ML & DL 살펴보기

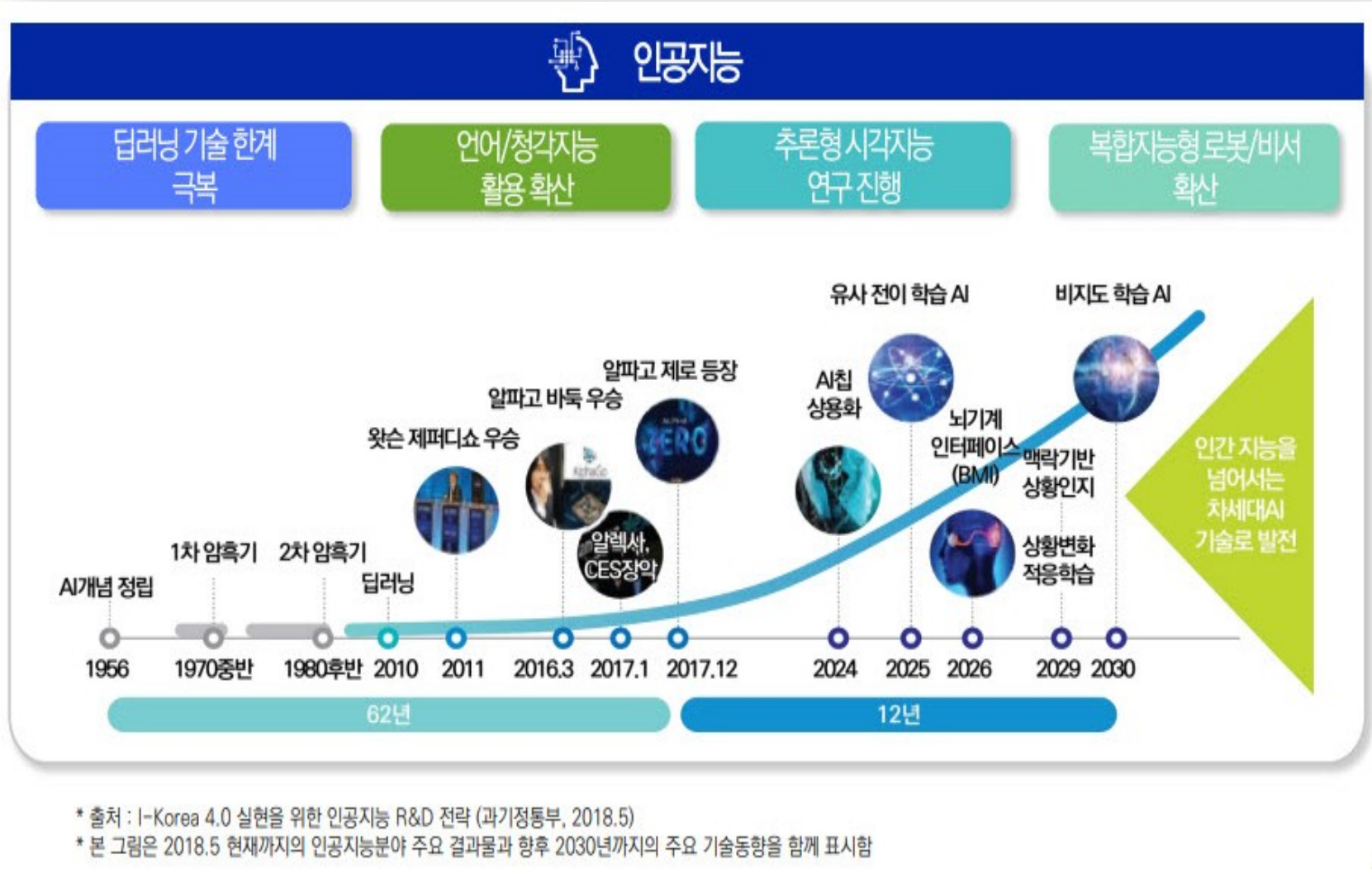
ML & DL 살펴보기

◆ AI 역사

- 1955년 존 매카시 논문 <지능이 있는 기계를 만들기 위한 과학과 공학> 처음 등장
- 1956년 다스머스 학회에서 공식적으로 인공지능이라는 용어 사용
- 뇌를 모사한 인공지능을 뜻하는 퍼셉트론(Perceptron) 용어도 탄생
- 1960년에 접어들면서 인공지능 연구가 본격화
- 1970년대 초까지 인간처럼 생각하고 문제를 푸는 인공지능 연구 계속
- 1970~1980년대 한계
- 1980~1990년대 신경망, 다층 인식론 등 연구
- 2000년대 머신러닝, 딥러닝

ML & DL 살펴보기

◆ AI 발전 역사



* 출처 : I-Korea 4.0 실현을 위한 인공지능 R&D 전략 (과기정통부, 2018.5)
* 본 그림은 2018.5 현재까지의 인공지능분야 주요 결과물과 향후 2030년까지의 주요 기술동향을 함께 표시함

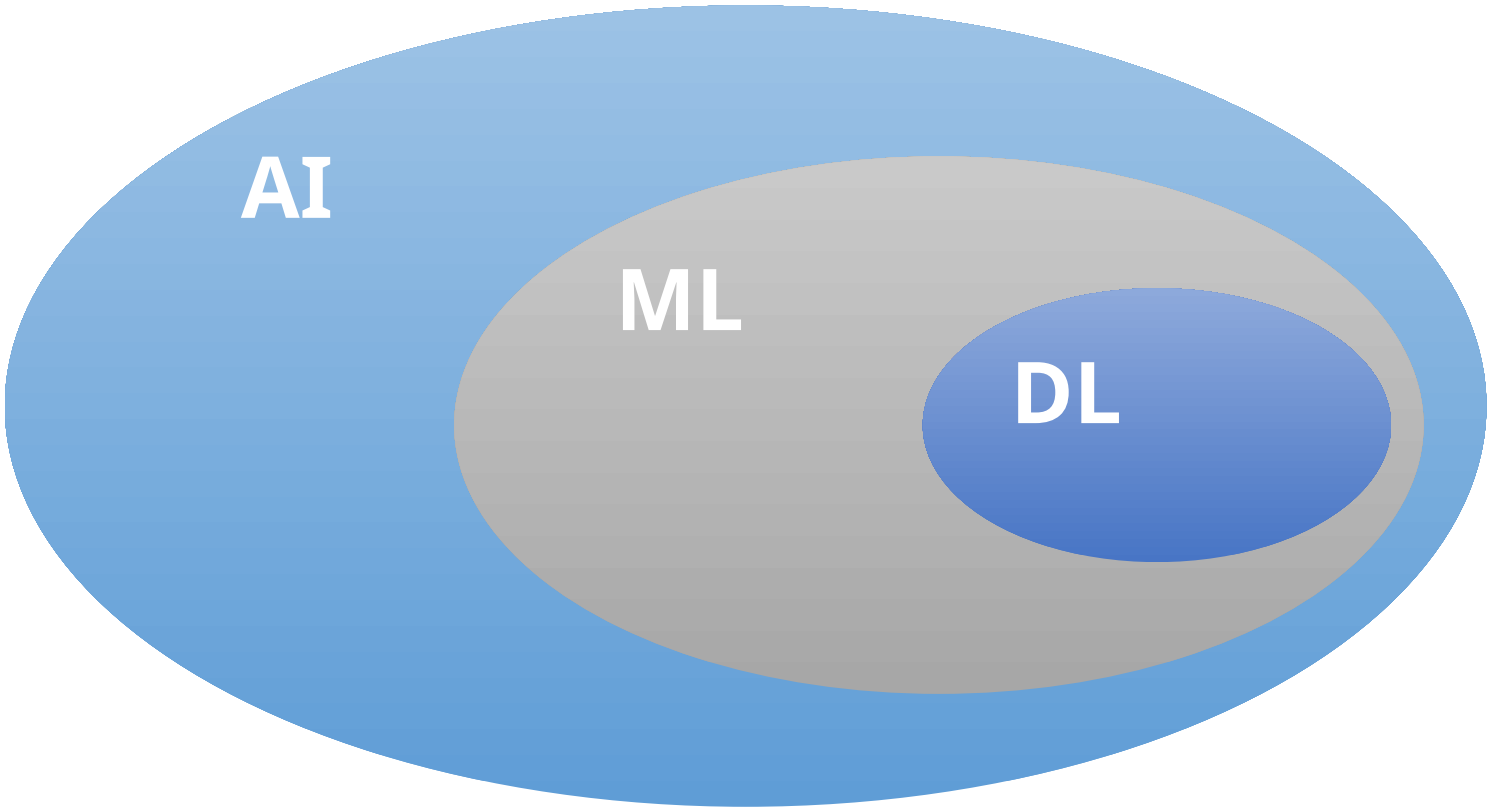
ML & DL 살펴보기

◆ AI 구현 기술

구분	1세대	2세대	3세대	4세대
시기	1950~1980년대	1990년대	2000년 대	2010년 대 ~ 현재
특징	제어 프로그램	경로 및 DB 탐색	머신러닝	딥러닝
내용	- 기계 및 가전제품에 탑재된 단순 제어 프로그램	-대량 정보와 규칙기반 경로 탐색 -DB검색 후 정답 파악 -전문가 시스템	-컴퓨터 스스로 규칙 및 지식 학습 -예측 방법 파악 -인공시경막	-추상화된 특징 표현 등 고급 지식 학습 -데이터 변형 및 인사이트 파악 -깊은 인공신경막
예시	자동 세탁기	검색 DB	문자 및 패턴인식	영상 및 음성인식 자연어 처리

ML & DL 살펴보기

- 머신러닝 & 딥러닝 관계



ML & DL 살펴보기

◆ 머신러닝(Machine Learning)

- 인공지능의 한 분야
- 패턴인식과 컴퓨터 학습 이론의 연구로부터 진화한 분야

- 1949년 Hubbian Learning Theroy를 발표하면서 시작
- 그 이후 1952년 Arthur Samuel이 경험으로부터 배우는 방법을 사용한 최초의 머신러닝 프로그램인 체커 프로그램 개발

- 경험적 데이터 기반 학습하고 예측 수행으로 스스로의 성능을 향상시키는 시스템과 이를 위한 알고리즘을 연구하고 구축하는 기술
- 입력 데이터를 기반으로 예측이나 결정을 이끌어내기 위해 특정한 모델 구축 방식

ML & DL 살펴보기

◆ 머신러닝(Machine Learning)

알고리즘	특징	
지도(supervised)학습	-입력 -출력 -기법	문제 + 결과 데이터 예측 결과 제공 회귀, 분류, 랭킹
강화(reinforcement)학습	-입력 -출력 -기법	문제 데이터만 제공 결과, 결과 평가 후 보상 알고리즘 트레이닝 *상 최대화 / 벌 최소화 방향
비지도(unsupervised)학습	-입력 -출력 -기법	문제 데이터만 제공 분석 결과 제공 군집화, 토픽 모델링, 밀도 추청, 차원 축소

ML & DL 살펴보기

◆ 머신러닝(Machine Learning)

기법 및 모델	특징
분류(Classification)	데이터의 특정 그룹 부여하여 구분
회귀(Regression)	수치 데이터에 사용되는 기법, 연속형 결과 예측
의사결정 나무 (Decision Tree)	트리 구조 형태의 예측 모델을 사용하는 기법
인공 신경망 (Neural Network)	생물의 신경 네트워크 구조와 기능 모방 기법
유 전 자 프 로 그 래 밍 (Genetic Programming)	생물의 진화 알고리즘에 기반한 기법
군집화(Clustering)	관측된 예를 군집하여 부분집합으로 배분 기법
몬테카를로 방법 (Monter Carlo method)	무작위 추출된 난수 통해 확률로 계산하는 기법

ML & DL 살펴보기

◆ 딥러닝(Deep Learning)

- 머신 러닝의 한 분야 => 신경망
- 학습 데이터를 구분하는 층을 많이 만들어 그 정확도를 올리는 방법

- 2016년 알파고에 적용
- 영상 인식, 음성 인식, 자연 언어 처리 등 분야에서 우수한 성능 발휘

ML & DL 살펴보기

◆ 딥러닝(Deep Learning)

알고리즘	특징
퍼셉트론(Perceptron)	-1957년 고안된 알고리즘 -딥러닝 신경망의 기원이되는 알고리즘 -다수의 신호를 입력 받아서 하나의 신호 출력(0 / 1) -뉴런 또는 노드로부터 신호 입력 -뉴런 또는 노드로 신호 출력
인공신경망 (ANN: Artificial Neural Network)	-1980년대부터 활발히 연구 -뇌신경망의 패턴 인식 방식의 통계학적 학습 알고리즘 -선형 맞춤(linear fitting)과 비선형 변환(nonlinear-transformation or activation) 반복 수행하여 최적화
심층신경망 (DNN: Deep Neural Network)	-신경망이 다수의 층의 깊이로써 구성된 개념 -입력층과 출력층 사이 다수의 은닉층으로 구성된 ANN -분류 및 수치 예측을 위한 것

