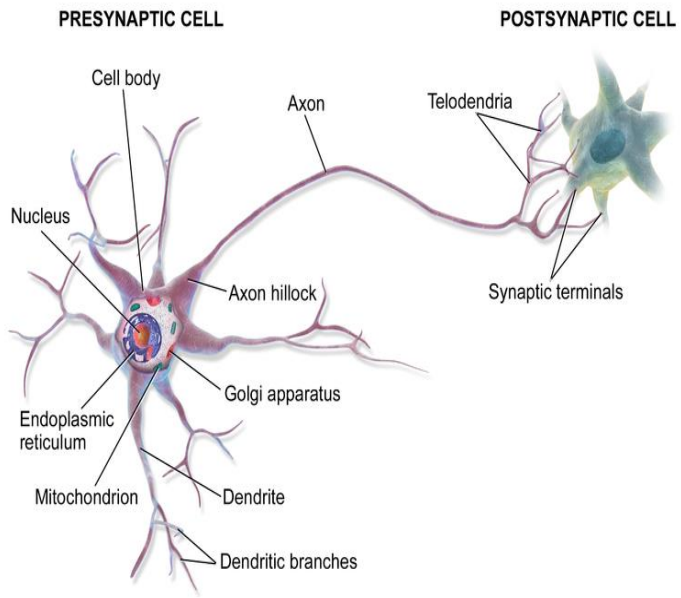


The Anatomy of a Multipolar Neuron

Artificial Intelligence

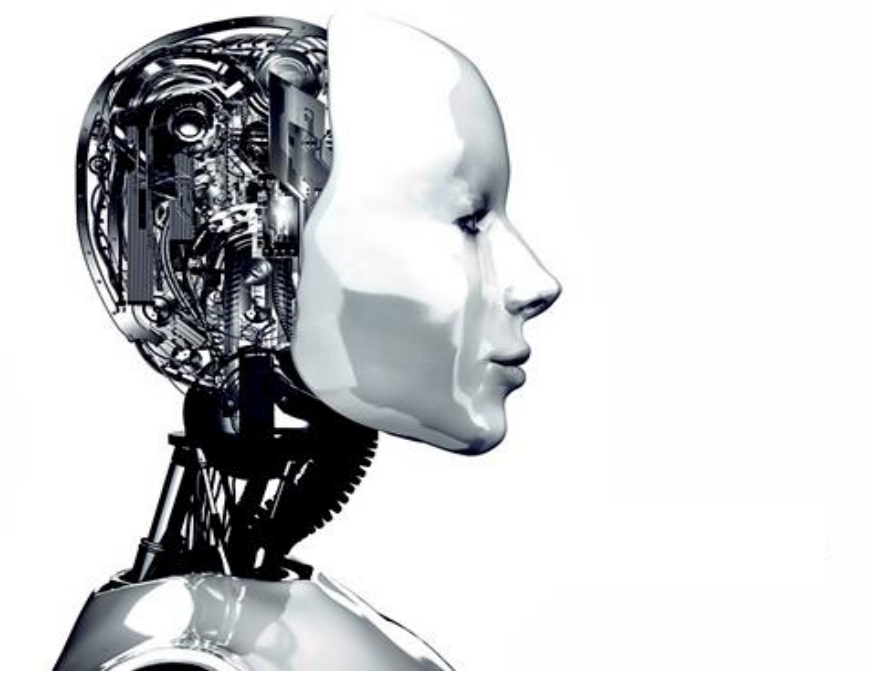
Machine Learning

Deep Learning

Lee Byung Han

byunghan.lee@goodus.com

목 차



Artificial Intelligence Concept & Trend

인공지능이란 무엇일까요?

"**인간의 지능 자체를 가진 기계**를 만들고자 하는 입장"

"**인간이 지능을 사용하는 것과 같은 방식을 기계**에게 시키려는 입장"

지능 활동 3가지 요소

- **상황 파악**
  입력된 정보(오감)를 바탕으로 현 상태 파악(분석)
- **추론 및 판단**
  분석된 상황을 바탕으로 어떤 결론에 도달
- **반응 혹은 행동**
  도달한 결론에 맞게 행동

출처: 코넬대학 로버트 스턴버그 교수

AI는 생명체의 이 같은 **지적 활동**을 **인공적으로 흉내** 내는 것을 말하며, 이를 위해 **학습하는 알고리즘(Software : ML, DL)**을 **특정의 목적 기계(컴퓨터, 로봇 자동차, 스피커, 스마트 폰.....)가 수행**하는 것을 지칭함.

인공지능 개발 방법론(방향성)

지식 기반형

전문가 시스템

의미 망

- **특정분야의 전문 지식을 인위적으로 컴퓨터에 부여(학습)**
- 사람의 지식을 기호의 조합으로 표현
- **일종의 자문형 Expert System**
- **의료진단, 화학분자, DNA 분석**
  (60만건 진단서, 200만쪽 의학 전문 서적, 150만명의 환자 기록)

데이터 기반형

인공 신경망

딥러닝

- **기존 데이터에서 공통 성질, 특성(Feature), 패턴 추출하여 학습**
- Machine Learning Algorithms, Big Data Analysis
- Neural Network, Deep Learning
- **예측, 추론형 인공지능**

경험 축적형

모의실험

강화학습

- **시행착오(Trail & Error)로부터 학습, 큰 보상(Reward) 추구 방향성**
- 불확실성 하의 모의실험, **Reinforcement Learning**
- Game, **Robotics**, **자율 주행**, Simulation

글로벌 기업들의 인공지능

글로벌 기업들은 나름대로의 명칭으로 AI 정의

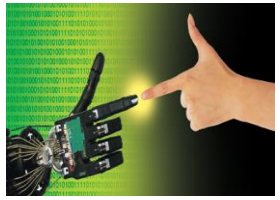
Artificial Intelligence	Machine Learning	Cognitive Computing	Machine Learning	Smart Machine
Google	Microsoft	IBM	amazon	Gartner

******* *Artificial Intelligence* *******

AI 개념 및 동향

AI 의 5가지 핵심기술

자연어 처리 (NLP)



소통 기술
텍스트 이해, 텍스트 작성, 번역

머신 러닝 (딥러닝)

예측 기술
데이터 인식, 알고리즘 발견, 변화 예측

패턴 인식

인간 감각모방 기술
스피치, 비전, 문자, 심볼 등을 관찰 & 분류

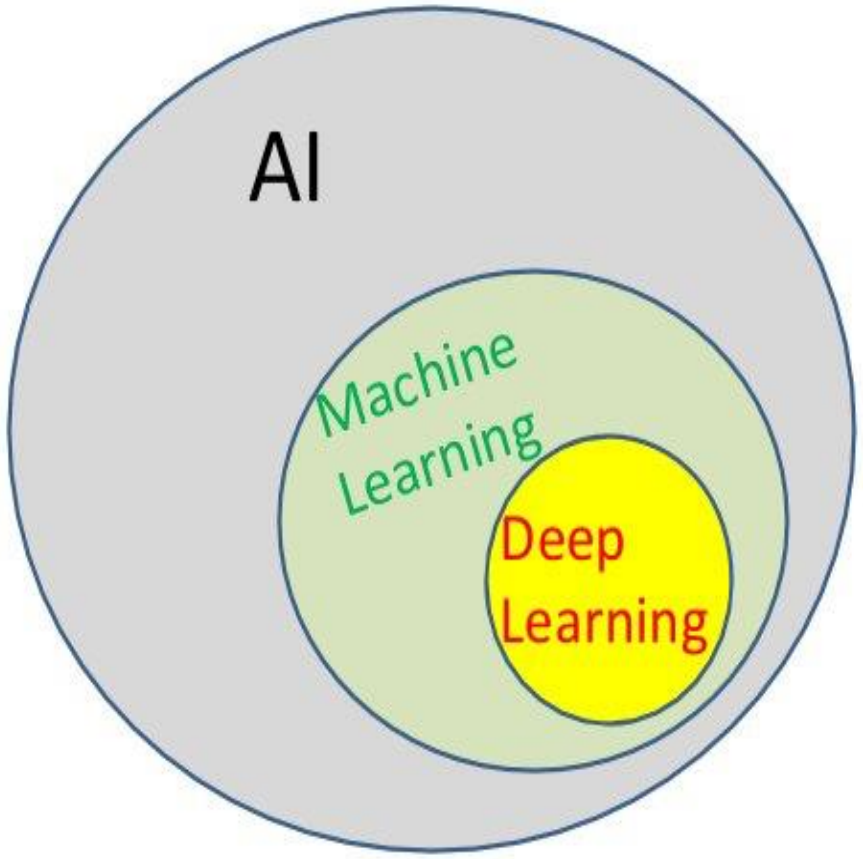
지식 표현 & 추론

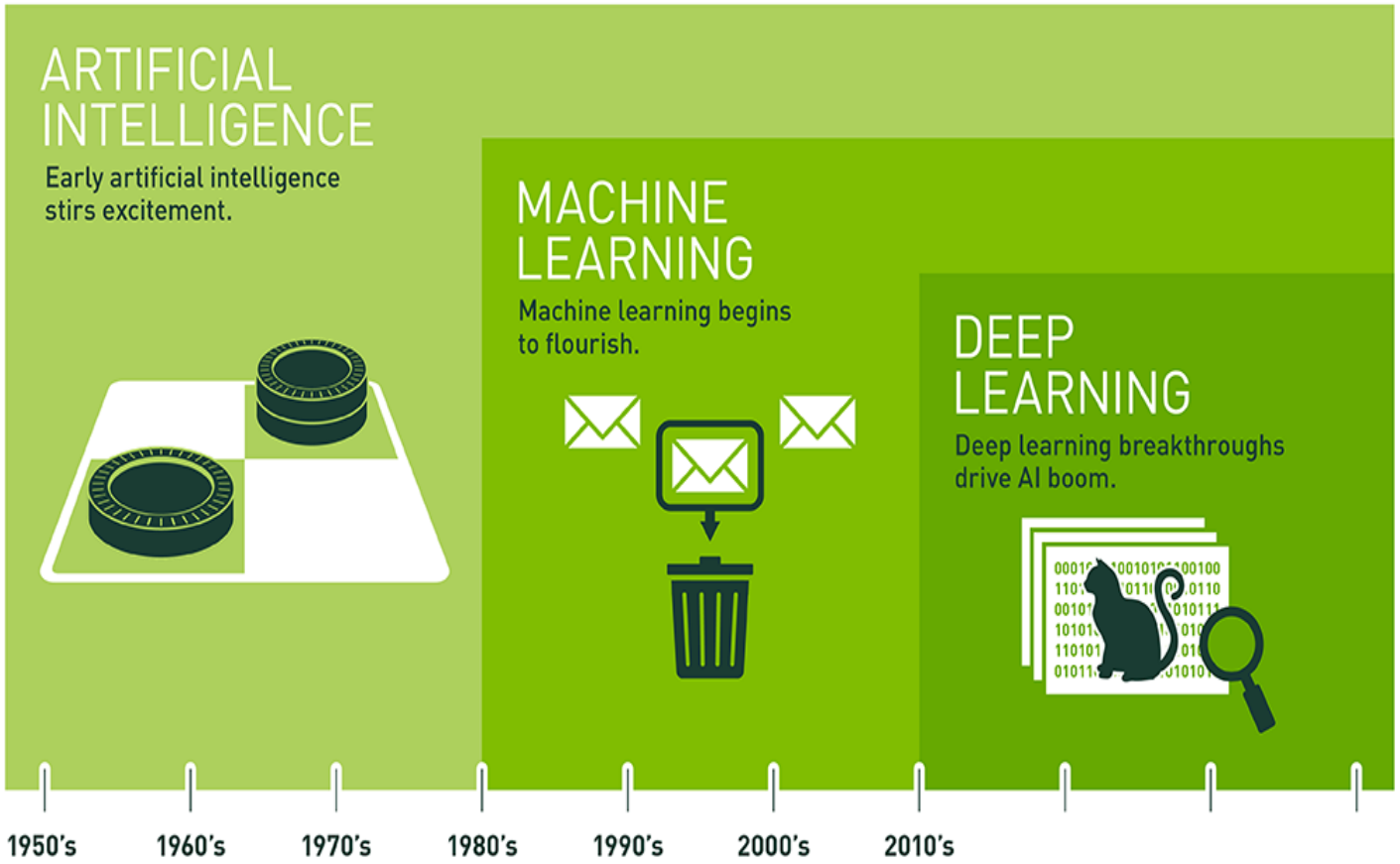
정보 구조화 기술
시멘틱, 온토로지, 추론엔진 등을 통해 구조화

관리 & 기획

지능형 에이전트 기술
자동로봇, 자율 주행차

AI > Machine Learning > Deep Learning





Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

AI Industry Eco-System

Application, Product, Service

공통 어플리케이션 : 챗봇, 가상비서, 프로세스 자동화(BI), 로봇틱스, 예측 분석, 이상 감지

산업 특화 솔루션 : 자동차, 물류, 리테일, 제조, 통신, 마케팅, 의료, 보안, 가전, 금융

선진기업 솔루션 & 서비스 : IBM, Google, Microsoft, Amazon AI APIs

AI Platform

선진기업

IBM Watson | Google Cloud Platform | Amazon Machine Learning (aws.amazon.com/machine-learning) | Windows Azure MACHINE LEARNING

Startup & 학계

ELEMENT AI | BONSAI AI

Framework, Algorithms

선진기업

TensorFlow | torch | The Microsoft Cognitive Toolkit | cuDNN

Startup & 학계

H2O | skymind | turi | Caffe | dmlc mxnet | R

Computing Infrastructure

GPU Cluster / HPC

NVIDIA GPU CLOUD | Wave Computing | SOFTLAYER an IBM Company

독자 인프라

Google | facebook

고성능 분석 인프라

Spark | neo4j

Processor

범용 프로세스(GPU 외)

NVIDIA TESLA PERSONAL SUPERCOMPUTER | Google Tensor Processing Unit | intel + nervana | Graphcore

특화 프로세스(FPGA, ASIC)

intel MOBILEYE | intel + Movidius | CEVA

뇌 모방 프로세스(NPU)

IBM | QUALCOMM ZEROTH NPU | intel | KNUEDGE

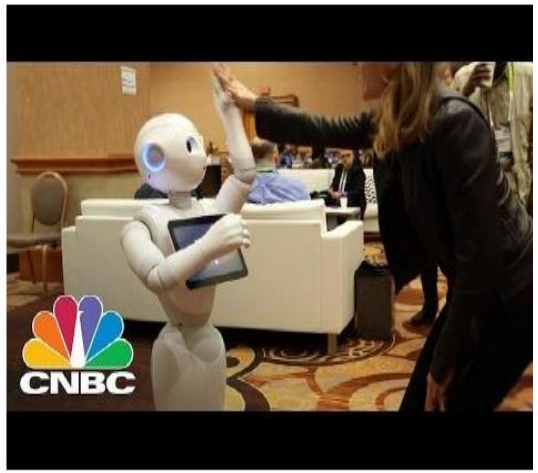
Machine Learning : Application

응용 분야	적용 사례
인터넷 분야	텍스트 마이닝, **웹 로그 분석, 스팸 필터, 추천**, 문서 분류,
컴퓨터 시각	**문자 인식, 패턴 인식, 물체 인식, 얼굴 인식**, 화상 복구, 이미지 인식, **자율 주행**
음성/언어 인식	**음성 인식**, 단어 모호성 제거, **번역**, 문법 학습, 대화 패턴 분석
모바일	동작 인식, 제스쳐 인식, **각종 센서 정보 인식 및 분석**, 떨림 방지....
통신, 제조업	**생산 & 관리 & 운영 자동화, 기계 장치의 각종 센서 정보 인식 및 분석을 통한 사전 장애 예측, 추론**
로보틱스	장애물 인식, 모터 제어, **각 산업(제조, 의료, 공공, 경찰, 보안, 군, 서비스..) 분야별 인공 로봇**
서비스 업	**고객 분석**, 시장 분석, **마케팅**, **상품 추천**, 고객 관리(CRM), 텔레마케터, 인공비서, **투자, 거래 분석**
바이오 메트릭스	홍채 인식, 심장 박동수 측정, 혈압 측정, 당뇨 측정, 암 세포 인식, 지문 인식
생물정보	유전자 인식, 단백질 분류, 유전자 조절 망 분석, 질병 진단

AI is already Here

Personal Assistant/Robot at Home

Google Home



Pepper

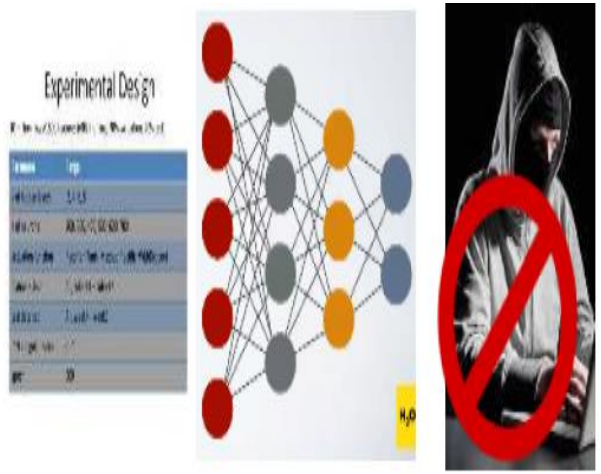
Amazon Echo "Alexa"

Jibo

자율 주행 차

바둑, 체스, 게임

Health Care



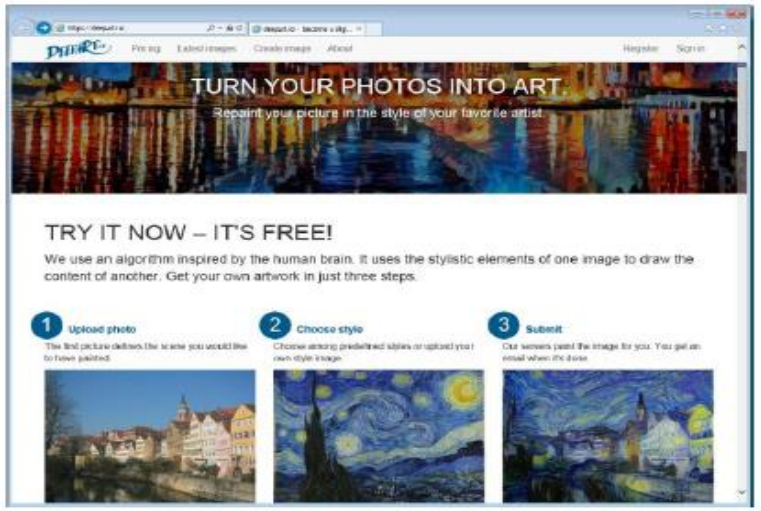
Paypal 사기거래 예측

AI 개념 및 동향

AI is already Here

주식투자 ' 신의 손' 될까?

- 글로벌 자산 운용사 BlackRock(자산 규모 약 5570조) 펀드 매니저 7명 해고 ➔ AI로 전환(2017.03)
- 투자은행 골드만 삭스 주식 트레이터 600명 ➔ 2명 (2017.04)



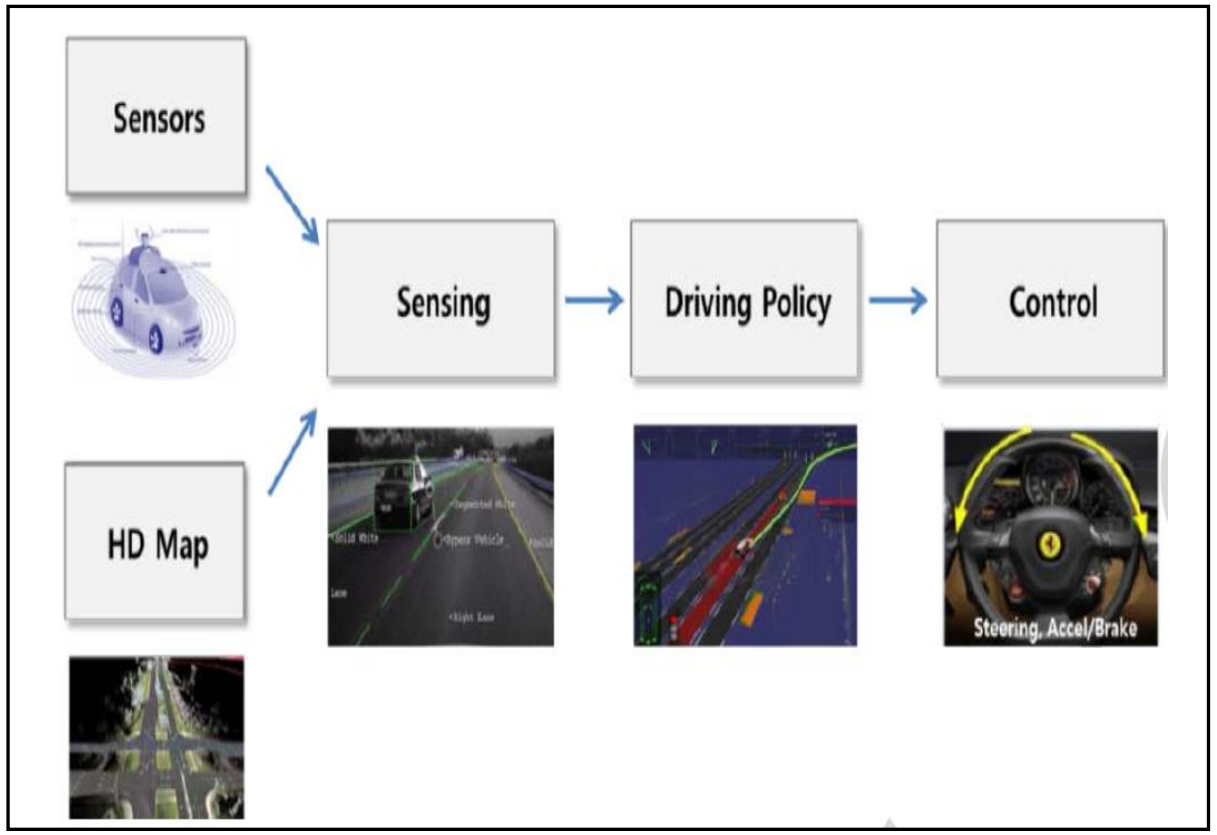
Deep Art (미술 분야)

- 원하는 사진 업로드
- 원하는 스타일(피카소 그림)을 뽑을 사진 업로드
- E-mail로 최종 그림 결과 받기

IBM 왓슨 (제과 분야)

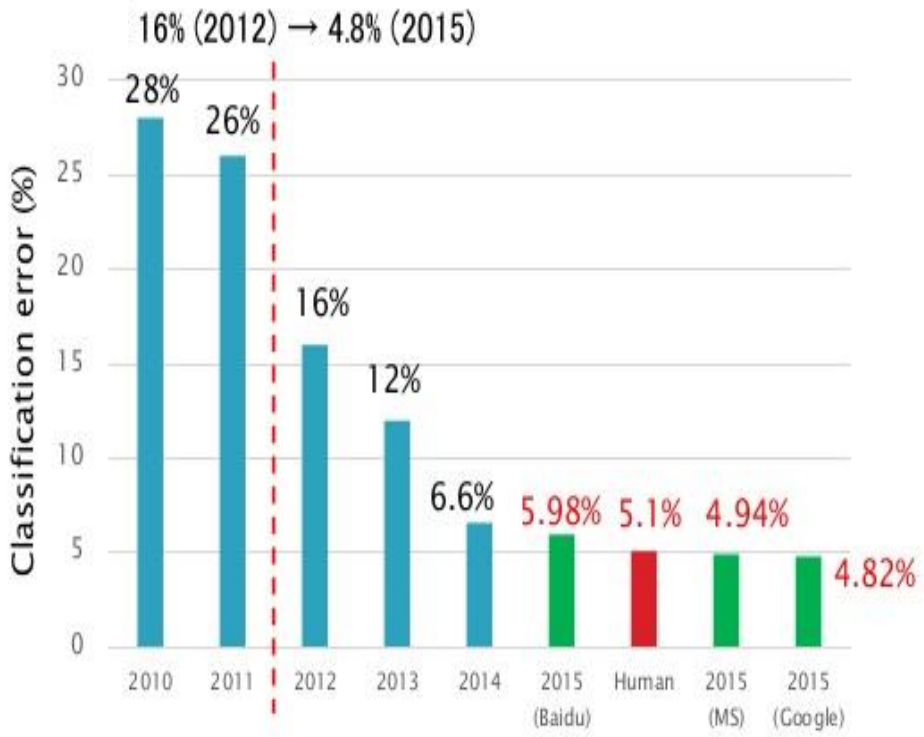
- 2017년 9월 인공지능이 기획한 빼빼로 2종 출시 (빼빼로 카카오닙스 & 깔라만시)
- 8만여 개의 식품 관련 인터넷 사이트에 게재된 약 1000만 건의 정보를 수집 및 인공지능 학습
- 분석 결과 식품 분야에서는 향후 맥주·치즈·고추 등의 소재가 인기가 높아질 것으로 예측, 과자 및 초콜릿 분야에서는 헤이즐넛,딸기,코코아,**카카오닙스,깔라만시** 등이 높은 관심을 얻을 것으로 예측

AI is already Here



- Computer Vision(시각 인식) 및 Machine Learning 기반의 자율 주행 차
- 대표적 구현 기술 : **인지(Sensing) – 판단(Driving Policy)** – 제어(Control)
- **인지, 판단 영역**에서 Deep Learning 및 Reinforcement Learning 활용

AI is already Here

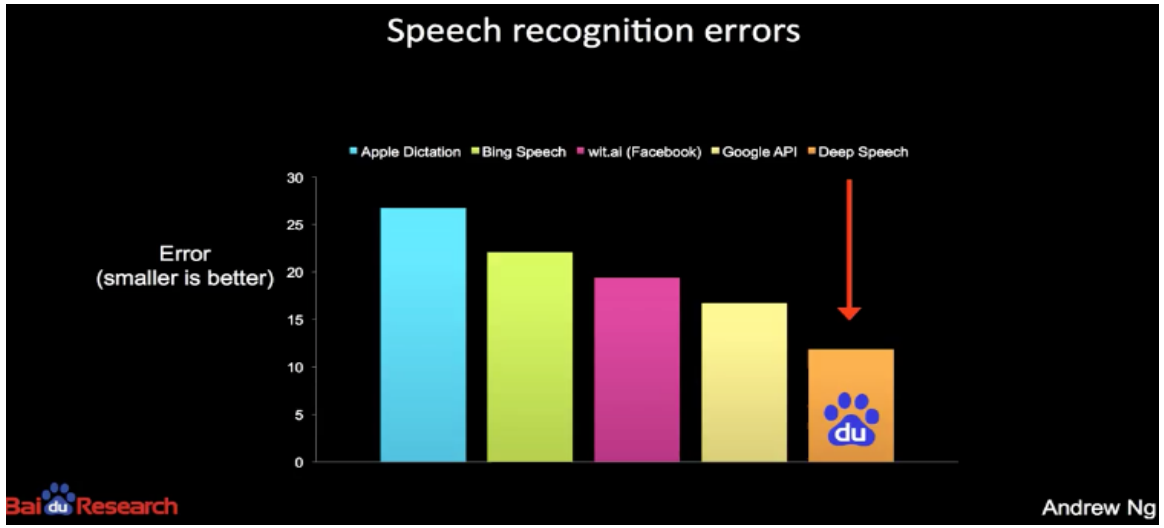


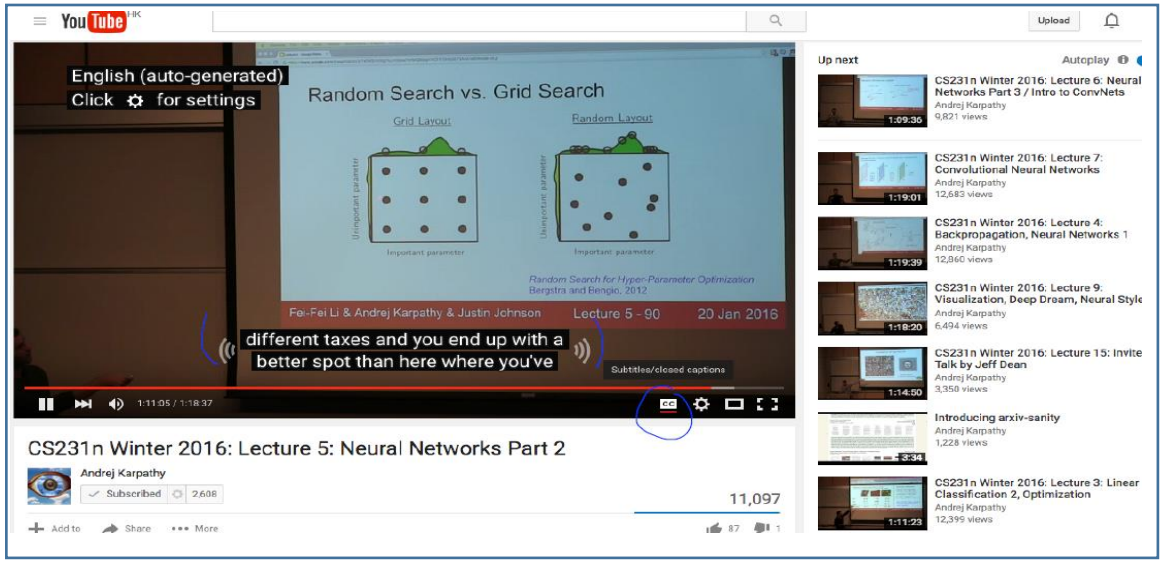
He et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", arXiv, 2015.
Ioffe et al., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", arXiv, 2015.

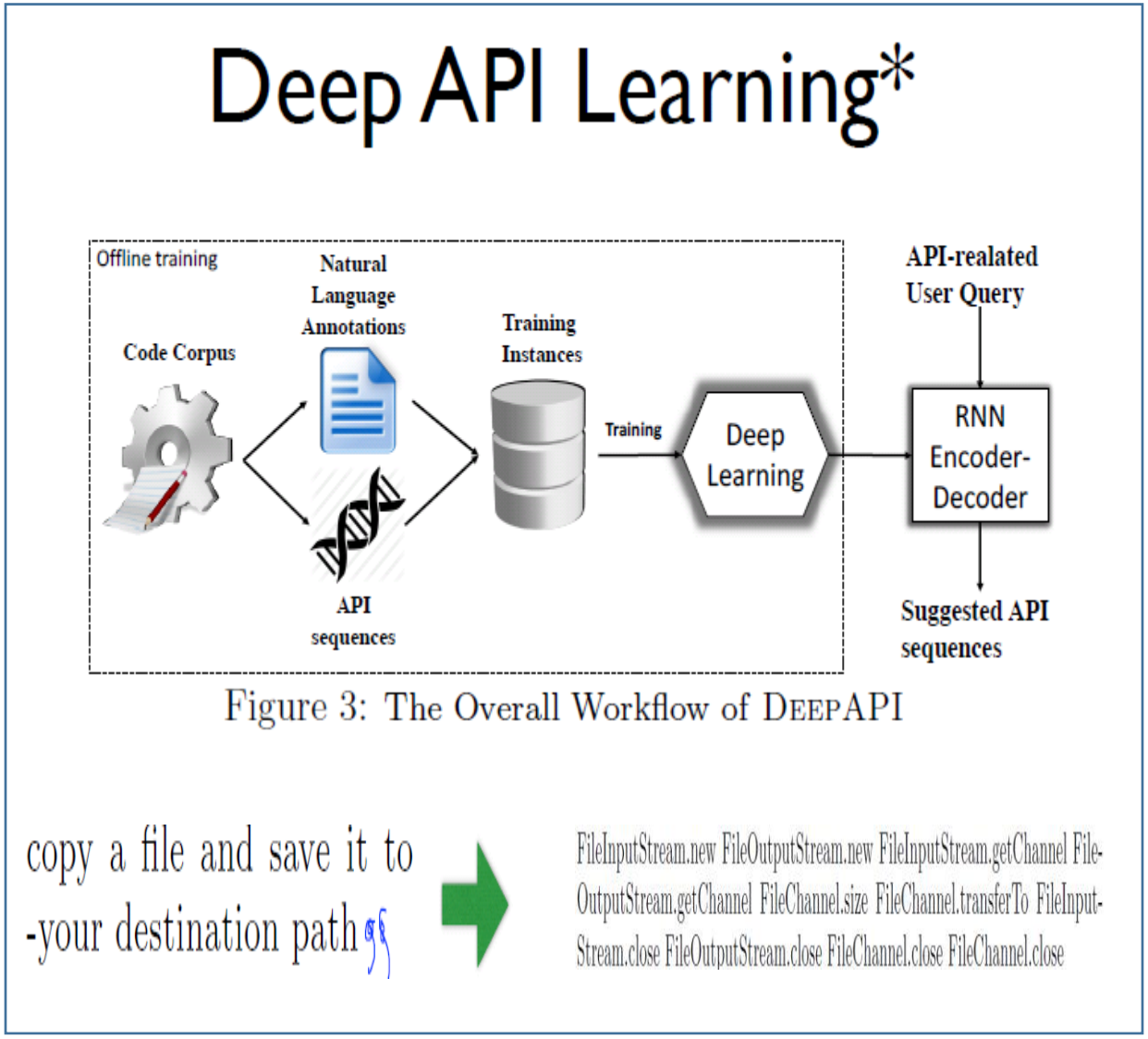
Image Question and Answering

Question

AI is already Here

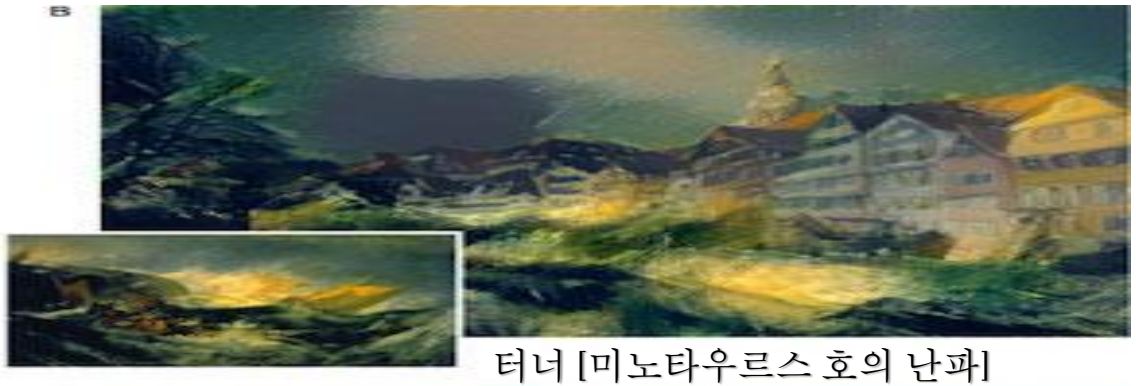






AI is already Here

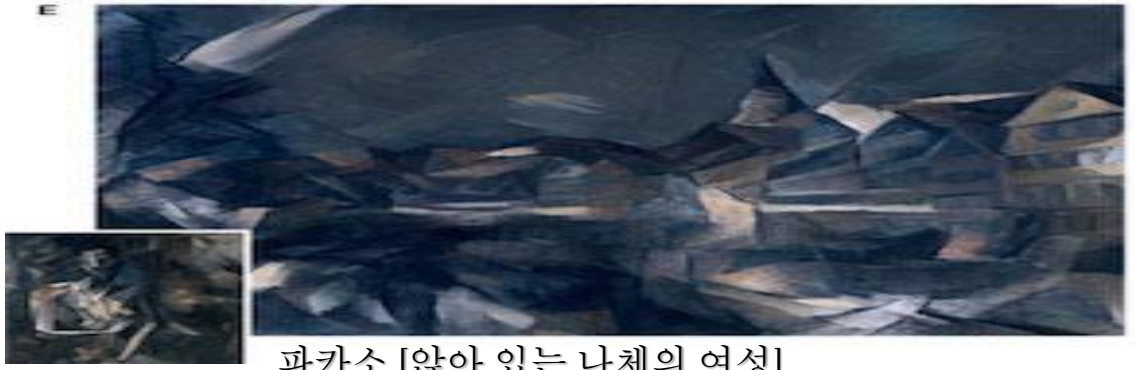
원본 풍경 사진(독일 튀빙겐)



터너 [미노타우르스 호의 난파]

빈센트 반 고흐 [별이 빛나는 밤]

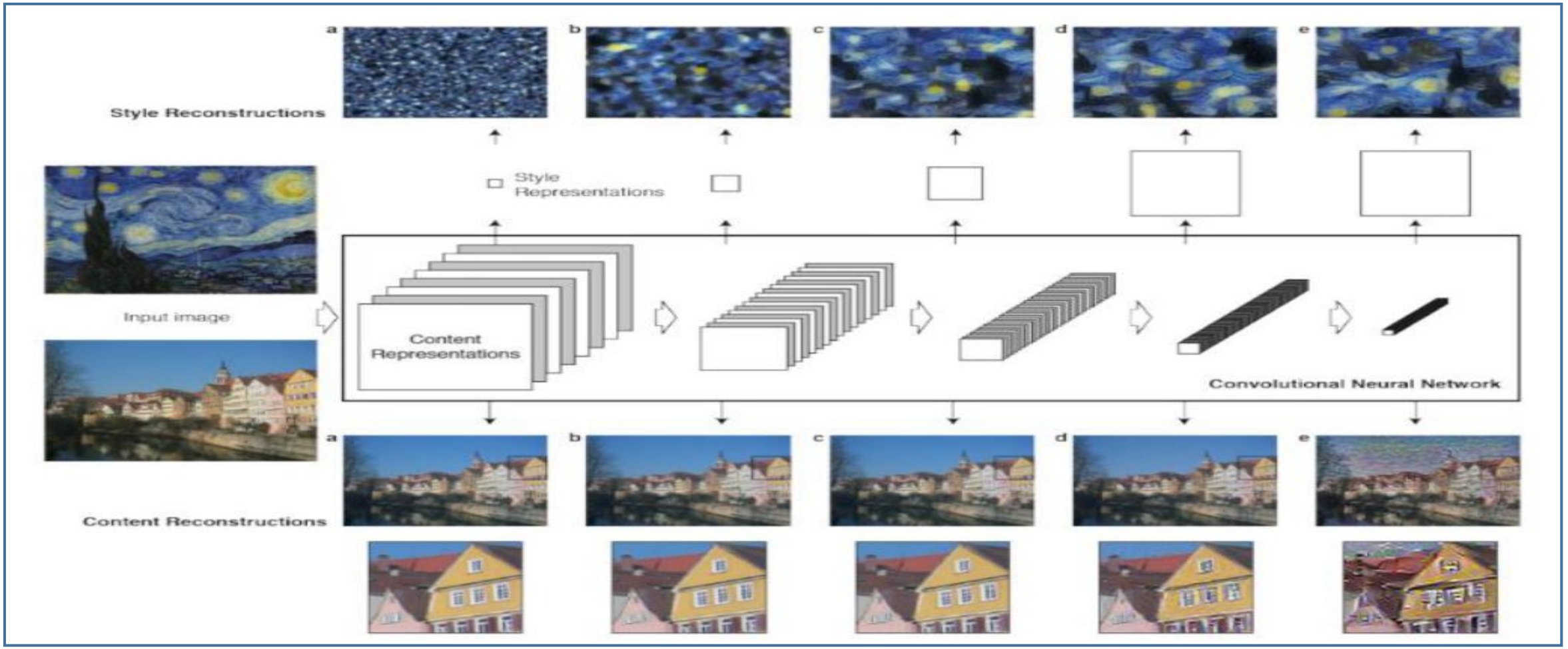
뭉크 [절규]



파카소 [앉아 있는 나체의 여성]

칸딘스키 [구성 VII]

AI is already Here



- Deep learning으로 고급스러운 예술적 이미지를 생성
- 그림의 **내용(contents)**과 **스타일(style)**을 구분하여 신경망으로 **표현을 학습**하고 이를 **재 조합 출력**

AI is already Here

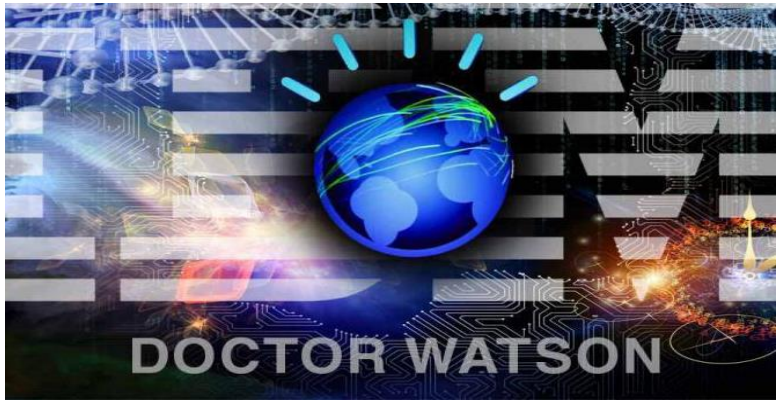
피카소(자화상) : Style

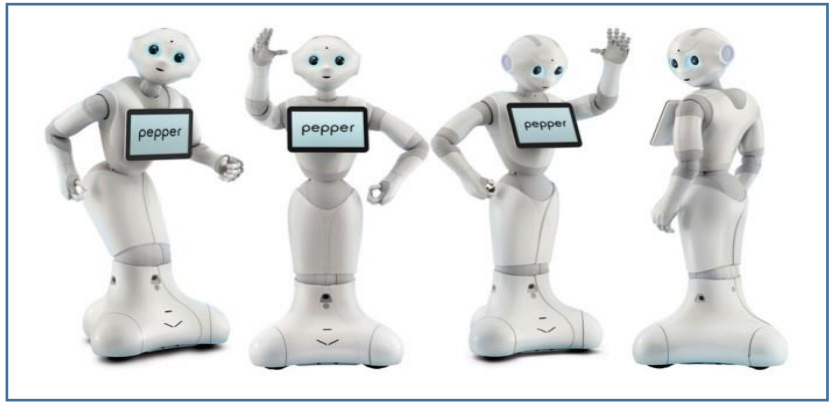
브래드 피트(사진) : Contents

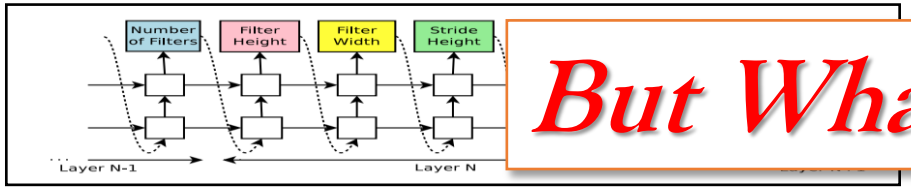
Weight(가중치) Value를 변경했을 때의 출력 그림 변화

AI is already Here

First, We've Seen All Of This (And More)







But What About Networking ?

$$+ g_t(\nabla_\theta J(\theta_t), \phi)$$

AI is already Here (Networking)

Combining SDN and Machine Learning

Knowledge-Defined Networking

Albert Mestres*, Alberto Rodriguez-Natal*, Josep Carner*, Pere Barlet-Ros*, Eduard Alarcón*,
Marc Solé†, Victor Muntés-Mulero†, David Meyer‡, Sharon Barkai§, Mike J Hibbett¶, Giovani Estrada¶,
Khaldun Ma'ruf‖, Florin Coras**, Vina Ermagan**, Hugo Latapie**, Chris Cassar**, John Evans**, Fabio Maino**,
Jean Walrand†† and Albert Cabellos*
* Universitat Politècnica de Catalunya † CA Technologies ‡ Brocade Communication § Hewlett Packard Enterprise
¶ Intel R&D ‖ NTT Communications ** Cisco Systems †† University of California, Berkeley

Abstract—The research community has considered in the past the application of Artificial Intelligence (AI) techniques to control and operate networks. A notable example is the Knowledge Plane proposed by D.Clark et al. However, such techniques have not been extensively prototyped or deployed in the field yet. In this paper, we explore the reasons for the lack of adoption and posit that the rise of two recent paradigms: Software-Defined Networking (SDN) and Network Analytics (NA), will facilitate the adoption of AI techniques in the context of network operation and control. We describe a new paradigm that accommodates and exploits SDN, NA and AI, and provide use cases that illustrate its applicability and benefits. We also present simple experimental results that support its feasibility. We refer to this new paradigm as Knowledge-Defined Networking (KDN).

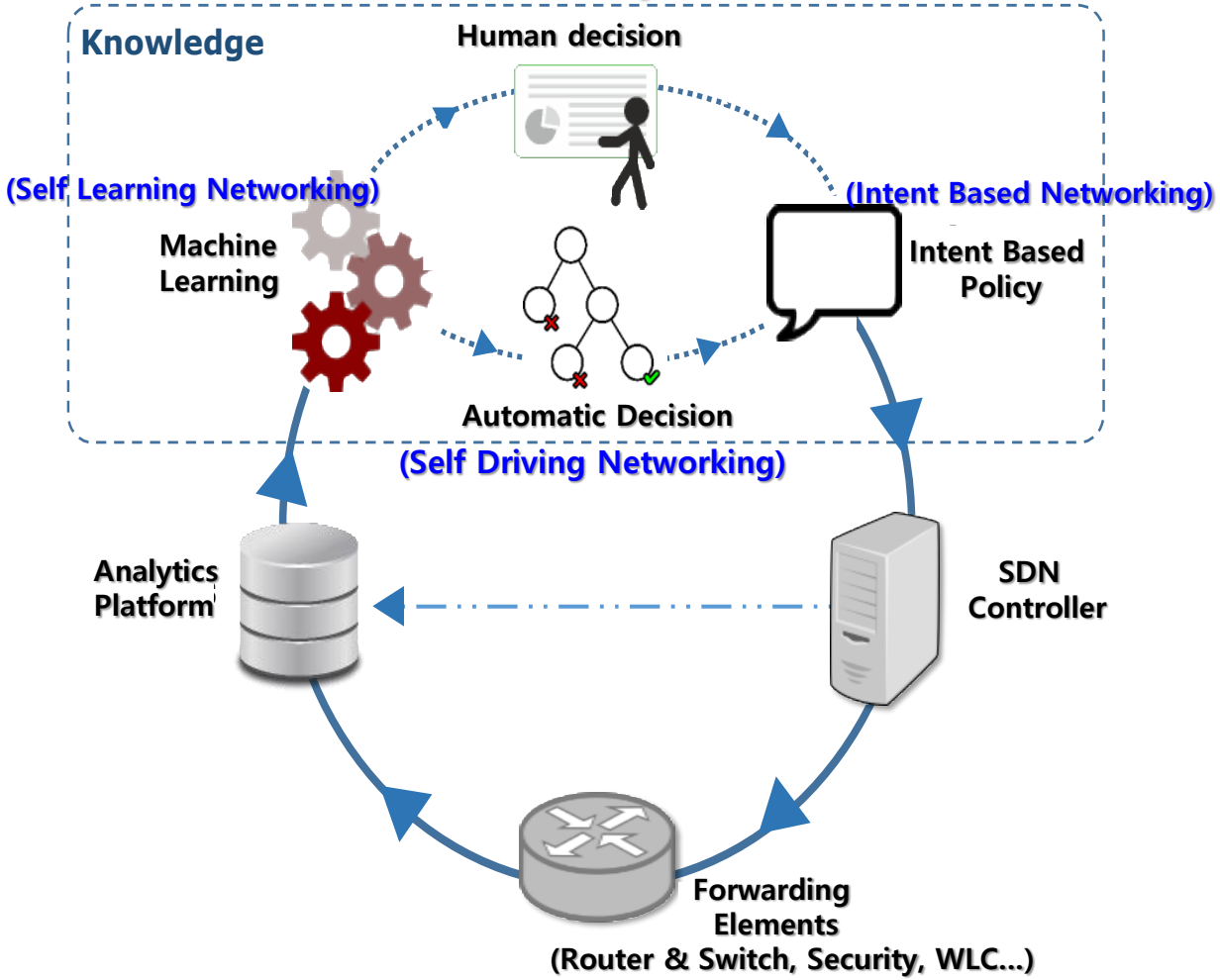
Keywords—*Knowledge Plane, SDN, Network Analytics, Machine Learning, NFV, Knowledge-Defined Networking*

techniques provide real-time packet and flow-granularity information, as well as configuration and network state monitoring data, to a centralized Network Analytics (NA) platform [4]. In this context, telemetry and analytics technologies provide a richer view of the network compared to what was possible with conventional network management approaches.

In this paper, we advocate that the centralized control offered by SDN, combined with a rich centralized view of the network provided by network analytics, enable the deployment of the KP concept proposed in [1]. In this context, the KP can use ML and Deep Learning (DL) techniques to gather knowledge about the network, and exploit that knowledge to control the network using logically centralized control capabilities provided by SDN. We refer to the paradigm resulting from combining SDN, telemetry, Network Analytics, and the Knowledge Plane as Knowledge-Defined Networking.

Knowledge-Defined Networking

Basic Control Loop



AI is already Here (Networking)

➢ IETF/IRTF Network Machine Learning Research Group (NMLRG)

- 2015
- Member : Global Vendors

➢ Main focus & Charters

"... intelligently learn the various environments of networks and react to dynamic situations better than a fixed algorithm. ... it would greatly accelerate the development of autonomic networking...."

