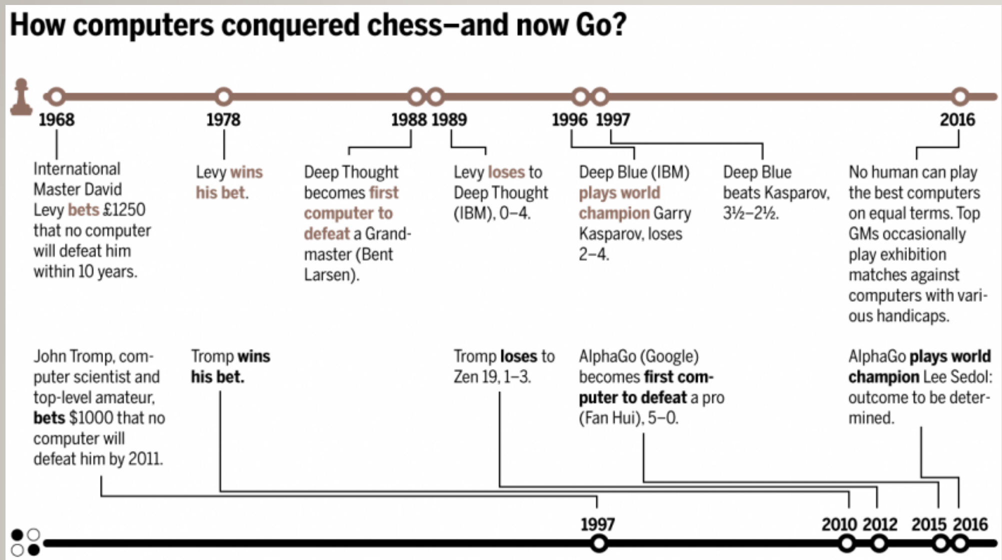
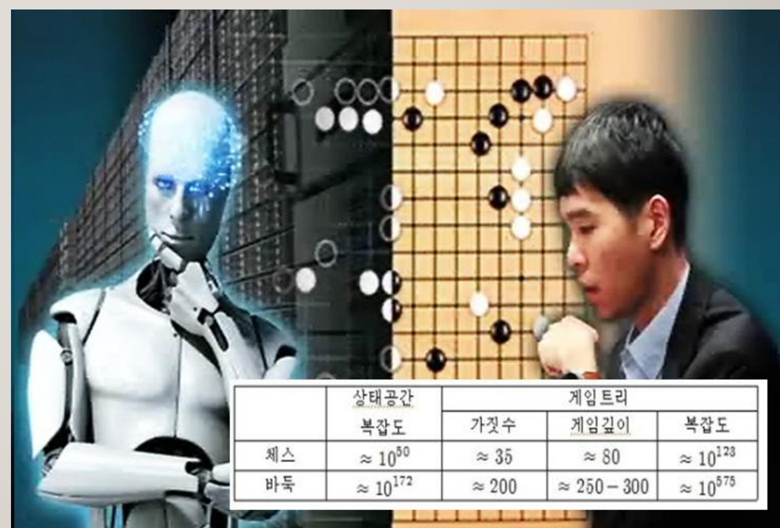
Industrial Data Science

Reinforcement Learning

Prof. Hyerim Bae

Alphago





	상태공간 복잡도	게임트리 가짓수	게임깊이	복잡도
체스	$\approx 10^{50}$	≈ 35	≈ 80	$\approx 10^{123}$
바둑	$\approx 10^{172}$	≈ 200	$\approx 250-300$	$\approx 10^{575}$

Contents

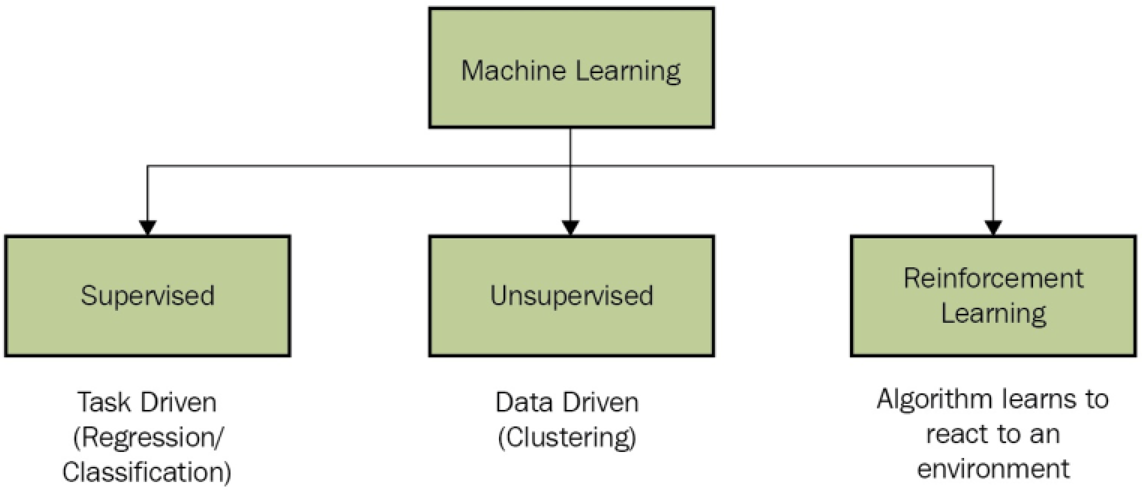
01 RL overview

02 MDP and RL

03 MC and TD

04 Q-Learning

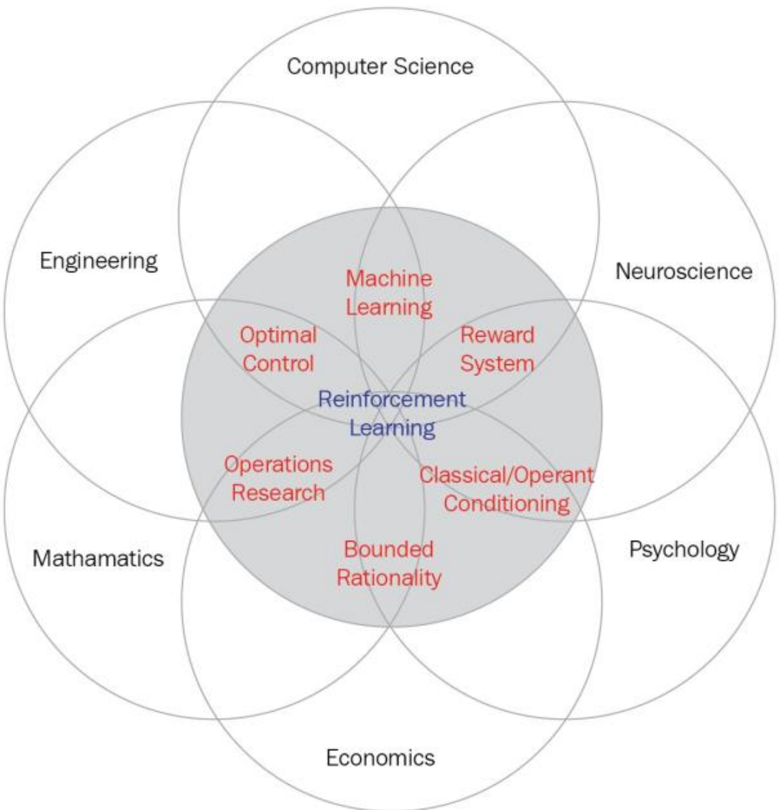
Types of Machine Learning



[Akhtar 2017]

