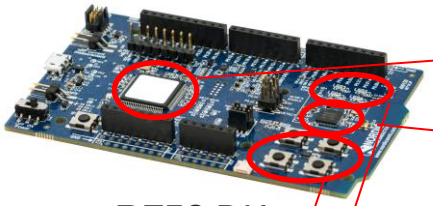
3일차

IoT 임베디드 시스템

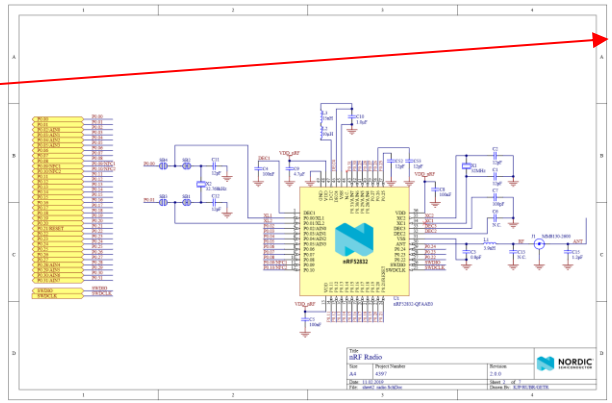
MCU · 임베디드 시스템 · 실습

PCB fabrication

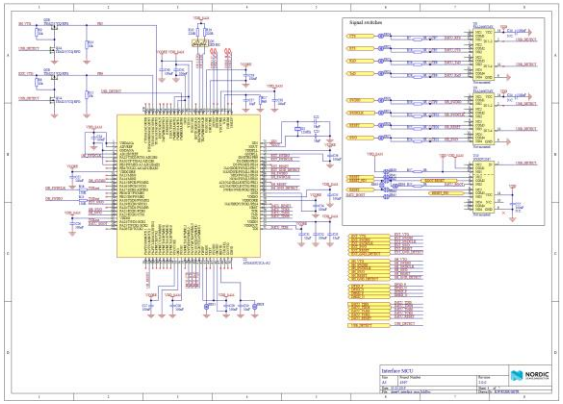
Nordic 사 제공 공식 하드웨어 파일



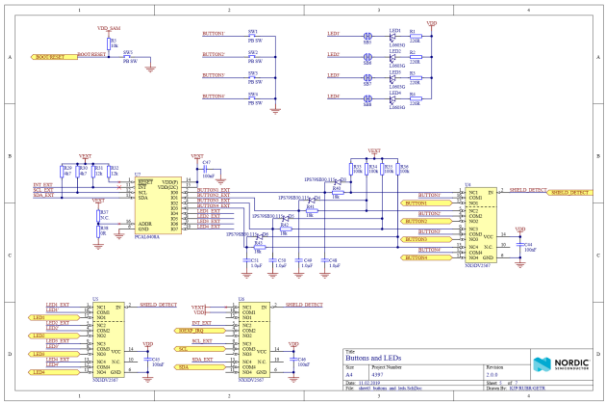
nRF52 DK



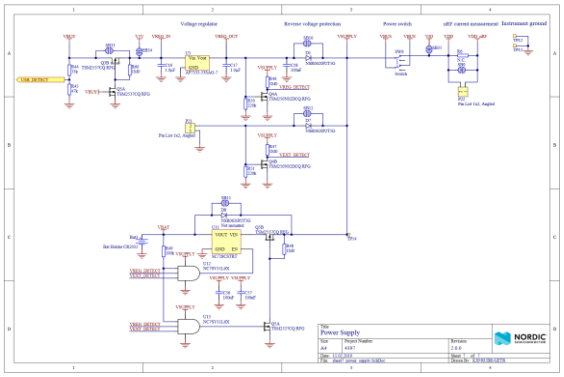
nRF52832



Interface MCU



Button and LED

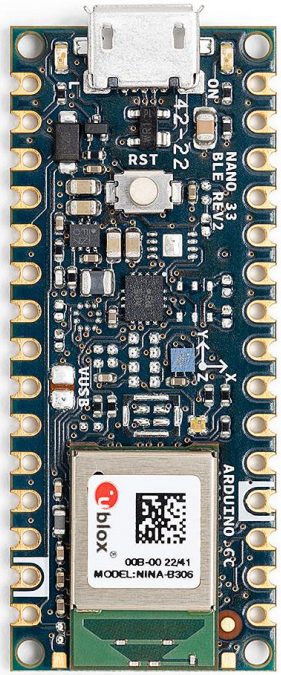


Power supply

다양한 목적의 사용을 위해 크기가 크고 여러 부가기능이 포함되어 있음

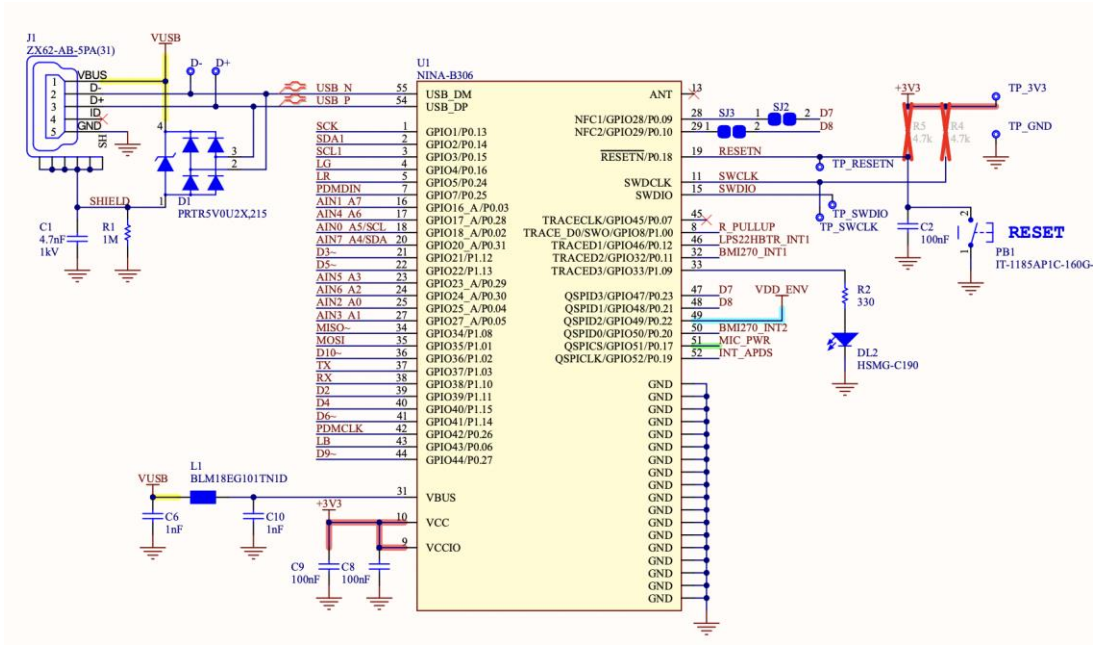
→ 최적화된 설계를 위해 소형화의 필요

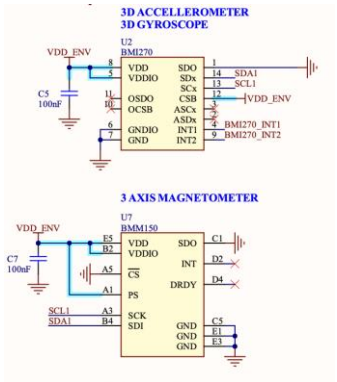
PCB fabrication

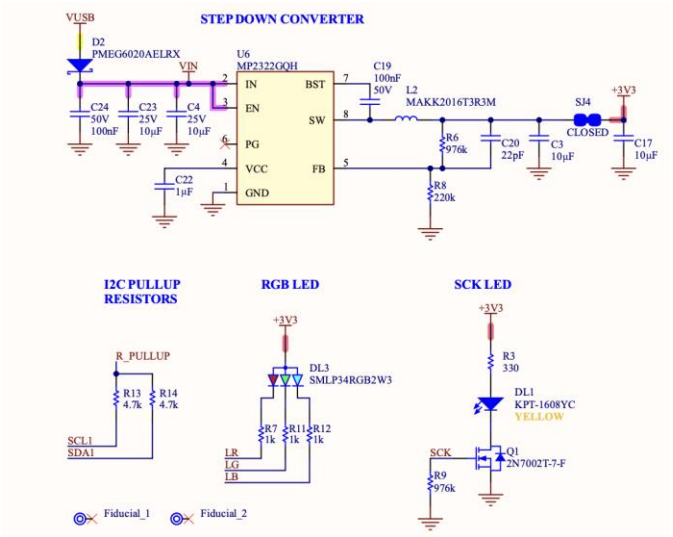


Nano 33 BLE

nRF52840 칩셋 사용







PCB fabrication

nRF52832 구동을 위한 필수 주변 회로

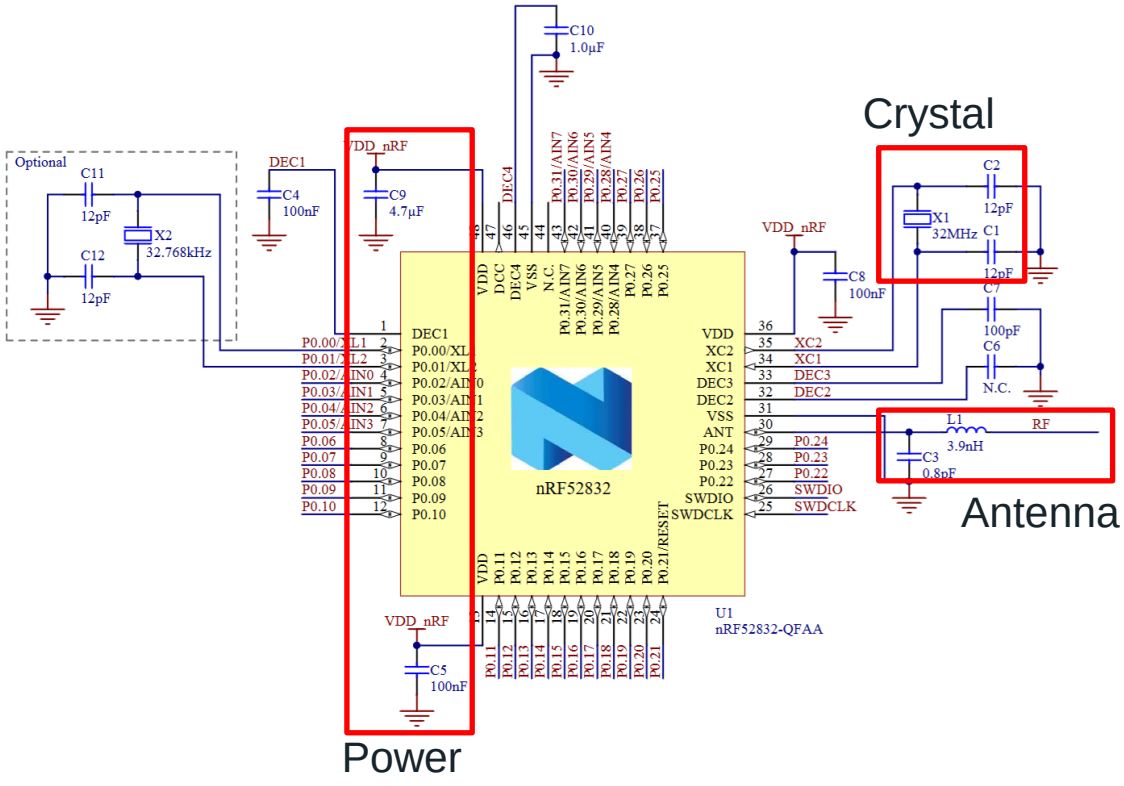


Figure 2: QFN48 pin assignments, top view

Table 3: QFN48 pin assignments

Pin	Name	Type	Description
Left Side of chip			
1	DEC1	Power	0.9 V regulator digital supply decoupling
2	P0.00	Digital I/O	General purpose I/O
	XL1	Analog input	Connection for 32.768 kHz crystal (LFXO)
3	P0.01	Digital I/O	General purpose I/O
	XL2	Analog input	Connection for 32.768 kHz crystal (LFXO)
4	P0.02	Digital I/O	General purpose I/O
	AIN0	Analog input	SAADC/COMP/LPCOMP input
5	P0.03	Digital I/O	General purpose I/O
	AIN1	Analog input	SAADC/COMP/LPCOMP input
6	P0.04	Digital I/O	General purpose I/O
	AIN2	Analog input	SAADC/COMP/LPCOMP input
7	P0.05	Digital I/O	General purpose I/O
	AIN3	Analog input	SAADC/COMP/LPCOMP input
8	P0.06	Digital I/O	General purpose I/O
9	P0.07	Digital I/O	General purpose I/O
Pin	**Name**	**Type**	**Description**
10	P0.08	Digital I/O	General purpose I/O
11	NFC1	NFC input	NFC antenna connection
	P0.09	Digital I/O	General purpose I/O[2]
12	NFC2	NFC input	NFC antenna connection
	P0.10	Digital I/O	General purpose I/O[2]
Bottom side of chip			
13	VDD	Power	Power supply
14	P0.11	Digital I/O	General purpose I/O
15	P0.12	Digital I/O	General purpose I/O
16	P0.13	Digital I/O	General purpose I/O
17	P0.14	Digital I/O	General purpose I/O
	TRACEDATA[3]		Trace port output
18	P0.15	Digital I/O	General purpose I/O
	TRACEDATA[2]		Trace port output
19	P0.16	Digital I/O	General purpose I/O
	TRACEDATA[1]		Trace port output
20	P0.17	Digital I/O	General purpose I/O
21	P0.18	Digital I/O	General purpose I/O
	TRACEDATA[0] / SWO		Single wire output Trace port output
22	P0.19	Digital I/O	General purpose I/O
23	P0.20	Digital I/O	General purpose I/O
	TRACECLK		Trace port clock output
24	P0.21	Digital I/O	General purpose I/O
	nRESET		Configurable as pin reset
Right Side of chip			
25	SWDCLK	Digital input	Serial wire debug clock input for debug and programming
26	SWDIO	Digital I/O	Serial wire debug I/O for debug and programming
27	P0.22	Digital I/O	General purpose I/O[2]
28	P0.23	Digital I/O	General purpose I/O[2]
29	P0.24	Digital I/O	General purpose I/O[2]
30	ANT	RF	Single-ended radio antenna connection
31	VSS	Power	Ground (Radio supply)
32	DEC2	Power	1.3 V regulator supply decoupling (Radio supply)
33	DEC3	Power	Power supply decoupling
34	XC1	Analog input	Connection for 32 MHz crystal
35	XC2	Analog input	Connection for 32 MHz crystal
36	VDD	Power	Power supply
Top side of chip			
37	P0.25	Digital I/O	General purpose I/O[2]
38	P0.26	Digital I/O	General purpose I/O[2]
39	P0.27	Digital I/O	General purpose I/O[2]
40	P0.28	Digital I/O	General purpose I/O[2]
	AIN4	Analog input	SAADC/COMP/LPCOMP input
41	P0.29	Digital I/O	General purpose I/O[2]
	AIN5	Analog input	SAADC/COMP/LPCOMP input
42	P0.30	Digital I/O	General purpose I/O[2]
	AIN6	Analog input	SAADC/COMP/LPCOMP input
43	P0.31	Digital I/O	General purpose I/O pin[3]
	AIN7	Analog input	SAADC/COMP/LPCOMP input
Pin	**Name**	**Type**	**Description**
44	NC		No connect Leave unconnected
45	VSS	Power	Ground
46	DEC4	Power	1.3 V regulator supply decoupling Input from DC/DC regulator Output from 1.3 V LDO
47	DCC	Power	DC/DC regulator output
48	VDD	Power	Power supply
Bottom of chip			
Die pad	VSS	Power	Ground pad Exposed die pad must be connected to ground (VSS) for proper device operation.

동작을 Power와 clock만 있으면 최소한의 동작이 가능함

(BLE 사용 시 Antenna 또한 필수)

추가적인 기능은 오른쪽 pin 역할을 고려하여 자유롭게 배치

PCB fabrication

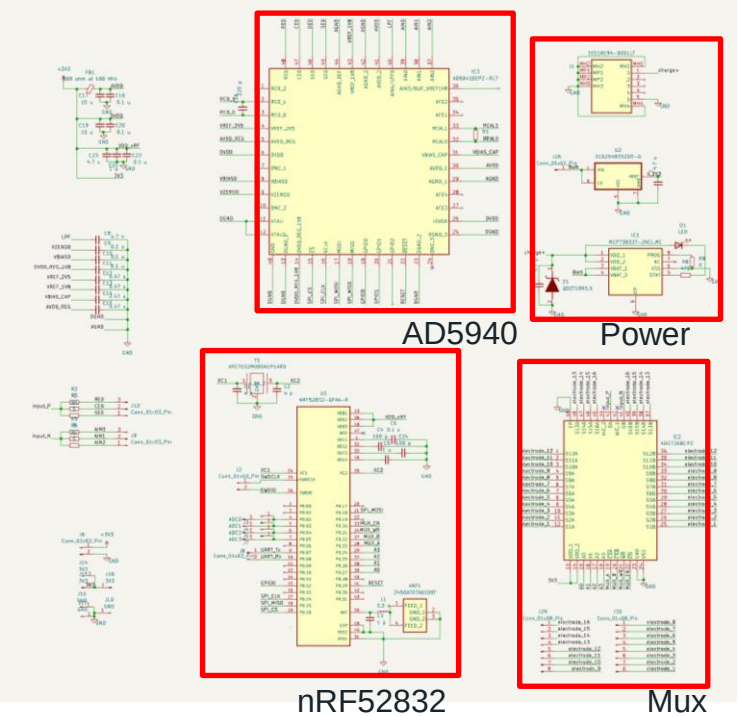
- **PCB 개발 tools**

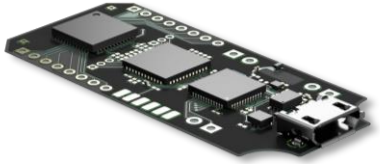
개발 용도와 수준에 맞는 개발 tools 사용

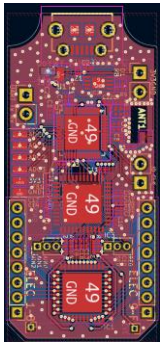
Kicad는 무료로 제공되어 간단한 회로를 만들기 용이함

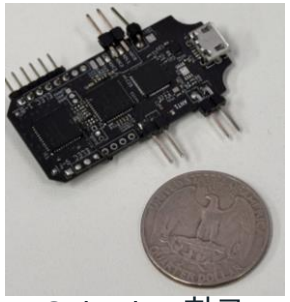
PCB fabrication

- **nRF52 회로 설계 예시**(다중 생체신호 계측 회로)

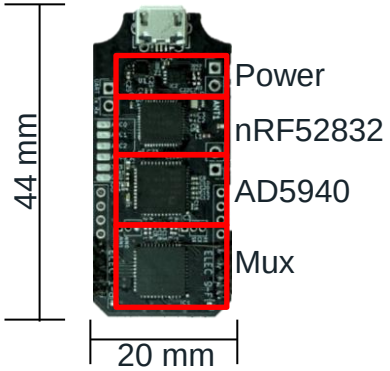






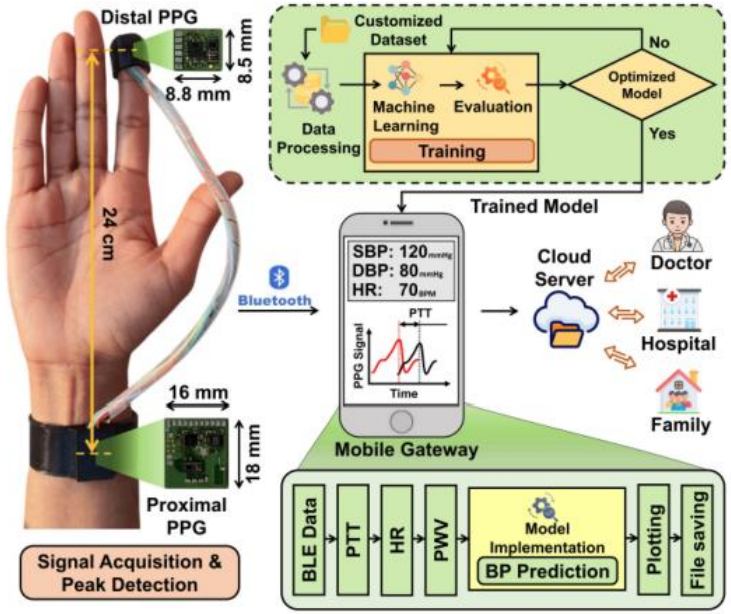


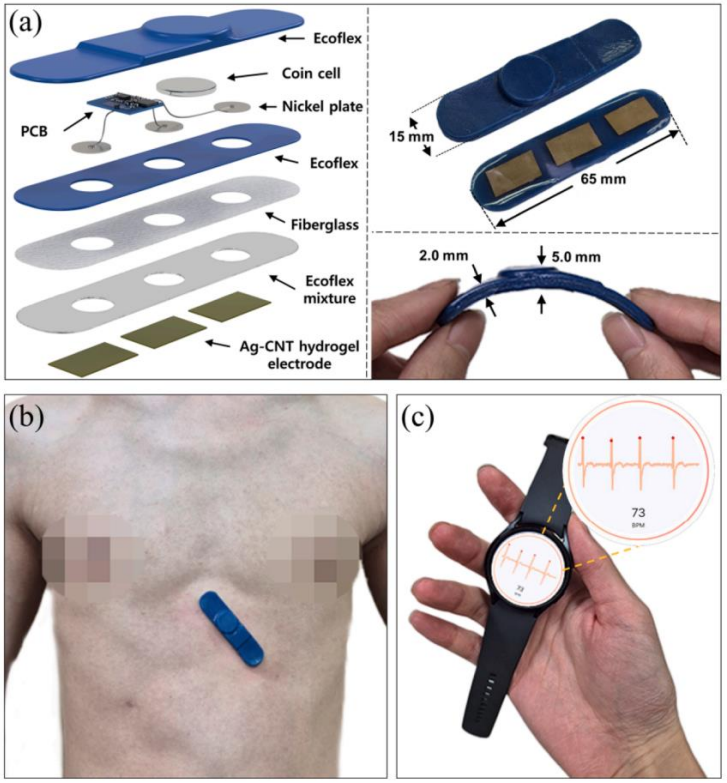
Coin size 회로

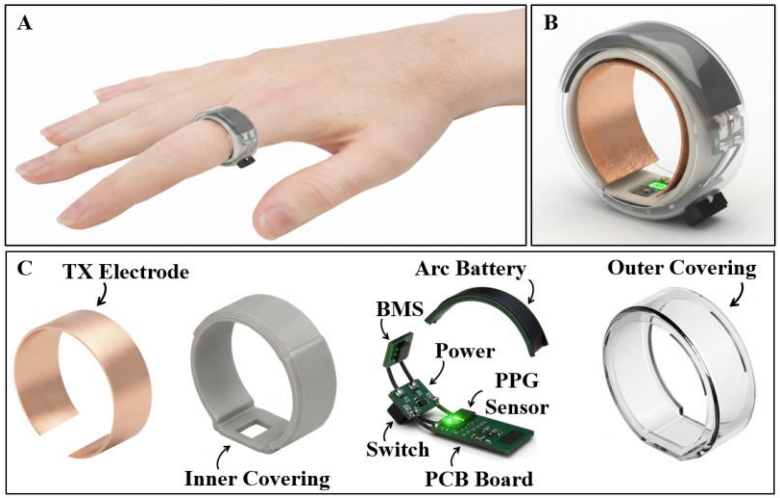


회로도 작성 → 회로 설계 → 제작
schematic layout

PCB fabrication

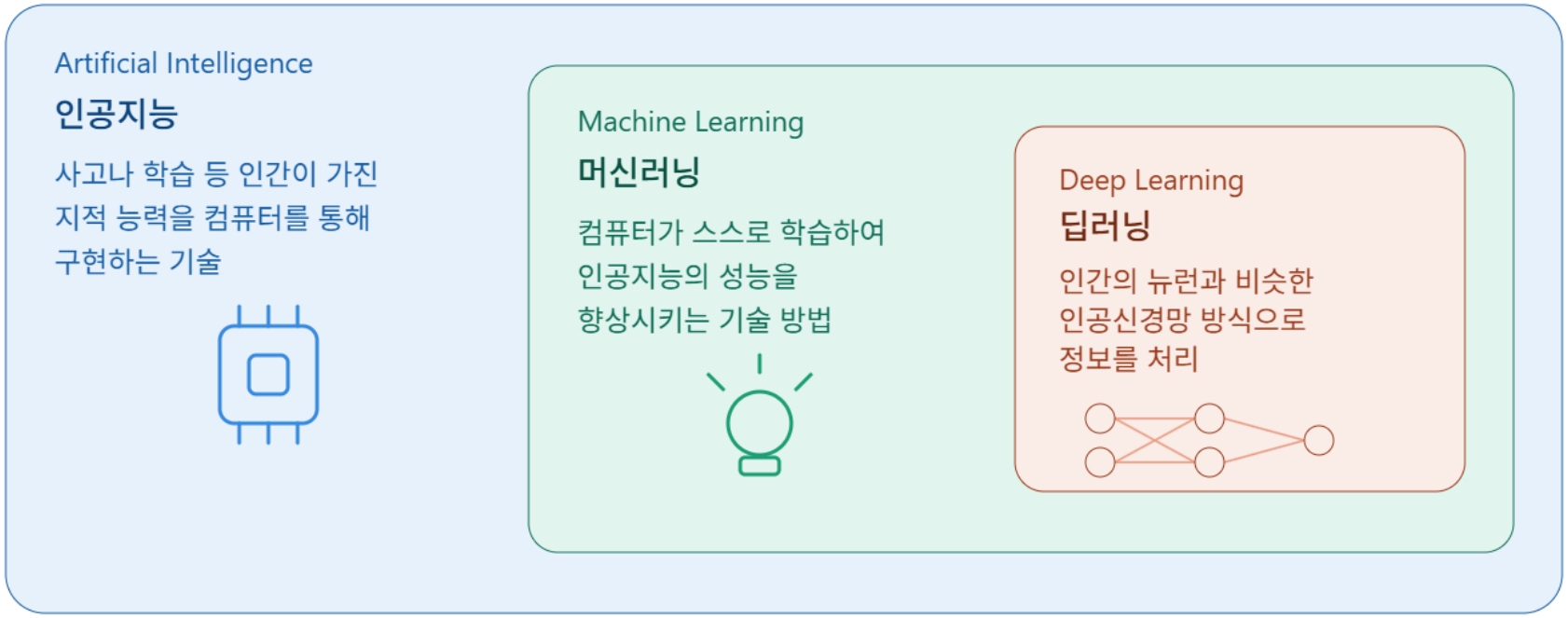




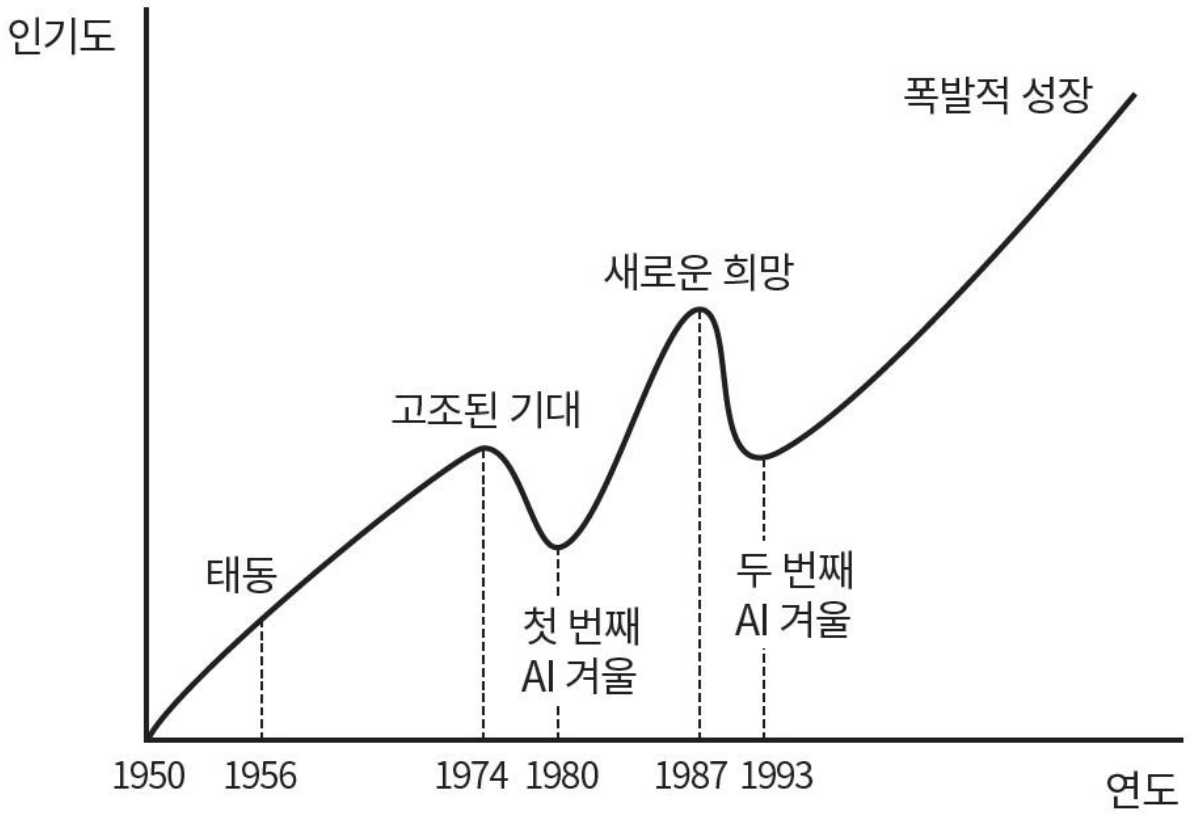


Artificial Intelligence

Artificial intelligence (AI): 컴퓨터로 만들어진 지능으로 “사람에 의해 통상적으로 수행되는 지적인 작업을 자동화 하는 것을 목표”(Chollet, 2018) 로 하는 학문 영역

