

강화학습은 무엇이고, 무엇이 가능한가?

김권현

숨은원리

2017. 12. 14(목), 17:30-18:20

차례

- ① 강화학습을 바탕으로 인간을 능가하고 있는 인공지능
- ② 기계 학습의 하나, 강화학습
- ③ 간단하게 설명하는 가치 기반 강화 학습
- ④ 심층신경망DNN, Deep Neural Networks
- ⑤ 강화학습과 심층신경망의 결합: DQN
- ⑥ 심리학, 신경과학
- ⑦ 최근의 발전 방향
- ⑧ 강화 학습의 응용 분야
- ⑨ 마무리

강화학습을 바탕으로 인간을 능가하고 있는 인공지능

자율 헬리콥터(Autonomous Helicopter)

- RC(Remote Control) 헬리콥터 묘기 ► (00:00:06 - 1:17:00)
- 자율 헬리콥터 ► (00:02:19 - 00:03:07)
- 스탠포드대 자율 헬리콥터 <http://heli.stanford.edu/>

아타리(ATARI) 게임에서 DQN(Deep Q-Network)

- 아타리(ATARI) 게임 소개 ►
- DQN의 아타리 게임 학습 예 ►

알파고

- 카스파로프 vs. 딥블루 ►
- 이세돌 vs. 알파고 ►

기계 학습의 하나, 강화학습

기계 학습의 3가지 분류

- 지도학습(SL, Supervised Learning)
- 비지도학습 (UL, Unsupervised Learning)
- 강화학습(RL, Reinforcement Learning)

강화학습의 두 줄기

- 가치 기반(Value based)
- 정책 기반(Policy based)
- 액터-크리틱(Actor-Critic)

간단하게 설명하는 가치 기반 강화 학습

결정론적 환경의 예

상태 state	행동 action		
	a	b	c
A	30($\rightarrow B$)	-30($\rightarrow C$)	10($\rightarrow A$)
B	-20($\rightarrow C$)	30($\rightarrow B$)	-10($\rightarrow A$)
C	20($\rightarrow A$)	-10($\rightarrow A$)	10($\rightarrow C$)

- 보상reward $r(s, a), r(s, a, s')$
- 할인율discount factor : 미래의 보상과 현재의 보상을 비교하는 방법 (이자, 인플레이션, 사망율 등의 해석)
- 반환값return : 즉각보상과 할인된 미래 보상의 총합
- 정책policy : 주어진 상태에서 어떤 행동을 할 것인가? 결정론적 정책 $\pi(s)$, 확률적인 정책 $\pi(a|s)$
- 끝없이 지속되는 과제continuous task, 끝이 있는 과제episodic task

결정론적 환경의 예

상태 state	행동 action		
	a	b	c
A	30($\rightarrow B$)	-30($\rightarrow C$)	10($\rightarrow A$)
B	-20($\rightarrow C$)	30($\rightarrow B$)	-10($\rightarrow A$)
C	20($\rightarrow A$)	-10($\rightarrow A$)	10($\rightarrow C$)

결정론적 정책의 예.¹

$$\pi_1(A) = a, \pi_1(B) = a, \pi_1(C) = a$$

이때, 상태 A에서 즉각보상과 할인된 미래 보상의 총합은,

$$V_1(A) = R(A, a) + \gamma^1 R(B, a) + \gamma^2 R(C, a) + \gamma^3 R(A, a) + \dots$$

¹ 확률적 정책으로 표현한다면, $\pi_1(a|A) = 1, \pi_1(b|A) = 0, \pi_1(c|A) = 0, \pi_2(a|B) = 1, \pi_1(b|B) = 0, \pi_1(c|B) = 0, \pi_1(a|C) = 1, \pi_1(b|C) = 0, \pi_1(c|C) = 0$

결정론적 환경의 예: 가치 함수

상태 state	행동 action		
	a	b	c
A	30($\rightarrow B$)	-30($\rightarrow C$)	10($\rightarrow A$)
B	-20($\rightarrow C$)	30($\rightarrow B$)	-10($\rightarrow A$)
C	20($\rightarrow A$)	-10($\rightarrow A$)	10($\rightarrow C$)

정책 π_1 을 따를 때, 상태 A, B, C 의 가치 함수를 구해보면,

$$V_1(A) = R(A, a) + \gamma^1 R(B, a) + \gamma^2 R(C, a) + \gamma^3 R(A, a) + \gamma^4 R(B, a) + \gamma^5 R(C, a) + \dots$$

$$V_1(B) = R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \gamma^4 R(C, a) + \gamma^5 R(A, a) + \dots$$

$$V_1(C) = R(C, a) + \gamma^1 R(A, a) + \gamma^2 R(B, a) + \gamma^3 R(C, a) + \gamma^4 R(A, a) + \gamma^5 R(B, a) + \dots$$

결정론적 환경의 예: 가치 함수

상태 state	행동 action		
	a	b	c
A	30($\rightarrow B$)	-30($\rightarrow C$)	10($\rightarrow A$)
B	-20($\rightarrow C$)	30($\rightarrow B$)	-10($\rightarrow A$)
C	20($\rightarrow A$)	-10($\rightarrow A$)	10($\rightarrow C$)

정책 π_1 을 따를 때, 상태 A, B, C 의 가치 함수를 구해보면,

$$V_1(A) = R(A, a) + \gamma^1 R(B, a) + \gamma^2 R(C, a) + \gamma^3 R(A, a) + \gamma^4 R(B, a) + \gamma^5 R(C, a) + \dots$$

$$V_1(B) = R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \gamma^4 R(C, a) + \gamma^5 R(A, a) + \dots$$

$$V_1(C) = R(C, a) + \gamma^1 R(A, a) + \gamma^2 R(B, a) + \gamma^3 R(C, a) + \gamma^4 R(A, a) + \gamma^5 R(B, a) + \dots$$

결정론적 환경의 예: 가치 함수

$$V_1(A) = R(A, a) + \gamma^1 R(B, a) + \gamma^2 R(C, a) + \gamma^3 R(A, a) + \gamma^4 R(B, a) + \gamma^5 R(C, a) + \dots$$

$$V_1(B) = R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \gamma^4 R(C, a) + \gamma^5 R(A, a) + \dots$$

$$V_1(C) = R(C, a) + \gamma^1 R(A, a) + \gamma^2 R(B, a) + \gamma^3 R(C, a) + \gamma^4 R(A, a) + \gamma^5 R(B, a) + \dots$$

반복적인 부분을 활용하면, 다음과 같이 다시 쓸 수 있다.

$$\begin{aligned} V_1(A) &= R(A, a) + \gamma \cdot R(B, a) + \gamma \cdot \gamma^1 R(C, a) + \gamma \cdot \gamma^2 R(A, a) + \gamma \cdot \gamma^3 R(B, a) + \dots \\ &= R(A, a) + \gamma(R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \dots) \end{aligned}$$

$$V_1(B) = R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \dots$$

결정론적 환경의 예: 벨만 가치 함수 방정식

$$\begin{aligned}
 V_1(A) &= R(A, a) + \gamma \cdot R(B, a) + \gamma \cdot \gamma^1 R(C, a) + \gamma \cdot \gamma^2 R(A, a) + \gamma \cdot \gamma^3 R(B, a) + \dots \\
 &= R(A, a) + \gamma(R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \dots) \\
 V_1(B) &= R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \dots
 \end{aligned}$$

