2019 CDE 학회
튜토리얼 세션

강화학습 개론 및 실습

홍익대학교 산업공학과
강윤철 교수
yckang@hongik.ac.kr

참고자료:
[1] Fei-Fei Li & Justin Johnson & Serena Yeung Lecture14-1, cs231n, Stanford University
[2] Sutton and Barto, Introduction to Reinforcement Learning

튜토리얼 순서

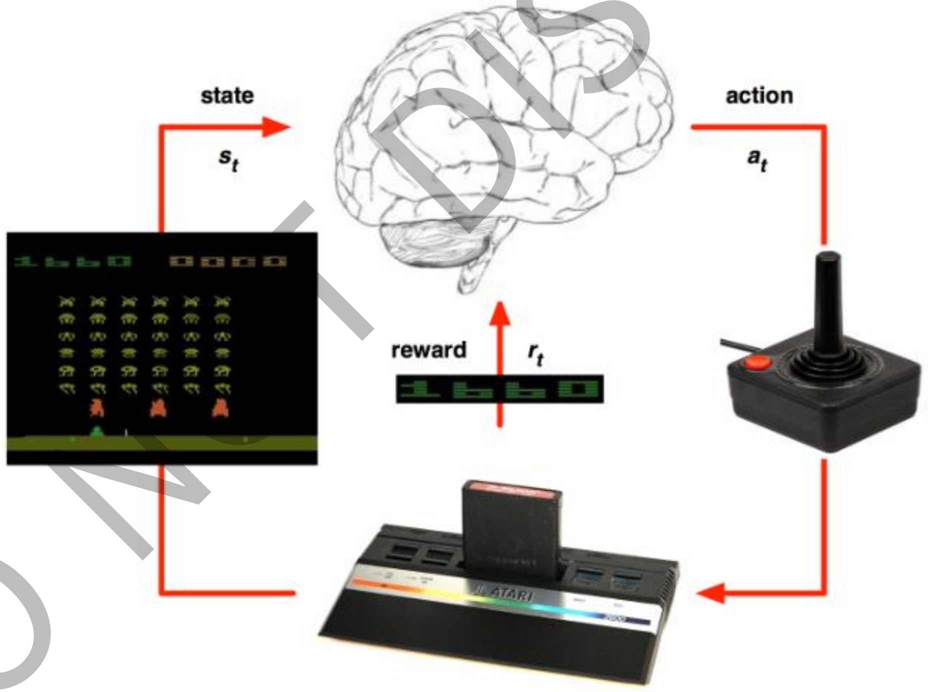
1. 강화학습 기초
2. 구성 이론
3. 최신 경향
4. 예제
   - Google Colab 을 활용한 실습

1. 강화학습 기초

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture14-1, cs231n, Stanford University

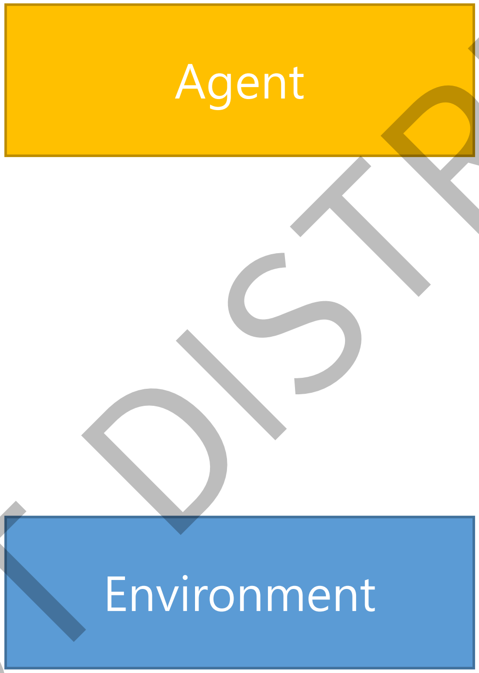
Reinforcement Learning (강화학습)

- 단일 에이전트(agent)가 환경(environment)과 지속적으로 상호작용하며 보상(reward)을 받게 되는 상황
- 목표: 보상(Reward)를 최대화(maximize)하기 위해 에이전트는 어떻게 **Action(행동;의사결정)**을 취할 것인가?

