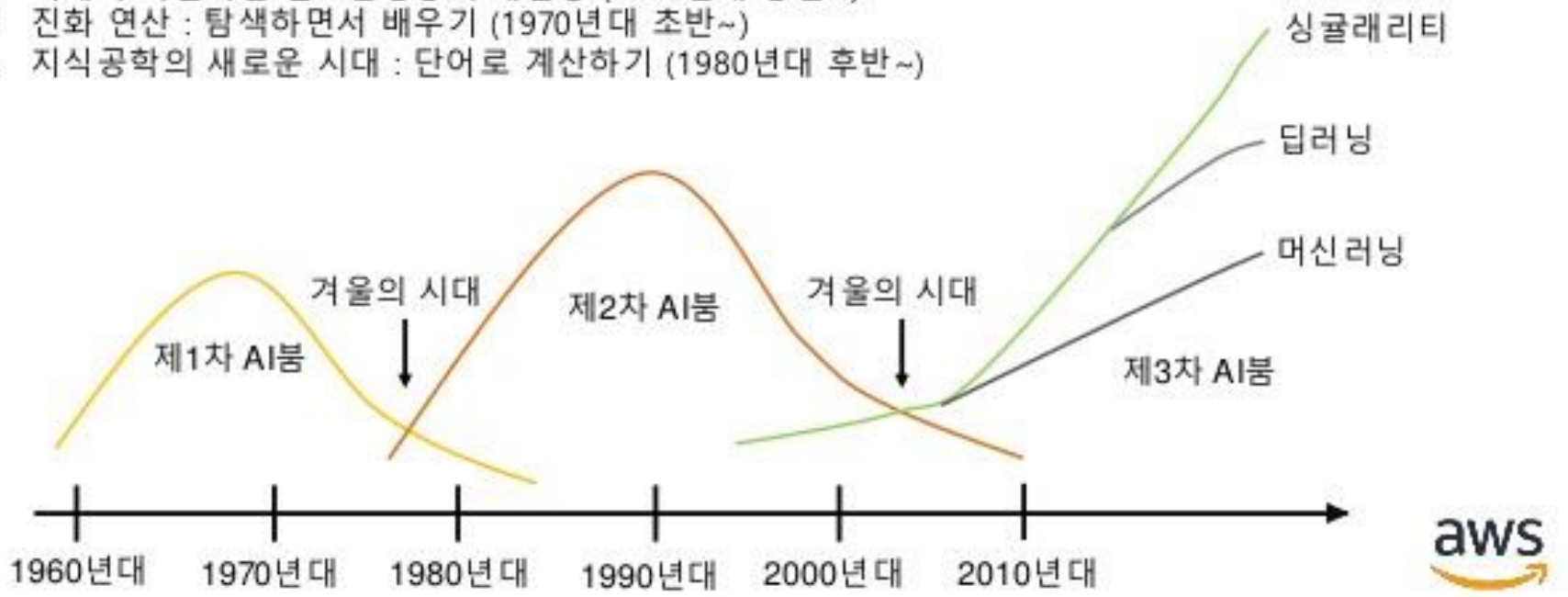
딥러닝 개요 및 동작 원리

2019.11

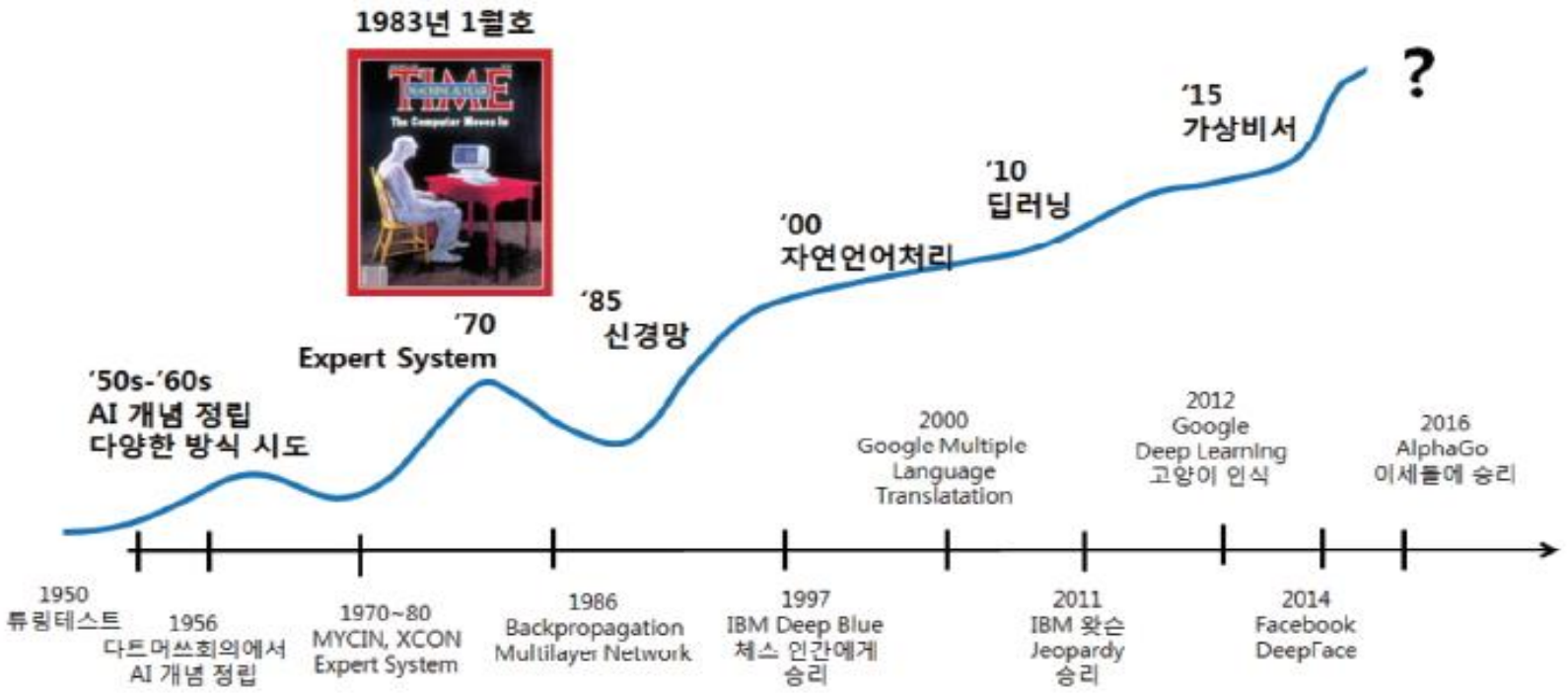
■ 인공지능의 역사

'암흑기'에서 지식기반 시스템에 이르는 AI의 역사

1. 암흑기 : AI의 탄생 (1943년~1956년)
2. AI의 융성 : 큰 기대의 시대 (1956년~1960년대 후반)
3. 이행되지 않은 약속 : 현실의 직면 (1960년대 후반~1970년대 초반)
4. 전문가 시스템의 기술 : 성공의 열쇠 (1970년대 초반~1980년대 중반)
5. 기계가 학습하는 법 : 신경망의 재탄생 (1980년대 중반~)
6. 진화 연산 : 탐색하면서 배우기 (1970년대 초반~)
7. 지식공학의 새로운 시대 : 단어로 계산하기 (1980년대 후반~)



■ 인공지능의 역사



※ 출처 : 소프트웨어정책연구소, 알파고의 능력은 어디에서 오는가?, (2016)

■ 인공지능/딥러닝 개요

❖ 인공지능의 정의

- 지능형 기계를 만드는 과학 및 공학 (John McCarthy)
- 컴퓨터에서 지능적인 행동을 시뮬레이션하는 컴퓨터 과학 분야
- 지능적인 인간의 행동을 모방하는 기계의 능력
- 일반적으로 사람의 지능을 필요로 하는 작업을 수행할 수 있는 컴퓨터 시스템(ex. 시각적 인식, 음성 인식, 의사결정, 언어 간의 번역)

