Introduction to Q - Learning

220150026
Ryu Joon su

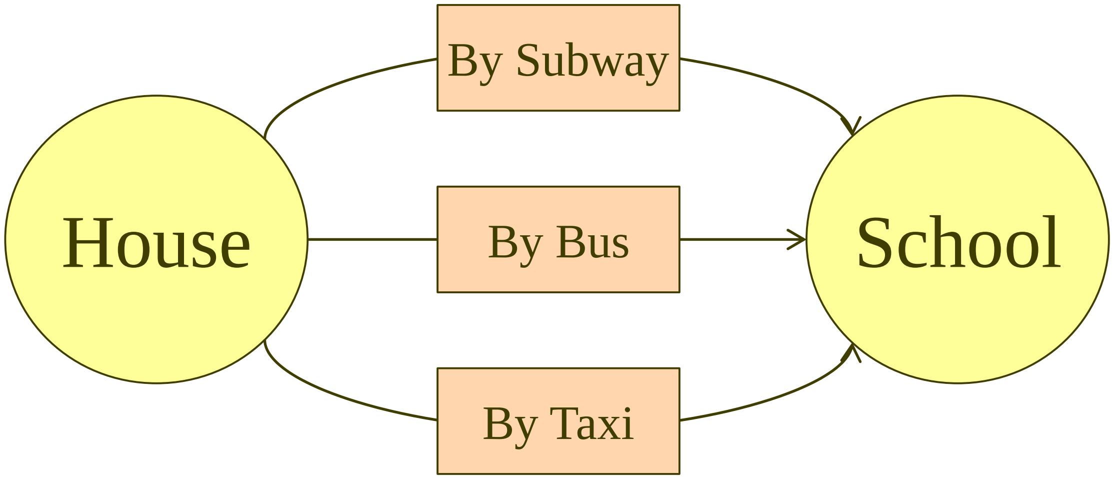
Overview

- 6/1
  - Markov Decision Process
  - Reinforcement Learning
  - Definition of Q – Learning and it's Algorithm
  - Simple Example
  - Proof of Convergence

- 6/8
  - Playing Atari with Deep Reinforcement Learning
  - Deep Q – Learning

Markov Decision Process

- Markov Decision Process (MDP) is a mathematical framework for modeling decision making in situations where outcome are partly random and partly under the control of a decision maker.



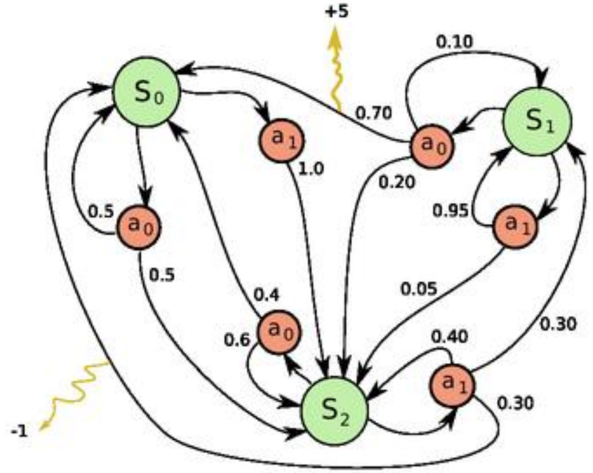
Markov Decision Process (cont.)

- Definition
  - A Markov Decision Process is a 5-tuple (S, A, T, R, γ)

  - S : finite set of states
  - A : finite set of actions
  - T : transition probability
  - R : reward function (or C : cost function)
  - γ : discount factor $(0 \leq \gamma \leq 1)$

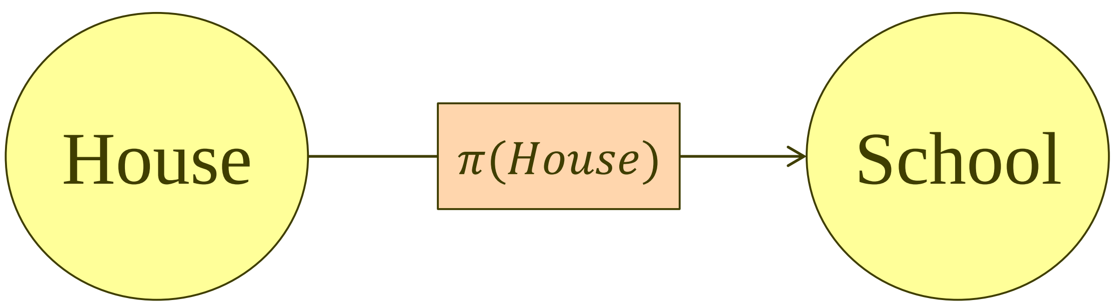
Markov Decision Process (cont.)

- Deterministic MDP
  - If you select an action, then you can do this action with probability 1.
- Nondeterministic MDP
  - If you select an action, then you can do this action with any probability.