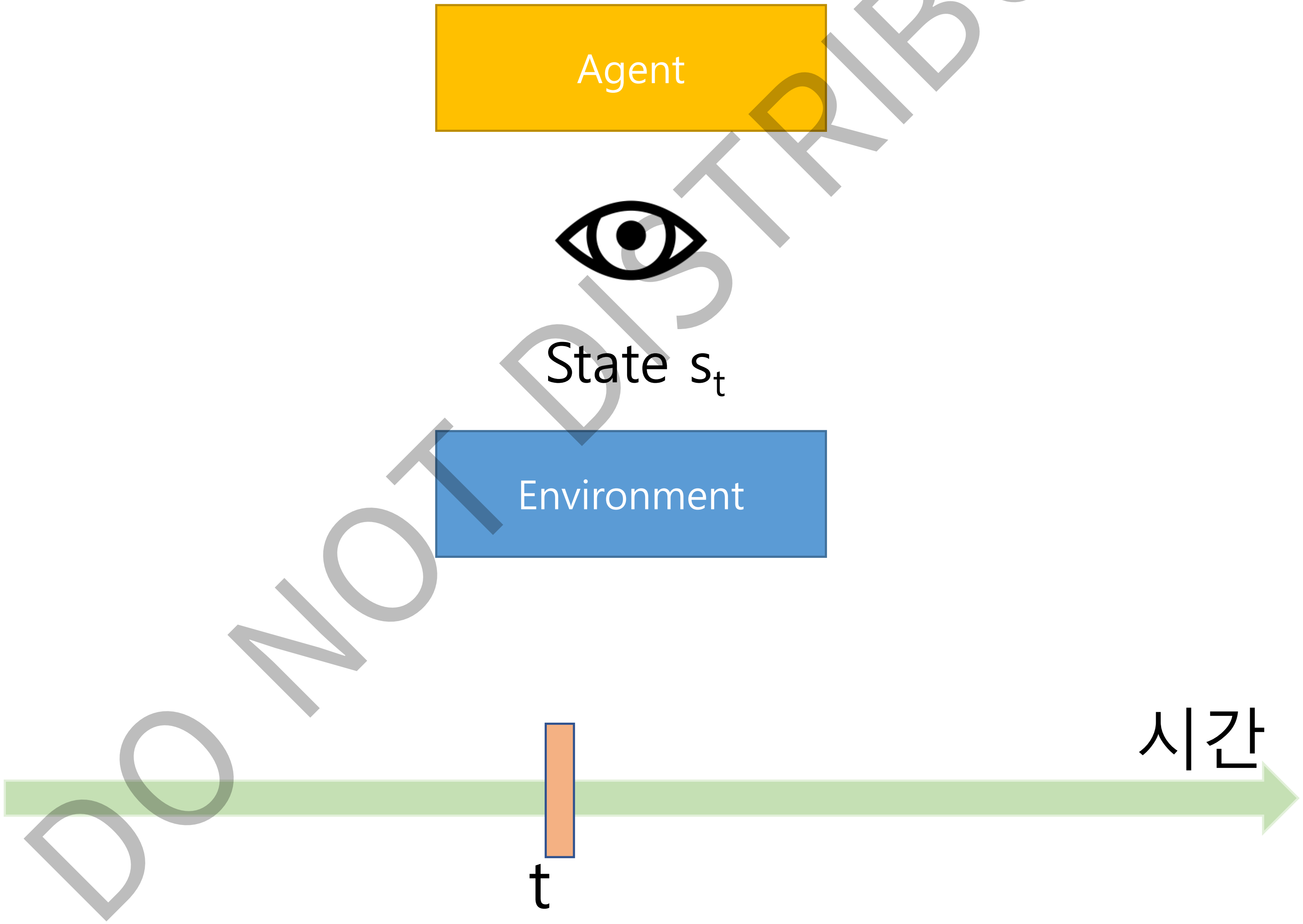


출처:https://keon.io/deep-q-learning/

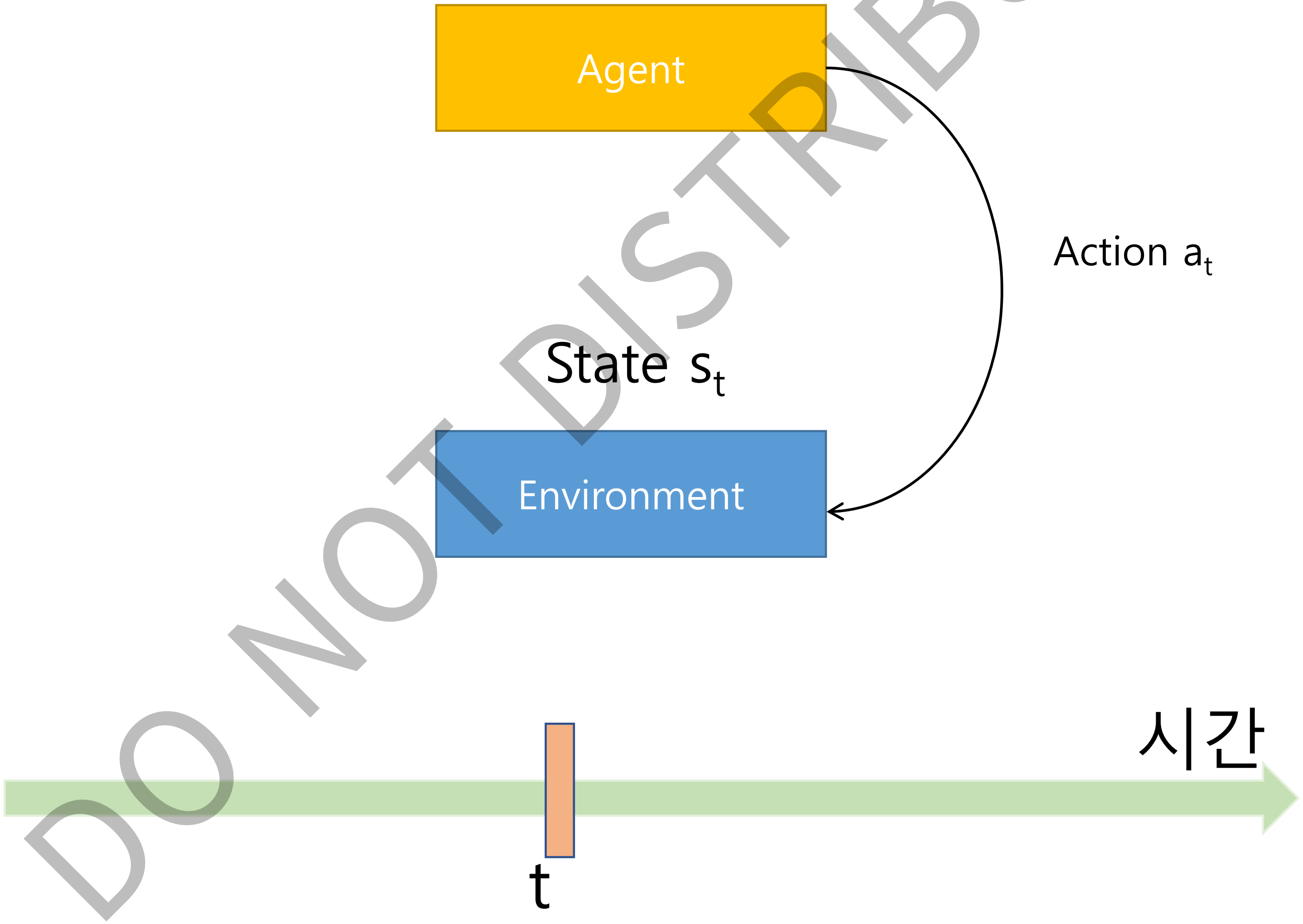
강화학습 기본 골격



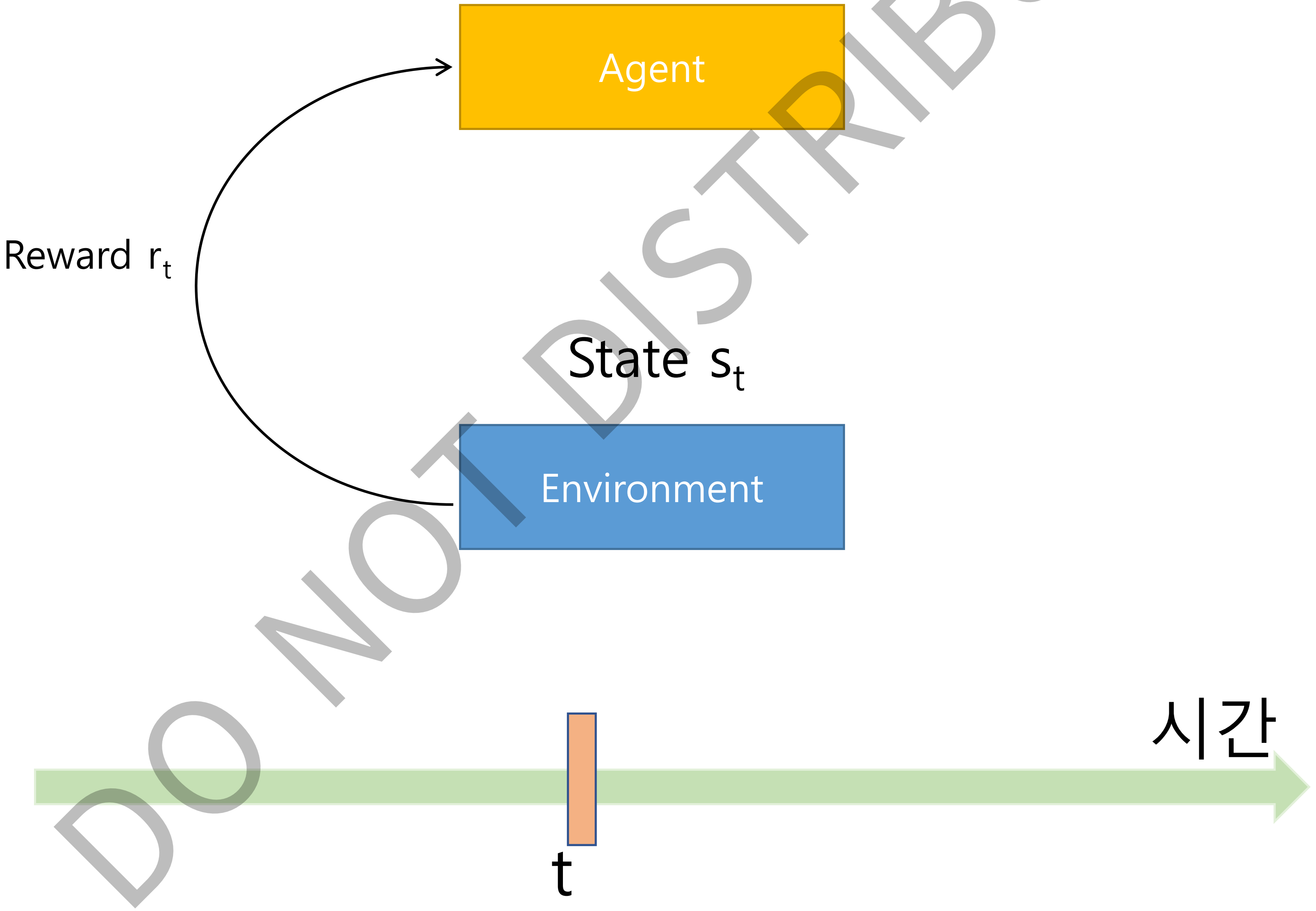
강화학습 기본 골격



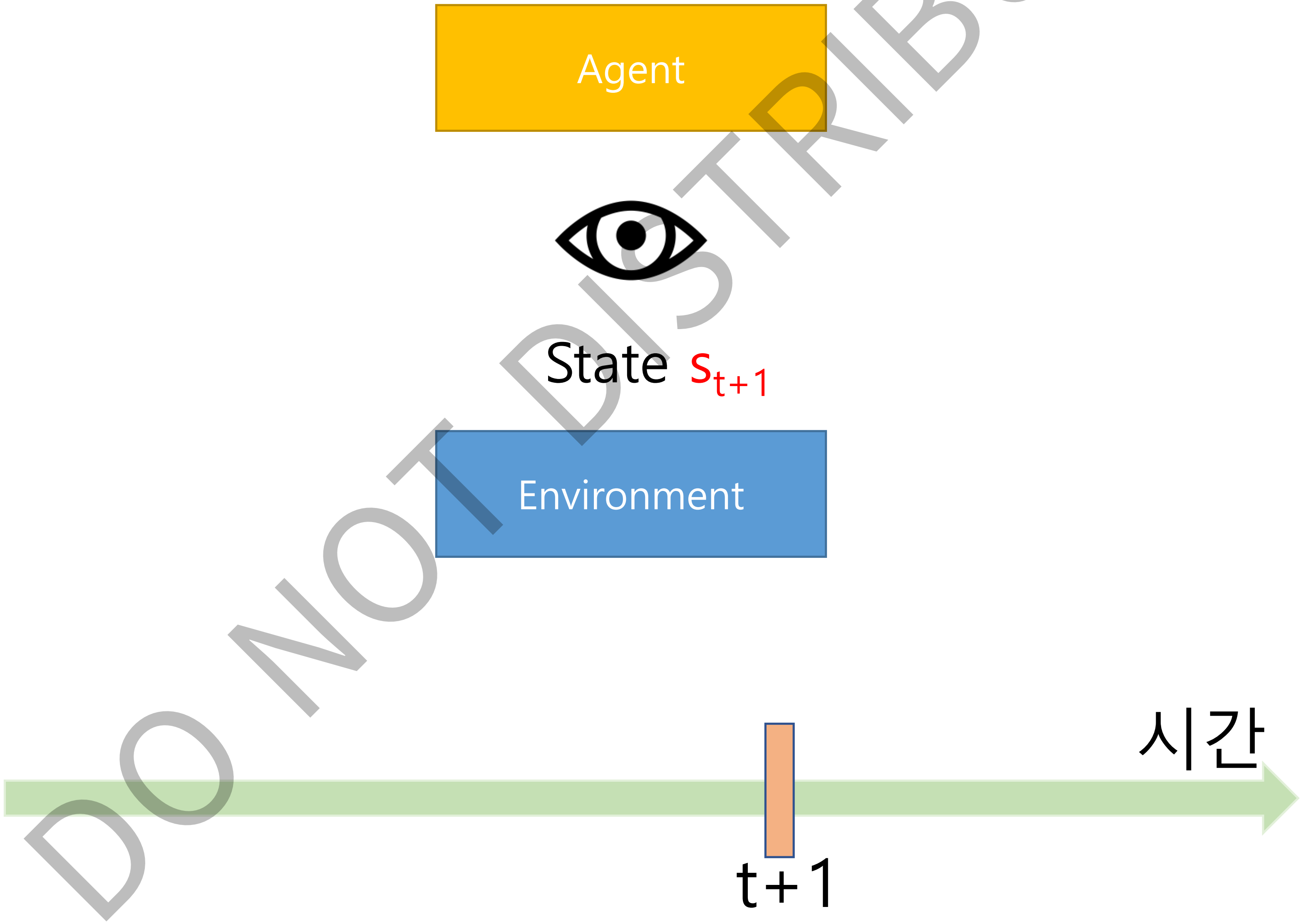
강화학습 기본 골격



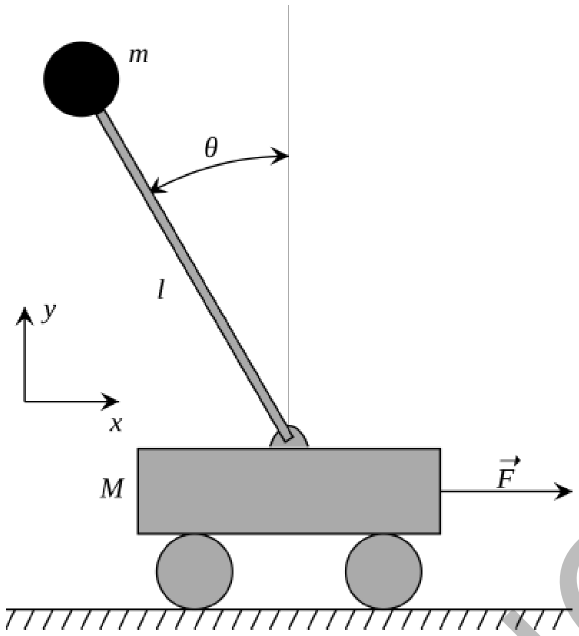
강화학습 기본 골격



강화학습 기본 골격



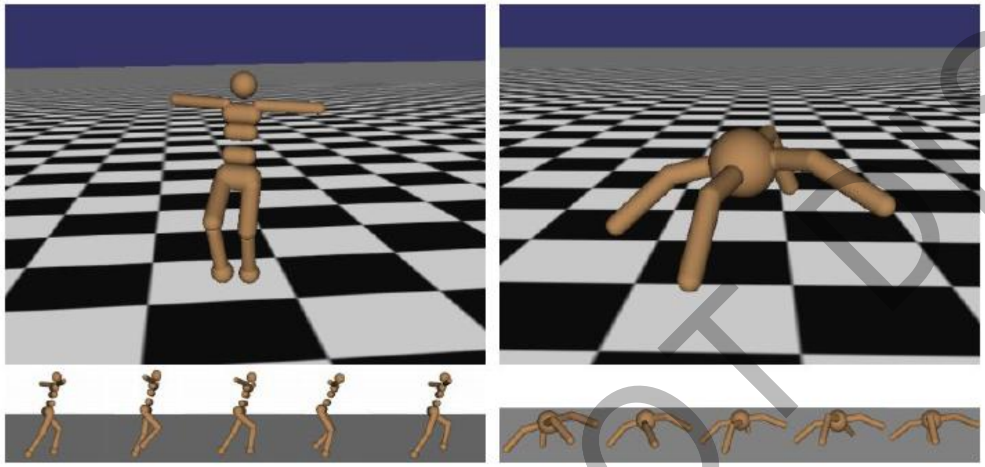
1) Cart-Pole 문제



Objective: Balance a pole on top of a movable cart

State: angle, angular speed, position, horizontal velocity
Action: horizontal force applied on the cart
Reward: 1 at each time step if the pole is upright

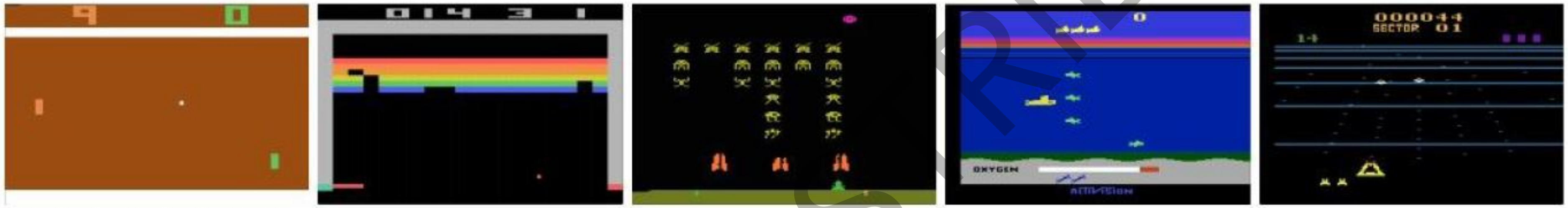
2) Let the robot move!



Objective: Make the robot move forward

State: Angle and position of the joints
Action: Torques applied on joints
Reward: 1 at each time step upright + forward movement

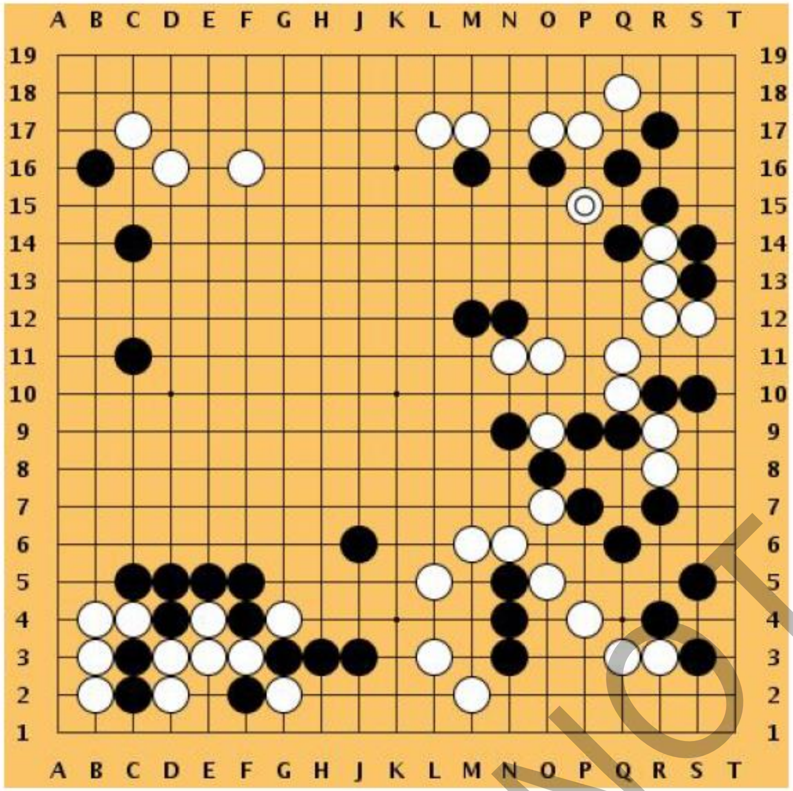
3) Atari Games



Objective: Complete the game with the highest score

State: Raw pixel inputs of the game state
Action: Game controls e.g. Left, Right, Up, Down
Reward: Score increase/decrease at each time step

4) Go



Objective: Win the game!

State: Position of all pieces
Action: Where to put the next piece down
Reward: 1 if win at the end of the game, 0 otherwise

2. 강화학습 구성이론

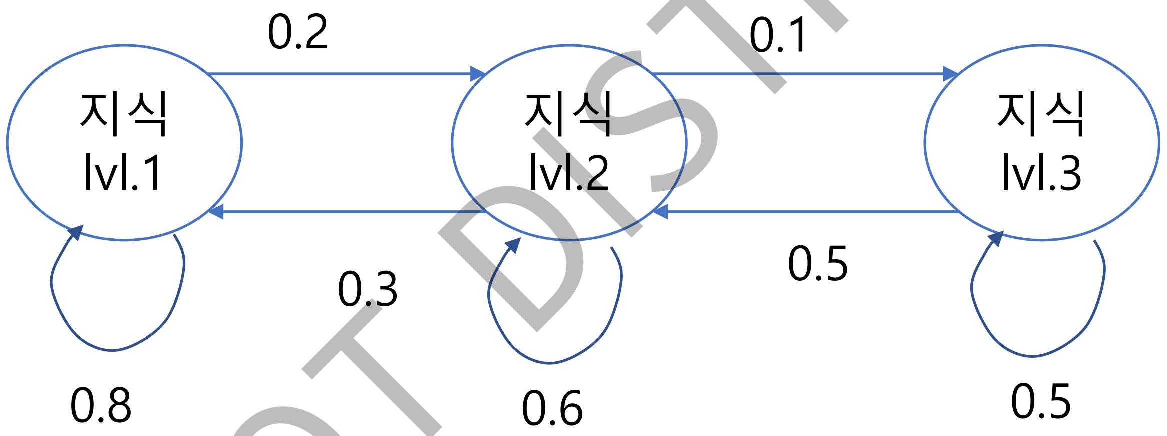
강윤철, 인공지능응용,
홍익대학교, 2018

강화학습 구성이론 설명 순서

1. Markov Chain
2. Markov Decision Process (MDP)
3. How to solve MDP
4. Difficulties in solving MDP
5. Approximate Dynamic Programming
6. Reinforcement Learning (RL)
7. RL을 이해하기 위한 핵심 개념들 소개
8. How to solve RL
9. Difficulties in solving RL

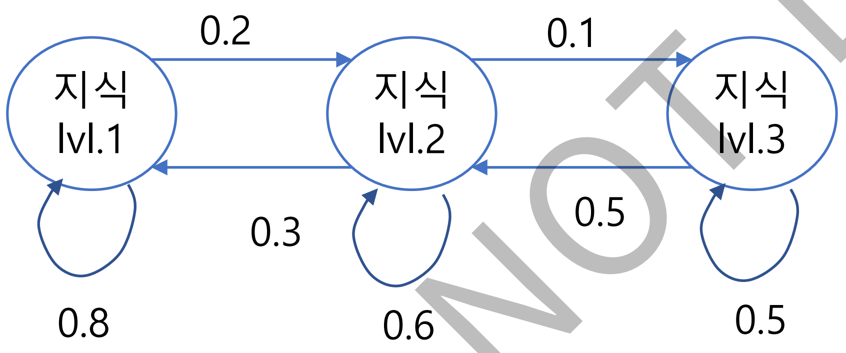
Markov Chain

인공지능응용을 수강하는 어떤 한 학생의 지식 수준 상태

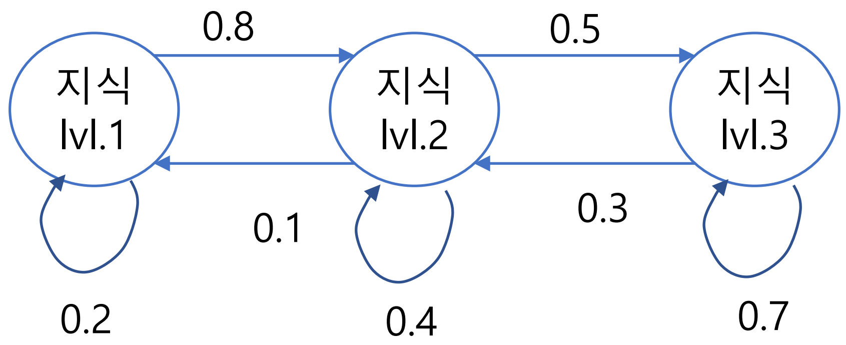


Markov Decision Process (MDP)

- Dynamic Programming (DP)의 한 분야
- Markov Decision Process (MDP) is a discrete time stochastic control process.
- MDP deals with multiple number of Markov Chain.
- For a given time, your state will be affected by one of the Markov chains in the MDP model, depending on your action!



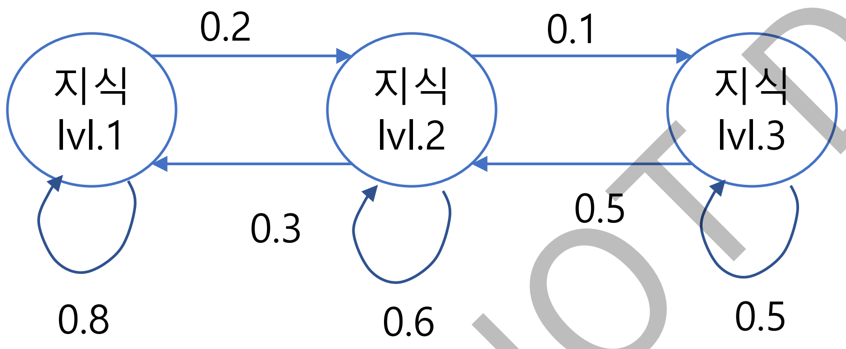
"놀기" 의 Markov Chain



"공부" 의 Markov Chain

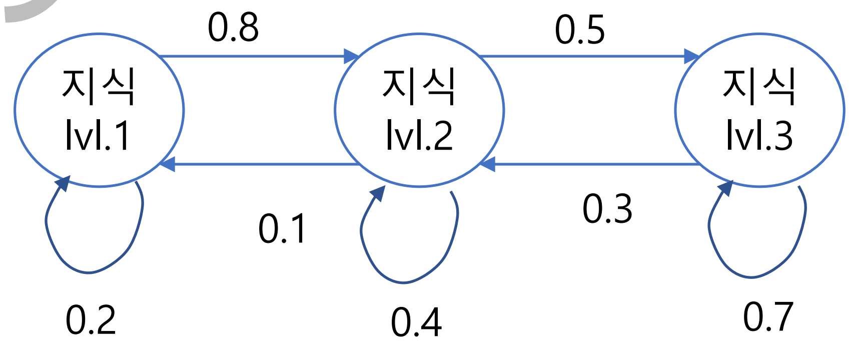
Markov Decision Process (MDP)

- What would you get from each your action?
- From your action, what's the gain and/or loss?
- What's your goal (i.e. objective function)?
- What would be the optimal "policy" for achieving your goal?



"놀기" 의 Markov Chain

컨디션: + 1점

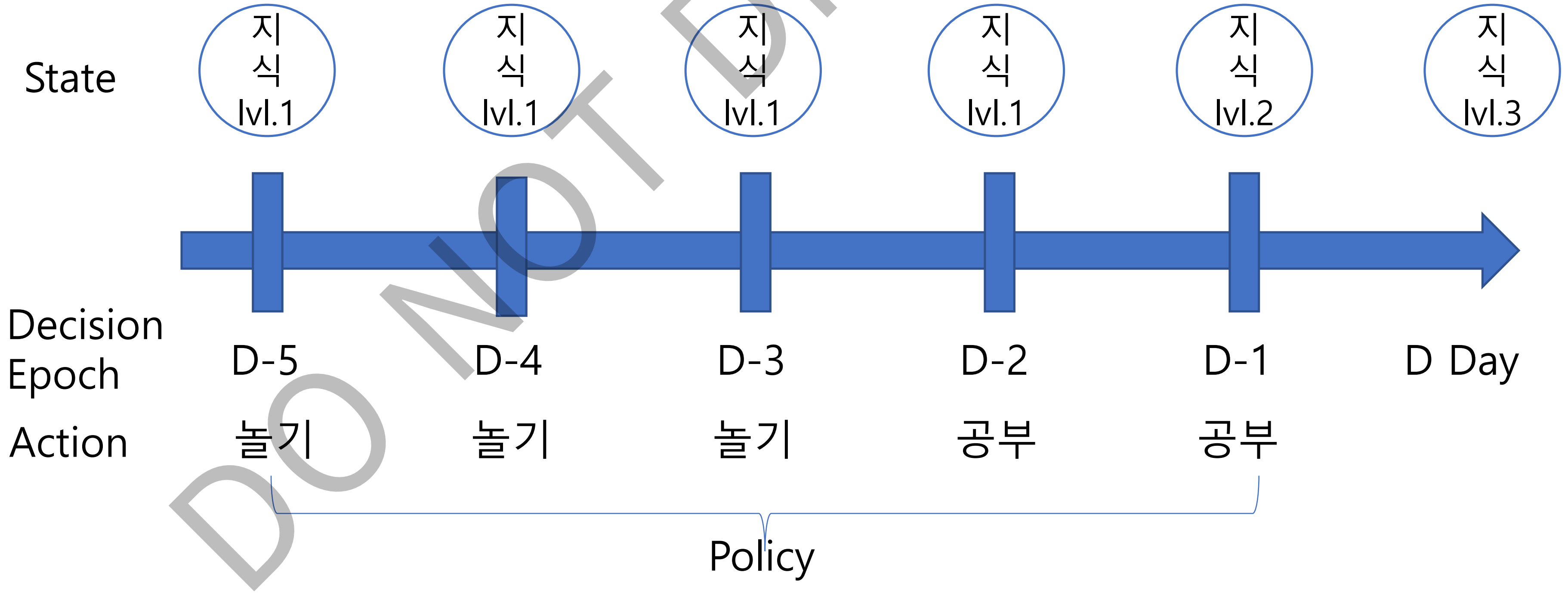


"공부" 의 Markov Chain

컨디션: - 1점

Markov Decision Process (MDP)

- Reward = 30*지식레벨 + 2*컨디션
- Let’s say your objective is to score as much as you can!
- The higher reward you get, the higher score you would get!
- So, maximizing your (discounted, overall, average) reward throughout the planning horizon would be your goal for this problem.



