실습1) MLP

Nov. 10. 2024

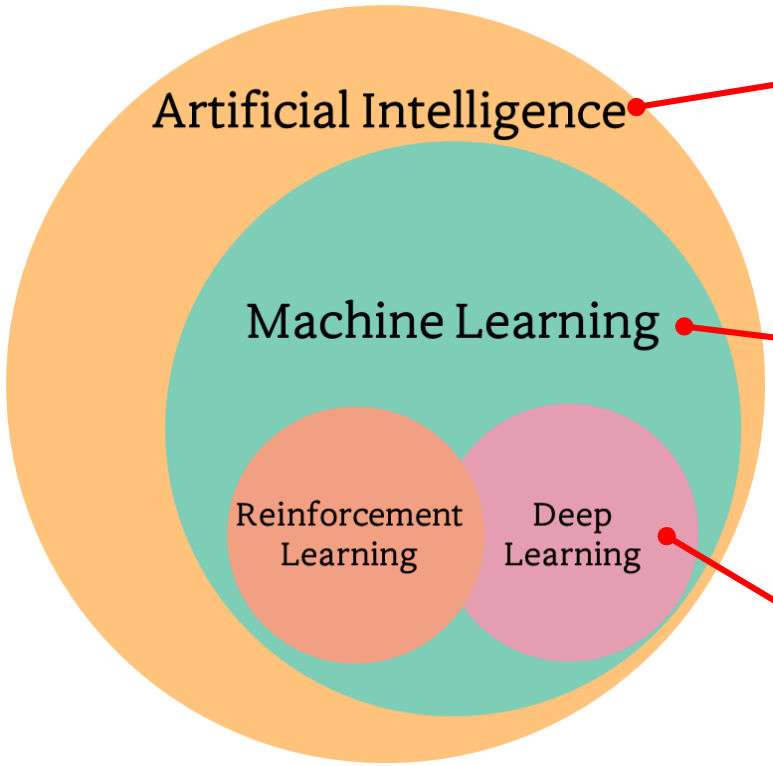
Spidercore

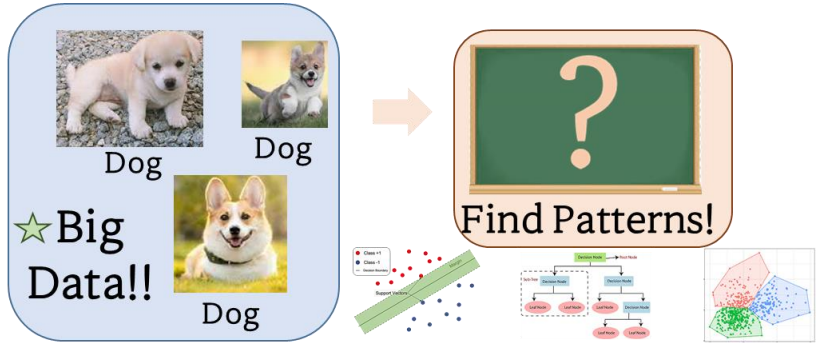
Contents

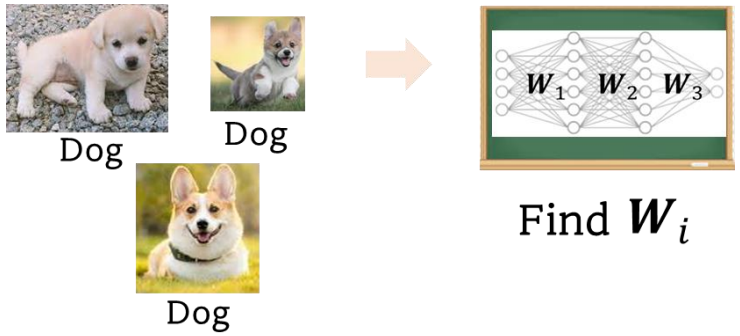
Contents

Recap.

- AI vs. ML vs. DL



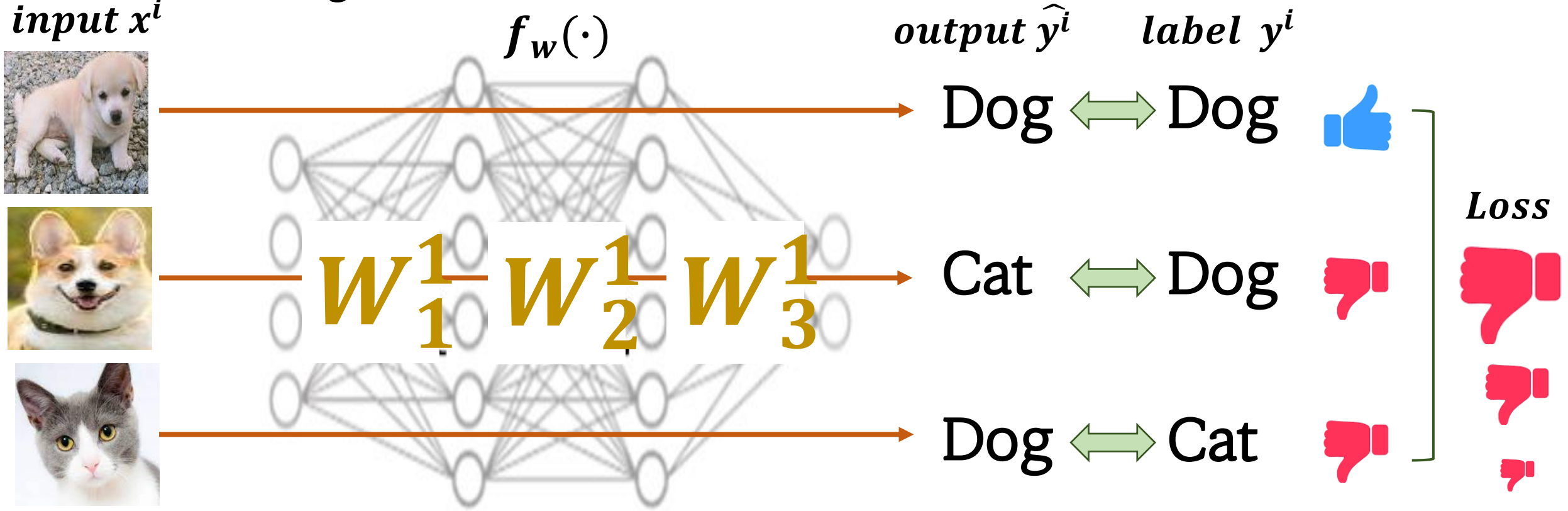




Dog!

Recap.

» Model Training



Repeat… Repeat… Repeat… until the loss goes to minimum…

Training : Determine W that minimizes loss btw. ground truth and prediction.

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, f_w(x^{(i)}))$$

$$= \hat{y}(x^{(i)})$$

Recap.

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, f_w(x^{(i)}))$$

Example : XOR Classification via Multilayer Perceptrons

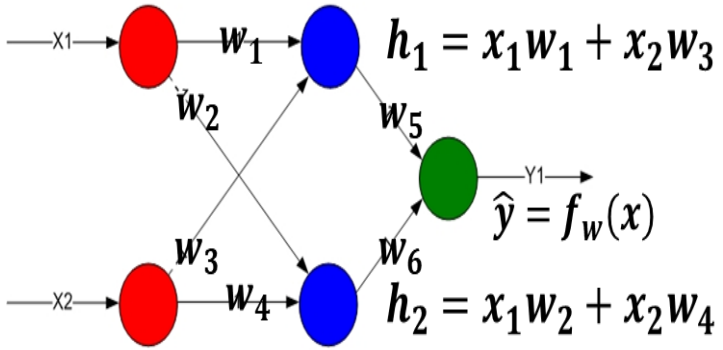
① Define $f_w(\cdot) \rightarrow$ 'Model' $\quad h_1 = x_1 w_1 + x_2 w_3, \qquad h_2 = x_1 w_2 + x_2 w_4$

② Define loss function $\mathcal{L}(y, \hat{y})$ $\quad \hat{y} = f_w(x) = h_1 w_5 + h_2 w_6$

③ Calculate $\hat{y}$ (prediction of each x) $\rightarrow$ 'Feed Forward'

④ Calculate loss of each sample and sum-up

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{4} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

$$= \sum_{i=1}^{4} \left(y^{(i)} - f_w(x^{(i)})\right)^2$$

x_1	x_2	$\hat{y} = f_w(x) = (x_1 w_1 + x_2 w_3) w_5 + (x_1 w_2 + x_2 w_4) w_6$	y
0	0	$(0w_1 + 0w_3)w_5 + (0w_2 + 0w_4)w_6$	0
0	1	$(0w_1 + 1w_3)w_5 + (0w_2 + 1w_4)w_6$	1
1	0	$(1w_1 + 0w_3)w_5 + (1w_2 + 0w_4)w_6$	1
1	1	$(1w_1 + 1w_3)w_5 + (1w_2 + 1w_4)w_6$	0

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{4} \left(y^{(i)} - f_w(x^{(i)})\right)^2 = (w_3 w_5 + w_4 w_6 - 1)^2 + (w_1 w_5 + w_2 w_6 - 1)^2 + (w_1 w_5 + w_3 w_5 + w_2 w_6 + w_4 w_6)^2$$

Recap.

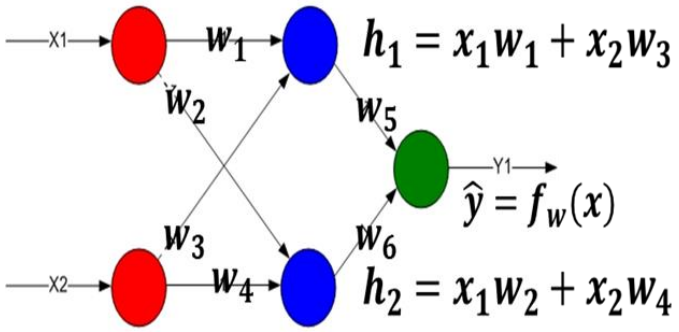
Example : XOR Classification via Multilayer Perceptrons

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, f_w(x^{(i)}))$$

⑤ Minimize loss s.t. w → 'Training'

$$\boldsymbol{W^k - \alpha \nabla_w \mathcal{L}(W^{k-1})}$$

$$\begin{bmatrix} w_1^k \\ \dots \\ w_6^k \end{bmatrix} = \begin{bmatrix} w_1^{k-1} - \alpha \left.\frac{\partial \mathcal{L}}{\partial w_1}\right|_{W=W^{k-1}} \\ \dots \\ w_6^{k-1} - \alpha \left.\frac{\partial \mathcal{L}}{\partial w_6}\right|_{W=W^{k-1}} \end{bmatrix}$$



$$\mathcal{L}(y,\hat{y}) = (w_3w_5 + w_4w_6 - 1)^2 + (w_1w_5 + w_2w_6 - 1)^2 + (w_1w_5 + w_3w_5 + w_2w_6 + w_4w_6)^2$$

Too complicated to calculate the partial derivatives…

$$\begin{cases} h_1 = x_1{}^{(i)} w_1 + x_2{}^{(i)} w_3 \\ \hat{y}^{(i)} = h_1 w_5 + h_2 w_6 \\ \mathcal{L}^{(i)}(y,\hat{y}) = \left(y^{(i)} - \hat{y}^{(i)}\right)^2 \\ \mathcal{L}(y,\hat{y}) = \sum \mathcal{L}^{(i)}(y,\hat{y}) \end{cases} \Rightarrow \frac{\partial \mathcal{L}}{\partial w_1} = \sum \frac{\partial \mathcal{L}^{(i)}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Chain Rule !!!

Recap.

Example : XOR Classification via Multilayer Perceptrons

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, f_w(x^{(i)}))$$

⑤ Minimize loss s.t. w → 'Training'

Update…

$$W^k - \alpha \nabla_W \mathcal{L}(W^{k-1})$$

$$\begin{bmatrix} w_1^k \\ \dots \\ w_6^k \end{bmatrix} = \begin{bmatrix} w_1^{k-1} - \alpha \left.\frac{\partial \mathcal{L}}{\partial w_1}\right|_{W=W^{k-1}} \\ \dots \\ w_6^{k-1} - \alpha \left.\frac{\partial \mathcal{L}}{\partial w_6}\right|_{W=W^{k-1}} \end{bmatrix}$$

$$w_1^0 = 0.1, \dots, w_6^0 = 0.1, \alpha = 0.003$$

$$w_1^1 = w_1^0 - \alpha \left.\frac{\partial \mathcal{L}}{\partial w_1}\right|_{W=W^0}$$

$$\left.\frac{\partial \mathcal{L}}{\partial w_1}\right|_{W=W^{k-1}} = 4w_1 w_5^2 + 4w_2 w_5 w_6 + 4w_3 w_5^2 + 4w_4 w_5 w_6 - 2w_5 \,|_{W=W^{k-1}}$$

$$= 0.1 - 0.003 * (4 * 0.1^3 + 4 * 0.1^3 + 4 * 0.1^3 + 4 * 0.1^3 - 2 * 0.1)$$

$$= 0.100564$$

$w_1^0 = 0.1 \rightarrow 0.100564$ ***updated***!!

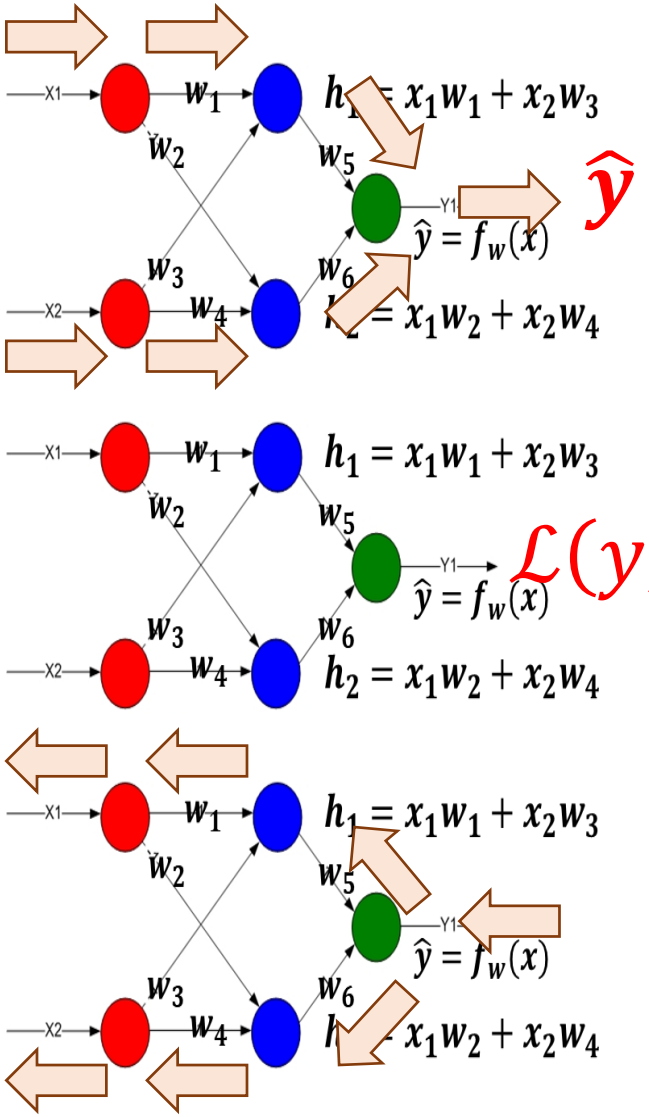
$$W^k \rightarrow W^k - \alpha \nabla_W \mathcal{L}(W^{k-1})$$

Repeat… Repeat… Repeat… until the loss goes to minimum…

Recap.