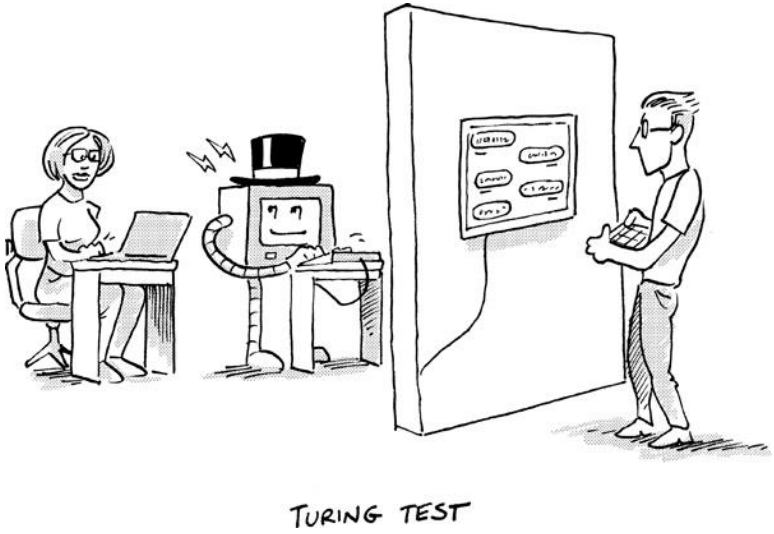


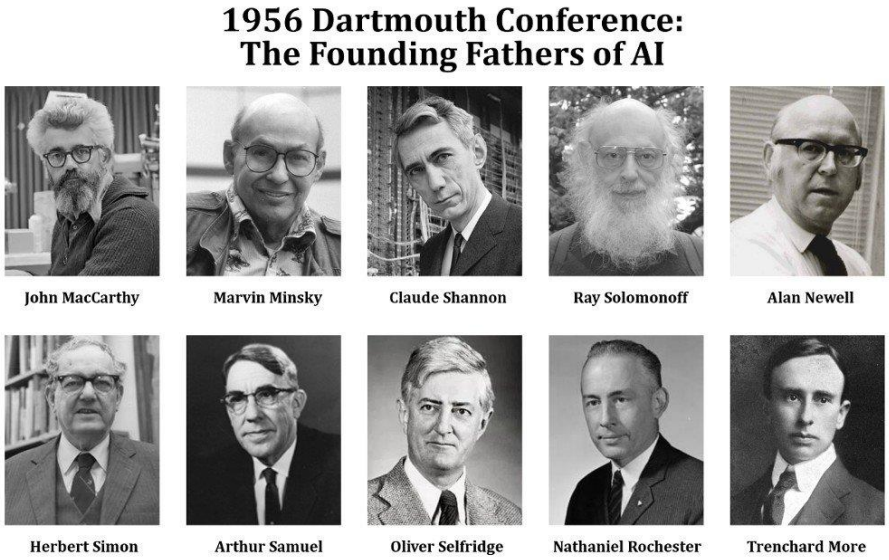
Artificial Intelligence: History



Artificial Intelligence: History

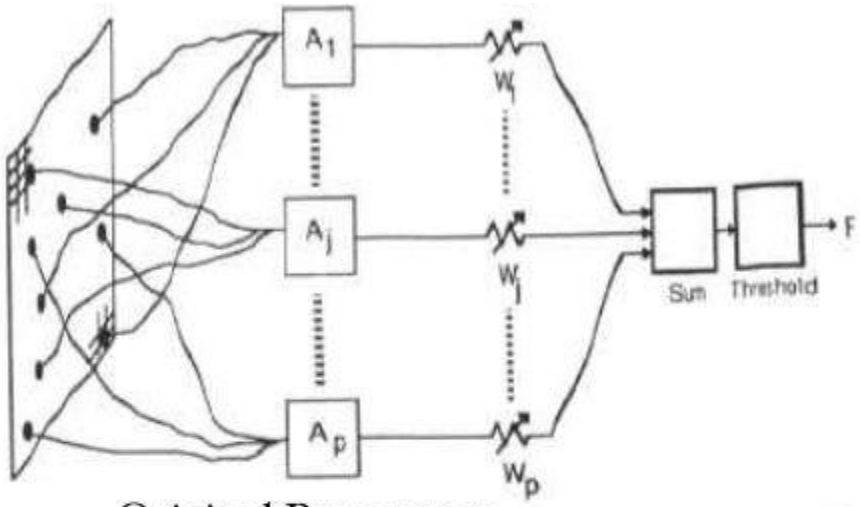
튜링 테스트(Turing test)는 기계가 인간과 얼마나 비슷하게 대화하고 생각할 수 있는지를 판별하기 위해 수학자 앨런 튜링이 1950년에 제안한 인공지능 평가 시험





Artificial Intelligence: History

Perceptron (1958, Frank Rosenblatt): First model of neural network

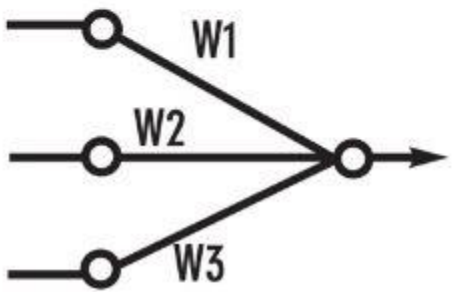


Original Perceptron

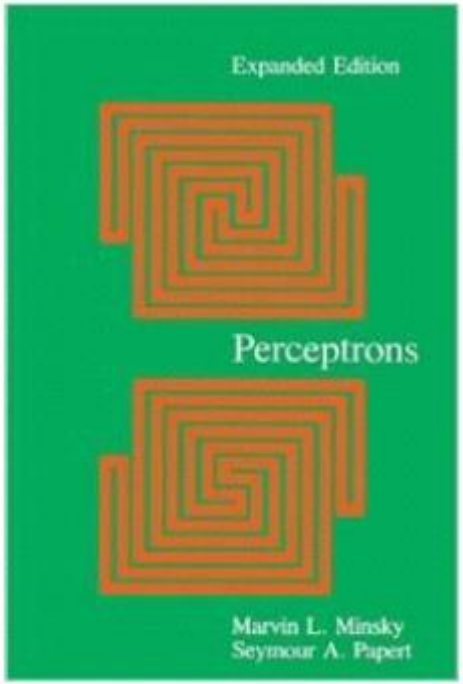
(From Perceptrons by M. L Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press.

Frank Rosenblatt
(1928-1971)

Simplified model:



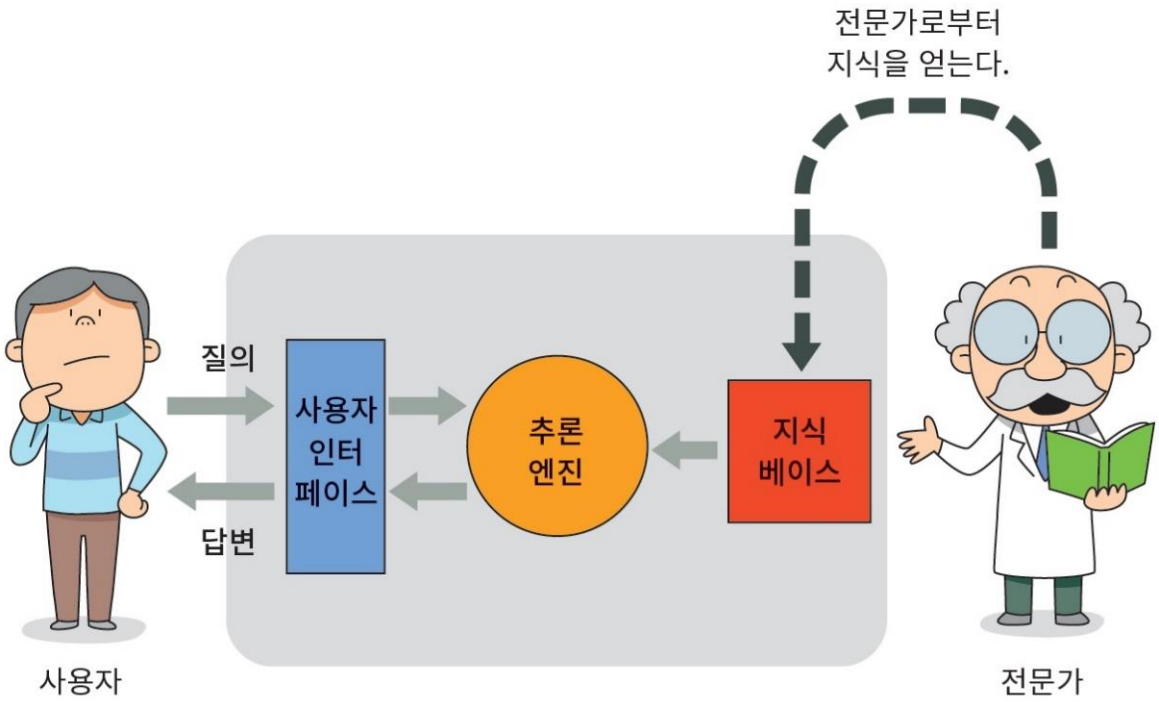
Artificial Intelligence: History



- **첫 번째 AI 겨울 (1974~1980)**
  - No one on earth had found a viable way to train MLPs(Multi-Layer Perceptrons) good enough to learn such simple functions
  - Perceptron의 XOR 계산 불가능을 이론적으로 증명
    - 연산에 필요한 CPU의 속도나 메모리의 부족
    - 최적 솔루션을 구하기 위해 필요한 계산 시간
    - 부족한 정보량, 학습 방법의 부재

Artificial Intelligence: History

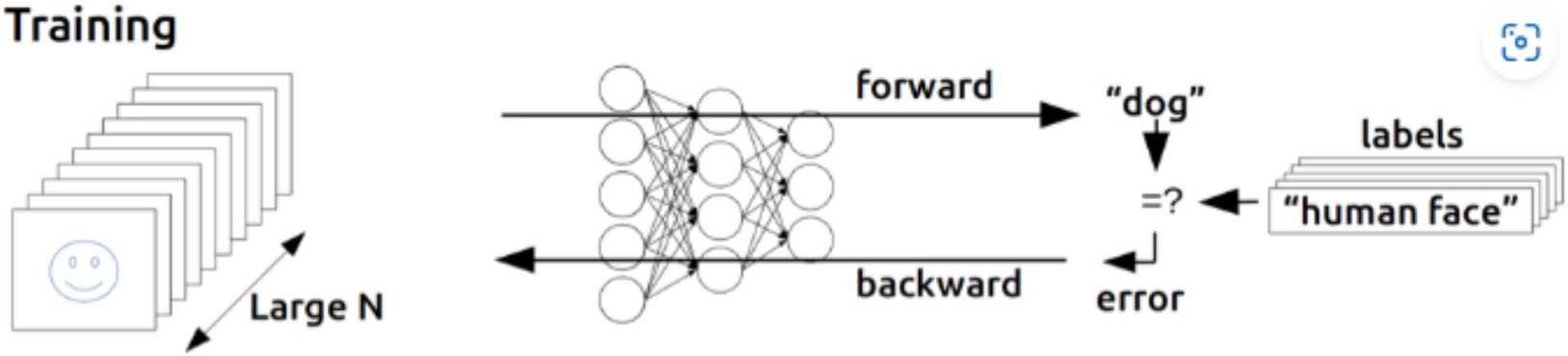
- 새로운 희망: 전문가 시스템



Artificial Intelligence: History

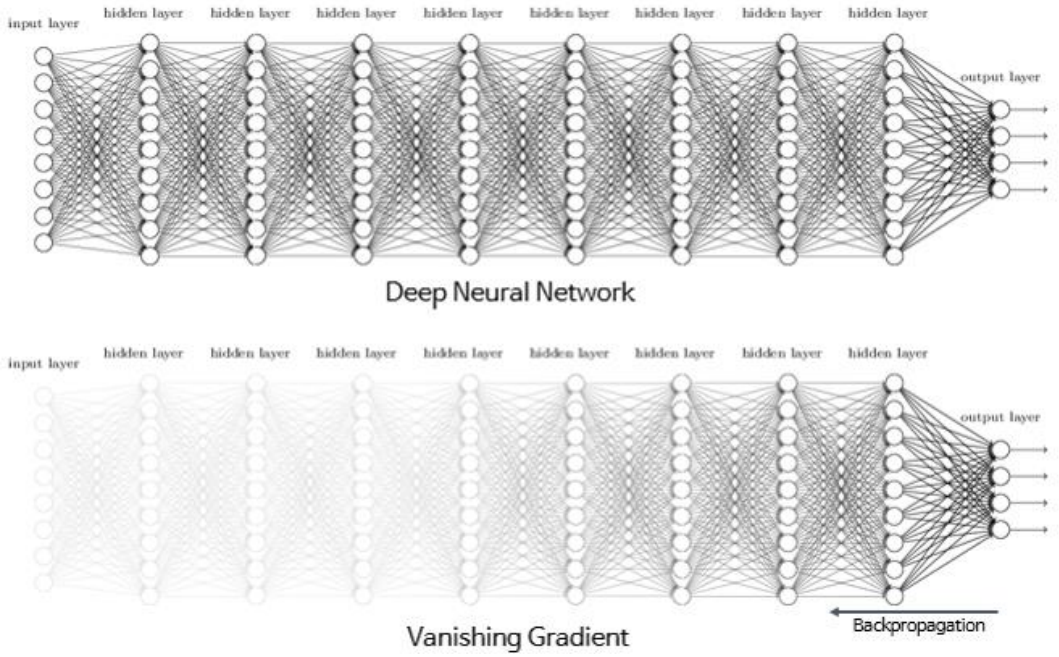
- 신경망의 부활: Back propagation (1974, 1982 by Paul Werbos, 1986 by Hinton)

https://developer.nvidia.com/blog/inference-next-step-gpu-accelerated-deep-learning/



Artificial Intelligence: History

- 두 번째 겨울 (1987~1993): Vanishing gradient



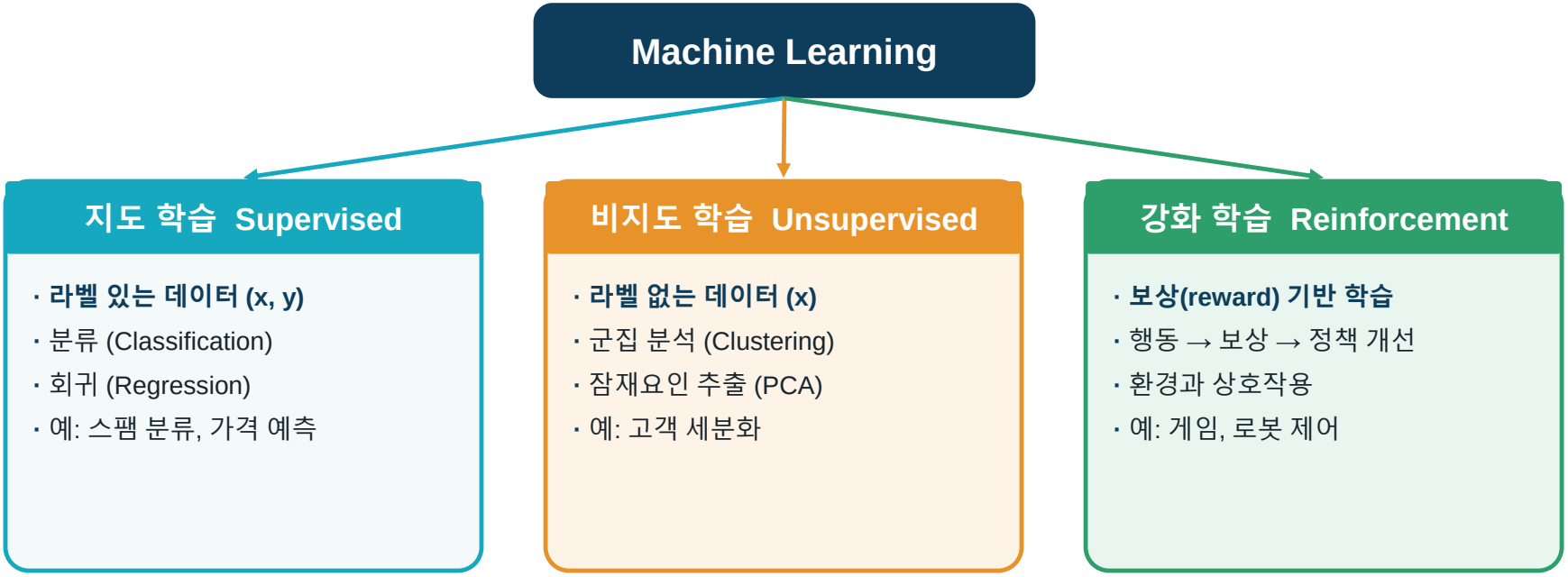
Artificial Intelligence: History

- **폭발적 성장**
  - **활성화 함수 변경:** 값이 0에서 1사이로 작아져 기울기 소실을 유발하던 Sigmoid 함수 대신, 양수 입력에 대해 기울기를 그대로 보존하는 ReLU 함수를 도입.
  - **정교한 가중치 초기화:** 신경망 학습 초기에 가중치(Weight)를 무작위로 적절하게 설정하는 기법을 사용해 기울기 소실이나 폭발을 방지.
  - **잔차 연결(ResNet):** 층이 매우 깊어지더라도 입력된 정보가 우회로를 통해 층을 건너뛰며 직접 전달되도록 설계하여 신호가 소실되는 것을 방지.

OpenAI

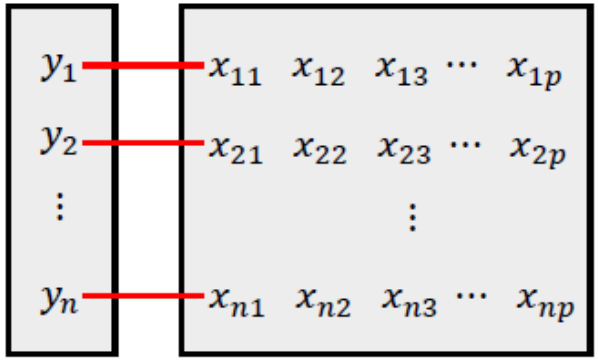
Machine Learning 분류

■ 학습에 사용하는 데이터의 형태와 학습 방식에 따라 크게 세 가지



지도 학습 (Supervised Learning)

- 표식이 있는 자료(labeled data)를 대상으로 함
- 입력 x로부터 표식이 있는 자료 y로의 mapping을 학습하는 방법



- y가 범주형인 경우 → 분류 (Classification)
  - 예) spam 메일 분류, face detection, 질병 분류
- y가 연속형인 경우 → 회귀 (Regression)
  - 주식 가격 예측, 소비자 연령 예측, 기온 예측

지도 학습 (Supervised Learning)

개념

사람이 입력 데이터와 정답(레이블)을 짝지어 주면, 모델이 그 관계를 학습해 처음 보는 데이터의 정답을 예측

사례 이해:
채점된 문제집을 수천 권 풀고 나면, 처음 보는 문제도 풀 수 있게 됨.

학습 과정

① 입력 + 정답 데이터 준비
예) 메일 100만 건 + "스팸 / 정상" 표시

↓

② 모델 학습
입력과 정답 사이의 규칙을 찾아냄

↓

③ 새 데이터 예측
처음 보는 메일 → "스팸일 확률 97%"

지도 학습 (Supervised Learning)

스팸 메일 필터

네이버·카카오 메일이 "스팸 / 정상"으로 표시된 수백만 건을 학습해, 새 메일의 스팸 여부를 자동 판별.

번역기 (파파고 · 구글 번역)

사람이 번역한 한국어–영어 문장 쌍(입력–정답)을 대량으로 학습해 새로운 문장을 번역.

은행 대출 심사 · 신용평가

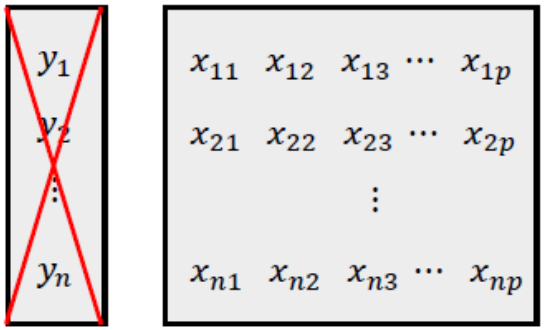
과거 고객 정보(입력)와 연체 여부(정답)를 학습해 새 고객의 연체 위험도를 예측.

아파트 시세 예측

면적·층수·역세권 여부와 실거래가의 관계를 학습합니다. 병원의 영상 판독 보조 AI도 같은 원리

비지도 학습 (Unsupervised Learning)

- 표식이 없는 자료(labeled data)를 대상으로 함
- 자료의 숨겨진 구조(hidden structure)를 찾고자 함



- 하지만 표식이 있는 y가 없기 때문에 찾아낸 structure가 정밀한지에 대한 검증은 불가능함
- 추출된 요인을 이용해 2차 분석을 할 때 주로 사용함
- 군집 분석(Clustering): feature x의 유사성을 이용해 몇 개의 그룹으로 분류
  - Ex) 광고대상자를 고객의 특성으로 바탕으로 분류, K-means clustering, Support Vector Machine (SVM)
- 잠재요인 추출(Latent factor extraction): feature 변수 x에서 관측되지 않은 잠재요인을 추출하는 방법
  - Ex) 주성분 분석 (Principal Component Analysis, PCA)

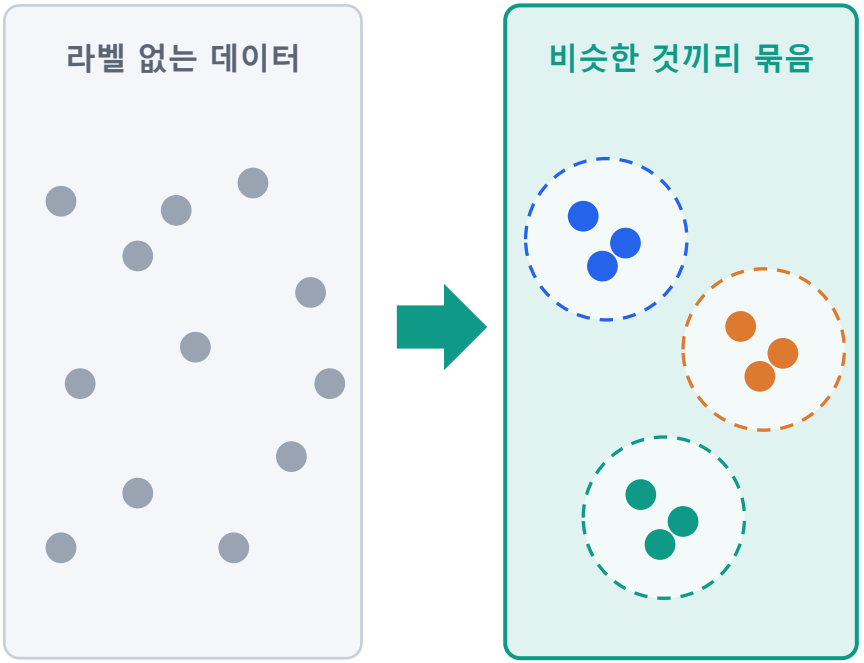
비지도 학습 (Unsupervised Learning)

개념

정답 없이 데이터만 주면, 모델이 스스로 숨어 있는 패턴이나 비슷한 그룹(군집)을 찾아냄.

사례 이해:
이름표 없는 사진 수천 장을 주고 "비슷한 것끼리 알아서 묶어 봐"라고 시키는 것과 동일.

학습 과정



사람이 알려주지 않아도 데이터 속 구조를 발견

비지도 학습 (Unsupervised Learning)

고객 세분화 (이마트 · 쿠팡)

구매 이력만 보고 "육아용품 고객", "심야 간편식 고객"처럼 비슷한 소비 패턴의 그룹을 발견해 맞춤 쿠폰을 보냄.

비슷한 곡 묶기 (멜론 · 스포티파이)

곡의 특성을 분석해 장르 태그 없이도 분위기가 비슷한 곡끼리 묶어 플레이리스트를 만듦.

카드 이상거래 탐지

평소 소비 패턴을 학습해 두었다가, 새벽 해외 고액 결제처럼 패턴에서 크게 벗어난 거래를 포착함

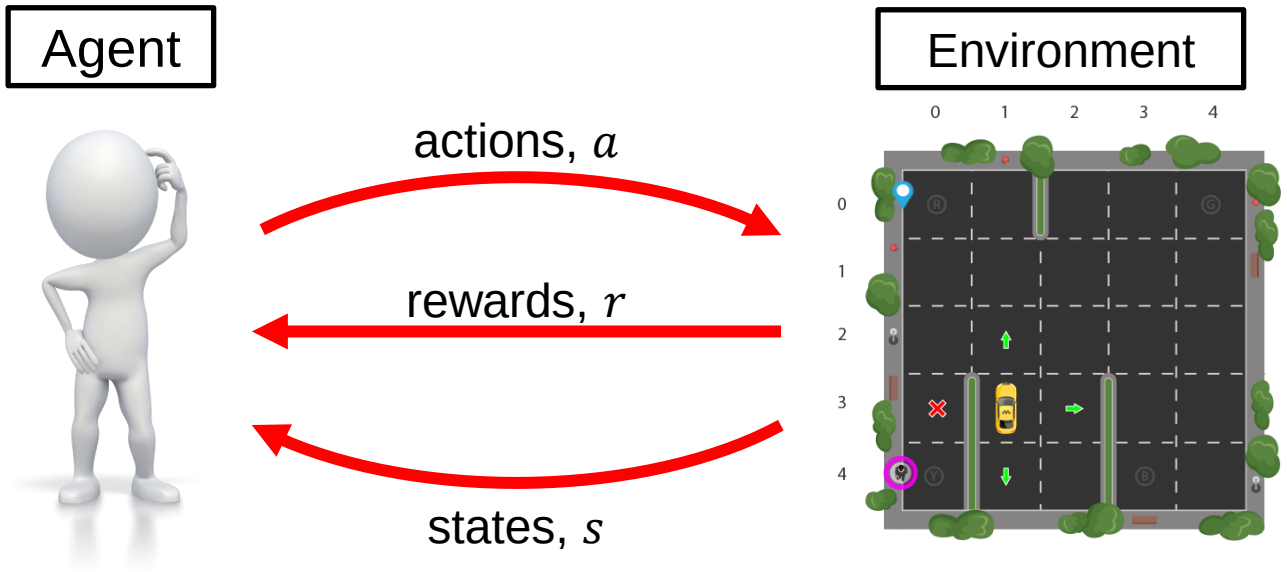
뉴스 기사 묶음 (네이버 뉴스)

같은 사건을 다룬 기사들을 정답 없이도 자동으로 한 묶음(클러스터)으로 모아 보여줌

강화 학습 (Reinforcement Learning)

■ 주어진 환경에 의해 시스템의 성능을 향상시키는 학습 알고리즘

■ 어떤 행동을 했을 때 받는 보상(reward)에 따라 미래의 행동을 바꿔가는 강화(reinforcement) 개념이 기본 아이디어



핵심 : 좋은 보상을 많이 받는 행동을 강화 → 시행착오를 거쳐 최적 정책(policy)을 학습

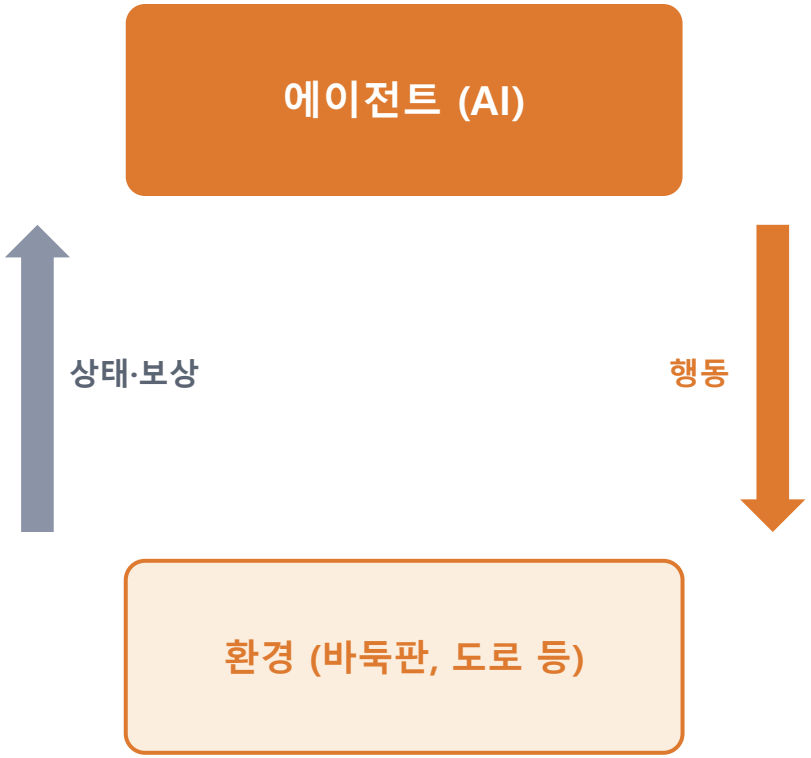
강화 학습 (Reinforcement Learning)

개념

정답 대신 행동의 결과로 보상이나 벌점을 주면, 모델이 시행착오를 통해 보상을 최대화하는 행동 전략을 스스로 터득함.