MDP 구성요소

- Elements of MDP
  - State (at time t): $s_t \in S$
  - Transition Probabilities: $p(s_{t+1} = j | s_t = i)$
  - Action: $a \in A$
  - Reward: $r(s_t, a)$
  - Planning Horizon: finite or infinite
  - Decision epoch: the moment you make a decision

- At the decision epoch (current time=t)
  - Observe the current state: $s_t = i$
  - Take an action, $a \in A$
  - Receive immediate reward $r(s_t, a)$
  - (According to the $p(s_{t+1} = j | s_t = i)$), proceed to the next state: j

- Policy = a series of actions

How to solve: Bellman's Equation

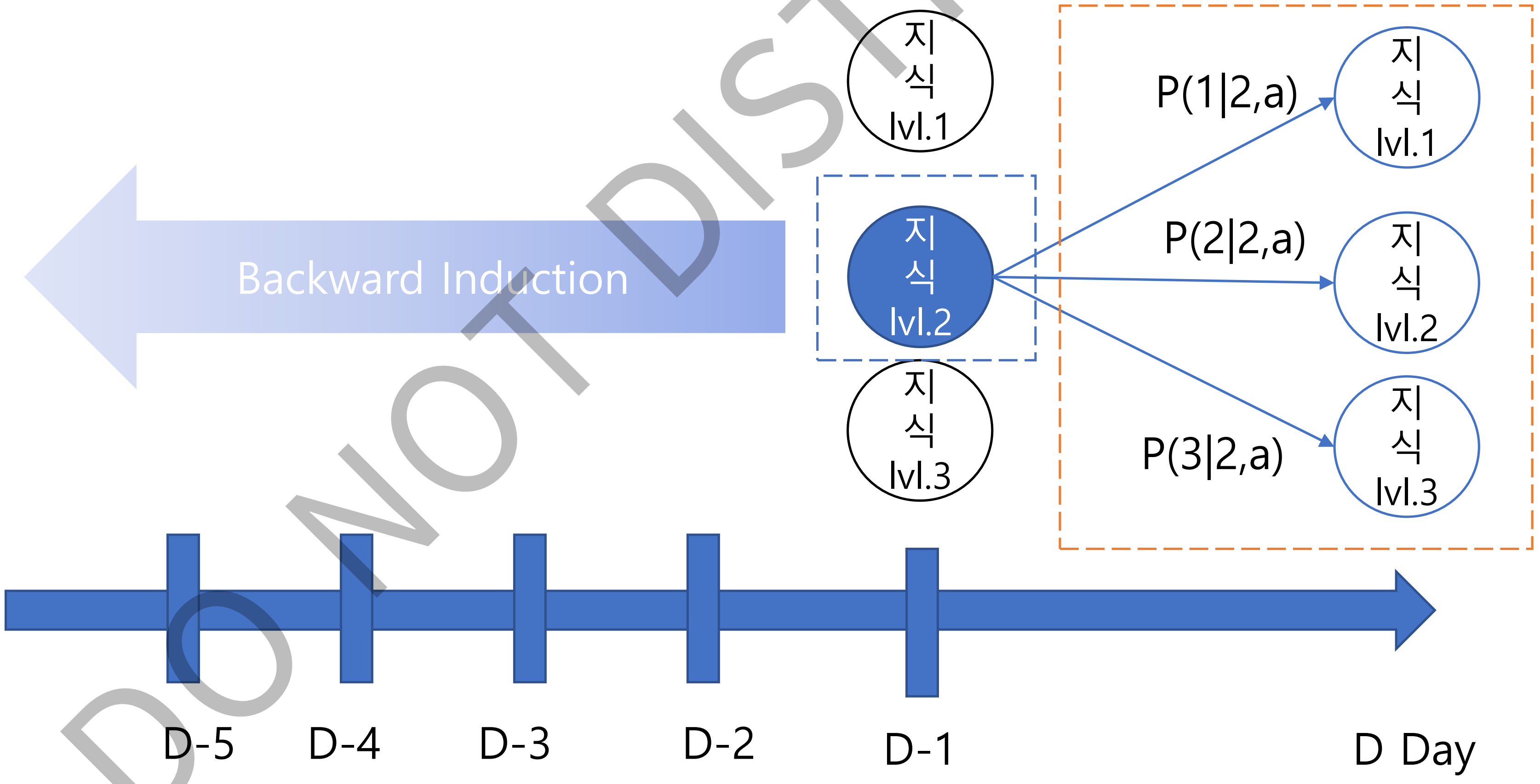
- "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" – Bellman's the Principle of Optimality(1957)
- "There exists a policy that is optimal for every state, at every stage" – Denardo(1982)

$$u_t^*(s_t) = \max_{a \in A_{s_t}} \{r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j)\}$$

$$d_t^*(s_t) = arg \max_{a \in A_{s_t}} \{r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j)\}$$

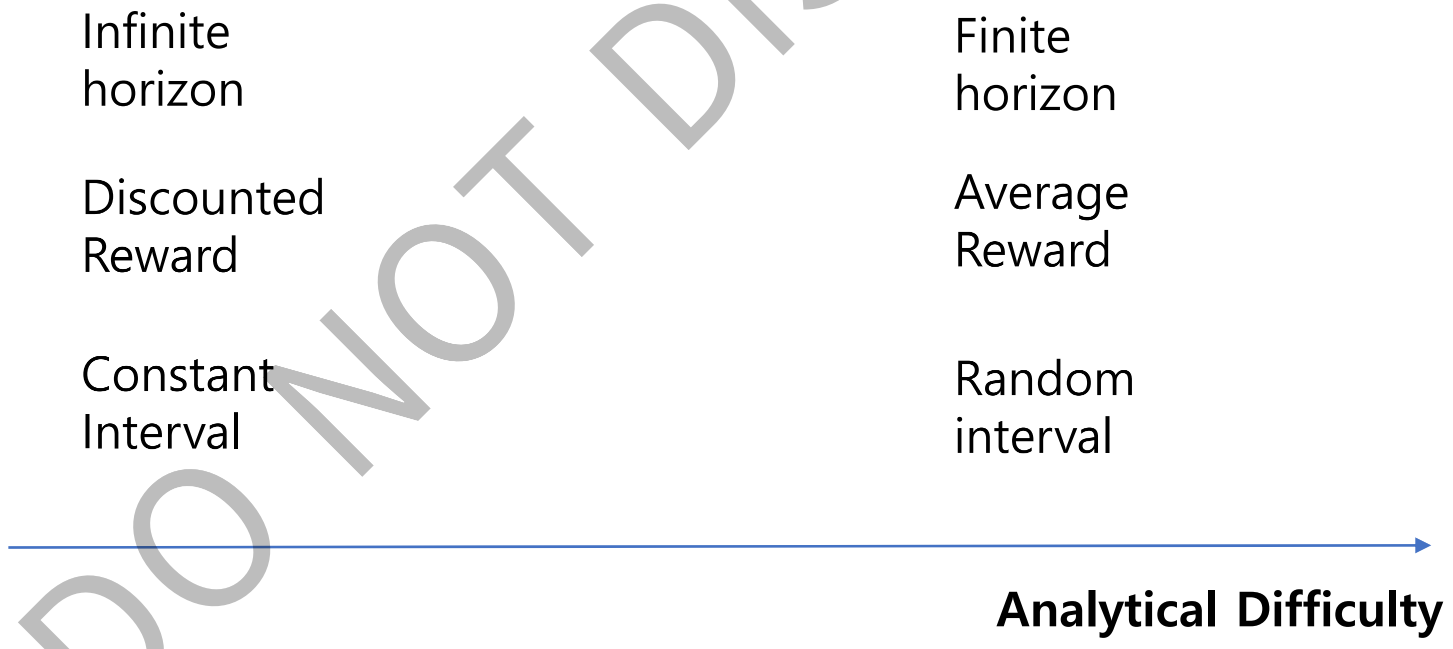
Bellman's Equation (finite horizon)

$$u_t^*(s_t) = \max_{a \in A_{s_t}} \left\{ \boxed{r_t(s_t, a)} + \boxed{\sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j)} \right\}$$



Categories of MDP

- Planning horizon: Finite vs Infinite
- Reward Criterion: Discounted Reward, Total Reward, Average Reward
- Decisions Interval: constant(DTMDP), randomly(SMDP, CTMDP)



Infinite horizon MDP

- Value Iteration

$$u^{n+1}(s) = \max_{a\in A_s}\{r(s,a) + \sum_{s'\in S} p(s'|s,a)u^n(s')\}$$

Stop the algorithm when $|u^{n+1} - u^n| < threshold$

- Policy Iteration

Policy Evaluation(PE) → Policy improvement(PI) → go back to policy evaluation until both policy used from PE and policy found in PI are the same

- Linear Programming

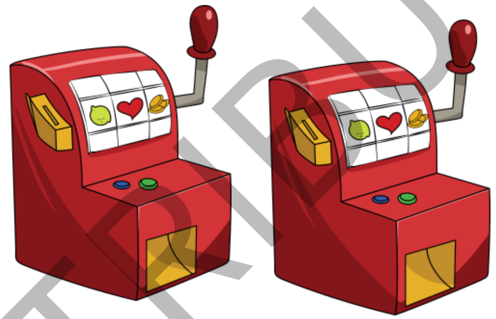
$$\min \sum_{s\in S} \alpha(s)u(s)$$ 단, $\Sigma\alpha(s) = 1$

Subject to $u(s) - \sum_{s'\in S} p(s'|s,a)u^n(s') \geq r(s,a)$

Some popular MDP examples

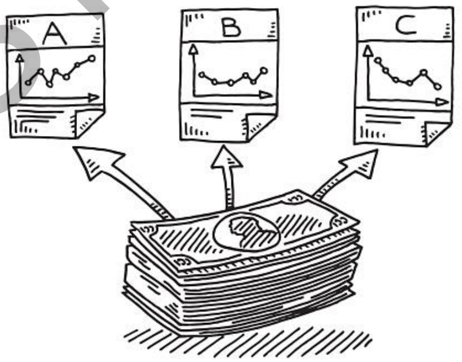
- n-armed Bandit

  State= 각 머신 당 정보(승률 등.)
  Action= 어떤 기계를 선택할 것인가
  Reward= 배당금-배팅금액



- Asset allocation

  State= 남은 자산양
  Action= 이번에 투자할 자산양
  Reward= 이번 수익



- Optimal Stopping (a.k.a secretary problem)

  State=지금 인터뷰하고 있는 사람이 best인지 여부
  Action= Stop or Continue
  Reward=Best 후보를 뽑았을때 얻을 수 있는 기대값

  Solution: Considering STOP after 37% (=1/e) of N

MDP Issues: 이론은 이론일 뿐, 현실은…

- Curse of Dimensionality (차원의 저주)
  - Transition probabilities
    - M states & N actions = M^2*N elements
    - E.g.) 1000 state, 2 actions = 2million elements
- Unable to construct MDP model
  - Uncertain transition probabilities
  - Uncertain reward
  - Uncertain state: fully observable, partially observable (POMDP)
- Continuous state/action space

대부분의 실제문제에 적용하기 힘듦!

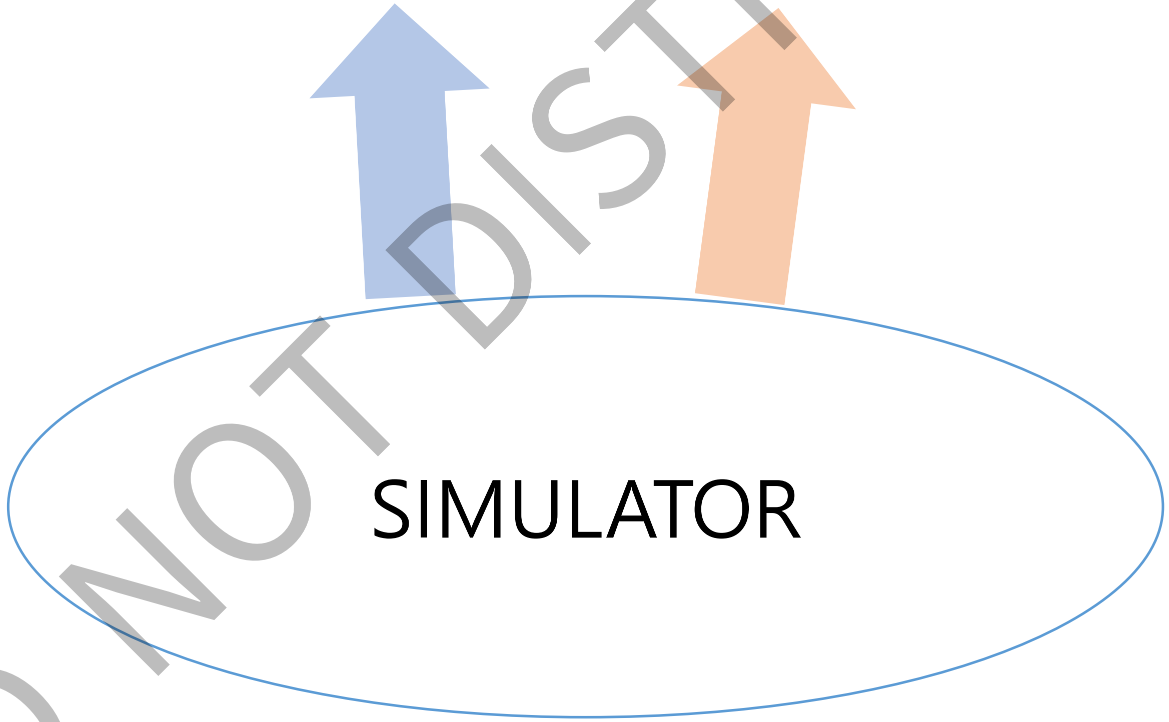
대안 1) Approximate Dynamic Programming

- For communication technology engineers
- For computer scientist (AI researchers)
- Breaking the curses
  - Should we care about the transition probabilities?
  - Should we care about analytical models?
- But, we still need a “good” solution
  - How about heuristic?
  - Well, if we could use the concept of MDP, and get some good solution based on that…

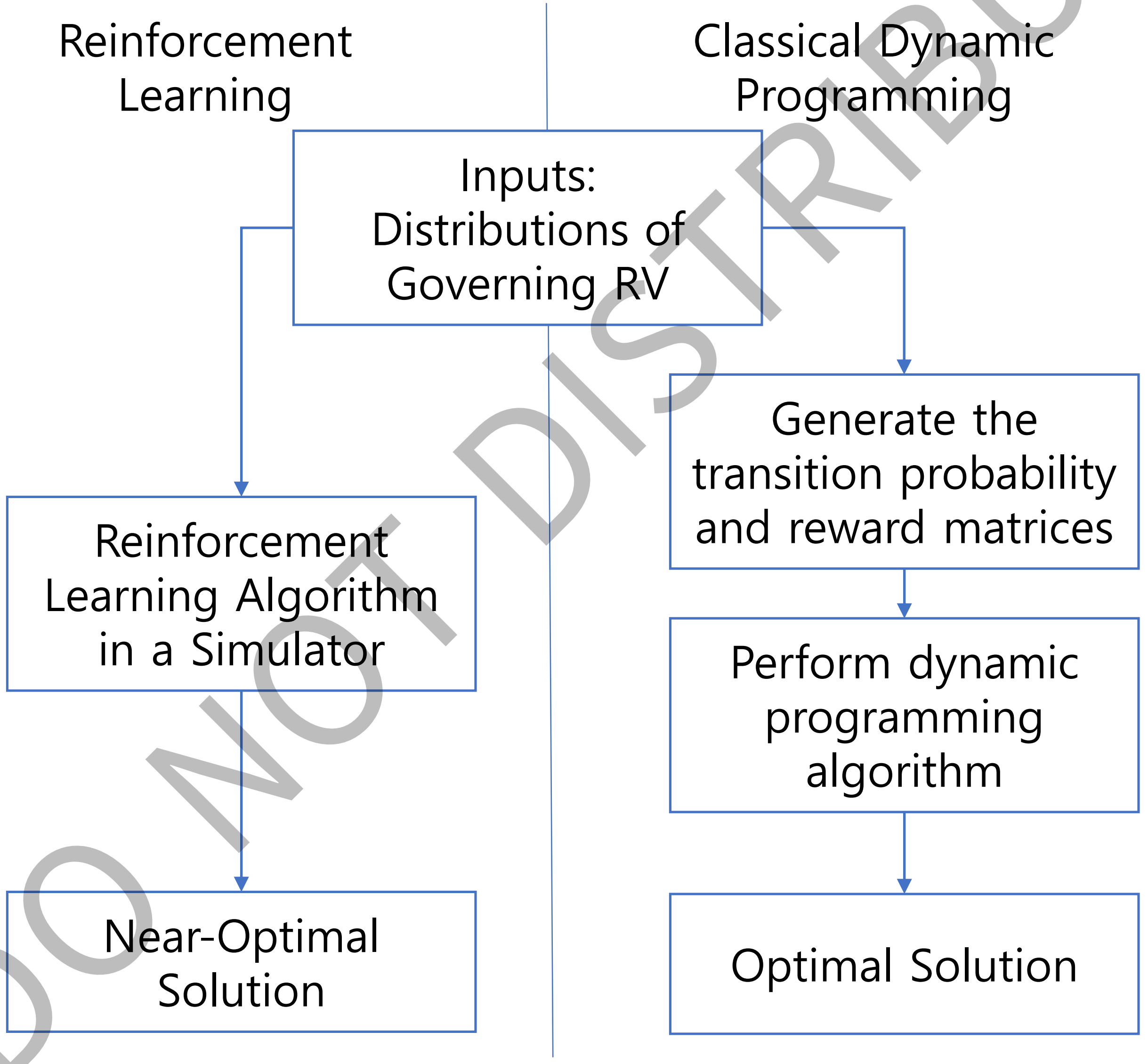
Method	Level of modeling effort	Solution quality
MDP	High	High
ADP	Medium	High
Heuristics	Low	Low

Bellman's Equation Revisit

$$u_t^*(s_t) = \max_{a \in A_{s_t}} \{ \boxed{r_t(s_t, a)} + \sum_{j \in S} \boxed{p_t(j|s_t, a)} u_{t+1}^*(j) \}$$



대안 2) Reinforcement Learning Fundamentals
from Gosavi(2015)


Reinforcement Learning
Classical Dynamic Programming
Inputs: Distributions of Governing RV
Reinforcement Learning Algorithm in a Simulator
Near-Optimal Solution
Generate the transition probability and reward matrices
Perform dynamic programming algorithm
Optimal Solution

Bellman Equation 의 다른 표현

Bellman 방정식

$$u^*(s) = \max_{a \in A_s}\{r(s,a) + \sum_{s' \in S} p(s'|s,a)u^*(s')\}$$

$$Q(s,a) = r(s,a) + \sum_{s' \in S} p(s'|s,a)u^*(s')$$

$$u^*(s) = \max_{a \in A_s} Q(s,a)$$

Q 함수

$$Q(s,a) = r(s,a) + \sum_{s' \in S} p(s'|s,a) \max_{b \in A_{s'}} Q(s',b)$$

그렇다면 이 Q함수로 이루어진 식을 어떻게 풀 것인가?

