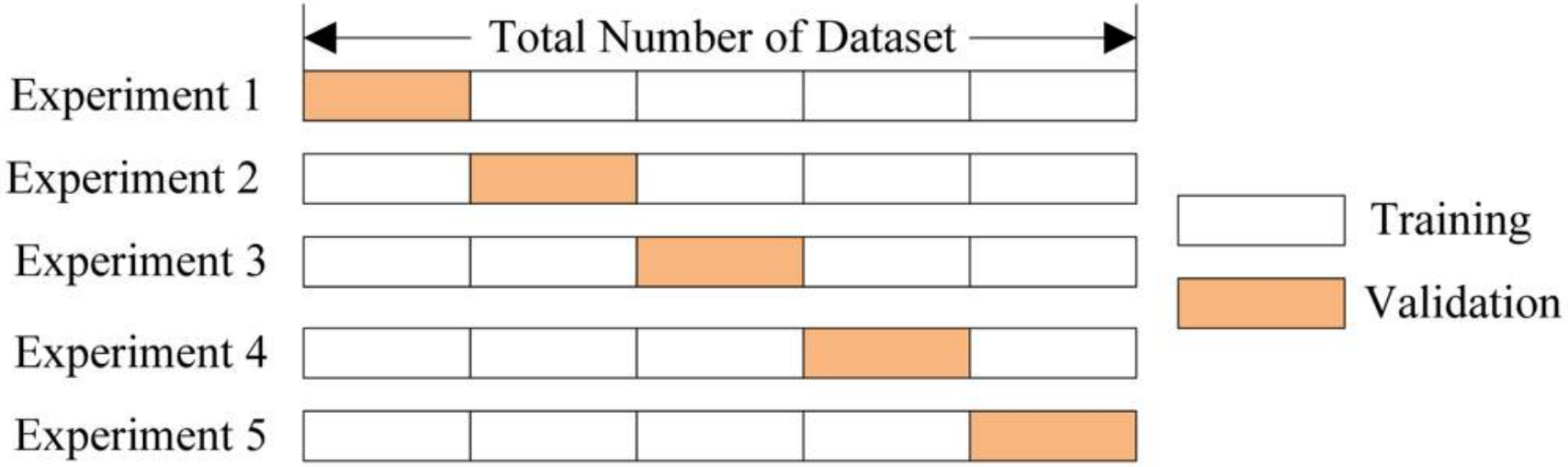
- Variance – infinite data sampling 으로 해결 가능
- Bias – true function 을 알면 해결 가능
- **BUT,** 현실에서는 infinite data sampling 도 할 수 없고 true function 도 알 수 없으므로 간접적 방법을 사용
  1. Cross Validation
  2. Precision / Recall / F1-Score

Training & Testing & Cross-Validation Set

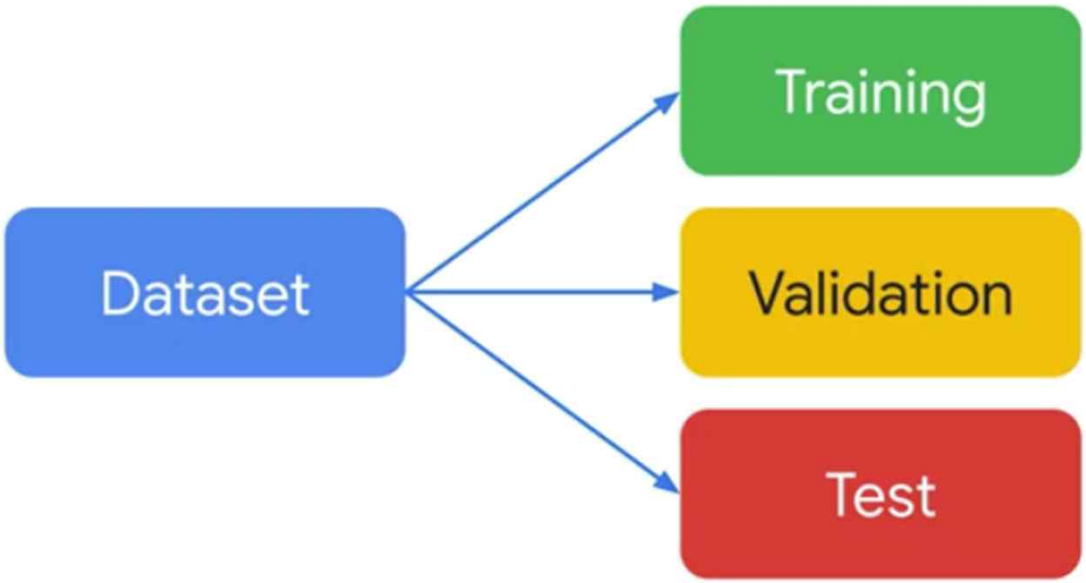
- Training Set
  - parameter 를 inference 하는 procedure 에 사용하는 data
  - 보지 못한 Data 의 분포가 Training set 과 상이할 경우 문제
  - Training set 내에서 cross-validation set 을 구성 (Data 가 불충분한 경우)
- Testing Set
  - 학습한 Machine Learning Model 을 Test 하기 위해 사용하는 data
  - 미래의 instance 인 것처럼 간주
- Training set 과 Testing set 은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

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

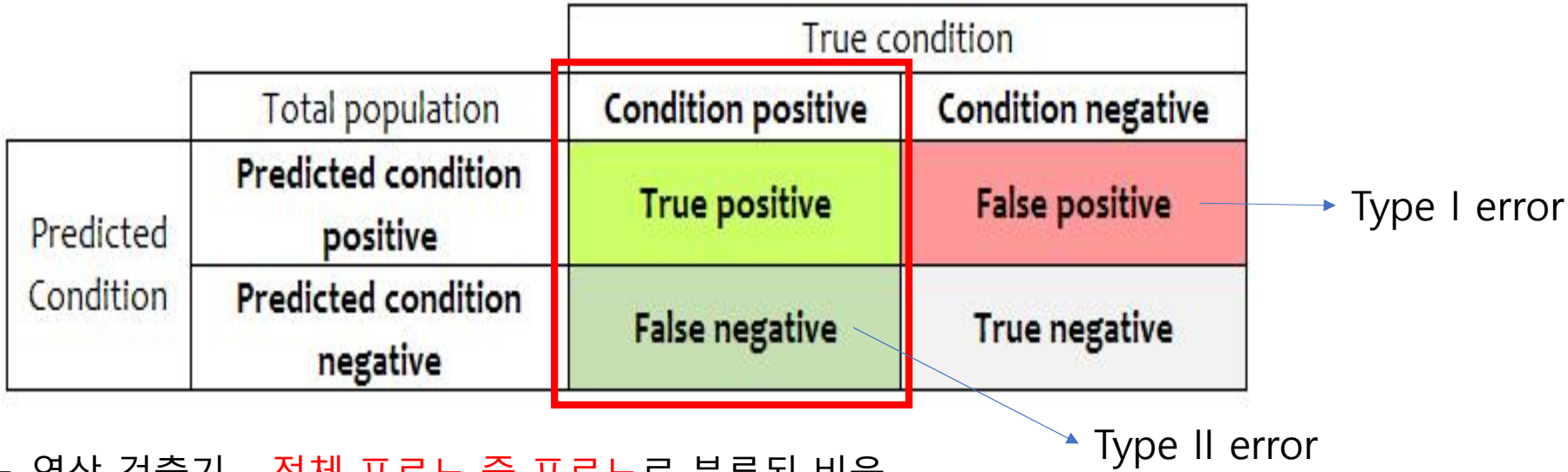
- Precision = TP / (TP + FP)
  - 정밀성. **Positive로 예측한 내용 중에, 실제 Positive**의 비율
    → Model 이 sample 을 True 로 분류했을 때 얼마나 자주 맞추었는가 ?
  - positive 분류의 정확성 측정 (1 에 가까울 수록 좋음)

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



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

Precision/Recall Trade-off

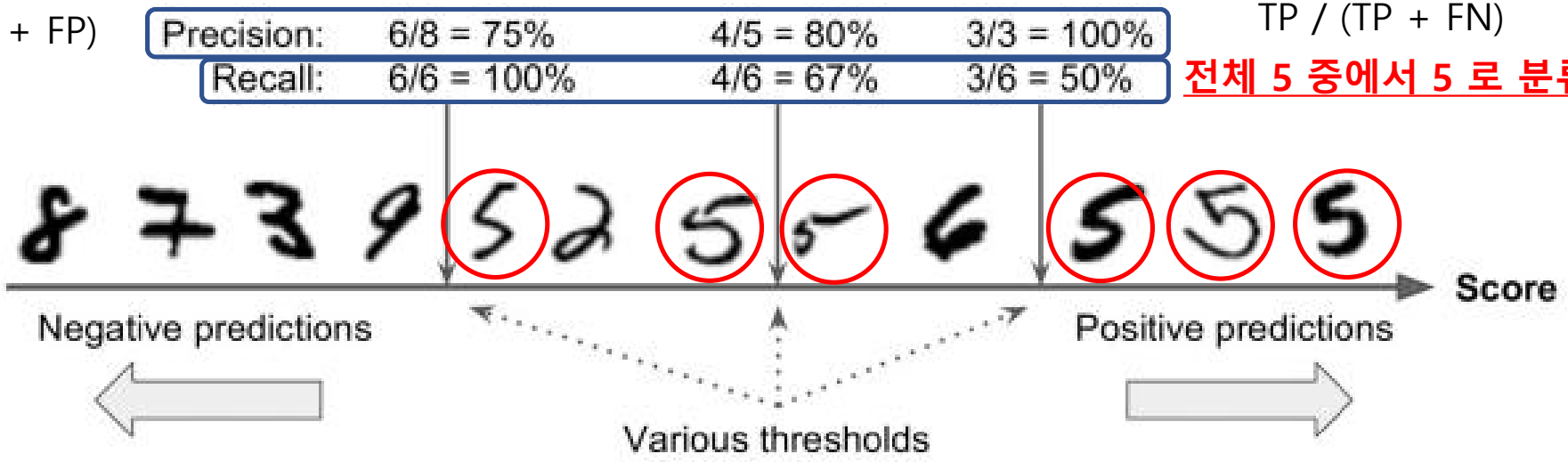
- 숫자 중 5 (positive) 를 검출

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 / Recall Trade-off

- 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.

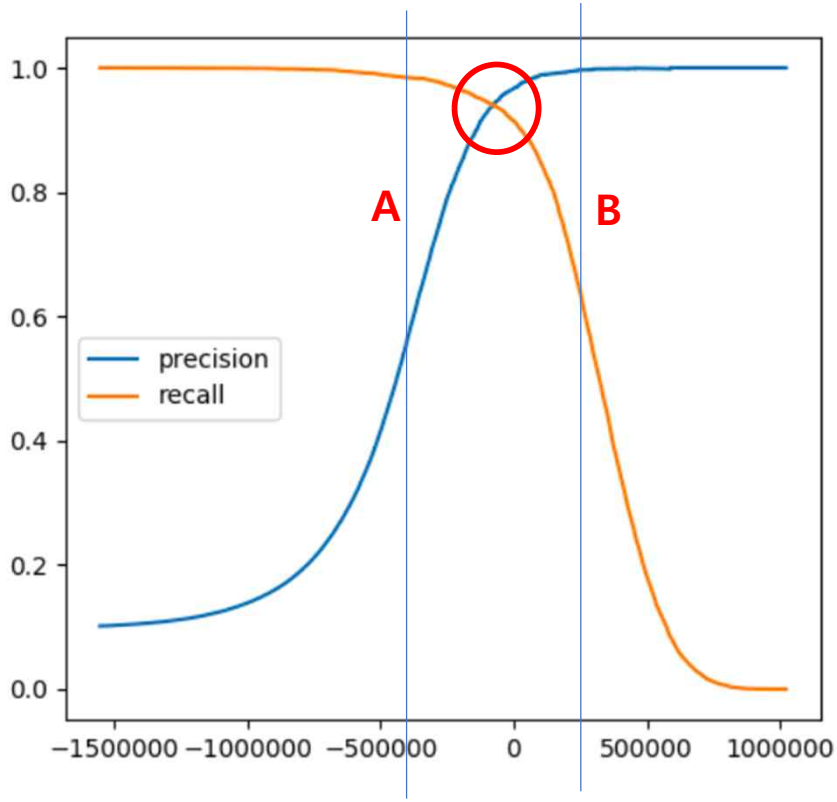
- 전체적 성능 측정에 활용 (조화평균)

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

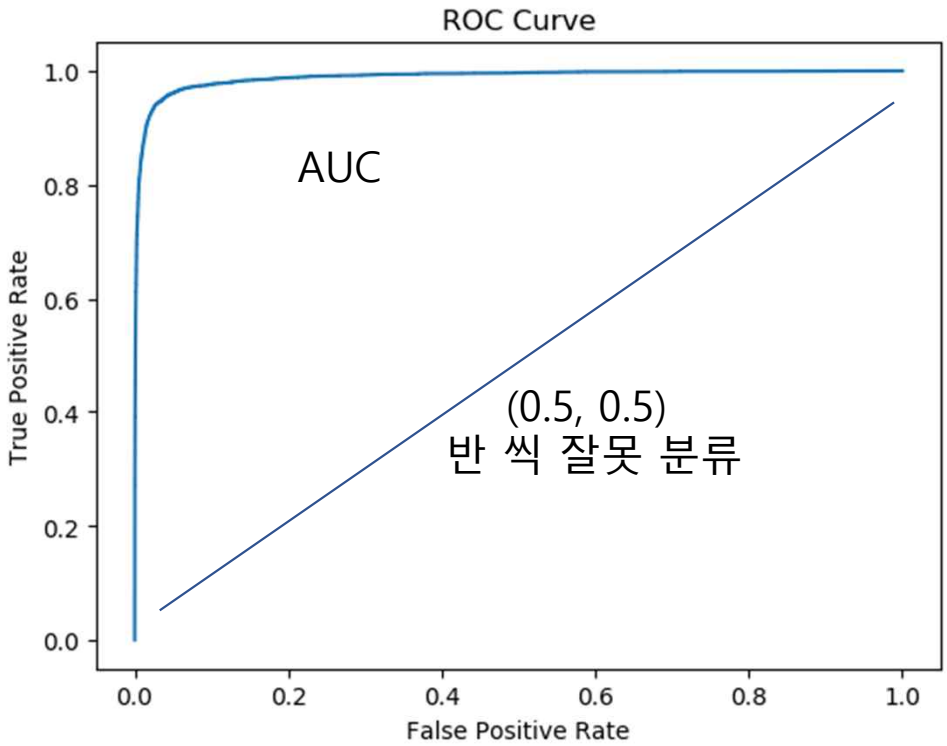
F1-Score = 0 ---> Poor (P=0 or R=0)
F1-Score = 1 ---> Perfect (P=1 and R=1)

A – High Recall / Low Precision

B – High Precision / Low Recall



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

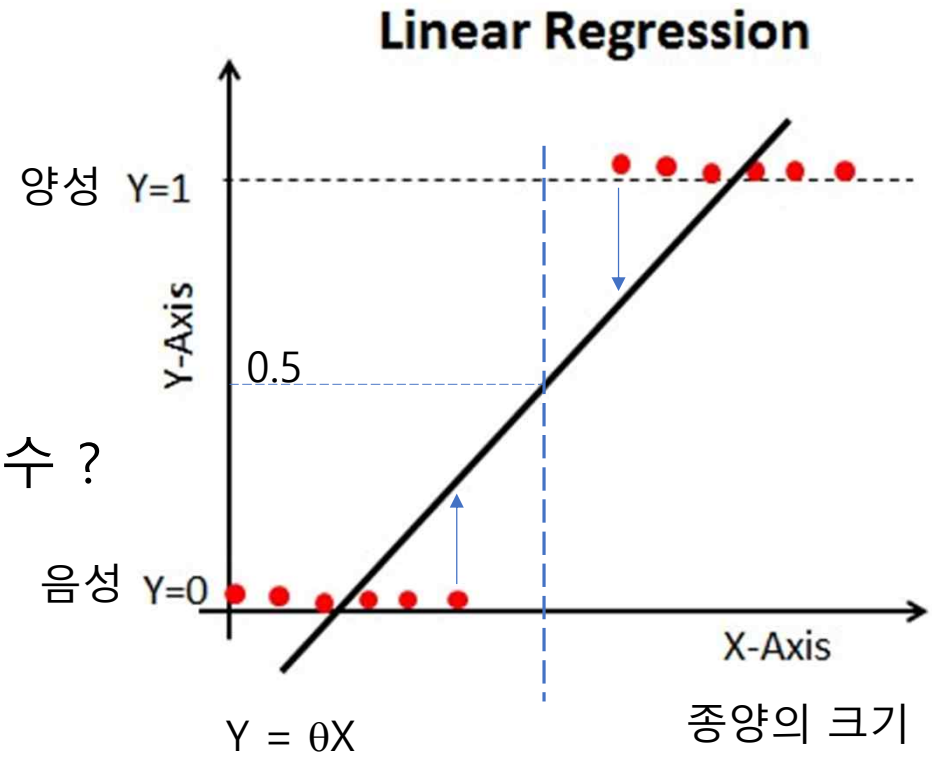
Logistic Regression

Logistic Regression (로지스틱 회귀)

- 선형회귀를 분류 문제에 적용 가능 ?

$$\hat{y} = 0 \text{ if } \theta X < 0.5$$
$$1 \text{ if } \theta X \geq 0.5$$

- **0 과 1** 로 구성된 분류 문제에 적합한 함수 ?
- 0 과 1 에 속할 **확률 값**을 return
- 미분가능한 성질



Sigmoid 함수

- S curve 형성
- Logistic 함수

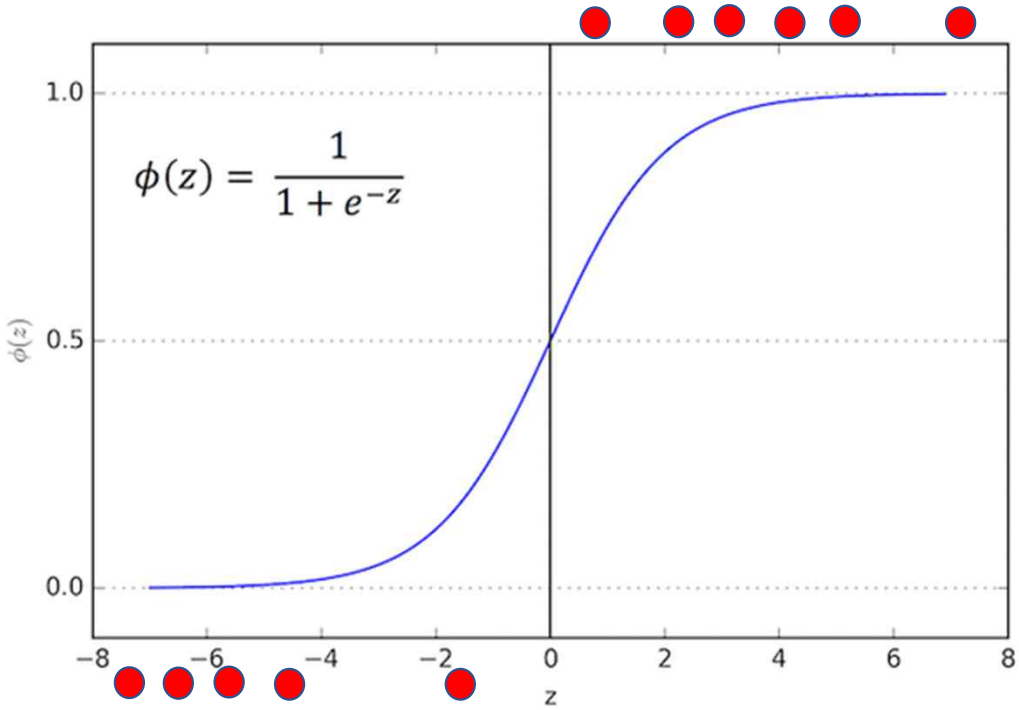
logit

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

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

1 if f(z) ≥ 0.5

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



Logistic Regression (로지스틱 회귀) classifier

1. 가장 단순한 분류기 → binary-class

2. 구현이 간단하고 모든 classification problem 의 기초

3. Logistic Regression 의 기본 concept 은 Deep Learning 에도 적용

4. 독립변수와 종속변수 간의 관계를 찾아내고 평가함

5. sklearn.linear_model.LogisticRegression(solver='lbfgs')

   * solver – optimization algorithm

실습: 040. Logistic Regression 을 이용한 이진분류

1. 특정 사용자가 구매를 할지 여부를 예측 (구매: 1, 구매 않음: 0)

2. Dataset 구성
   사용자 id, 성별, 연령, 추정급여, 구매여부
   이중 연령, 추정급여 두가지 feature 를 이용하여 구매여부 예측

3. Train / Test dataset 은 8 : 2 로 분리

4. Feature Scaling 실시 (standard scaling)

5. Model evaluation by Confusion Matrix, f1-score

Ensemble Learning

Random Forest/Gradient Boosting

What is Ensemble Learning ?

- 다수의 약한 학습기 (weak learner) 를 조합(Ensemble) 하여 더 높은 성능 추출
- Decrease Variance by **Bagging** (Bootstrap Aggregating)
- Decrease Bias by **Boosting**

What is Bagging (Bootstrap Aggregating) ?

- Training sample 의 sub-set 을 무작위로 추출하여 classifier 훈련

- Bootstrap – 중복을 허용하는 random sampling 방법 (통계학)

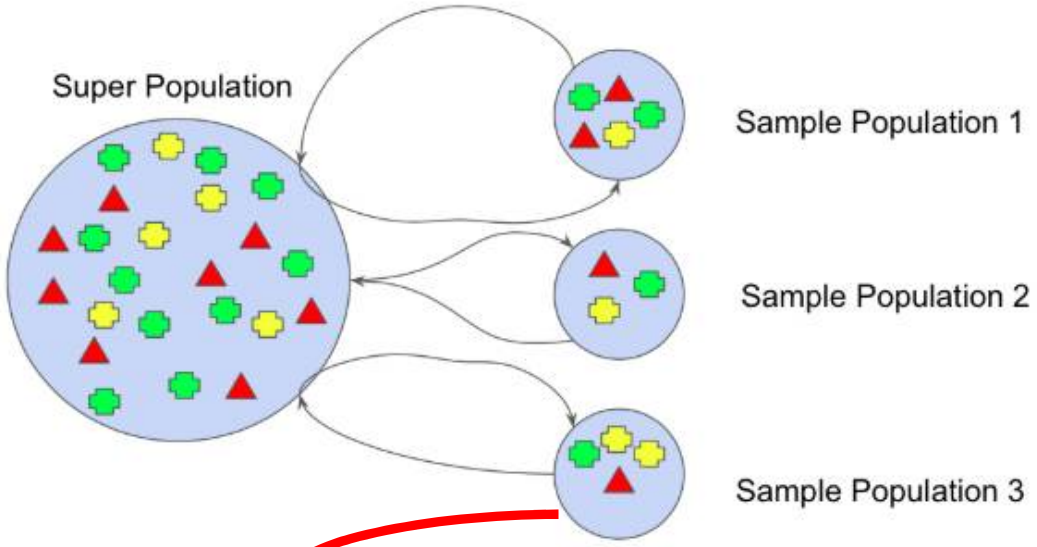
  ➔ b 개의 independent training set 을 평균하면,

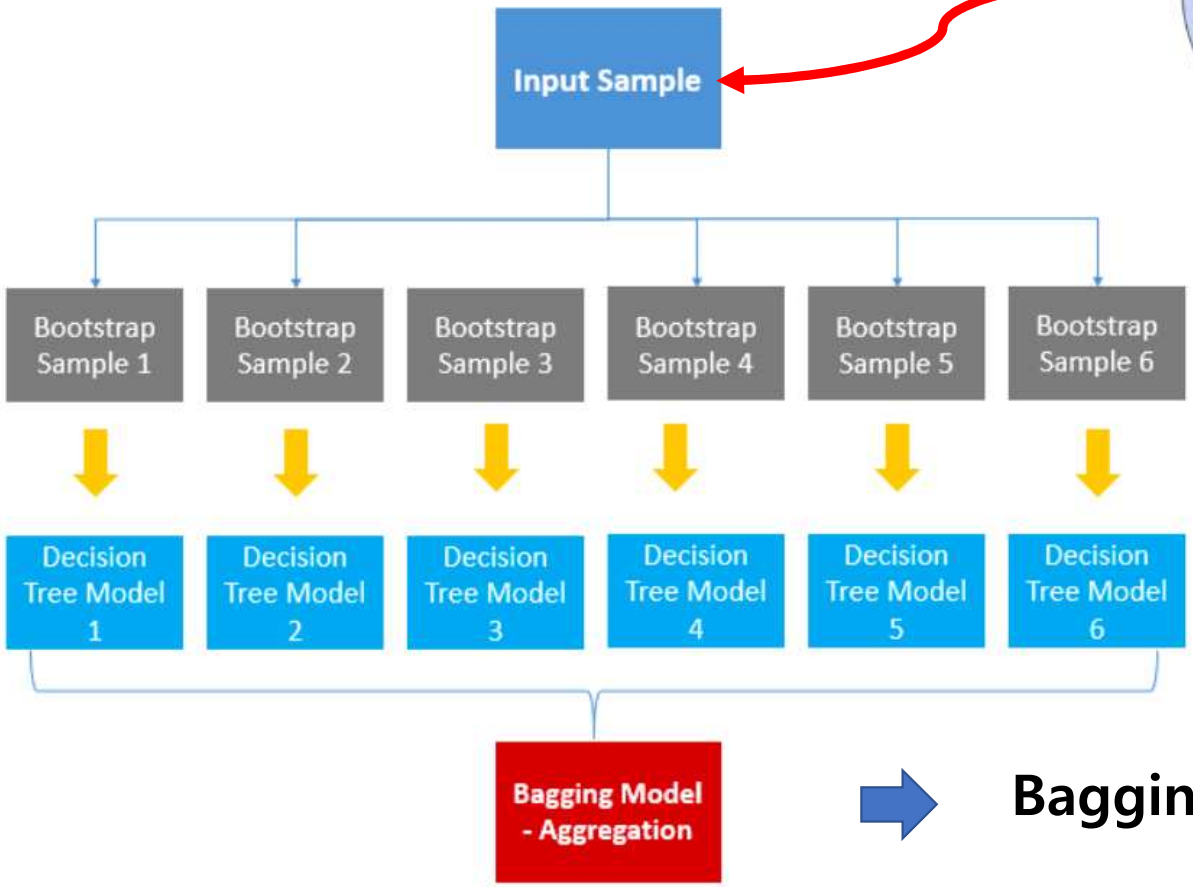
  variance → $\frac{\sigma^2}{b}$ 로 감소, mean → 동일

- Aggregating – voting for classification

- **Random Forest** 가 대표적인 method

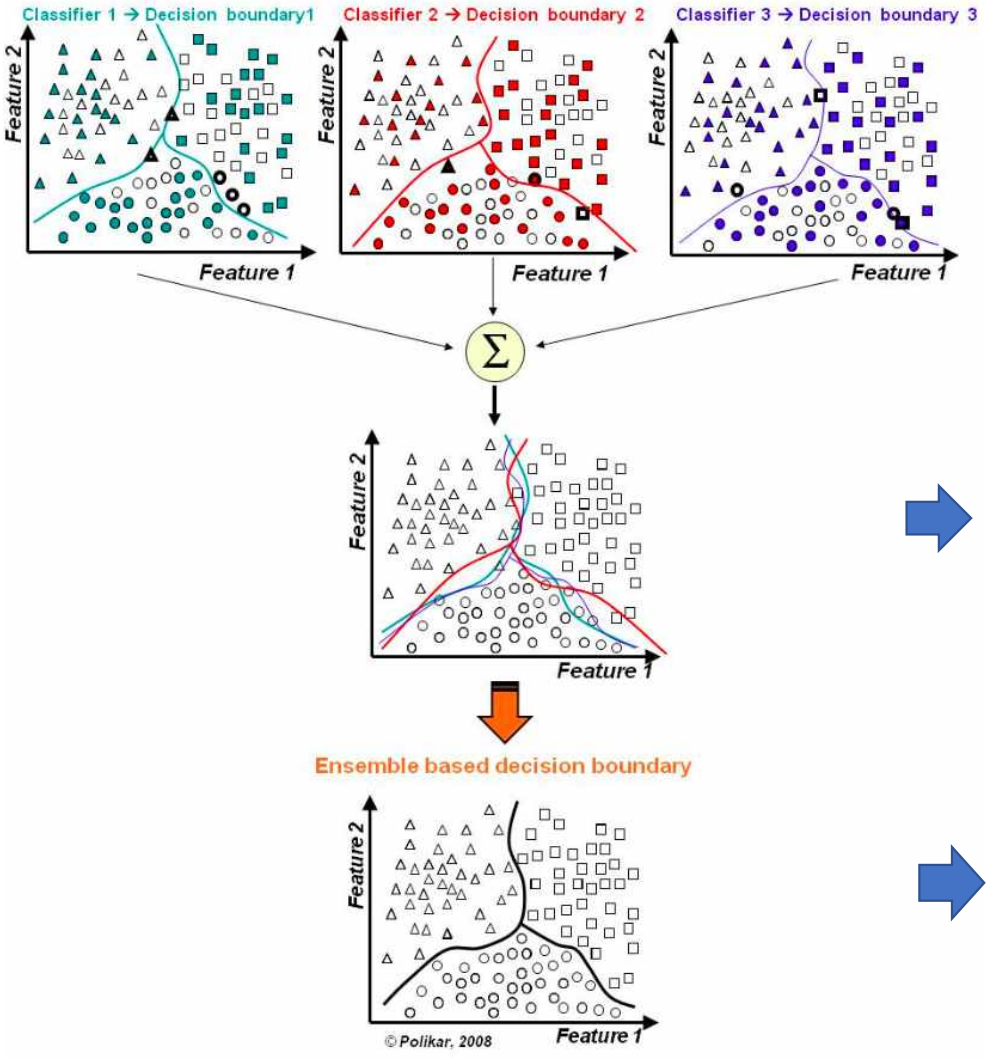
Bootstrapping (복원추출)





Decision Tree

Bagging Tree (Random Forest)



Decision Trees

Majority Voting

Random Forest

Bagging Tree 알고리즘

1. training data 에서 random sampling 하여 X_b, y_b 를 고름
   (ex. 1,000 개의 data 중 100 개 sampling)

2. X_b, y_b 를 이용하여 **ID3 알고리즘**으로 **decision tree** 구성

3. B 개의 tree 가 만들어질 때까지 1, 2 반복

4. B 개의 모든 tree 를 이용하여 classification 한 후 majority vote 로 결정

5. Bias 를 유지하며 Variance 를 줄인다.

   ➔ bagging 은 random sampling 에 의해 model 을 fit 하므로 **bias (model complexity) 가 커지지 않으면서 variance 를 줄일 수 있는 특징**이 있다.

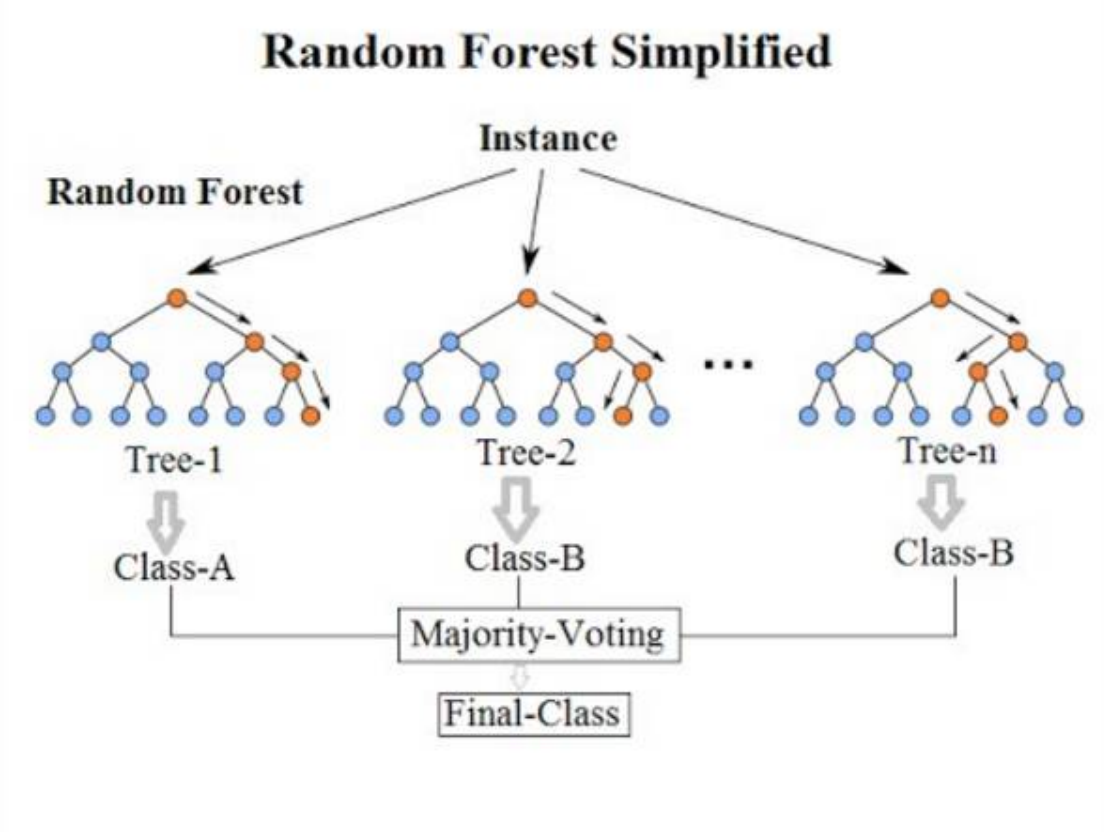
Random Forest

- Bagging Tree

  ➔ Random Forest 로 발전

- **Randomly Choose Attributes**

- bootstrapping 에 의한 복원추출
  +
  attribute 의 random 선택에 따른 independent trees 구성



Random Forest Algorithm

1. Decision Tree 에 포함될 attribute 들을 random 하게 선정

   ➔ 모든 attribute 를 가지고 Tree 를 만들 경우 매우 강한 attribute 가 모든 tree 에 항상 포함되는 것을 막기 위한 방법
   (ex. 30 개 attribute 중 10 개만 random selection)

   ➔ 좀 더 Random 하고 독립적인 classifier (Tree) 들을 생성시킬 수 있으므로 Random Forest 로 명명

2. Tree-based model 이므로 **white box 특징을 유지** (feature_importance_)

3. **High prediction accuracy**

4. **병렬적**으로 생성 가능하므로 속도가 빠르다

5. 각 tree 는 **매우 deep 하게 생성**된다.
   ➔ 인위적인 prune 을 하지 않으나 Ensemble 을 하면 low bias, low variance 가 된다.