이를 활용하면, 다음과 같은 상태 가치 함수 방정식을 얻는다.

$$V_1(A) = R(A, a) + \gamma V_1(B)$$

$$V_1(B) = R(B, a) + \gamma V_1(C)$$

$$V_1(C) = R(C, a) + \gamma V_1(A)$$

벨만 방정식을 풀기

상태 state	행동 action		
	a	b	c
A	30($\rightarrow B$)	-30($\rightarrow C$)	10($\rightarrow A$)
B	-20($\rightarrow C$)	30($\rightarrow B$)	-10($\rightarrow A$)
C	20($\rightarrow A$)	-10($\rightarrow A$)	10($\rightarrow C$)

우리는 $R(A, a)$, $R(B, a)$, $R(C, a)$ 를 모두 알고 있으므로, 만약 할인율 $\gamma = 0.9$ 라면,

$$\textcolor{red}{V_1(A)} = 30 + 0.9 \textcolor{blue}{V_1(B)}$$

$$\textcolor{blue}{V_1(B)} = -20 + 0.9 \textcolor{brown}{V_1(C)}$$

$$\textcolor{brown}{V_1(C)} = 20 + 0.9 \textcolor{red}{V_1(A)}$$

벨만 방정식을 풀기 (행렬 표현)

앞에서 구한 벨만 방정식

$$\mathbf{V}_1(\mathbf{A}) = 30 + 0.9 \mathbf{V}_1(\mathbf{B})$$

$$\mathbf{V}_1(\mathbf{B}) = -20 + 0.9 \mathbf{V}_1(\mathbf{C})$$

$$\mathbf{V}_1(\mathbf{C}) = 20 + 0.9 \mathbf{V}_1(\mathbf{A})$$

이를 행렬식으로 표현해보자. $\mathbf{V}_1 = \begin{pmatrix} \mathbf{V}_1(\mathbf{A}) \\ \mathbf{V}_1(\mathbf{B}) \\ \mathbf{V}_1(\mathbf{C}) \end{pmatrix}$ 로 놓으면,

$$\begin{aligned} \mathbf{V}_1 &= \begin{pmatrix} R(\mathbf{A}, a) \\ R(\mathbf{B}, a) \\ R(\mathbf{C}, a) \end{pmatrix} + \gamma \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \mathbf{V}_1 \\ &= \mathbf{R}_1 + 0.9 \mathbf{T}_1 \mathbf{V}_1. \end{aligned}$$

벨만 방정식을 푸는 두 가지 방법

$$\mathbf{V}_1 = \mathbf{R}_1 + 0.9 \mathbf{T}_1 \mathbf{V}_1$$

- 행렬 방정식을 푼다.

$$(\mathbf{I} - 0.9 \mathbf{T}_1) \mathbf{V}_1 = \mathbf{R}_1$$

$$\mathbf{V}_1 = (\mathbf{I} - 0.9 \mathbf{T}_1)^{-1} \mathbf{R}_1$$

- 반복적인 갱신 update를 이용한다.

$$\mathbf{V}_1^{(1)} = \mathbf{R}_1 + 0.9 \mathbf{T}_1 \mathbf{V}_1^{(0)}$$

$$\mathbf{V}_1^{(2)} = \mathbf{R}_1 + 0.9 \mathbf{T}_1 \mathbf{V}_1^{(1)}$$

⋮

벨만 방정식의 해: 반복적인 갱신

첫 번째 갱신에서 가치 함수 V_1 은 다음과 같이 결정된다.

$$V_1^{(1)}(A) = R(A, a) + \gamma V_1^{(0)}(B)$$

$$V_1^{(1)}(B) = R(B, a) + \gamma V_1^{(0)}(C)$$

$$V_1^{(1)}(C) = R(C, a) + \gamma V_1^{(0)}(A)$$

두 번째 갱신에서 가치 함수 V_2 은 다음과 같다.

$$V_1^{(2)}(A) = R(A, a) + \gamma V_1^{(1)}(B)$$

$$V_1^{(2)}(B) = R(B, a) + \gamma V_1^{(1)}(C)$$

$$V_1^{(2)}(C) = R(C, a) + \gamma V_1^{(1)}(A)$$

그리고 다음과 같이 다시 쓸 수 있다.

$$\begin{aligned} V_1^{(2)}(A) &= R(A, a) + \gamma(R(B, a) + \gamma V_1^{(0)}(C)) \\ &= R(A, a) + \gamma R(B, a) + \gamma^2 V_1^{(0)}(C) \end{aligned}$$

상태-행동 가치 함수 Q-function

주어진 정책, 주어진 상태에서 “행동”을 평가하기

$$V_1(A) = R(A, a) + \gamma^1 R(B, a) + \gamma^2 R(C, a) + \gamma^3 R(A, a) + \gamma^4 R(B, a) + \gamma^5 R(C, a)$$

$$Q_1(A, a) = R(A, a) + \gamma^1 R(B, a) + \gamma^2 R(C, a) + \gamma^3 R(A, a) + \gamma^4 R(B, a) + \gamma^5 R(C, a)$$

$$Q_1(A, b) = R(A, b) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \gamma^4 R(C, a) + \gamma^5 R(A, b)$$

$$Q_1(A, c) = R(A, c) + \gamma^1 R(A, a) + \gamma^2 R(B, a) + \gamma^3 R(C, a) + \gamma^4 R(A, a) + \gamma^5 R(B, c)$$

이를 가치 함수를 활용하여 표현해보면,

$$V_1(A) = R(A, a) + \gamma^1 V_1(B)$$

$$Q_1(A, a) = R(A, a) + \gamma^1 V_1(B)$$

$$Q_1(A, b) = R(A, b) + \gamma^1 V_1(C)$$

$$Q_1(A, c) = R(A, c) + \gamma^1 V_1(A)$$

상태-행동 가치 함수 Q-function 벨만 방정식

$$V_1(A) = R(A, a) + \gamma^1 V_1(B)$$

$$Q_1(A, a) = R(A, a) + \gamma^1 V_1(B)$$

$$Q_1(A, b) = R(A, b) + \gamma^1 V_1(C)$$

$$Q_1(A, c) = R(A, c) + \gamma^1 V_1(A)$$

여기서 $Q_1(A, a) = V_1(A)$ 를 활용하면, 다음의 방정식을 구할 수 있다.

$$Q_1(A, a) = R(A, a) + \gamma^1 Q_1(B, a)$$

$$Q_1(A, b) = R(A, b) + \gamma^1 Q_1(C, a)$$

$$Q_1(A, c) = R(A, c) + \gamma^1 Q_1(A, a)$$

$$Q_1(B, a) = R(B, a) + \gamma^1 Q_1(C, a)$$

$$Q_1(B, b) = R(B, b) + \gamma^1 Q_1(B, a)$$

⋮

상태-행동 가치 함수 Q-function 벨만 방정식의 해 구하기

- 역행렬을 이용하여 해를 구하기
- 반복적으로 갱신 update하기

상태-행동 가치 함수 Q-function 활용하여 최상의 정책 구하기 : 정책을 반복적으로 개선하기

Table: 정책 π_1 을 따를 때 Q-함수값

state	action		
	a	b	c
A	104.059	72.28782	103.6531
B	82.28782	104.059	83.65314
C	113.6531	64.05904	112.2878

Q-함수를 활용하여 개선된 정책 π_2 를 다음과 같이 정할 수 있다.

$$\pi_2(A) = a, \pi_2(B) = b, \pi_2(C) = a$$

그리고 이에 대해 다시 Q-함수를 구하고, 다시 정책을 개선하는 과정을 반복함으로써 최상의 정책에 이를 수 있다.

