

인공 지능과 데이터 사이언스

박 경수

전주대학교 게임콘텐츠학과

인공 지능

- 지능이란?
 - 자연 지능 / 인공 지능
- 인공 지능
 - 지능을 구현한 컴퓨터 시스템
 - 논리 기반 시스템 / 경험(자료) 기반 시스템
- 기계 학습
 - 기계가 경험을 통하여 스스로 배우는 것
 - 지도 학습 / 비지도 학습 / 강화 학습

목차

- 지도 학습
- 비지도 학습
- 강화 학습

지도 학습

supervised learning

- 회귀(regression)
 - 내일 비가 올 확률
 - 우리 아이의 키는?
 - 투자의 손익
- 분류(classification)
 - 사진 속 꽃의 종 판별
 - 글자 인식

자료

- 실험이나 관측으로 얻은 자료
- 입력과 출력

x	x_1	x_2	\dots	x_n
y	y_1	y_2	\dots	y_n

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

목표

- 자료

$$\{(x_1, y_1), \dots, (x_n, y_n)\}$$

- 훈련 집합(training set)

- 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
 - 최소 제곱 오차

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

- L 을 최소로 하는 $\hat{f}(x)$ 를 구한다

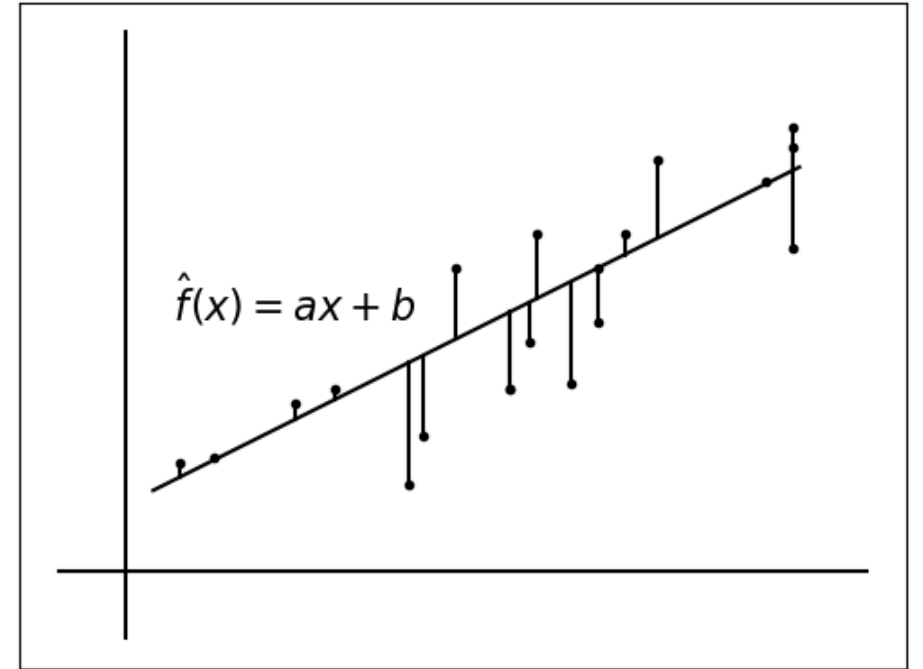
보기

x	0.08	0.13	...	0.99
y	0.16	0.17	...	0.66

- $\hat{f}(x) = ax + b$ 라 추정
- L 은

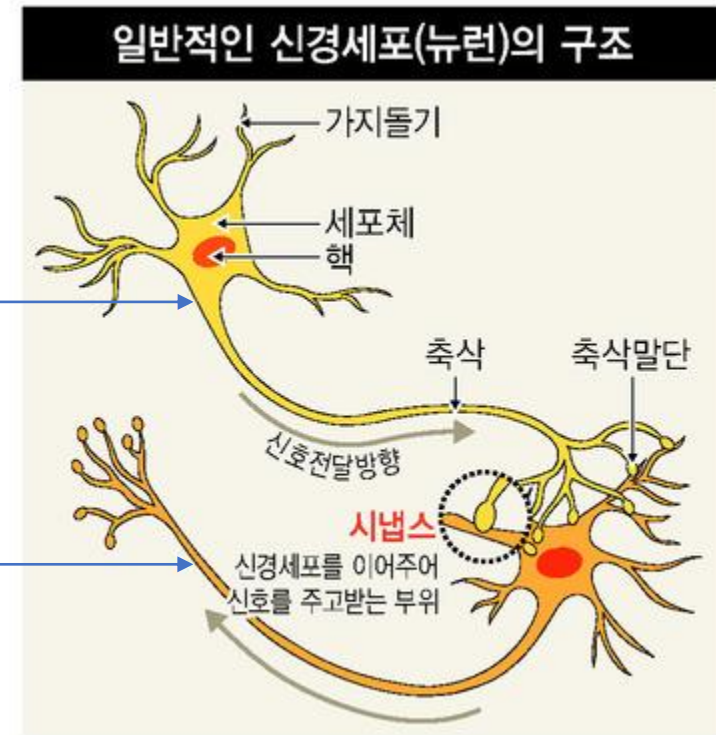
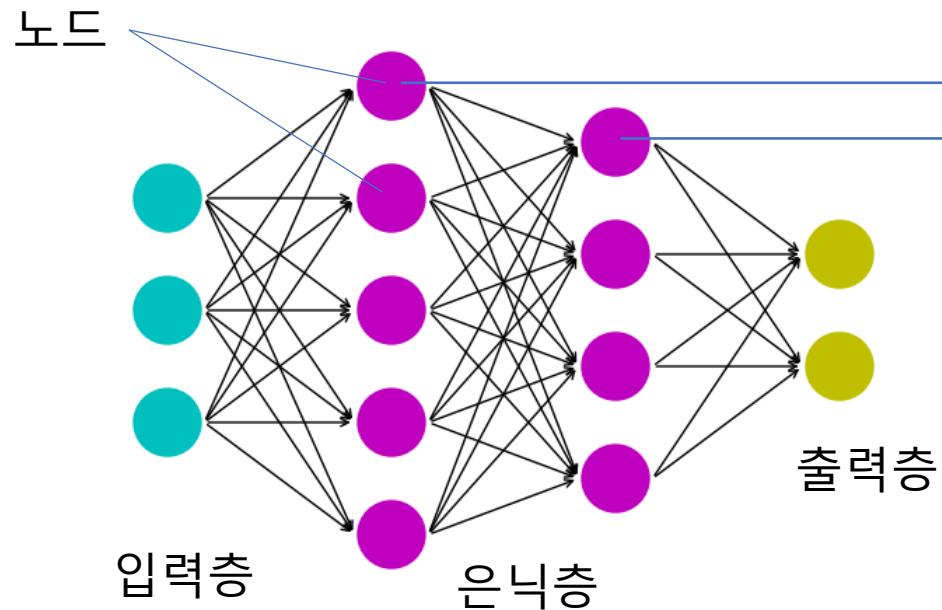
$$L = (0.16 - 0.08a - b)^2 + \dots + (0.66 - 0.99a - b)^2$$

- L 을 최소로 하는 a 와 b 를 구한다



순방향 신경망 Feedforward Neural Network

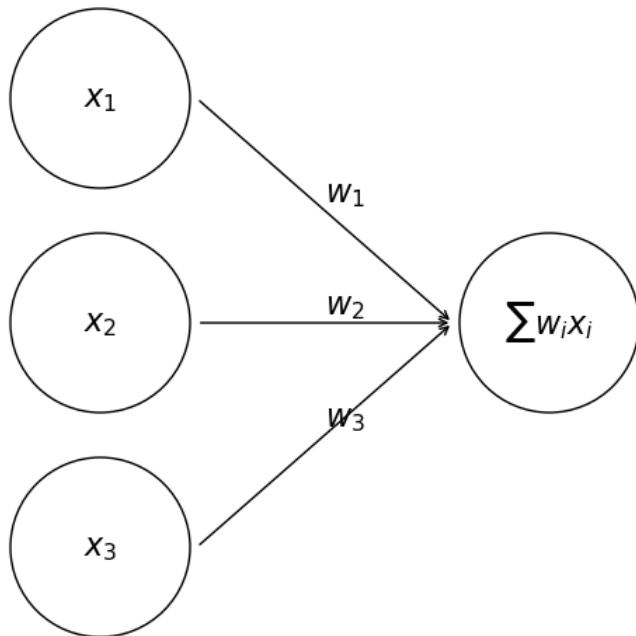
- 순환이 없는 신경망



출처: 한겨레

자녀의 키

아빠의 키(x_1)	0.173	0.165	...	0.180
엄마의 키(x_2)	0.156	0.159	...	0.165
가계 소득(x_3)	0.300	0.450	...	0.280
아들의 키(y_1)	0.179	0.175	...	0.180
딸의 키(y_2)	0.160	0.165	...	0.158