Robbins-Monro Algorithm

$$E[X] = \lim_{k\to\infty} \frac{\sum_{i=1}^{k} x_i}{k}$$

This can be generated in a simulator!!

$$X^k = \frac{\sum_{i=1}^{k} x_i}{k}$$

$$X^{k+1} = \frac{\sum_{i=1}^{k+1} x_i}{k+1} = \frac{\sum_{i=1}^{k} x_i + x_{k+1}}{k+1} = \frac{X^k k + x_{k+1}}{k+1} = \frac{X^k k + X^k - X^k + x_{k+1}}{k+1}$$

$$= \frac{X^k(k+1) - X^k + x_{k+1}}{k+1} = \frac{X^k(k+1)}{k+1} - \frac{X^k}{k+1} + \frac{x_{k+1}}{k+1}$$

$$= X^k - \frac{X^k}{k+1} + \frac{x_{k+1}}{k+1}$$

$$= \left(1 - \alpha^{k+1}\right)X^k + \alpha^{k+1} x_{k+1} \qquad if \;\; \alpha^{k+1} = \frac{1}{k+1}$$

Robbins-Monro 와 Q 함수

$$Q(s,a) = r(s,a) + \sum_{s' \in S} p(s'|s,a) \max_{b \in A_{s'}} Q(s',b)$$

$$Q(s,a) = \sum_{s' \in S} p(s'|s,a)[r(s,a,s') + \max_{b \in A_{s'}} Q(s',b)]$$

$$= \mathrm{E}[r(s,a,s') + \max_{b \in A_{s'}} Q(s',b)]$$

E(•)는 기대값 함수

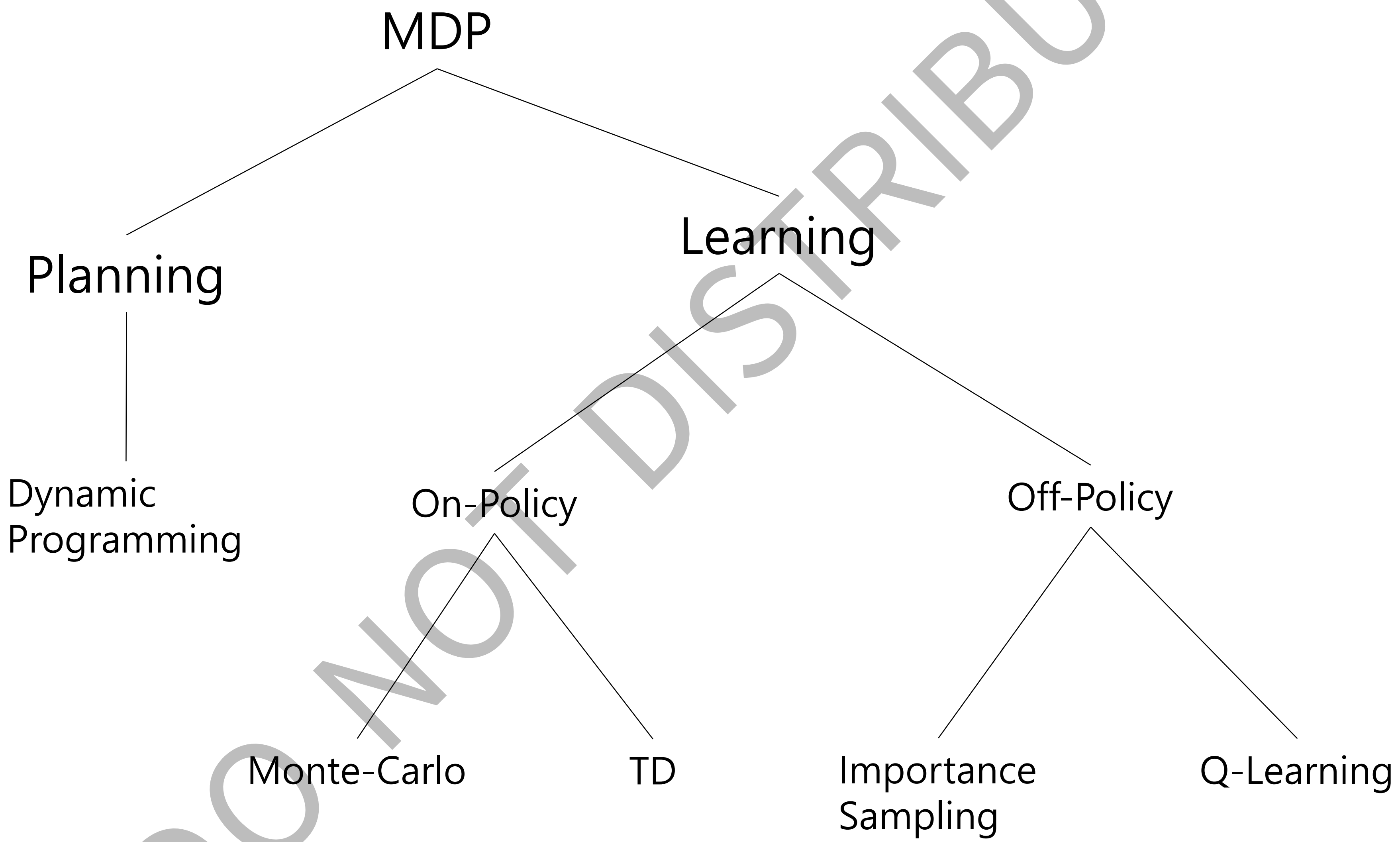
Robbins-Monro

$$\boxed{Q^{k+1}(s,a) \leftarrow (1-\alpha^{k+1})Q^k(s,a) + \alpha^{k+1} x_{k+1}}$$

$$\boxed{Q^{k+1}(s,a) \leftarrow (1-\alpha^{k+1})Q^k(s,a) + \alpha^{k+1}[r(s,a,s') + \max_{b \in A_{s'}} Q(s',b)]}$$

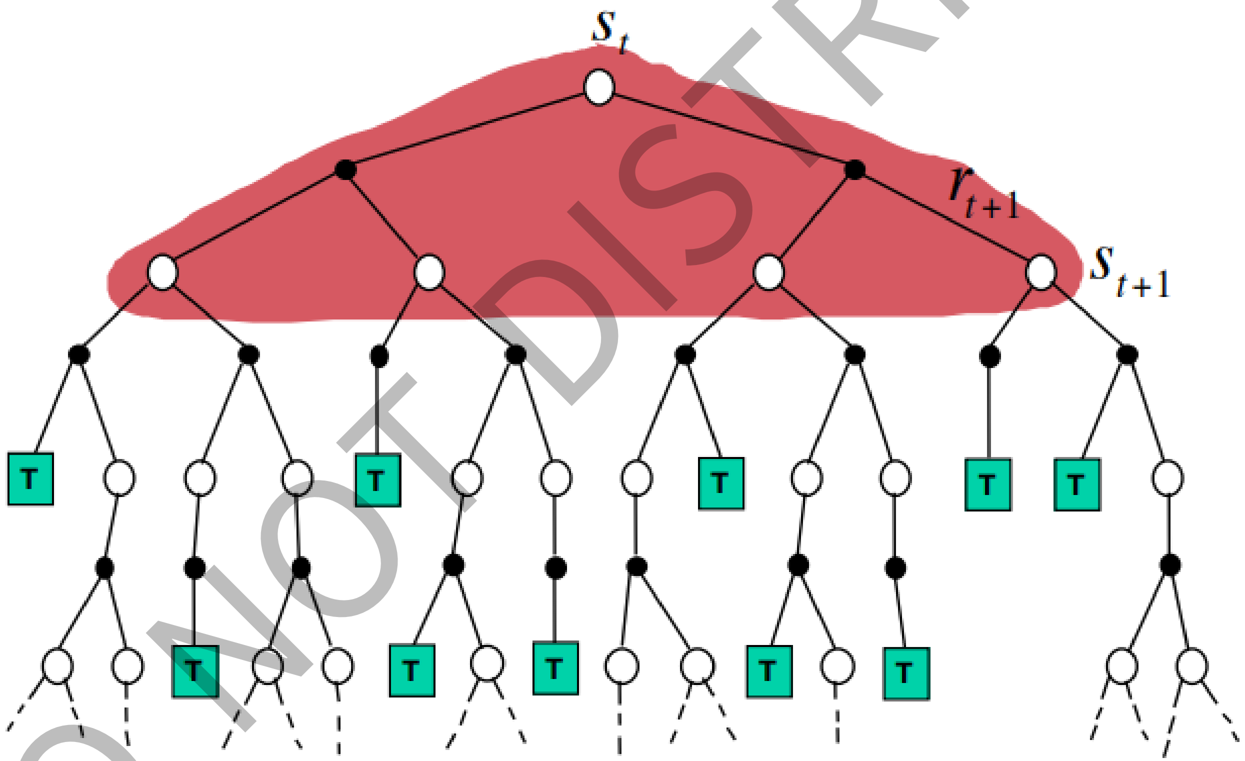
Q-Learning update rule

Reinforcement Learning 분류



Dynamic Programming 이 문제를 푸는 방식

$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



https://www.analyticsvidhya.com/blog/2019/03/reinforcement-learning-temporal-difference-learning/

핵심개념 1: Exploration vs Exploitation

- E.g. Multi-Armed Bandit Problem

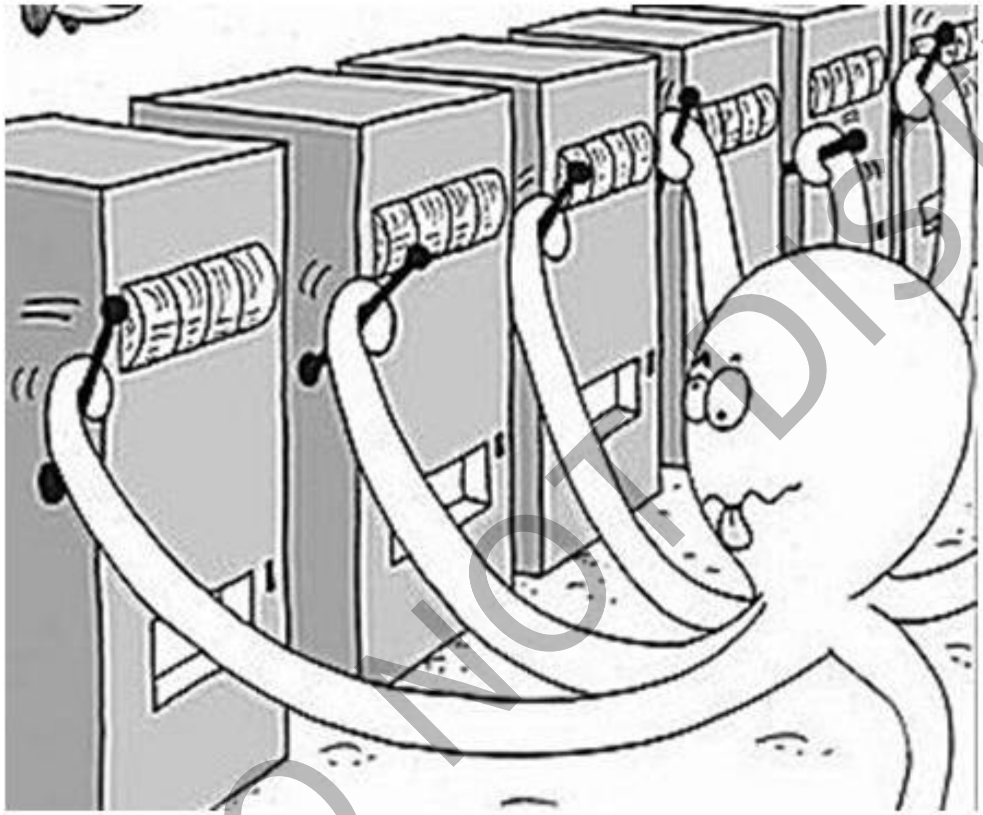
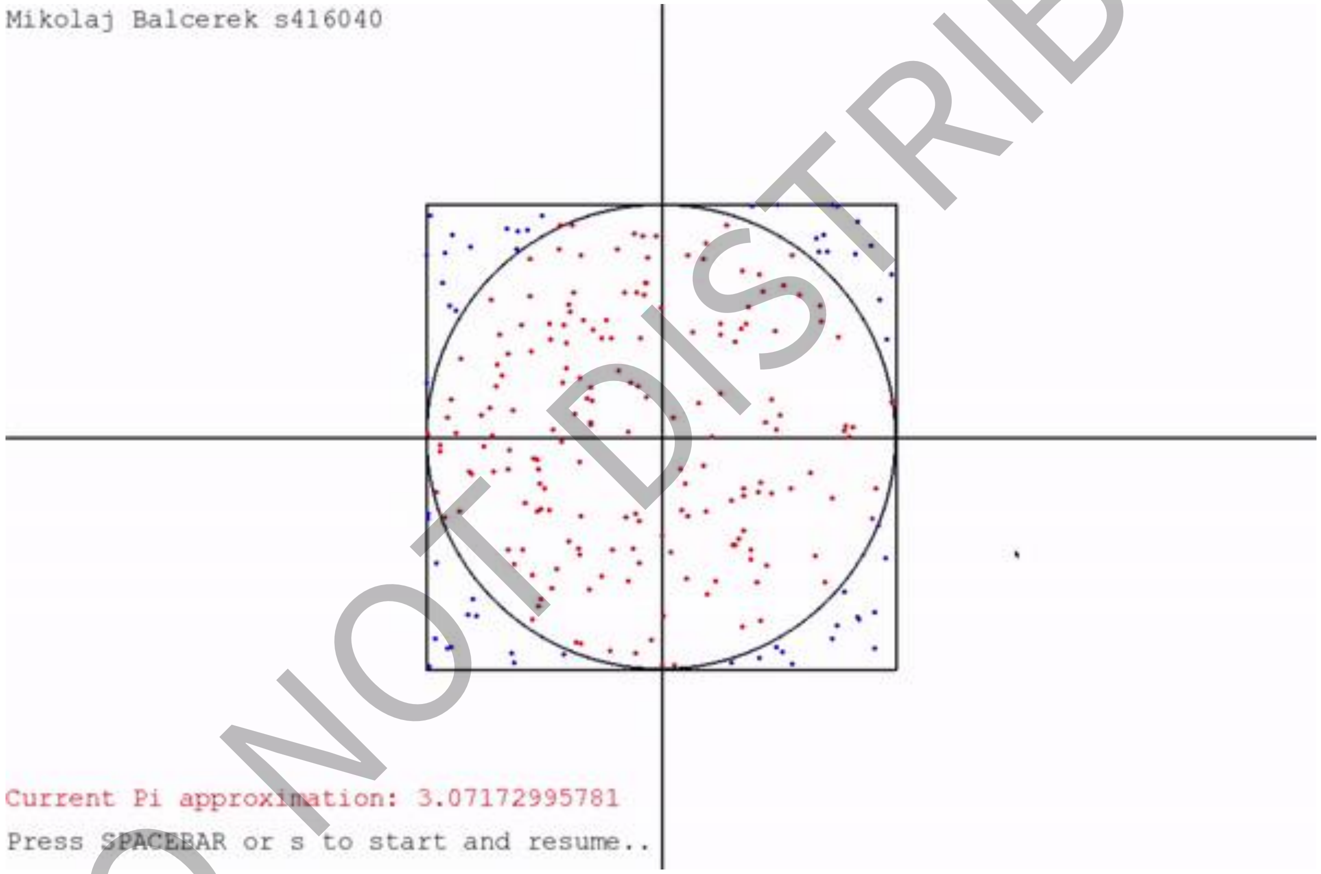


Image from research.microsoft.com

- Pure exploration
- Pure exploitation
- Mixed approach
  - e-greedy
  - Softmax
    (e.g. Boltzmann)

Monte Carlo Methods



Current Pi approximation: 3.07172995781

Press SPACEBAR or s to start and resume..

