인공 지능과 기계 학습

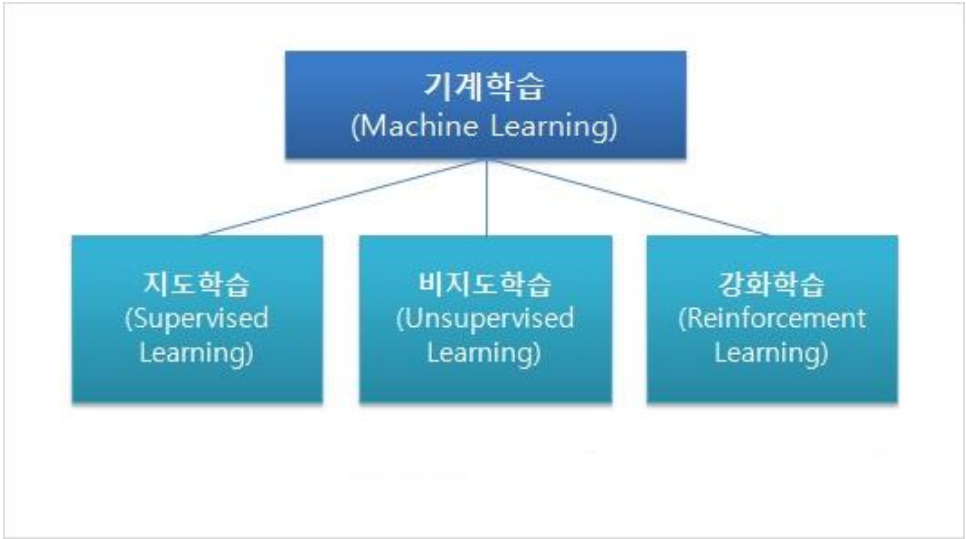
박 경수
전주대학교 게임콘텐츠학과

서론

- 지능
  - 상황을 인식하고 판단하는 기능
  - 자연 지능 / 인공 지능
- 인공 지능
  - 지능을 구현한 컴퓨터 시스템
  - 논리 기반 시스템 / 데이터(경험) 기반 시스템
- 기계 학습
  - 기계가 경험을 통하여 스스로 배우는 알고리즘
  - 인공 지능 구현의 한 방법

목차

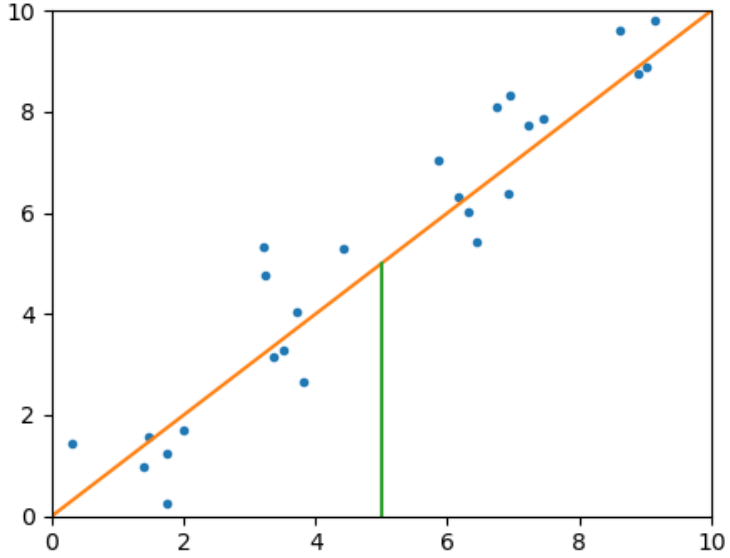
- 지도 학습
- 비지도 학습
- 강화 학습



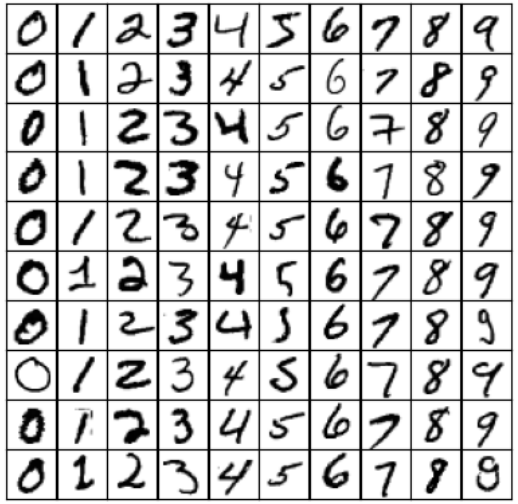
지도 학습(supervised learning)

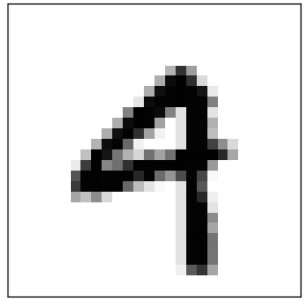
- 훈련 데이터로부터 함수를 유추
  - 출력을 알고 있는 데이터로 학습
  - 출력을 모르는 데이터의 출력을 유추

- 회귀(regression)
  - 우리 아이의 키는?
  - 투자의 손익



- 분류(classification)
  - 개, 고양이 판별
  - 글자 인식
  - 확률로 이해하고 회귀 문제로 접근





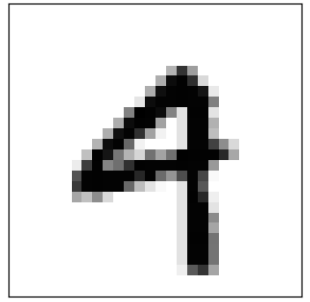
자료

- 실험이나 관측으로 얻은 자료

입력	X_1	X_2	$\cdots$	X_n
출력	Y_1	Y_2	$\cdots$	Y_n

- 자료에는 예측할 수 없는 다양한 잡음(noise)이 포함된다
  - 환경 요인, 측정 오차

- 입력 $X = (x_1, \ldots, x_p)$, 출력 $Y = (y_1, \ldots, y_q)$
- 예
  - 15세 자녀의 키
    - 입력: 엄마의 키, 아빠의 키, 가계 소득
    - 출력: 아들의 키
  - 내일의 날씨
    - 입력: 오늘의 온도, 습도, 바람의 속도, 바람의 방향
    - 출력: 내일 비가 올 확률
  - 숫자 인식
    - 입력: $p = 28 \times 28 = 784$pixel의 색상
    - 출력: $q = 10$, (숫자가 0일 확률, ..., 숫자가 9일 확률)



학습의 목적

- 가정:

$$Y = f(X) + \alpha$$

  - α: 잡음
- 목표:

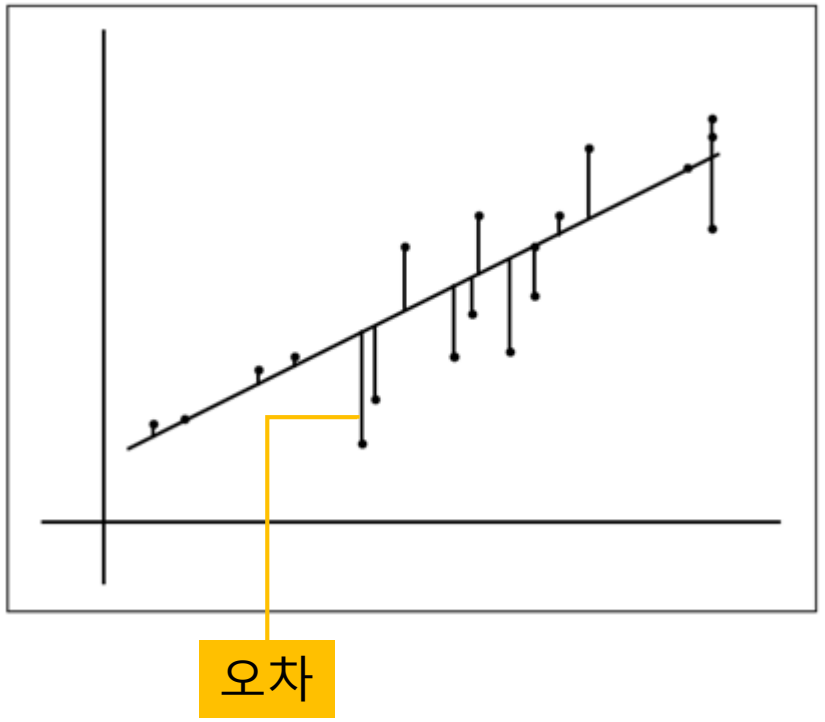
$f(X)$의 근사식 $\hat{f}(X)$ 찾기

통계적 방법

- $\hat{f}(X)$의 형태를 추정
  - 일차 함수, 다항 함수, …
- 최소 제곱법
  - 제곱 오차(squared error)