ML & DL 살펴보기

◆ ML & DL용 Python 필수 라이브러리

라이브러리	언어	특징
NumPy	Python	- www.numpy.org - 파이썬에서 수치해석, 통계, 과학 계산용 모듈 - Numeric라이브러리를 이어 SciPy Core로 다시 2005년에 NumPy로 개명 - array자료 생성, 색인, 처리, 연산 수행 - Scipy, Pandas, matplotlib 등 다른 Python 패키지와 함께 사용
SciPy	Python	- www.scipy.org/ - 과학 계산용 함수를 모아 놓은 오픈 소스 파이썬 패키지 - 선형 대수, 함수 최적화, 신호 처리, 특수 수학 함수, 통계 분포 등
Matplotlib	Python	- www.matplotlib.org - Matlab을 기반으로 파이썬으로 확장한 패키지로 그래프 패키지 - 데이터와 분석 결과를 다양한 관점으로 시각화 가능
Pandas	Python	- www.pandas.pydata.org/ - Series와 DataFrame 데이터 구조 제공 및 데이터 처리 패키지

ML & DL 살펴보기

◆ ML 라이브러리 프레임워크

라이브러리	언어	특징
Scikit-lean	Python	- 다양한 지도학습, 비지도학습 알고리즘 구현 - Python기반으로 NumPy, SciPy이용한 고속화 지원
Statsmodels	Python	- 검정 및 추정, 회귀분석, 시계열분석등 다양한 통계분석 - 회귀분석의 경우 patsy 패키지 포함
NLTK	Python	- http://www.nltk.org/ - Natural Language Toolkit 약자 자연어 처리 라이브러리
MLib	Python,Java, Scala	- 아파치 스마크용 머신러닝 라이브러리
Weka	Java	- 자바로 쓰인 데이터마이닝 라이브러리 / GUI 제공
OpenCV	C/C++, Java, Python	- 2000년에 공개되었으며 이미지 데이터 처리 함수 풍부
Pytorch	Python	- 페이스북 인공지능 연구팀에서 개발

ML & DL 살펴보기

◆ DL 라이브러리 프레임워크

라이브러리	언어	특징
Tensorflow	**Python/ C++**	구글 AI 연구팀에서 개발한 오픈소스 라이브러리 머신 러닝 & 딥 러닝 알고리증 충족 풍부한 자연어용 딥러닝 함수, 데이터 모델 라이브러리 제공
Caffe	**Python, Matlab**	• caffe.berkeleyvision.oeg) • 캘리포니아 버클리대 오픈소스로 개발된 딥러닝 프레임워크 • 유연한 구조, 코드 확장성, 고속으로 처리가능, 커뮤니티 활성화 • 미리 학습된 다양한 분류 모델 사용 가능
Torch	**C, Lua**	뉴욕 대학에서 C, LuaJIT로 개발한 라이브러리 스크립트 언어인 루아로 구현 빠른 계산 속도 및 직관적인 API 제공

ML & DL 살펴보기

◆ DL 라이브러리 프레임워크

라이브러리	GPU	특징
CUDA	**NVIDIA**	NVIDIA에서 만든 GPU 플랫폼이자 API 많은 딥러닝 라이브러리에서 CUDA 지원 NVIDIA의 GPU에서 사용
OpenCL	**Intel,** **AMD**	Apple에서 개발하고 Khronos Group에서 관리하는 연산 플랫폼 CPU/GPU 동시동작용 프로그램 개발 라이브러리 Intel/AMD 외 이기종 컴퓨터 표준 제공을 위해 개발 많은 딥러닝 프레임워크에서 OpenCL 미지원

MACHING LEARNING

ML WITH PYTHON

◆ 머신러닝

➢ 개발자가 작성한 프로그램에 따라 동작하는 SW가 아닌

많은 데이터 기반 학습 통해 **답을 출력**할 수 있는 **방법 터득**

[스팸필터 기능]

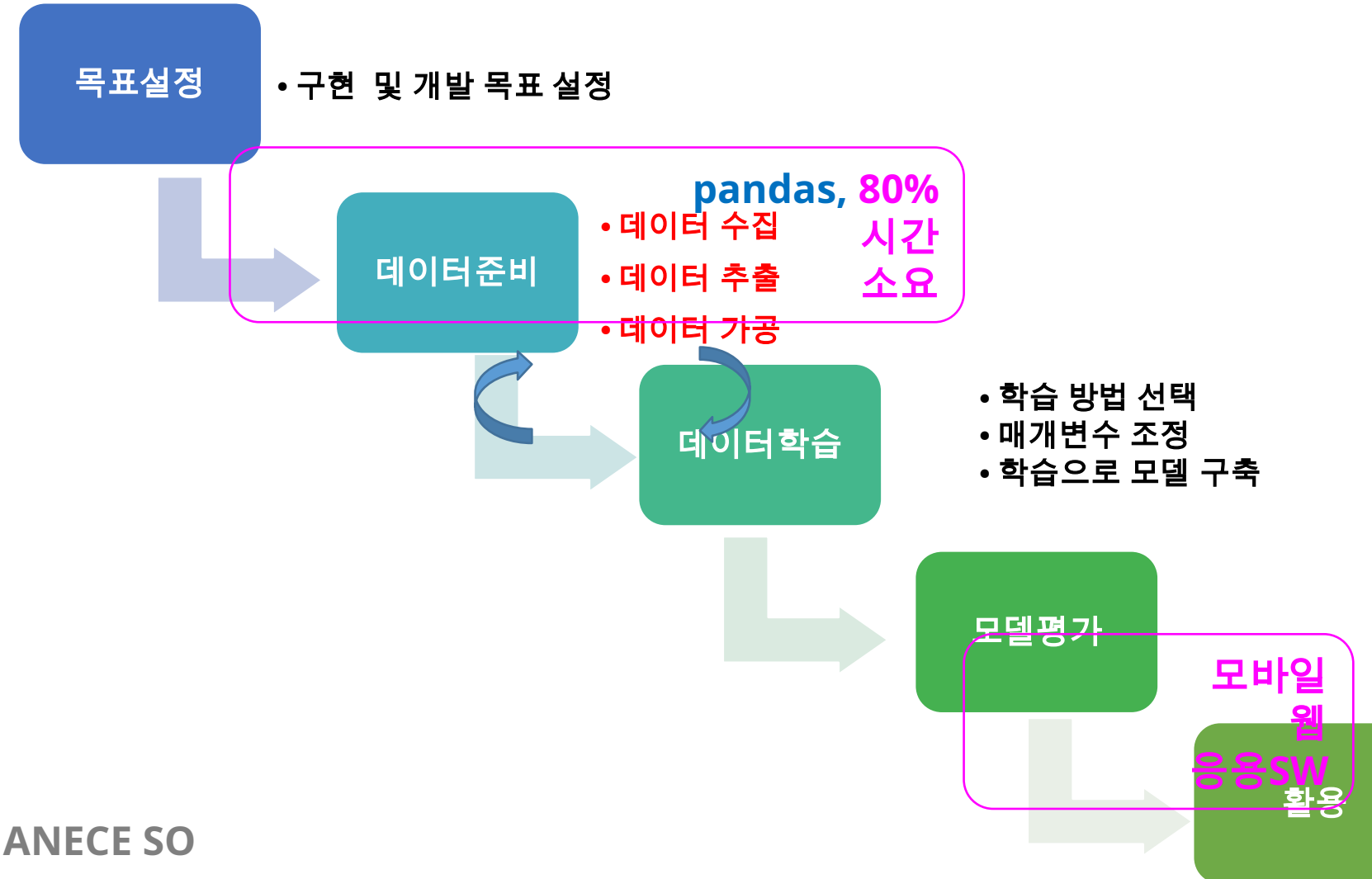
- 스팸의 기준 다양
- 다양한 형태 스팸 데이터

[집값 예상]

- 주거지역 정보
- 집값 정보 데이터

ML WITH PYTHON

◆ 머신러닝 과정



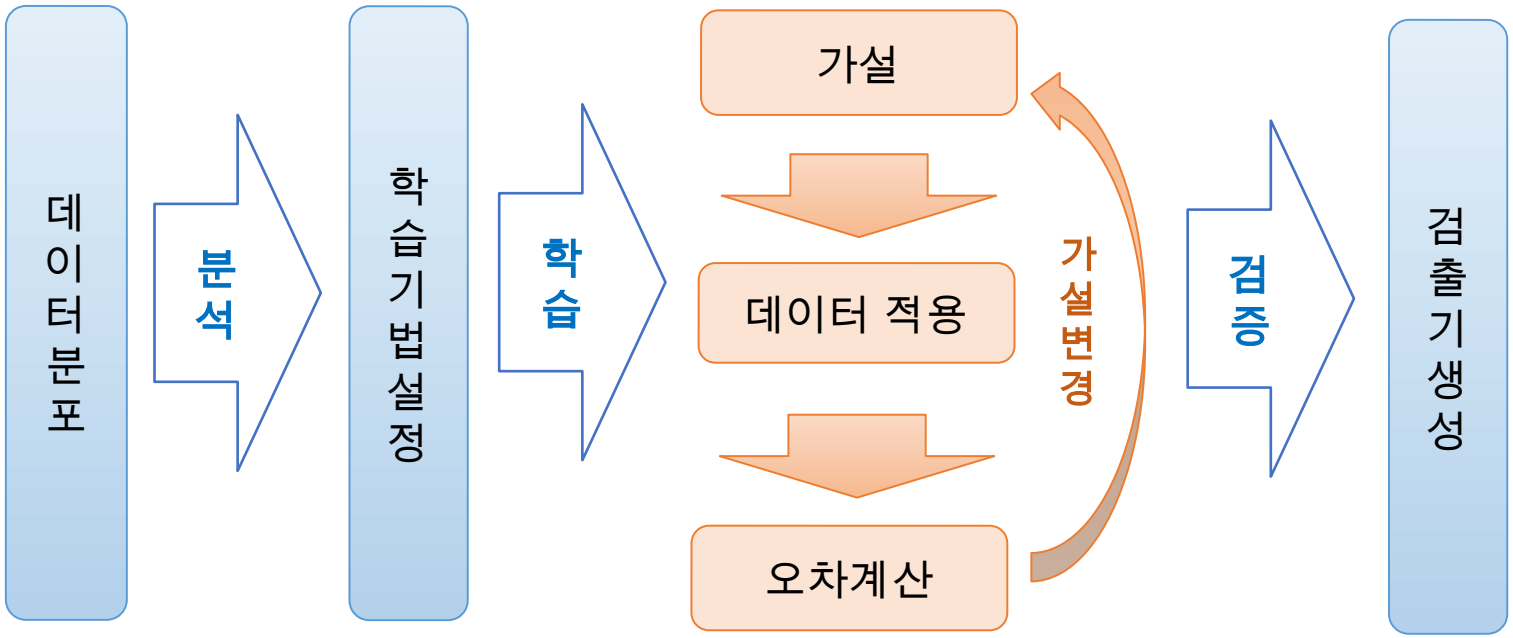
ML WITH PYTHON

◆ 학습 방법

학습 알고리즘	특징	
지도(supervised)학습	-입력	**문제 + 결과 데이터 모두 제공**
	-출력	예측 결과 제공
	-기법	회귀, 분류, 랭킹
강화(reinforcement)학습	-입력	문제 **데이터만 제공**
	-출력	결과, 결과 평가 후 **보상**
	-기법	알고리즘 트레이닝 (상 최대화 / 벌 최소화 방향)
비지도(unsupervised)학습	-입력	문제 **데이터만 제공**
	-출력	**분석 결과** 제공
	-기법	군집화, 토픽 모델링, 밀도 추청, 차원 축소

ML WITH PYTHON

◆ 머신러닝 학습 단계



머신러닝 학습 준비

➢ 데이터셋(Dataset)

정학한 규칙/패턴을 찾기 위해 필요한 데이터 집합

종류

규칙/패턴을 찾기 위한 데이터 => **학습용 데이터셋**

규칙/패턴을 찾는 과정 중 검사 데이터 => **검증용 데이터셋**

규칙/패턴을 평가 위한 데이터 => **테스트용 데이터셋**

머신러닝 학습 준비

데이터셋(Dataset)

분류

규칙/패턴 찾기 위해 **입력되는 데이터**

➔ **입력/문제/피쳐(특성)**

입력 데이터에 해당하는 **정답 데이터**

➔ **타겟/정답/클래스/ 라벨**

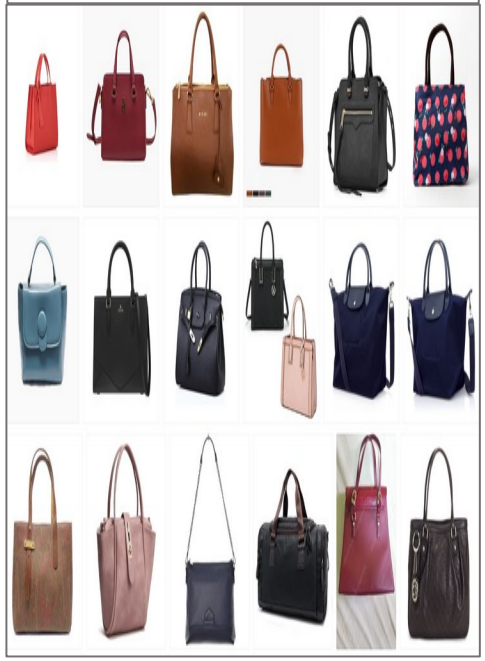
지도 학습 시 제공

ML WITH PYTHON

◆ 머신러닝 학습 준비

➢ 데이터셋(Dataset) => 데이터+라벨

데이터셋

핸드백(Label)	고양이 (Label)
	