■ 인공지능/딥러닝 개요

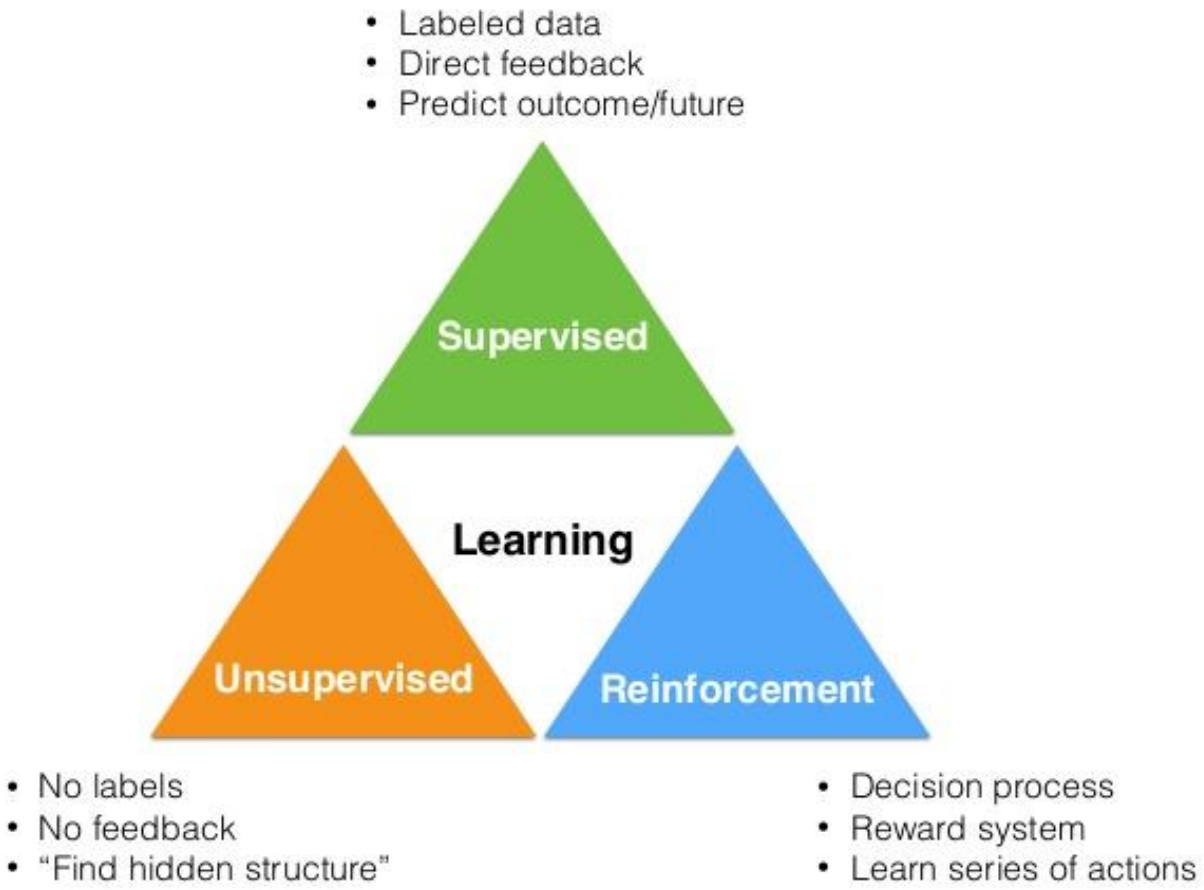
❖ 머신 러닝: 스스로 변화하는 프로그램

- 인공지능의 부분 집합
- 컴퓨터에게 명시적으로 프로그래밍 하지 않고 학습을 할 수 있는 능력을 주는 연구 분야 (Arthur Samuel, 1959)
- 더 많은 데이터를 접했을 때 스스로를 수정하는 능력
- 추측 값과 실제 정답 값(Ground Truth 레이블)을 대조하여 오류 값을 측정할 수 있음
- 최적화 알고리즘을 올바르게 설정하면 위 과정을 반복하여 오류를 최소화

❖ 딥 러닝: 정확도, 수학 및 컴퓨팅 기능 향상

- 머신 러닝의 부분 집합
- 이미지 인식, 음성 인식, 추천 시스템, 자연어 처리 분야
- Deep이란 신경망에서 레이어의 수를 일컫는 기술 용어
- 머신 러닝에 비해 더욱 높은 정확도를 도출하지만 더 많은 하드웨어와 학습시간을 필요로 함

■ 머신 러닝 알고리즘 분류

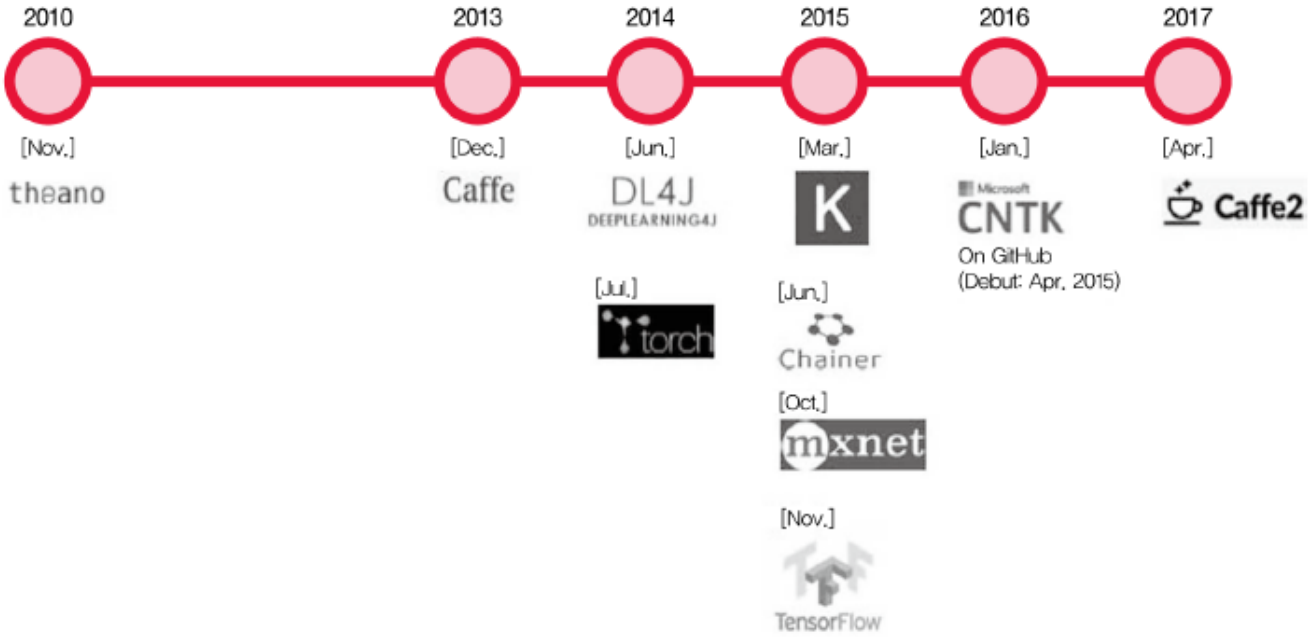

• Labeled data
• Direct feedback
• Predict outcome/future
Supervised
Learning
Unsupervised
Reinforcement
• No labels
• No feedback
• "Find hidden structure"
• Decision process
• Reward system
• Learn series of actions

■ 향후 10년 동안 인공지능(AI)이 활용될 분야

1. 안전한 자율주행차 운전
2. 대량의 이미지 처리 및 분류
3. 비정형화된 환자 데이터 처리
4. 금융 관련 데이터 분석
5. 매핑을 통해 차량 목적지까지 안내
6. 항공기, 차량 등의 부품 유지 보수
7. 데이터 분석으로 사이버 위협 방지
8. 서류를 디지털 데이터로 변환
9. 채용 과정에서 최적의 지원자 선별
10. 의료 영상 분석