출처: https://github.com/MikolajBalcerek/Pi-Monte-Carlo-Method-Visual

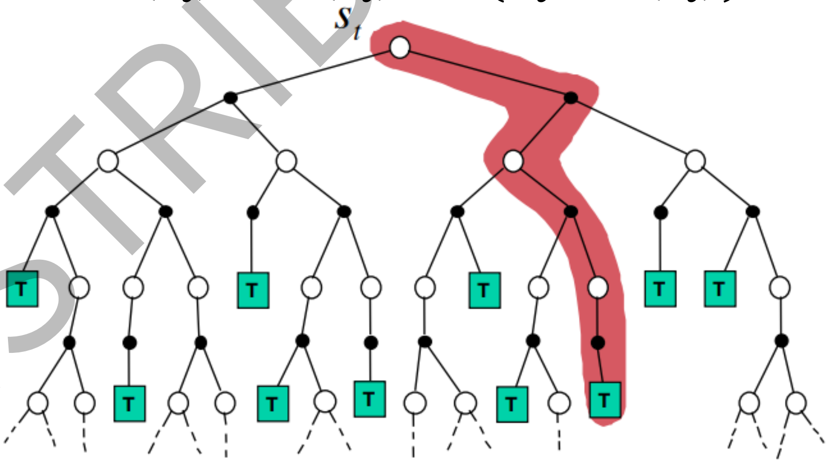
Episode

- 시뮬레이션 상에서 특정 task를 수행 시, 더 이상 진행되지 않고 종료되는 경우 해당 task의 시작부터 끝까지 나왔던 state-action 쌍의 history 집합
  - 컴퓨터 게임의 경우는 game over 상태
  - 자동차 주행의 경우에는 운전 시작부터 종료까지
  - 야구경기의 경우?
- Markov Chain에서 terminal state (종단 state)가 존재하는 경우
- 컴퓨터 공학 분야에서 다루는 대부분의 강화학습 문제유형은 episodic task 형태의 문제임
- Non-episodic task 형태도 존재

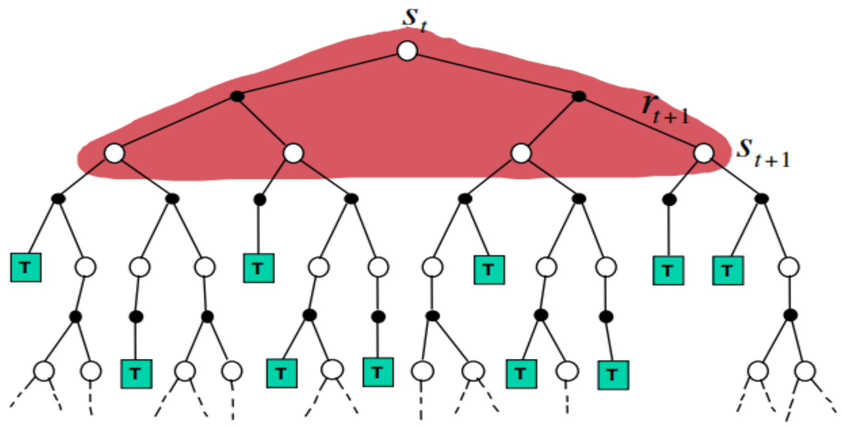
강화학습에서의 Monte Carlo 기법 활용

- MC (Monte Carlo) is *model-free* : MDP를 완전히 정의할 수 없는 경우에도 사용 가능함
- DP는 모든 state를 순차적으로 방문하지만 MC는 일련의 에피소드 (episode)만을 이용하여 근사값을 이용함
- 어떤 Policy에 대하여 한 에피소드가 끝날 때까지 각 상태를 랜덤하게 방문하면서 각 state에서 받은 총 reward를 측정함
- 에피소드를 충분히 여러 번 반복하면서 state 마다 측정된 reward를 누적한 후, 에피소드 반복이 끝나면 누적된 reward의 평균값으로 State-value를 추정함.
- 한 에피소드를 끝까지 완료해야 하기 때문에 (Episodic task) 각 상태의 value가 실시간으로 업데이트 되지 않는 단점이 있음

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$



$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



Temporal Difference (TD) – 시간 차분 학습

- TD 는 앞서 언급한 Monte Carlo 방식과 Dynamic Programming 방식을 절충
- TD methods learn directly from episodes of experience (Monte Carlo)
- TD learns from incomplete episodes by bootstrapping (Dynamic Programming)
- TD is model-free, no knowledge of MDP transitions / reward
- TD is used for
  - the prediction problem (or policy evaluation): estimating the value function $V(s_t)$
  - the control problem: finding an optimal policy

Monte Carlo vs. TD

- Predict $V(s_t)$ online from experience under policy π
- Every visit Monte-Carlo learning (G_t= actual return)

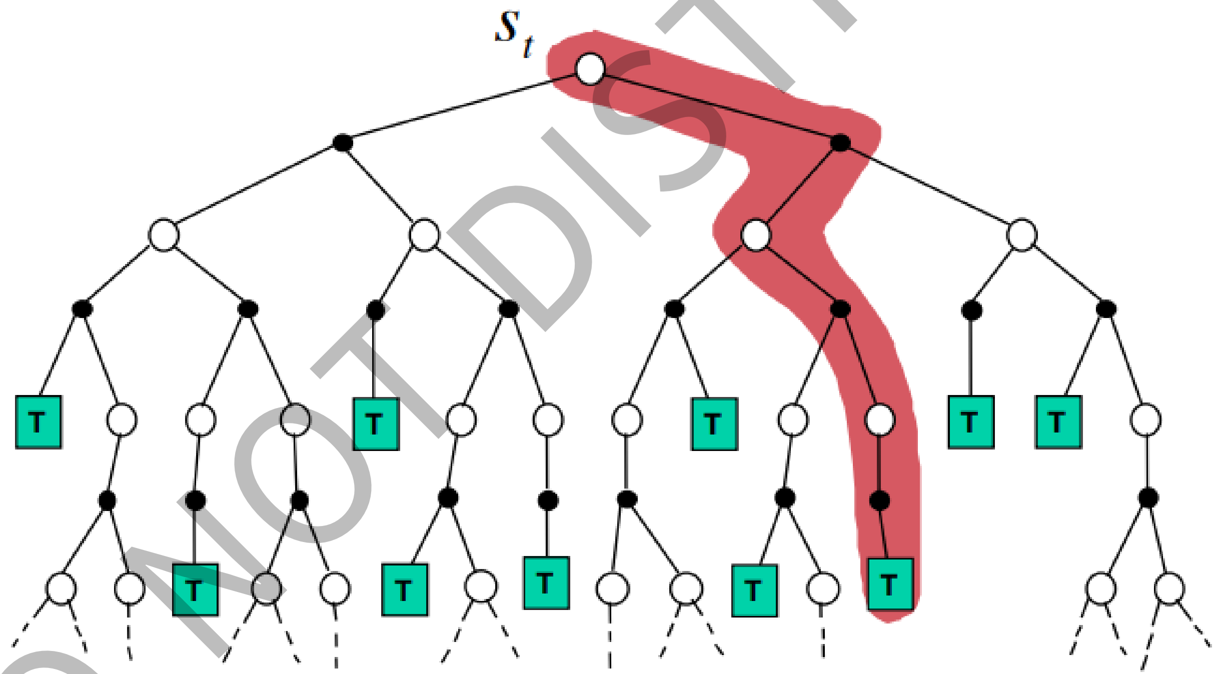
$$V(s_t) \leftarrow V(s_t) + \alpha[G_t - V(s_t)]$$

- The simplest TD learning: TD(0) ($R_{t+1} + \gamma V(s_{t+1})$= estimated return)

$$V(s_t) \leftarrow V(s_t) + \alpha[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

Monte-Carlo Method

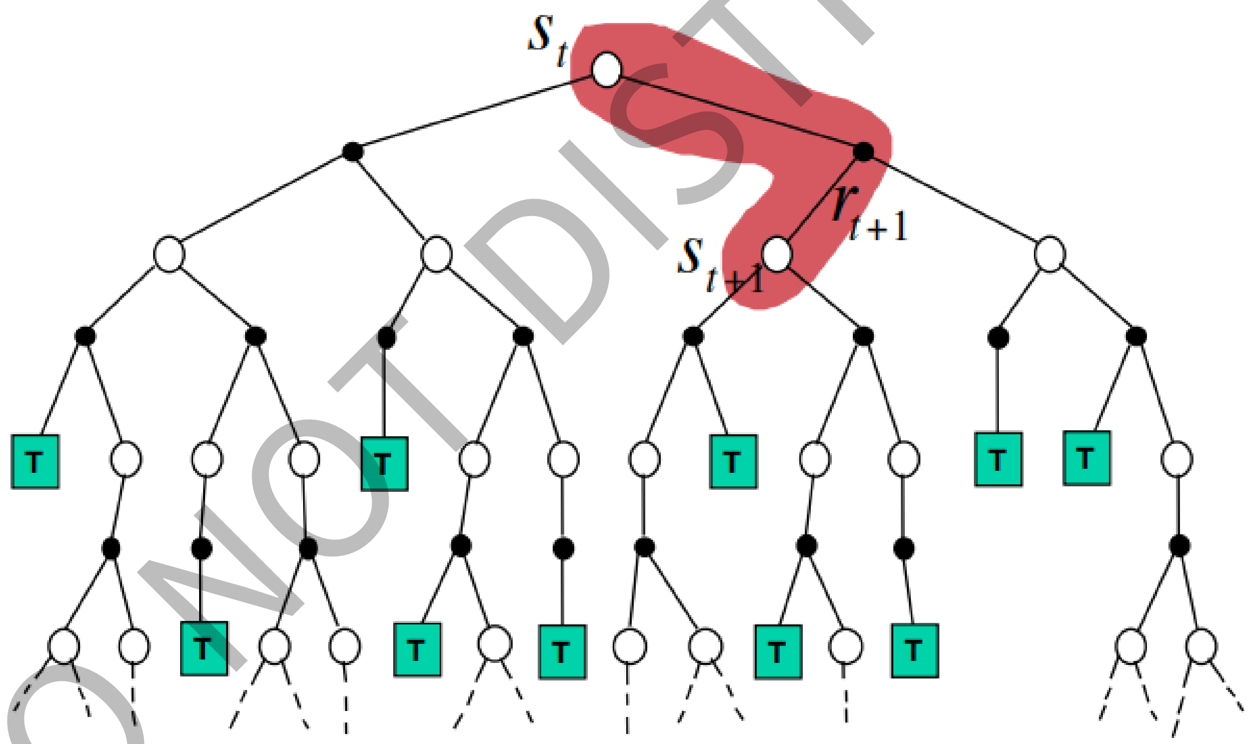
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



https://www.analyticsvidhya.com/blog/2019/03/reinforcement-learning-temporal-difference-learning/

Temporal Difference (TD)

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



https://www.analyticsvidhya.com/blog/2019/03/reinforcement-learning-temporal-difference-learning/

Reinforcement Learning 해법 정리

$$Q^{k+1}(s,a) \leftarrow \left(1-\alpha^{k+1}\right)Q^{k}(s,a) + \alpha^{k+1} x_{k+1}$$
$$= Q^{k}(s,a) + \alpha^{k+1}\left(x_{k+1} - Q^{k}(s,a)\right)$$

New Estimate ← Old Estimate + StepSize [Target – OldEstimate]

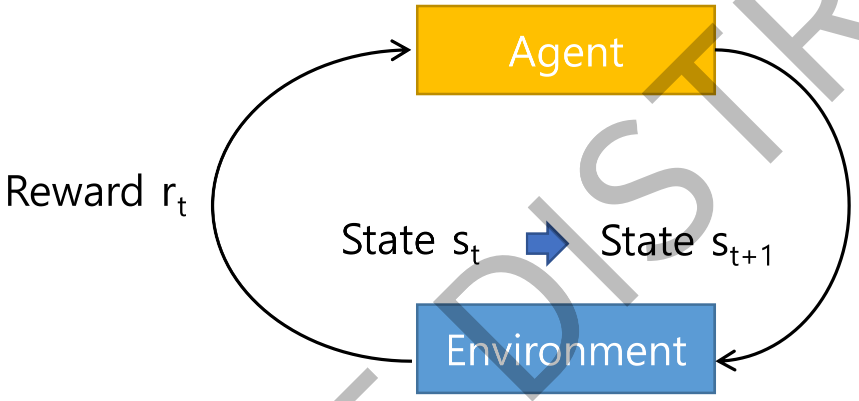
Monte-Carlo $V(s_t) \leftarrow V(s_t) + \alpha[G_t - V(s_t)]$

SARSA $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$

Q-Learning $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_{a} Q(S_{t+1}, a) - Q(S_t, A_t)]$

Behavioral Policy and Target Policy

agent는 어떤 것을 보며 학습
하는 걸까?
→ Target Policy (목표 정책)



액션은 어떤 기준에서 선택되는
가?
→ Behavioral Policy (행동 정책)

SARSA $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$

Q-Learning $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$

핵심개념 2: On-Policy vs Off-Policy

On-policy control	Off-policy control
• Lear about target policy π from experience sampled from π • SARSA (on-policy TD control) • Use a sample $\{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}\}$ • Bellman expectation equation → Policy iteration → SARSA	• Lear about target policy π from experience sampled from behavior policy μ • Importance Sampling, Q-Learning (Off-policy TD control) • Use a sample $\{S_t, A_t, R_{t+1}, S_{t+1}\}$ • Bellman optimality equation → Value iteration → Q-Learning

1. 시간 투자 (10000시간의 법칙?)

2. 좋은 레퍼런스 (코칭, 참고자료 등.)

핵심개념 3: Off-line vs On-line

- Implementation perspective
  - Off-line implementation
    - Simulator
    - Plan ahead and make a policy before the “actual” implementation
  - On-line implementation
    - Run the model in real time alongside an actual physical process
    - Plan adaptively as time goes by

- Update perspective (internal issue of the algorithm)
  - On-line update: use results from the very last stage for updating Q value (i.e. update Q immediately after a trial)
  - Off-line update: use results from finite # of past stages for updating Q value (i.e. update Q after a finite number of state transitions)

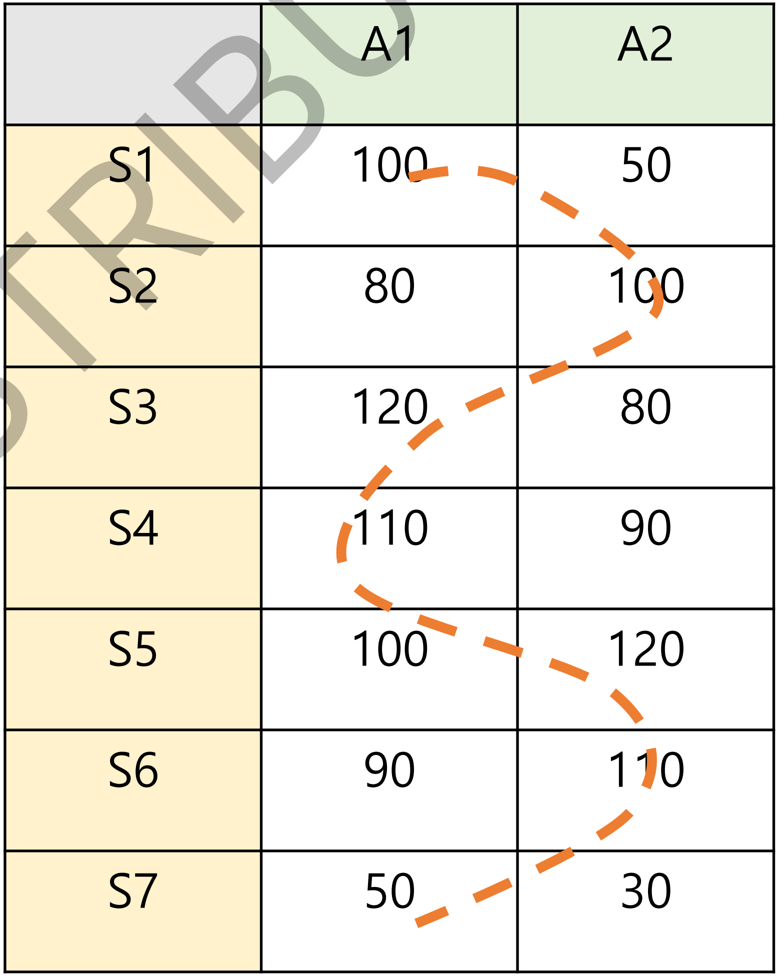
Some notable techniques in RL

- Function Approximation
  - # of state-action pairs
  - cf. look-up table approach (a rule of thumb?)
  - State Aggregation
  - Neural Network
    - Input: state-action pair, Output: Q-value
    - Input: state Output: Q-value of actions
    - …

Q(S,A) 값 저장

	A1	A2
S1	100	50
S2	80	100
S3	120	80
S4	110	90
S5	100	120
S6	90	110
S7	50	30

Tabular Form



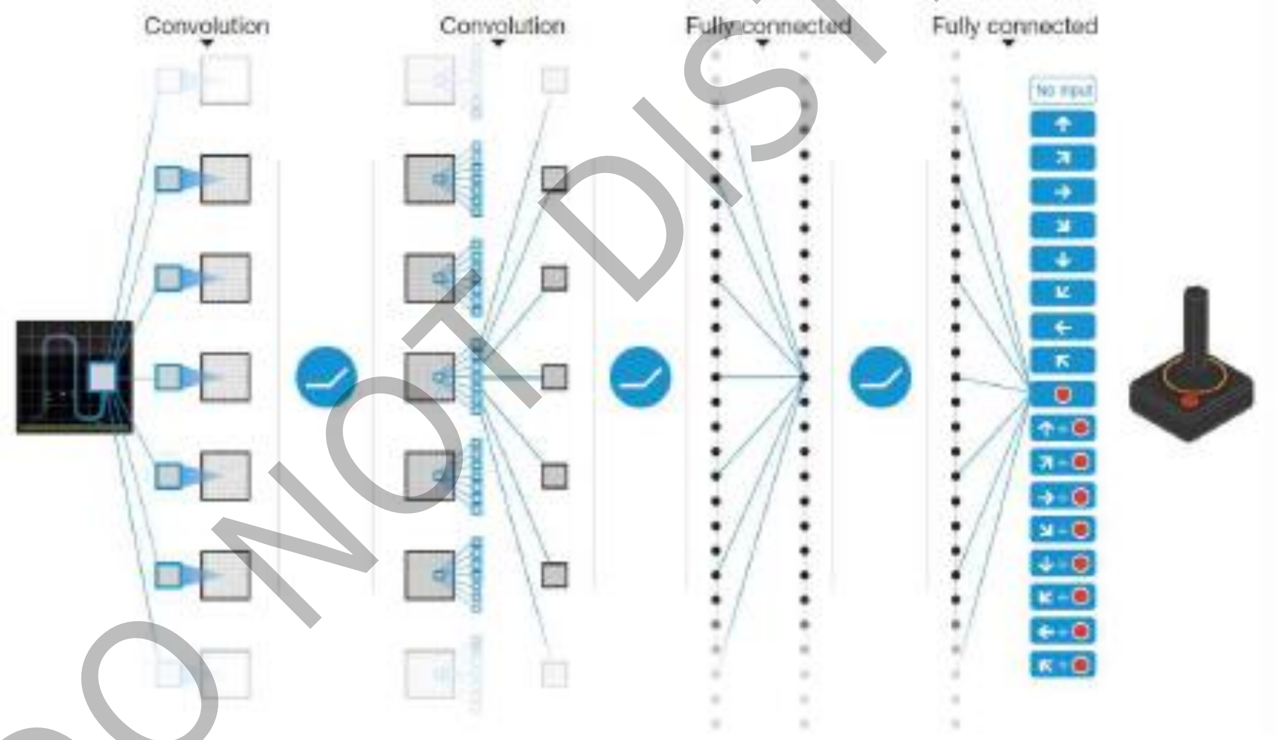
	A1	A2
S1	100	50
S2	80	100
S3	120	80
S4	110	90
S5	100	120
S6	90	110
S7	50	30

Function Approx.