아들의 키

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

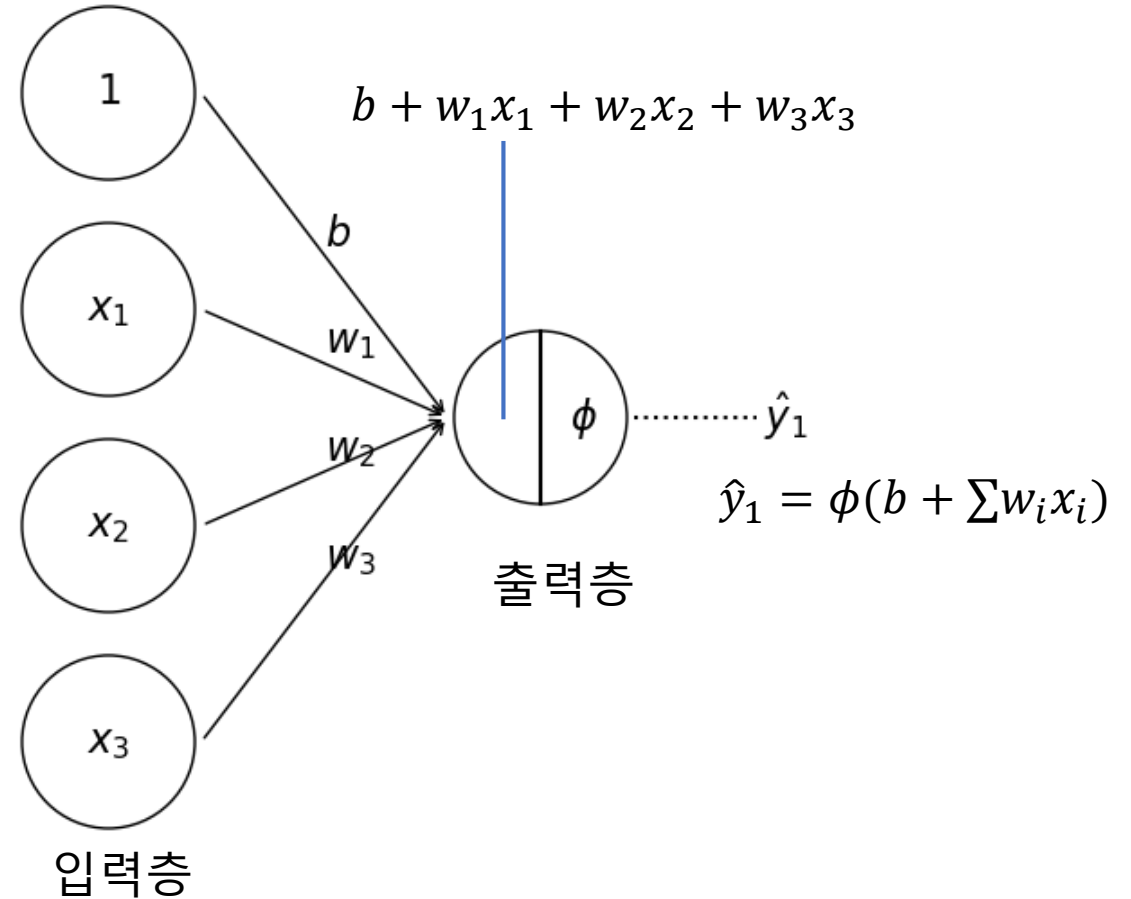
$$w_1 \cdot 0.173 + w_2 \cdot 0.156 + w_3 \cdot 0.300 = 0.1806$$

오차: 0.0016

오차의 제곱의 합을 최소로 하는 w_1, w_2, w_3 는?

단층 퍼셉트론

- 은닉층이 없는 신경망
 - w_i : 가중치
 - 각 요소의 비중
 - b : 바이어스(bias, 편향)
 - 상수항
 - ϕ : 활성화 함수
 - 비선형, 증가, 미분가능
 - \hat{y}_j : 출력



활성함수

- 시그모이드 함수

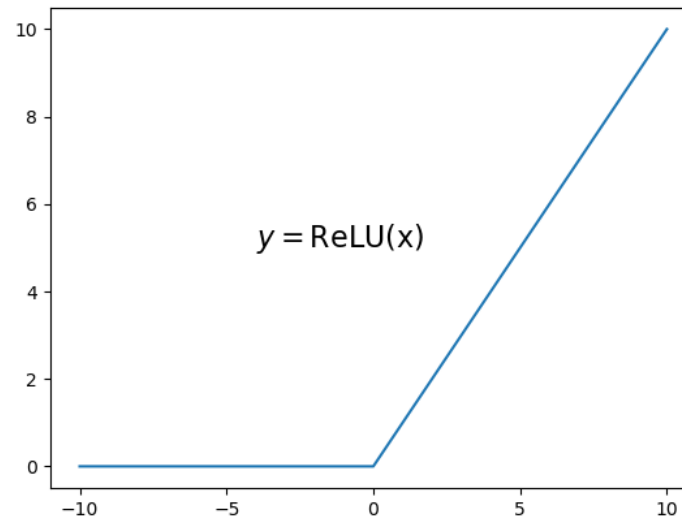
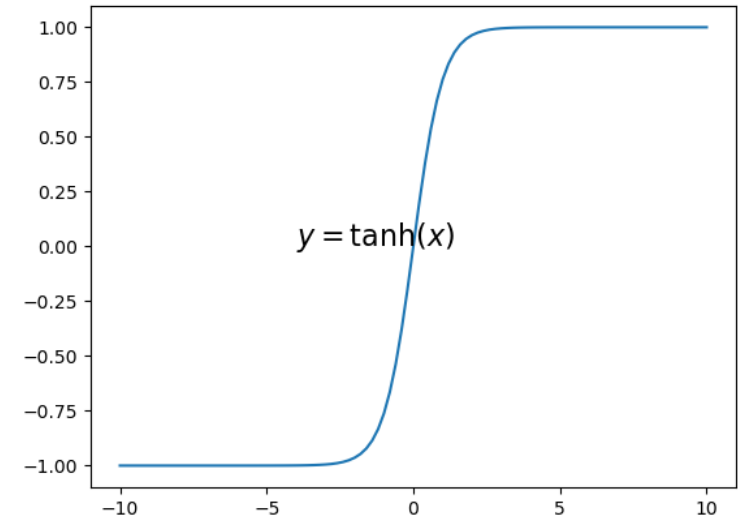
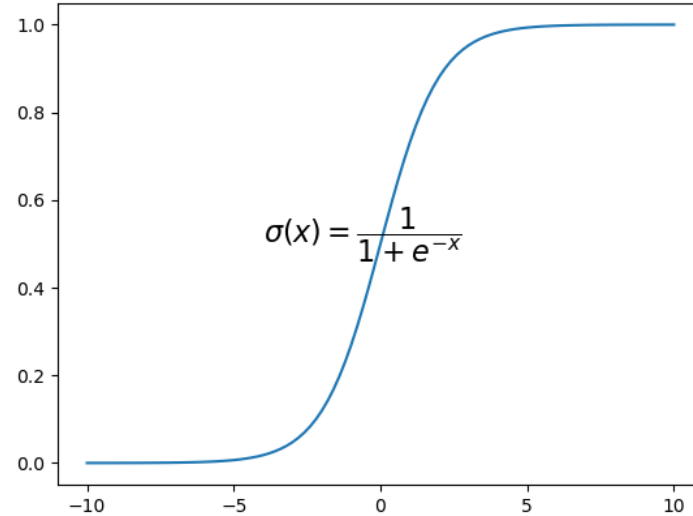
- $\sigma(x) = \frac{1}{1+e^{-x}}$

- 쌍곡 탄젠트

- $\tanh(x)$

- Rectifier

- $\text{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$



목표

- 아들의 키를 가장 잘 근사시키는 바이어스 b 와 가중치 w_i 는?

아빠의 키(x_1)	x_{11}	x_{12}	...	x_{1n}
엄마의 키(x_2)	x_{21}	x_{22}	...	x_{2n}
가계 소득(x_3)	x_{31}	x_{32}	...	x_{3n}
아들의 키(y_1)	y_{11}	y_{12}	...	y_{1n}

- $\hat{y}_{1i} = \phi(b + w_1x_{1i} + w_2x_{2i} + w_3x_{3i})$
- 오차: $L = \frac{1}{2} \sum_{i=1}^n (y_{1i} - \hat{y}_{1i})^2$

그래디언트 Gradient

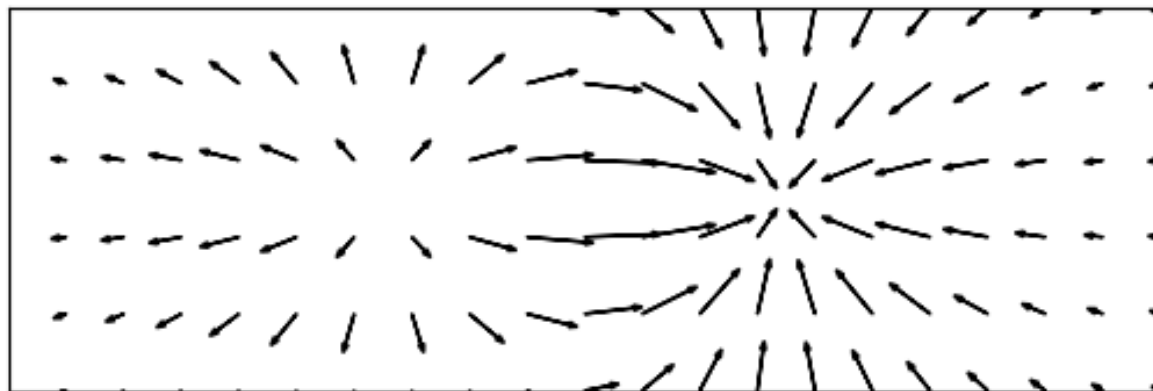
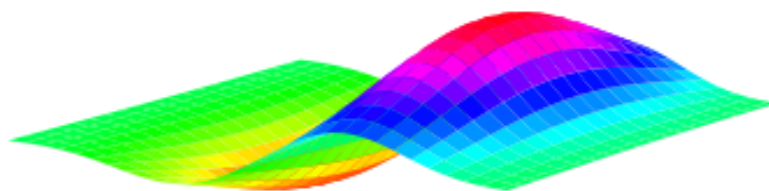
- 함수 $f: \mathbb{R}^n \rightarrow \mathbb{R}$
- 점 $x = (x_1, \dots, x_n)$ 에서 f 의 그래디언트