What is Boosting ?

- Misclassified data 에 더 높은 weight 를 부여하여 다음 번 model 의 sampling 에 포함될 확률을 높이는 것

- 다수의 weak learner (small decision tree) 를 훈련 시켜 이전 learner가 발생시킨 오차를 다음 learner 가 학습하는 방법

- **AdaBoost, Gradient Boost (XGBoost)** 가 대표적인 method

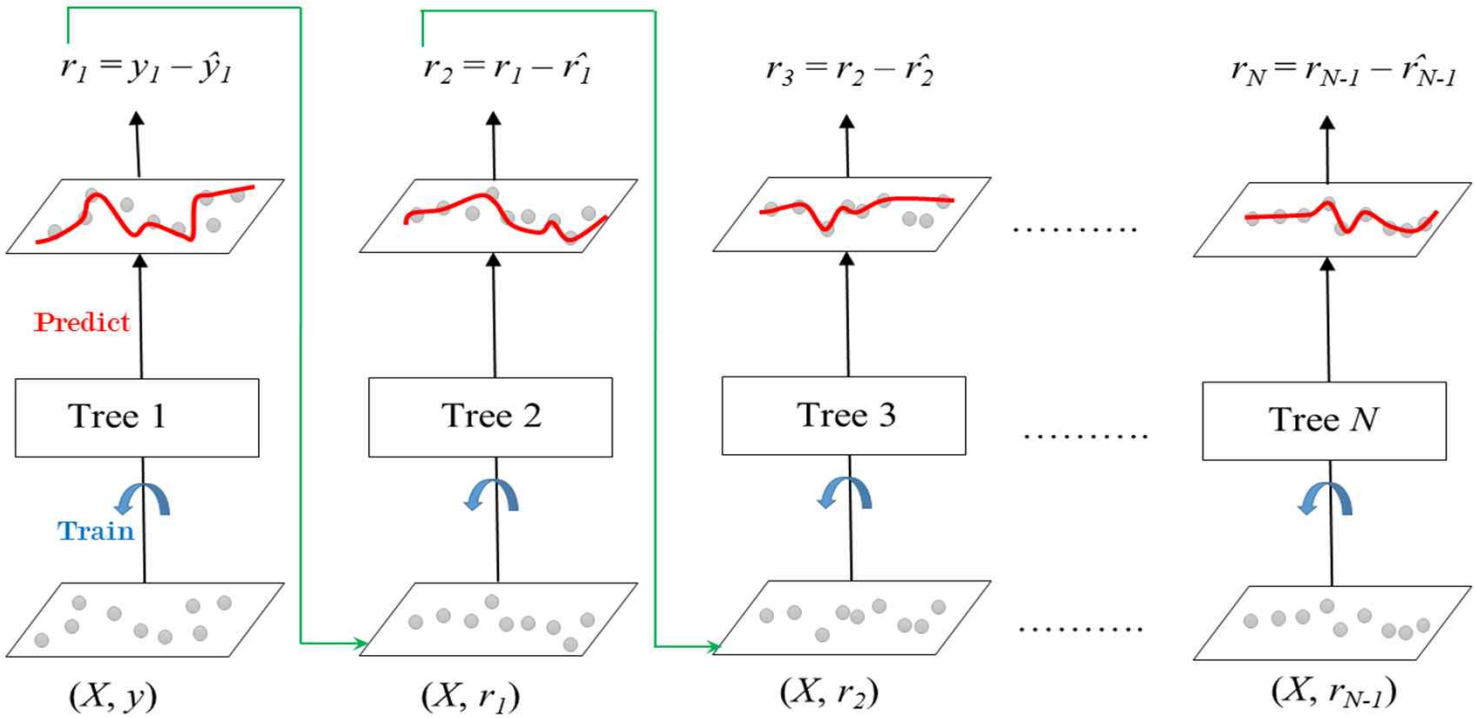
Gradient Boost 알고리즘

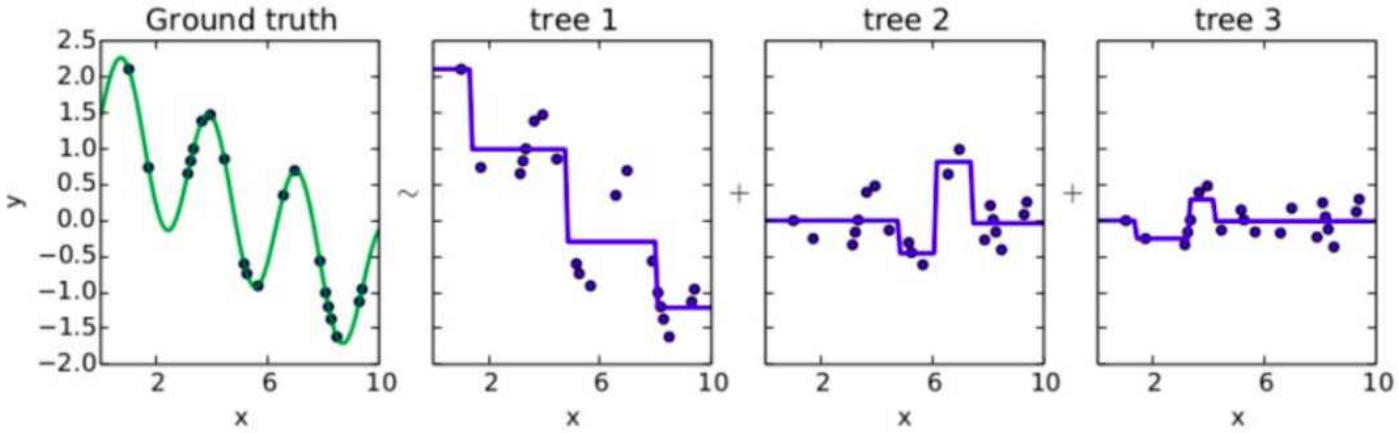
- **Weak Learner** - random choice 보다 약간 더 나은 성능의 모델
  ➔ Decision Tree 사용

- 기존의 weak learner 가 생성한 residual error 를 감소시키는
  추가 tree 를 더 이상의 감소 효과가 없을 때까지 생성

- Gradient Descent (경사하강법)에 의해 loss function (ex. MSE) 을 optimize

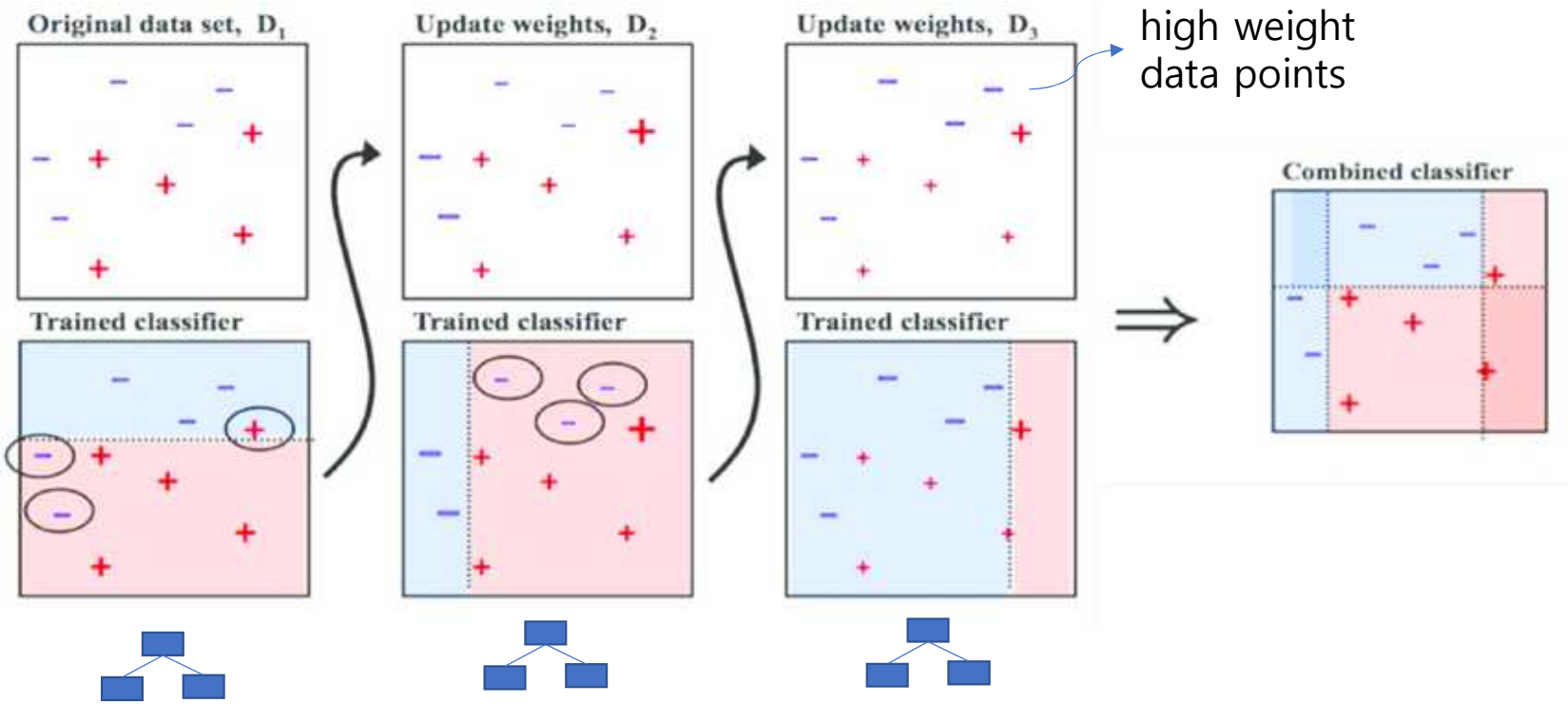
Gradient Boost





- 이전 tree 에서 발생한 잔차 (residual error) 를 next tree 에서 **보정**

(참고) AdaBoost (Adaptive Boosting)



Stump 로 구성 – weak learners (기존 weak learner의 단점 보완)

이전 stump 에서 잘 못 분류한 example 이 다음 sampling 에 포함되도록 가중치를 높이는 기법

Gradient Boost for Regression

* Gradient Boost는 single leaf 부터 시작

모든 target 값의 평균 Weight

1st Tree 71.2

(88+76+56+73+77+57)/6=71.2

input

	Height (m)	Favorite Color	Gender	Weight (kg)	1st Tree Residual	2nd Tree Residual
1	1.6	Blue	Male	88	16.8	15.1
2	1.6	Green	Female	76	4.8	4.3
3	1.5	Blue	Female	56	-15.2	-13.7
4	1.8	Red	Male	73	1.8	1.4
5	1.5	Green	Male	77	5.8	5.4
6	1.4	Blue	Female	57	-14.2	-12.7

target

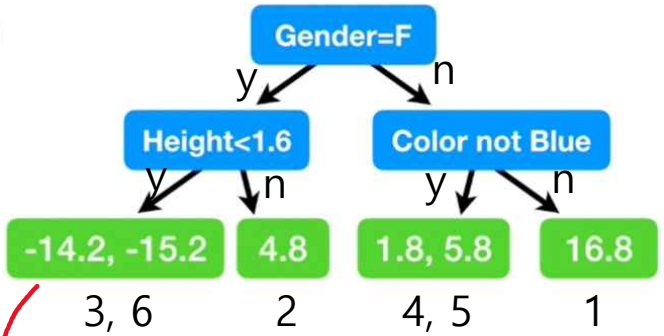
잔차

Residual 감소

\+

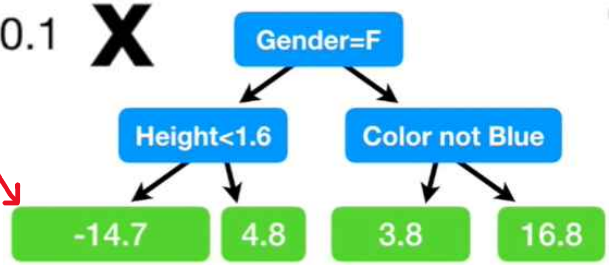
* 실제로는 leaf가 8 ~ 32인 트리 사용

2nd Tree



마지막 leaf 노드에 두개의 residual 값이 있는 경우, 두 값의 평균으로 치환

0.1 X



71.2 + 0.1 * 16.8 = 72.9

88 – 72.9 = 15.1

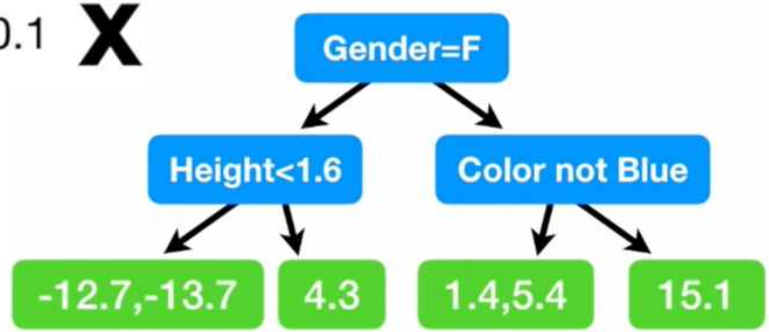
- 첫 single leaf 모델이 예측한 71.2kg보다는 실제 값(88kg)에 더 가까워짐.
- Gradient Boost 모델은 이런 식으로 실제 값에 조금씩 가까워지는 방향으로 학습.

71.2 + 0.1 * 16.8 + 0.1 * 15.1 = 74.39
88 – 74.39 = 13.6

	Height (m)	Favorite Color	Gender	Weight (kg)	1st Tree Residual	2nd Tree Residual	3rd Tree Residual
1	1.6	Blue	Male	88	16.8	15.1	13.6
2	1.6	Green	Female	76	4.8	4.3	3.9
3	1.5	Blue	Female	56	-15.2	-13.7	-12.4
4	1.8	Red	Male	73	1.8	1.4	1.1
5	1.5	Green	Male	77	5.8	5.4	5.1
6	1.4	Blue	Female	57	-14.2	-12.7	-11.4

3rd tree

0.1 X



Residual ≅ 0 이 될 때까지 반복

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

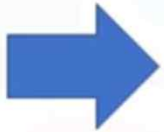
Data Formatting

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 호의 생존 예측

- Titanic 호 생존여부 data 이용
- Survival 여부 예측 model 작성
- Feature Engineering
- Random Forest / Gradient Boost model을 이용한 분류

Clustering

Clustering 이란 ?

- 비슷한 object 들끼리 모으는 것

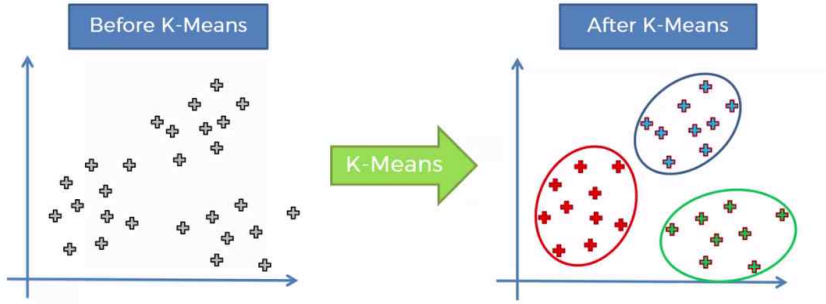
- label data 가 없는 것이 classification 과 차이

  ➔ unsupervised machine learning

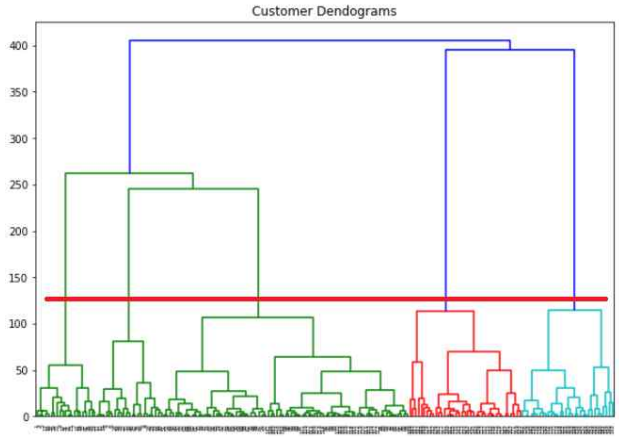
- 적용 사례
  - 고객의 구매 형태별 분류
  - 고객의 취향에 맞는 책, 동영상 등의 추천
  - 신용카드 사용의 fraud detection
  - 뉴스 자동 분류 및 추천
  - 유전자 분석 등

Clustering 알고리즘의 종류

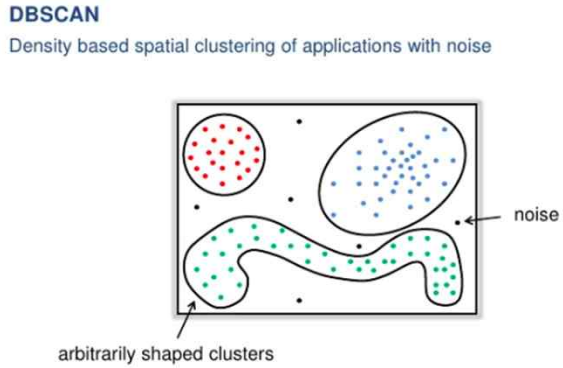
- K-Means Clustering



- Hierarchical Clustering (dendrogram)



- Density-based Clustering (DBSCAN)



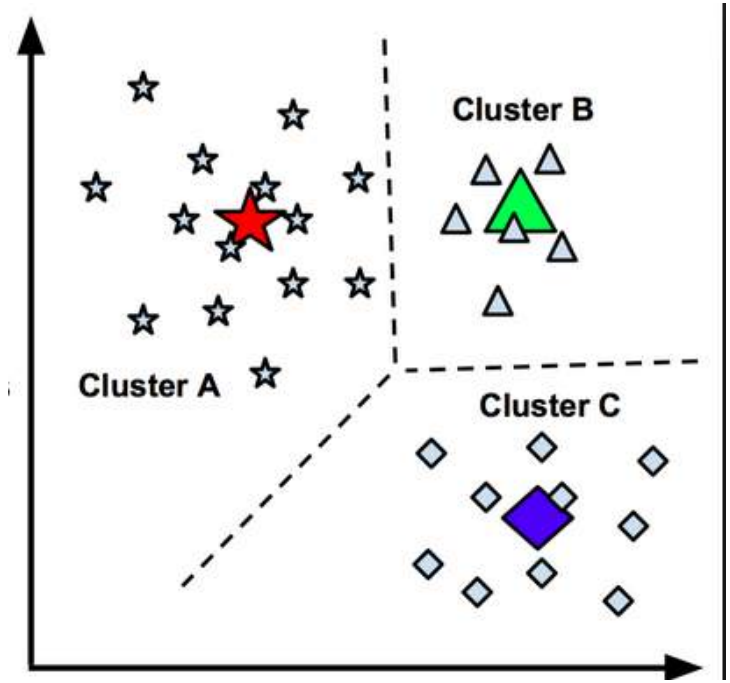
K-Means Clustering 알고리즘 – Distance 계산

Distance = Euclidean Distance

$$= \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

Ex)

고객	나이	수입	교육	
1	54	190	3	→ x1
2	50	200	8	→ x2



Distance(x1, x2) $= \sqrt{(54-50)^2 + (190-200)^2 + (3-8)^2} = 11.87$

K-Means Clustering 알고리즘

1. Random 하게 k 개의 centroid (중심점) 를 정한다.

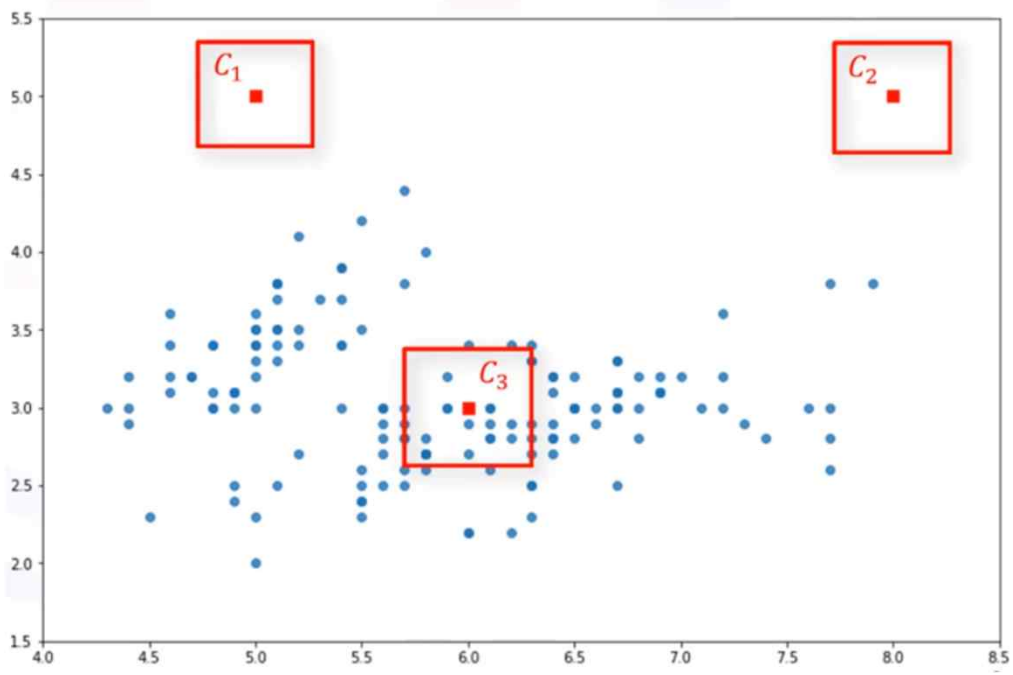
2. 각 centroid 로 부터 각 data point 까지의 거리를 계산.

3. 각 data point 를 가장 가까운 centroid 에 할당하여 cluster 를 생성.

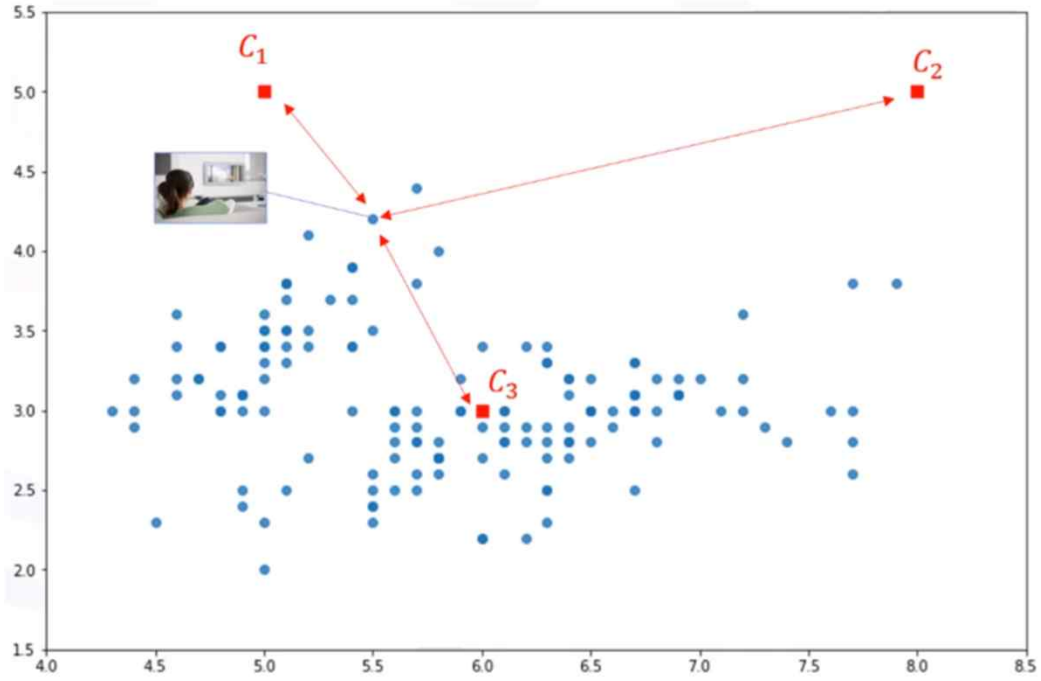
4. K centroid 의 위치를 다시 계산

5. centroid 가 더 이상 움직이지 않을 때까지 2-4 단계를 반복

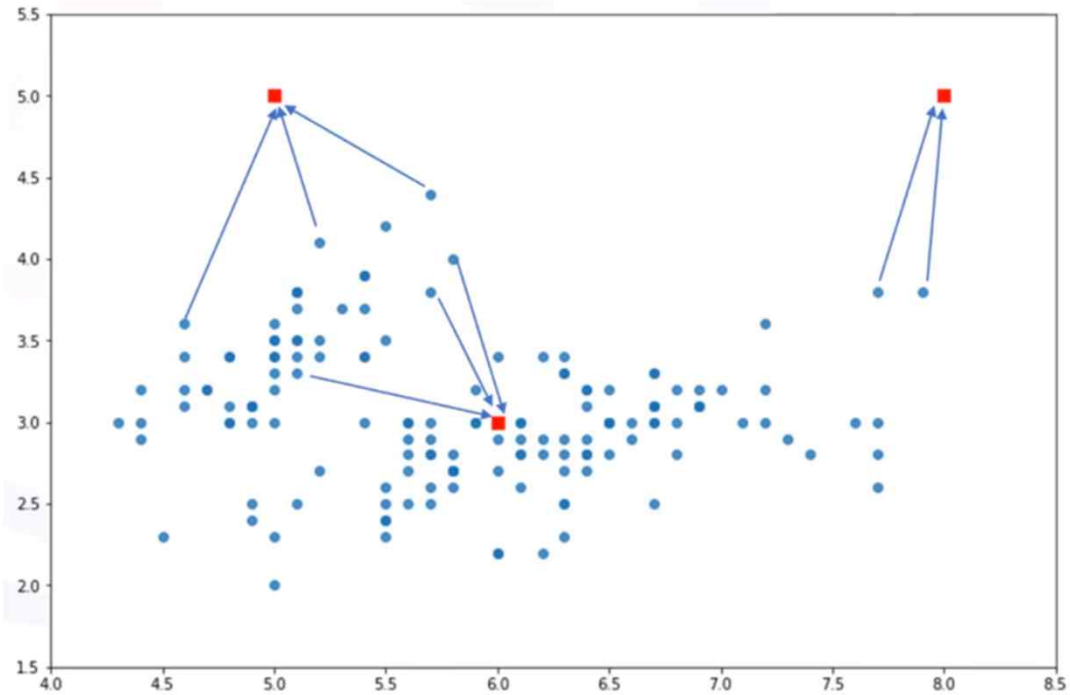
1. 임의의 centroid 선정 : k=3

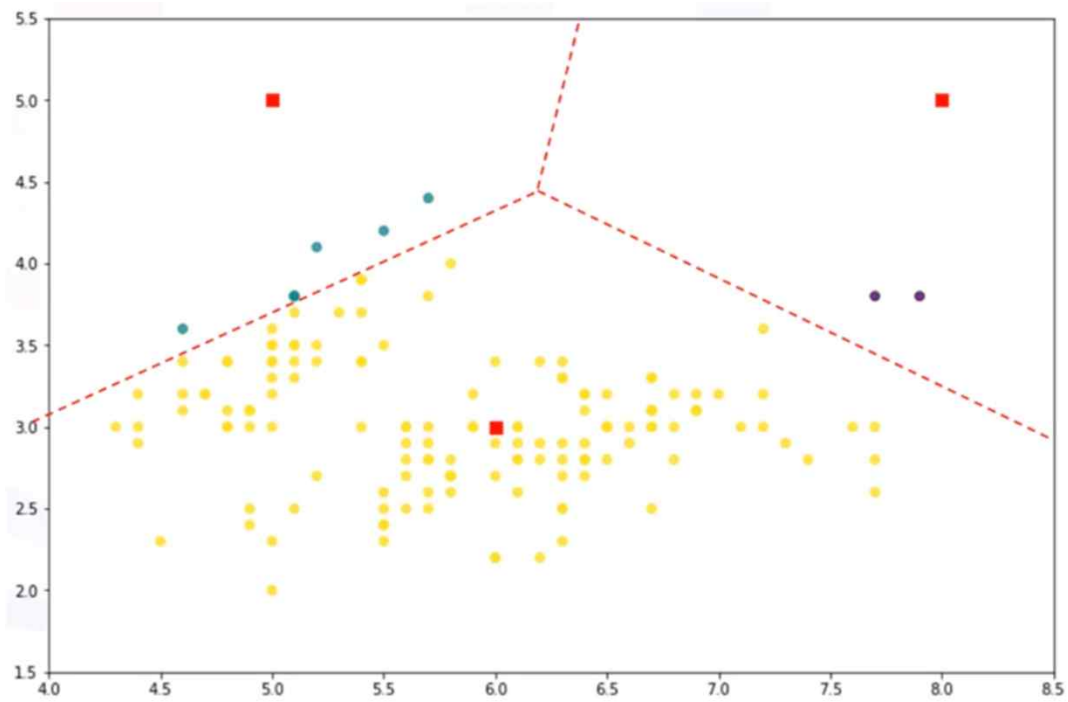


2. 거리 계산 for each data point

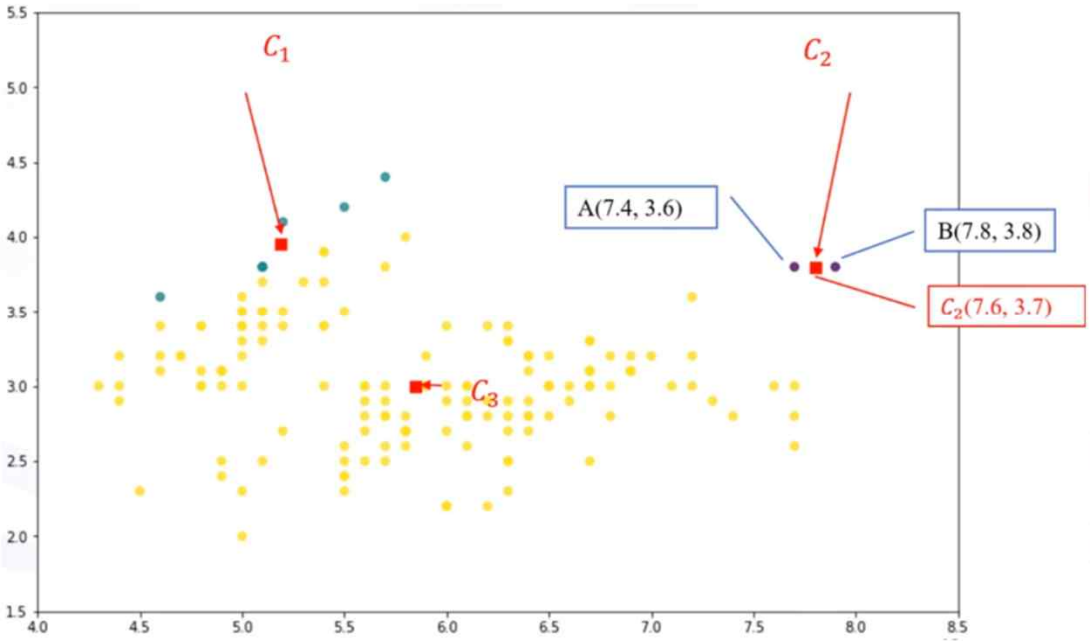


3. 가장 가까운 centroid 로 각 data point 할당

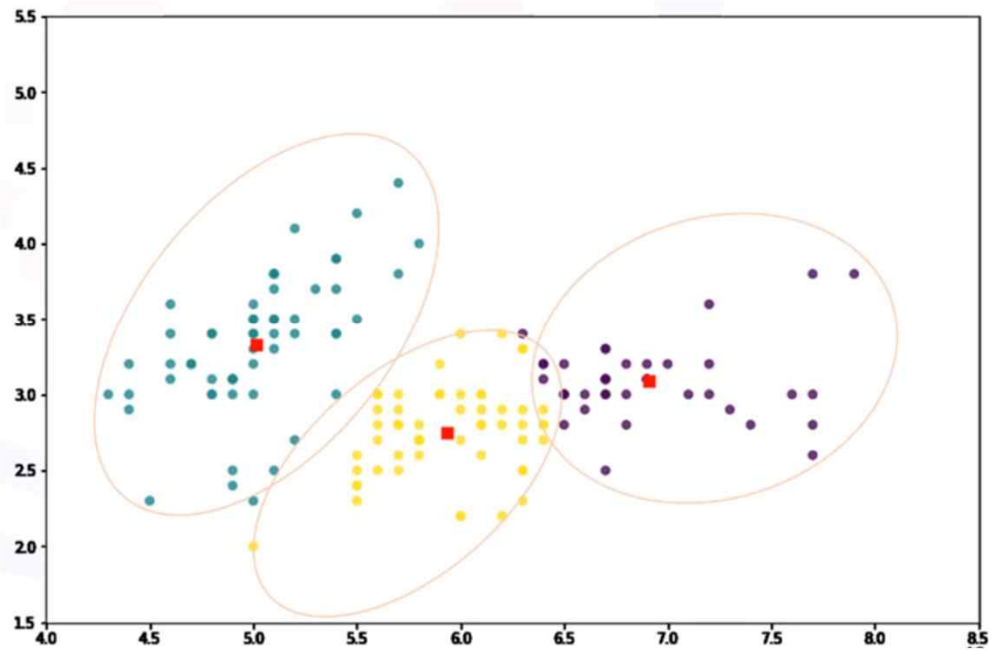




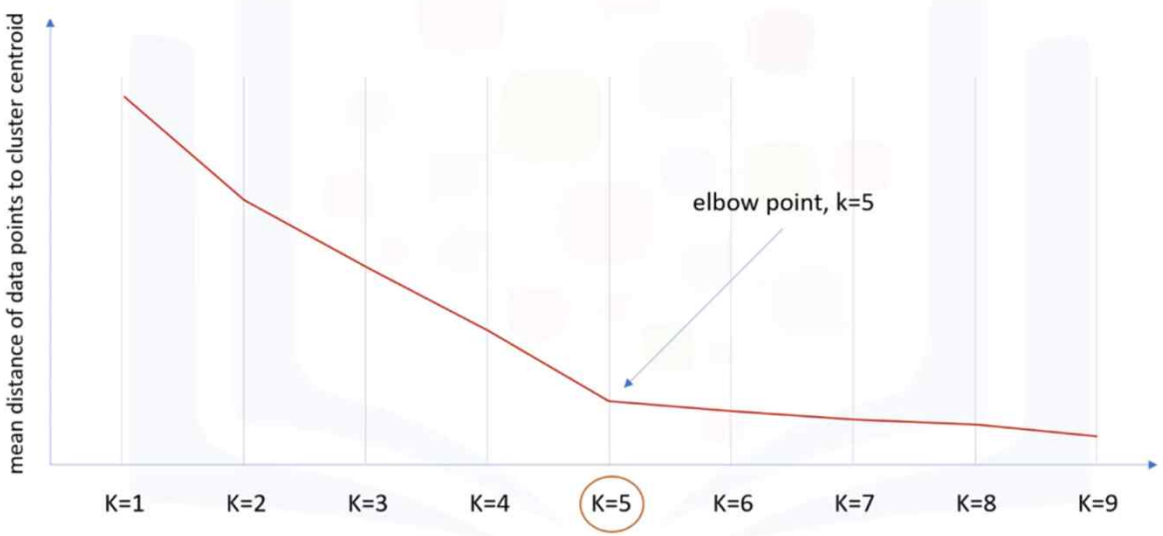
4. 각 cluster 의 new centroid 계산



5. Centroid 변화 없을 때까지 반복



Choosing k

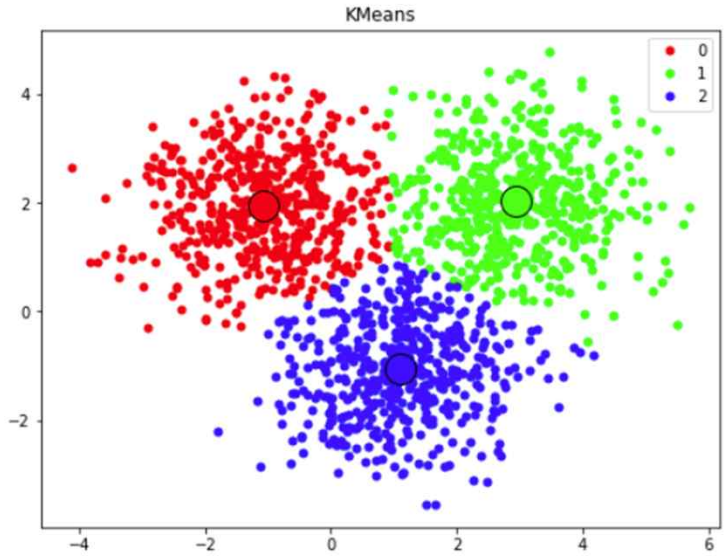


- K 를 잘 정하는 것이 중요
  - How ? → 경험적으로 $\mathrm{k} = \sqrt{n}$ (n: data sample 개수)