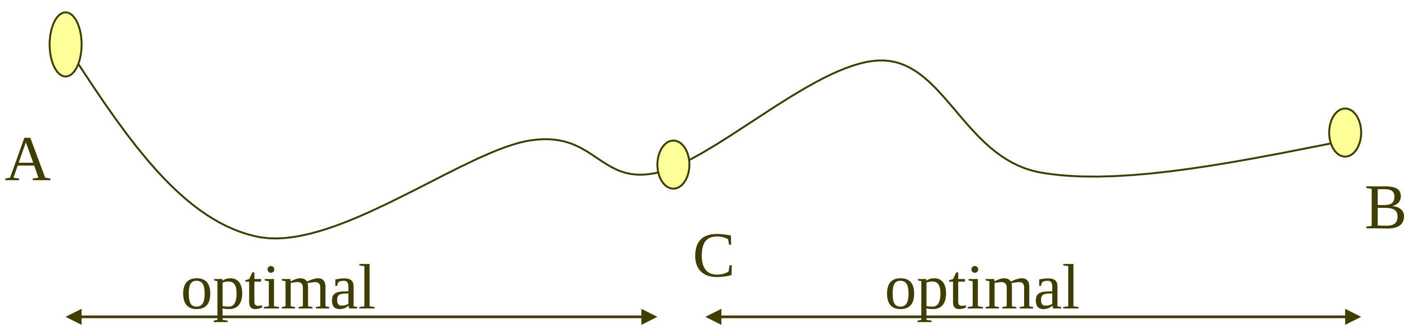
Markov Decision Process (cont.)

- Goal
  - To find a "policy" for the decision maker
  - A function π that specifies the action $\pi(s)$ that the decision maker will choose when in state s.
  - The goal is to choose a policy π that will maximize some cumulative function of the random rewards, typically the expected discount sum over a potentially infinite horizon.



Markov Decision Process (cont.)

- Solution Concept
  - If C belongs to an optimal path from A to B, then the sub-path A to C and C to B are also optimal.
  - Therefore, all sub-path of an optimal path is optimal.



Markov Decision Process (cont.)

- Solution

  - Bellman's Optimality Equation

$$V^*(s) = \sum_{s'} P(s, s', \pi(s))\big(R(s, s', \pi(s)) + \gamma V^*(s')\big)$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left\{ \sum_{s'} P(s, s', a)\big(R(s, s', a) + \gamma V^*(s')\big) \right\}$$

Can be implemented by **linear programming** or **dynamic programming**.

Markov Decision Process (cont.)

- Solution Algorithms

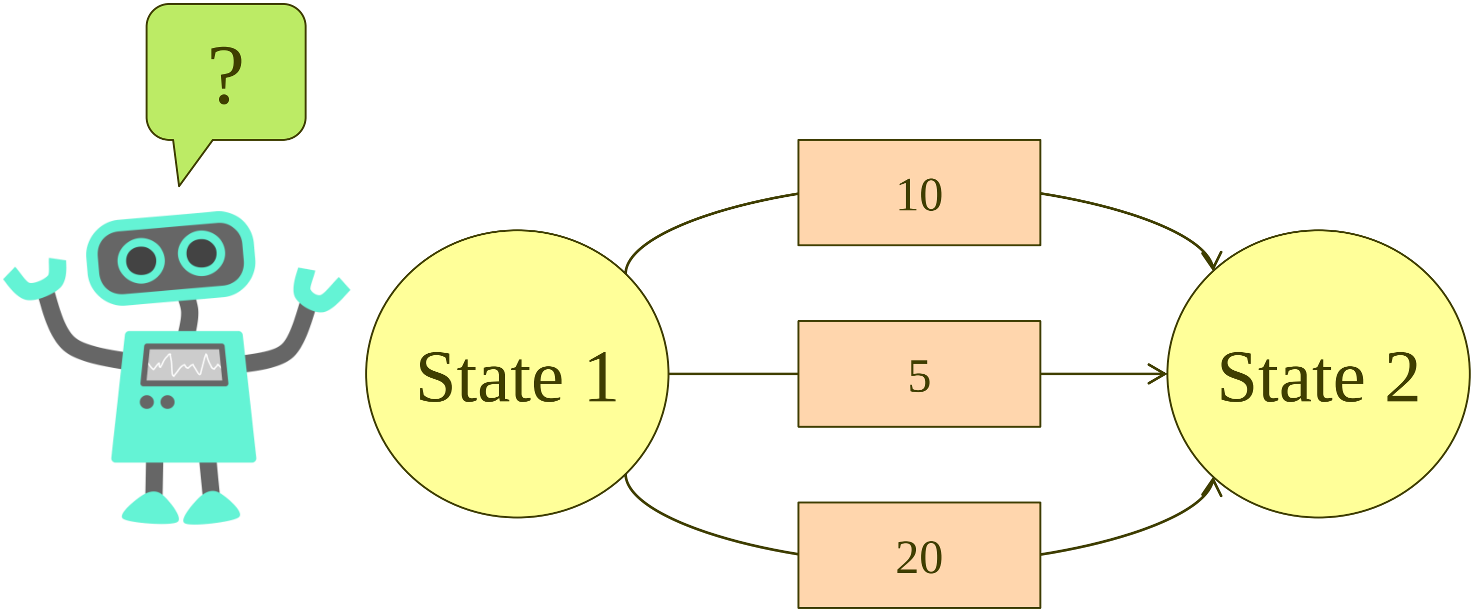
  - Value Iteration (VI)
    - Also called **Backward Induction**.
    - π function is not used.
    - Computing $V_0 \rightarrow V_1 \rightarrow \cdots$ until V converges