■ 딥러닝 소프트웨어

- Tensorflow: 구글 브레인팀 개발, 현재 가장 많은 사람이 사용
- Caffe: UC Berkeley에서 관리
- Theano: 딥러닝 알고리즘을 파이썬으로 쉽게 구현할 수 있도록 해줌
- 토치: 페이스북과 구글 딥마인드가 사용
- CNTK: 마이크로소프트 개발
- Matlab: 상용 소프트웨어
- R: 통계분석



■ TensorFlow

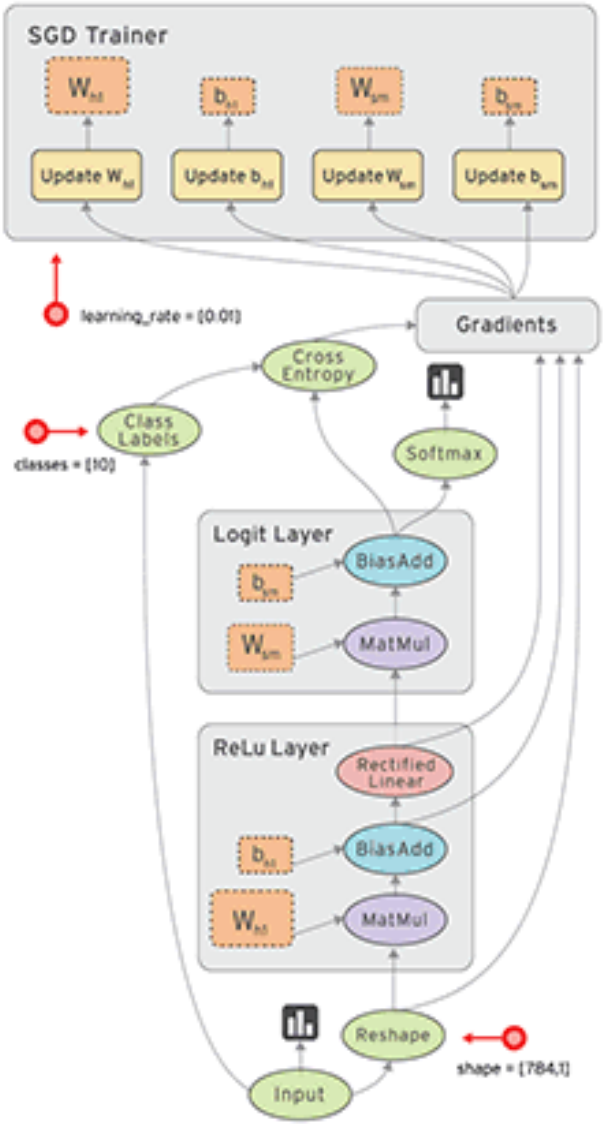
❖ 개요

- 개발자 : 구글 브레인팀
- 발표 : 2015년 11월
- 버전 : 1.15 → 2.0 ('19.10)
- C++로 개발
- 사용 가능한 언어
  - 버전 1 : 파이썬, C++
  - 버전 2 : 파이썬, C++, Java, JavaScript
- 오픈 소스
- 데이터 플로우 그래프를 통한 풍부한 표현력
- 코드 수정 없이 CPU/GPU 모드로 동작

■ TensorFlow

❖ 데이터 플로우 그래프

- 노드(Node)와 엣지(Edge)를 사용한 방향 그래프(Directed Graph)
- 노드는 수학적 계산, 데이터 입/출력, 데이터의 읽기/저장 등의 작업 수행
- 엣지는 노드들 간 데이터의 입출력 관계를 표현



■ Keras

- 구글에서 파이썬으로 작성된 오픈 소스 신경망 라이브러리
- Tensorflow, Microsoft Cognitive Toolkit(CNTK), Theano 위에서 수행
- 2017년, 구글의 텐서플로 팀은 텐서플로의 코어 라이브러리에 케라스를 지원하기로 결정
  → 버전 2.0에서는 고수준 API로 케라스를 채택
- 원칙
  - 사용자 친화성
  - 모듈형

    신경층(neural layer), 비용 함수(cost function), 옵티마이저(optimizer), 초기화 방식(initialization scheme), 활성화 함수(activation function), 정규화 방식(regularization scheme) 모두 독립적인 모듈이며 결합을 통해 새로운 모델을 만들 수 있다.
  - 손쉬운 확장
  - 파이썬과의 연계
  - "기계가 아닌 사람을 위해 설계됐으며 인지 부하를 낮추기 위한 모범 사례에 따른다."
- 구글, 마이크로소프트, 아마존, 애플, 엔비디아, 우버 등 쟁쟁한 기업들이 지지

■ 실습 환경

❖ Anaconda3 설치

- Anaconda: https://www.anaconda.com/download/
- 현재 version: Anaconda 2019.07, Python 3.7
- 아나콘다(Anaconda) 설치 하기 on Windows(https://wonderbout.tistory.com/22) 참조

❖ Keras/Tensorflow 설치

- pip install tensorflow
- pip install keras

❖ Google Colaboratory(colab) 서비스

- gmail 계정 필요
- http://colab.research.google.com/
- 사용법: 데이터 사이언스 스쿨 (https://datascienceschool.net/view-notebook/338fe23b46464e9b9c4d4c8f8c7c7258/) 참조

■ 선형 회귀의 정의

❖ “학생들의 중간고사 성적이 다르다.”

- 위 문장이 나타낼 수 있는 정보는 너무 제한적
- 학급의 학생마다 제각각 성적이 다르다는 당연한 사실 외에는 알 수 있는 게 없음

❖ "학생들의 중간고사 성적이 [ ]에 따라 다르다."

- 이 문장은 정보가 담길 여지를 열어 놓고 있음
- [ ] 부분에 시험 성적을 좌우할 만한 여러 가지 것이 들어간다면 좀 더 많은 사실을 전달할 수 있음
- 예를 들면 공부한 시간, 시험 당일의 컨디션, 사교육비 지출액 등이 들어갈 수 있음
- 무엇이 들어가든지 해당 성적의 이유를 나름대로 타당하게 설명할 수 있음
- 따라서 이 문장이 중간고사 성적의 차이와 이유를 나타낼 때 더욱 효과적
- 여기서 [ ]에 들어갈 내용을 ‘정보’라고 함
- 머신러닝과 딥러닝은 이 정보가 필요함

  → 많은 정보가 더 정확한 예측을 가능케 하며, 이때의 ‘많은 정보’가 곧 ‘빅데이터’

■ 선형 회귀의 정의

❖ 독립변수/종속변수

- 성적을 변하게 하는 '정보' 요소를 x라고 하고, 이 x 값에 의해 변하는 '성적'을 y라할때,
- x 값이 변함에 따라 y 값도 변한다
- 이때, 독립적으로 변할 수 있는 값 x를 독립 변수라고 함
- 이 독립 변수에 따라 종속적으로 변하는 y를 종속 변수라고 함

❖ 선형 회귀란 독립 변수 x를 사용해 종속 변수 y의 움직임을 예측하고 설명하는 작업

❖ 단순/다중 선형 회귀

- 단순 선형 회귀(simple linear regression): 하나의 x 값만으로 y 값을 설명 할 수 있을 때
- 다중 선형 회귀(multiple linear regression): x 값이 여러 개 필요할 때

■ 가장 훌륭한 예측선이란?