지능적으로 학습하고 반응 ➔ 자동화된 네트워크 제어(관리)

– to use ML for routing control and optimization

Routing 제어 및 최적화에 Machine Learning 기법 활용

"... the NMLRG will work on potential approaches that apply ML technologies in network control, network management, and supplying network data for upper-layer applications..."

- 네트워크 제어 및 관리에 ML 기술 접목
- 상위계층에 표준 포맷의 네트워크 데이터 제공

– to use ML in network management to predict future network status

미래 네트워크 상태를 예측하는 NMS에 ML 적용

– to use ML to analyze network faults and support recovery learning network attacks and their behavior, so that protection mechanisms could be self-developed

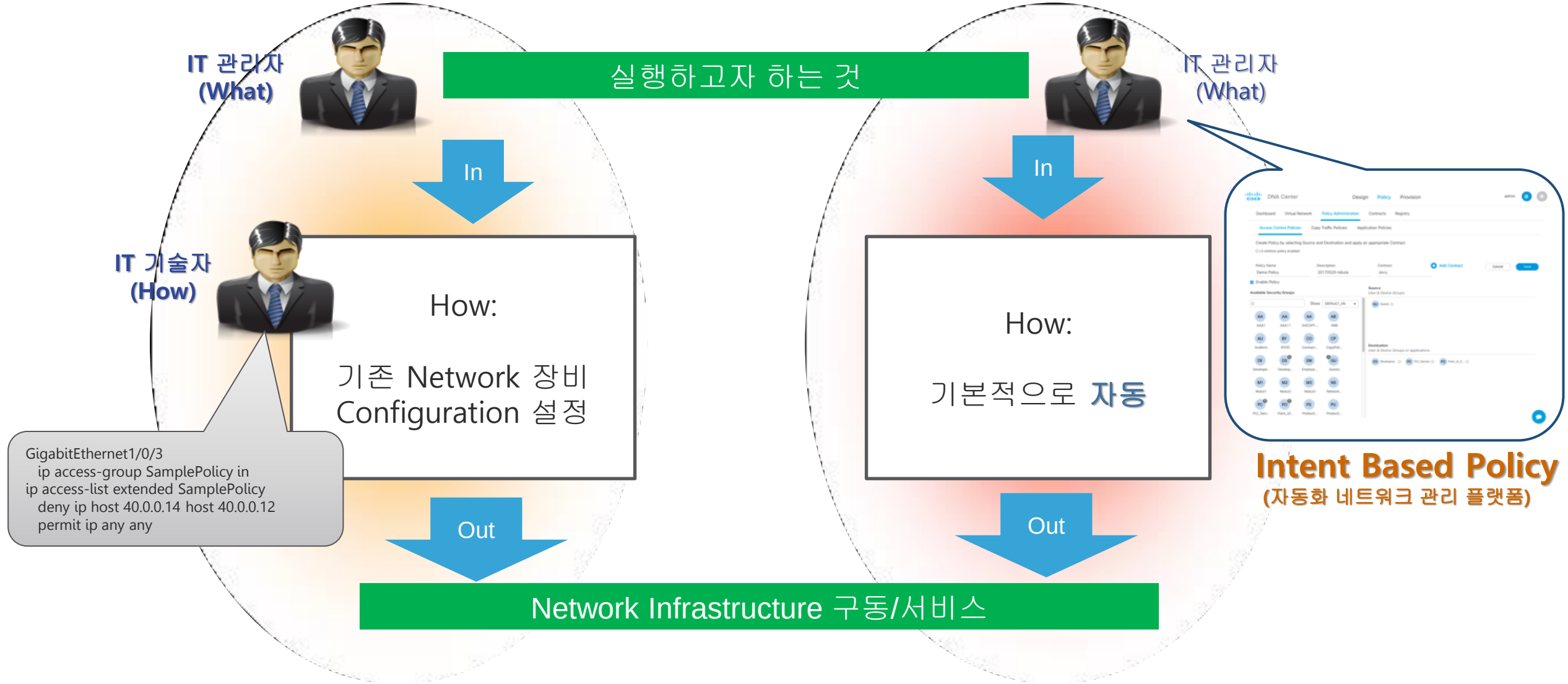
- ML을 이용한 네트워크 결함 및 공격 분석
- 자동화된 보호기법 적용

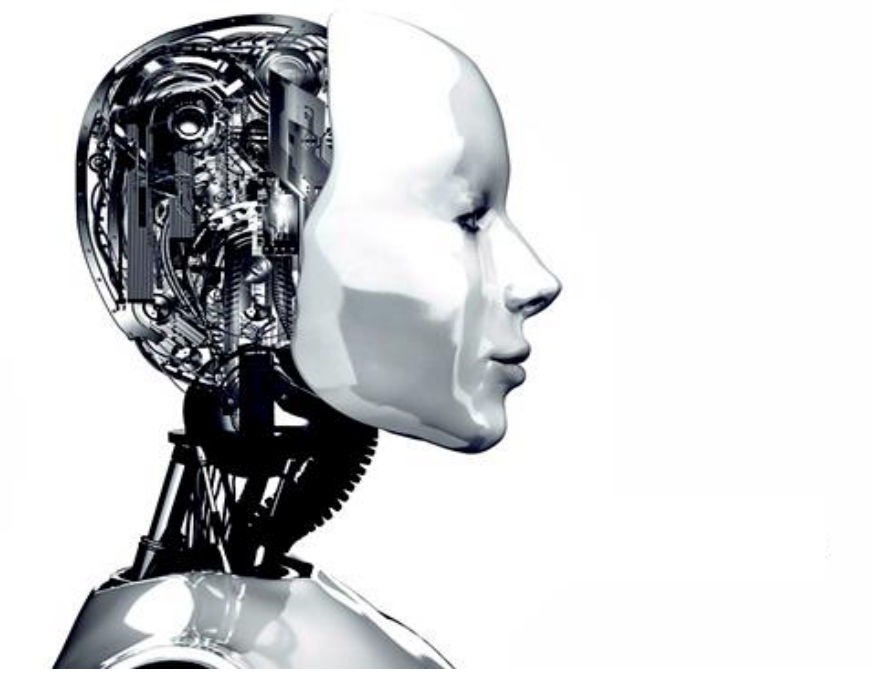
– to use ML to autonomic and dynamically manage the network

자동화된 동적 네트워크 관리에 ML 적용

Network Transformation
– "Intent Base Policy" Concept

- **"의도나 목적"**을 입력하면 **자동적으로 해석, 판단하고 적절한 기능**을 네트워크에 대해서 전개
- 다양한 애플리케이션과 네트워크가 연계하기가 쉽다





AI 발전 원동력 (4가지 요소)

AI (Deep Learning) 발전 원동력

4가지 원동력

Computing Power

(특히 GPU 기반 병렬 계산)

BIG Data

획기적인

알고리즘

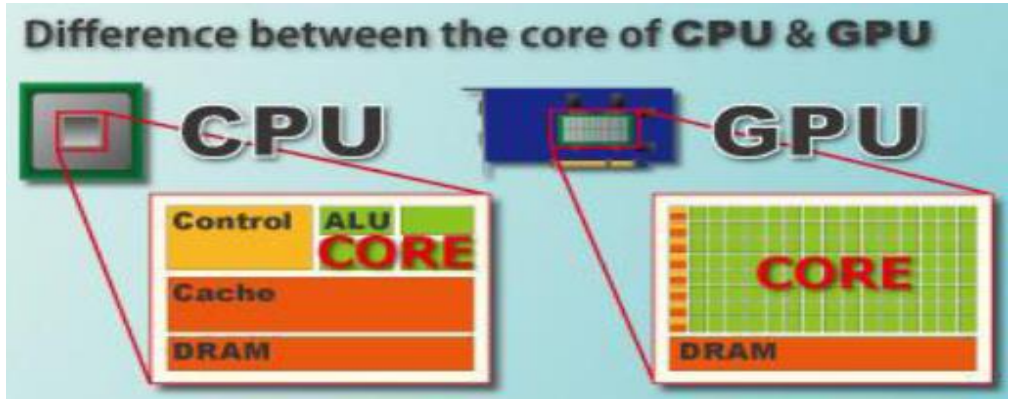
개발 환경

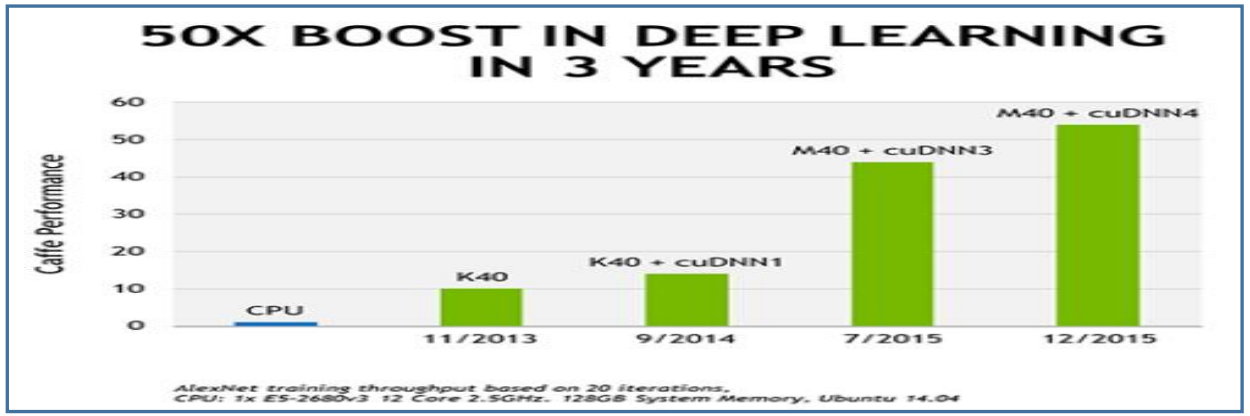
진화

AI (Deep Learning) 발전 원동력

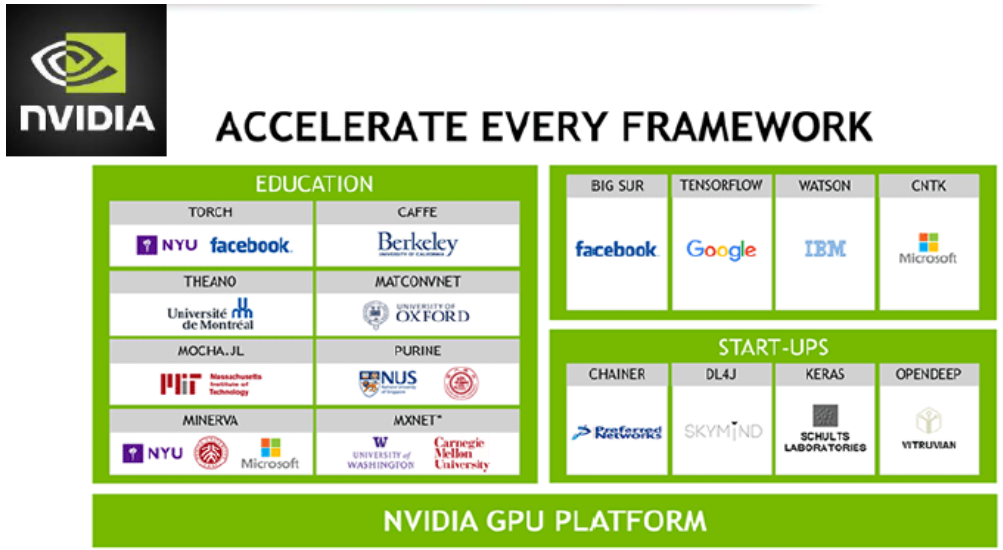
Computing Power

Computer의 계산(연산) 속도 향상으로 대규모 모델 학습 가능
특히, Graphical Processing Unit(GPU)으로 CPU보다 **훨씬 빠르게 병렬 연산** 제공





Deep Learning Model 학습에 대규모로 GPU 적용하여
획기적 속도 향상됨을 보임



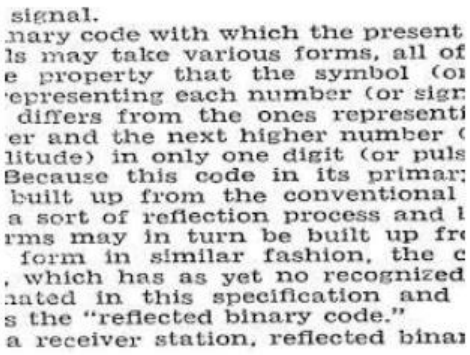
Cloud Computing

대용량 저장 장치(DB)

AI (Deep Learning) 발전 원동력

BIG Data

지능형 시스템 구성에 필요한 **기초적인 데이터**를 얼마든지 제공

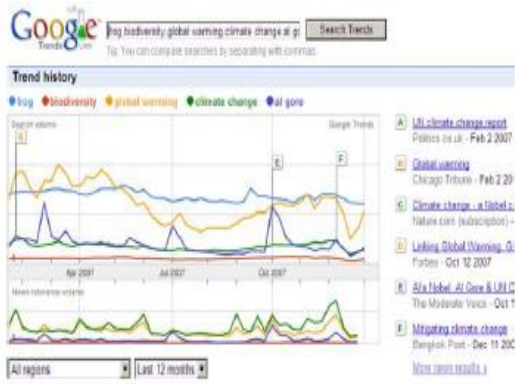

signal.
nary code with which the present
ls may take various forms, all of
e property that the symbol (o
epresenting each number (or sign
differs from the ones representi
er and the next higher number
litude) in only one digit (or puls
Because this code in its primar
built up from the conventional
a sort of reflection process and
rms may in turn be built up fr
form in similar fashion, the c
, which has as yet no recognized
nated in this specification and
s the "reflected binary code."
a receiver station, reflected binar

글자(Text) Data

시각(Visual) Data

소리(Audio) Data

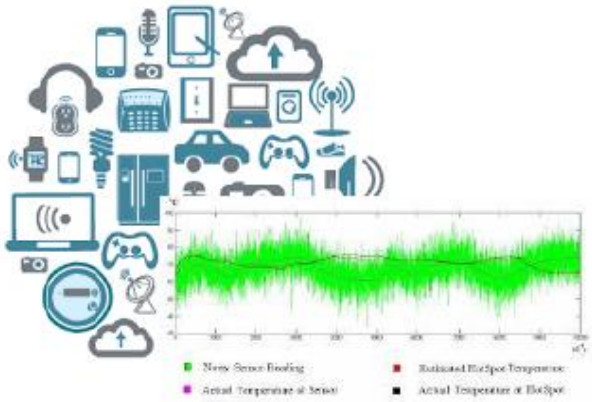
Wikipedia등 무료 정보 Data



전세계 누적된 검색 Data

인터넷 사용 기록 Data

사용자 정보(위치, Activity) Data

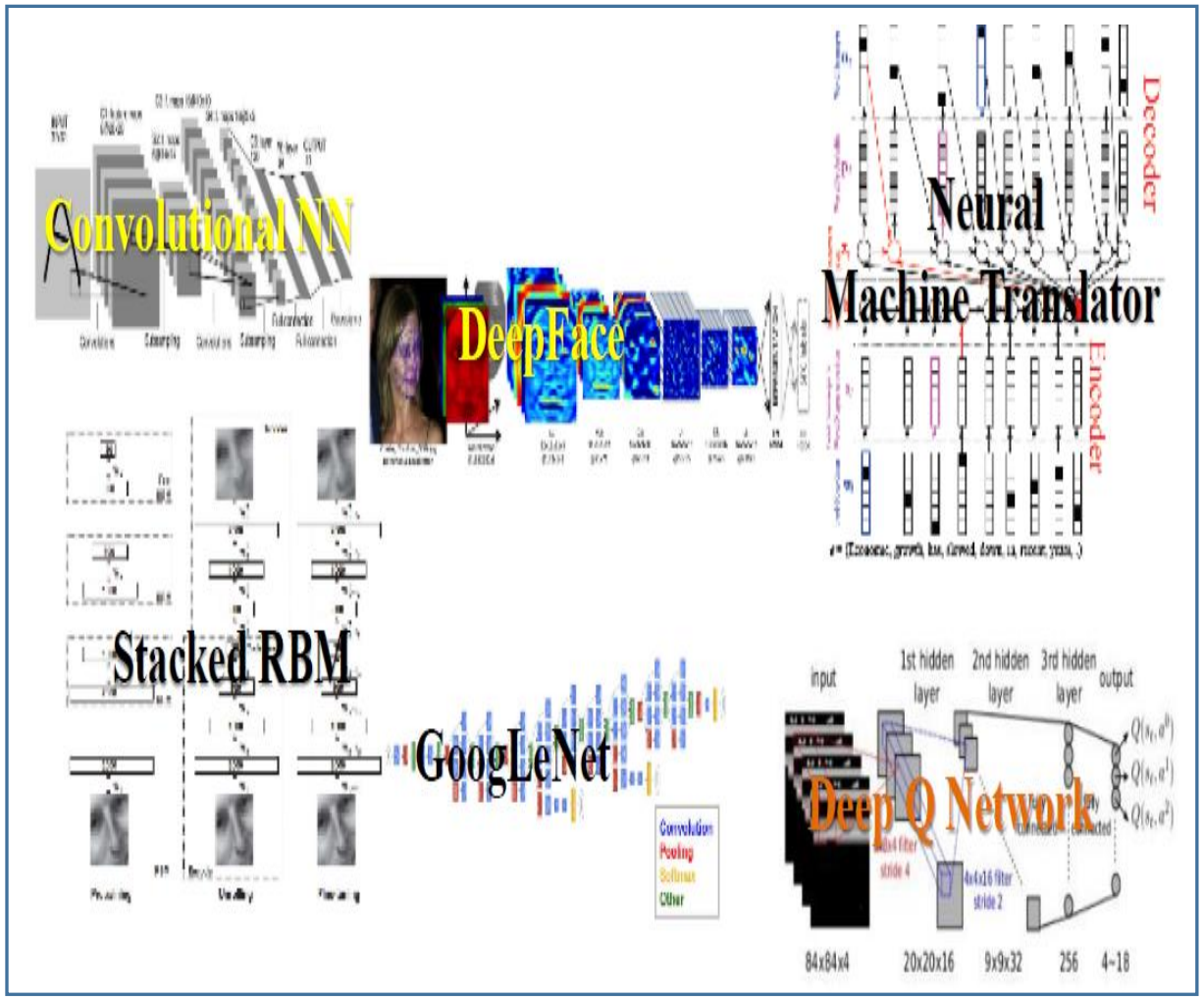


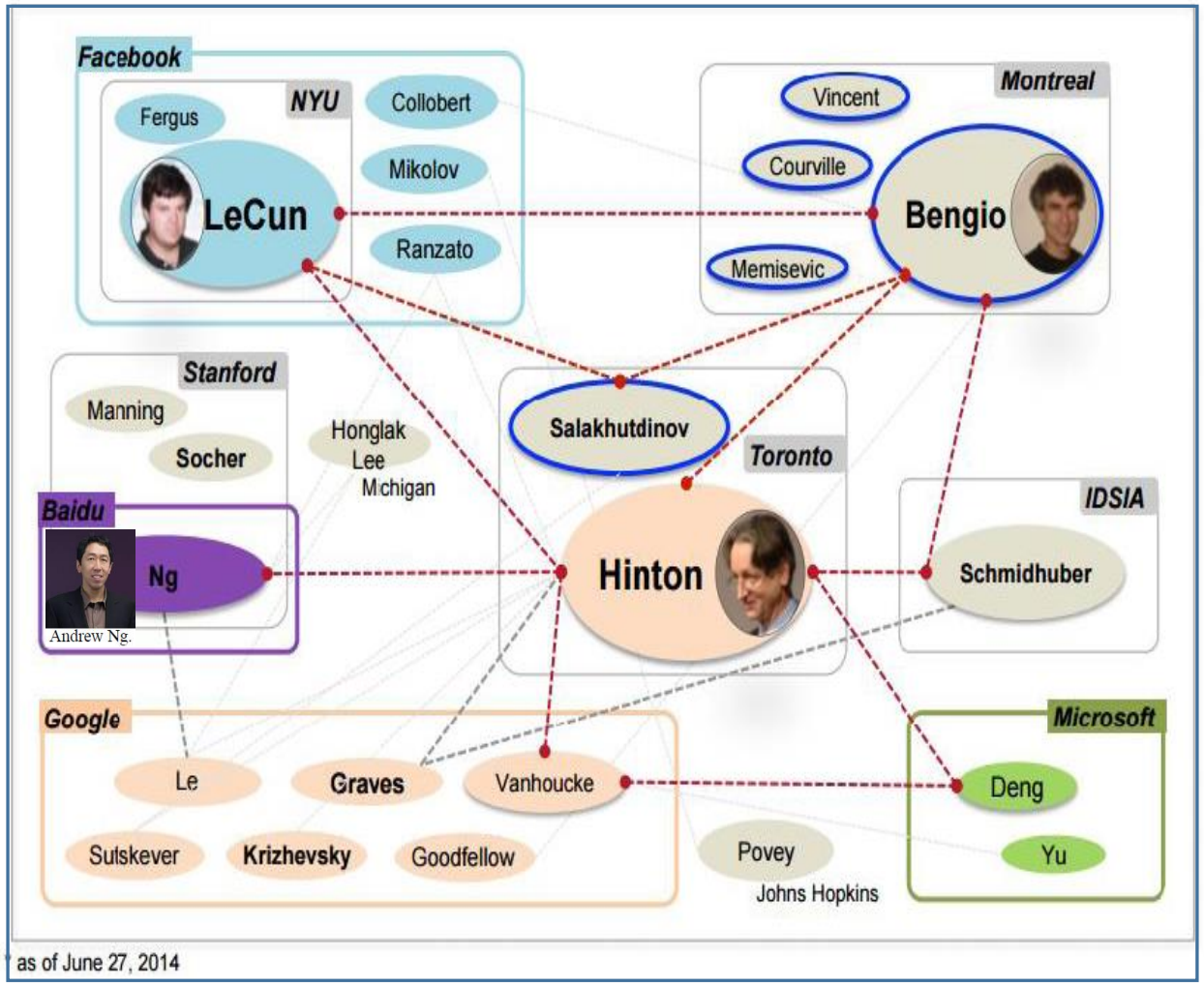
각종 Device들의 센서(Sensor) Data

AI (Deep Learning) 발전 원동력

획기적인 알고리즘

기업과 학계의 최고 전문 인력이 **새로운 ML, DL 기반의 AI 해법**을 개발, 제공

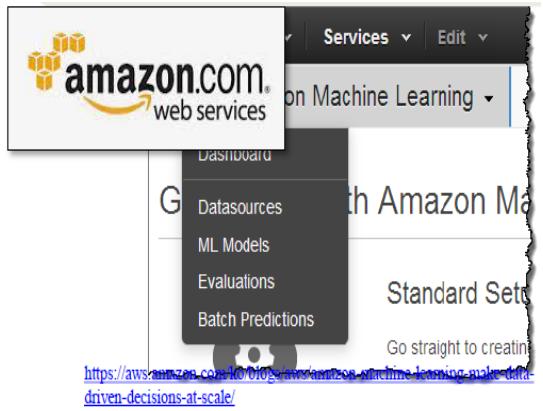


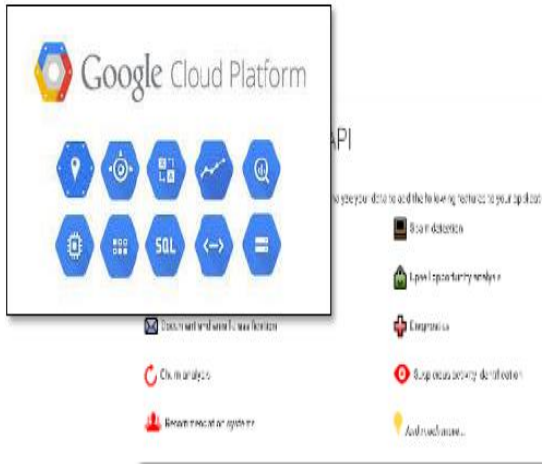


AI (Deep Learning) 발전 원동력

개발 환경 진화

[Cloud Based ML + DL] Platform





[Open Source Based ML + DL] Engine

Microsoft CNTK — https://github.com/Microsoft/CNTK

TensorFlow — https://github.com/tensorflow/tensorflow

theano — https://github.com/Theano/Theano

torch — https://github.com/torch/torch7

Caffe — https://github.com/BVLC/caffe

Machine Learning Concept

Machine Learning 개념

인공지능을
단순화해서 생각해보면.....

인공지능(AI)

인지(Cognitive)

Dog

Cat

입력(x)이 주어지면

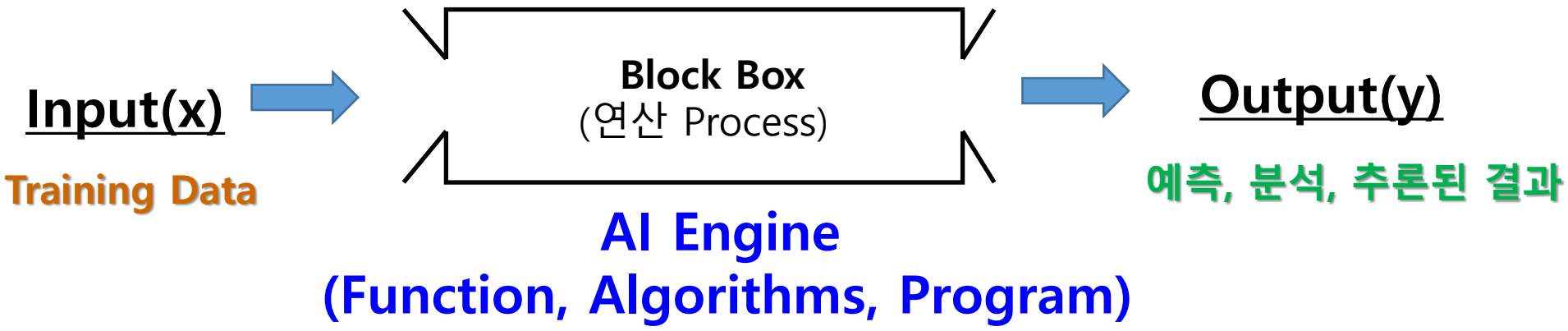
출력(y)을 내보낸다.

Machine Learning 개념

아주 기본적이고, 상식적인

입력(x)이 주어지면, **출력(y)**을 내보낸다.

단순 도식화 하면 !



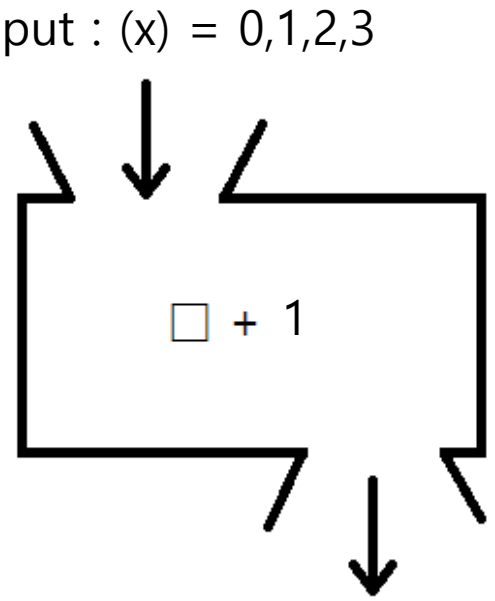
Machine Learning 개념

Block Box : f

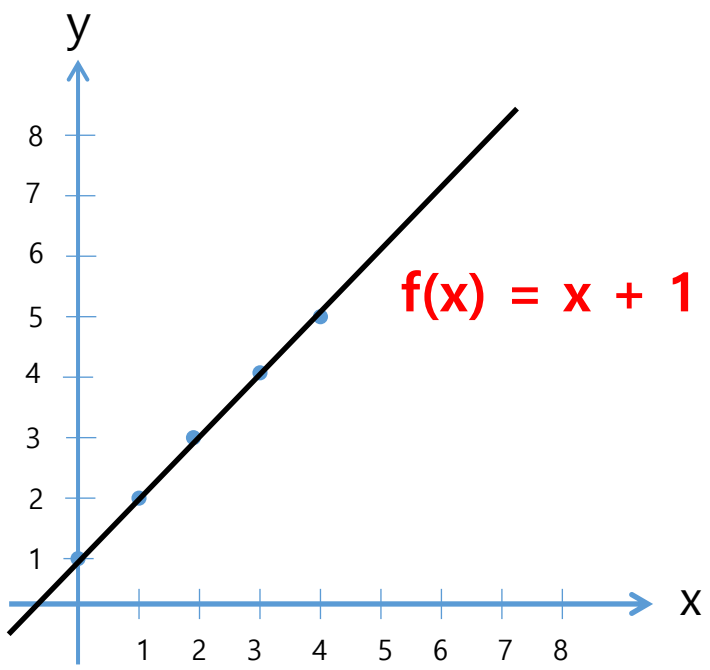
y = f(x)

f(x) = **w** * x + **b**

함수(Function)



그래프로 표현

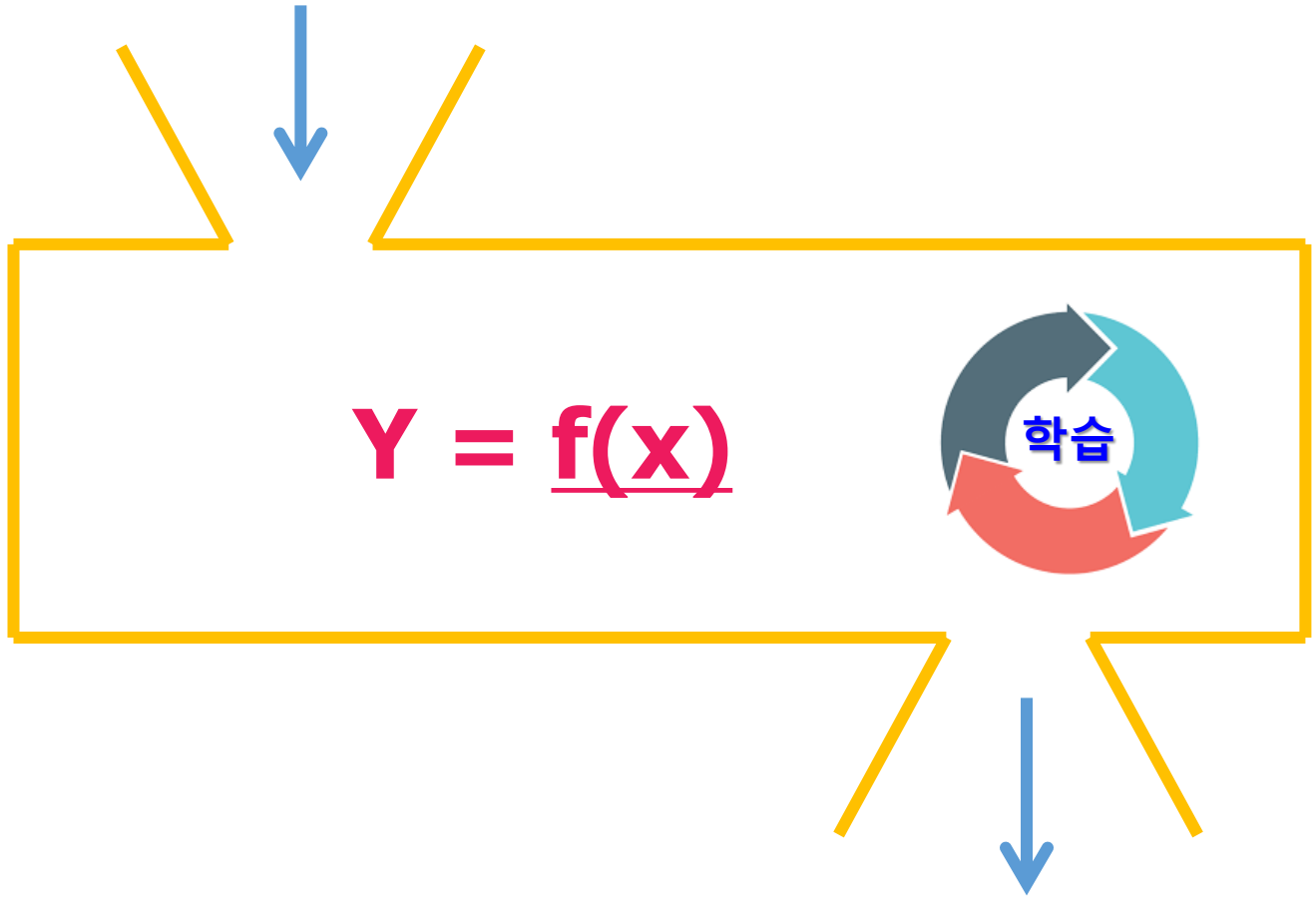


함수 (Functions)

Algorithms, Program 라고도 불립니다.

Machine Learning 개념

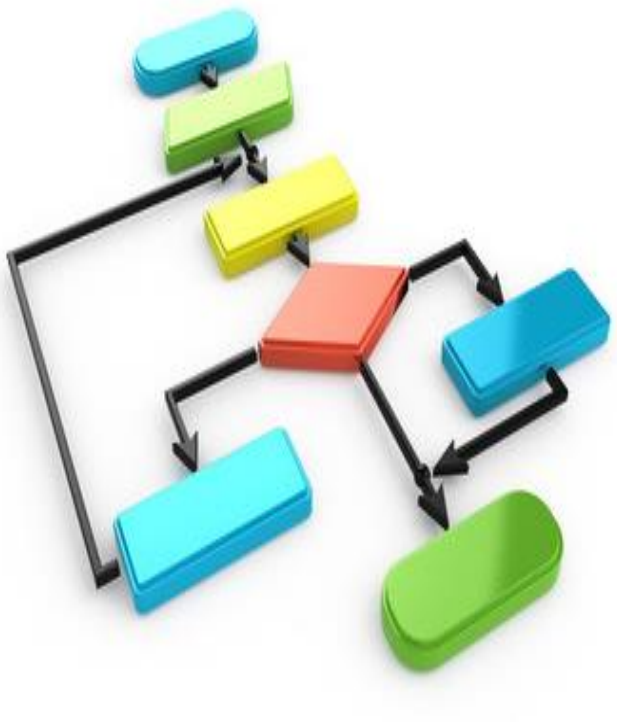
x_i : 여러 종류의 입력 데이터들(그림, 음성, 문자, 로그 데이터, 숫자,....등등) 입력



ML, DL Algorithms
➔ AI Program Engine

y : 새로운 입력 X에 관련된 "예측 값 Y는 땡땡이야" 하고
추론, 예측, 실행(Action)

Machine Learning 개념



Machine Learning, Deep Learning에서 **Algorithms** 의미?

원하는 형태로 모델(수학적 Procedure)을 정의/개발
이 모델을 통해 내가 가지고 있는 데이터를 어떻게 Learning할까?
즉 어떤 적당한 Algorithms을 사용하여 "Learning"할 건지 결정 要

Algorithms ?

주어진 **일(Task)**을 하기 위한 **"일련의 (수학적)처리 절차(단계)의 집합"**

알고리즘의 주요 요소로서
- 명확한 목적 (문제 해결을 위한)
- 입력 정의
- 결과 출력

ex : Shortest Path Algorithms

목적 : 최단거리 Path를 찾음
입력 : 시작점, 끝점, 경유 노드 수
출력 : Shortest Path

➔ **이런 것을 행할 수 있는 일종의 Procedure가 바로 Algorithms 임.**

머신 러닝(Machine Learning)이란

"머신 러닝 또는 기계 학습은 인공 지능의 한 분야로, 컴퓨터가 **명시적 프로그래밍 없이** **학습**할 수 있도록 하는 **알고리즘과 기술**을 개발하는 분야를 말한다." - 위키피디아

"Field of study that gives **computers** the ability to **learn** without being **explicitly programmed**." - Arthur Samuel, 1959

머신 러닝을 보다 **형식화**하여 정의하면 **"환경(Environment, E)"**과의 상호작용을 통해서 축적되는 경험적인 **"데이터(Data, D)"**를 바탕으로 **학습**하여, **"모델(Model, M)"**을 자동으로 구축하고 스스로 **"성능(Performance, P)"**을 향상하는 시스템 이다 (Mitchell, 1997)

Performance(P)

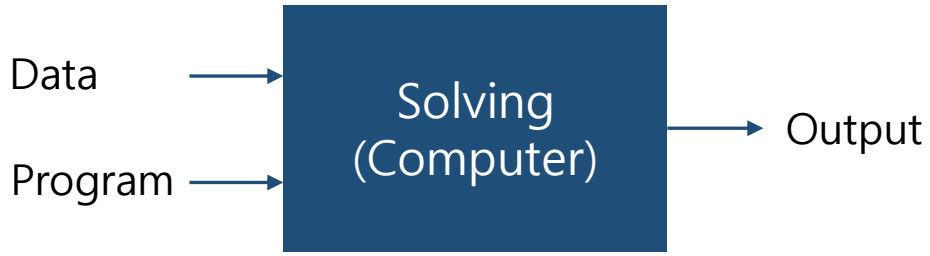
Machine Learning : Data(D) ⟶ Model(M)

일반 Programming 방식과의 차이점

- **Traditional Computer Programming**

  - 사람이 **일일이 명시적으로 알고리즘 설계 및 코딩(explicitly Programming)**

  - 주어진 문제(데이터)에 대한 **답** 출력



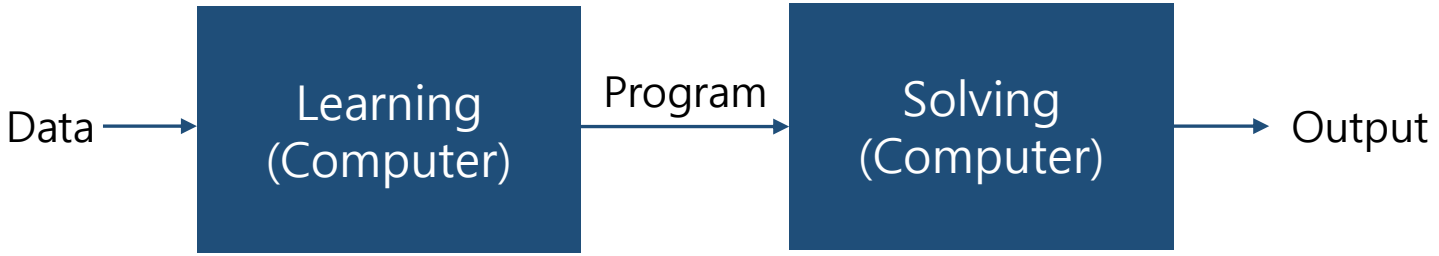
Human Programming

- **Machine Learning Programming**

  - 사람이 코딩

  - 기계가 **알고리즘**을 **자동 프로그래밍 (Automatic Programming)**

  - 데이터에 대한 **최적화된 Program**을 출력

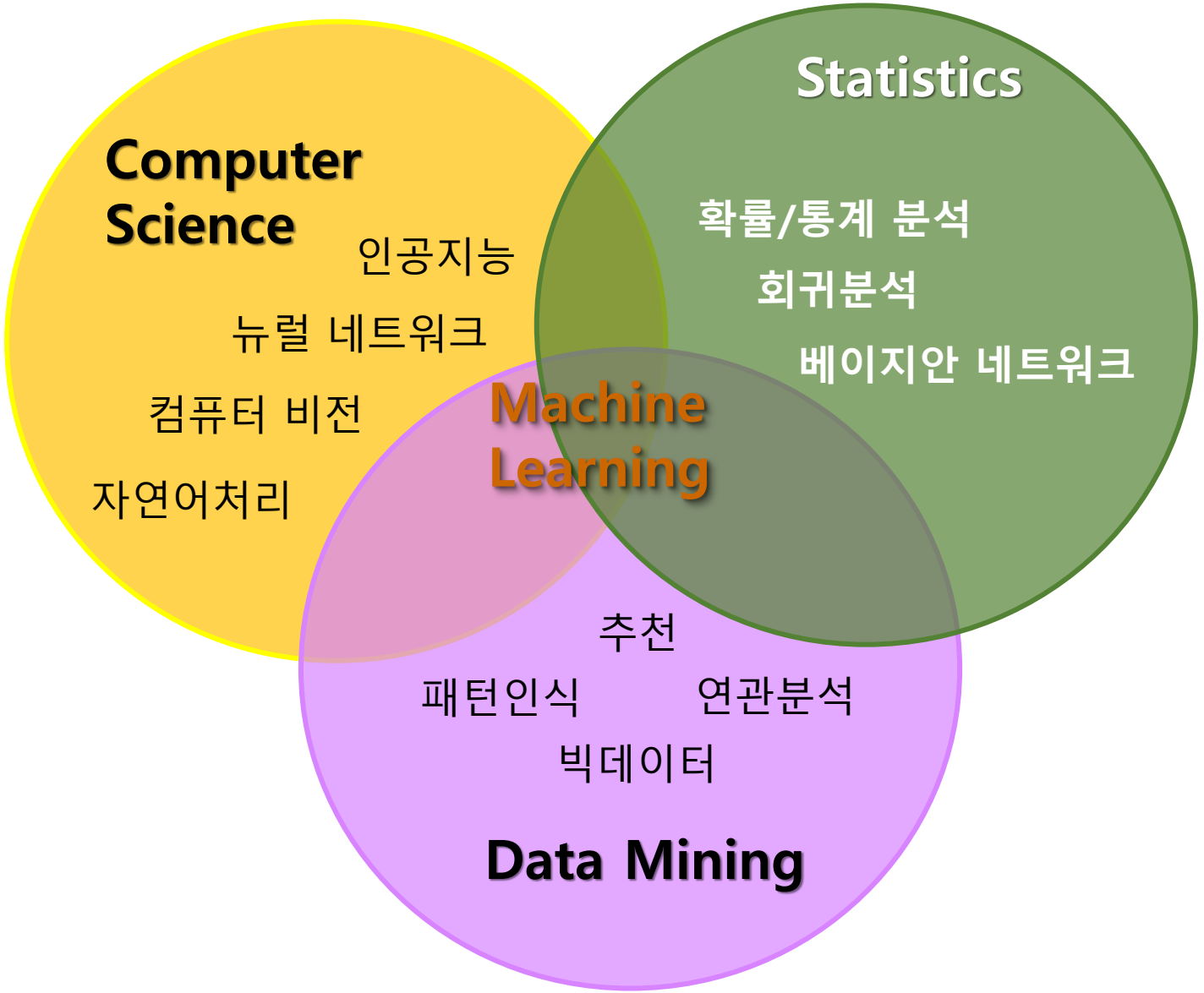


Automatic Programming(ML, DL)

Machine Learning 왜 중요한가?

- **프로그래밍이 어려운 문제들**
  - No explicit knowledge (명시적 문제 해결 부재)
  - Hard to encode by humans (음성, 화상 인식)
  - Environmental change (자율 주행 차, 로봇)
- **IT 환경의 변화**
  - Big data
  - Computing power
  - Algorithms power
- **직접적인 비즈니스 가치 창출**
  - User preference
  - Advertisement
  - Recommendation
  - Prediction, Analysis

리소스 분류


Computer Science
인공지능
뉴럴 네트워크
컴퓨터 비전
자연어처리
Statistics
확률/통계 분석
회귀분석
베이지안 네트워크
Machine Learning
추천
패턴인식
연관분석
빅데이터
Data Mining

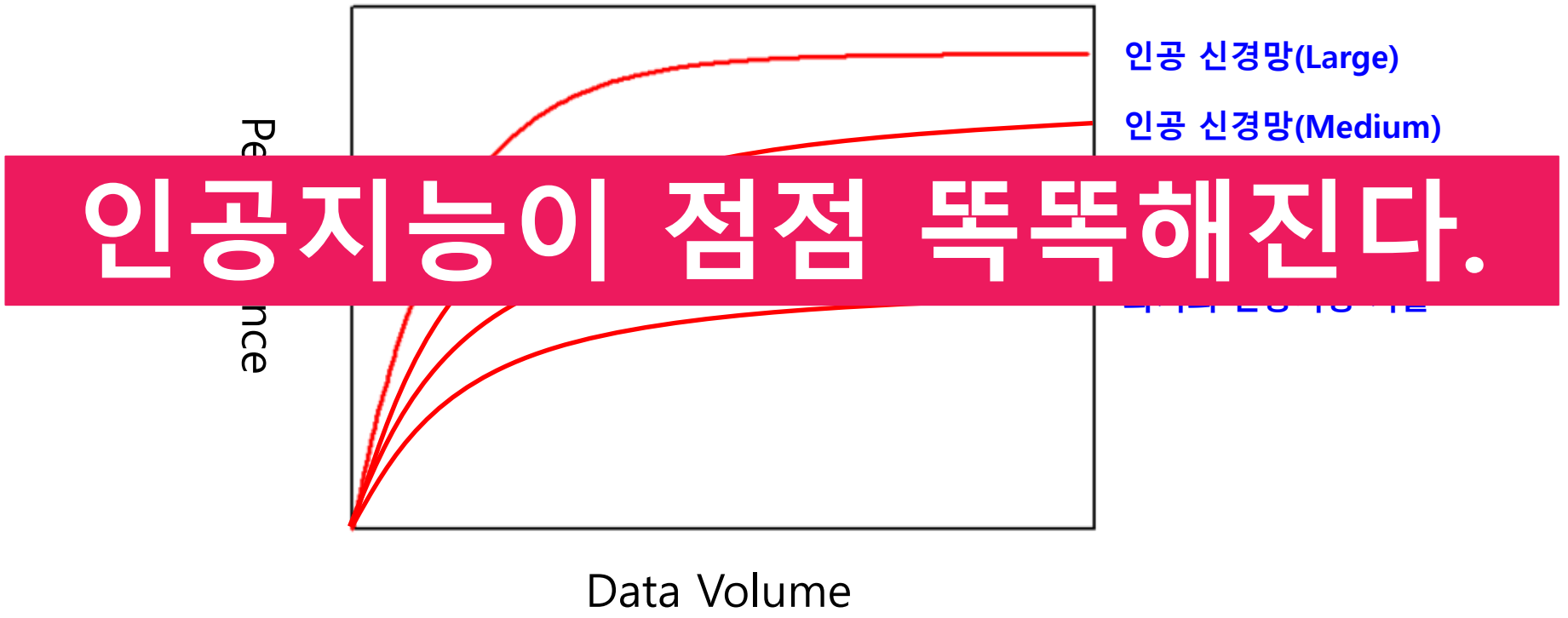
러닝(Learning) = 학습
→ Adaptation/Update

Function = Algorithms = Program

인공지능이 점점 똑똑해진다.

= training data Base (기존의 Email Data)

러닝(Learning) = 학습
→ Adaptation/Update



인공지능이 점점 똑똑해진다.

Machine Learning 개념

사람 정보 습득(학습) 과정

지식, 정보 : 수업, 책..