  - Policy Iteration (PI)
    - Policy Improvement Technique
    - Given π defines a new policy π' such that

$$\pi'(s) = \operatorname*{argmax}_{a \in A} \left\{ \sum_{s'} P(s, s', a) R(s, s', a) + \gamma \sum_{s'} P(s, s', a) V(s') \right\}$$

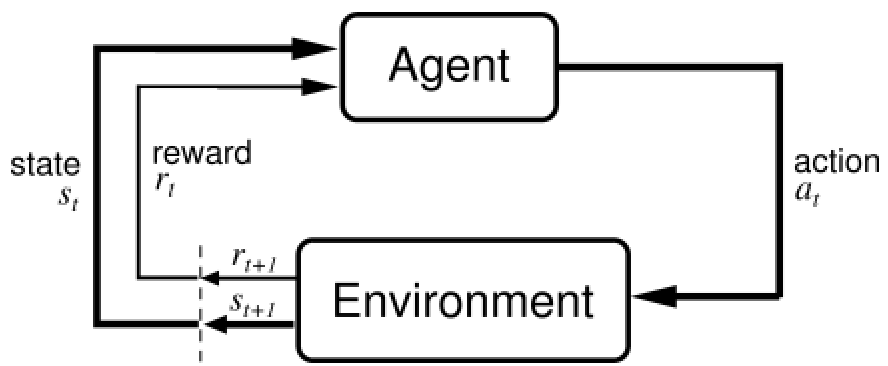
Limitation

- Most of cases, computers do not know reward function until they really do actions.



To Overcome Limitation

- Reinforcement Learning
  - An area concerned with how an agent ought to take actions in an environment so as to maximize some notion of reward.
  - The task of Reinforcement Learning (RL) is to use **observed rewards** to find an optimal policy for the environment.



Reinforcement Learning

- Kind of Reinforcement Learning
  - **Q – Learning**
  - Temporal Difference (TD) Learning
  - State-Action-Reward-State-Action (SARSA) Learning
  - Etc.

Q – Learning

- Q – Learning is a model-free reinforcement technique.
- Can be used to find an optimal action policy for any given MDPs.
- Converges to an optimal policy in both deterministic and nondeterministic MDPs.

Q – Learning (cont.)

- Q function (nondeterministic)
  - Define a functions $Q^* : S \times A \rightarrow \mathbb{R}$ such that

$$Q^*(s,a) = \sum_{s' \in S} P(s,s',a)R(s,s',a) + \gamma \sum_{s' \in S} P(s,s^*,a)V^*(s')$$

  - Measure of how good to take an action $a \in A$ at state $s \in S$ if an optimal policy is followed from the possible next state of s.

Q – Learning Algorithm

1. Initialize $Q_0(s, a)$ arbitrarily
2. Repeat
3. Choose a_t from s_t using an exploratory policy ϕ_t
4. Take action a_t, observe r, s_{t+1}
5. Update Q – value function such that

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha \left[r + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right]$$

6. $t \leftarrow t + 1$

Q – Learning Algorithm (cont.)

- Learning rate (α)
  - The learning rate determines to what extent the newly acquired information will override the old information.
  - A factor of 0 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information.

- Discount factor (γ)
  - The discount factor determines the importance of future rewards.
  - A factor of 0 will make the agent "opportunistic" by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward.

Exploratory Policy

- ε-greedy policy

At time t

$$\phi_t \begin{cases} \text{with probability } \varepsilon_t(s) = \dfrac{c}{n_t(s)} \text{ select action } a \in A \text{ with prob } \dfrac{1}{|A|} \\ \text{with probability } 1 - \varepsilon_t(s) \text{ select an action } in \ \underset{a \in A}{\operatorname{argmax}} Q_t(s_t, a\ ) \end{cases}$$

※ $n_t(s)$ is number of visits to state s in time step t, $0 < c < 1$

Note if

$$\lim_{t \to \infty} n_t(s) = \infty, \lim_{t \to \infty} \varepsilon_t(s) = 0$$

So that ϕ_t becomes more greedy selection rule with respect to Q_t at t.

Exploratory Policy (cont.)

- Boltzmann Exploration – Exploitation rule

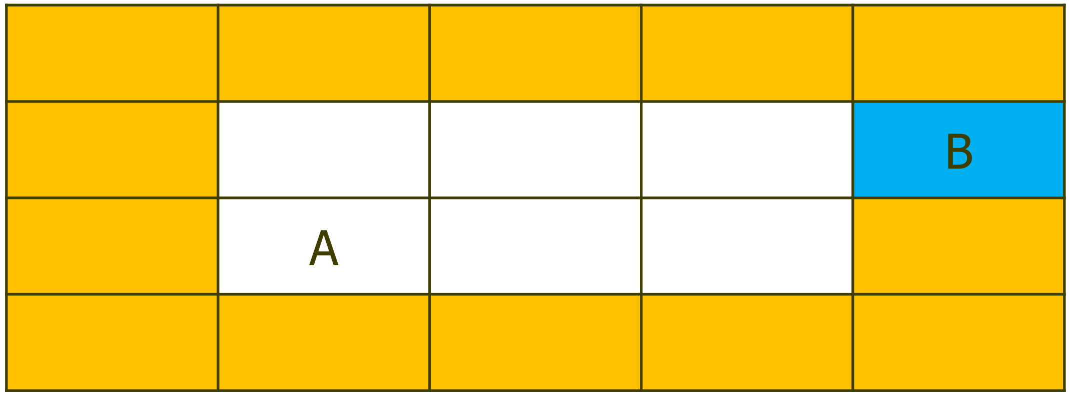
$$\phi_t^a(s) = \frac{e^{T_t\ (s) Q_t\ (s,a)}}{\sum_{b \in A} e^{T_t\ (s) Q_t\ (s,b)}}$$