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

- 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

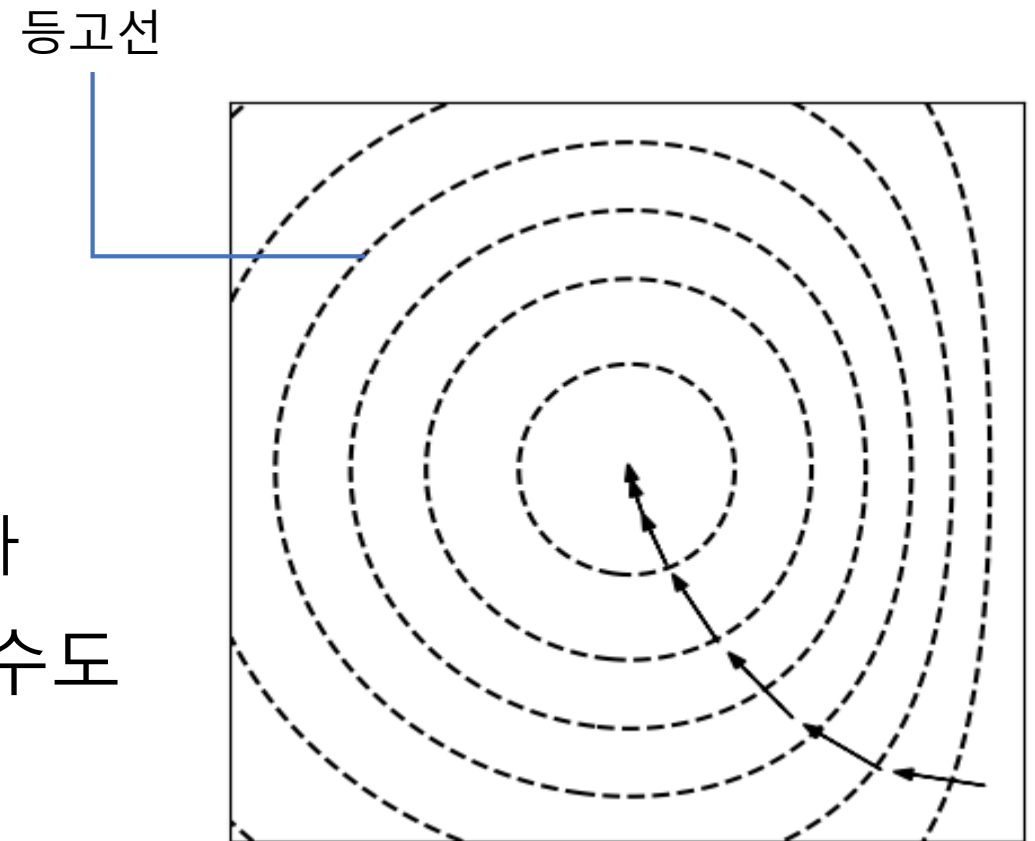
- $z = 1.5e^{-(x-1)^2-y^2} - e^{-(x+1)^2-y^2}$



경사 하강법

Gradient Descent Method

- 함수 f 의 극소점 찾기
- 알고리즘
 - p = 임의의 점, λ = 작은 실수
 - $f(p)$ 의 변화가 큰 동안 다음을 반복
 - $p \leftarrow p - \lambda \cdot \nabla f(p)$
- 학습률 λ
 - 실험을 반복하여 적당한 값을 찾는다
- p 의 초기값에 따라 극소점이 다를 수도 있다



합성함수의 그래디언트

- 두 함수 $g: \mathbb{R}^m \rightarrow \mathbb{R}^n$, $f: \mathbb{R}^n \rightarrow \mathbb{R}$ 의 합성 함수

$$\begin{aligned}\mathbb{R}^m &\xrightarrow{g} \mathbb{R}^n \xrightarrow{f} \mathbb{R} \\ y &= g(x) \\ z &= f(y) = f(g(x))\end{aligned}$$

- g 의 야코비 행렬

$$J_g(x) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$$

- 그래디언트(열벡터)

$$\nabla_x z = J_g(x)^T \nabla_y z$$

계산

아빠의 키(x_1)	x_{11}	x_{12}	\cdots	x_{1n}
엄마의 키(x_2)	x_{21}	x_{22}	\cdots	x_{2n}
가계 소득(x_3)	x_{31}	x_{32}	\cdots	x_{3n}
아들의 키(y_1)	y_{11}	y_{12}	\cdots	y_{1n}

- 근사값

$$\hat{y}_{1i} = \phi(b + \sum_{j=1}^3 w_j x_{ji})$$

- 오차

$$L = \frac{1}{2} \sum_{i=1}^n (y_{1i} - \hat{y}_{1i})^2$$

- 미분

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= \sum_i (y_{1i} - \hat{y}_{1i}) \frac{\partial \hat{y}_{1i}}{\partial w_1} \\ &= \sum_i (y_{1i} - \hat{y}_{1i}) \phi'(b + \sum_j w_j x_{ji}) x_{1i} \end{aligned}$$

$$\begin{aligned}
\nabla L &= \left[\frac{\partial L}{\partial b} \quad \frac{\partial L}{\partial w_1} \quad \frac{\partial L}{\partial w_2} \quad \frac{\partial L}{\partial w_3} \right]^T \\
&= \begin{bmatrix} \sum_i (y_{1i} - \hat{y}_{1i}) \phi'(b + \sum_j w_j x_{ji}) \\ \sum_i (y_{1i} - \hat{y}_{1i}) \phi'(b + \sum_j w_j x_{ji}) x_{1i} \\ \sum_i (y_{1i} - \hat{y}_{1i}) \phi'(b + \sum_j w_j x_{ji}) x_{2i} \\ \sum_i (y_{1i} - \hat{y}_{1i}) \phi'(b + \sum_j w_j x_{ji}) x_{3i} \end{bmatrix}
\end{aligned}$$

- 행렬 표현

- 훈련 집합

$$\{(x_1, y_1), \dots, (x_n, y_n)\}$$

- 변수

$$W = [b \quad w_1 \quad w_2 \quad w_3]$$

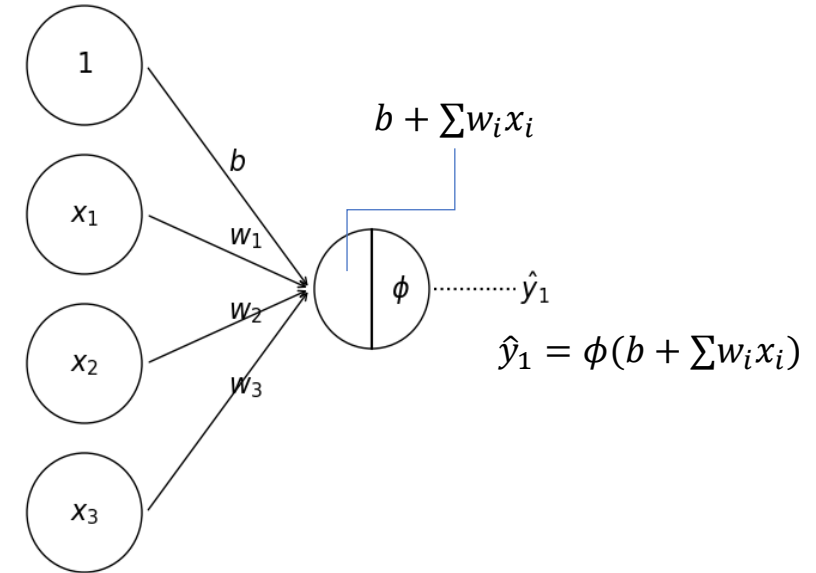
$$X = \begin{bmatrix} 1 & \dots & 1 \\ x_{11} & \dots & x_{1n} \\ x_{21} & \dots & x_{2n} \\ x_{31} & \dots & x_{3n} \end{bmatrix}$$

$$Z = F(X, W) = WX$$

$$\hat{Y}_1 = \phi(Z)$$

- L 의 그래디언트

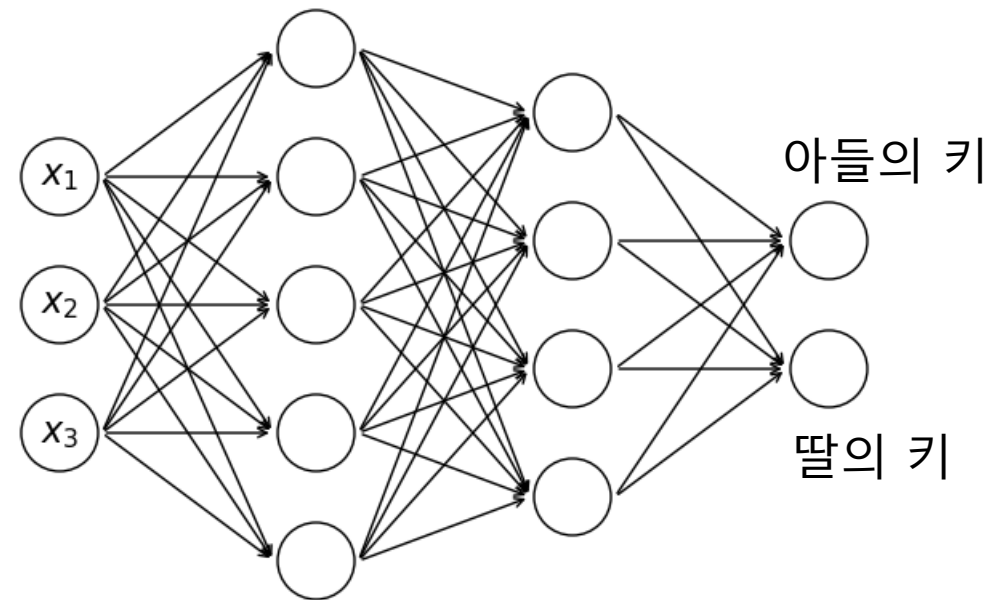
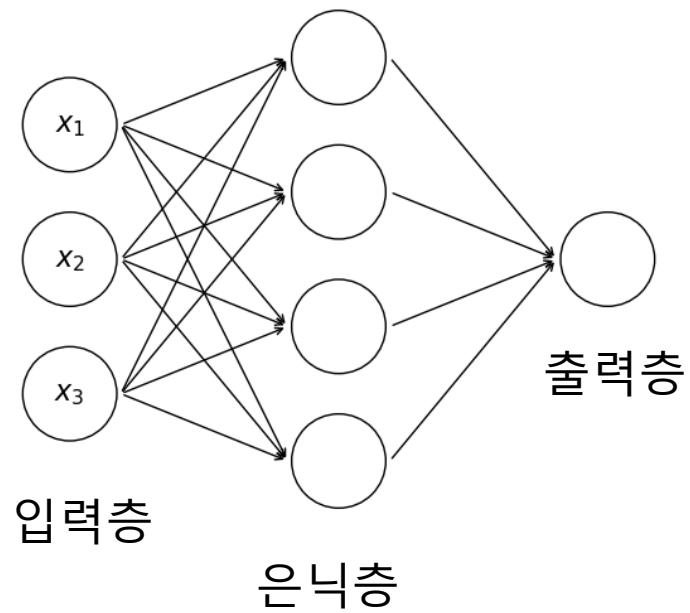
$$\nabla_W L = J_\phi(Z)^T (\hat{Y}_1 - Y_1) X^T$$



$$\begin{array}{ccccc} W & & & & \\ & \searrow & & & \\ X & \rightarrow & Z & \rightarrow & \hat{Y}_1 \end{array}$$