사례 이해:
"앉아"를 성공하면 간식을 주는 강아지 훈련처럼, 수많은 시도 끝에 보상을 가장 많이 받는 행동을 배움.

학습 구조



이 순환을 수백만 번 반복하며 전략을 개선

강화 학습 (Reinforcement Learning)

알파고 (AlphaGo)

수백만 번의 자가 대국에서 "이기면 보상"을 받으며 바둑 전략을 스스로 터득해 이세돌 9단을 이겼습니다.

자율주행 (테슬라 · 현대차)

시뮬레이션에서 "안전 주행 = 보상, 충돌 = 벌점"을 반복하며 운전 정책을 학습합니다.

배달 배차 최적화 (배달의민족)

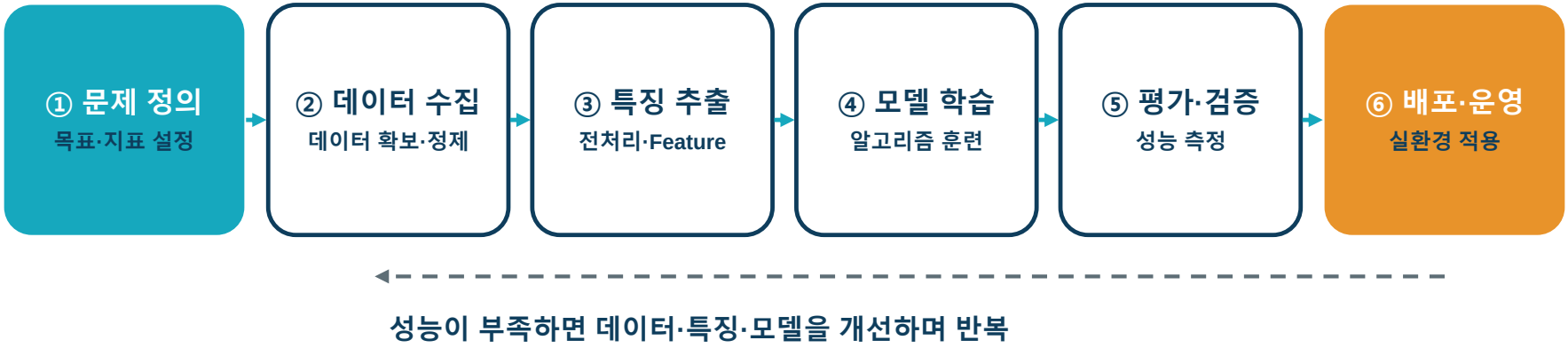
어떤 라이더에게 어떤 주문을 배정할지 시도하고, 배달 시간이 줄면 보상을 주는 식으로 배차 전략을 개선합니다.

콘텐츠 추천 (유튜브 · 넷플릭스)

추천한 영상을 오래 시청하면 보상, 바로 이탈하면 벌점으로 간주해 추천 전략을 계속 조정합니다.

Machine Learning 분석 절차

■ 데이터 준비 → 학습 → 평가 → 배포의 반복적 과정

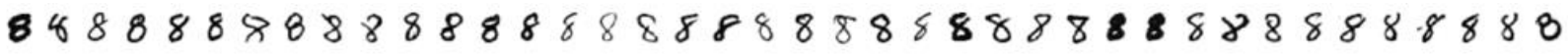


가장 중요한 단계 : 데이터 수집·전처리 (전체 작업 시간의 70~80% 차지)

좋은 특징(feature)이 좋은 모델을 만든다

지식기반 방식에서 기계 학습으로의 대전환

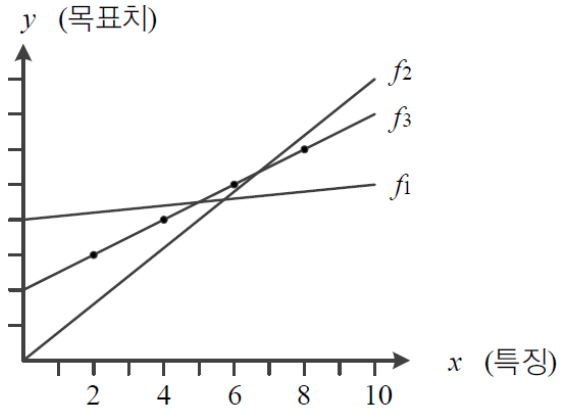
- **큰 깨달음**
  - 지식기반의 한계
  - 단추를 “가운데 구멍이 몇 개 있는 물체”라고 규정하면 많은 오류 발생



  - 사람은 변화가 심한 장면을 아주 쉽게 인식하지만, 왜 그렇게 인식하는지 서술하지는 못함

기계 학습 개념

- **간단한 기계 학습 예제**
  - 가로축은 시간, 세로축은 이동체의 위치
  - 관측한 4개의 점이 데이터

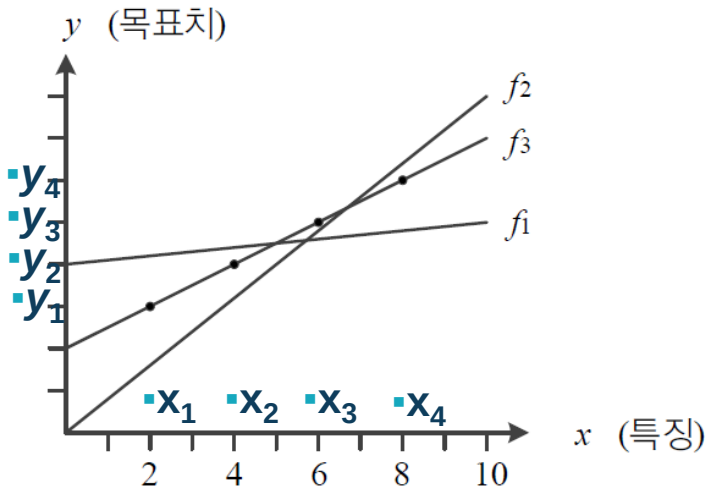


- **예측prediction 문제**
  - 임의의 시간이 주어지면 이때 이동체의 위치는?
  - 회귀regression 문제와 분류classification 문제로 나뉨
    - 회귀는 목표치가 실수, 분류는 부류값 (위는 회귀 문제)

기계 학습 개념

- **훈련집합**
  - 가로축은 특징, 세로축은 목표치
  - 관측한 4개의 점이 훈련집합을 구성함

$$\mathbb{X} = \{\mathbf{x}_1 = (2.0), \mathbf{x}_2 = (4.0), \mathbf{x}_3 = (6.0), \mathbf{x}_4 = (8.0)\}$$
$$\mathbb{Y} = \{y_1 = 3.0, y_2 = 4.0, y_3 = 5.0, y_4 = 6.0\}$$



기계 학습 개념

- **데이터를 어떻게 모델링할 것인가**
  - 눈대중으로 보면 직선을 이루므로 직선을 선택하자 → 모델로 직선을 선택한 셈
  - 직선 모델의 수식
    - 2개의 매개변수 *w*와 *b*

$$y = wx + b \qquad (1.2)$$

- **기계 학습은**
  - 가장 정확하게 예측할 수 있는, 즉 최적의 매개변수를 찾는 작업
  - 처음에는 최적값을 모르므로 임의의 값에서 시작하고, 점점 성능을 개선하여 최적에 도달
  - [그림 1-4]의 예에서는 f_1에서 시작하여 $f_1 \rightarrow f_2 \rightarrow f_3$
    - 최적인 f_3은 w=0.5와 b=2.0

기계 학습 개념

- **학습을 마치면,**
  - 예측에 사용
  - 예) 10.0 순간의 이동체 위치를 알고자 하면, $f_3(10.0)=0.5*10.0+2.0=7.0$이라 예측함

- **기계 학습의 궁극적인 목표**
  - 훈련집합에 없는 새로운 샘플에 대한 오류를 최소화 (새로운 샘플 집합: 테스트 집합)
  - 테스트 집합에 대한 높은 성능을 일반화generalization 능력이라 부름

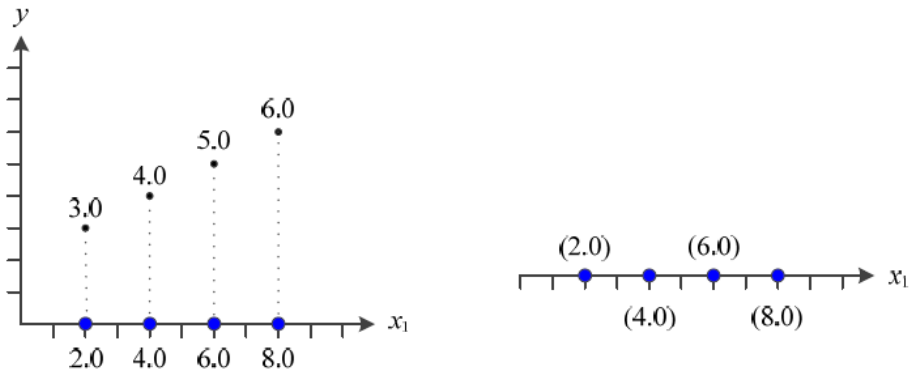
사람의 학습과 기계 학습

사람의 학습과 기계 학습의 비교

기준	사람의 학습	기계 학습
학습 과정	능동적	수동적
데이터 형식	자연에 존재하는 그대로	일정한 형식에 맞추어 사람이 준비함
동시에 학습 가능한 과업 수	자연스럽게 여러 과업을 학습	하나의 과업만 가능
학습 원리에 대한 지식	매우 제한적으로 알려져 있음	모든 과정이 밝혀져 있음
수학 의존도	매우 낮음	매우 높음
성능 평가	경우에 따라 객관적이거나 주관적	객관적(수치로 평가, 예를 들어 정확률 99.8%)
역사	수백만 년	60년 가량

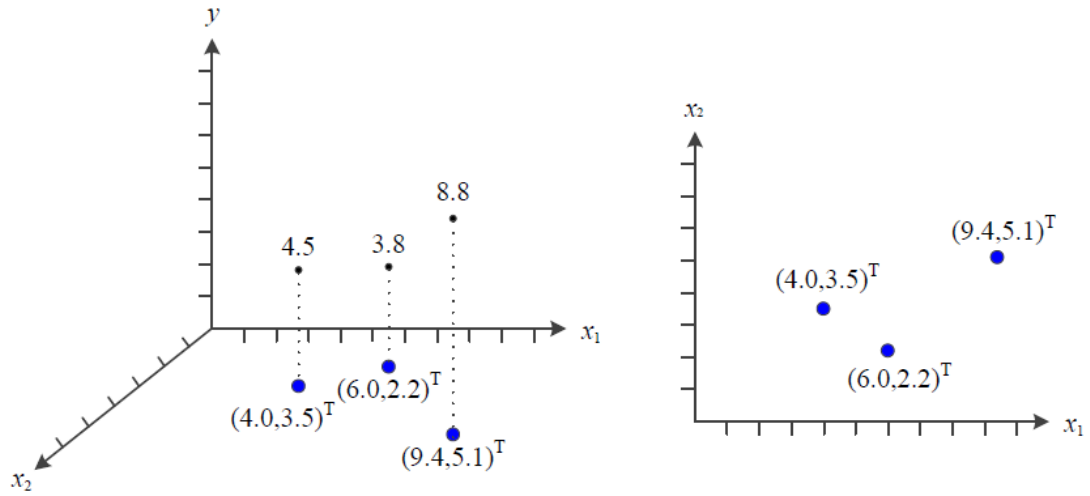
1차원과 2차원 특징 공간

- **1차원 특징 공간** →



(a) 1차원 특징 공간(왼쪽: 특징과 목푯값을 축으로 표시, 오른쪽: 특징만 축으로 표시)

- **2차원 특징 공간** →
  - 특징 벡터 표기
    - $x=(x_1,x_2)^T$
  - 예시
    - x=(몸무게,키)^T, y=장타율
    - x=(체온,두통)^T, y=감기 여부

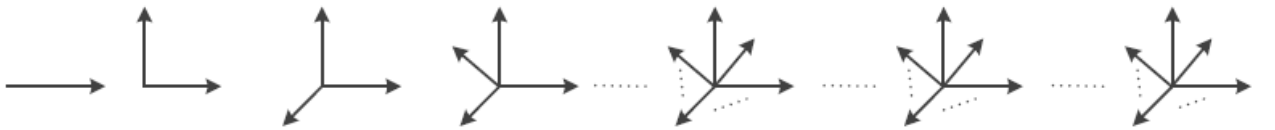


(b) 2차원 특징 공간(왼쪽: 특징 벡터와 목푯값을 축으로 표시, 오른쪽: 특징 벡터만 축으로 표시)

특징 공간과 데이터의 표현

다차원 특징 공간

- **다차원 특징 공간 예제**



1차원	2차원	3차원	4차원	13차원	784차원	54877차원
		Haberman survival	Iris	Wine	MNIST	Farm ads

Haberman survival: $\mathbf{x} = (\text{나이}, \text{수술년도}, \text{양성 림프샘 개수})^T$

Iris: $\mathbf{x} = (\text{꽃받침 길이}, \text{꽃받침 너비}, \text{꽃잎 길이}, \text{꽃잎 너비})^T$

Wine: $\mathbf{x} = (\text{Alcohol}, \text{Malic acid}, \text{Ash}, \text{Alcalinity of ash}, \text{Magnesium}, \text{Total phenols}, \text{Flavanoids}, \text{Nonflavanoid phenols}$
$\text{Proanthocyanins}, \text{Color intensity}, \text{Hue}, \text{OD280 / OD315 of diluted wines}, \text{Proline})^T$

MNIST: $\mathbf{x} = (\text{화소1}, \text{화소2}, \cdots, \text{화소784})^T$

Farm ads: $\mathbf{x} = (\text{단어1}, \text{단어2}, \cdots, \text{단어54877})^T$

다차원 특징 공간

다차원 특징 공간

- ***d*-차원 데이터**
  - 특징 벡터 표기: $x=(x_1,x_2, \ldots ,x_d)^T$
- ***d*-차원 데이터를 위한학습 모델**
  - 직선 모델을 사용하는 경우 매개변수 수=d+1

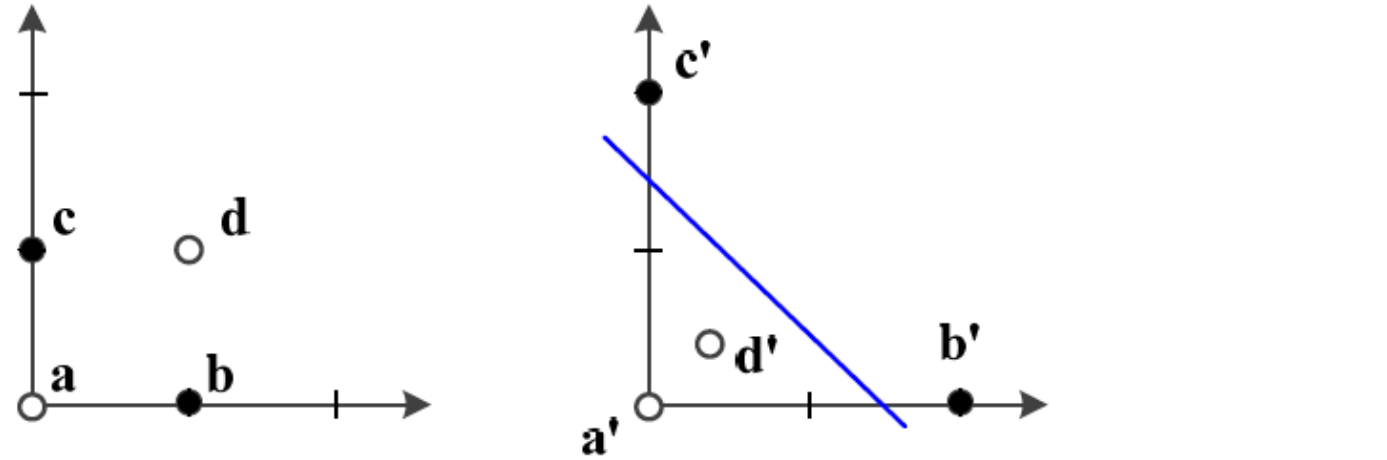
$$y = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b \tag{1.3}$$

  - 2차 곡선 모델을 사용하면 매개변수 수가 크게 증가
    - 매개변수 수=d^2+d+1
    - 예) Iris 데이터: d=4이므로 21개의 매개변수
    - 예) MNIST 데이터: d=784이므로 615,441개의 매개변수

$$y = w_1 x_1^2 + w_2 x_2^2 + \cdots + w_d x_d^2 + w_{d+1} x_1 x_2 + \cdots + w_{d^2} x_{d-1} x_d + w_{d^2+1} x_1 + \cdots + w_{d^2+d} x_d + b \tag{1.5}$$

특징 공간 변환과 표현 학습

- **선형 분리 불가능**linearly non-separable**한 원래 특징 공간**
  - 직선 모델을 적용하면 75% 정확률이 한계



(a) 원래 특징 공간 (b) 분류에 더 유리하도록 변환된 새로운 특징 공간

특징 공간 변환

특징 공간 변환과 표현 학습

- **식 (1.6)으로 변환된 새로운 특징 공간 ([그림 1-7(b)])**
  - 직선 모델로 100% 정확률

$$\text{원래 특징 벡터 } \mathbf{x} = (x_1, x_2)^{\mathrm{T}} \rightarrow \text{변환된 특징 벡터 } \mathbf{x}' = \left(\frac{x_1}{2x_1x_2 + 0.5}, \frac{x_2}{2x_1x_2 + 0.5} \right)^{\mathrm{T}} \quad (1.6)$$

$$\mathbf{a} = (0,0)^{\mathrm{T}} \rightarrow \mathbf{a}' = (0,0)^{\mathrm{T}}$$

$$\mathbf{b} = (1,0)^{\mathrm{T}} \rightarrow \mathbf{b}' = (2,0)^{\mathrm{T}}$$

$$\mathbf{c} = (0,1)^{\mathrm{T}} \rightarrow \mathbf{c}' = (0,2)^{\mathrm{T}}$$

$$\mathbf{d} = (1,1)^{\mathrm{T}} \rightarrow \mathbf{d}' = (0.4,0.4)^{\mathrm{T}}$$

- **표현 학습representation learning**
  - 좋은 특징 공간을 자동으로 찾는 작업
  - 딥러닝은 다수의 은닉층을 가진 신경망을 이용하여 계층적인 특징 공간을 찾아냄
    - 왼쪽 은닉층은 저급 특징(에지, 구석점 등), 오른쪽은 고급 특징(얼굴, 바퀴 등) 추출
  - [그림 1-7]은 표현 학습을 사람이 직관으로 수행한 셈

특징 공간 변환과 표현 학습

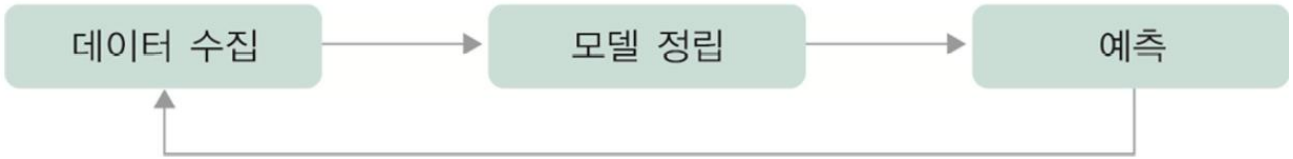
- **차원에 대한 몇 가지 설명**
  - 차원에 무관하게 수식 적용 가능함
    - 예) 두 점 a=(a_1,a_2, … ,a_d)^{T}와 b=(b_1,b_2, … ,b_d)^T 사이의 거리는 모든 d에 대해 성립

$$dist(\mathbf{a},\mathbf{b}) = \sqrt{\sum_{i=1}^{d}(a_i - b_i)^2} \qquad (1.7)$$

    - 보통 2~3차원의 저차원에서 식을 고안한 다음 고차원으로 확장 적용
  - 차원의 저주
    - 차원이 높아짐에 따라 발생하는 현실적인 문제들
    - 예) d=4인 Iris 데이터에서 축마다 100개 구간으로 나누면 총 100^4=1억 개의 칸
    - 예) d=784인 MNIST 샘플의 화소가 0과 1값을 가진다면 2^{784}개의 칸. 이 거대한 공간에 고작 6만 개의 샘플을 흩뿌린 매우 희소한 분포

데이터에 대한 이해

- **과학 기술의 발전 과정**



  - 예) 튀코 브라헤는 천동설이라는 틀린 모델을 선택함으로써 자신이 수집한 데이터를 설명하지 못함. 케플러는 지동설 모델을 도입하여 제1, 제2, 제 3법칙을 완성함

- **기계 학습**
  - 기계 학습이 푸는 문제는 훨씬 복잡함
    - 예) '8' 숫자 패턴과 '단추' 패턴의 다양한 변화 양상
  - 단순한 수학 공식으로 표현 불가능함
  - 자동으로 모델을 찾아내는 과정이 필수

데이터 생성 과정

- 데이터 생성 과정을 완전히 아는 인위적 상황의 예제
  - 예) 두 개 주사위를 던져 나온 눈의 합을 x라 할 때, $y=(x-7)^2+1$ 점을 받는 게임
  - 이런 상황을 '데이터 생성 과정을 완전히 알고 있다'고 말함
    - x를 알면 정확히 y를 예측할 수 있음
      - 실제 주사위를 던져 $\mathbb{X} = \{3,10,8,5\}$를 얻었다면, $\mathbb{Y} = \{17,10,2,5\}$
    - x의 발생 확률 $P(x)$를 정확히 알 수 있음
    - $P(x)$를 알고 있으므로, 새로운 데이터 생성 가능

- 실제 기계 학습 문제
  - 데이터 생성 과정을 알 수 없음
  - 단지 주어진 훈련집합 $\mathbb{X}, \mathbb{Y}$로 예측 모델 또는 생성 모델을 근사 추정할 수 있을 뿐

데이터베이스의 중요성

- **데이터베이스의 품질**
  - 주어진 응용에 맞는 충분히 다양한 데이터를 충분한 양만큼 수집 → 추정 정확도 높아짐
  - 예) 정면 얼굴만 가진 데이터베이스로 학습하고 나면, 기운 얼굴은 매우 낮은 성능
  - 주어진 응용 환경을 자세히 살핀 다음 그에 맞는 데이터베이스 확보는 아주 중요함

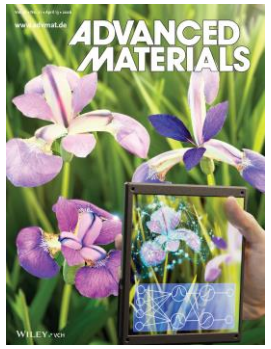
- **아주 많은 공개 데이터베이스**
  - 기계 학습의 초파리로 여겨지는 3가지 데이터베이스: Iris, MNIST, ImageNet
  - 위키피디아에서 ‘list of datasets for machine learning research’로 검색
  - UCI 리퍼지토리 (2017년11월 기준으로 394개 데이터베이스 제공)

데이터베이스의 중요성

- Iris 데이터베이스는 통계학자인 피셔 교수가 1936년에 캐나다 동부 해안의 가스페 반도에 서식하는 3종의 붓꽃(setosa, versicolor, virginica)을 50송이씩 채취하여 만들었다[Fisher1936]. 150개 샘플 각각에 대해 꽃받침 길이, 꽃받침 너비, 꽃잎 길이, 꽃잎 너비를 측정하여 기록하였다. 따라서 4차원 특징 공간이 형성되며 목푯값은 3종을 숫자로 표시함으로써 1, 2, 3 값 중의 하나이다. http://archive.ics.uci.edu/ml/datasets/Iris에 접속하여 내려받을 수 있다.

Sepal length	Sepal width	Petal length	Petal width	Species
5.2	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
5.5	2.3	4.0	1.3	*I. versicolor*
6.3	3.3	6.0	2.5	*I. virginica*
5.8	2.7	5.1	1.9	*I. virginica*
7.1	3.0	5.9	2.1	*I. virginica*
6.3	2.9	5.6	1.8	*I. virginica*

Setosa

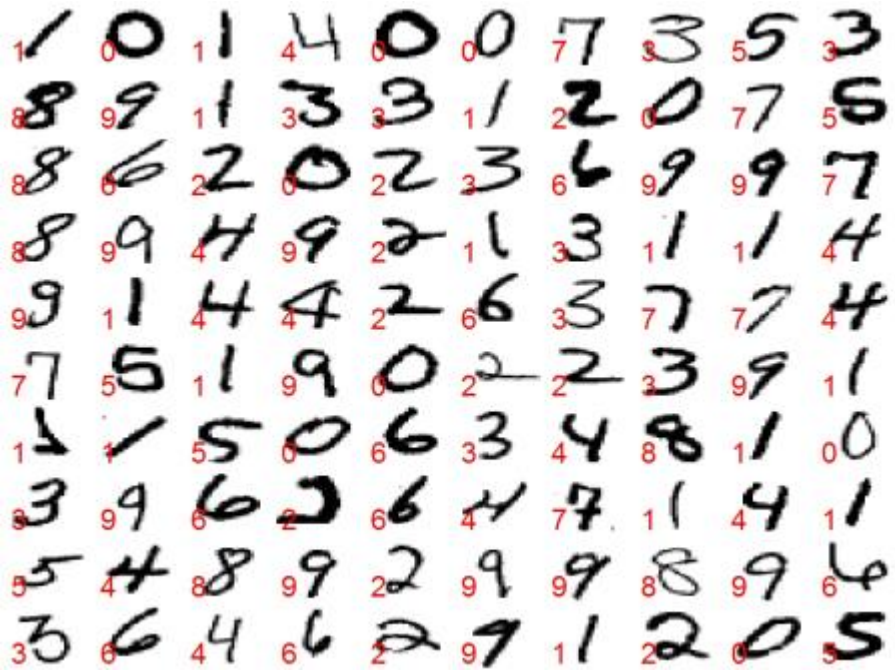


Versicolor

Virginica

데이터베이스의 중요성

- MNIST 데이터베이스는 미국표준국(NIST)에서 수집한 필기 숫자 데이터베이스로, 훈련집합 60,000자, 테스트집합 10,000자를 제공한다. http://yann.lecun.com/exdb/mnist에 접속하면 무료로 내려받을 수 있으며, 1988년부터 시작한 인식률 경쟁 기록도 볼 수 있다. 2017년 8월 기준으로는 [Ciresan2012] 논문이 0.23%의 오류율로 최고 자리를 차지하고 있다. 테스트집합에 있는 10,000개 샘플에서 단지 23개만 틀린 것이다.