Definition

- A type of machine learning that determines the action within a specific environ ment in order to maximize a reward
- Goal directed learning from interaction
- A mathematical structure that captures this kind of trial-and-error learning
  - Goal: to learn about the system through interaction with the system
  - Learning from the environment and learning to be more accurate with time
  - Learn about the system through the interaction
  - Learning from reward and punishment in the absence of detailed supervision

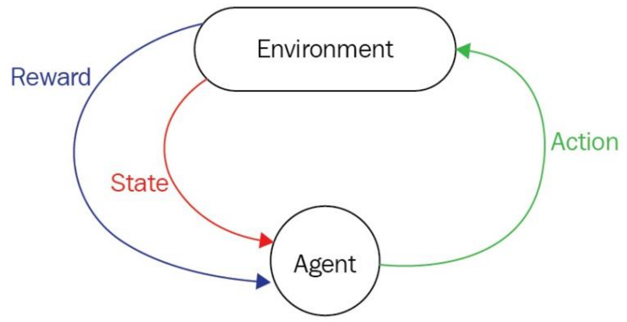


Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Mathamatics
Bounded Rationality
Psychology
Economics

[Akhtar 2017]

Reinforcement Learning Elements

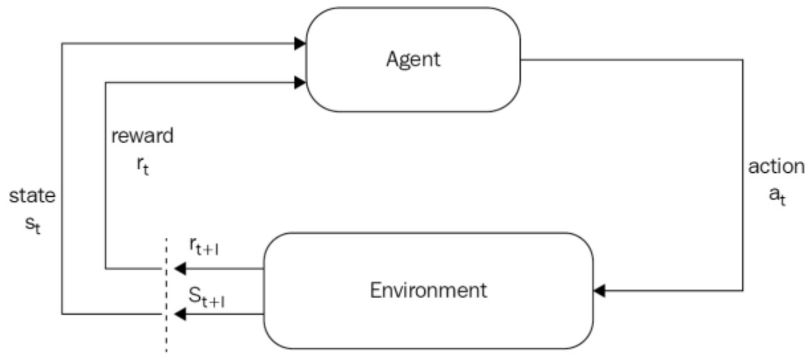
- State: the place or the situation in which the agent finds itself
- Agent: always takes actions; can perform certain actions; goal is to maximize the reward
- Action: some work based on certain scenarios
- Environment: everything outside the Agent; responds to all actions and presents new situations to the Agent
- Reward: a feedback signal; it can be plus or minus (numeric value); the environ ment gives feedback for all the actions



[Akhtar 2017]

Interaction between Agent and Environment

- Agent looks the state in environments
- Select action
- Execute the action
- Get reward with moving to the next state
- Agent update the information



$$s_0, a_0, r_1, s_1, a_1, r_2, \cdots, s_T$$

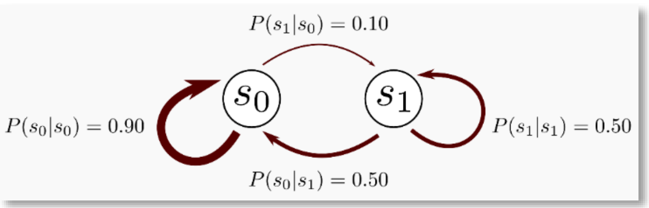
Markov Chain

- A **Markov Chain** is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

- A Markov Chain has a set of **states** $S = \{s_0, s_1, \ldots, s_m\}$ and a **process** that can move successively from one state to another. To summarize a Markov Chain is defined by:
  - Set of possible States: $S = \{s_0, s_1, \ldots, s_m\}$
  - Initial State: s_0
  - Transition Model: $T(s, s')$

- Markov Property: given the present, the future is conditionally **independent** of the past.
  - The state in which the process is now, it is dependent only from the state it was at t-1.

Fundamental Reinforcement Learning Component



source: Patacchiola (2016)

$$T = \begin{bmatrix} 0.90 & 0.10 \\ 0.50 & 0.50 \end{bmatrix}$$

The process after 2 steps:

$$\begin{pmatrix} 0.90 & 0.10 \\ 0.50 & 0.50 \end{pmatrix} \times \begin{pmatrix} 0.90 & 0.10 \\ 0.50 & 0.50 \end{pmatrix} = \begin{pmatrix} 0.86 & 0.14 \\ 0.70 & 0.30 \end{pmatrix}$$

The process after 3 steps:

$$\begin{pmatrix} 0.844 & 0.156 \\ 0.78 & 0.22 \end{pmatrix}$$

The Process after 50 steps:

$$\begin{pmatrix} 0.83333333 & 0.16666667 \\ 0.83333333 & 0.16666667 \end{pmatrix}$$

If we have initial distribution as vector V=(1,0):

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0.86 & 0.14 \\ 0.70 & 0.30 \end{pmatrix} = \begin{pmatrix} 0.844 & 0.156 \end{pmatrix}$$

The prob. of s0 and s1 after two steps

Markov Decision Process (MDP)

- Minimum path to the goal( ) from start

↑ (P=0.1) → (p=0.8) (1,3)	↑ (P=0.1) → (p=0.8) (2,3)	→ (p=0.8) (3,3)	+1
↑ (P=0.8) → (p=0.1) (1,2)		(3,2)	-1
↑ (P=0.8) → (p=0.1) Start	(2,1)	(3,1)	(4,1)

