

Learning: Reinforcement Learning

(Based on Shoham and Leyton-Brown (2008). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge.)

Leen-Kiat Soh

How do we learn?

When goal is too far away or path is unclear, how do we learn to stay on target or learn to stay on the correct path?



Think about us. Think about equipping a robot/rover with the reasoning that will decide to do the right things towards its “goal”.



Especially in a complex environment with incomplete information, uncertainty, dynamic properties, etc.

How do we learn?

When goal is too far away or path is unclear, how do we learn to identify and stay on the **best** target or learn to identify and stay on the **best** path?



Think about us. Think about equipping a robot/rover with the reasoning that will decide to do the right things towards its “goal”.



Especially in a complex environment with incomplete information, uncertainty, dynamic properties, etc.

Introduction

- **Reinforcement learning does *not* explicitly model the opponent's strategy**
- The specific family of techniques we look at are derived from the **Q-learning algorithm** for learning in **unknown (single-agent) MDPs**



Q-learning: most popular and fundamental in agent-related learning, real-time, online



Single-agent MDPs: An agent does not care about what others are doing, or their probabilities of their mixed strategies, or their types/models ...



How can an agent afford to not model how other agents do things if they are in the same system or environment? (**Key: in the "same" system or environment**)



Markov Decision Process (MDP)

- A Markov Decision Process (MDP) is a discrete time *stochastic* control process
 - At each time step, given state s , the decision maker chooses action a that is available in state s
 - With a *state transition function*, $p(s, a, s')$, the process *probabilistically* transitions into a new state s'
 - This transition gives a *reward* for that state-action decision: $r(s, a, s')$.
- Given s and a , it is *conditionally independent* of all previous states and actions
 - i.e., the state transitions of an MDP meet the Markov property



MDP | Action Selection via Value Iteration

- **Goal:** To **maximize the total reward** over time
- **Strategy:** By assigning the **best possible action to each state**
- **Method:** **Value iteration** is the most popular algorithm, to find control policies
- It recursively calculates the utility of each action relative to a reward function

$$Q^{\pi^*}(s, a) = r(s, a, s') + \beta \sum_{s'} p(s, a, s') V^{\pi^*}(s')$$

Q-value (utility) of the best policy for the state-action pair of (s, a)

reward

Discount factor

Value of the best policy π^* for s

- Then it updates:

$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s, a)$$



MDP | Action Selection via Value Iteration 2

- The algorithm

$$Q_{t+1}(s, a) \leftarrow r(s, a, s') + \beta \sum_{\hat{s}} p(s, a, s') V_t(s')$$

$$V_t(s) \leftarrow \max_a Q_t(s, a)$$

- In a multiagent MDP, any (global) action a is really **a vector of local actions** (a_1, \dots, a_n) , one by each of n agents.

Notes: The future term is the *expected cumulative reward* of state s .



What if we don't know the transition probabilities?

Introduction 2

- Considering (single-agent) MDPs again: Value iteration assumes that the MDP is known
- **What if we do not know the rewards or transition probabilities of the MDP?**
- It turns out that, if we always know what state we are in and the reward received in each iteration, we can still converge to the correct Q-values using Q-learning
 - **Intuition:** We approximate the unknown transition probability by using the actual distribution of states **reached in the game** itself

Do humans do this?



Q-Learning

Initialize the Q-function and V values (arbitrarily, for example)

Repeat until convergence

Observe the current state s_t .

Carry out actions as
it learns ...

Select action a_t and take it.

Observe the reward $r(s_t, a_t)$ and next state s_{t+1}

Perform the following updates (and do *not* update any other Q-values):

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha_t)Q_t(s_t, a_t) + \alpha_t(r(s_t, a_t) + \beta V_t(s_{t+1}))$$

$$V_{t+1}(s) \leftarrow \max_a Q_t(s, a)$$

Value iteration

End Repeat

When is convergence?



Q-Learning 2



How good is the next state? What have I gotten myself into?
The future term, the look ahead term

Updated Q

Current Q

Current Reward

Value of the resulting state

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha_t)Q_t(s_t, a_t) + \alpha_t(r(s_t, a_t) + \beta V_t(s_{t+1}))$$

Learning Rate

Discount Factor

High learning rate vs. low learning rate: Implications?



Learning rate: why is it dependent on time?



Why discount factor? Uncertainty?



Implications with the future term? **Path-finding?**



Is Q-Learning intuitive to you?

We learn the best target or best path
not haphazardly

We use **past** experience, are adaptive to
current situations (e.g., rewards), look
ahead to the promise of **future**



Q-Learning 3

Updated value of a state

Maximum Q-value that an agent can get from acting on the state

$$V_{t+1}(s) \leftarrow \max_a Q_t(s, a)$$

V = sort of intrinsic value of a state, regardless of which action is carried out

Q(s,a) = inform us which a to select given s



- Knowledge accumulated in Qs
- is then used to update Vs
- which is then used to compute Qs in the next time step
- This iterates

Q-Learning and Optimality

Theorem 7.4.2 Q-learning guarantees that the Q and V values **converge to those of the optimal policy**, provided that *each state-action pair is sampled an infinite number of times*, and that the *time-dependent learning rate α_t obeys $0 \leq \alpha_t < 1$, $\sum_0^\infty \alpha_t = \infty$ and $\sum_0^\infty \alpha_t^2 < \infty$* .