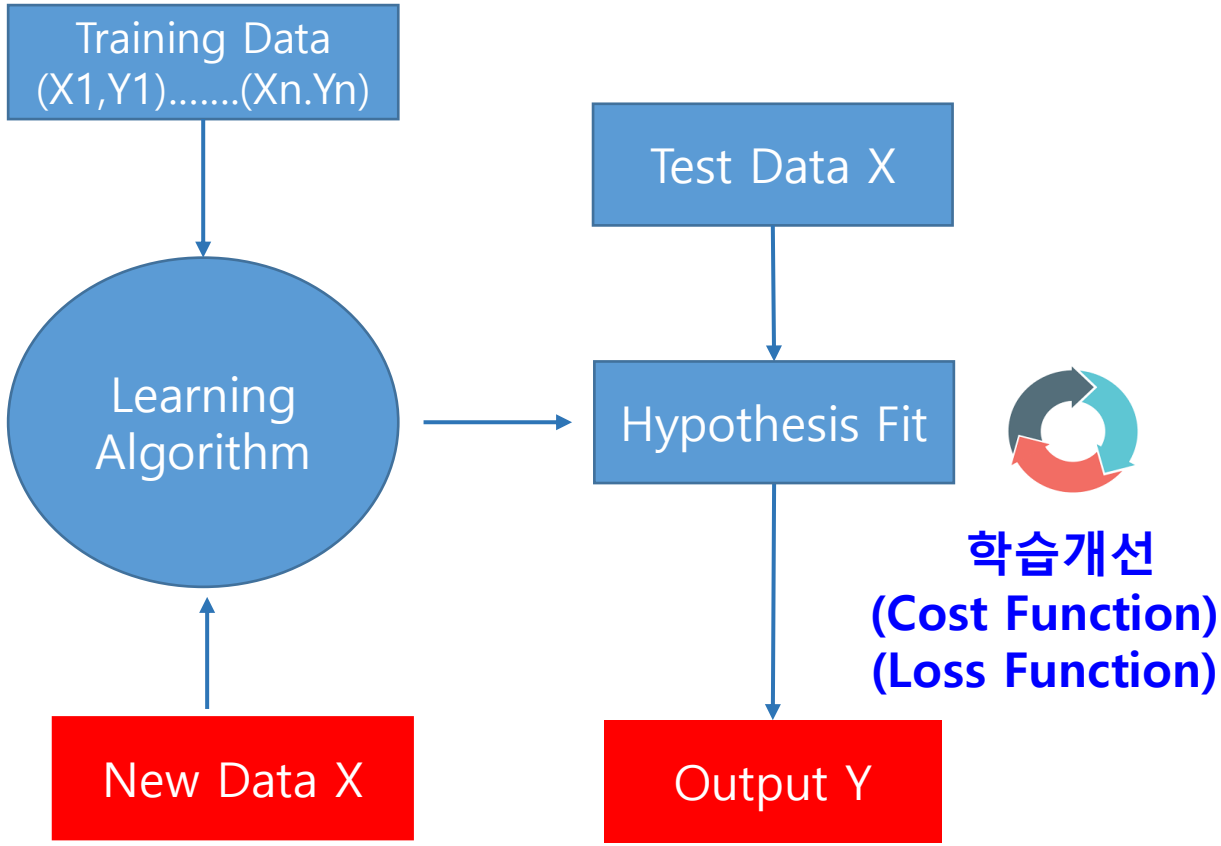
시험 : 모의고사

학습

학습 방법 개선

수능 합격

Machine Learning 과정

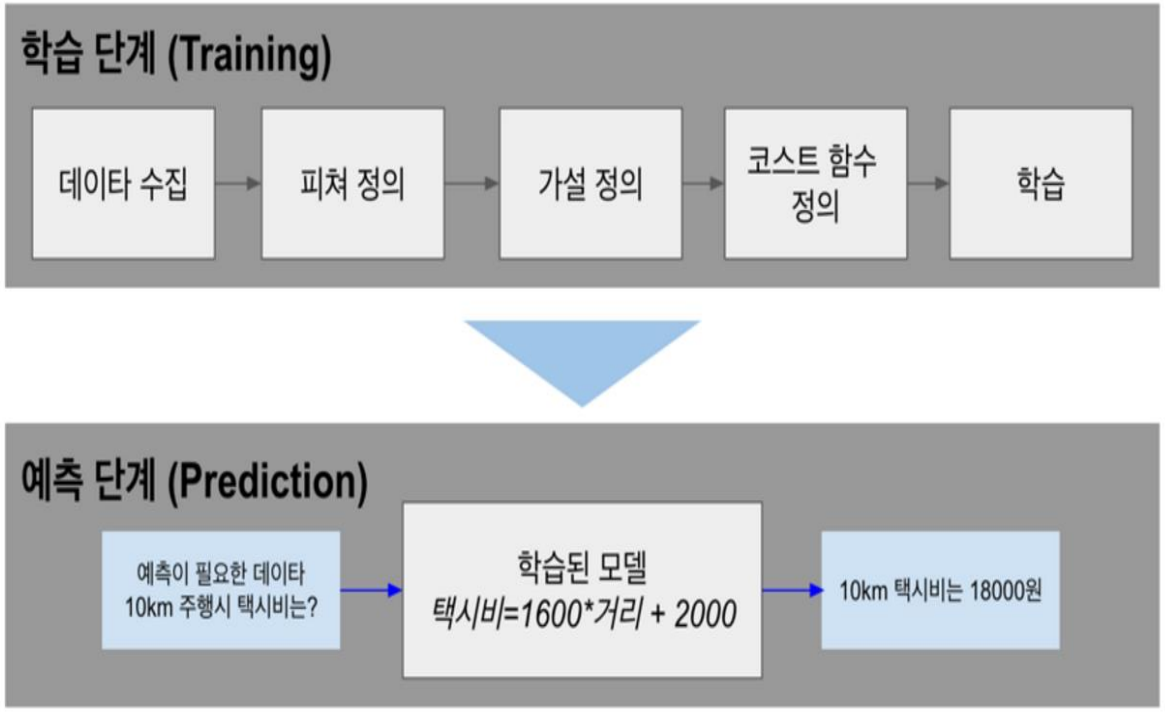


ML으로 문제를 풀기 위해 정의 사항

1. Input Data
   (지식, 정보, 교과서 : Training Data)
2. Output Data
   (시험 결과 : 예측 결과)
3. Target Function
   (수능합격 : 패턴분석, 예측)
4. Loss Minimize .. Algorithms
   (학습 방법 개선 : 최적의 알고리즘)

머신 러닝의 순서

기본 개념은 데이터를 기반으로 해서 어떤 가설(공식:함수, 알고리즘)을 만들어 낸(선택) 다음, **그 학습에서 나온 값(예측 값)**이 **실제 값과의 차이(Cost or Loss 함수 정의 : 에러, 오차)**가 **최소한의 값(0에 수렴)**을 가지도록 **변수(Weight, bias)에 대한 값**을 컴퓨터가 자동으로 찾은 후**(최적화, 이 과정을 학습)**, 이 찾아진 값**(최적화된 W, b 값)**을 가지고 **학습된 모델(최적화된 알고리즘)**을 **정의(Fix)**해서 **최종 목적인 새로운 입력에 대한 결과 예측, 추론**을 수행 하는 것.



학습 단계

예측, 분석 모델을 만들기 위해서, **실제 데이터를 수집**하고, 이 수집된 데이터에서 **어떤 특징(Feature, 상관관계, 패턴)**를 가지고 예측 할 것인지 **데이터 특징(Feature)들을 정의**한 다음에, 이 데이터 특징을 기반으로 예측을 할 **가설을 정의(함수, 알고리즘 선택)**하고, 이 **가설을 기반으로 데이터를 학습** 시킴.

예측 단계

학습이 끝나면 **최적화된 학습모델 (함수, 알고리즘)**가 주어지고, 예측은 단순하게, 이 최적화 된 모델(함수, 알고리즘)에 예측 하고자 하는 새로운 값을 입력, **학습된 모델에 의해서 결과 값을 예측**.

Machine Learning 개념

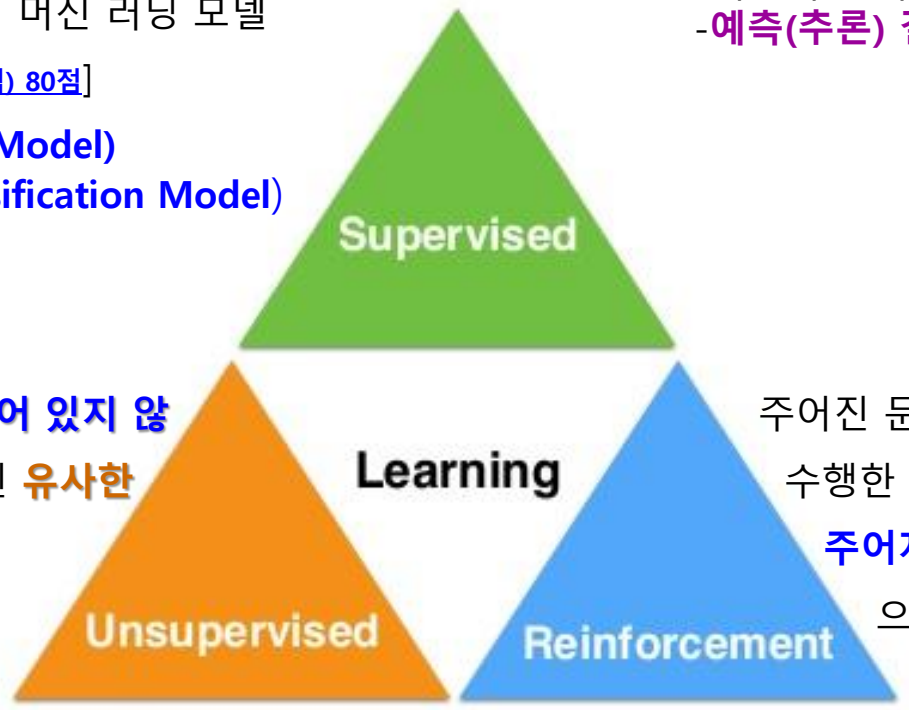
머신 러닝 3가지 타입 (학습 방법)

지도학습

이미 **정답**을 알고(다른 의미로는 **레이블이 되어 있는 데이터**) 있는 **Training Data(학습 데이터)를 이용**하여 머신 러닝 모델 학습시키는 방식 [X_i(공부시간) 2시간 = **Y_i(시험성적) 80점**]

ex : 시험 성적 예측, 주가 예측 **(Regression Model)**
Image Labeling, 스팸 메일 필터링 **(Classification Model)**

-**분류된 데이터(Labeled Data)**
-직접적인 피드백
-**예측(추론) 결과 (미래)**



비지도 학습

정답을 모르는 Training 데이터(즉 **레이블 되어 있지 않은 데이터**)를 이용, 학습하여 데이터에 내재된 **유사한 특성, 패턴, 구조** 등을 발견, 분석

ex : 구글 뉴스 그룹핑(비슷한 뉴스 그룹핑)
Word clustering (비슷한 단어끼리 **Clustering Model**),
Security

-**분류가 안됨(No Labels)**
-피드백 없음
-**숨겨진 구조 찾기(특징, 패턴, 군집)**

강화학습

주어진 문제의 답이 명확히 떨어지지 않지만,알고리즘이 수행한 결과에 따라서 **보상(Reward)과 손실(Penalty)이 주어져**, 이를 통해 **보상(Reward)을 최대화 하는 방향**으로 진행하는 학습 방법. **시행착오(trial and error)적인 방법**으로 적합한 행동을 탐색함

ex : **게임, 제어 전략**, 금융 시장의 매매 전략,
Robotics, 자율(Autonomous) 주행 차

-의사결정 프로세스
-**보상 제도**
-**행동(Action)시리즈 배우기**

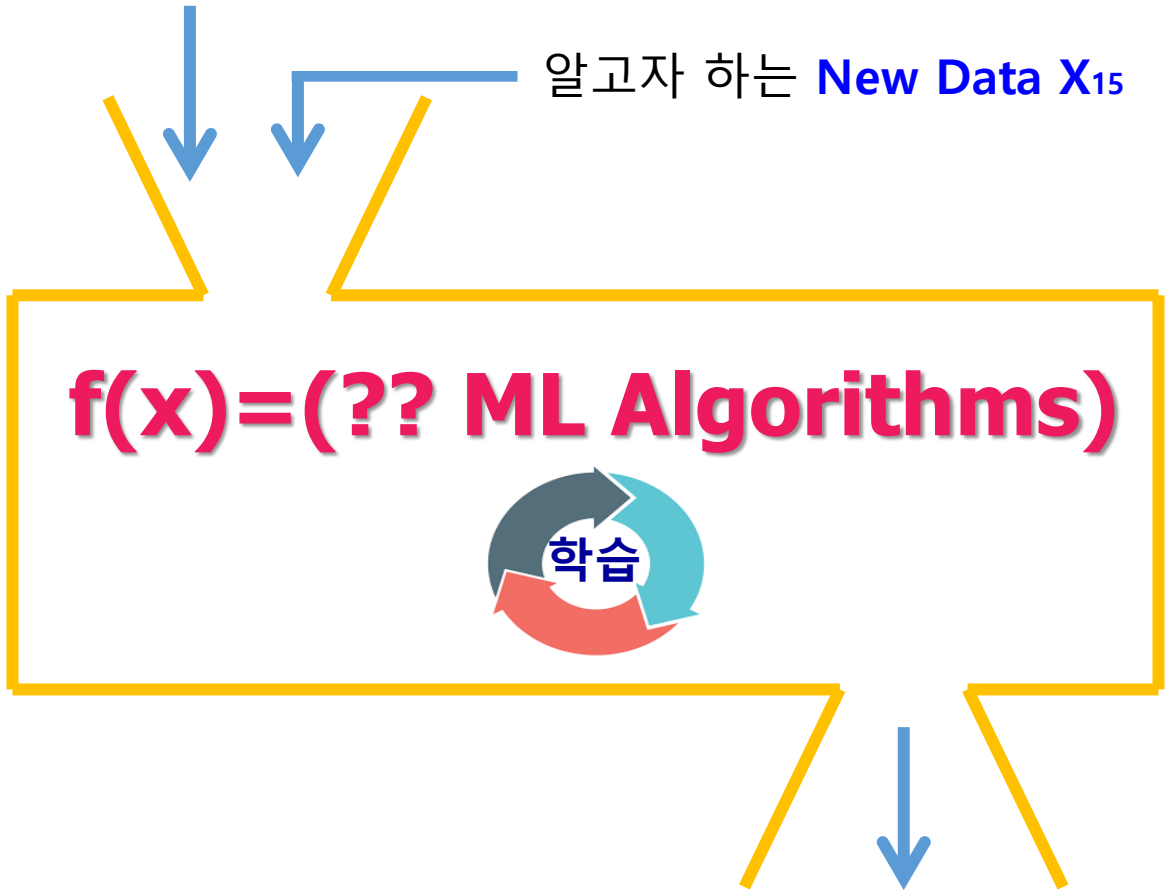
Machine Learning 개념

Supervised Learning (지도 학습)

(X_1, Y_1) (X_2, Y_2) .....(X_{10}, Y_{10}) : Training Data Set **(Y Value 有 : Label 有)**

- Learning **with labeled** examples : **training data set**
- Most common problem type in ML

  - Predicting exam score : **(linear Regression)**
    learning from previous exam score and time spent
  - Image labeling : **(classification)**
    learning from tagged imaged
  - Email spam filter :
    learning from labeled(spam or ham) email



- 정답이 주어진다.
- (비교적)문제 풀이가 쉽다

Machine Learning 개념

Supervised Learning (예시 : Linear Regression Model)

어떤 New Data X = 15

input

Xi(평수)	yi(시세)
1	100
10	200
20	300
30	400

Label (정답)

Training Data Set

ML-Learning Algorithms

H(X) = W * X + b

ML이 학습을 통해 어떤 모델(LRM)이 생겨남.
(최적화된 알고리즘 FIX 함 = w, b 찾음)

y = 250 (새로운 데이터 X_{test}에 대한 예측)

Supervised Learning(예시 : image label)

An example training set for four visual categories.

Y값(label) = cat, dog, mug, hat

X값=image

Training Data Set

새로운 image X_{16}=  , X_{19} = 

ML-Learning Algorithms

학습

Y_{16} = dog, Y_{19} = cat 예측(추론)

Supervised Learning(Model Type)

- Predicting final exam score(**연속 값**) based on time spent
  ➔ **regression Model (Algorithms)**

0점
.
.
.
100점

- Pass/non-pass based on time spent
  ➔ **binary classification Model**

Pass (1)

Non-pass (0)

- Letter grade (A,B,C,D,E and F) based on time spent
  ➔ **multi-label classification Model**

A학점
B학점
C
D
E
F

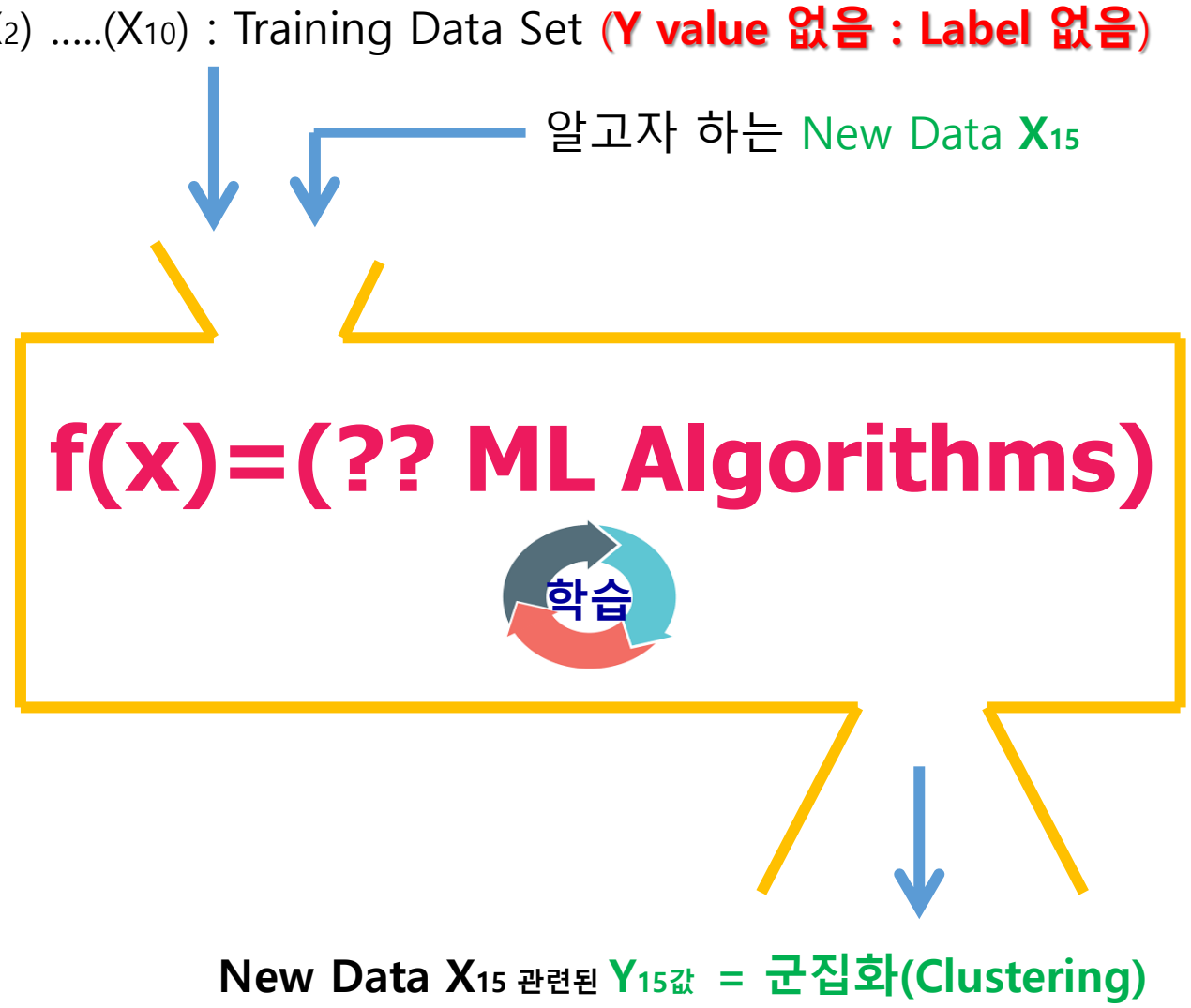
Machine Learning 개념

Unsupervised Learning

- Learning **without labeled** examples : **training data set**
- Data들 간의 내재된 **특징, 패턴, 구조 파악해서 분류, 군집화**
- Most common problem type in ML

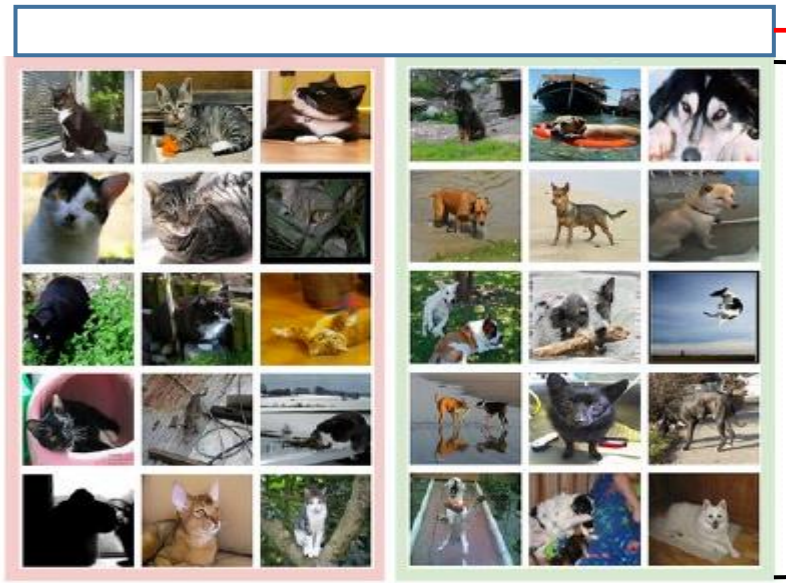
  - Google news grouping
    :비슷한 뉴스끼리 Grouping (정치, 연예, 경제, 사고..)
  - Word clustering
    : 유사한 텍스트 토픽 Grouping
  - **Anomaly Detection**
    : 보안분야 이상행위(패턴) 탐지

- 정답이 주어지지 않는다
- 문제 풀이가 어렵다



Machine Learning 개념

Unsupervised Learning (예시 : Clustering)



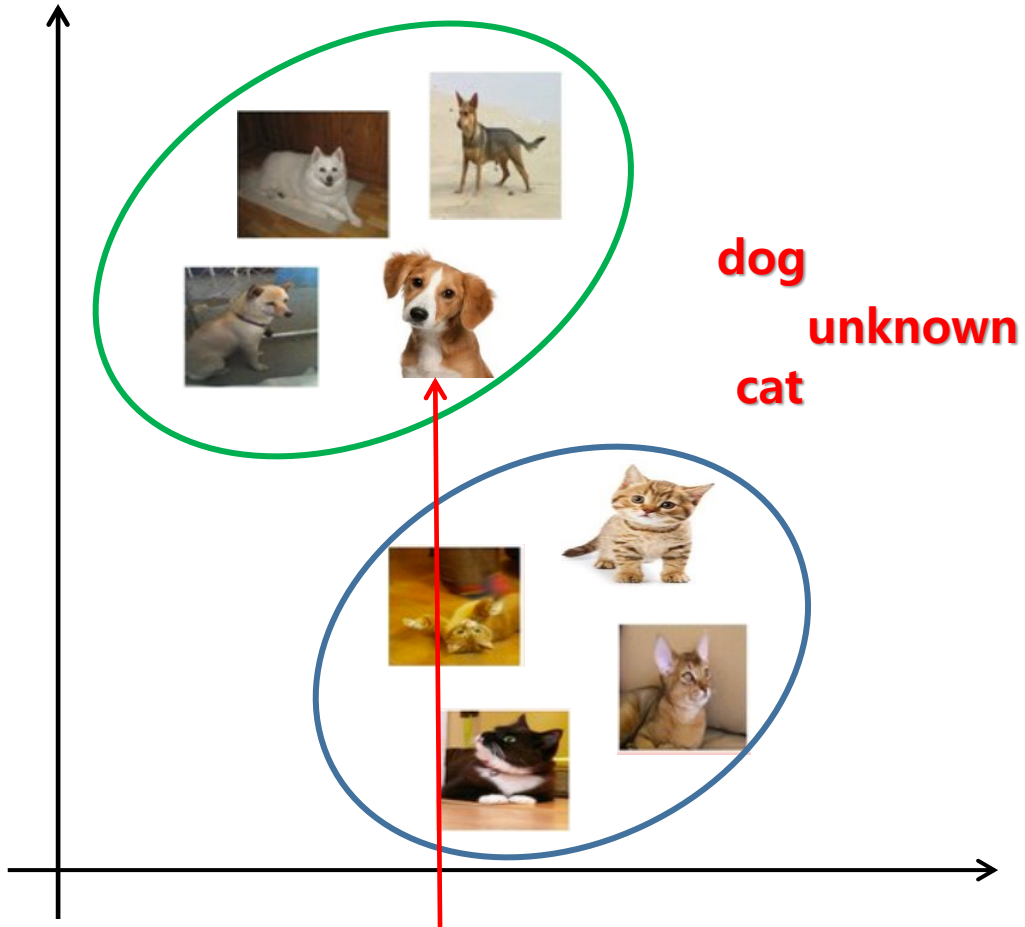
Y : Label
없음

X : Training
Data Set

New Data X= 

ML-Learning Algorithms

학습



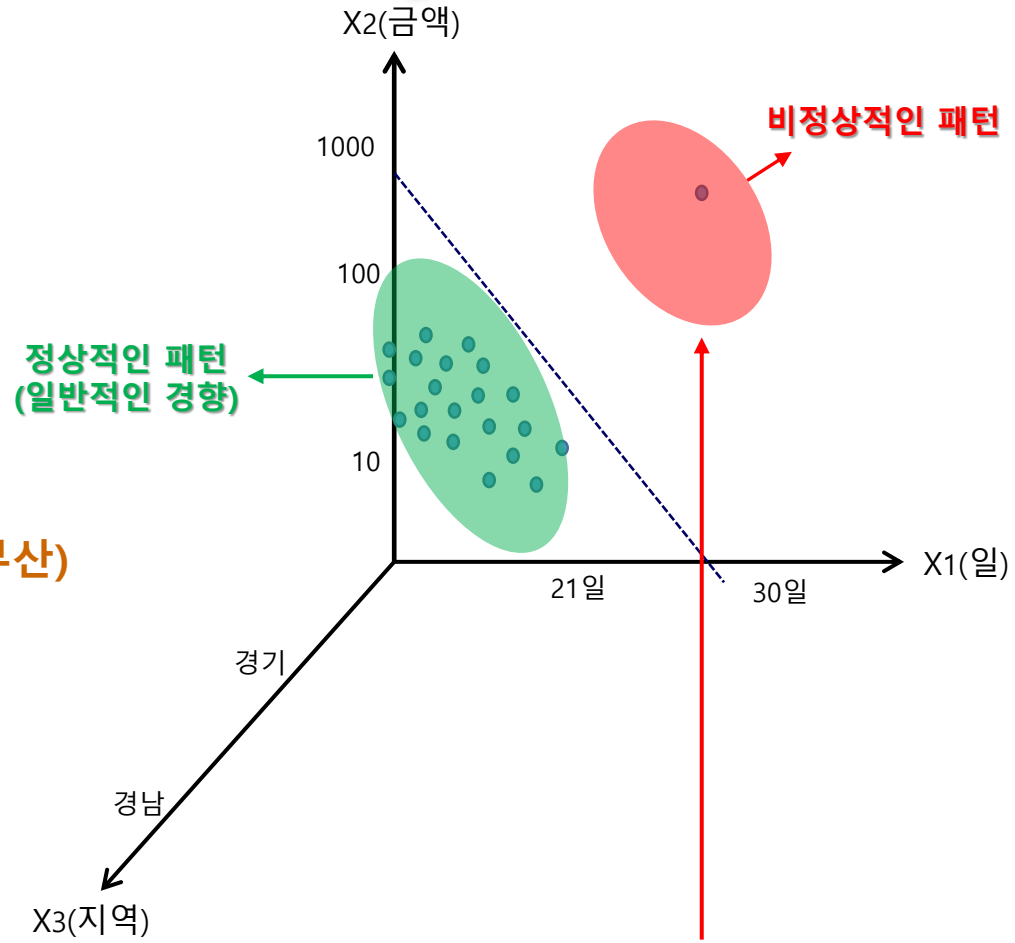
New Data X에 대한 Output Y =  dog인지는 모르나 이쪽

Machine Learning 개념

Unsupervised Learning (ex : Clustering)

X1(인출 일자)	X2(인출 지역)	X3(인출 금액)
20,	수원,	10,
21,	수원,	10,
22,	서울,	10,
21,	인천,	50,
20,	서울,	20,
21,	서울,	30,
23,	안양,	50,
19,	인천,	100,
21,	인천,	100,
21,	안양,	20,
.....	.....	.....

Xi : (X1, X2, X3)
Training Data Set
X와 관련된 **label된**
Y1, Y2, Y3 data 없음
(정답이 없음)



New Data (X1=30일, X2=1000만원, X3=부산)

ML-Learning Algorithms

학습

New Data X1,2,3에 대한 Output Y= "비정상 패턴" 군집화

25일 이후 or 전혀 다른 지역 or 너무 많은 금액 인출

이상거래 탐지 시스템(FDS)
=Anomaly Detection (Security 분야)

= **불법**이라 **단정**(**오탐 확률 존재**)할 수 없으나 **인출 패턴(이상 경향)이 이상** 한 거래로 인식

= **오탐(거래 중지)을 줄이고자, Protection 개념**으로 보다 자세한 인출단계 요구 예 : **"비밀번호"**외에 **추가로 "주민번호" 입력 要**

Algorithms

Supervised Learning	Classification Model	Binary Classification
		Multinomial Classification
		Support Vector machine
		Decision Tree
	Regression Model	Linear regression
		Locally weighted linear regression
		Ridge
		Lasso
Unsupervised Learning	Clustering Model	K means
		Density estimation
		Expectation maximization
		Pazen window
		DBSCAN

Reinforcement Learning

" Optimal Control"
(Trial and Error)
(시행착오)

어떤 모르는 환경에서 어떻게 **"행동(Action)"** 하면

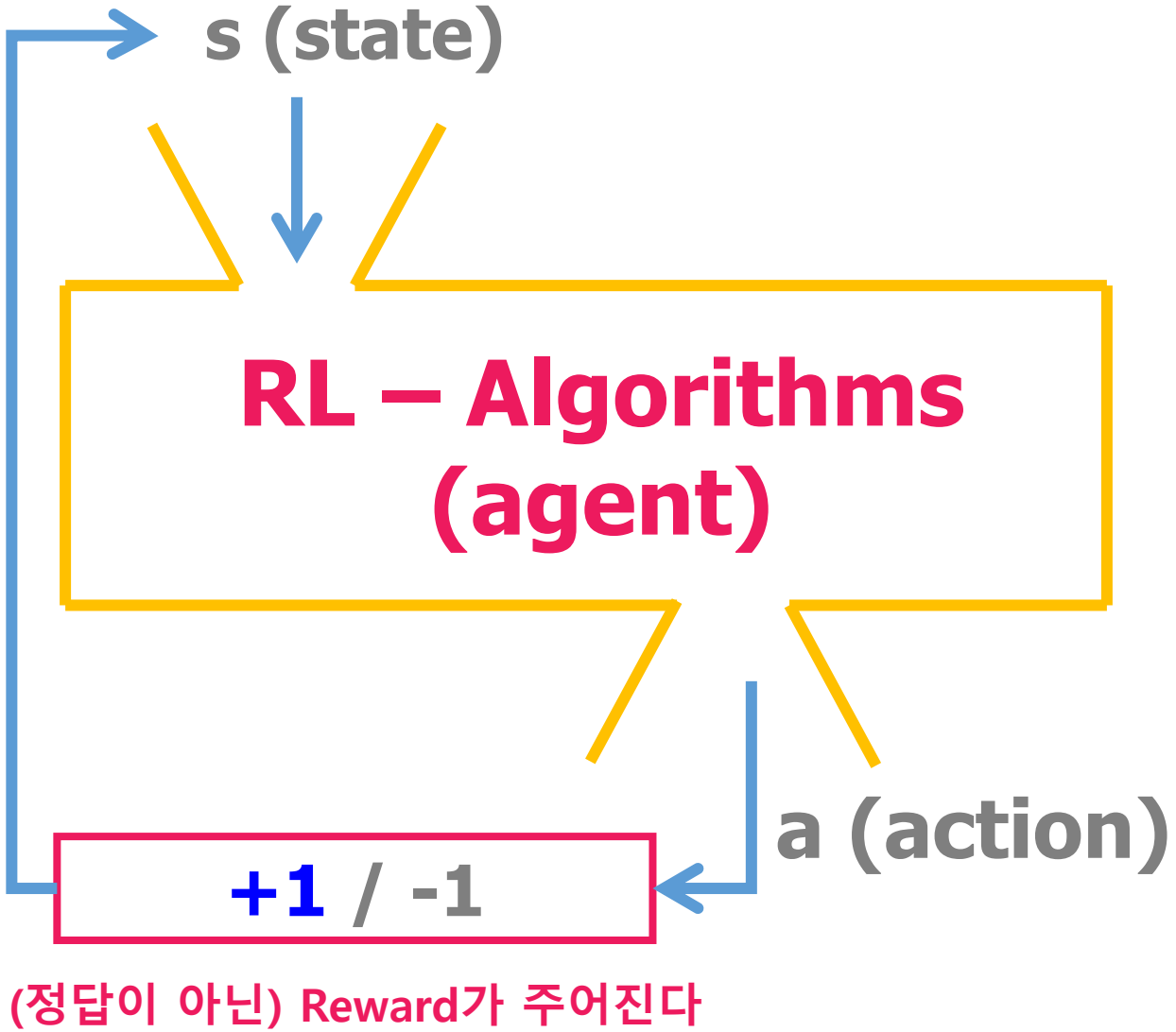
"상(Reward :+1)"을 **더 많이 받을 정책(Policy)방향**

으로 " **학습하는 것이 핵심**

➔ 에이전트가 앞으로 **누적될 포상**을 **최대화 하는 일련의**

행동으로 정의되는 최적의 정책(Policy)을 찾는 방법

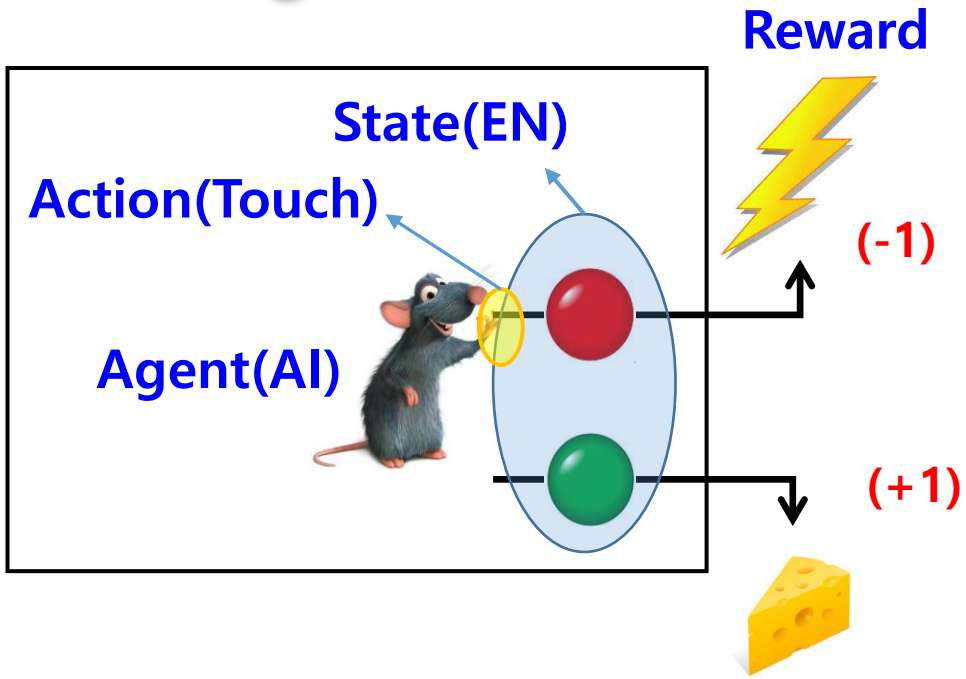
(최적의 의사 결정 알고리즘)



Reinforcement Learning

- Learning **without labeled** examples : **training data set**

  어떤 모르는 환경에서 **정답은 모르지만 Agent(주체)**가 변화하는 **환경과 자신의 상태(State)**와의 **상호작용 (Action ←→Reward)**을 통해서, 즉 자신이 한 **행동(Action)**에 대한 **"보상(Reward)"**을 받으면서 최종적으로 특정 목적(예: 로봇 걷기)을 수행할 수 있게 끔 **모든 보상(Reward)의 합**이 **가장 좋게 만드는 Policy(일련의 행동 들)**를 **Self Learning 방식**(의사 결정 알고리즘)



- Most common problem type in ML

  - **Robotics**
    : Humanoid robot walk

  - Business operation
    : Inventory management: how much to purchase of inventory, spare parts

  - **Finance**
    : Investment decisions, portfolio design

  - **Game strategy**

  - E-commerce / Media
    : What content to present to users
    : What ads to present to users (avoiding ad fatigue)

좋은 행동을 했을 경우에 **좋은 반응**이 **환경**으로부터 옴**(Reward:+1)**

Machine Learning 개념

Reinforcement Learning

➢ **문제의 정의**

이 **AI 로봇**은 **보석**을 얻기 위해 **어떻게 해야 할 지를 학습**
(최적 행동을 선택하는 **의사결정:최적의 정책**을 찾는 것)

Definition

A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states → AI 로봇(Agent)이 있는 현재 위치(상태)
- $\mathcal{A}$ is a finite set of actions → AI 로봇이 앞,뒤,좌,우로 이동하는 행위
- $\mathcal{P}$ is a state transition probability matrix, → AI 로봇이 특정 t라는 시점의 state에서 action을 했을 다음 상태인 s'에 도착할 확률(전이 확률 행렬)

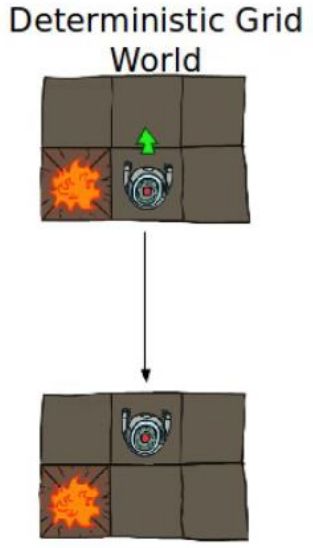
$$\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

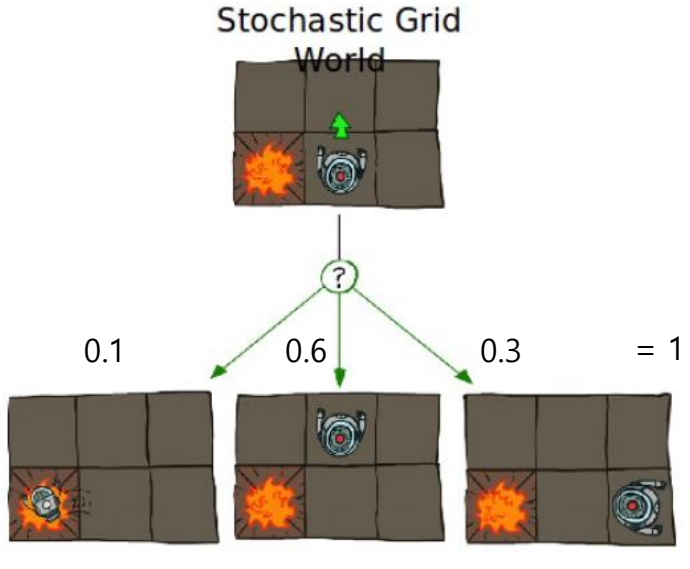
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
  action을 취했을 때 "Environment"가 알려주는 보상(+1)
- γ is a discount factor $\gamma \in [0, 1]$.
  0에서 1사이 값으로 reward에 대한 현재, 미래의 가치평가

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

즉 각 상태(state)에서의 선택할 최적의 행동(action) Policy를 결정

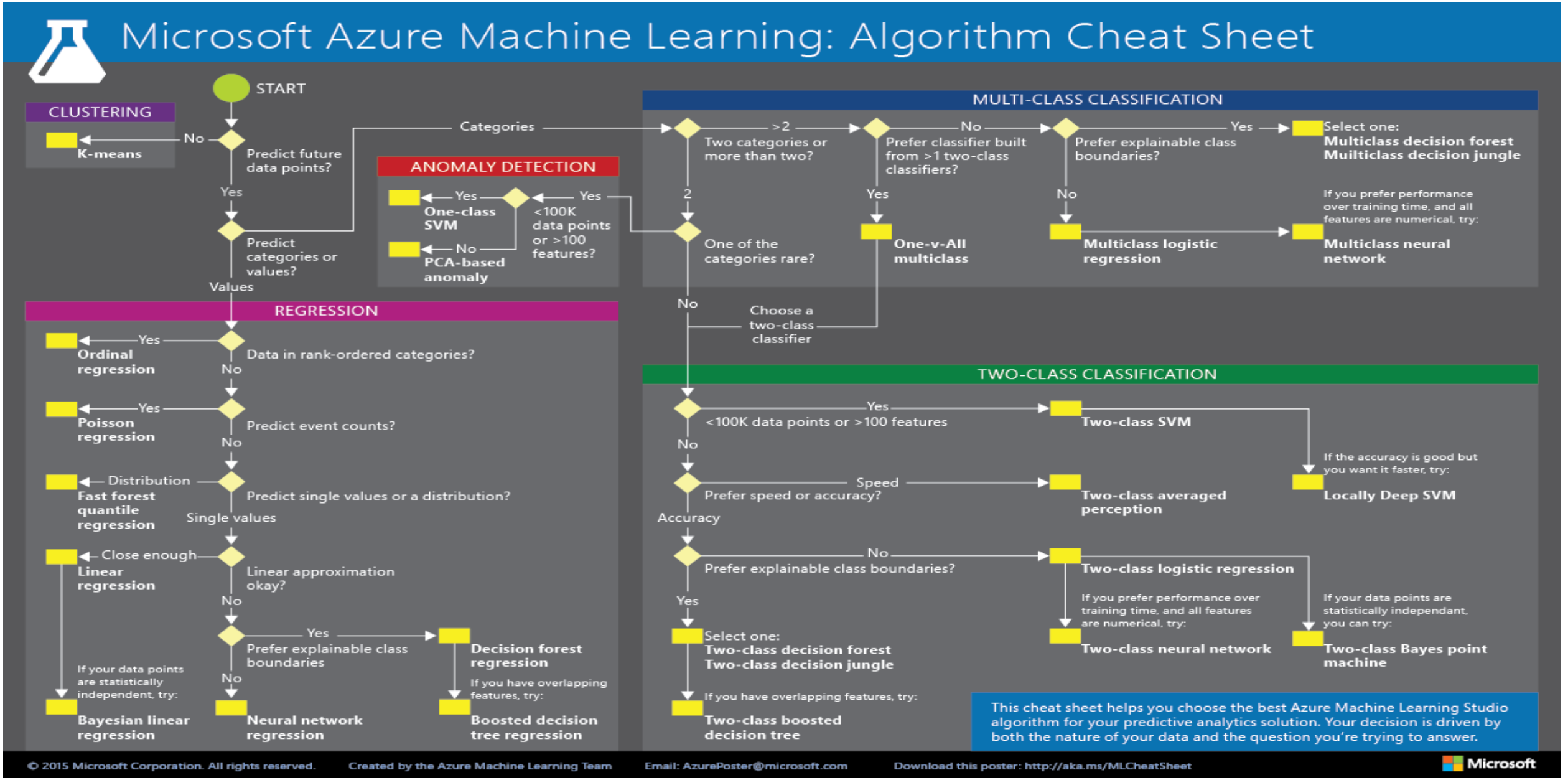
Markov Decision Processes





$$\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

머신 러닝 알고리즘 참조도



Machine Learning 개념

대표적 알고리즘

- Machine Learning
  - Supervised Learning
    - **선형 회귀 분석 (Linear Regression)**
      : Continues 한 Value 예측(실수)
    - **로지스틱 회귀 분석 (Logistic Regression)**
      : Classification
    - **Multinomial Classification**
  - Unsupervised Learning
    - **K means**
      : Clustering
  - Reinforcement Learning
    - **Deep Q-Learning**
- Deep Learning
  - **Neural Network, Convolution neural network**, RNN,,,

Machine Learning Principle
(Linear Regression Model)

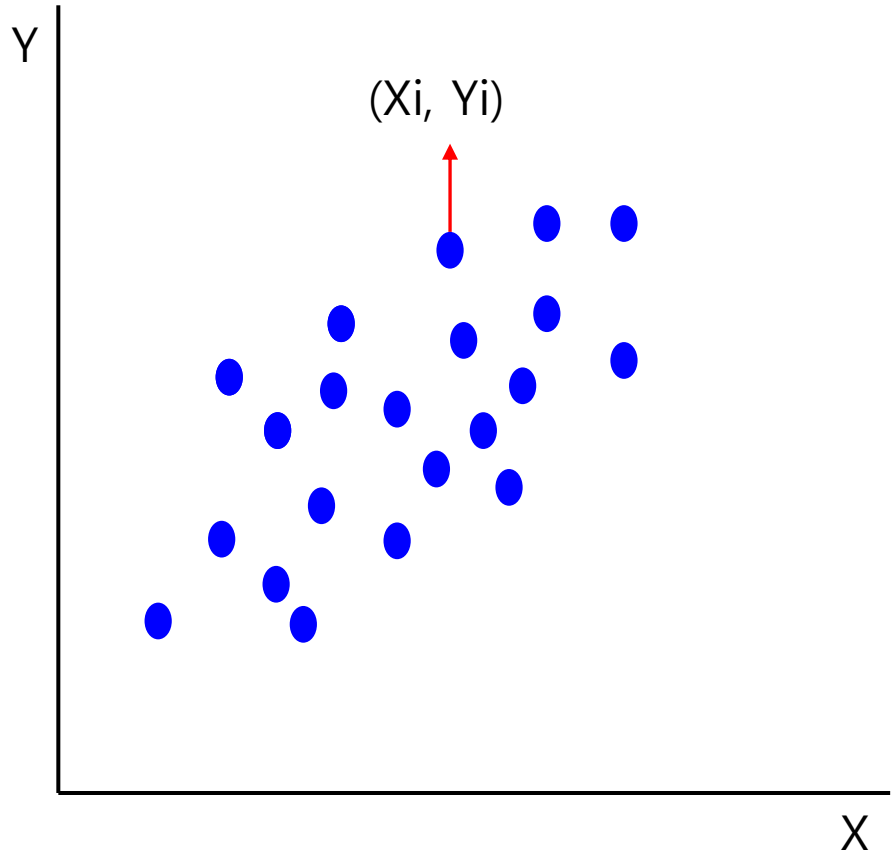
ML Principle (Linear Regression Model)

Linear Regression (Supervised Learning)
(선형) 회귀분석

Tabular Form

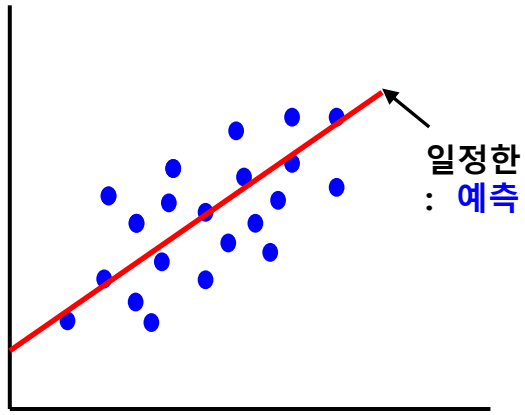
Xi(평수)	Yi(시세)
2	100
5	150
10	190
20	250
. . .	. . .
Xn	Yn

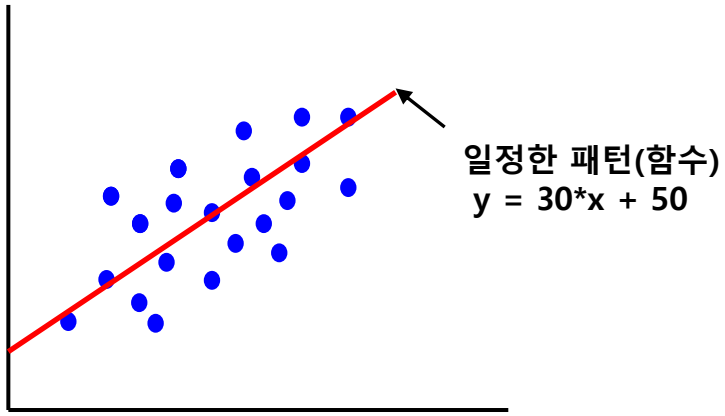
Graphical Form

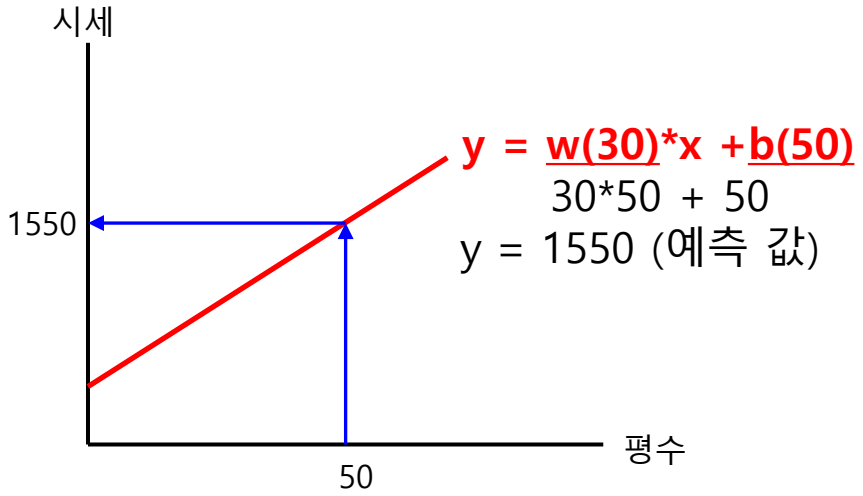


ML Principle (Linear Regression Model)

Linear Regression (Supervised Learning)
(선형) 회귀분석







회귀 분석
: 데이터 표본으로 부터 얻어낸 결과가 "이럴 것이다 " 라고 추리한 예측 값

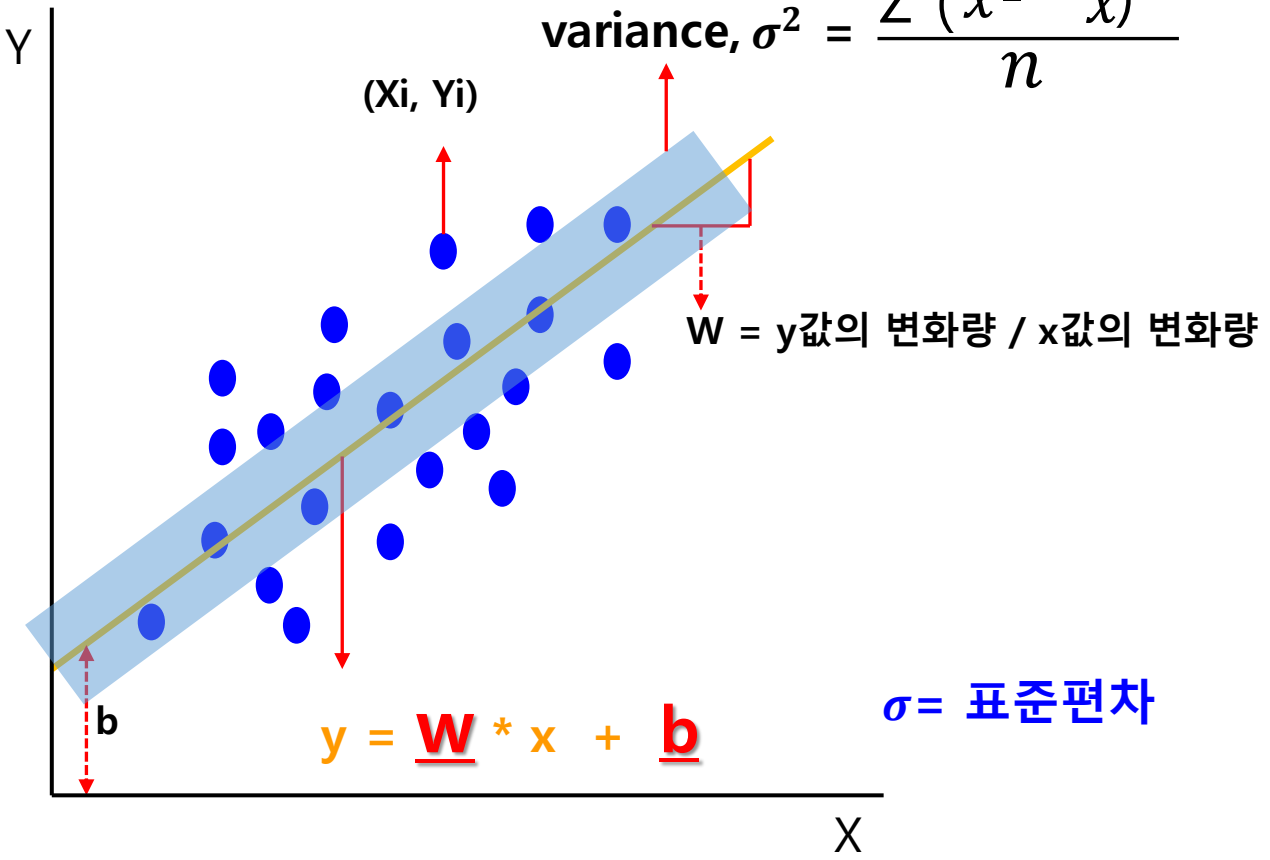
ML Principle (Linear Regression Model)

Linear Regression (Supervised Learning)
(선형) 회귀분석

Tabular Form

Xi(평수)	Yi(시세)
2	100
5	150
10	190
20	250
. . .	. . .
Xn	Yn

Graphical Form



Variance(분산, σ^2)

이런 데이터들이 **평균을 기준으로 퍼져있는 정도**, 즉 **분포 특성을 나타내는 수치로서 원래 데이터 값과 평균값의 오차**.
분산이 클 수록 데이터의 분포도가 평균에서 들쭉날쭉 불안정하다, 즉 **오차가 많다**는 의미

ML Principle (Linear Regression Model)

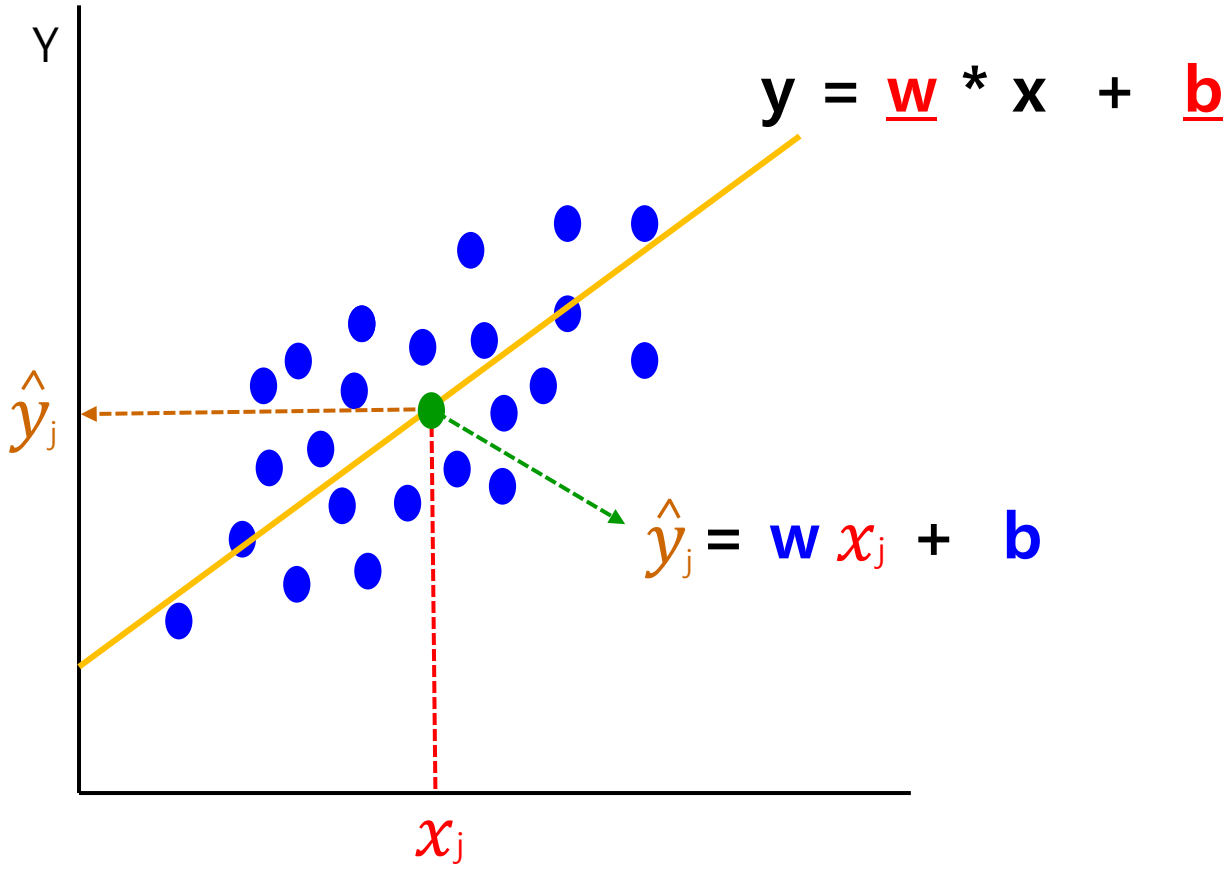
Linear Regression (Supervised Learning)
선형회귀분석

Tabular Form

Xi(평수)	Yi(시세)
2	100
5	150
10	190
20	250
x_j	$\hat{y}_j(?)$
. .	. .
Xn	Yn

New Data

Graphical Form



ML Principle (Linear Regression Model)

Linear Regression (Supervised Learning)
선형회귀분석 (회귀식 구하기)

표본으로 추리한 "예측 값"

$$\hat{y} = \underline{w}\,x + \underline{b}$$

기울기 w, 절편 b

➔ w, b 값을 알아야 예측($\hat{y}$)을 할 수 있다.