- feature 가 많을 경우 계산량 증가

실습: 080. K-Means

1. sklearn.cluster.Kmeans
2. Dataset : sklearn.dataset.samples_generator.make_blobs 사용
3. Matplotlib 을 이용하여 visualize

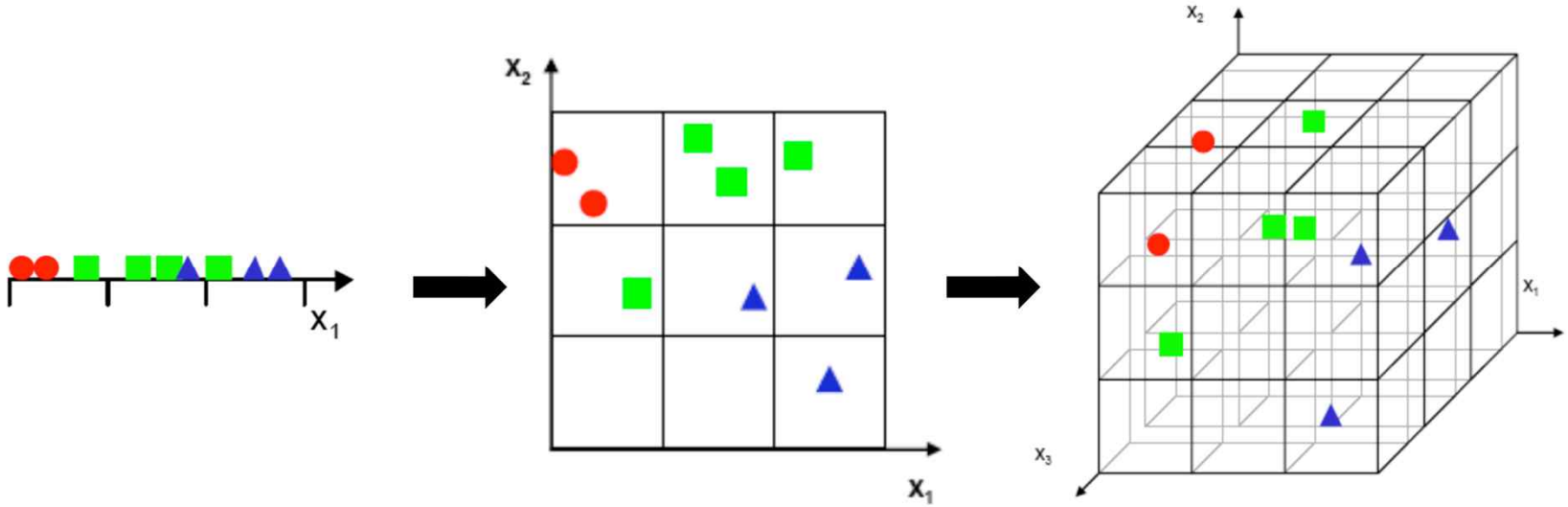


차원축소

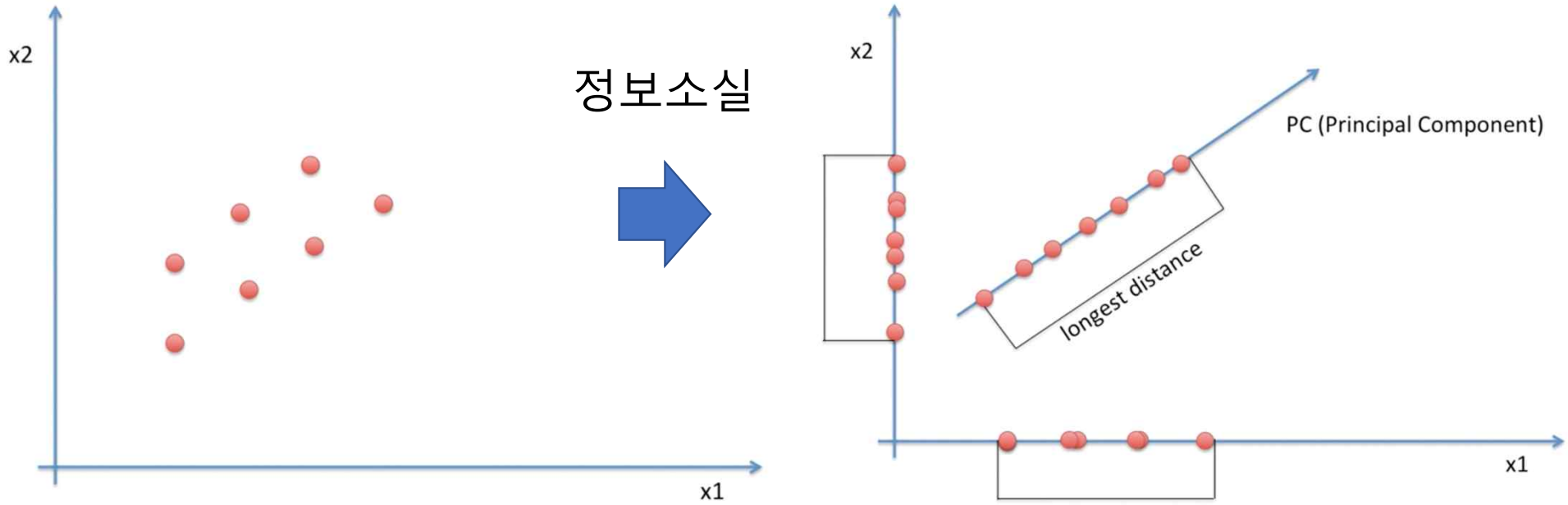
차원의 저주 (Curse of Dimensionality)

- 차원이 증가함에 따라 vector 공간내의 space 도 증가하는데 데이터의 양이 적으면 빈공간이 많이 발생하여 예측의 정확도가 떨어진다.

- 유사한 성격의 feature (예, 키, 신장, 앉은키, 기온, 수도관 동파, 빙판길 미끄러짐 사고 등) 는 하나의 새로운 feature 로 성분을 합칠 수 있음

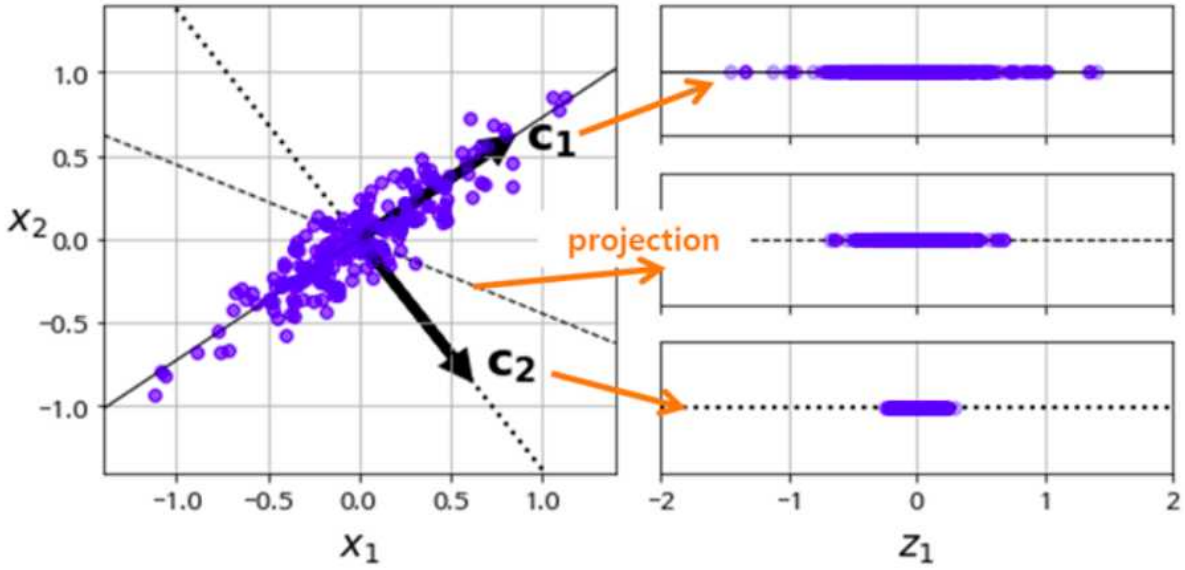


How to reduce dimension ?



PCA (Principal Component Analysis) - 주성분 분석

- **선형대수학의 SVD (특이값 분해)** 를 이용하여 **분산이 최대인 축**을 찾음
- 데이터의 **분산(variance)**을 최대한 보존하면서 서로 직교하는 새 축을 찾아, 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 변환



분산을 최대한 보존

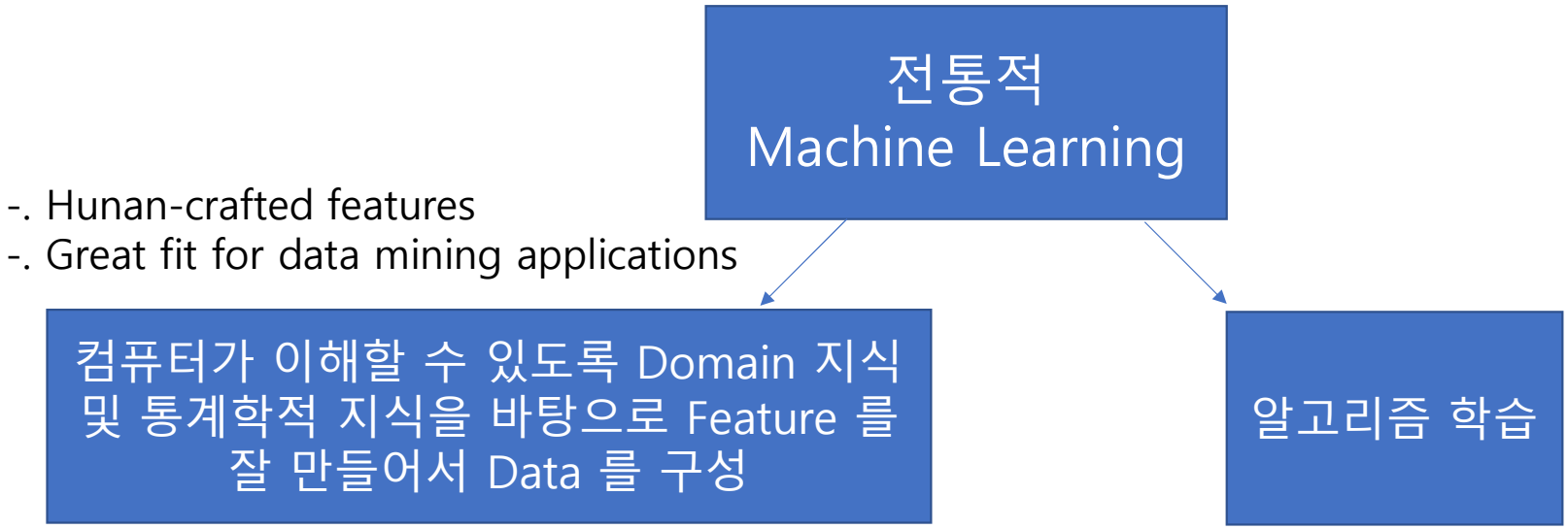
https://i.imgur.com/Uv2dlsH.gif

085. PCA 를 이용한 차원 축소

- PCA 적용 전 후의 Logistic Regression model 정확성 비교
- 차원 축소된 dataset 시각화

Neural Network
and
Deep Learning

전통적 Machine Learning 의 학습



80~90 % 의 비중

Domain 전문지식을 가진 석,박사급 인재 필요

10~20% 의 비중

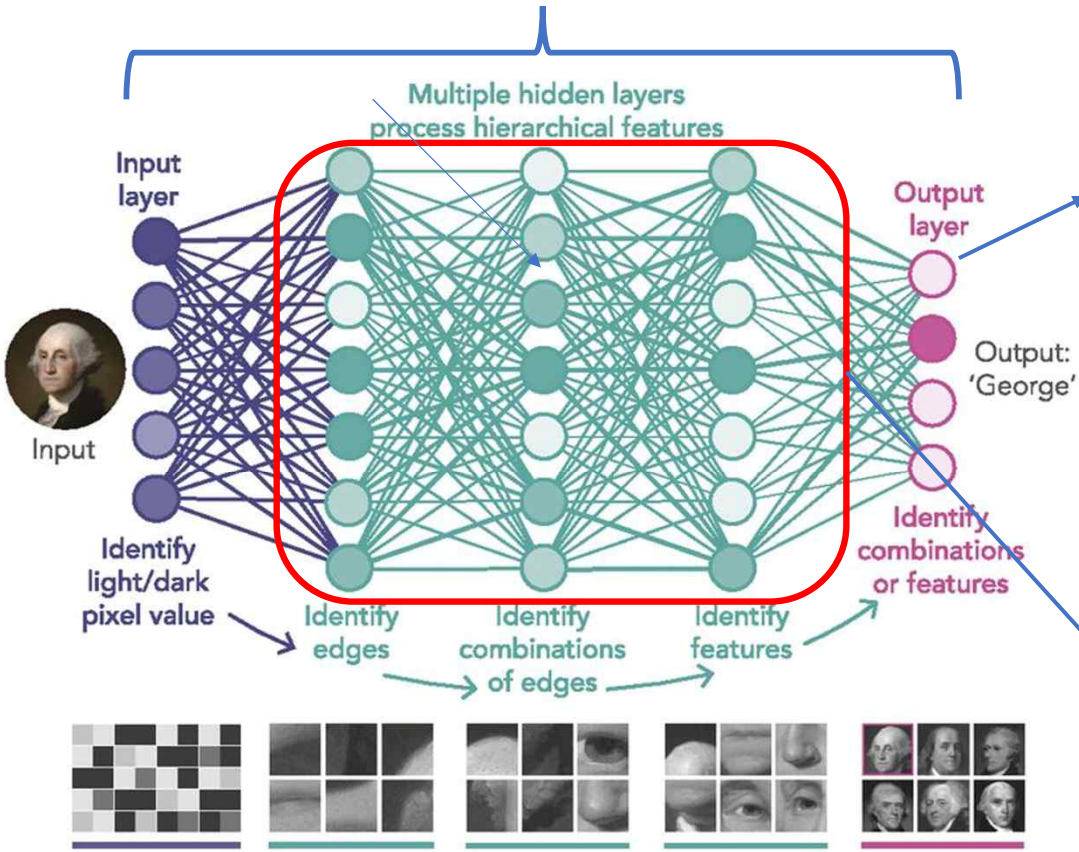
각 feature 의 weight 를 optimize

Deep Learning 의 학습

- 중요한 Feature 를 스스로 구분하여 weight 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



neuron

예측 결과

Big Mathematical Equation

Artificial Neuron (Perceptron)

구성요소:

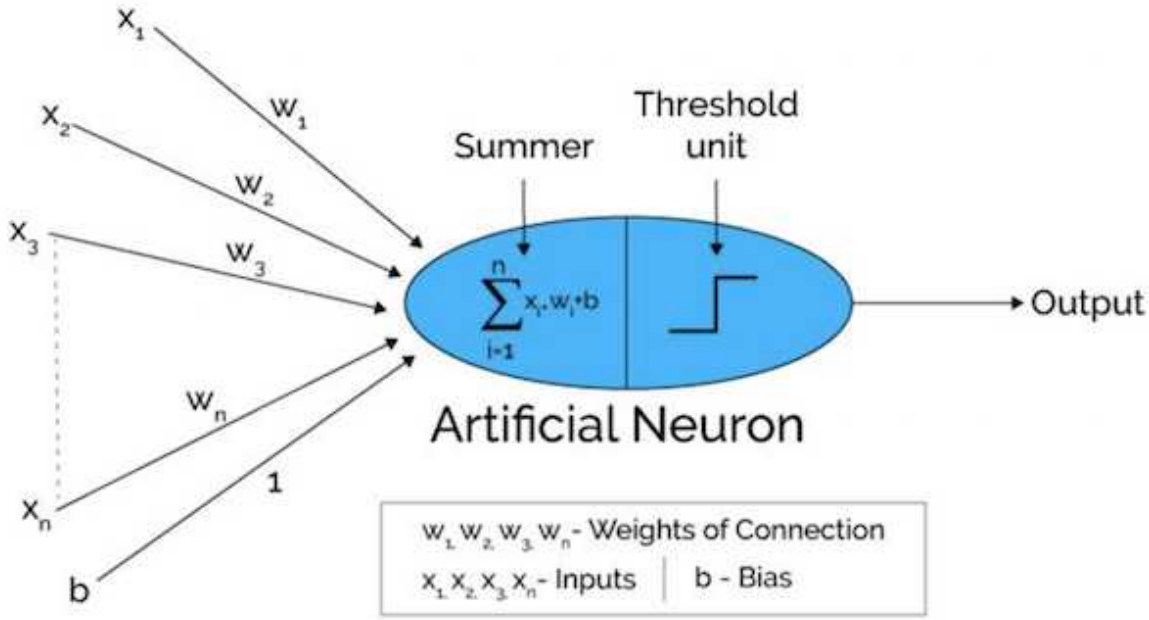
Pre-Activation :

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

Activation :

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

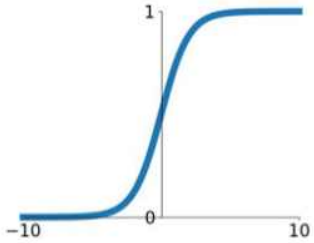
w : connection weights
b : bias
g : activation function



Activation Functions

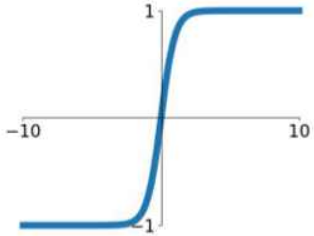
Sigmoid

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



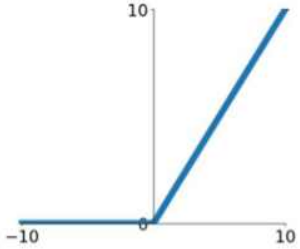
tanh

$$\tanh(x)$$



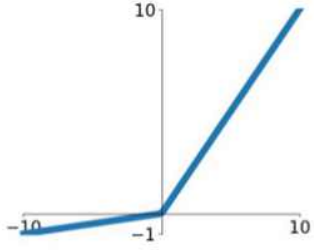
ReLU

$$\max(0, x)$$



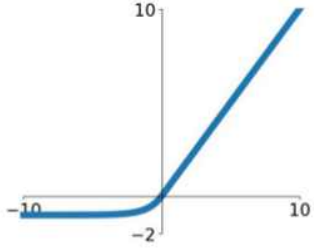
Leaky ReLU

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



ELU

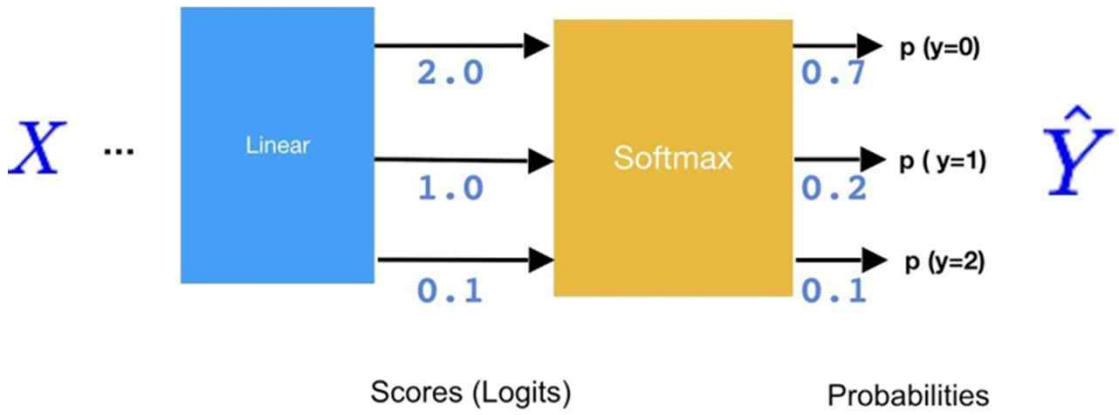
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



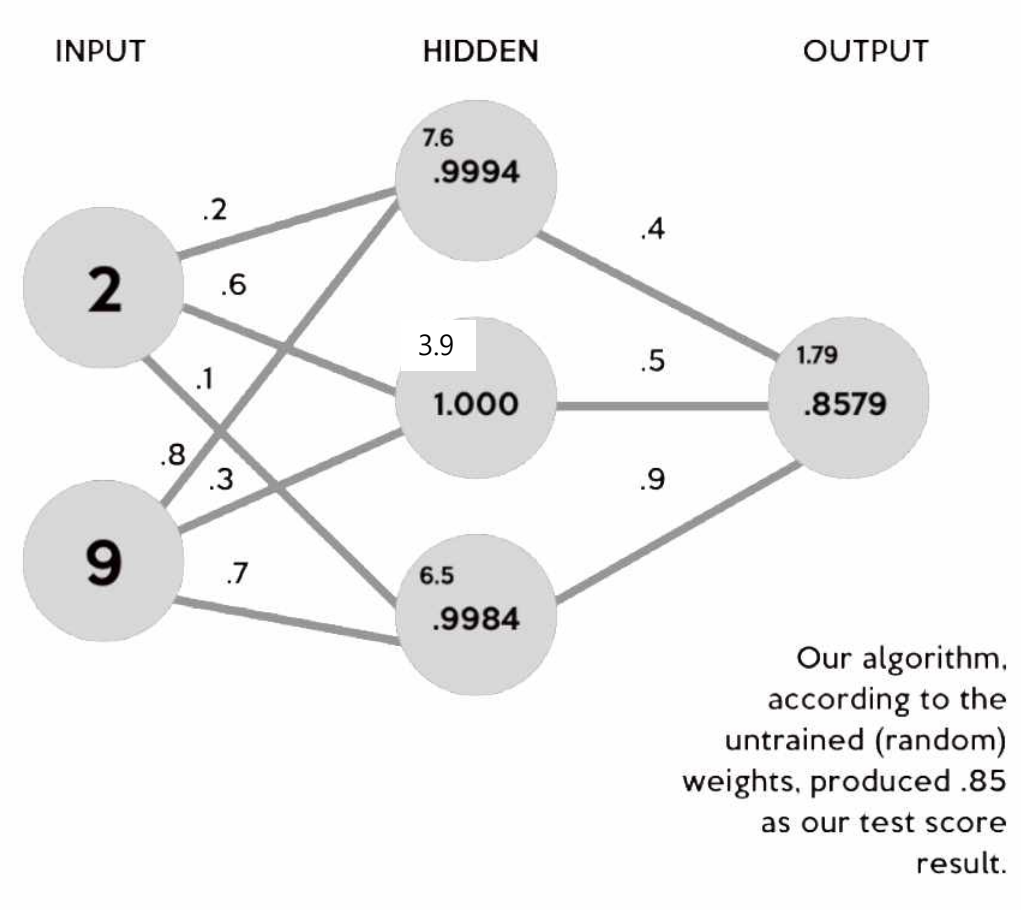
Softmax Activation Function

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

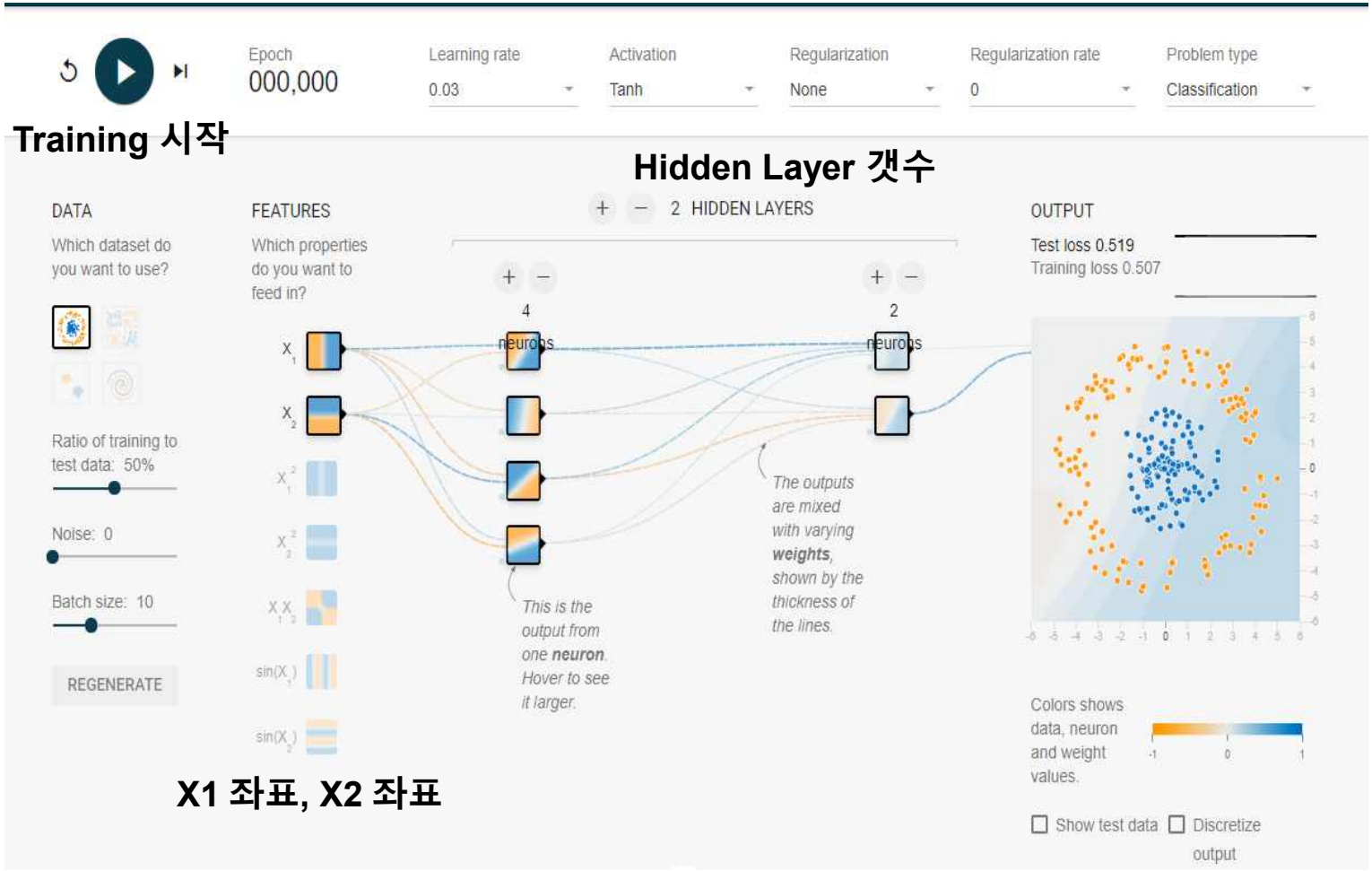
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



Neural Network 의 작동 원리



실습 : Tensorflow Playground

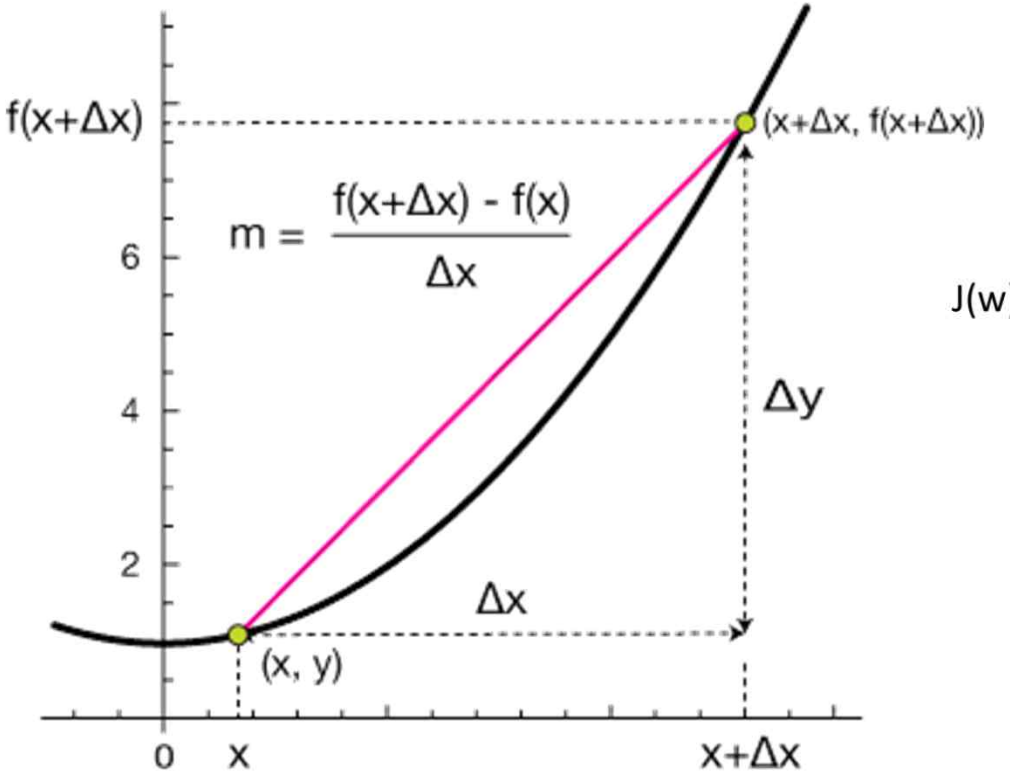


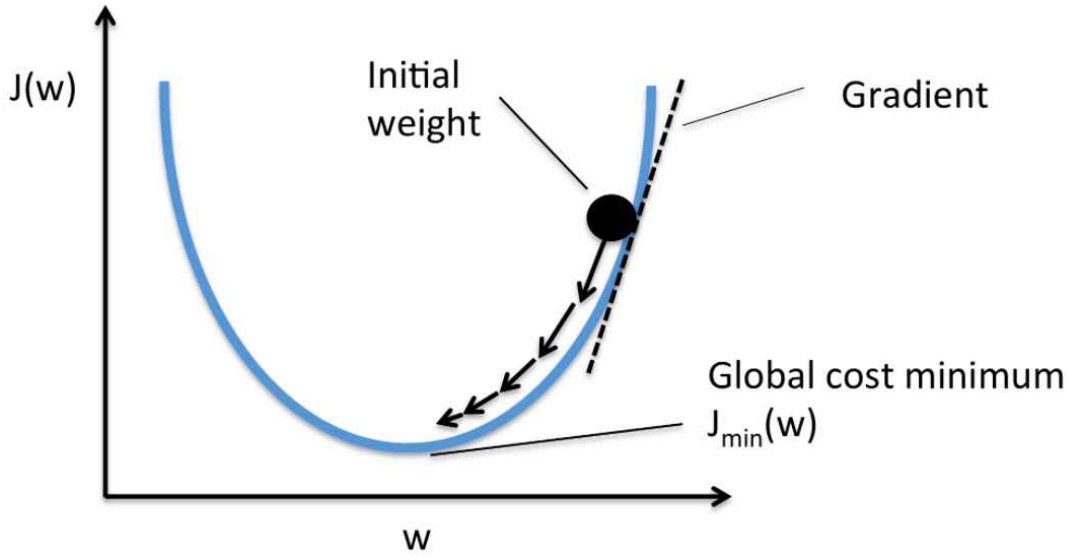
Neural Network
훈련 원리

Deep Neural Network 훈련의 핵심

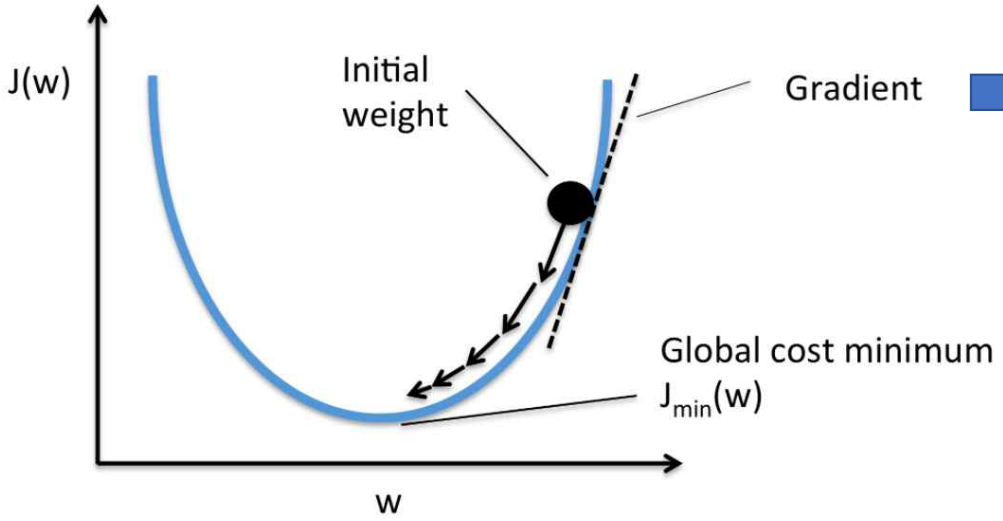
- Gradient Descent (경사하강법)
  - 목적 – 실제값과 예측값의 차이를 최소화 하는 parameter(θ) 발견
  - 방법 – 손실함수를 정의하여 손실함수의 값이 0 으로 수렴하도록 parameter(θ) 조절
- Backpropagation (오차역전파)
  - 손실함수를 최소화 하는 방향으로 신경망 전체의 parameter 가 update 되도록 하는 기법
- 손실함수
  - 비용함수(cost function), 목적함수(object function) 등으로도 불림
  - 경사하강법이 가능하도록 미분 가능한 함수를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

