

732A96/TDDE15 Advanced Machine Learning

Reinforcement Learning

Jose M. Peña
IDA, Linköping University, Sweden

Lectures 8: Q-Learning Algorithm

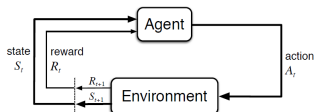
Contents

- ▶ Learning through Interaction
- ▶ Markov Decision Processes
- ▶ Bellman Equations
- ▶ Value Iteration
- ▶ Policy Iteration
- ▶ Q-Learning
- ▶ Example: Grid Worlds

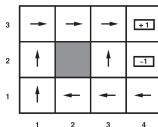
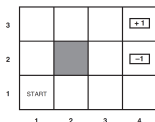
Literature

- ▶ Main source
 - ▶ Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, 2018. Chapters 1-7.
- ▶ Additional source
 - ▶ Russel, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Pearson, 2010. Chapters 16, 17 and 21.

Learning through Interaction



- ▶ **Agent:** The learner and decision maker.
- ▶ **Environment:** The agent interacts with it.
 - ▶ **State:** State of the agent and the environment.
 - ▶ **Action:** The agent decides the next action on the basis of the current state.
 - ▶ **Reward:** Numerical response to the action chosen by the agent. The agent aims to learn how to act so as to maximize the cumulative reward.
 - ▶ **Trajectory:** $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, S_3, \dots$
- ▶ **Policy:** Probability of doing an action in a state. The agent acts according to it. The agent aims to learn an optimal one.
- ▶ **Example:** A robot moves with probability 0.8 in the intended direction, and at the right angles of it otherwise. The reward for non-terminal states is $R(s) = -0.04$. All this is unknown to the robot. Optimal policies shown.



$R(s) < -1.6284$



$-0.4278 < R(s) < -0.0850$



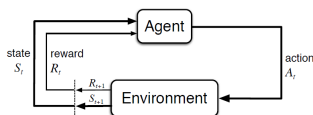
$-0.0221 < R(s) < 0$



$R(s) > 0$

- ▶ RL is neither supervised nor unsupervised learning.

Markov Decision Processes



- ▶ We assume that the agent-environment interaction follows a finite **Markov decision process**:
 - ▶ Finite sets of states, actions and rewards. Fully observable state.
 - ▶ Markovian and stationary transition model:

$$p(s_t, r_t | s_{0:t-1}, a_{0:t-1}, r_{1:t-1}) = p(s_t, r_t | s_{t-1}, a_{t-1}) = p(s', r | s, a).$$

- ▶ The **transition model** is typically unknown to the agent. Note the randomness of the next state and reward.
- ▶ The objective of the agent is to learn a policy $\pi(a|s)$ that maximizes the **expected discounted return**:

$$E_\pi[G_t] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] = E_\pi[R_{t+1} + \gamma G_{t+1}]$$

where $0 < \gamma \leq 1$ describes our preference between present and future rewards. Note the infinite horizon. However, the expectation is finite if $\gamma < 1$. For episodic tasks, $\gamma = 1$ and $R_{t+k+1} = 0$ for all $t + k + 1 > T$.

Markov Decision Processes

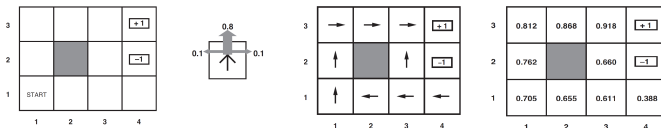
- ▶ The **state value** function $v_\pi(s)$ is the expected return of following policy π starting from state s :

$$v_\pi(s) = E_\pi[G_t | S_t = s] = \sum_a \pi(a|s) E_\pi[G_t | S_t = s, A_t = a] = \sum_a \pi(a|s) q_\pi(s, a).$$

- ▶ The **action value** function $q_\pi(s, a)$ is the expected return of doing action a in state s and then following policy π :

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] = E_\pi[R_{t+1} + \gamma G_{t+1} | s, a] = \sum_{s', r} p(s', r | s, a) (r + \gamma v_\pi(s')).$$

- ▶ Example: Environment, policy and state values.



- ▶ We can define the objective of the agent as learning a policy π_* such that

$$v_*(s) \geq v_\pi(s) \text{ for all } \pi, s.$$

For MDPs, there is always **at least one** such optimal policy.

Bellman Equations

- ▶ The state value function satisfies a recursive relationship known as **Bellman equation**:

$$v_{\pi}(s) = \sum_a \pi(a|s) q_{\pi}(s, a) = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) (r + \gamma v_{\pi}(s')).$$

- ▶ Moreover, v_{π} is the **unique solution** to the equations. Note that there are as many equations as unknowns. Since the equations are linear, they can be solved by linear algebra methods in $O(n^3)$. But this requires knowing the transition model.
- ▶ Likewise for the action value function:

$$q_{\pi}(s, a) = \sum_{s', r} p(s', r|s, a) (r + \gamma \sum_{a'} \pi(a'|s') q_{\pi}(s', a')).$$

- ▶ The Bellman equations of an **optimal policy** are

$$v_{*}(s) = \max_a \sum_{s', r} p(s', r|s, a) (r + \gamma v_{*}(s'))$$

and

$$q_{*}(s, a) = \sum_{s', r} p(s', r|s, a) (r + \gamma \max_{a'} q_{*}(s', a')).$$

- ▶ As before, v_{*} and q_{*} are the **unique solutions** to these equations. Note that the equations are now non-linear due to the max operator and, thus, harder to solve. Again, this requires knowing the transition model.