$(0.8)^5 = 0.32768$

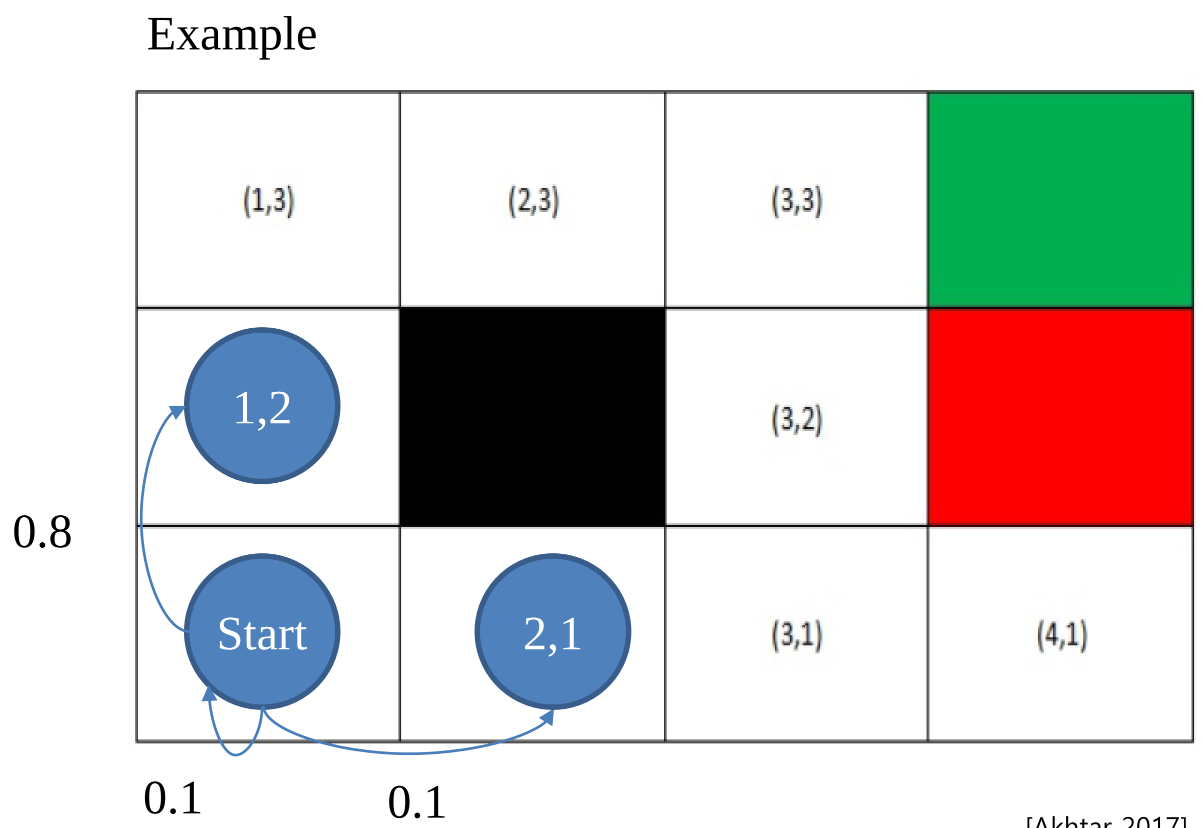
$0.1X0.1X0.1X0.1X0.8 = 0.00008$

$0.32668 + 0.00008 = 0.32776$

[Akhtar 2017]

Markov Decision Process

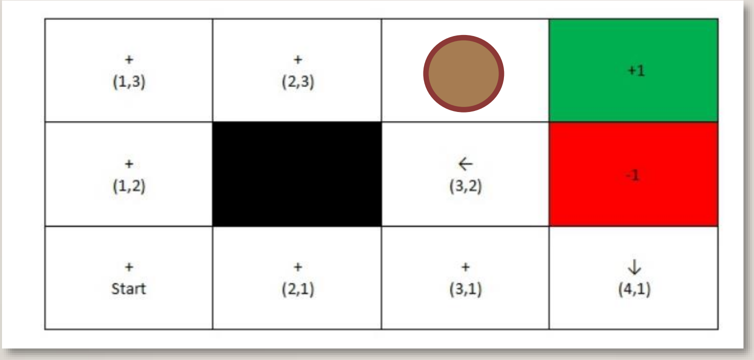
- State : 12 states, in a form of coordinates
- Action : 4 actions {UP, DOWN, LEFT, RIGHT}
- Model :



[Akhtar 2017]

Markov Decision Process

We would never go this way
since it will end the game.
Remember, our goal is
to maximize rewards

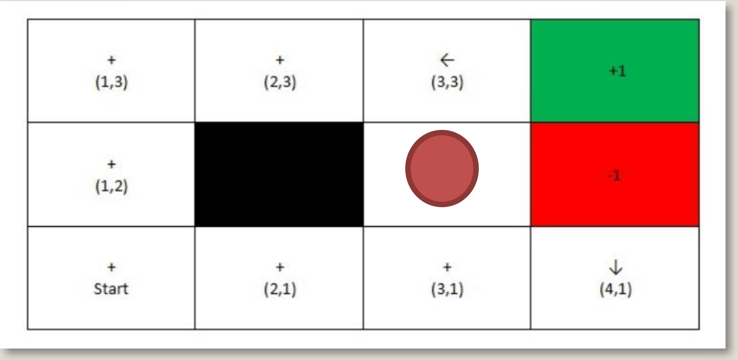


(For every step we take, we would get reward +2)

[Akhtar 2017]

Markov Decision Process

Lead to failure state, avoid this!

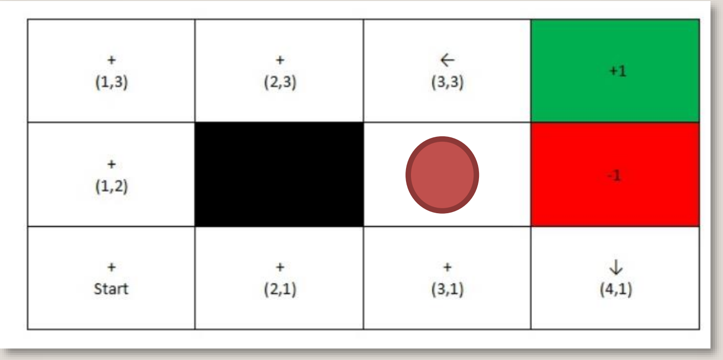


(For every step we take, we would get reward +2)

[Akhtar 2017]

Markov Decision Process

Blocked state!

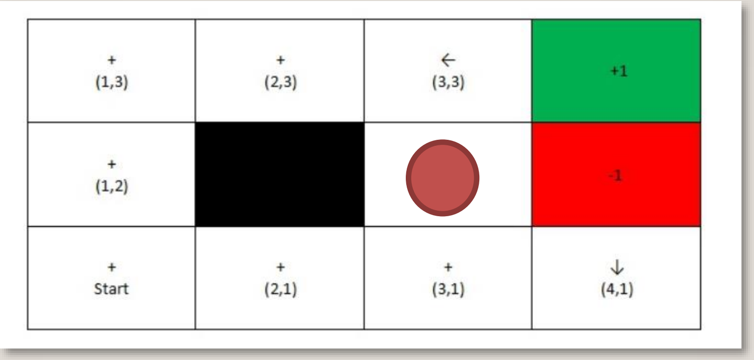


(For every step we take, we would get reward +2)

[Akhtar 2017]

Markov Decision Process

Since the reward is positive and we just accumulate it, we always avoid the goal and failure states.



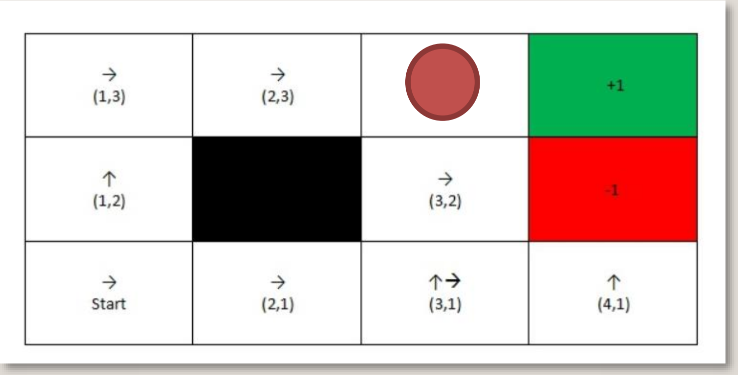
(For every step we take, we would get reward +2)

[Akhtar 2017]

Markov Decision Process

Definitely choose to go right to
end the game

Goal: exit the game as soon as possible



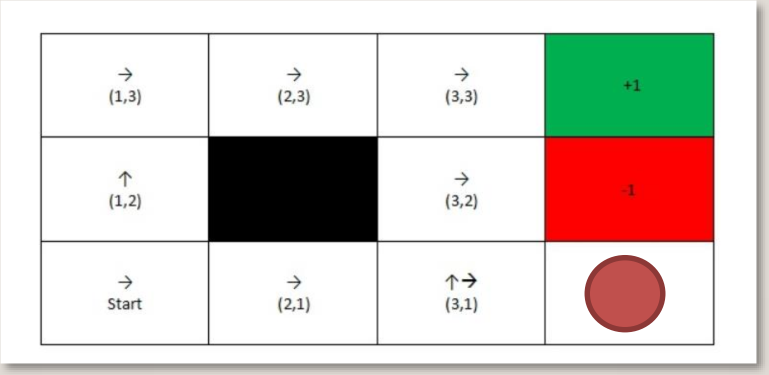
(For every step we take, we would get reward -2)