Where $T_t\ (s)$ is called **temperature** associated with s

Note if more $T_t\ (s)$ lower, $\phi_t^a(s)$ become uniform selection

if more $T_t\ (s)$ higher, $\phi_t^a(s)$ become greedy selection with respect to a

Q – Learning Example



A is start and our goal is end in B

- Orange location reward = -8
- White location reward = 0
- Blue location reward = 8

Q – Learning Example (cont.)

- **States**

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

- **Actions**

North (↑), South(↓), East(→), West(←)

Q – Learning Example (cont.)

- The $Q(s, a)$ function

states

actions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
↑																				
↓																				
←																				
→																				

Q – Learning Example (cont.)

1. **Initialize $Q_0(s, a)$ arbitrarily**
2. Repeat
3. Choose a_t from s_t using an exploratory policy ϕ_t
4. Take action a_t, observe r, s_{t+1}
5. Update Q – value function such that

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha \left[r + \gamma \max_a Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t) \right]$$

6. $t \leftarrow t + 1$

Q – Learning Example (cont.)

- The $Q(s, a)$ function

states

actions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
↑	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	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Q – Learning Example (cont.)

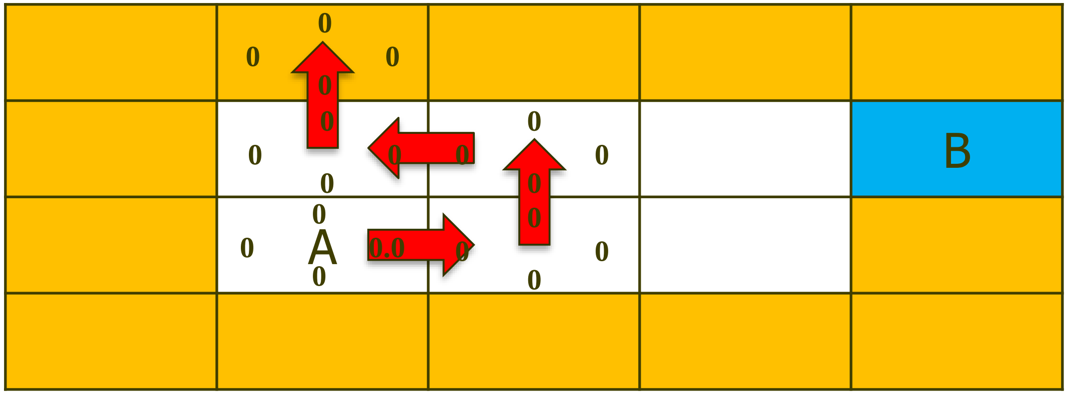
1. Initialize $Q_0(s,a)$ arbitrarily
2. Repeat
3. **Choose $\boldsymbol{a_t}$ from $\boldsymbol{s_t}$ using an exploratory policy $\boldsymbol{\phi_t}$**
4. **Take action $\boldsymbol{a_t}$, observe $\boldsymbol{r}$, $\boldsymbol{s_{t+1}}$**
5. Update Q – value function such that

$$Q_{t+1}(s_t,a_t) \leftarrow Q_t(s_t,a_t) + \alpha\left[r + \gamma \max_a Q_t(s_{t+1},a_t) - Q_t(s_t,a_t)\right]$$

6. $t \leftarrow t+1$

Q – Learning Example (cont.)

- An episode



Q – Learning Example (cont.)

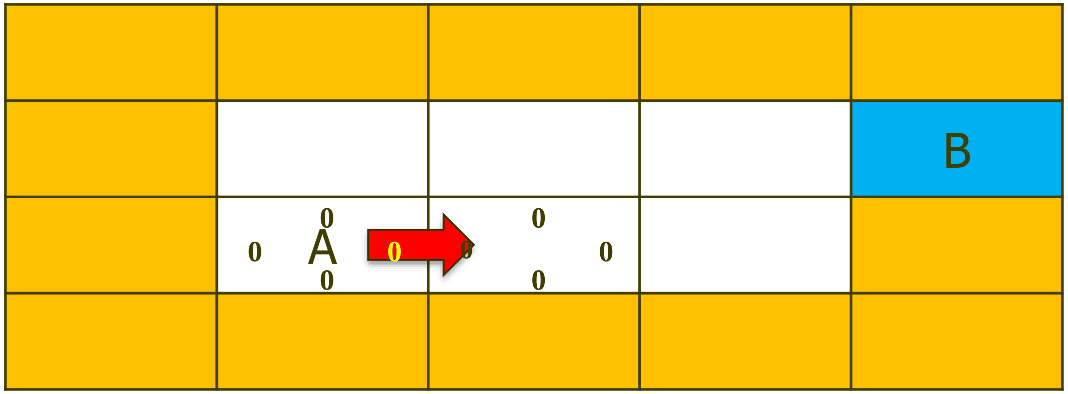
1. Initialize $Q_0(s, a)$ arbitrarily
2. Repeat
3. Choose a_t from s_t using an exploratory policy ϕ_t
4. Take action a_t, observe r, s_{t+1}
5. **Update Q – value function such that**

$$\boldsymbol{Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha\left[r + \gamma \max_a Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t)\right]}$$

6. $t \leftarrow t + 1$

Q – Learning Example (cont.)