Bellman Equations

- ▶ Once we have v_* or q_* , it is easy to determine an **optimal policy**:

$$\pi_*(a|s) = \arg \max_a q_*(s, a)$$

or

$$\pi_*(a|s) = \arg \max_a \sum_{s', r} p(s', r|s, a)(r + \gamma v_*(s')).$$

- ▶ Note that the optimal policy is **deterministic**. So, we can consider only deterministic policies without loss of generality, i.e. $\pi(s)$ instead of $\pi(a|s)$.

Value Iteration

- ▶ We can avoid solving the Bellman equations for the state values of an optimal policy by turning them into update rules.

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

```
|  $\Delta \leftarrow 0$   
| Loop for each  $s \in \mathcal{S}$ :  
|    $v \leftarrow V(s)$   
|    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$   
|    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$   
until  $\Delta < \theta$ 
```

Output a deterministic policy, $\pi \approx \pi_*$, such that

$$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

- ▶ VI converges asymptotically for $\gamma < 1$. Since the Bellman optimality equations have a unique solution, VI converges to v_* .
- ▶ VI still requires knowing the transition model.

Policy Iteration

- Policy evaluation: Turn the ordinary Bellman equations into update rules.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement

policy-stable \leftarrow true

For each $s \in \mathcal{S}$:

old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

If *old-action* $\neq \pi(s)$, then *policy-stable* \leftarrow false

If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Policy Iteration

- ▶ Theorem: If $q_{\pi}(s, \pi'(s)) \geq q_{\pi}(s, \pi(s))$ for all s , then $v_{\pi'}(s) \geq v_{\pi}(s)$ for all s . Thus, we can modify π into a better policy π' by doing

$$\pi'(s) = \arg \max_a q_{\pi}(s, a) \text{ for all } s.$$

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement

policy-stable \leftarrow true

For each $s \in \mathcal{S}$:

old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow \arg \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$

If *old-action* $\neq \pi(s)$, then *policy-stable* \leftarrow false

If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Policy Iteration

- PI terminates since each iteration improves the policy and there is a finite number of policies. When PI halts, the Bellman optimality equations hold and, thus, π is optimal. Again, PI requires knowing the transition model.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement

policy-stable \leftarrow true

For each $s \in \mathcal{S}$:

old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

If *old-action* $\neq \pi(s)$, then *policy-stable* \leftarrow false

If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Q-Learning

- ▶ If the transition model were so that s and a are always followed by s' and r , then $q_*(s, a) = r + \gamma \max_{a'} q_*(s', a')$ by the Bellman optimality equation and, thus, $0 = r + \gamma \max_{a'} q_*(s', a') - q_*(s, a)$. We can try to enforce this constraint by executing π one step from s and a and, then, updating the estimate of $q_*(s, a)$ as

$$q_*(s, a) \leftarrow q_*(s, a) + \alpha(r + \gamma \max_{a'} q_*(s', a') - q_*(s, a)).$$

where $\alpha > 0$ is the learning rate.

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

 until S is terminal

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

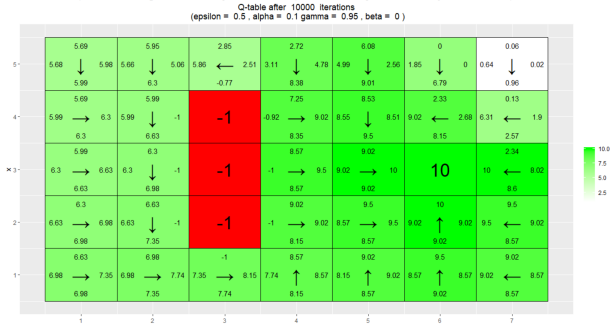
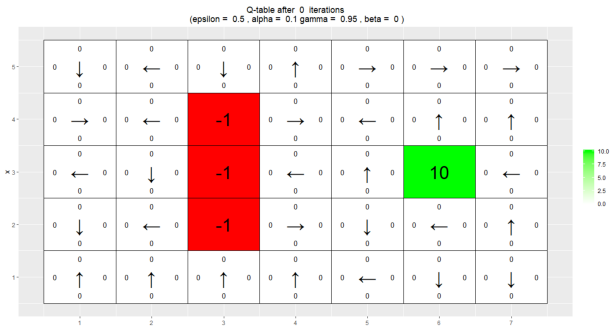
$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

 until S is terminal

- ▶ Q-learning converges asymptotically if e.g. $\alpha(t) = O(1/N(s, t))$.
- ▶ Q-learning converges asymptotically to π_* if e.g. an **ϵ -greedy** policy is used to keep updating all the state-action pairs: Choose the action with maximal estimated value with probability $1 - \epsilon$, and a random one with probability ϵ .
- ▶ Q-learning also works for stochastic transition models, since the number of times that s and a are followed by s' and r in the sampled episodes is proportional to the transition probability.
- ▶ Q-learning does **not** require knowing the transition model.

Example: Grid Worlds



Summary

- ▶ Learning through Interaction
- ▶ Markov Decision Processes
- ▶ Bellman Equations
- ▶ Value Iteration
- ▶ Policy Iteration
- ▶ Q-Learning
- ▶ Example: Grid Worlds
- ▶ Interested in more ? Check out [AlphaGo - The Movie](#).

Thank you