다층 퍼셉트론

- 은닉층이 있는 순방향 신경망

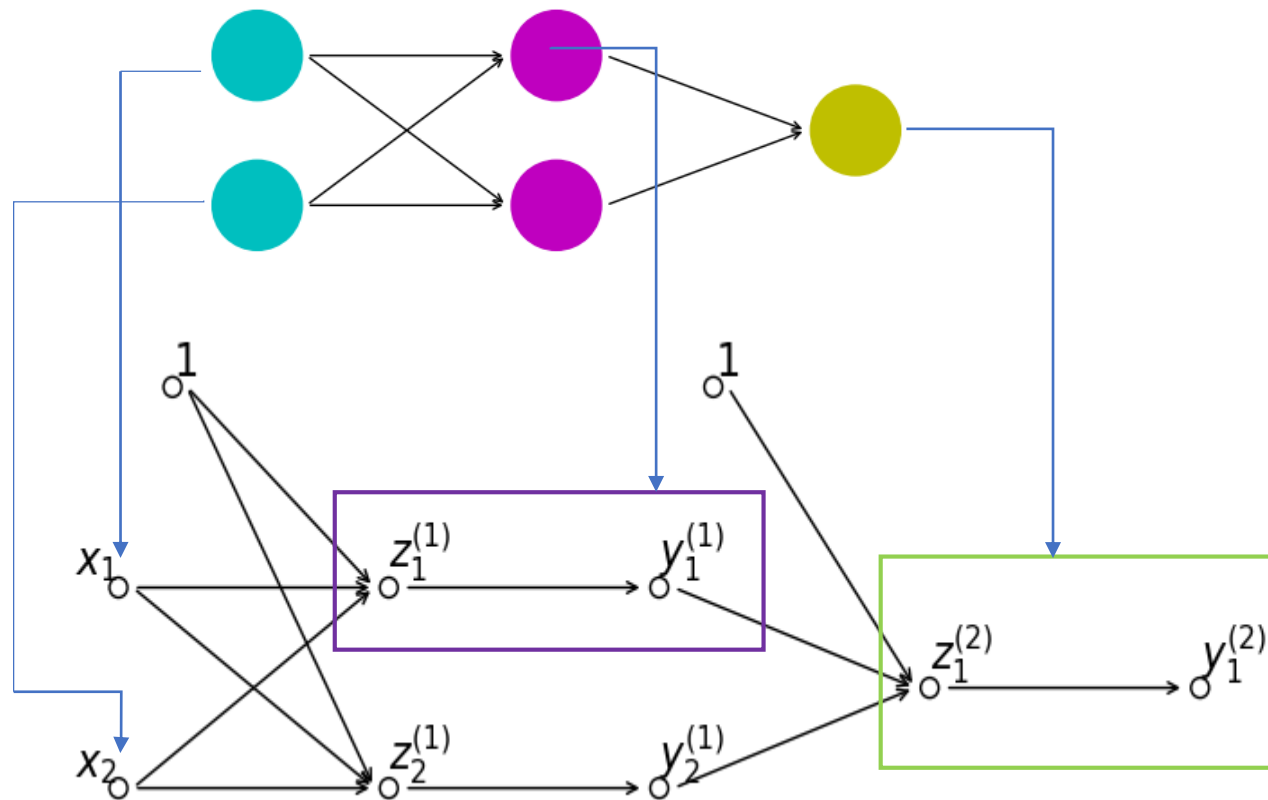
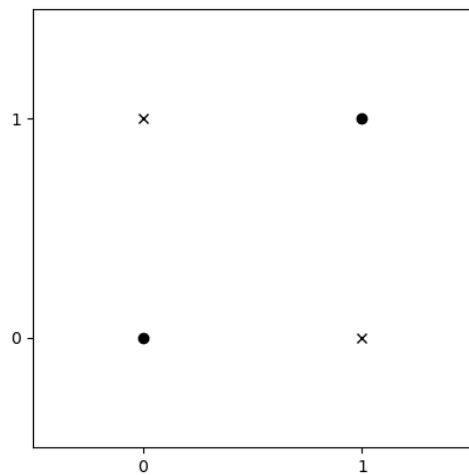


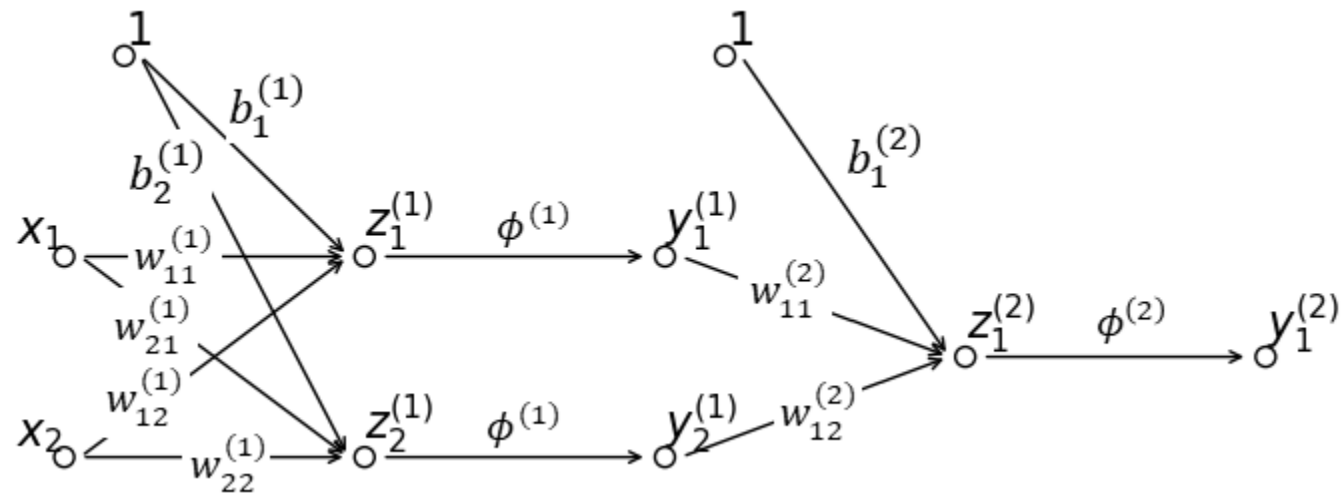
배타적 논리합(XOR)

$$p \oplus q = (\sim p \wedge q) \vee (p \wedge \sim q)$$

p	q	$p \oplus q$
F	F	F
F	T	T
T	F	T
T	T	F

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





$W^{(1)}$

$W^{(2)}$

$$\begin{array}{ccccccc}
 & & \searrow & & \searrow & & \\
 X & \rightarrow & Z^{(1)} & \rightarrow & Y_1^{(1)} & \rightarrow & Z^{(2)} \rightarrow Y^{(2)}
 \end{array}$$

벡터 표현

$$\begin{aligned}
 z_1^{(1)} &= b_1^{(1)} + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 \\
 y_1^{(1)} &= \phi^{(1)}(z_1^{(1)}) \\
 &\vdots
 \end{aligned}$$

행렬 표현

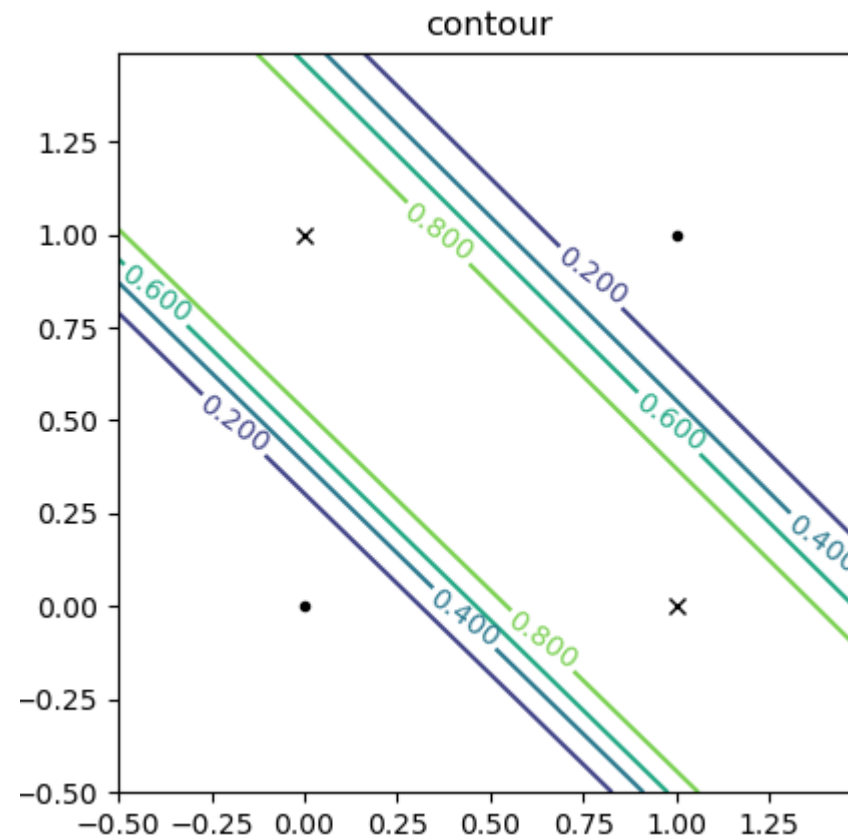
$$\begin{aligned}
 Z^{(1)} &= W^{(1)} X \\
 Y^{(1)} &= \phi^{(1)}(Z^{(1)}) \\
 &\vdots
 \end{aligned}$$