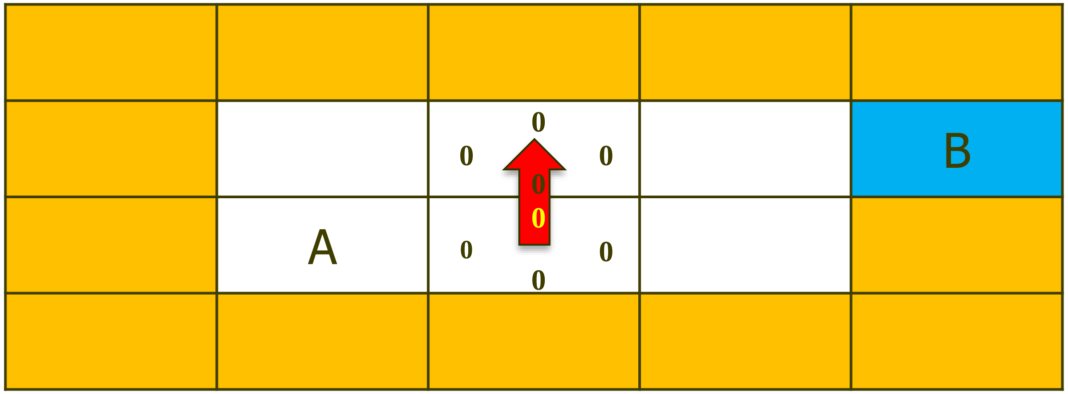
- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_{12}, \rightarrow) \leftarrow 0 + 1 \times [0 + 0.5 \times 0 - 0]$$

Q – Learning Example (cont.)

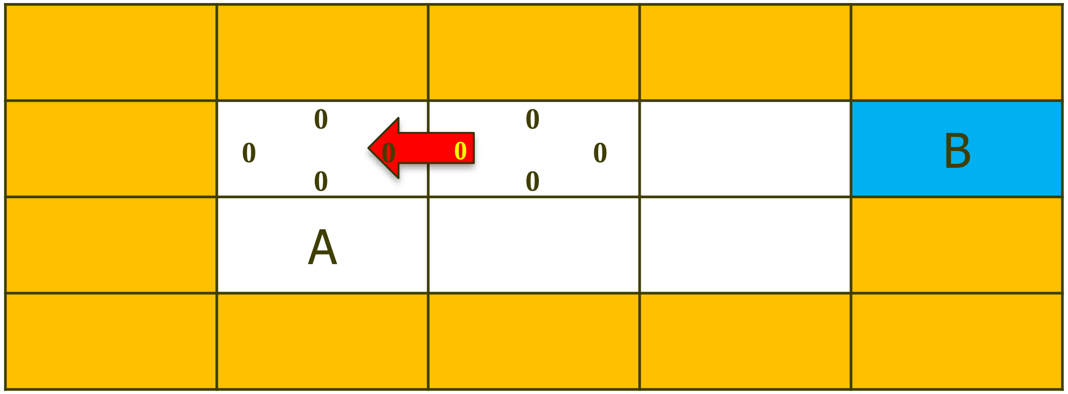
- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_{13}, \uparrow) \leftarrow 0 + 1 \times [0 + 0.5 \times 0 - 0]$$

Q – Learning Example (cont.)

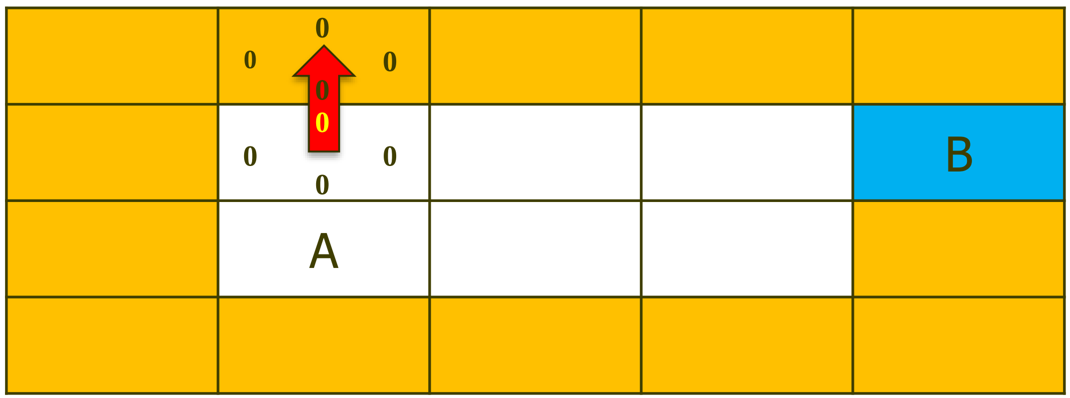
- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_8, \leftarrow) \leftarrow 0 + 1 \times [0 + 0.5 \times 0 - 0]$$

Q – Learning Example (cont.)

- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_7, \uparrow) \leftarrow 0 + 1 \times [-8 + 0.5 \times 0 - 0]$$

Q – Learning Example (cont.)

- The Q table after the first episode

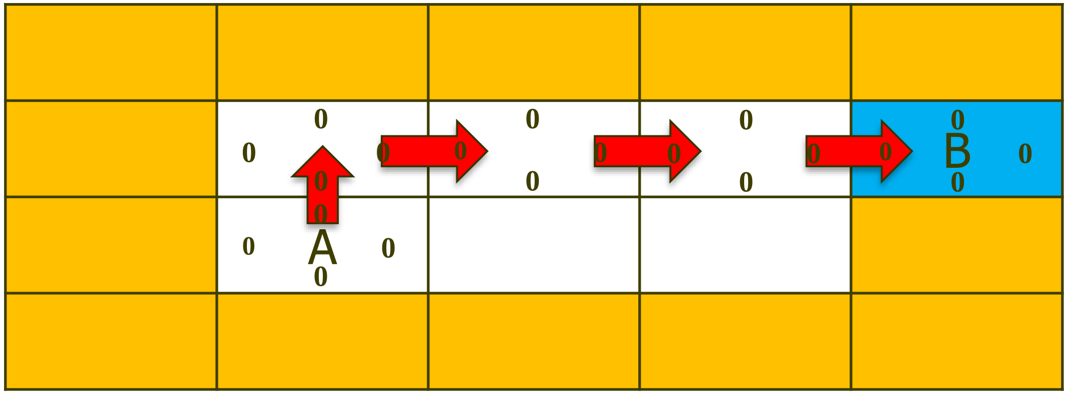
states

actions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
↑	0	0	0	0	0	0	**-8**	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	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	**0**	0	0	0	0	0	0	0	0

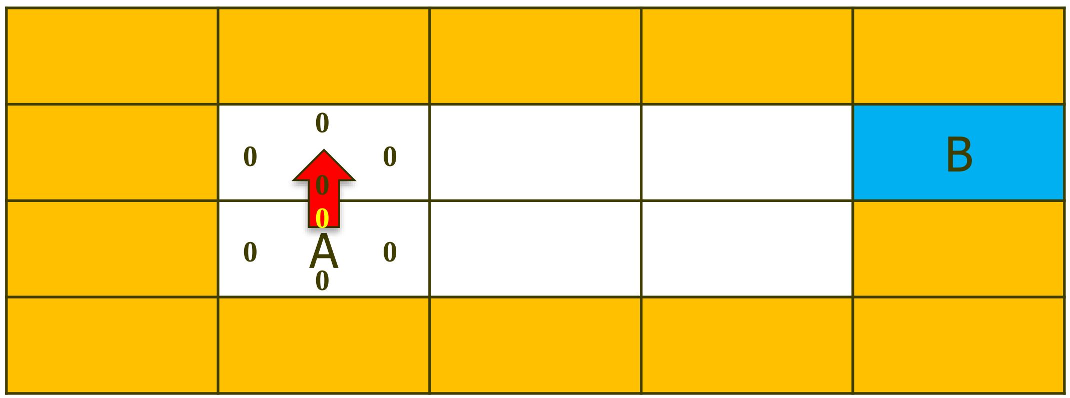
Q – Learning Example (cont.)

- A second episode



Q – Learning Example (cont.)

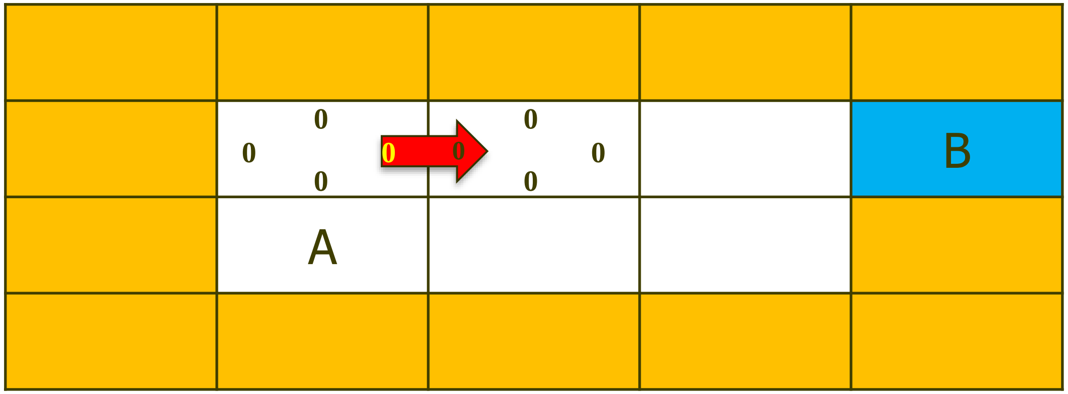
- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_{12}, \uparrow) \leftarrow 0 + 1 \times \left[0 + 0.5 \times \max\{-8, 0, 0, 0\} - 0\right]$$

Q – Learning Example (cont.)