데이터베이스의 중요성

- ImageNet 데이터베이스는 정보검색 분야에서 만든 WordNet의 단어 계층 분류를 그대로 따랐고, 부류마다 수백에서 수천 개의 영상을 수집하였다[Deng2009]. 총 21,841개 부류에 대해 총 14,197,122개의 영상을 보유하고 있다. 그중에서 1,000개 부류를 뽑아 ILSVRC(ImageNet Large Scale Visual Recognition Challenge)라는 영상인식 경진대회를 2010년부터 매년 개최하고 있다. 대회 결과에 대한 자세한 내용은 4.4절을 참조하라. http:// image-net.org에서 내려받을 수 있다.

(a) 'swing' 부류

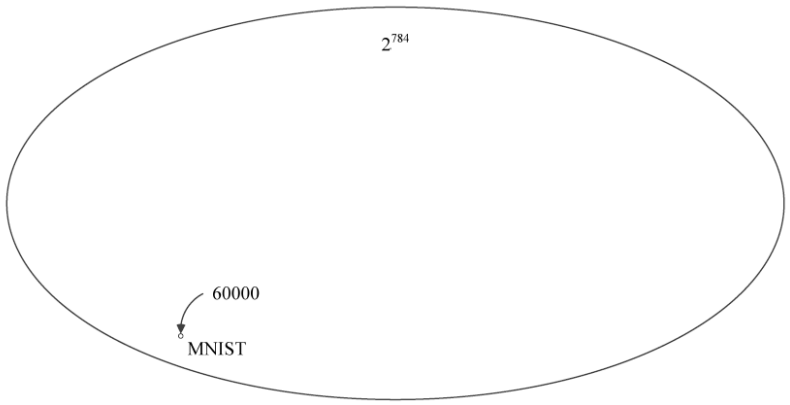
(b) 'Great white shark' 부류

ImageNet의 예제 영상

데이터베이스 크기와 기계 학습 성능

- **데이터베이스의 왜소한 크기**
  - 예) MNIST: 28*28 흑백 비트맵이라면 서로 다른 총 샘플 수는 2^{784}가지이지만, MNIST는 고작 6만 개 샘플

$$\frac{60{,}000}{2^{784}} \approx \frac{6 \times 10^4}{10^{236}} \approx 10^{-231}$$



1) 데이터 공간은 텅 비어 있다

784차원 공간의 점을 무작위로 하나 뽑으면 거의 항상 의미 없는 흑백 노이즈가 나옵니다. 사람이 "숫자"로 알아볼 수 있는 이미지는 그 공간에서 극히 좁은 영역에만 몰려 있음.

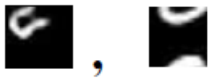
2) 그런데도 학습이 된다 (핵심)

2^{784}개를 다 채우려면 데이터가 무한히 필요할 텐데, 실제로는 6만 개만으로도 손글씨 숫자를 잘 분류합니다. 왜냐하면 진짜 손글씨 숫자들은 784차원 전체에 흩어져 있는 게 아니라, 훨씬 낮은 차원의 매니폴드(곡면) 위에 몰려 있기 때문임. 숫자 이미지를 결정하는 실제 자유도는 획의 굵기·기울기·크기·위치·필기 스타일 정도 — 수십 개 수준이지 784개가 아님.

3) 차원의 저주에 대한 답

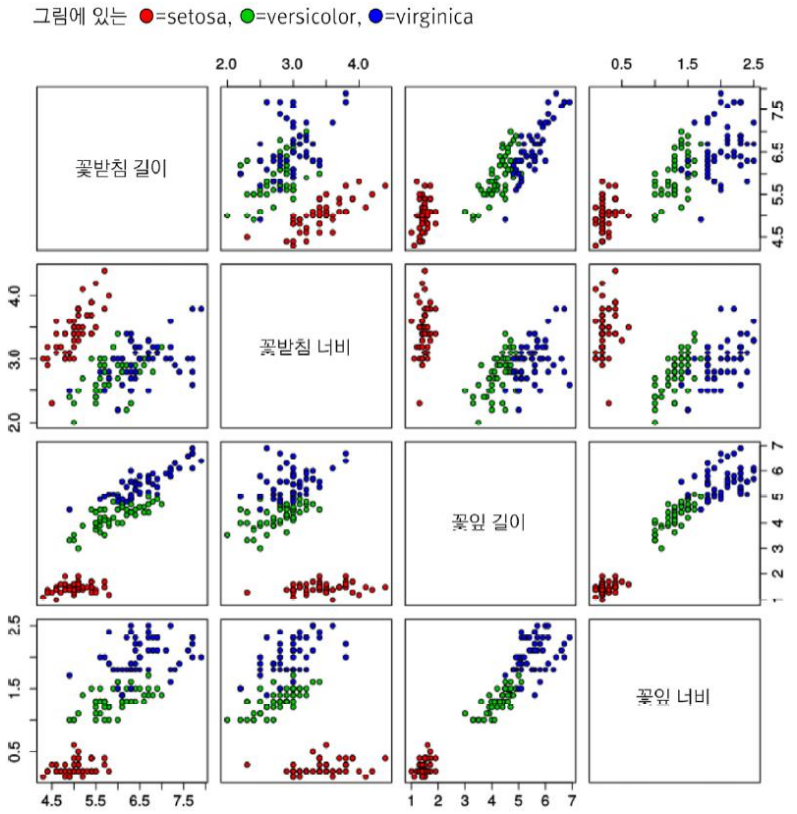
겉보기 차원(784)이 아무리 커도 본질 차원이 낮으면 적은 샘플로 학습이 가능하며 딥러닝이 고차원 데이터에서 작동하는 이유가 바로 이것임.

데이터베이스 크기와 기계 학습 성능

- **왜소한 데이터베이스로 어떻게 높은 성능을 달성하는가?**
  - 방대한 공간에서 실제 데이터가 발생하는 곳은 매우 작은 부분 공간임
    -  와 같은 샘플의 발생 확률은 거의 0
  - 매니폴드 가정
    -  와 같이 일정한 규칙에 따라 매끄럽게 변화

데이터 가시화

- **4차원 이상의 초공간은 한꺼번에 가시화 불가능**
- **여러 가지 가시화 기법**
  - 2개씩 조합하여 여러 개의 그래프 그림

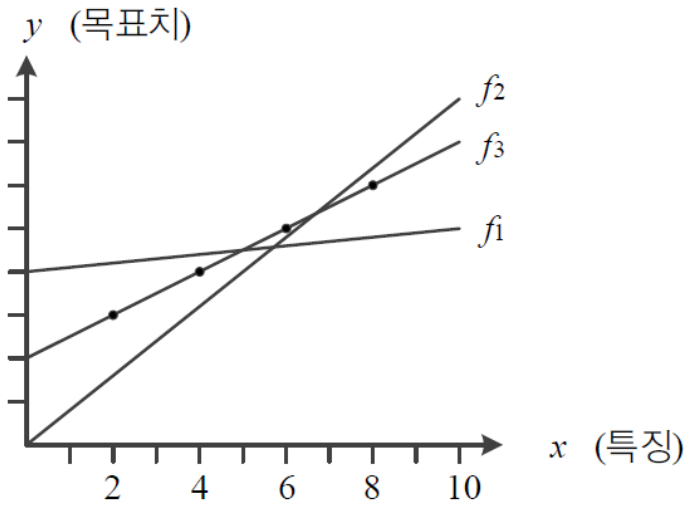


  - 고차원 공간을 저차원으로 변환하는 기법들

간단한 기계 학습의 예

- 선형 회귀 문제
  - [그림 1-4]: 식 (1.2)의 직선 모델을 사용하므로 두 개의 매개변수 $\Theta = (w, b)^{\mathrm{T}}$

$$y = wx + b \qquad (1.2)$$



간단한 기계 학습의 예

- 목적 함수objective function (또는 비용 함수cost function)
  - 식 (1.8)은 선형 회귀를 위한 목적 함수
    - $f_\Theta(\mathbf{x}_i)$는 예측함수의 출력, y_i는 예측함수가 맞추어야 하는 목푯값이므로 $f_\Theta(\mathbf{x}_i)$-y_i는 오차
    - 식 (1.8)을 평균제곱오차MSE(mean squared error)라 부름

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(f_\Theta(\mathbf{x}_i) - y_i)^2 \qquad (1.8)$$

  - 처음에는 최적 매개변수 값을 알 수 없으므로 난수로 $\Theta_1 = (w_1, b_1)^T$ 설정 → $\Theta_2 = (w_2, b_2)^T$로 개선 → $\Theta_3 = (w_3, b_3)^T$로 개선 → Θ_3는 최적해 $\hat{\Theta}$
    - 이때 $J(\Theta_1) > J(\Theta_2) > J(\Theta_3)$

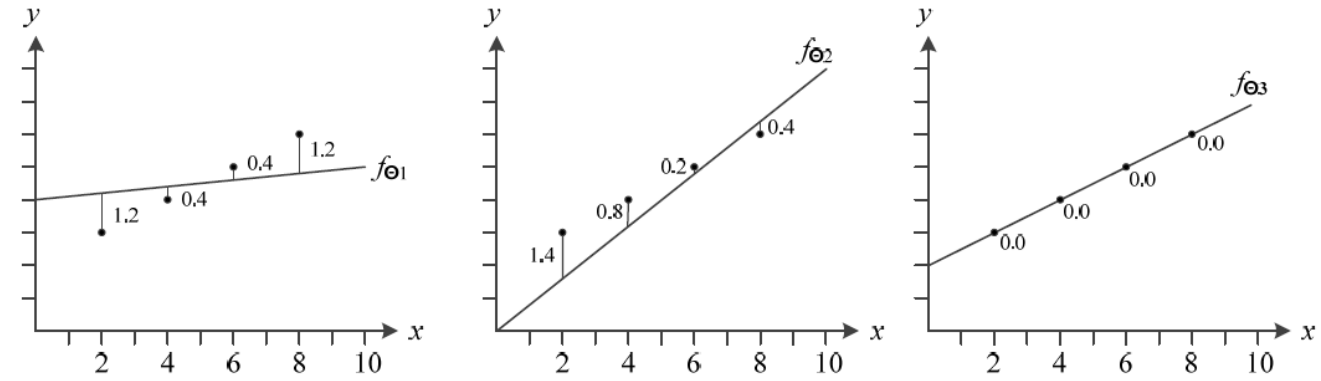
간단한 기계 학습의 예

- 훈련집합
  - Θ_1을 개선하여 $\Theta_2 = (0.8, 0.0)^T$가 되었다고 가정

$$\mathbf{x}_1, y_1 \rightarrow \left(f_{\Theta_2}(2.0) - 3.0\right)^2 = \left((0.8 * 2.0 + 0.0) - 3.0\right)^2 = 1.96$$
$$\mathbf{x}_2, y_2 \rightarrow \left(f_{\Theta_2}(4.0) - 4.0\right)^2 = \left((0.8 * 4.0 + 0.0) - 4.0\right)^2 = 0.64$$
$$\mathbf{x}_3, y_3 \rightarrow \left(f_{\Theta_2}(6.0) - 5.0\right)^2 = \left((0.8 * 6.0 + 0.0) - 5.0\right)^2 = 0.04$$
$$\mathbf{x}_4, y_4 \rightarrow \left(f_{\Theta_2}(8.0) - 6.0\right)^2 = \left((0.8 * 8.0 + 0.0) - 6.0\right)^2 = 0.16$$

$$\rightarrow J(\Theta_2) = 0.7$$

  - Θ_2를 개선하여 $\Theta_3 = (0.5, 2.0)^T$가 되었다고 가정
  - 이때 $J(\Theta_3) = 0.0$이 되어 Θ_3은 최적값 $\hat{\Theta}$ 이 됨



(a) 초기 매개변수 Θ_1 (b) Θ_1을 개선하여 Θ_2가 됨 (c) Θ_2를 개선하여 최적의 Θ_3을 찾음

간단한 기계 학습의 예

- **기계 학습이 할 일을 공식화하면,**

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} J(\Theta) \qquad (1.9)$$

  - 기계 학습은 작은 개선을 반복하여 최적해를 찾아가는 수치적 방법으로 식 (1.9)를 사용

- **알고리즘 형식으로 쓰면,**

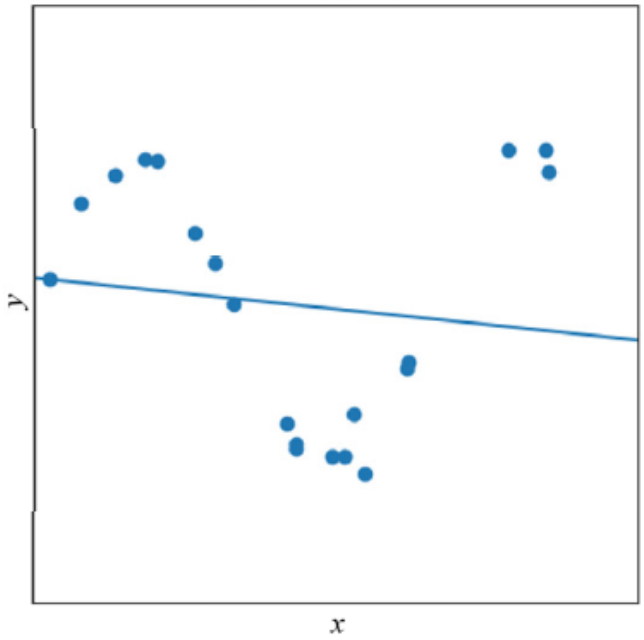
알고리즘 1–1 기계 학습 알고리즘

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$
출력: 최적의 매개변수 $\hat{\Theta}$

```
1  난수를 생성하여 초기 해 Θ1을 설정한다.
2  t=1
3  while (J(Θt)가 0.0에 충분히 가깝지 않음)      // 수렴 여부 검사
4      J(Θt)가 작아지는 방향 ΔΘt를 구한다.      // ΔΘt는 주로 미분을 사용하여 구함
5      Θt+1 = Θt + ΔΘt
6      t=t+1
7  Θ̂ = Θt
```

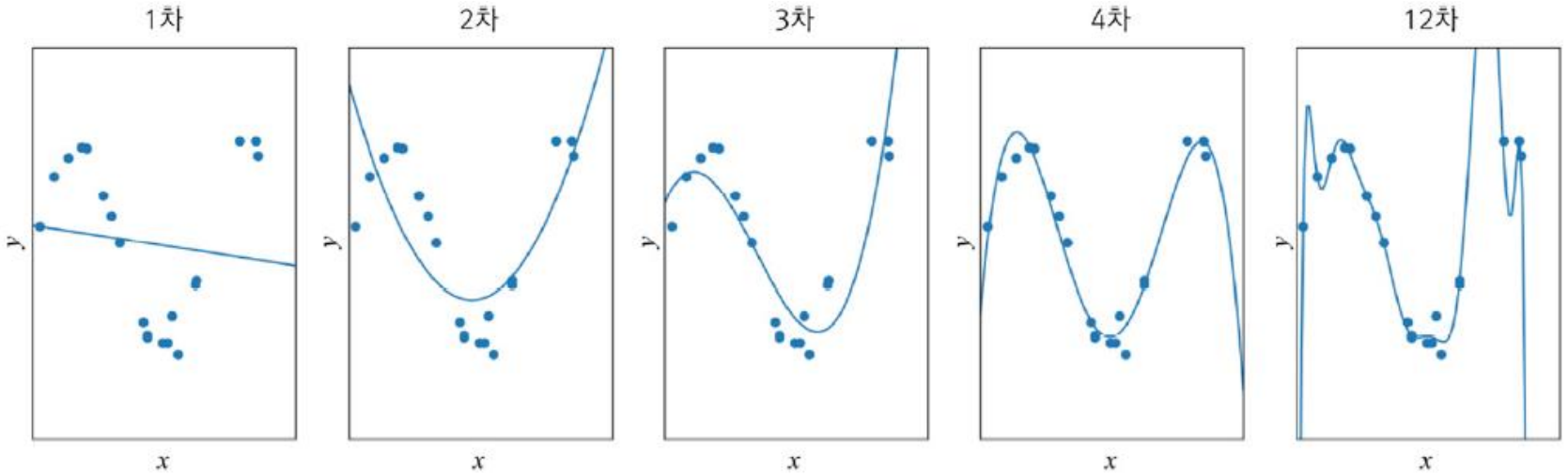
간단한 기계 학습의 예

- **좀더 현실적인 상황**
  - 지금까지는 데이터가 선형을 이루는 아주 단순한 상황을 고려함
  - 실제 세계는 선형이 아니며 잡음이 섞임 → 비선형 모델이 필요



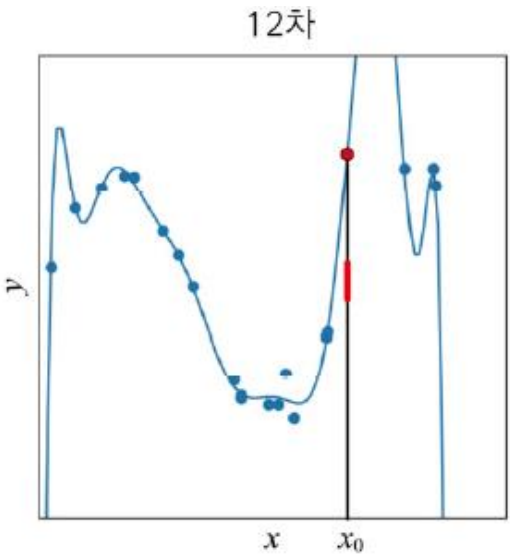
과소적합과 과잉적합

- **1차 모델은 과소적합**
  - 모델의 '용량이 작아' 오차가 클 수밖에 없는 현상
- **비선형 모델을 사용하는 대안**
  - 2차, 3차, 4차, 12차는 다항식 곡선을 선택한 예
  - 1차(선형)에 비해 오차가 크게 감소함



과소적합과 과잉적합

- **과잉적합**
  - 12차 다항식 곡선을 채택한다면 훈련집합에 대해 거의 완벽하게 근사화함
  - 하지만 '새로운' 데이터를 예측한다면 큰 문제 발생
    - x_0에서 빨간 막대 근방을 예측해야 하지만 빨간 점을 예측
  - 이유는 '용량이 크기' 때문. 학습 과정에서 잡음까지 수용 → 과잉적합 현상
  - 적절한 용량의 모델을 선택하는 모델 선택 작업이 필요함

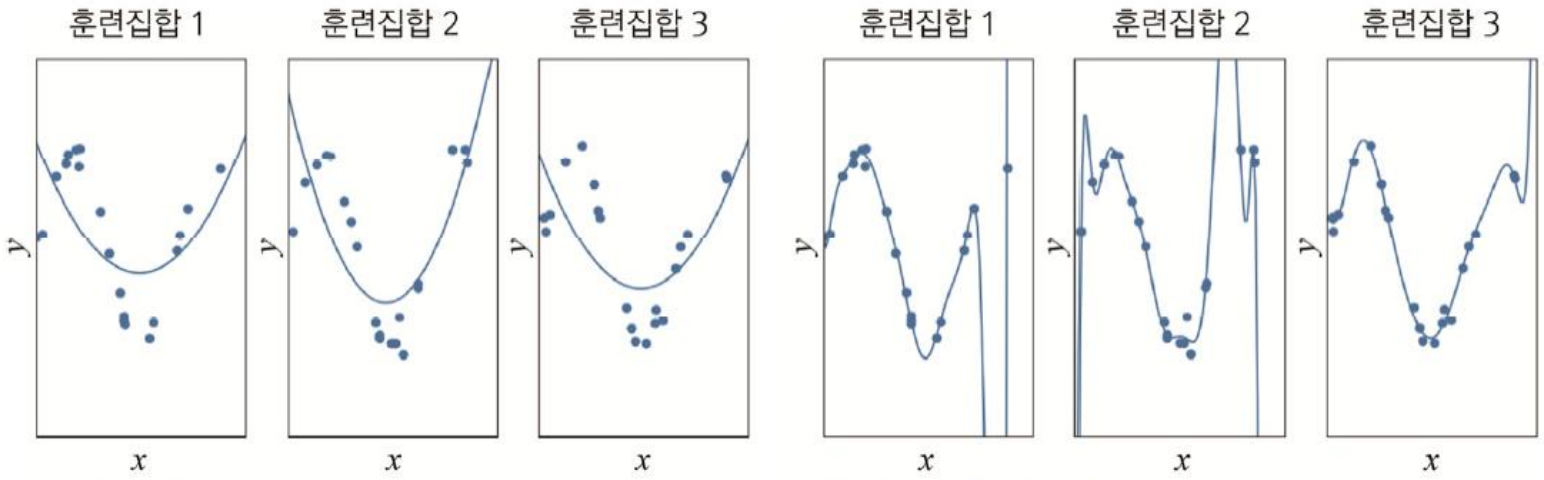


바이어스와 분산

- **1차~12차 다항식 모델의 비교 관찰**
  - 1~2차는 훈련집합과 테스트집합 모두 낮은 성능
  - 12차는 훈련집합에 높은 성능을 보이나 테스트집합에서는 낮은 성능 → 낮은 일반화 능력
  - 3~4차는 훈련집합에 대해 12차보다 낮겠지만 테스트집합에는 높은 성능 → 높은 일반화 능력

바이어스와 분산

- **훈련집합을 여러 번 수집하여 1차~12차에 적용하는 실험**
  - 2차는 매번 큰 오차 → 바이어스가 큼. 하지만 비슷한 모델을 얻음 → 낮은 분산
  - 12차는 매번 작은 오차 → 바이어스가 작음. 하지만 크게 다른 모델을 얻음 → 높은 분산
  - 일반적으로 용량이 작은 모델은 바이어스는 크고 분산은 작음. 복잡한 모델은 바이어스는 작고 분산은 큼
  - 바이어스와 분산은 트레이드오프 관계

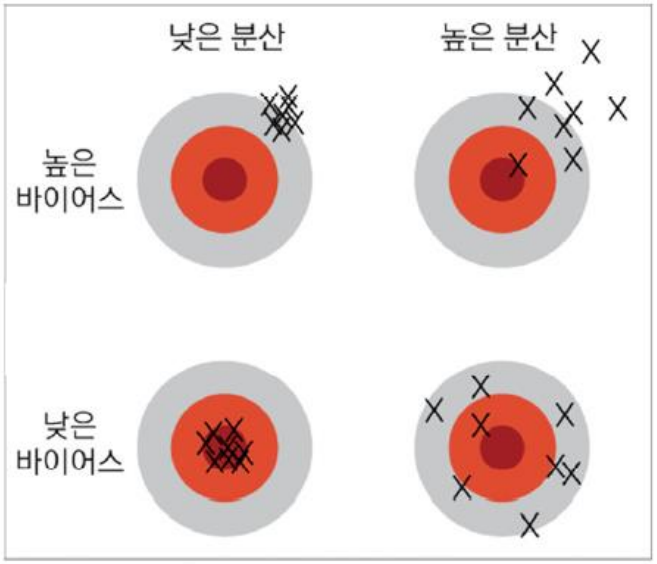


(a) 2차 모델(바이어스는 크고, 분산은 작음)

(b) 12차 모델(바이어스는 작고, 분산은 큼)

바이어스와 분산

- **기계 학습의 목표**
  - 낮은 바이어스와 낮은 분산을 가진 예측기 제작이 목표. 즉 왼쪽 아래 상황



  - 하지만 바이어스와 분산은 트레이드오프 관계
  - 따라서 바이어스 희생을 최소로 유지하며 분산을 최대로 낮추는 전략 필요

모델 선택 알고리즘

- **검증집합을 이용한 모델 선택**
  - 훈련집합과 테스트집합과 다른 별도의 검증집합을 가진 상황

알고리즘 1-2 검증집합을 이용한 모델 선택

입력 : 모델집합 Ω, 훈련집합, 검증집합, 테스트집합
출력 : 최적 모델과 성능

```
1  for (Ω에 있는 각각의 모델)
2      모델을 훈련집합으로 학습시킨다.
3      검증집합으로 학습된 모델의 성능을 측정한다.   // 검증 성능 측정
4  가장 높은 성능을 보인 모델을 선택한다.
5  테스트집합으로 선택된 모델의 성능을 측정한다.
```

모델 선택 알고리즘

- **교차검증cross validation**
  - 비용 문제로 별도의 검증집합이 없는 상황에 유용한 모델 선택 기법
  - 훈련집합을 등분하여, 학습과 평가 과정을 여러 번 반복한 후 평균 사용

알고리즘 1-3 교차검증에 의한 모델 선택

입력 : 모델집합 Ω, 훈련집합, 테스트집합, 그룹 개수 k
출력 : 최적 모델과 성능

```
1  훈련집합을 k개의 그룹으로 등분한다.
2  for (Ω에 있는 각각의 모델)
3        for (i=1 to k)
4              i번째 그룹을 제외한 k-1개 그룹으로 모델을 학습시킨다.
5              학습된 모델의 성능을 i번째 그룹으로 측정한다.
6        k개 성능을 평균하여 해당 모델의 성능으로 취한다.
7  가장 높은 성능을 보인 모델을 선택한다.
8  테스트집합으로 선택된 모델의 성능을 측정한다.
```

모델 선택 알고리즘

- **부트스트랩boot strap**
  - 난수를 이용한 샘플링 반복

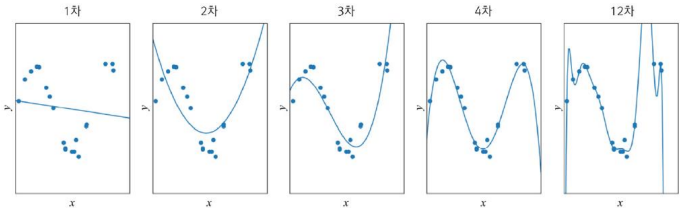
알고리즘 1-4 부트스트랩을 이용한 모델 선택

입력 : 모델집합 Ω, 훈련집합, 테스트집합, 샘플링 비율 $p(0<p\leq 1)$, 반복횟수 T
출력 : 최적 모델과 성능

```
1  for (Ω에 있는 각각의 모델)
2      for (i=1 to T)
3          훈련집합 𝕏에서 pn개 샘플을 뽑아 새로운 훈련집합 𝕏′를 구성한다. 이때 대치를 허용한다.
4          𝕏′로 모델을 학습시킨다.
5          𝕏 − 𝕏′를 이용하여 학습된 모델의 성능을 측정한다.
6      T개 성능을 평균하여 해당 모델의 성능으로 취한다.
7  가장 높은 성능을 보인 모델을 선택한다.
8  테스트집합으로 선택된 모델의 성능을 측정한다.
```

모델 선택의 한계와 현실적인 해결책

- **[알고리즘 1-2, 1-3, 1-4]에서 모델 집합 Ω**
  - 오른쪽에서는 서로 다른 차수의 다항식이 Ω인 셈
  - 현실에서는 아주 다양
    - 신경망, 강화 학습, 확률 그래피컬 모델, SVM, 트리 분류기 등이 선택 대상
    - 신경망을 채택하더라도 MLP, 깊은 MLP, CNN 등 아주 많음



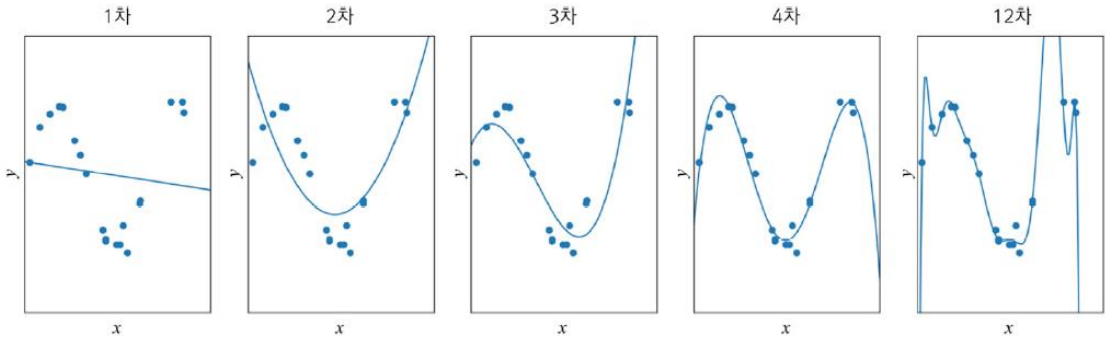
- **현실에서는 경험으로 큰 틀 선택한 후**
  - 모델 선택 알고리즘으로 세부 모델 선택하는 전략 사용
  - 예) CNN을 사용하기로 정한 후, 은닉층 개수, 활성함수, 모멘툼 계수 등을 정하는데 모델 선택 알고리즘을 적용함

모델 선택의 한계와 현실적인 해결책

- **이런 경험적인 접근방법에 대한『Deep Learning』책의 비유**

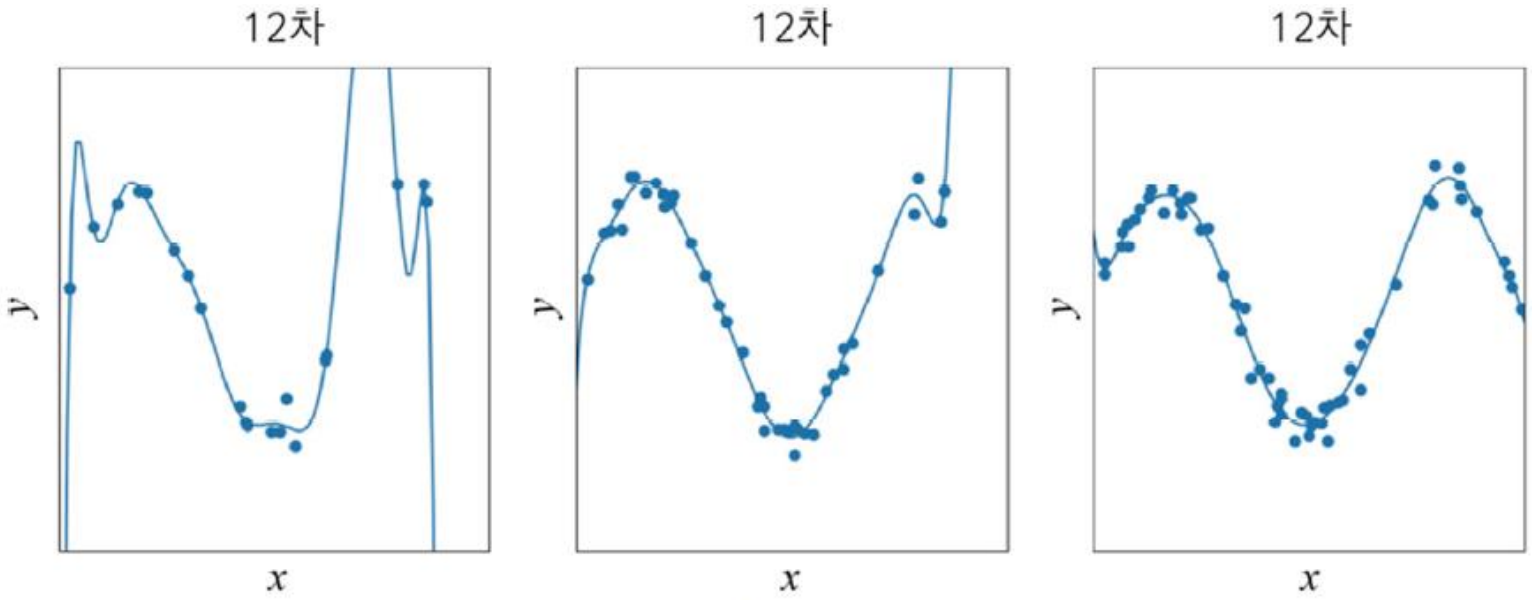
"To some extent, we are always trying to fit a square peg(the data generating process) into a round hole(our model family). 어느 정도 우리가 하는 일은 항상 둥근 홈(우리가 선택한 모델)에 네모 막대기(데이터 생성 과정)를 끼워 넣는 것이라고 말할 수 있다[Goodfellow2016(222쪽)]."