코드

```
import numpy as np
from keras import Sequential, losses
from keras.layers import Dense
import matplotlib.pyplot as plt

x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], 'float32')
t = np.array([[0, 1, 1, 0]], 'float32').T

model = Sequential()
model.add(Dense(2, input_dim=2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss=losses.mse, optimizer='adam')
result = model.fit(x, t, epochs=10000, verbose=2)
```

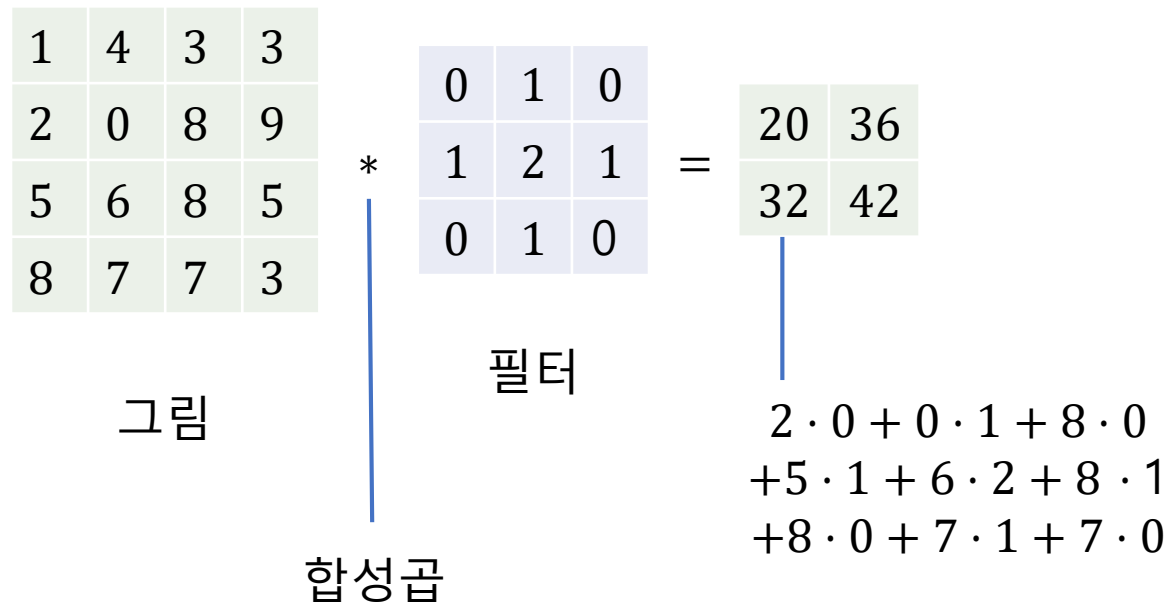


심층 학습

Deep Learning

- 심층 신경망을 사용하는 기계 학습
 - 심층 신경망 = 은닉층이 많은 신경망

합성곱 Convolution



0	0	0
1	1	1
0	0	0

가로선

0	1	0
0	1	0
0	1	0

세로선

1	0	0
0	1	0
0	0	1

대각선

0	0	1
0	1	0
1	0	0

대각선



*

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

=



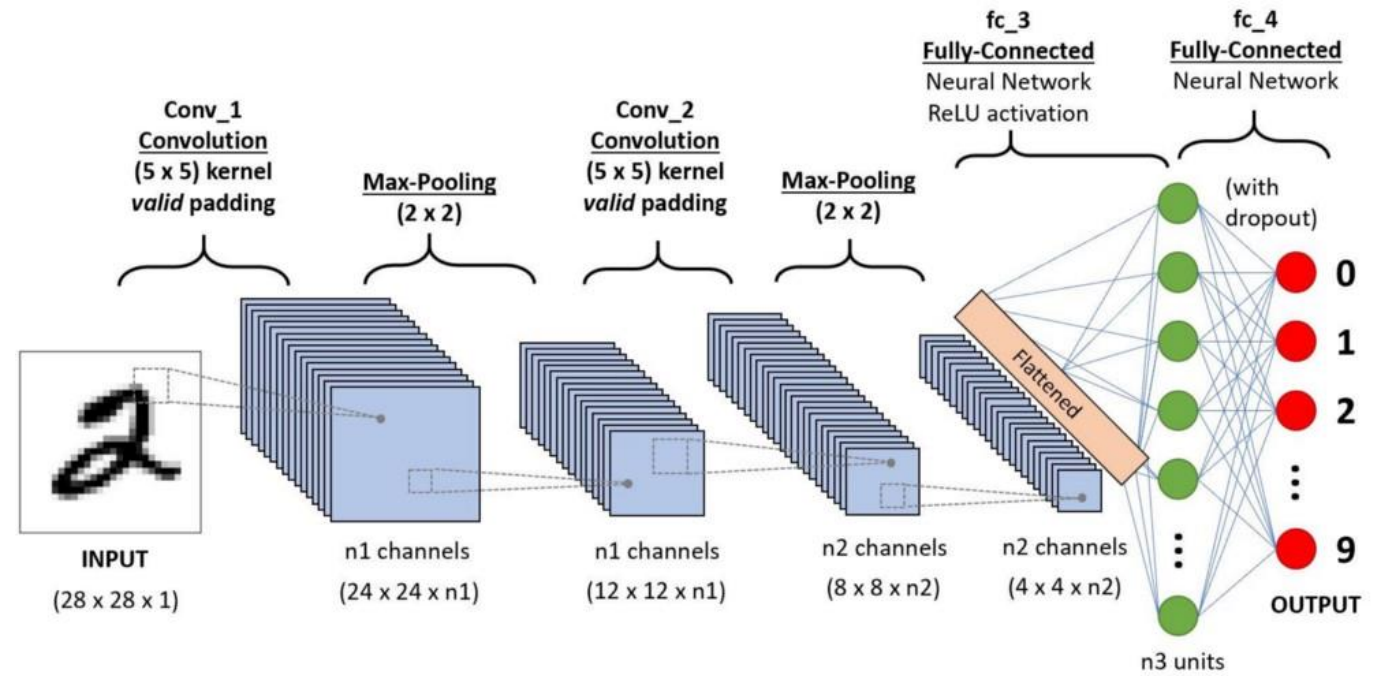
합성곱 신경망

CNN – Convolutional Neural Network

- 합성곱을 포함하는 신경망

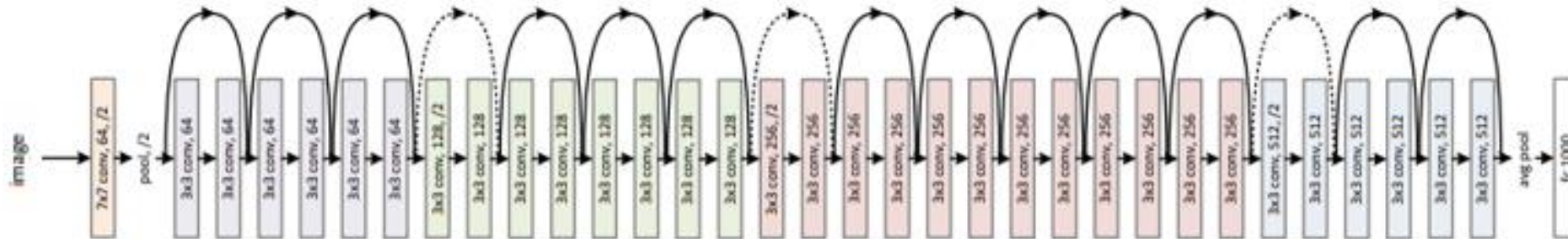
숫자 인식

- Mnist
 - 28×28 손글씨 6만개



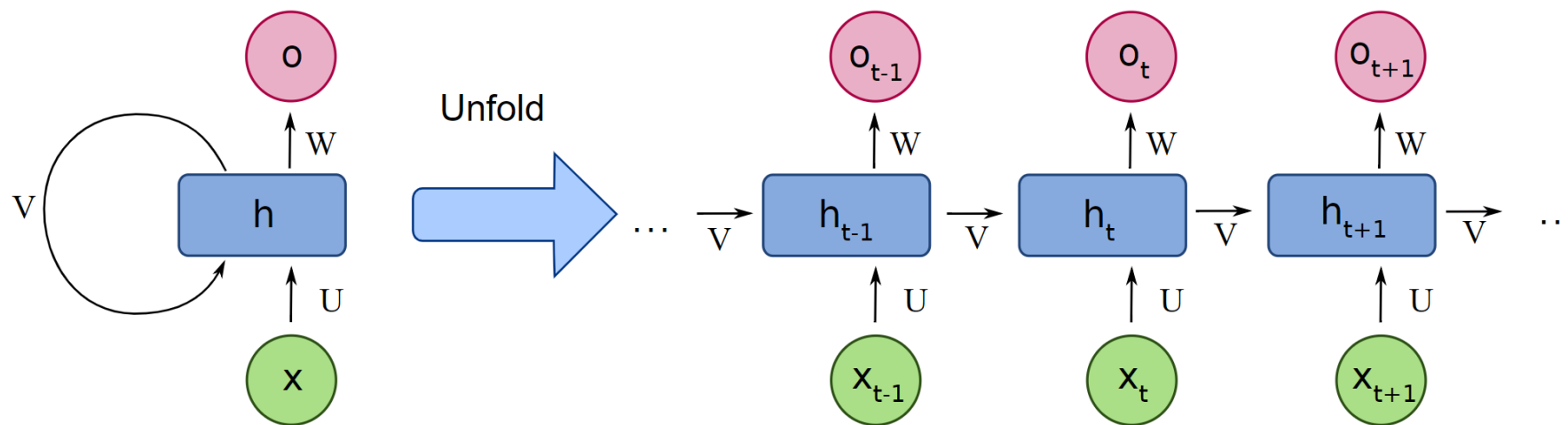
이미지 인식

- ResNet – 2015
 - 사람보다 좋은 인식률 95%
 - 34층
 - 노드 개수 \approx 6천만



RNN – Recurrent Neural Network

- 작곡, 소설



인공 신경망의 장단점

- 정확도가 높다
- 단점
 - 너무 복잡하여 인간이 이해하기 어렵다
 - ResNet은 변수가 6천만개인 함수의 극솟값을 계산한다
 - 오류가 있어도 사전에 발견할 수 없다
 - 하드웨어 비용이 너무 커서 접근이 어렵다
 - 이미 발표된 네트워크는 따라해 볼 수 있으나
 - 새로운 네트워크 개발에는 시간의 제약이 따른다