❖ 단순 선형 회귀 예

- 4명 학생의 중간고사 성적과 공부한 시간

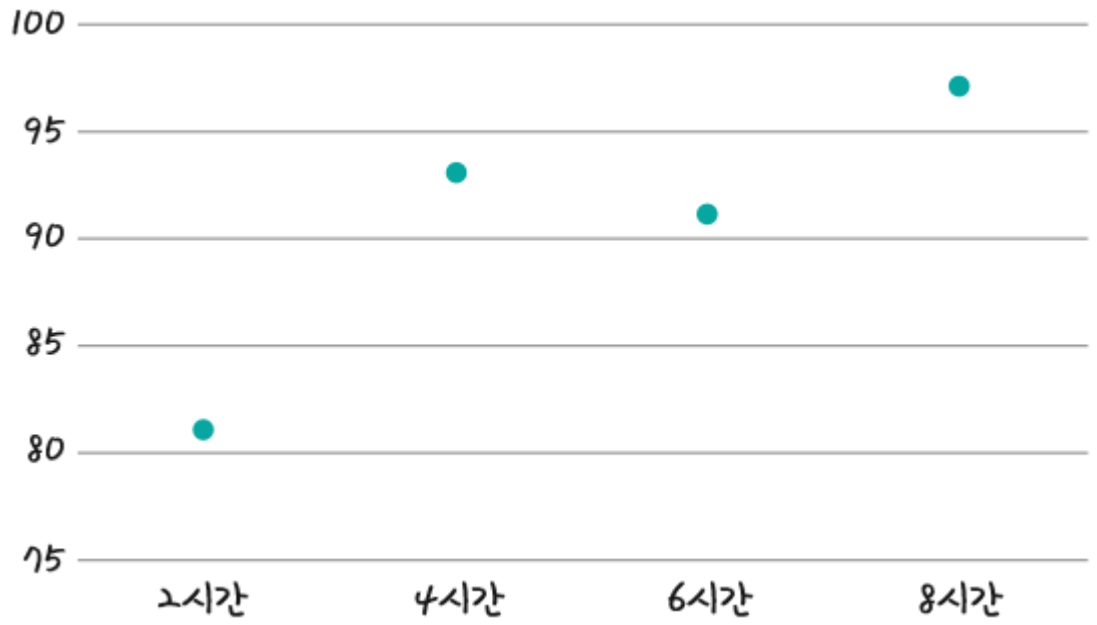
공부한 시간	2시간	4시간	6시간	8시간
성적	81점	93점	91점	97점

- 공부한 시간을 x라 하고 성적을 y라 할 때 집합 x와 집합 y를 다음과 같이 표현할 수 있음

$$x = \{2,\ 4,\ 6,\ 8\}$$
$$y = \{81,\ 93,\ 91,\ 97\}$$

■ 가장 훌륭한 예측선이란?

❖ 단순 선형 회귀 예



- 선형 회귀를 공부하는 과정은 이 점들의 특징을 가장 잘 나타내는 선을 그리는 과정과 일치
- 여기에서 선은 직선이므로 곧 일차 함수 그래프

$$y = ax + b$$

■ 가장 훌륭한 예측선이란?

❖ 예측선을 그리는 이유

- 잘 그어진 직선을 통해 공부한 시간과 중간고사 성적 데이터에 들어 있지 않은 여러 가지 내용을 유추할 수 있음
- 예를 들어, 표에 나와 있지 않은 또 다른 학생의 성적을 예측하고 싶을때, 정확한 직선을 그어 놓았다면 이 학생이 몇 시간을 공부했는지만 물어보면 됨
- 정확한 a와 b의 값을 따라 움직이는 직선에 학생이 공부한 시간인 x 값을 대입하면 예측 성적인 y 값을 구할 수 있는 것

❖ 딥러닝과 머신러닝의 '예측' 이란?

- 기존 데이터(정보)를 가지고 어떤 선이 그려질지를 예측한 뒤,
- 아직 답이 나오지 않은 그 무언가를 그 선에 대입해 보는 것

■ 최소 제곱법

- 정확한 기울기 a와 정확한 y 절편의 값 b를 알아내는 간단한 방법
- 회귀 분석에서 사용되는 표준 방식으로, 실험이나 관찰을 통해 얻은 데이터를 분석하여 미지의 상수를 구할 때 사용되는 공식
- 최소 제곱법 공식을 알고 적용한다면, 일차 함수의 기울기 a와 y 절편 b를 바로 구할 수 있음

$$a = \frac{(x - x\text{ 평균})(y - y\text{ 평균})\text{의 합}}{(x - x\text{ 평균})\text{의 합의 제곱}}$$

- 식으로 표현하면

$$a = \frac{\sum_{i=1}^{n} (x - mean(x))(y - mean(y))}{\sum_{i=1}^{n} (x - mean(x))^2}$$

■ 최소 제곱법

- 성적(y)과 공부한 시간(x)을 가지고 최소 제곱법을 이용해 기울기 a를 구하면
  1. x 값의 평균과 y 값의 평균을 각각 구한다.
     - 공부한 시간(x) 평균: (2 + 4 + 6 + 8) ÷ 4 = 5
     - 성적(y) 평균: (81+ 93 + 91 + 97) ÷ 4 = 90.5
  2. 이를 식에 대입한다.

$$a = \frac{(2-5)(81-90.5)+(4-5)(93-90.5)+(6-5)(91-90.5)+(8-5)(97-90.5)}{(2-5)^2+(4-5)^2+(6-5)^2+(8-5)^2}$$

$$= \frac{46}{20}$$

$$= 2.3$$

■ 최소 제곱법

- y 절편인 b를 구하는 공식

$$b = y\text{의 평균} - (x\text{의 평균} \times \text{기울기 } a)$$

- 식으로 표현하면

$$b = mean(y) - (mean(x) * a)$$

- 식에 대입하면

$$\begin{aligned} b &= 90.5 - (2.3 \times 5) \\ &= 79 \end{aligned}$$

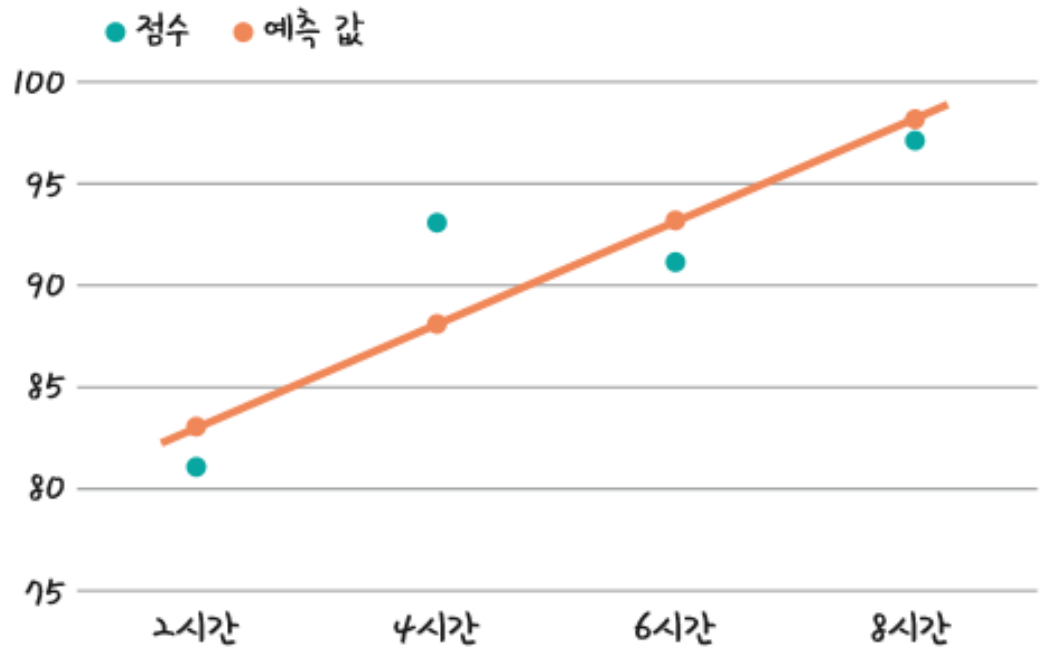
- 직선의 방정식

$$y = 2.3x + 79$$

■ 최소 제곱법

❖ 최소 제곱법 공식으로 구한 성적 예측 값과 실제 값

공부한 시간	2	4	6	8
성적	81	93	91	97
예측 값	**83.6**	**88.2**	**92.8**	**97.4**



■ 최소 제곱법

❖ 코딩으로 확인 ▶ 01_최소제곱법.ipynb

■ 평균 제곱근 오차

- 최소 제곱법의 한계

  → '여러 개의 입력(x)' 값이 있는 경우 이 공식만으로 처리할 수 없음
- 딥러닝은 대부분 입력 값이 여러 개인 상황에서 이를 해결해야 함

- 여러 개의 입력 값을 계산하는 방법
  - 임의의 선을 그리고 난 후
  - 이 선이 얼마나 잘 그려졌는지를 평가하여
  - 조금씩 수정해 가는 방법을 사용