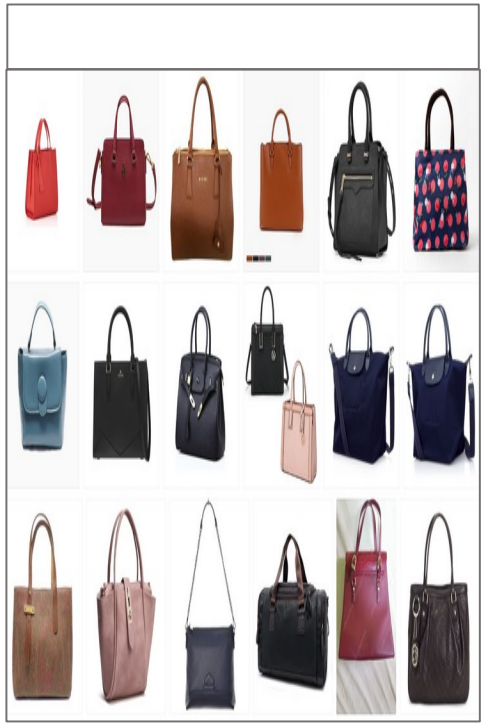
평수(데이터)	관리비(라벨)
18	90,000
24	140,000
29	190,000
32	210,000
39	250,000
42	280,000
49	330,000
54	370,000

ML WITH PYTHON

◆ 머신러닝 학습 준비

➢ 데이터셋(Dataset) => 데이터로만 구성

데이터셋



평수(데이터)
18
24
29
32
39
42
49
54

ML WITH PYTHON

◆ 지도/교사학습(Supervised Learning)

가장 널리 사용되는 방법

구성 : **입력(문제) 데이터 + 타겟(정답) 데이터**

일반적으로 우리가 학교에서 배우는 방식

ML WITH PYTHON

◆ 지도/교사학습(Supervised Learning)

➢ 학습 종류

회귀(Regression)		예) 시간 : 점수와 관계 -> 공부 시간에 따른 점수 예측
분류 (Classification)	이진분류 (Binary Classification)	예) 시간 : 합격여부 관계 -> 공부 시간에 따른 합격 / 불합격
	다중 분류 (Multi lable Classification)	예) 시간 : 학점 관계 -> 공부 시간에 따른 학점 (A ~ F) 예측

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ 공부시간과 점수

공부시간	2	4	6	8	9
점 수	24	44	64	84	94

5시간 공부하면 몇점?

공부시간과 점수 사이의 일정한 규칙(패턴) 찾기

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ 공부시간과 점수

공부시간	2	4	6	8	9
점 수	24	44	64	84	94

5시간 공부하면 몇점?

시간 * X = 점수

2 ---- 24 2 * 10 + 4 => 24

4 ---- 44 4 * 10 + 4 => 24

6 ---- 64 6 * 10 + 4 => 64

8 ---- 84 8 * 10 + 4 =>

9 ---- 94

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ 공부시간과 점수

공부시간	2	4	6	8	9
점 수	24	44	64	84	94

5시간 공부하면 몇점?

5 * 10 + 4 = 54

점

규칙(패턴)

시간 * 10 + 4 = 점수

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ 공부시간과 합격여부

공부시간	2	4	4.5	5.5	6
점 수	불합격	불합격	합불격	합격	합격

5시간 공부하면 합격? 불합격?

2021년 고3수험생의 1일 평균 공부시간과 합격/불합격 데이터 수집 후 평가

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ 공부시간과 학점

공부시간	2	4	4.5	5.5	6
점 수	D	C+	C+	B+	A+

5시간 공부하면 학점은?

A+, A, B+, B, …. , F

10가지 학점 중 하나의 결과

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ Regression(회귀)

데이터를 기반으로 새로운 데이터에 대한 **결과 예측하는 기술**

데이터와 데이터 분석 기반 가설 필요

해당 가설에 **오차를 줄여 예측 적중률을 높이는 것이 목표**

◆ **구현 기법** ◆

- Linear Regression
- Logistic Regression
- Multi-variable Linear Regrssion & Matrix
- 그 외 다양

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ Classification(분류)

데이터 분석 결과 결과(라벨)로 데이터를 나누는 경우

- 두가지 결과로 나누는 경우 => 이진 분류
- 세가지 이상 결과로 나누는 경우 => 다중 분류

ML WITH PYTHON

◆ 지도학습(Supervised Learning)

➢ Classification(분류)

◆ 구현 기법 ◆

- KNN (K-nearest neighbor)
- Native Bayes
- SVM(Support Vector Machine)
- Decision Tree
- Random Forest
- 그 외 다양 .....

Scikit-learn 살펴보기

ML WITH SCIKIT-LEARN

◆ Scikit-learn

머신러닝 알고리즘, 전처리, 검증 등 기능 제공 **라이브러리**

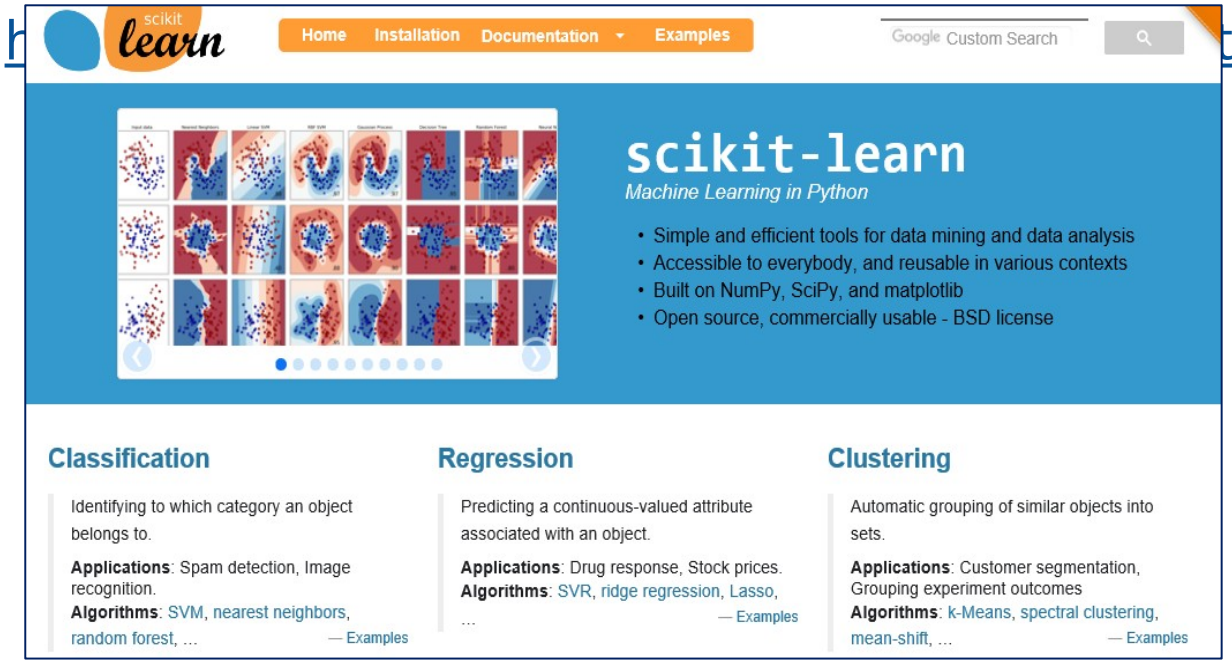
Numpy기반 속도 최적화

바로 테스트 가능한 **다양한 샘플 데이터 제공**

많이 사용되는 다른 라이브러리와 호환 가능

BSD 라이선스로 **무·료상업적 사용 가능**

ML WITH SCIKIT-LEARN

- **Scikit-learn**
  - https://scikit-learn.org
  - API : h 

tml

ML WITH SCIKIT-LEARN

◆ Scikit-learn

분류	모듈명	내장 기능
예제 데이터	sklearn.datasets	연습용 데이터셋
피처 처리	sklearn.preprocessing	전처리 관련 기법 (원핫 인코딩, 정규화, 스케일링 등)
	sklearn.feature_selection	모델에 중요한 영향을 미치는 피처를 탐색 및 선택하는 기법
	sklearn.feature_extraction	원시 데이터로부터 피처를 추출하는 기능 이미지에 대한 피처 추출은 하위 모듈 image에, 텍스트 데이터의 피처 추출은 하위 모듈 text에 지원 API가 있음.
차원 축소	sklearn.decomposition	차원 축소 관련 알고리즘 계열 (PCA, NMF, Truncated SVD 등)

ML WITH SCIKIT-LEARN

◆ Scikit-learn

분류	모듈명	내장 기능
기계학습 알고리즘	sklearn.ensemble	앙상블 알고리즘 계열 (랜덤 포레스트, 에이다 부스트, 배깅 등)
	sklearn.linear_model	선형 알고리즘 계열 (선형회귀, 로지스틱 회귀, SGD 등)
	sklearn.naive_bayes	나이브 베이즈 알고리즘 계열 (베르누이 NB, 가우시안 NB, 다항분포 NB 등)
	sklearn.neighbors	최근접 이웃 알고리즘 계열 (K-NN 등)
	sklearn.svm	Support Vector Machine 계열 알고리즘
	sklearn.tree	의사 결정 나무 계열 알고리즘
	sklearn.cluster	비지도 학습(클러스터링) 알고리즘 (KMeans, , DBSCAN 등)

ML WITH SCIKIT-LEARN

◆ Scikit-learn

분류	모듈명	내장 기능
검증, 하이퍼 파라미터 튜닝, 데이터 분리	sklearn.model_selection	검증, 하이퍼 파라미터 튜닝, 데이터 분리 등 (cross_validate, GridSearchCV, train_test _split, learning_curve 등)
모델 평가	sklearn.metrics	모델의 성능을 측정 및 평가하는 기법 (accuray, precision, recall, ROC curve 등)
유틸리티	sklearn.pipeline	피처 처리 등의 변환과 기계학습 알고리즘 등을 연쇄적으로 실행하는 기능

ML WITH SCIKIT-LEARN

◆ Scikit-learn.metrics 모듈

- scikit-learn 에서 **모델 및 성능평가에 사용되는 모듈**
- https://scikit-learn.org/stable/modules/classes.html

sklearn.metrics: Metrics

See the Metrics and scoring: quantifying the quality of predictions section and the Pairwise metrics, Affinities and Kernels section of the user guide for further details.

The `sklearn.metrics` module includes score functions, performance metrics and pairwise metrics and distance computations.

ML WITH SCIKIT-LEARN

◆ Scikit-learn.metrics 모듈

함수	역할
accuracy_score(y_true, y_pred)	정확도 측정
confusion_matrix(y_true, y_pred)	Classification에서 실제와 결과를 메트릭스로 표현 성능지표(confusion matrix) 출력, 정확도 평가
precision_score(y_true, y_pred)	정밀도 측정
recall_score(y_true, y_pred)	재현율 즉정
f1_score(y_true, y_pred)	정빌도와 재현율의 평균
classfication_report(y_true, y_pred)	분류 학습 후 결과 보고서를 출력 precision, recall, f1-score, support 항목 출력

REGRESSION(회귀)

REGRESSION 회귀

◆ REGRESSION

지도학습의 학습 방법 중 하나

선형과 비선형

입력데이터에 대한 **타겟 값(수치)를 예측**하는 것

예) 키에 따른 몸무게, 피자 도착 시간, 경계 성장률 등등

REGRESSION 회귀

◆ REGRESSION

『 구현 단계 』

1. 데이터를 기반으로 **데이터의 관계 분석**
2. 데이터 분석 결과 **수학적 계산 가능 공식 가설** 설정
3. 데이터의 가설 공식에 적용 학습
4. 공식에 사용되는 계수 찾기
5. 테스트 데이터 활용 성능 확인