(기울기 w를 알아야, y 절편 b를 구함)

변수 x 값의 평균, 변수 y 값의 평균

$$w = \frac{\sum (x - \overline{x})\,(y - \overline{y})}{\sum (x - \overline{x})^2}$$

$$b = \overline{y} - w\,\overline{x}$$

$(x - \overline{x})$: 편차(원래 데이터 값과 평균값 차이)

$(x - \overline{x})^2$: 편차 제곱 (- 값을 없애기 위해)

$\sum (x - \overline{x})^2$: 편차 제곱 합

$\frac{\sum (x - \overline{x})^2}{N}$: 편차 제곱의 평균(분산, σ^2)이며, 이 값에 루트 한 것이 표준편차(σ)

ML Principle (Linear Regression Model)

(수동으로 회귀식 구하기)

공부 시간(x)	10	20	30	40	50
시험 성적(y)	60	70	80	90	100

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
10	60	10-30=-20	60-80=-20	-20 * -20=400	400
20	70	20-30=-10	70-80=-10	-10*-10=100	100
30	80	30-30=0	80-80=0	0 * 0 = 0	0
40	90	40-30=10	90-80=10	10*10=100	100
50	100	50-30=20	100-80=20	20*20=400	400
				Σ 1000	1000

$\bar{x}$ (x 변수 평균) : 30

$\bar{y}$ (y 변수 평균) : 80

$(x - \bar{x})$: 편차

$(x - \bar{x})^2$: 편차 제곱 (- 값을 없애기 위해)

$\sum (x - \bar{x})^2$: 편차 제곱 합

$\frac{\sum (x - \bar{x})^2}{N}$: 편차 제곱의 평균(분산)이며, 이 값에 루트 한 것이 표준편차

$$w = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2} \frac{1000}{1000} = 1$$

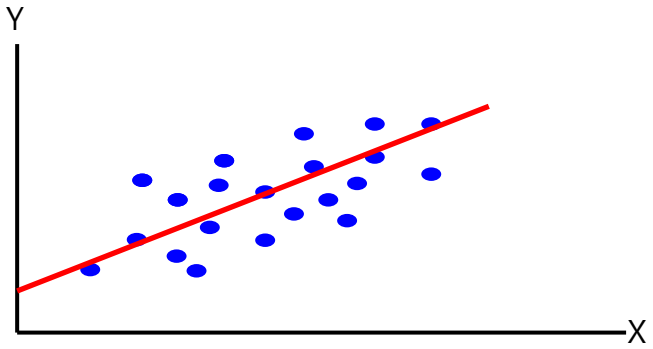
$$b = \bar{y} - w\bar{x} = 80 - 1 * 30 = 50$$

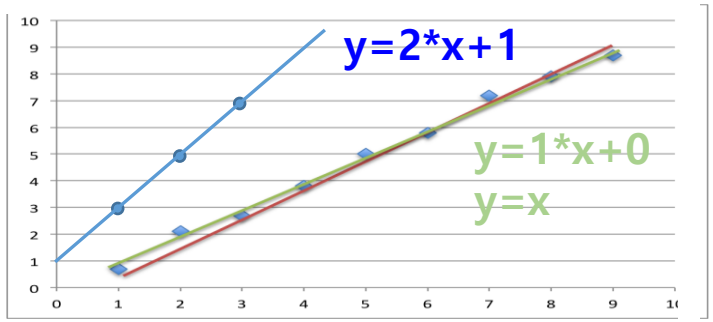
$$\hat{y} = w x + b = 1 * x + 50$$

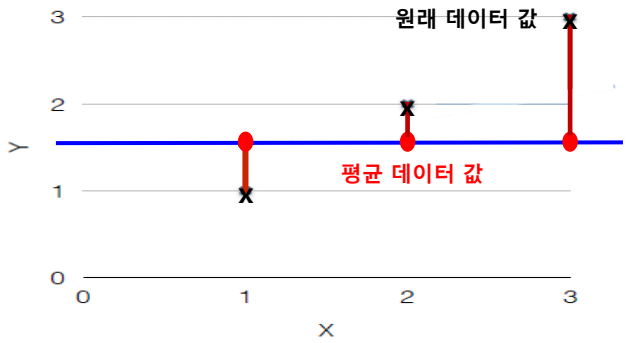
ML Principle (Linear Regression Model)

Linear Regression (Mathematics)

정리를 하면.....







*** (선형) 회귀 분석**

선형 함수(linear function)을 이용하여 **회귀 분석**(어떤 변수가 다른 변수에 의해 설명, 상관관계가 된다고 보고 그 변수간 **관계**를 조사하는 **통계적 해석수법**)을 수행하는 것.

*** 선형 함수(linear Function)**

입력(x)과 출력(y)의 **관계**가 **직선적(linear)**인 성격을 나타내며, 이를 우리가 중학교 때 배운 **1차 함수(function)**.

선형(직선) 함수 = 1차 함수 (y = 2 * x +1)

비선형(곡선) 함수 : $y = 2 * x^2 + 3$ (X)
$y = \sqrt{x} + 3$ (X)

*** 선형회귀 모델 정의(회귀식)**

$y = w * x + b$

기울기(weight) 편향(bias)

- **가중치(weight)**
  : 이 함수가 만들 직선의 ***기울기*** **변화의 값**

$$(\text{기울기}) = \frac{y\text{의 변화량}}{x\text{의 변화량}} = \frac{\Delta y}{\Delta x}$$

- **편향(bias)**
  : 이 함수가 만들 직선을 **위나 아래로 치우침의 변화 값**

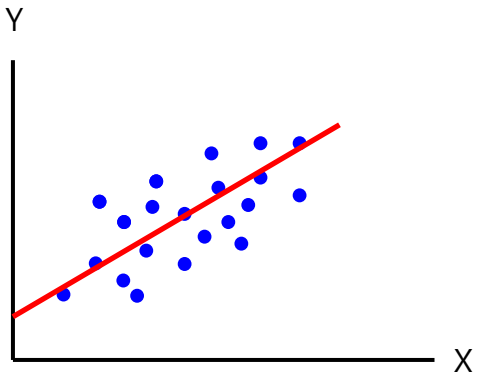
*** 편차 : 원래 데이터 값과 평균 값간의 차이**

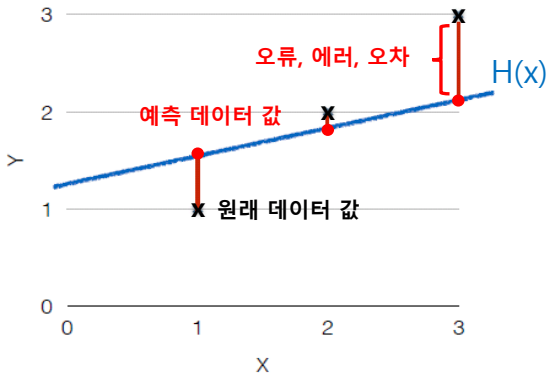
$(x - \bar{x})$ $(y - \bar{y})$

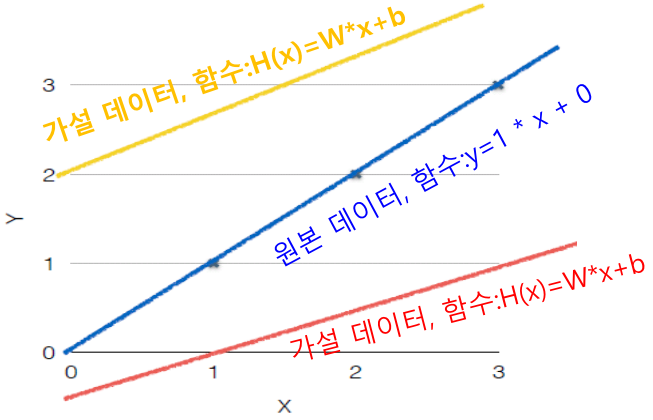
*** 분산 : 이런 데이터들이 평균을 기준으로 퍼져있는 정도, 즉 분포 특성을 나타내는 수치로서 원래 데이터 값과 평균값의 오차.**

$$\frac{\sum (x - \bar{x})^2}{N} \qquad \frac{\sum (y - \bar{y})^2}{N}$$

ML Principle (Linear Regression Model)







Linear Regression Model (Machine Learning)

* Hypothesis (가설) 정의

"데이터가 x일 때 y의 결과가 000 나올 것이다"라고 예측을 위한 두 변수 사이의 관계 함수 정의. 즉, 지금 내가 가지고 있는 Data Set들을 보니 이런 **선형적인 회귀분석 함수 특성**을 가지고 있으니, 이 함수(알고리즘)을 정의(선택)해서 새로운 데이터간 **특성을 예측하는 이 함수를 찾는 것이 ML의 학습**.

➔ **수학적 모델(회귀분석)** : y = w * x + b

➔ **ML 알고리즘 모델** : **H(x) = w * x + b** **(hypothesis = w * x + b)**

* Cost(Loss) Function (오류 함수)

H(x) 함수로 예측된 **빨간 점(예측 값)**과 **검정색(원 데이터 값)** 간의 **차이**(오차, 오류)를 계산하는 함수. 즉, H(x)로 유추한 값(H(x))과 실제 데이터 값(Yi)의 차이. 오차가 클수록 이 예측 값의 함수(직선)는 이들 **데이터들을 대표하는 직선이 아니다**.

➔ **수학적 모델**

$$\sigma^2 = \frac{\sum (y - \bar{y})^2}{n}$$

➔ **ML 알고리즘 모델**

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

cost = tf.reduce_sum(tf.square(hypothesis - Y))

* Gradient Descent Algorithms (경사 하강법)

빨간 선과 **노란 선** 중 어느 Hypothesis가 더 좋다고 할 수 있을까? 만약 **빨간 선 함수**를 최적화하여 원본 데이터 함수에 근접하게 끔, ML Engine이 정확하게 scoring할 수 있는 무언가(**오류가 작은 함수**)를 **찾아주는 알고리즘**. 결론은 ML은 최적의 (Cost가 최소화된) w와 b 값을 찾는 작업(학습)

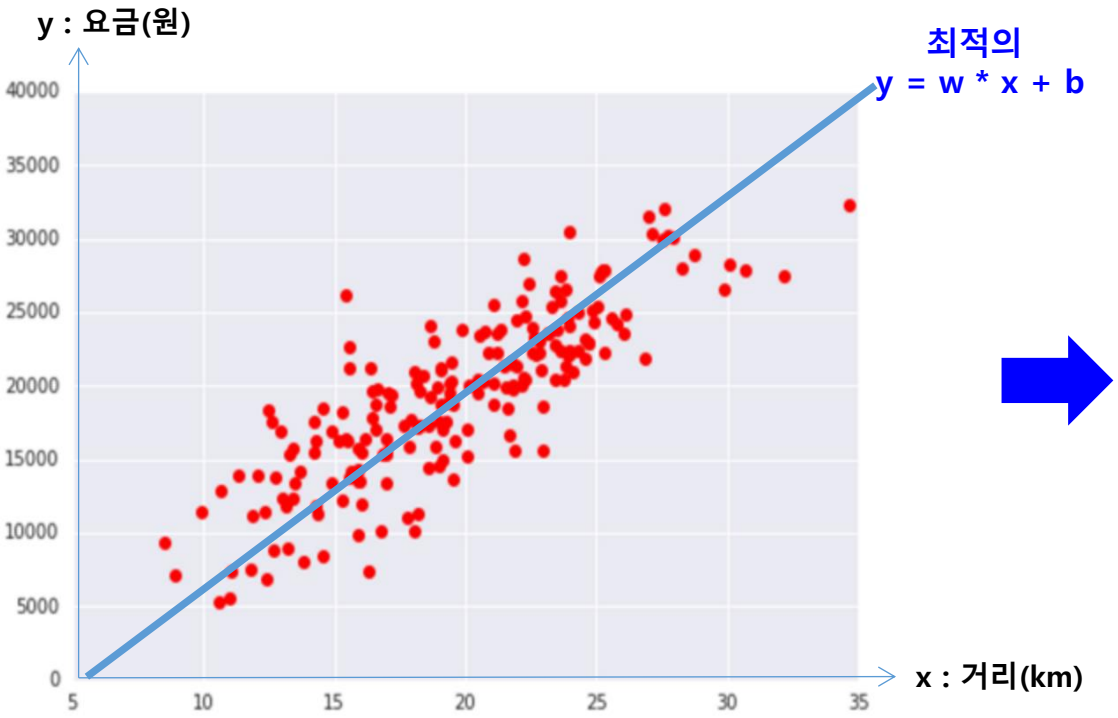
$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
train = optimizer.minimize(cost)

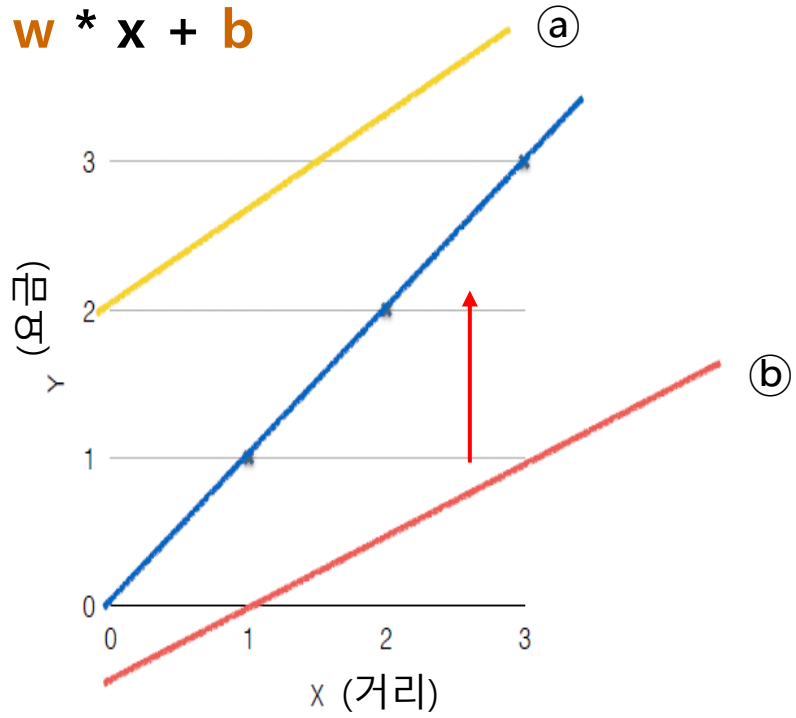
ML Principle (Linear Regression Model)

Linear Regression (Supervised Learning)

1. Hypothesis(가설) 정의



H(x) = w * x + b



Model : **H(x)** = W(weight) * x + b(bias), **Training Data** : (x_1:거리, y_1:요금), (x_2, y_2)...(x_n, y_n) **Parameter** : (**W, b**)

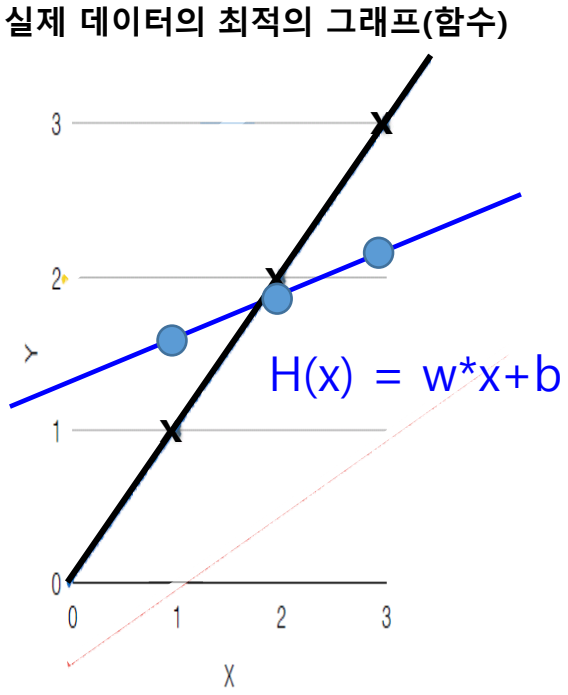
➔ **[Goal] Find (w, b)** which best fits the given data

ML Principle (Linear Regression Model)

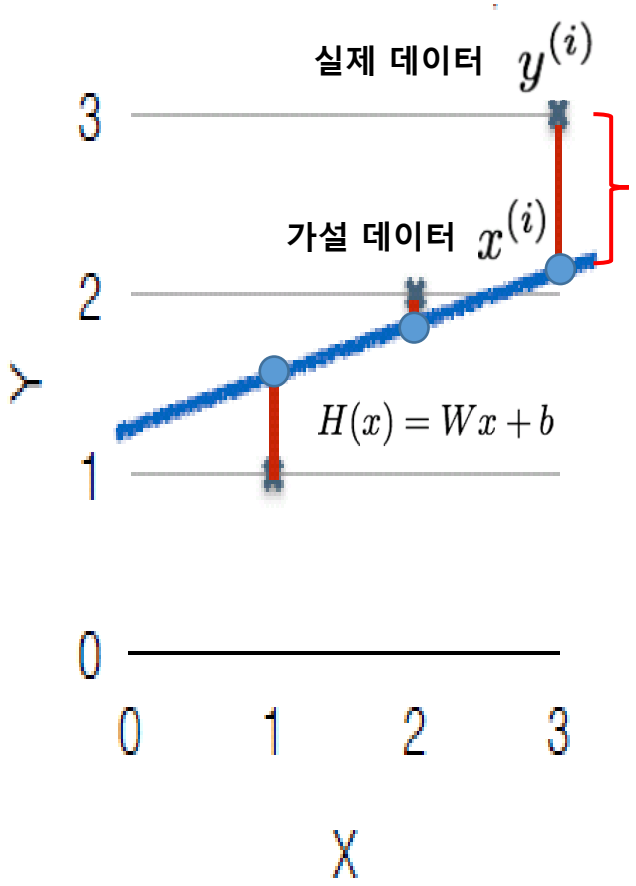
Linear Regression (Supervised Learning)

2. Cost Function : How fit the line to our training data

우리가 세운 "가설 H(x) 값(예측 값)과 실제 Data 값이 얼마나 다른가"를 의미. 이 "차이(오류, 에러)"를 "최소화"



최적의 그래프를 만들기 위한 기울기(w)와 편향(b)를 어떻게 찾아야 하나?
즉, Cost function을 정의함으로써 이 함수가 기존에 주어진 데이터에 대해서 얼마나 정확한지 유추할 수 있음.



Cost Function은 **실제 데이터** $y^{(i)}$ 와 **가설로 측정된 데이터** $x^{(i)}$ 간의 **거리 차이**를 계산 알고리즘**(최소화)**

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

Goal = $\underset{W,b}{\text{minimize}}\, cost(W,b)$

➔ 최소화된 w(기울기)와 b(편향) 값을 찾아 보정

ML Principle (Linear Regression Model)

Linear Regression (Supervised Learning)

3. Optimizer 정의

우리가 구하고자 하는 함수 그래프는 **실제 값**에서 **그래프(가설, 예측)의 값**까지 **차이가 가장 작은 값**을 구하고자 함(Cost Function).
이 Cost Function가 최소가 되는 W와 b를 자동으로 구해주는 최적화 알고리즘

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

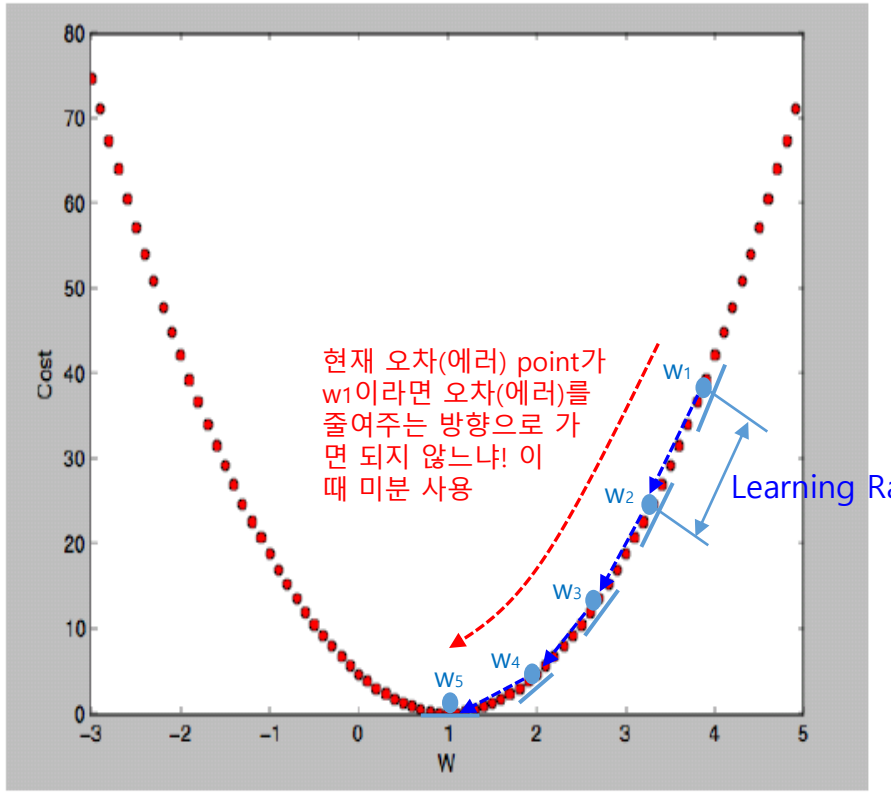
Gradient descent algorithm

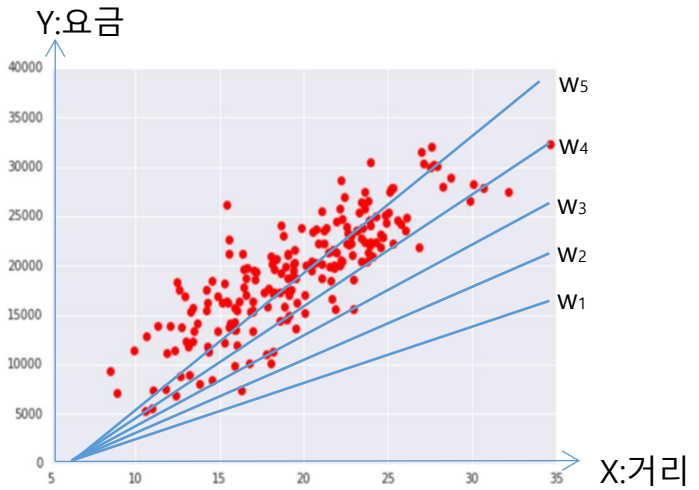
$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

$$b := b - \alpha\frac{\partial}{\partial b}cost(b)$$

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

현재 w값 / Learning Rate / Cost Function 미분 값(기울기)





최적의(cost function이 최소가 되는) W와 b(무시) 를 찾는 것(최적화 된 함수, 오차가 가장 적은 cost 함수 찾기)이 학습 과정이며, 목표임.

➔ **우리가 가지고 있는 데이터(Training Data)를 통해(학습 시켜) 이 Cost를 최소화 시키는 w,b 값을 찾는 것이 Machine Learning에서 선형회귀분석 목표임**

ML Principle (Linear Regression Model)

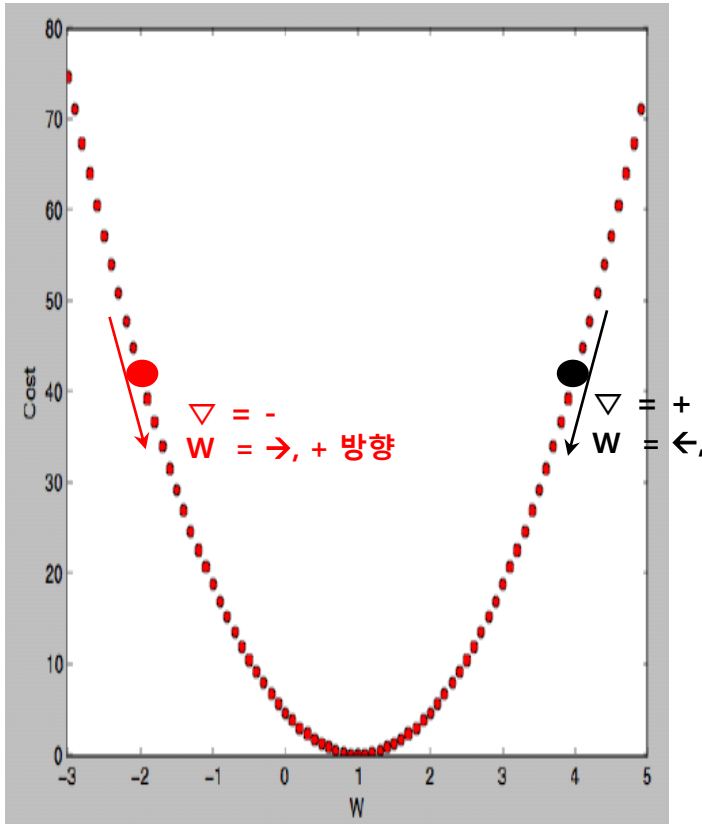
Linear Regression (Supervised Learning)

3. Optimizer 정의

How it works?

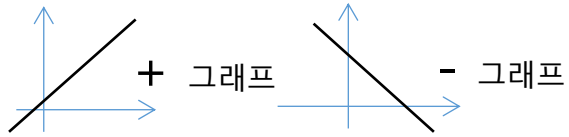
- Start with initial guesses
  - Start at 0,0 (or any other value)
  - Keeping changing W and b a little bit to try and reduce cost(W, b)
- Each time you change the parameters, you select the gradient which reduces cost(W, b) the most possible
- Repeat
- Do so until you converge to a local minimum

Gradient descent algorithm



▽ = -
W = →, + 방향

▽ = +
W = ←, - 방향

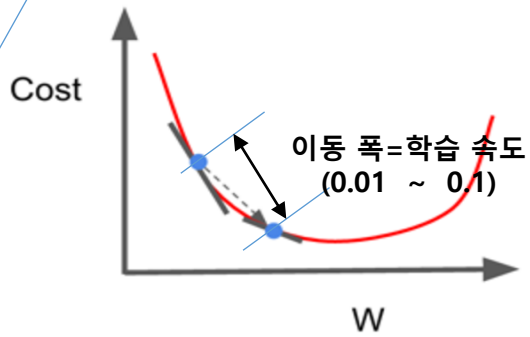


$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)}) x^{(i)}$$

Cost F를 미분한 값
(cost 함수의 기울기)
▽

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

α : Learning Rate



ML Principle (Linear Regression Model)

Linear Regression (Supervised Learning)

3. Optimizer 정의(Gradient Descent Algorithms : Manual 증명)

Hypothesis : H(x) = W * x + b

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

Simplified 정의 Hypothesis : H(x) = W * x + b(=0)

H(x) = W * x

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

W = 뭐일 때, cost(W)의 값은 ?

x	Y
1	1
2	2
3	3

- W=1, cost(W)=0

$$\frac{1}{3}((1*1-1)^2 + (1*2-2)^2 + (1*3-3)^2)$$

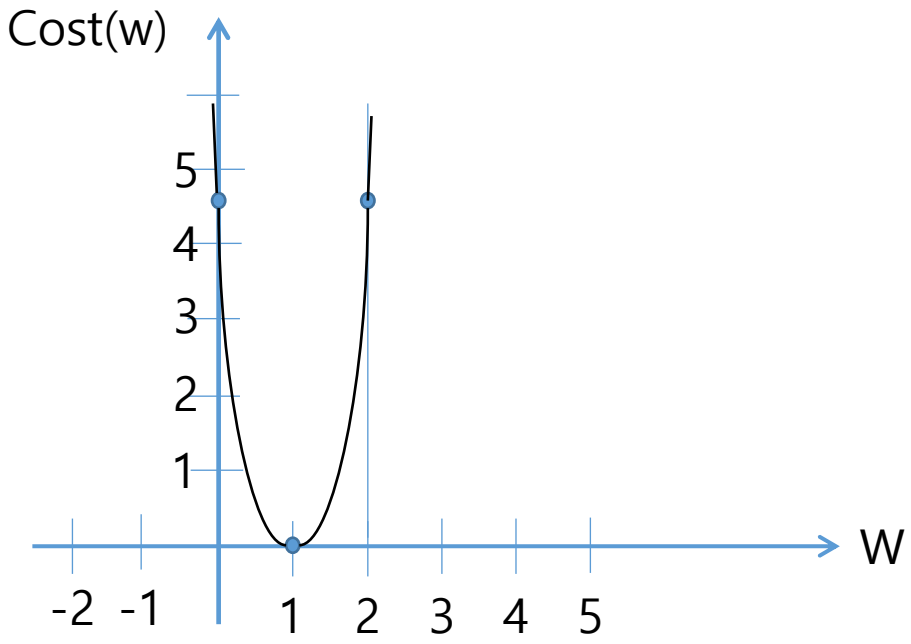
- W=0, cost(W)=4.67

$$\frac{1}{3}((0*1-1)^2 + (0*2-2)^2 + (0*3-3)^2)$$

- W=2, cost(W)=?

$$\frac{1}{3}((2*1-1)^2 + (2*2-2)^2 + (2*3-3)^2)$$

1 4 9 = 14/3

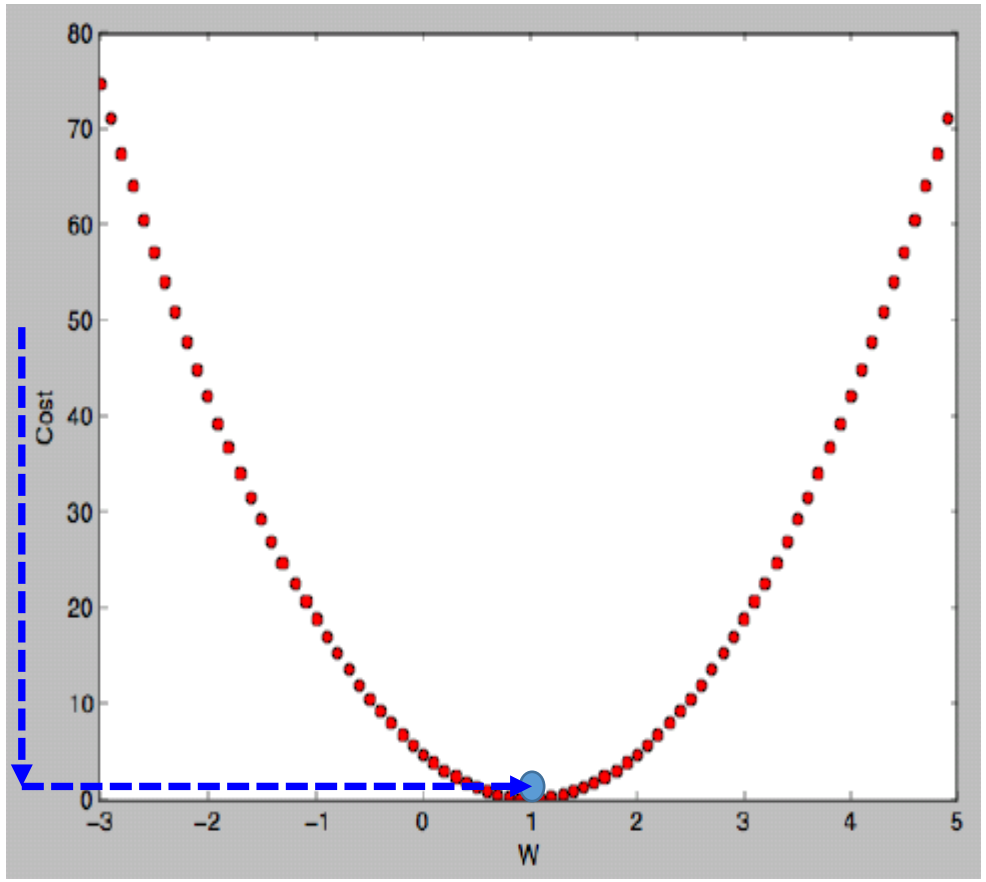


➔ y = w(1) * x + b(0) ➔ y=x

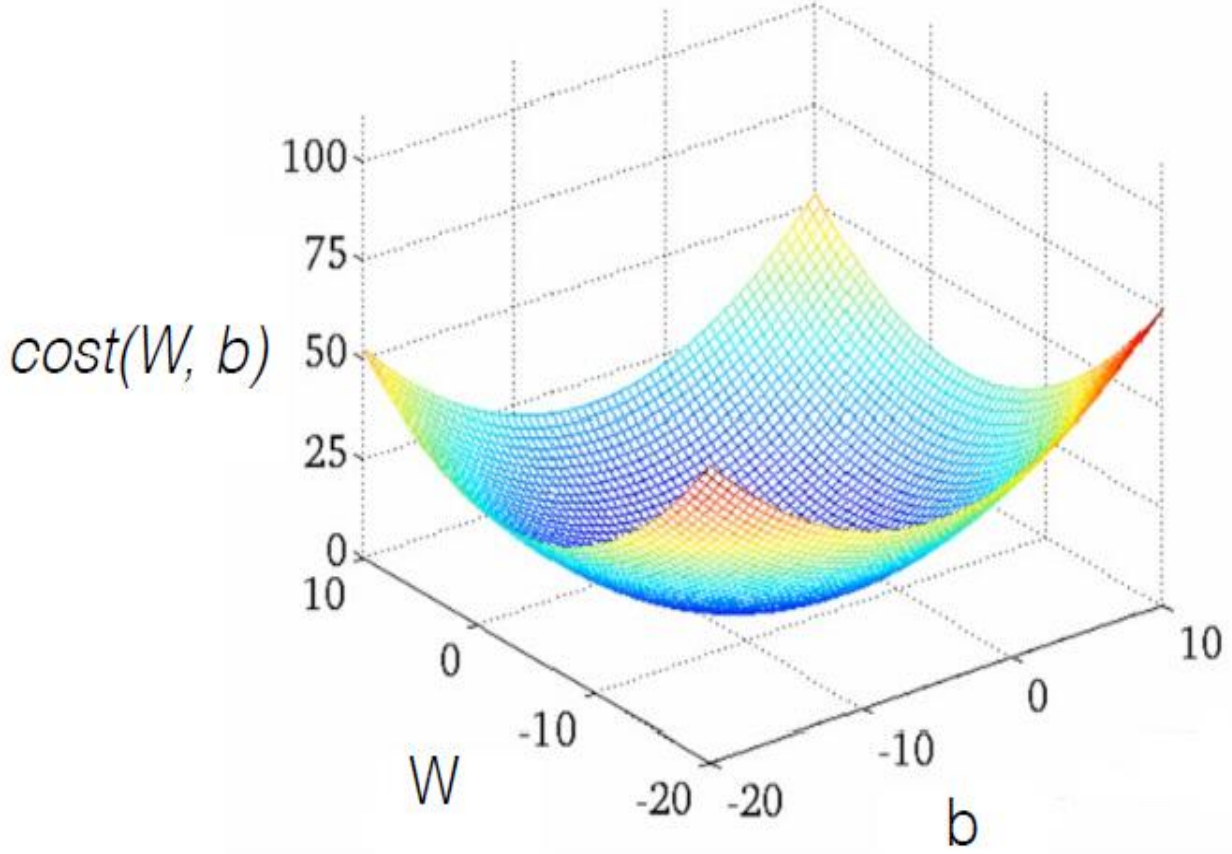
Linear Regression (Supervised Learning)

3. Optimizer 결론

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$



$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$



Machine Learning Principle
(실습 Programming)

ML Principle (Linear Regression Model) 실습

TensorFlow

Machine Learning(ML)과 **Deep Learning(DL)**을 위해 Google에서 만든 **Open Source Library (Software)**

Deep learning libraries:
Accumulated GitHub metrics

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3		
#1:	172.29	tensorflow/tensorflow
#2:	89.78	BVLC/caffe
#3:	69.70	fchollet/keras
#4:	53.09	dmlc/mxnet
#5:	38.23	Theano/Theano
#6:	29.86	deeplearning4j/deeplearning4j
#7:	27.99	Microsoft/CNTK
#8:	17.36	torch/torch7
#9:	14.43	baidu/paddle
#10:	13.10	pfnet/chainer
#11:	12.37	NVIDIA/DIGITS
#12:	10.42	tflearn/tflearn
#13:	9.20	pytorch/pytorch

- 많은 사람들이 사용함으로 검증 및 공부할 수 있는 자료들이 상당히 많음
- 또 많은 사전 검증 모델들이 open source로 공개되어 있기 때문에 source code 참조 유리
- Python Language Programming

  ➔ 지금 시점에서는 TensorFlow Library가 가장 좋은 선택 (사견)

ML Principle (Linear Regression Model) 실습

TensorFlow Mechanics

```
import tensorflow as tf

# tf Graph Input
X = [1, 2, 3]
Y = [1, 2, 3]

# Set wrong model weights
W = tf.Variable(5.0)
# Linear model
hypothesis = X * W
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize: Gradient Descent Magic
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
train = optimizer.minimize(cost)
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

for step in range(100):
    print(step, sess.run(W))
    sess.run(train)
```

이런 x, y값이 입력되고, 굉장히 간단한 training data(x=1➔y=1, x=2➔y=2)

여기서 weight 값을 5로 임의적으로 지정. 또는 순차적으로 자동화 가능(random)

예측할 hypothesis H(x) = W*x + b 정의. 간소화하기 위해 b=0로 생략 $H(x) = Wx$

Cost Function 정의 $cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$

위에 정의된 optimizer의 minimize 함수를 호출하면서 무엇을 최소화 할 것인가 하면 이 cost를 최소화하라 주면, 위의 w값을 조정해서 minimize하게 됨

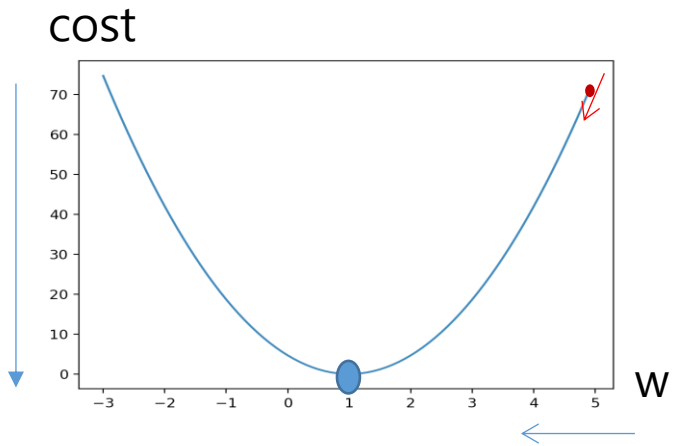
: 실행 명령

: 실행 전 초기화

Cost(Loss) Function의 최소화된 W값을 찾는 최적화 알고리즘인 Gradient Descent Algorithms정의

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

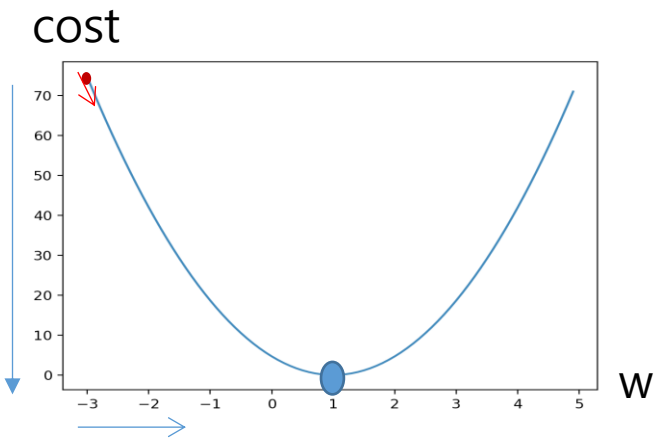
Output when W=5



Step	W(weight) Value
0	5.0
1	1.26667
2	1.01778
3	1.00119
4	1.00008
5	1.00001
6	1.0
7	1.0
8	1.0
9	1.0

ML Principle (Linear Regression Model) 실습

TensorFlow Mechanics




```
import tensorflow as tf

# tf Graph Input
X = [1, 2, 3]
Y = [1, 2, 3]   ➔ Y = [2, 4, 6] : y 값을 변경해서 실행, 변화 볼 것(W 값)

# Set wrong model weights
W = tf.Variable(-3.0)   ➔ W 값을 변경해서 실행

# Linear model  (y = w * x + b)
Hypothesis = X * W                                   ➔ H(x) = Wx
# cost/loss function
Cost = tf.reduce_mean(tf.square(hypothesis - Y))     ➔ cost(W) = 1/m Σ_{i=1}^{m} (Wx^(i) − y^(i))^2
# Minimize: Gradient Descent Magic
Optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1) ➔ W := W − α 1/m Σ_{i=1}^{m} (Wx^(i) − y^(i)) x^(i)
Train = optimizer.minimize(cost)
                                  LR를 큰 값(1) 또는 작은 값(0.001)
                                  으로 변경 해서 변화를 볼 것

# Launch the graph in a session.
Sess = tf.Session()
# Initializes global variables in the graph.
Sess.run(tf.global_variables_initializer())

For step in range(100):          ➔ 학습 단계를 2, 4번 등 변경하면 서 실행
   print(step, sess.run(W))      ➔ PRINT를 COST 값까지 출력하여 변화 실행
   sess.run(train)
```

Output when W=-3

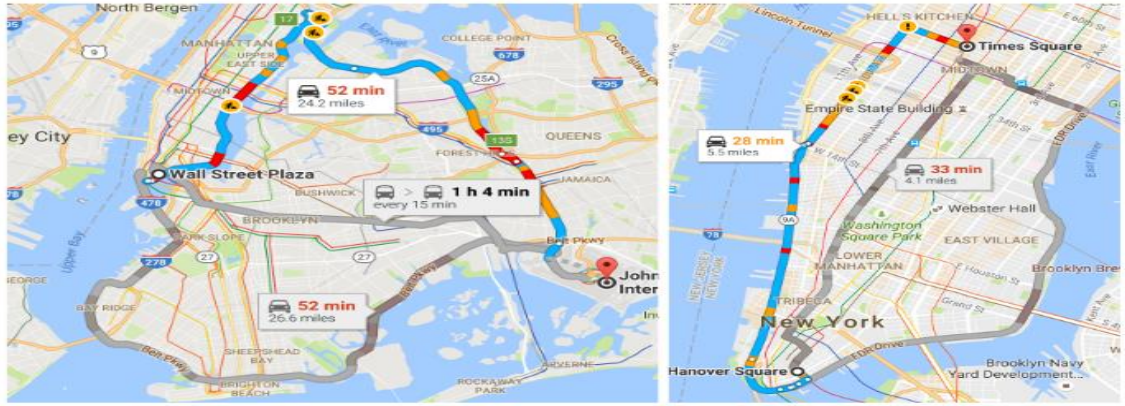
STEP	W
0	-3.0
1	0.733334
2	0.982222
3	0.998815
4	0.999921
5	0.999995
6	1.0
7	1.0
8	1.0
9	1.0

ML Principle (Linear Regression Model) 실습

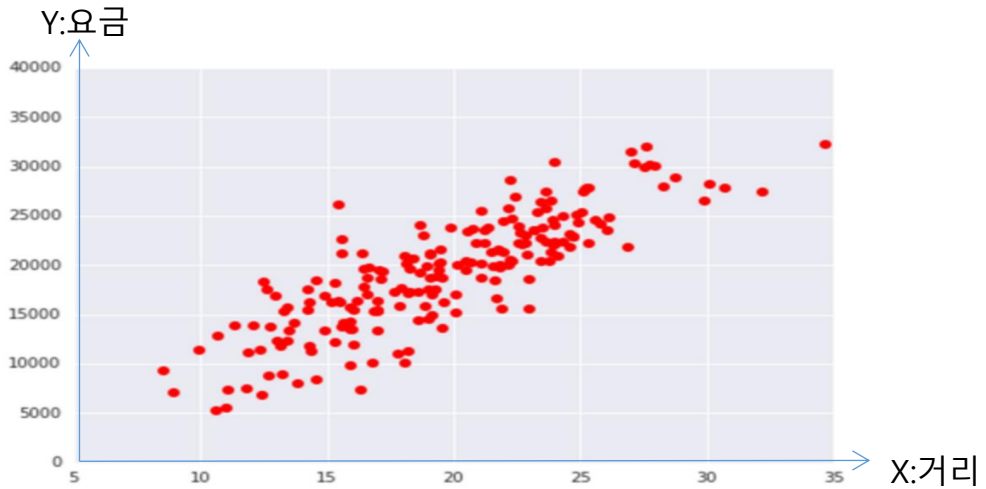
선형 회귀 분석 (Linear Regression)

쉽게 설명하면 **결과값 (output value)**이 있고 그 결과값을 결정할 것이라고 **추정(추론, 예측)**되는 **입력 값 (input value:X)과 결과 값(Output value : Y)**의 **연관 관계**를 찾는 것이고, 이를 **선형 관계**를 통해 찾는 방법이 **선형 회귀 (Linear regression)**이다.