- 이를 위해 주어진 선의 오차를 평가하는 오차 평가 알고리즘이 필요

  → 가장 많이 사용되는 방법: 평균 제곱근 오차(root mean square error)

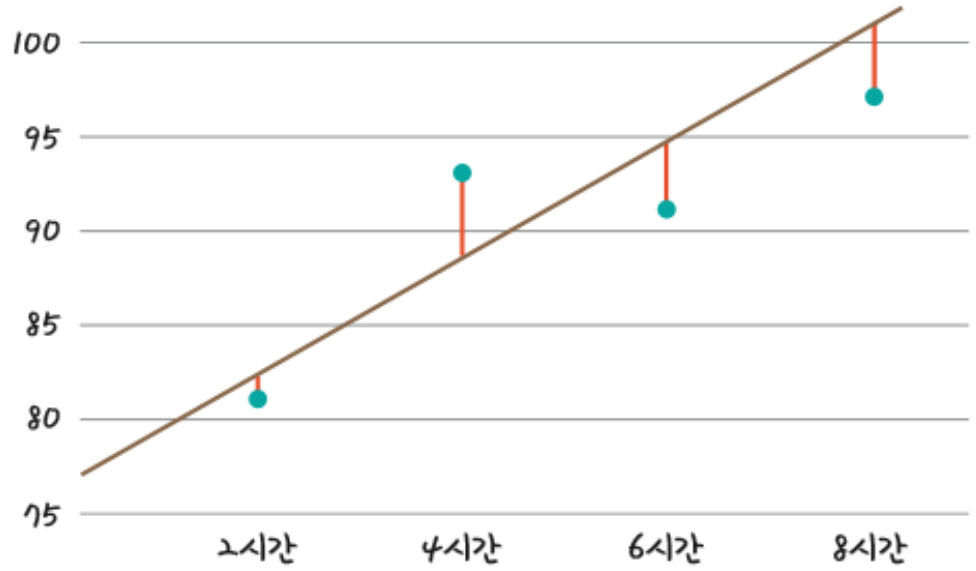
■ 잘못 그은 선 바로잡기

- '일단 그리고 조금씩 수정해 나가기' 방식에 대하여
  → 가설을 하나 세운 뒤 이 값이 주어진 요건을 충족하는지를 판단하여 조금씩 변화를 주고, 이 변화가 긍정적이면 오차가 최소가 될 때까지 이 과정을 계속 반복하는 방법

- 나중에 그린 선이 먼저 그린 선보다 더 좋은지 나쁜지를 판단하기 위해 필요한 것은?
  - 각 선의 오차를 계산할 수 있어야 한다.
  - 이 오차가 작은 쪽으로 바꾸는 알고리즘이 필요하다.

- 대강의 선을 긋기 위해 기울기 a와 y 절편 b를 임의의 수 3과 76이라고 가정해 본다면
  - y = 3x + 76인 선을 그리고
  - 이 직선과 어느 정도의 오차가 있는지를 확인한 후
  - 오차를 최소화하도록 수정

■ 잘못 그은 선 바로잡기



$y = 3x + 76$

- 주어진 데이터를 대입하여 얻을 수 있는 모든 오차의 값을 정리하면

공부한 시간(x)	2	4	6	8
성적(실제 값, y)	81	93	91	97
예측 값	82	88	94	100
오차	1	−5	3	3

■ 잘못 그은 선 바로잡기

- 부호를 없애야 정확한 오차를 구할 수 있음.
- 따라서 오차의 합을 구할 때는 각 오차의 값을 제곱해 줌

$$\text{오차의 합} = \sum_{i=1}^{n} (p_i - y_i)^2$$

- 오차 제곱의 합을 n으로 나누면 오차 제곱합의 평균을 구할 수 있고, 이를 평균 제곱 오차(Mean Squared Error, MSE)라 함

$$\text{평균 제곱 오차(MSE)} = \frac{1}{n}\sum_{i=1}^{n} (p_i - y_i)^2$$

$$\text{평균 제곱근 오차(RMSE)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (p_i - y_i)^2}$$

■ 잘못 그은 선 바로잡기

- 잘못 그은 선 바로잡기는 곧 '평균 제곱근 오차(RMSE)'의 계산 결과가 가장 작은 선을 찾는 작업

- 선형 회귀란?
  - 임의의 직선을 그어 이에 대한 평균 제곱근 오차를 구하고
  - 이 값을 가장 작게 만들어 주는 a와 b 값을 찾아가는 작업!

■ 잘못 그은 선 바로잡기

❖ 코딩으로 확인 ▶ 02_평균제곱근오차.ipynb

■ 잘못 그은 선 바로잡기

❖ 코딩으로 확인

- 실행 결과

```
공부한 시간=2, 실제 점수=81, 예측 점수=82
공부한 시간=4, 실제 점수=93, 예측 점수=88
공부한 시간=6, 실제 점수=91, 예측 점수=94
공부한 시간=8, 실제 점수=97, 예측 점수=100
rmse 최종값: 3.31662479036
```

- 이를 통해 우리가 처음 가정한 a = 3, b = 76은 오차가 약 3.3166이라는 것을 알게 됨

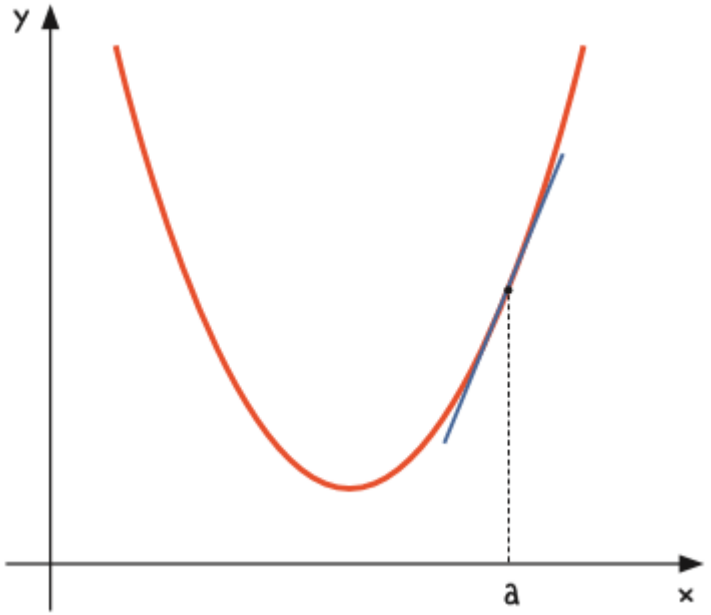
- 이제 남은 것은 이 오차를 줄이면서 새로운 선을 긋는 것

- 이를 위해서는 a와 b의 값을 적절히 조절하면서 오차의 변화를 살펴보고, 그 오차가 최소화되는 a와 b의 값을 구해야 함

■ 미분의 개념

❖ 순간 변화율의 의미

- x 값의 변화량이 0에 가까울 만큼 아주 미세하게 변화했다면,
- y 값의 변화 역시 아주 미세하게 변화했을 것
- 순간 변화율은 '어느 쪽'이라는 방향성을 지니고 있으므로 이 방향에 맞추어 직선을 그릴 수가 있음
- 이 선이 바로 이 점에서의 '기울기'라고 불리는 접선



a에서의 순간 변화율은 곧 기울기

■ 미분의 개념

❖ 미분이란?