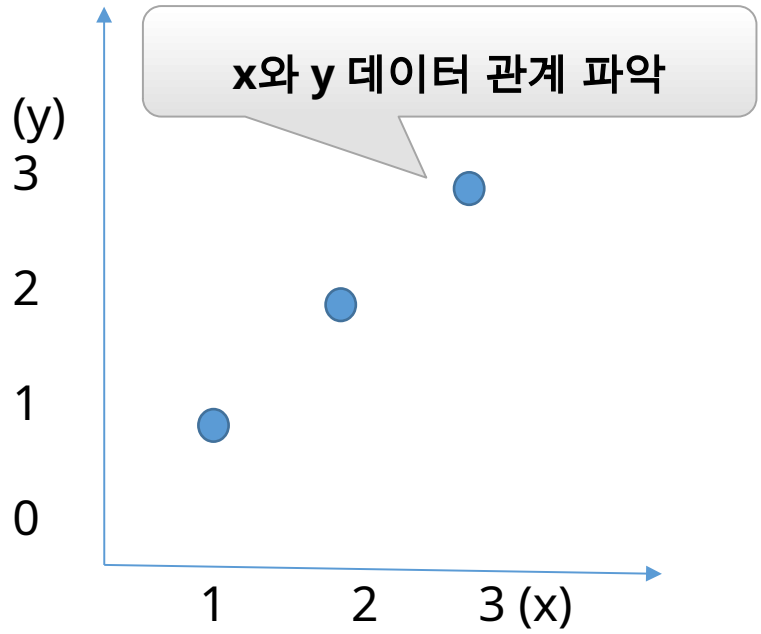
REGRESSION 회귀

◆ REGRESSION

x(hour)	y(score)
1	1
2	2
3	3

데이터 & 타겟

분석

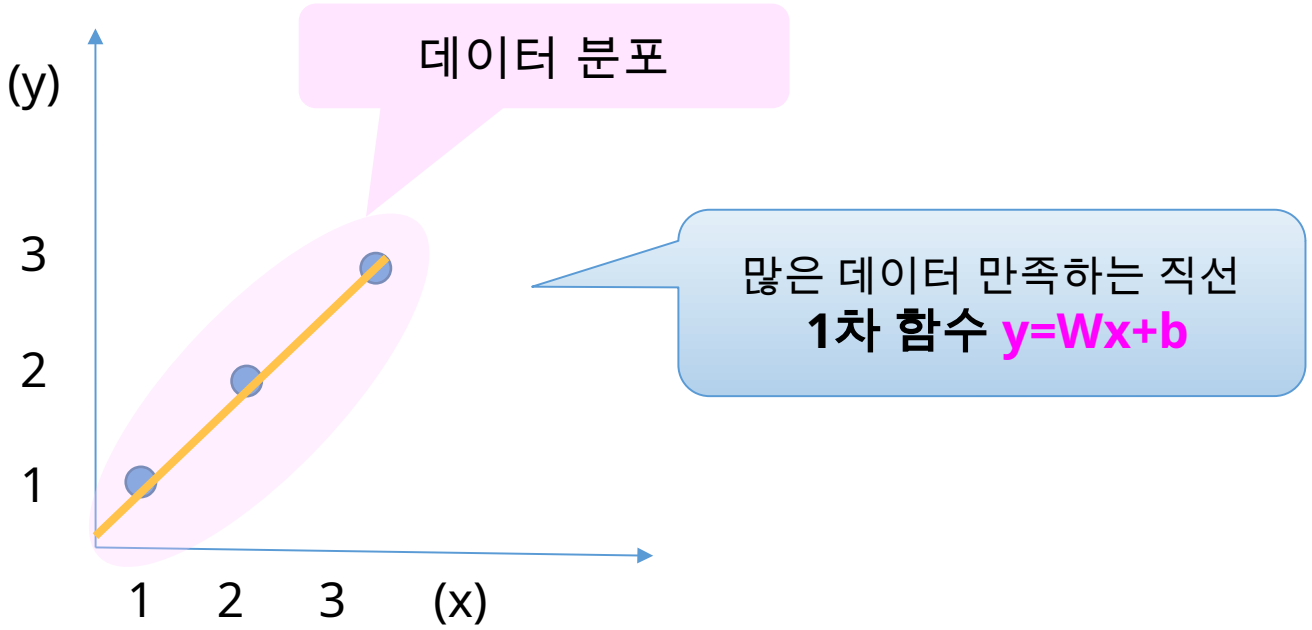


REGRESSION 회귀

◆ 선형 REGRESSION

➢ Linear Regression (선형 회귀)

- **선형 형태** 데이터 분포

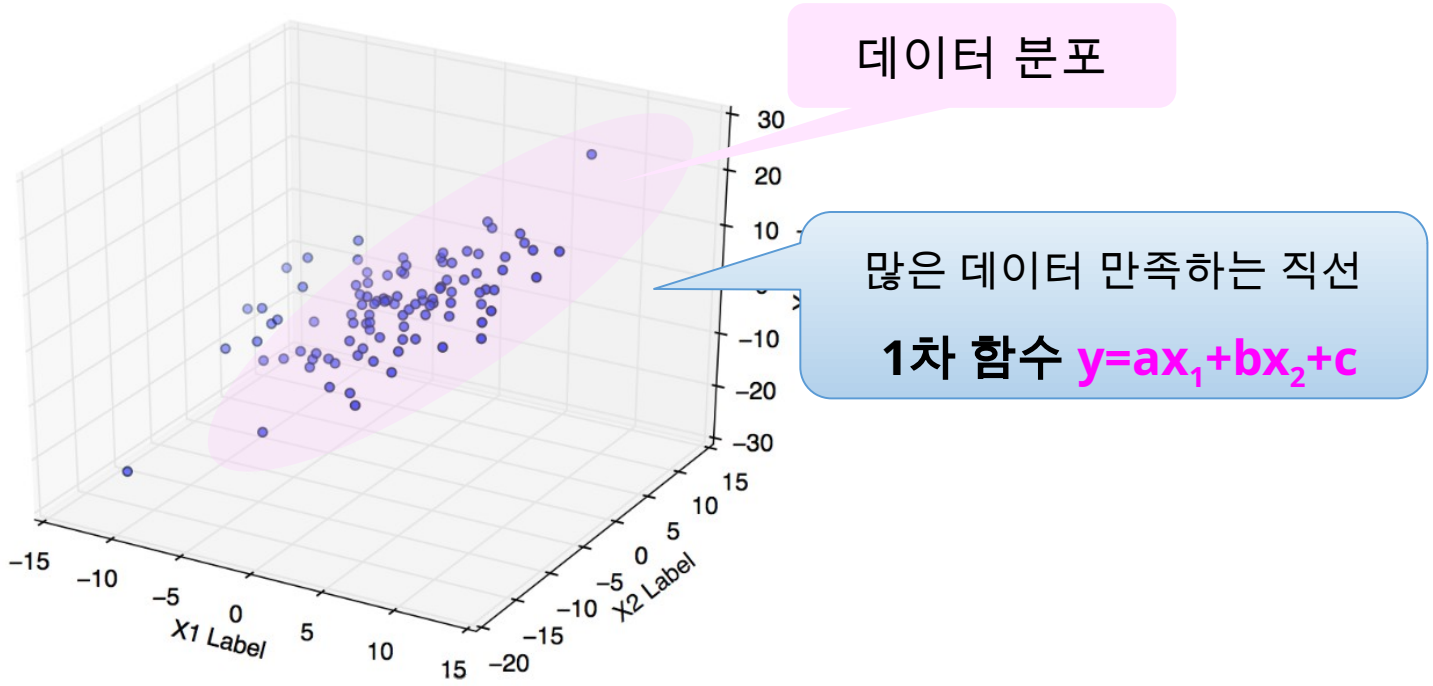


REGRESSION 회귀

◆ 선형 REGRESSION

➢ Multiple Regression (다중 회귀)

- **선형 형태** 데이터 분포

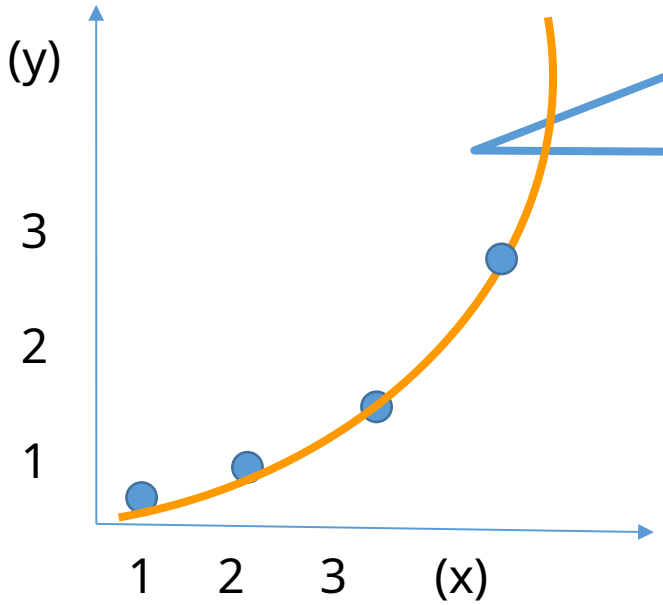


REGRESSION 회귀

◆ 비선형 REGRESSION

➢ Polynomial Regression (다항 회귀)

- **곡선 형태** 데이터 분포



많은 데이터 만족하는 곡선

2차 함수 $y=a_2x^2+a_1x^1+c$

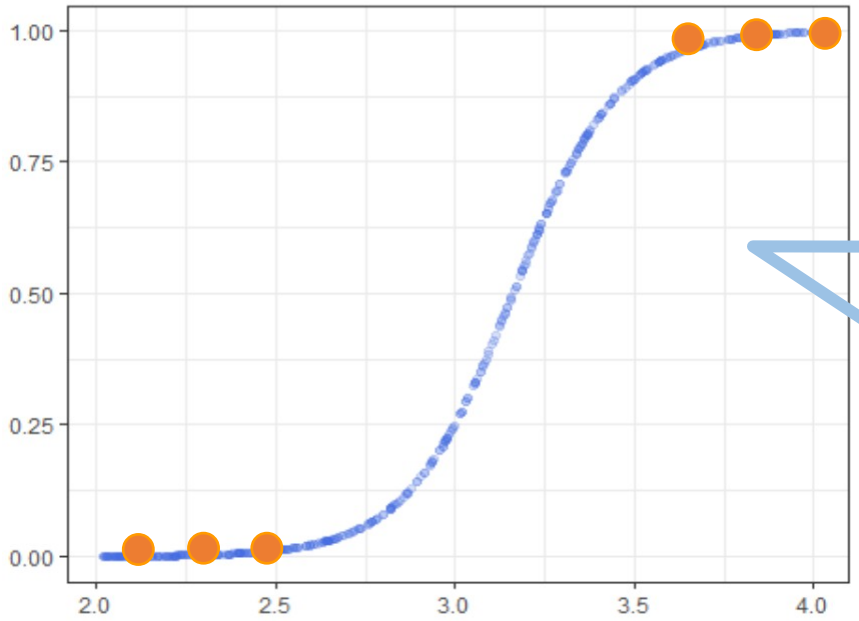
N차 함수 $y=a_dx^d+..+a_2x^2+a_1x^1+b$

REGRESSION 회귀

◆ 비선형 REGRESSION

➢ Logistic Regression (로지스틱 회귀) ← 신경망DL

- **S자 형태** 데이터 분포



많은 데이터 만족하는

S형 곡선 $\frac{e^{a+bX}}{1+e^{a+bX}}$

REGRESSION 회귀

- **REGRESSION**

직선 $y = ax + b$

$y = ax_1 + a_2x_2 + b$

곡선 $y = a_dx^d + \dots + a_2x^2 + a_1x^1 + b$

y = 타겟
x = 특성

모델 학습 통해 최적의 a, b 찾기

a, b는 모델 파라미터

REGRESSION 회귀

◆ REGRESSION

➢ 모델 성능 평가

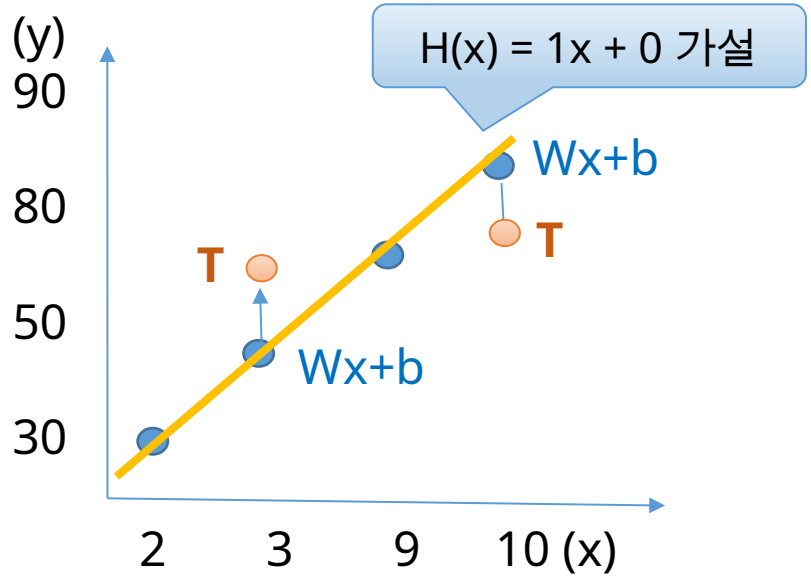
- 실제값과 모델 예측값 과의 차이 ➔ 오차
- 오차 = 0 >>> 모델 100% : 불가능
- 오차 최소화되도록 모델 구현

REGRESSION 회귀

◆ REGRESSION

➢ 오차 함수 (Cost/Loss Fuction)

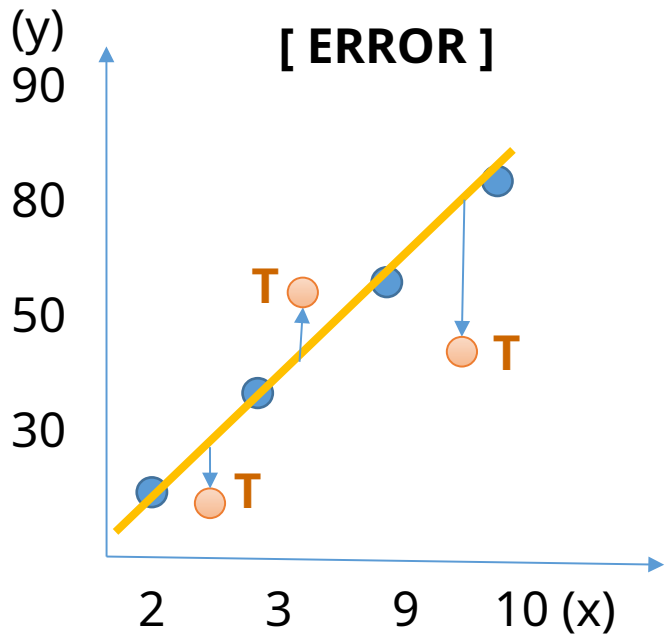
오차
T – y = T – (Wx+b)