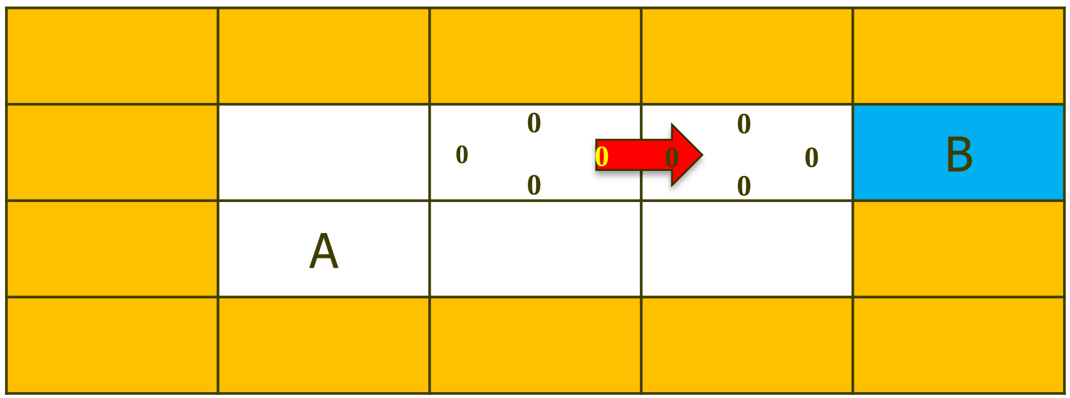
- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_7, \rightarrow) \leftarrow 0 + 1 \times [0 + 0.5 \times 0 - 0]$$

Q – Learning Example (cont.)

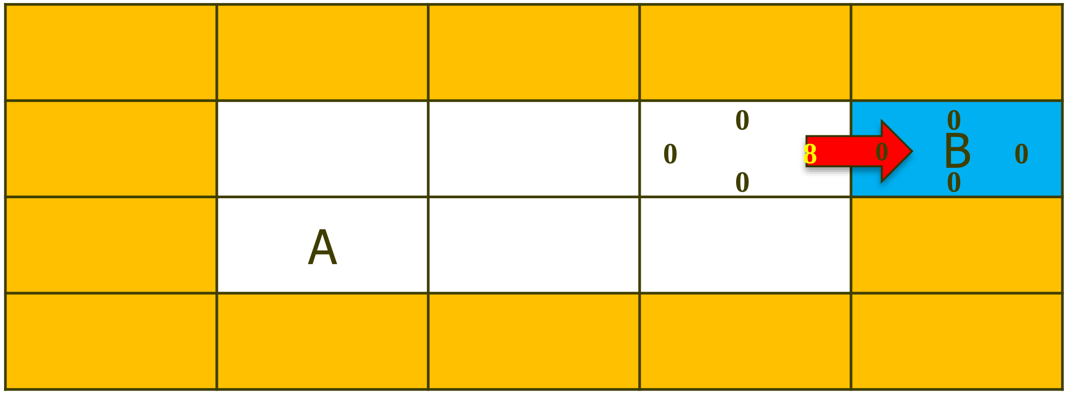
- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_8, \rightarrow) \leftarrow 0 + 1 \times [0 + 0.5 \times 0 - 0]$$

Q – Learning Example (cont.)

- We assume that $\alpha = 1, \gamma = 0.5$



$$Q(s_9, \rightarrow) \leftarrow 0 + 1 \times [8 + 0.5 \times 0 - 0]$$

Q – Learning Example (cont.)

- The Q table after the second episode (blue – updated in first episode)

states

actions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
↑	0	0	0	0	0	0	**-8**	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	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**	**8**	0	0	**0**	0	0	0	0	0	0	0	0

Q – Learning Example (cont.)

- The Q table after a few episode

states

actions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
↑	0	0	0	0	0	0	-8	-8	0	0	0	1	2	4	0	0	0	0	0	0
↓	0	0	0	0	0	0	0.5	1	0	0	0	-8	-8	-8	0	0	0	0	0	0
←	0	0	0	0	0	0	-8	1	0	0	0	-8	0.5	1	0	0	0	0	0	0
→	0	0	0	0	0	0	2	4	8	0	0	1	2	-8	0	0	0	0	0	0

Q – Learning Example (cont.)