Derivative (도함수, 미분, 접선의 기울기)





Gradient Descent (경사하강법) Optimization



방향 – Gradient
(derivative of Cost Function)

이동 속도 – Learning Rate

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

경사하강법과 parameter update (선형회귀)

- $y = \theta_0 + \theta_1 x$

- Loss Function $L(\theta_0, \theta_1) = \frac{1}{N}\sum_{i=0}^{m} (y_i - (\theta_0 + \theta_1 x_i))^2$

- Gradient $\frac{\partial L(\theta_0,\theta_1)}{\partial \theta_0} = -2\frac{1}{N}\sum_{i=0}^{m}(y_i - (\theta_0 + \theta_1 x_i))$

  $\frac{\partial L(\theta_0,\theta_1)}{\partial \theta_1} = -2\frac{1}{N}\sum_{i=0}^{m} x_i(y_i - (\theta_0 + \theta_1 x_i))$

- Update $\theta_0 := \theta_0 - \alpha \frac{\partial L(\theta_0,\theta_1)}{\partial \theta_0}$

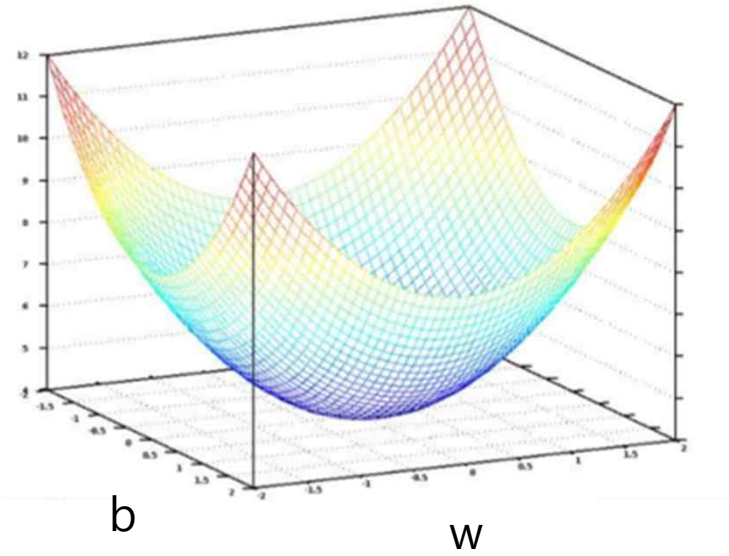
  $\theta_1 := \theta_1 - \alpha \frac{\partial L(\theta_0,\theta_1)}{\partial \theta_1}$

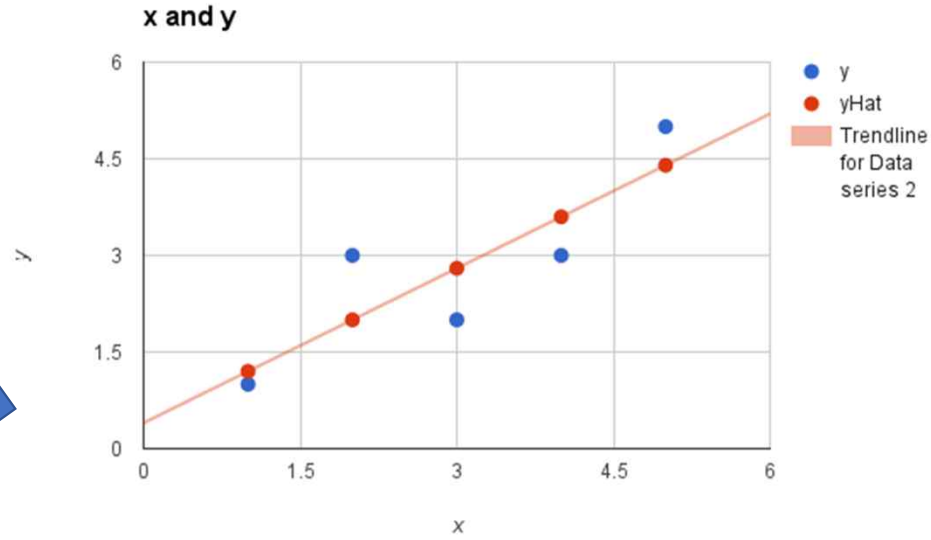
대표적 손실 함수

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

Cost Function - Linear Regression

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





y = Wx + b

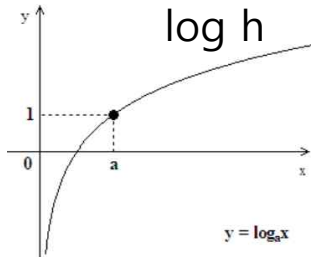
MSE(Mean Squared Error) 를 최소화 하는 W 와 b 를 optimize

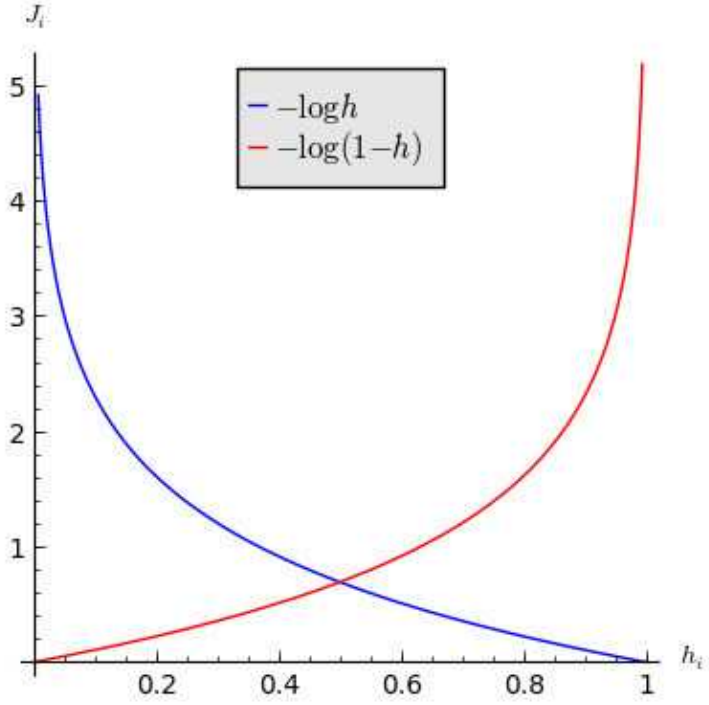
Cost Function - Logistic Regression (Binary Cross-entropy)

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

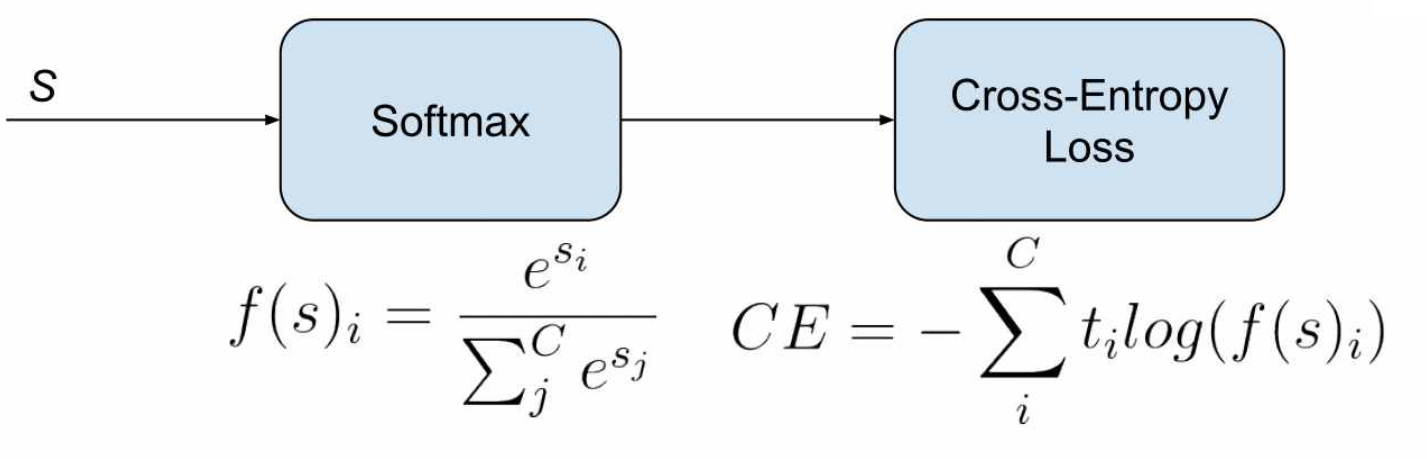
If $y^{(i)} = 1 : J(\theta) = -\log h_\theta(x^{(i)})$
where $h_\theta(x^{(i)})$ should be close to 1

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





Cost Function - Categorical Crossentroy (Softmax Loss)

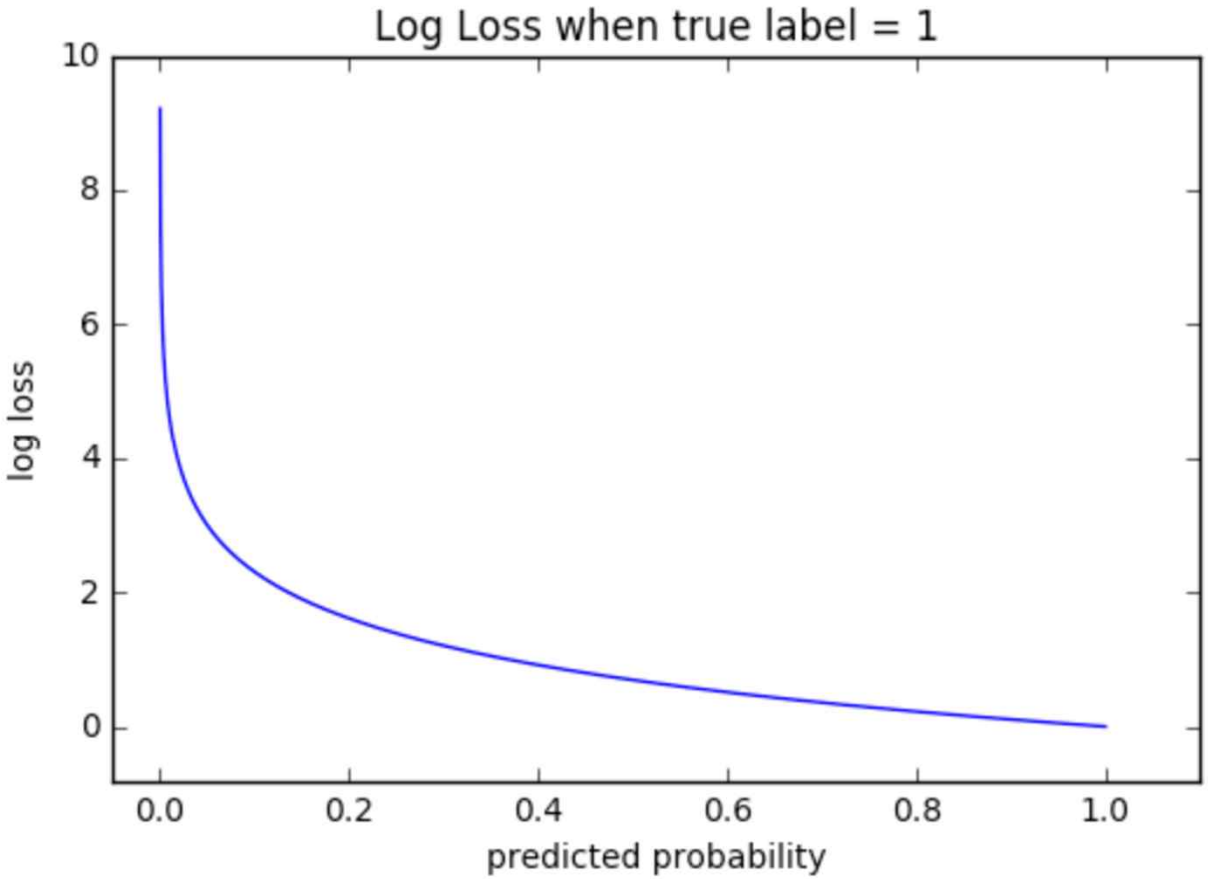


t_i : 0 이 아닌 target (one-hot encoded 되어 있으므로 multi-class 중 오직 1 개만 1)

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

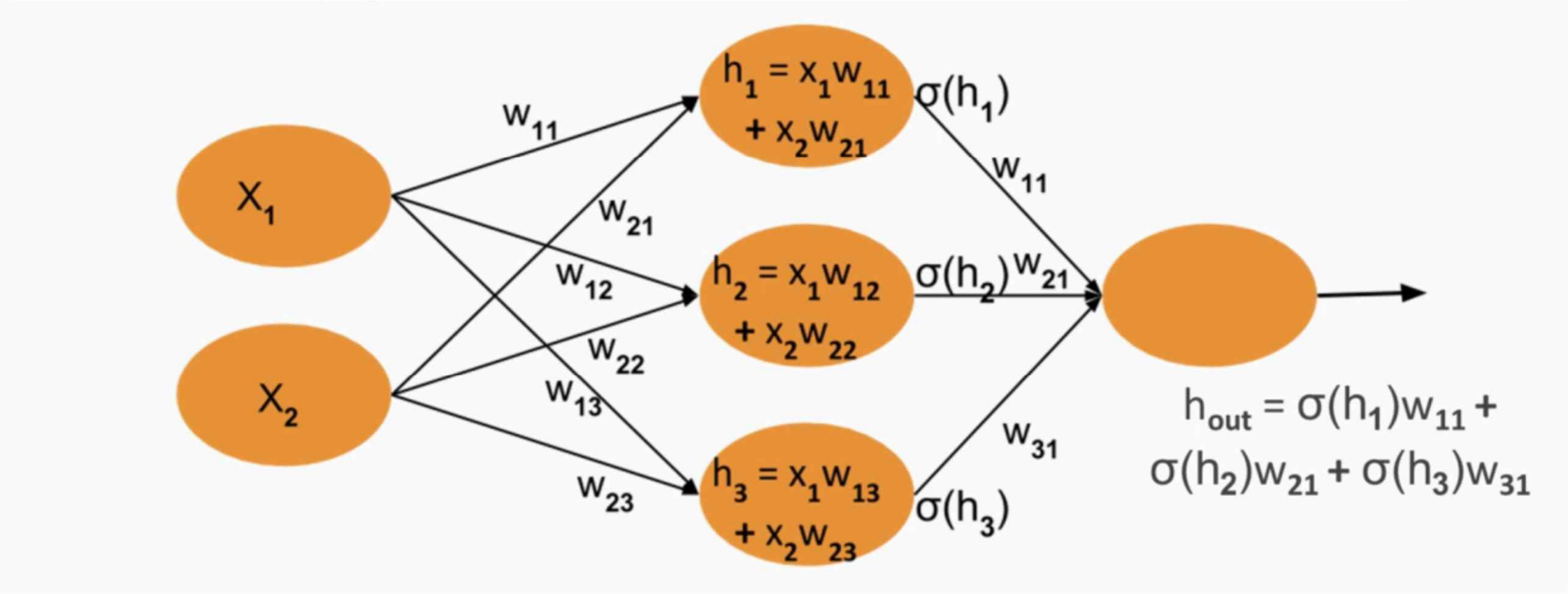
Cross-Entropy Loss



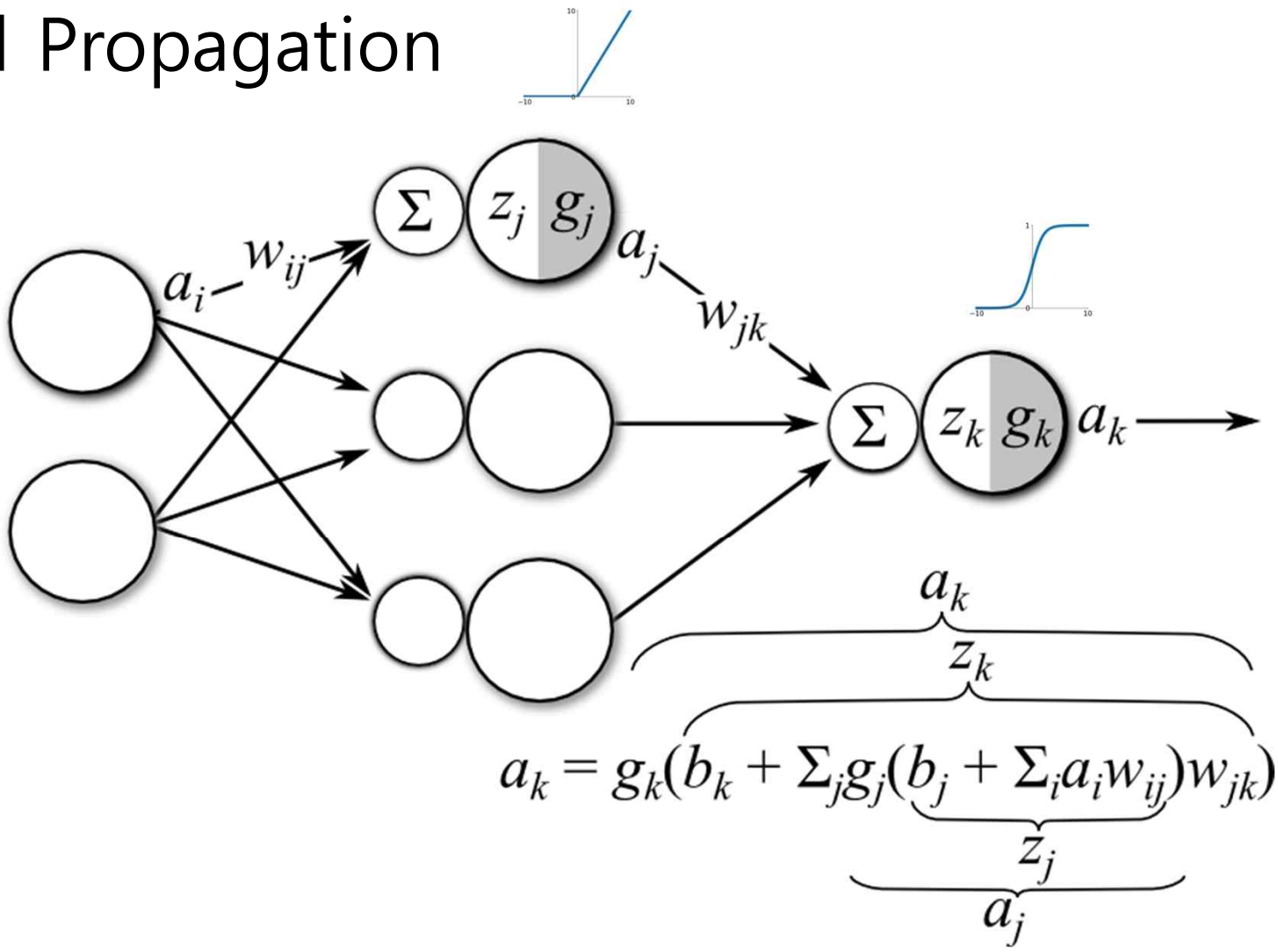
$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

```
def CrossEntropy(yHat, y):
    if y == 1:
      return -log(yHat)
    else:
      return -log(1 - yHat)
```

Forward Propagation



Forward Propagation



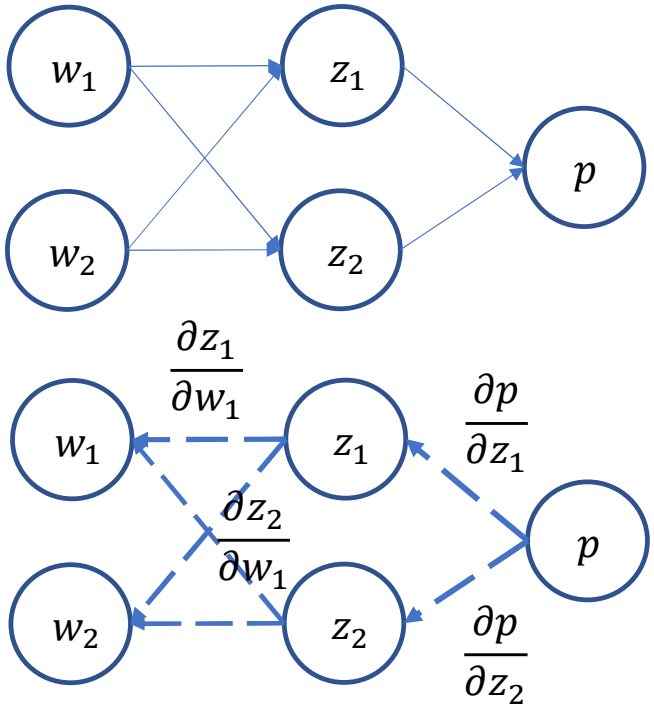
Backward Propagation 기초 공식

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

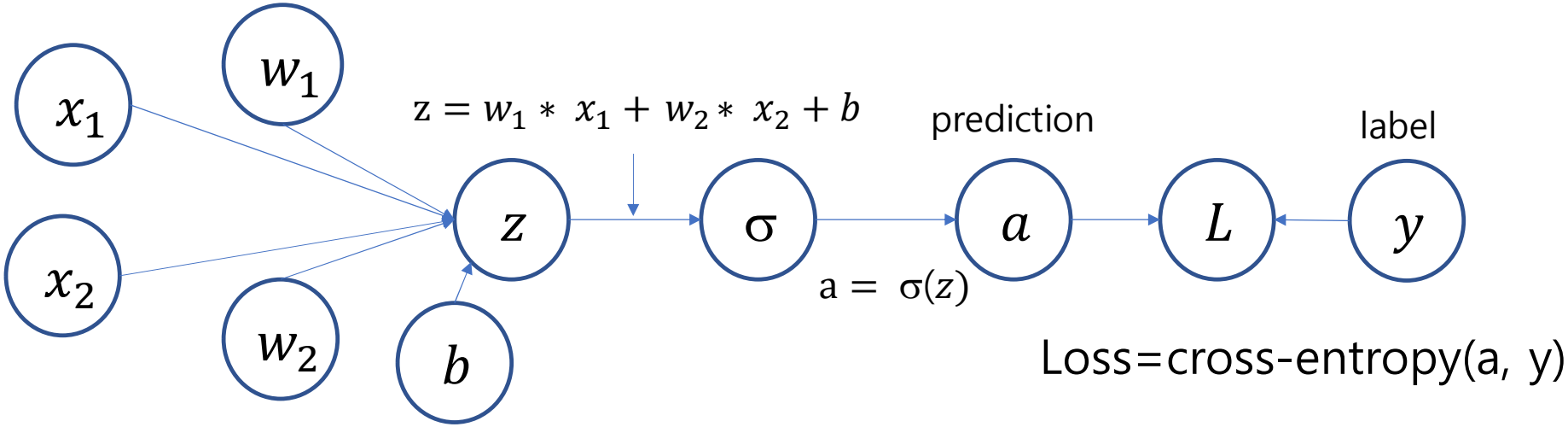
- Chain Rule :

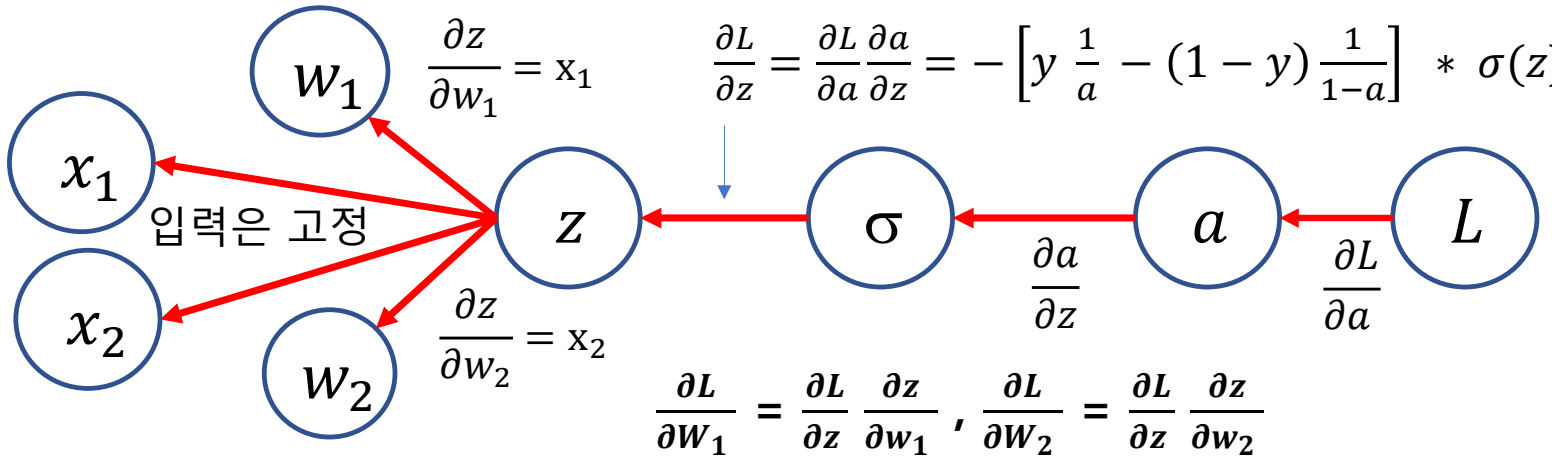
$$p = f(z_1, z_2) : \text{합성함수}$$
$$z_1 = f(w_1, w_2)$$
$$z_2 = f(w_1, w_2)$$

$$\frac{\partial p}{\partial w_1} = \frac{\partial p}{\partial z_1}\frac{\partial z_1}{\partial w_1} + \frac{\partial p}{\partial z_2}\frac{\partial z_2}{\partial w_1}$$
$$\frac{\partial p}{\partial w_2} = \frac{\partial p}{\partial z_1}\frac{\partial z_1}{\partial w_2} + \frac{\partial p}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$



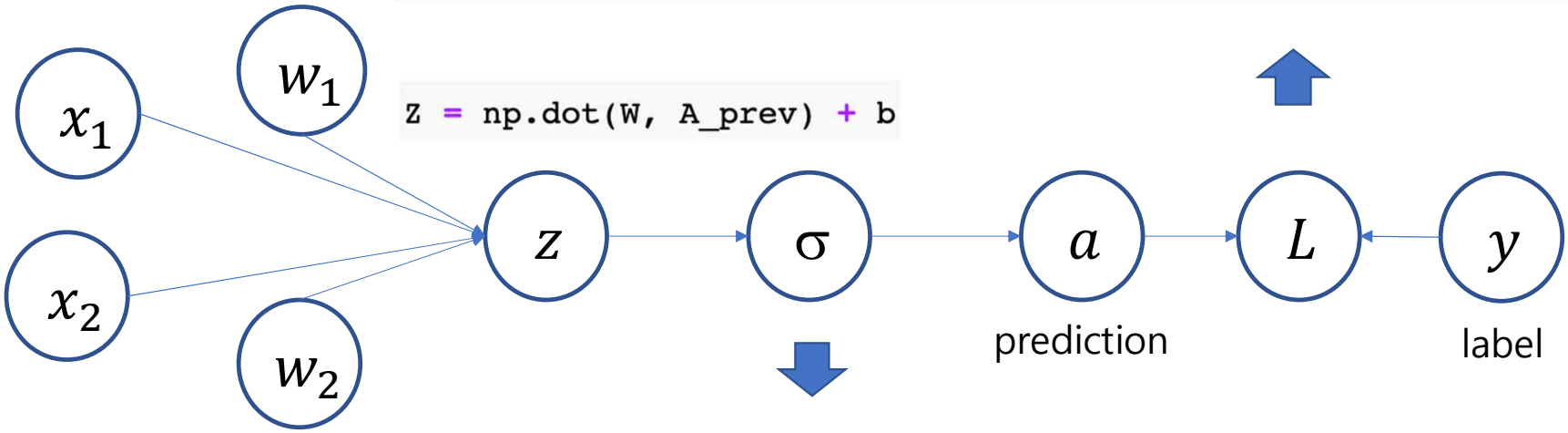
Example : 2 개의 feature 를 가진 1 layer + sigmoid activation





Forward Propagation 구현

```python
def compute_cost(AL, y):
    m = y.shape[1]
    cost = - (1 / m) * np.sum(
        np.multiply(y, np.log(AL)) + np.multiply(1 - y, np.log(1 - AL)))
    return cost
```




```python
Z = np.dot(W, A_prev) + b
```

```python
def sigmoid(Z):
    A = 1/(1+np.exp(-Z))
    return A, Z
```

or

```python
def relu(Z):
    A = np.maximum(0,Z)
    return A, Z
```

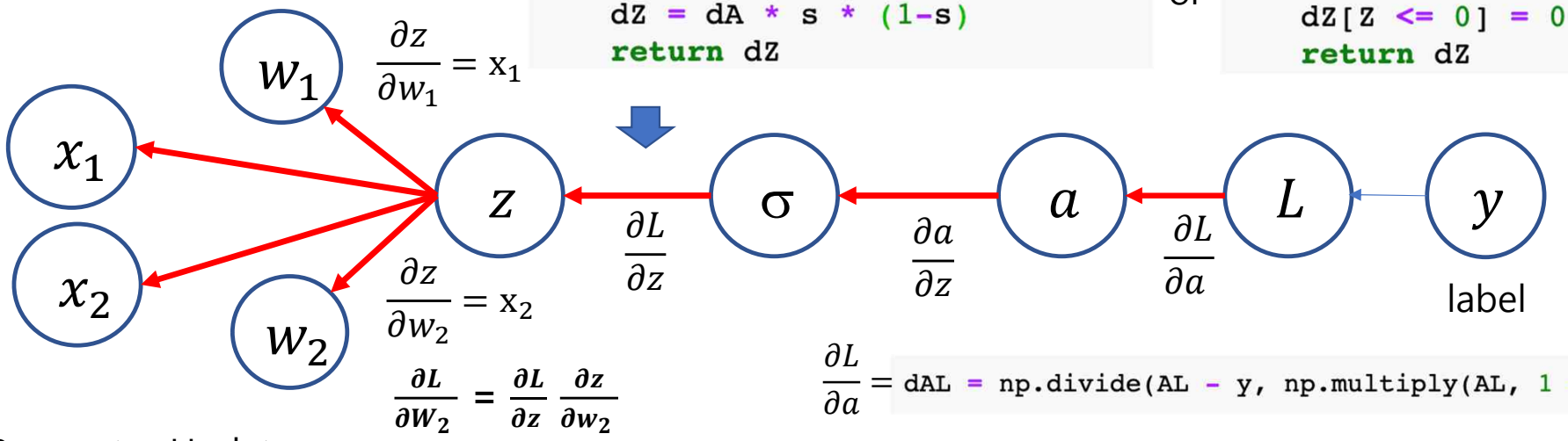
Backward Propagation 구현

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1}$$

```
def sigmoid_backward(dA, cache):
    Z = cache
    s = 1/(1+np.exp(-Z))
    dZ = dA * s * (1-s)
    return dZ
```

or

```
def relu_backward(dA, cache):
    Z = cache
    dZ = 1
    dZ[Z <= 0] = 0
    return dZ
```



$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_2}$$

$$\frac{\partial L}{\partial a} =$$ `dAL = np.divide(AL - y, np.multiply(AL, 1 - AL))`

Parameter Update

```
for l in range(1, L + 1):
    parameters["W"+str(l)] = parameters["W"+str(l)] - learning_rate * grads["dW"+str(l)]
    parameters["b"+str(l)] = parameters["b"+str(l)] - learning_rate * grads["db"+str(l)]
```

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}
$$

$$
\frac{dL}{da} = -[y \cdot \frac{1}{a} - (1-y) \cdot \frac{1}{1-a}] = -\frac{y}{a} + \frac{1-y}{1-a}
$$

$$
= -\frac{y(1-a)}{a(1-a)} - \frac{a-ay}{a(1-a)} = \frac{-y+ay+a-ay}{a(1-a)} = \frac{a-y}{a(1-a)}
$$

Backpropagation (Chain Rule 적용)