예 : 택시 요금
택시 요금은 물론 막히냐 마냐에 따라 편차가 있지만, 대부분 **거리**에 비례해서 요금이 부과. 그래서 결과값 (Y: 요금)과 입력값 (X: 거리)의 **상관 관계**를 찾아야 한다.

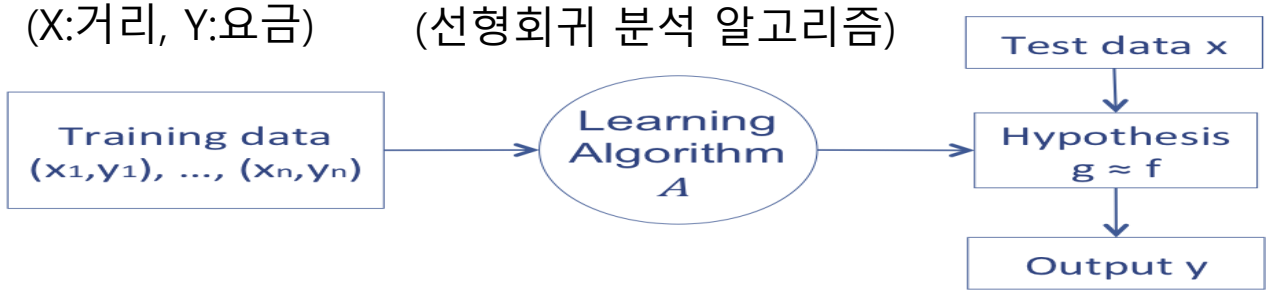


예 : 택시 요금(거리별 택시 요금 분포도 Graph)
원본 데이터의 거리를 X, 해당 거리에 측정된 택시 요금 Y
(원본 Data Set ➔ Training Data)



(X:거리, Y:요금)

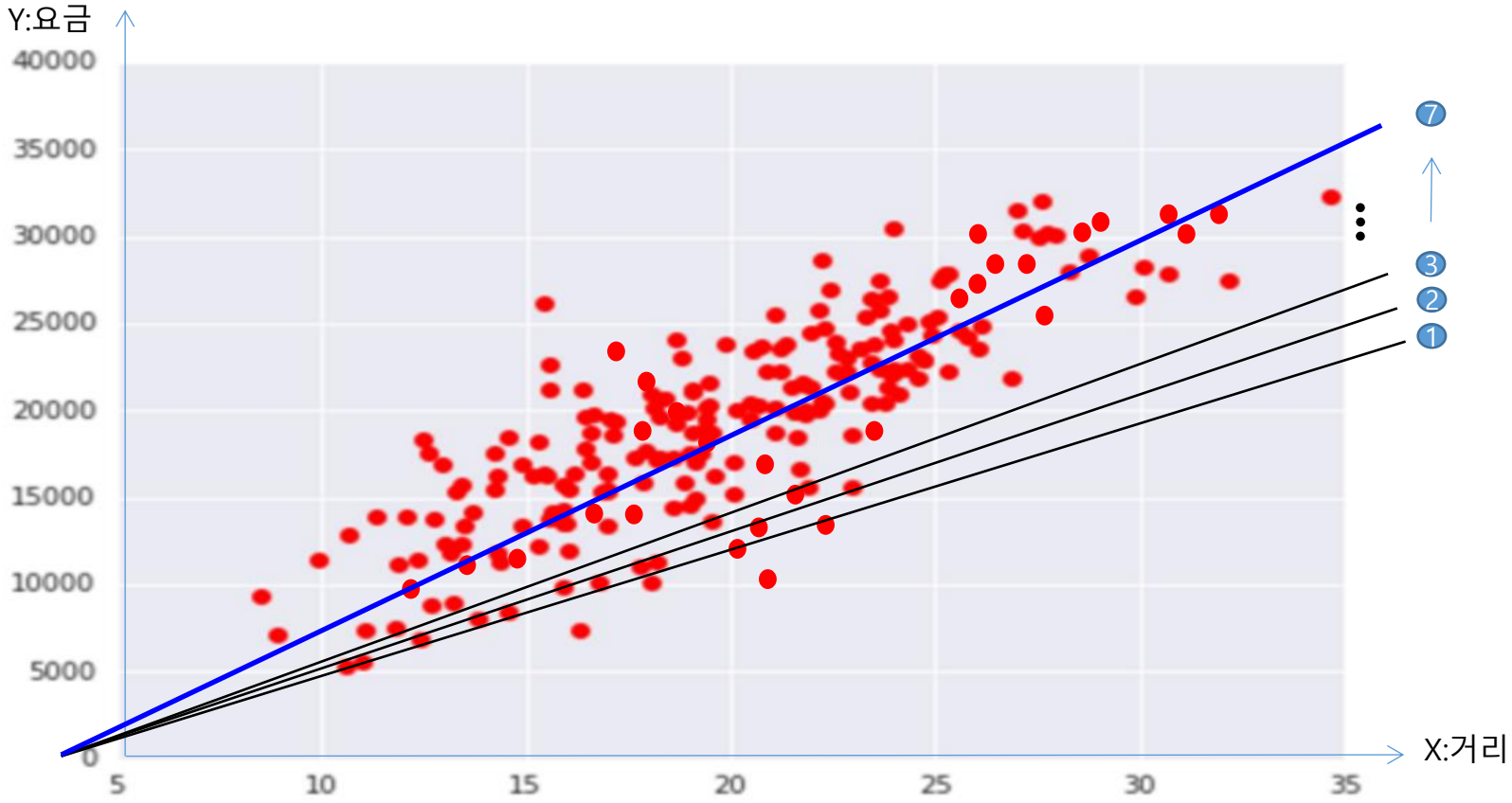
(선형회귀 분석 알고리즘)



예 : **가설(Hypothesis) 정의**
거리(Y)와 요금(X)이 서로 **비례관계(선형적)** 이기 때문에, 거리(X)와 요금(Y)간의 상관관계는 다음과 같은 **일차 방정식(함수)과 그래프**를 그리게(Linear Regression Algorithms 적용) 된다고 가정(가설 정의)

ML Principle (Linear Regression Model) 실습

선형 회귀 분석 (Linear Regression)



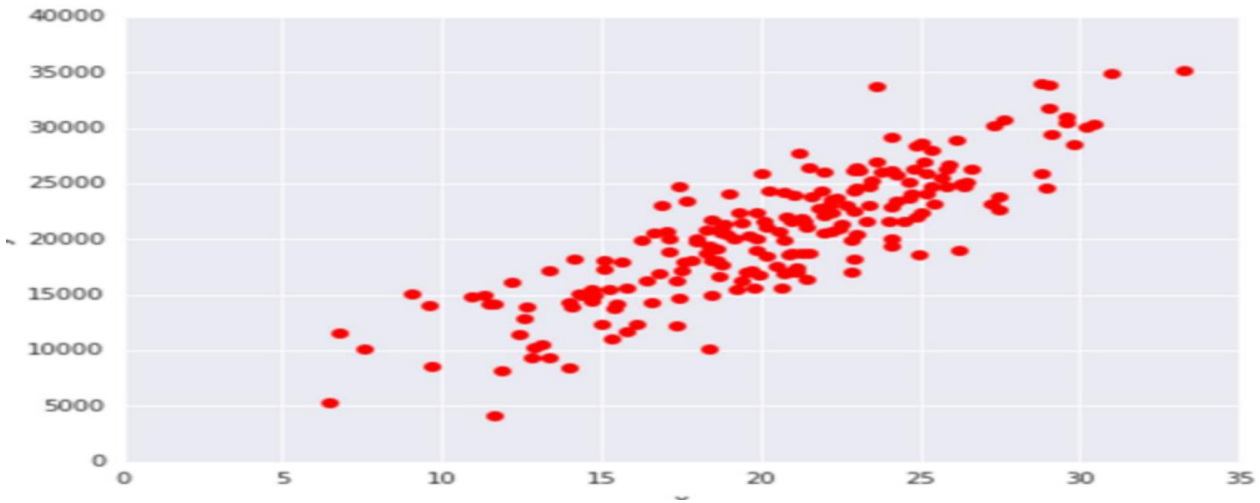
1. x는 거리, y는 요금의 training **Data 수집** (분석하고자 하는 Data 수집)
2. 이 Data들의 feature(특징)은 **실수이며, 선형적인(비례, 직선) Feature**를 가지고 있다. **(데이터 Feature 정의)**
3. 그러므로 **가설(Hypothesis)**로 이 Data들은 **Linear Regression Algorithms**이 적당하지 않을까? **(가설 정의 : H(x) = w * x + b)**
4. **Cost(Loss) 함수 정의** : 최적화 알고리즘인 **Gradient Descent Algorithms**을 적용하여 학습을 시킴 **(해당 알고리즘에 학습 진행)**
5. **W(weight:기울기) Value를 조정**(여러 번 반복 step으로 학습 시킴)해서 가장 **최적화된 함수(알고리즘) fits**
6. **예측 모델 적용** : 50km 주행시 택시비는? **학습된 모델로 예측** : 택시비 = **? (w)** * 50 + **?(b)** ➔ 50km 택시비는 얼마하고 출력 **y값 예측($\hat{y}$)**

ML Principle (Linear Regression Model) 실습

선형 회귀 분석 (Linear Regression)

```
import numpy as np
num_points = 200
vectors_set = [ ]
for i in xrange(num_points):
 x = np.random.normal(5,5)+15
 y =  x*1000+(np.random.normal(0,3))*1000

 vectors_set.append([x,y])

x_data = [v[0] for v in vectors_set ]
y_data = [v[1] for v in vectors_set ]

import matplotlib.pyplot as plt
plt.plot(x_data,y_data,'ro')
plt.ylim([0,40000])
plt.xlim([0,35])
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
# x range로 200개 sample data 생성


# x축은 택시 주행 거리로, 정규분포를 따르는 난수 생성. 5를 중심으로 표준편차가 5인 데이터 생성.
# y축은 택시비인데, 주행거리(x 축 값) * 1000 + 정규분포를 따르는 난수를 중심 값은 0, 그리고 표준편차를 따르는 난수
  생성 후
# 이 값에 1000을 곱하였다. 그리고 x data에는 x 값, y data에는 y값들을 배열 형태로 저장



# Pyplot이라는 모듈을 이용하여 그래프를 그림

# y 축은 0에서 40000
# x 축은 0에서 35까지의 범위
```



ML Principle (Linear Regression Model) 실습

선형 회귀 분석 (Linear Regression)

```
import tensorflow as tf

W = tf.Variable(tf.random_uniform([1],-1.0,1.0))
b = tf.Variable(tf.zeros([1]))
Y = W * x_data + b

loss = tf.reduce_mean(tf.square(y-y_data))
Optimizer = tf.train.GradientDescentOptimizer(0.0015)
Train = optimizer.minimize(loss)

init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)

for step in xrange(10):
  sess.run(train)
  print(step,sess.run(W),sess.run(b))
  print(step,sess.run(loss))

  plt.plot(x_data,y_data,'ro')
  plt.plot(x_data,sess.run(W)*x_data + sess.run(b))
  plt.xlabel('x')
  plt.ylabel('y')
  plt.legend()
  plt.show()
```

w의 초기값은 random 값으로 생성,(-1.0 ~ 1.0 사이의 값으로 생성 [1] :1차원 배열 의미, 2면 2차원 행렬 의미

b값은 tf.zero([1])로 정의. 1차원으로 0 값 정의

 # Hypothesis(y) = w * x data + b(0)

$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

 # cost(loss) 함수 정의

 # Gradient Descent Algorithms 정의 (learning rate로 0.0015 정의)

train model 정의로 cost(loss) 함수 값이 이전의 **Gradient Descent Algorithms**을 이용하여 최소화되는 w를 자동으로 계산하여 찾으라는 의미

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

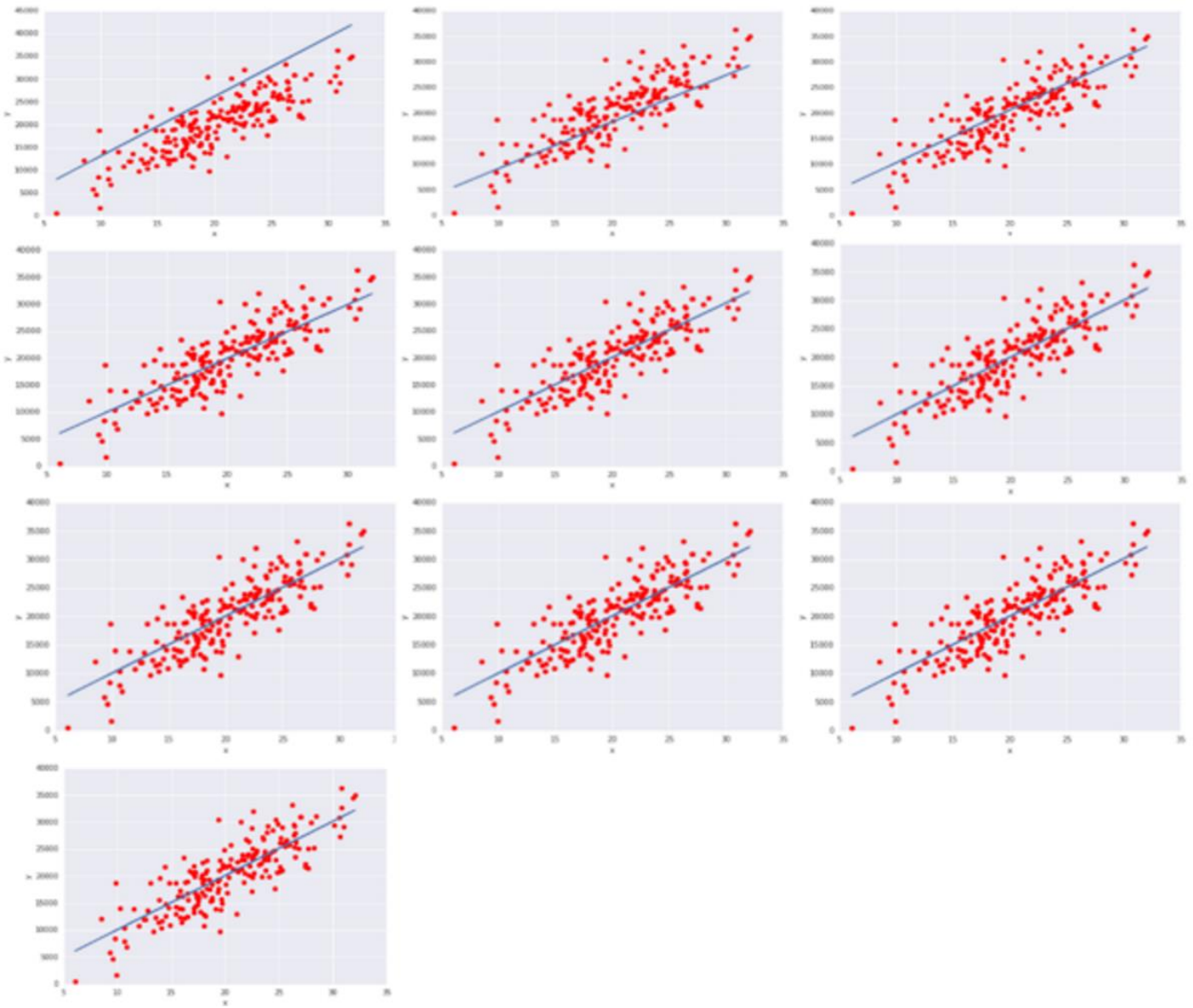
for loop를 돌려 학습을 10번 반복하라

학습된 결과로 학습 단계(0~9), 각 단계별 w 값, b 값을 출력 정의

학습이 어떻게 되는지 그래프로 표현. x_data를 가로축으로 하고, w * x +b의 값을 그래프로 출력

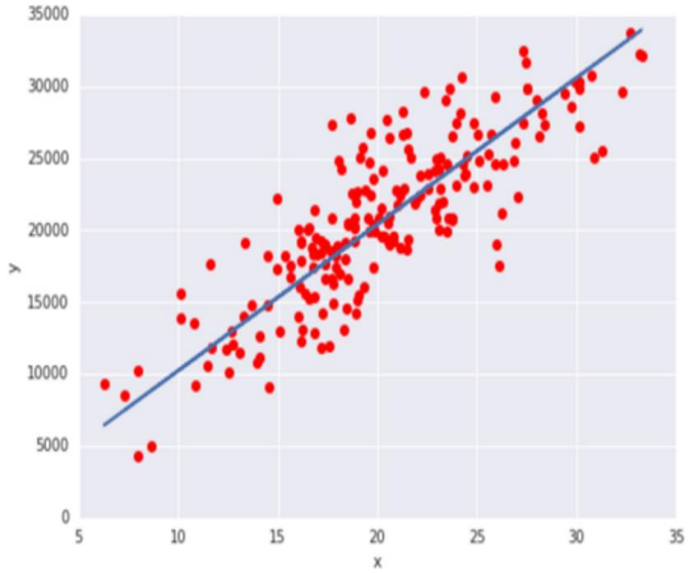
ML Principle (Linear Regression Model) 실습

선형 회귀 분석 (Linear Regression)



그래프가 점점 **데이터의 중앙에 수렴**하면서 **조정(학습)**되는 것을 확인할 수 있으며, 이렇게 해서 맨 마지막 학습(10번)에 다음과 같은 결과가 출력된다.

```
(9, array([ 1018.00439453], dtype=float32), array([ 51.36595535], dtype=float32))
(9, 10272684.0)
```



학습된 **W 값은 1018**, **b는 51** 그리고 **cost 값은 10272684.0**이 됨을 확인할 수 있다.

이렇게 최종 학습이 끝났고, 이제 거리에 따른 택시비는
(택시비) = 1018 * (거리 : new x_data) + 51로 이 공식을 가지고
새로운 거리(New x_data)에 따른 택시비(y 값)를 예측.

ML Principle (Linear Regression Model)

Recap

Hypothesis and Cost Function and Gradient Desent Algorithms

$$H(x) = Wx + b$$

주어진 입력 x 값에 대해서 예측을 어떻게 할 것인가? Hypothesis H(x). W(weight) 곱하기 x에 bias 합 표기. 단순화하기 위해 b=0

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

예측을 한 값 H(x)에서 **실제 데이터 값(y)** 간의 **차이**(얼마나 예측을 잘 했나를 의미)에 제곱한 것을 평균 낸 것이 **Cost Function**. W와 b의 값에 따라 cost function이 작아질 수도 커질 수 도 있다. 우리가 **학습을 한다!**는 의미는 w, b값을 조정해서 **cost(loss) 값을 최소화(에러, 오차)** 하는 것임.

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

이 **cost 함수 값이 최소화**되는 **w(weight:기울기)를 자동으로 찾아주는 Gradient Descent Algorithms**을 적용. 결론적으로 우리가 가지고 있는 사용할 트레이닝 데이터들을 가지고 학습을 시켜 이런 형태의 **최소화된 cost 함수(function)**을 만들어 낼 수 만 있다면 **"아주 정확한(예측 정확도),** 또는 **학습이 잘된(최적화된) Machine Learning Algorithms**"이 되는 것임.

Machine Learning Principle
(Linear Regression Model : Multi-Variable)

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Predicting exam score:
regression using **one input (x)**

one-variable
one-feature
x (hours)

x (hours)	y (score)
10	90
9	80
3	50
2	60
11	40

Hypothesis

$$H(x) = Wx + b$$

Cost Function

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

Gradient Descent Algorithms

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Predicting exam score:
regression using **three input (x1, x2, x3)**

multi-variable/feature

x_1 (quiz 1)	x_2 (quiz 2)	x_3 (midterm 1)	Y (final)
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Hypothesis

$$H(x) = Wx + b$$

$$H(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Multi-variable

$$H(x_1, x_2, x_3, ..., x_n) = w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n + b$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Cost Function

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

$$cost(W, b) = \frac{1}{m} \sum_{I=1}^{m} (H(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) - y^{(i)})^2$$

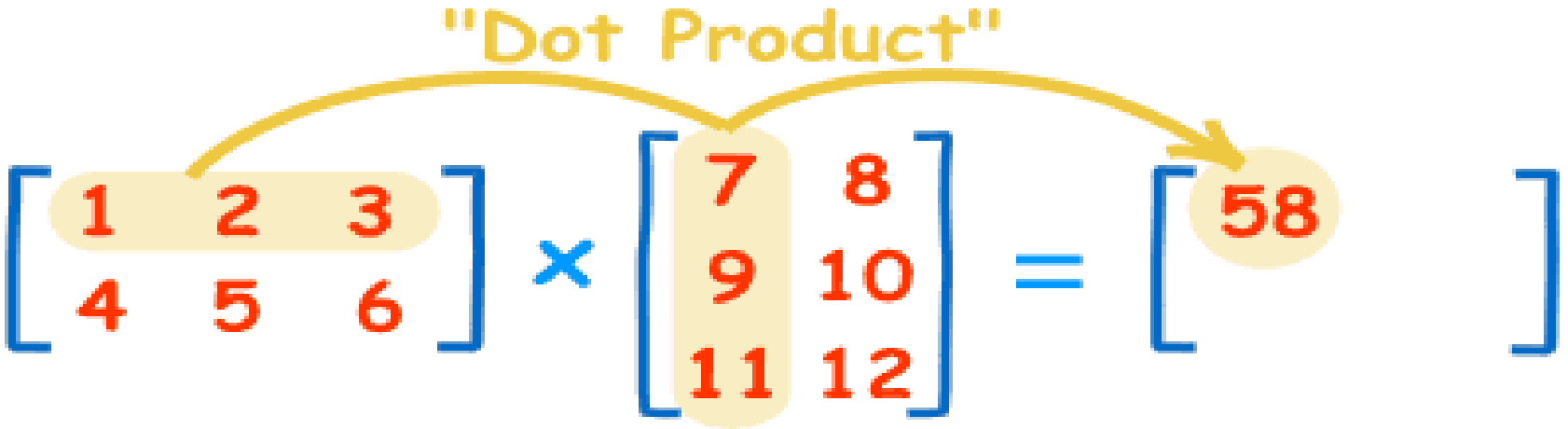
ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Matrix

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

Matrix multiplication (행렬 곱)



ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Hypothesis using Matrix

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1w_1 + x_2w_2 + x_3w_3 \end{pmatrix}$$

Hypothesis using multi-variable

$$H(X) = XW$$

Hypothesis using single-variable

$$H(x) = Wx + b$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Hypothesis using Matrix

Multi-variable(x1, x2, x3)

instance

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

$$H(x_1, x_2, x_3) = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

1x3 • 3x1 = 1x1

$$X \cdot W = H(X)$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Hypothesis using Matrix

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

$$X \cdot W = H(X)$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Hypothesis using Matrix

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

[5, 3] • [3, 1] = [5, 1]

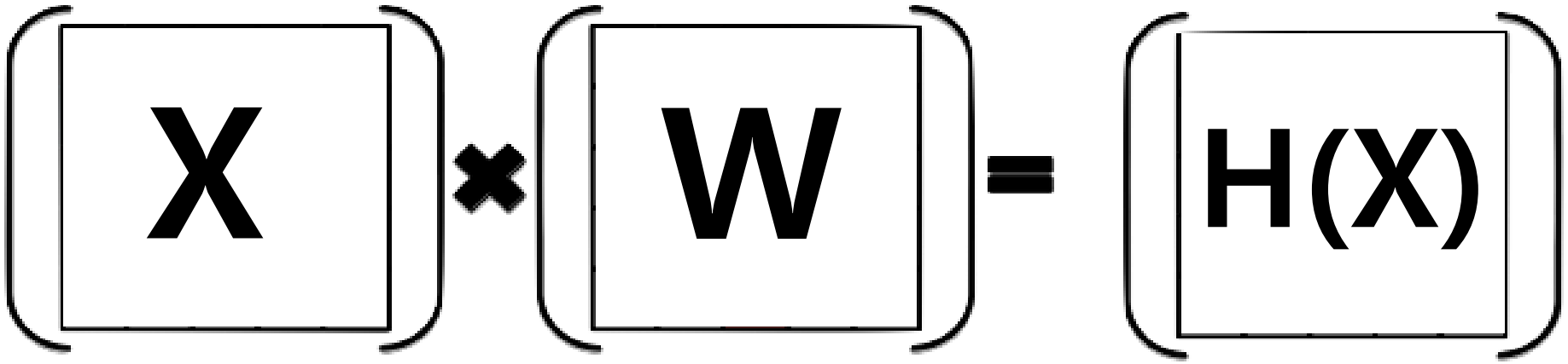
$$X \cdot W = H(X)$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Hypothesis using Matrix

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142



[5, 3] [?, ?] [5, 1]

각 입력 변수 별 instance 개수 입력 변수 개수

$$X \cdot W = H(X)$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Hypothesis using Matrix

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

[n, 3] [3, 1] [n, 1]

$$X \cdot W = H(X)$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Hypothesis using Matrix(n output)

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} = \begin{pmatrix} x_{11}w_{11} + x_{12}w_{21} + x_{13}w_{31} & x_{11}w_{12} + x_{12}w_{22} + x_{13}w_{32} \\ x_{21}w_{11} + x_{22}w_{21} + x_{23}w_{31} & x_{21}w_{12} + x_{22}w_{22} + x_{23}w_{32} \\ x_{31}w_{11} + x_{32}w_{21} + x_{33}w_{31} & x_{31}w_{12} + x_{32}w_{22} + x_{33}w_{32} \\ x_{41}w_{11} + x_{42}w_{21} + x_{43}w_{31} & x_{41}w_{12} + x_{42}w_{22} + x_{43}w_{32} \\ x_{51}w_{11} + x_{52}w_{21} + x_{53}w_{31} & x_{51}w_{12} + x_{52}w_{22} + x_{53}w_{32} \end{pmatrix}$$

?

[n, 3] [3, 2] [n, 2]

$$X \cdot W = H(X)$$

ML Principle (Linear Regression Model : Multi-Variable)

Multi-Variable/feature Model

Wx vs XW

- Lecture (theory):

$$H(x) = Wx + b$$

- Implementation (TensorFlow)

$$H(X) = XW$$

Machine Learning Principle
(Multi-Variable/Feature 실습 Programming)

ML Principle (Linear Regression Model : Multi-Variable) 실습 Programming

Linear Regression (Multi-Variable/feature)

Multivariable example

Matrix

$H(x_1, x_2) = w_1 x_1 + \dots + w_n x_n$

$+ w2 \times x2 + w3 \times x3 \,]$

x1	x2	
1	0	
0	2	
3	0	
0	4	4
5	0	5

Data에서 요구하는 정답은 w1 = 1, w2 = 1, b = 0

1 * 1 + 0 * 1 + 0 = 1
0 * 1 + 2 * 1 + 0 = 2
3 * 1 + 0 * 1 + 0 = 3
0 * 1 + 4 * 1 + 0 = 4
5 * 1 + 0 * 1 + 0 = 5

```
# Try to find values for W
W = tf.Variable(tf.random_uniform([1,2], -1.0, 1.0))
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
```

ML Principle (Linear Regression Model : Multi-Variable) 실습 Programming

```
Import tensorflow as tf
x1_data = [1, 0, 3, 0, 5]
x2_data = [0, 2, 0, 4, 0]
y_data  = [1, 2, 3, 4, 5]

W1 = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
W2 = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b  = tf.Variable(tf.random_uniform([1], -1.0, 1.0))

hypothesis = W1*x1_data + W2*x2_data + b
cost = tf.reduce_mean(tf.square(hypothesis - y_data))

rate = tf.Variable(0.1)
optimizer = tf.train.GradientDescentOptimizer(rate)
train = optimizer.minimize(cost)

init = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init)

for step in range(2001):
    sess.run(train)
    if step%20 == 0:
        print(step, sess.run(cost), sess.run(W1), sess.run(W2), sess.run(b))

sess.close()
```

```
Import tensorflow as tf

x_data = [[1., 0., 3., 0., 5.], [0., 2., 0., 4., 0.]]   # x_data는 2x5(2행5열 행렬) 행렬 변환
y_data  = [1, 2, 3, 4, 5]                                # y_data는 1x5 (1행5열)

W = tf.Variable(tf.random_uniform([1, 2], -1.0, 1.0))    # 행렬 곱셈을 해야 하니 w는 1X2 행렬(w1, w2)
b = tf.Variable(tf.random_uniform([1], -1.0, 1.0))       # b는 1차원

hypothesis = tf.matmul(W, x_data) + b                    # Hypothesis는 행렬 곱 w(1x2) *x_data (2x5) = (1x5)
cost = tf.reduce_mean(tf.square(hypothesis - y_data))

rate = tf.Variable(0.1)
optimizer = tf.train.GradientDescentOptimizer(rate)
train = optimizer.minimize(cost)

init = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init)

print(sess.run(W))
print(sess.run(tf.matmul(W, x_data)))                    # print 행렬 곱 연산 결과

for step in range(2001):
    sess.run(train)
    if step%20 == 0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))


sess.close()
```

Matrix Transformation Coding

Machine Learning Principle
(Logistic Classification Model)

ML Principle (Logistics Classification Model)

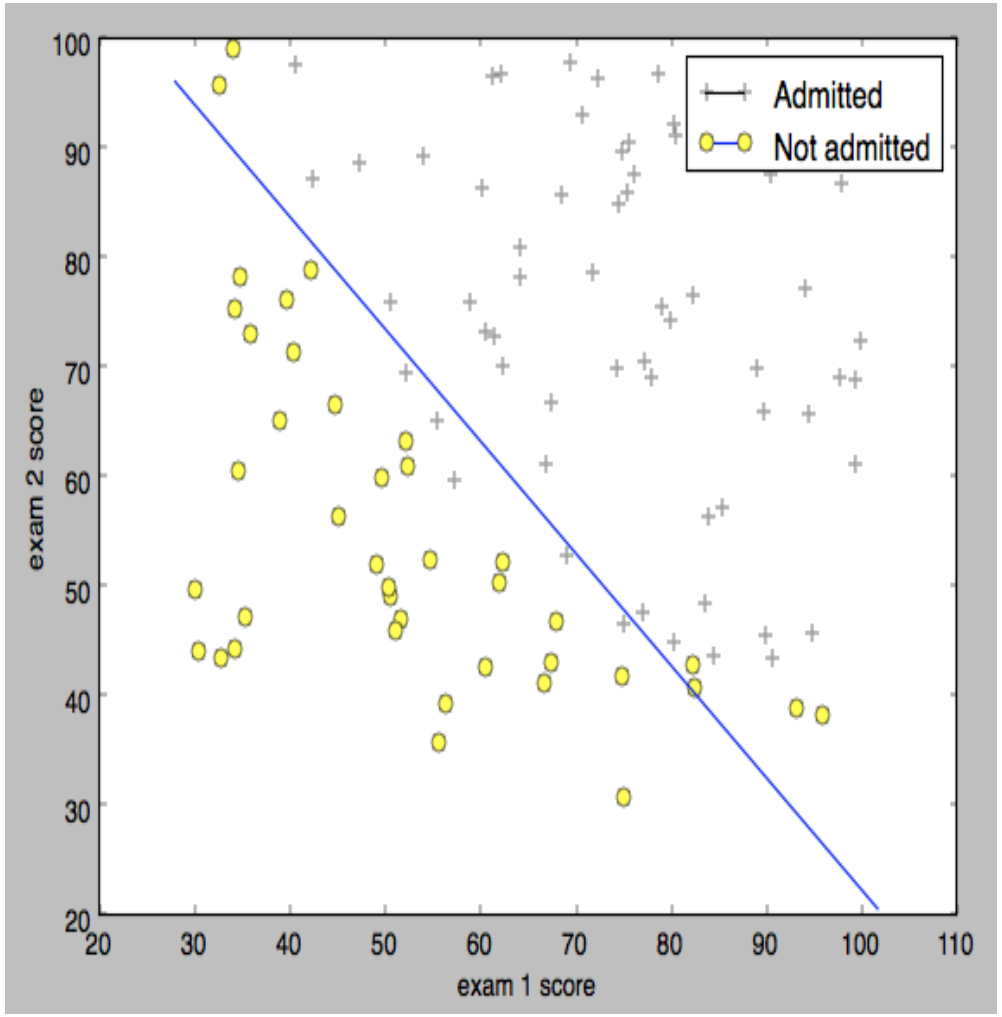
Logistics Regression (Binary Classification)

Classification

0; 1 encoding

- Spam Detection: Spam or Ham
- Facebook feed: show or hide
- Credit Card Fraudulent Transaction detection: legitimate/fraud

- Spam Detection: Spam (1) or Ham (0)
- Facebook feed: show(1) or hide(0)
- Credit Card Fraudulent Transaction detection: legitimate(0) or fraud (1)

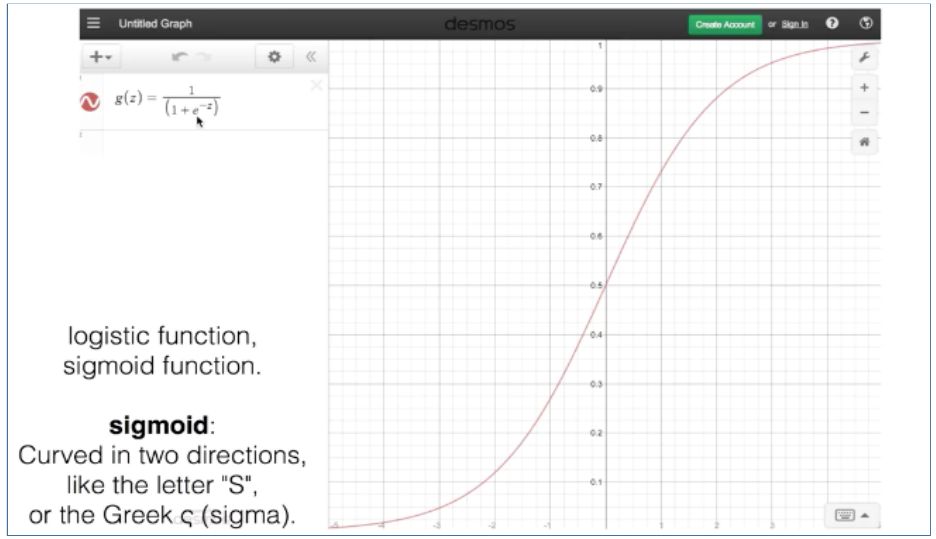


ML Principle (Logistics Classification Model)

Logistics Regression (Binary Classification)

Hypothesis H(X) :
Sigmoid Function

$$H(X) = \frac{1}{1 + e^{-X}}$$



Cost Function :

$$c(W) = -\frac{1}{m}\sum ylog(H(x)) + (1-y)log(1-H(x))$$

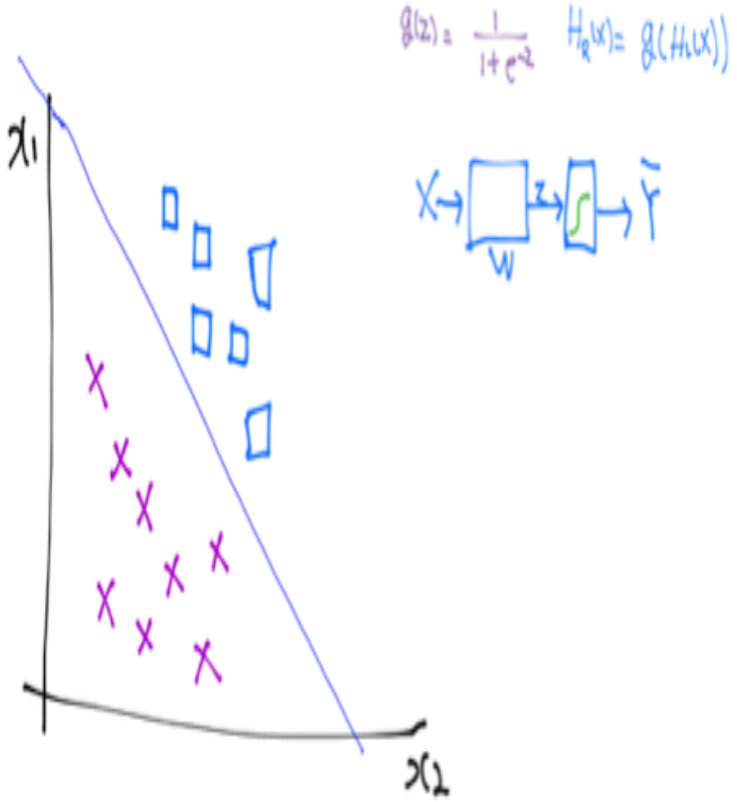
Optimizer Algorithms :

$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

Machine Learning Principle
(Multinomial Classification Model)

ML Principle (Multinomial Classification Model)

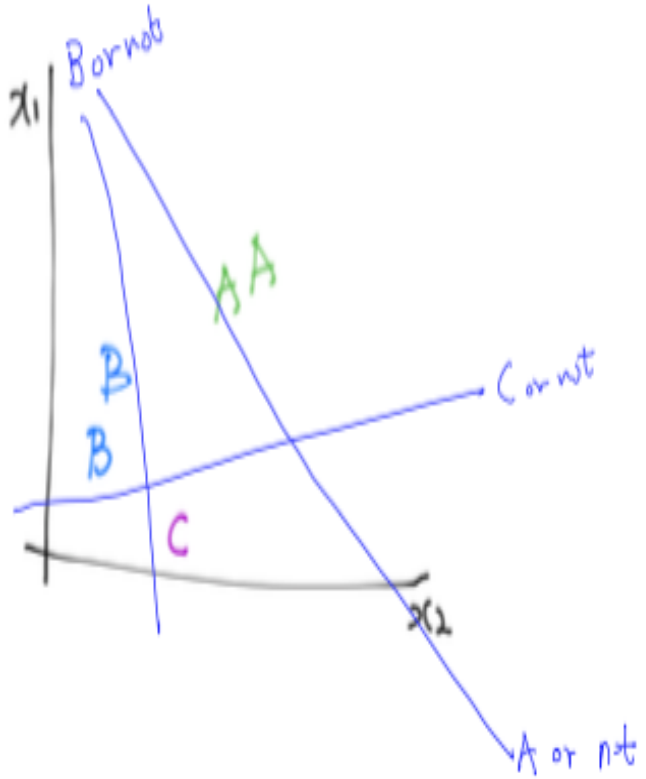
Logistic regression



Multinomial classification

x1 (hours)	x2 (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C

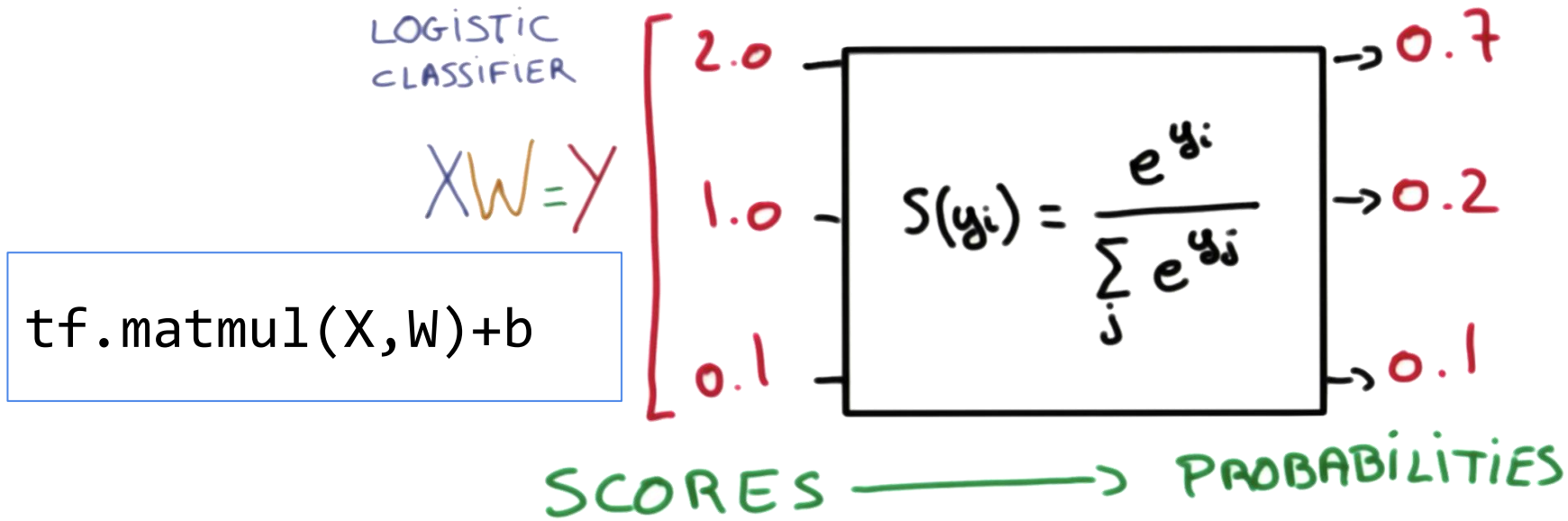
Multinomial classification



ML Principle (Multinomial Classification Model)

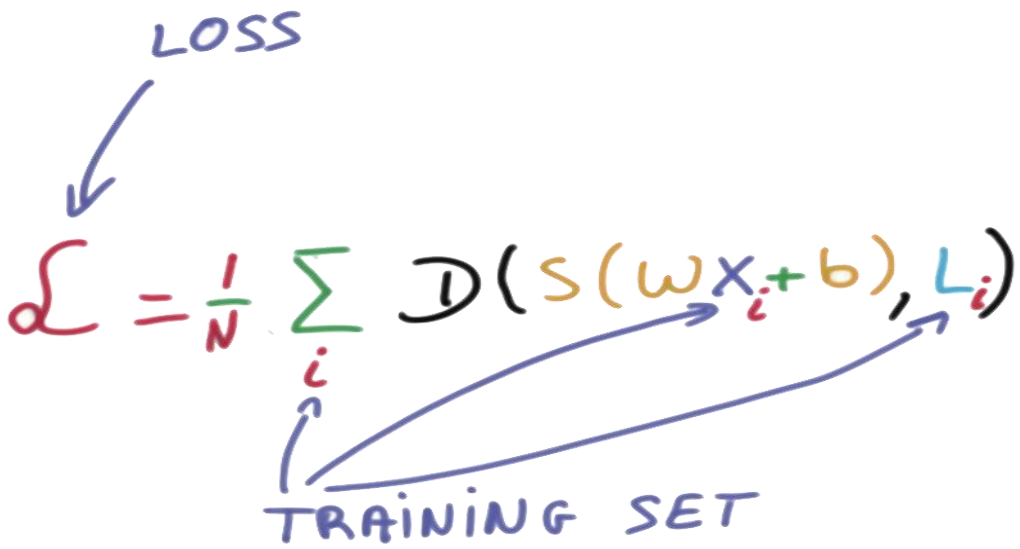
Hypothesis(H(X)) = `tf.nn.softmax(tf.matmul(X,W)+b)`

Softmax Function



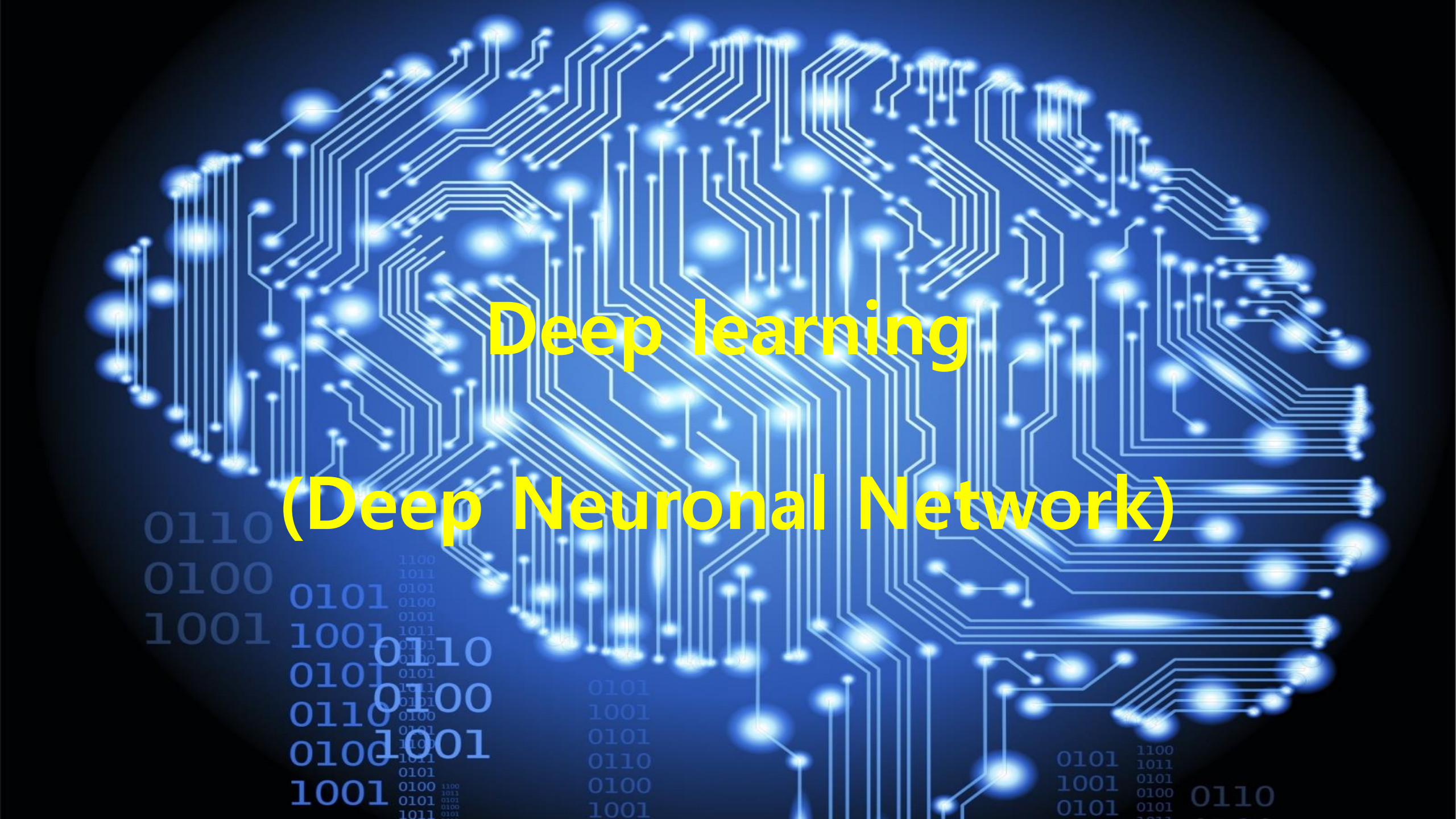
ML Principle (Multinomial Classification Model)

Cost Function :
(Cross Entropy)



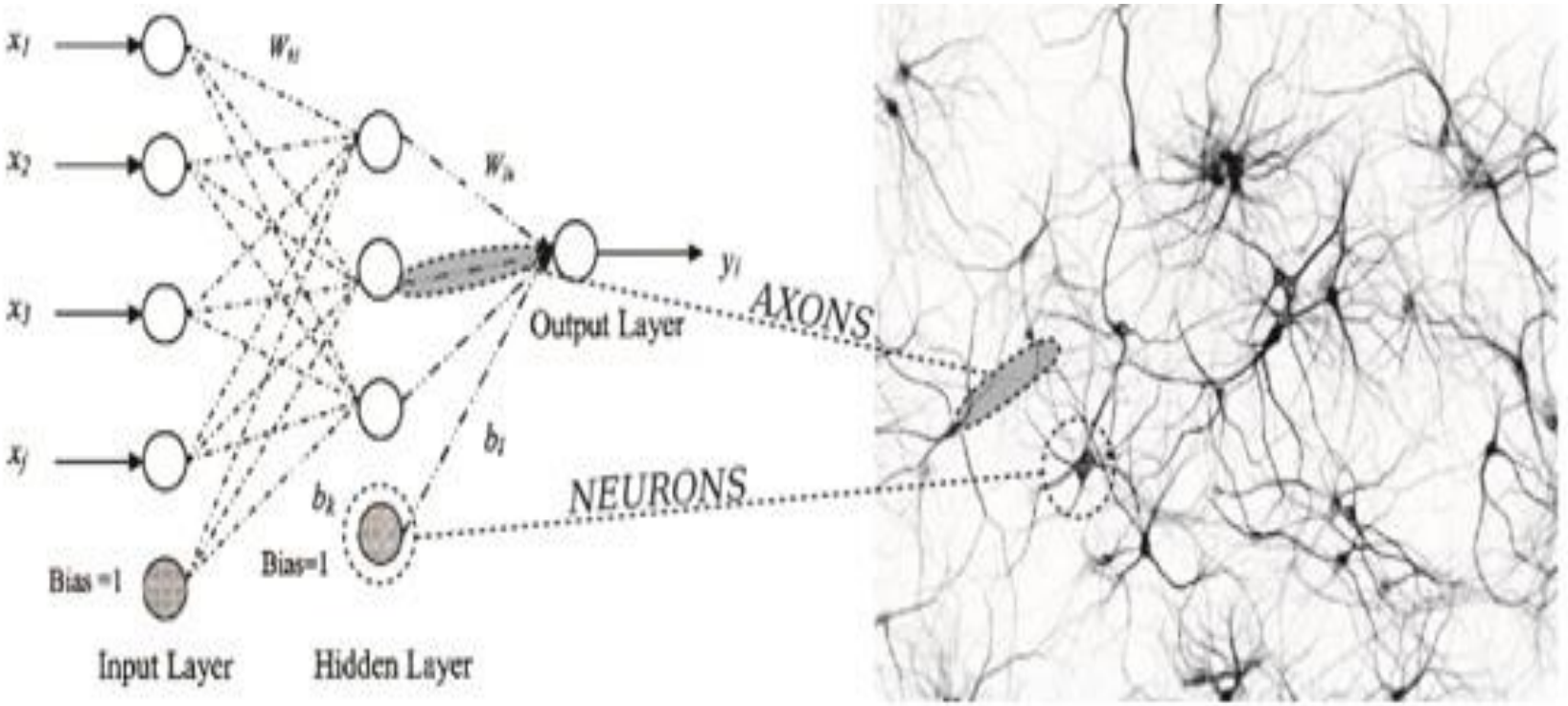

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

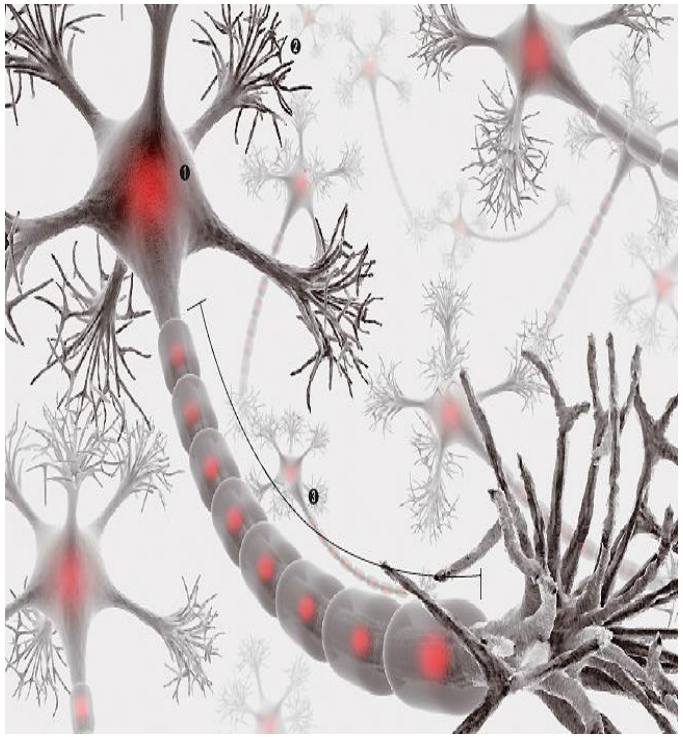
Machine Learning Principle
(Clustering Model)


Deep learning
(Deep Neuronal Network)

Deep Learning (Neuron & Neural Network) 개념

Deep Learning을 이해하기 위해서는
Neural Network(신경망)을 알아야 한다.