Example : XOR Classification via Multilayer Perceptrons

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, f_w(x^{(i)}))$$



③ Calculate $\hat{y}$ (prediction of each x) → 'Feed Forward'

x_1	x_2	$\hat{y} = f_w(x)$ $= (x_1w_1 + x_2w_3)w_5 + (x_1w_2 + x_2w_4)w_6$	y
0	0	$(0w_1 + 0w_3)w_5 + (0w_2 + 0w_4)w_6$	0
0	1	$(0w_1 + 1w_3)w_5 + (0w_2 + 1w_4)w_6$	1
1	0	$(1w_1 + 0w_3)w_5 + (1w_2 + 0w_4)w_6$	1
1	1	$(1w_1 + 1w_3)w_5 + (1w_2 + 1w_4)w_6$	0

④ Calculate loss of each sample and sum-up

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{4} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_{i=1}^{4} \left(y^{(i)} - f_w(x^{(i)})\right)^2$$

⑤ Minimize loss s.t. w → 'Training'

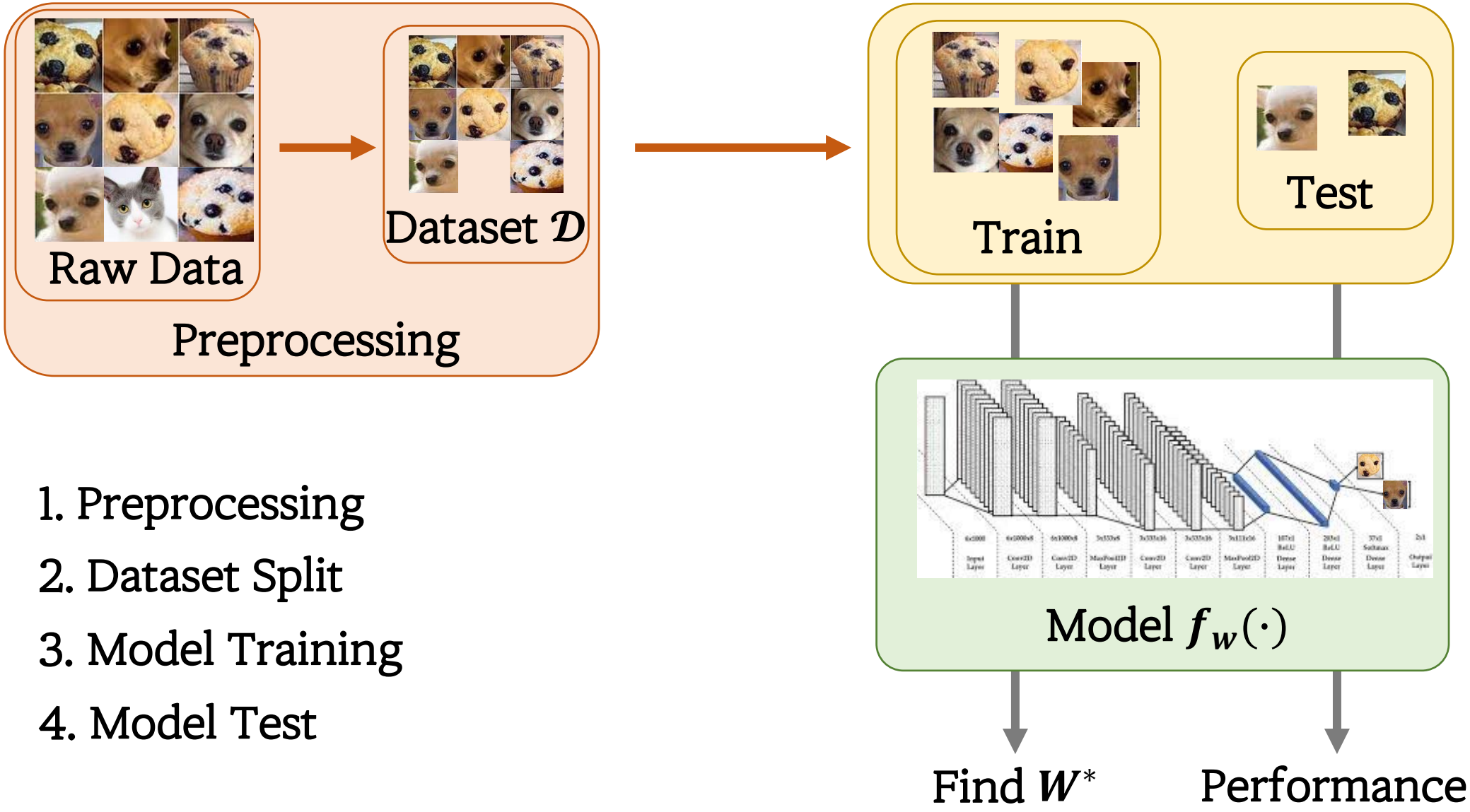
$$\frac{\partial \mathcal{L}}{\partial w_1} = \sum \frac{\partial \mathcal{L}^{(i)}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$

Back-propagation via chain-rule

Update

$$W^k \rightarrow W^k - \alpha \nabla_w \mathcal{L}(W^{k-1})$$

Deep Learning - Flow



1. Preprocessing
2. Dataset Split
3. Model Training
4. Model Test

Recap.

Q&A – Partial Derivative

→ Squared Error는 미분하면 $2(y^{(i)} - \hat{y}^{(i)})$ 가 되므로 숫자 2가 계속 따라다니기 때문에 계산이 번거로움

→ 그래서 수식 전개의 편리성을 위해 Loss function에 $\frac{1}{2}$을 곱해준 형태로 정의함

→ L(x)를 최소화 하든 $\frac{1}{2}$L(x)를 최소화하든 결과는 동일

Model Training

Example : Predicting the weight loss effects of Wegovy

$$\min_{w} \sum_{i=1}^{m} \ell(y^{(i)}, f_w(x^{(i)}))$$

⑤ Minimize loss s.t. w → 'Training'

$$\mathcal{L}(y, \hat{y}) = 4w_0^2 + 30\, w_0 w_1 - 26.2\, w_0 + 79 w_1^2 - 129.8 w_1 + 54.83$$

Let $\boldsymbol{W^k} = \begin{bmatrix} w_0^k \\ w_1^k \end{bmatrix}$ denote the updated parameter at the k-th iteration (2D case)

Then the gradient of $\boldsymbol{W^k}$, $\nabla_w \mathcal{L}(\boldsymbol{W}^{k-1}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0}\Big|_{W=\boldsymbol{W}^{k-1}} \\ \frac{\partial \mathcal{L}}{\partial w_1}\Big|_{W=\boldsymbol{W}^{k-1}} \end{bmatrix} = \begin{bmatrix} 4w_0^{k-1} + 15w_1^{k-1} - 13.1 \\ 15w_0^{k-1} + 79w_1^{k-1} - 64.9 \end{bmatrix}$

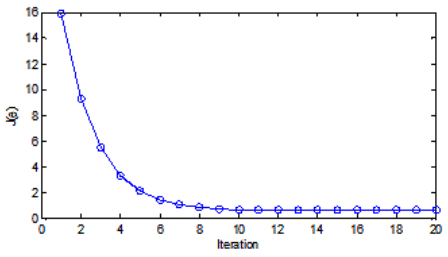
The update equation is given by

$$\begin{bmatrix} \boldsymbol{w_0^k} \\ \boldsymbol{w_1^k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w_0^{k-1}} - \alpha \frac{\partial \mathcal{L}}{\partial w_0}\Big|_{W=\boldsymbol{W}^{k-1}} \\ \boldsymbol{w_1^{k-1}} - \alpha \frac{\partial \mathcal{L}}{\partial w_1}\Big|_{W=\boldsymbol{W}^{k-1}} \end{bmatrix} = \begin{bmatrix} w_0^{k-1} - \alpha(4w_0^{k-1} + 15w_1^{k-1} - 13.1) \\ w_1^{k-1} - \alpha(15w_0^{k-1} + 79w_1^{k-1} - 64.9) \end{bmatrix} =: \boldsymbol{W^k} - \alpha \nabla_w \mathcal{L}(\boldsymbol{W}^{k-1})$$

$\boldsymbol{w_0^0 = 0, w_1^0 = 0, \alpha = 0.003}$

Update...

w_0	w_1
0.0000	0.0000
0.0393	0.1947
0.0692	0.3427
0.0921	0.4551
0.1095	0.5405
.....	
0.6752	0.6928
0.6754	0.6929



Solution:
$(w_0, w_1) = (0.6747, 0.6934)$

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{4} (y^{(i)} - \hat{y}^{(i)})^2$$

$$\mathcal{L}(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^{4} (y^{(i)} - \hat{y}^{(i)})^2$$

Recap.

Q&A – Anomaly Detection

https://github.com/xuhongzuo/DeepOD

Tabular Anomaly Detection models:

Model	Venue	Year	Type	Title
Deep SVDD	ICML	2018	unsupervised	Deep One-Class Classification [1]
REPEN	KDD	2018	unsupervised	Learning Representations of Ultrahigh-dimensional Data for Random Distance-based Outlier Detection [2]
RDP	IJCAI	2020	unsupervised	Unsupervised Representation Learning by Predicting Random Distances [3]
RCA	IJCAI	2021	unsupervised	RCA: A Deep Collaborative Autoencoder Approach for Anomaly Detection [4]
GOAD	ICLR	2020	unsupervised	Classification-Based Anomaly Detection for General Data [5]
NeuTraL	ICML	2021	unsupervised	Neural Transformation Learning for Deep Anomaly Detection Beyond Images [6]
ICL	ICLR	2022	unsupervised	Anomaly Detection for Tabular Data with Internal Contrastive Learning [7]
DIF	TKDE	2023	unsupervised	Deep Isolation Forest for Anomaly Detection [19]
SLAD	ICML	2023	unsupervised	Fascinating Supervisory Signals and Where to Find Them: Deep Anomaly Detection with Scale Learning [20]
DevNet	KDD	2019	weakly-supervised	Deep Anomaly Detection with Deviation Networks [8]
PReNet	KDD	2023	weakly-supervised	Deep Weakly-supervised Anomaly Detection [9]
Deep SAD	ICLR	2020	weakly-supervised	Deep Semi-Supervised Anomaly Detection [10]
FeaWAD	TNNLS	2021	weakly-supervised	Feature Encoding with AutoEncoders for Weakly-supervised Anomaly Detection [11]
RoSAS	IP&M	2023	weakly-supervised	RoSAS: Deep semi-supervised anomaly detection with contamination-resilient continuous supervision [21]

Tier : ICML,ICLR > KDD, IJCAI > Others

```python
# unsupervised methods
from deepod.models.tabular import DeepSVDD
clf = DeepSVDD()
clf.fit(X_train, y=None)
scores = clf.decision_function(X_test)

# weakly-supervised methods
from deepod.models.tabular import DevNet
clf = DevNet()
clf.fit(X_train, y=semi_y) # semi_y uses 1 for known anomalies, and 0 for
scores = clf.decision_function(X_test)

# evaluation of tabular anomaly detection
from deepod.metrics import tabular_metrics
auc, ap, f1 = tabular_metrics(y_test, scores)
```

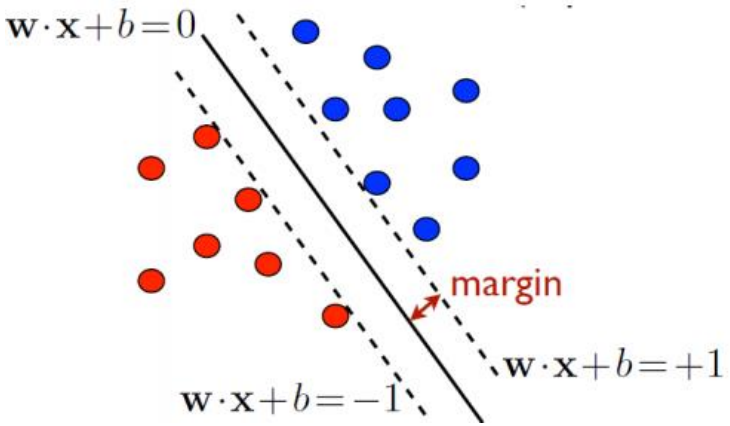
Easy implementation through built-in functions.

Recap.

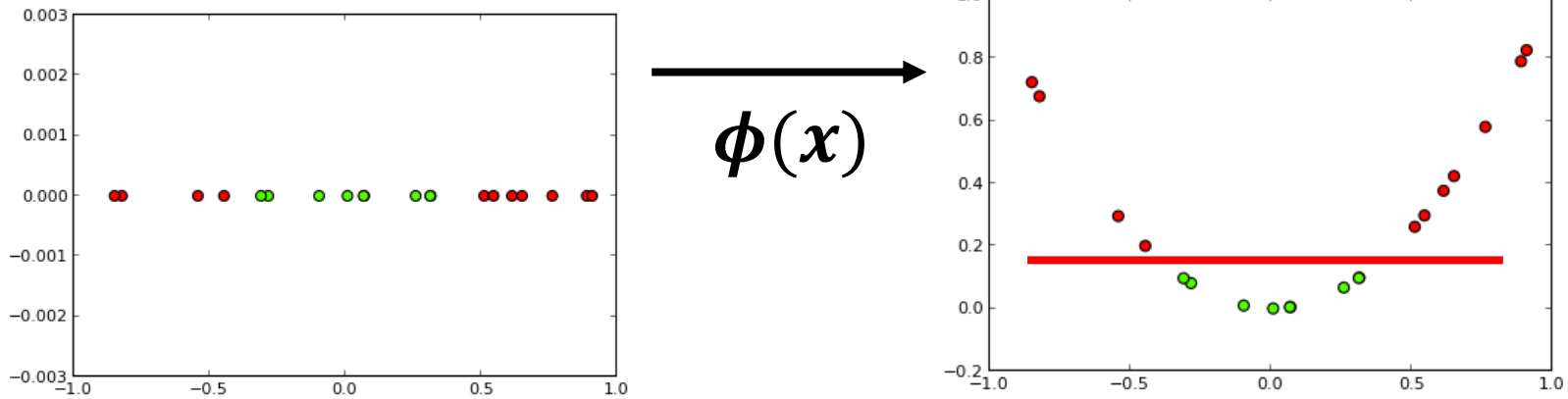
Q&A – Anomaly Detection

https://github.com/xuhongzuo/DeepOD

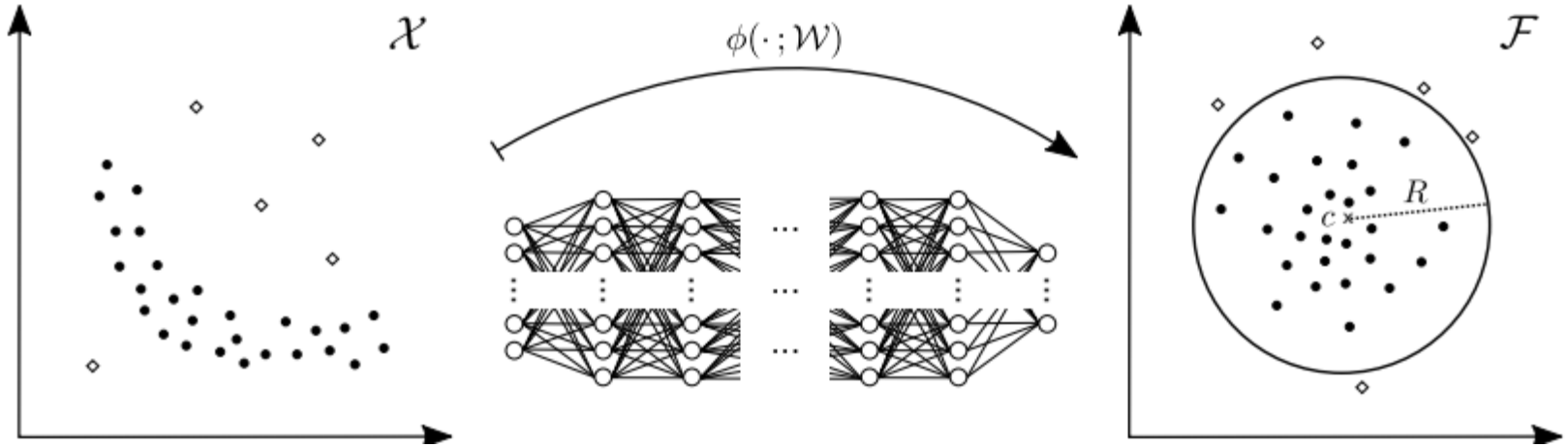
Example : Deep One-Class Classification (Deep SVDD , ICML2018)



< SVM >



< SVM w/ Kernel Trick >



< SVDD >

Recap.