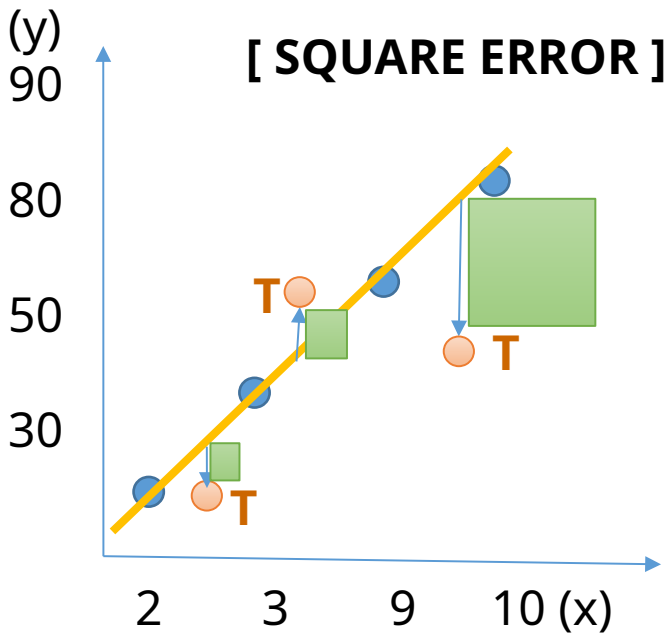


REGRESSION 회귀

◆ REGRESSION

➢ 오차 함수 (Cost/Loss Fuction)





REGRESSION 회귀

◆ REGRESSION

➢ 오차 함수 (Cost/Loss Fuction)

***단순오차함수 (RSS:residual sum of squares)**

- 예측값과 타겟값과의 차이를 제곱해서 모두 더 한 함수

***평균제곱오차 함수 (MSE:mean squared error)**

- 예측값과 타겟값과의 차이를 제곱해서 모두 더 한 후 평균 취하는 함수
- **이상치 값이 존재**하면 수치가 많이 커짐 **(민감함)**

REGRESSION 회귀

◆ REGRESSION

➢ 오차 함수 (Cost/Loss Fuction)

***평균절대값오차함수 (MAE:mean absoulte error)**

- 예측값과 타겟값과의 차이를 절대값을 모두 더 한 후 평균 취하는 함수
- 변동성이 큰 지표와 작은 지표 함께 비교에 좋음

***평균절대값오차함수 (RMAE:Root Mean Squared Error)**

- MSE에 루트를 취한 값
- 에러에 따른 손실이 기하 급수적으로 올라가는 상황에서 쓰기 적합
- **MAE와 함께 가장 일반적으로 많이 쓰이는 회귀모델 성능분석지표**

REGRESSION 회귀

◆ REGRESSION

➢ 오차 함수 (Cost/Loss Fuction)

***R2 Score (Coefficient of Determination)**

- 주어진 데이터들에 기반해 만들어진 모델이 주어진 데이터들을 얼마나 잘 '설명하는가'에 대한 수치 ➔ **결정계수**

- 예측 모델과 실제 모델이 얼마나 강한 상관관계(Correlated) 가지는가

- 총 제곱합 : 관측값과 평균의 차이 기반

- 회귀제곱합 : 회귀예측값과 평균의 차이 기반

- 1 - ((y_true - y_pred) 2).sum() /**

((y_true - y_true.mean()) ** 2).sum()

- 값이 1에 가까울 수록 성능이 좋음

REGRESSION 회귀

◆ REGRESSION

[a, b 업데이트 최적화 기법]

- **손실 함수의 값이 최소화**가 될 수 있도록 a, b 값 계속 업데이트 필요
- **처음 랜덤한 값으로 초기화 후 적정한 값으로 계속 업데이트 진행**
- 최적의 값 찾아내기

- 대표적인 최적화 기법
  - 경사하강법 (Gradient Descent)
  - 아담스최적화 기법
  - RMShop 기법 등등....

REGRESSION 회귀

◆ REGRESSION

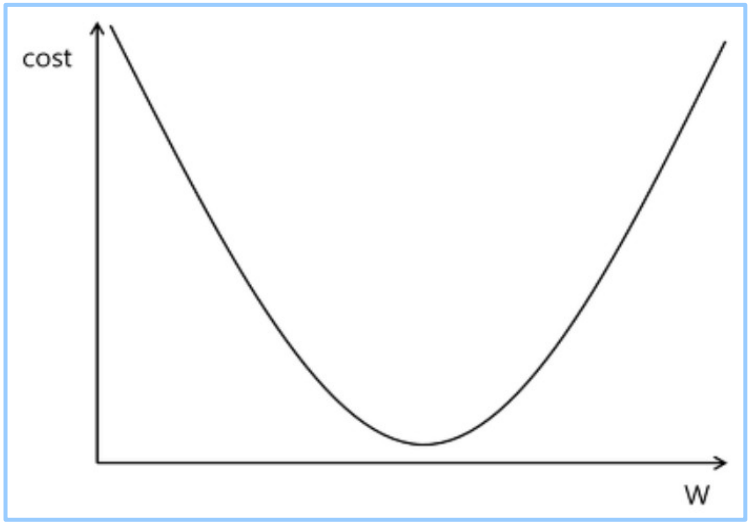
[경사하강법 (Gradient Descent)]

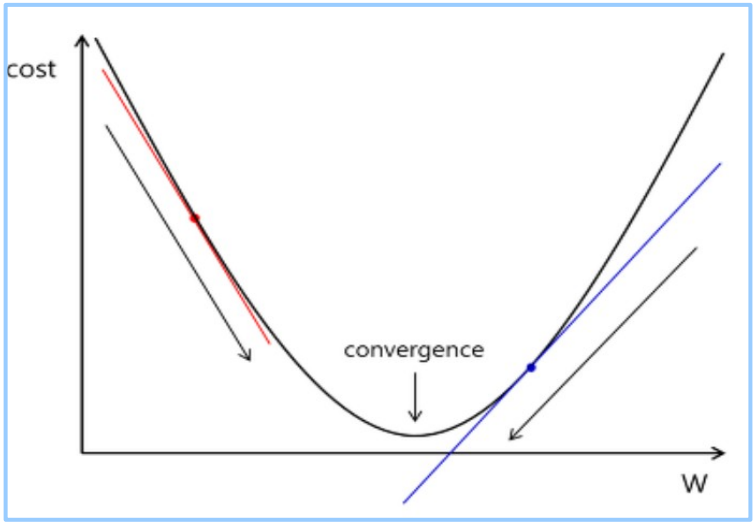
정답이 주어진 데이터가 있을 경우 최적의 선형회귀모델 직선을 긋고

그 직선의 MSE 즉 cost functin을 최소로 만드는 직선을 찾아야 함

cost를 최소로 하는 직선을 구하는 과정 => 학습

학습에 사용되는 알고리즘이 바로 '경사하강법'



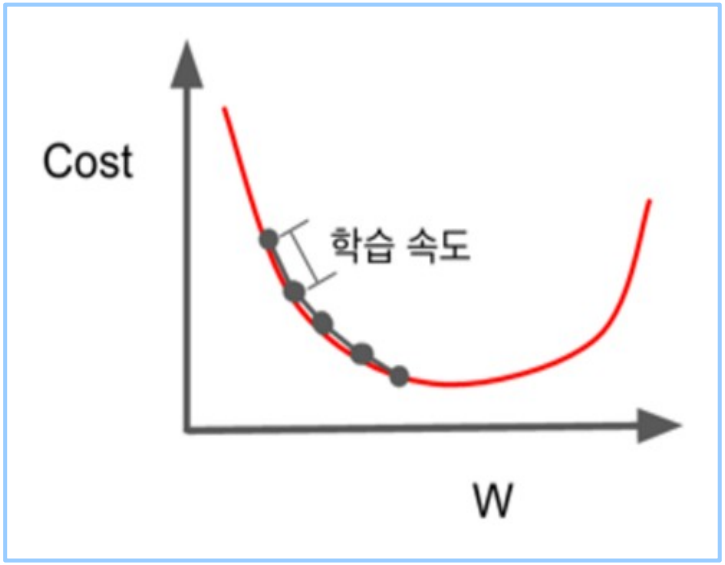


REGRESSION 회귀

◆ REGRESSION

[학습속도 : Learning Rate]

- 경사 하강법에서 **학습 단계별로 움직이는 학습 속도 정의**
- W값을 조정해 가면서 Cost 값이 최소가 되는 값을 찾기 위한 것
- W값이 다음 W값이 되는 속도



오버슈팅(Over shooting)

학습속도가 큰 경우 발생
최소값으로 내려가지 않고 반대편으로 넘어가
무한대

스몰 러닝 레이트(Small Learning Rate)

학습속도가 매우 작은 경우 발생
예) 0.0001
최소값 가기전에 학습 종료

비선형 REGRESSION

비선형 REGRESSION

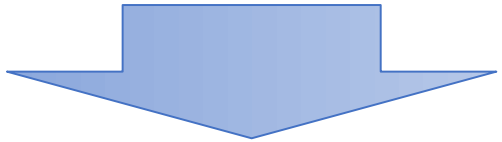
◆ LOGISTIC REGRESSION

- **입력 데이터와 타겟값의 관계**를 **구체적인 함수**로 나타내어 향후 예측 모델에 사용하는 회귀 분석과 목표 동일
- Linear Regression과 달리 **타켓값**이 **범주형 데이터**
- 입력 데이터에 대한 결과가 **특정 분류로 나뉘어 짐**
- 기본 규제 : **L2 적용**
- 비선형적 데이터에도 적용 가능 ➔ 커널 로지스틱 함수

비선형 REGRESSION

◆ LOGISTIC REGRESSION

- 종류
  - **이진 분류** : 2가지(0과 1) 분류
  - **다중 분류** : 3가지 이상으로 분류



각 **범주로 분류될 확률**로 **결과 반환**

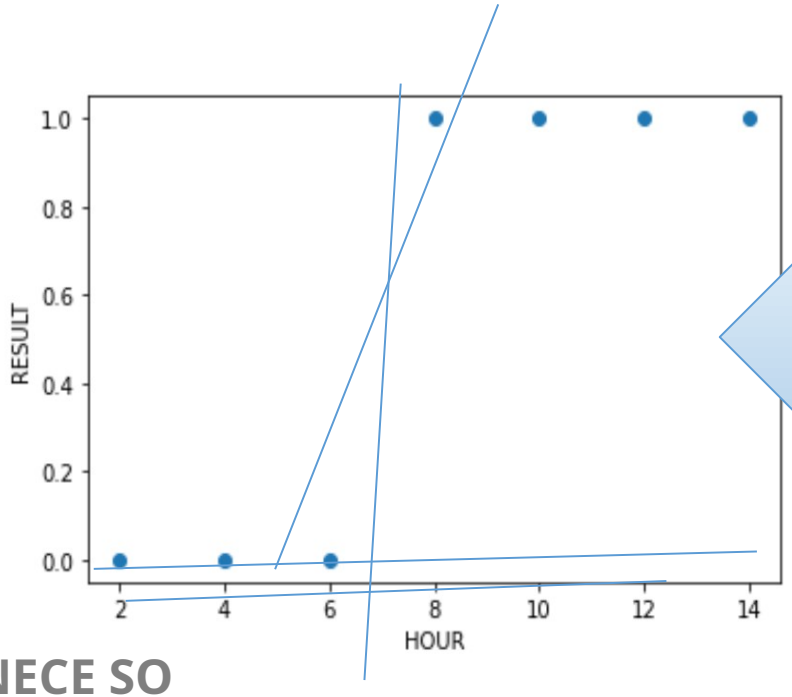
모든 범주의 **분류 확률 합 = 1**

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 예시 – 공부 시간과 성적

공부시간	2시간	4시간	6시간	8시간	10시간	12시간	14시간
성 적	40	48	54	68	72	89	91



y=ax+b

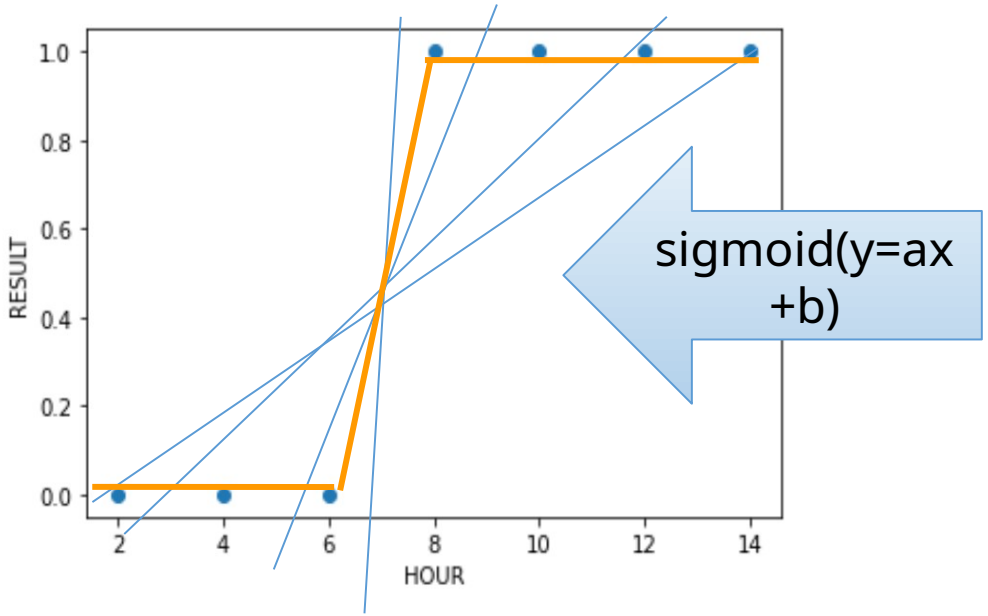
40=2a+b
48=4a+b
54=6a+b
68=8a+b
72=10a+b
89=12a+b
91=14a+b