관련 분야

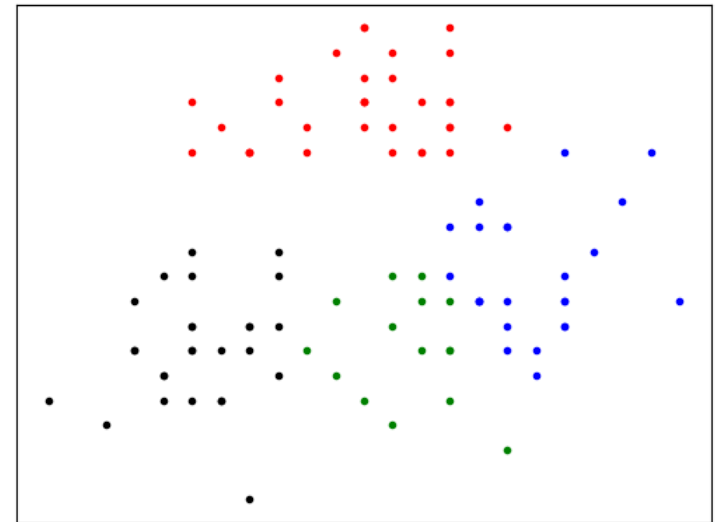
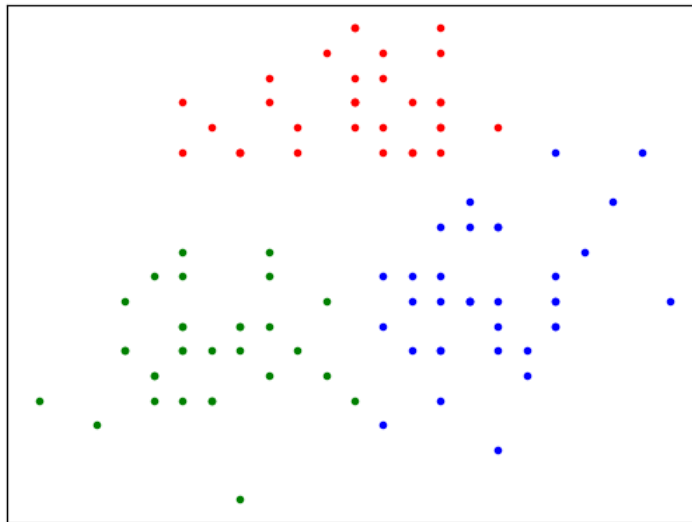
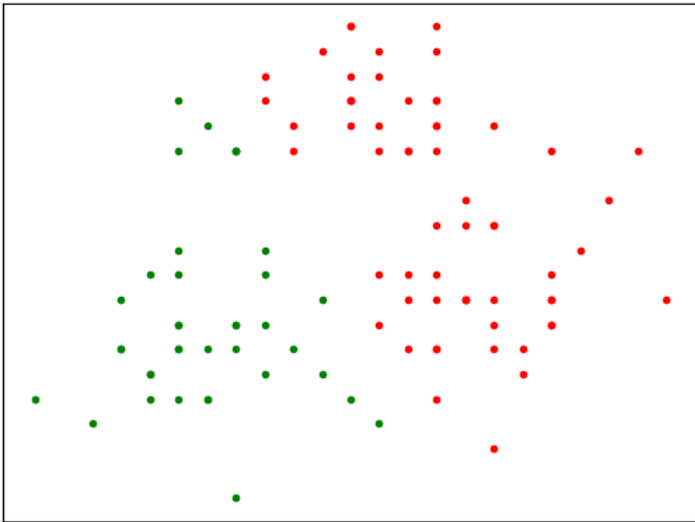
- 인공 신경망의 이해 및 설계
 - 미분, 벡터, 행렬, 확률, 통계
 - 함수에 대한 깊은 이해가 필요
 - 컴퓨터 프로그래밍
- 인공 신경망을 이용한 자료 분석
 - 이미 설계된 신경망에 대입하여 결과를 이해
 - 컴퓨터 프로그래밍
 - 파이썬 – 텐서플로우 / 케라스
 - 약간의 수학적 지식이 필요
 - 다양한 분야에서 많은 연구자들이 이용하고 있음

인공 신경망의 현재와 미래

- 결과에 대한 이유를 이해하는 것은 불가능에 가깝다
 - 복잡한 신경망을 이해하는 사람은 없다
 - 인공 지능 자동차가 호수로 돌진한다면?
- 특정 분야에서는 인간보다 정확하다
 - 인간도 자동차를 몰고 호수로 돌진한다
- 새로운 수학 이론이 필요?

비지도 학습 Unsupervised Learning

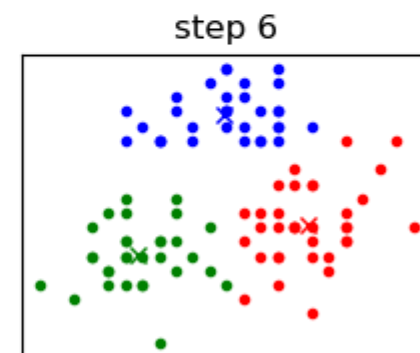
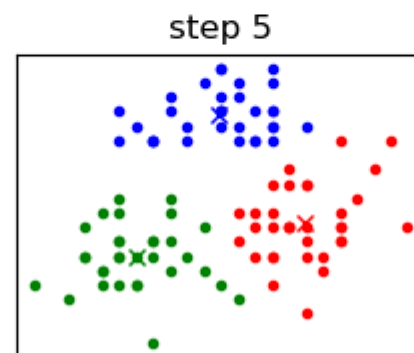
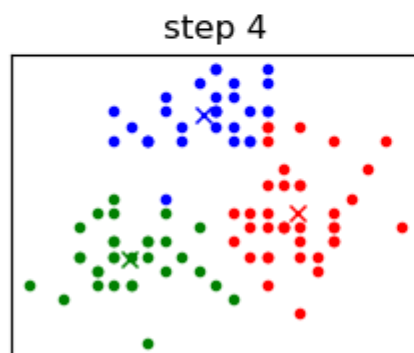
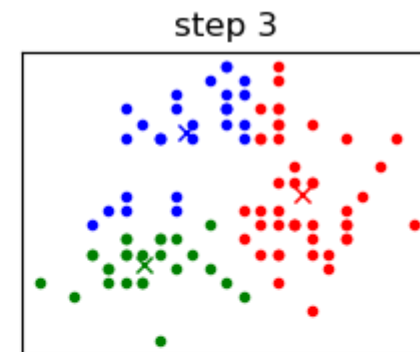
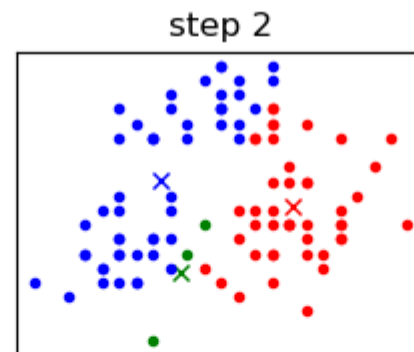
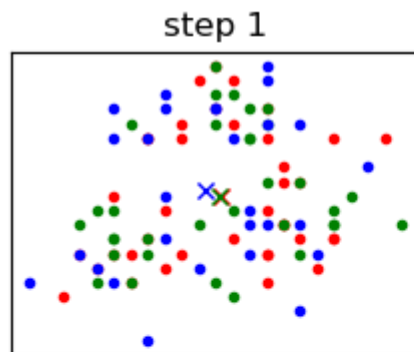
- 미지의 자료를 그룹으로 분류