3. 강화학습 최신 경향

DQN (Deep Q Network)

- 합성곱 계층(CNN)을 통한 고 수준의 이미지 입력
- Experience Replay를 통한 다양한 경험 축척
- 별도로 구분된 Target Network를 사용한 안정된 학습



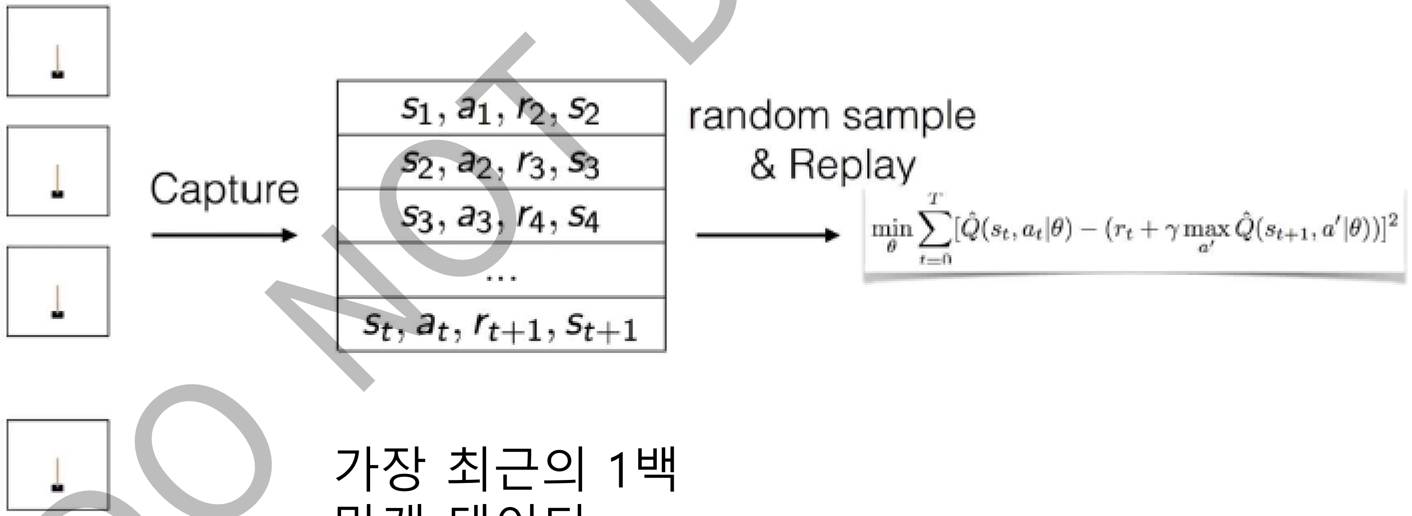
Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529.

DQN-1) 합성곱 신경망 사용

- Convolutional Neural Network (CNN; 합성곱신경망)을 이용한 높은 수준의 이미지 입력정보 활용
  - High-dimensional sensory inputs으로 게임에서 표출하는 화면의 이미지 픽셀정보를 입력 State로 사용
  - raw Input size는 210*160이나 84*84 로 리사이즈.
  - Input에 들어오는 이미지는 가장 최근 4장을 쌓아서 제공. 즉
  - 4*84*84 가 input.
  - CNN(3개층의 Convolutional layer와 2개층의 Fully connected Layer로 구성)을 통해 Q함수를 Approximation

DQN-2) Experience Replay

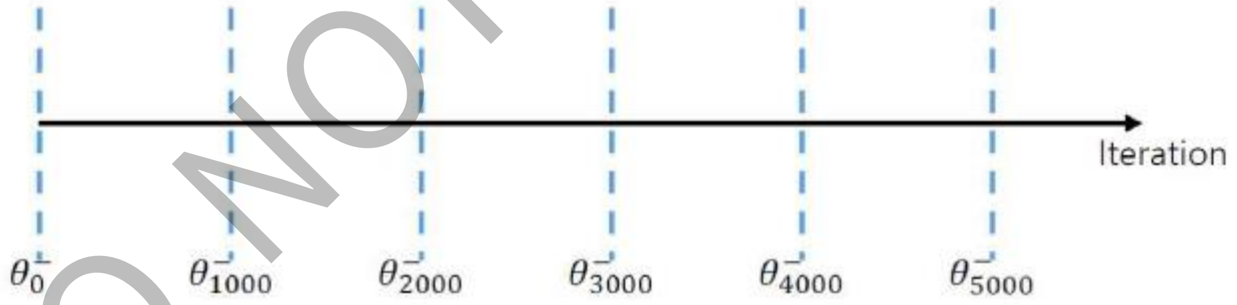
- Correlations between samples 문제
  - observation의 시퀀스상의 correlation때문에 발생하는 문제. 샘플간 차이가 크지 않아서 발생.
  - 이를 효율적으로 해결하기 위해 Replay Buffer라는 것을 두고 여기에 Transition 정보값을 차곡차곡 쌓아놓고, 이 중 랜덤하게 샘플링하여 학습에 활용

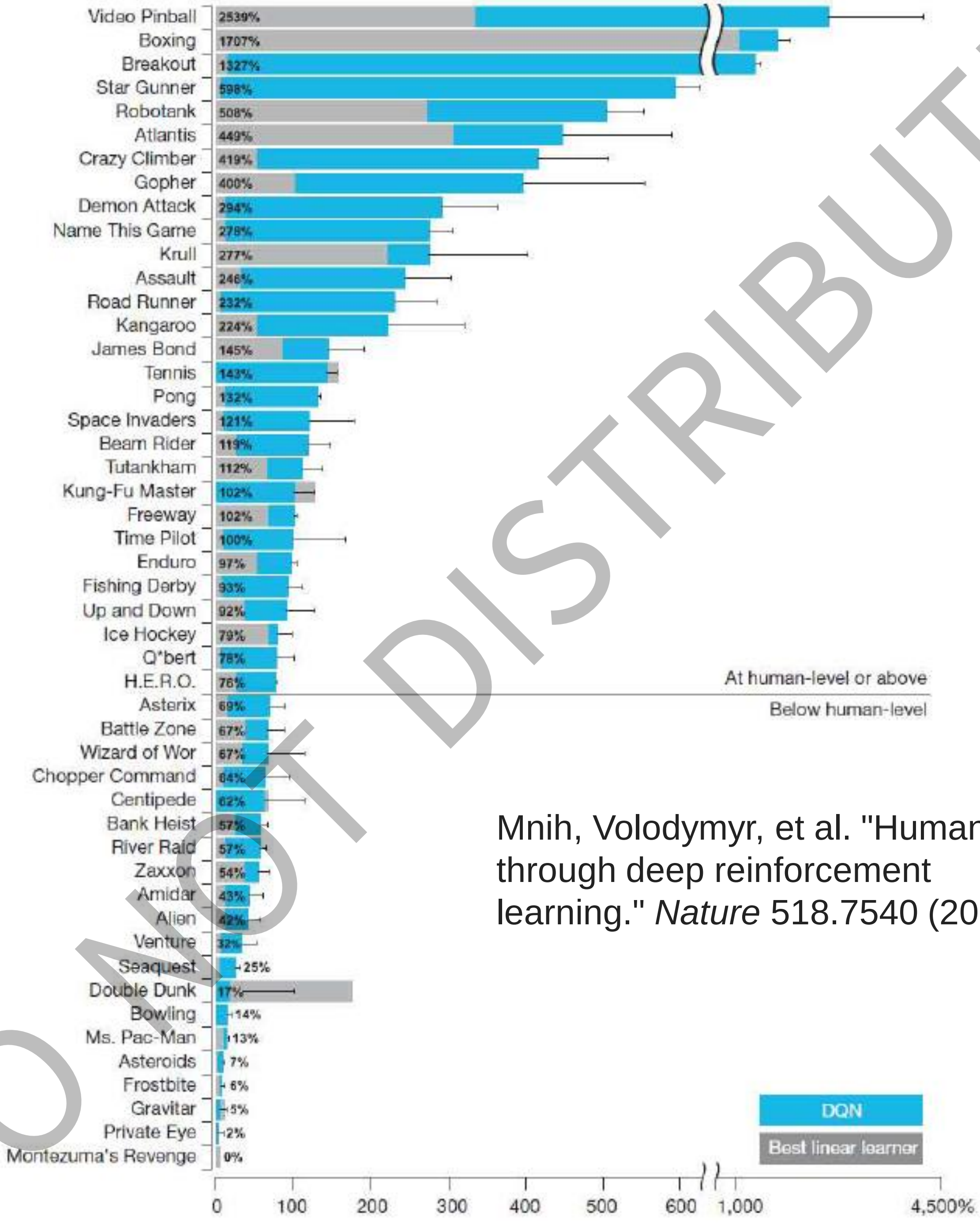

Capture
s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\dots$
$s_t, a_t, r_{t+1}, s_{t+1}$
random sample & Replay
$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2$

가장 최근의 1백 만개 데이터

DQN-3) 별도의 Target Network 신경망

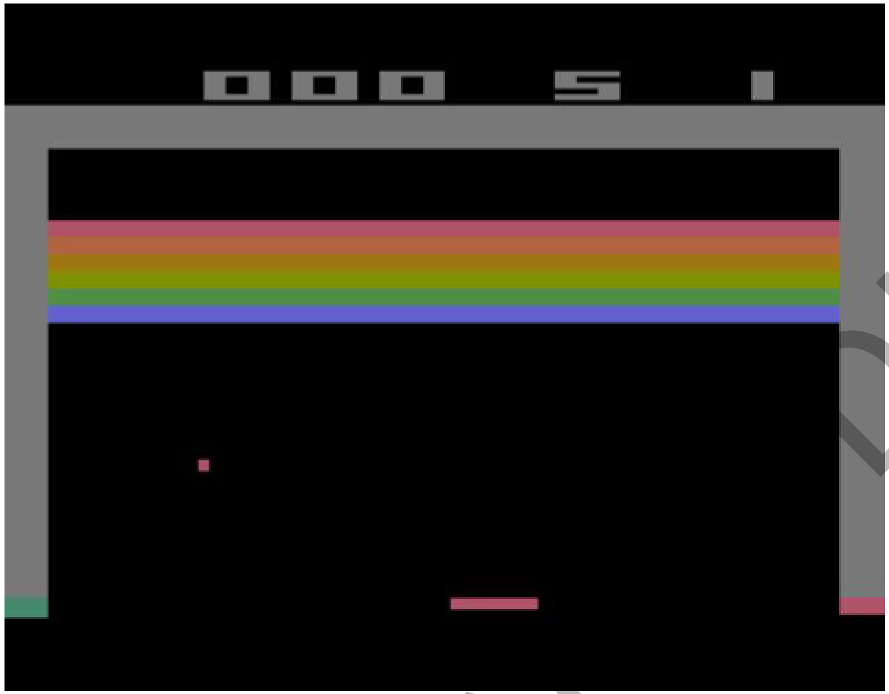
- 매번 학습때마다 신경망의 파라미터 값이 계속 바뀌는 문제로 인해 Q값이 불안정해질 수 있음 (Non-stationary target 이슈)
- 이러한 이슈를 줄이기 위해, Target 신경망을 별도로 두고, Target 신경망의 업데이트 간격을 길게 고정하여 잡아 둠.

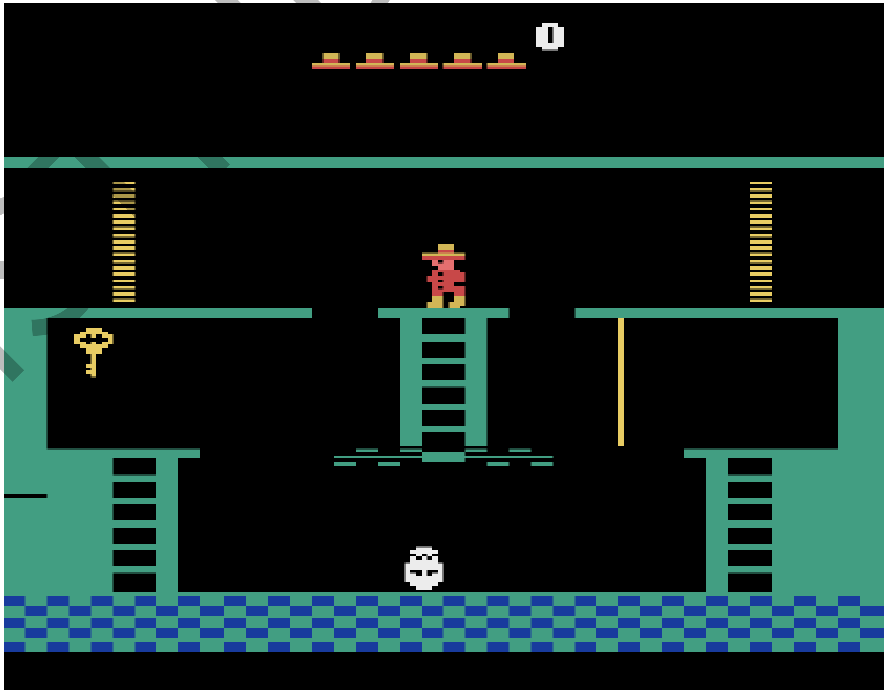




Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529.

Breakout vs Montezuma's Revenge

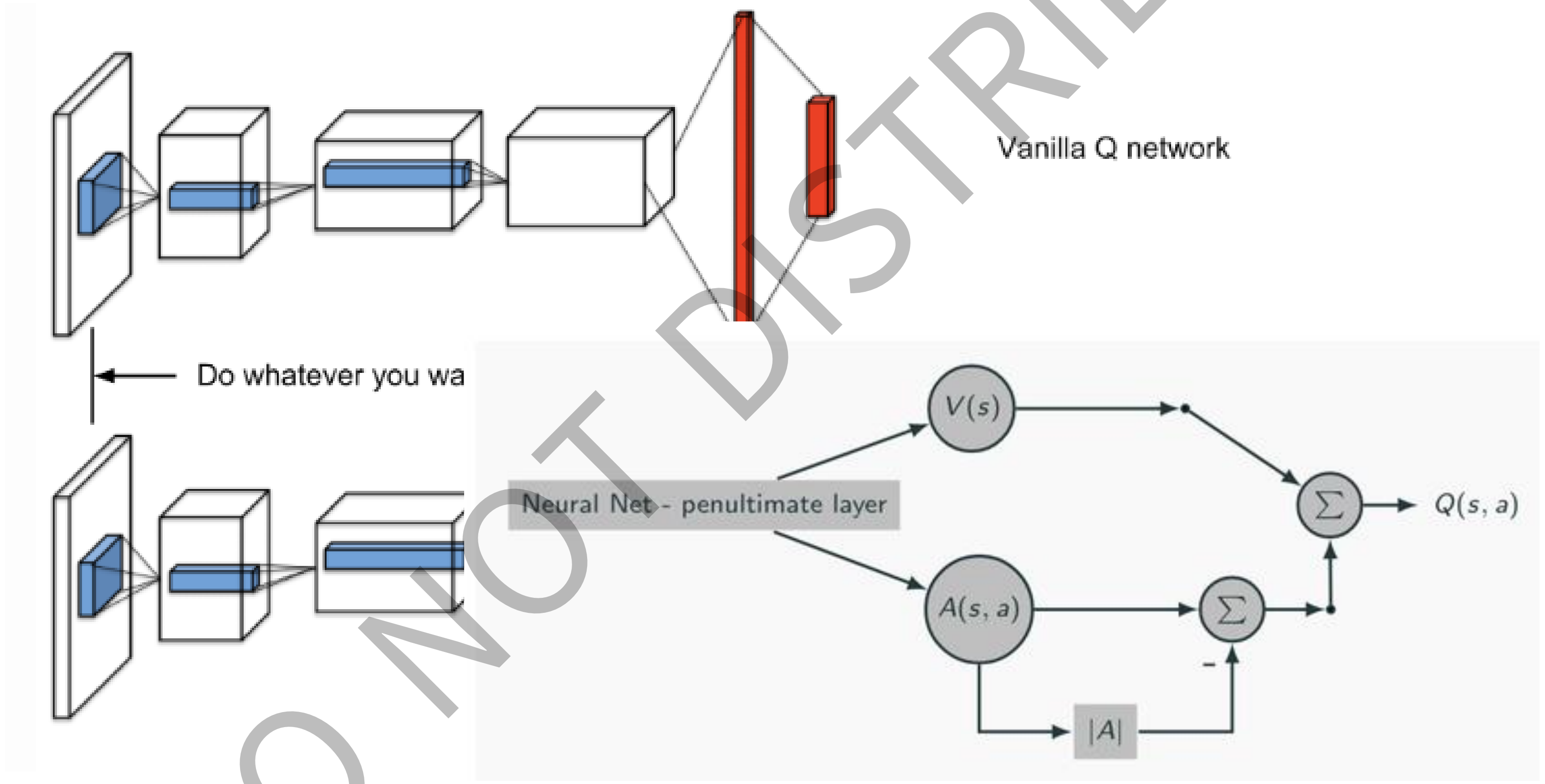




Hierarchical RL (계층적 강화학습)

Dueling DQN

- Q(s,a) =V(s) + A(a)



Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581*(2015).