- One of the optimal policies

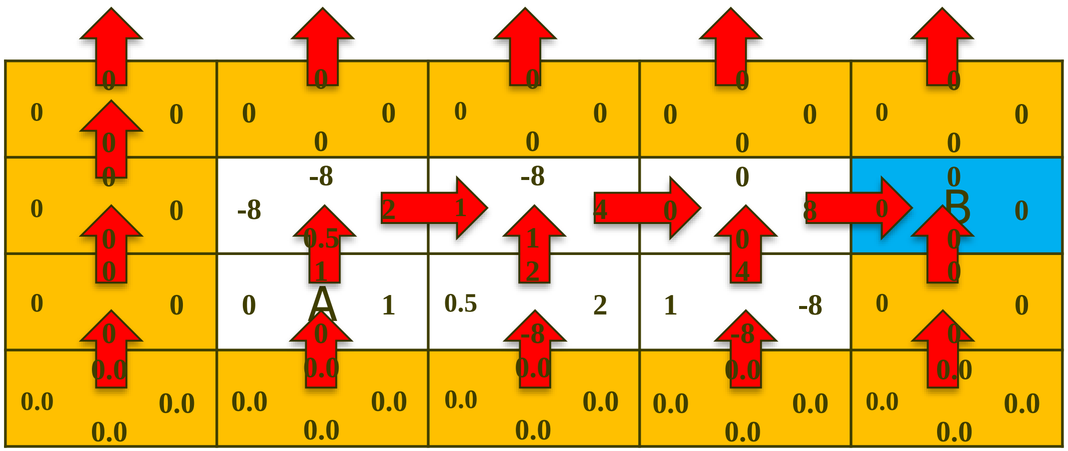
states

actions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
↑	0	0	0	0	0	0	-8	-8	0	0	0	1	2	4	0	0	0	0	0	0
↓	0	0	0	0	0	0	0.5	1	0	0	0	-8	-8	-8	0	0	0	0	0	0
←	0	0	0	0	0	0	-8	1	0	0	0	-8	0.5	1	0	0	0	0	0	0
→	0	0	0	0	0	0	2	4	8	0	0	1	2	-8	0	0	0	0	0	0

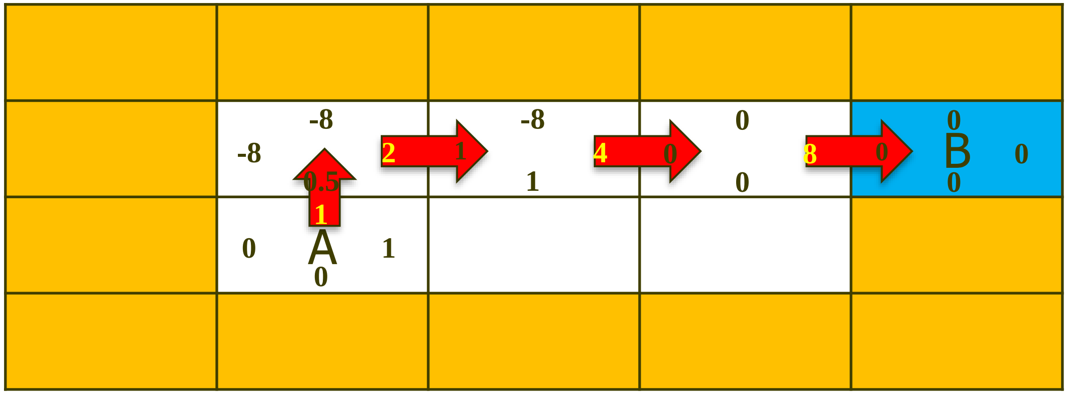
Q – Learning Example (cont.)

- An optimal policy graphically



Q – Learning Example (cont.)

- So we can find the this optimal way to go B



The Problem of Q – Learning

- Q – learning can require many thousands of training iterations to converge in even modest-sized problems.

- Very often, the memory resourced required by this method become too large.

Convergence of Q – Learning

- **Case1** : Deterministic MDP

- Condition for Convergence
  - The immediate reward values are bounded.

    $$|r(s,a)| < c \text{ for all } s,a$$

  - The agent selects actions in such a fashion that it visits every possible state-action pair infinitely often.

Convergence of Q – Learning (cont.)

- **Theorem 1**
  - $Q_n(s,a)$ converges to $Q^*(s,a)$ as $n \to \infty$, for all s, a.

Proof.

$$\Delta_n \equiv \max_{s,a} |Q_n(s,a) - Q^*(s,a)|$$

$$|Q_{n+1}(s,a) - Q^*(s,a)| = \left| \left(r + \gamma \max_{a'} Q_n(s',a')\right) - \left(r + \gamma \max_{a'} \hat{Q}(s',a')\right) \right|$$

$$= \gamma \left| \max_{a'} Q_n(s',a') - \max_{a'} \hat{Q}(s',a') \right|$$

$$\leq \gamma \max_{a'} \left| Q_n(s',a') - \hat{Q}(s',a') \right|$$

$$\leq \gamma \max_{s'',a'} \left| Q_n(s'',a') - \hat{Q}(s'',a') \right|$$

$$|Q_{n+1}(s,a) - Q^*(s,a)| \leq \gamma \Delta_n$$

$$|Q_{n+1}(s,a) - Q^*(s,a)| \leq \gamma^n \Delta_0$$

■

Convergence of Q – Learning (cont.)

- **Case2** : Nondeterministic MDP
- Condition for Convergence
  - Learning rate condition

$$\sum_{t=0}^{\infty} \alpha_t\,(s,a) = 0$$

$$\sum_{t=0}^{\infty} \alpha_t^2\,(s,a) < \infty$$

For all $(s,a) \in S \times A$

Convergence of Q – Learning (cont.)

- **Theorem 2**
  - $Q_n(s, a)$ converges to $Q^*(s, a)$ as $n \to \infty$, for all s, a.

Proof.

Omit. (Show in Appendix)

Summary

- Reinforcement learning addresses the problem of learning control strategies for autonomous agents.
- The reinforcement learning algorithms fit a problem setting known as a Markov decision process (MDP).
- Q learning is one form of reinforcement learning in which the agent learns an evaluation function over states and actions.
- Q learning can be proven to converge to the correct Q function under certain assumptions.
- Reinforcement learning is closely related to dynamic programming approaches to Markov decision processes.


Q
&
A