K-Means Clustering

- 알고리즘

- K: 그룹의 개수
- K 그룹으로 임의 분할
- 반복
 - 각 그룹의 평균점 계산
 - 가까운 평균점으로 그룹 다시 배정



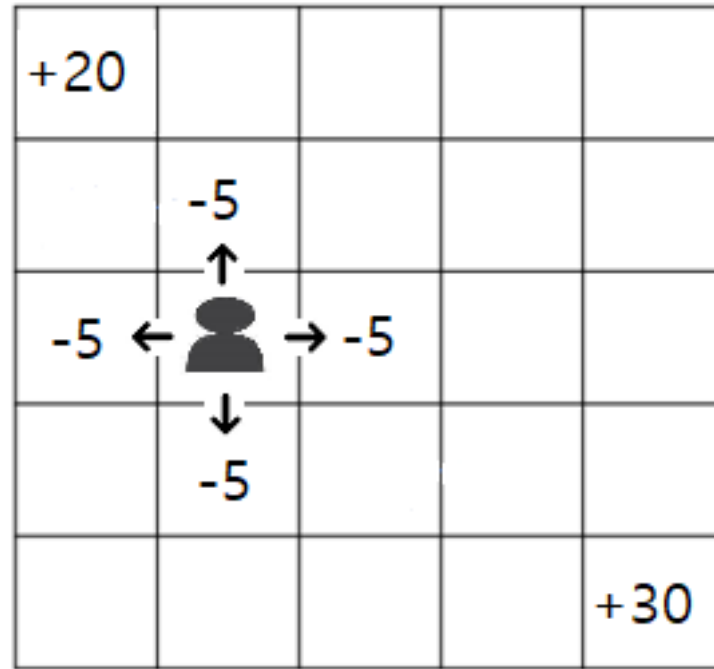
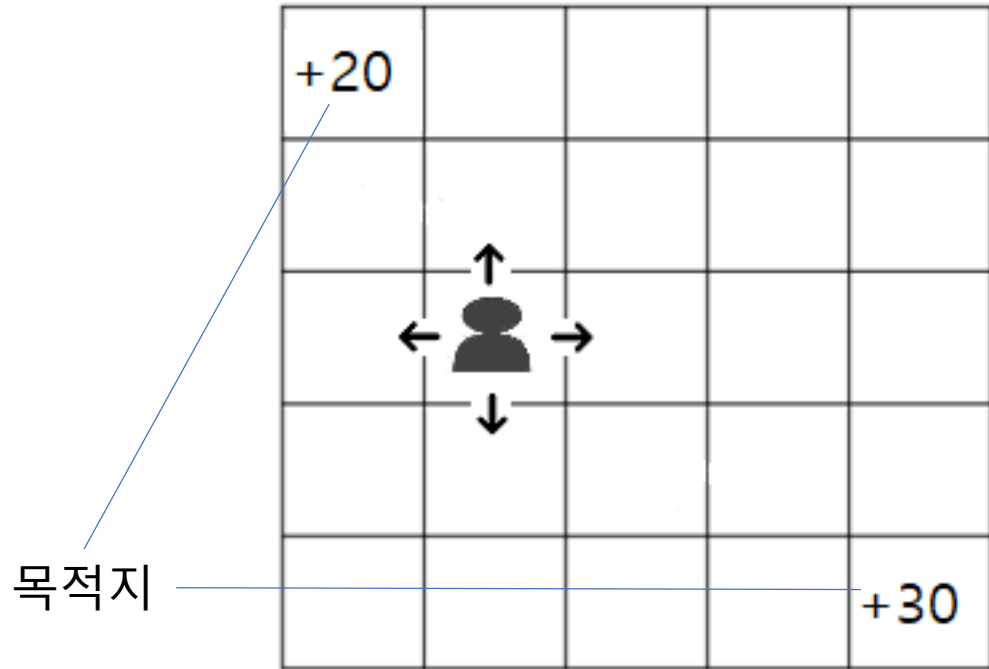
강화 학습

Reinforcement Learning

- 최적의 보상을 얻는 행동을 찾아가는 기계 학습 방법
 - [Cart Pole](#)
 - 알파고
 - [게임](#)

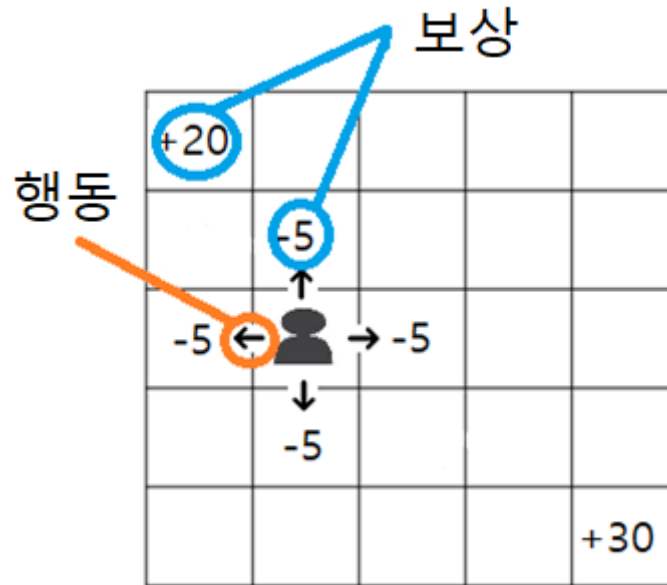
격자 세계

- 이득이 가장 큰 방향은 어디인가?



변수

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



예

- 올해는 무엇을 심을까?
 - 상태 – 예상 강수량, 예상 기온, 가격, 등
 - 행동 – 작물 선택
 - 보상 – 수확 / 소득
- 우산을 가지고 나갈까?
- 자동차는 어떤 방향으로 가야할까?
- 바둑 선수는 어떤 수를 둘까?

행위자와 환경

Agent and Environment

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

예

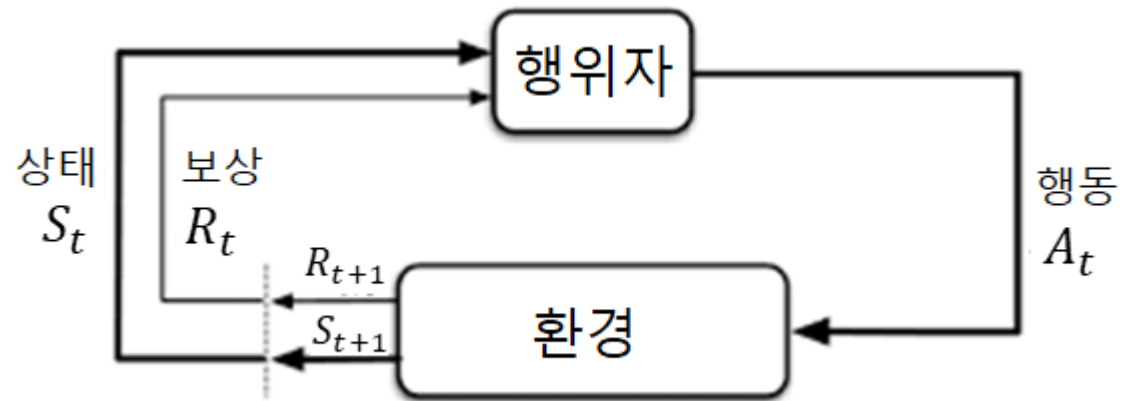
- 올해는 무엇을 심을까?
 - 행위자 - 농부
 - 환경 - ?
- 우산을 가지고 나갈까?
 - 환경 - ?
- 자동차는 어떤 방향으로 갈까?
 - 행위자 - 자동차
 - 환경 - ?
- 바둑 선수는 어떤 수를 둘까?
 - 환경 - 규칙 + 상대방

마르코프 결정 과정

MDP – Markov Decision Process

- 의사결정을 위한 수학적 모델
- 직전의 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 s 에서 취할 수 있는 행동의 집합
- \mathcal{R} – 보상의 집합
 - $\mathcal{R}(s, a)$ – 상태 s 에서 행동 a 를 취했을 때 얻는 보상의 집합
- $p(s', r | s, a) = \Pr(S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a)$
 - 상태 s 에서 행동 a 를 취했을 때 상태 s' 이 되고 보상 r 을 얻을 확률

참고
대문자: 변수
소문자: 값

에피소드 Episode

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

반환 Return

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

격자 세계

- 에피소드

- $S_0 \xrightarrow{A_0} R_1, S_1 \xrightarrow{A_1} R_2, S_2 \xrightarrow{A_2} R_3, S_3 \xrightarrow{A_3} R_4, S_4$

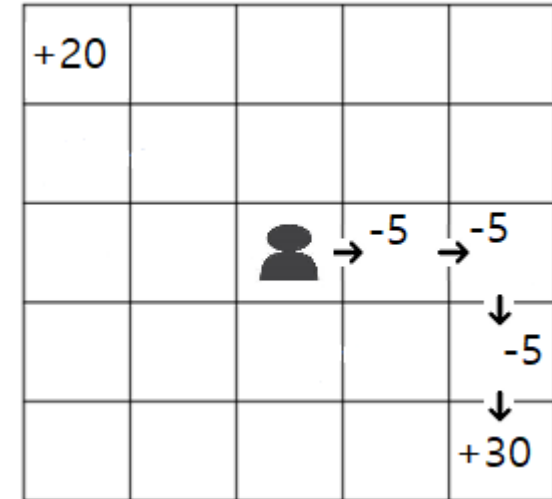
- 할인율

- $\gamma = 0.9$

- 반환

- $$\begin{aligned} G_0 &= R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \gamma^4 R_5 \\ &= -5 + 0.9 \cdot (-5) + 0.81 \cdot (-5) + 0.729 \cdot 30 \\ &= 8.32 \end{aligned}$$

- $G_1 = ?$



정책 Policy

- 행위자가 행동을 선택하는 방법
- 종류
 - 결정적 정책
 - 각 상태에서 취할 행동이 결정되어 있음
 - 확률적 정책
 - 각 상태에서 확률에 따라 행동을 취함

정책의 표현

- $\pi(a|s) = \Pr(A_t = a \mid S_t = s)$
- 예
 - 결정적 정책
 - $\pi(a|s) = 0$ or 1
 - 랜덤 정책
 - $\pi(a|s)$ 가 균일

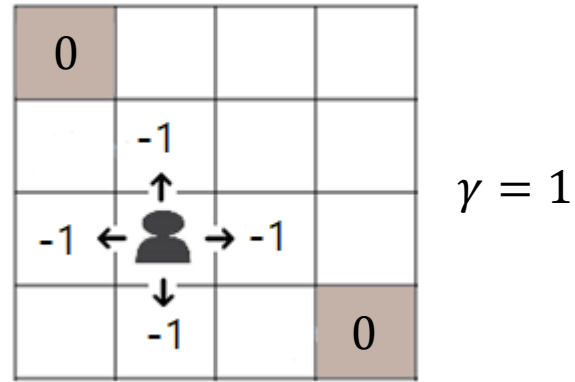
정리

- 행위자, 환경, 상태, 행동, 보상, 정책, 에피소드, 반환, 할인율
- 확률적으로 결정되는 것들은 무엇인가?
- 올해는 무엇을 심을까?
 - 콩을 10개 심으면 몇 개의 싹이 틀까?
- 우산을 가지고 나갈까?
 - 상태는 어제와 비슷한데 오늘도 비가 올까?
- 자동차는 어떤 방향으로 갈까?
- 바둑 선수는 어떤 수를 둘까?
 - 내가 둔 수에 대한 상대의 반응은?