- x 값이 아주 미세하게 움직일 때의 y 변화량을 구한 뒤,
- 이를 x의 변화량으로 나누는 과정
- 한 점에서의 순간 기울기

❖ "함수 f(x)를 미분하라"는 $\frac{d}{dx}f(x)$ 라고 표기함.

$$\frac{d}{dx}f(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

③ y 변화량의 차이를

④ x 변화량으로 나눈 값(= 순간 변화율)을 구하라는 뜻!

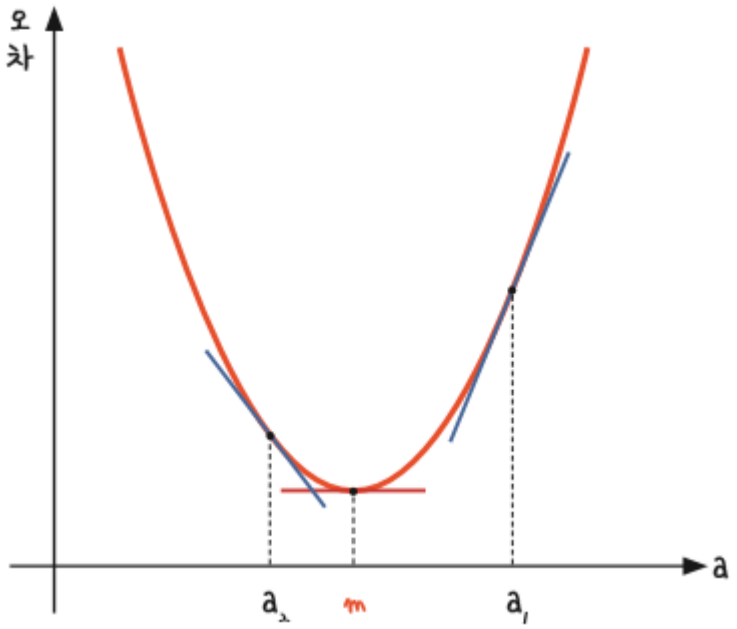
② x의 변화량이 0에 가까울 만큼 작을 때

① 함수 f(x)를 x로 미분하라는 것은

■ 경사 하강법

❖ 개요

- $y = x^2$ 그래프에서 x에 a_1, a_2 그리고 m을 대입하여 그 자리에서 미분하면 그림처럼 각 점에서의 기울기가 그려짐
- 기울기가 0인 점이 최소값
- 따라서 우리가 할 일은 '미분 값이 0인 지점'을 찾는 것!



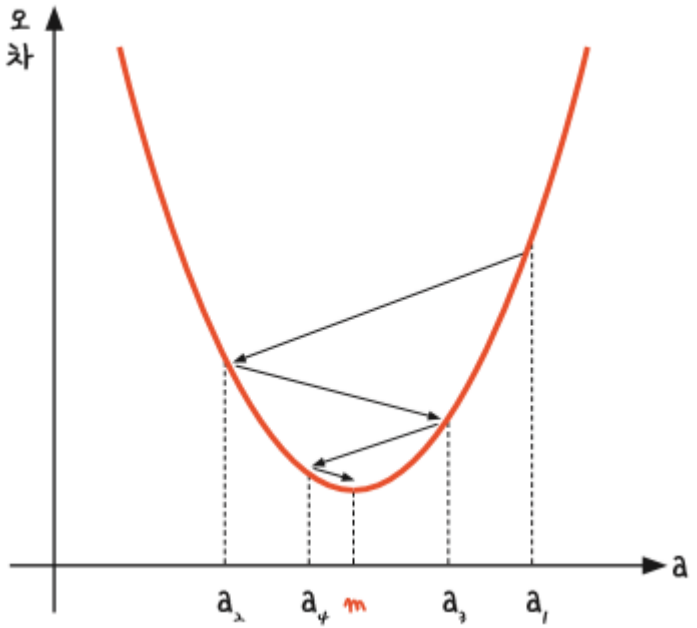
기울기가 0인 점이 최소값

■ 경사 하강법

❖ 기울기가 0인 점을 찾는 방법

1) a_1에서 미분값을 구한다.
2) 구해진 기울기의 반대 방향으로 얼마간 이동시킨 a_2에서 미분값을 구한다.
3) 위에서 구한 값이 0이 아니면 a_2에서 2)번 과정을 반복한다.
4) 그러면 그림처럼 이동 결과가 한 점으로 수렴함

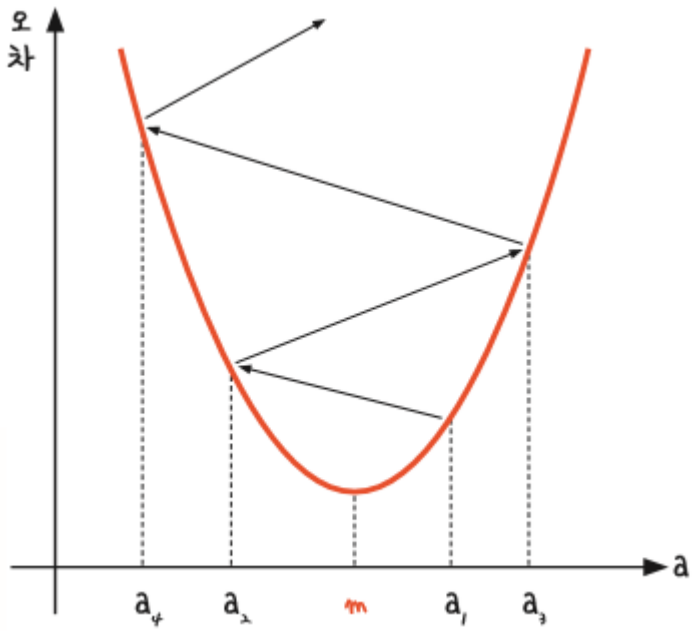
❖ 경사 하강법은 이렇게 반복적으로 기울기 a를 변화시켜서 m의 값을 찾아내는 방법



기울기가 0인 점 m을 찾는 방법

■ 학습률

- 기울기의 부호를 바꿔 이동시킬 때 적절한 거리를 찾지 못해 너무 멀리 이동시키면 a 값이 한 점으로 모이지 않고 위로 치솟아 버림
- 어느 만큼 이동시킬지를 정해주는 것
  → 학습률(Learning Rate)



학습률을 너무 크게 잡으면
한 점으로 수렴하지 않고 발산함

■ 코딩으로 확인하는 경사 하강법

▶ 03_경사하강법.ipynb

■ 다중 선형 회귀

- 4시간 공부한 친구는 88점을 예측했는데 이보다 좋은 93점을 받았고, 6시간 공부한 친구는 93점을 받을 것으로 예측했지만 91점을 받았음
  → 예측과 실제값에 차이가 있음

- 차이가 생기는 이유는 공부한 시간 이외의 다른 요소가 성적에 영향을 끼쳤기 때문
- 더 정확한 예측을 하려면 추가 정보를 입력해야 함
- 정보를 추가해 새로운 예측 값을 구하려면 변수의 개수를 늘려 '다중 선형 회귀'를 만들어 주어야 함