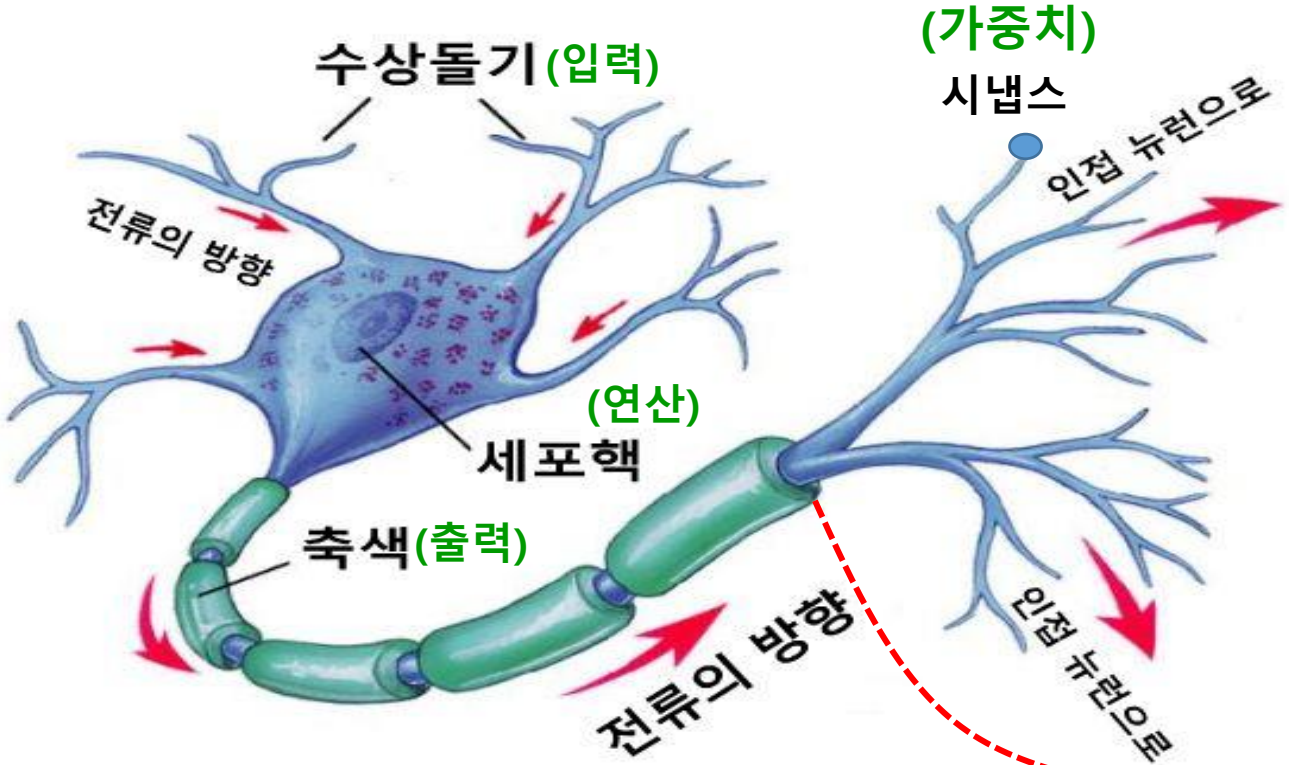




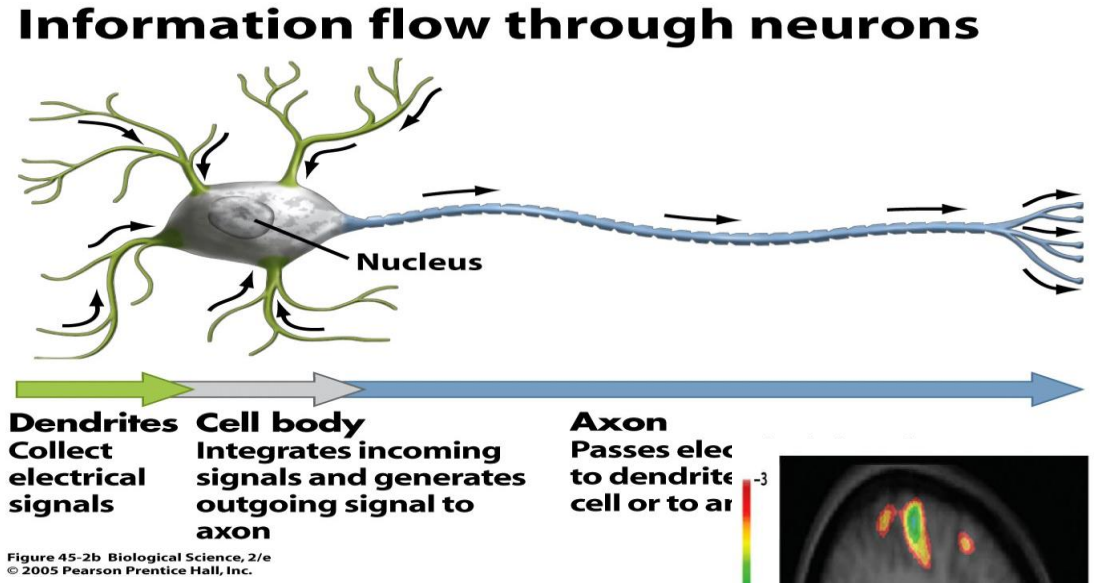
Deep Learning (Neuron & Neural Network) 개념

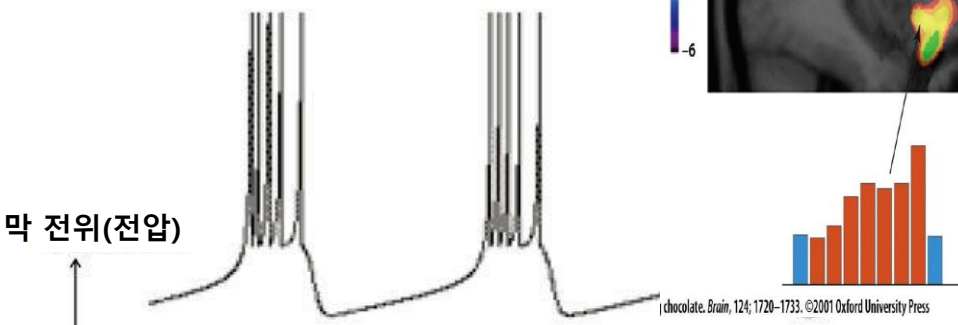
뉴런(Neuron)

여러개의 **수상돌기**에서 신호가**(입력:X_i)** 합해져서
그 값이 어느 값 이상일 경우 **축색돌기(출력:Y_i)**로
자극(신호, 정보)을 발생시킨다.



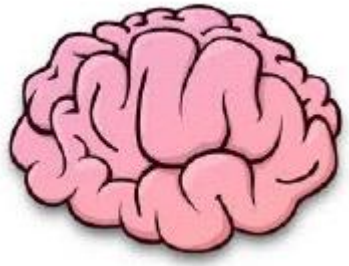
뉴런의 생물학적 도식

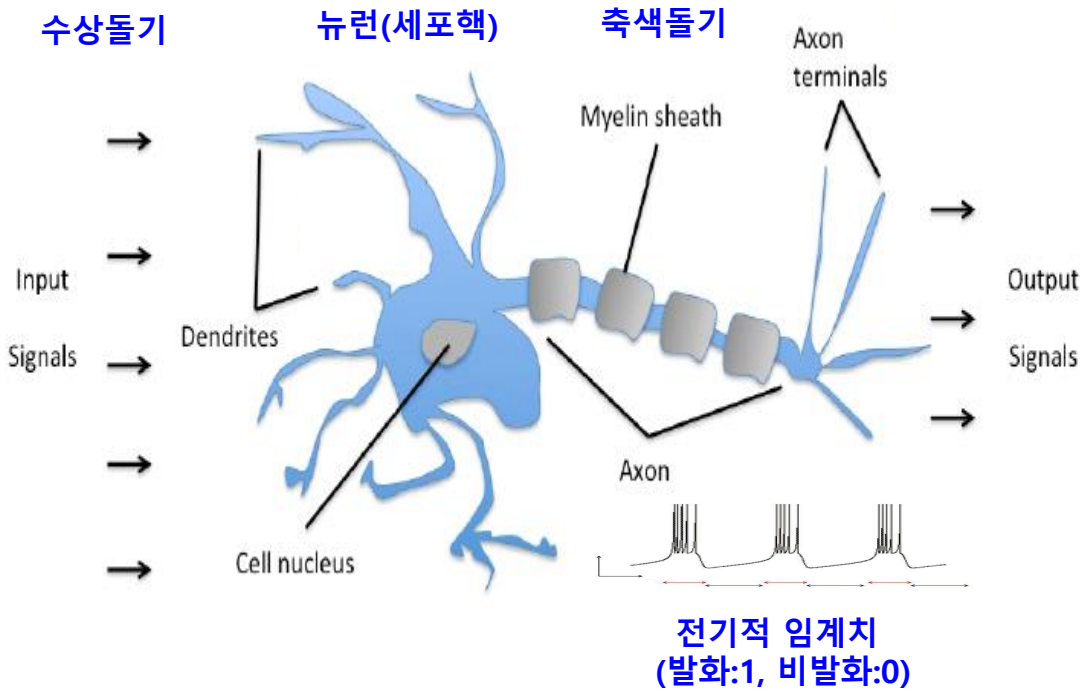


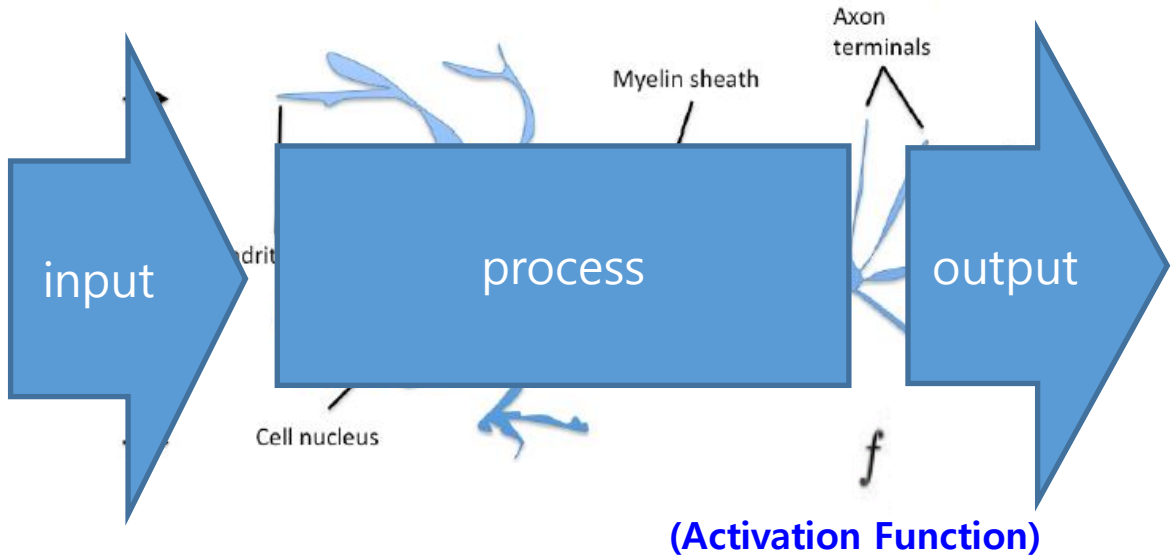


Deep Learning (Neuron & Neural Network) 개념

Neural Network(신경망)





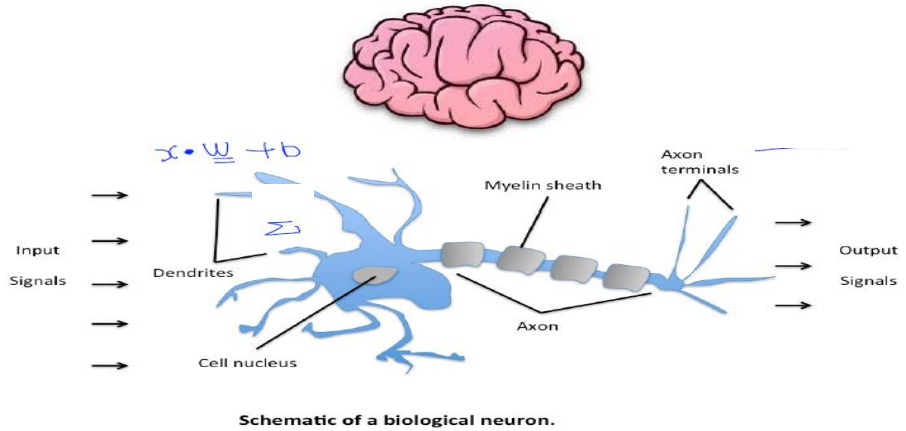


하나의 뉴런 구조(Biological Model)

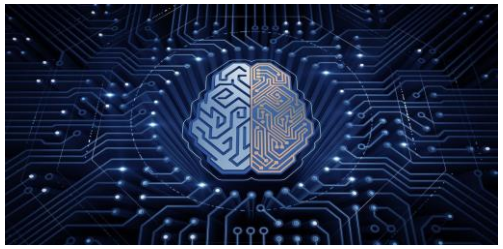
Modeling Neurons (Computer Science)

Deep Learning (Neuron & Neural Network) 개념

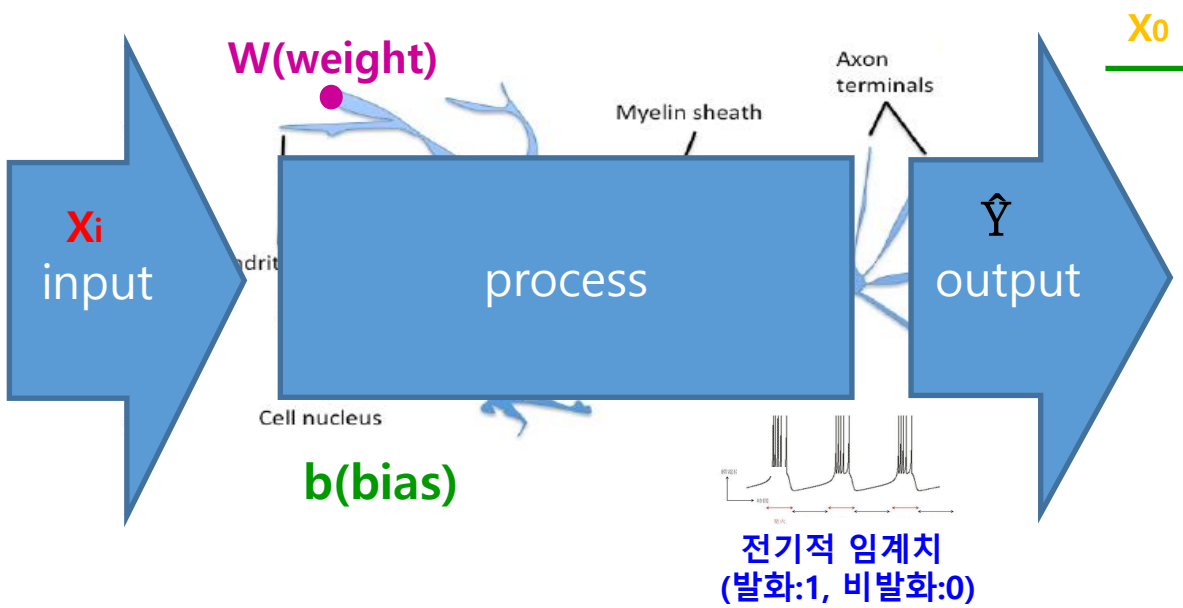
Neural Network(신경망)



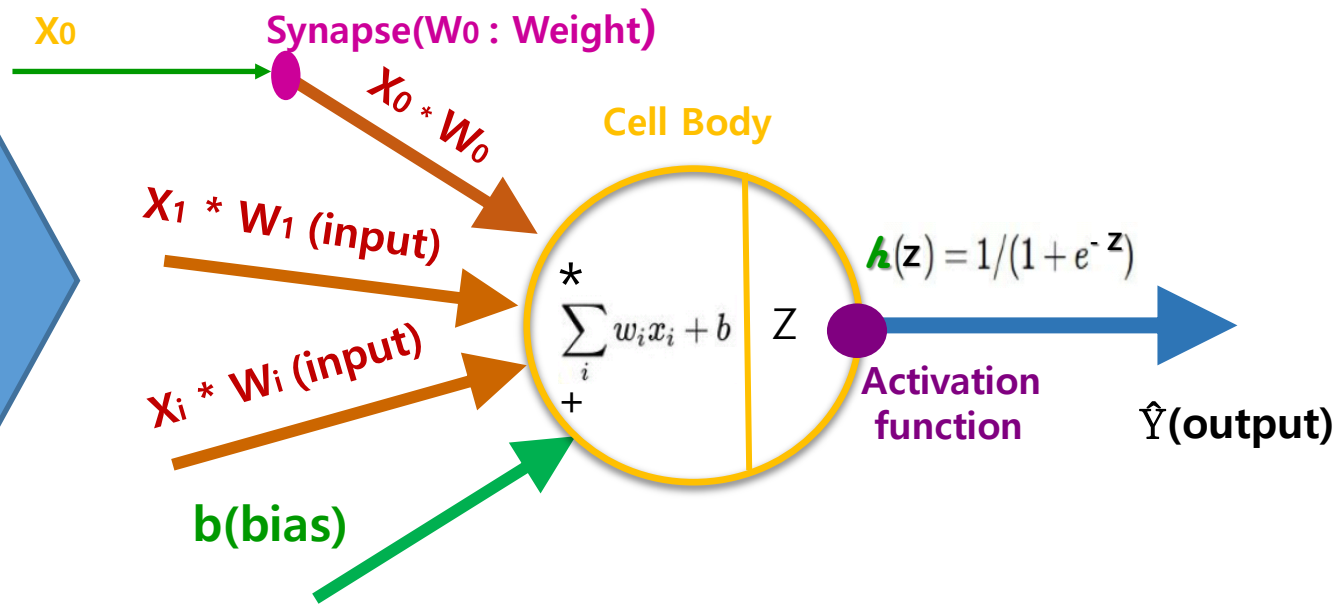
Schematic of a biological neuron.



선형회귀 함수 **(y = w*x + b)**
Single Neuron의 수학적 Model



Modeling Neurons (Computer Science)

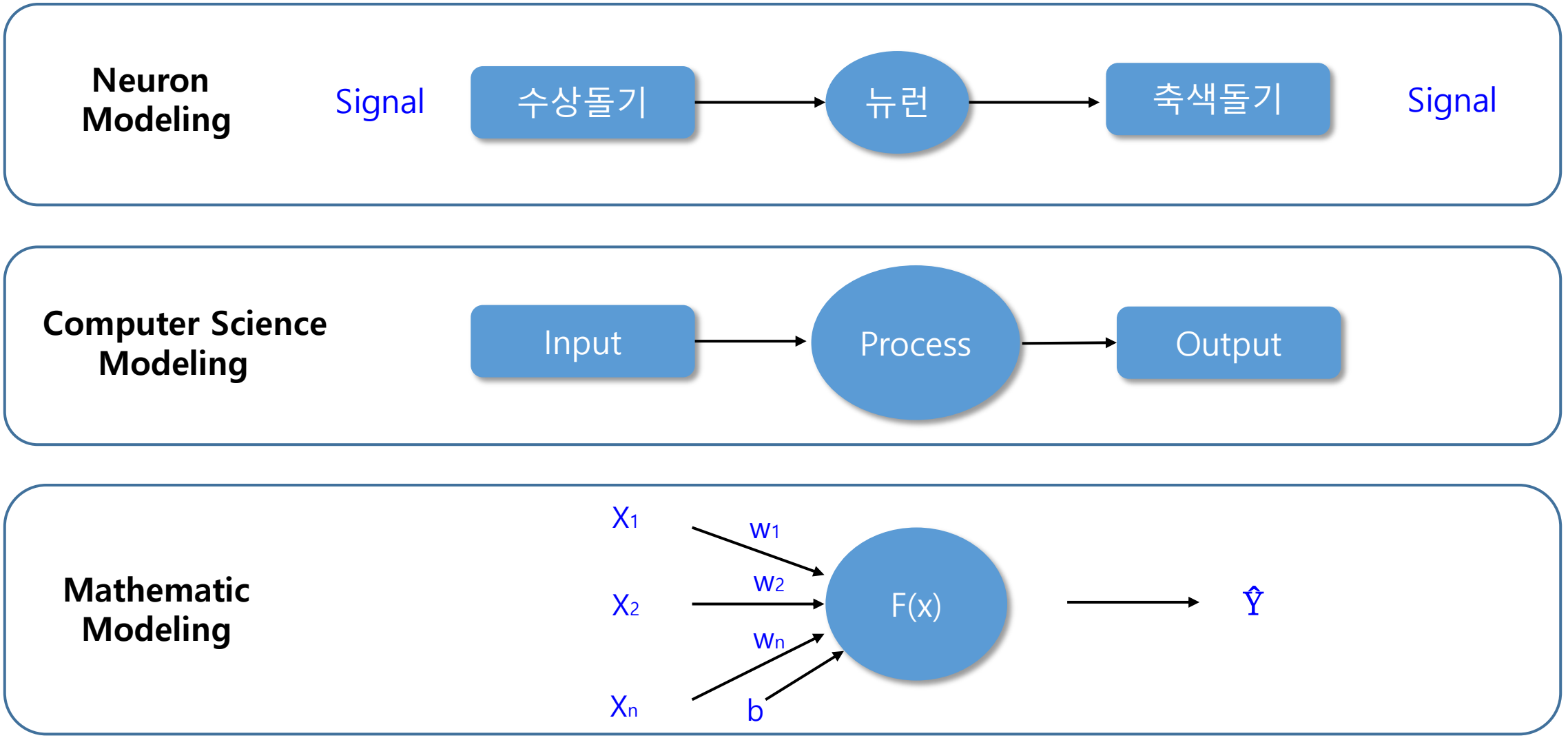


General Model for Neuron (수학적 Model)

Deep Learning (Neuron & Neural Network) 개념

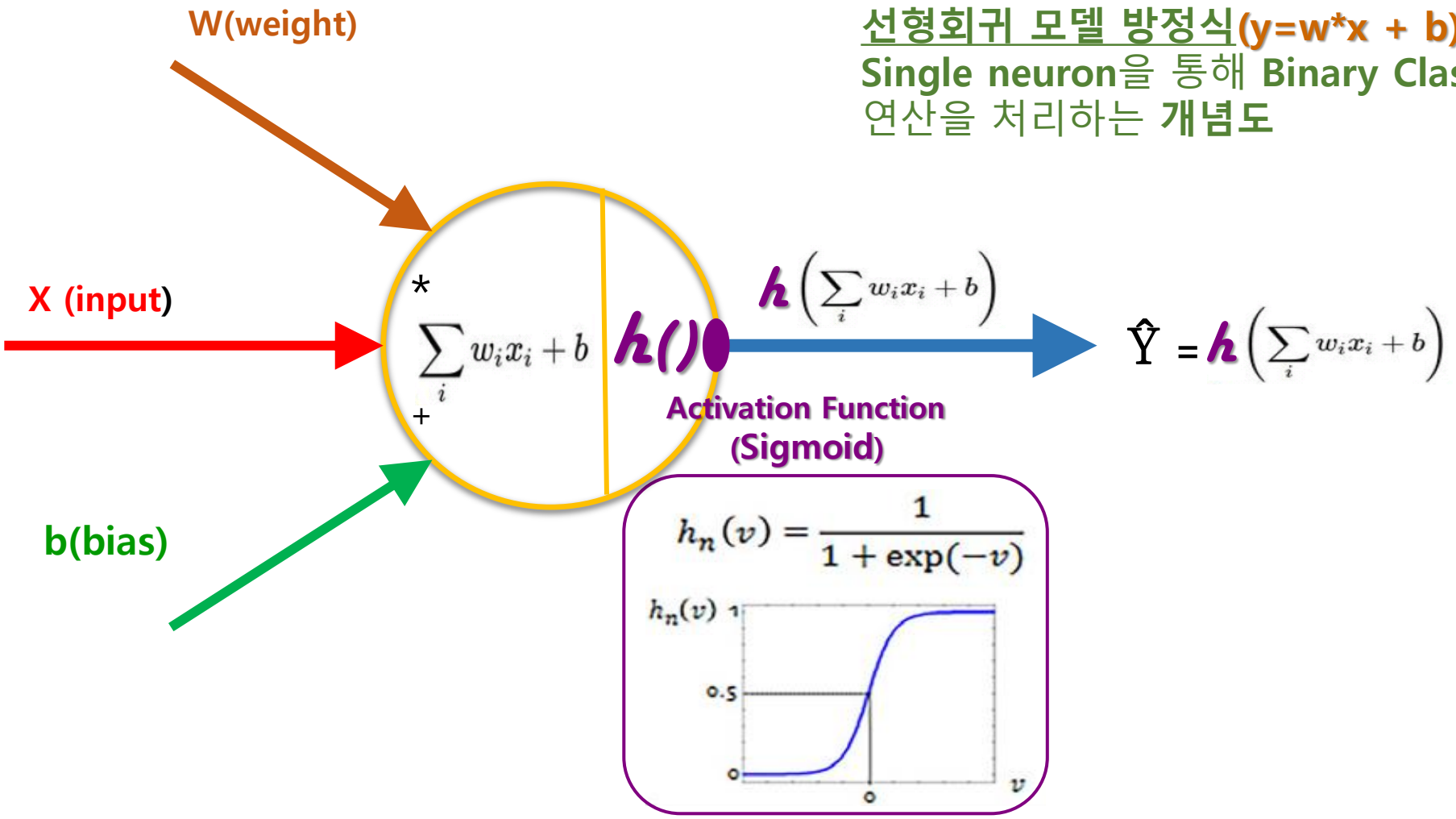
Neural Network(신경망)

자, 다시 정리해 보면.....



Deep Learning (Neuron & Neural Network) 개념

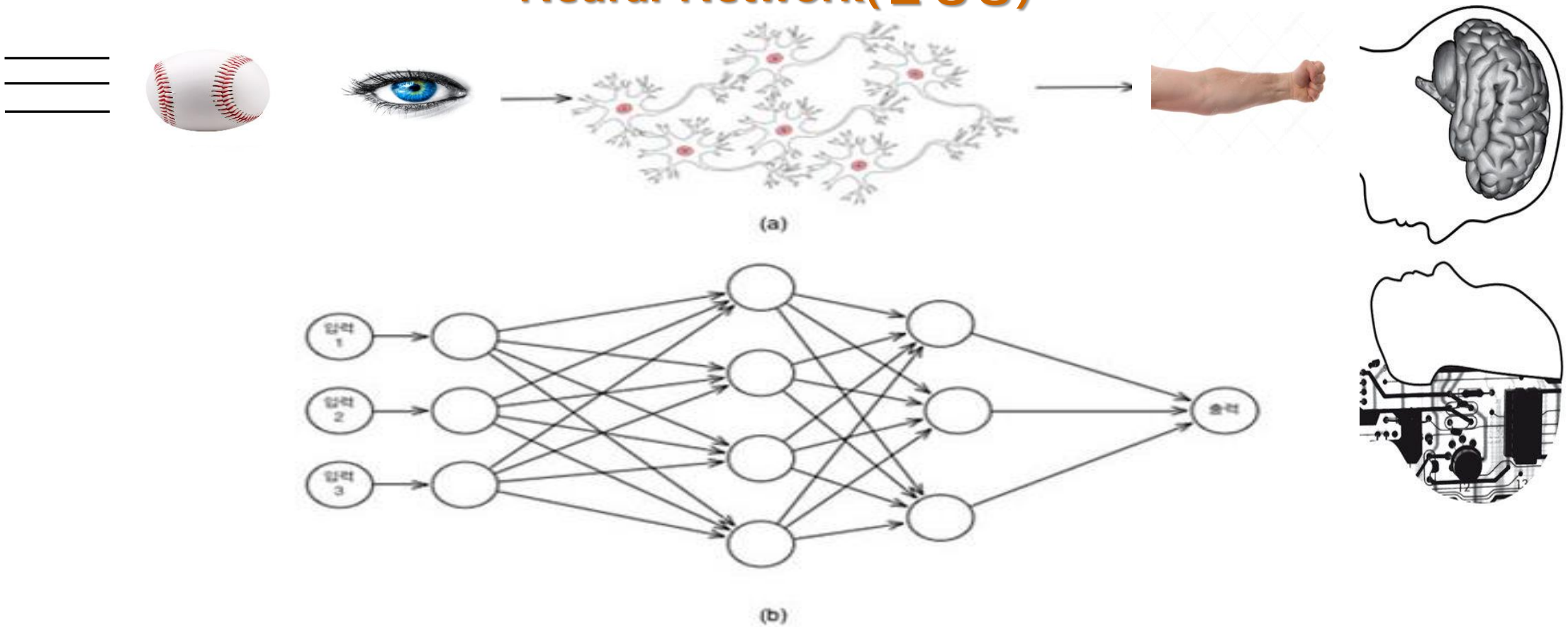
하나의 인공 뉴런 개념도



선형회귀 모델 방정식(y=w*x + b)을
Single neuron을 통해 Binary Classification
연산을 처리하는 **개념도**

Deep Learning (Neuron & Neural Network) 개념

Neural Network(신경망)



생물학적 신경망 (a)와 인공 신경망 (b)의 비교

인간 생물학적 신경망이 감각기관으로부터 **공이 날아오는 것을 감지(입력)**하여 이에 반응해 근육을 움직여 **공을 받는 반응(출력)**을 하듯이, **인공 신경망** 또한 주어진 **입력 값**에 반응해서 **출력 값**을 내놓는다 **(예측, 추론, 분석 한다).**

인간 생물학적 신경망이 공을 받는 행위가 **훈련**을 통해 숙련되듯, **인공 신경망**도 **학습**을 통해 이런 기능을 갖춘다.

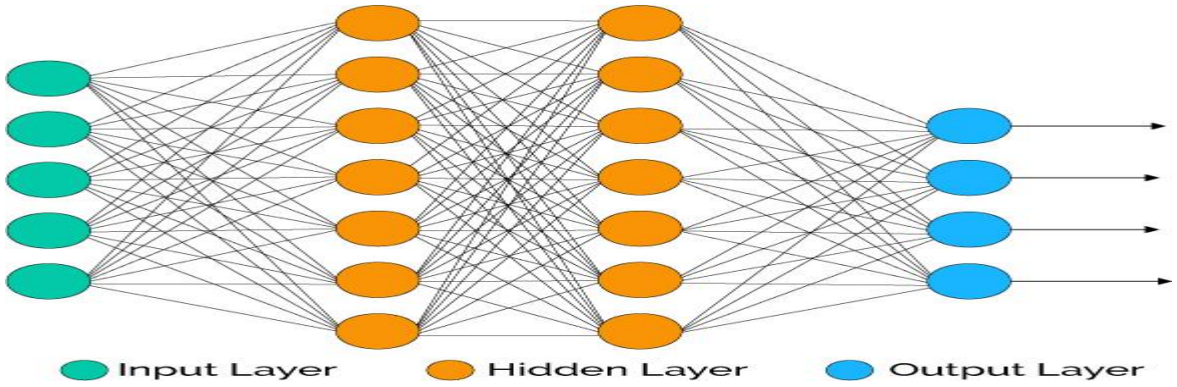
Deep Learning (Neuron & Neural Network) 개념

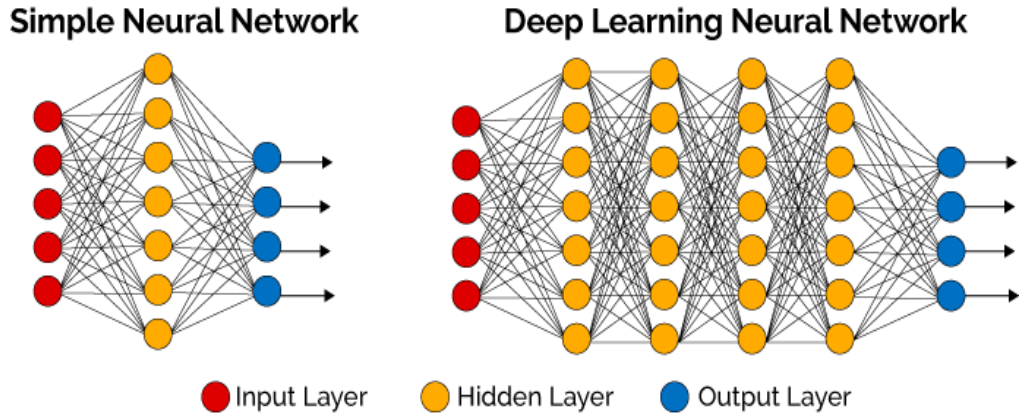
Neural Network(신경망)

인공 신경망의 구조

전형적인 신경 회로망은 일련의 층(Layer)으로 배열 된 많은 수의 인공 신경세포(artificial neurons)을 포함합니다

전형적인 인공 신경망(네트워크)에서, 여러 층(Layer)로 구성된다. Multi-Layer Perceptron(MLP)





- **Input layer (입력 층)** : 학습하거나 인식하거나 처리 할 외부 세계로부터 입력을 받는 유닛 (뉴런으로 인식 안하고 단순 Buffer)

- **Hidden layer** : 이 유닛은 입력 Layer와 출력 Layer 사이에 있으며(Hidden Layer), 이 기능은 입력을 받아 어떤 방식 으로든 출력 단위가 사용할 수 있는 무언가 변환(연산 처리)하는 기능.

- **Output Layer (출력 층)** : 이 모든 작업을 통해 학습된 방법에 대한 결과 정보를 출력(응답) 장치 기능

Neural Network(신경망)

신경망의 4 가지 다른 용도

- **예측**
  신경망은 주어진 입력(Training Data Set)으로부터 기대되는 출력(예측, 추론)을 산출하도록 훈련
  예 : 주식 시장 예측, 성능, 장애, 용량 예측, 음성, 이미지 인식 등등

- **분류**
  신경망은 주어진 Raw Data들의 일정한 패턴 또는 데이터 집합을 미리 정의 된 클래스로 분류
  예 : 이상징후, Traffic 패턴, 보안 탐지 등등

- **클러스터링**
  신경 네트워크는 데이터(Training Data Set)의 사전 지식 없이 데이터의 특징을 파악하고 서로 같은 범
  주로 분류 예 : 금융 이상 거래, smart factory(장비 감시 및 감지) 등등

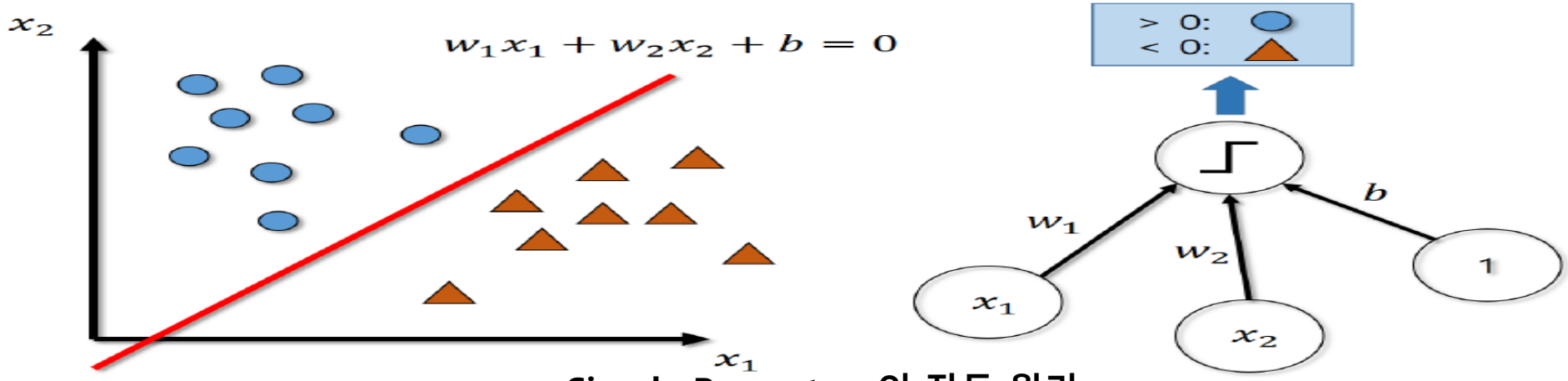
- **Association (연관) 분석**
  신경망은 특정 패턴을 기억하도록 훈련 될 수 있으므로 네트워크에 노이즈 패턴이 제시 될 때 네트워크는 메모리에 있는 가장 가까운 패턴과 연관 시키거나 폐기
  예 : 인식, 분류 및 클러스터링 등을 수행하는 Hopfield Networks에 응용

Applications of Neural Network

Application	Architecture / Algorithm	Activation Function
Process modeling and control	Radial Basis Network	Radial Basis
Machine Diagnostics	**Multilayer Perceptron**	**Tan - Sigmoid Function**
Portfolio Management	Classification Supervised Algorithm	Tan - Sigmoid Function
Target Recognition	**Modular Neural Network**	**Tan - Sigmoid Function**
Medical Diagnosis	**Multilayer Perceptron**	**Tan - Sigmoid Function**
Credit Rating	Logistic Discriminant Analysis with ANN, Support Vector Machine	Logistic function
Targeted Marketing	Back Propagation Algorithm	Logistic function
Voice recognition	**Multilayer Perceptron, Deep Neural Networks (Convolutional Neural Networks)**	**Logistic function**
Financial Forecasting	**Backpropagation Algorithm**	**Logistic function**
Intelligent searching	Deep Neural Network	Logistic function
Fraud detection	**Gradient - Descent Algorithm and Least Mean Square (LMS) algorithm.**	**Logistic function**

Deep Learning (Neuron & Neural Network) 개념

선형(직선 형태) Perceptron Learning 예제

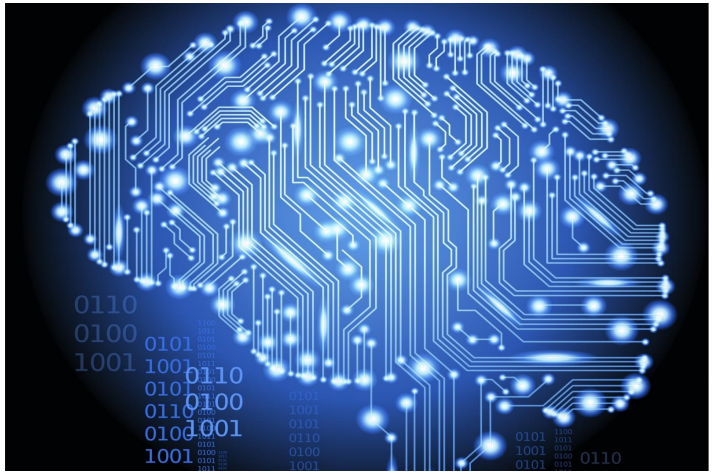


Simple Perceptron의 작동 원리

상기 그림과 같이 먼저 **가중치(w1, w2)와 바이어스(b) 값**을 임의의 값으로 초기화. 학습시킬 **Raw Data(Training Data) 집합**을 입력하여 **학습을 시키면**서 perceptron의 **예측 값 H(x)**이 **목표 출력 값(정답) y값**과 **다르면 가중치(w1, w2)와 편향(b) 값**을 **변경**(cost 함수 값, 즉 에러 값이 작은 방향)하고, **예측 값**과 **목표 출력 값**이 **같으면 변경하지 않고 최적의 알고리즘으로 fit(Model화)**

- **목표 출력 값 y = 1** 인데 **예측 값 H(x) = -1**인 경우 : 가중치 w 값을 w + x로 **증가**
- **목표 출력 값 y = -1** 인데 **예측 값 H(x) = 1**인 경우 : 가중치 w 값을 w – x로 **감소**

이와 같은 방식으로 **학습의 단계를 반복적으로 수행**하면서 가중치(w)와 바이어스(b) 값을 **변경(갱신)**하면서 올바른 분류(에러가 가장 적게 하는 함수를 위한 w와 b를 탐색)를 하기 위해 **학습해 나가는 Deep Learning 원리**

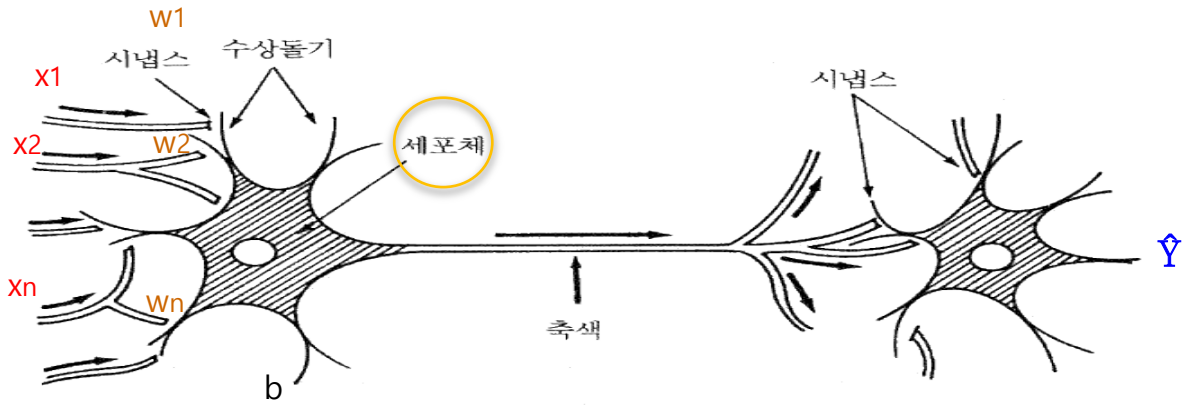


Machine Learning vs Deep Learning 차이점 (Deep Learning 특징)

ML vs DL 차이점 (Deep Learning 특징)

Machine Learning
기법들과의 차이점 1 :

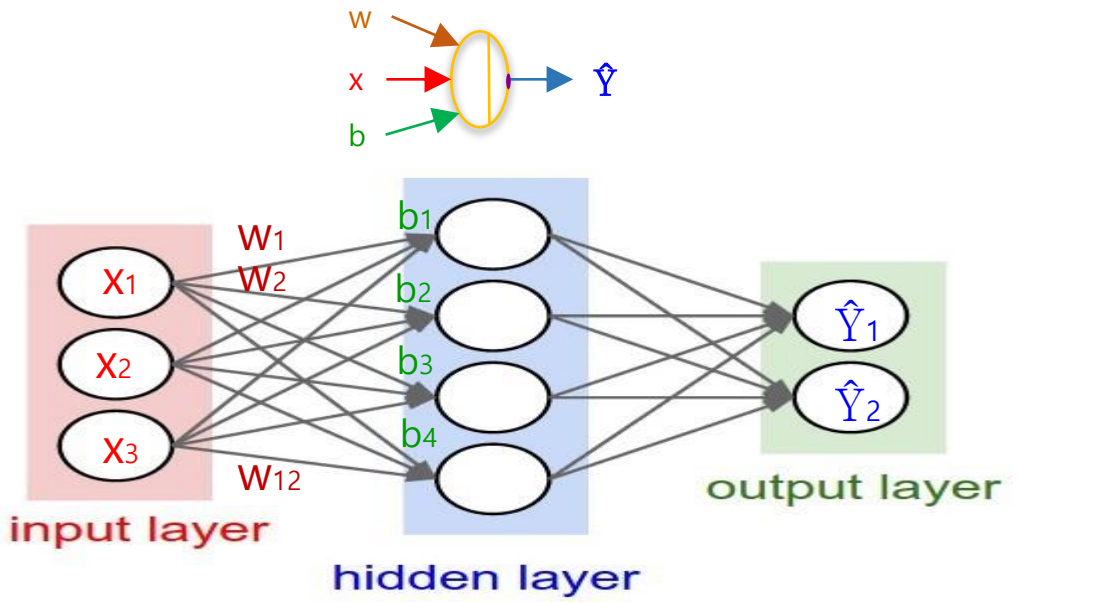
생물학적 인간 뇌(Neuron) 의 정보(신호) 전달 과정을 모방해서 모델링