Q&A – 교재

(★~★★) O'REILLY Series

- 개발자를 위한 필수수학 (https://product.kyobobook.co.kr/detail/S000213417669)
- 개발자를 위한 실전 선형대수학 (https://product.kyobobook.co.kr/detail/S000209345747)
- 핸즈온 머신러닝 등등

(★~★★) 딥러닝을 위한 수학

- 엄청 쉬워보이긴 함, python 코드도 있긴 함 (https://jpub.tistory.com/1304)

(★★~★★★) Mathmatics for Machine Learning (MML)

- 필요한 내용만 Depth있게 다루는 것 같음, Reddit Recommend 600+
- PDF file Download : http://mml-book.github.io/book/mml-book.pdf

(★★★★) An Introduction to Statistical Learning (with Applications in Python)

- PDF file Download : https://www.statlearning.com/

(★★★★★★★★★) 딥러닝을 위한 선형대수학

- 선형대수의 대가 Gilbert Strang 作, 선형대수 완강한 사람들 추천
- 직강 유툽 있음 (https://www.youtube.com/playlist?list=PLUl4u3cNGP63oMNUHXqIUcrkS2PivhN3k)

Contents

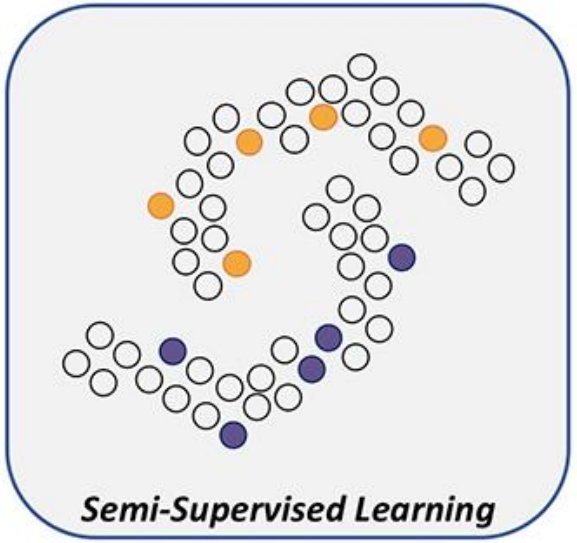
Deep Learning Flow

Learning Methods

모든 데이터의 label이 존재
Task :
- Classification
- Regression

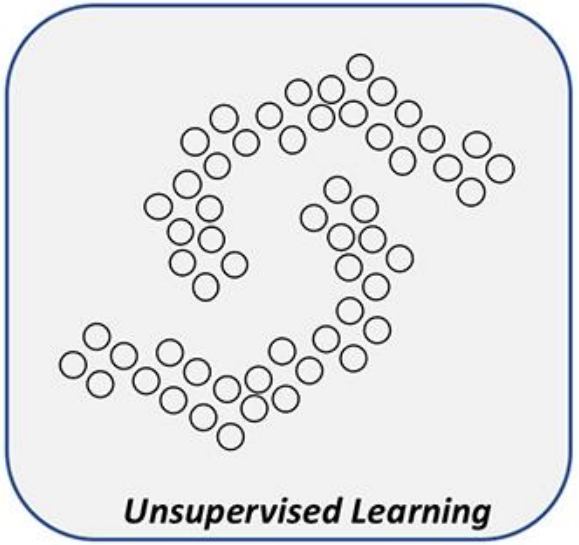
$$Given = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$$

$$Goal = Learn\ a\ rule\ (f : x \rightarrow y)$$



데이터 일부의 label이 존재
Task :
- Classification
- Regression

$$Loss = L_s + L_u$$



모든 데이터의 label이 없음
Task :
- Clustering
- Dimensionality Reduction

$$Given = \{(x^{(1)}), \ldots, (x^{(N)}\}$$

$$\boldsymbol{Goal = Discover\ hidden\ patterns\ w/o\ instruction}$$

Deep Learning Flow

Overview

1. Preprocessing (Exploratory Data Analysis)

- Remove dummy data and variable
- Transformation into numeric type
- Statistical analysis
- Missing values

2. Dataset Split

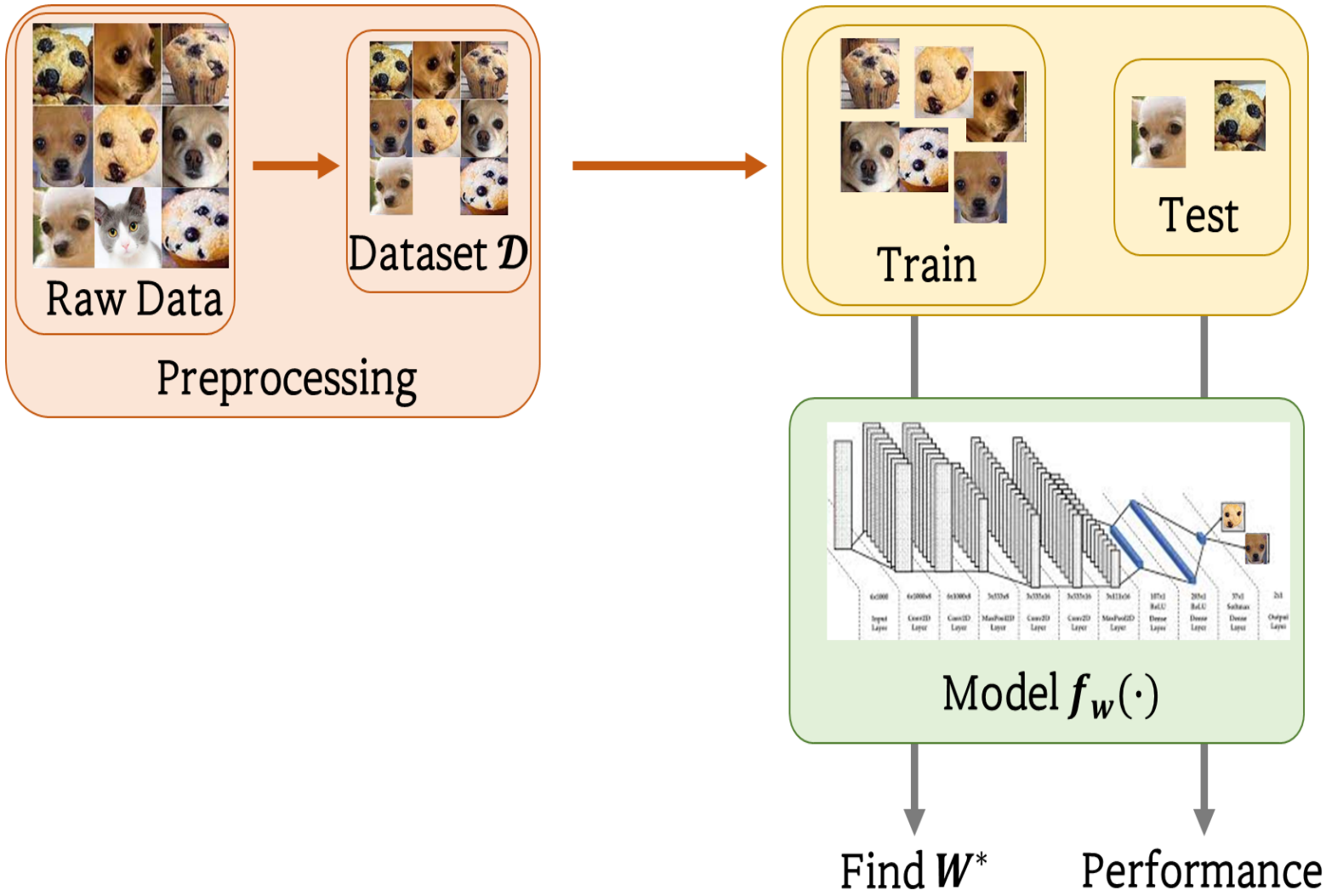
- Train vs. Validation vs. Test
- K-fold cross validation

3. Model Training

- Epoch, Batch size
- Optimizer, Learning Rate
- Initializer
- Activaition Function
- Over-fitting, Under-fitting

4. Model Test

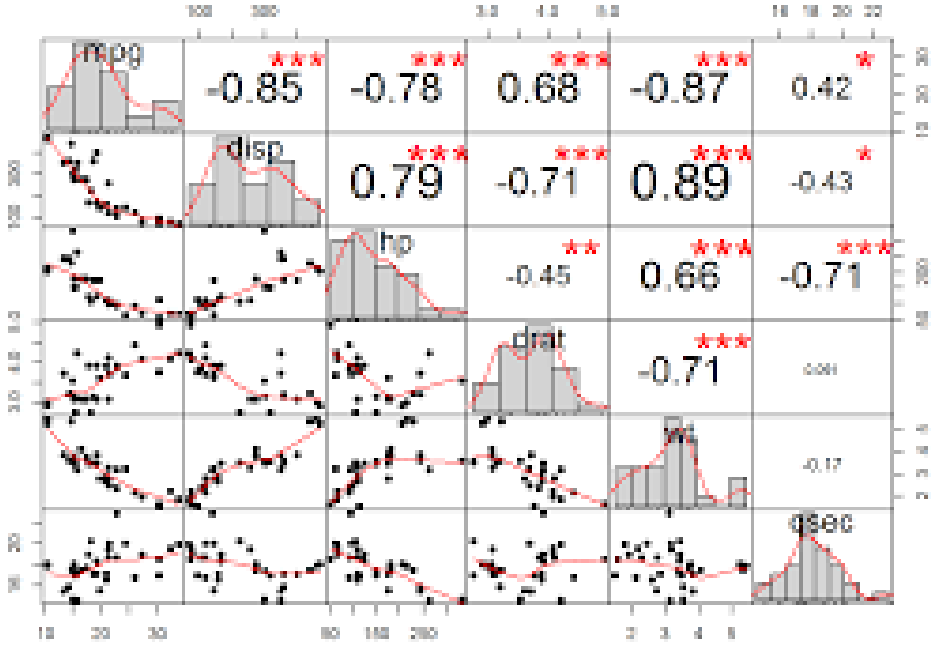
- Performance Measure

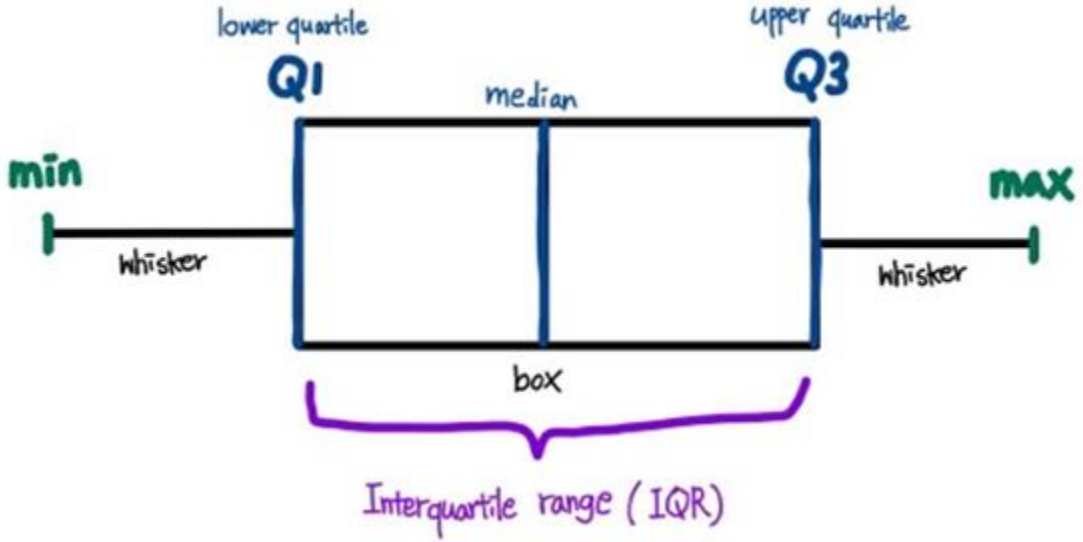


Deep Learning Flow – 1. Preprocessing

EDA (Exploratory Data Analysis)

- **벨연구소의 수학자 '존 튜키'가 개발한 데이터분석 과정에 대한 개념**
  - 데이터를 분석하고 결과를 내는 과정에 있어서 지속적으로 '탐색과 이해'를 진행해야 함
- **데이터를 분석하기 전에 그래프나 통계적인 방법으로 자료를 직관적으로 바라보는 과정**
- https://dacon.io/codeshare/4899





Deep Learning Flow – 1. Preprocessing

Remove Dummy Data and Variables

- 데이터 일부를 랜덤으로 샘플링하여 육안으로 확인하면서 Cleaning 작업 진행
- 불필요한 변수가 있는 경우 미리 제거 (∵ 노이즈로 작용)

'아파트 가격 ~ 평수 + 화장실 개수 + 층 + 부동산업자'

	Pas erId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	T	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

Deep Learning Flow – 1. Preprocessing

Transformation into numeric type

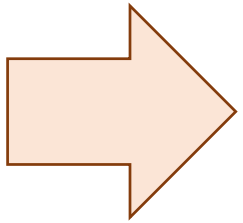
모든 입출력 값은 숫자형 데이터만 가능

- 문자형 데이터의 경우 dtype = Object / 숫자형 데이터의 경우 dtype = int, float, double

문자형 데이터의 경우, Encoding 필요 (one-hot or Label)

- 숫자형 데이터더라도 Categorical variable인 경우 encoding 필요
- One-hot encoding → Dummy Variable Trap (multicollinearity) issue
- Quiz) 각각 언제 사용해야 하나?