상태-행동 가치 함수 Q-function 를 활용하여 최상의 정책 구하기 : 최적 벨만 방정식

$$Q_*(A, a) = R(A, a) + \gamma^1 Q_*(B, ?)$$

$$Q_*(A, b) = R(A, b) + \gamma^1 Q_*(C, ?)$$

$$Q_*(A, c) = R(A, c) + \gamma^1 Q_*(A, ?)$$

$$Q_*(B, a) = R(B, a) + \gamma^1 Q_*(C, ?)$$

$$Q_*(B, b) = R(B, b) + \gamma^1 Q_*(B, ?)$$

최상의 정책이라면 행동은 $Q_*(s, a)$ 를 최대로 하는 행동을 선택할 것이다.

$$Q_*(A, a) = R(A, a) + \gamma^1 \max_{a'} Q_*(B, a')$$

$$Q_*(A, b) = R(A, b) + \gamma^1 \max_{a'} Q_*(C, a')$$

$$Q_*(A, c) = R(A, c) + \gamma^1 \max_{a'} Q_*(A, a')$$

$$Q_*(B, a) = R(B, a) + \gamma^1 \max_{a'} Q_*(C, a')$$

최적 벨만 방정식의 해 구하기

$$Q_*(A, a) = R(A, a) + \gamma^1 \max_{a'} Q_*(B, a')$$

$$Q_*(A, b) = R(A, b) + \gamma^1 \max_{a'} Q_*(C, a')$$

$$Q_*(A, c) = R(A, c) + \gamma^1 \max_{a'} Q_*(A, a')$$

$$Q_*(B, a) = R(B, a) + \gamma^1 \max_{a'} Q_*(C, a')$$

$$Q_*(B, b) = R(B, b) + \gamma^1 \max_{a'} Q_*(B, a')$$

⋮

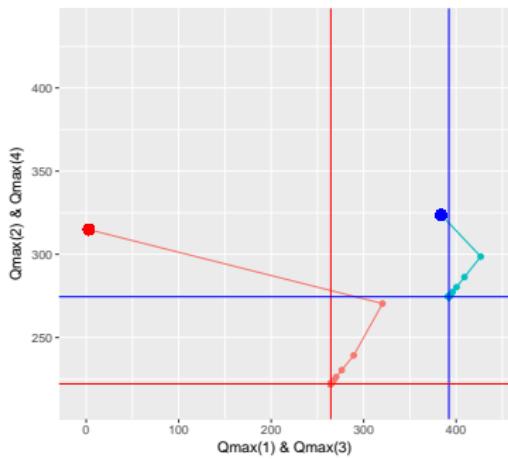
위의 방정식은 역행렬을 활용하여 풀 수 없지만, 반복적인 갱신을 활용하여 풀 수 있다!

반복적인 갱신의 예

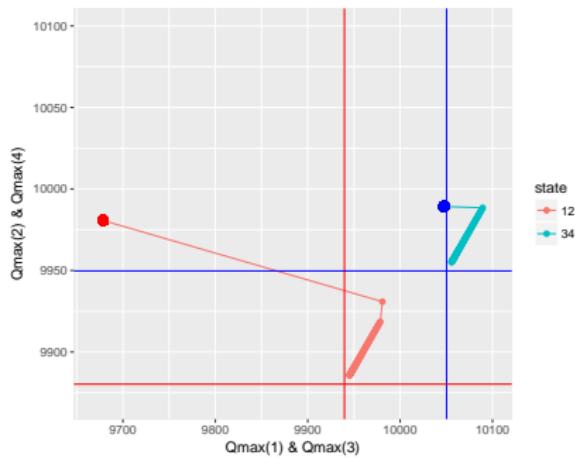
상태 state	행동 action		
	a	b	c
A	100($\rightarrow B$)	-10($\rightarrow C$)	-20($\rightarrow C$)
B	0($\rightarrow A$)	50($\rightarrow B$)	30($\rightarrow D$)
C	200($\rightarrow D$)	70($\rightarrow A$)	50($\rightarrow B$)
D	-100($\rightarrow C$)	0($\rightarrow A$)	0($\rightarrow C$)

반복적인 갱신의 예: γ 의 효과

gamma = 0.7



gamma = 0.99



확률적 환경, 확률적 정책에서 벨만 방정식

상태 가치 함수

$$\begin{aligned} V_1(s) &= \mathbb{E}_{S' \sim p(S'|s, a'), a' \sim \pi_1(a'|s)} \left[R(s, a') + \gamma V_1(S') | S = s \right] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \left[r + \gamma V_1(s') \right] \end{aligned}$$

상태-행동 가치 함수

$$\begin{aligned} Q_1(s, a) &= \mathbb{E}[R(s, a)] + \gamma \mathbb{E}_{s' \sim \mathbb{P}(s'|s, a)} \left[V(s') \right] \\ &= \mathbb{E}[R(s, a)] + \gamma \mathbb{E} \left[\sum_{a'} \pi(a'|s') \sum_{s'', r} p(s'', r|s', a) [r + \gamma V_1(s'')] \right] \\ &= \mathbb{E}[R(s, a)] + \gamma \mathbb{E} \left[\sum_{a'} \pi(a'|s') Q_1(s', a') \right] \end{aligned}$$

결정론적 환경의 예: 벨만 가치 함수 방정식

$$\begin{aligned}
 V_1(A) &= R(A, a) + \gamma \cdot R(B, a) + \gamma \cdot \gamma^1 R(C, a) + \gamma \cdot \gamma^2 R(A, a) + \gamma \cdot \gamma^3 R(B, a) + \dots \\
 &= R(A, a) + \gamma(R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \dots) \\
 V_1(B) &= R(B, a) + \gamma^1 R(C, a) + \gamma^2 R(A, a) + \gamma^3 R(B, a) + \dots
 \end{aligned}$$

이를 활용하면, 다음과 같은 상태 가치 함수 방정식을 얻는다.