■ 다중 선형 회귀

- 예를 들어, 일주일 동안 받는 과외 수업 횟수를 조사해서 이를 기록해 보면,
  → 두 개의 독립 변수 x_1과 x_2가 생긴 것

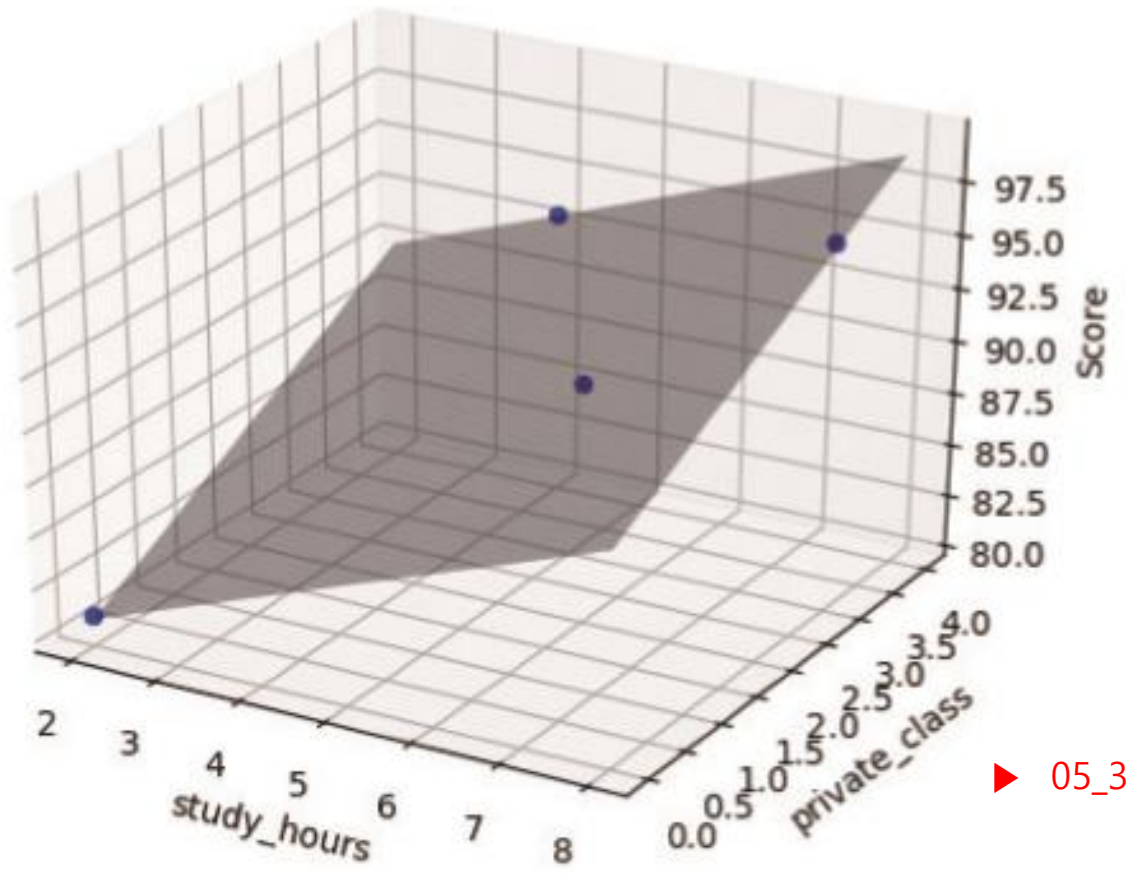
공부한 시간(x_1)	2	4	6	8
과외 수업 횟수(x_2)	**0**	**4**	**2**	**3**
성적(y)	81	93	91	97

- 이를 사용해 종속 변수 y를 만들 경우 기울기를 두 개 구해야 하므로 다음과 같은 식이 나옴

$$y = a_1x_1 + a_2x_2 + b$$

■ 코딩으로 확인하는 다중 선형 회귀

▶ 04_다중선형회귀.ipynb



▶ 05_3차원 그래프.ipynb

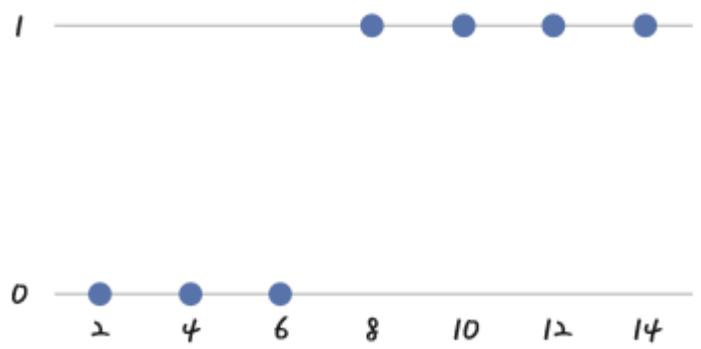
■ 개요

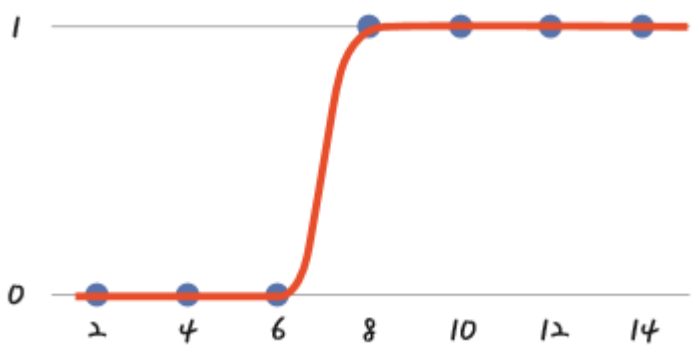
❖ 정의

- 참과 거짓중 하나의 값만을 판단

❖ 예

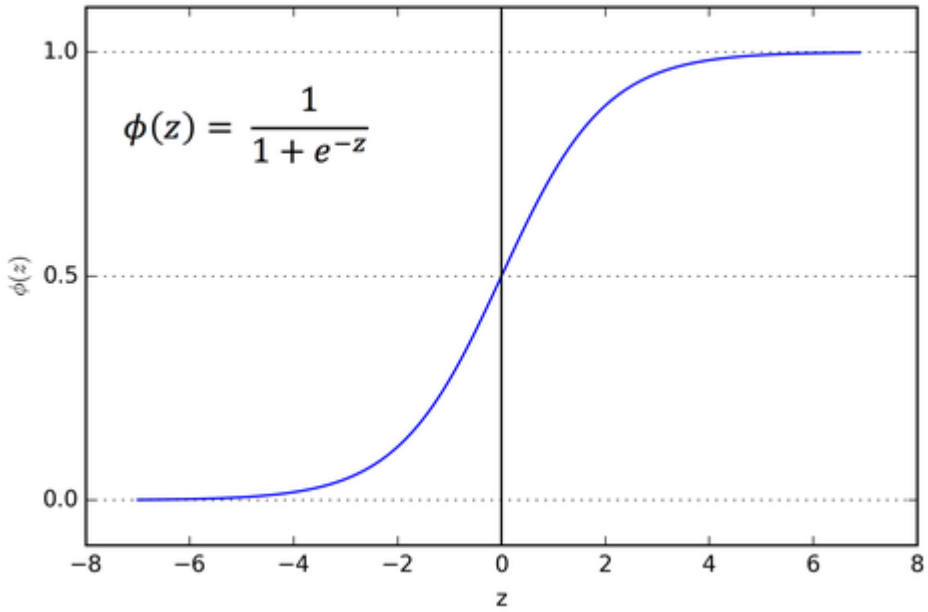
공부한 시간	2	4	6	8	10	12	14
합격 여부	불합격	불합격	불합격	합격	합격	합격	합격