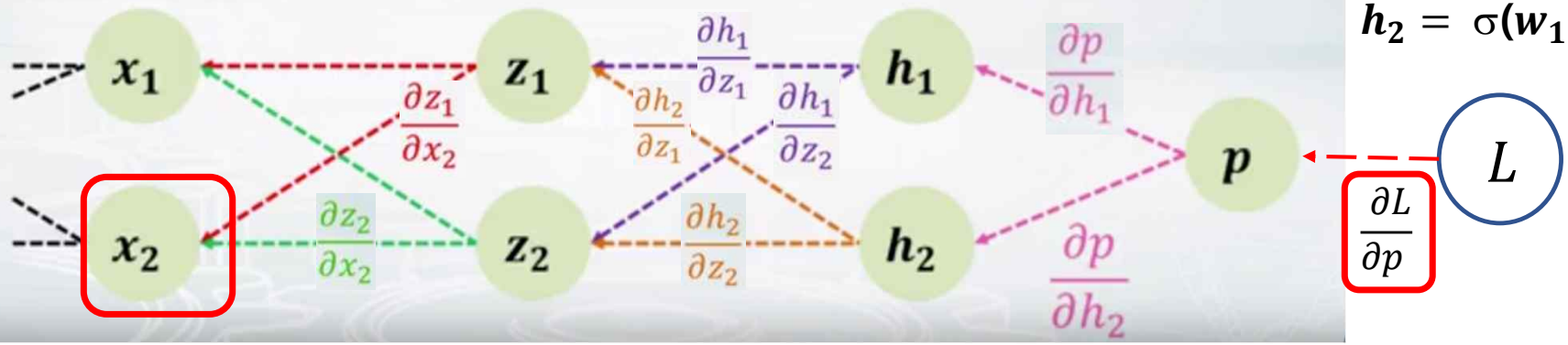
* 같은 색으로 표시된 부분은 한번 계산하면 여러 번 reuse

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

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

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

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



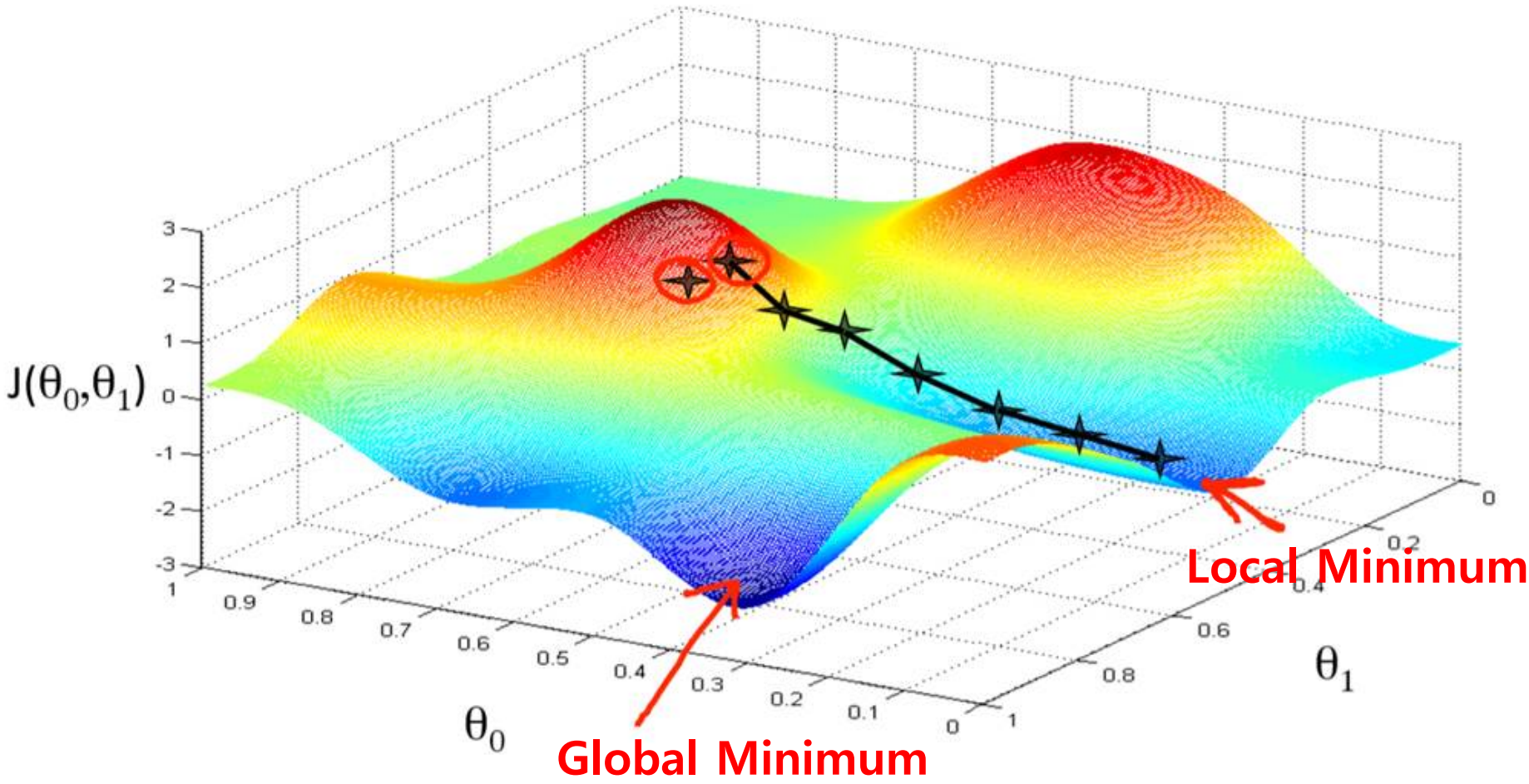
$$\frac{\partial L}{\partial p} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial h_1} + \frac{\partial L}{\partial p}\frac{\partial p}{\partial h_2}$$

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

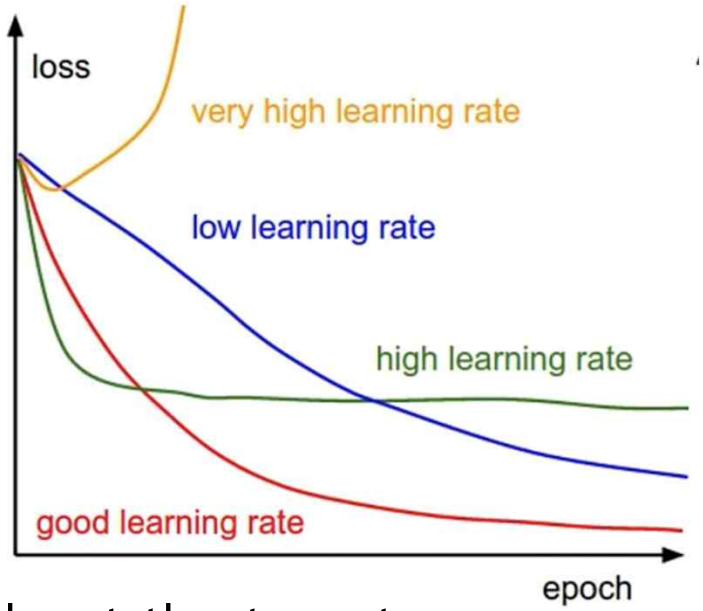
Backpropagation 요약

- 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

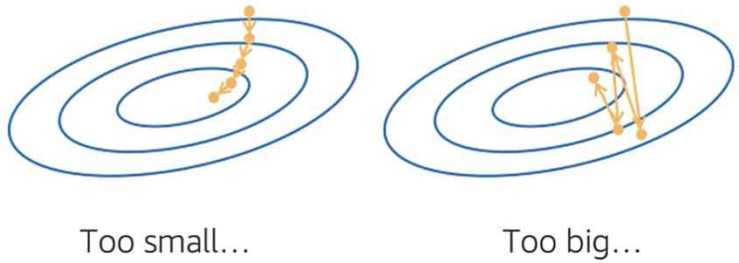
Global Minima / Local Minima



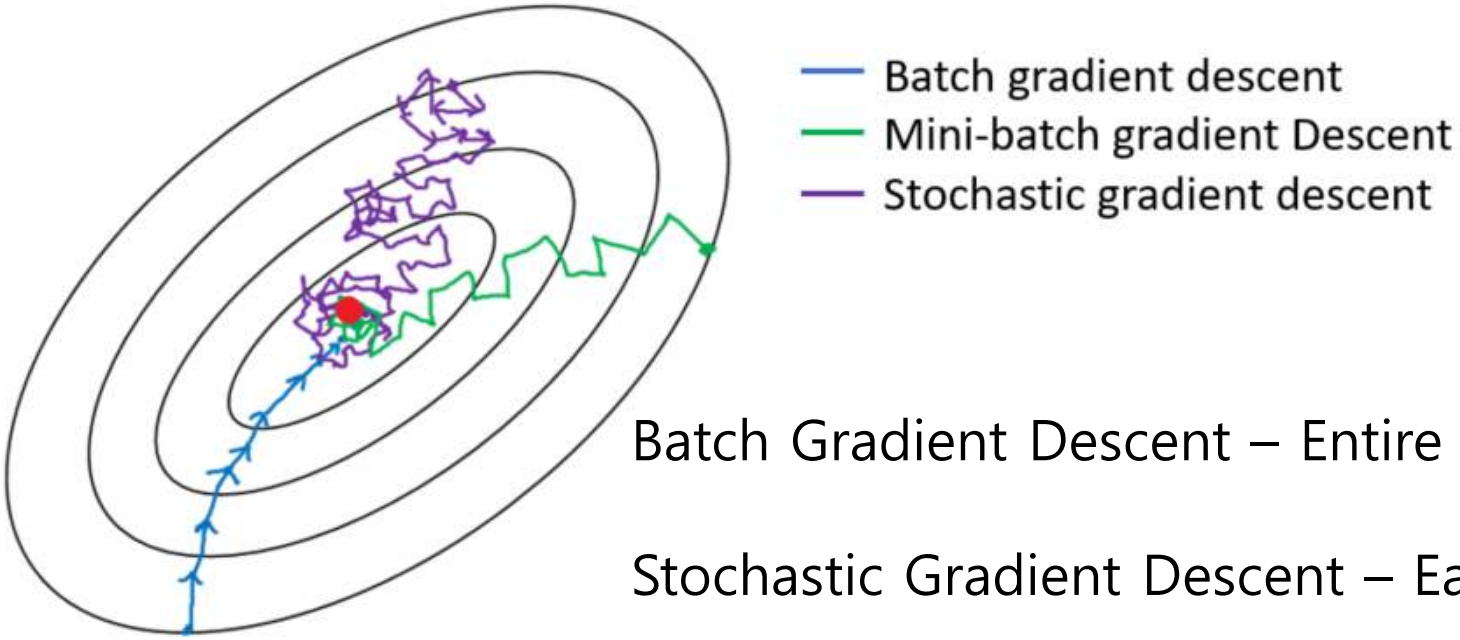
Learning Rate (α)



- Step size
- Range : 1e-6 ~ 1.0 (default 0.01)
  - High learning rate – fast learning, may overshoot the target
  - Low learning rate – slow learning, may take long time
- Adaptive Learning Rates
  초기값을 크게 주고 학습 진행에 따라 slow down



SGD (Stochastic Gradient Descent, 확률적 경사하강법)

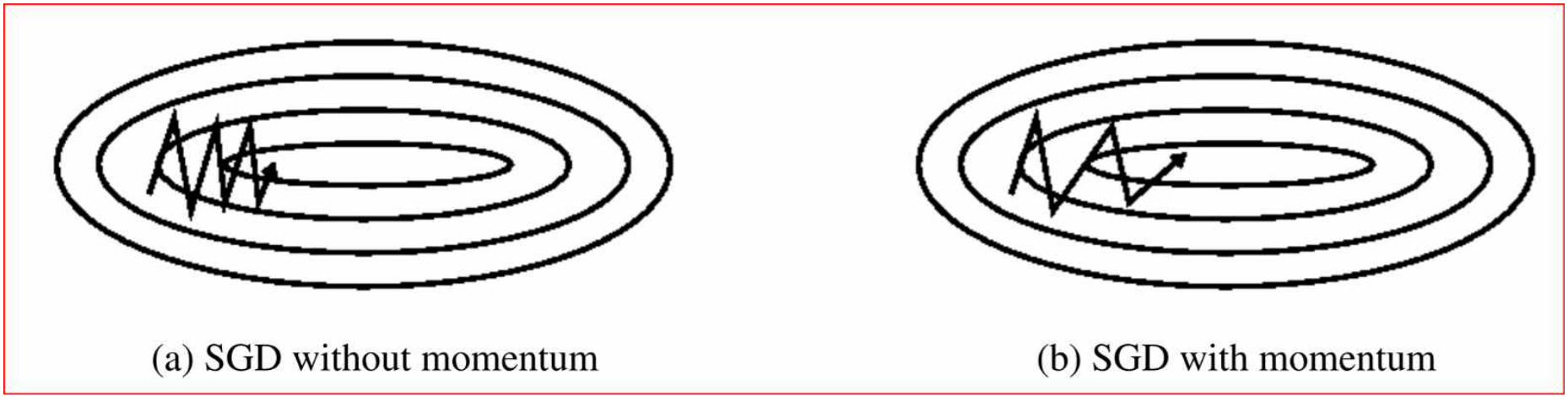


Batch Gradient Descent – Entire samples

Stochastic Gradient Descent – Each sample (size 1)

Mini-batch Gradient Descent – small size of samples

Momentum : 방향성을 유지하며 가속

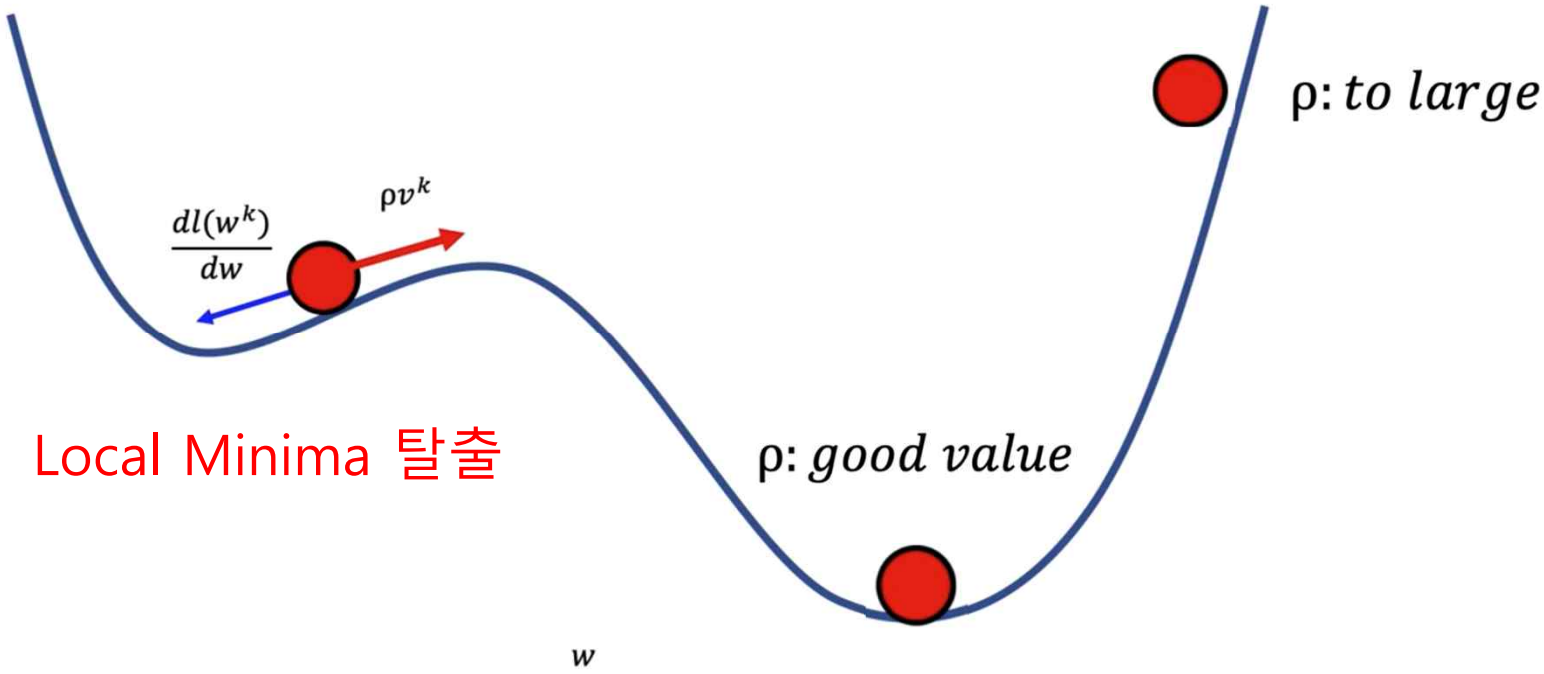


(a) SGD without momentum

(b) SGD with momentum

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

Momentum 과 Local Minima

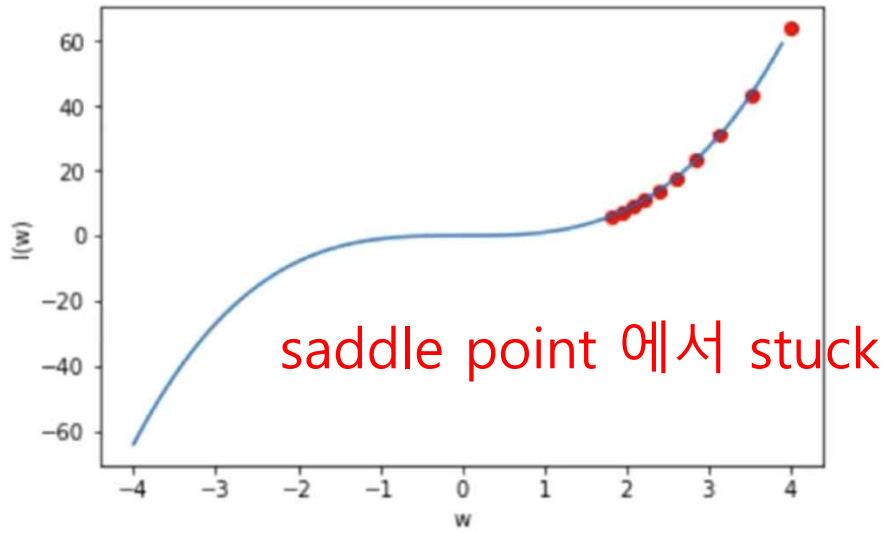


Momentum 과 weight update 비교

ρ: momentum

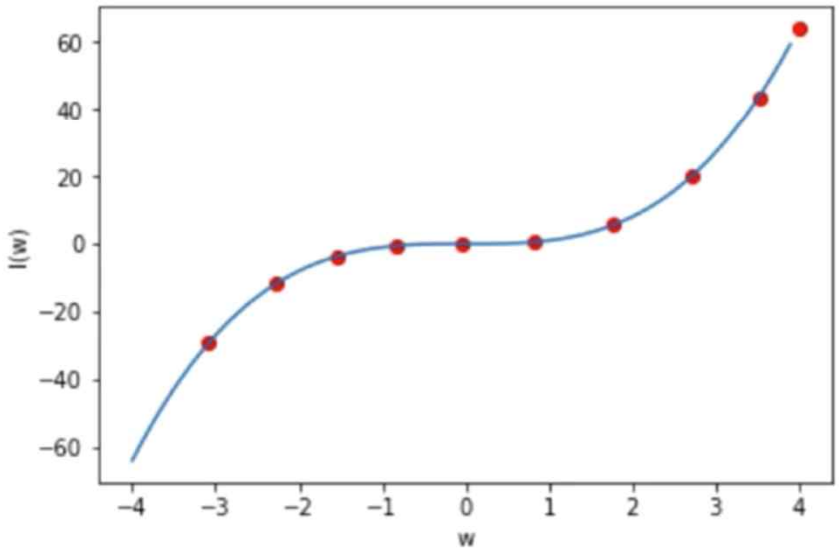
η: learning rate (lr)

lr=0.01,momentum=0



saddle point 에서 stuck

lr=0.01,momentum=0.90



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

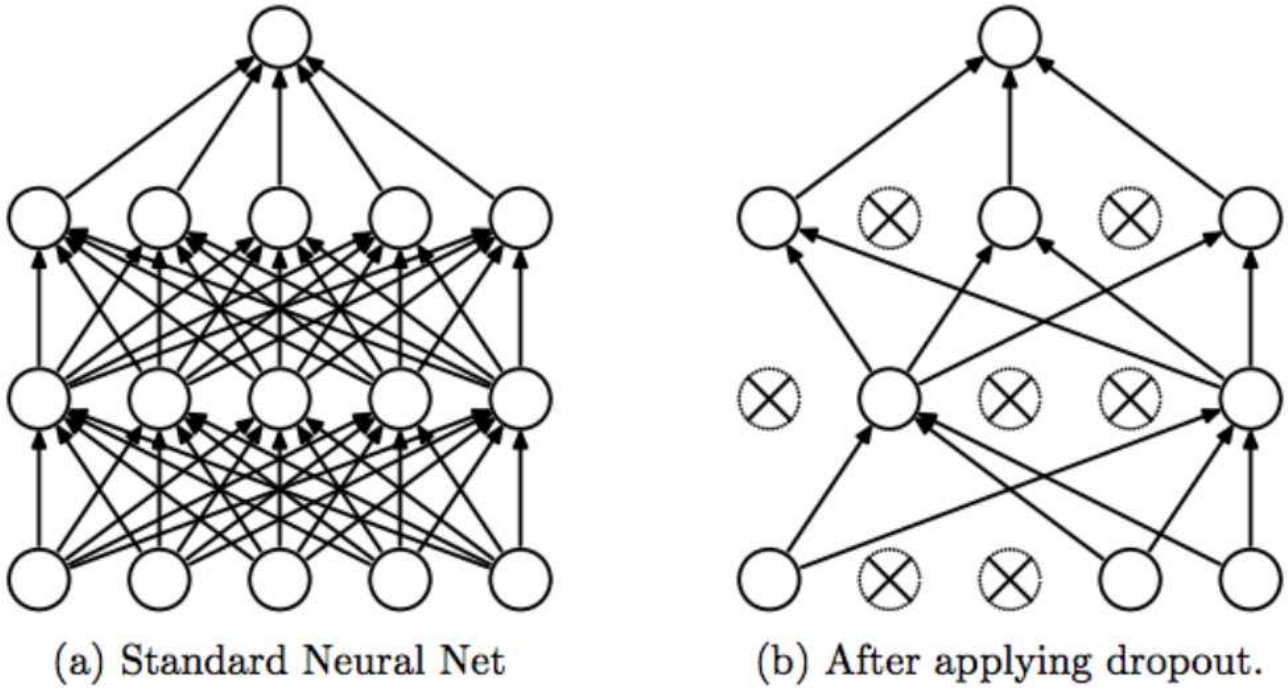
- 정의 – 전체 dataset 이 neural network 을 통해 한번 처리된 것
- Epoch 은 model 의 training 시에 hyperparameter 로 횟수 지정
- 하나의 epoch 은 한번에 처리하기 어려운 size 이므로 여러 개의 batch 로 나누어 처리
- Parameter training 을 위해서는 여러 번 epoch 을 반복해야 한다.
- One epoch 내에서의 iteration 횟수는 total sample size / batch size
- Ex) 1 epoch = 4 iterations =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

Hyper-parameters

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

Dropout regularization

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



(a) Standard Neural Net

(b) After applying dropout.

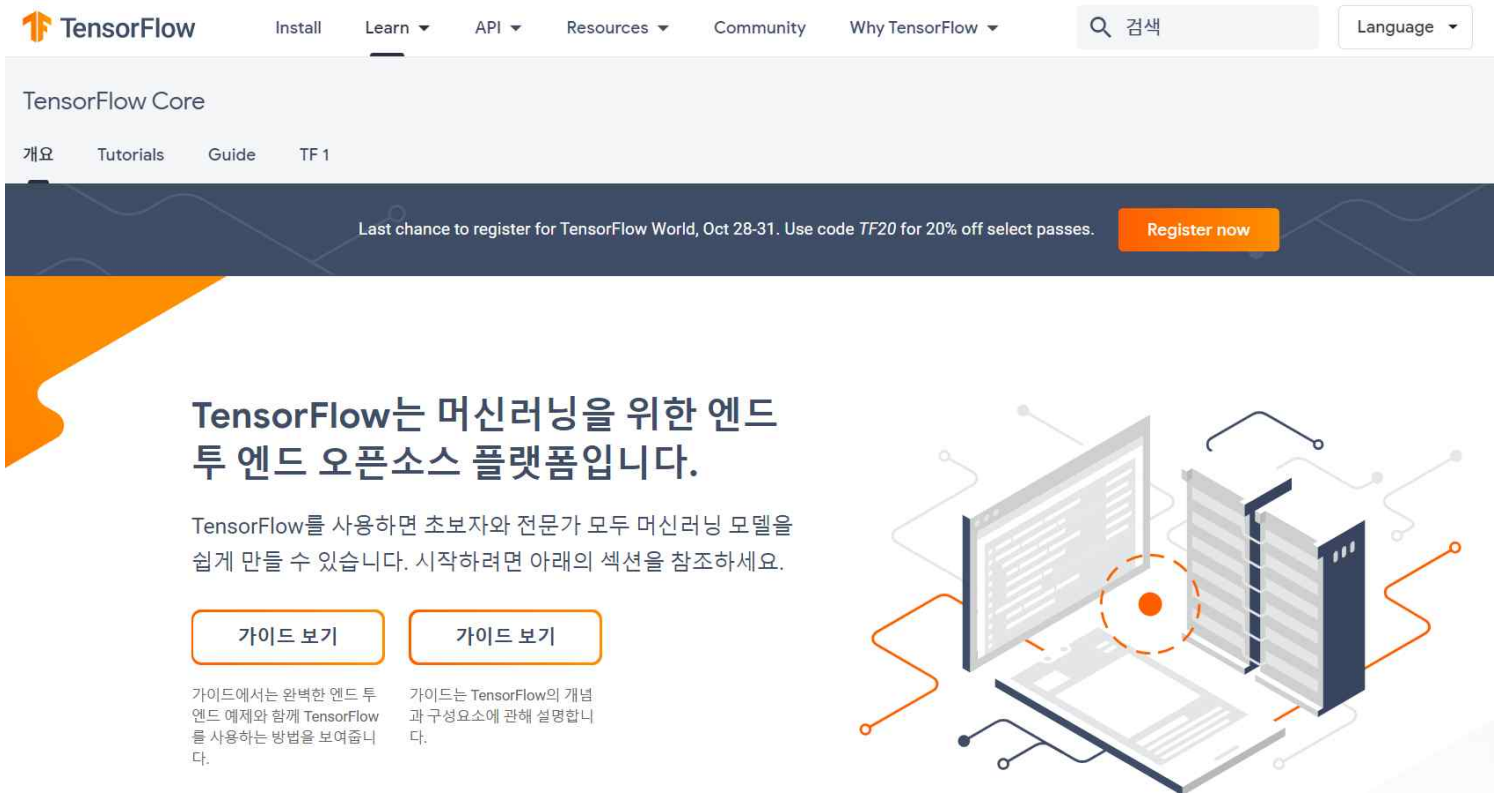
Hyper-parameter 값 결정 ?

- 정해진 RULE 이 없음
- 유사한 model 참조
- 경험에 의한 guessing
- Grid search – computationally expensive

Tensorflow 2.0

What is Tensorflow ?

- https://www.tensorflow.org/overview/?hl=ko



Tensorflow Installation

- pip install --upgrade tensorflow

- import tensorflow as tf
- tf.__version__

일반용 – Sequential API

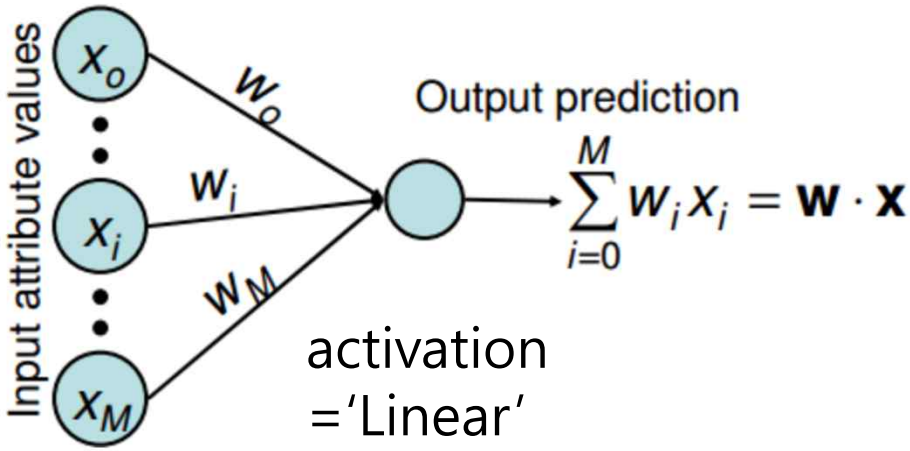
```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

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

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

전문가용 – Subclassing API