[Akhtar 2017]

Markov Decision Process

Cumulative rewards (-2) + (-2) + (-2) + 1 = -5

Goal: exit the game as soon as possible

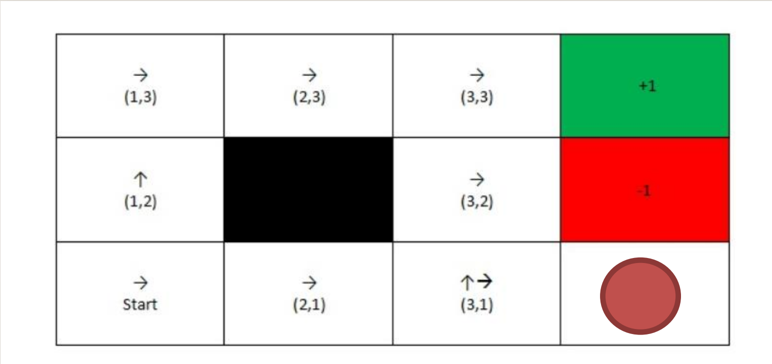


(For every step we take, we would get reward -2)

Markov Decision Process

Cumulative rewards -1, which is better than the previous options

Goal: exit the game as soon as possible



(For every step we take, we would get reward -2)

[Akhtar 2017]

Value function

- Representation of value function

$$v(s) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots \mid S_t = s]$$

$$v(s) = E[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} \cdots) \mid S_t = s]$$

$$v(s) = E[R_{t+1} + \gamma \underline{G_{t+1}} \mid S_t = s]$$

Expected value

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

Reward and state change

- Reward will be provided if an action is taken in a state

$$R_s^a = E[R_{t+1}|S_t=s, A_t=a]$$

- If an action is taken, state will change

$$P_{s->s'}^a = P[S_{t+1}=s'|S_t=s, A_t=a]$$

- Reward is delayed

$$\text{Sum of reward} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$$

Value function with reward

- Expected sum of future reward at a state

$$q(s,a) = \boldsymbol{E}[R_{t+1} + \gamma R_{t+2} + \cdots \mid S_t = s, A_t = a]$$

- Expected sum of future reward at a state when an action is taken

Policy

- Policy: Probability of selecting a in s

$$\pi(a|s) = \boldsymbol{P}[A_t = a|S_t = s]$$

- Value function in the policy:

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma R_{t+2} + \cdots |S_t = s]$$

- Q function in the policy:

$$q_\pi(s,a) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma R_{t+2} + \cdots |S_t = s, A_t = a]$$

Policy by Q function

- Greedy policy

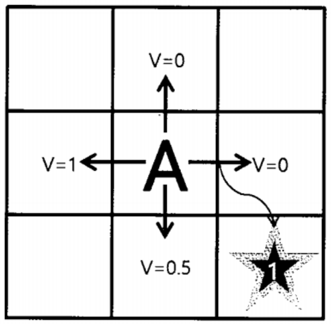
$$\pi'(s) = argmax_a \, q_{\pi}(s, a)$$

Bellman equation

- Bellman expectation equation

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s)\left(R_{t+1} + \gamma \sum_{s' \in S} P_{ss'}^{a} v_\pi(s')\right)$$

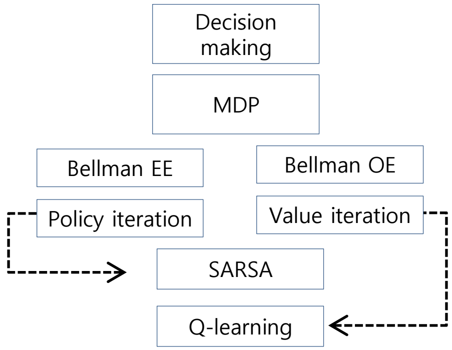


1	Up	0.25X(0+0.9X0)=0
2	Down	0.25X(0+0.9X0.5) = 0.1125
3	Left	0.25X(0+0.9X1)=0.225
4	Right	0.25X(1+0.9X0)=0.25
	V(S)	0.5875

- Bellman optimality equation

$$v_*(s) = \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

$$q_*(s,a) = E[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a]$$



Dynamic Programming

- Complexity of computation: $O(n^3)$
- Curse of Dim.
- Need perfect information of environment

Value function of DP: Expected value

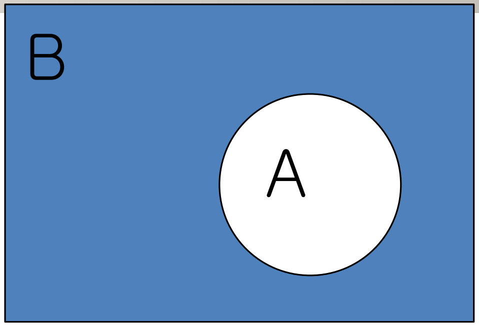
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

DP	RL
Need environment info. (Prob. of state change, reward)	Environment model is not necessary
Policy evaluation	Evaluation by Prediction
Policy iteration (policy evaluation + policy improvement)	Control (Policy improvement by prediction)
	Do it once -> evaluate -> update

Monte Carlo

- How to calculate the area of the circle A

=> Monte Carlo



Value function of DP: Expected value

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

Value function of RL: Estimate the value from samples

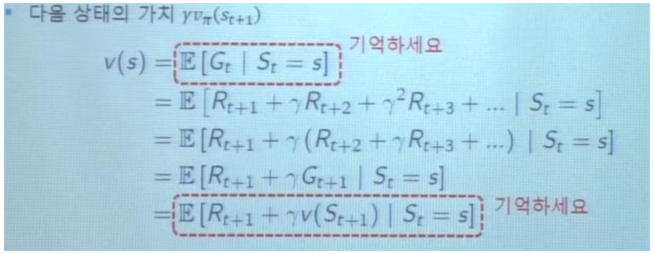
$$v_{n+1}(s) = \frac{1}{n}\left(G_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} G_i\right)$$

v_{n+1}은 현재 받은 반환값 G_n 과 이전에 받았던 반환값의 합 $\sum_{i=1}^{n-1} G_i$ 를 더한 값의 평균

$$v_{n+1}(s) = \frac{1}{n}(G_n + (n-1)V_n) = \frac{1}{n}(G_n + nV_n - V_n) = V_n + \frac{1}{n}(G_n - V_n)$$

Monte Carlo vs. TD

- Monte carlo learning



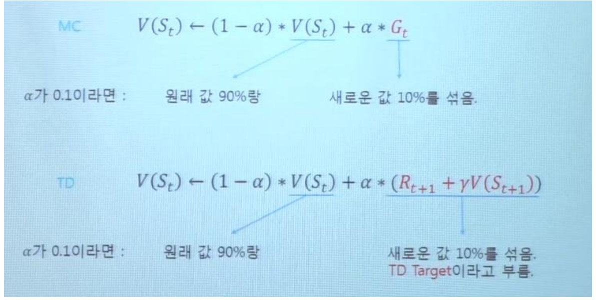
$$V(s) \leftarrow V(s) + \frac{1}{n}\underbrace{(G(s) - V(s))}_{\text{Error}}$$

- Update of value function in MC

스텝사이즈(StepSize)로서 업데이트할 때
오차의 얼마를 가지고 업데이트할지 정하는 것

$$V(s) \leftarrow V(s) + a(G(s) - V(s))$$

- Generalized update of value function in MC



- Time difference

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

- Value function in TD

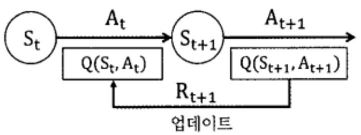
$$V(S_t) \leftarrow V(S_t) + a(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

- Update of value function in TD

현재 에이전트가 가지고 있는 값을
'가치함수 ' 일것이라고 예측

SARSA vs. Q-Learning

- In TD method,



Bellman EE → Policy Iter. → SARSA

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma Q(s',a') - Q(s,a)\right)$$

Bellman OE → Value Iter. → Q-Learning

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1},a') - Q(S_t,A_t)\right)$$

How to learn?