$$V_1(A) = R(A, a) + \gamma V_1(B)$$

$$V_1(B) = R(B, a) + \gamma V_1(C)$$

$$V_1(C) = R(C, a) + \gamma V_1(A)$$

확률적 환경의 예: 벨만 가치 함수 방정식

만약 다음 상태가 결정론적이라면,

$$V_1(A) = R(A, a) + \gamma V_1(B)$$

$$V_1(B) = R(B, a) + \gamma V_1(C)$$

$$V_1(C) = R(C, a) + \gamma V_1(A)$$

만약 다음 상태가 확률적이라면, 전이확률 $\mathbb{P}(S'|S, a)$ 를 이용해서,

$$V_1(A) = R(A, a) + \mathbb{P}(A|A, a)\gamma V_1(A) + \mathbb{P}(B|A, a)\gamma V_1(B) + \mathbb{P}(C|A, a)\gamma V_1(C)$$

$$V_1(B) = R(B, a) + \mathbb{P}(A|B, a)\gamma V_1(A) + \mathbb{P}(B|B, a)\gamma V_1(B) + \mathbb{P}(C|B, a)\gamma V_1(C)$$

$$V_1(C) = R(C, a) + \mathbb{P}(A|C, a)\gamma V_1(A) + \mathbb{P}(B|C, a)\gamma V_1(B) + \mathbb{P}(C|C, a)\gamma V_1(C)$$

환경을 모를 때

상태 전이 확률 $\mathbb{P}(s'|s, a)$ 를 모를 때,
어떻게 주어진 정책에 대한 상태 가치 함수를 구하고,
더 나아가, 최상의 정책을 찾아낼 수 있을까?

주어진 상태에서 주어진 정책에 따라 행동을 여러 번 해보기

몬테카를로 (MC; Monte Carlo) 방법 : 주어진 상태 s 에서 주어진 정책 π 에 따라 행동을 계속함으로써 얻어지는 보상의 할인된 총합 (G)을 구한다. 그리고 이를 여러 번 반복해서 평균을 구한다.

$$G = R_{(0)} + \gamma R_{(1)} + \gamma^2 R_{(2)} \cdots \gamma^T R_{(T)}$$

$$v(s) = \mathbb{E}_{\pi|s}[G]$$

$$\hat{v}(s) = \sum_{i=1}^n g_i / n$$

- 끝이 있는 과제에서만 사용할 수 있는 방법이다.
- 분산이 크다.

환경을 모를 때 주어진 정책에 대한 상태 가치 함수 학습하기

시간차Temporal Difference 방법을 사용하여 상태 가치 함수 갱신update하기

- DP fullbackup

$$\begin{aligned} V_1(s) &= \mathbb{E}_{S' \sim p(S'|s, a'), a' \sim \pi_1(a'|s)} \left[R(s, a') + \gamma V_1(S') | S = s \right] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \left[r + \gamma V_1(s') \right] \end{aligned}$$

- TD update(step-size: α)

$$V_1(s) = V_1(s) + \alpha \left[r + \gamma V_1(s') - V_1(s) \right]$$

벨만 방정식

평균적으로 현재 상태의 가치 함수 값은 즉각 보상과 다음 상태의 가치 함수 값의 합이다.

$$V_1(s) = \mathbb{E} \left[R(s, a') + \gamma V_1(S') \middle| S = s \right]$$

평균적으로 현재 상태-행동의 가치 함수 값은 즉각 보상과 다음 상태-행동 가치 함수 값의 합이다.

$$Q_1(s, a) = \mathbb{E} \left[R(s, a) + \gamma V_1(S', a') \middle| S = s, A = a \right]$$

벨만 최적 방정식

최적의 정책을 가정할 때, 현재 상태-행동 가치 함수 값은 평균적으로 즉각보상과 다음 상태-최적 행동의 가치 함수 값의 합이다.

$$Q_*(s, a) = \mathbb{E} \left[R(s, a) + \gamma \max_{a'} Q_*(S', a') \middle| S = s, A = a \right]$$

함수 근사 function approximation

Tabular method의 문제점

- ① 메모리를 너무 많이 차지한다.
- ② 일반화(Generalize)를 하지 못한다.
- ③ 연속적인 상태 또는 행동을 다루기 힘들다.

근사 함수의 종류

- ① 선형 함수 : $V(s) = \theta_1\phi_1(s) + \theta_2\phi_2(s) + \cdots + \theta_k\phi_k(s)$
- ② 비선형 함수: ANN Artificial Neural Network 등

함수 근사시 손실함수

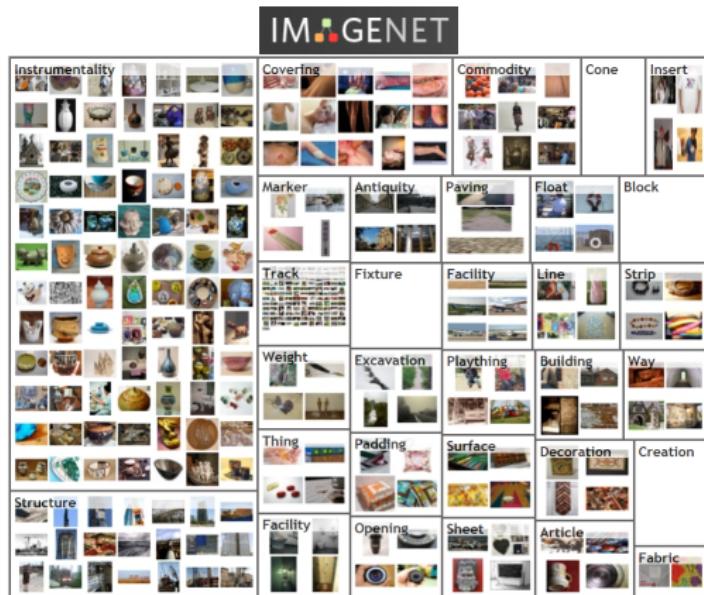
- ① 함수 근사를 사용하지 않을 경우

$$Q_*(s, a) = \mathbb{E} \left[R(s, a) + \gamma \max_{a'} Q_*(S', a') \middle| S = s, A = a \right]$$

- ② 함수 근사를 사용할 경우의 손실 함수

$$\left(Q_*(s, a) - \mathbb{E} \left[R(s, a) + \gamma \max_{a'} Q_*(S', a') \middle| S = s, A = a \right] \right)^2$$