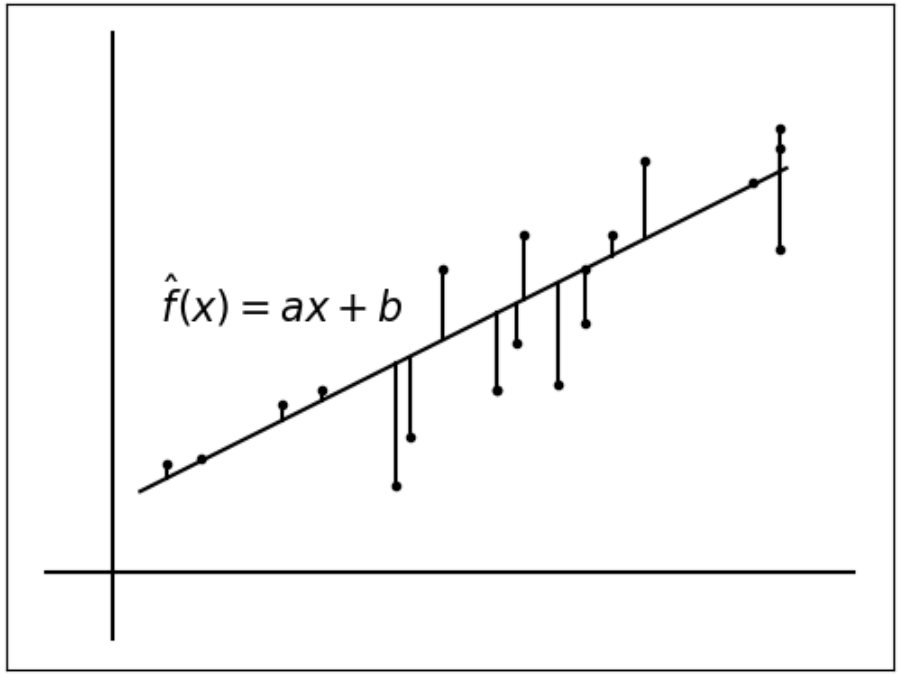
$$L = \sum_{i=1}^{n} \| Y_i - \hat{f}(X_i) \|^2$$

  - 오차를 최소로 하는 $\hat{f}(X)$를 구한다



보기

x	0.08	0.13	⋯	0.99
y	0.16	0.17	⋯	0.66



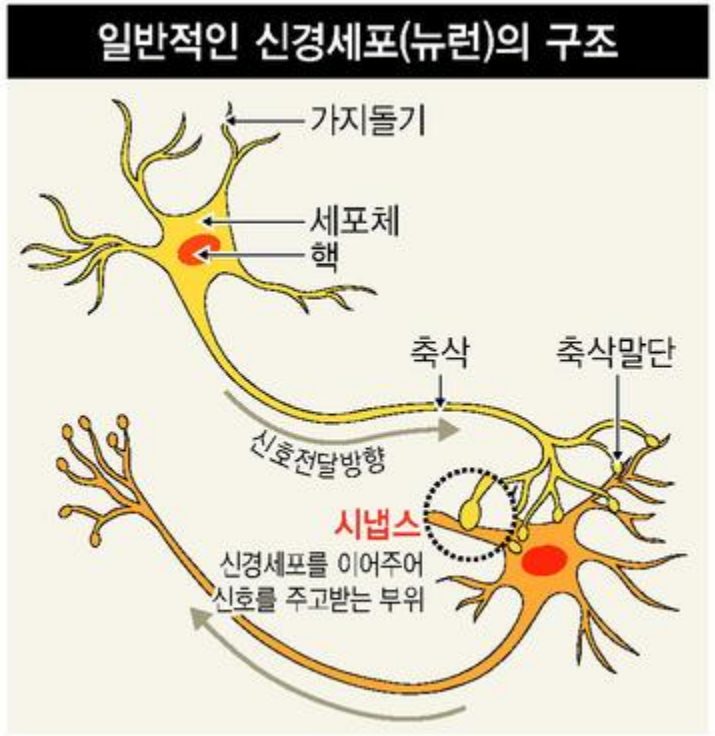
- $\hat{f}(x) = ax + b$라 추정
- 오차의 제곱의 합

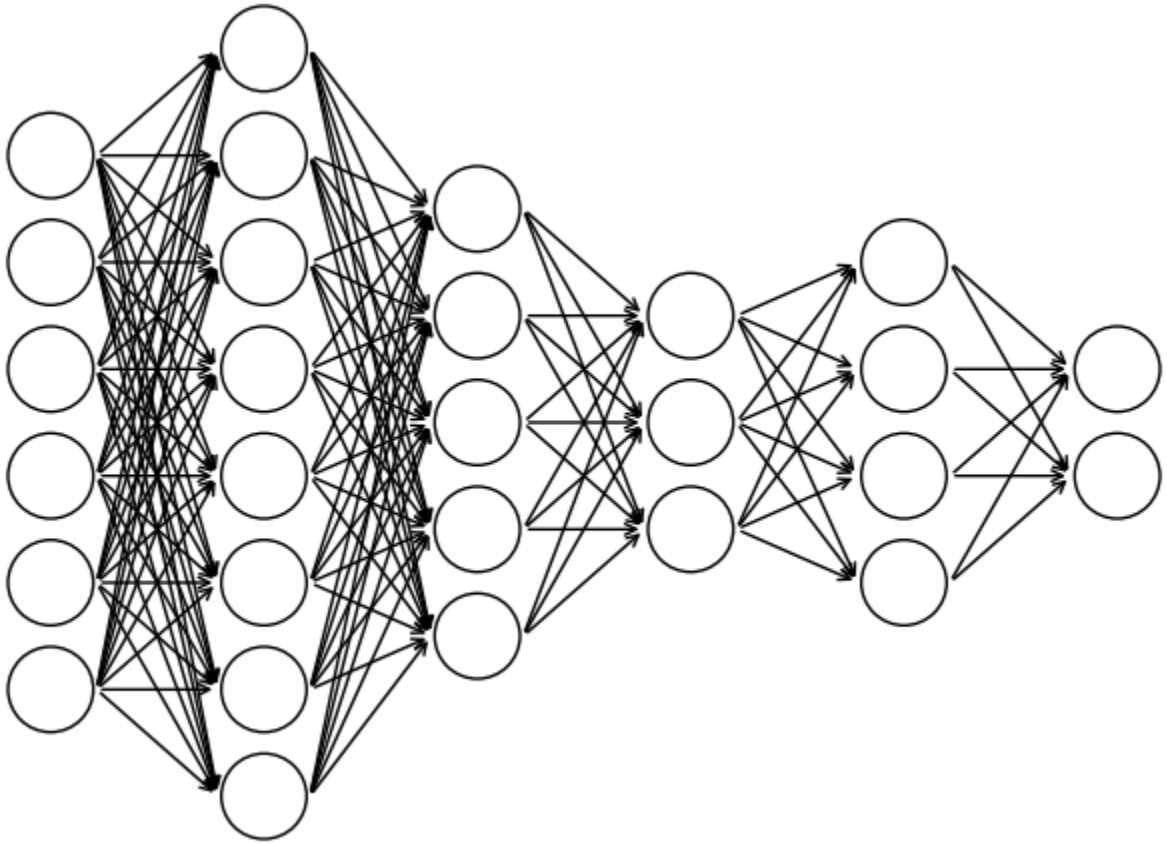
$$L = (0.16 - 0.08a - b)^2 + \cdots + (0.66 - 0.99a - b)^2$$

- L을 최소로 하는 a와 b를 구한다

인공 신경망(artificial neural network)

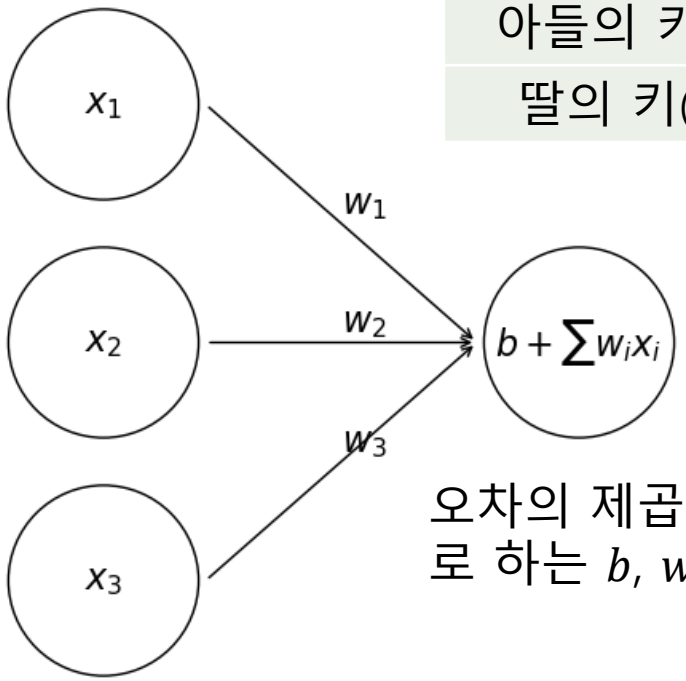
- 신경망을 구성하여 학습





최소 제곱법의 인공 신경망 표현

아빠의 키(x_1)	0.17	0.16	⋯	0.18
엄마의 키(x_2)	0.15	0.16	⋯	0.16
가계 소득(x_3)	0.30	0.45	⋯	0.28
아들의 키(y_1)	0.18	0.17	⋯	0.19
딸의 키(y_2)	0.16	0.16	⋯	0.17



오차의 제곱의 합을 최소로 하는 b, w_1, w_2, w_3는?