- Bellman equation

$$q_\pi(s,a) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma(R_{t+2} + \cdots)|S_t = s, A_t = a]$$

$$q_\pi(s,a) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

- Learning

Q-function <- Reward+discount rate X next Q

$$q(s,a) \leftarrow r + \gamma q_\pi(s', a')$$

Deterministic or stochastic

Q-function update

$$q(s,a) = q(s,a) + \alpha(r + \gamma \max_{a'} q(s',a') - q(s,a))$$

Dummy Q-learning algorithm

For each s, a initialize table entry $\hat{Q}(s, a) \leftarrow 0$

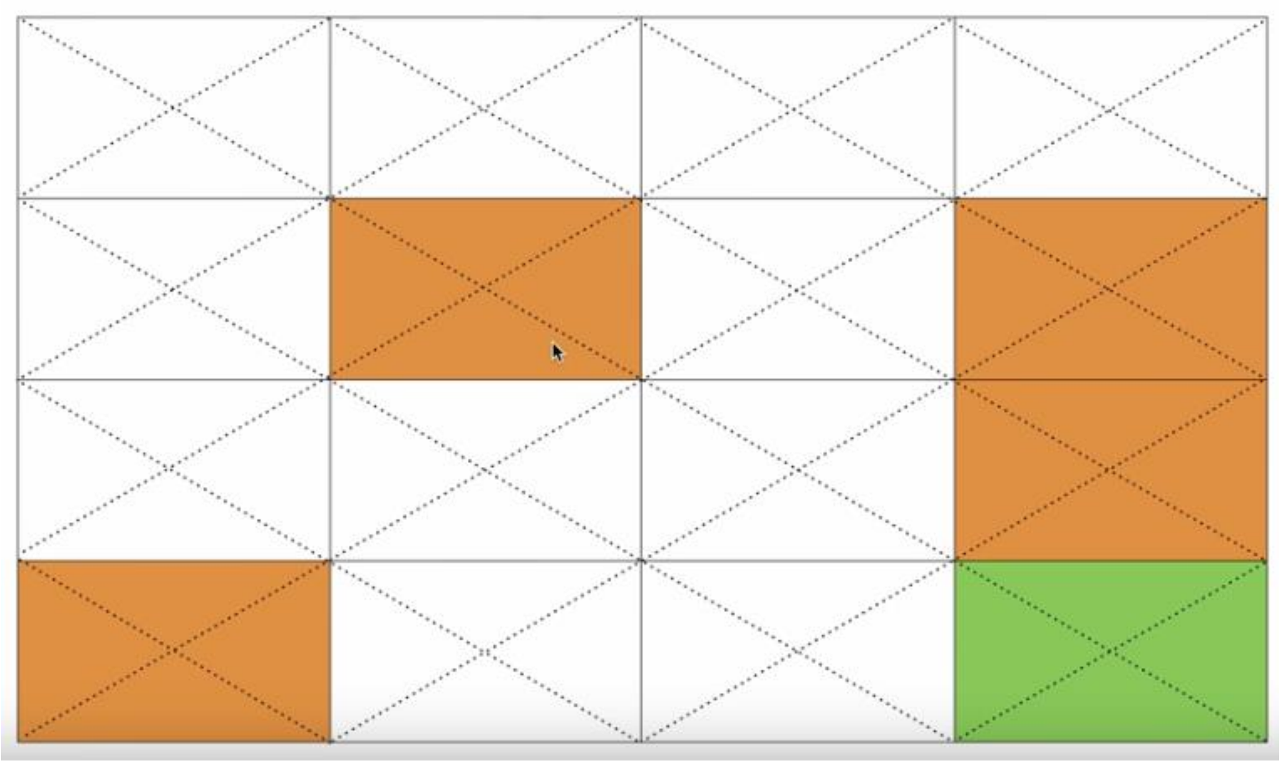
Observe current state s

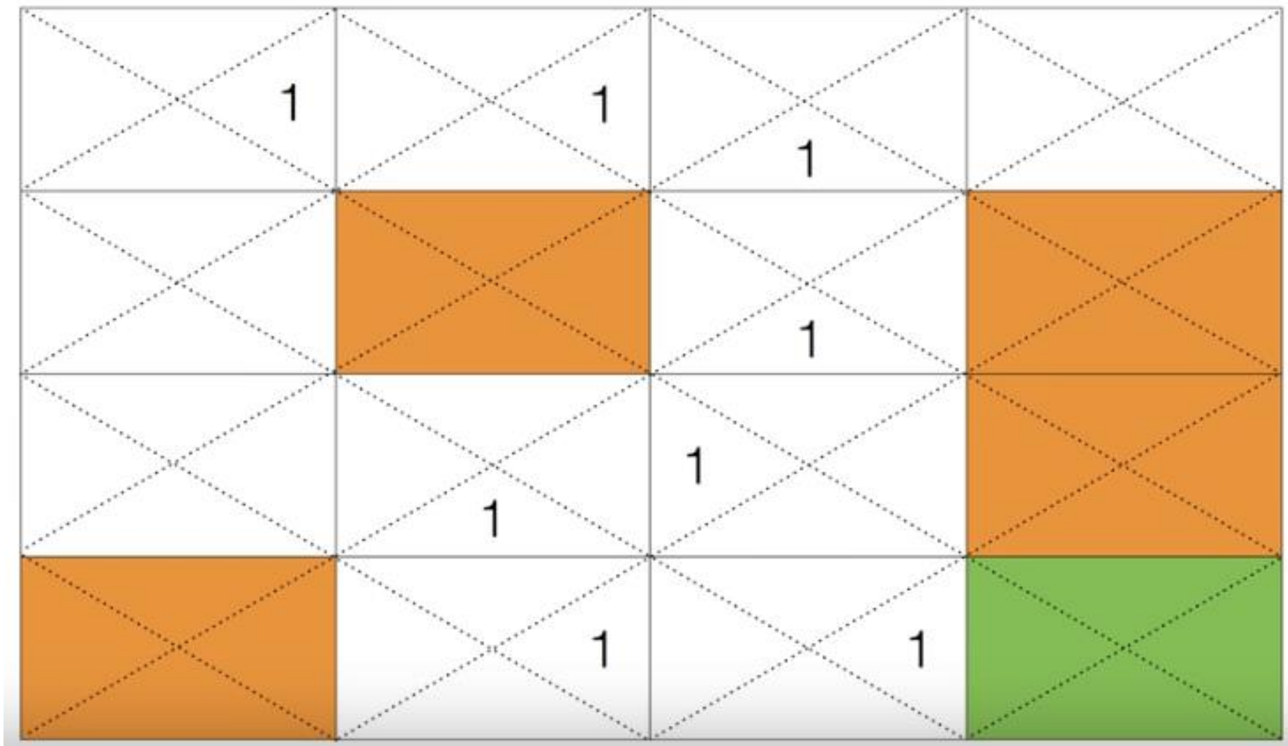
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \quad \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$