IMAGENET



1,000 object classes, 1,431,167 images

- <http://image-net.org/about-stats>
- <http://image-net.org/explore>

연도별 ILSVRC 결과

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

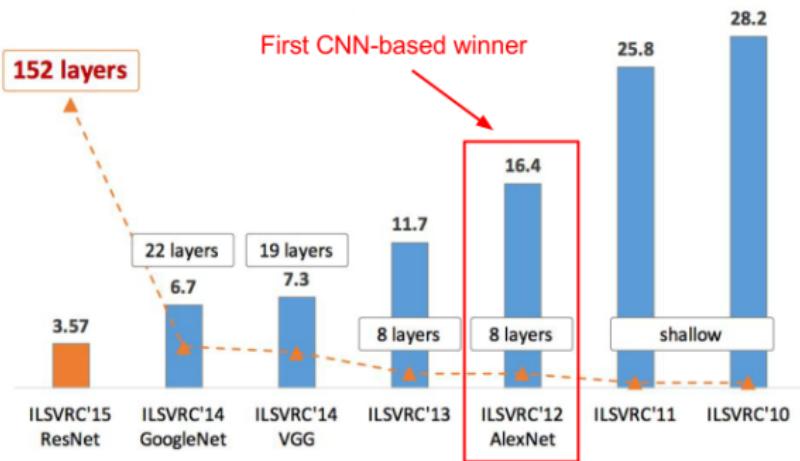
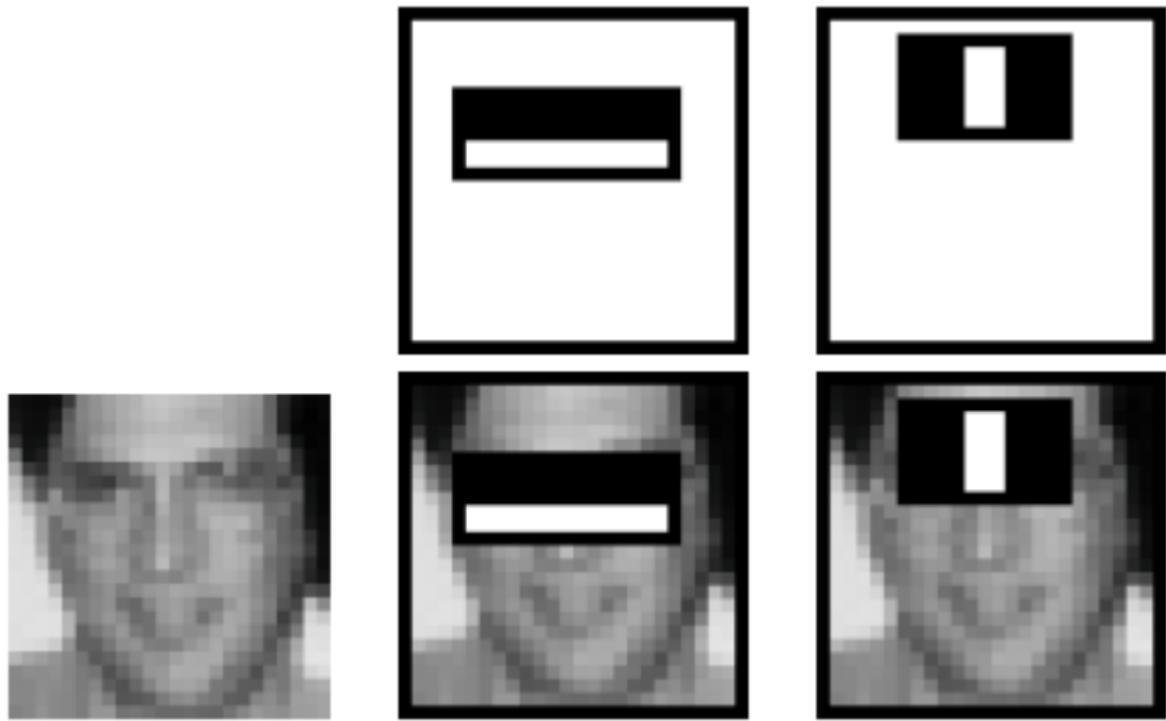


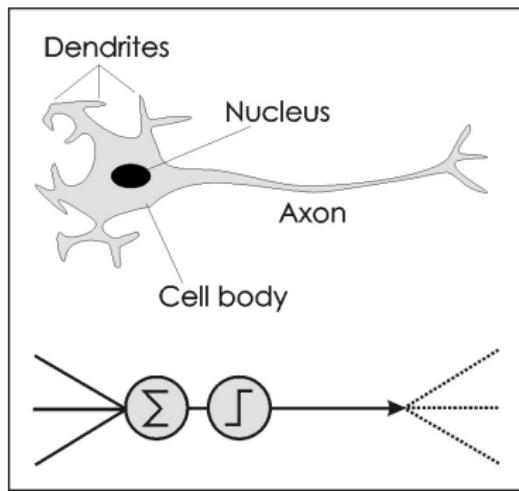
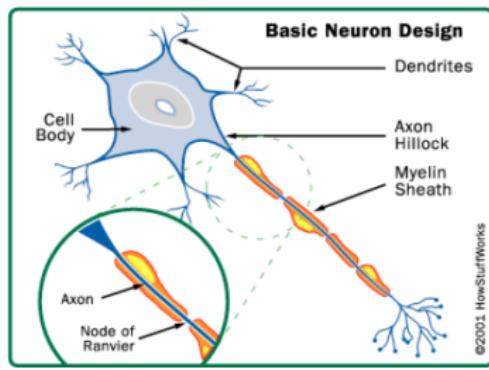
Figure copyright Kaiming He, 2016. Reproduced with permission.

전통적인 시각 처리 방법 : hand-crafted features

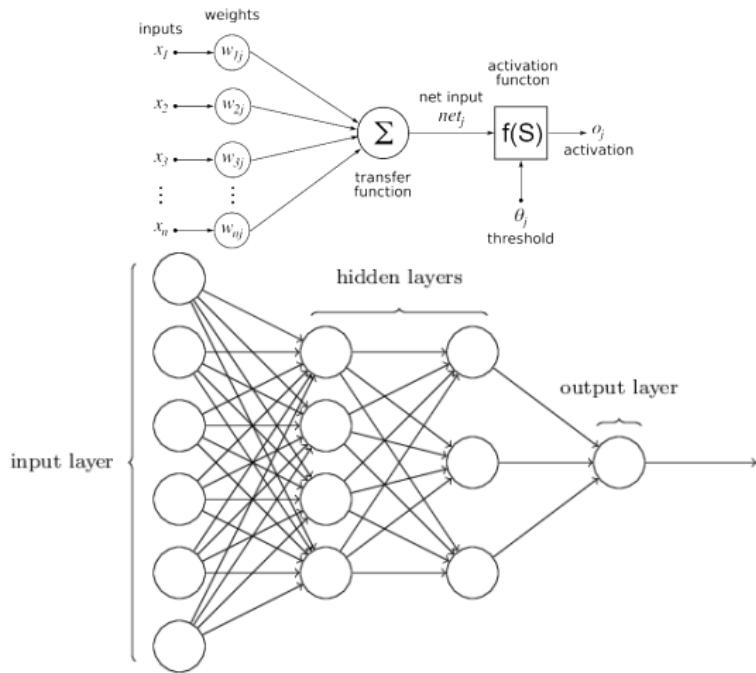
예) 얼굴 인식 (Viola & Jones, 2001)