아들의 키

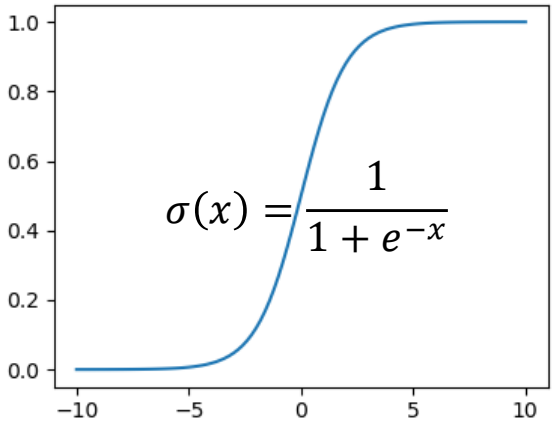
$b = 0$, $w_1 = 0.6$, $w_2 = 0.3$, $w_3 = 0.1$이라 하면

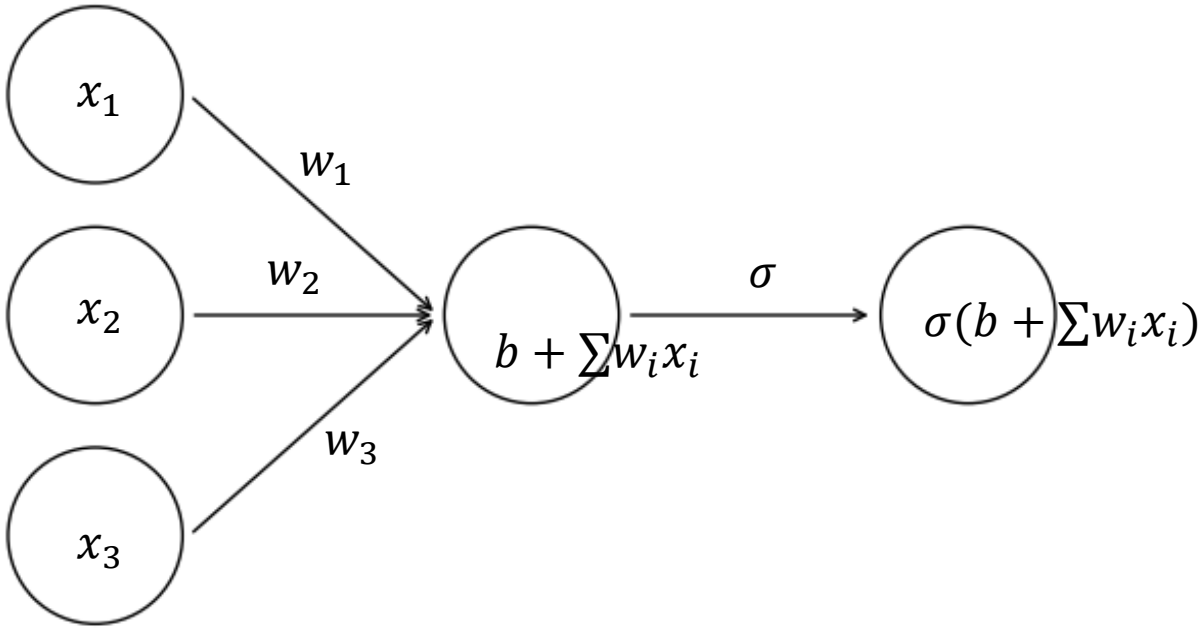
$0.17 \cdot w_1 + 0.15 \cdot w_2 + 0.30 \cdot w_3 = 0.177$

오차: 0.003

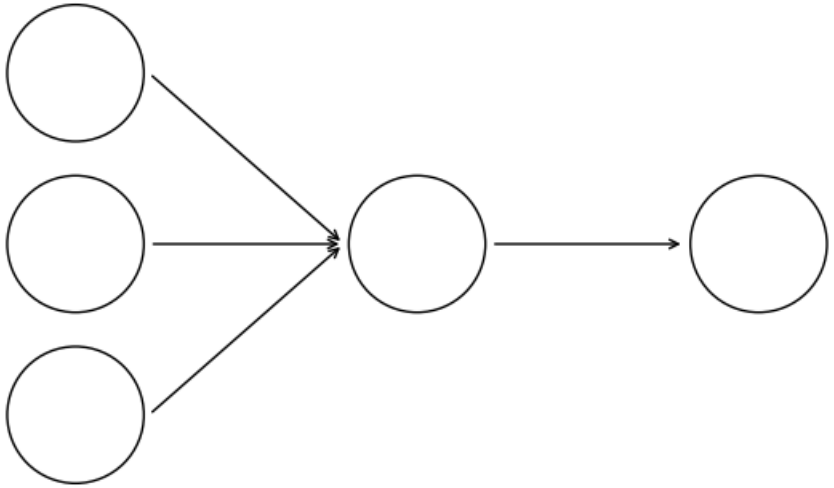
단층 퍼셉트론 – 초기 모델

- w_i: 각 요소의 비중
- b: 상수항
- σ: 활성 함수
  - 비선형, 증가, 미분가능

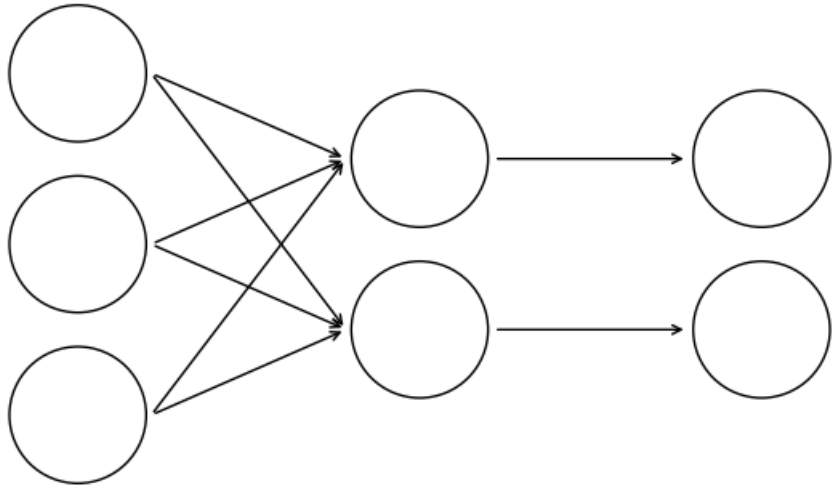




아빠의 키(x_1)	0.17	0.16	…	0.18
엄마의 키(x_2)	0.15	0.16	…	0.16
가계 소득(x_3)	0.30	0.45	…	0.28
아들의 키(y_1)	0.18	0.17	…	0.19
딸의 키(y_2)	0.16	0.16	…	0.17



아들의 키



아들의 키와 딸의 키

목표

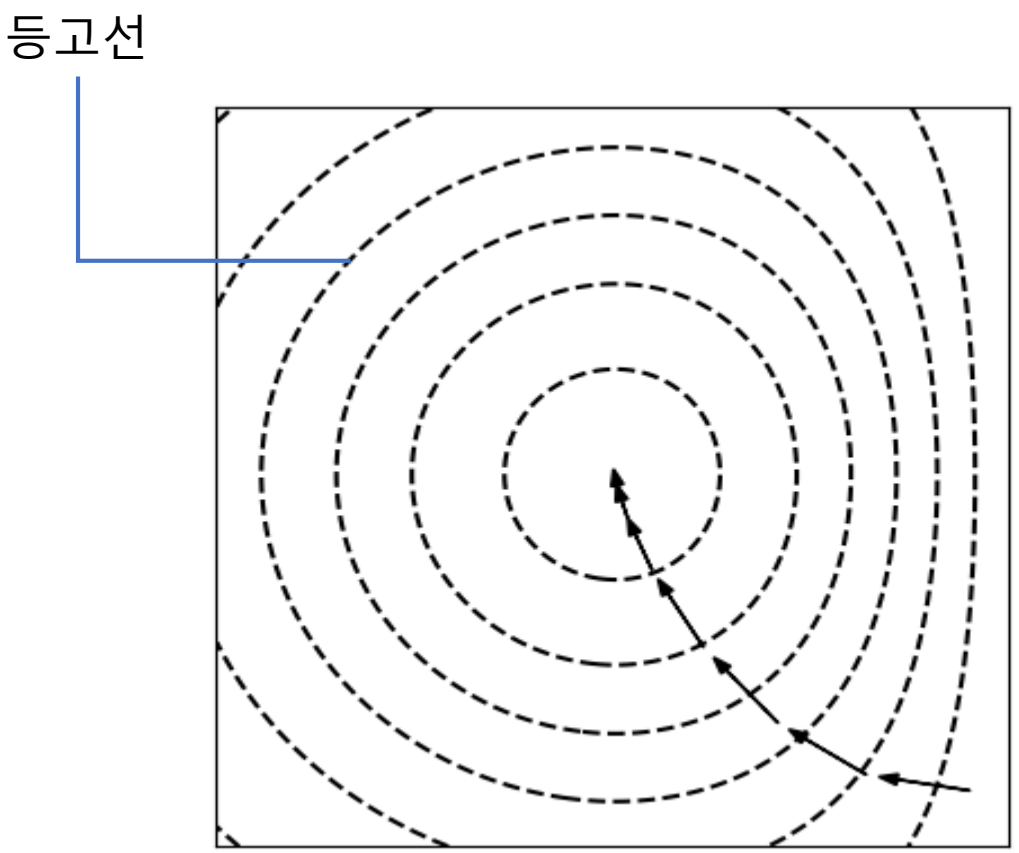
- y_1을 가장 잘 근사시키는 b와 w_i는?

x_1	x_{11}	x_{12}	$\cdots$	x_{1n}
x_2	x_{21}	x_{22}	$\cdots$	x_{2n}
x_3	x_{31}	x_{32}	$\cdots$	x_{3n}
y_1	y_{11}	y_{12}	$\cdots$	y_{1n}

- $\hat{y}_{1i} = \sigma(b + w_1 x_{1i} + w_2 x_{2i} + w_3 x_{3i})$
- 오차 $|y_{1i} - \hat{y}_{1i}|$
- $L = \frac{1}{2}\sum_{i=1}^{n}(y_{1i} - \hat{y}_{1i})^2$을 최소로 하는 b와 w_i

경사 하강법(gradient descent)