1
1
1
1
1
1
1
1

Dummy Q-learning algorithm

For each s, a initialize table entry $\hat{Q}(s,a) \leftarrow 0$

Observe current state s

Do forever:

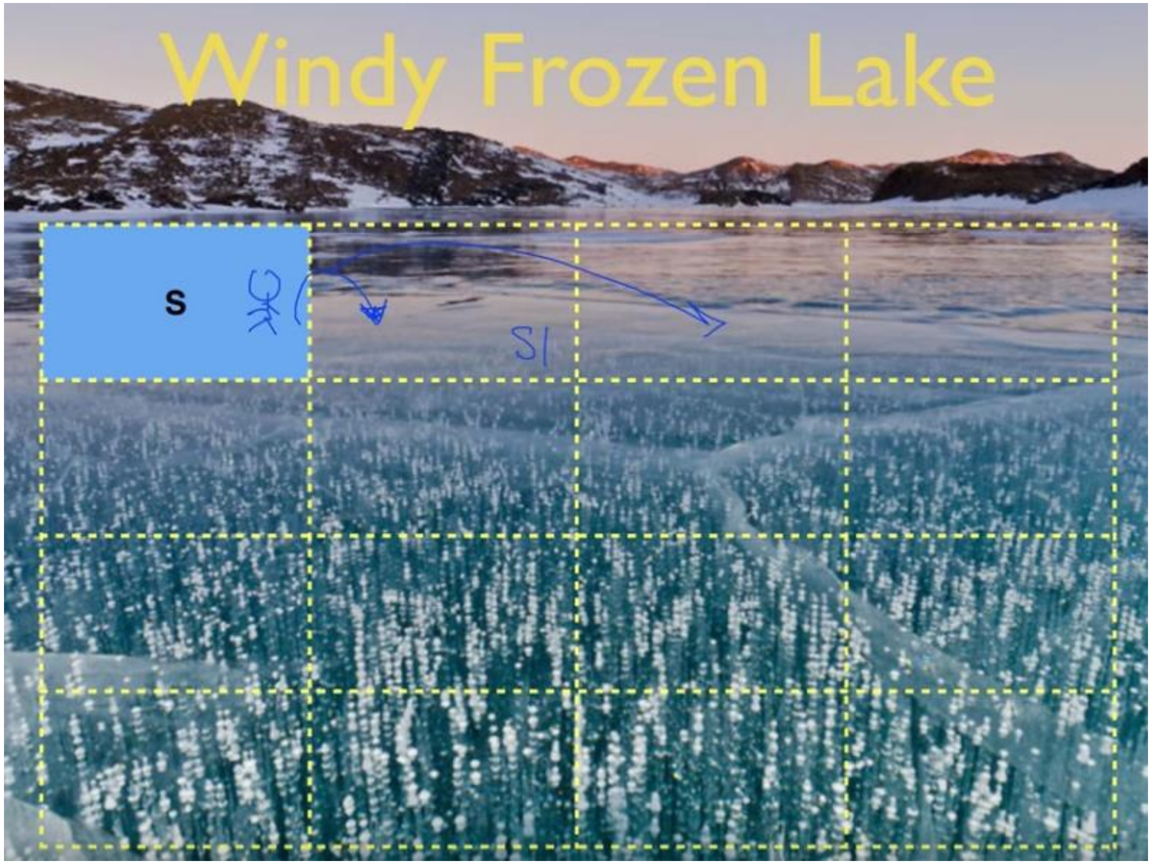
- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a')]$$

- $s \leftarrow s'$

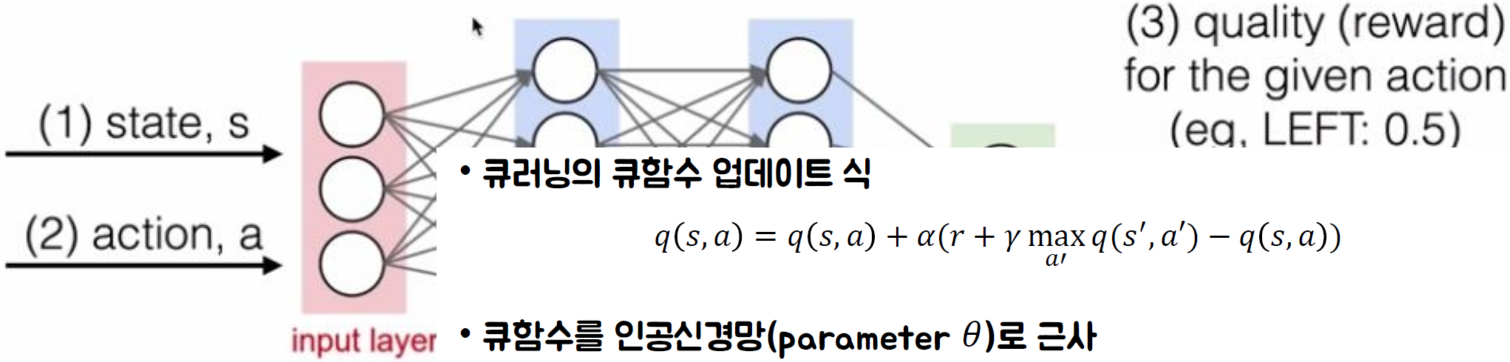
```
e = 0.1

if rand < e:
  a = random

else:
  a = argmax(Q(s, a))
```

- How much do we believe Q ?



- https://www.youtube.com/watch?v=RTu7G0y4Os4

DQN



- **큐러닝의 큐함수 업데이트 식**

$$q(s,a) = q(s,a) + \alpha(r + \gamma \max_{a'} q(s',a') - q(s,a))$$

- **큐함수를 인공신경망(parameter θ)로 근사**

$$q_\theta(s,a) = q_\theta(s,a) + \alpha(r + \gamma \max_{a'} q_\theta(s',a') - q_\theta(s,a))$$