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 예시 : 공부 시간과 합격 여부

공부시간	2시간	4시간	6시간	8시간	10시간	12시간	14시간
합격 여부	불합격	불합격	불합격	합격	합격	합격	합격



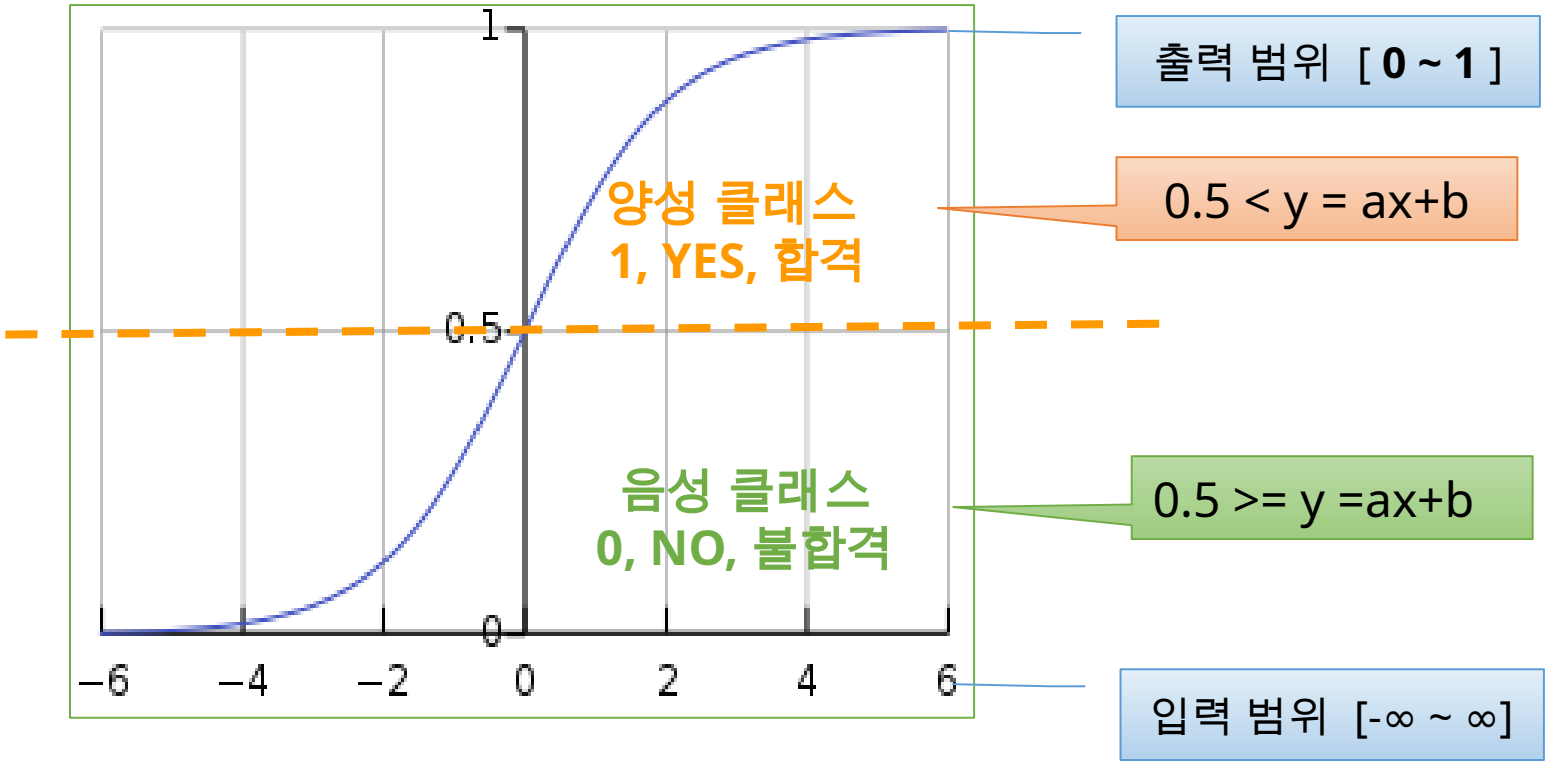
0=2a+b
0=4a+b
0=6a+b
1=8a+b
1=10a+b
1=12a+b
1=14a+b

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 시그모이드(Sigmoid) 함수

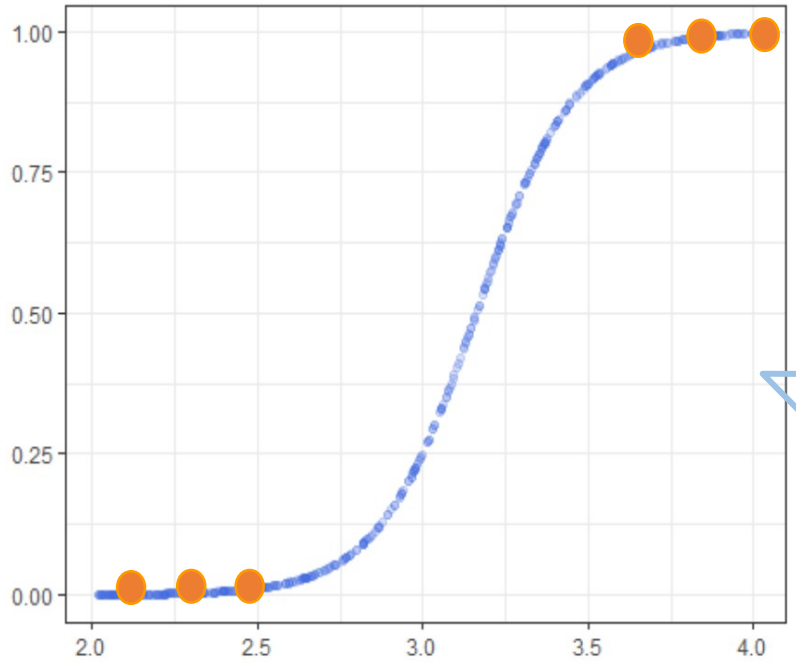
: S자형 곡선 또는 시그모이드 곡선을 갖는 수학 함수



비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ **S자 형태** 데이터 분포



많은 데이터 만족 S형 곡선

z=ax+b

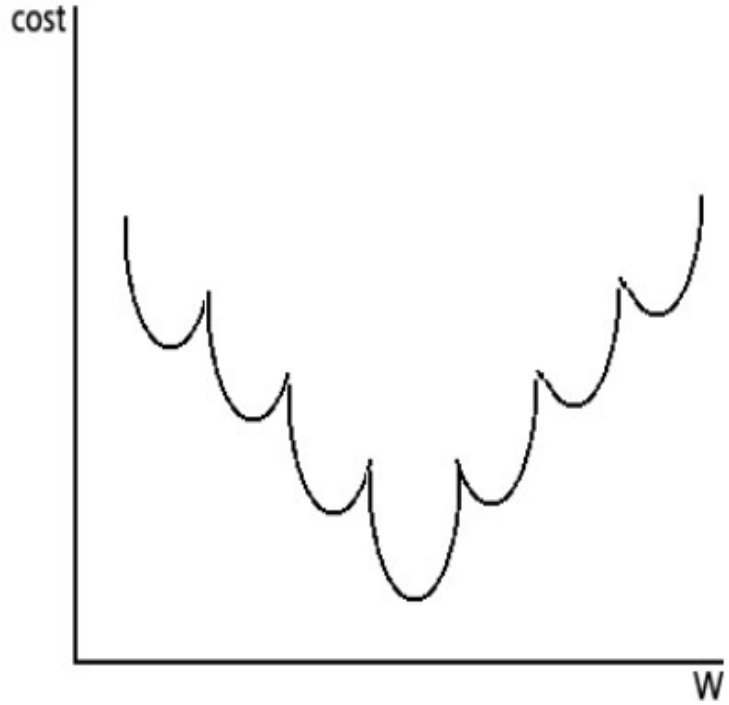
Sigmoid(z)

$$p = \frac{1}{1+e^{-z}}$$

p : 확률값

비선형 REGRESSION

- **LOGISTIC REGRESSION**
  - **선형회귀 적용 손실함수**



선형회귀 cost function을
로지스틱 회귀 가설 적용 하면
W의 값에 대한 cost function

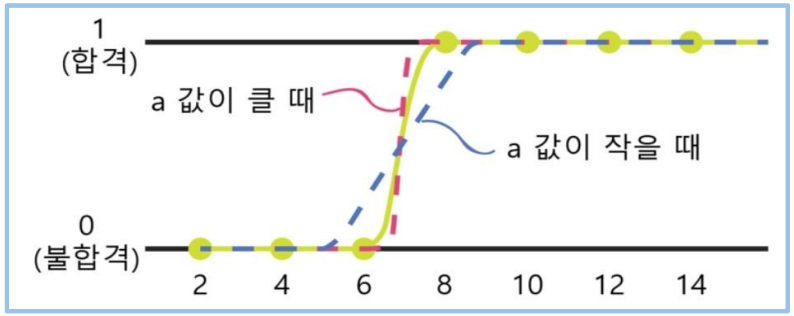
비선형 REGRESSION

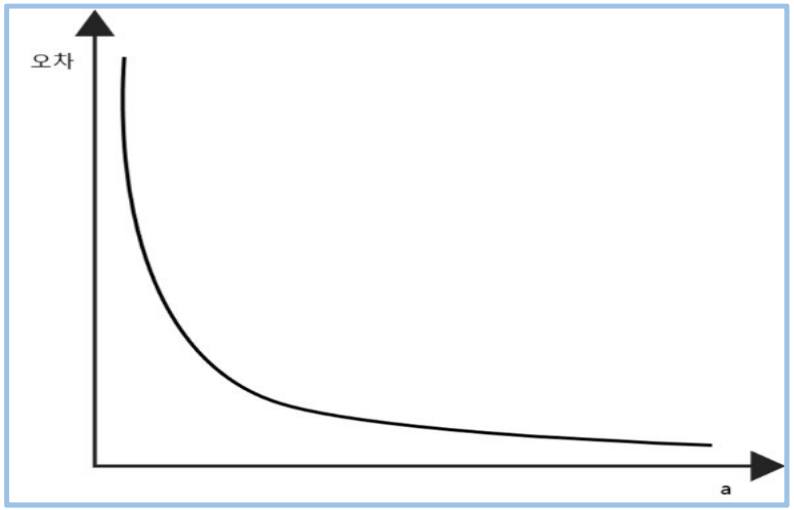
- **LOGISTIC REGRESSION**
  - 선형회귀 적용 손실함수

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 실제값과 오차 관계





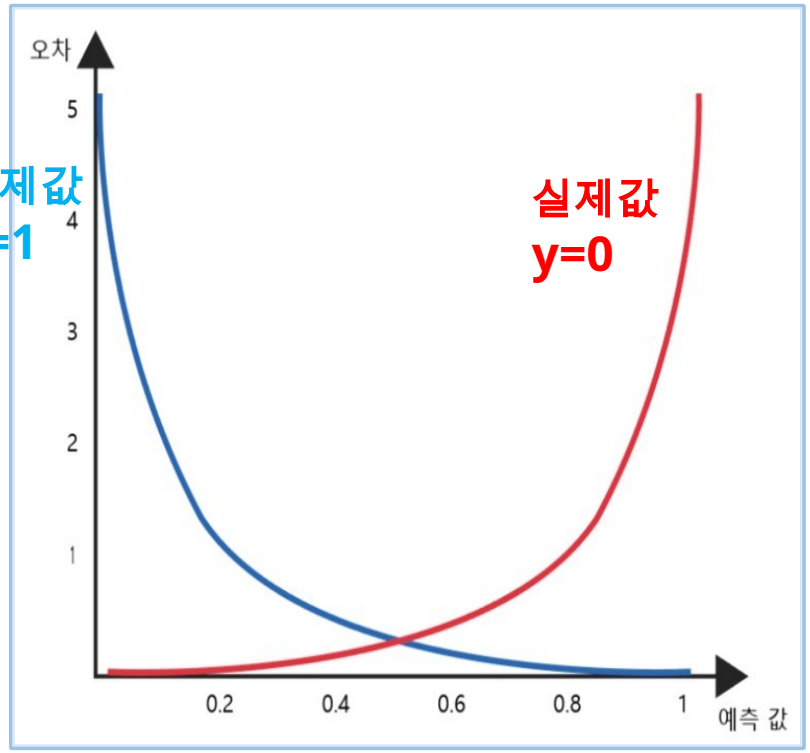
➔ a가 클수로 경사 커짐

➔ a가 작을 수록 경사 작아짐
➔ 수평이 됨

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 실제값과 오차 관계



파랑색 선 ➔ **실제 값 1**

: -ylog(H(x))

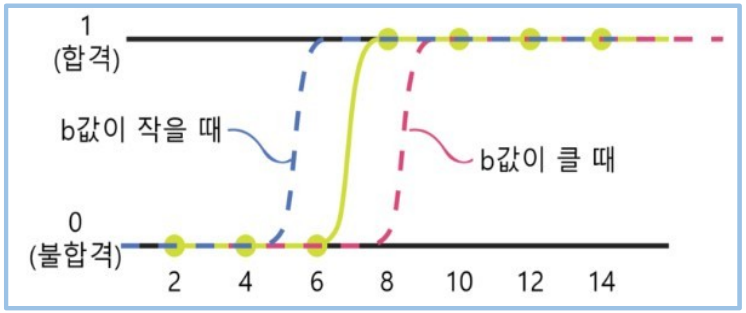
빨강색 선 ➔ **실제 값 0**

: (1-y)log(1-H(x))