- 함수 f 의 극소점
- 알고리즘
  - $p \leftarrow p - \lambda \cdot \nabla f(p)$를 반복
  - $\nabla f(p)$는 f의 그래디언트



그래디언트(gradient)

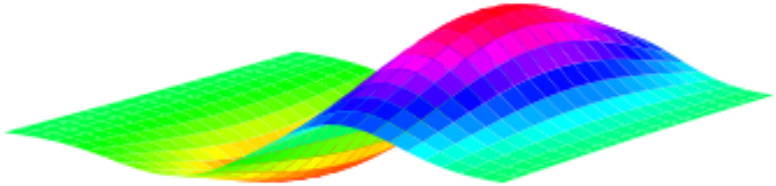
- 함수 $f: \mathbb{R}^n \to \mathbb{R}$
- 점 $x = (x_1, \ldots, x_n)$에서 f 의 그래디언트

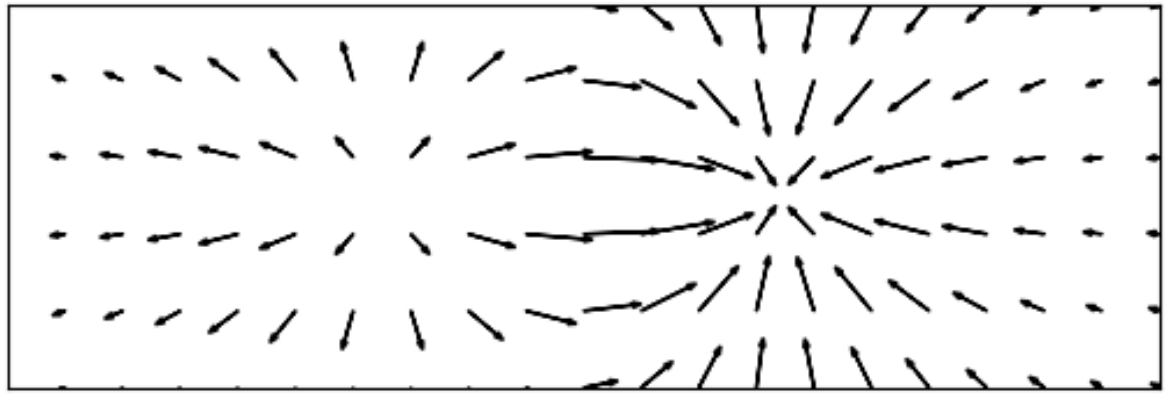
$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right)$$

  - 함수가 가장 빨리 증가하는 방향
- 보기:

$$f(x_1, x_2) = x_1^2 x_2$$
$$\nabla f(x_1, x_2) = (2x_1 x_2, x_1^2)$$

- 그래디언트 벡터
  - $z = f(x, y)$

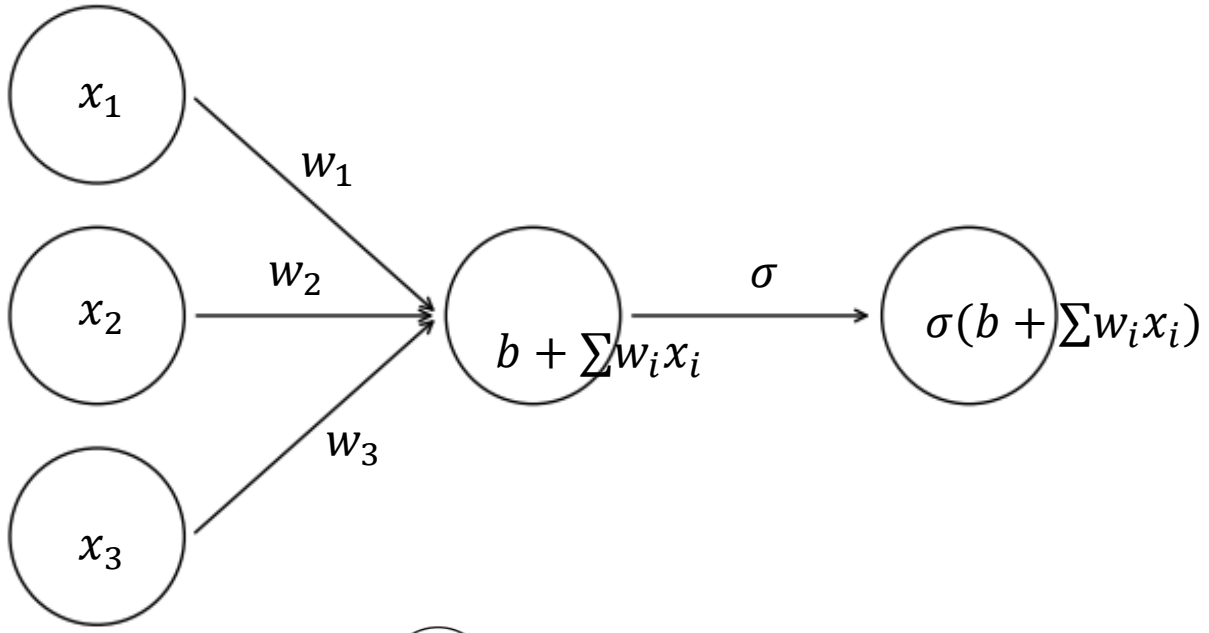


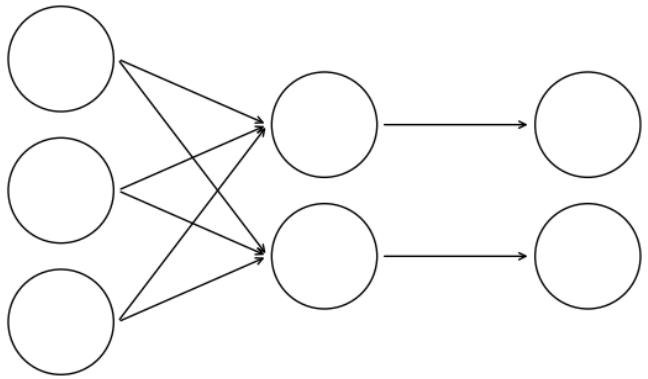


- 경사 하강법
  - $p \leftarrow p - \lambda \cdot \nabla f(p)$

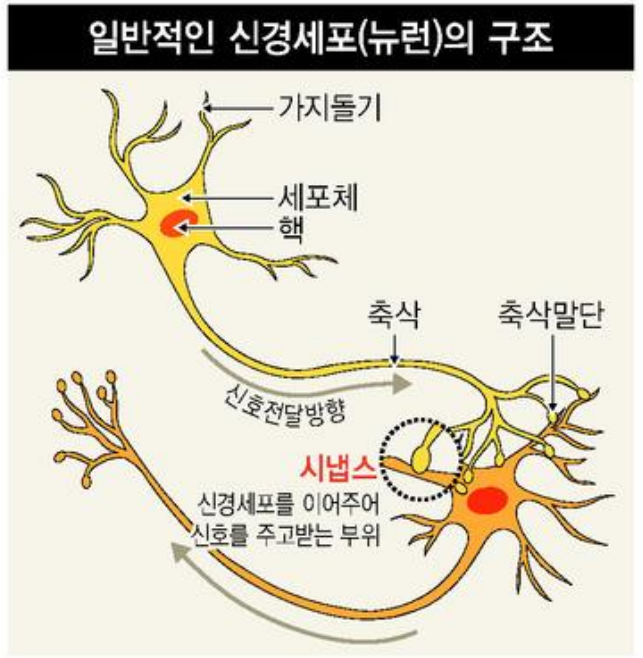
단층 퍼셉트론

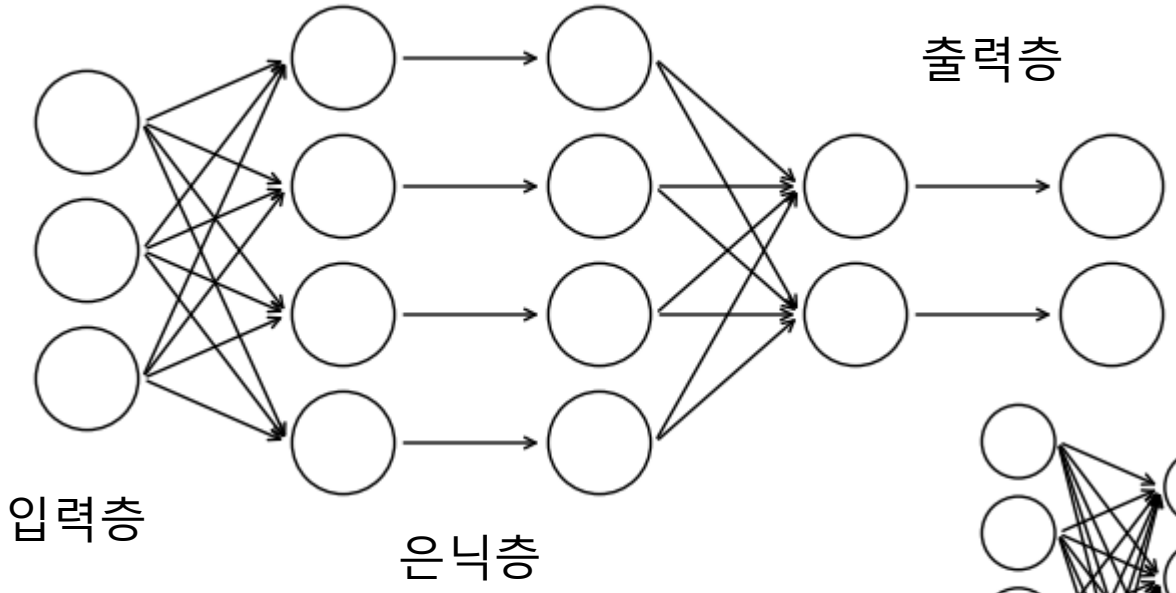
- 훈련 집합
  $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
- 변수
  b, w_1, w_2, w_3
- 근사식
  $$\hat{Y}_j = \sigma\left(b + \sum w_i x_{ij}\right)$$
  $$= \sigma(b + WX_j)$$
- 오차의 제곱의 합
  $\sum_j \| \sigma\left(b + \sum_i w_i x_{ij}\right) - Y_j \|^2$
- 경사 하강법