- **큐함수가 아닌 큐함수를 근사한 인공신경망을 업데이트(MSE loss function)**

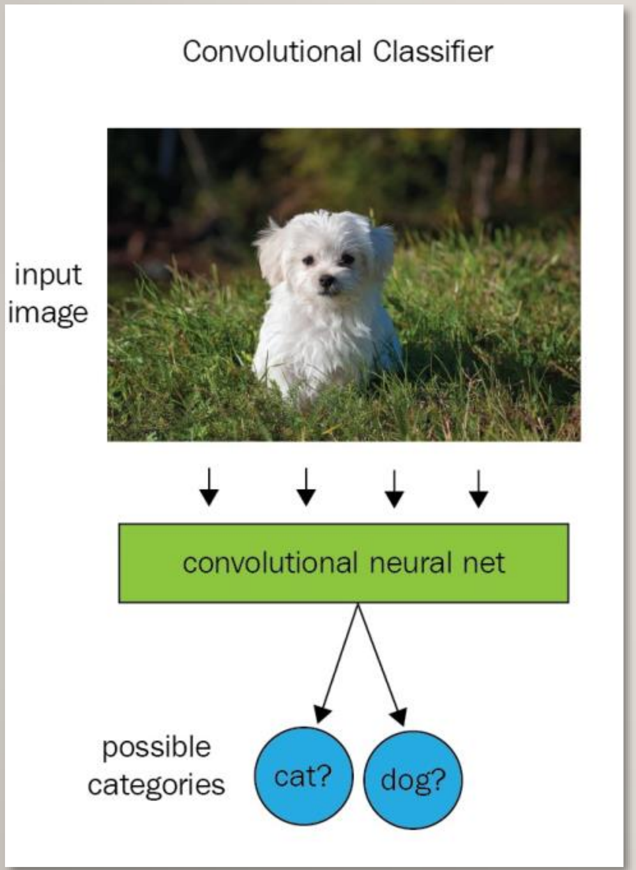
$$MSE = \Big(\underbrace{r + \gamma \max_{a'} q_\theta(s',a')}_{\text{정답}} - \underbrace{q_\theta(s,a)}_{\text{예측}}\Big)^2$$

Neural Network and Reinforcement Learning

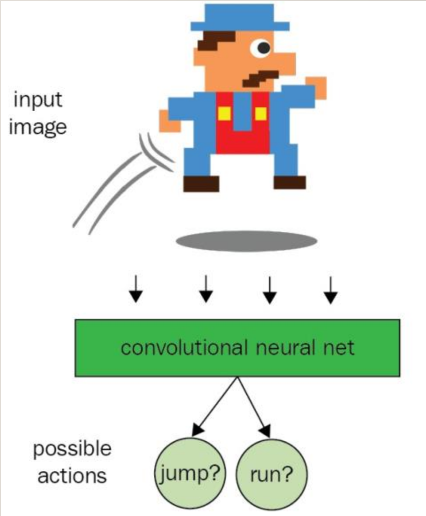
- Neural network ➔ the structure is like any other kind of network. There are interconnected nodes, which are called neurons and the edges that join them together.
  - A neural network comes in layers: **input layer**, **hidden layer** and **output layer**.
- Reinforcement learning ➔ convolutional networks are used to recognize an agent's state.

Convolutional

- Convolutional Classifier



- Convolutional Agent



RL's Drawbacks

- Slow learning
- Time and cost
- Not safe
- Hard to calculate reward
- Difficult in complex problems

Applications of RL

- Driverless car
- Natural Language Processing

- No MDP
- Hierarchy
- Multi-agent
- Efficient Learning
- Learning by user feedback

https://www.youtube.com/watch?v=UHLiMk_rwjE

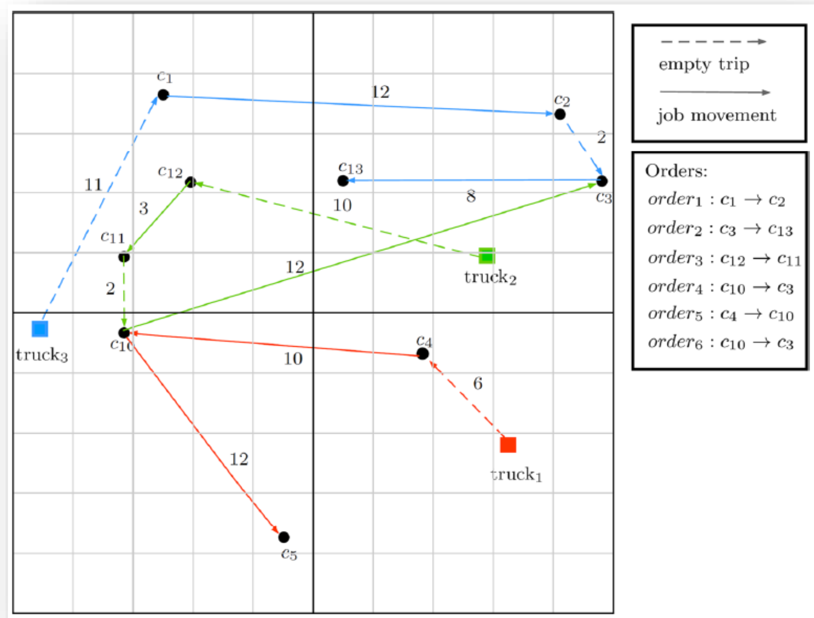
https://www.youtube.com/watch?v=_U78JajCKaw

Deep Reinforcement Learning Application

(The Inter-Terminal Truck Routing Optimization – Case Study)

What is ITTRP?

Inter-Terminal Truck Routing (ITTRP) is a combinatorial optimization and integer programming which aims to produce an optimal truck route to deliver a container between container terminal within a port.



Background

1. The truck is still a primary transportation mode to move containers among container terminal in the most Port in the world.
2. In a major Port, Inter-Terminal Transportation forms a complex transportation network.
3. A lack of Truck Scheduling and Route Planning could lead to the high operational cost (such as empty-truck trips cost), produce significant environmental impact, and low customer satisfaction.
4. in the real world, the condition is dynamically changing, and Real-time Truck Route Planning highly required and challenging.
5. The **mathematical model** and **metaheuristics approach** is not suitable for dynamic conditions.

Objective

1. Develop a robust technique/method (**using Reinforcement Learning**) to produces optimal truck routes that meet the objective of ITT Routing: Minimum Total Cost

Critical Decision

How to produce truck routes (Choose the proper order/task to be served)

That meets

Objectives of ITT Routing

1. Minimize Total Cost (TC). in our case, TC is:
   1. Empty-truck trip cost (indirectly)
   2. Transport Cost
   3. Late Cost
   4. Idle Cost

The Reinforcement Learning – State Design

Task Characteristics

Time	AT	CTP	TOeTP	TNDD	TFDD

1. **Time:** 24 Hours, 1440 Minutes – {0,1, 2, ..., 1440}
2. **Available Task (AT):** {0, 1}. 0 = No Available Task, 1 = There is Available Task
3. **Current Truck Position (CTP):** {1, 2, 3, ..., 5}. 1 = Terminal 1, etc
4. Task with Origin Location == Truck Position **(TOeTP)**: {0, 1}
5. Task with Nearest Due-Date **(TNDD)**: {0, 1}
6. Task with Farthest Due-Date **(TFDD)**: {0, 1}

Threshold < 2 hours from current Time

Threshold > 2 hours from current Time

Example:

55	1	4	1	0	1

The above state example means:

at minute 55 (Time = 55), **there are available tasks (AT = 1)**. The position of the **Truck is at Container Terminal 4 (CTP = 4).**

There are two task characteristics from the available tasks: **The Task with Origin Location equal with Current Truck Position (TOeTP = 1)** and **Task with Farthest Due-Date (TFDD = 1)**

The Reinforcement Learning – Action Design

Reinforcement Learning is utilized to learn to choose the best action from a given state.