■ 시그모이드 함수

- 시그모이드 함수(sigmoid function): S자 모양으로 그래프가 그려지는 함수
- 수식과 그래프



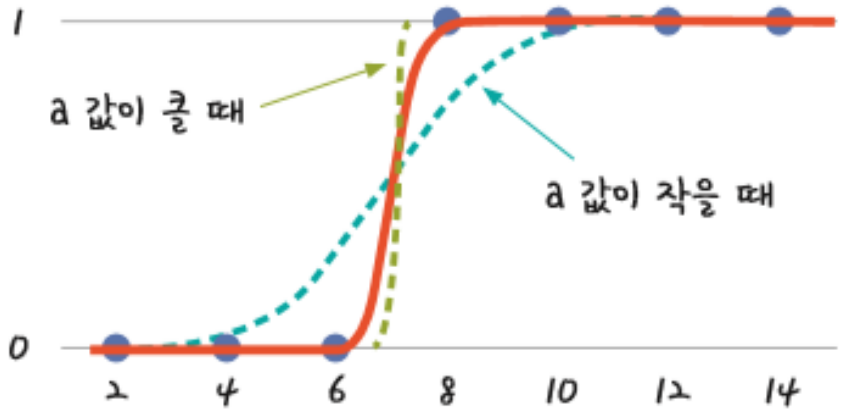
z = ax + b

▶ 06_Sigmoid.ipynb

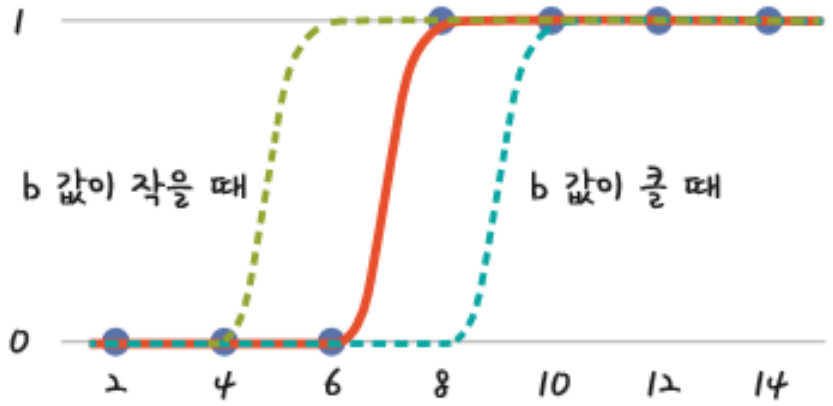
■ 시그모이드 함수

❖ a 값과 b 값의 변화에 따른 그래프 모양

- a 값이 커지면 경사가 커지고 a 값이 작아지면 경사가 작아짐



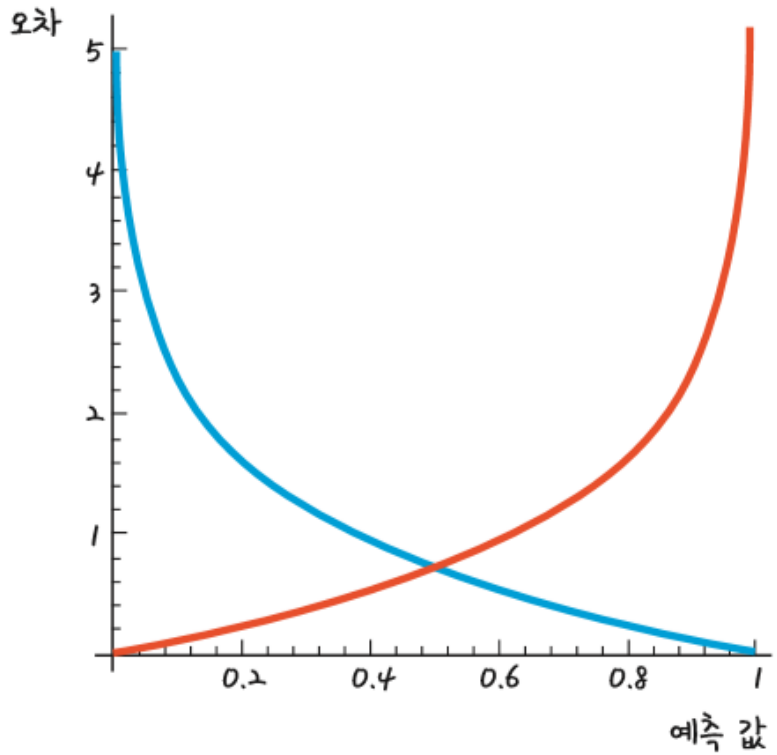
- b 값에 따라 그래프가 이동함



■ 시그모이드 함수

❖ 오차

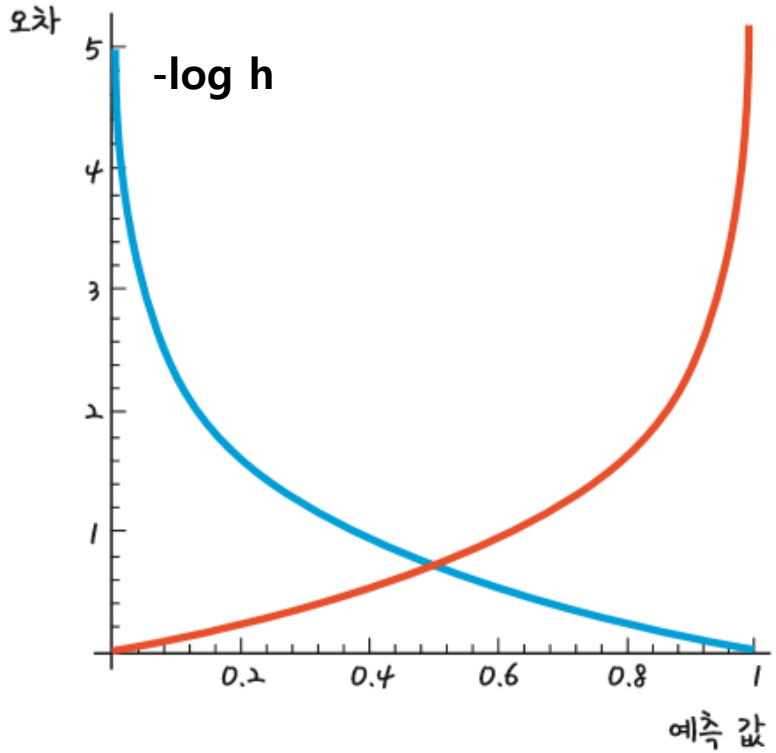
- 시그모이드 함수의 특징은 y 값이 0과 1 사이라는 것
- 따라서 실제 값이 1일 때 예측 값이 0에 가까워지면 오차가 커져야 함
- 반대로, 실제 값이 0일 때 예측 값이 1에 가까워지는 경우에도 오차는 커져야 함
- 이를 공식으로 만들 수 있게 해 주는 함수: 로그 함수



실제 값이 1일 때(파란색)와 0일 때(빨간색) 로그 함수 그래프

■ 시그모이드 함수

❖ 오차



▶ 07_오차 함수.ipynb

- y의 실제 값이 1일 때 –log h 그래프를 쓰고, 0일 때 –log (1 - h) 그래프를 써야 함

$$-\underbrace{\{y\log h}_{\text{A}}+\underbrace{(1-y)\log(1-h)\}}_{\text{B}}$$

■ 코딩으로 확인하는 로지스틱 회귀

▶ 08_로지스틱 회귀.ipynb

■ 코딩으로 확인하는 다중 로지스틱 회귀

▶ 09_다중 로지스틱 회귀.ipynb

❖ 실제 값 적용하기

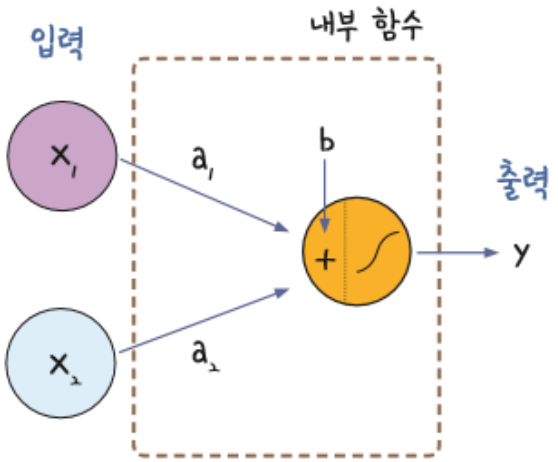
- 예를 들어 7시간 공부하고 과외를 6번 받은 학생의 합격 가능성은?
- 이를 실행하면 다음과 같이 출력됨

```
공부한 시간: 7, 과외 수업 횟수: 6
합격 가능성:  85.66 %
```

■ 로지스틱 회귀에서 퍼셉트론으로

❖ 퍼셉트론(Perceptron)

- 로지스틱 회귀를 퍼셉트론 방식으로 표현한 예



- x1과 x2가 입력되고, 각각 가중치 a1, a2를 만남
- 여기에 b 값을 더한 후 시그모이드 함수를 거쳐 1 또는 0의 출력 값 y를 출력

- 1957년, 코넬 항공 연구소의 프랑크 로젠블라트라는 사람이 발표