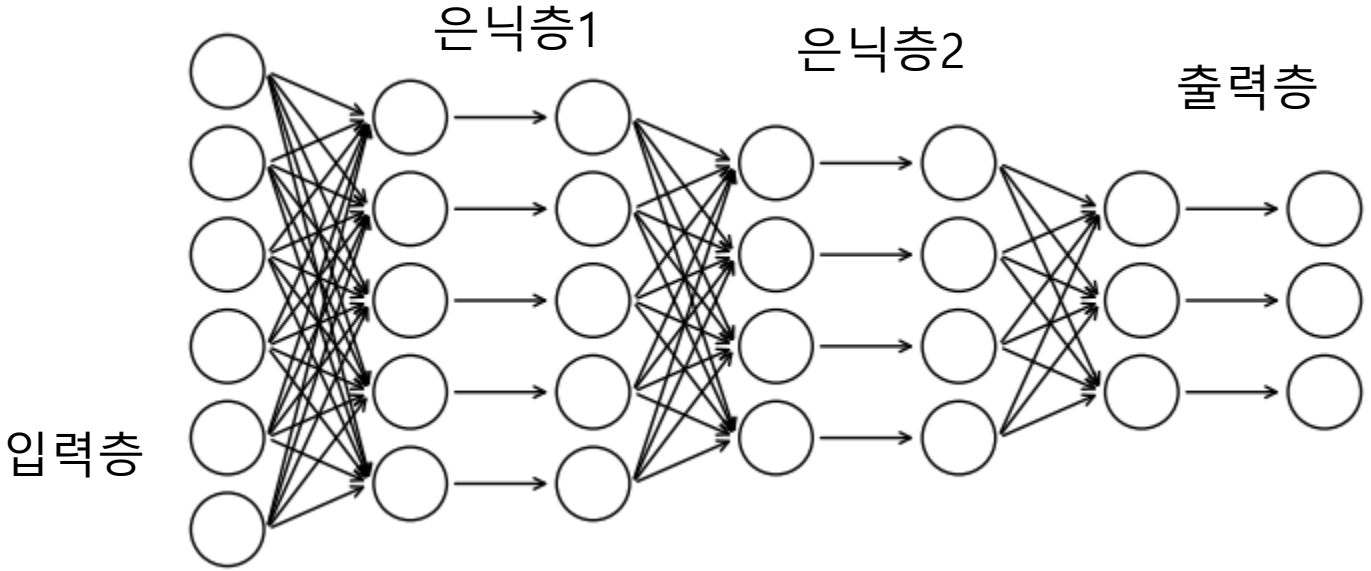




다층 퍼셉트론


일반적인 신경세포(뉴런)의 구조
가지돌기
세포체
핵
축삭
축삭말단
신호전달방향
시냅스
신경세포를 이어주어
신호를 주고받는 부위


출력층
입력층
은닉층


은닉층1
은닉층2
출력층
입력층

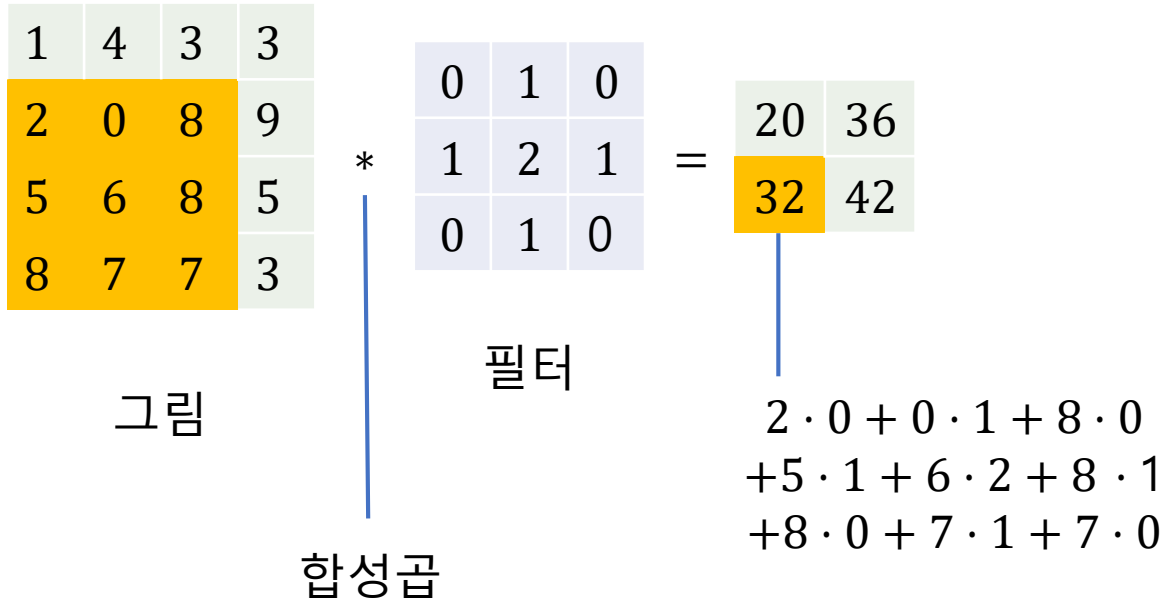
심층 학습(deep learning)

- 은닉층이 여럿인 신경망을 사용하는 기계 학습
- 다층 퍼셉트론을 포함한 다양한 층을 포함
- CNN, RNN

이미지 인식(분류)

- 합성곱 신경망(CNN, convolutional neural network)
  - 심층 학습
  - 합성곱(convolution)을 사용
  - 낮은 오류율

합성곱(convolution)



0	0	0
1	1	1
0	0	0

수평선 찾기

0	1	0
0	1	0
0	1	0

수직선 찾기

1	0	0
0	1	0
0	0	1

대각선 찾기

0	0	1
0	1	0
1	0	0

대각선 찾기

*

0.2	0.2	0.2	0.2	0.2
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

=

0.2	0	0	0	0
0.2	0	0	0	0
0.2	0	0	0	0
0.2	0	0	0	0
0.2	0	0	0	0

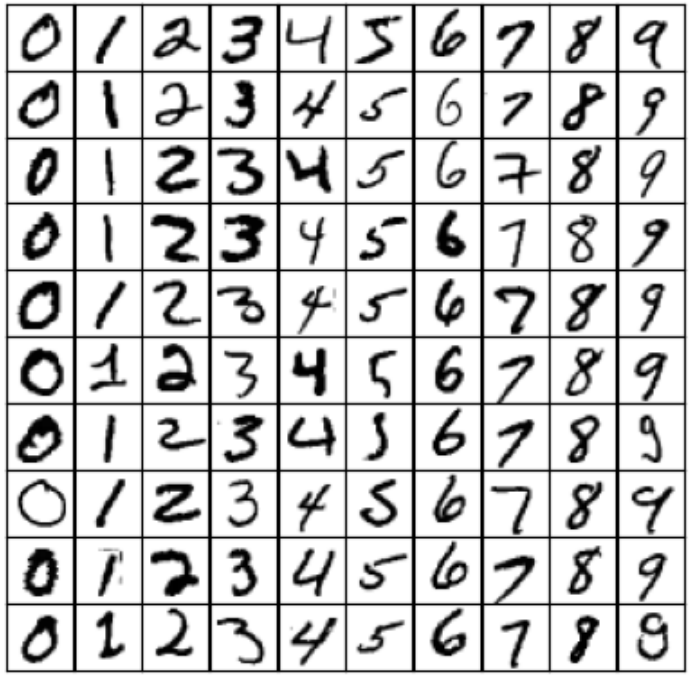
=

합성곱 신경망 (CNN, convolutional neural network)