Current Trends in RL

- Double DQN
- Policy Gradient
- Actor-Critic
- Asynchronous Advantage Actor-Critic
- Trust Regions (TRPO, PPO, ACKTR)

4. 예제

참조한
블로그정보는
코드에서 주석으로
표기

Blackjack

- Object: 21을 초과하지 않고 플레이어의 카드 합이 딜러의 카드 합보다 크면 승리
- State
  - 플레이어의 현재 카드 합 (12-21)
  - 딜러가 오픈한 카드 (ace-10)
  - 플레이어가 1혹은 11로 플레이 가능한 Ace를 가지고 있는지 여부
- Reward
  - 승리 +1 , 무승부 0, 패배 -1
- Actions
  - stick (카드 그만 받기;stand 혹은 stay): 0
  - hit (카드 한 장 더 받기): 1

Blackjack 실습 준비사항

실습파일

- 다음 링크에서 다운로드

https://github.com/charleskang/cde/blob/master/Bj_cde.ipynb

혹은

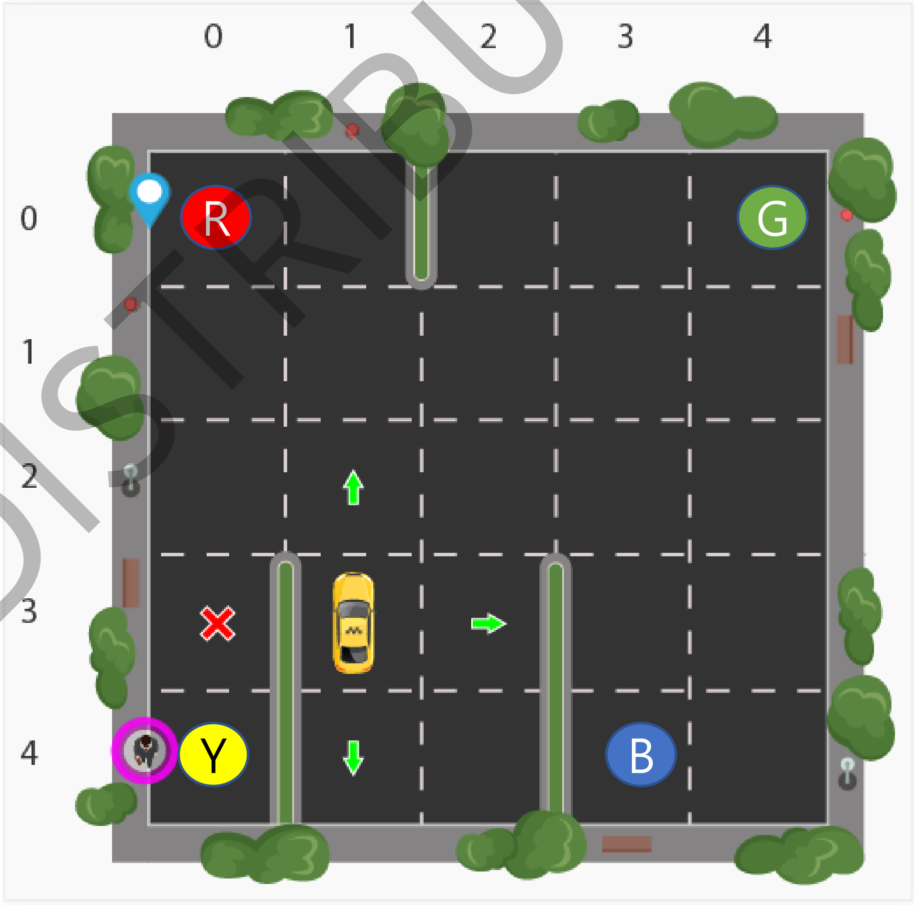
http://tiny.cc/6b2dbz

참고링크

- 블랙잭 시뮬레이터(environment) 원본파일: https://github.com/openai/gym/blob/master/gym/envs/toy_text/blackjack.py

Self-Driving Cab

- Object: 승객을 정해진 목적지로 이동
- State (5*5*5*4)
  - 5 X 5 가능한 위치
  - pickup 혹은 dropoff가 가능한 위치: R,G,Y,B
  - 승객의 상태: 택시 타고 이동중인지(1) + R,G,B,Y(4)=5
- Action (6)
  - south, north, east, west, pickup, dropoff
- Reward
  - 성공적으로 승객을 정해진 장소에 drop off 했을때 높은 보상이 주어짐
  - 승객을 잘못된 곳에 drop off 했을때는 패널티가 주어짐
  - 매 이동시간마다 “약간 " 의 패널티가 주어짐



Self Driving Cab 실습 준비사항

실습파일

- 다음 링크에서 다운로드

https://github.com/charleskang/cde/blob/master/cab_cde.ipynb

혹은

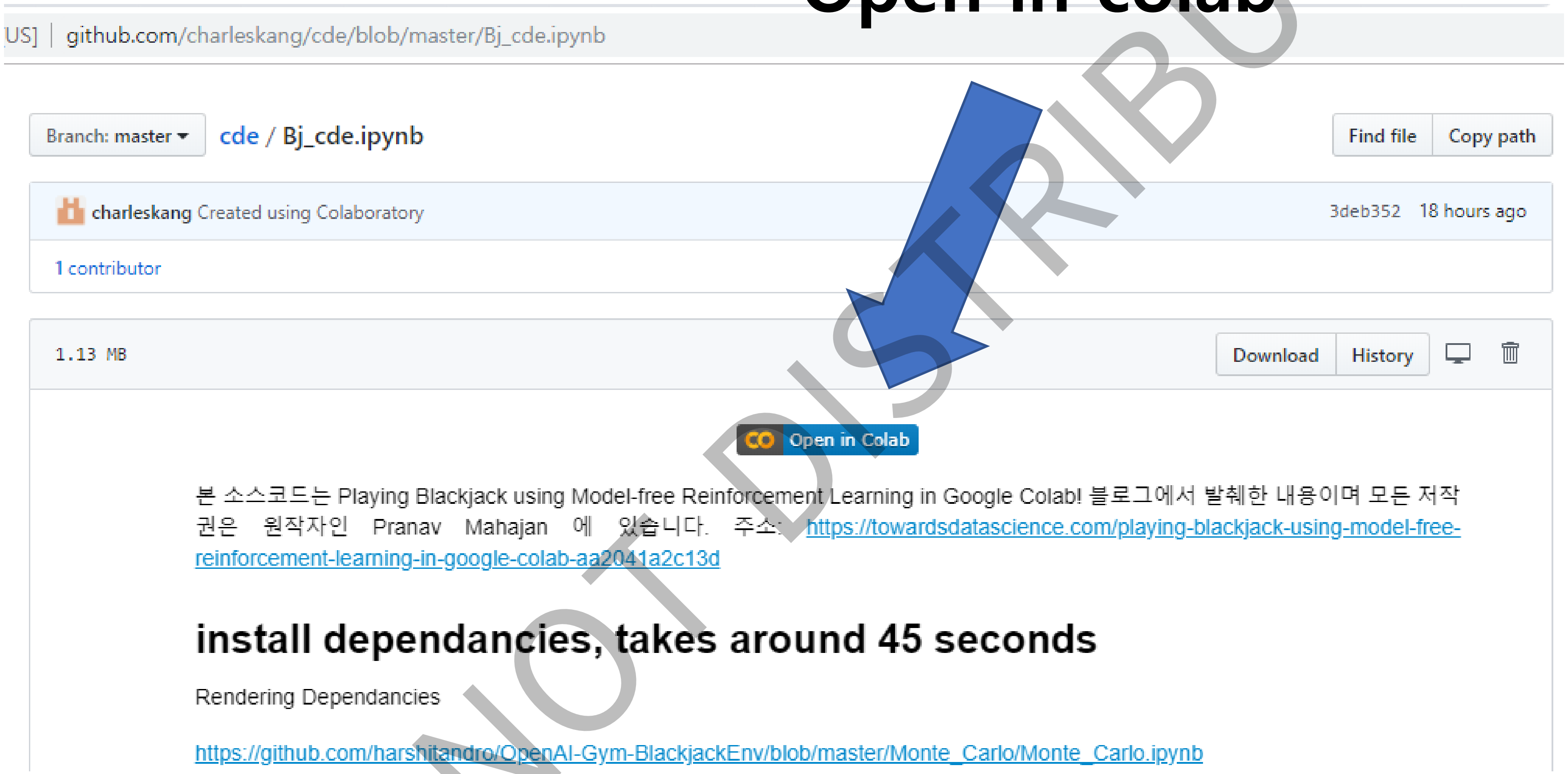
http://tiny.cc/5ow9az

참고링크

- 택시 시뮬레이터 원본 파일
  https://github.com/openai/gym/blob/master/gym/envs/toy_text/taxi.py

Appendix

Open in colab

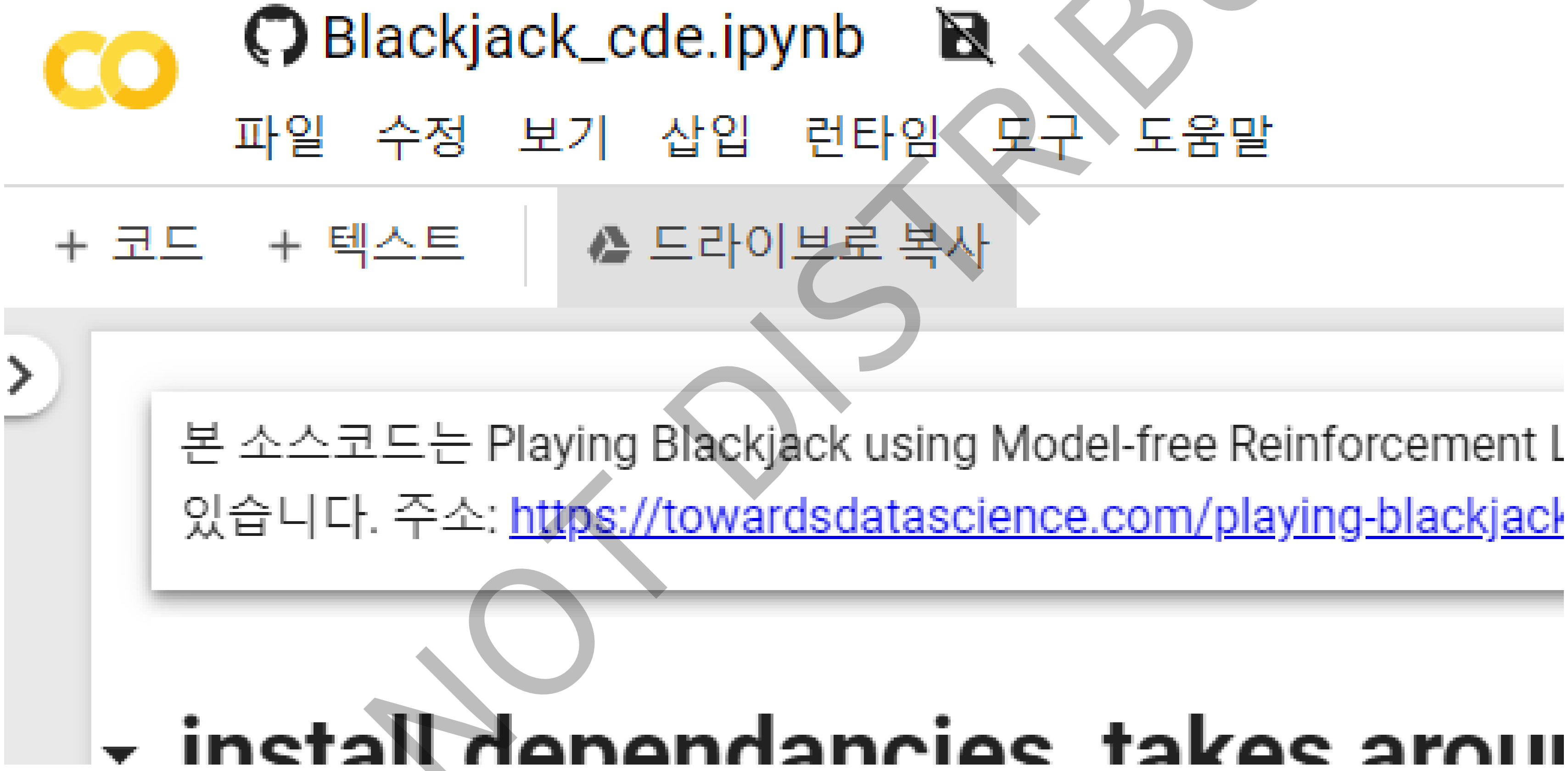


본 소스코드는 Playing Blackjack using Model-free Reinforcement Learning in Google Colab! 블로그에서 발췌한 내용이며 모든 저작권은 원작자인 Pranav Mahajan 에 있습니다. 주소: https://towardsdatascience.com/playing-blackjack-using-model-free-reinforcement-learning-in-google-colab-aa2041a2c13d

install dependancies, takes around 45 seconds

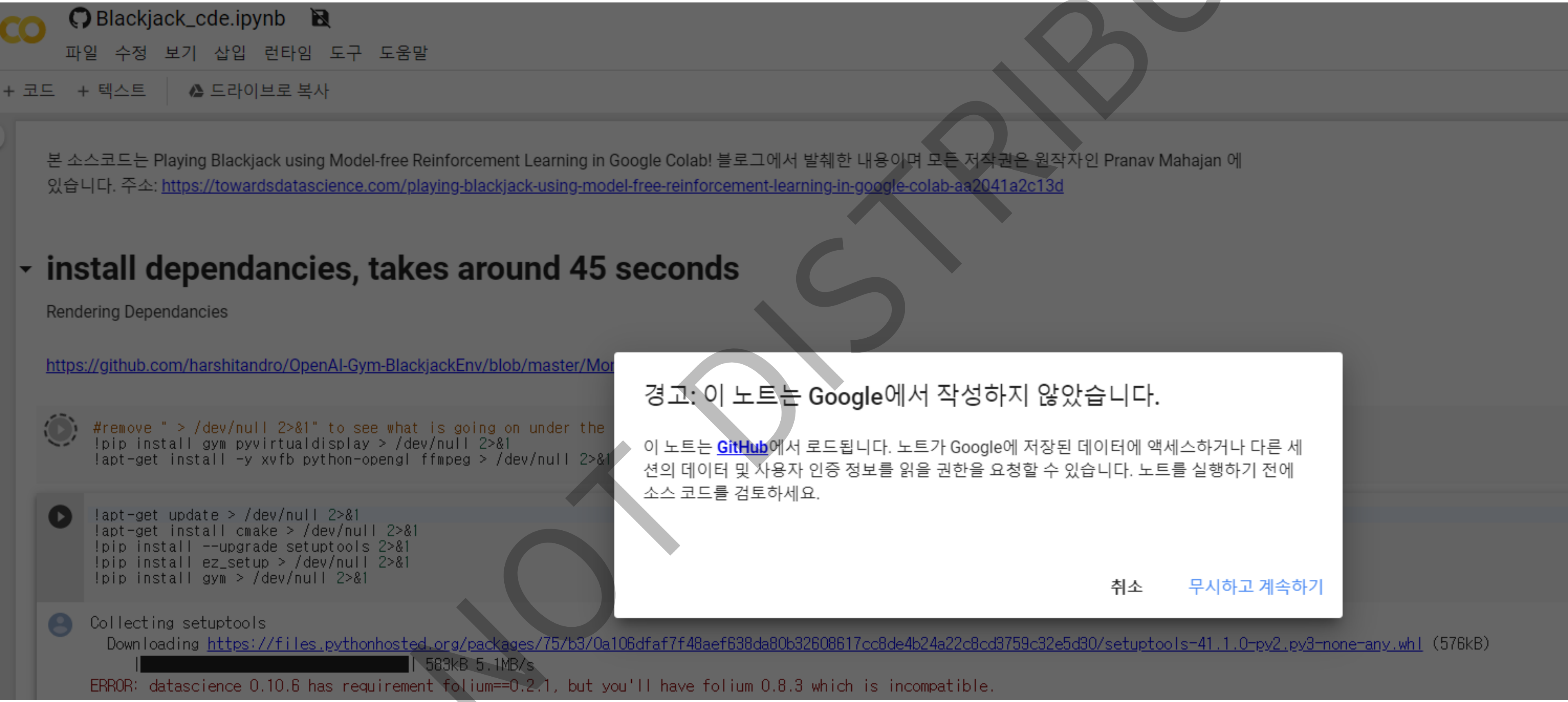
Rendering Dependancies

https://github.com/harshitandro/OpenAI-Gym-BlackjackEnv/blob/master/Monte_Carlo/Monte_Carlo.ipynb



본 소스코드는 Playing Blackjack using Model-free Reinforcement L
있습니다. 주소: https://towardsdatascience.com/playing-blackjack

install dependancies, takes arou


Blackjack_cde.ipynb
파일 수정 보기 삽입 런타임 도구 도움말
+ 코드 + 텍스트 드라이브로 복사

본 소스코드는 Playing Blackjack using Model-free Reinforcement Learning in Google Colab! 블로그에서 발췌한 내용이며 모든 저작권은 원작자인 Pranav Mahajan 에 있습니다. 주소: https://towardsdatascience.com/playing-blackjack-using-model-free-reinforcement-learning-in-google-colab-aa2041a2c13d

install dependancies, takes around 45 seconds

Rendering Dependancies

https://github.com/harshitandro/OpenAI-Gym-BlackjackEnv/blob/master/Mo

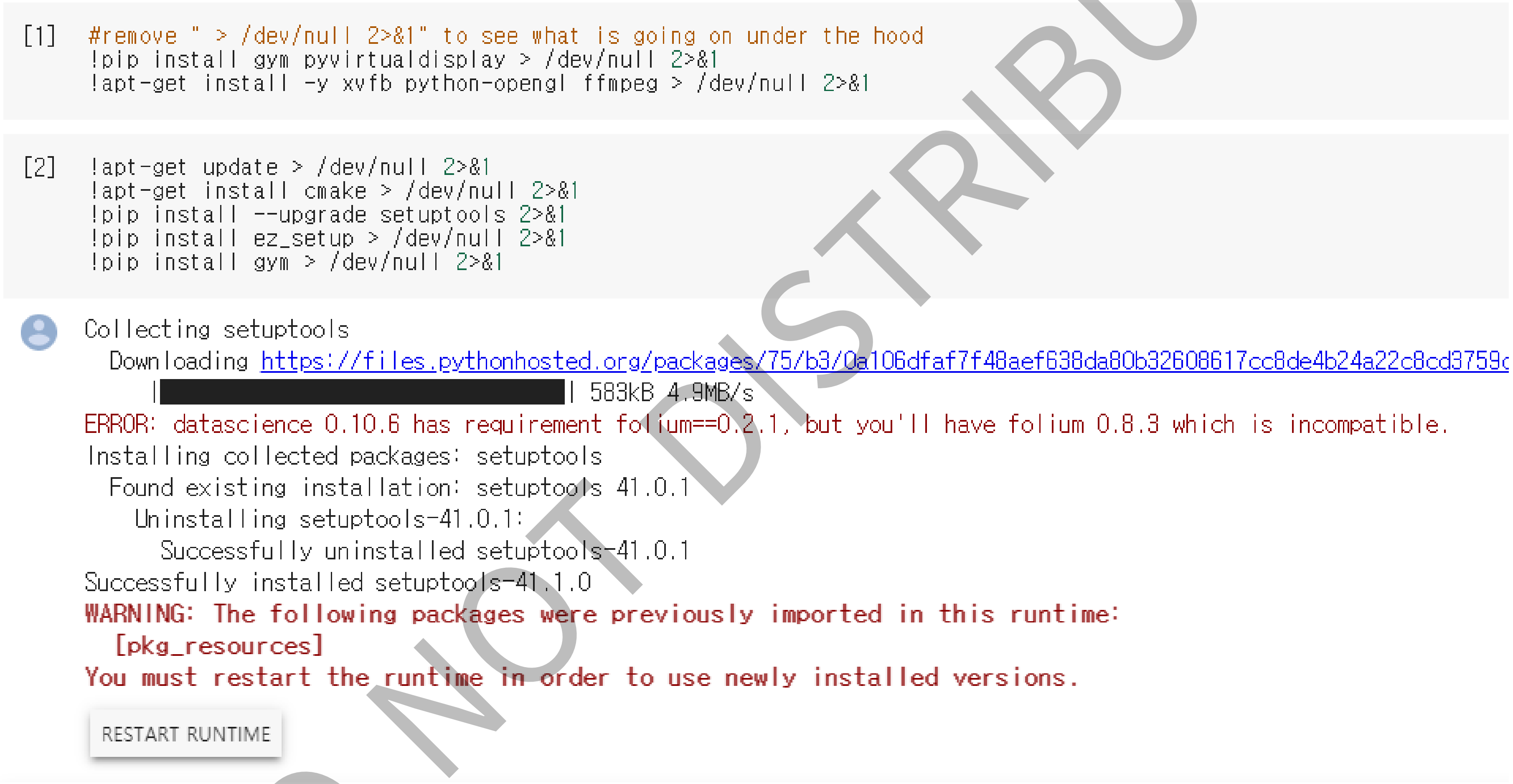
```
#remove " > /dev/null 2>&1" to see what is going on under the
!pip install gym pyvirtualdisplay > /dev/null 2>&1
!apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1
```

```
!apt-get update > /dev/null 2>&1
!apt-get install cmake > /dev/null 2>&1
!pip install --upgrade setuptools 2>&1
!pip install ez_setup > /dev/null 2>&1
!pip install gym > /dev/null 2>&1
```

```
Collecting setuptools
  Downloading https://files.pythonhosted.org/packages/75/b3/0a106dfaf7f48aef638da80b32608617cc8de4b24a22c8cd3759c32e5d30/setuptools-41.1.0-py2.py3-none-any.whl (576kB)
     |████████████████████████████████| 583kB 5.1MB/s
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible.
```

경고: 이 노트는 Google에서 작성하지 않았습니다.