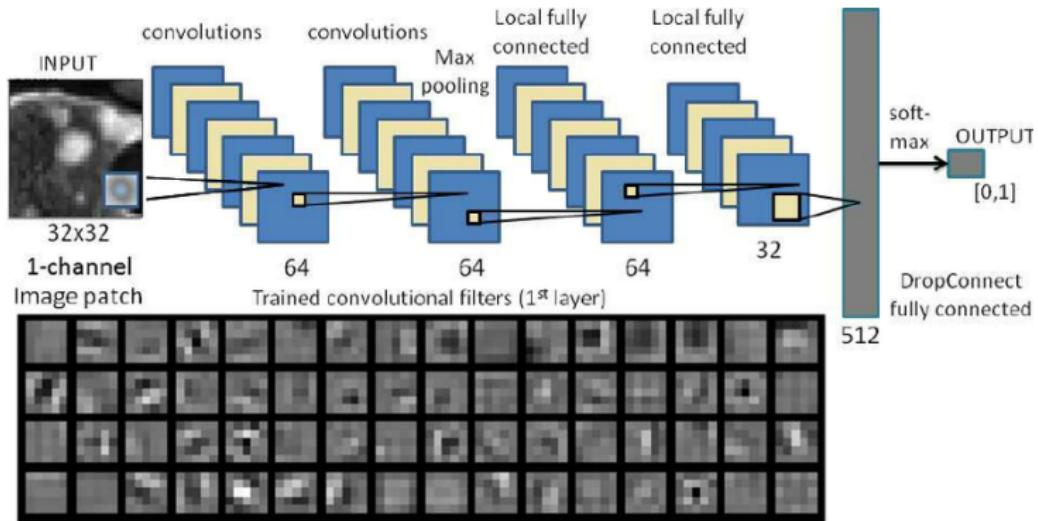
인공신경 Artificial Neuron



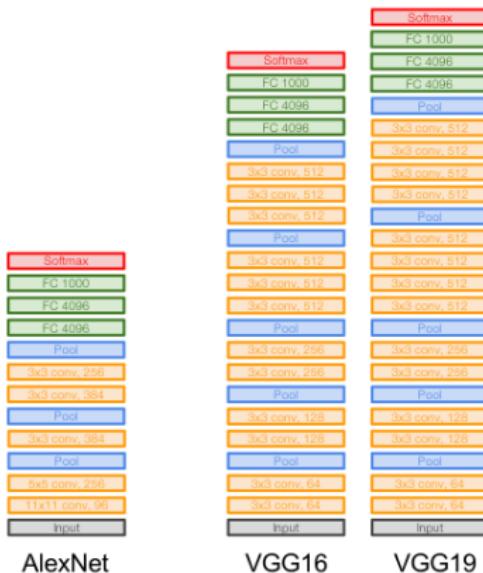
인공신경망 Artificial Neuronal Network



Convolution Filters

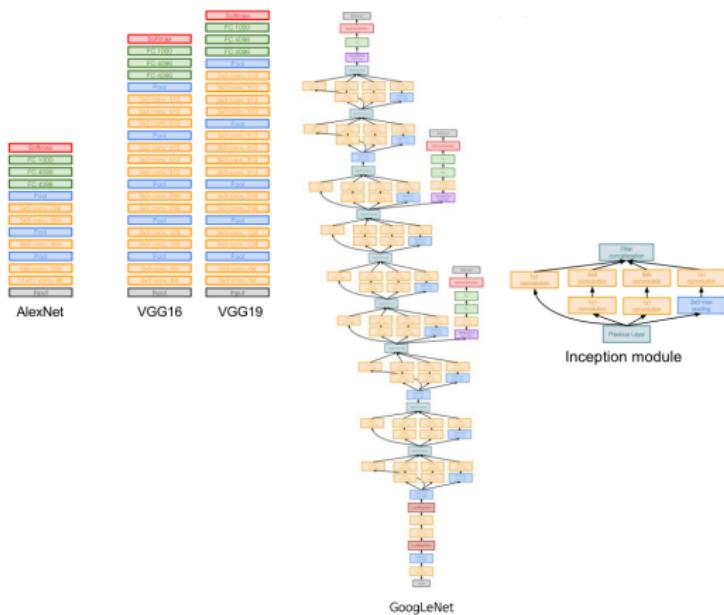


AlexNet, VGG-Net



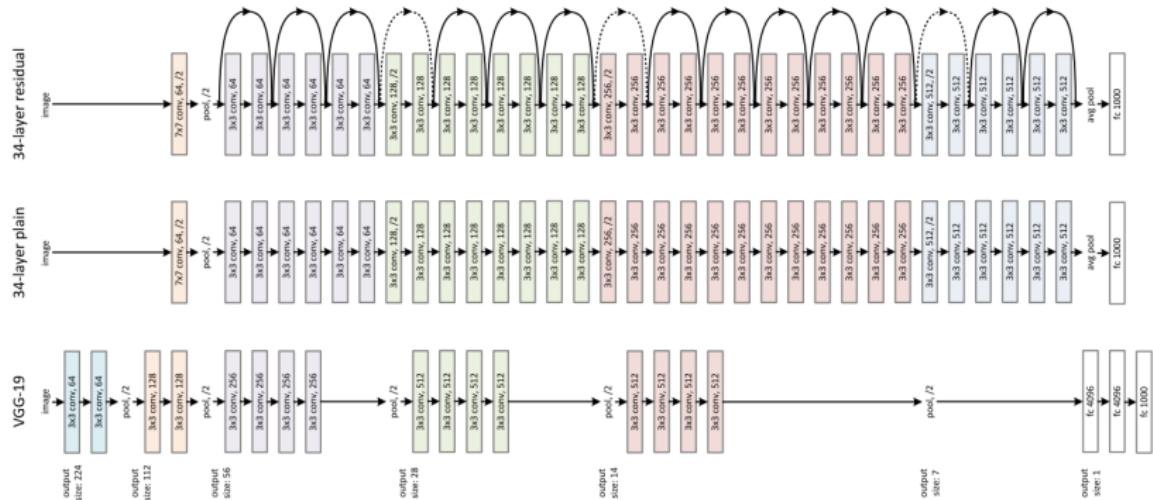
- 5x5 conv. 256 = 5x5 convolution with 256 filters
- FC 1000 = Fully Connected with 1000 nodes