```python
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.flatten = Flatten()
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.conv1(x)
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)
grads = tape.gradient(loss_value, model.trainable_variable
optimizer.apply_gradients(zip(grads, model.trainable_varia
```

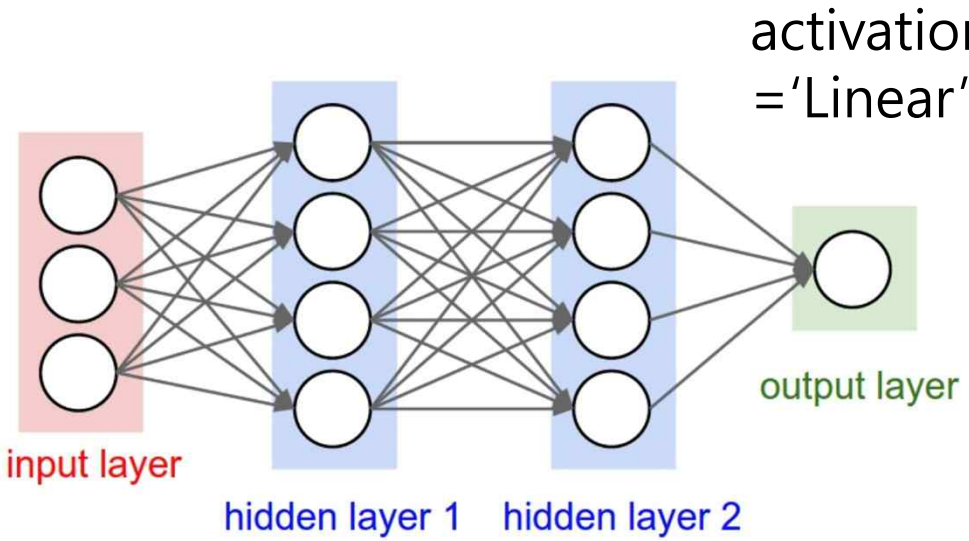
Neural Network 을 이용한 Regression

NN 을 이용한 Linear Regression



One-Layer

Multiple-Layers



실습 : 100. 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)

Logistic Regression

(이진 분류, Binary Classification)

Sigmoid 함수

- S curve 형성
- Logistic 함수

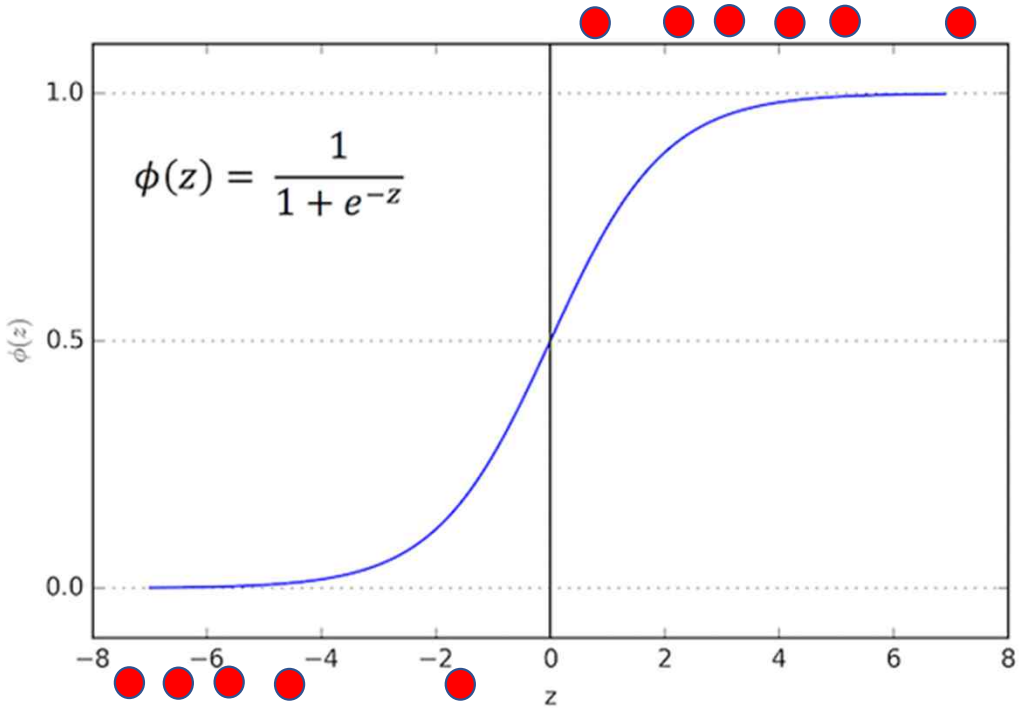
logit

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

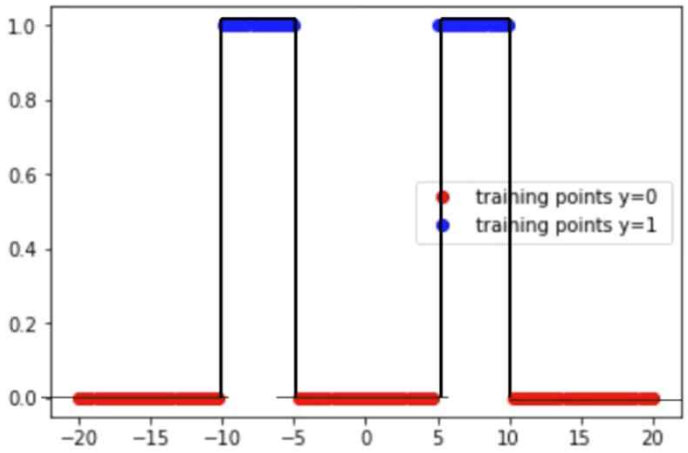
$\hat{y}$ = 0 if f(z) < 0.5

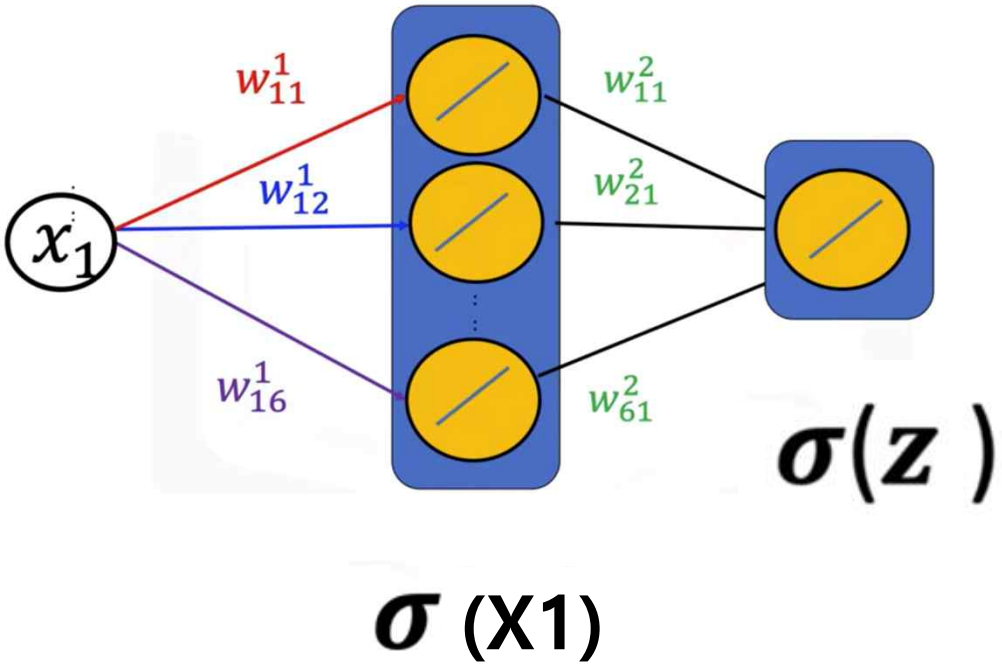
1 if f(z) ≥ 0.5

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



NN 을 이용한 Logistic Regression





σ (X1)

실습 : 110. Malware Detection - Logistic Regression

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

CNN (Convolutional Neural Network)

Multiple Levels of Abstraction

Pixel 정보 인식

↓

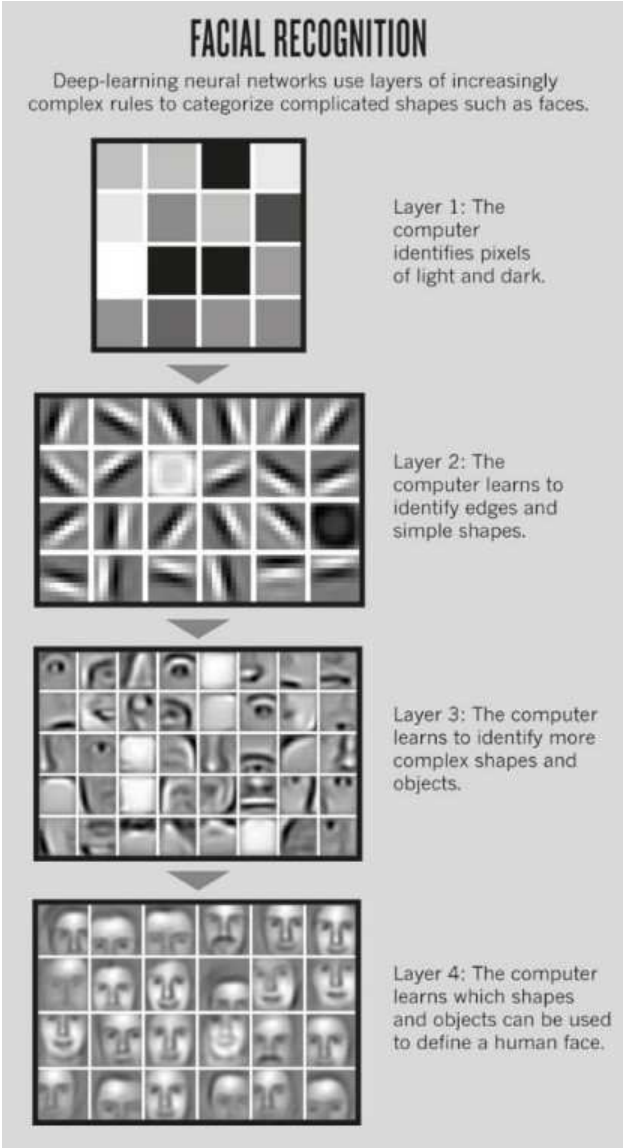
Edge/Simple Shape 특성 학습

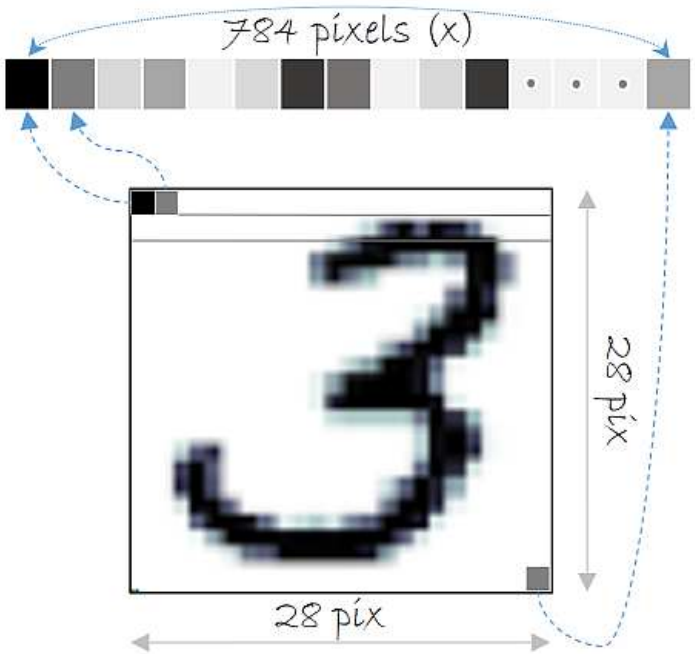
↓

Complex Shape 특성 학습

↓

얼굴인식에 필요한 특성 학습





2차원 이미지의 1차원 vector 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

CNN 의 특별한 Layers

- **Convolutional Layer (합성곱층)**

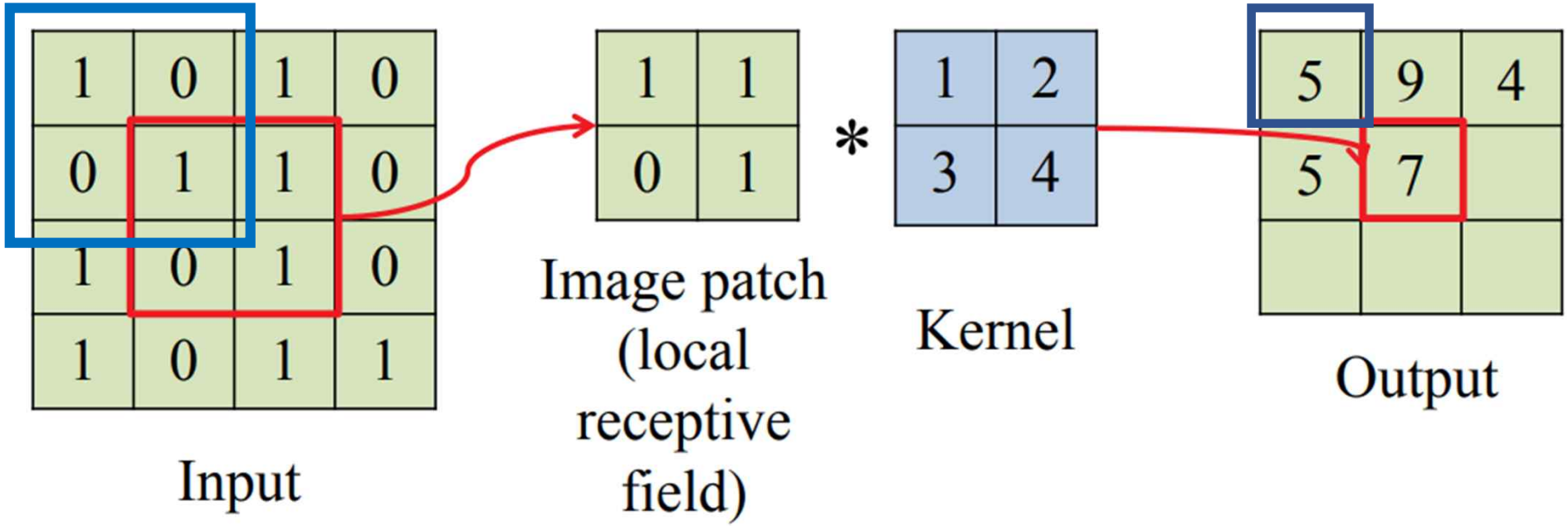
  - Image 정보의 공간적 지역 특성 보존
  - Filter (Kernel) 을 이용한 이미지 특성 추출

- **Pooling Layer (풀링층)**

  - Image data 의 정보 손실 없는 압축

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

How Convolution works ?

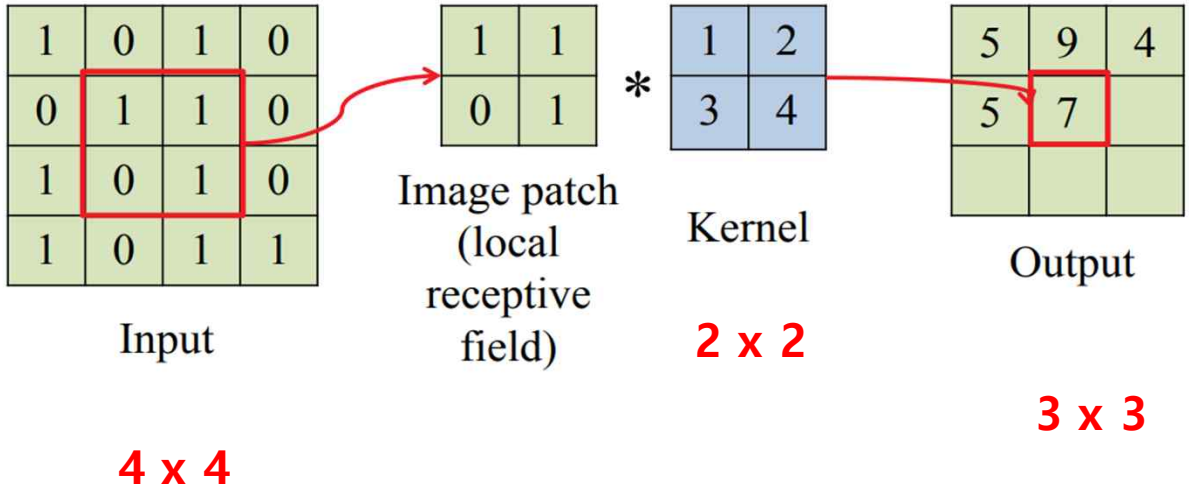


Kernel_size=(2, 2), stride=(1, 1), No padding

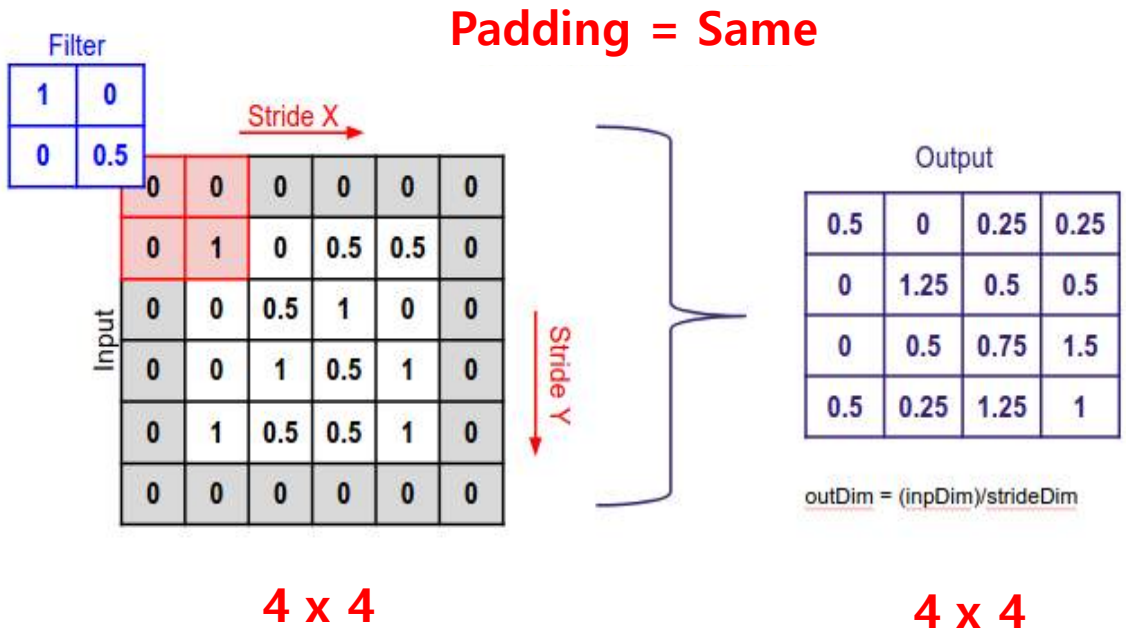
Padding

- Convolution 때 마다 image 의 edge 정보가 소실

Padding = **Valid (No Padding)**

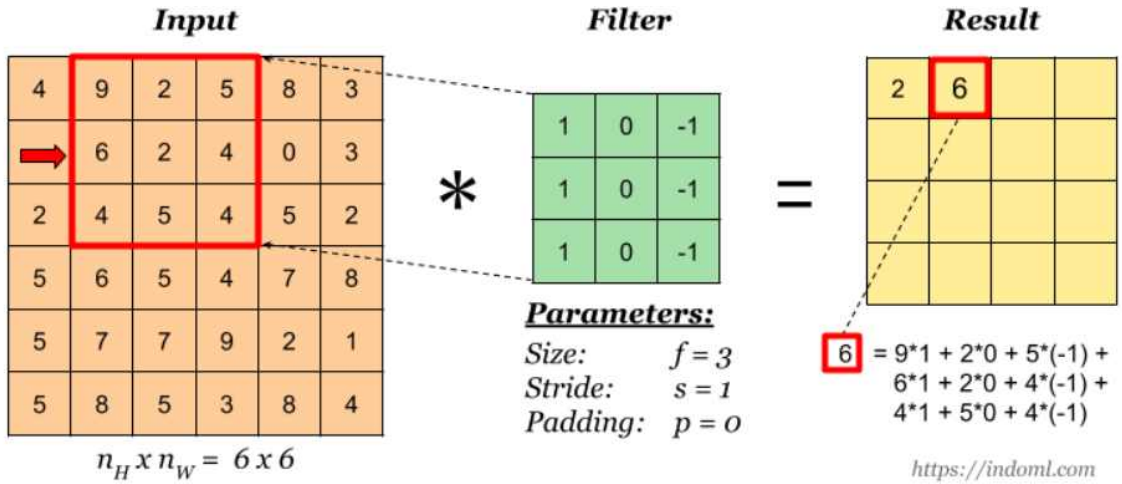


- Input size = output size (➔ Input 의 주위에 0 pixel padding)
- Padding = “same” convolution



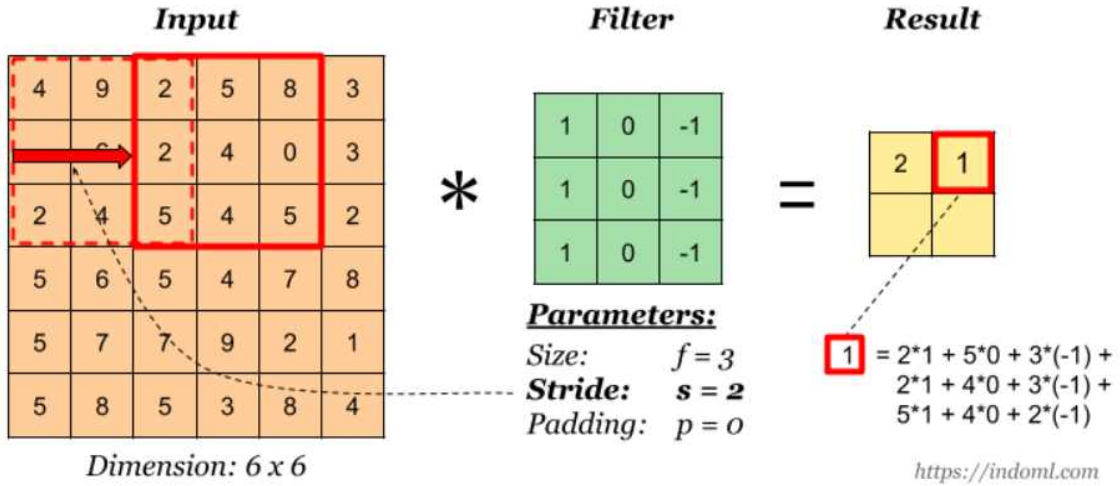
Striding

Stride = 1



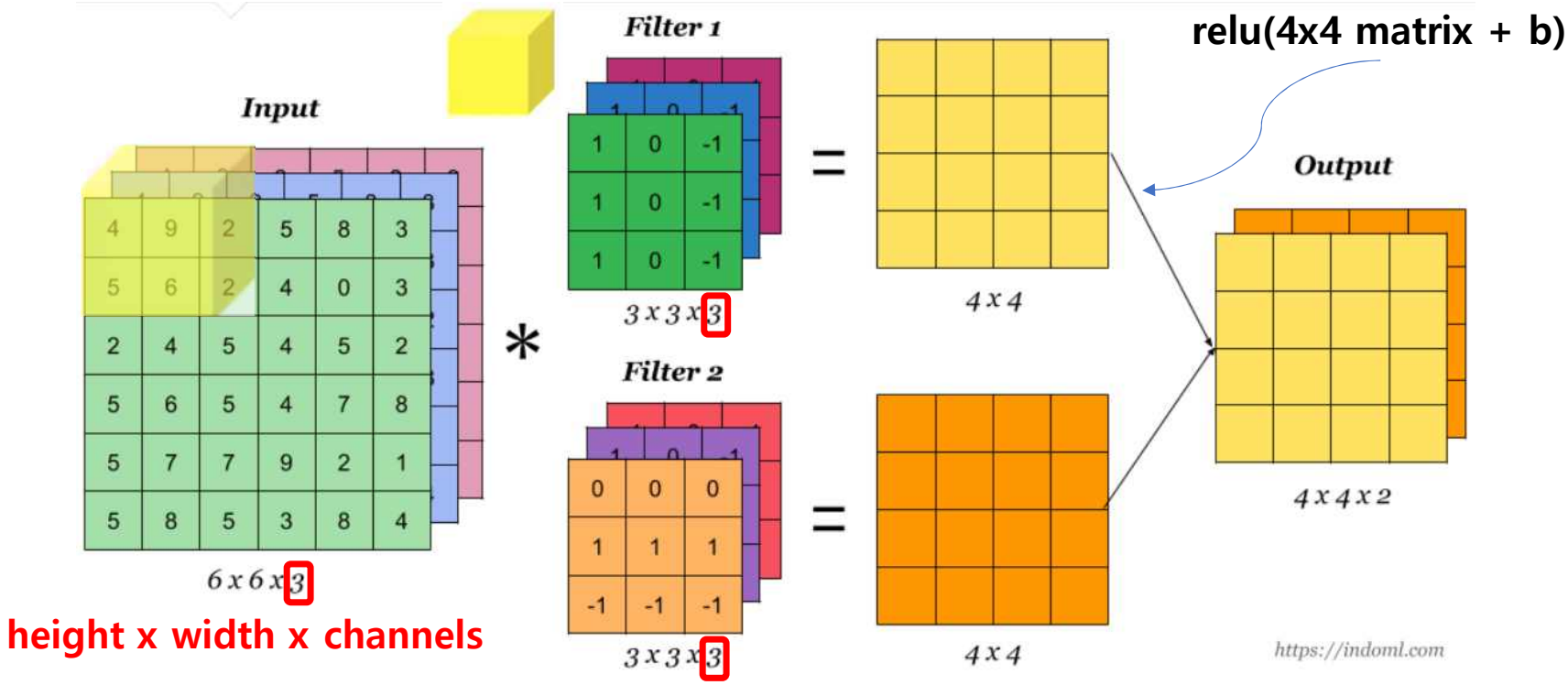
4 x 4

Stride = 2



2 x 2

Convolutions over Volumes (RGB images)



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

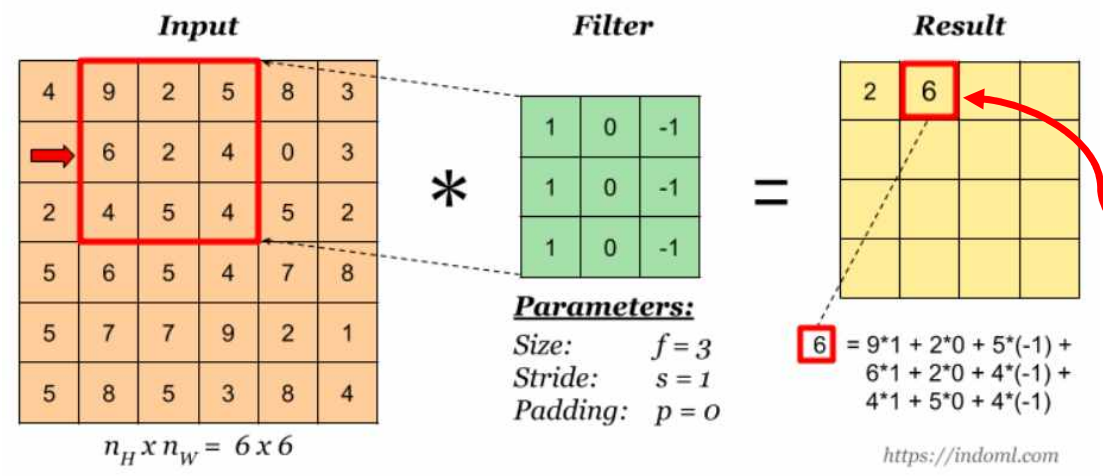


Image 전체가 아니라 3x3 area 에만 연결

*** Fully connected 의 경우 image 전체와 연결**

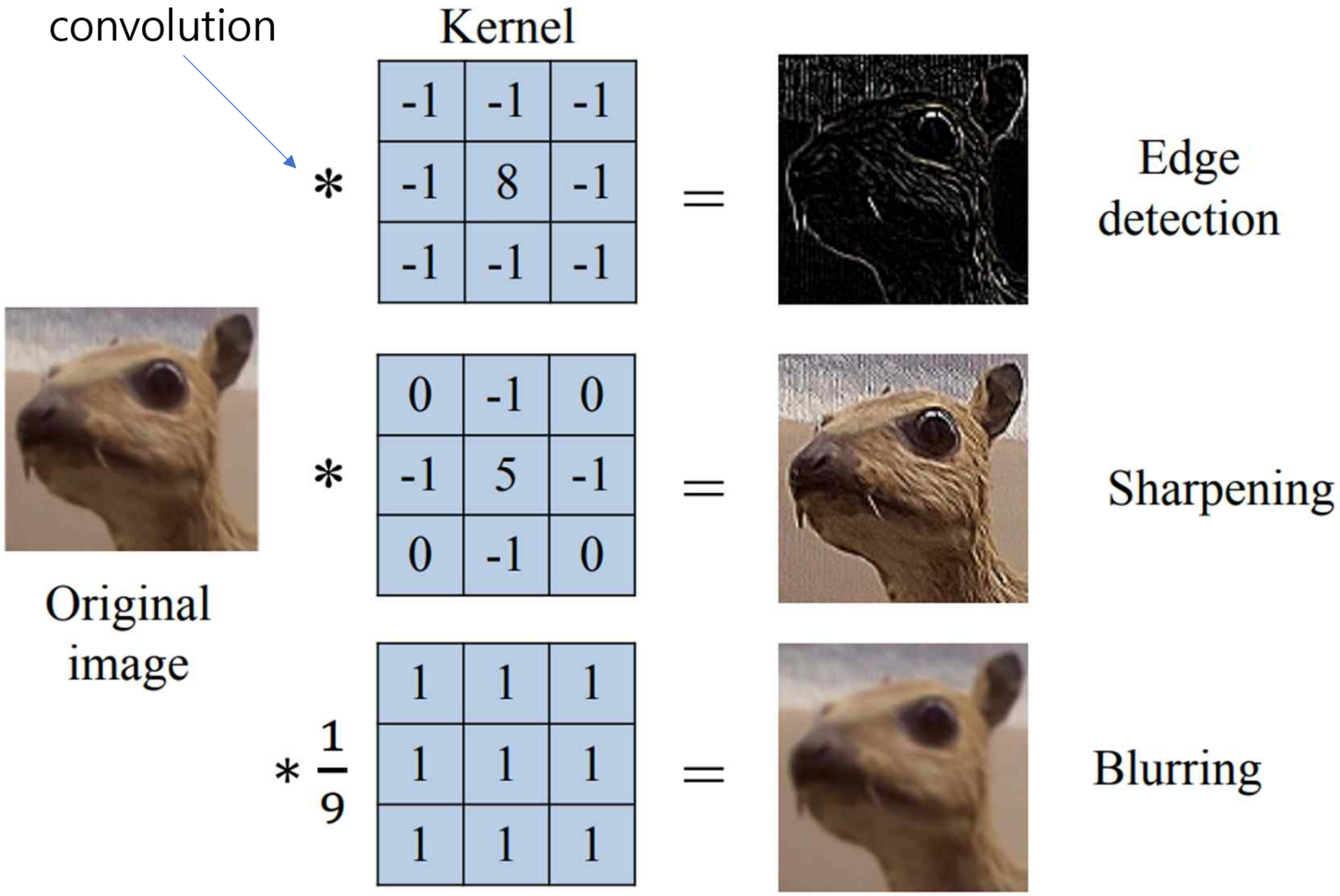
2. **Parameter Sharing**

- 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



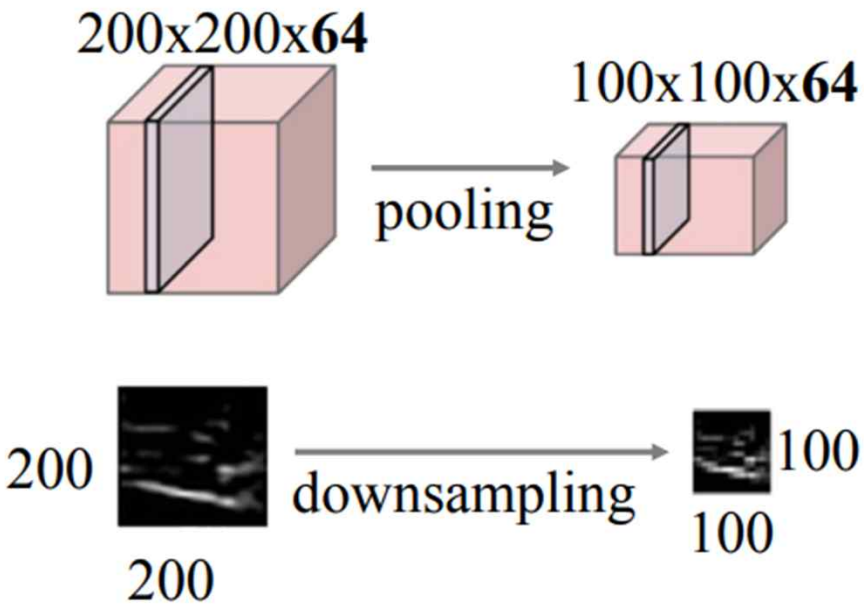
- CNN 이전에는 edge detection filter 를 computer vision 전문가들이 모두 manually 만들어 줌

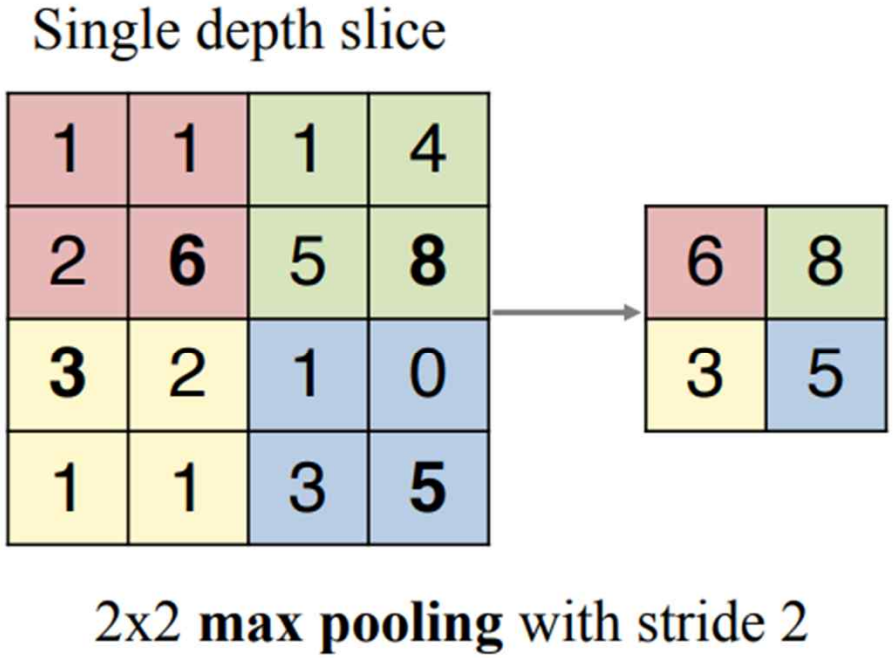