- 합성곱에 의한 정보 추출 과정

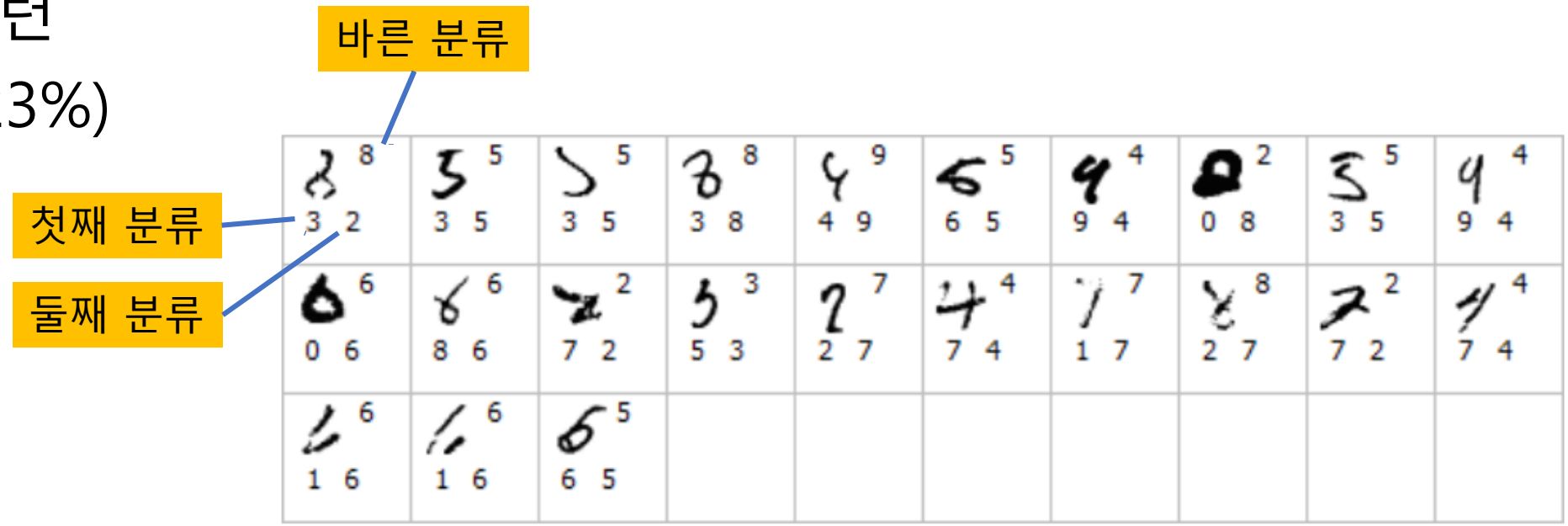
MNIST

- NIST(National Institute of Standards and Technology)가 미국 우편번호에서 수집한 손글씨
  - 28×28 pixel
  - 훈련용 – 6만개
  - 검증용 – 1만개
- 분석 역사
  - 통계적 방법 - 99.48%(2007)
  - 인공 신경망 – 99.65%(2010)
  - 합섭곱 신경망 – 99.77%(2012)

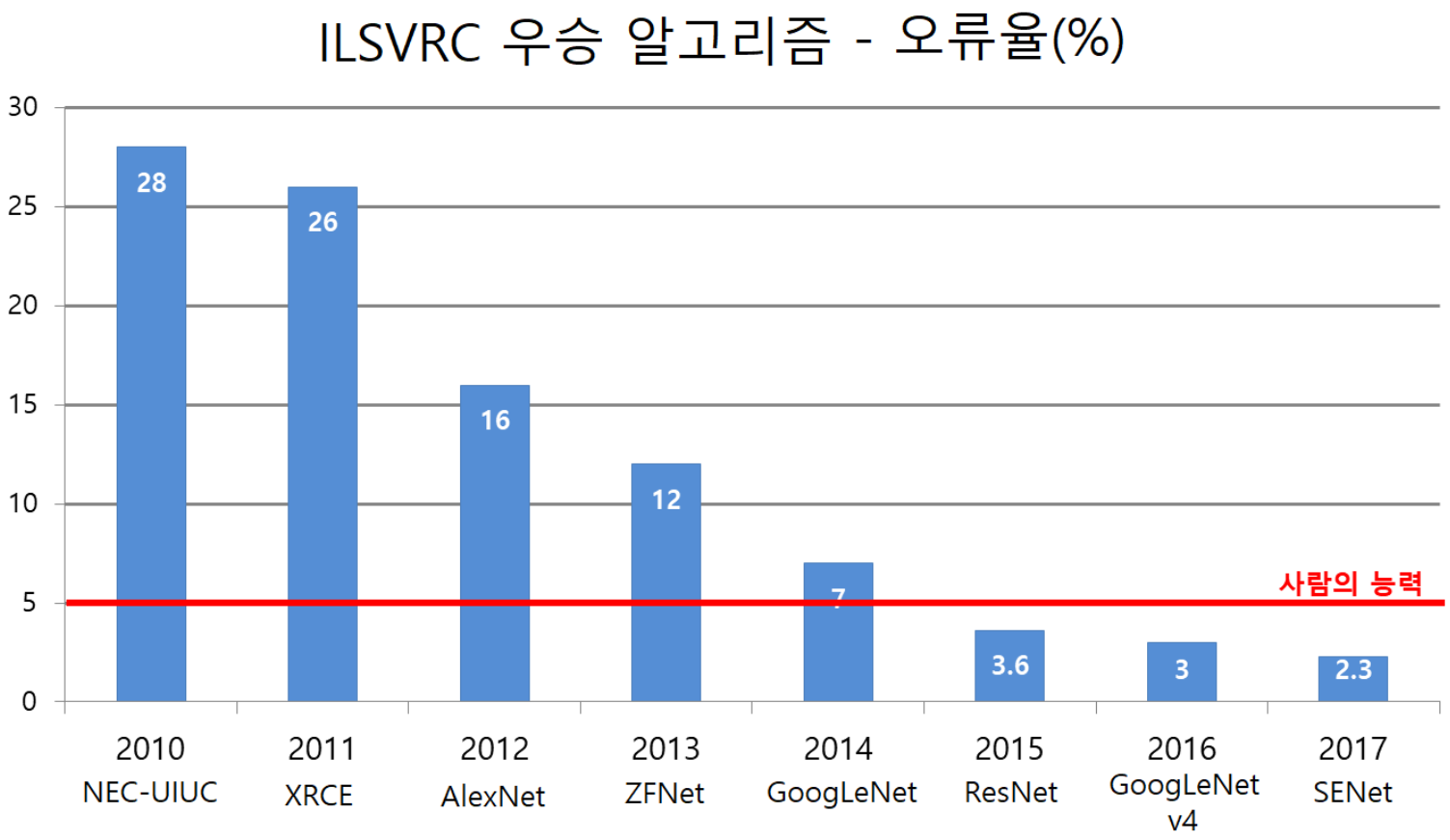


MCDNN

- 합성곱 신경망
- 1x29x29-20C4-MP2-40C5-MP3-150N-10N DNN
  - 변수 개수 ≈ 18만
- 15시간 훈련
- 오류율(0.23%)

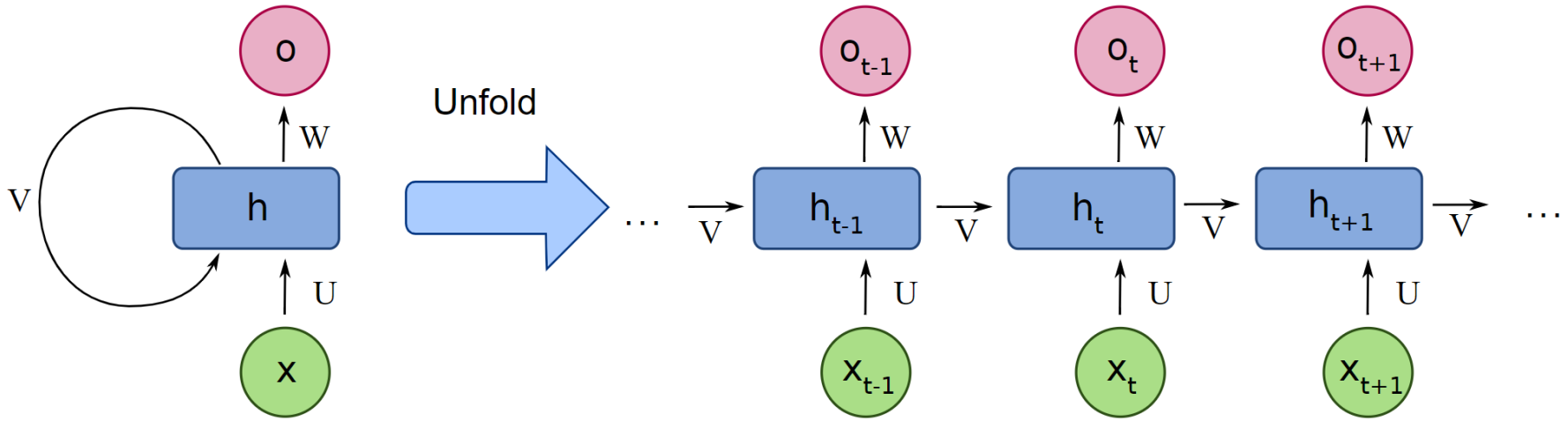


이미지 인식률


ILSVRC 우승 알고리즘 - 오류율(%)
30
25
20
15
10
5
0
28
26
16
12
7
3.6
3
2.3
사람의 능력
2010
NEC-UIUC
2011
XRCE
2012
AlexNet
2013
ZFNet
2014
GoogLeNet
2015
ResNet
2016
GoogLeNet
v4
2017
SENet

RNN(recurrent neural network)

- 작곡, 소설



인공 신경망의 장단점

- 장점
  - 정확도가 높다
  - 스스로 학습한다.
- 단점
  - 너무 복잡하여 인간이 이해하기 어렵다
    - ResNet은 변수가 6천만개인 함수의 극소점을 찾는다
  - 오류가 있어도 이유를 알기 어렵다
  - 비용(하드웨어, 시간)이 너무 커서 접근이 어렵다
    - 이미 발표된 네트워크는 따라해 볼 수 있으나
    - 새로운 네트워크 개발에는 시간의 제약이 따른다