- **현대 기계 학습의 전략**
  - 용량이 충분히 큰 모델을 선택 한 후, 선택한 모델이 정상을 벗어나지 않도록 여러 가지 규제regularization 기법을 적용함
  - 예) 12차 다항식을 선택한 후 적절히 규제를 적용



데이터 확대

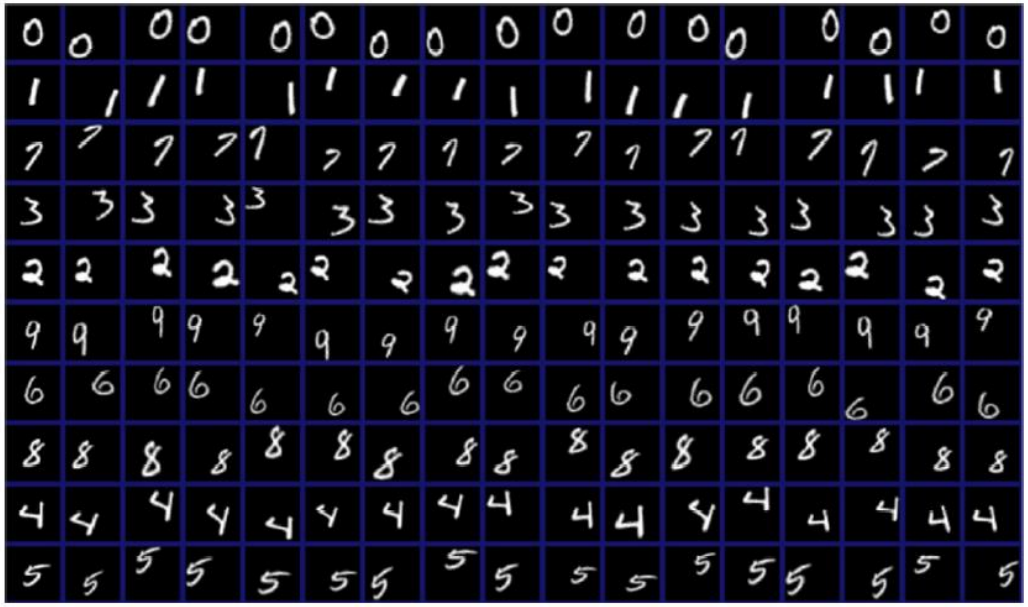
- 데이터를 더 많이 수집하면 일반화 능력이 향상됨



(a) 훈련집합의 크기 = 20 (b) 훈련집합의 크기 = 40 (c) 훈련집합의 크기 = 60

데이터 확대

- **데이터 수집은 많은 비용이 듦**
  - 그라운드 트루스 (Ground truth)를 사람이 일일이 레이블링해야 함
- **인위적으로 데이터 확대**
  - 훈련집합에 있는 샘플을 변형함
  - 약간 회전 또는 와핑 (부류 소속이 변하지 않게 주의)



가중치 감쇠

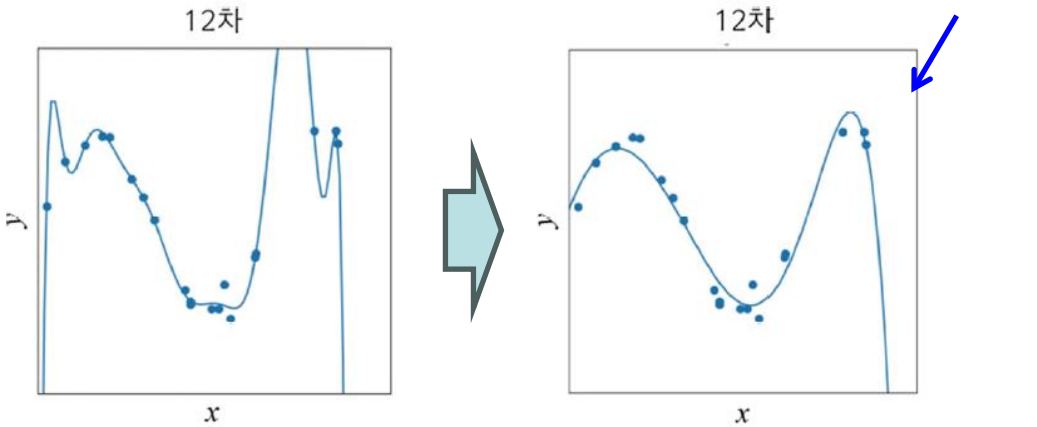
- **가중치를 작게 조절하는 기법**
  - [그림 1-18(a)]의 12차 곡선은 가중치가 매우 큼

$$y = 1005.7x^{12} - 27774.4x^{11} + \cdots - 22852612.5x^{1} - 12.8$$

  - 가중치 감쇠는 개선된 목적함수를 이용하여 가중치를 작게 조절하는 규제 기법
    - 식 (1.11)의 두 번째 항은 규제 항으로서 가중치 크기를 작게 유지해줌

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(f_{\Theta}(\mathbf{x}_i) - y_i)^2 + \lambda\|\Theta\|_2^2 \qquad (1.11)$$

$$y = 10.779x^{12} - 42.732x^{11} + \cdots - 2.379x^{1} + 0.119$$



(a) 가중치 감쇠 적용 안 함[식 (1.8)의 목적함수] (b) 가중치 감쇠 적용함[식 (1.11)의 목적함수]

지도 방식에 따른 유형

- 지도 학습
  - 특징 벡터 $\mathbb{X}$와 목푯값 $\mathbb{Y}$가 모두 주어진 상황
  - 회귀와 분류 문제로 구분
- 비지도 학습
  - 특징 벡터 $\mathbb{X}$는 주어지는데 목푯값 $\mathbb{Y}$ 가 주어지지 않는 상황
  - 군집화 과업 (고객 성향에 따른 맞춤 홍보 응용 등)
  - 밀도 추정, 특징 공간 변환 과업

지도 방식에 따른 유형

- 강화 학습
  - 목푯값이 주어지는데, 지도 학습과 다른 형태임
  - 예) 바둑
    - 수를 두는 행위가 샘플인데, 게임이 끝나면 목푯값 하나가 부여됨
      - 이기면 1, 패하면 -1을 부여
    - 게임을 구성한 샘플들 각각에 목푯값을 나누어 주어야 함

- 준지도 학습
  - 일부는 $\mathbb{X}$와 $\mathbb{Y}$를 모두 가지지만, 나머지는 $\mathbb{X}$만 가진 상황
  - 인터넷 덕분으로 $\mathbb{X}$의 수집은 쉽지만, $\mathbb{Y}$는 수작업이 필요하여 최근 중요성 부각

다양한 기준에 따른 유형

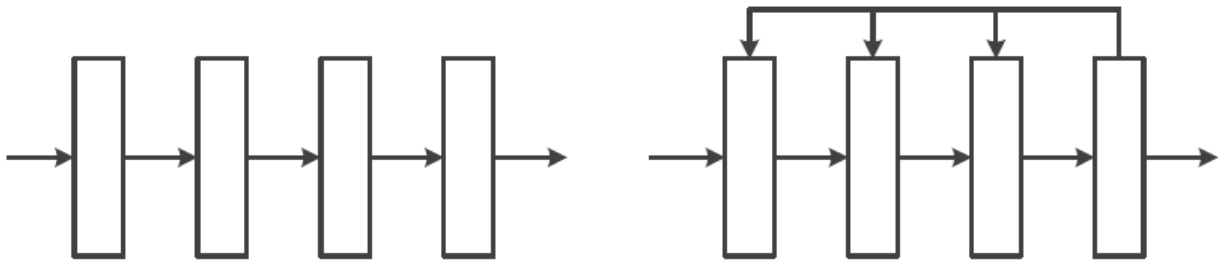
- **오프라인 학습과 온라인 학습**
  - 온라인 학습은 인터넷 등에서 추가로 발생하는 샘플을 가지고 점증적 학습
- **결정론적 학습과 스토캐스틱 학습**
  - 결정론적에서는 같은 데이터를 가지고 다시 학습하면 같은 예측기가 만들어짐
  - 스토캐스틱 학습은 학습 과정에서 난수를 사용하므로 같은 데이터로 다시 학습하면 다른 예측기가 만들어짐. 보통 예측 과정도 난수 사용
- **분별 모델과 생성 모델**
  - 분별 모델은 부류 예측에만 관심. 즉 $P(y|x)$의 추정에 관심
  - 생성 모델은 $P(x)$ 또는 $P(x|y)$를 추정함
    - 따라서 새로운 샘플을 '생성'할 수 있음

신경망 (Neural Network)

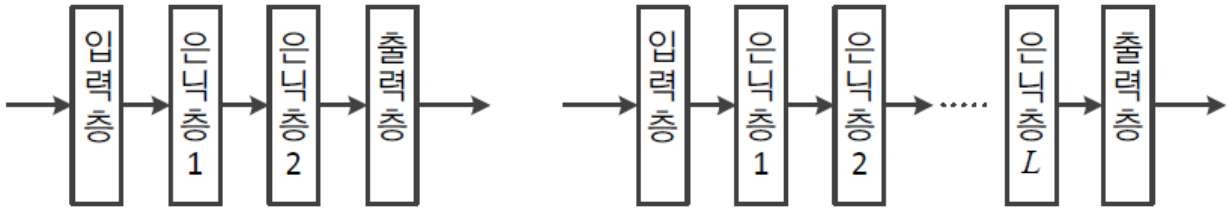
- **신경망**
  - 기계 학습 역사에서 가장 오래된 기계 학습 모델이며, 현재 가장 다양한 형태를 가짐
  - 1950년대 퍼셉트론 → 1980년대 다층 퍼셉트론
  - 3장은 4장 딥러닝의 기초가 됨

신경망

- **신경망에는 아주 다양한 모델이 존재함**
  - 전방 신경망과 순환 신경망
  - 얕은 신경망과 깊은 신경망
  - 결정론 신경망과 스토캐스틱 신경망



(a) 전방 신경망과 순환 신경망



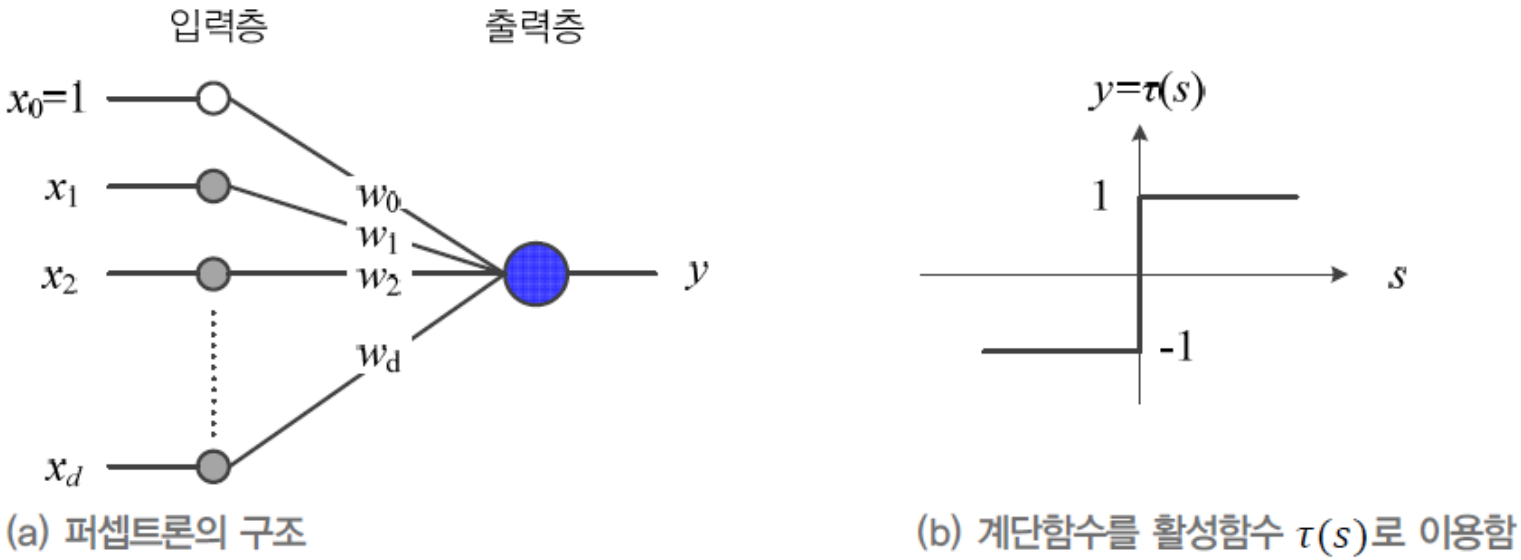
(b) 얕은 신경망과 깊은 신경망

신경망의 종류

퍼셉트론 (Perceptron)

퍼셉트론의 구조

- 입력층과 출력층을 가짐
  - 입력층은 연산을 하지 않으므로 퍼셉트론은 단일 층 구조라고 간주
- 입력층의 i번째 노드는 특징 벡터 $\mathbf{x} = (x_1, x_2, \cdots, x_d)^{\mathrm{T}}$의 요소 x_i를 담당
- 항상 1이 입력되는 바이어스 노드
- 출력층은 한 개의 노드
- i번째 입력층 노드와 출력층을 연결하는 에지는 가중치 w_i를 가짐



(a) 퍼셉트론의 구조

(b) 계단함수를 활성함수 $\tau(s)$로 이용함

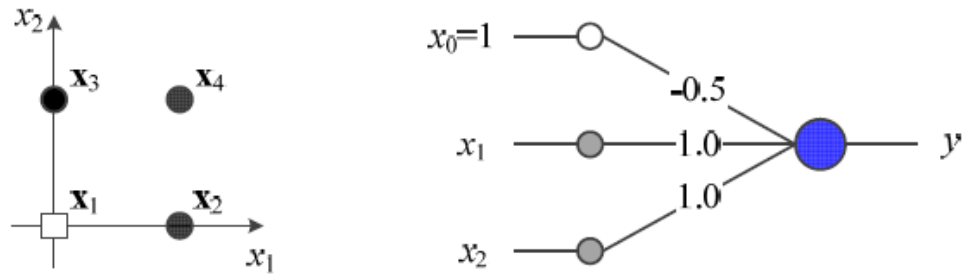
퍼셉트론의 구조와 동작

퍼셉트론

<예제> 퍼셉트론의 동작

2차원 특징 벡터로 표현되는 샘플을 4개 가진 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, $\mathbb{Y} = \{y_1, y_2, y_3, y_4\}$를 생각하자. [그림 3-4(a)]는 이 데이터를 보여준다.

$$\mathbf{x}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, y_1 = -1, \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, y_2 = 1, \quad \mathbf{x}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, y_3 = 1, \quad \mathbf{x}_4 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, y_4 = 1$$



(a) 훈련집합 (b) 퍼셉트론

OR 논리 게이트를 이용한 퍼셉트론의 동작 예시

샘플 4개를 하나씩 입력하여 제대로 분류하는지 확인해 보자.

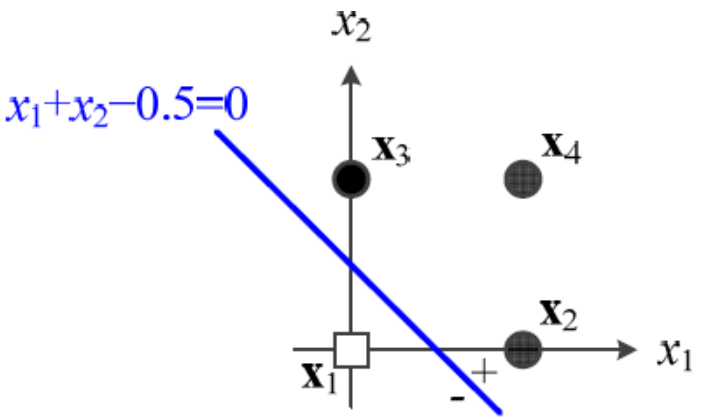
$$\begin{aligned}
\mathbf{x}_1&: s = -0.5 + 0 * 1.0 + 0 * 1.0 = -0.5, && \tau(-0.5) = -1 \\
\mathbf{x}_2&: s = -0.5 + 1 * 1.0 + 0 * 1.0 = 0.5, && \tau(0.5) = 1 \\
\mathbf{x}_3&: s = -0.5 + 0 * 1.0 + 1 * 1.0 = 0.5, && \tau(0.5) = 1 \\
\mathbf{x}_4&: s = -0.5 + 1 * 1.0 + 1 * 1.0 = 1.5, && \tau(1.5) = 1
\end{aligned}$$

결국 [그림 3-4(b)]의 퍼셉트론은 샘플 4개를 모두 맞추었다. 이 퍼셉트론은 훈련집합을 100% 성능으로 분류한다고 말할 수 있다.

퍼셉트론

위 예제를 기하학적으로 설명하면,

- 결정 직선 $d(\mathbf{x}) = d(x_1, x_2) = w_1 x_1 + w_2 x_2 + w_0 = 0 \cdot x_2 - 0.5 = 0$
  - w_1과 w_2는 직선의 방향, w_0은 절편을 결정
  - 결정 직선은 전체 공간을 +1과 -1의 두 부분공간으로 분할하는 분류기 역할



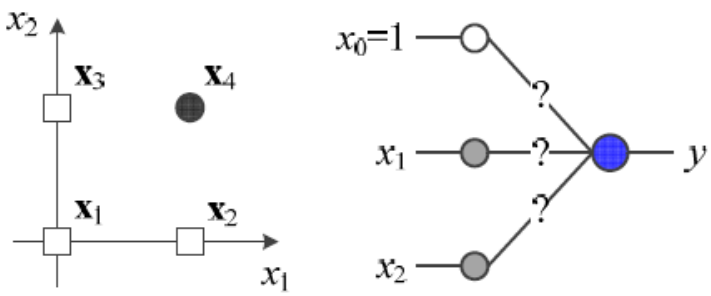
<예제> 의 퍼셉트론에 해당하는 결정 직선

- d차원 공간에서는 $d(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + w_0 = 0$
  - 2차원은 결정 직선, 3차원은 결정 평명, 4차원 이상은 결정 초평면

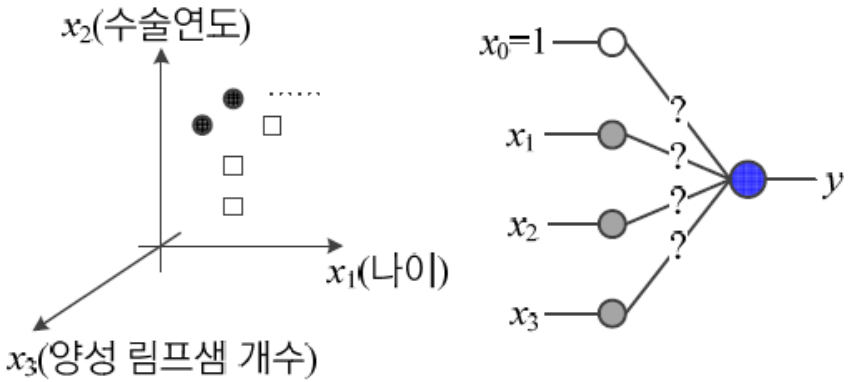
학습

학습 문제

- 지금까지는 학습을 마친 퍼셉트론을 가지고 동작을 설명한 셈
- 학습 문제: 가중치 w_1과 w_2, w_0이 어떤 값을 가져야 100% 옳게 분류할까?
- 2차원 공간에 4개 샘플이 있는 훈련집합이지만, 현실 세계는 d차원 공간에 수백~수만 개의 샘플이 존재



(a) AND 분류 문제



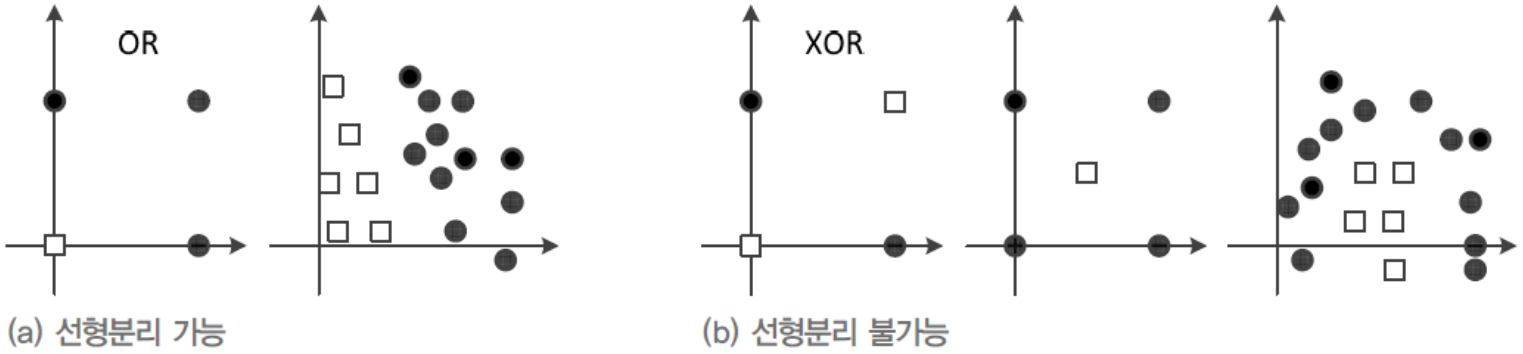
(b) Haberman survival 분류 문제

학습은 반대 방향으로 일어남. 출력과 정답의 차이(손실)를 계산한 뒤, 역전파(backpropagation)로 각 가중치가 오차에 얼마나 기여했는지 미분으로 구하고, 그 방향의 반대로 가중치를 조금씩 수정함 (경사하강법). 이걸 수만 번 반복 (epoch) 하는 것이 "학습"입니다.