  ex) 수직 / 수평 / 45 도 / 명암 구분 filter 등

  ➔ 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)





Pooling Layer 의 2 가지 특성

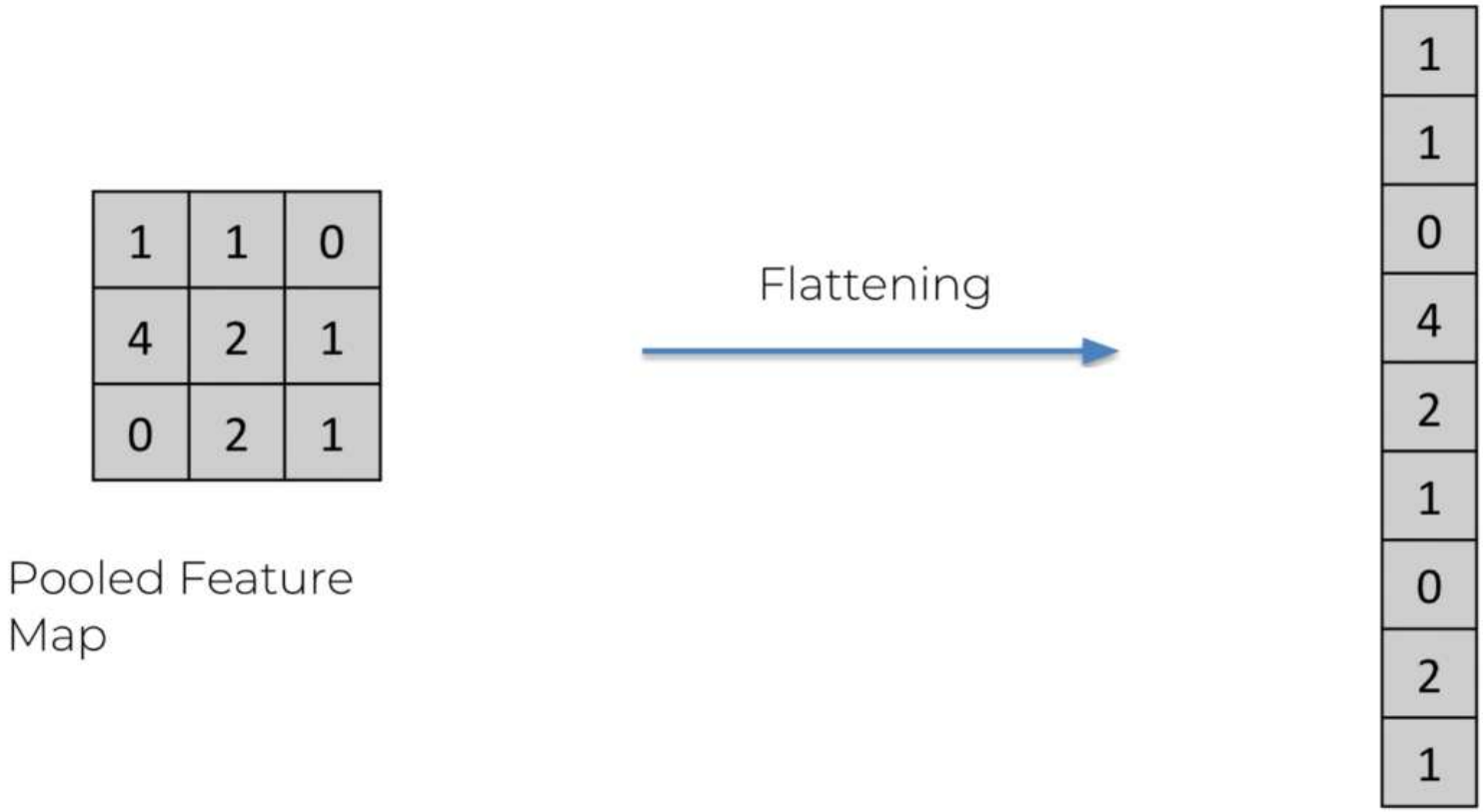
1. Positional Invariance

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

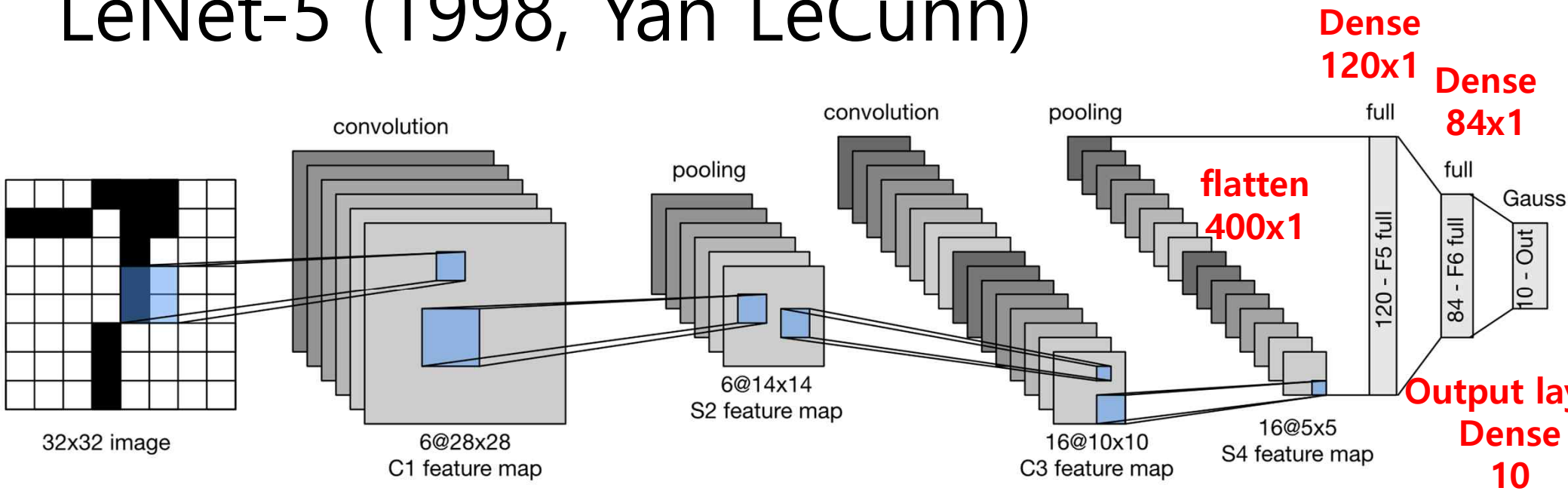
2. Size 축소

   - 계산량을 크게 줄임
   - 과적합 방지

What is Flattening ?



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 : Le Net 을 이용한 손글씨 인식

1. Keras 를 이용한 Le Net 구축

2. 구축한 Le Net model 을 이용하여 Mnist 손글씨 분류

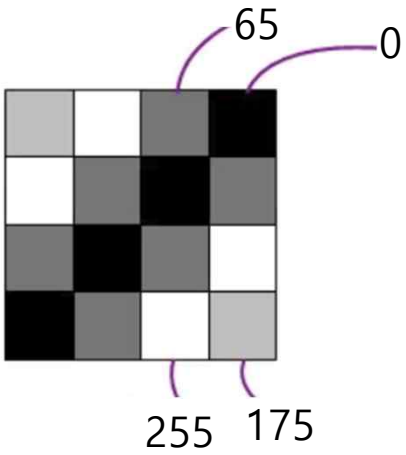
Mnist Dataset 소개

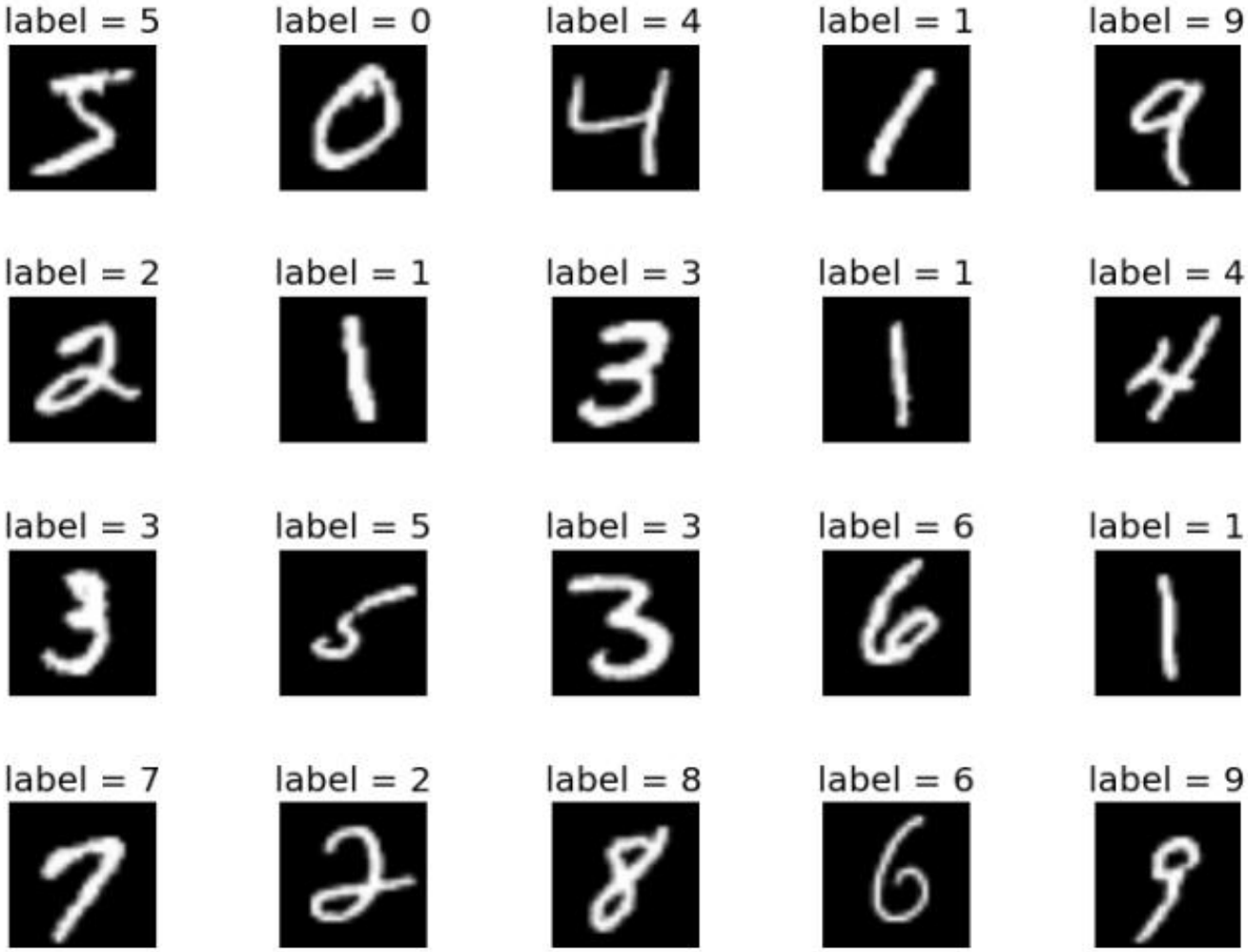
- Mixed National Institute of Standards and Technology

- 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

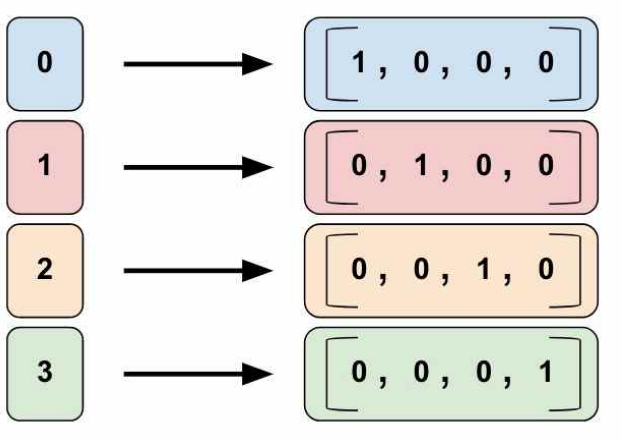
Pixel 의 구성

0 – black
255 - white

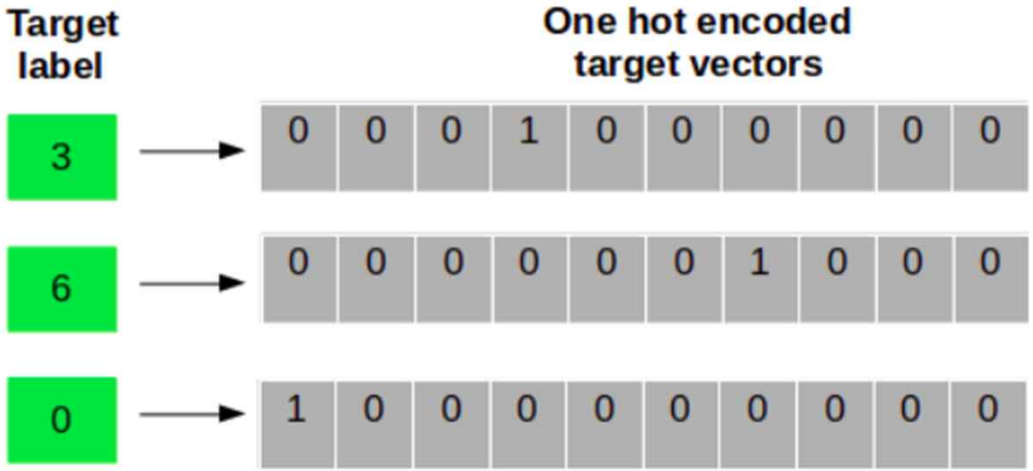




One-Hot encoding



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)

실습 : Deeper CNN 을 이용한 CIFAR-10 분류

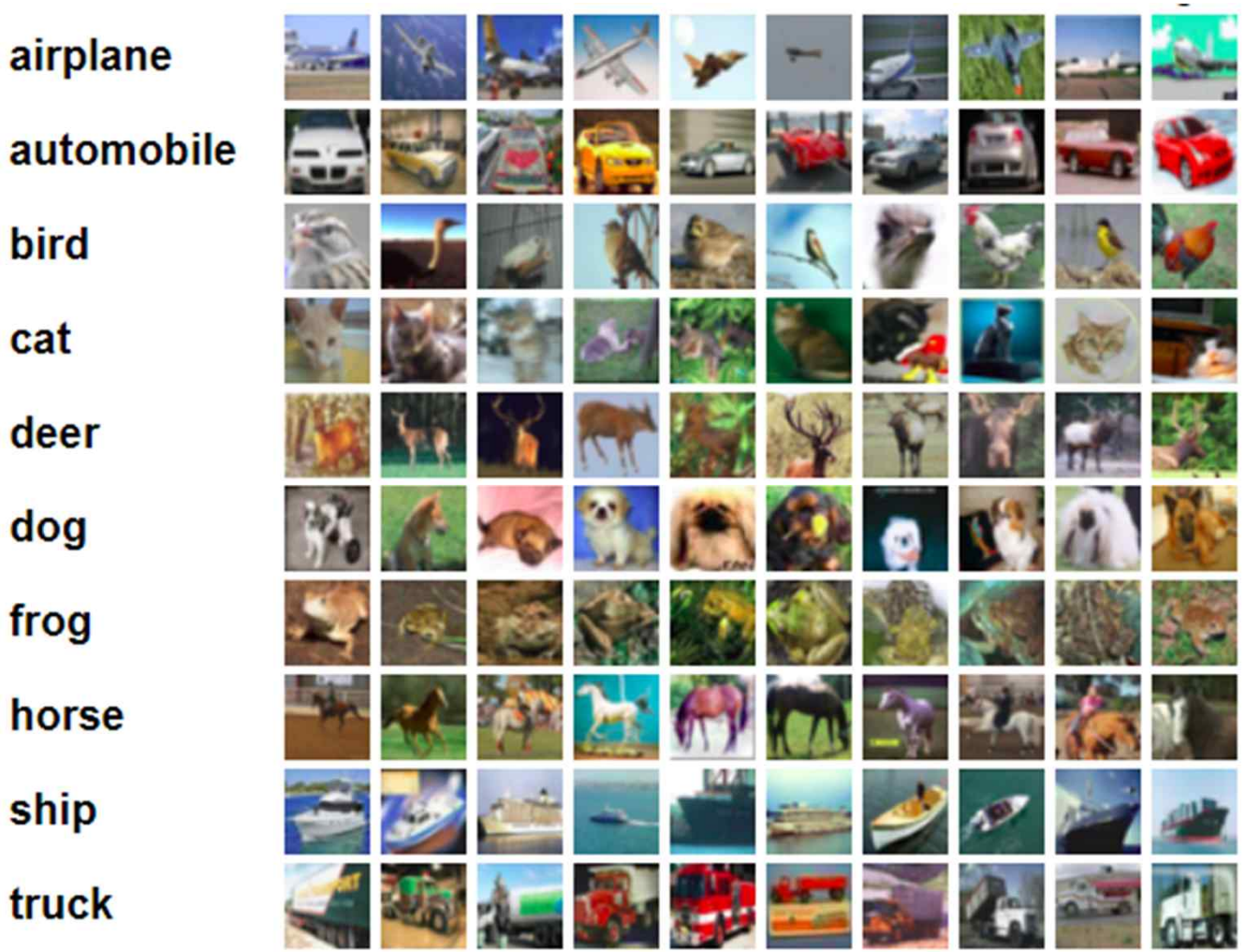
1. CIFAR-10 dataset 은 32x32 color image 를 가진 10개의 class (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)

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

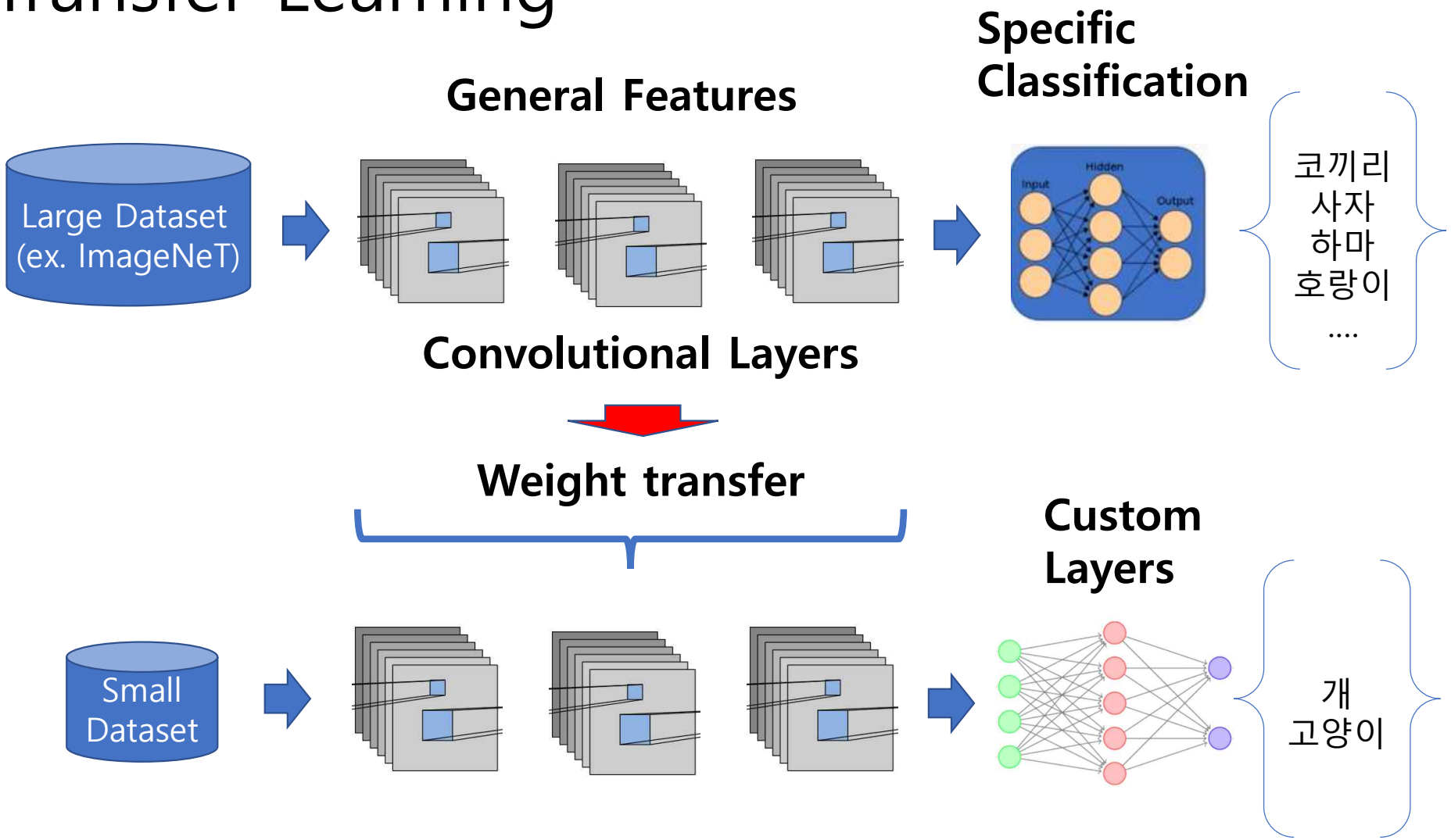
Dropout : 0.5

Output

Dense
10
Activation : softmax

Transfer Learning (전이학습)

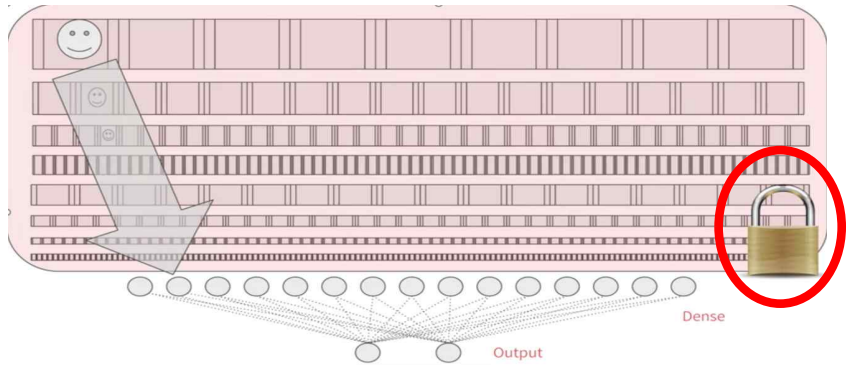
Transfer Learning



Transfer Learning Strategy

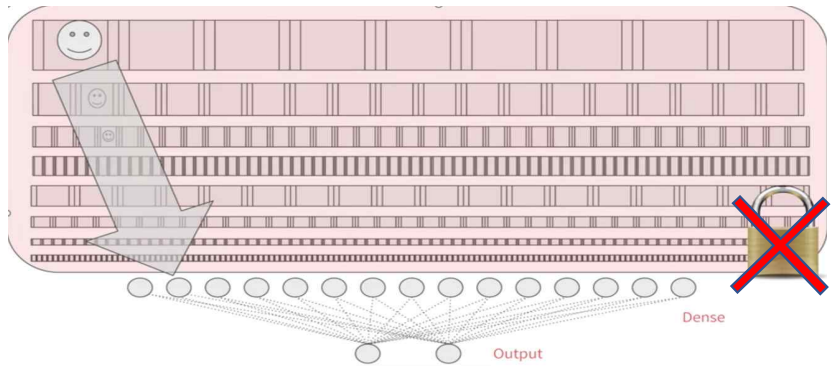
- Strategy 1

CNN layer 는 freeze 하고,
추가한 완전연결층만 새로이 train



- Strategy 2

전체 Layer 를 매우 작은 learning rate 로 re-train



Transfer Learning 고려사항

- **목적에 맞는 dataset 선택**

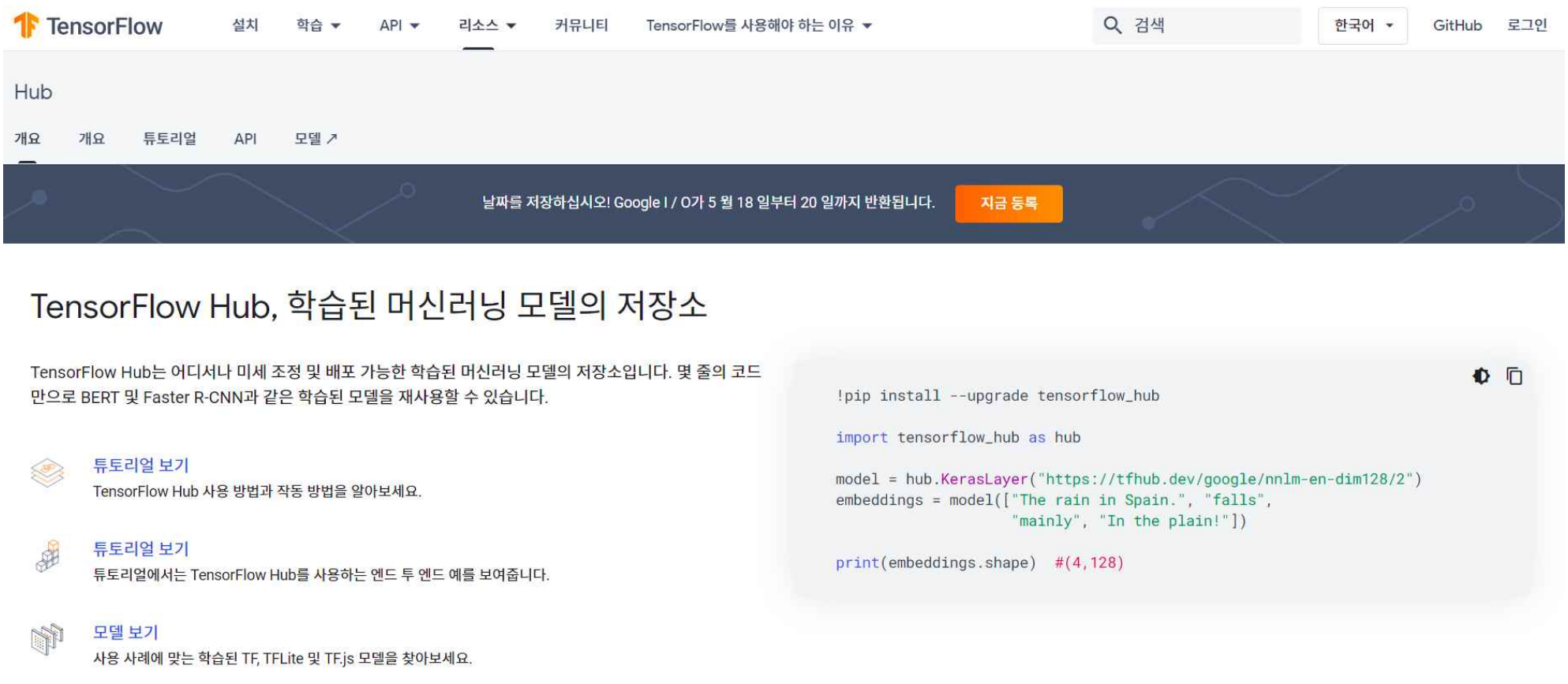
  ex) Cat & Dog 구분 → ImageNet 에 포함

  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)

TensorFlow Hub



실습 : 150. 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

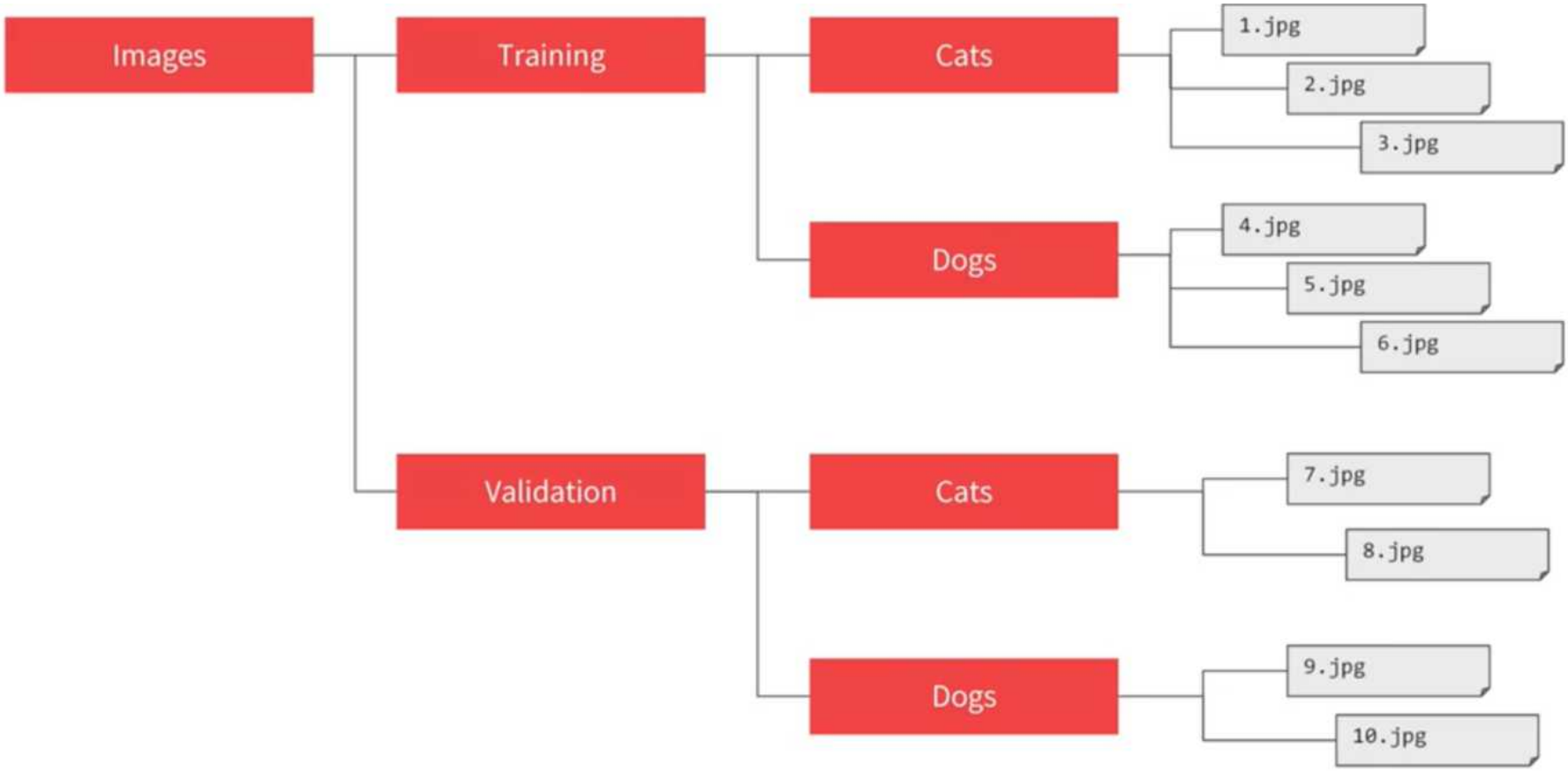
```
model = tf.keras.Sequential([
  MobileNet_feature_extractor_layer,
  tf.keras.layers.Dense(flowers_data.num_classes, activation='softmax')
])
```

MobileNet 소개

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

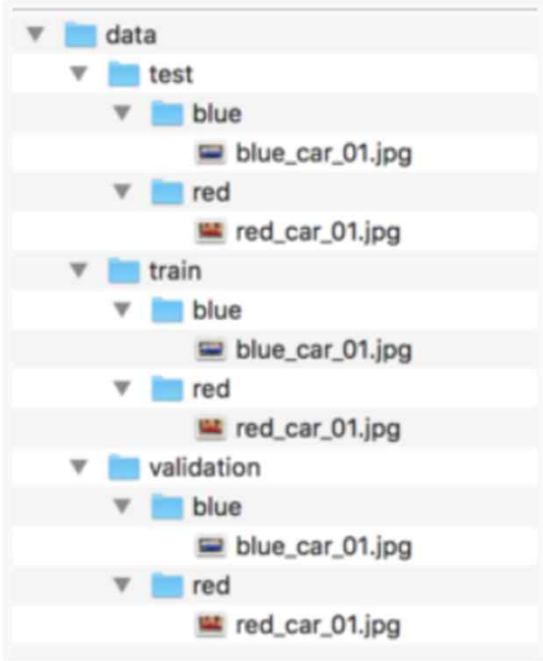
class name	probability	imagenet class id
Egyptian cat	0.478	285
tabby, tabby cat	0.300	281
tiger cat	0.167	282
remote control, remote	0.016	761
Siamese cat, Siamese	0.008	284

ImageDataGenerator folder structure



ImageDataGenerator methods

- .flow_from_directory
  - 대용량 data 를 directory 에서 직접 load
    (with data augmentation)
  - directory 구조에 의해 자동으로 label 인식



flow_from_directory 사용법

- from tensorflow.keras.preprocessing.image import ImageDataGenerator
- Instance 생성 :

  train_data_gen = ImageDataGenerator(rescale=1/255.)

- flow_from_directory method 호출 :