- Infinite number of times to be **complete**
- α_t is smaller and smaller as t progresses
 - *Experienced \rightarrow refined \rightarrow no need to learn that quickly*



Think about us. If something is working well, we only tweak it to refine it, and don't change it much ...



... unless things change in the current situations (e.g., rewards)

Q-Learning Issues

- How to design the **order in which the algorithm selects actions**?
- What is the **rate of convergence**?
- It gives **no** assurance regarding the accumulation of optimal future discounted rewards by the agent
 - It could well be, depending on the discount factor, that **by the time the agent converges to the optimal policy it has paid too high a cost, which cannot be recouped by exploiting the policy going forward**
 - This is **not** a concern if the learning takes place during training sessions, and **only** when learning has converged sufficiently is the agent unleashed on the world
 - e.g., think of a fighter pilot being trained on a simulator before going into combat
- But in general Q-learning should be thought of as guaranteeing good learning, but **neither quick learning nor high future discounted rewards**

Revisiting Q-Learning

Initialize the Q-function and V values (arbitrarily, for example)

Repeat *until convergence*

Observe the current state s_t .

Select action a_t and take it.

Observe the reward $r(s_t, a_t)$ and next state s_{t+1}

Perform the following updates (and do *not* update any other Q-values):

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha_t)Q_t(s_t, a_t) + \alpha_t(r(s_t, a_t) + \beta V_t(s_{t+1}))$$
$$V_{t+1}(s) \leftarrow \max_a Q_t(s, a)$$

End Repeat

This does not tell us how to select actions?



How should we choose actions?
When do we reap the benefits of what we have learned?

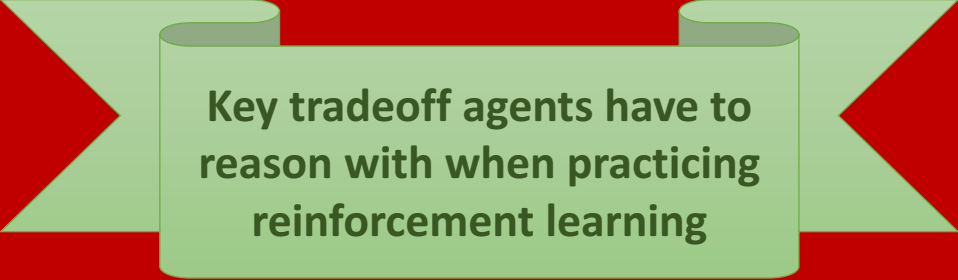


Should we explore and learn until convergence? Would that be too late?



What's the danger of reaping the benefits too early? (Think: **suboptimal** solutions)

Exploration vs. Exploitation



Key tradeoff agents have to reason with when practicing reinforcement learning

- When should one continue to explore (e.g., trying out all possible state-action combinations)?
 - Exploring for too long may not leave enough time for agent to recoup lost rewards—that were sacrificed when agent chooses less rewarding actions during exploration
- When should one start exploiting (e.g., choosing actions that give highest $Q(s,a)$)?
 - Exploiting too early may lead to sub-optimal solutions
 - And agent may never find the the optimal solutions
- One option: exploring offline—i.e., training; obtaining all $Q(s,a)$ values; finding the best paths; exploiting when deployed (with learning turned off)

Connection to MAS?

In a complex environment, agent autonomy in reasoning is necessary; we design them to have that reasoning power so that they can handle the “unforeseen”, especially when there are other agents present in the environment



Or, we learn offline (**training phase**), simulating and going through thousands and thousands of runs to obtain convergence on all $Q(s,a)$ values, and then deploy (**testing phase**) the agents with their “learning” mechanism turned off, allowing them to select actions with the best Q given a state




Silly Question: Suppose that in order to reduce costs (from manufacturing and installing road signs), your state no longer puts up road signs 2, 1.5, 1, 0.5, or 0.25 miles away from an exit. Instead, your state only puts up one road sign indicating the exit just right (0.25 miles) before the exit. What would happen? What would you learn in order to deal with this new change as a driver?



Belief-Based Reinforcement Learning

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha_t)Q_t(s_t, a_t) + \alpha_t(r(s_t, a_t) + \beta V_t(s_{t+1}))$$

$$V_t(s) \leftarrow \max_{a_i} \sum_{a_{-i} \in A_{-i}} Q_t(s, (a_i, a_{-i})) Pr_i(a_{-i})$$


- **Explicit modeling of the other agent(s)**
- The agent updates the value of the game using the **probability** it assigns to the opponent(s) playing each action profile
 - The belief function is updated after each play

Other Learning Behaviors

- **No-Regret Learning.** A learning rule is said to exhibit no regret if it guarantees that with high probability the agent will experience no positive regret: *no worse-off than the agent could have obtained by playing any one of its pure strategies throughout*
 - Risk averse
- **Targeted Learning.** An alternative sense of “good,” which retains the requirement of best response, but *limits it to a particular class of opponents*
 - Intuition: one has **some sense of the agents** in the environment
 - E.g., a chess player has studied previous plays of his opponent, and so on. And so it *makes sense to try to optimize against this set of opponents, rather than against completely unknown opponents*

Other Learning Behaviors 2

- **Difference between No-Regret Learning and Targeted Learning**
 - Consider learning in a repeated Prisoner's Dilemma game
 - Suppose that the target class consists of all opponents whose strategies rely on the past iteration (e.g., Tit-for-Tat)
 - Successful **targeted learning will result in constant cooperation**
 - while **no-regret learning prescribes constant defection**