다중 퍼셉트론

퍼셉트론은 선형 분류기라는 한계

- 선형 분리 불가능한 상황에서는 일정한 양의 오류
- 예) XOR 문제에서는 75%가 정확률 한계

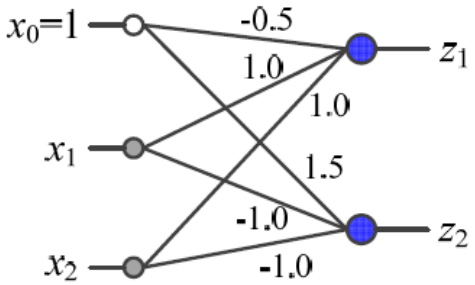


선형분리가 가능한 상황과 불가능한 상황

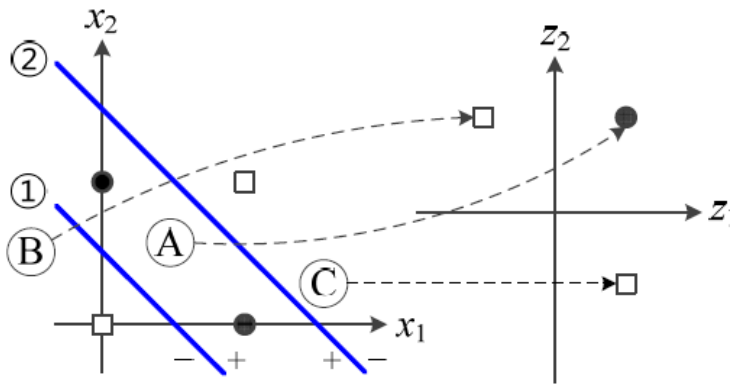
- 민스키의『Perceptrons』
  - 퍼셉트론의 한계를 지적하고 다층 구조를 이용한 극복 방안 제시. 당시 기술로 실현 불가능
  - 1974년 웨어보스는 박사 논문에서 오류 역전파 알고리즘 제안
  - 1986년 루멜하트의 저서『Parallel Distributed Processing』다층 퍼셉트론 이론 정립하여 신경망 부활

공간 변환

- **퍼셉트론 2개를 병렬로 결합하면,**
  - 원래 공간 $\mathbf{x} = (x_1, x_2)^{\mathrm{T}}$를 새로운 특징 공간 $\mathbf{z} = (z_1, z_2)^{\mathrm{T}}$로 변환
  - 새로운 특징 공간 $\mathbf{z}$에서는 선형 분리 가능함



(a) 두 퍼셉트론을 병렬로 결합



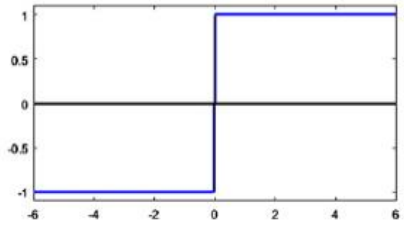
(b) 원래 특징 공간 x를 새로운 특징 공간 z로 변환

특징 공간의 변환

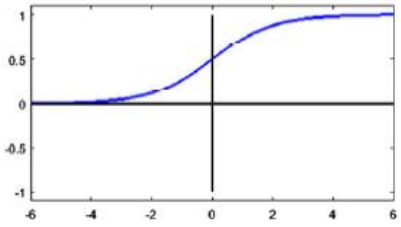
활성 함수

딱딱한 공간 분할과 부드러운 공간 분할

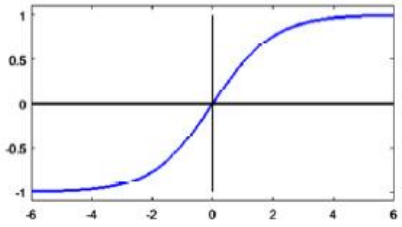
- 계단함수는 딱딱한 의사결정(영역을 점으로 변환). 나머지 활성함수는 부드러운 의사결정(영역을 영역으로 변환)



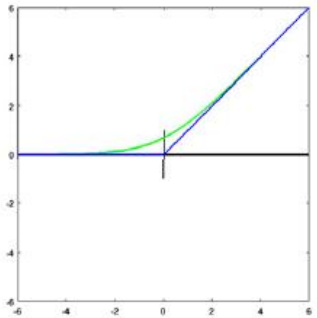
(a) 계단 함수



(b) 로지스틱 시그모이드

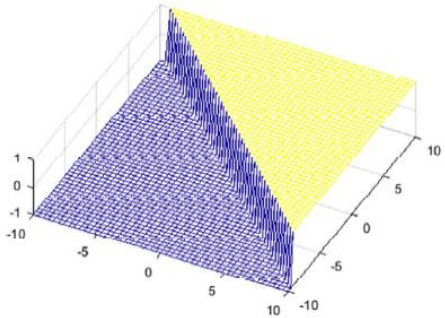


(c) 하이퍼볼릭 탄젠트 시그모이드

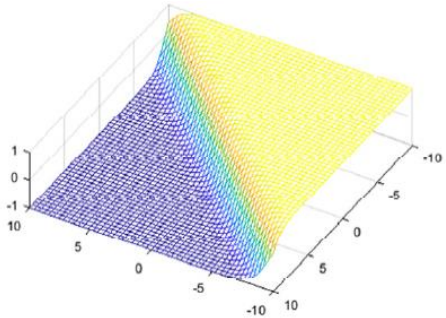


(d) softplus와 rectifier

신경망이 사용하는 활성함수



(a) 계단함수의 딱딱한 공간 분할



(b) 로지스틱 시그모이드의 부드러운 공간 분할

활성 함수

신경망이 사용하는 다양한 활성함수

- 로지스틱 시그모이드와 하이퍼볼릭 탄젠트는 a가 커질수록 계단함수에 가까워짐
- 모두 1차 도함수 계산이 빠름 (특히 ReLU는 비교 연산 한 번)

표 3-1 활성함수로 사용되는 여러 함수

함수 이름	함수	1차 도함수	범위
계단	$\tau(s)=\begin{cases}1 & s\geq 0\\ -1 & s<0\end{cases}$	$\tau'(s)=\begin{cases}0 & s\neq 0\\ \text{불가} & s=0\end{cases}$	-1과 1
로지스틱 시그모이드	$\tau(s)=\frac{1}{1+e^{-as}}$	$\tau'(s)=a\tau(s)(1-\tau(s))$	(0,1)
하이퍼볼릭 탄젠트	$\tau(s)=\frac{2}{1+e^{-as}}-1$	$\tau'(s)=\frac{a}{2}(1-\tau(s)^2)$	(-1,1)
소프트플러스	$\tau(s)=\log_e(1+e^s)$	$\tau'(s)=\frac{1}{1+e^{-s}}$	(0, ∞)
렉티파이어(ReLU)	$\tau(s)=\max(0,s)$	$\tau'(s)=\begin{cases}0 & s<0\\ 1 & s>0\\ \text{불가} & s=0\end{cases}$	[0, ∞)

딥러닝 (Deep Learning)

딥러닝

- 다층 퍼셉트론에 은닉층을 여러 개 추가하면 깊은 신경망이 됨
- 딥러닝은 깊은 신경망을 학습시키는 알고리즘
- 딥러닝은 새로운 응용을 창출하고 인공지능 제품의 성능을 획기적으로 향상 → 현대 기계 학습을 주도

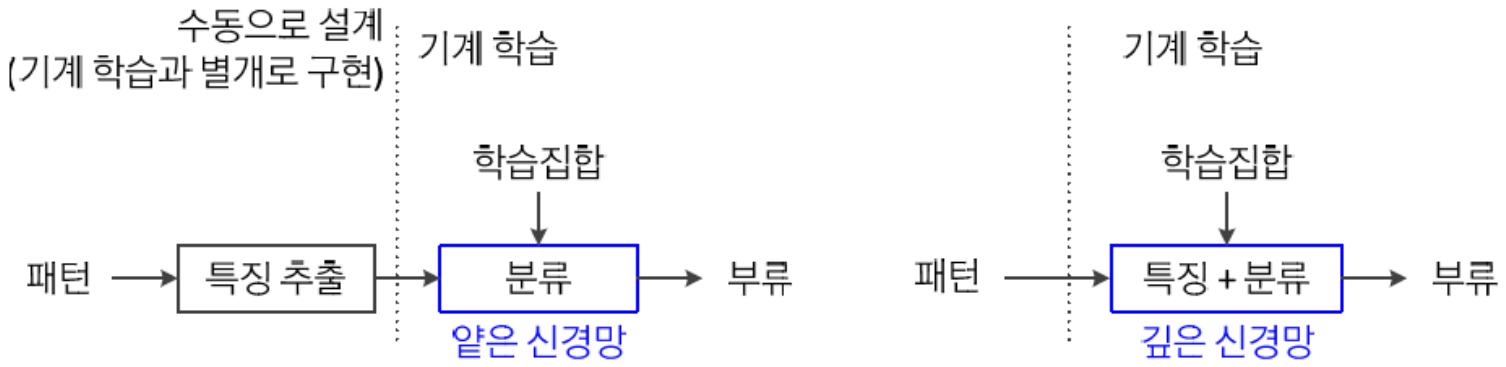
등장 요인

- 컨볼루션 신경망이 딥러닝의 가능성을 엶
- 값싼 GPU의 등장
- 인터넷 덕분으로 학습 데이터가 크게 늘어남
- 계산은 단순한데 성능은 더 좋은 활성함수
- 과잉적합을 방지하는데 효과적인 다양한 규제 기법
- 층별 예비학습 기법 개발

특징 (Feature) 학습 부각

기계 학습의 패러다임의 변화

- 고전적인 다층 퍼셉트론
  - 은닉층은 특징 추출기
  - 얕은 구조이므로 센서로 획득한 원래 패턴을 그대로 입력하면 낮은 성능
  - 따라서 사람이 수작업 특징을 구상하고 구현하여 신경망에 입력함
- 현대 기계 학습 (딥러닝)
  - 특징 추출을 학습으로 설계 ← 특징 학습
  - 원래 패턴이 신경망의 입력 ← 통째 학습end-to-end learning



(a) 고전적 패러다임에서의 수작업 특징

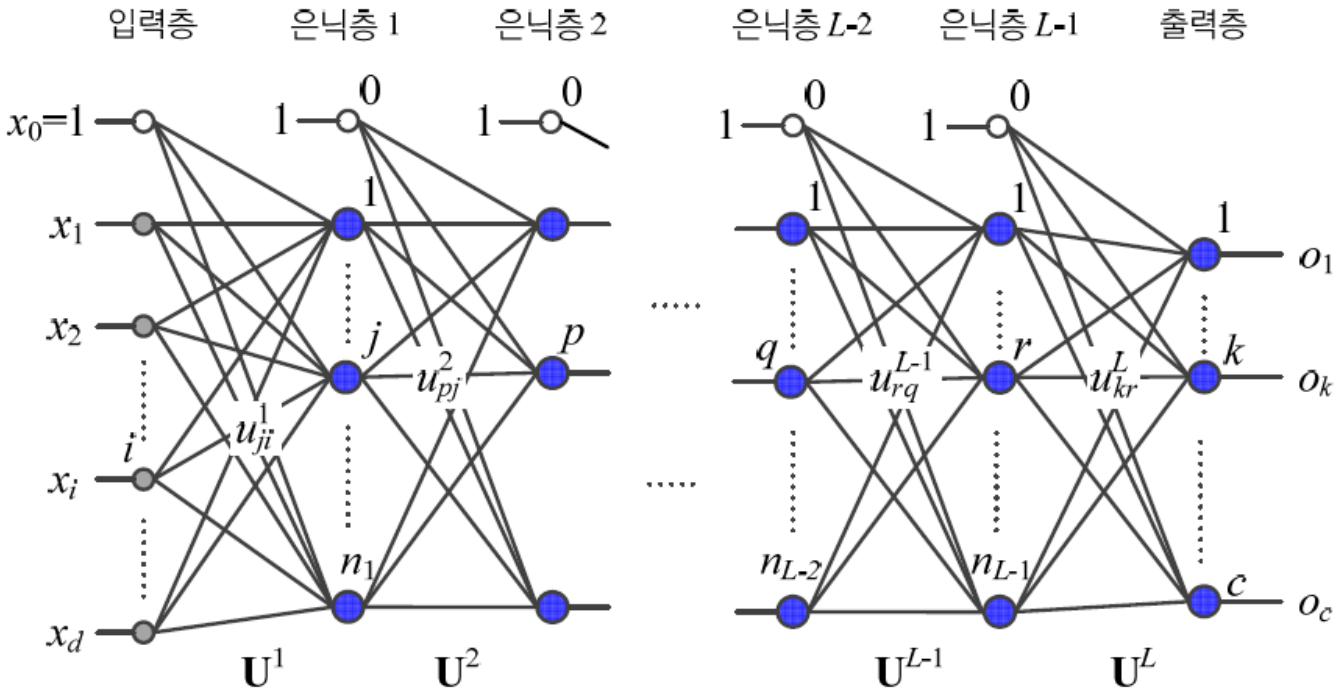
(b) 딥러닝에서의 특징 학습

기계 학습의 패러다임 변화

구조 및 동작

깊은 MLP(DMLP, deep MLP)의 구조

- 입력층(d+1개의 노드)과 출력층(c개의 노드)
- L-1개의 은닉층 (입력층은 0번째 은닉층, 출력층은 L번째 은닉층으로 간주)
  - l번째 은닉층의 노드 수를 n_l로 표기

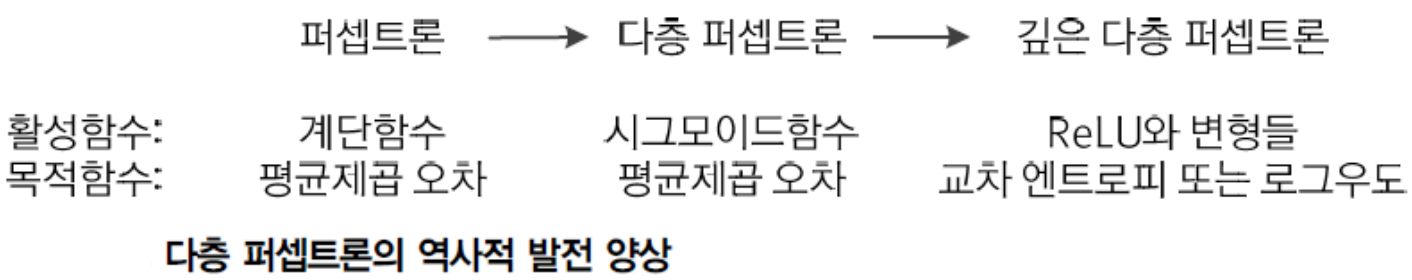


깊은 MLP(DMLP)의 구조

학습

역사적 고찰

- 학습 알고리즘의 주요 개선



다층 퍼셉트론의 역사적 발전 양상

- CNN의 부상
  - 예) MNIST 인식 경쟁
    - 2010년 784-2500-2000-1500-1000-500-10 구조의 DMLP 0.35% 오류율
    - 2011년 CNN 0.35% 오류율
    - 2012년 35개 CNN을 이용한 앙상블 모델 0.23% 오류율 [Ciresan2012]
  - 예) ILSVRC 자연영상 인식 경쟁: CNN이 DMLP보다 확연히 우월

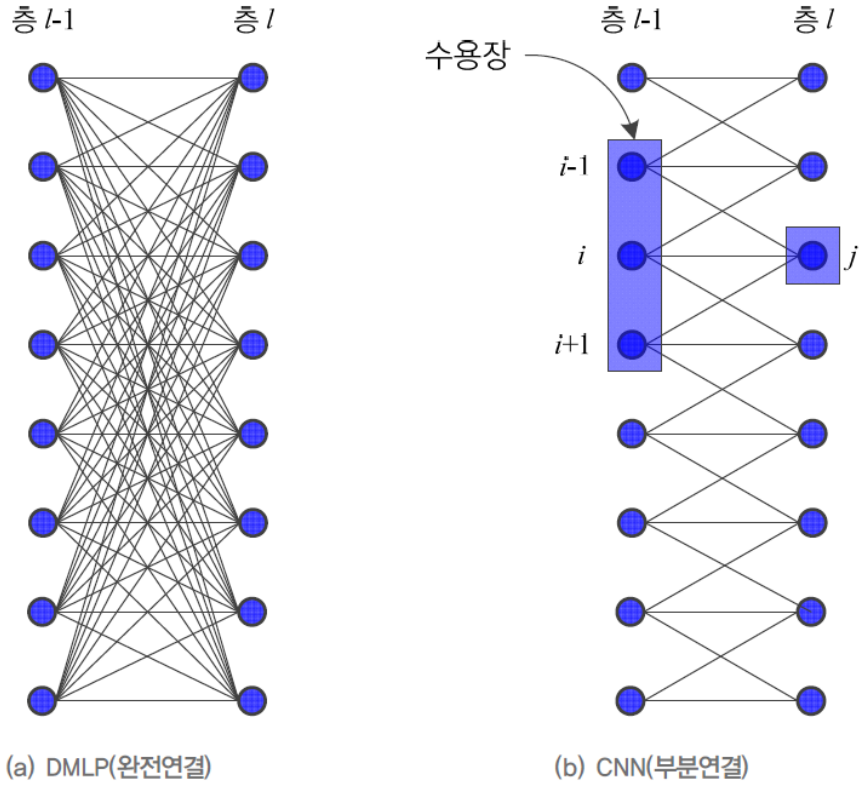
컨볼루션 신경망 (CNN)

DMLP와 CNN의 비교

- DMLP
  - 완전연결 구조로 높은 복잡도
  - 학습이 매우 느리고 과잉적합 우려
- CNN
  - 컨볼루션 연산을 이용한 부분연결(희소 연결) 구조로 복잡도 크게 낮춤
  - 컨볼루션 연산은 좋은 특징 추출

CNN

- 격자 구조(영상, 음성 등)를 갖는 데이터에 적합
- 수용장은receptive field 인간시각과 유사
- 가변 크기의 입력 처리 가능



CNN의 부분연결과 수용장

컨볼루션 신경망 (CNN)

컨볼루션 연산

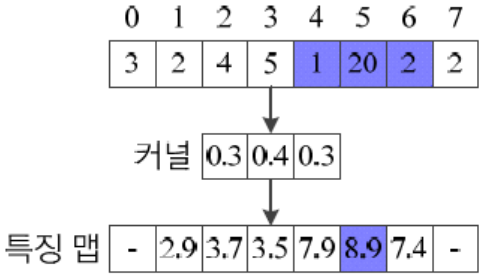
- 컨볼루션은 해당하는 요소끼리 곱하고 결과를 모두 더하는 선형 연산
- 식 (4.10)과 식 (4.11)에서 u는 커널, z는 입력, s는 출력(특징 맵)

$$s(i) = z \circledast u = \sum_{x=-(h-1)/2}^{(h-1)/2} z(i+x)u(x) \qquad (4.10)$$

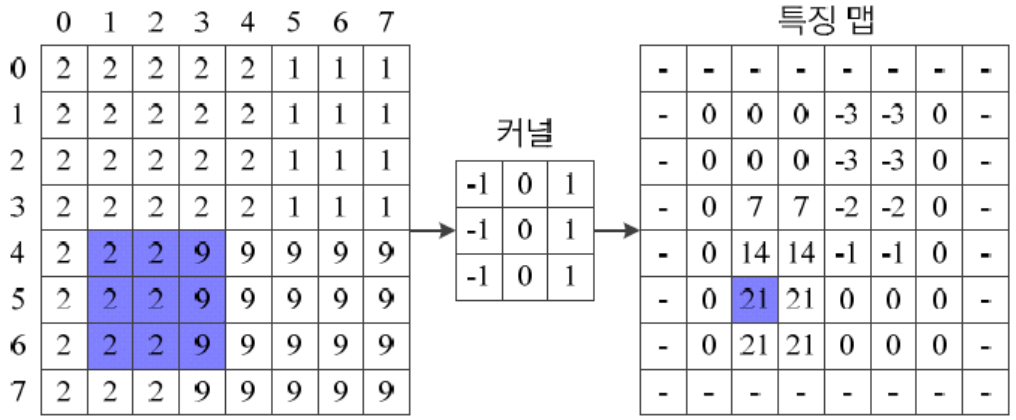
← 1차원 입력

$$s(j,i) = z \circledast u = \sum_{y=-(h-1)/2}^{(h-1)/2} \sum_{x=-(h-1)/2}^{(h-1)/2} z(j+y, i+x)u(y,x) \qquad (4.11)$$

← 2차원 입력



(a) 1차원 컨볼루션

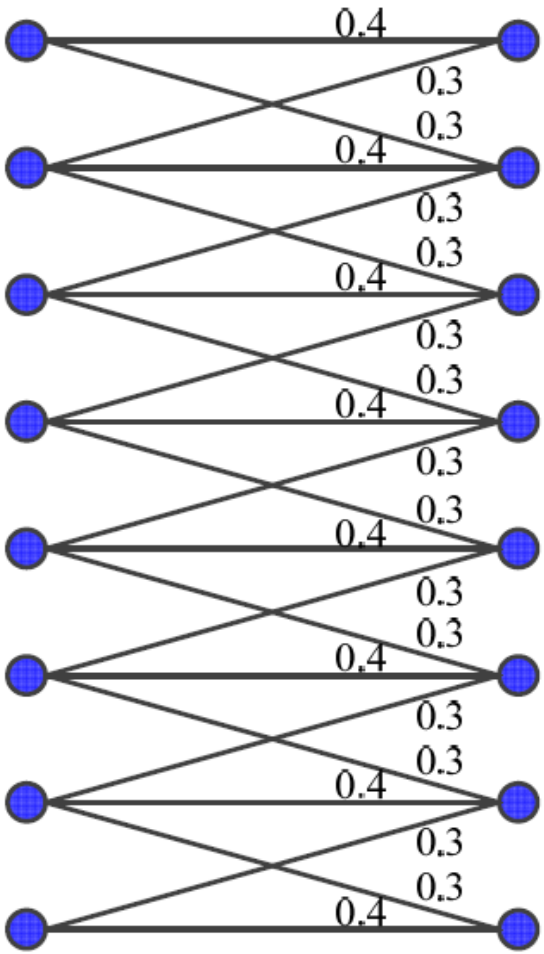


(b) 2차원 컨볼루션

컨볼루션층

가중치 공유(묶인 가중치)

- 모든 노드가 동일한 커널을 사용(즉 가중치를 공유)하므로 매개변수는 3개에 불과
- 모델의 복잡도가 크게 낮아짐



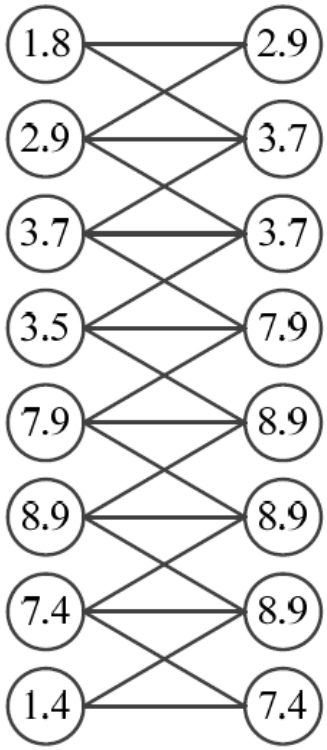
특징 학습

- 커널을 사람이 설계하지 않고, 학습으로 알아냄
  - u_i^k는 k번째 커널의 i번째 매개변수
- 예) 2차원 영상이 7*7 커널을 64개 사용한다면, 학습은 (7*7+1)*64=3200개의 매개변수를 찾아내야 함
- DMLP와 마찬가지로 오류 역전파로 커널을 학습함

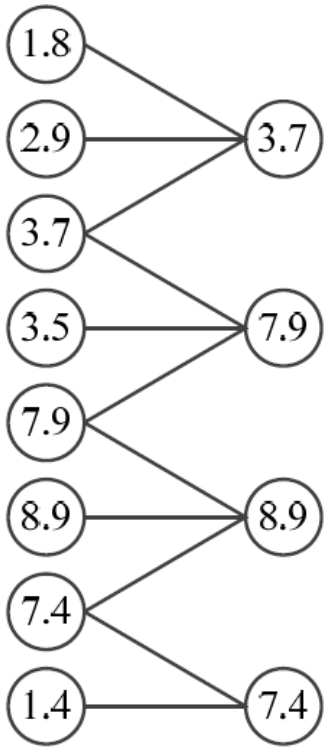
컨볼루션층

풀링 연산

- 아래 그림은 최대 풀링 (Max pooling) 예
- 최대 풀링, 평균 풀링, 가중치 평균 풀링 등
- 보폭을 크게 하면 다운샘플링 효과



(a) 보폭 1



(b) 보폭 2

얻는 효과 세 가지

1. 계산량과 메모리 절감 — 크기가 1/4로 줄면 다음 합성곱층이 처리할 데이터도 1/4이 됨. 합성곱+풀링을 반복할수록 이미지는 작아지고, 대신 필터 수를 늘려 "공간은 좁게, 의미는 깊게" 압축해 감.

2. 위치 변화에 대한 둔감함 (translation invariance) — 고양이 귀가 2~3픽셀 옆으로 이동해도, 같은 2×2 구역 안이라면 최댓값은 그대로 살아남아 출력이 변하지 않음. "귀가 정확히 (137, 52) 픽셀에 있다"가 아니라 "이 근방에 귀가 있다"만 기억하는 셈이라, 물체가 조금 움직이거나 틀어져도 인식이 유지됨.

3. 과적합 억제 — 세부 위치 정보를 강제로 버리기 때문에, 학습 이미지의 픽셀 단위 배치를 통째로 외우기 어려워짐.

컨볼루션 신경망 사례 연구

자연영상 분류라는 도전적 문제

- ImageNet 데이터베이스
  - 2만 부류에 대해 부류별로 500~1000장의 영상을 인터넷에서 수집하여 구축하고 공개
- ILSVRC 대회 (CVPR 학술대회에서 개최)
  - 1000부류에 대해 분류, 검출, 위치 지정 문제: 1순위와 5순위 오류율로 대결
  - 120만 장의 훈련집합, 5만 장의 검증집합, 15만 장의 테스트집합
  - 우승: AlexNet(2012) → Clarifi팀(2013) → GoogLeNet&VGGNet(2014) → ResNet(2015)
- 우승한 CNN은 프로그램과 가중치를 공개함으로써 널리 사용되는 표준 신경망이 됨

(a) 'swing' 부류

(b) 'Great white shark' 부류

TinyML & 엣지 AI 개요

마이크로컨트롤러에서 직접 머신러닝을 실행하는 TinyML의 개념과 필요성 이해

클라우드 AI vs 엣지(온디바이스) AI

■ 수집한 센서 신호를 **어디서 처리하느냐**가 시스템 설계의 핵심

클라우드 AI

- 데이터를 서버로 전송해 처리
- **강력한 연산 · 큰 모델**
- 네트워크 필요 · 지연 발생
- 전송 비용 · 프라이버시 우려

엣지 · 온디바이스 AI

- **기기 안에서 바로 추론**
- 실시간 · 저지연 응답
- **오프라인 동작 · 저전력**
- 데이터가 기기 밖으로 안 나감

TinyML이란? — 왜 온디바이스인가

■ **TinyML** = 수 mW 전력·수백 KB 메모리의 마이크로컨트롤러에서 돌아가는 경량 머신러닝

- **정의 :** 배터리로 동작하는 초소형 기기에서 딥러닝 모델을 직접 실행하는 기술
  - 클라우드 왕복 없이 센서 옆에서 즉시 판단 → 실시간·저전력·프라이버시
- **제약 조건 :** 메모리(RAM/Flash)·연산·전력이 매우 제한적
  - 모델을 작게 → 양자화(int8), 경량 신경망, 효율적 특징추출이 필수
- **대표 응용 :** 키워드 인식 · 동작/제스처 인식 · 이상 진동 감지 · 낙상 감지 · 예지보전
- 이 과정에서는 **Edge Impulse** 로 코딩을 최소화하며 전체 파이프라인을 체험

Arduino Nano 33 BLE

■ 한 손톱만 한 보드에 **고성능 MCU + 다양한 센서 + BLE** 가 모두 내장

- **프로세서 :** Nordic nRF52840 — ARM Cortex-M4F @ 64MHz (부동소수점 연산용 FPU 내장)
  - 메모리 : 1MB Flash · 256KB RAM → 작은 신경망을 올리기에 충분
- **무선 :** Bluetooth 5.0 LE(BLE) 내장 → 추론 결과를 스마트폰·PC로 전송 가능
- **전원 :** USB 또는 코인셀·리튬 배터리, 저전력 동작(수 mA)
- **개발 :** Arduino IDE / CLI 로 프로그래밍, Edge Impulse가 이 보드를 공식 지원
- **참고 :** Rev2는 IMU·온습도 칩이 일부 변경(BMI270 등) — Edge Impulse는 두 버전 모두 지원

온보드 센서 & 개발 환경

■ 코딩 없이도 **여러 센서 데이터를 바로 수집** 할 수 있어 데이터셋 만들기가 쉬움

- **IMU(관성센서) :** 3축 가속도 + 3축 자이로 + 3축 지자기 → **이번 실습의 주인공(제스처/동작)**
- **마이크(PDM) :** 소리 → 키워드 인식·소리 분류에 사용
- **환경 센서 :** 온도·습도, 기압 → 환경 모니터링 응용
- **광학 센서(APDS9960) :** 제스처·근접·색·조도 감지
- **개발 환경 준비물 :** Arduino IDE(또는 CLI), Edge Impulse 계정, USB 케이블, 크롬 브라우저
  - Edge Impulse CLI(edge-impulse-daemon)로 보드를 스튜디오에 바로 연결

Edge Impulse란?