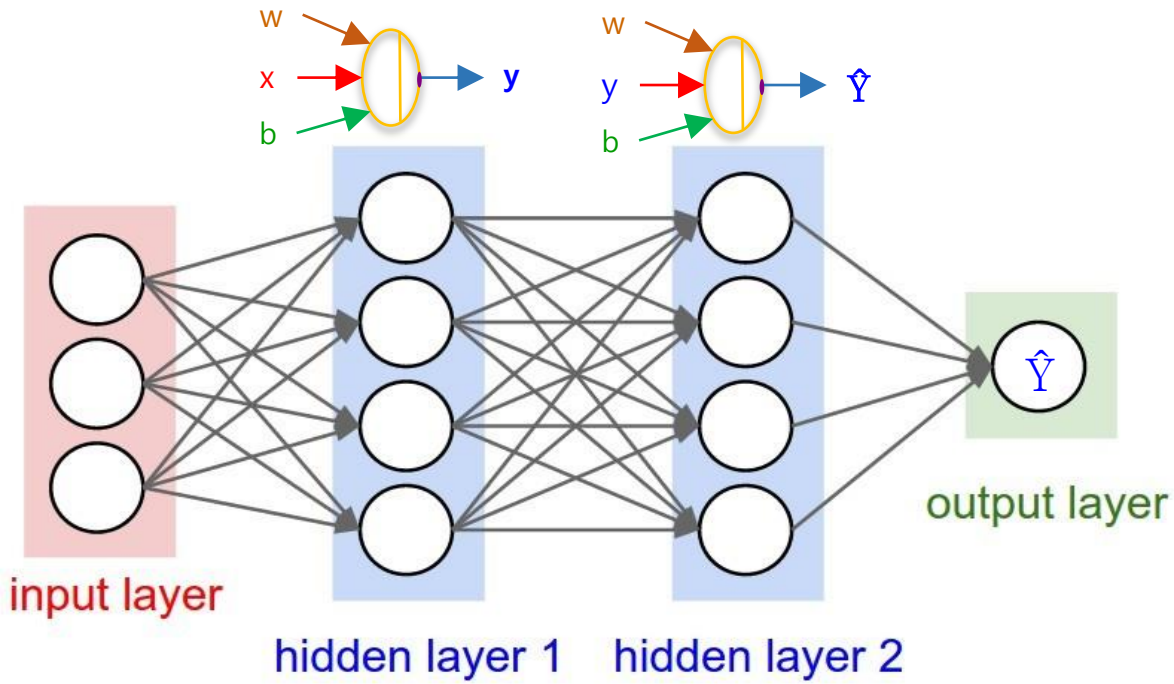


Machine Learning
기법들과의 차이점 2 : → **엄청 복잡한 함수(문제) (알고리즘), 복잡한 패턴 분류를 할 수 있다.**

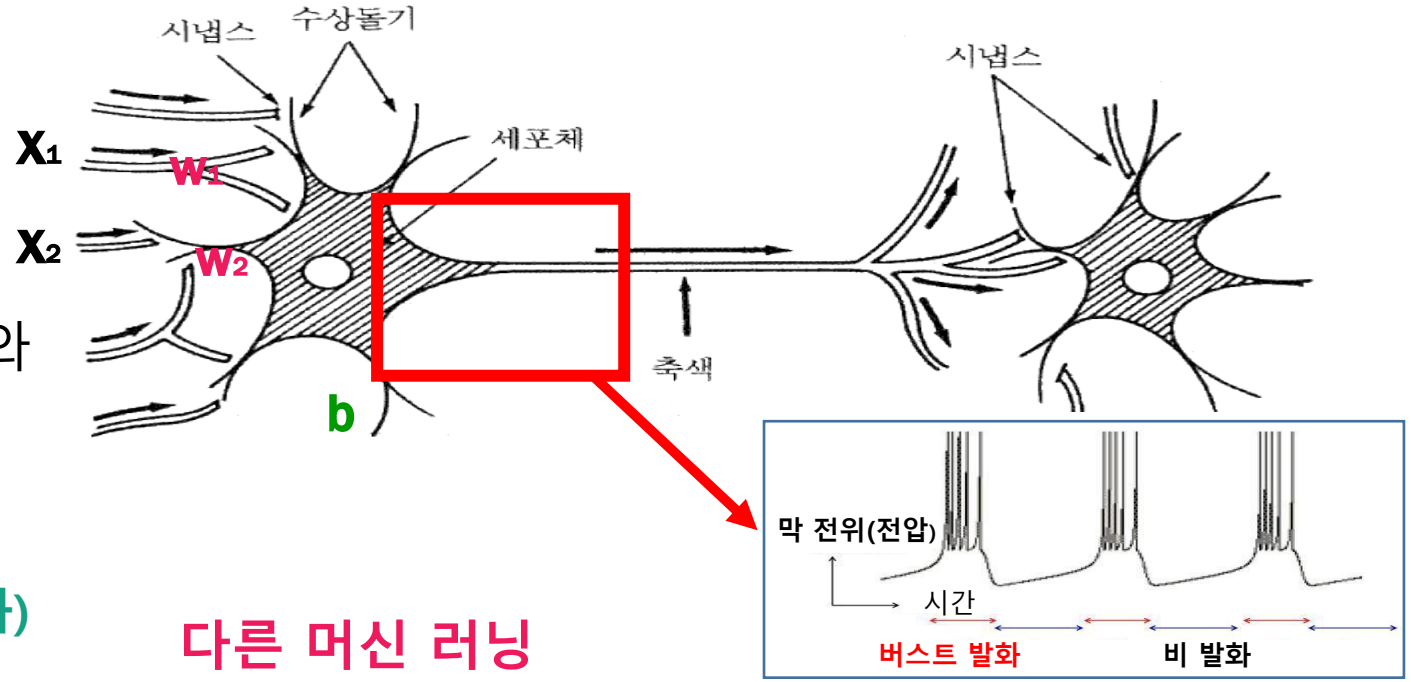
다층 구조(Multiple Layer) 제공

Hidden layer가 **2개 이상**인 NN를 **Deep Learning** 함





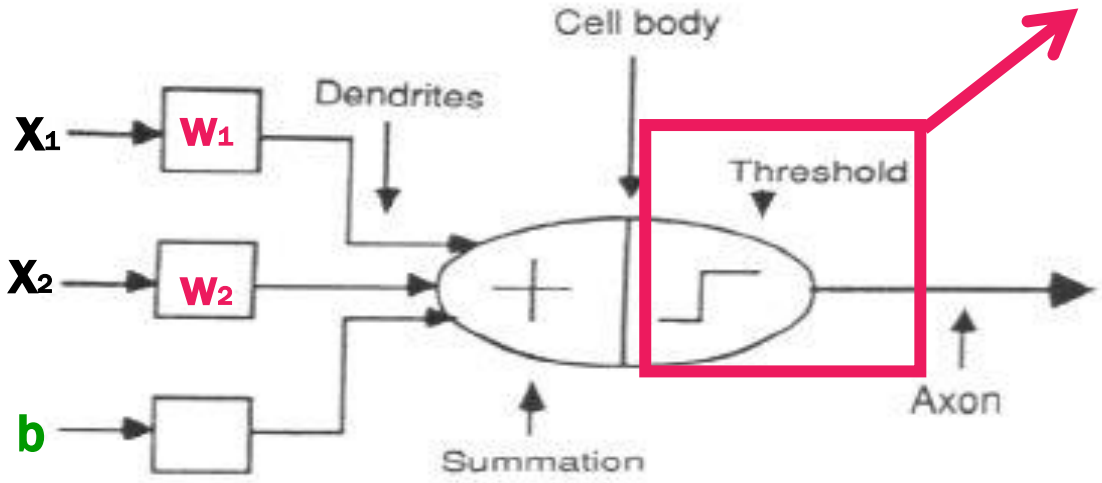
ML vs DL 차이점 (Deep Learning 특징)



생물학적 뉴런(Neuron) 발화와

인공 뉴런의 활성화 함수

(Activation Function : 발화)



다른 머신 러닝 기법들과의 차이점 3:

Nonlinear(복잡한) activation function

Sigmoid Function : $h(z) = 1/(1+e^{-z})$

tanh Function : tanh(x)

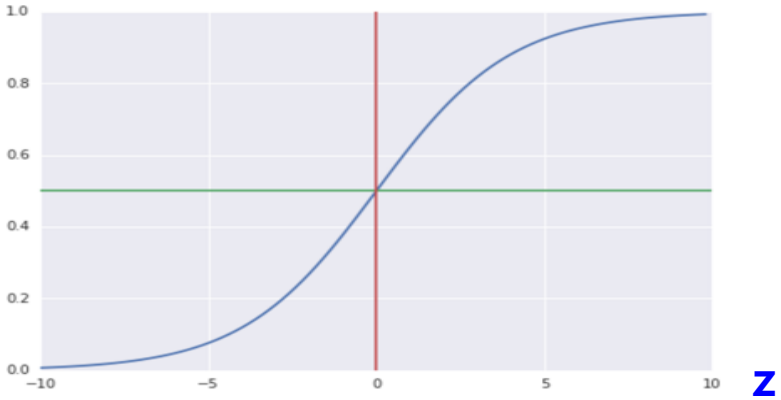
ReLU Function : max(0,x)

ELU, MaxOut, Leaky ReLU .....

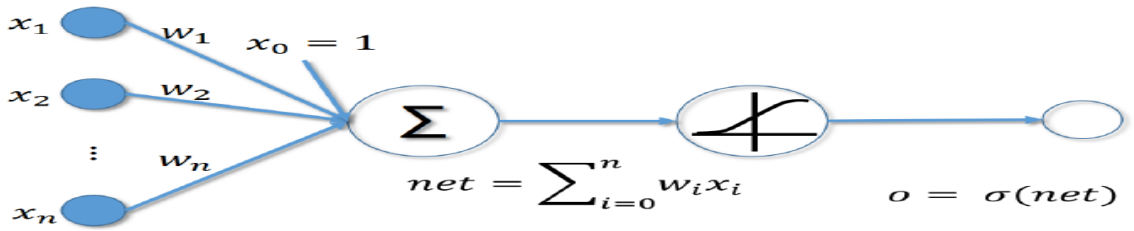
ML vs DL 차이점 (Deep Learning 특징)

Activation Function (활성화 함수 : Sigmoid Function)

h(z)

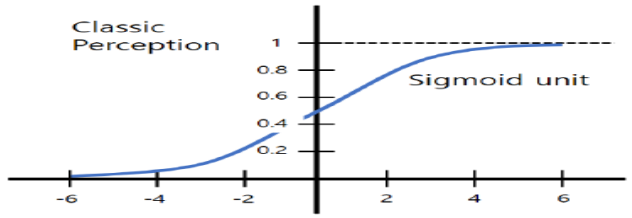


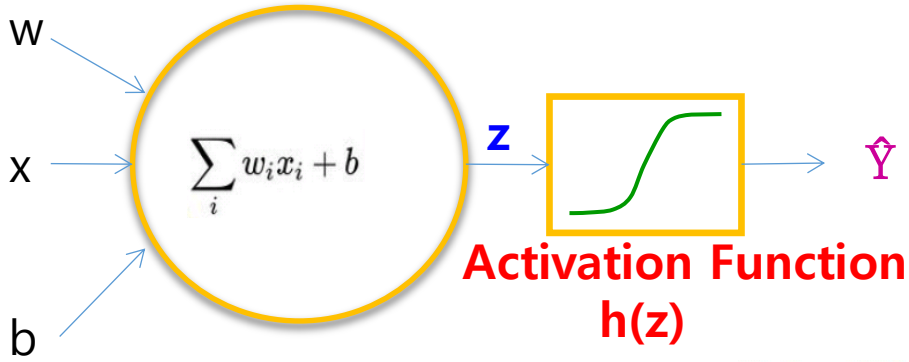
z



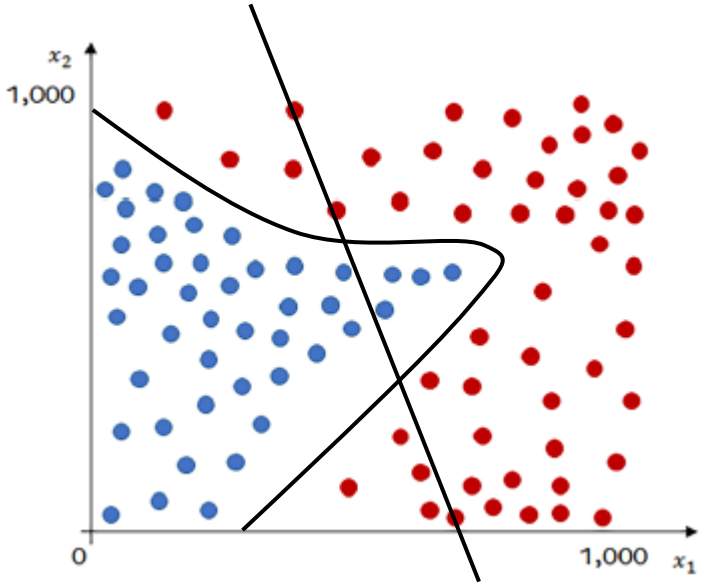
Sigmoid function is **Differentiable**

$$\frac{df(x)}{dx} = f(x)(1 - f(x))$$



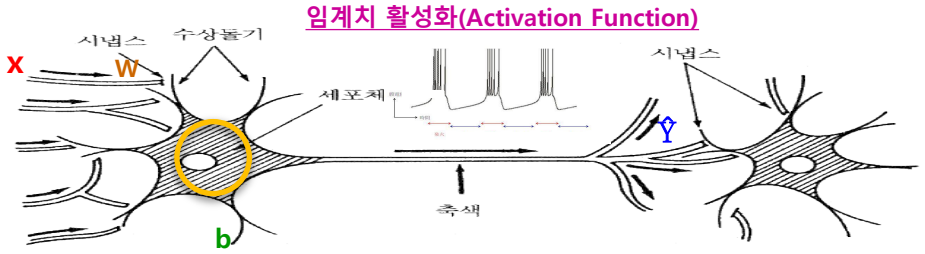


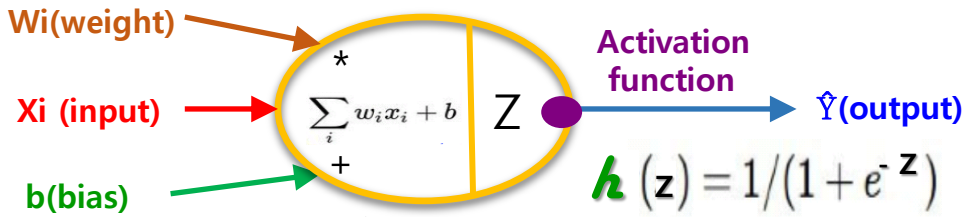
Sigmoid Function : $h(z) = 1/(1+e^{-z})$

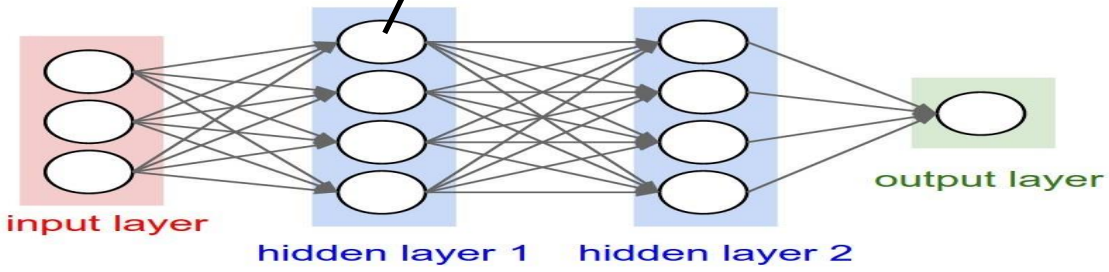


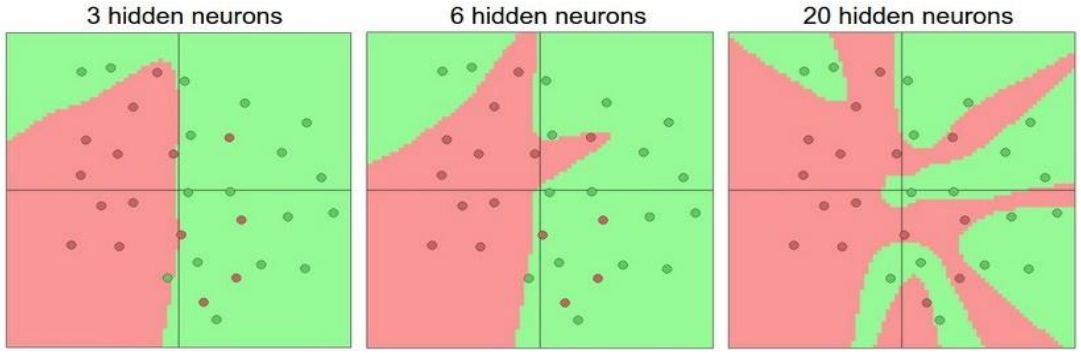
$$Z = \sum_i w_i x_i + b \qquad \hat{Y} = h(z) = 1/(1+e^{-z})$$

ML vs DL 차이점 (Deep Learning 특징)



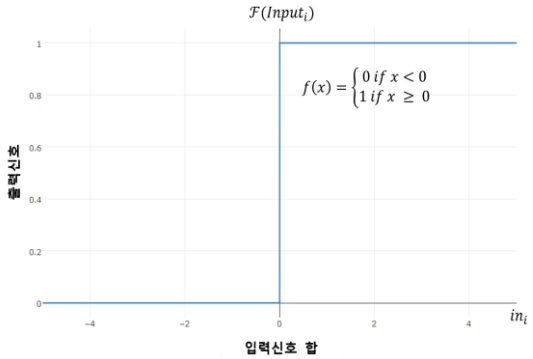




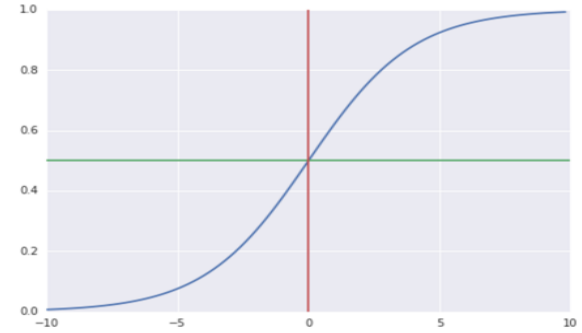


다시 정리를 하면.....

Activation Function (비선형의 활성화 함수)

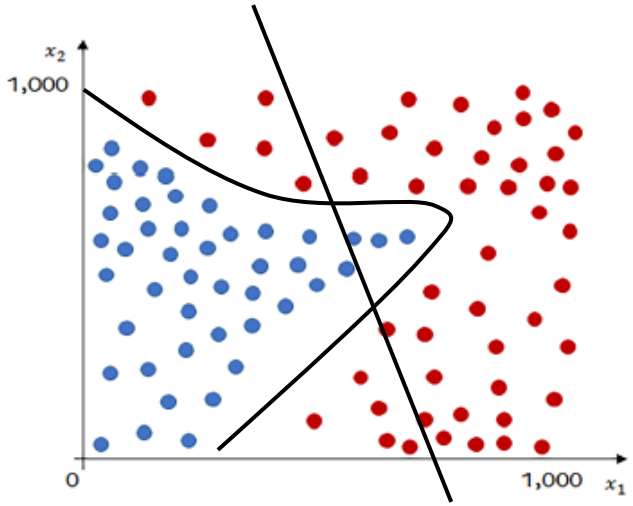


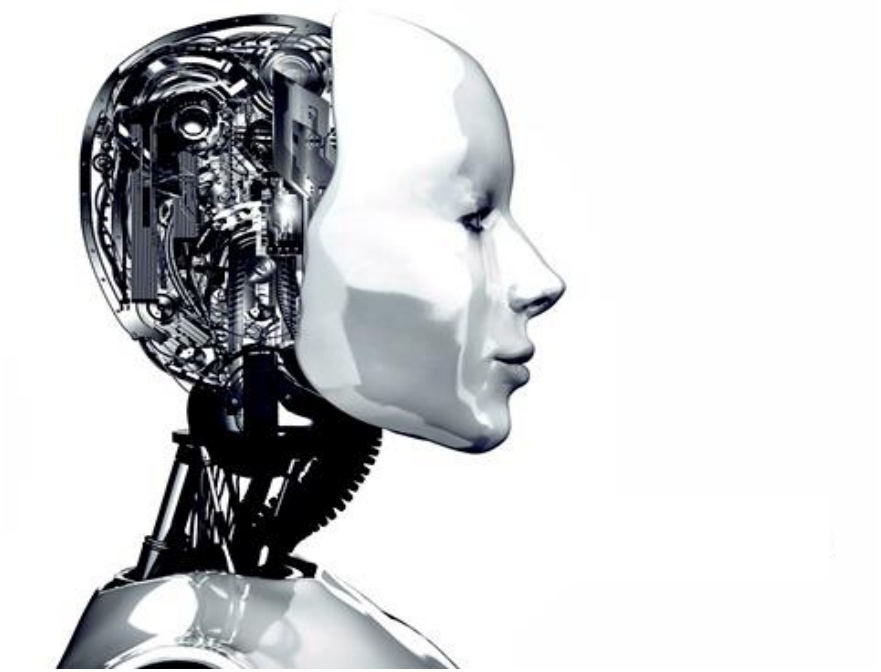
STEP Function(선형 활성화 함수)



Sigmoid Function(비선형 활성화 함수)

Multilayer Neuron은 Non linear Function
(비선형의 복잡한 문제 해결) : XOR 문제 해결

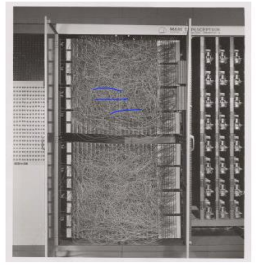




Artificial Intelligence History

AI(Artificial Intelligence) History

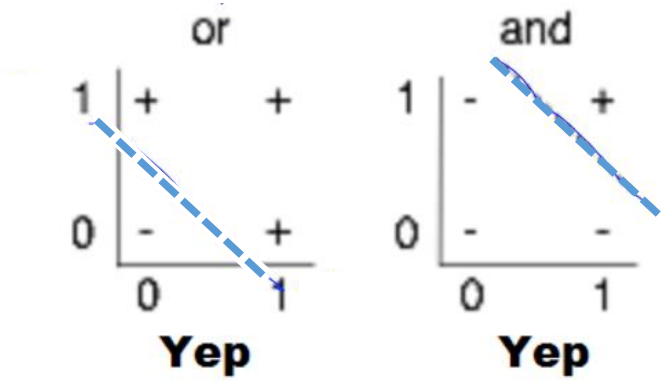
H/W Implementation(57년)



Frank Rosenblatt, ~1957: Perceptron

Widrow and Hoff, ~1960: Adaline/Madaline

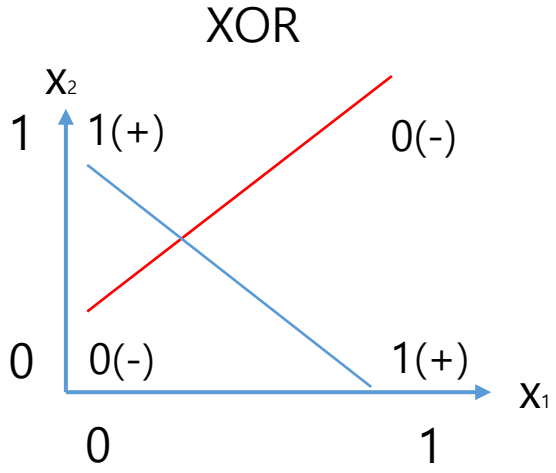
(Simple) AND/OR problem: linearly separable?



False Promises

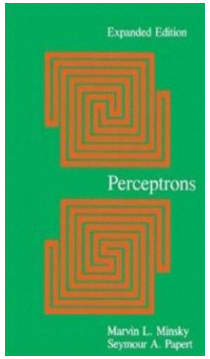
"The Navy revealed the embryo of an electronic computer today that *it expects will be able to walk, talk, see, write, reproduce itself* an be conscious of its existence … Dr. Frank Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers" The New York Times July 08, 1958

(Simple) XOR problem: linearly separable?



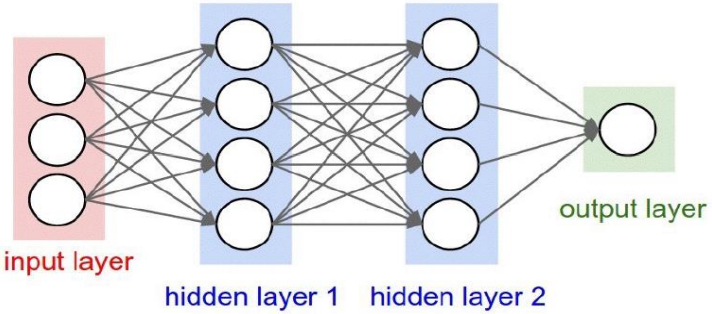
Perceptrons (1969)

by Marvin Minsky, founder of the MIT AI Lab



- We need to use MLP, multilayer perceptrons (multilayer neural nets)
- No one on earth had found a viable way to train MLPs good enough to learn such simple functions.

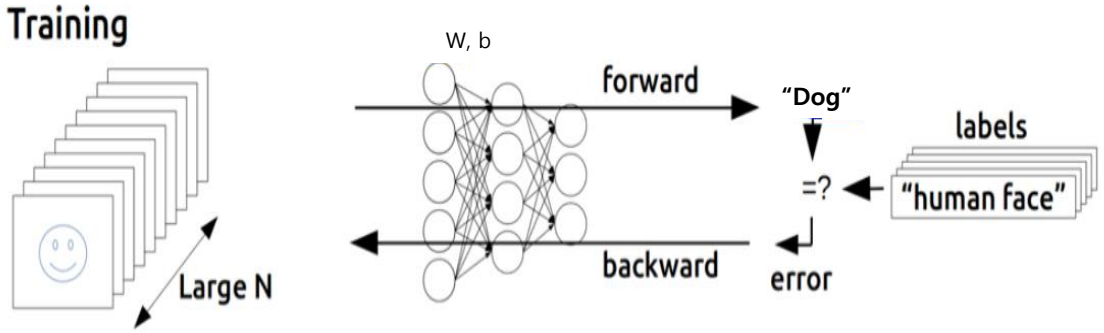
"No one on earth had found a viable way to train*"



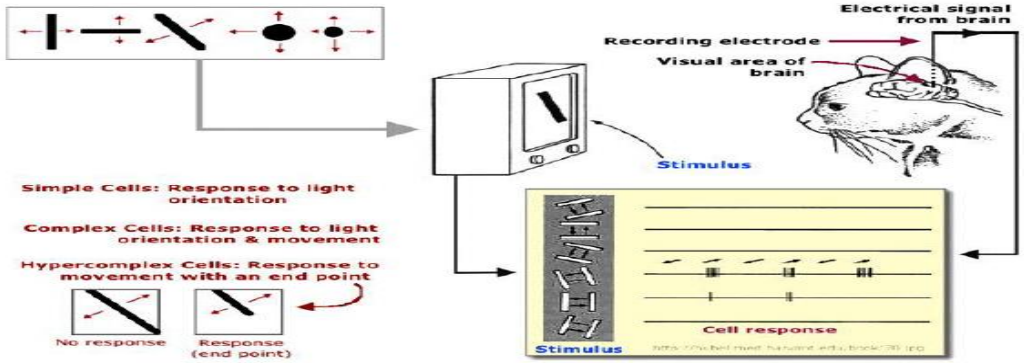
*Marvin Minsky, 1969

AI(Artificial Intelligence) History

Backpropagation ※
(1974, 1982 by Paul, 1986 by Hinton)



Convolutional Neural Networks



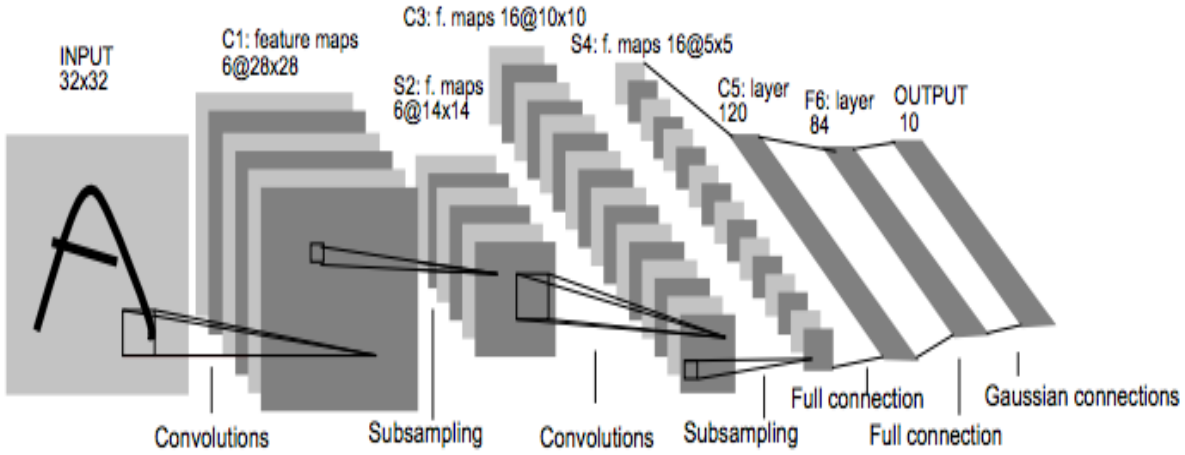
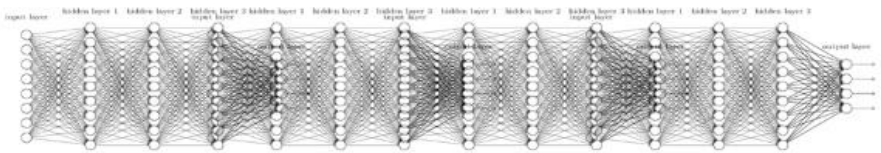


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

A BIG problem

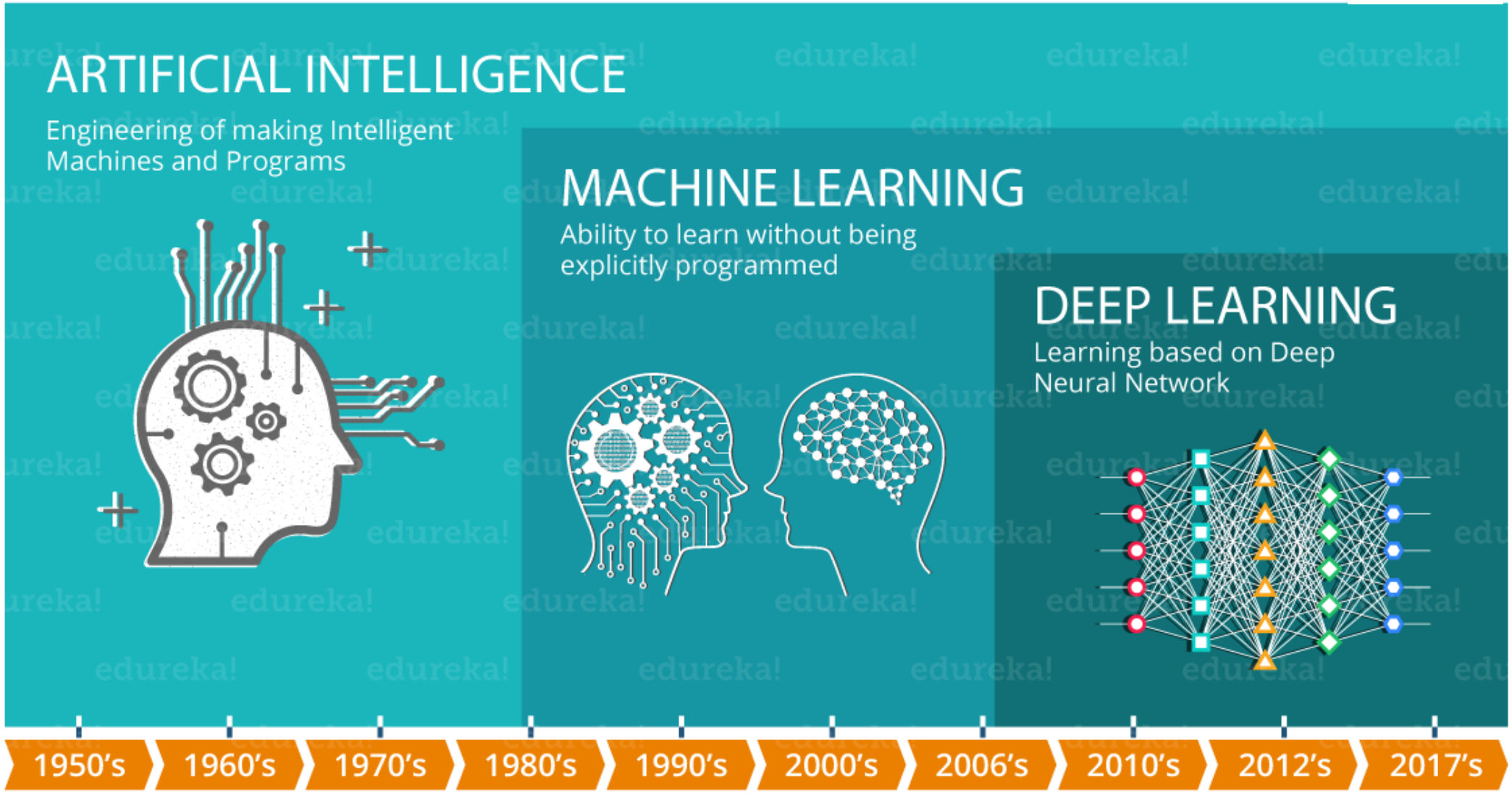
- Backpropagation just did not work well for normal neural nets with many layers
- Other rising machine learning algorithms: SVM, RandomForest, etc.
- **1995** "Comparison of Learning Algorithms For Handwritten Digit Recognition" by LeCun et al. found that this new approach worked better



http://neuralnetworksanddeeplearning.com/chap6.html

AI(Artificial Intelligence) History

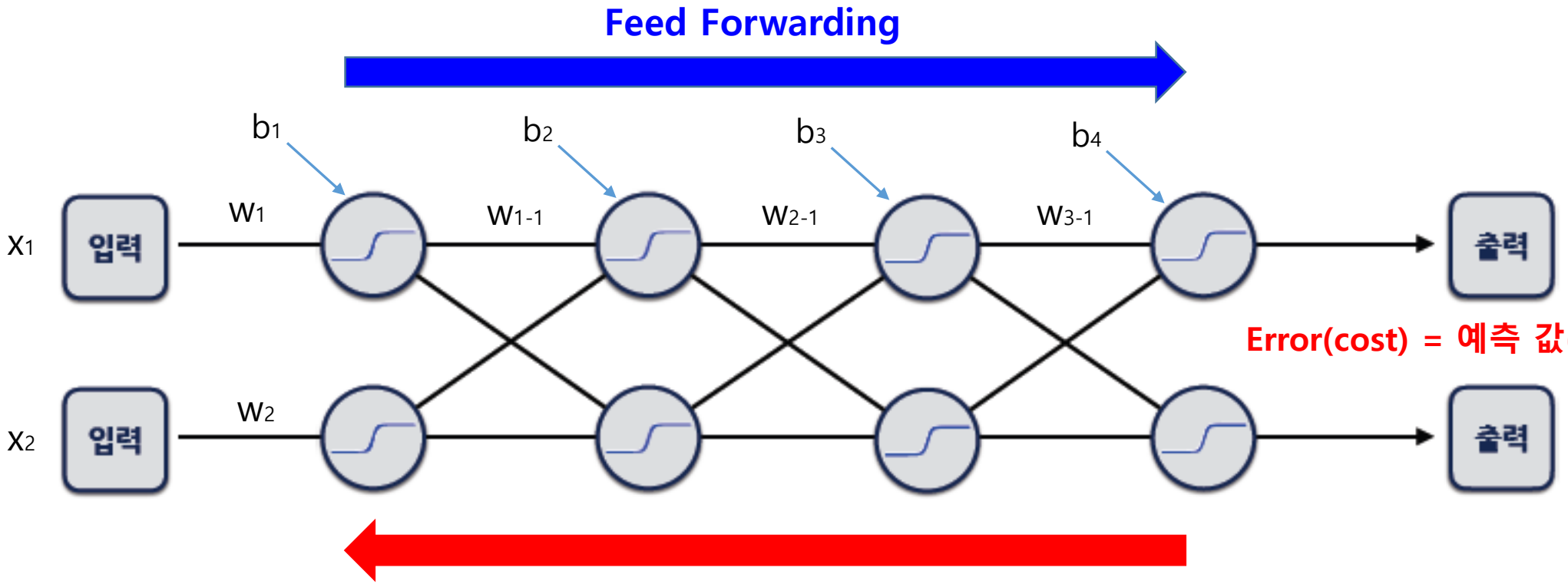
AI Technology Timeline




Deep Learning Principle

Deep Learning Principle

Neural Network의 Learning 방법

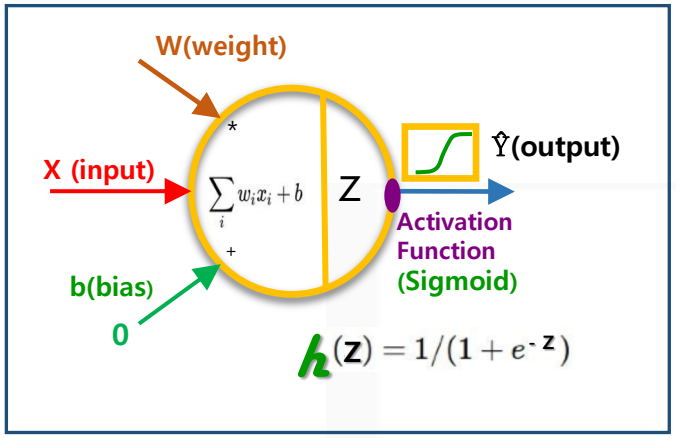


뭐를 뒤로 전달 하는가? 현재 **오차가 있는 정보(예측 값-정답)를 "미분(기울기)"** 한 값.
W(가중치)에 대한 Error(오차)에 대한 미분 값

$$\frac{\partial E_d}{\partial w_i}$$

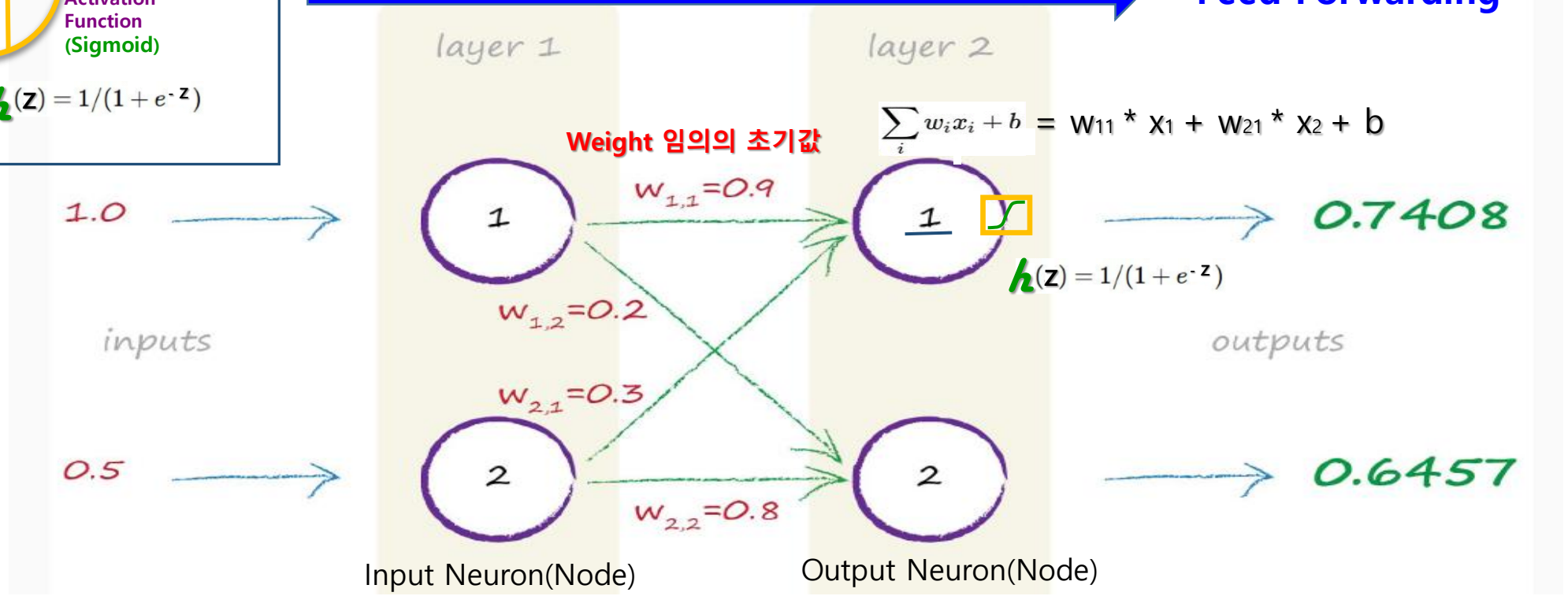
오차(에러:예측 값-정답) 를 최소화하기 위한 **w, b 값을 보정(변화)** 시키기 위해 각 Node마다 미분(편 미분)하고, 곱하고, 더하고 하여 역 방향으로 반복하여 **w, b 값(cost Function 최소화)을 업데이트 함**.

Deep Learning Principle



Neural Network의 Learning 방법

Feed Forwarding



출력 1번 Node = w11(0.9) * X1(1.0) + W21(0.3) * X2(0.5) = 0.9 + 0.15 = 1.05 ➔

Activation Function (Sigmoid Function)

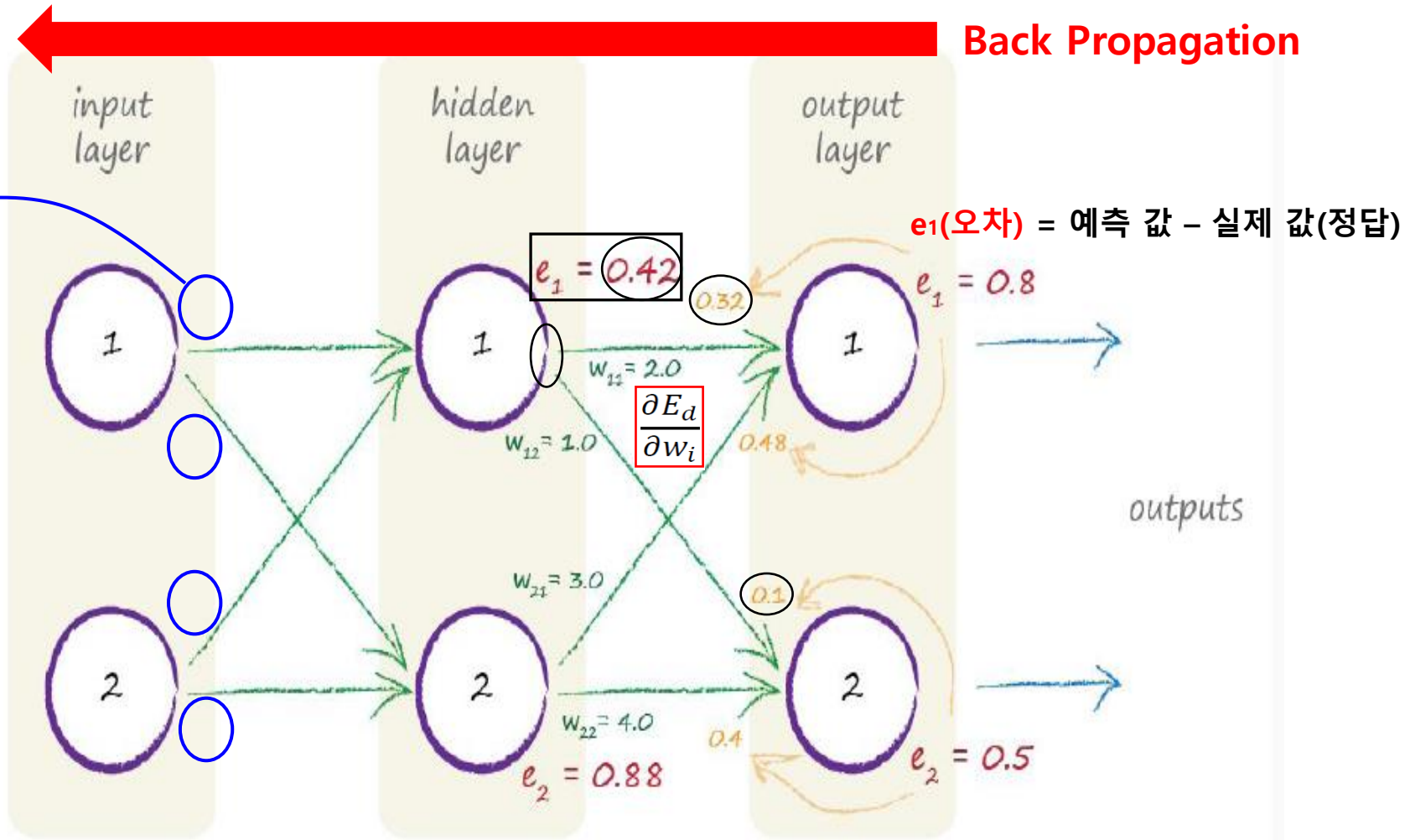
$h(z) = 1/(1+e^{-z})$ = 1 / 1.3499 = 0.7408

출력 2번 Node = ~~~~~~ ====

= 0.6457

Deep Learning Principle

Neural Network의 Learning 방법



Deep Learning Principle

Neural Network의 Learning 방법

Back Propagation

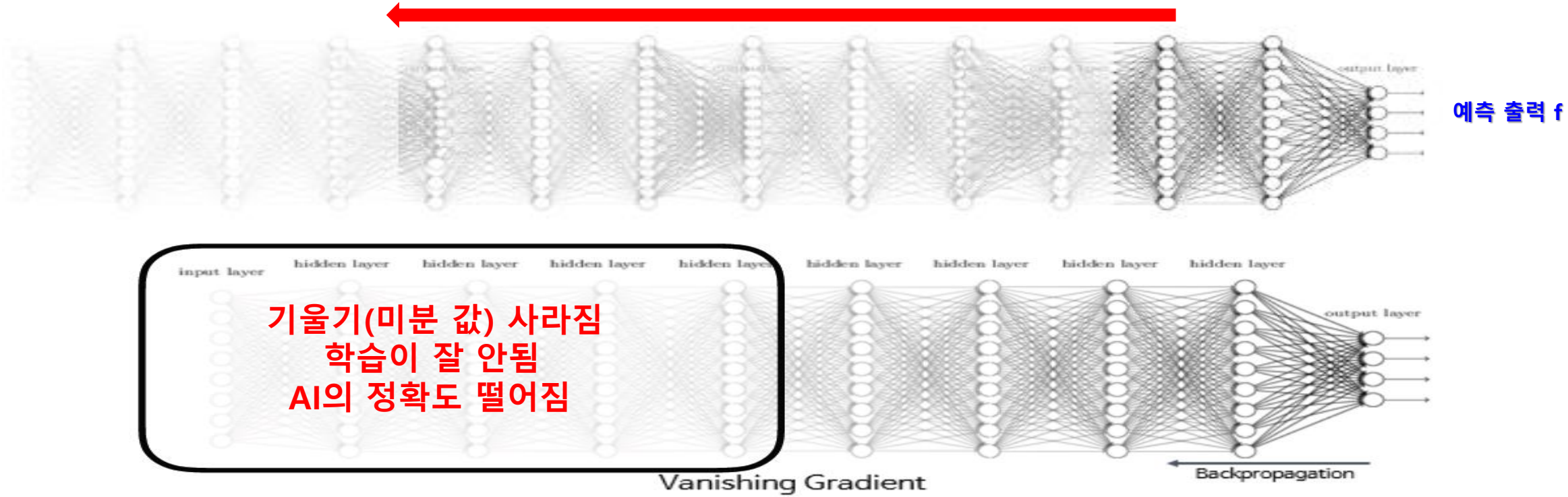
move w_{jk} in the opposite direction to the slope

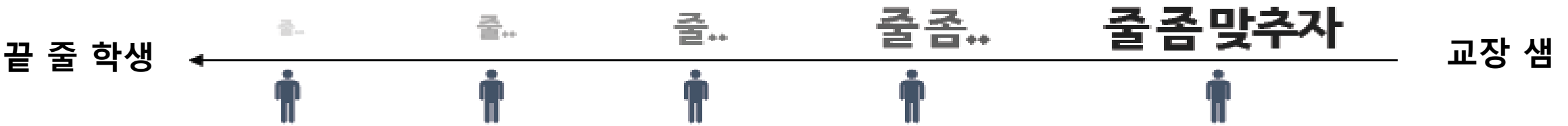
$$\text{new } w_{jk} = \text{old } w_{jk} - \alpha \cdot \frac{\partial E}{\partial w_{jk}}$$

remember that learning rate

Vanishing gradient 현상 (NN winter 2 : 1986 – 2006)

Back Propagation : Layer가 Deep할 수록 **오차 보정을 위해 전달되는 미분(기울기) 값들이 뒤로 갈 수록 없어지는 현상** 발생. 즉, 결론적으로 AI가 **학습이 안됨 (정확도, 성능이 떨어짐)**

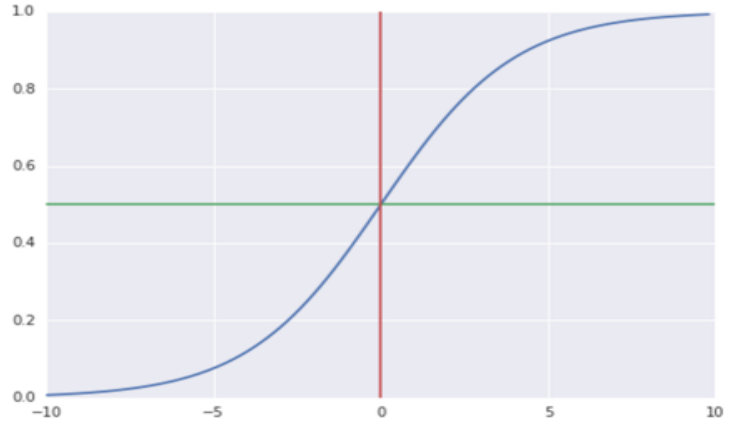




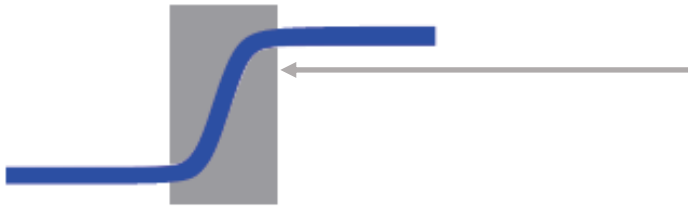
Vanishing gradient 현상 (NN winter 2 : 1986 – 2006)

문제는 ?