Color
Red
Green
Blue
Red
Green
Red
Blue



Color
2
1
0
2
1
2
0

< Label Encoding >

Dummy_r	Dummy_b	Dummy_g
1	0	0
0	0	1
0	1	0
1	0	0
0	0	1
1	0	0
0	1	0

< One-hot Encoding >

Deep Learning Flow – 1. Preprocessing

Transformation into numeric type

Ex.) Titanic Survival Prediction

- One-hot or label? Survived, Pclass, Sex, Cabin, Embarked

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

Surv_0	Surv_1
1	0
0	1
1	0
0	1
1	0

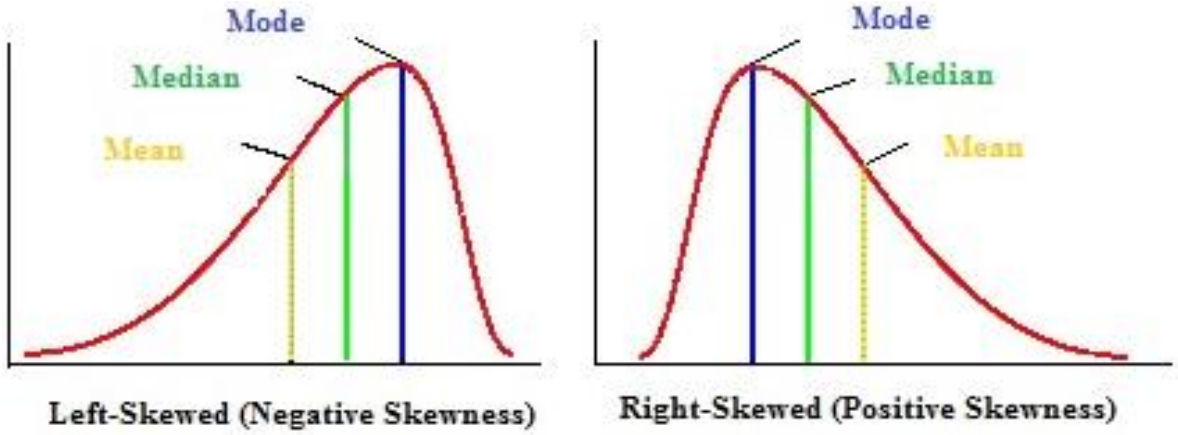
(Embarked) S : Southampton, Q : Queenstown, C : Cherbourge

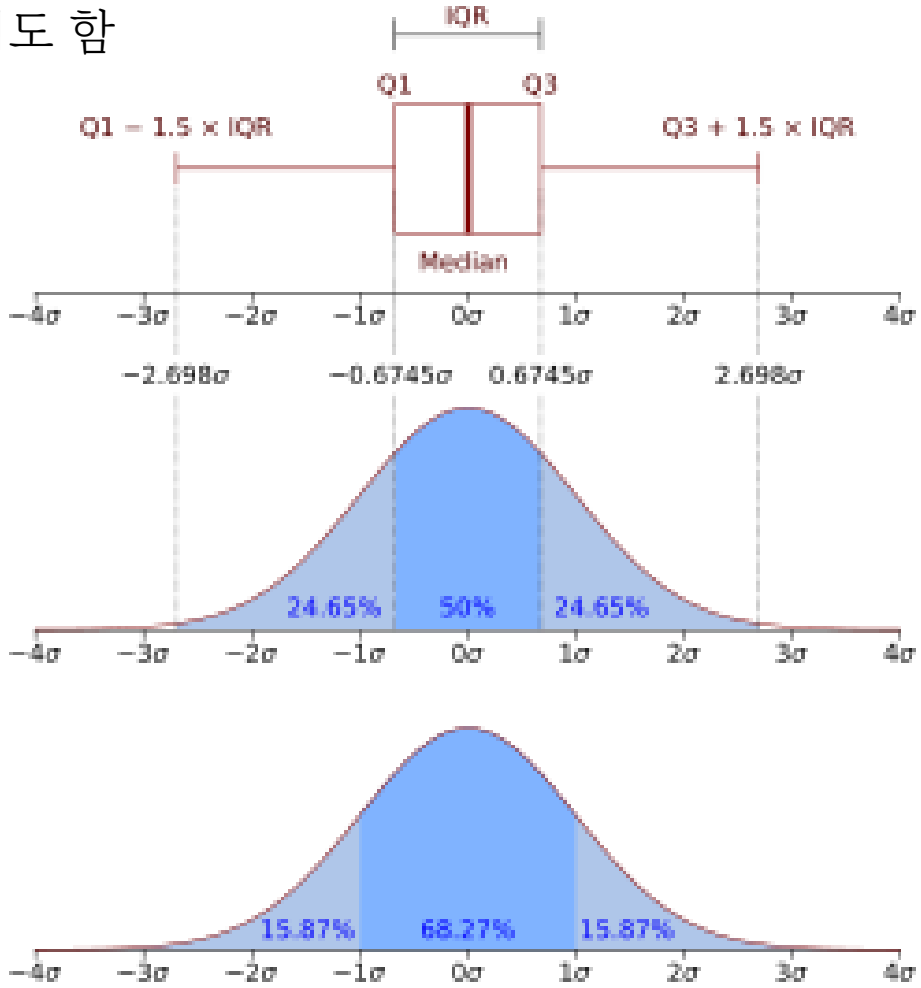
Deep Learning Flow – 1. Preprocessing

Statistical analysis

Mean, median, mode / min, max / variance / percentile (0,25,50,75,100%)

- 정규분포 상에서 $\pm 3\sigma$ 안에 포함될 확률은 99.7% → 3σ 이상의 값들은 outlier로 취급
- IQR(InterQuartile Range, Q3-Q1) 기준 Q1-(IQR*1.5)이하 또는 Q3+(IQR*1.5) 이상은 outlier로 취급
- Skeness가 심한 경우, log-transform을 통해 정규분포로 만들어주기도 함





Deep Learning Flow – 1. Preprocessing

Missing values

결측값이 있으면 모델 학습이 불가능

결측값은 3가지 종류가 있음

- MCAR (Missing Completely At Random, 완전 무작위 결측)
  - 결측값이 관찰값과 상관없이 랜덤하게 발생
  - 센서 오류, 전산 오류, 통신 중 데이터 누락, 응답자 실수로 누락

- MAR (Missing At Random, 무작위 결측)
  - 결측값이 관찰값이랑 관련이 있음, 다른 변수값에 따른 조건부 발생
  - 해시계 측정값 (시간이란 변수가 밤이 되면 측정 불가), 일정 습도 이상에서는 온도 측정 불가 -> 원인이 분명히 존재!

- MNAR (Missing At Not Random, 비 무작위 결측)
  - 결측값이 관찰값이랑 관련이 있음, 다른 변수와는 관련이 없음
  - 응답자들이 고의로 사실과 다르게 응답

Deep Learning Flow – 1. Preprocessing

Missing values

결측치 처리 방법

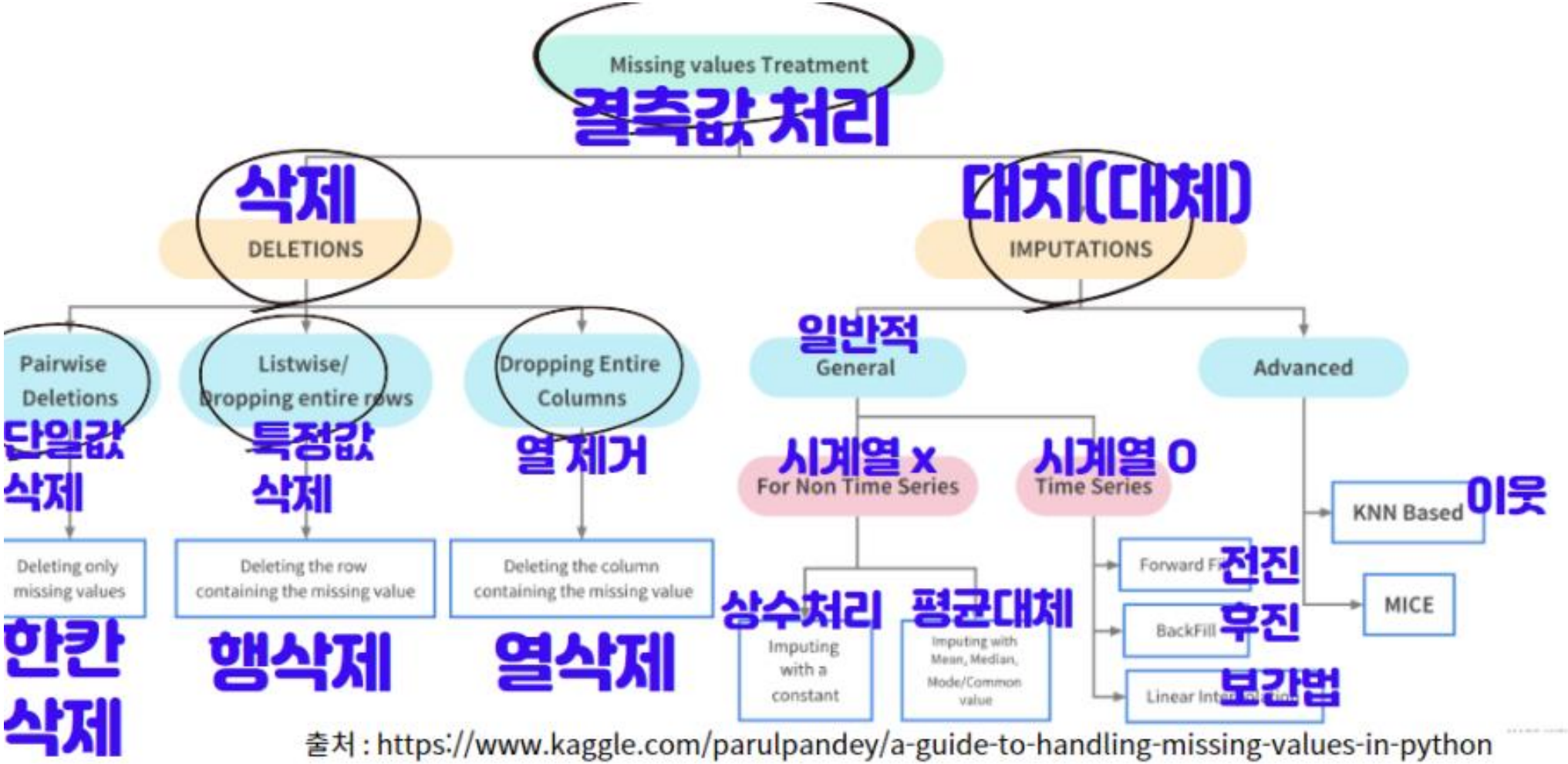
- 결측값이 매우 소량인 경우, 해당 샘플을 삭제해도 무방
- 보통 0 또는 평균값이나 최빈값으로 대체
- 좀 더 복잡하게는 regression 모델을 만들어서 채우기도 함

제거법	완전제거법	하나의 변수라도 결측치 존재시 분석 대상에서 제외하고, 모든 변수값이 존재할 때만 분석 대상에 포함
단일대체법	평균대체법	관측자료의 평균값으로 대체
	연역적 대체법	같은 조사표 상의 비슷한 다른 항목에서 주어진 응답 패턴을 근거로 연역적으로 유추하여 결측값을 대체
	일치대응대체법	결측된 정보를 다른 조사자료로부터 얻을 수 있는 경우, 동일한 조사 단위에 해당하는 다른 외부자료의 값으로 대체
	핫덱대체법	다른 변수에서 동일한 특성을 갖는 응답 값을 랜덤 샘플링하여 그 값으로 대체
	회귀대체방법	회귀분석을 실시한 결과로 얻은 추정치로 대체
다중대체법		결측치를 제외한 나머지 변수들로 해당 결측치를 예측하며, 여러 번 반복을 통해 하나의 결측치에 대해 다수의 대체값을 생성

Deep Learning Flow – 1. Preprocessing

Missing values

결측치 처리 방법 (참고)



출처 : https://www.kaggle.com/parulpandey/a-guide-to-handling-missing-values-in-python

Deep Learning Flow – 2. Data Split

Train only

- 여러분에게 치와와, 쿠키 사진 10장이 주어짐
- 치와와, 쿠키 사진들을 가지고 둘을 구분하는 모델을 학습
- 모델을 이것저것 바꿔가며 학습 진행

$f_{w_1}(\cdot)$ — w_1: i-th weight values; f: Model structure (CNN, GNN 등)

→ $f_{w_1}(\cdot)$ W^*_{f1} → 치와와 쿠키

...

→ $g_{w_1}(\cdot)$ W^*_{g1} → 치와와 쿠키

Loss 가장 작은 모델이랑 Loss를 최소화시키는 W^*를 찾음!!!

찾은 모델이 성능은 어느정도야? 얼마나 잘 구분해?

학습하느라 다 써서 실제로 테스트해볼 데이터가 없는데...? 1개는 성능평가로 빼놓자!

Deep Learning Flow – 2. Data Split

Train and Test

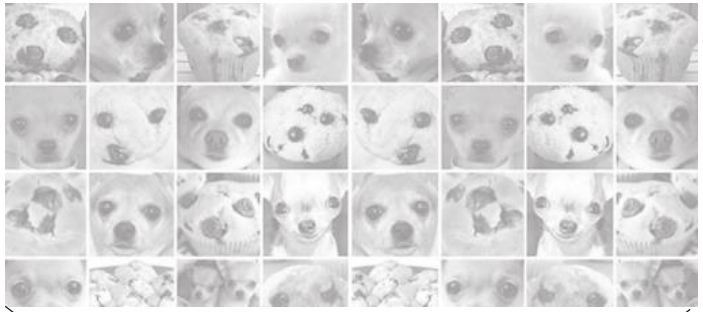
→ $f_{w_1}(\cdot)$ W^*_{f1} → 치와와 쿠키

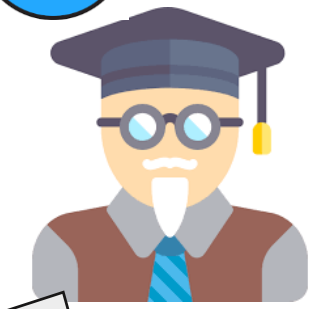
→ $g_{w_1}(\cdot)$ W^*_{g1} → 치와와 쿠키

→ $f_{w_1}(\cdot)$ W^*_{f1} → 쿠키 !!
80%!

→ $g_{w_1}(\cdot)$ W^*_{g1} → 치와와 !!
100%!!
(Best Model)

(제일 잘나온거로 말해야지 룰루~)
학습하지 않은 데이터로 성능을 확인해봤는데 정확도 100% 나와요!!