■ **Edge Impulse** = 임베디드 ML을 위한 웹 기반 개발 플랫폼(브라우저에서 대부분 처리)

▪ 코드를 거의 안 써도 **데이터 수집 → 학습 → 최적화 → 배포** 전 과정을 GUI로 진행가능
전문가 모드 등을 활용하여 코드 기반의 복잡한 모델 적용 가능

- **Studio(웹) :** 데이터 라벨링, 특징추출(DSP) 설계, 신경망 학습, 성능 확인을 한 곳에서
- **자동 최적화 :** EON Tuner가 MCU 자원(메모리·지연)에 맞는 모델 구성을 자동 탐색
- **배포 :** 학습된 모델을 Arduino 라이브러리 등으로 내보내 기기에 바로 탑재
- 무료 개발자 계정으로 학습·실습 가능

전체 워크플로우

■ 센서 데이터를 모아 학습하고, 작은 모델로 만들어 보드에 올리는 **5단계** 흐름

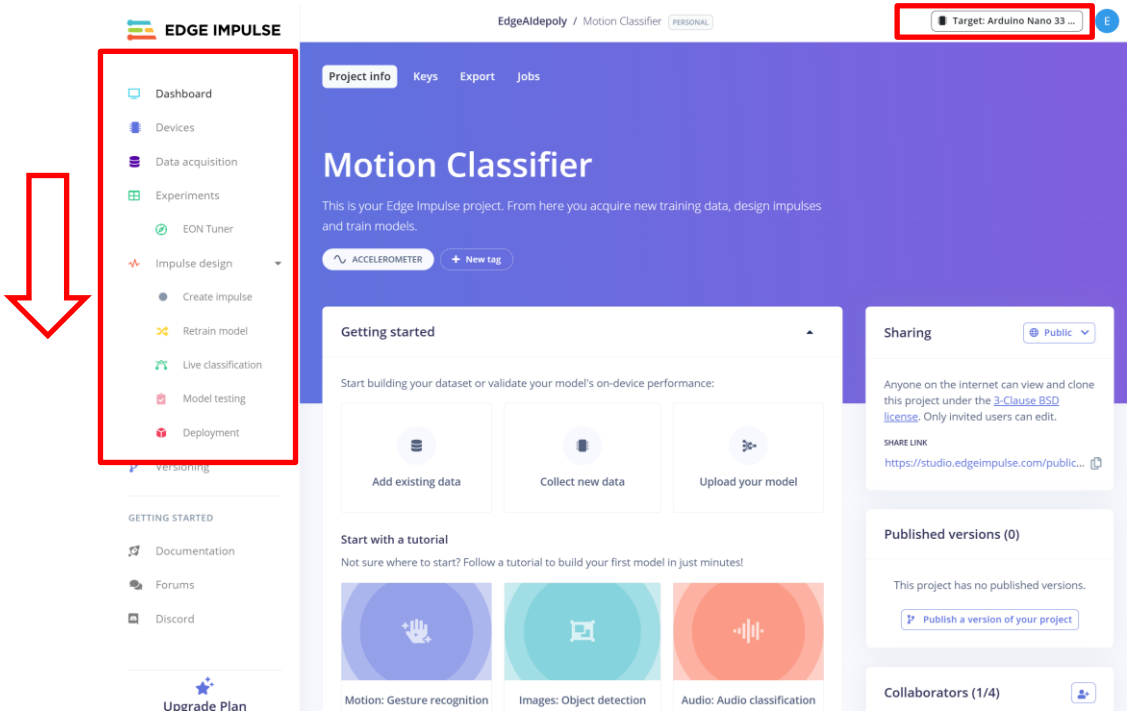
① 데이터 수집
보드→Studio

② 모델 설계
특징추출+NN

③ 학습
Train model

④ 테스트
정확도 검증

⑤ 배포
Arduino로



실습 준비

■ **실습 목표 :** 보드의 가속도 센서로 **손동작(제스처)을 분류**하는 모델 만들기

- **준비물 :** Nano 33 BLE Sense, USB 케이블, PC(크롬), 인터넷
- ① Edge Impulse 계정 생성 후 **새 프로젝트(New project)** 만들기
- ② PC에 **Arduino CLI** 와 **Edge Impulse CLI** 설치(Node.js 기반)
- ③ 보드에 Edge Impulse **공식 펌웨어** 를 한 번 굽기(flash) — 데이터 수집용
- **분류할 클래스(라벨) 예 :** idle(정지) · wave(좌우 흔들기) · circle(원) · updown(위아래)

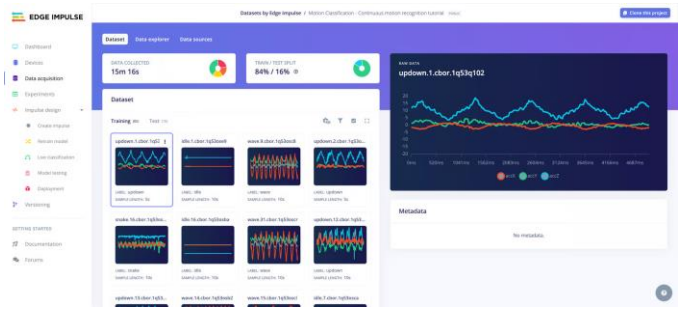
좋은 데이터셋 만들기

■ 모델 성능의 8할은 **데이터 품질** — 아래 원칙을 지키며 수집

- **클래스당 충분히 :** 각 동작을 여러 번, 최소 2~3분 분량씩(많을수록 좋음)
- **다양하게 :** 속도·세기·방향을 조금씩 다르게, 여러 사람이 수집하면 일반화↑
- **정지(idle) 클래스 필수 :** 아무 동작도 아닐 때를 반드시 포함해야 오작동↓
- **Train / Test 분할 :** 보통 **80% 학습 / 20% 테스트** 자동 분할 사용
- **라벨 일관성 :** 같은 동작은 같은 라벨로, 애매한 샘플은 제거
  - Studio가 클래스별 데이터 균형을 그래프로 보여줌 → 치우치면 보완

4가지 동작 : 5-10 s의 동작별 data file 101개, 학습과 test용으로 분할함.

- **Idle** - just sitting on your desk while you're working.
- **Snake** - moving the device over your desk as a snake.
- **Wave** - waving the device from left to right.
- **Updown** - moving the device up and down.

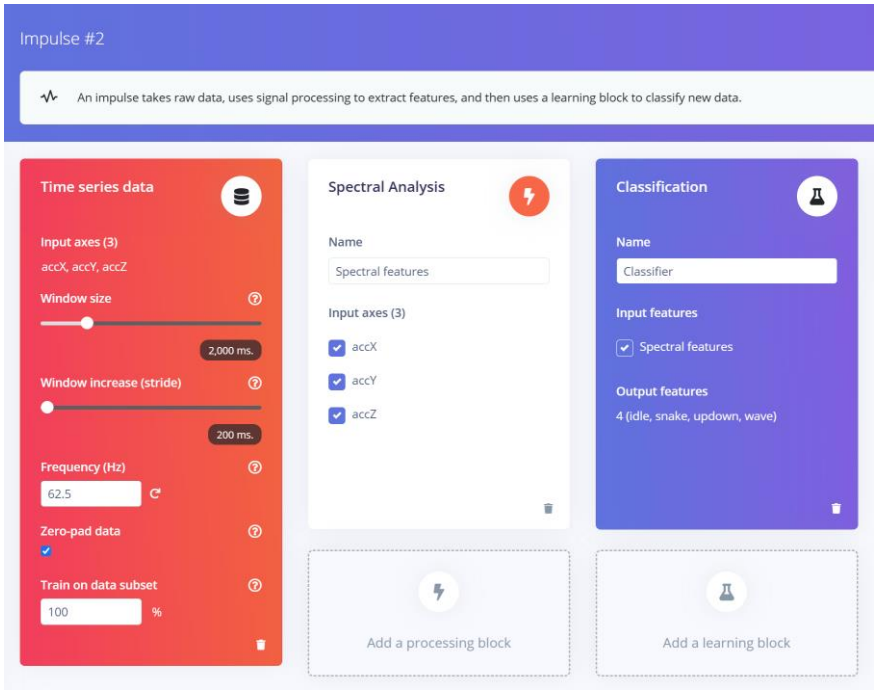


Preset data 활용

Impulse 설계 & 특징 추출(DSP)

■ **Impulse** = 원시 신호를 처리해 학습에 쓰는 '처리 파이프라인' — 3개 블록으로 구성

- ① **입력 블록 :** 시계열 센서 데이터를 일정 길이 창(window)으로 잘라 입력(예: 2초, 62.5Hz)
- ② **처리 블록(DSP) :** 원시 신호에서 **특징(feature)** 추출 → 학습이 쉬워지고 모델이 작아짐
  - 동작/제스처 → **Spectral Analysis** (주파수·통계 특징) / 소리 → MFCC·MFE
- ③ **학습 블록 :** 추출한 특징으로 **신경망(분류기)** 학습 → 클래스 출력
- ④ **Generate features** 실행 → 특징 공간에서 **클래스가 잘 뭉치는지** 확인
  - 클래스별 점들이 서로 잘 분리돼 보이면 학습이 잘 될 신호



전처리 과정 (DSP)

■ 같은 신호를 **다른 관점**에서 보는 것
예) 시계열 data -> 주파수 변환, 필터링 기능 등

시간 영역 (Time domain)

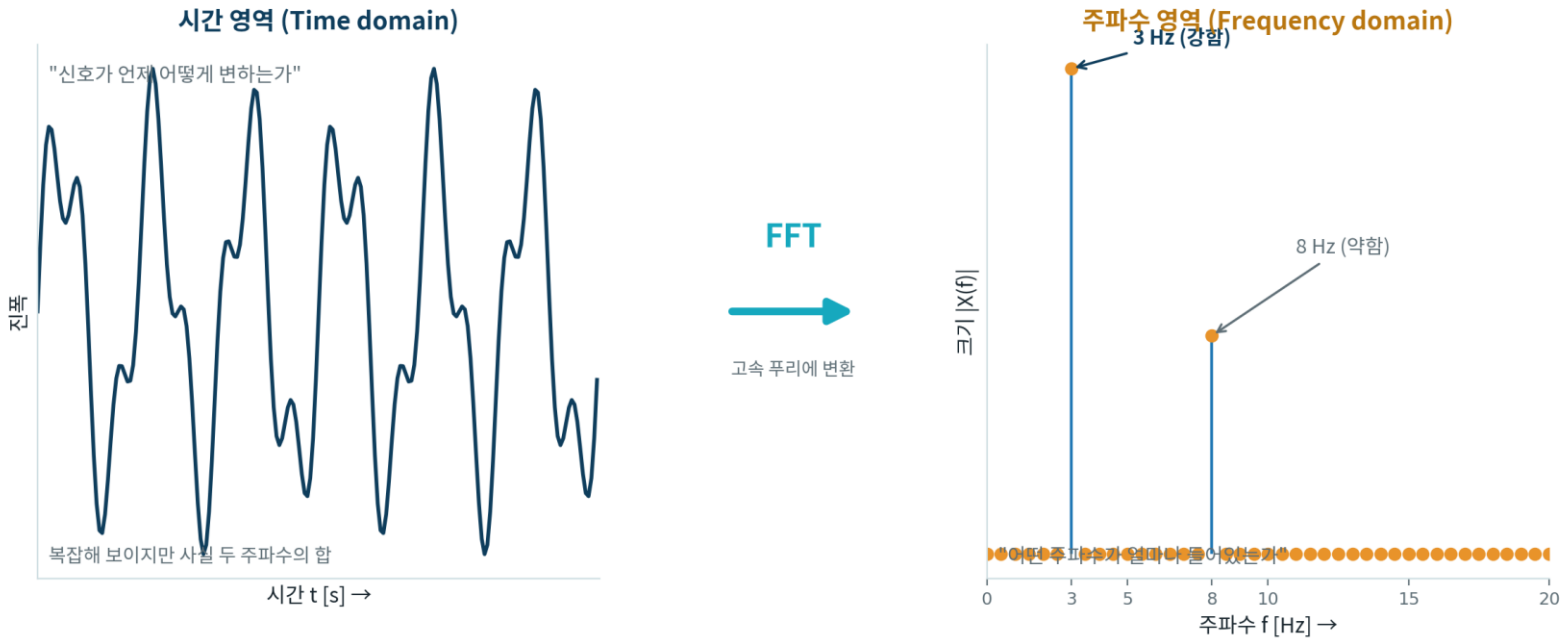
- **가로축 = 시간**
- 언제 얼마나 흔들렸는가
- 파형이 복잡하면 해석 어려움
- 진폭·평균·분산 등 통계 특징

주파수 영역 (Frequency domain)

- **가로축 = 주파수**
- 어떤 빠르기 성분이 얼마나 있는가
- **반복 패턴이 뚜렷한 피크로 나타남**
- 동작의 '리듬'을 포착

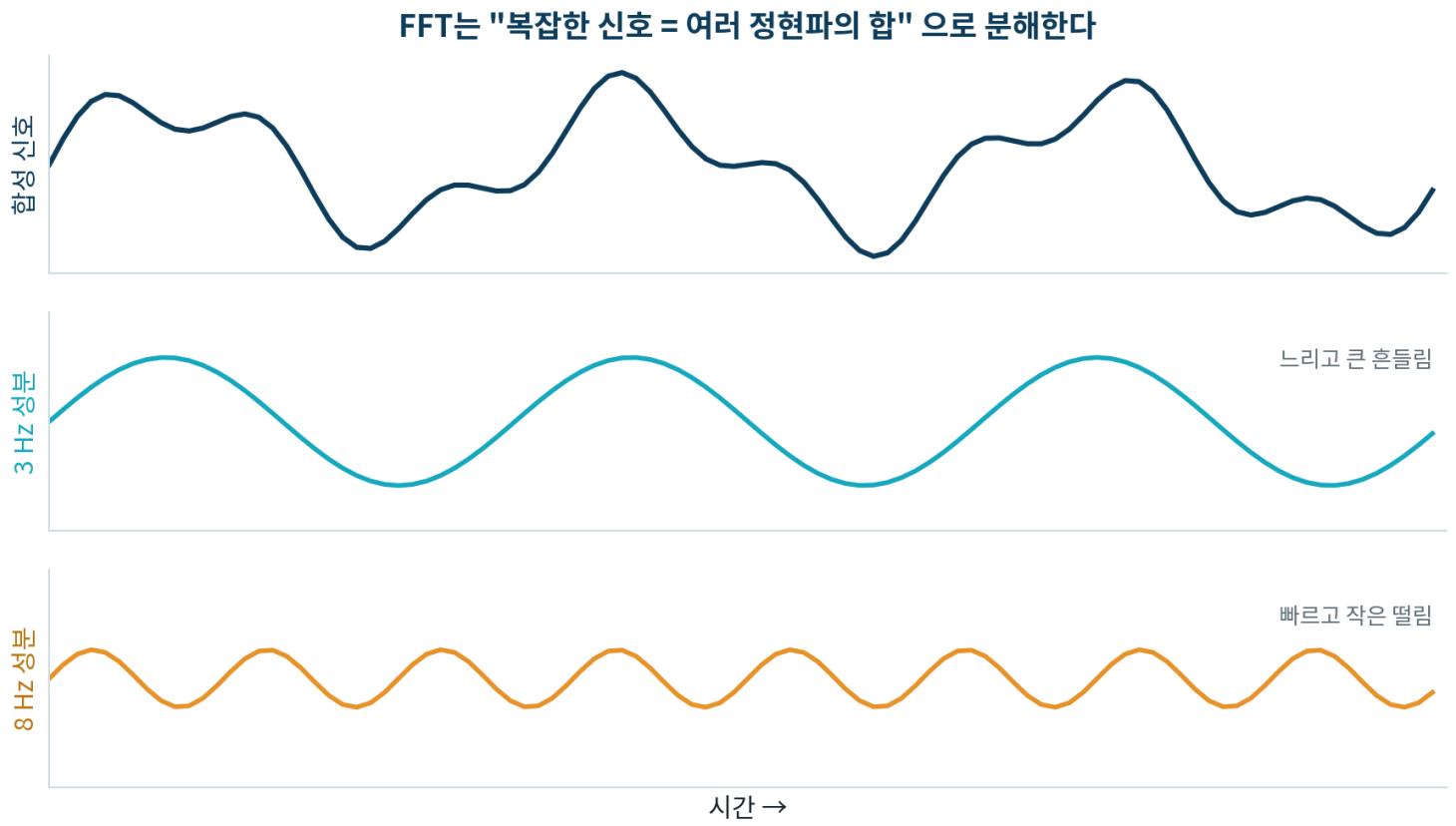
FFT

- **FFT** (Fast Fourier Transform) : 신호에 **어떤 주파수가 얼마나 섞여 있는지** 계산



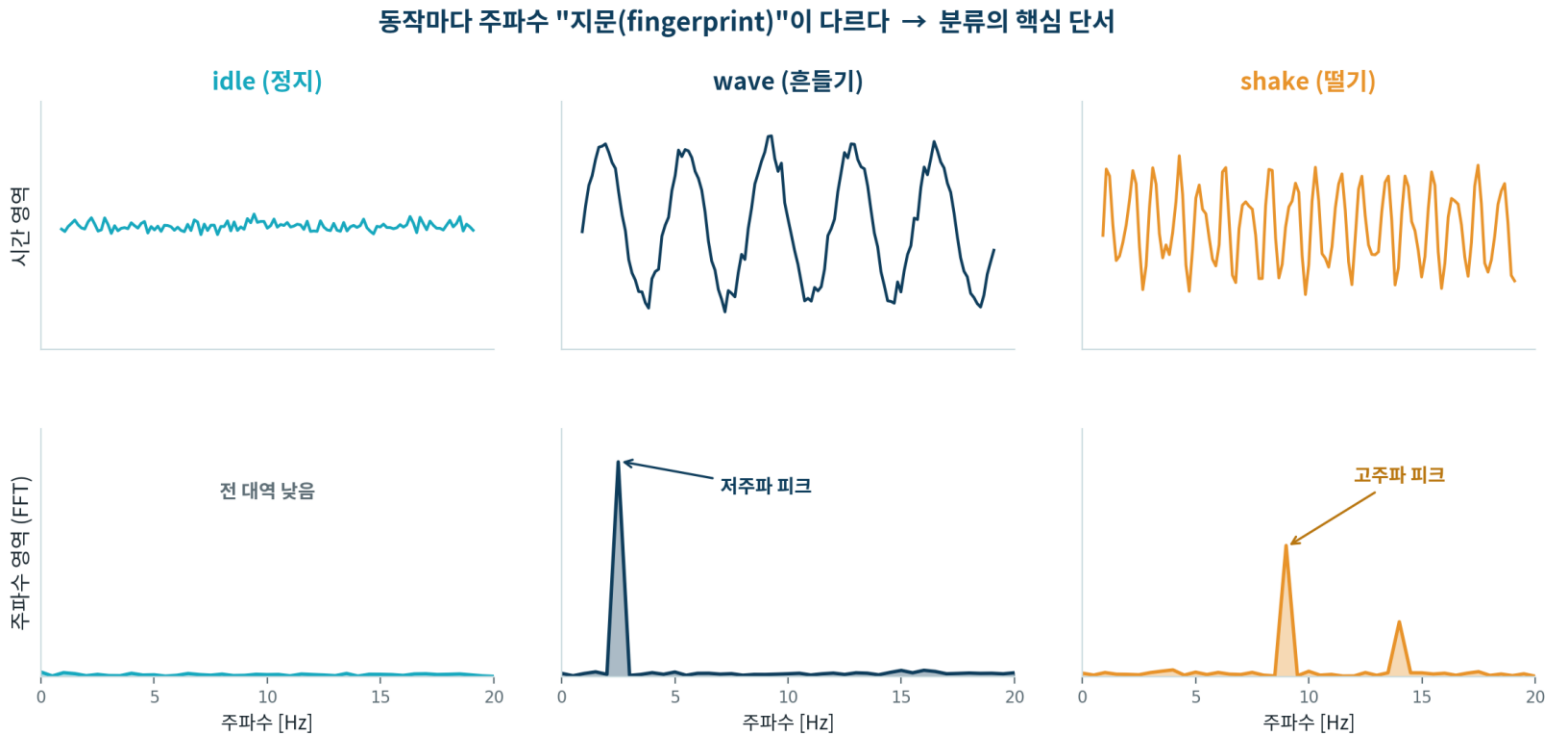
복잡한 신호 = 정현파의 합

- 모든 신호는 **여러 사인파(정현파)의 합**으로 분해할 수 있다 (푸리에 급수)



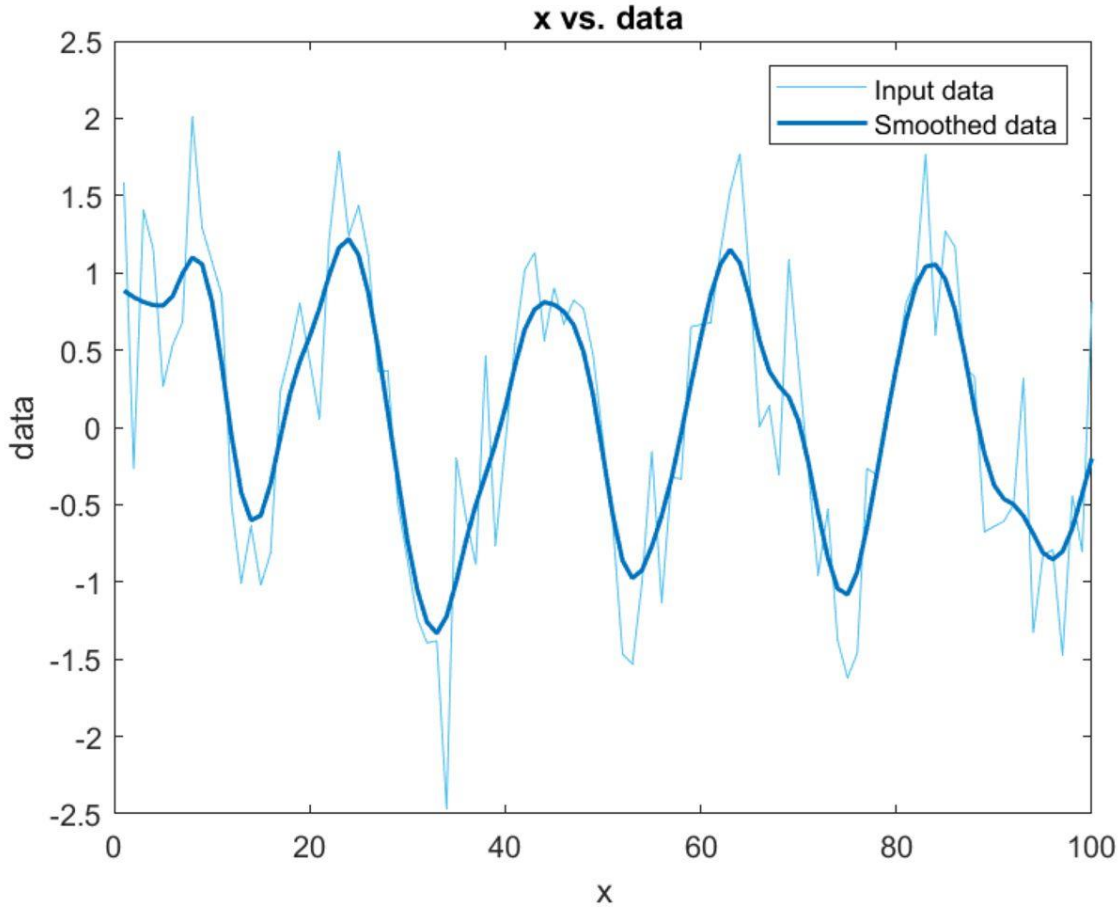
동작에 따른 신호의 주파수 형태

- 정지 · 흔들기 · 떨기 — 시간 파형은 헷갈려도 **스펙트럼은 확연히 구분됨.**



그 외 전처리 방법

- Filtering, 이동평균, convolution 연산 등 다양한 특성 추출을 위한 전처리 방법이 사용됨.



신경망 학습(Training)

■ **Classifier** 탭에서 학습 파라미터를 정하고 모델을 훈련

- **주요 설정 :** epochs, learning rate (학습률), 네트워크 구조(Dense 층, ANN 등)
- **기본값으로 시작해도 무방 :** epochs 30~50, learning rate 0.0005 부근
- **[Start training]** → 학습 곡선과 정확도가 실시간 표시
- **양자화(int8) 옵션 :** On으로 두면 MCU용 경량 모델까지 함께 생성
- 결과에 **정확도 · 손실 · 예상 RAM/Flash · 추론시간** 이 함께 표시됨

신경망 · 양자화 · 성능 평가

■ MCU에 올리려면 모델이 **작고 빨라야** 함 — 핵심 개념 3가지

- **경량 신경망 :** 완전연결(Dense) 몇 층이면 충분한 경우가 많음 (수십 KB)
- **양자화(Quantization) :** 32bit 실수(float) → **8bit 정수(int8)** 로 변환해 크기·연산을 대폭 축소(정확도 손실 최소)
- **성능 지표 :** 정확도, **혼동행렬(confusion matrix)**, 온디바이스 지연시간·RAM·Flash 사용량
  - Studio가 보드에서의 예상 추론시간과 메모리 사용량을 미리 알려줌
- **과적합 주의 :** 학습 정확도만 높고 테스트가 낮으면 데이터 다양성·양을 늘려야 함

결과 읽기 & 자동 최적화

■ 정확도 숫자 하나가 아니라 **혼동행렬** 로 어디서 헷갈리는지 봐야함

- **혼동행렬(confusion matrix) :** 실제 vs 예측을 표로 — 대각선이 진할수록 좋음
  - 특정 두 동작이 서로 혼동되면 그 클래스 데이터를 더 수집
- **Model testing :** 떼어둔 테스트셋으로 **일반화 성능** 확인(학습 정확도와 큰 차이면 과적합)
- **EON Tuner(선택) :** 여러 DSP·모델 조합을 자동 실험해 **정확도-크기-속도 균형** 최적안 추천
- **목표 :** 테스트 정확도를 높이면서 RAM/Flash·지연이 보드 budget 안에 들게

Arduino 라이브러리로 배포

■ **Deployment** 탭에서 모델을 **Arduino 라이브러리(.zip)** 로 내보냅니다

- ① **Deployment > Arduino library** 선택
- ② **최적화 옵션 : Quantized(int8)** 권장 — 더 작고 빠름
- ③ [Build] → **.zip** 파일 다운로드
- ④ **Arduino IDE :** Sketch > Include Library > Add .ZIP Library
- ⑤ **예제 열기 :** 파일 > 예제 > (프로젝트명)_inferencing > **nano_ble33_sense_accelerometer**
  - 이 예제가 센서를 읽어 실시간으로 추론까지 수행