비선형 REGRESSION

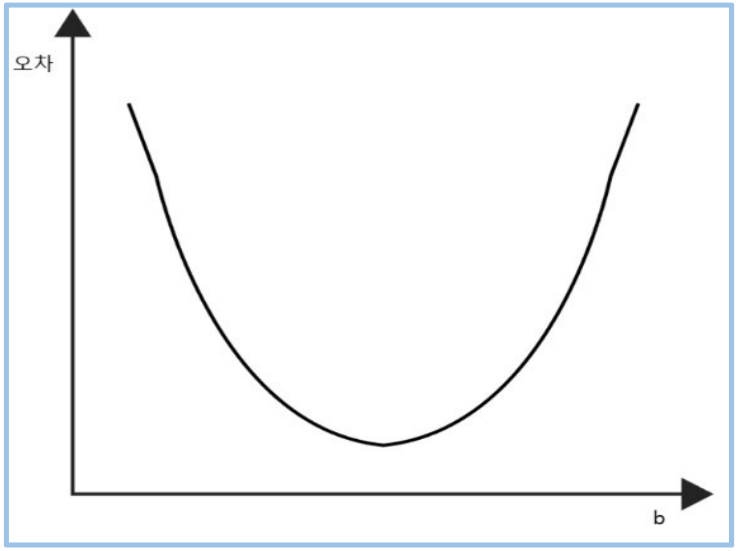
◆ LOGISTIC REGRESSION

➢ 절편과 오차 관계



➔ b가 클수로 오른쪽 이동

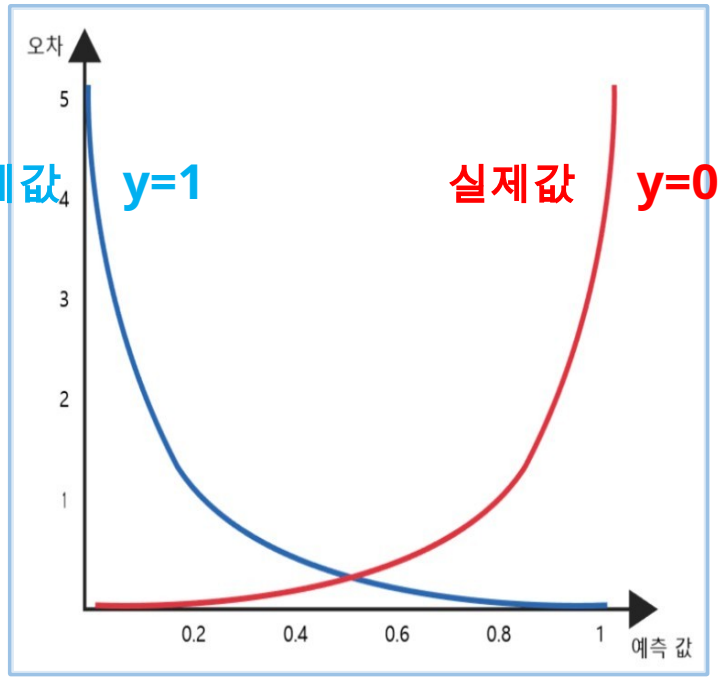
➔ b가 작을 수록 왼쪽 이동



비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 손실함수 ➔ 로그 함수



실제 값이 1인 경우 Log

실제 값이 0인 경우 Log

$$cost(H(x), y) = -y \log(H(x)) - (1-y) \log(1-H(x))$$

손실(오차) =

-평균(ylog(H(x)) + (1-y) log(1-H(x))

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 시그모이드(Sigmoid) 함수

numpy 모듈

numpy.**exp(1개)** → **확률값**

scipy 모듈

scipy.special.**expit(1개)** → **확률값**

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 소프트맥스(Softmax) 함수

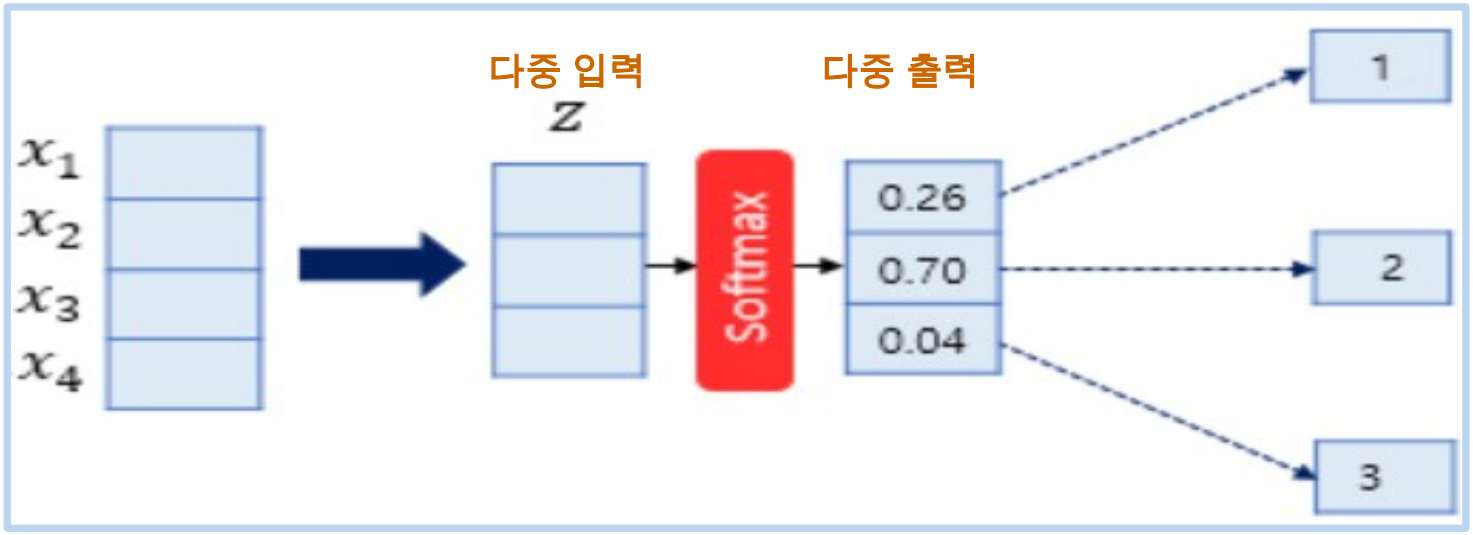
- 다중 분류 시에 각 라벨(클래스/타겟)에 대한 **확률 추정 함수**
- 시그모드 함수에서 유래
  - 입력값 : 여러개
  - 출력값 : 0~1 사이의 실수 ➔ '확률'로 해석 가능
  - 출력 총합 : 1

$$y_k = \frac{e^{a_k}}{\sum_{i=1}^{n} e^{a_i}}$$

비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 소프트맥스(Softmax) 함수



비선형 REGRESSION

◆ LOGISTIC REGRESSION

➢ 소프트맥스(Softmax) 함수

scipy 모듈

- scipy.special.**softmax(여러개) ➔ 확률값**

비선형 REGRESSION

◆ LOGISTIC REGRESSION

◺ Scikit-Learn Lib 사용 - **학습객체**

from sklearn.linear_model **import** **LogisticRegression**

(**penalty = L2**	**규제 사용 기준 지정 (L2 = MSE+가중치 제곱합)**
dual	이중 또는 초기 공식
tol	정밀도
C=1.0	**규제 강도 (Cost Funciton) , 큰값(약) - 작은값(강)**
fit_intercept=True	절편 존재 여부 설정
intercept_scaling=1	정규화 효과 정도
class_weight =1	클래스 가중치
random_state	난수 seed 설정

비선형 REGRESSION

◆ LOGISTIC REGRESSION

◺ Scikit-Learn Lib 사용 - 학습객체

```
from sklearn.linear_model import LogisticRegression
(
solver='lbfgs'      # 최적화 문제 사용 알고리즘
max_iter            # 계산 작업 수
multi_class         # 다중 분류 시 (ovr, multinomial, auto)로 설정
verbose             # 동작 과정에 대한 출력 메시지
warm_start          # 이전 모델을 초기화로 적합하게 사용할 것인지 여부
n_jobs              # 병렬 처리 할 때 사용되는 CPU 코어 수
l1_ratio            # L1 규제의 비율(Elastic-Net에만 사용) )
```

ML OPTIMIZATION & MODEL

ML OPTIMIZATION

◆ 최적화

목적함수(Objective Function)를 최대한, 혹은 최소화하는 파라미터 조합을 찾는 과정

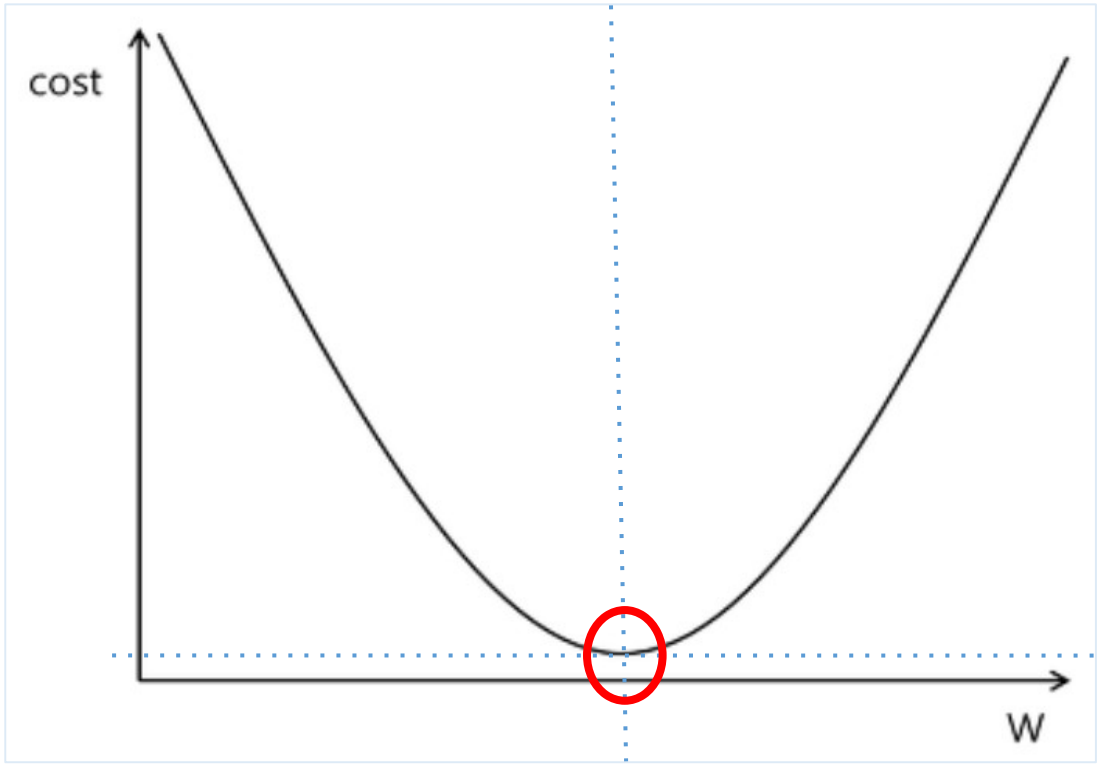
- 경사하강법 (Gradient Decent) : 비용(cost)/손실함수(loss), 오차(error)
  경사상승법 (Gradient Ascent) : 이익(profit), 점수(score)

모델 평가 시 **손실/비용함수 값이 최소가 될때**가 **최적의 모델**
손실/비용함수 값이 최소가 되는 **모델 파라미터**를 찾는 것

ML OPTIMIZATION

◆ 최적화

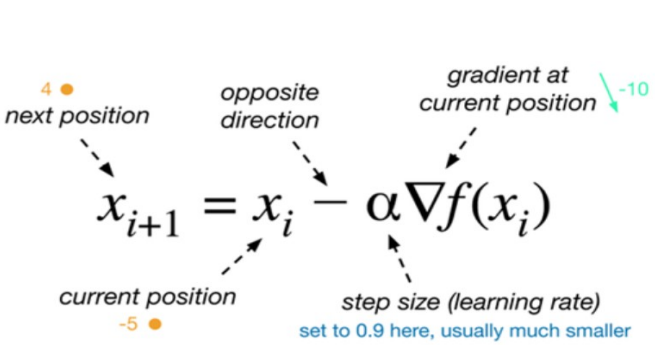
- ML/DL 오차와 기울기 관계

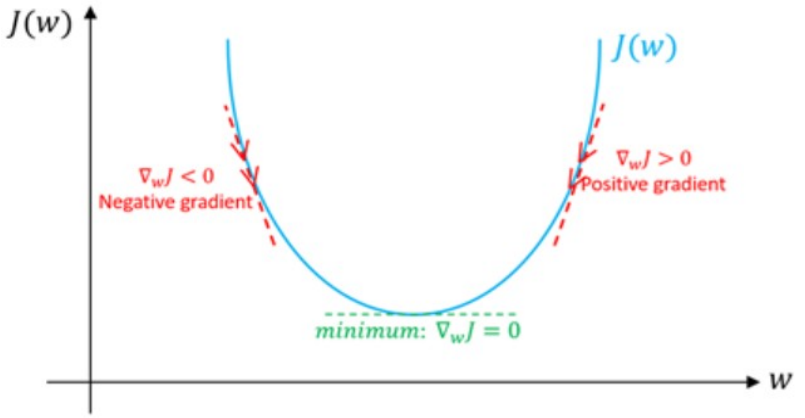


ML OPTIMIZATION

◆ 경사하강법 (Gradient Descent)