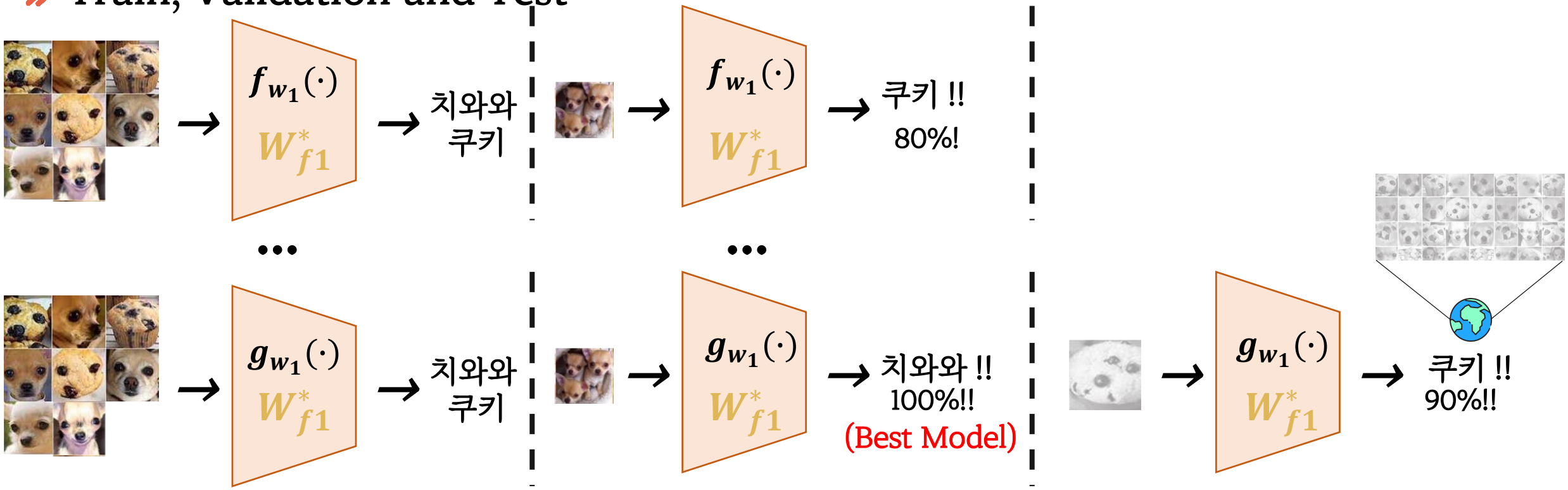


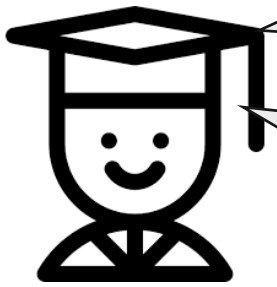


근데 그거 이세상에 존재하는 치와와랑 쿠키 다 테스트 해봄? 100% 보장 가능????

Deep Learning Flow – 2. Data Split

Train, Validation and Test



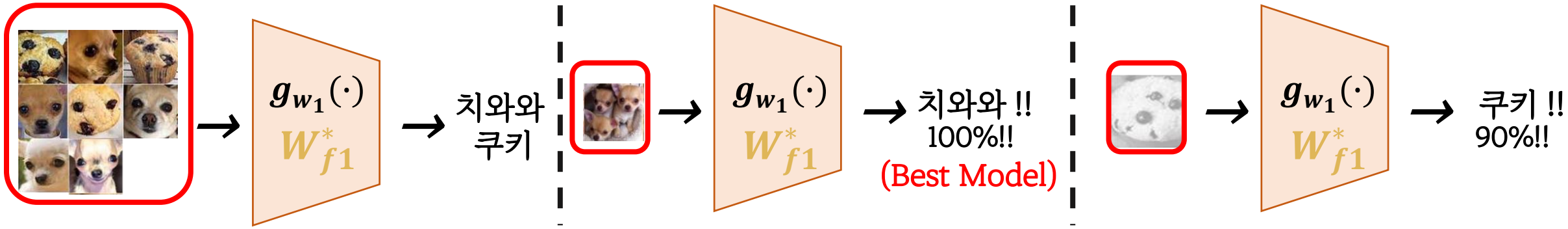


이세상에 모든 치와와랑 쿠키 사진을 가져올수도 없고...ㅠㅠㅠ
그래 그럼 이세상의 모든 치와와랑 쿠키 사진은 내가 아직 볼 수 없는 Unseen data니까... 내가 가지고 있는 사진 중 일부를 unseen이라고 가정하자! 그럼 이 unseen까지 잘 맞추면 실제로 내가 아직 보지 못한 이세상 사진들도 잘 구분할거야!!

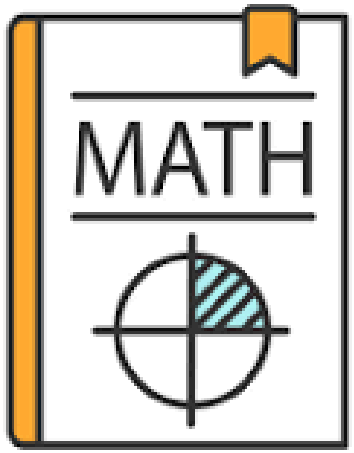
Unseen data에서는 정확도가 90%로 떨어졌네! 내가 아직 구하지 못한 이세상 데이터들에서는 정확도가 90%정도 나오겠구만!!
그러면 이렇게 사용한 unseen data는 실제로 test에 사용된 거니까 'test set'이라 하고, 최고의 모델을 찾는데 사용한 데이터는 'validation set'이라고 하자!! 학습에 사용된 것은 'train set'이겠네!!

Deep Learning Flow – 2. Data Split

Train, Validation and Test

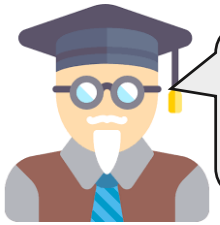


‘Train set’ ‘Validation set’ ‘Test set’



Deep Learning Flow – 2. Data Split

K-fold Cross Validation



Train이랑 validation이 럭키비키하게 나눠진 것일 수도 있으니, 두 데이터 조합을 다양하게 평가해보게나.

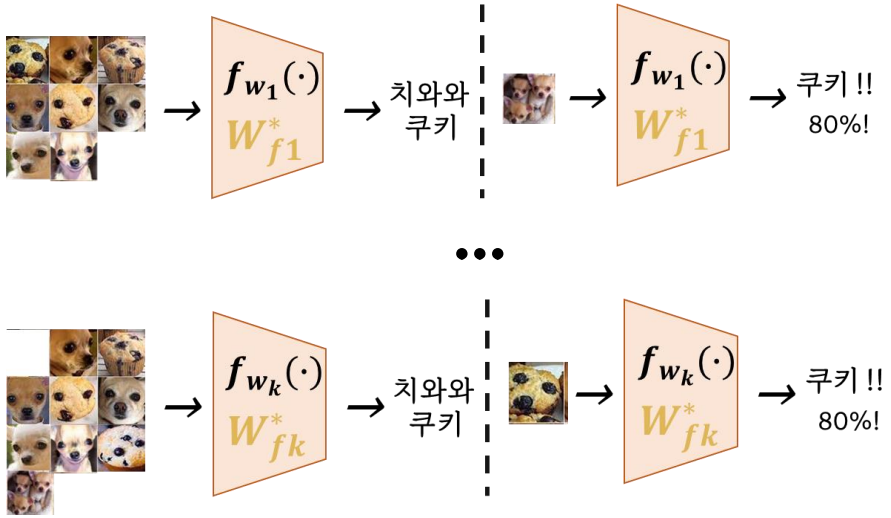
All Data

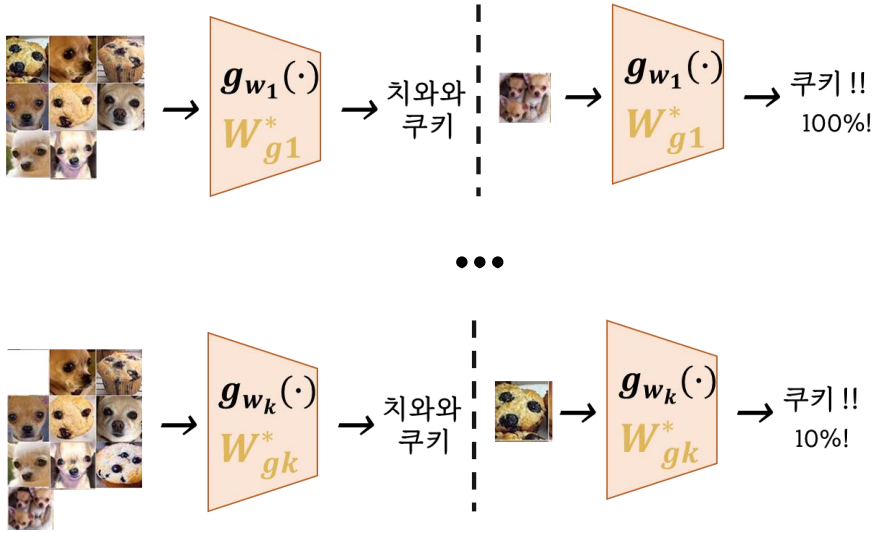
Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Finding Parameters

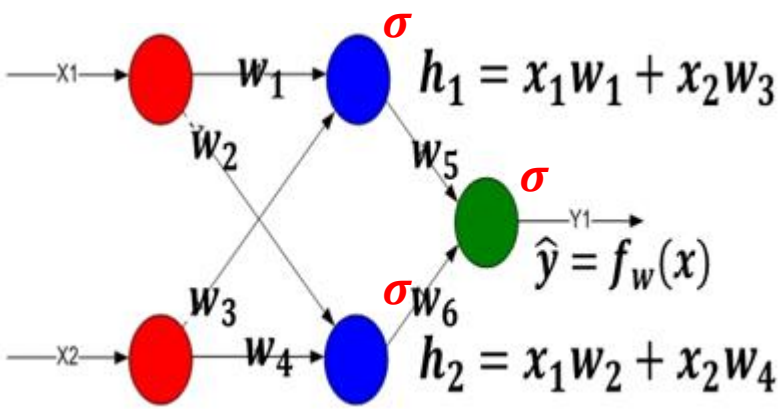
Final evaluation: Test data





Deep Learning Flow – 3. Model Training

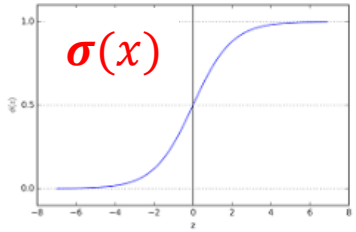
Activation Function



① Define $f_w(\cdot) \rightarrow$ 'Model'

$h_1 = \boldsymbol{\sigma}(x_1w_1 + x_2w_3),\ h_2 = \boldsymbol{\sigma}(x_1w_2 + x_2w_4)$

$\hat{y} = f_w(x) = \boldsymbol{\sigma}(h_1w_5 + h_2w_6)$



If it doesn't exist,

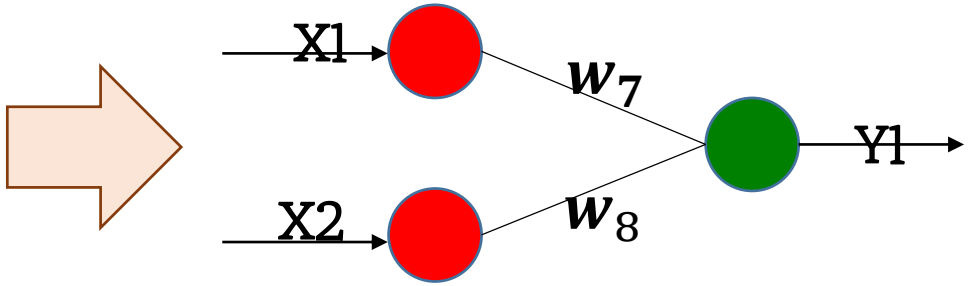
$$\bullet = \bullet \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}$$

$$\bullet = \bullet \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}$$

Then,

$$\bullet = \bullet \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} w_5 \\ w_6 \end{bmatrix}$$

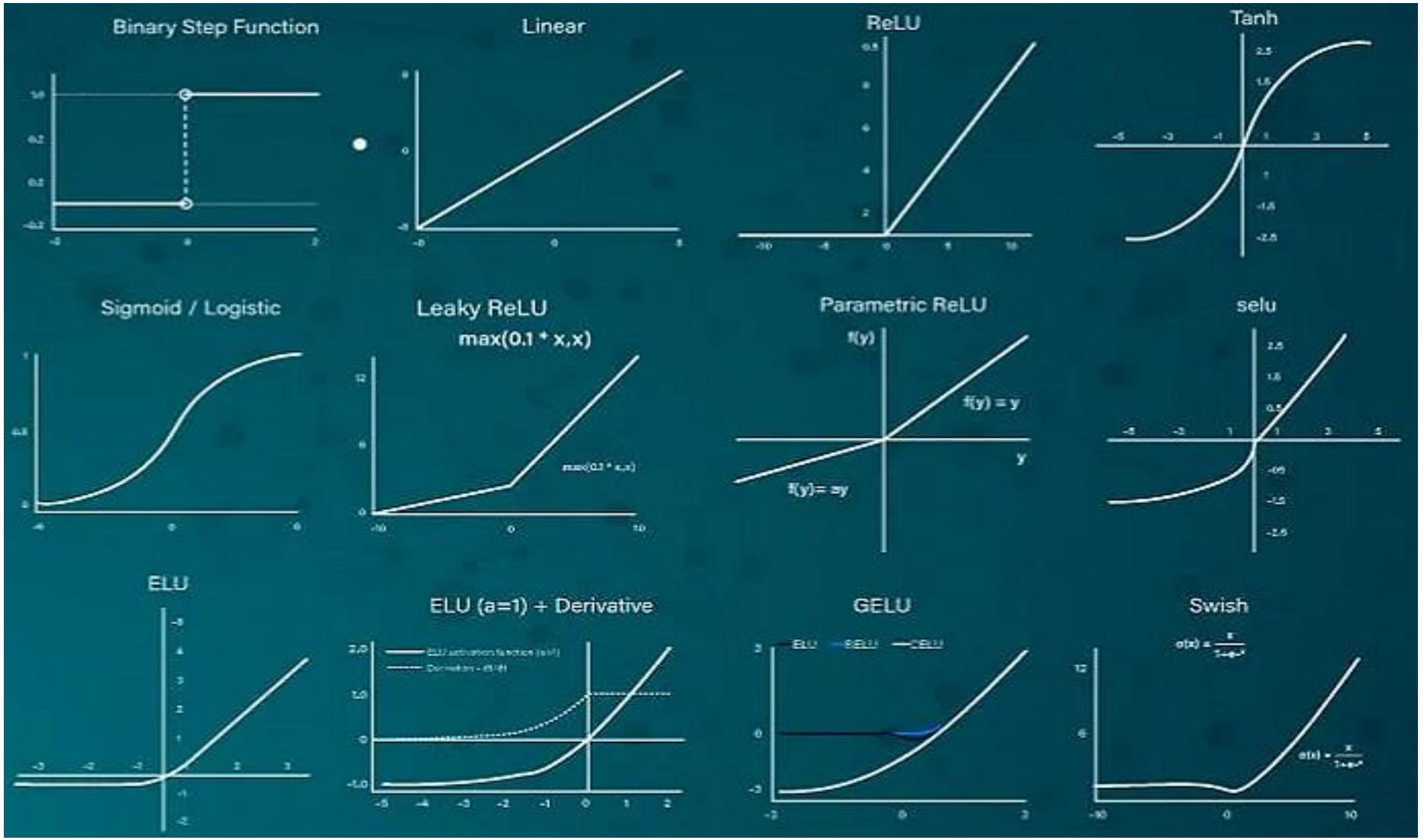
$$= \bullet \begin{bmatrix} w_1w_5 + w_2w_6 \\ w_3w_5 + w_4w_6 \end{bmatrix} =: \begin{bmatrix} w_7 \\ w_8 \end{bmatrix}$$