GoogLeNet

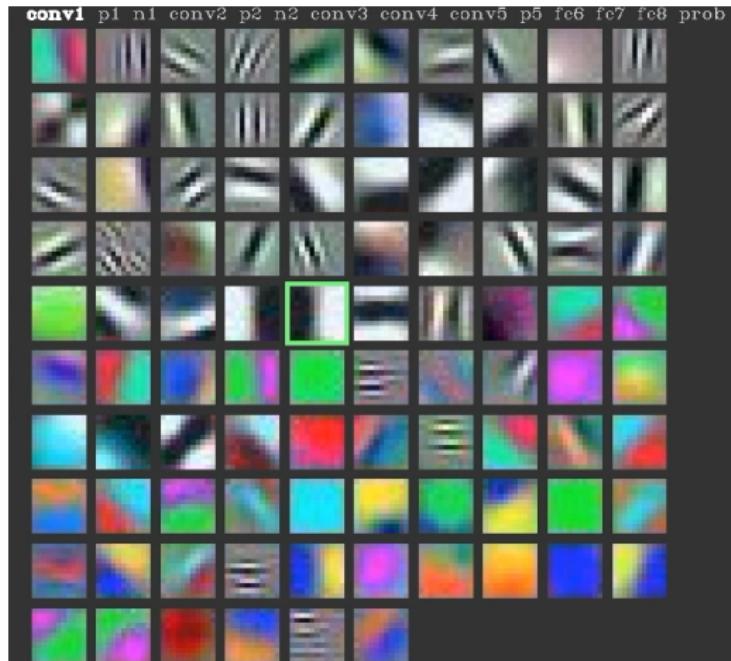


- Going deeper with convolution

ResNet



알렉스 넷(Alexnet)의 말단 필터들



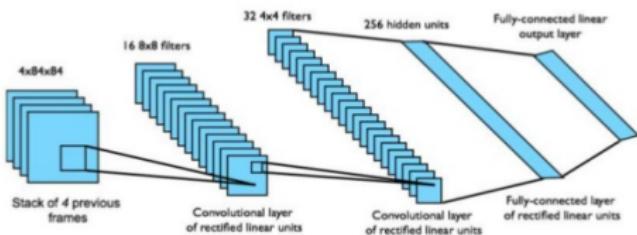
- Deep Visualization Toolbox ►

강화학습과 심층신경망의 결합: DQN

DQN(Deep Q-Network)



DeepMind Atari Deep-Q Network



Layer	Input	Filter size	Stride	Num filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			18	Linear	18

강화학습의 두 줄기(반복)

- 가치 기반(Value based)
- 정책 기반(Policy based)
- 액터-크리틱(Actor-Critic)

심리학, 신경과학

최근의 발전 방향

DQN의 개선 : Rainbow

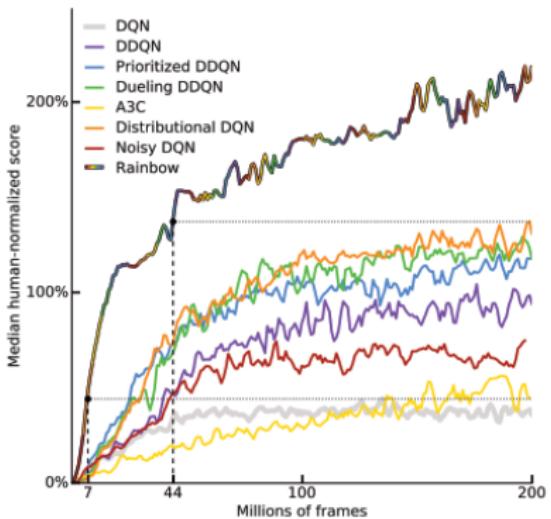
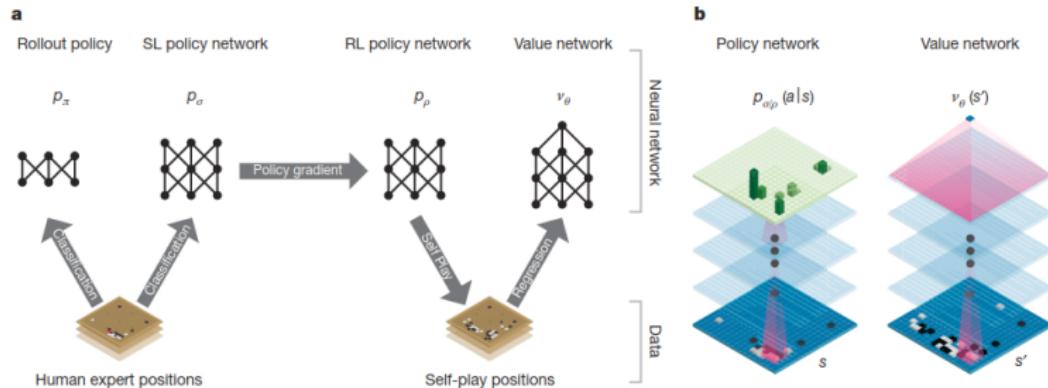


Figure 1: **Median human-normalized performance** across 57 Atari games. We compare our integrated agent (rainbow-colored) to DQN (grey) and six published baselines. Note that we match DQN's best performance after 7M frames, surpass any baseline within 44M frames, and reach substantially improved final performance. Curves are smoothed with a moving average over 5 points.

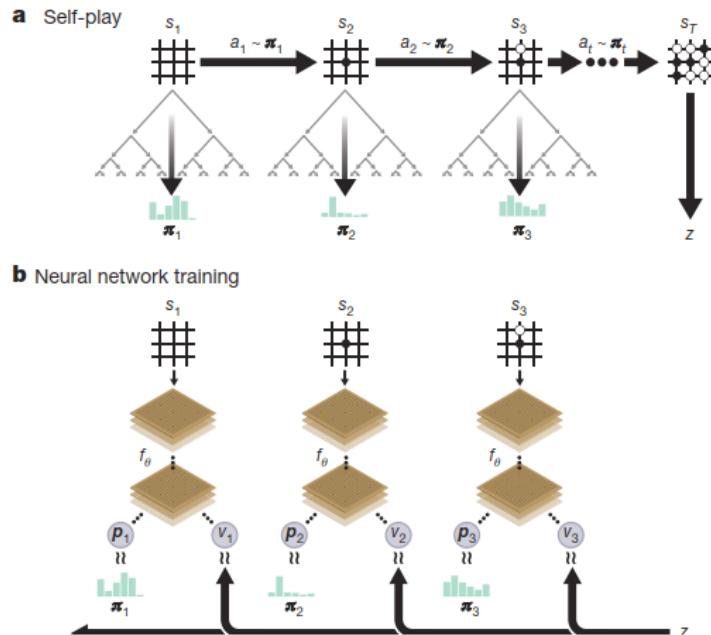
알파고에서 알파고 제로로의 진화

알파고



알파고에서 알파고 제로로의 진화

알파고에서 알파고 제로(AlphaGo Zero)로의 진화



체스까지 정복한 알파고

2.01815v1 [cs.AI] 5 Dec 2017

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

David Silver,^{1*} Thomas Hubert,^{1*} Julian Schrittwieser,^{1*}
Ioannis Antonoglou,¹ Matthew Lai,¹ Arthur Guez,¹ Marc Lanctot,¹
Laurent Sifre,¹ Dharshan Kumaran,¹ Thore Graepel,¹
Timothy Lillicrap,¹ Karen Simonyan,¹ Demis Hassabis¹

¹DeepMind, 6 Pancras Square, London N1C 4AG.

*These authors contributed equally to this work.