관련 분야

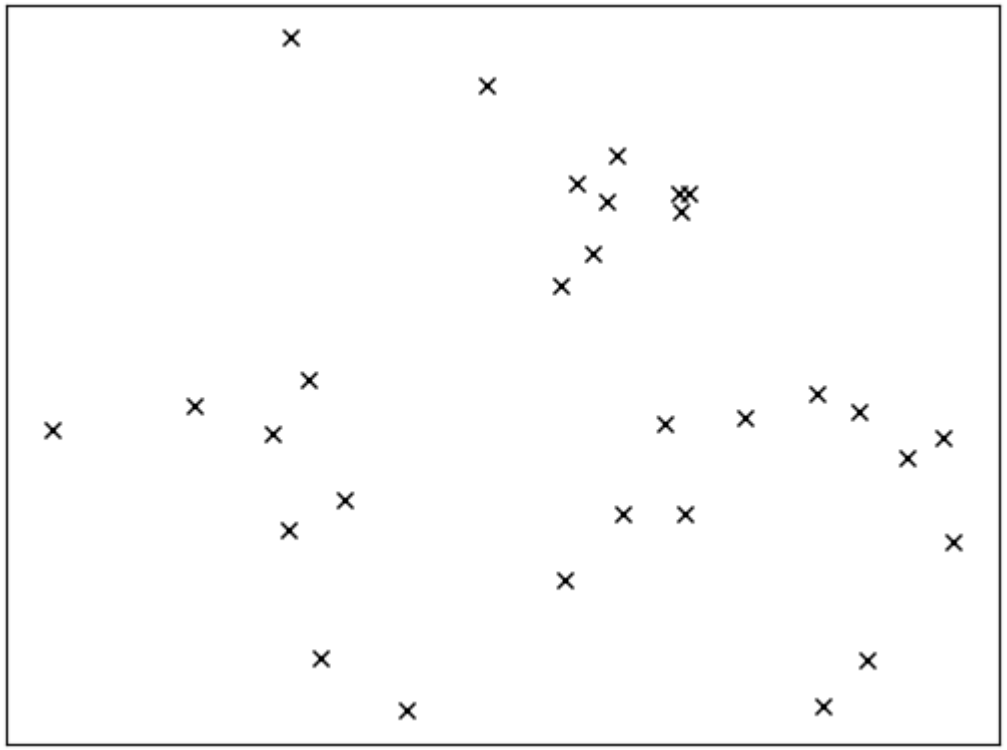
- 인공 신경망의 이해 및 설계
  - 미분, 벡터, 행렬, 확률, 통계
  - 컴퓨터 프로그래밍
- 인공 신경망을 이용한 자료 분석
  - 컴퓨터 프로그래밍
  - 약간의 수학적인 지식
  - 다양한 분야에서 이용되고 있음

인공 신경망의 문제들

- 네트워크의 이해
  - 구조 및 동작
- 인간의 개입을 최소화하는 자동화 알고리즘
  - 현재는 인간이 구조를 설계하고 컴퓨터는 계산만 한다
- 알고리즘의 진화
  - 성능이 좋은 알고리즘으로 스스로 진화

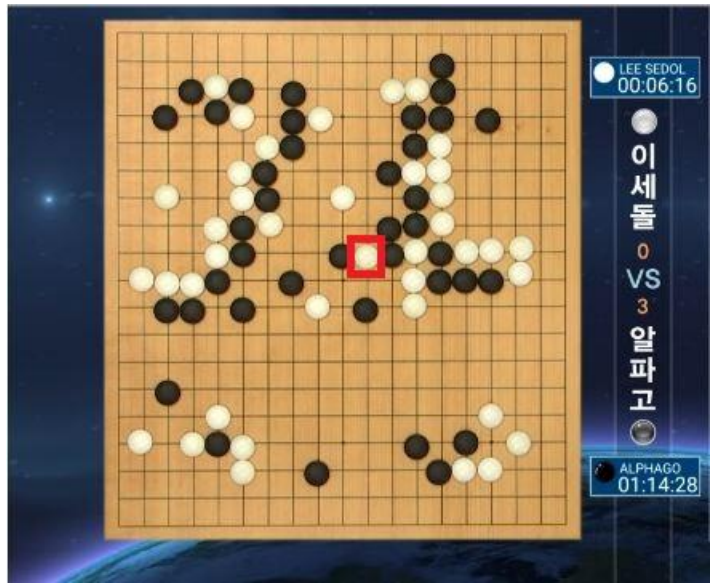
비지도 학습(unsupervised learning)

- 미지의 자료를 그룹으로 분류



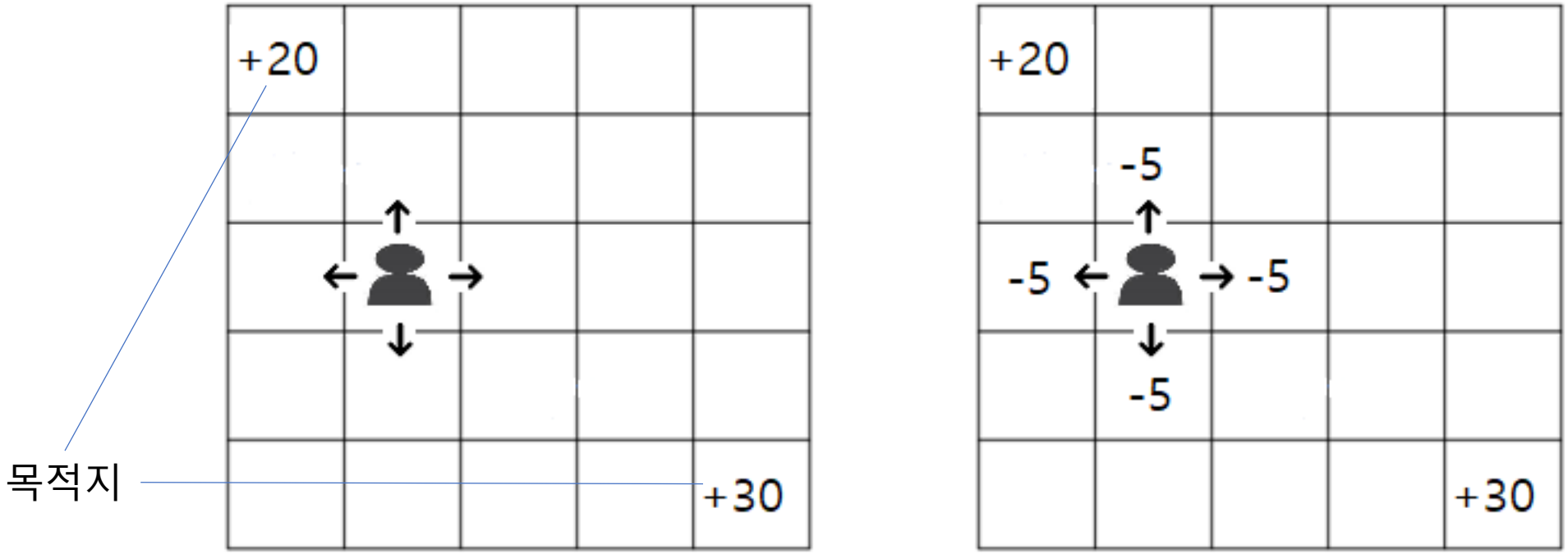
강화 학습(reinforcement learning)

- 최적의 보상을 얻는 행동을 찾아가는 기계 학습 방법
  - Cart Pole
- 실제 경험이나 모의 실험으로 획득한 자료로 반복 학습
  - 올해는 밭에 무엇을 심을까?
  - 우산을 가지고 갈까?
  - 자동차는 어느 방향으로?
  - 어디에 돌을 놓을까?



격자 세계

- 이득이 가장 큰 방향은 어디인가?

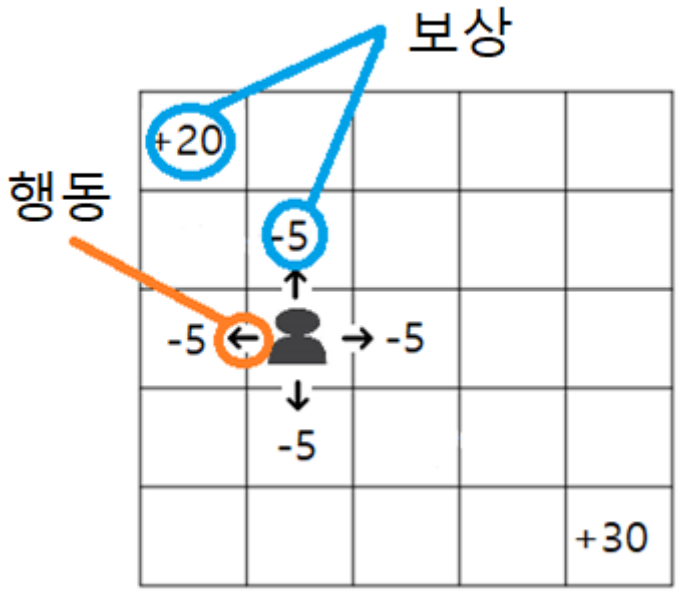


행위자와 환경

- 행위자(agent)
  - 행동 주체
  - 각 상태에서 행동 선택
- 환경(environment)
  - 행위자의 행동에 대한 보상 제공
  - 행위자의 행동에 대하여 다음 상태 결정

변수

- 상태(state)
- 행동(action)
- 보상(reward)
  - 행동에 대한 보상

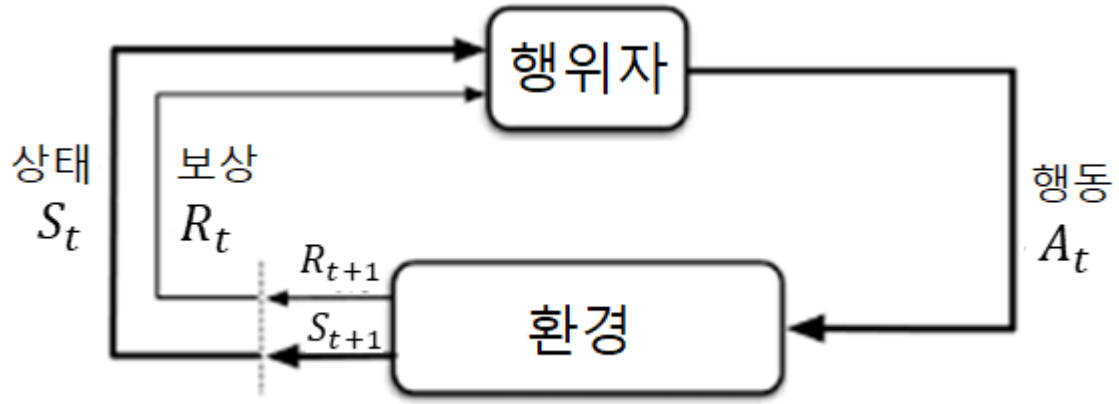


예

- 올해는 밭에 무엇을 심을까?
  - 상태 – 예상 강수량, 예상 기온, 가격
  - 행동 – 작물 선택
  - 보상 – 수확(소득) / 비용(손해)
- 어디에 돌을 놓을까?
  - 상태 – 돌이 놓인 위치
  - 행동 –
  - 보상 – 종료 후 승패
- 자동차는 어느 방향으로?