상태 가치 함수

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

격자 세계



- 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

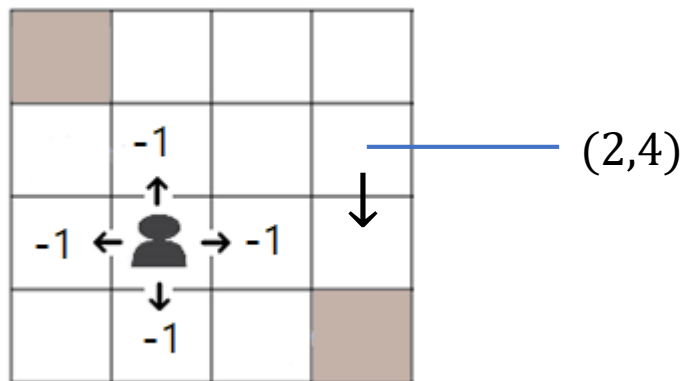
0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

행동 가치 함수

- 행동 가치
 - 정책에 의하여 행동을 선택할 경우 각 행동의 가치
- 정책 π 에 대한 행동 가치 함수
 - $q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$
- 상태 가치와 행동 가치의 관계는?

예

- 랜덤 정책을 취할 때 상태 (2,4)에서 행동 ↓의 가치는?



벨만 방정식

Bellman equation

- 벨만 방정식

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

- 행동 가치 함수

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

벨만 방정식의 해

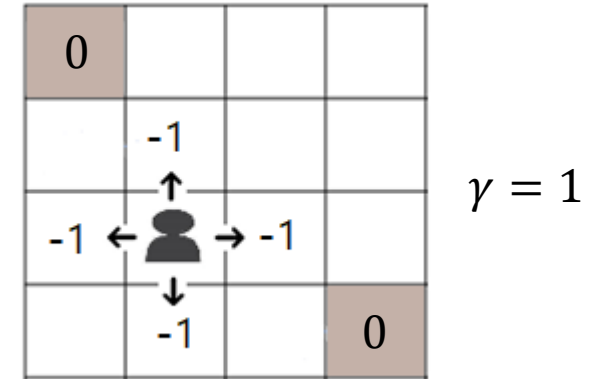
- 일차 연립방정식
 - 미지수의 개수 = 상태의 개수
 - $X = AX + b$ 꼴
- 점화식
 - X_0 : 임의로 초기화
 - $X_{i+1} = AX_i + b$
 - 해는 $\lim_{n \rightarrow \infty} X_n$

벨만 방정식

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

해 구하기

- $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_\infty = v_\pi$



랜덤 정책 π 에 대한 가치 함수

$$v_\pi(s) = \sum_{a=1}^4 \frac{1}{4} [-1 + v_\pi(s')]$$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 0$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 1$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 2$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 3$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

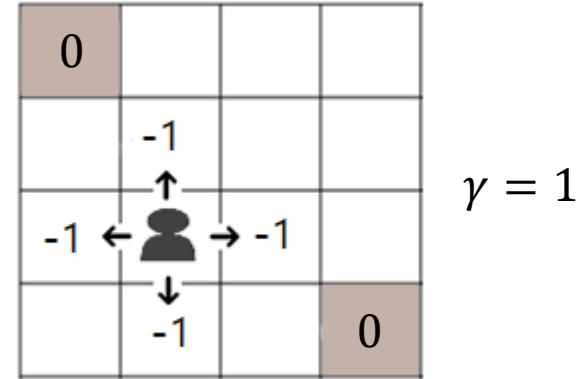
$k = 10$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

$k = \infty$

최적 가치 함수와 최적 정책

- 최적 가치 함수
 - 상태의 최대 가치
 - $v_*(s) = \max_{\pi} v_{\pi}(s)$
- 최적 정책
 - 가장 우수한 정책 π_*
 - $v_{\pi_*}(s) = v_*(s)$

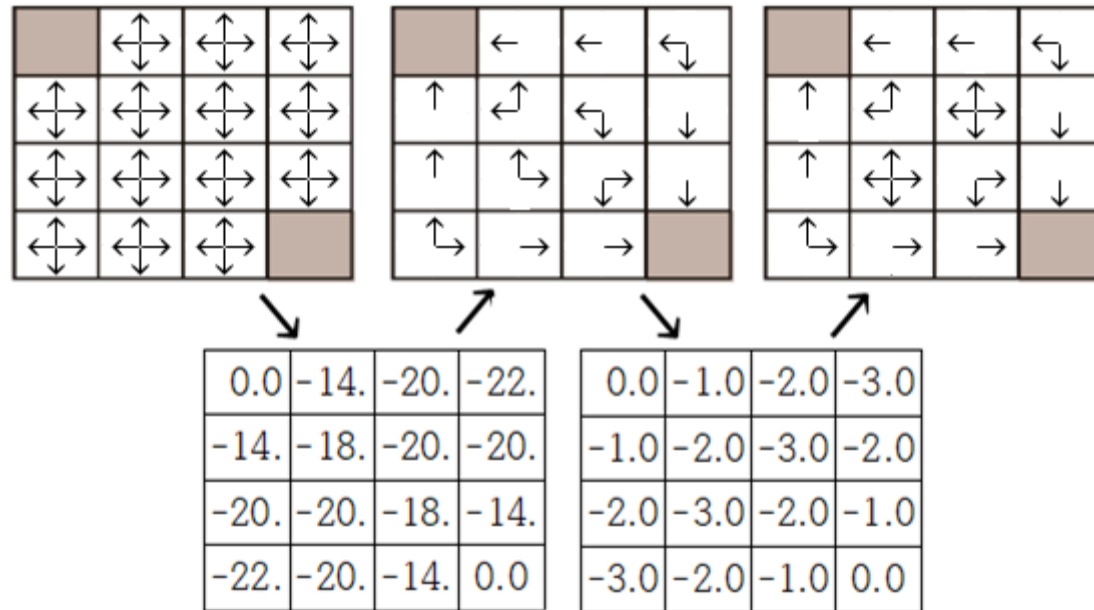


최적 가치

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

최적 정책 구하기

- π_0 : 임의의 정책
- $\pi_0 \rightarrow v_{\pi_0} \rightarrow \pi_1 \rightarrow v_{\pi_1} \rightarrow \dots \rightarrow \pi_*$
- $v_{\pi_*} = v_*$



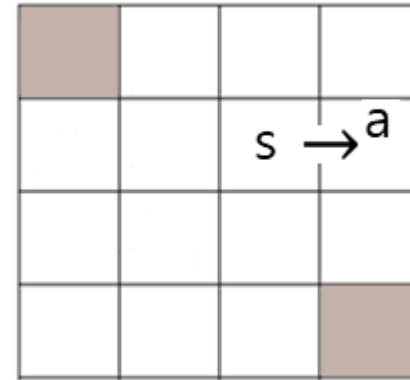
Dynamic Programming

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

몬테 카를로 방법

Monte Carlo Methods

- q_π 의 근삿값
 - 에피소드의 표본에 대하여 계산



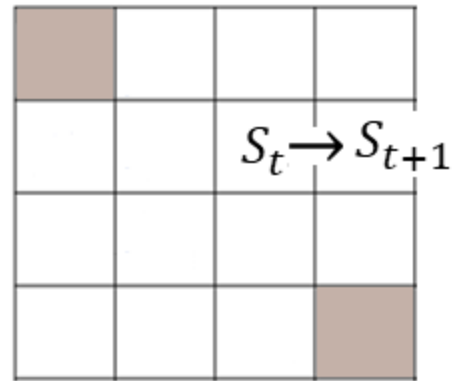
$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

모든 에피소드에 대한 반환의 평균을 계산

시간차 학습

Temporal Difference Learning

- 에피소드의 각 단계에서 상태 가치 함수를 다시 설정
- 상태의 가치를 다음 상태의 가치로 보정
 - $v(S_t) \leftarrow v(S_t) + \alpha[R_{t+1} + \gamma v(S_{t+1}) - v(S_t)]$
 - α - 학습률



예: 시간차 학습

$$\pi(a|s) = 0.25, \alpha = 0.01, \gamma = 1$$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

에피소드 0

0.0	0.0	0.0	0.0
-.03	-.02	-.02	-.02
-.01	-.04	-.02	0.0
-.03	-.07	-.01	0.0

에피소드 1

0.0	0.0	0.0	0.0
-.05	-.02	-.02	-.02
-.01	-.04	-.02	0.0
-.03	-.08	-.01	0.0

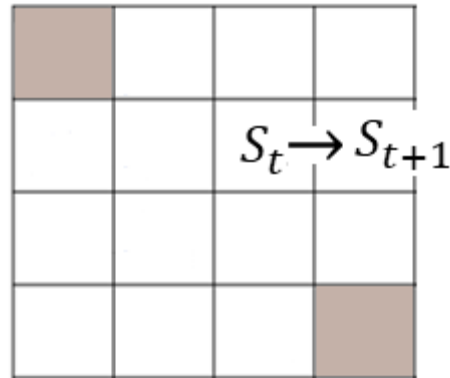
에피소드 2

0.0	-13.	-19.	-21.
-14.	-17.	-19.	-19.
-19.	-19.	-17.	-13.
-21.	-19.	-15.	0.0

에피소드 100000

Sarsa

- 에피소드를 따라 행동 가치 함수를 다시 설정
 - $q(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha[R_{t+1} + \gamma q(S_{t+1}, A_{t+1}) - q(S_t, A_t)]$
- $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$ 를 사용하기 때문에 "sarsa"라 부른다



Q-learning

- $q(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a q(S_{t+1}, a) - q(S_t, A_t)]$
 - $\max_a q(S_{t+1}, a)$ 는 정책에 무관
 - 에피소드를 따라 계산

알파고



AlphaGo

- 가치가 큰 수(행동)를 선택
- 모든 수를 시도해 볼 수는 없다
- 이길 확률이 높은 그림과 유사한 그림을 만드는 수들을 시도해 본다
 - 합성곱 신경망

강화 학습의 현재

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

프로그래밍

- 파이썬
 - 배우기 쉬운 언어