```
train_generator = train_data_gen.flow_from_directory(
                  train_dir,
                  target_size(150, 150),   ➔ image size 통일
                  batch_size=20,
                  class_mode='binary' or 'categorical)
```

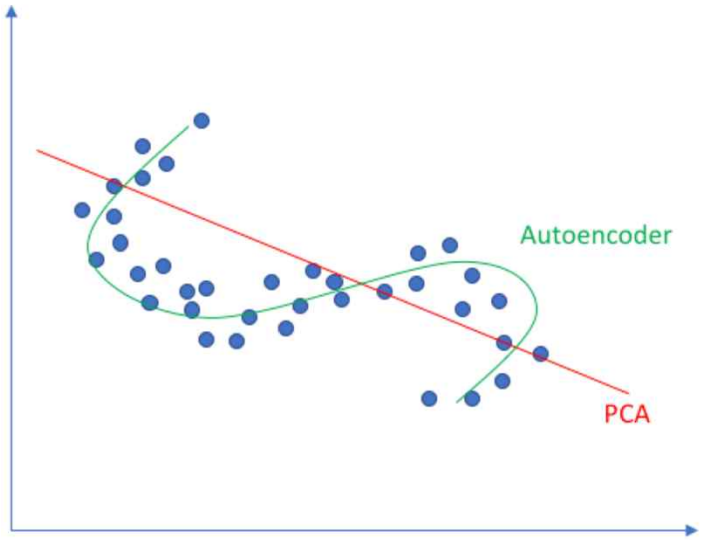
Deep Neural Network

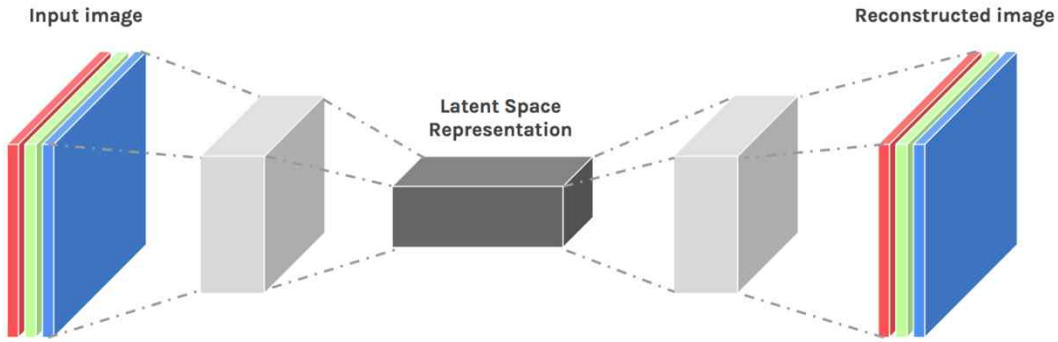
비지도 학습 방법

Autoencoder

Autoencoder

- 차원의 감소 : most relevant feature 추출
  - PCA : linear transformation
  - Autoencoder : non-linear transformation
- 적용 분야
  - 정보의 압축
  - Noise 제거
  - 유사한 image 검색
  - Image 변형에 의한 새로운 image 생성
  - Pre-Training





latent representation (잠재 표현) - intuition

83 12 21 42 99 18 51

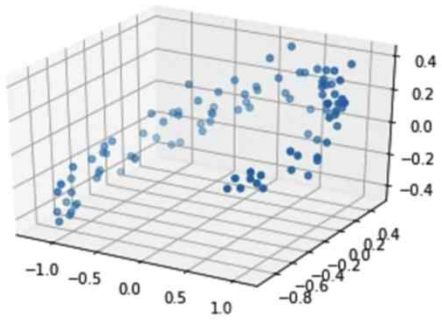
4 9 16 25 36 49 64 81 100 121 144 169

기억하기 쉬운 수열은 ?

- 첫번째 sequence 는 no pattern
- 두번째 sequence 는 제곱 pattern → latent pattern 존재

Autoencoder 의 기본 구조

input : 3 차원



encoder

latent representation 학습

decoder

X1

X2

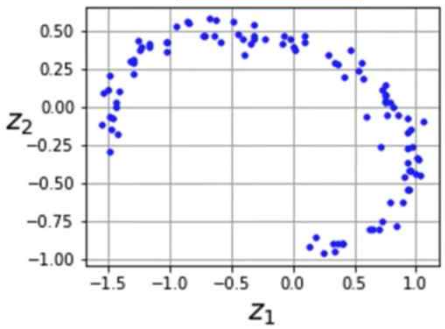
X3

2 차원

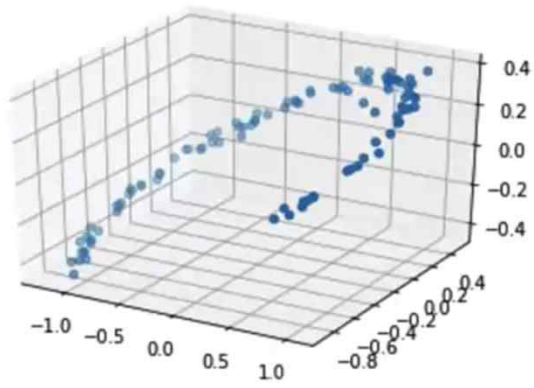
Y1

Y2

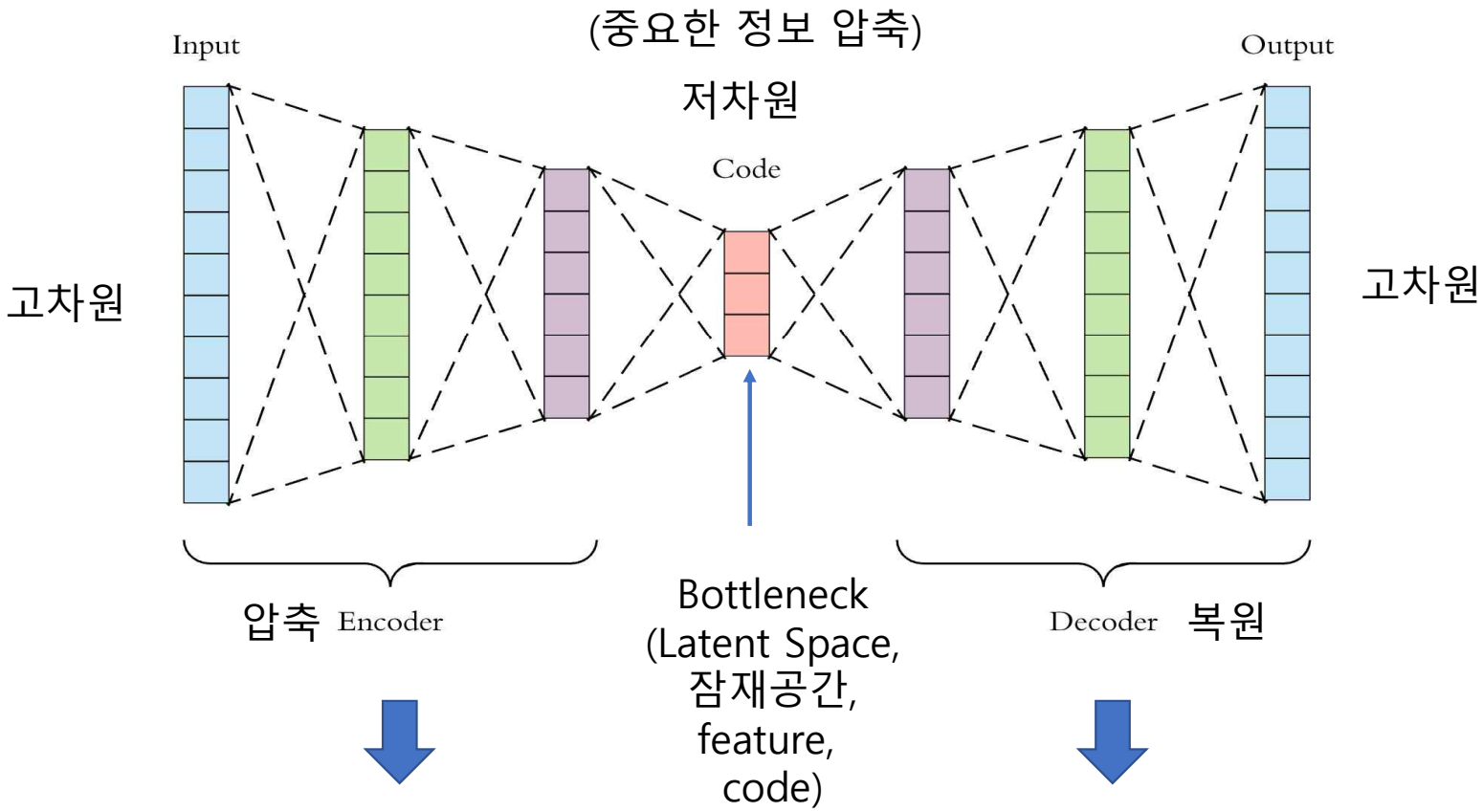
Y3



Output : 3 차원



noise 제거된 input

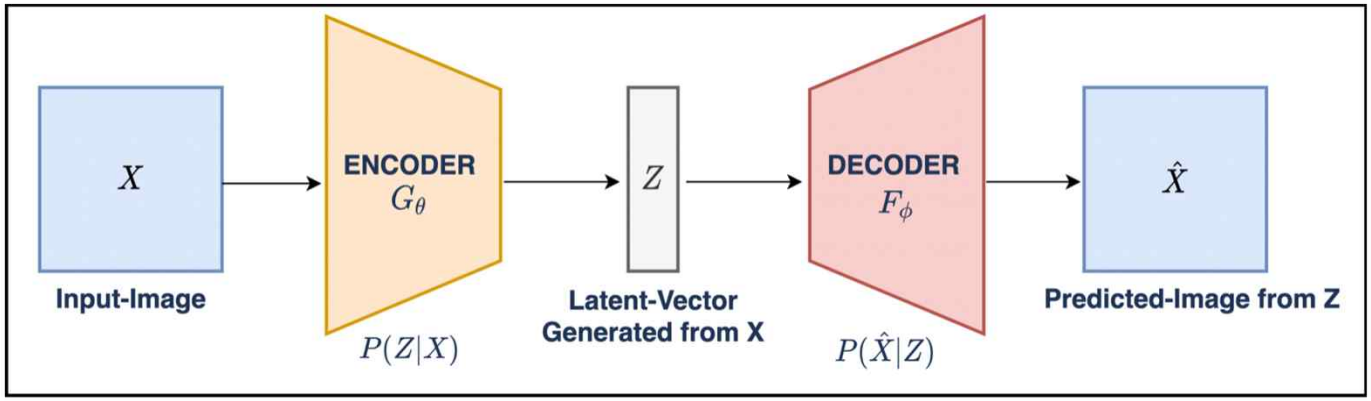


이미지와 같은 고차원 데이터를 Encoding 을 통해 저차원 Hidden 공간으로 표현

→ Latent Space (= Latent Variable)

Latent Variable 의 정보를 기반으로 **원래의 이미지를 복원**(Reconstruction) 하는 과정 수행

Autoencoder – 작동 방식



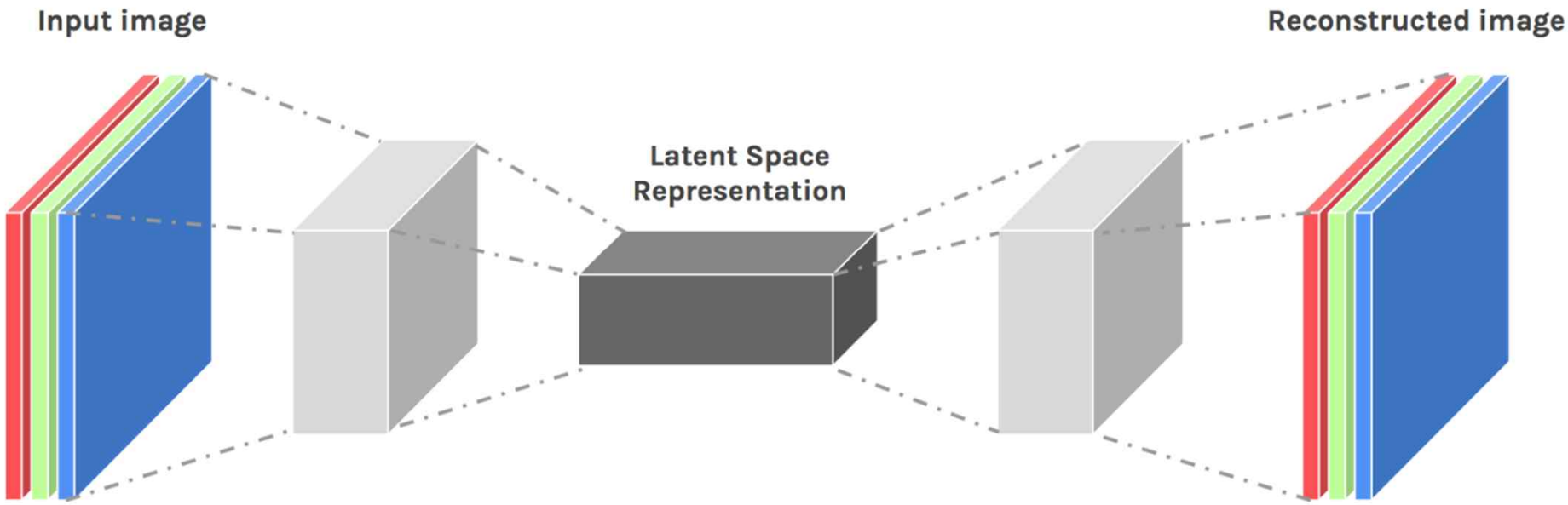
$$L(x, y) = \|X - \hat{X}\|^2$$

$$z = G(X)$$
$$\hat{X} = F(z) = F\big(G(X)\big)$$

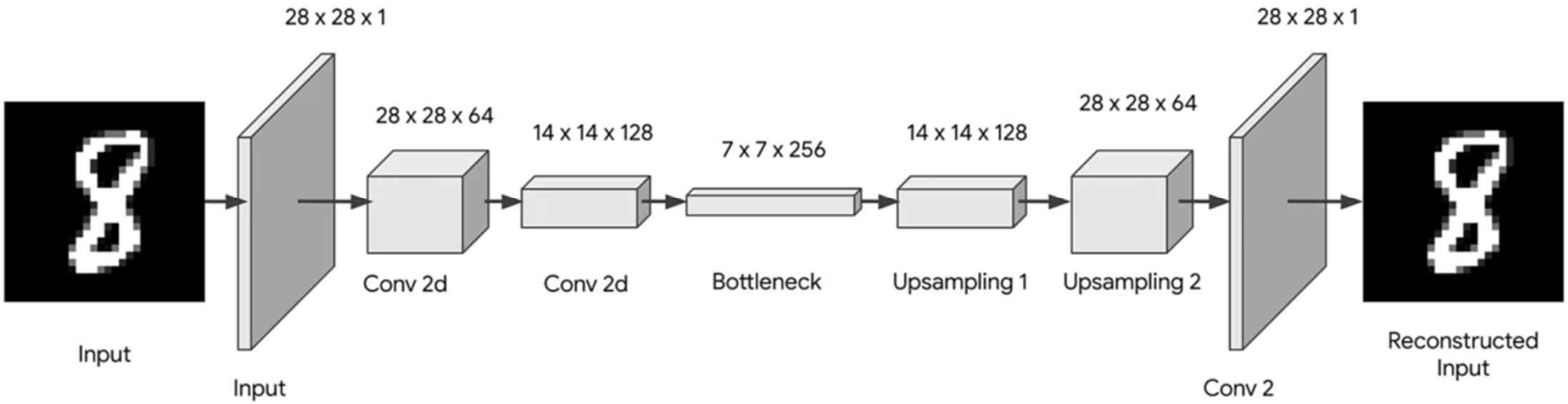
비지도학습 문제를 지도학습 문제로 바꾸어 해결
(자신을 Label 로 사용)

$L(x, y)$ - mse(입력이 정규분포) or cross-entropy(입력이 베르누이 분포)

Convolutional Auto-Encoders

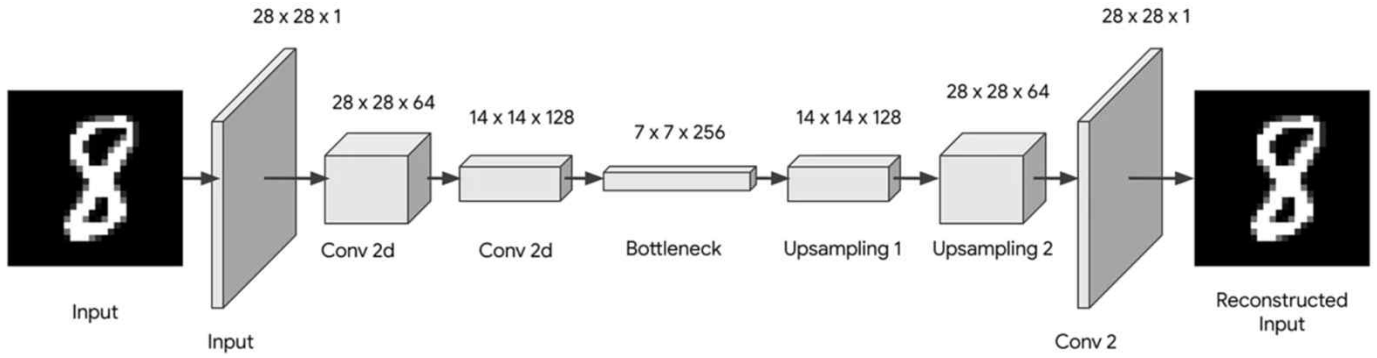


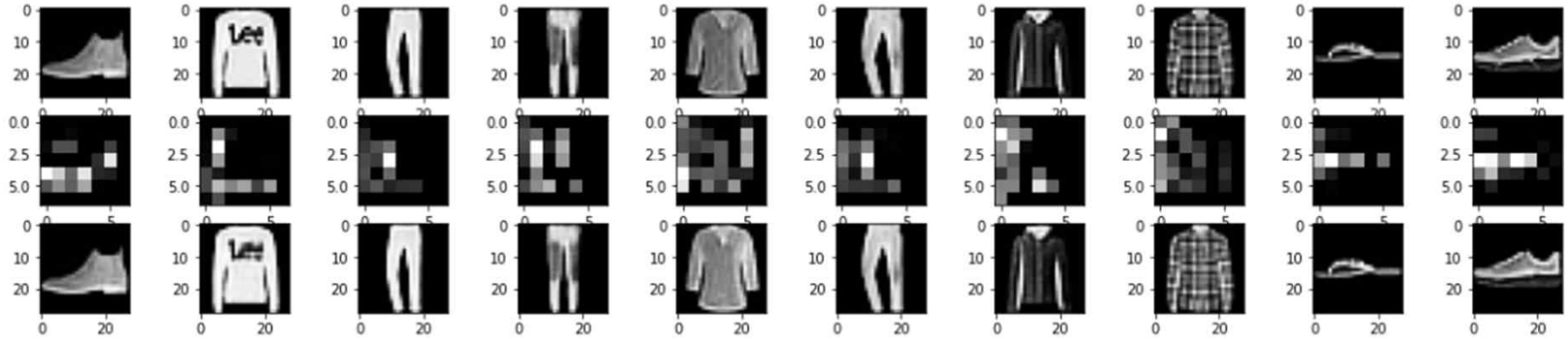
Convolutional Auto-Encoders



실습 : 300. Convolutional Autoencoder

- (Fashion) mnist dataset 을 encoding 후 decoding 하여 복원





Deep Learning

Sequence Model

What is sequence data ?

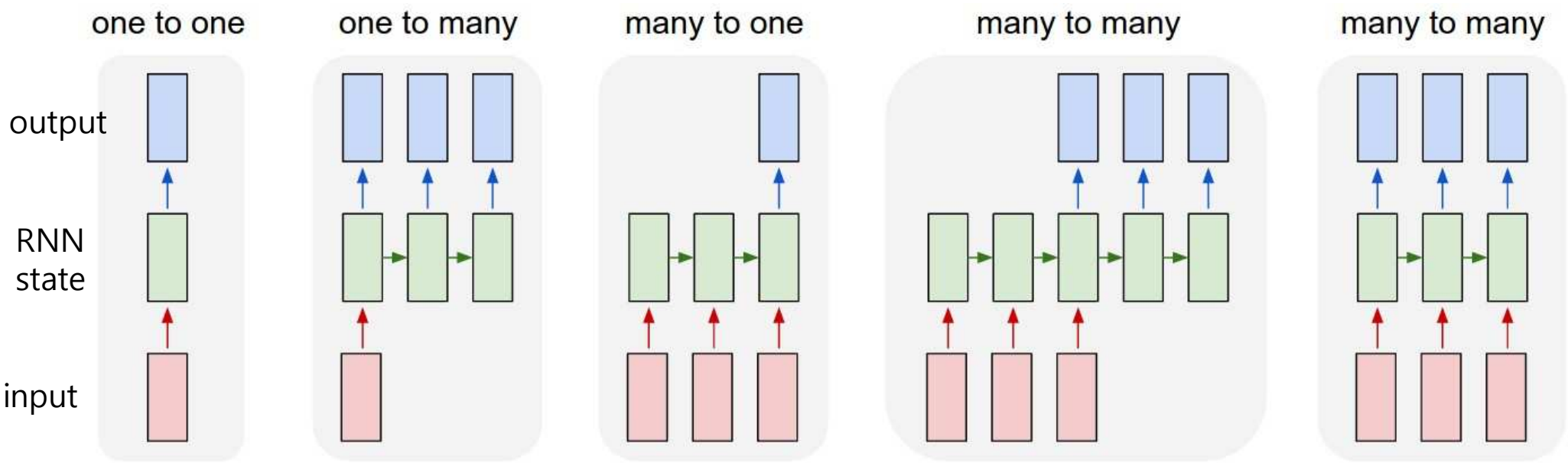
- Speech recognition : 파동의 연속 → 단어의 연속으로 변환
- Music generation : 연속된 음표 출력
- Sentiment classification : Text → 평점, 부정/긍정 판단
- DNA 분석 : 염기서열 → 질병유무, 단백질 종류 등
- 자동 번역 : 한국어 → 영어
- Video activity recognition : 연속된 장면 → 행동 판단
- Financial Data : 시계열자료 → 주가, 환율 예측 등

RNN (Recurrent Neural Network)

RNN (Recurrent Neural Network)

- 시퀀스 데이터에 특화
- '기억' 능력을 갖고 있음
  * 네트워크의 기억 - 지금까지의 입력 데이터를 요약한 정보
  (새로운 입력이 들어올 때 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

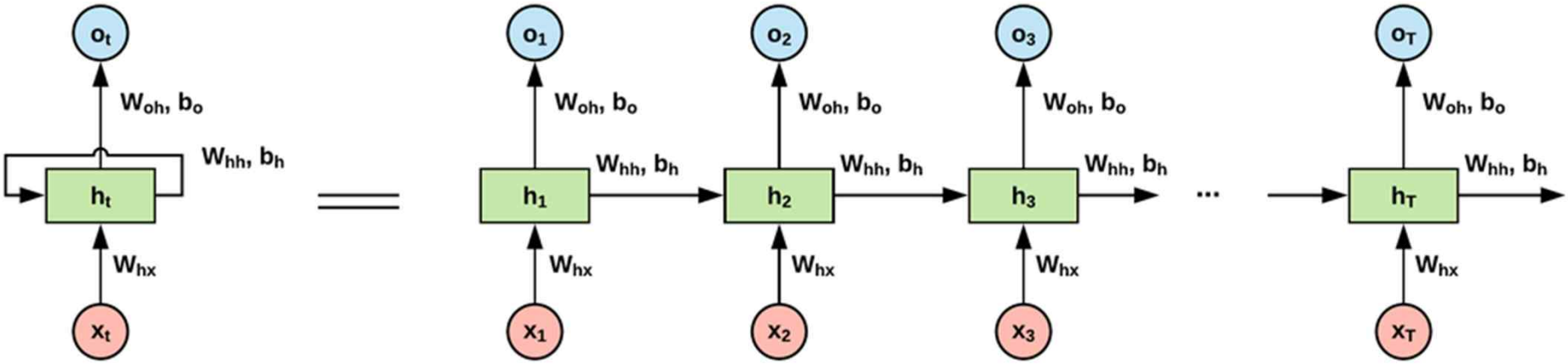
RNN (Unfold 표시)

Internal State :

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

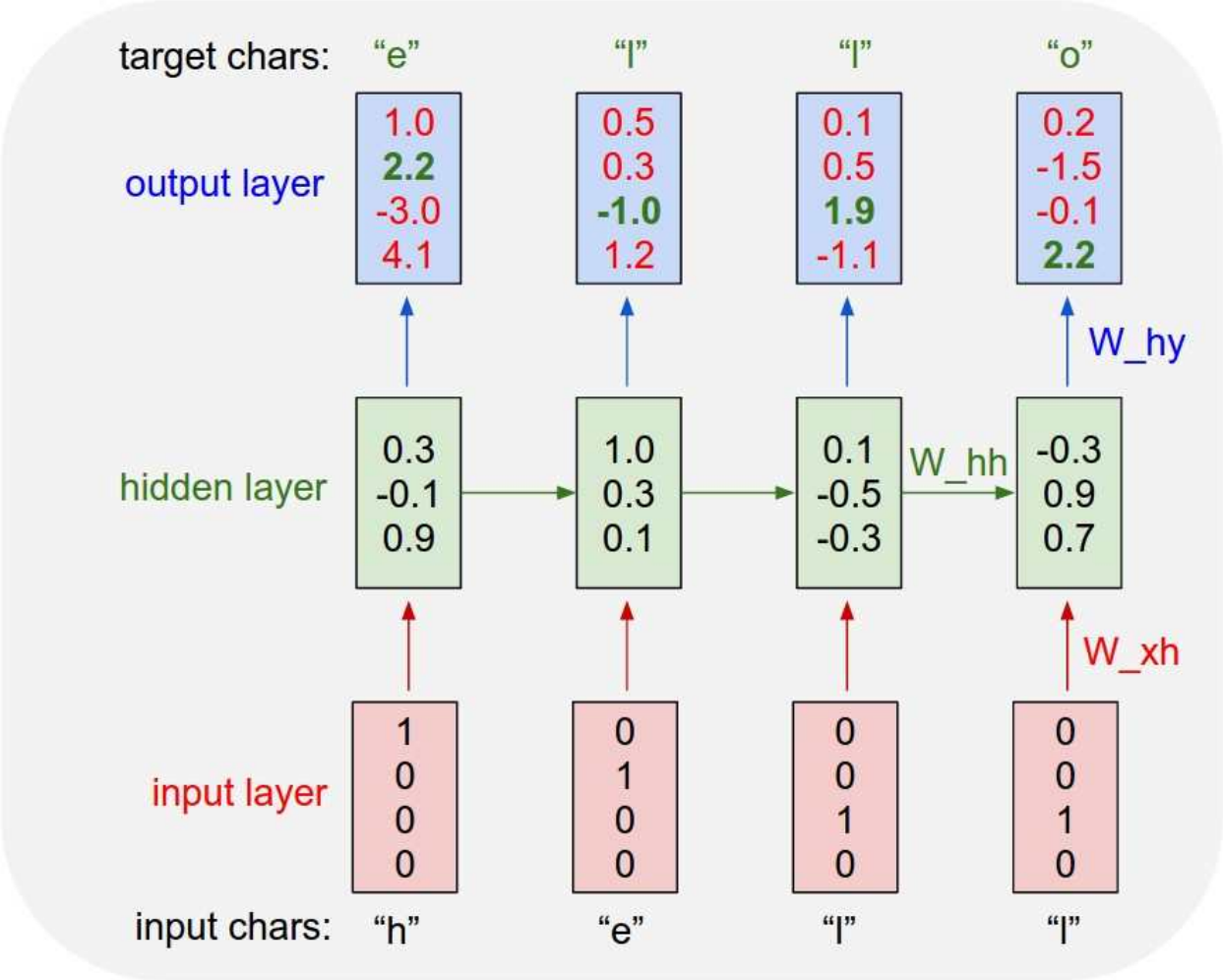
Output :

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



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

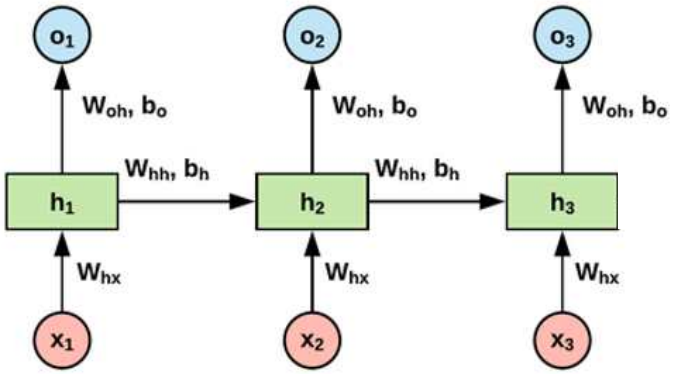
How RNN is trained ?



- "h", "e", "l", "o" 4 개 문자만 있다고 가정
- 각 문자를 One-hot encoding
- "h" 를 시작 문자로 주면 "hello" 가 출력되도록 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)

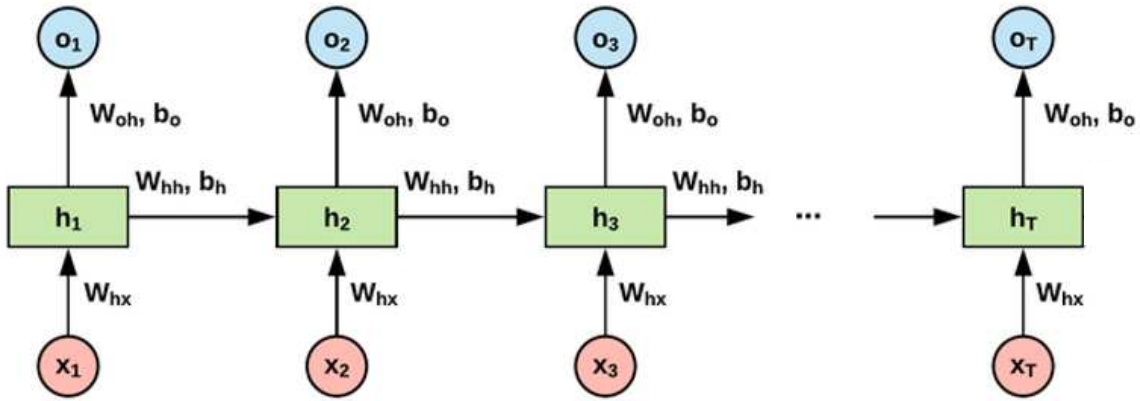
green



Tree color is

Simple RNN 으로 기억 가능

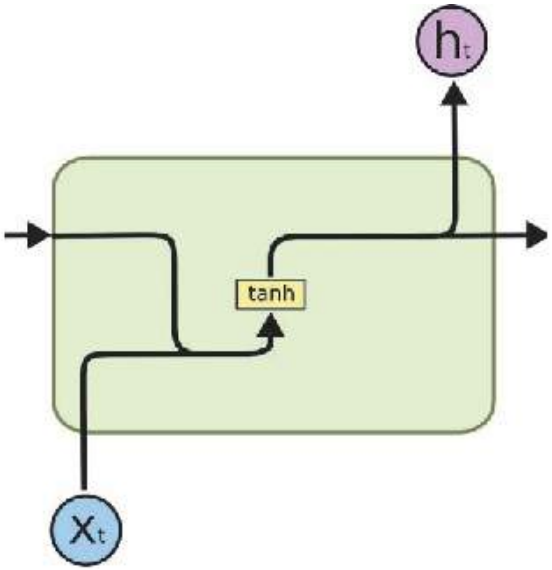
English



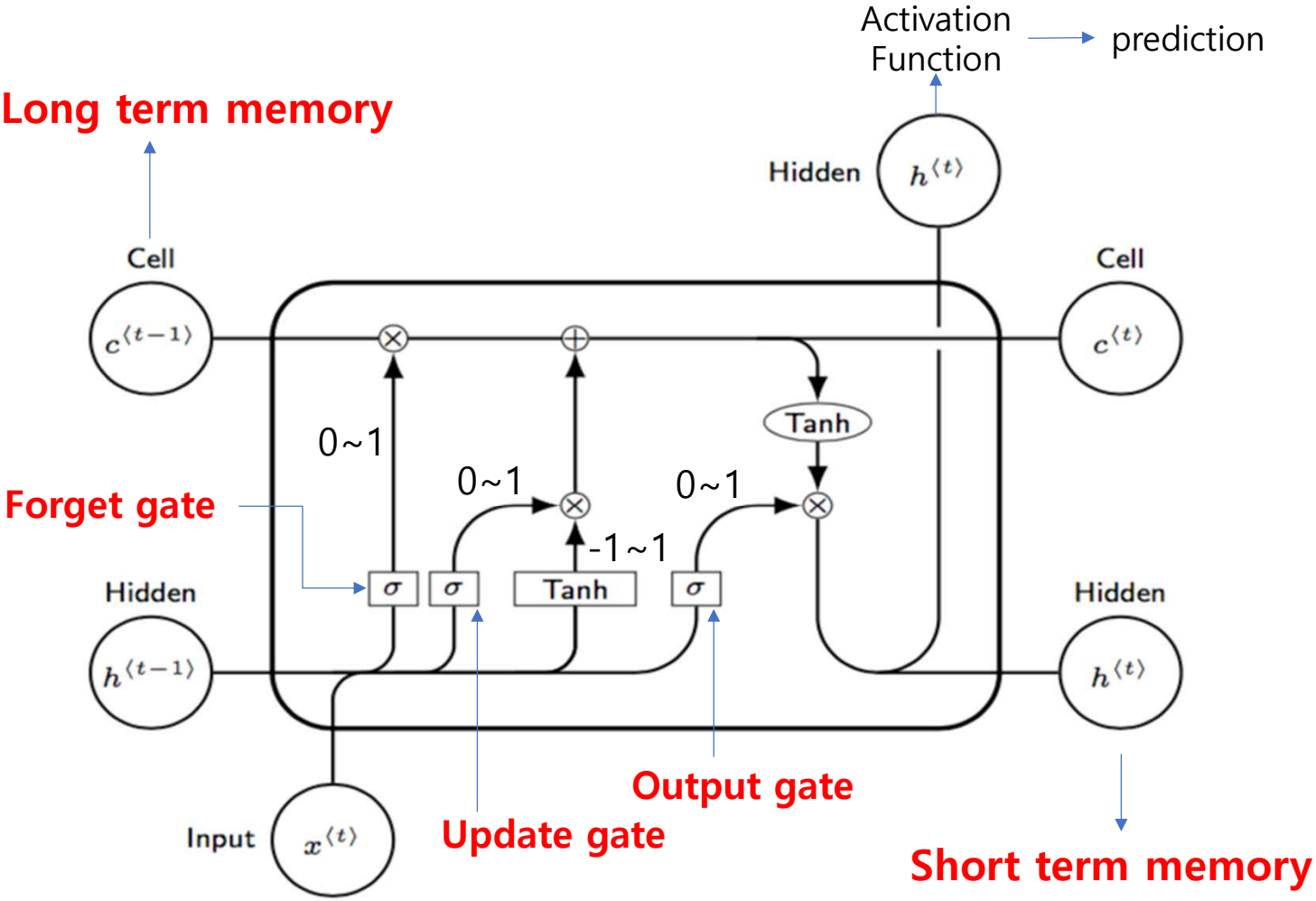
He was born in **England** and moved to Asia and his mother language is

Long-Term Memory 필요

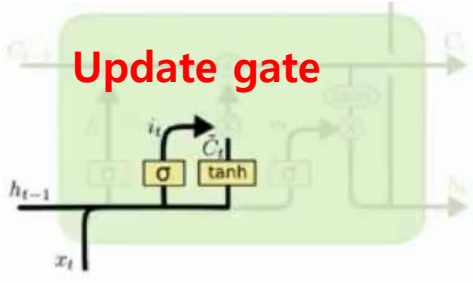
SimpleRNN



LSTM (Long Short-Term Memory)

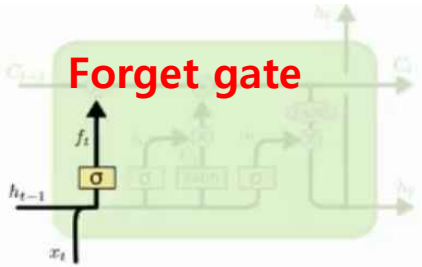


LSTM 내부 구조



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

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

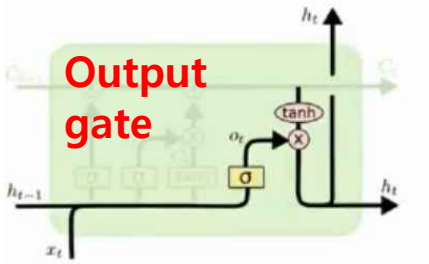


- **Update gate** – input 을 어느정도 받아들일지 결정 (0-무시, 1-전체)

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

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

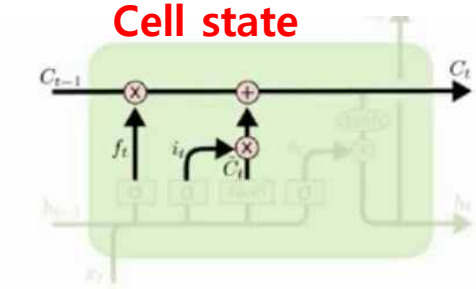
  $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$



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

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

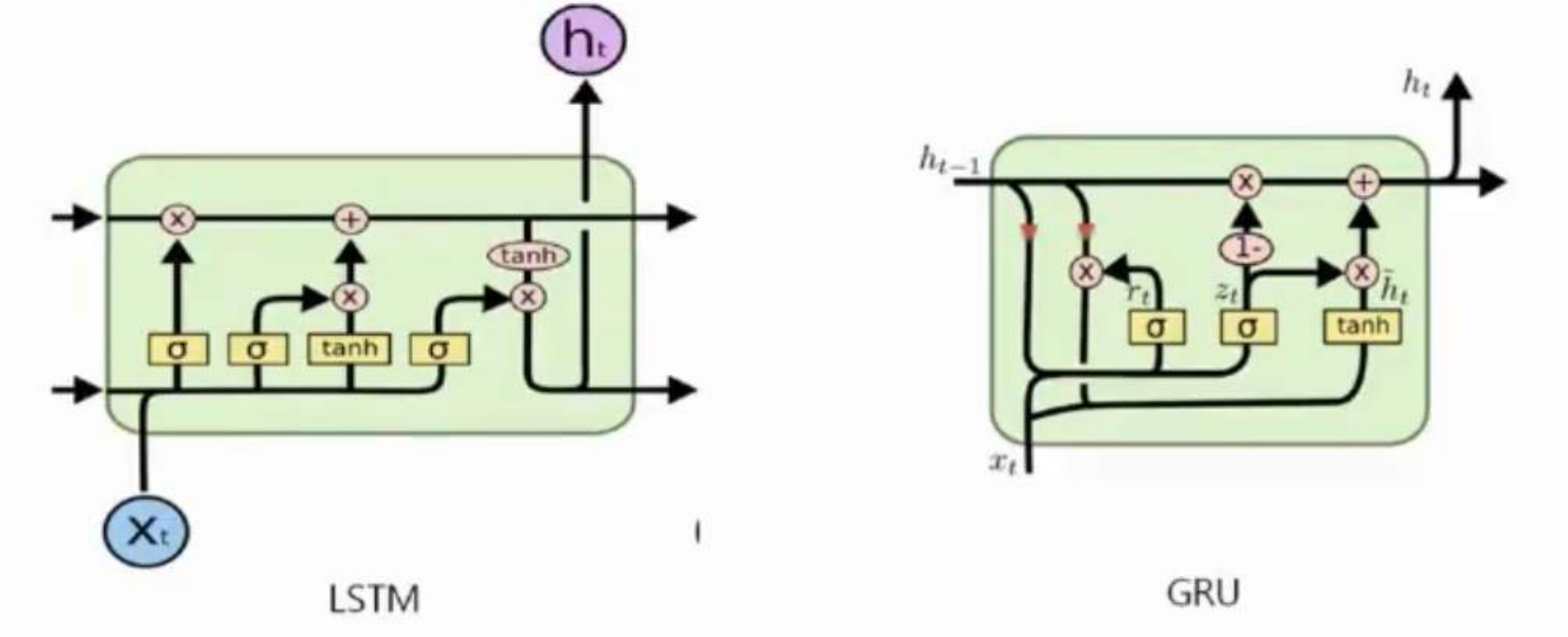
- $C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + \Gamma_f * C^{<t-1>}$



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

GRU (Gated Recurrent Unit)

- LSTM 의 장점을 유지하면서 일부 Gate 를 생략하여 계산의 복잡성을 낮춤



실습: 165. 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: 200. LSTM 을 이용한 주가 예측

- Apple 주식의 가격 추세 예측

- Yahoo finance data 이용