마르코프 결정 과정 (MDP, Markov decision process)

- 의사결정을 위한 수학적 모델
- 현 상태와 행동이 다음 상태와 보상을 결정
- $S_0 \xrightarrow{A_0} R_1, S_1 \xrightarrow{A_1} R_2, S_2 \longrightarrow$
  - S_t - 상태
  - A_t - 행동
  - R_t - 보상



MDP

- $\mathcal{S}$ – 상태의 집합
- $\mathcal{A}$ – 행동의 집합
  - $\mathcal{A}(s)$ – 상태 s에서 취할 수 있는 행동의 집합
- $\mathcal{R}$ – 보상의 집합
  - $\mathcal{R}(s,a)$ – 상태 s에서 행동 a를 취했을 때 얻는 보상의 집합
- $p(s',r \mid s,a) = \Pr(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a)$
  - 상태 s에서 행동 a를 취했을 때 상태 s'이 되고 보상 r을 얻을 확률

참고
대문자: 변수
소문자: 값

에피소드(episode)

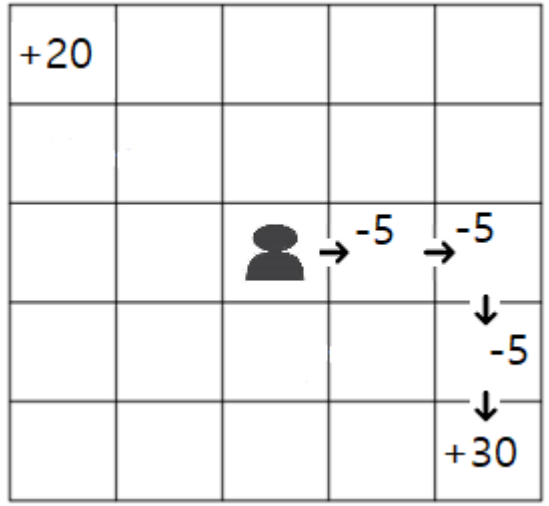
- 행위자와 환경의 상호작용이 완료되는 과정
- $S_0 \xrightarrow{A_0} R_1, S_1 \xrightarrow{A_1} R_2, S_2 \xrightarrow{A_2} \cdots \xrightarrow{A_{T-1}} R_T, S_T$

반환(return)

- 각 타임 스텝에서 에피소드가 종료될 때까지 얻는 보상의 총합
- $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$
  - γ – 할인율(discount rate)
    - 미래에 얻을 이익을 현재 가치로 환산하기 위한 비율
    - $0 < \gamma \leq 1$

격자 세계

- 에피소드
  - $S_0 \xrightarrow{A_0} R_1, S_1 \xrightarrow{A_1} R_2, S_2 \xrightarrow{A_2} R_3, S_3 \xrightarrow{A_3} R_4, S_4$
- 할인율
  - $\gamma = 0.9$
- 반환
  - $G_0 = R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \gamma^4 R_5$
    $= -5 + 0.9 \cdot (-5) + 0.81 \cdot (-5) + 0.729 \cdot 30$
    $= 8.32$
  - $G_1 = ?$



정리

- 상태, 행동, 보상, 에피소드, 반환, 할인율
- 확률적으로 결정되는 것들은 무엇인가?

- 올해는 밭에 무엇을 심을까?
  - 콩을 10개 심으면 몇 개의 싹이 틀까?
- 어디에 돌을 놓을까?
  - 전에 비슷한 상황이 있었는데 이번에도 결과가 같을까?

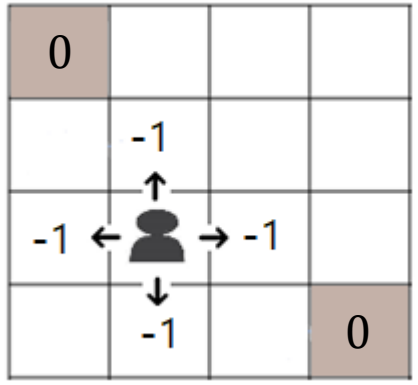
정책(policy)

- 행위자가 행동을 선택하는 방법
- $\pi(a|s) = \Pr(A_t = a \mid S_t = s)$
  - 상태 s에서 정책 π를 취할 확률
- 예
  - 결정적 정책
    - $\pi(a|s) = 0$ or 1
  - 랜덤 정책
    - $\pi(a|s)$가 균일

상태 가치 함수

- 상태 가치
  - 정책에 의하여 행동을 선택할 경우 각 상태의 가치
- 정책 π에 대한 상태 가치 함수
  - $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
  - 에피소드의 반환의 기댓값

격자 세계



$\gamma = 1$

- 랜덤 정책을 취할 때 각 상태의 가치

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

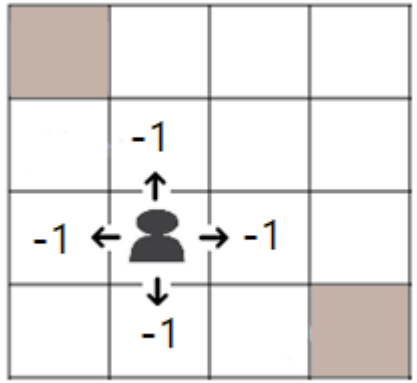
- 최적의 정책을 취할 때 각 상태의 가치

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

벨만 방정식(Bellman equation)

- 벨만 방정식

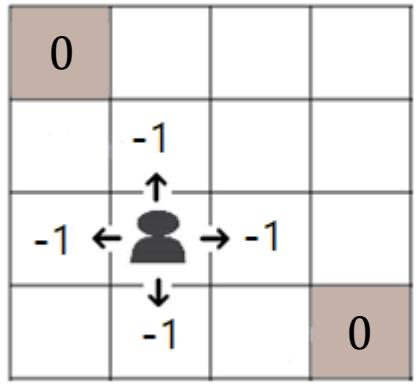
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$



최적 가치 함수와 최적 정책

- 최적 가치 함수
  - 상태의 최대 가치
  - $v_*(s) = \max_{\pi} v_\pi(s)$

- 최적 정책
  - 가장 우수한 정책 π_*
  - $v_{\pi_*}(s) = v_*(s)$



$\gamma = 1$

최적 가치

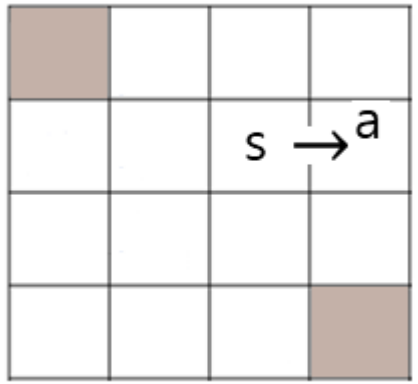
0	−1	−2	−3
−1	−2	−3	−2
−2	−3	−2	−1
−3	−2	−1	0

Dynamic Programming

- 최적화 문제를 해결하는 것
  - 가장 우수한 정책 찾기
- 장점
  - 거의 정확한 값을 계산할 수 있다
- 단점
  - 모든 경우의 수를 생각해야 한다
    - 시간의 한계
    - 컴퓨터 메모리의 한계
    - 바둑에서 상태의 개수 ≈ 361!

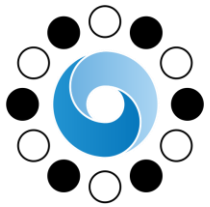
몬테 카를로 방법
Monte Carlo Methods

- v_π의 근삿값
  - 표본을 추출하여 평균으로 계산



$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$
모든 에피소드에 대한 반환의 평균을 계산

알파고



AlphaGo

- 가치가 큰 수(행동)을 선택
- 모든 수를 시도해 볼 수는 없다
- 이길 확률이 높은 그림과 유사한 그림을 만드는 수들을 시도해 본다
  - 합성곱 신경망
- 역사
  - 알파고 리(2016)
    - 이세돌과 대국
  - 알파고 마스터(2017)
    - 프로기사와 대국에서 60연승
  - 알파고 제로(2017)
    - 기존 기보에 대한 지식없이 바둑 규칙만으로 스스로 발전

강화 학습의 현재

- 상태의 가치를 모두/정확히 알 수 없다
  - 상태가 너무 많다
- 어떤 행동을 시도해 볼 것인가?
  - 확률에 의하여 결정된다
  - 불확실성이 존재한다