Deep Learning Flow – 3. Model Training

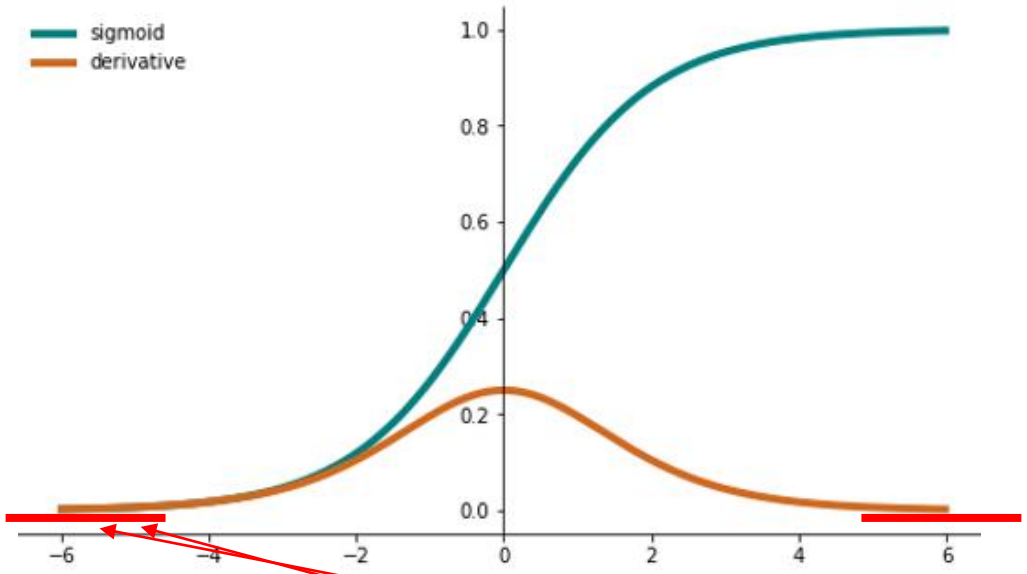
Activation Function

- There are various types of activation functions.

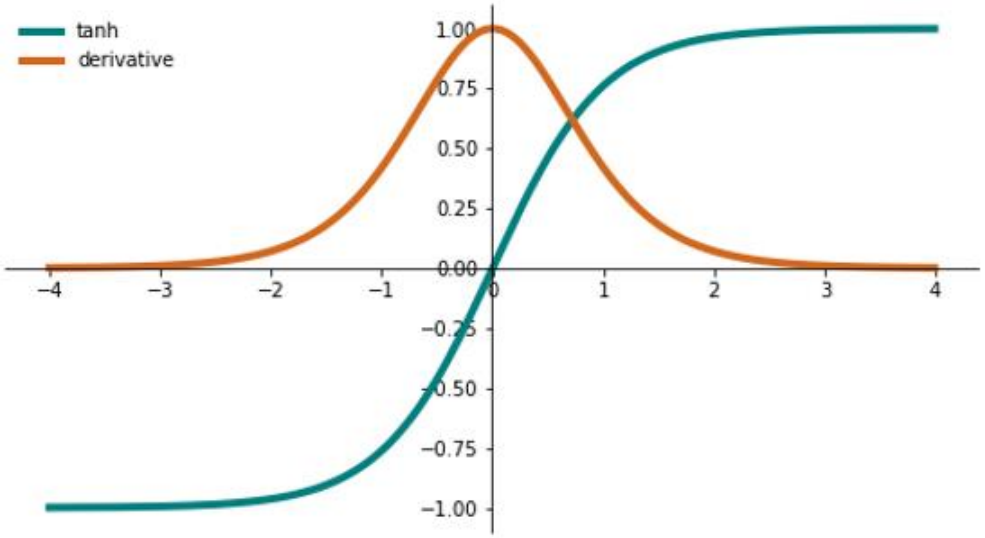


Deep Learning Flow – 3. Model Training

Activation Function



< Sigmoid >



< tanh >

$$
\begin{cases}
h_1 = \boldsymbol{\sigma}(x_1{}^{(i)} w_1 + x_2{}^{(i)} w_3) \\
\hat{y}^{(i)} = \boldsymbol{\sigma}(h_1 w_5 + h_2 w_6) \\
\mathcal{L}^{(i)}(y, \hat{y}) = \left(y^{(i)} - \hat{y}^{(i)}\right)^2 \\
\mathcal{L}(y, \hat{y}) = \sum \mathcal{L}^{(i)}(y, \hat{y})
\end{cases}
$$

$$
\frac{\partial \mathcal{L}}{\partial w_1} = \sum \frac{\partial \mathcal{L}^{(i)}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \boldsymbol{\sigma}} \frac{\partial \boldsymbol{\sigma}}{\partial h_1} \frac{\partial h_1}{\partial \boldsymbol{\sigma}} \frac{\partial \boldsymbol{\sigma}}{\partial w_1}
$$

만약 $\frac{\partial \boldsymbol{\sigma}}{\partial h_1}$ or $\frac{\partial \boldsymbol{\sigma}}{\partial w_1}$가 -6보다 작거나 6보다 크다면 미분값=0이 되어버림

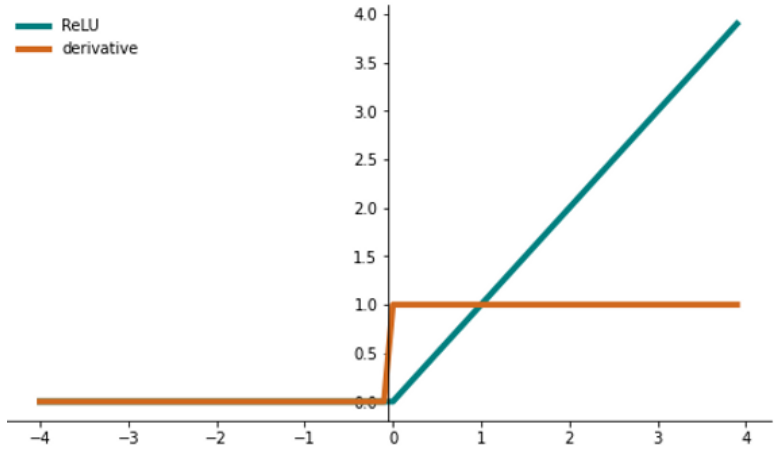
$$
\boldsymbol{W^k} - \alpha \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W^{k-1}})
$$

$$
\begin{bmatrix} \boldsymbol{w_1^k} \\ \cdots \\ \boldsymbol{w_6^k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w_1^{k-1}} - \alpha \left.\frac{\partial \mathcal{L}}{\partial w_1}\right|_{W=\boldsymbol{W^{k-1}}} \\ \cdots \\ \boldsymbol{w_6^{k-1}} - \alpha \left.\frac{\partial \mathcal{L}}{\partial w_6}\right|_{W=\boldsymbol{W^{k-1}}} \end{bmatrix}
$$

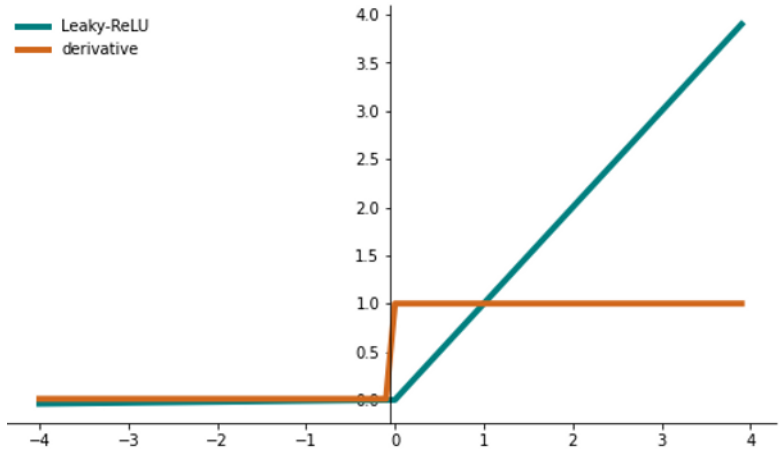
Weight가 업데이트되지 않고 중단됨

Deep Learning Flow – 3. Model Training

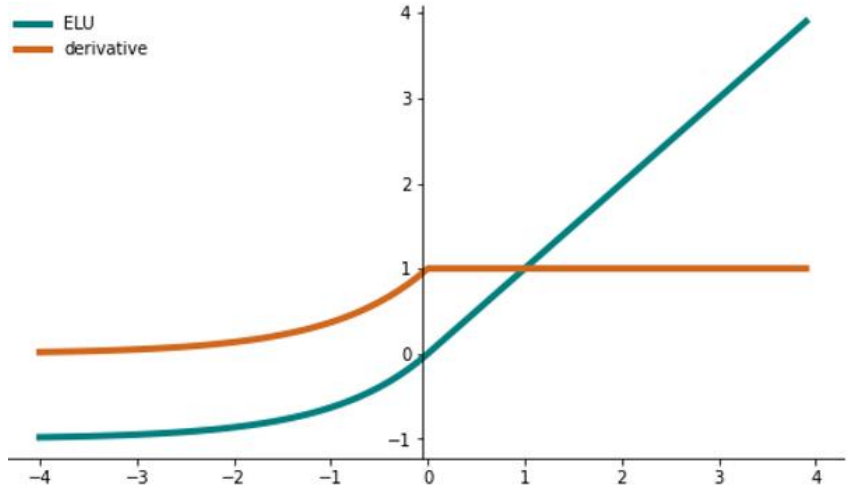
Activation Function



< ReLu >



< Leaky ReLu >



< ELU >

반면, ReLU는 미분값이 양수일 경우에는 gradient가 1만큼 흘러가기 때문에 업데이트가 지속됨.
또한 ReLU를 약간 수정해서 미분값이 음수일 경우에도 약간의 gradient가 흘러가도록 한 변종들이 존재

$$
\begin{cases}
h_1 = \boldsymbol{\sigma}(x_1{}^{(i)} w_1 + x_2{}^{(i)} w_3) \\
\hat{y}^{(i)} = \boldsymbol{\sigma}(h_1 w_5 + h_2 w_6) \\
\mathcal{L}^{(i)}(y, \hat{y}) = \left(y^{(i)} - \hat{y}^{(i)}\right)^2 \\
\mathcal{L}(y, \hat{y}) = \sum \mathcal{L}^{(i)}(y, \hat{y})
\end{cases}
\Rightarrow
\frac{\partial \mathcal{L}}{\partial w_1} = \sum \frac{\partial \mathcal{L}^{(i)}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \boldsymbol{\sigma}} \frac{\boldsymbol{\partial \sigma}}{\partial h_1} \frac{\partial h_1}{\partial \boldsymbol{\sigma}} \frac{\boldsymbol{\partial \sigma}}{\partial w_1}
$$

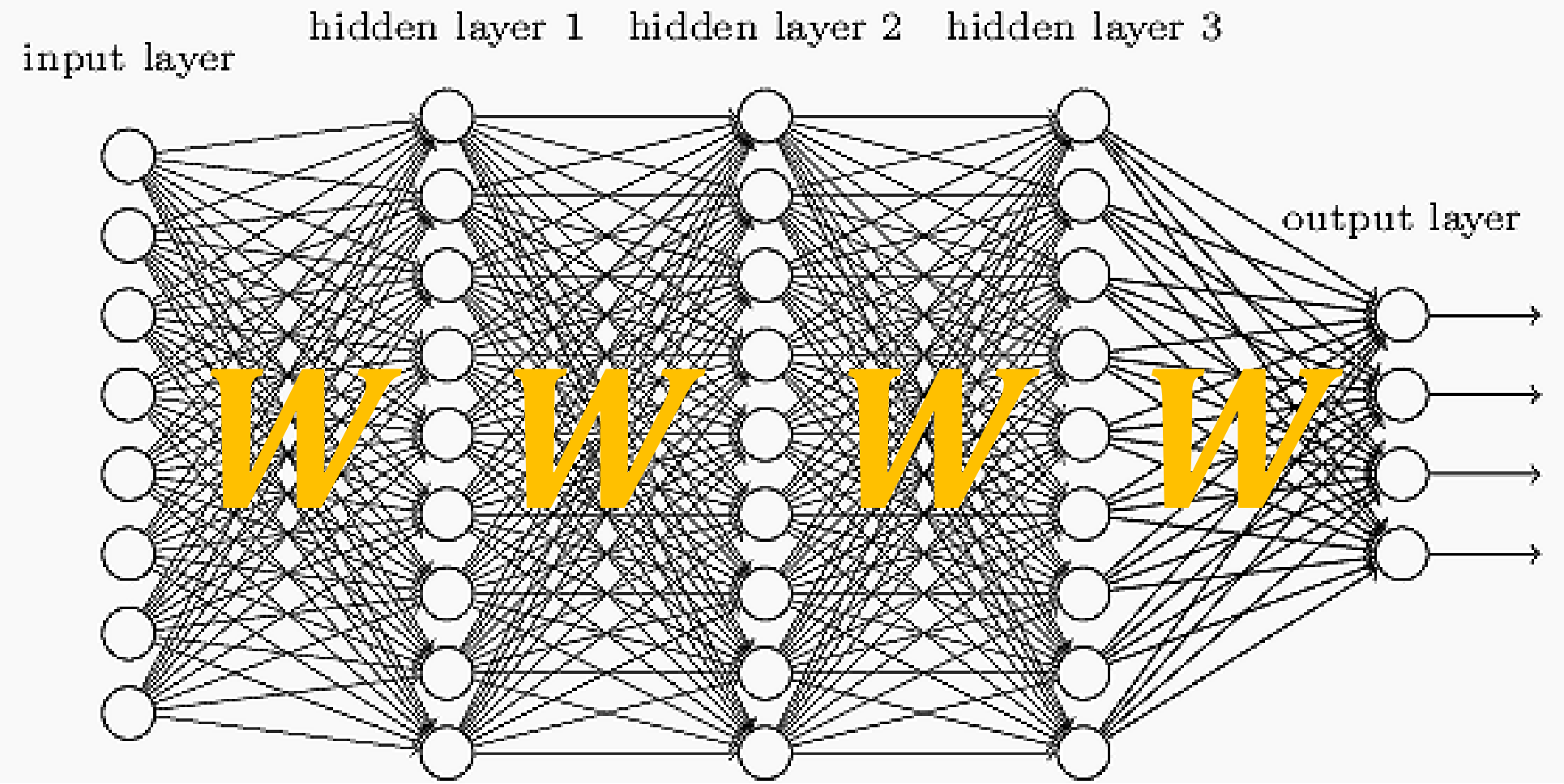
$$
\boldsymbol{W^k - \alpha \nabla_w \mathcal{L}(W^{k-1})}
$$

$$
\begin{bmatrix} \boldsymbol{w_1^k} \\ \dots \\ \boldsymbol{w_6^k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w_1^{k-1}} - \alpha \left.\dfrac{\partial \mathcal{L}}{\partial w_1}\right|_{W=\boldsymbol{W^{k-1}}} \\ \dots \\ \boldsymbol{w_6^{k-1}} - \alpha \left.\dfrac{\partial \mathcal{L}}{\partial w_6}\right|_{W=\boldsymbol{W^{k-1}}} \end{bmatrix}
$$