Abstract

The game of chess is the most widely-studied domain in the history of artificial intelligence. The strongest programs are based on a combination of sophisticated search techniques, domain-specific adaptations, and handcrafted evaluation functions that have been refined by human experts over several decades. In contrast, the *AlphaGo Zero* program recently achieved superhuman performance in the game of Go, by *tabula rasa* reinforcement learning from games of self-play. In this paper, we generalise this approach into

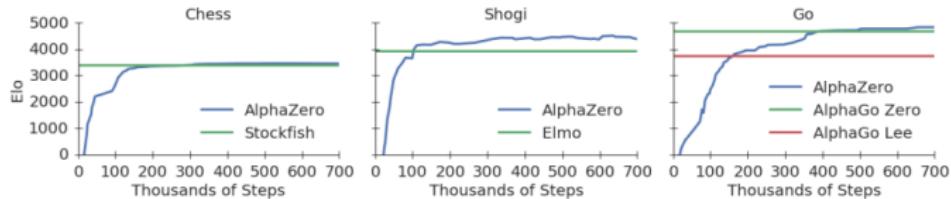


Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation

그 밖에

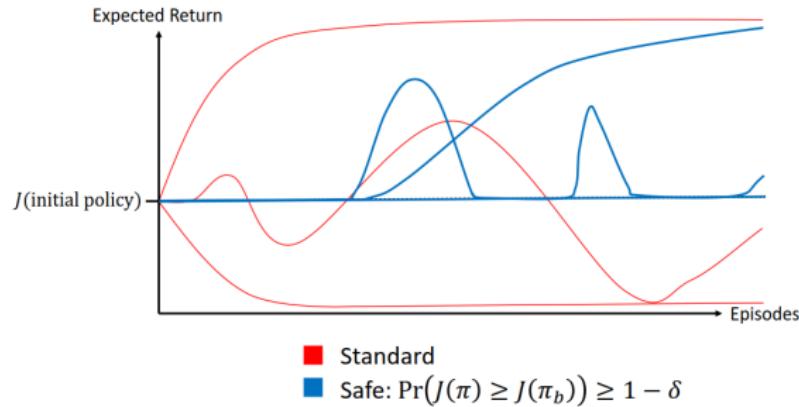
- Apprenticeship Learning(=Inverse Reinforcement Learning)
- Transfer Learning

강화 학습의 응용 분야: 로보틱스

- Juggling Pingpong ball ►
- A Ball in a cup ►
- Robots Learn ►
- Deep LocoPaper ►
- End-to-End Driving ►

강화 학습의 응용 분야

- 자연언어처리
- 경영/재무/투자
- 의료 : safe RL



- 에너지 ►

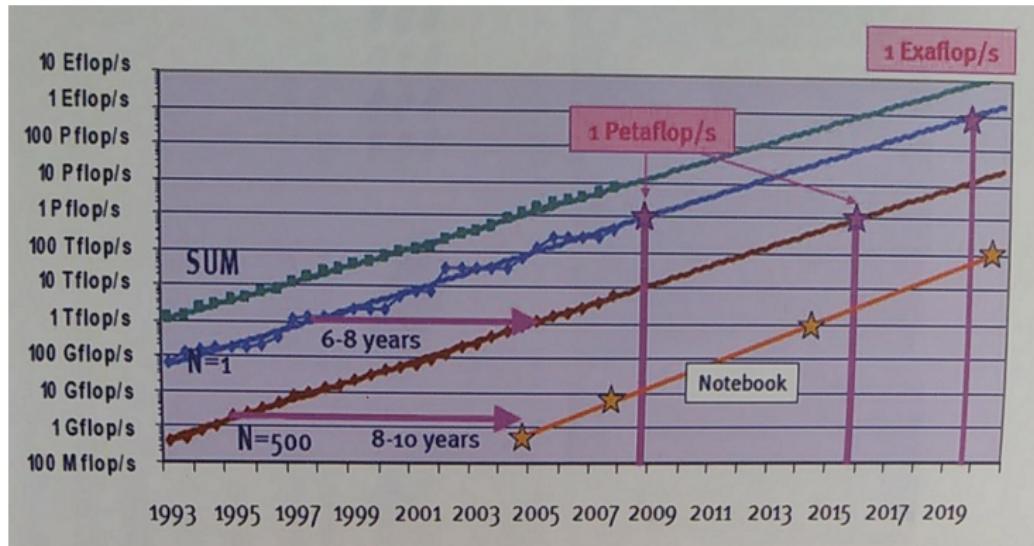
마무리

딥블루 이후의 체스, 알파고 이후의 바둑



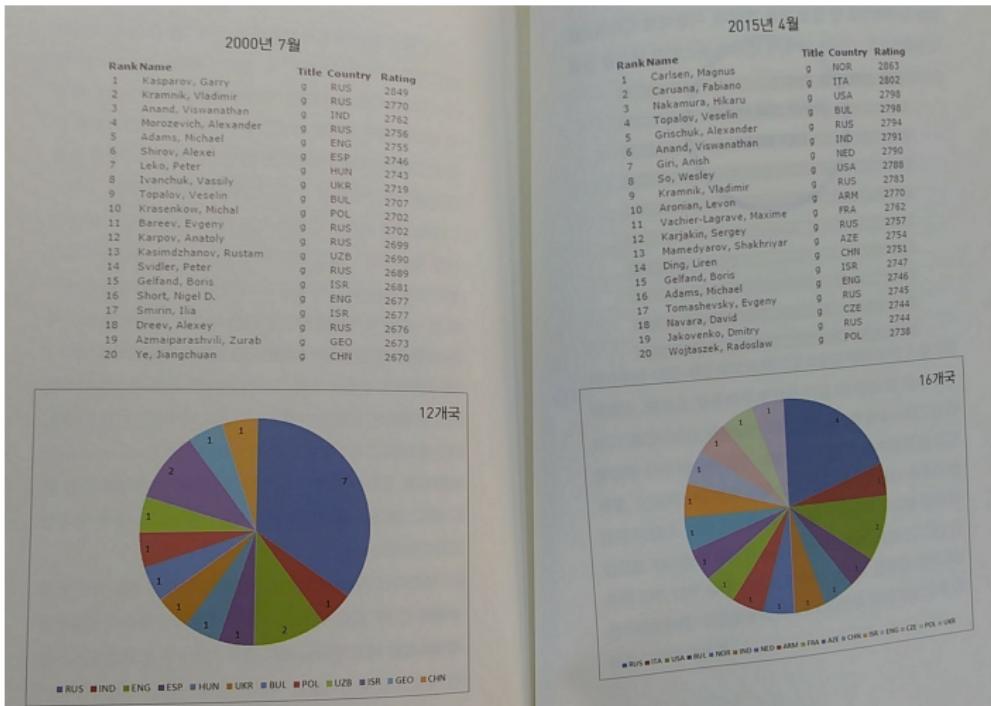
출처: 바둑으로 읽는 인공지능(감동근, 2016)

딥블루 이후의 체스, 알파고 이후의 바둑



출처: 바둑으로 읽는 인공지능(감동근, 2016)

딥블루 이후의 체스, 알파고 이후의 바둑



출처: 바둑으로 읽는 인공지능(감동근, 2016)

감사합니다!