- 등산 후 하산 해야 하는 상황
- 현재 내 위치로부터 **경사가 가장 가파른 방향으로 이동** 하는 것
- **최소점을 만족하는 파라미터 값을 찾는 것**
- **일차 미분 이용**한 최적화 기법
- 미분 통해 gradient를 구한 후 **반대 방향(음수)으로 이동**



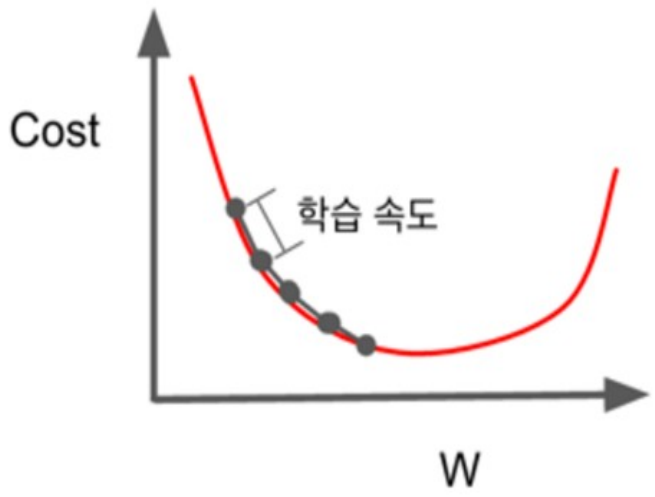


ML OPTIMIZATION

◆ 경사하강법 (Gradient Descent)

학습속도(Learning Rate)

- 경사 하강법에서 **학습 단계별로 움직이는 학습 속도 정의**
- W값 조정해 가면서 Cost 값이 최소가 되는 값을 찾기 위한 것
- W값이 다음 W값이 되는 속도



오버슈팅(Over shooting)

학습속도가 큰 경우 발생
최소값으로 내려가지 않고 반대편으로 넘어가
무한대

스몰 러닝 레이트(Small Learning Rate)

학습속도가 매우 작은 경우 발생
예) 0.0001
최소값 가기전에 학습 종료

ML OPTIMIZATION

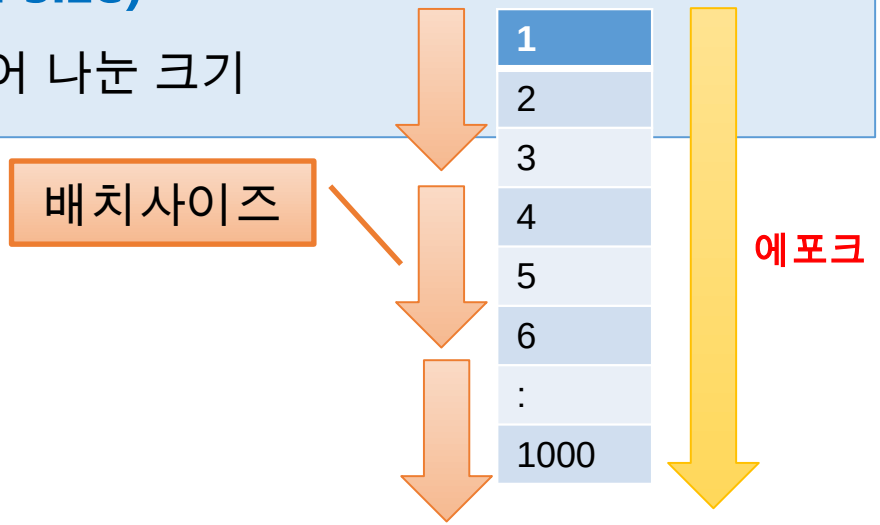
- **경사하강법 (Gradient Descent)**

에포크(Epoch) <- scikit-learn에서 max_iter 파라미터

- 전체 샘플을 모두 사용하는 한 번 반복 의미

배치 사이즈(Batch-size)

- 전체 샘플을 쪼개어 나눈 크기



ML OPTIMIZATION

◆ 경사하강법 (Gradient Descent)

배치 학습 / 오프라인 학습

- 모든 데이터를 **한꺼번에 학습**
- **시간과 자원이 많이 소모**되어 오프라인에서 수행
- 새로운 데이터 학습 위해 전체 데이터를 처음부터 다시 학습

점진적 학습 / 온라인 학습

- 데이터를 순차적으로 **한 개씩 또는 작은 묶음으로 학습 진행**
- 연속적으로 데이터를 받고 변화에 빠르게 적용 가능
- 기존 모델에 **새로운 데이터 추가 학습 진행**

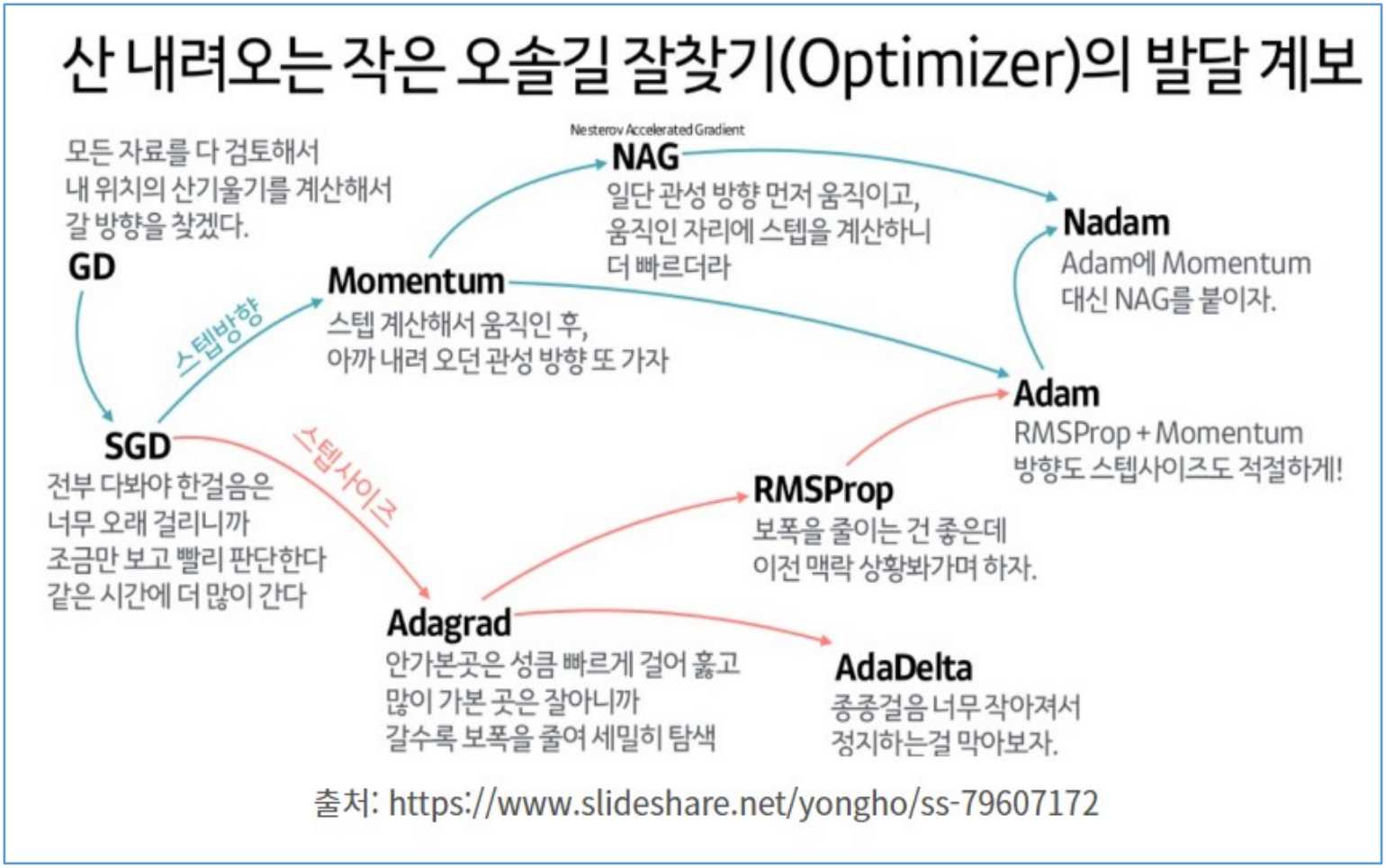
ML OPTIMIZATION

◆ 경사하강법 (Gradient Descent)

BGD (Batch Gradient Descent) 배치경사하강법	전체 학습 데이터 Gradient 계산 많은 시간 및 계산량 소요되지만 정확도 높음
SGD (Stochastic Gradien Descent) 확률적경사하강법	한 번에 한 개 데이터만 랜덤 샘플링 통해 추출 후 Gradient 계산 BGD에 비해 다소 부정확할 수 있지만 속도가 훨씬 빠름
MSGD (mini-batch Gradient Descent) 미니배치경사하강법	BGD와 SGD를 절충하여 일부 학습 데이터(mini-batch)를 Gradient 계산 **SGD**에 비해 정확도 높고 **BGD**에 비해 효율적
Momentum	기울기 방향으로 힘을 받아 물체가 가속되어 공이 구르는 듯한 움직임 이전 값과 비교 후 같은 방향으로 업데이트 진행 -> 경사하강 + 관성, **SGD**보다 빠름
AdaGrad (Adaptive Gradient)	변수들을 update할 때 각각의 변수마다 step size를 다르게 설정해서 이동하는 방식, 기존 기울기 값을 제곱한 값을 더하여 학습률을 조정
RMSProp / AdaDelta	AdaGrad의 갱신 속도 느려지는 단점을 해결한 방법 먼 과거의 기울기는 조금 반영하고 최신의 기울기를 많이 반영
Adam (Adaptive Moment Estimation)	과거 미분값 계속 가중평균 내면서 효율적 업데이트 AdaGrd + Momentum 방식 결합

ML OPTIMIZATION

- 경사하강법 (Gradient Descent)



BY ANECE SO

SGD OPTIMIZER

SGD OPTIMIZER

◆ SGD 최적화

- **확률적 경사하강법(SGD, Stochastic Gradient Descent)을 이용하여 구현하는 모델들**
- 분류와 회귀 모두에 적용

SGD OPTIMIZER

◆ SGD 최적화

- **SGDClassifer**
  - 이진분류 : 로지스틱회귀 , 이진 크로스엔트로피 손실함수
  - 다중분류 : 크로스엔트로피 손실함수
- **SGDRegressor**
  - 평균제곱근오차 손실함수 사용

SGD OPTIMIZER

◆ SGDClassifier

◺ Scikit-Learn Lib 사용 - 확률경사하강법

```
from sklearn.linear_model import SGDClassifier
(loss='hinge'                    : 손실함수 설정
 penalty= ' l2'                  : 규제 방법 설정
alpha=0.0001                     : 규제 값 설정 ( 클수록 강력한 규제)
l1_ratio=0.15                    :  L1규제 비율
fit_intercept=True               :  절편 사용 여부 설정
max_iter=1000                    :  학습 횟수 설정
tol=0.001                        :  정밀도
```

SGD OPTIMIZER

◆ SGDClassifier

➢ Scikit-Learn Lib 사용 - **학습객체**

```
from sklearn.linear_model import SGDClassifier
( shuffle=True                  : 에포크 후 데이터 섞는 유무
 verbose=0                      : 설명 출력 여부
 epsilon=0.1                    : 손실 함수에서 사용되는 값
 n_jobs=None                    : 병렬처리 설정 (CPU 수)
 random_state=None              : 난수 설정
 learning_rate= ' optimal'      : 학습 속도
 eta0=0.0                       : 초기 학습속도
```

SGD OPTIMIZER

◆ SGDClassifier

◺ Scikit-Learn Lib 사용 - 확률경사하강법

```
from sklearn.linear_model import SGDClassifier
( power_t=0.5                    : 역 스케일링 학습률
  early_stopping=False                  : 조기 중지 여부 설정
  validation_fraction=0.1        : 조기 중지 위한 검증 셋  비율
  n_iter_no_change=5                    : 조기 중비 전 반복횟수
  class_weight=None                     : 클래스별  가중치
  warm_start=False               : 초기화 유무
  average=False                  : True -> 모든 업데이트에 대한 평균 SGD 가중치
                                           계산 후 결과를 coef_속성에 저장
)
```

SGD OPTIMIZER

◆ SGDClassifier

◺ Scikit-Learn Lib 사용 - 확률경사하강법

```
from sklearn.linear_model import SGDClassifier
( power_t=0.5                    : 역 스케일링 학습률
  early_stopping=False                  : 조기 중지 여부 설정
  validation_fraction=0.1        : 조기 중지 위한 검증 셋  비율
  n_iter_no_change=5                    : 조기 중비 전 반복횟수
  class_weight=None                     : 클래스별  가중치
  warm_start=False               : 초기화 유무
  average=False                  : True -> 모든 업데이트에 대한 평균 SGD 가중치
                                           계산 후 결과를 coef_속성에 저장
)
```