Deep Learning Flow – 3. Model Training

Initializer

- **모델들의 W값들을 초기화**
- **다음 layer로 입력되는 값들은 이전 layer output들의 weighted sum.**
  - 이전 layer의 node 수가 많아질수록 weight 값은 반비례해야 다음 layer에 입력이 exploding하지 않음



Deep Learning Flow – 3. Model Training

Initializer

Xavier Initialization (w/ Sigmoid, tanh)

$$W \sim N(0, Var(W)) \qquad Var(W) = \sqrt{\frac{2}{n_{in} + n_{out}}}$$

< Xavier Normal Initialization >

$$W \sim U(-\sqrt{\frac{6}{n_{in} + n_{out}}}, \quad +\sqrt{\frac{6}{n_{in} + n_{out}}})$$

< Xavier Uniform Initialization >

He Initialization (w/ ReLU)

$$W \sim N(0, Var(W)) \qquad Var(W) = \sqrt{\frac{2}{n_{in}}}$$

< He Normal Initialization >

$$W \sim U(-\sqrt{\frac{6}{n_{in}}}, \quad +\sqrt{\frac{6}{n_{in}}})$$

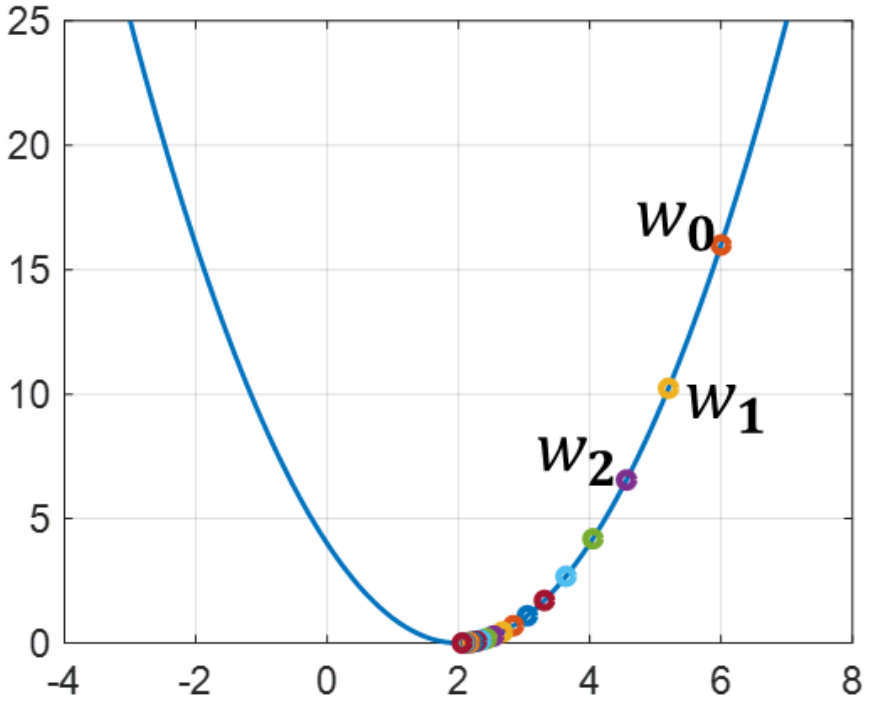
< He Uniform Initialization >

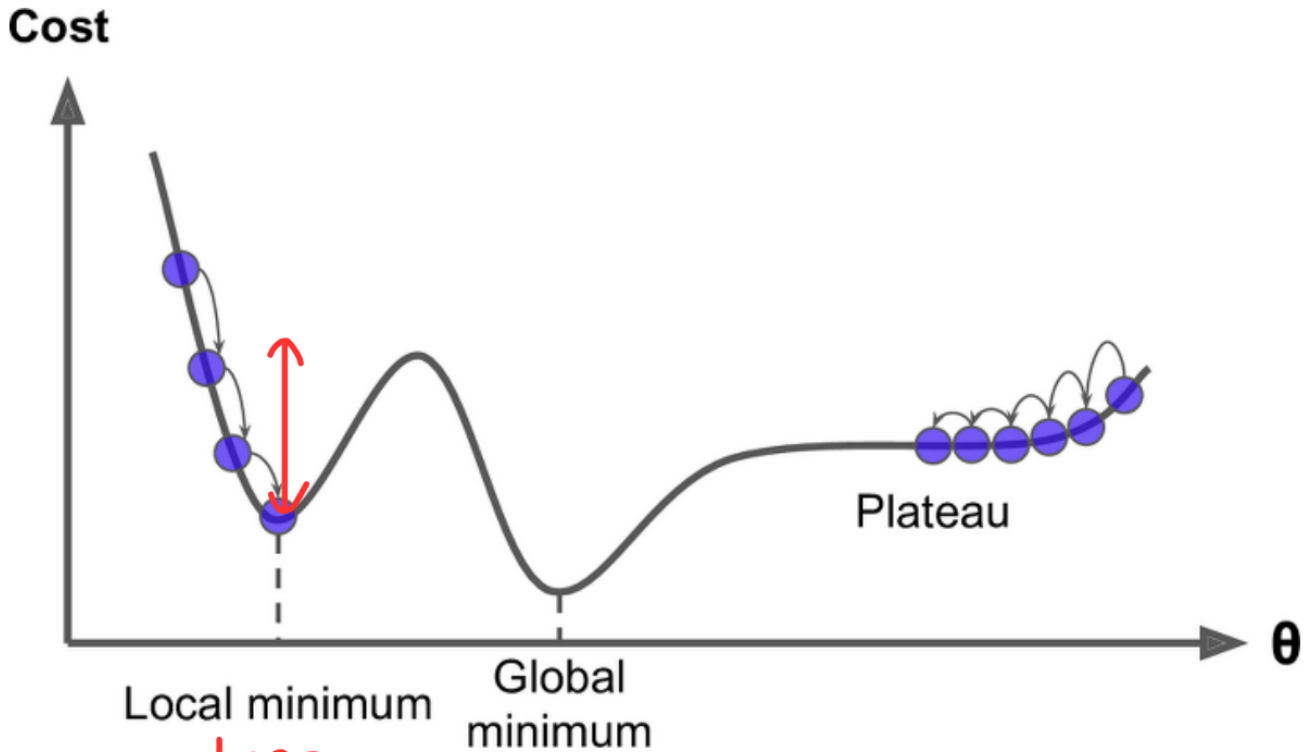
Deep Learning Flow – 3. Model Training

Optimizer

- Weight를 얼마나 어떻게 업데이트할 것인가!
- Goal : Local Minimum & Plateau 구간 탈출

$$w_k = w_{k-1} - \alpha \cdot \left.\frac{dL}{dw}\right|_{w=w_{k-1}}$$

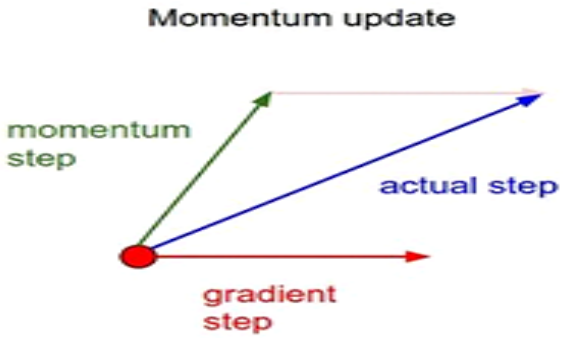




Deep Learning Flow – 3. Model Training

Optimizer

- Momentum : 이전에 왔던 방향도 함께 고려한다!
- Adaptive : 많이 업데이트된 weight는 적게, 적게 업데이트된 weight는 많이!



$$w_i^t = w_i^{t-1} - \eta g_i^t$$

< Standard GD >

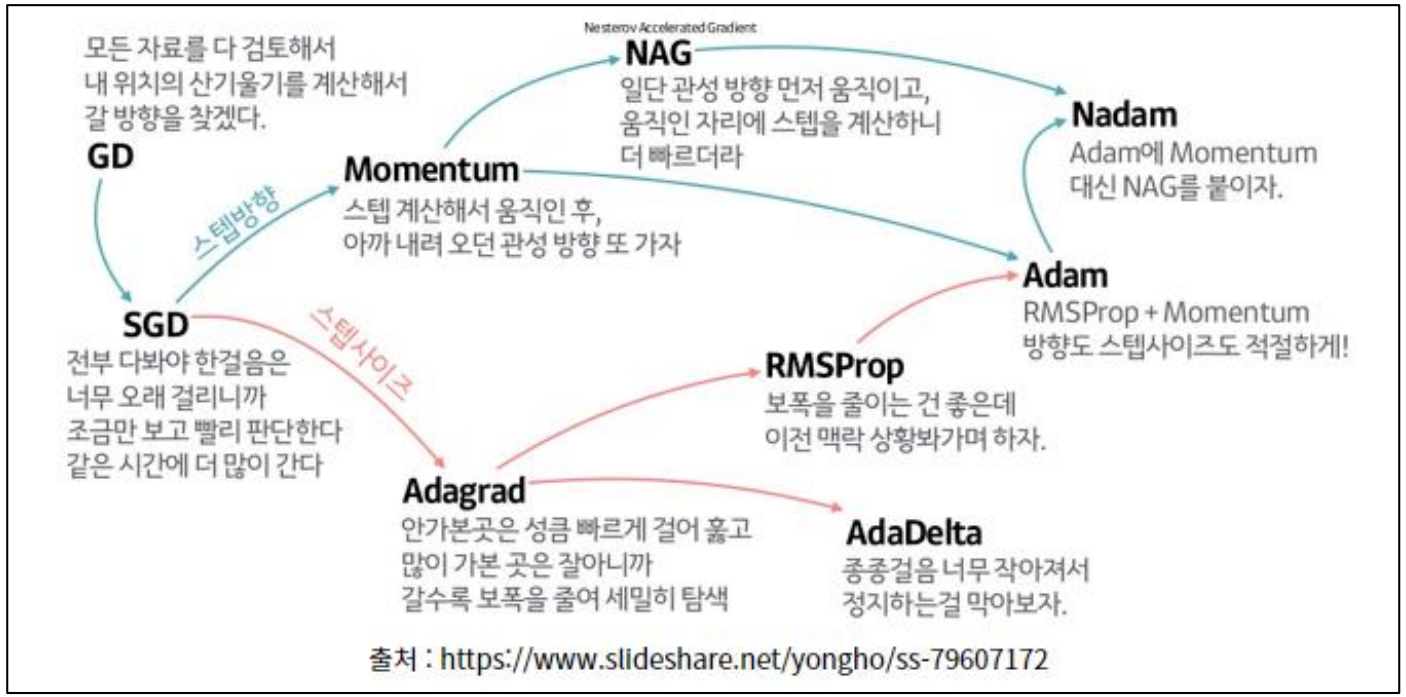
$$G_i^t = G_i^{t-1} + \left(g_i^t\right)^2$$

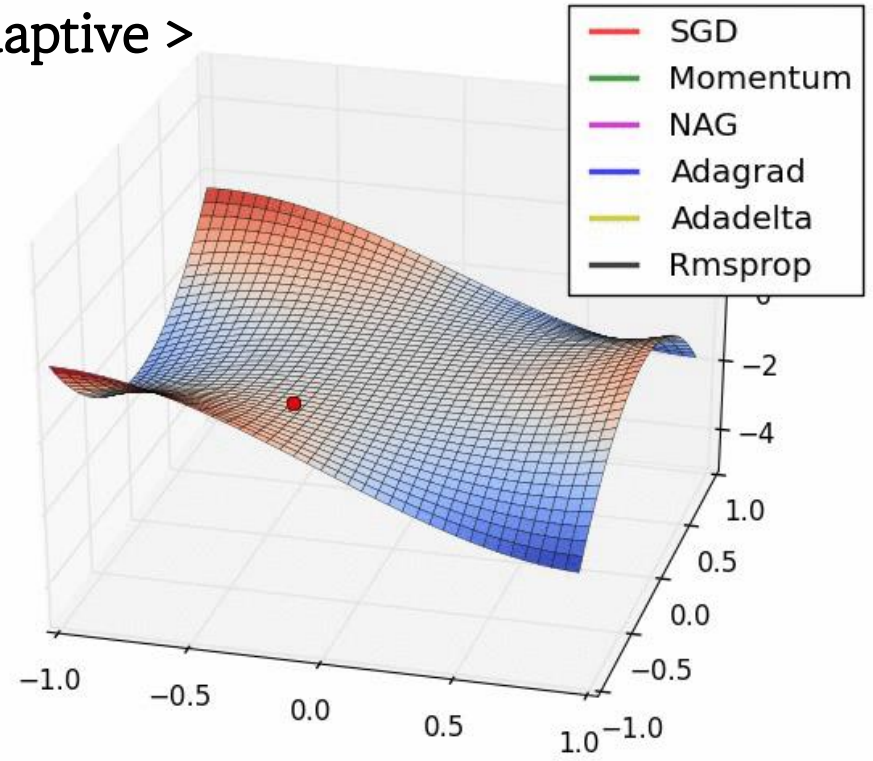
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

< Adaptive >

g_i^t: parameter i의 시간 t에서의 기울기

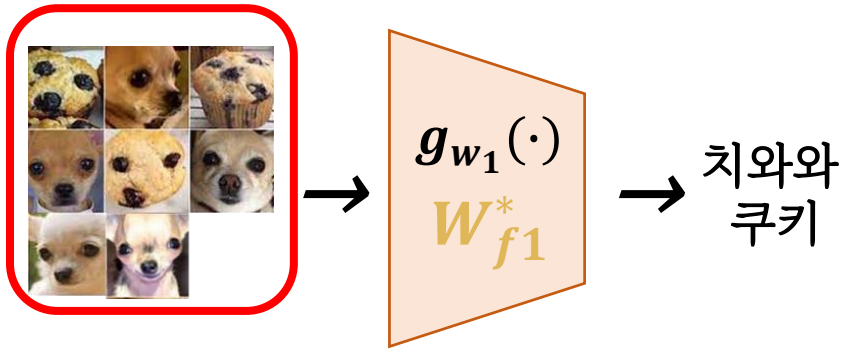
G_i^t: parameter i가 지금까지 update된 총량



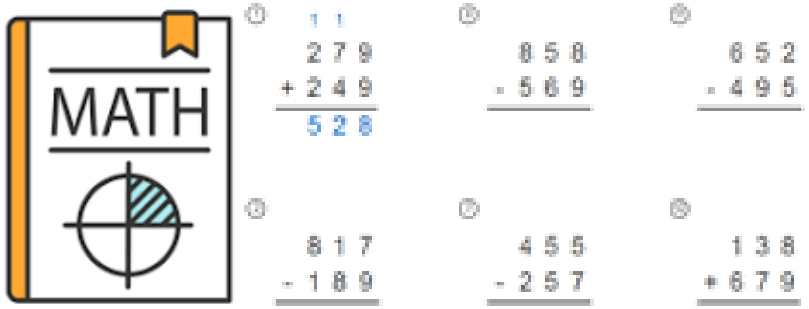


Deep Learning Flow – 3. Model Training

Epoch, Batch Size

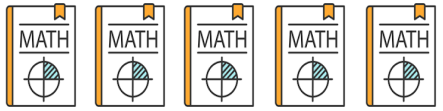


'Train set'



Epoch : 책을 총 몇 번 풀어볼 것이냐!