MODULE 05

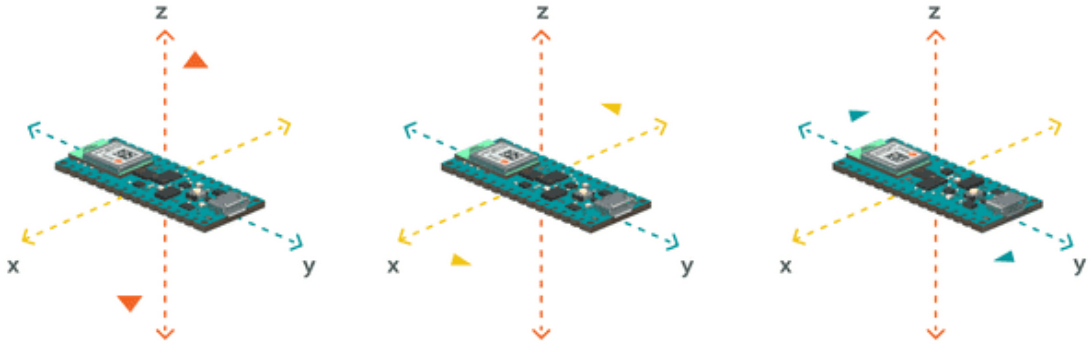
실습

Hands-on Practice

Practice #1~2 — 기계학습 모델과 실습

실습 예제 #1 IMU sensor

- **Arduino nano 33 BLE rev2 보드는 IMU (Inertial Measurement Unit) 센서를 포함하고 있음**



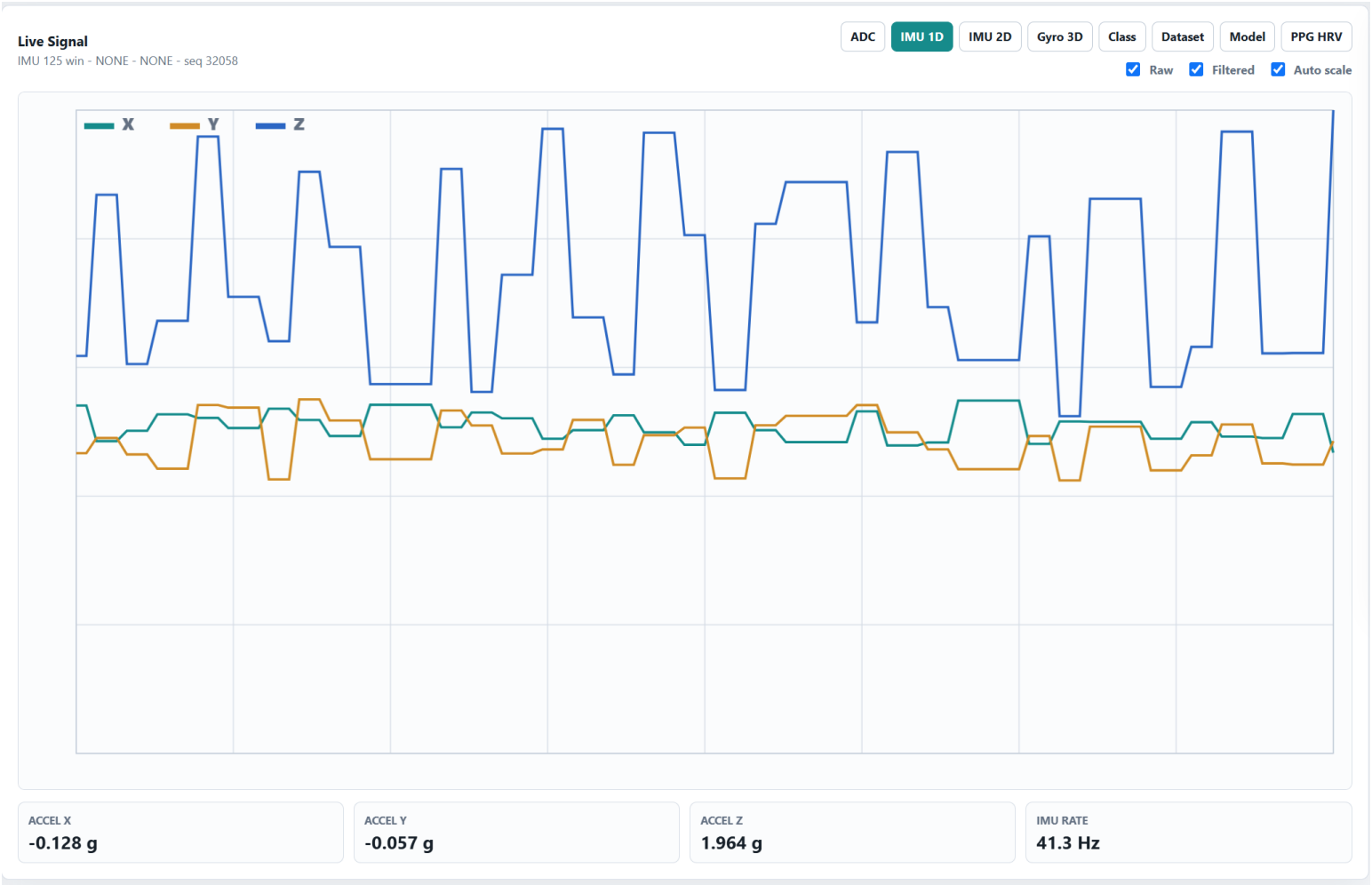
BMI270 Accelerometer
통신 방식: I^2C

사용 예제파일

ex06_imu_raw_stream_rev2.ino

실습 예제 #1 IMU sensor

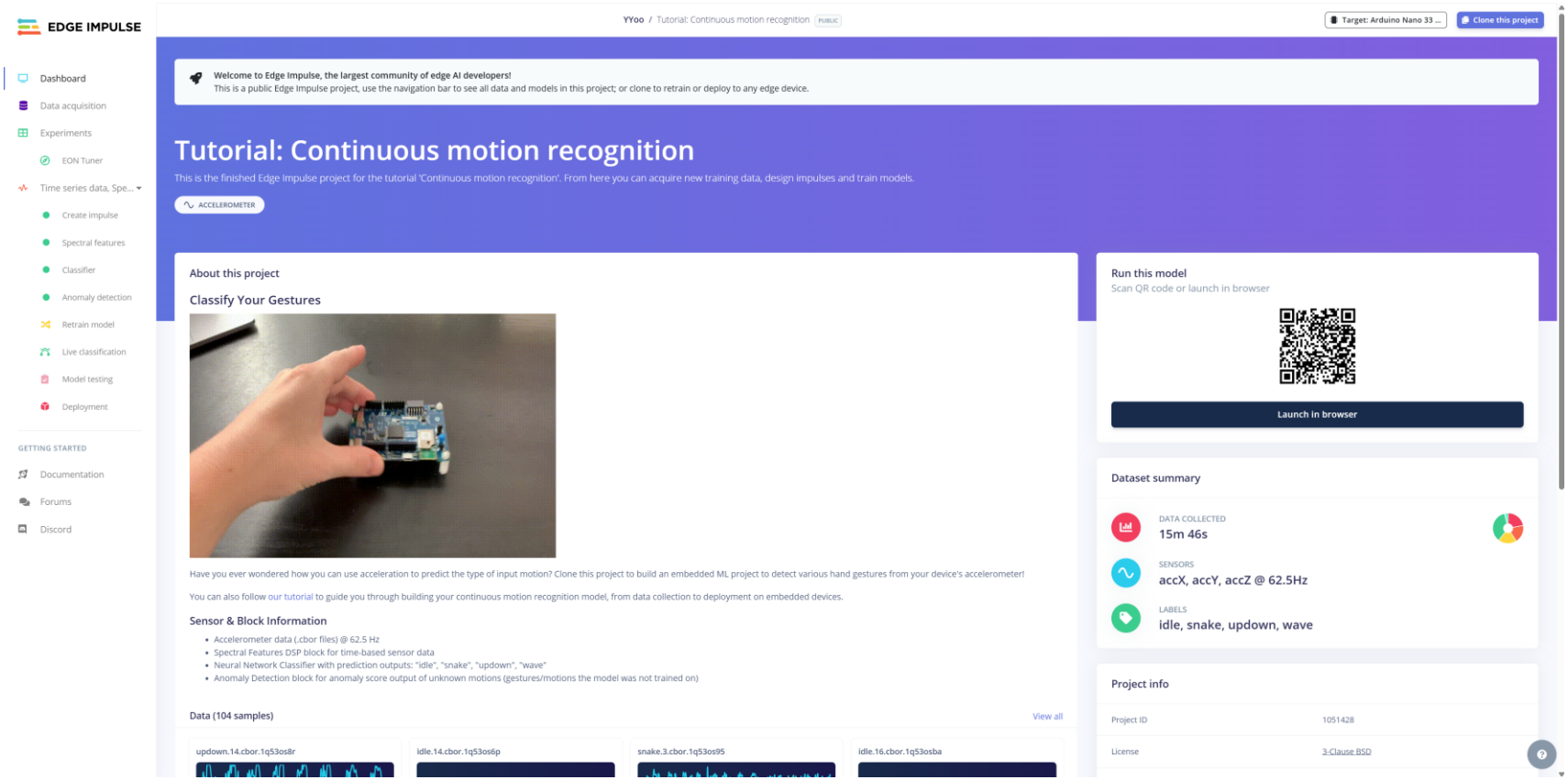
- IMU 1D 탭을 통해 IMU 센서 값(가속도)이 나오는 것을 확인



실습 예제 #1 Edge Impulse

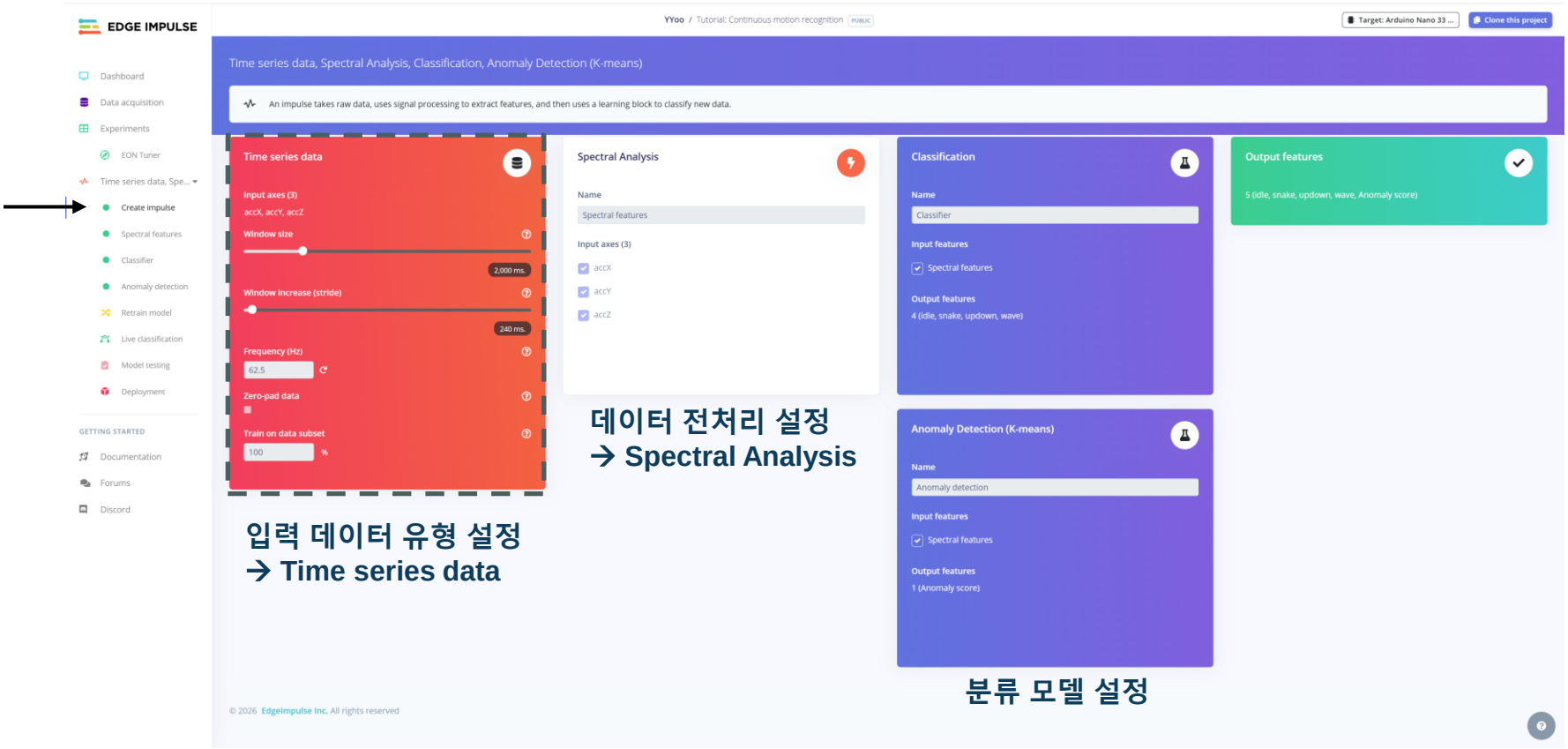
- **IMU 센서 기반 동작분류를 위한 온보드 모델 예제**

  https://studio.edgeimpulse.com/public/1051428/live



실습 예제 #1 Edge Impulse

- 실시간 분류를 위한 모델 구조 설계



입력 데이터 유형 설정
→ Time series data

데이터 전처리 설정
→ Spectral Analysis

분류 모델 설정

실습 예제 #1 Edge Impulse

- **Feature 추출을 위한 데이터 전처리**

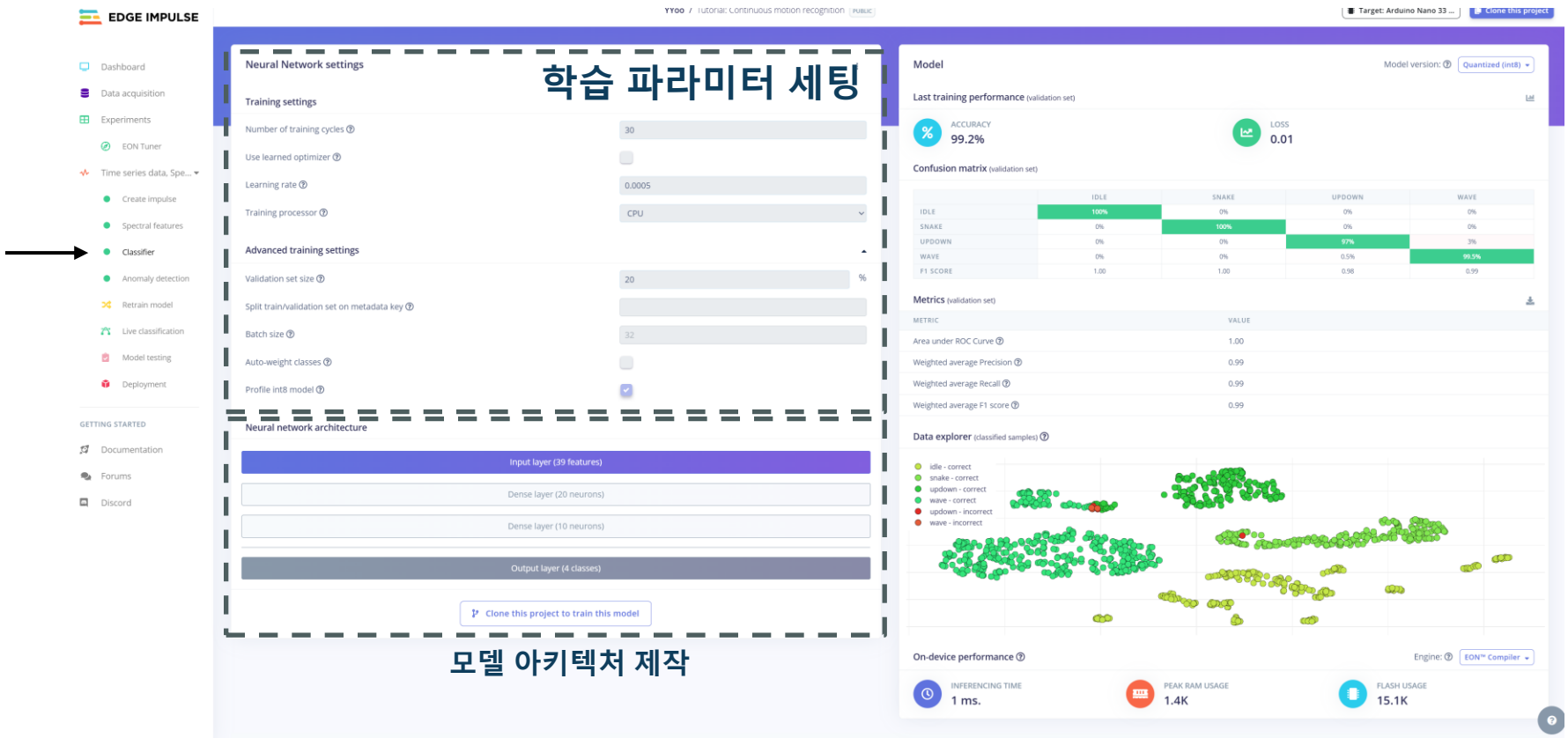
각 축의 가속도를 주파수 성분으로 분석

Low pass filter 적용 후 FFT 수행

데이터 전처리를 통해 입력 데이터의 특징을 추출하고 분류 성능을 향상시킬 수 있음

실습 예제 #1 Edge Impulse

- 분류 모델 세팅



입력 데이터의 특성에 따라 ANN, CNN 등 알맞은 모델 아키텍처를 선정

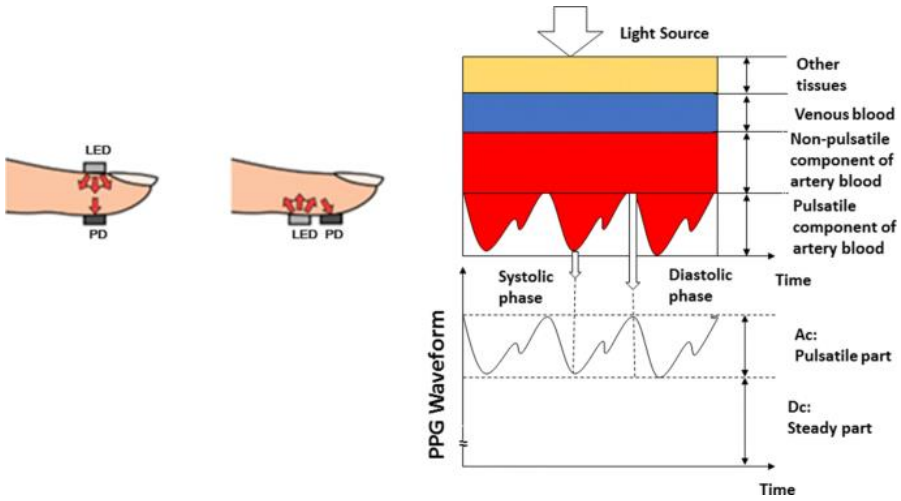
실습 예제 #1 Edge Impulse

- 모델 배포

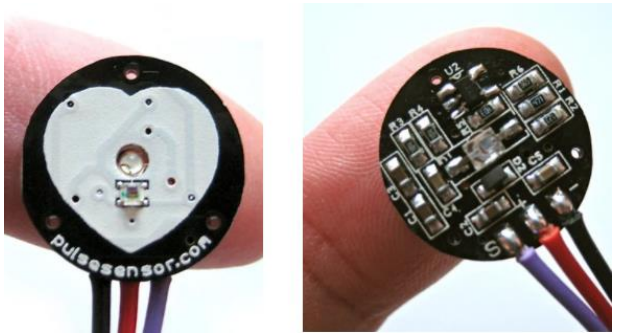
Arduino library 선택 후 Build → 라이브러리 다운로드

실습 예제 #2 PPG

- **Photoplethysmography**
- **혈류 측정을 위한 광 기반 측정법**



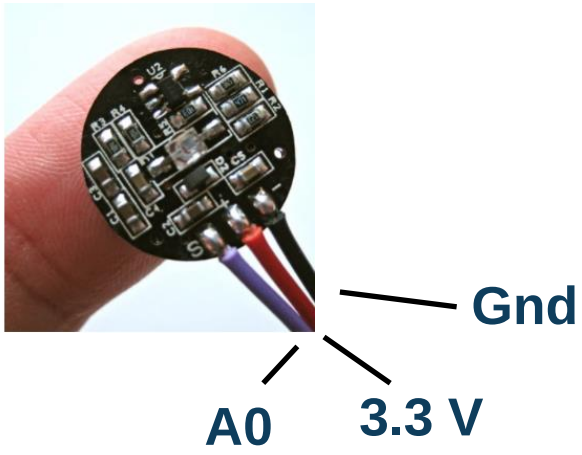
PPG 측정 방식

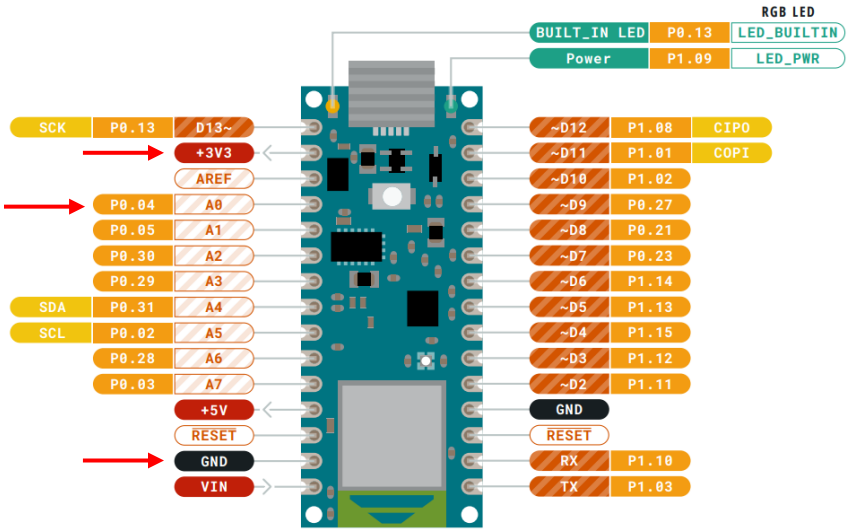


상용 PPG 센서

실습 예제 #2 PPG

- PPG 센서 연결법





실습 예제 #2 PPG

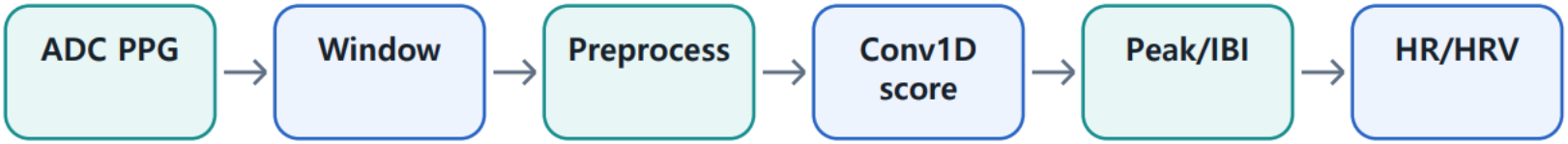
- **PPG**는 빛을 이용해 혈류 변화를 측정한 맥파 신호이다. 심장이 뛸 때마다 혈류량이 변하고, 이 변화가 파형의 봉우리로 나타난다. 봉우리 사이의 시간 간격을 보면 심박수와 심박 변동성을 계산할 수 있다

- PPG 파형에서 심장 박동이 발생한 위치를 찾는다. 박동 위치를 알면 HR과 HRV를 계산할 수 있다. 이 실습에서는 Conv1D Peak Detector와 Ridge Regression을 비교한다.

- 두 모델은 출력이 다르다. Conv1D는 박동 위치를 찾고, regression은 HR 숫자를 바로 예측한다

HR: Heart Rate 1분에 심장이 몇 번 뛰는지
HRV: Heart Rate Variability. 박동 간격이 얼마나 흔들리는지
Regression: 연속적인 숫자 하나를 예측하는 방법

실습 예제 #2 PPG



- ADC에서 들어온 PPG 신호를 일정한 window로 자른다.
- 전처리 후 Conv1D path와 regression path로 동시에 보낸다.
- Conv1D path는 peak timing을 찾고, regression path는 HR 값을 직접 예측한다

실습 예제 #2 PPG

- **PPG DaLia dataset 활용하여 학습 :** 170명의 피실험자로부터 PPG와 HR데이터 측정 (추가적으로 가속, ECG, 온도, EDA 까지 제공됨.)

Time	acc_x	acc_y	acc_z	ppg	eda	temp	hr	activity	activity_lat	Subject
44:52.0	-27	1	58	0	0	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.0	-27	2	59	0	0	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.1	-27	1	59	0	0	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.1	-27	1	58	0	0	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.1	-27	1	59	0	0	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.2	-27	1	59	0	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.2	-27	1	59	0.01	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.2	-27	1	58	-0.03	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.2	-27	1	58	-0.05	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.3	-27	1	59	0.13	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.3	-27	1	59	0.66	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.3	-27	1	59	1.37	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.4	-27	1	59	2.06	0.832765	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.4	-27	1	59	2.79	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.4	-27	1	58	3.8	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.5	-27	1	59	5.06	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.5	-27	1	58	6.2	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.5	-27	1	59	7.15	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.6	-27	1	59	8.31	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.6	-27	1	59	10.09	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.6	-27	1	59	11.88	1.180231	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.7	-27	2	59	13.14	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.7	-27	1	59	14.65	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.7	-27	1	59	17.91	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.7	-27	1	58	22.9	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.8	-27	1	59	27.58	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.8	-27	1	59	31.68	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.8	-27	1	58	37.81	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.9	-27	1	58	48.15	1.618462	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.9	-27	1	59	59.83	2.09644	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:52.9	-27	1	59	69.58	2.09644	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:53.0	-27	1	58	79.34	2.09644	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:53.0	-27	1	58	93.76	2.09644	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1
44:53.0	-27	1	59	112.62	2.09644	32.15	76	NO_ACTIV	NO_ACTIV	Subject_1

실습 예제 #2 PPG

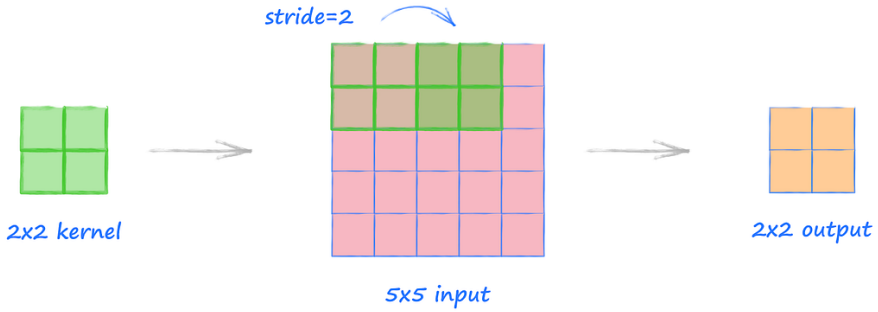
- 전처리 과정
  - 실제 센서 신호에는 느린 흔들림과 잡음이 포함된다.
  - Baseline removal은 천천히 변하는 기준선을 제거한다.
  - Smoothing은 작은 고주파 잡음을 줄인다.
  - Z-score normalization은 window 평균을 0, 표준편차를 1로 맞춘다

전처리는 모델 성능에 직접 영향을 준다. 같은 심박이라도 센서 접촉이나 움직임에 따라 신호 크기와 기준선이 달라질 수 있기 때문이다.

실습 예제 #2 PPG

- Conv1D는 1차원 시간 신호 위를 움직이는 작은 탐지기이다.
- 이 탐지기는 현재 구간이 맥박 봉우리처럼 생겼는지 확인한다. 비슷하면 큰 값, 다르면 작은 값을 출력한다.
- 현재 모델은 7, 9, 11 tap filter를 사용한다

이미지 CNN은 작은 필터가 사진 위를 훑으며 edge나 texture를 찾는다. 여기서는 필터가 시간축 PPG 위를 훑으며 pulse peak 모양을 찾는다



실습 예제 #2 PPG

- ReLU는 음수 반응을 0으로 바꾼다.
- 양수 반응은 peak-like evidence로 남긴다. 관련 없는 반응이나 반대 모양의 반응은 제거된다.

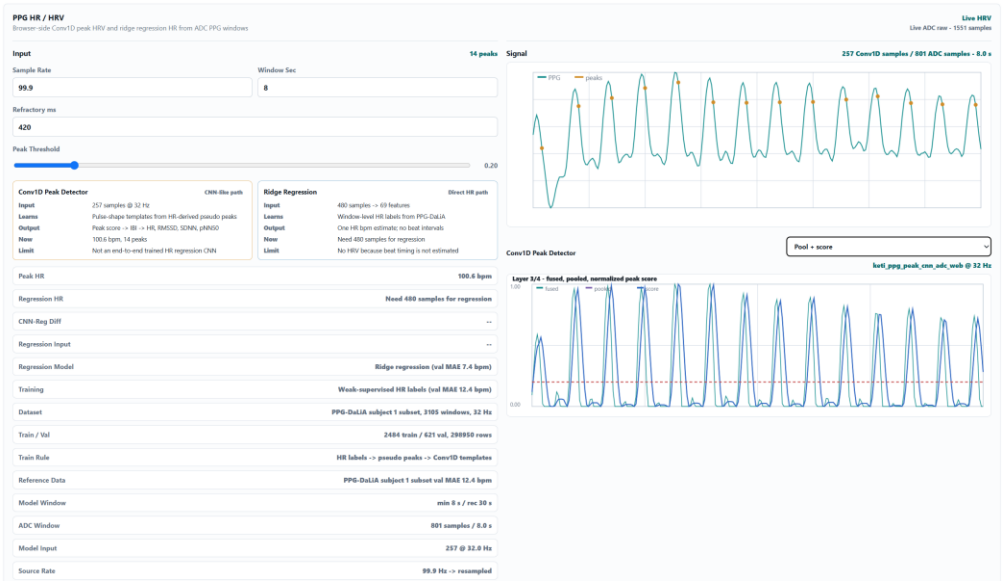
fused[t] = average(relu_1[t], relu_2[t], relu_3[t])

- 7, 9, 11 tap filter는 서로 다른 폭의 peak를 본다.
- 각 filter의 ReLU map을 평균해 하나의 fused map을 만든다. 좁은 peak, 중간 peak, 넓은 peak에 대한 증거가 통합된다

- 현재 pooling은 downsampling이 아니라 시간축 smoothing이다. Pooling radius는 0.06초이며, 32 Hz 기준 앞뒤 약 2 sample을 포함한다. 작은 spike를 줄이고, peak score가 너무 흔들리지 않게 만든다. 너무 큰 pooling은 가까운 beat를 합치거나 peak 위치를 흐리게 만들 수 있다

실습 예제 #2 PPG

- Pooling 결과를 0-1 범위의 peak score로 정규화한다. Threshold보다 큰 local maximum을 peak 후보로 선택한다. 너무 가까운 peak는 refractory rule로 하나만 남긴다. Threshold가 낮으면 false peak가 늘고, 높으면 실제 peak를 놓칠 수 있다.
- 데이터셋에는 HR label은 있지만 정확한 beat 위치 annotation은 없다. HR label로 예상 beat period를 계산한다. PPG local extrema 중 예상 period와 맞는 후보를 pseudo peak로 선택한다. pseudo peak 주변 waveform snippet을 평균해 Conv1D filter template을 만든다



실습 예제 #2 PPG

- **Regression 모델**
- Regression은 peak를 찾지 않고 HR 숫자 하나를 직접 예측한다. 15초 PPG window를 69개 feature로 변환한다. Autocorrelation, spectrum, slope 관련 feature를 사용한다. Ridge regression은 HR label과 예측 HR의 오차를 줄이도록 weight를 학습한다

$$HR = bias + w1\ f1 + w2\ f2 + ... + w69\ f69$$

Regression은 성능 지표상 HR 예측이 더 안정적일 수 있지만, beat timing을 출력하지 않기 때문에 HRV를 계산할 수 없다

실습 예제 #2 PPG

- **Regression 모델**
- Regression은 peak를 찾지 않고 HR 숫자 하나를 직접 예측한다. 15초 PPG window를 69개 feature로 변환한다. Autocorrelation, spectrum, slope 관련 feature를 사용한다. Ridge regression은 HR label과 예측 HR의 오차를 줄이도록 weight를 학습한다

$$HR = bias + w1\ f1 + w2\ f2 + ... + w69\ f69$$

Regression은 성능 지표상 HR 예측이 더 안정적일 수 있지만, beat timing을 출력하지 않기 때문에 HRV를 계산할 수 없다