From the example below, an agent needs to learn what is the best action to be taken {idle, choose random task, etc.} from state

0	1	1	1	1	1

Agent

0	1	1	1	1	1

Current State

Action

1. **Idle**
2. **Choose Random Task**
3. **Choose TOeTP**
4. **Choose TNDD**
5. **Choose TFDD**

Given Orders / Tasks

	A	B	F	G
1	Origin	Destination	Start Time Window (seconds)	End Time Window (hours)
2	2	3	0	300
3	0	3	180	960
4	2	1	240	1380
5	2	4	300	840
6	4	0	300	840
7	2	0	480	1380
8	1	4	480	1320
9	0	1	480	1080
10	1	0	480	840

The Reinforcement Learning – Reward Design

A reward in Reinforcement Learning is part of the feedback from the environment.

An agent uses this reward (positive for a good action or negative for bad action) to conclude how to behave in a state.

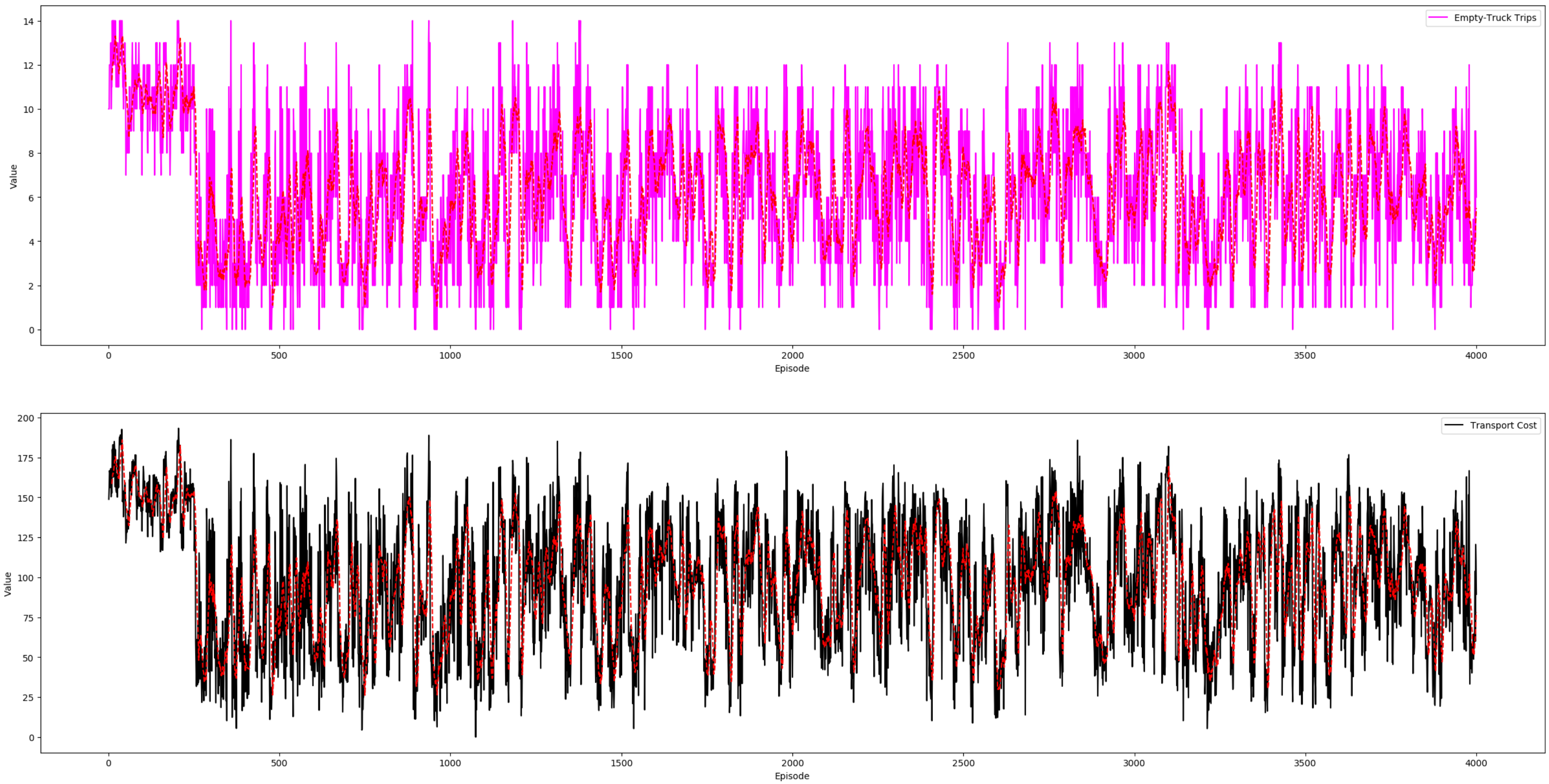
In the ITTRP case, the reward is the total cost.

Total Cost (TC) = Transport Cost + Idle Cost + Late Cost + Empty-Truck Trips Cost

The total cost from an origin to the destination of taken order/task will be a reward for the corresponding state-action pair.

The Reinforcement Learning – Training Result

The figure below is the example result of the training process that focuses on minimizing empty-truck trips and total transport costs.



The Reinforcement Learning – Testing Result (1)

The following figure shows the testing result summary in producing truck routes for 700 tasks. The result showing the computational time, and other important information.

```
++++++++++++++++++++++++++++++++++++++++++
 RL for ITTRP Testing Information
++++++++++++++++++++++++++++++++++++++++++
Number of Task Assignment File(s): 50
Computational Time: 124.70Seconds (0.03 Hours)

                                        Min.           Max.             Avg.
Num. of Served Task:                    14             18               16.16
Num. of Unserved Task:                  0              0                0.00
Num. of Late Task:                      0              4                1.12

Num. of Empty-Truck Trips:              7              16               12.06
Reward:                                 1              10               4.10

---------------------
Cost & Profit
---------------------
Empty-Truck Trips Cost:                 43.47          96.53            71.29
Transport Cost:                         140.80         201.60           168.17
Late Cost:                              0.00           63.67            14.33

Total Cost:                             144.20         243.40           182.50

Profit:                                 156.60         278.20           221.50
```

700 task in 2 minutes
0.17 sec / task

The Reinforcement Learning – Testing Result (2)

Each testing will display the routes for every agent (truck) as shown below:

```
********************************
Current Task Assigment Data File(s):  20
********************************
-------------------------
Route Agent : 1
-------------------------
Number of Served Task: 11  of  17
T5-> T4 | T4-> T1 () T3-> T5 () T3-> T4 | T4-> T2 () T3-> T2 () T4-> T5 | T5-> T2 | T2-> T4 () T1-> T3 () T1-> T4
==========================


-------------------------
Route Agent : 2
-------------------------
Number of Served Task: 6  of  17
T1-> T2 | T2-> T3 () T2-> T5 | T5-> T3 | T3-> T1 | T1-> T5
==========================


Total Task:  17
Served Task:  17
Unserved Task:  0
Total Reward:  10

Num. of Empty-truck trips:  7
Num. of Late Task:  0

Revenue:  425
Empty-truck trips Cost:  43.46666666666667
Transport Cost:  146.8
Late Cost:  0
Total Cost:  146.8
Profit:  278.19999999999993
```

In term of empty-truck trips, Reinforcement Learning outperformed Simulated Annealing as shown in the following table:

		Reinforcement Learning		Simulated Annealing	
No	**Orders**	**Lowest Empty Truck Trips Found**	**t (s)**	**Lowest Empty Truck Trips Found**	**t (s)**
1	10	4	0.7	4	1.6
2	20	10	1.43	9	6.2
3	40	19	6.64	21	50.23
4	80	29	9.22	34	125
5	180	62	59.46	89	1066