Epoch = 1 ; 

Epoch = 5 ; 

→ **데이터를 총 몇 번 반복학습할 것인가!!**

Batch size : 몇 문제씩 풀어보고 답을 맞춰볼 것이냐!

Batch = 1 ; 

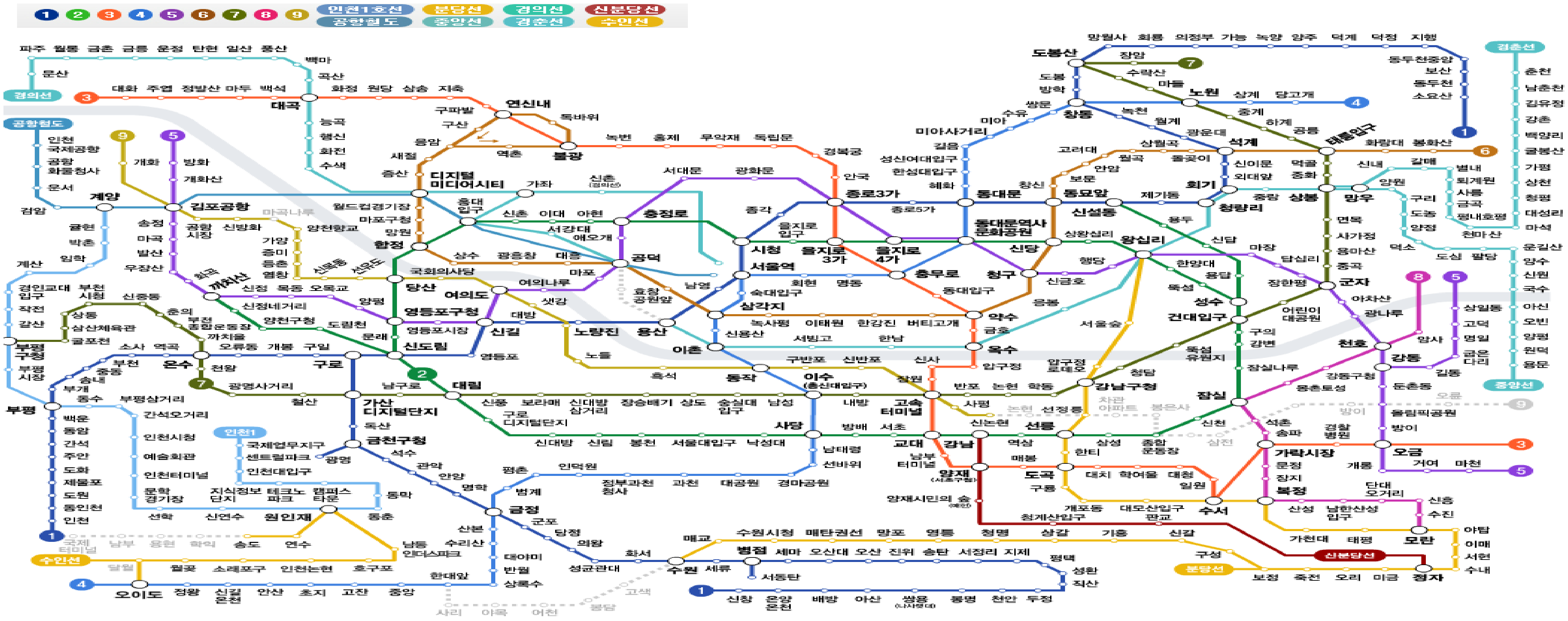
Batch = 5 ; 

→ **몇 개의 데이터를 보고 난 후에**
loss 계산 / weight update를 할 것인가!!

Contents

Regression Task : 하루동안 숙대입구에서 하차한 사람은 몇 명일까?

- 수도권에는 24개 노선, 648개 역이 존재
- 숙대입구에서 하차한 인원수는 다른 역에서 하차한 사람들과 연관성이 있다고 가정 ('숙대입구 하차인원 ~ 용산역 + 공덕역 + … 하차인원')
- 하차 인원수를 파악하여 상권 분석 등에 활용 (?)



Activity

Preprocessing (raw_data.ipynb)

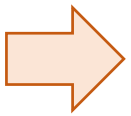
데이터셋은 서울교통공사에서 제공하는 '서울시 지하철호선별 역별 승하차 인원 정보' 데이터를 활용

- 2015년 1월부터 2024년 10월까지 일별 역별 승하차 인원수 데이터를 제공

Preprocessing

- csv file 별 인코딩 호환성 확인 및 매칭
- 사용일자별 '노선명_역명'을 column으로 하여 하차총승객수를 값으로 갖도록 변환
- 데이터 병합

	사용일자	노선명	역명	승차총승객수	하차총승객수	등록일자
1	20240701	9호선2~3단계	석촌고분	7665	7414	20240704
2	20240701	중앙선	도농	12178	10957	20240704
3	20240701	중앙선	구리	14053	14085	20240704
4	20240701	4호선	길음	22687	22215	20240704
5	20240701	분당선	복정	1	0	20240704



사용일자	1호선_동대문	1호선_동묘앞	1호선_서울역	1호선_시청	1호선_신설동	1호선_제기동	1호선_종각	1호선_종로3가	1호선_종로5가
20150101	10579.0	5654.0	40197.0	8943.0	6196.0	7891.0	17648.0	15817.0	9216.0
20150102	16551.0	8261.0	67738.0	25791.0	16121.0	18609.0	55013.0	41956.0	28787.0
20150103	16545.0	10657.0	51578.0	15098.0	10755.0	19106.0	32763.0	32560.0	24532.0

Activity

Code Snippet

```
1 tr_X_scaled
```

1일치 다른 역 하차자 수(총 2883개)

```
array([[-1.11099256, -0.83203816, -0.92640458, ..., -0.2285853 ,
        -0.23163808, -0.12341046],
       [-1.64227996, -0.70787291, -2.34830104, ..., -0.2285853 ,
        -0.23163808, -0.12341046],
       [-0.67830764,  1.35889116, -0.28443319, ..., -0.2285853 ,
        -0.23163808, -0.12341046],
       ...,
       [-0.72241261, -0.83749136,  0.27444142, ..., -0.2285853 ,
        -0.23163808, -0.12341046],
       [-1.15408362,  1.13321244, -2.31600365, ..., -0.2285853 ,
        -0.23163808, -0.12341046],
       [ 0.92696101,  0.83118887, -0.73416211, ..., -0.2285853 ,
        -0.23163808, -0.12341046]])
```

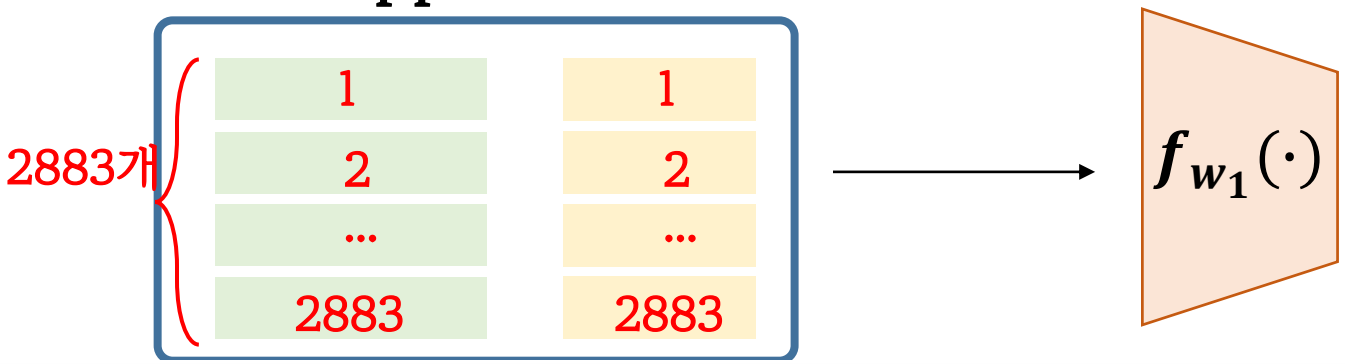
```
1 tr_Y_scaled
```

1일치 숙대입구역 하차자 수

```
array([[0.37869798],
       [0.1384672 ],
       [0.33851631],
       ...,
       [0.46642661],
       [0.14104505],
       [0.30250829]])
```

Activity

Code Snippet




```python
# Creating the dataset class
class CustomDataset(Dataset):
    # Constructor
    def __init__(self, x_, y_): ### (self는 무시하고) 데이터 보따리에 함수 순서 맞춰서 데이터 가져옴
        self.x_data = x_
        self.y_data = y_

    # getting data length
    def __len__(self): ### (self는 무시하고) 데이터 보따리 안에 있는 총 데이터 개수 카운트
        return len(self.x_data)

    # Getter
    def __getitem__(self, idx): ### (self는 무시하고) 데이터 보따리 안에 있는 데이터에 순서대로 index 붙여서 꺼내옴.
        return torch.FloatTensor(self.x_data[idx]), torch.FloatTensor(self.y_data[idx])
```

② tr_X_scaled, tr_Y_scaled를 보따리에 넣음

③ 보따리 안에 있는 총 개수 카운트

④ 순서대로 index 붙이고 꺼낼 준비

```python
tr_dataset = CustomDataset(tr_X_scaled, tr_Y_scaled)
tr_dataloader = DataLoader(tr_dataset, batch_size=256, shuffle=True)

val_dataset = CustomDataset(val_X_scaled, val_Y_scaled)
val_dataloader = DataLoader(val_dataset, batch_size=len(val_X_scaled), shuffle=False)

test_dataset = CustomDataset(test_X_scaled, test_Y_scaled)
test_dataloader = DataLoader(test_dataset, batch_size=len(test_X_scaled), shuffle=False)
```

① 함수 실행, 데이터 보따리 생성

⑤ 주문 내역에 맞게 모델에 배송준비 완료

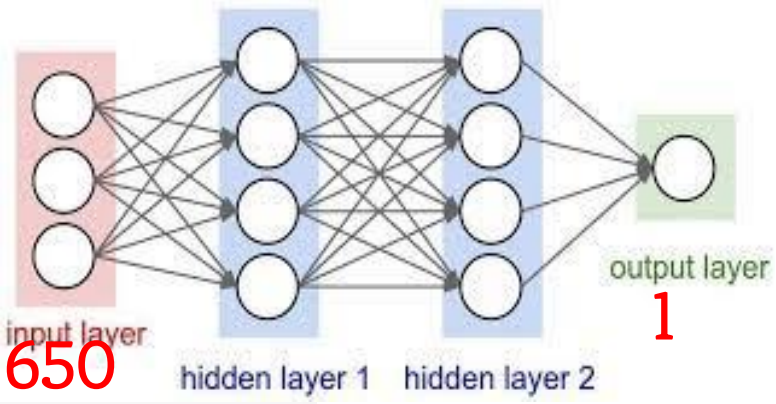
Activity

Code Snippet

2883개

1	1
2	2
...	...
2883	2883

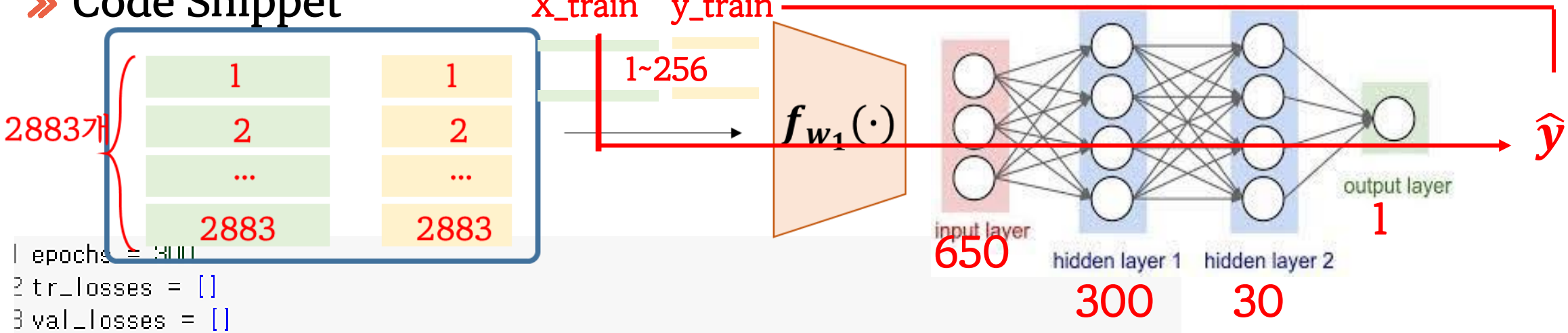
$f_{w_1}(\cdot)$




```python
class Regression(nn.Module):
    def __init__(self):
        super().__init__() # 모델 연산 정의
        self.hidden = nn.Sequential(
            nn.Linear(650, 300), # 입력층(650) -> 은닉층1(300)으로 가는 연산
            nn.ReLU(),
            nn.Linear(300, 30), # 은닉층1(300) -> 은닉층2(30)으로 가는 연산
            nn.ReLU(),
            nn.Linear(30, 1), # 은닉층2(30) -> 출력층(1)으로 가는 연산
            nn.ReLU() ## output 변수는 >0
        )

        #### initializer
        for m in self.modules():
            if isinstance(m, nn.Linear):
                nn.init.xavier_normal_(m.weight.data)



    def forward(self, x): # 모델 연산의 순서를 정의, model() 함수가 호출될 때마다 자동으로 실행됨!!!!
        y_pred = self.hidden(x)
        return y_pred
```

Activity

Code Snippet




```
epochs = 3000
tr_losses = []
val_losses = []
for epoch in range(epochs + 1):
    model.train() ### Train mode로 변경
    for batch_idx, samples in enumerate(tr_dataloader): ### 다음 batch 가져옴
        x_train, y_train = samples
        x_train = x_train.to(device) ### GPU에 올림
        y_train = y_train.to(device)

        y_train_pred = model(x_train) ### forward() 실행 -> feed forward!!

        tr_loss = loss_f(y_train_pred, y_train) ### loss 계산
        optimizer.zero_grad() ### 이전에 계산했던 gradient들 초기화 (누적되면 안됨)
        tr_loss.backward() ### gradient계산 및 back propagation
        optimizer.step() #### update
    tr_losses.append(tr_loss.item())
```

① 다음 batch 가져옴

② 순서대로 x, y 분리

③ model.forward(x_train) 실행 → feed forward!

④ loss 계산

⑤ gradient 계산 및 update

```
def forward(self, x): # 모델
    y_pred = self.hidden(x)
    return y_pred
```

References

References List

- https://www.kdnuggets.com/2019/11/tips-class-imbalance-missing-labels.html
- https://x.com/levikul09/status/1718193560682455358
- https://www.shutterstock.com/ko/search/mean-median-mode
- https://www.indusmic.com/post/activation-functions-in-neural-network
- https://medium.com/@byanalytixlabs/activation-functions-in-neural-networks-its-components-uses-types-23cfc9a7a6d7