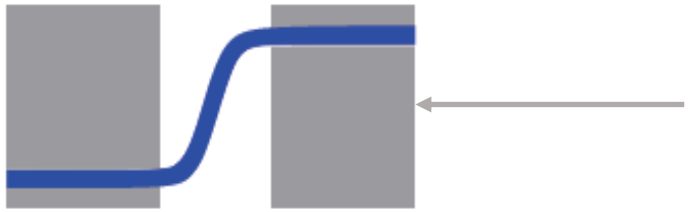
Activation Function(활성화 함수)으로 **Sigmoid 함수** 사용



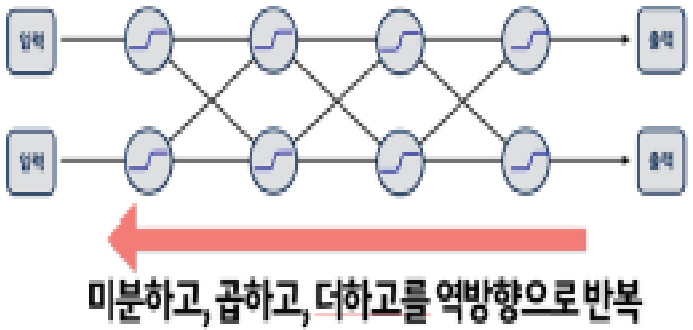
Sigmoid Function 특징



이 부분의 **기울기(미분)**은 존재 하나!,,,



그런데 여기 이 부분의 **기울기(미분)**는 0.00001...(0에 수렴)
이런 값을 중간에 곱해서 **뭔가 뒤로 전달할 값이 없어진다**.
기울기 사라진다 : 0.0001 * 0.00001 = 1e-9 (0.000000001)



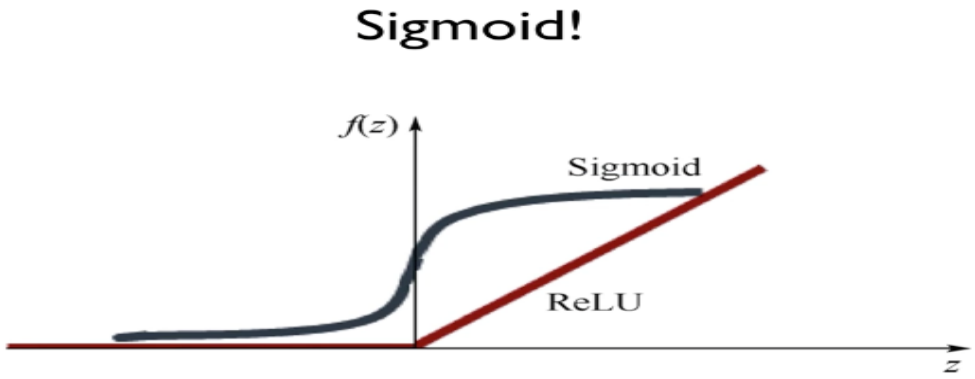
이런 상황에서 **이런 연산 프로세스를 반복하면 ????**

: 0 에 수렴하는 값(아주 작은, 미세한 값)이 맨 끝 단 뉴런에 전달되어 보정(변화)된 w, b값의 **변화가(너무 작아) 오차(에러)**가 있는 **출력 값에 영향(출력 값 변화)을 미치지 못함 (결론적으로 에러 Gap을 최소화 못함)**

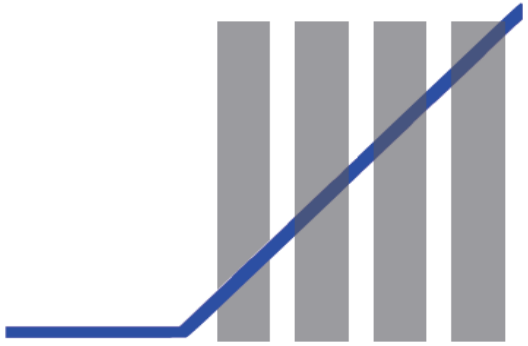
Vanishing gradient 현상 (NN winter 2 : 1986 – 2006)

그래서, **해결 방법은 ????**

기울기가 사라지는 **Sigmoid Function** 대신
이런 현상이 없는 **Activation Function** 개발!!



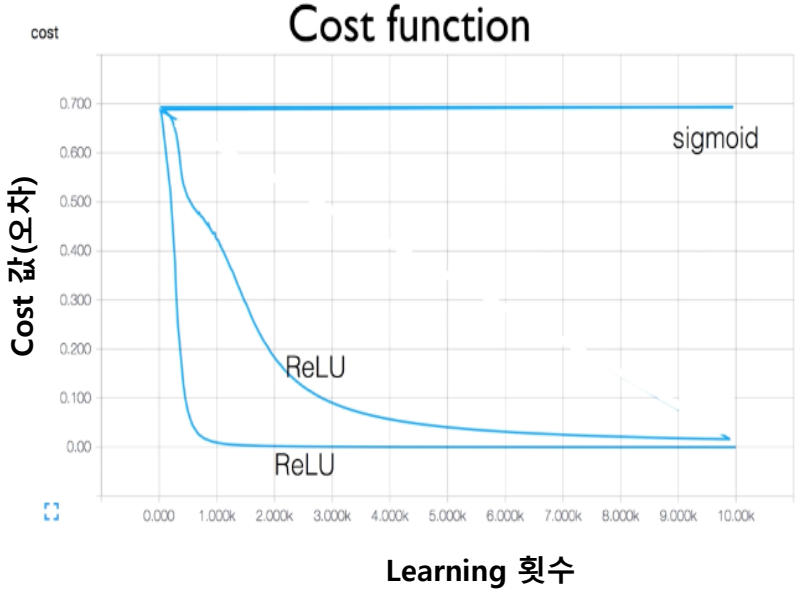
➔ **ReLU** (Rectified Linear Units)



\+ 구간에서는 **입력 값을 그대로 전달**
: + 구간에서는 전부 **미분 값(기울기) 존재**

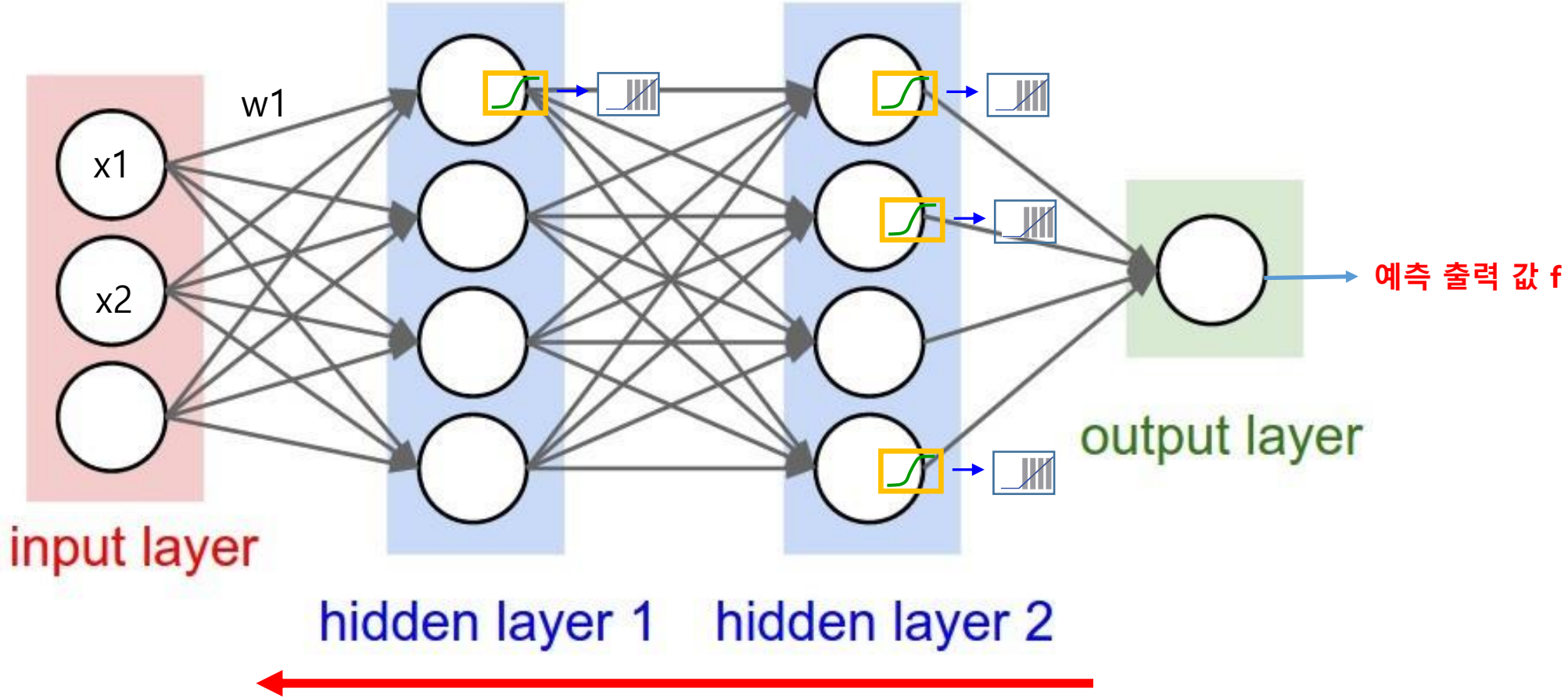
끝 줄 학생까지 이야기가 잘 전달되어, 학생들이 줄도 잘 맞춘다

기울기가 사라지지 않음 학습이 잘 됨(정확도가 높다)



Deep Learning Principle

Vanishing gradient 현상 (NN winter 2 : 1986 – 2006)



Back Propagation Algorithms(Deep Learning 학습 방법)

문제? 뒤 node(Neuron)로 전달하다가, **사라져(기울기) 버린다** (Vanishing Gradient 현상)

해결! Activation Function을 **Sigmoid Function ➔ ReLU Function (뒤로 전달 ok)**

Back Propagation Algorithms

오류역전파(error back-propagation) 학습 알고리듬

Given: 학습 데이터 $D = \{(x^{(d)}, y^{(d)})\}_{n=1}^{N}$, 학습률 η, 다층퍼셉트론 구조 $f(\mathbf{x}, \mathbf{w})$
Find: 다층퍼셉트론 연결가중치 벡터

$$\mathbf{w}_{BP} = argmin_{\mathbf{w}} \left\{ \frac{1}{2} \sum_{d=1}^{N} \sum_{k=1}^{K} \left(f_k(\mathbf{x}^{(d)}, \mathbf{w}) - y_k^{(d)} \right)^2 \right\}$$

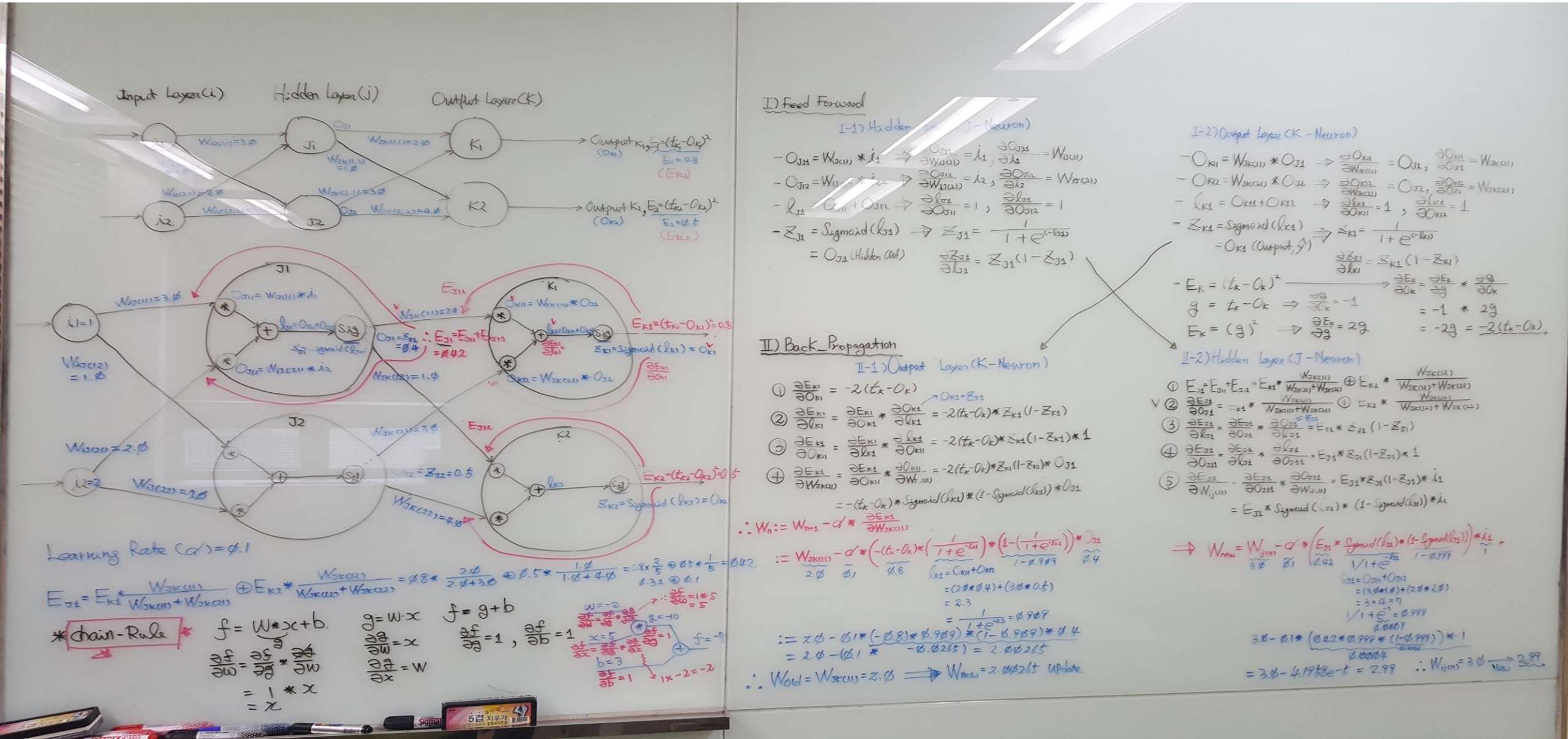
Procedure: 다층퍼셉트론 학습

1. 연결가중치 w_{ji}를 임의의 작은 값으로 초기화한다.
2. 입력 데이터 $\mathbf{x}^{(d)}$ 하나를 입력층에 넣고 활성화값을 전방으로 전달하면서 출력값 $f_k(\mathbf{x}^{(d)}, \mathbf{w}), k = 1, \ldots, k$를 계산한다.
3. 각 출력 뉴런 k에 대한 오차 $\delta_k = (f_k - y_k) f_k (1 - f_k)$를 계산한다 $(f_k = f_k(\mathbf{x}^{(d)}, \mathbf{w}))$.
4. 출력 뉴런 k에 대한 연결 가중치 값의 증분 $\Delta w_{kh} = -\eta \frac{\partial E_d}{\partial w_{kh}} = \eta \delta_k x_h$ 을 계산한다.
5. 은닉 뉴런 h에 대한 오차 $\delta_h = f_h(1 - f_h) \sum_k w_{kh} \delta_k$를 계산한다.
6. 은닉 뉴런 h에 대한 연결 가중치의 증분 $\Delta w_{hi} = -\eta \frac{\partial E_d}{\partial w_{hi}} = \eta \delta_h x_i$를 계산한다.
7. 가중치의 값을 모두 수정한다.

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} \quad \text{for } j \in Upper(i)$$

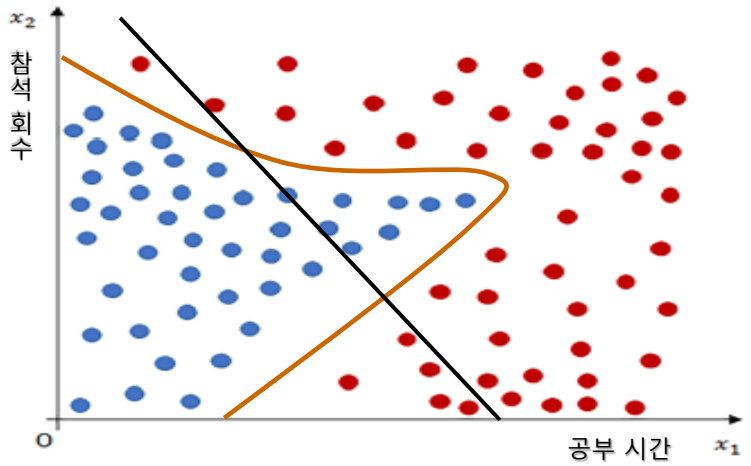
Deep Learning Principle

Back Propagation Algorithms



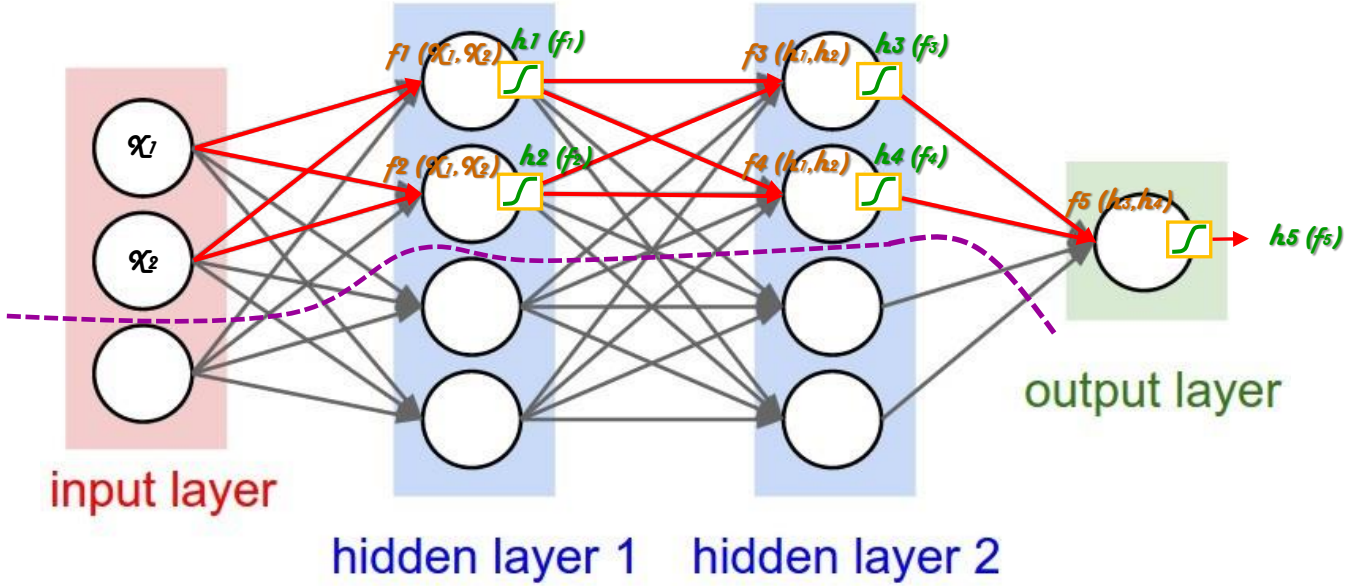
Deep Learning Principle : Sample

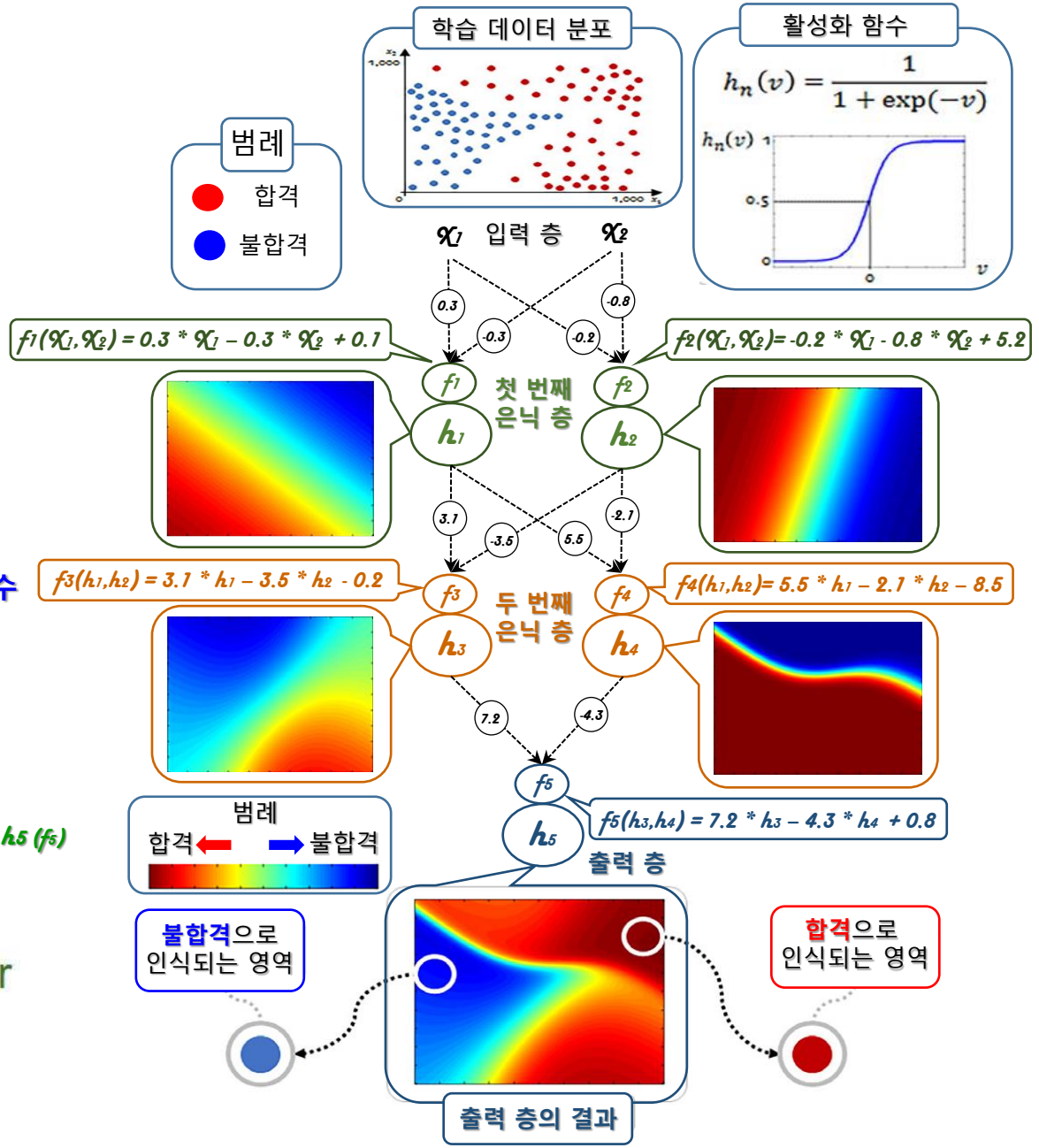
비선형 분포를 보이는 학습 데이터



❖ **문제 풀이 정의 :**

- **비선형 문제** 해결은 Deep Learning **(Multiple Neuron으로)**
- **Binary Classification 문제** 해결은 Activation Function으로 비선형의 **Sigmoid 함수**

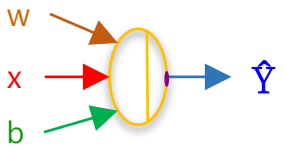




Multiple Neural Nets(NN) for XOR Problem

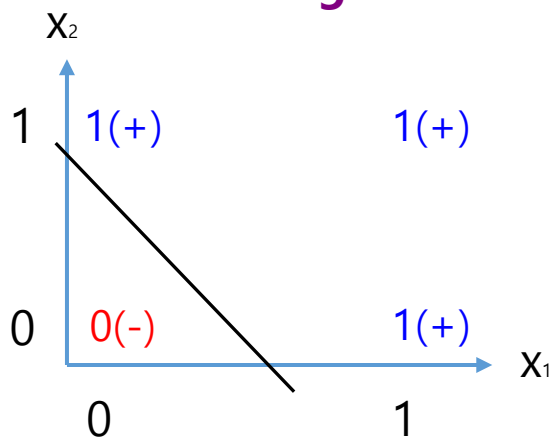
XOR Problem : Multiple Neuron

Single Neuron : 해결



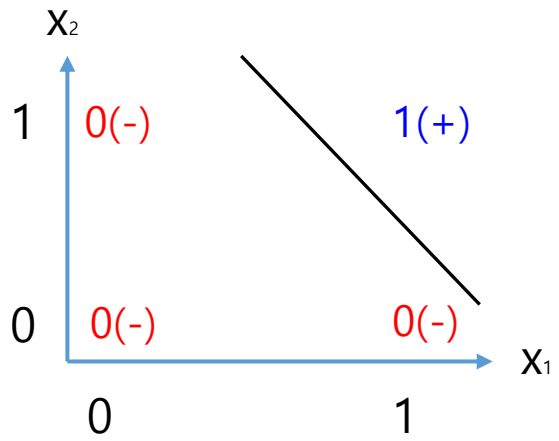
해결 못함

OR Logic



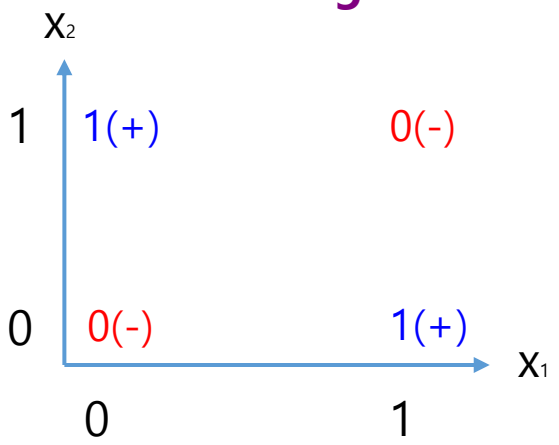
X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

AND Logic



X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

XOR Logic



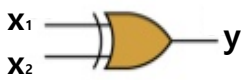
X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

XOR Problem : Multiple Neuron

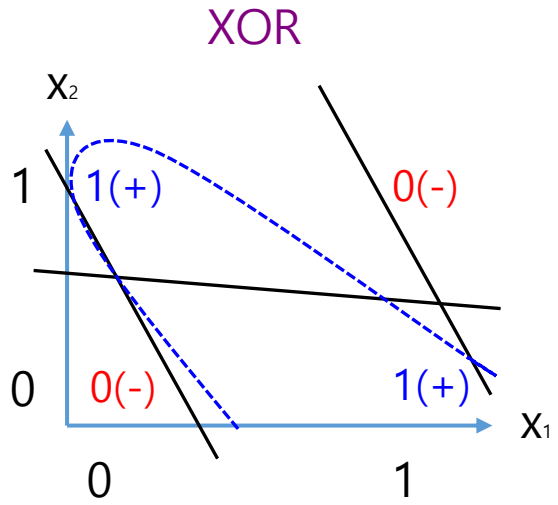
XOR using NN

Boolean Expression	Logic Diagram Symbol	Truth Table

$y = x_1 \oplus x_2$



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



❖ **문제 풀이 정의 :**

- **XOR 문제**는 **하나의 Neuron으로는 풀 수 없다**.

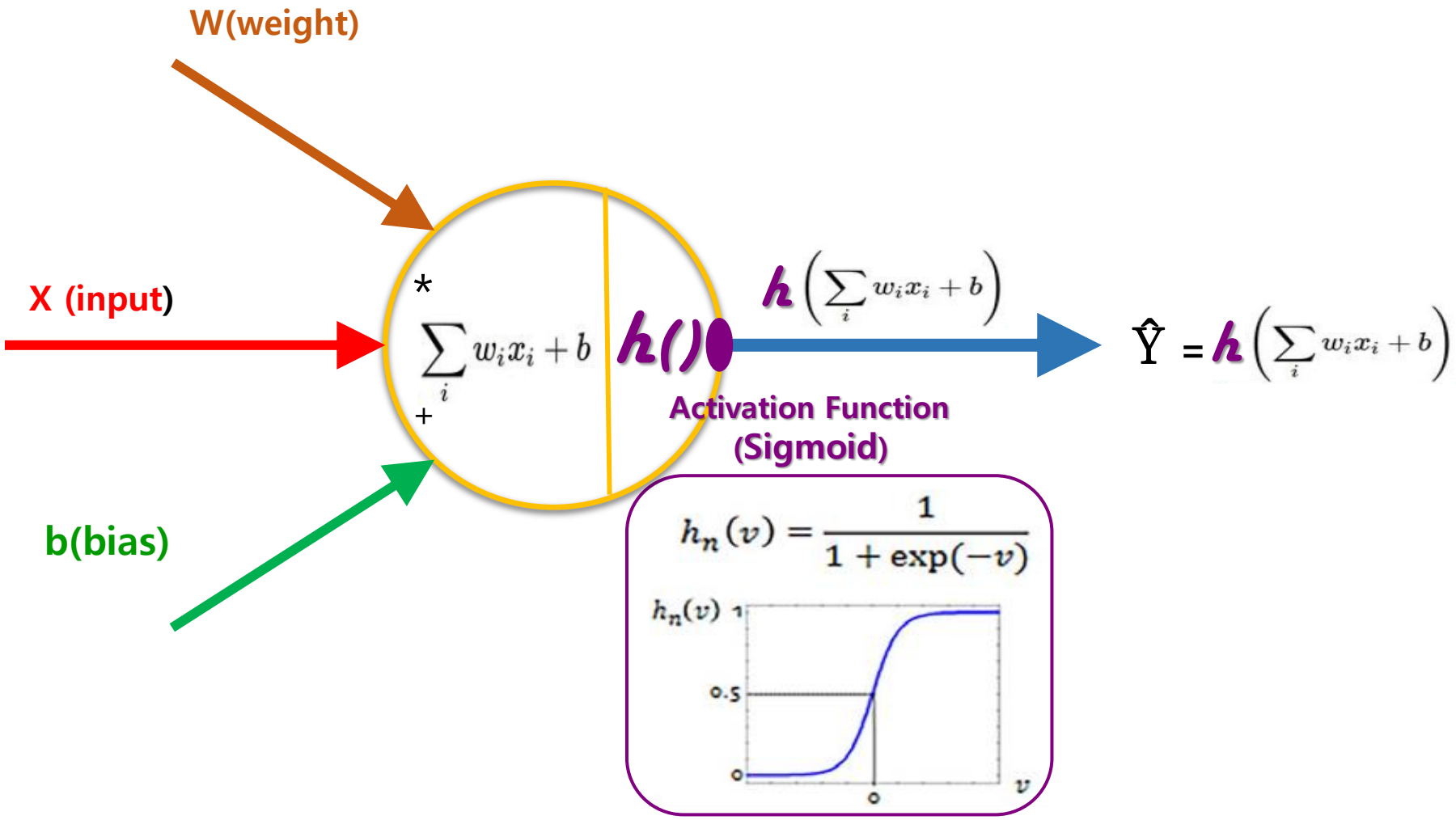
  ➔ **선형적**으로 상기 그래프의 **답에 대한 구분 불가**

  ➔ **비선형 문제** 해결은 Deep Learning **(Multiple Neuron 으로 해결)**

- **Binary Classification 문제(0 or 1) 해결**은 Activation Function으로 비선형의 **Sigmoid 함수**

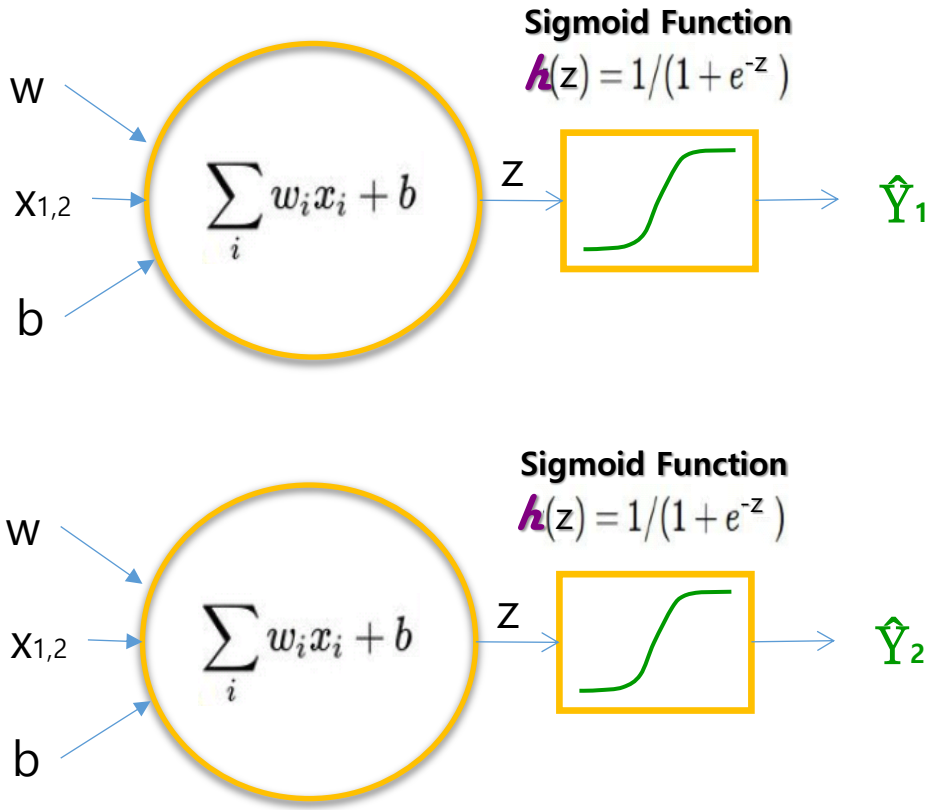
XOR Problem : Multiple Neuron

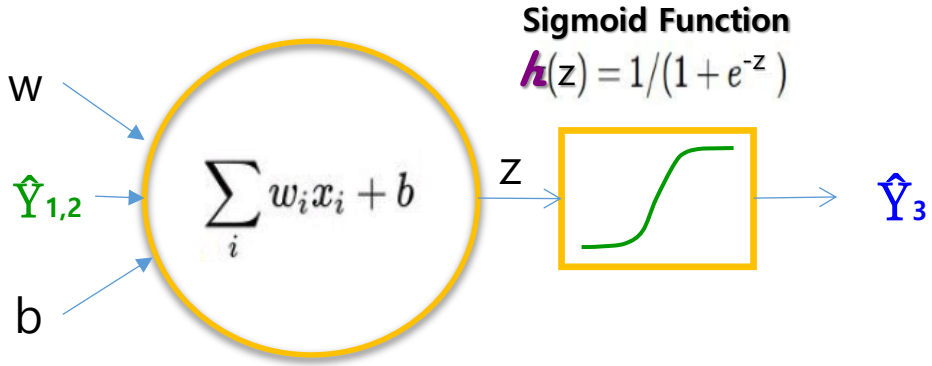
One logistics regression unit (one neuron) cannot separate XOR



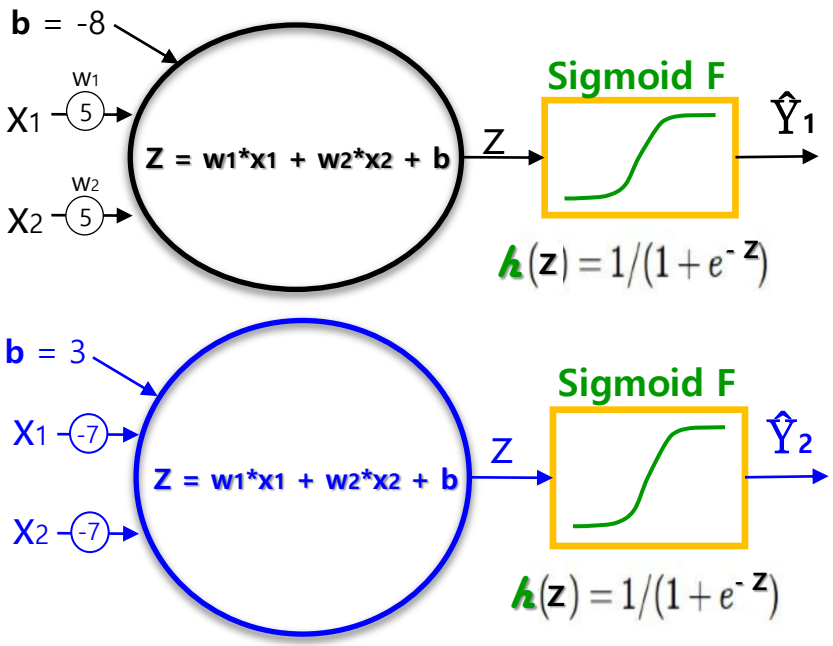
XOR Problem : Multiple Neuron

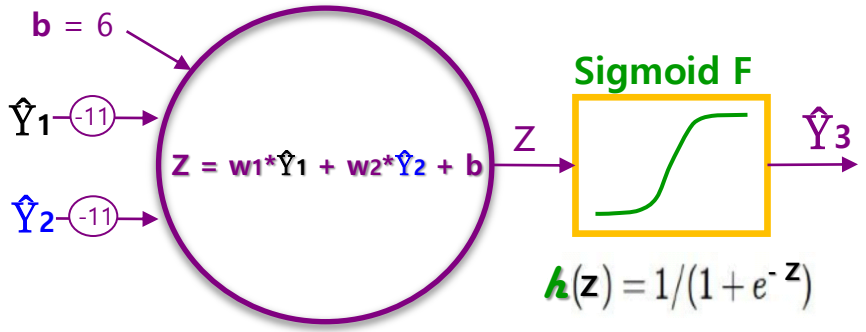
Multiple logistic regression unit (Multi neuron)





XOR Problem : Multiple Neuron

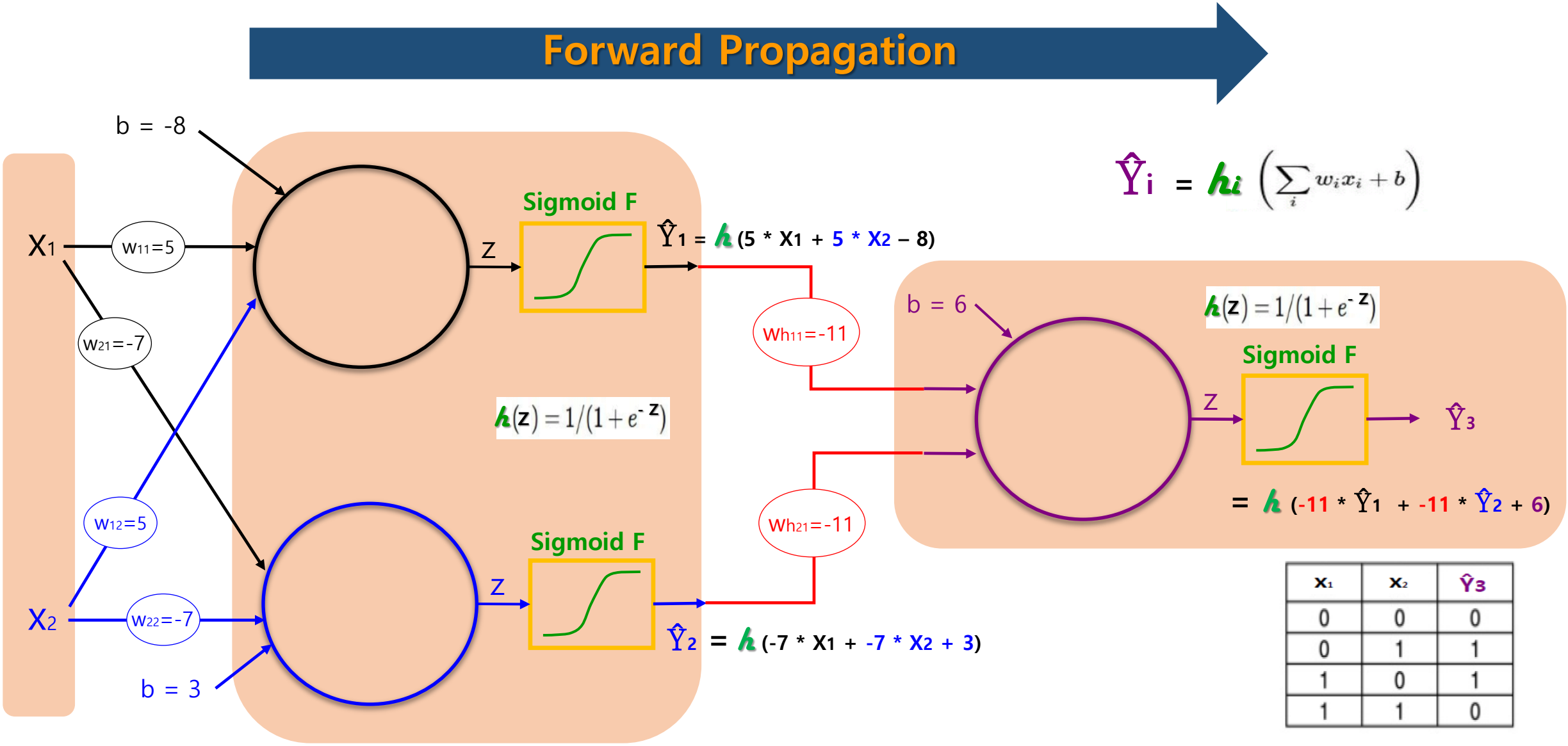




x_1 x_2	w	b	z
[0 , 0] •	$\begin{bmatrix} 5 \\ 5 \end{bmatrix}$	- 8	= - 8 , $\hat{Y}_1$ = Sigmoid (8) = 0.000335.. ➔ 0
[0 , 0] •	$\begin{bmatrix} -7 \\ -7 \end{bmatrix}$	+ 3	= 3 , $\hat{Y}_2$ = Sigmoid (-3) = 0.95 ➔ 1
$\hat{Y}_1$ $\hat{Y}_2$			
[0 , 1] •	$\begin{bmatrix} -11 \\ -11 \end{bmatrix}$	+ 6	= -5 , $\hat{Y}_3$ = Sigmoid (5) = 0.00669.. ➔ 0

X_1	X_2	$\hat{Y}_1$	$\hat{Y}_2$	$\hat{Y}_3$	XOR
0	**0**	**0**	**1**	**0** =	**0**
0	1	0	0	1 =	1
1	0	0	0	1 =	1
1	1	1	0	0 =	0

XOR Problem : Multiple Neuron



$$\hat{Y}_i = h_i\left(\sum_i w_i x_i + b\right)$$

x_1	x_2	$\hat{Y}_3$
0	0	0
0	1	1
1	0	1
1	1	0

XOR Problem : Single Neuron Programming

```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)
W = tf.Variable(tf.random_normal([2, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))

# Launch graph
with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())

   for step in range(10001):
       sess.run(train, feed_dict={X: x_data, Y: y_data})
       if step % 100 == 0:
           print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}), sess.run(W))

   # Accuracy report
   h, c, a = sess.run([hypothesis, predicted, accuracy], feed_dict={X: x_data, Y: y_data})
   print("\nHypothesis: ", h, "\nCorrect: ", c, "\nAccuracy: ", a)
```

XOR with
logistic regression?

But
it doesn't work!

```
Hypothesis:
[[ 0.5]
 [ 0.5]
 [ 0.5]
 [ 0.5]]
Correct:
[[ 0.]
 [ 0.]
 [ 0.]
 [ 0.]]
Accuracy:  0.5
```

XOR Problem : Multiple Neuron Programming

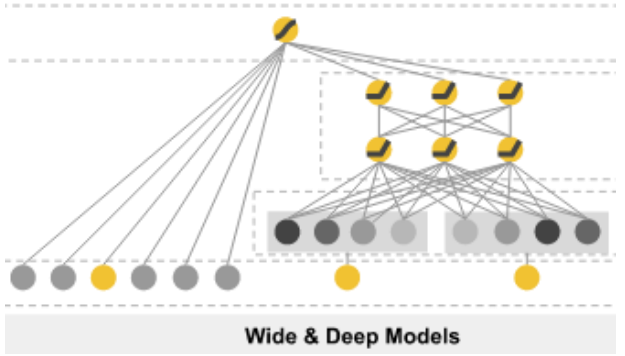
```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
b1 = tf.Variable(tf.random_normal([2]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        sess.run(train, feed_dict={X: x_data, Y: y_data})
        if step % 100 == 0:
            print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}), sess.run([W1, W2]))

    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy],
                       feed_dict={X: x_data, Y: y_data})
    print("\nHypothesis: ", h, "\nCorrect: ", c, "\nAccuracy: ", a)
```

NN(Multiple Neuron) for XOR

```
Hypothesis:
[[ 0.01338218]
 [ 0.98166394]
 [ 0.98809403]
 [ 0.01135799]]
Correct:
[[ 0.]
 [ 1.]
 [ 1.]
 [ 0.]]
Accuracy:  1.0
```

XOR Problem : Multiple Neuron Programming

Deep NN for XOR




```
W1 = tf.Variable(tf.random_normal([2, 10]), name='weight1')
b1 = tf.Variable(tf.random_normal([10]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([10, 10]), name='weight2')
b2 = tf.Variable(tf.random_normal([10]), name='bias2')
layer2 = tf.sigmoid(tf.matmul(layer1, W2) + b2)

W3 = tf.Variable(tf.random_normal([10, 10]), name='weight3')
b3 = tf.Variable(tf.random_normal([10]), name='bias3')
layer3 = tf.sigmoid(tf.matmul(layer2, W3) + b3)

W4 = tf.Variable(tf.random_normal([10, 1]), name='weight4')
b4 = tf.Variable(tf.random_normal([1]), name='bias4')
hypothesis = tf.sigmoid(tf.matmul(layer3, W4) + b4)
```

4 layers

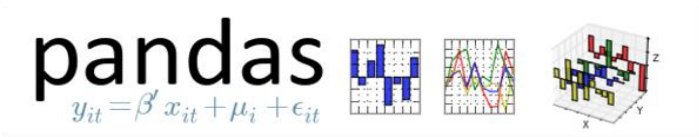
Hypothesis:
[[7.80e-04]
 [9.99e-01]
 [9.98e-01]
 [1.55e-03]]
Correct:
[[0.]
 [1.]
 [1.]
 [0.]]
Accuracy: 1.0

2 layers

Hypothesis:
[[0.01338218]
 [0.98166394]
 [0.98809403]
 [0.01135799]]
Correct:
[[0.]
 [1.]
 [1.]
 [0.]]
Accuracy: 1.0

어떤 것을 사용 할 것인가?

“Don’t be a hero”



결론

The Coming Age of Smart Machines

미래는 이미 와 있었다 !!

THE FUTURE
IS ALREADY HERE

It's just not evenly
distributed yet.

William Gibson

" 미래는 이미 와 있다.
단지 공평하게 퍼져있지
않을 뿐이다 "

" **AI로 바꿔가는 미래** " 는
머지않아 대부분의 고객은 기업과의
상호작용 시 인공지능에 기반한 것과
그렇지 못한 것의 차이를 구별하게
될 것이다