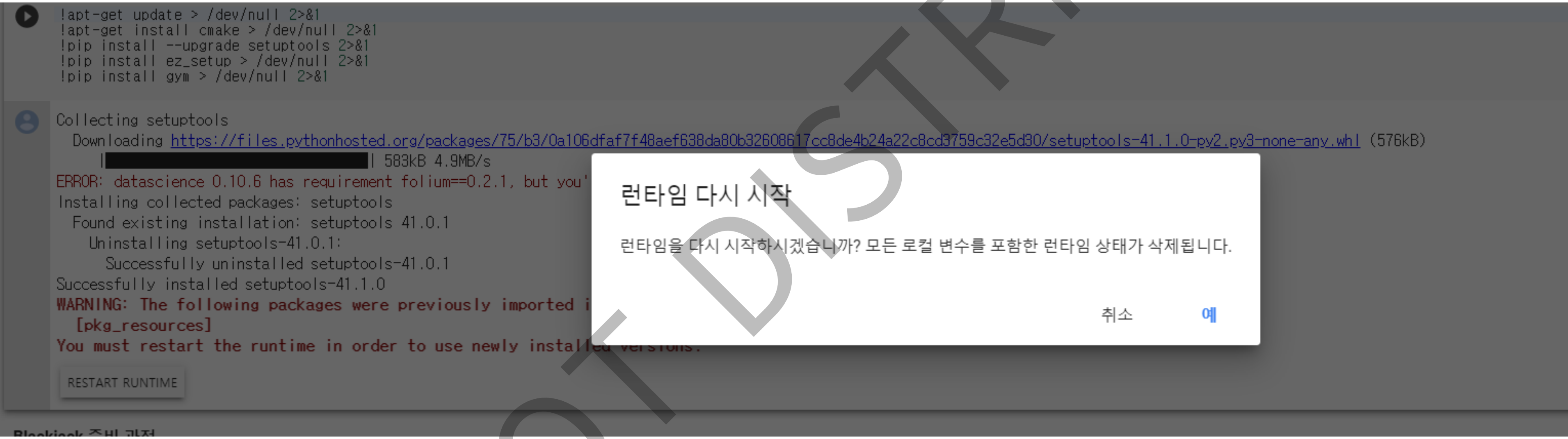
이 노트는 GitHub에서 로드됩니다. 노트가 Google에 저장된 데이터에 액세스하거나 다른 세션의 데이터 및 사용자 인증 정보를 읽을 권한을 요청할 수 있습니다. 노트를 실행하기 전에 소스 코드를 검토하세요.



```
[1]  #remove " > /dev/null 2>&1" to see what is going on under the hood
     !pip install gym pyvirtualdisplay > /dev/null 2>&1
     !apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1
```

```
[2]  !apt-get update > /dev/null 2>&1
     !apt-get install cmake > /dev/null 2>&1
     !pip install --upgrade setuptools 2>&1
     !pip install ez_setup > /dev/null 2>&1
     !pip install gym > /dev/null 2>&1
```

```
Collecting setuptools
  Downloading https://files.pythonhosted.org/packages/75/b3/0a106dfaf7f48aef638da80b32608617cc8de4b24a22c8cd3759c
     |████████████████████████████████| 583kB 4.9MB/s
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible.
Installing collected packages: setuptools
  Found existing installation: setuptools 41.0.1
    Uninstalling setuptools-41.0.1:
      Successfully uninstalled setuptools-41.0.1
Successfully installed setuptools-41.1.0
WARNING: The following packages were previously imported in this runtime:
  [pkg_resources]
You must restart the runtime in order to use newly installed versions.
```



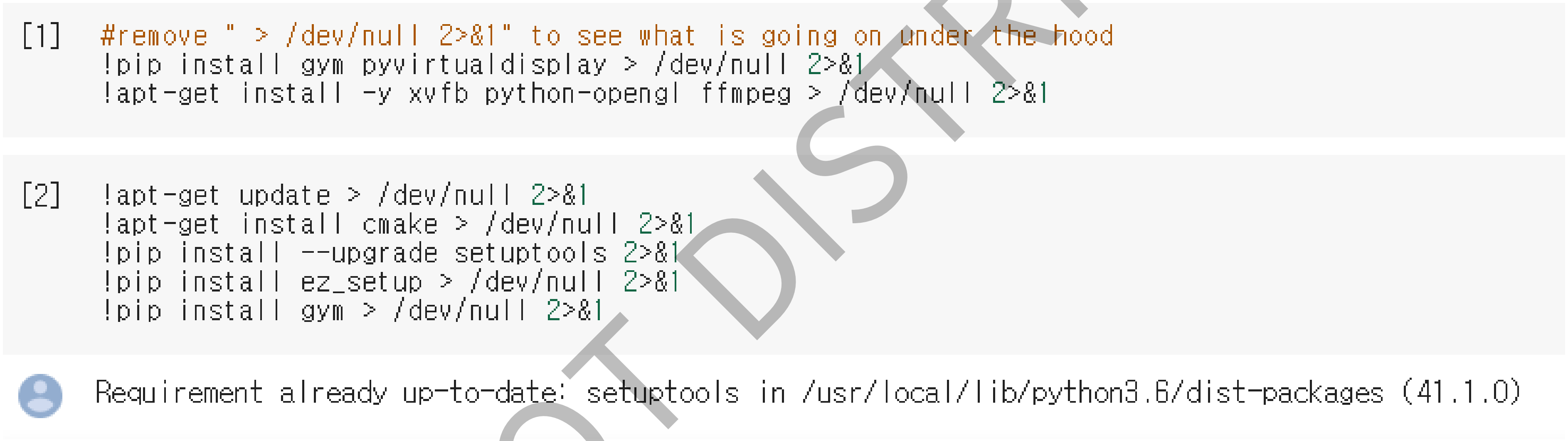
```
!apt-get update > /dev/null 2>&1
!apt-get install cmake > /dev/null 2>&1
!pip install --upgrade setuptools 2>&1
!pip install ez_setup > /dev/null 2>&1
!pip install gym > /dev/null 2>&1
```

```
Collecting setuptools
  Downloading https://files.pythonhosted.org/packages/75/b3/0a106dfaf7f48aef638da80b32608617cc8de4b24a22c8cd3759c32e5d30/setuptools-41.1.0-py2.py3-none-any.whl (576kB)
     |████████████████████████████████| 583kB 4.9MB/s
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'
Installing collected packages: setuptools
  Found existing installation: setuptools 41.0.1
    Uninstalling setuptools-41.0.1:
      Successfully uninstalled setuptools-41.0.1
Successfully installed setuptools-41.1.0
WARNING: The following packages were previously imported i
  [pkg_resources]
You must restart the runtime in order to use newly installed versions.
```

런타임 다시 시작

런타임을 다시 시작하시겠습니까? 모든 로컬 변수를 포함한 런타임 상태가 삭제됩니다.

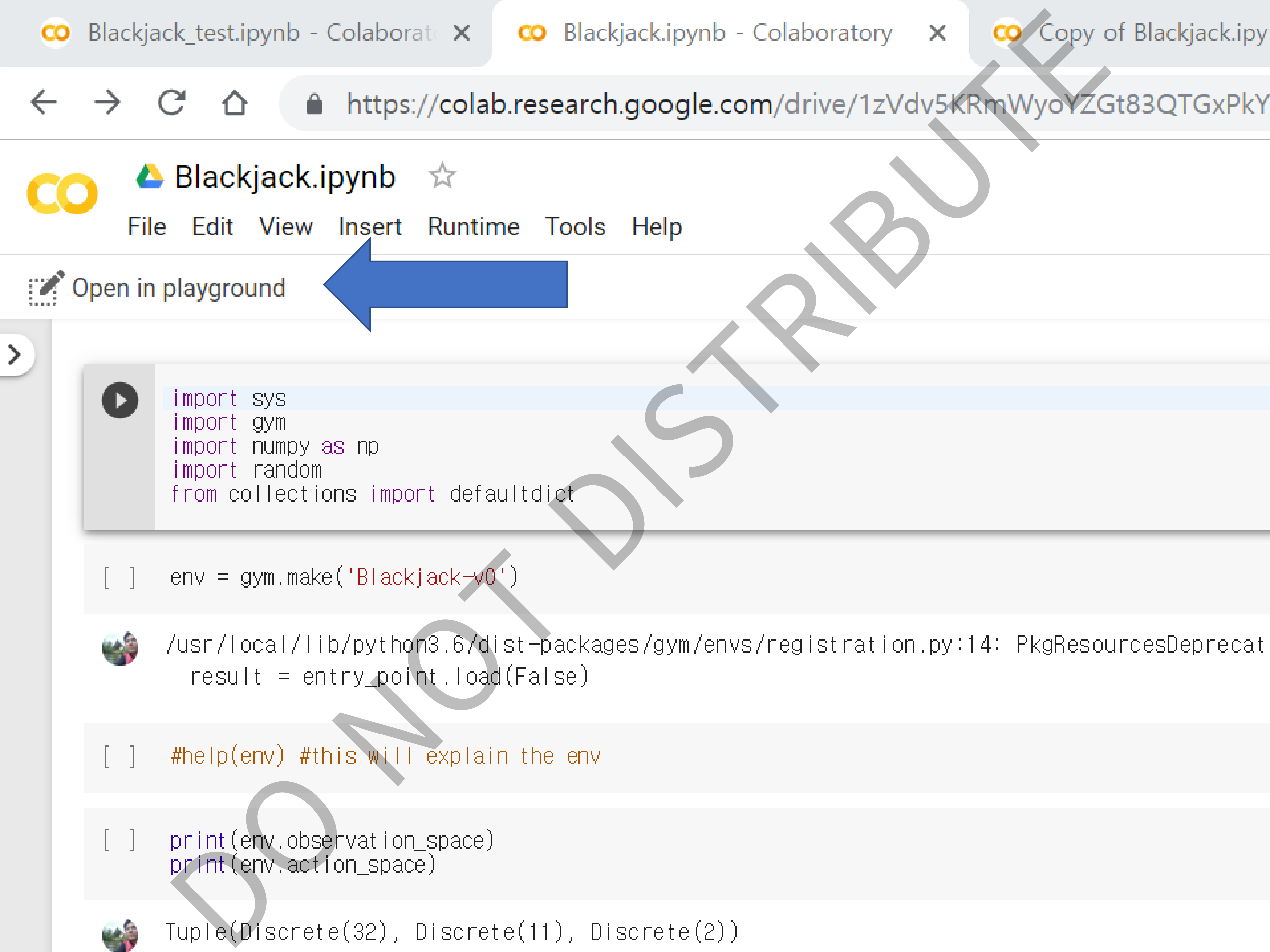
Blackjack 준비 과정



```
[1]  #remove " > /dev/null 2>&1" to see what is going on under the hood
     !pip install gym pyvirtualdisplay > /dev/null 2>&1
     !apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1
```

```
[2]  !apt-get update > /dev/null 2>&1
     !apt-get install cmake > /dev/null 2>&1
     !pip install --upgrade setuptools 2>&1
     !pip install ez_setup > /dev/null 2>&1
     !pip install gym > /dev/null 2>&1
```

```
Requirement already up-to-date: setuptools in /usr/local/lib/python3.6/dist-packages (41.1.0)
```


Blackjack_test.ipynb - Colaborat
Blackjack.ipynb - Colaboratory
Copy of Blackjack.ipy
https://colab.research.google.com/drive/1zVdv5KRmWyoYZGt83QTGxPkY
Blackjack.ipynb
File Edit View Insert Runtime Tools Help
Open in playground

```
import sys
import gym
import numpy as np
import random
from collections import defaultdict
```

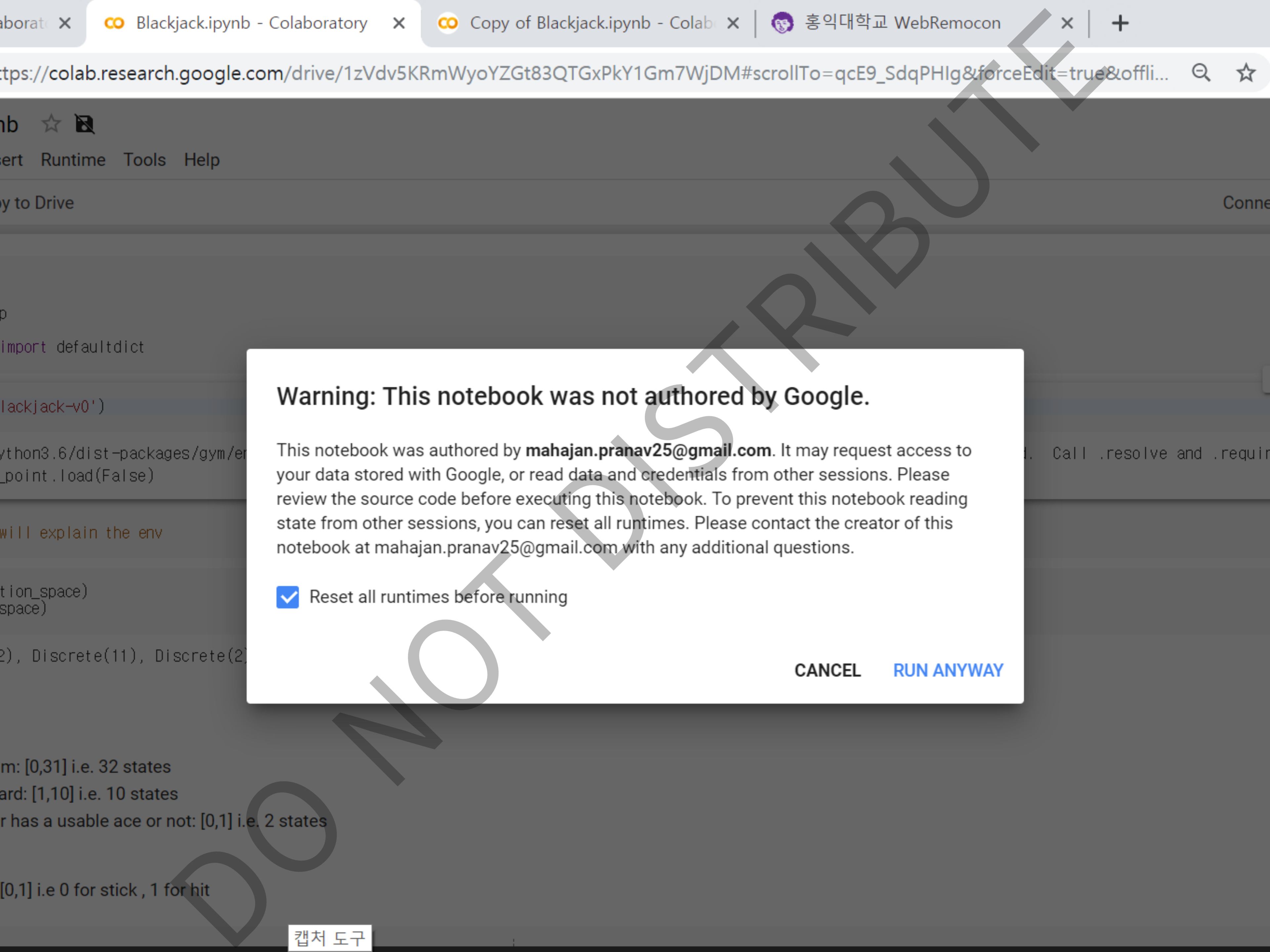
```
env = gym.make('Blackjack-v0')
```

```
/usr/local/lib/python3.6/dist-packages/gym/envs/registration.py:14: PkgResourcesDeprecat
  result = entry_point.load(False)
```

```
#help(env) #this will explain the env
```

```
print(env.observation_space)
print(env.action_space)
```

```
Tuple(Discrete(32), Discrete(11), Discrete(2))
```


Blackjack.ipynb - Colaboratory
Copy of Blackjack.ipynb - Colab
홍익대학교 WebRemocon
tps://colab.research.google.com/drive/1zVdv5KRmWyoYZGt83QTGxPkY1Gm7WjDM#scrollTo=qcE9_SdqPHlg&forceEdit=true&offli...
ert Runtime Tools Help
y to Drive
Conne
캡처 도구

```
import defaultdict
lackjack-v0')
ython3.6/dist-packages/gym/e
point.load(False)
will explain the env
tion_space)
space)
2), Discrete(11), Discrete(2
```

. Call .resolve and .requir

m: [0,31] i.e. 32 states
ard: [1,10] i.e. 10 states
r has a usable ace or not: [0,1] i.e. 2 states

[0,1] i.e 0 for stick , 1 for hit

Warning: This notebook was not authored by Google.

This notebook was authored by **mahajan.pranav25@gmail.com**. It may request access to your data stored with Google, or read data and credentials from other sessions. Please review the source code before executing this notebook. To prevent this notebook reading state from other sessions, you can reset all runtimes. Please contact the creator of this notebook at mahajan.pranav25@gmail.com with any additional questions.

[✓] Reset all runtimes before running

CANCEL RUN ANYWAY