Reinforcement Learning

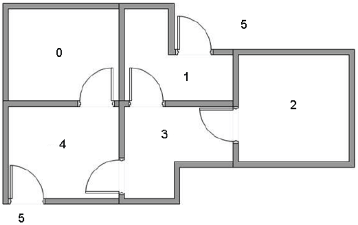
Wei Qi Yan

AUT, NZ

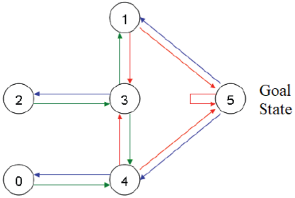
August 8, 2024

Reinforcement Learning

Example: Intelligent Navigation



Building Map



Graph

Table of Contents

Reinforcement Learning

Reinforcement Learning

- Agent and environment
- Action a, reward r
- *Policy* (π): State (s) and action $a \triangleq \pi(s)$
- *Samples*: $(s_1, a_1, r_1, \cdots, s_t)$, $t = 1, 2, \cdots$
- *Reinforcement learning*: $\max(r)$,
  s.t. $(s_1, a_1, r_1, \cdots, s_t) \to \pi$

MDP: Markov Decision Process

- A state s_t is *Markov* if $P(s_{t+1}|s_t) = P(s_{t+1}|s_1, \cdots, s_t)$.
- *Value function*: $v(s) \triangleq \mathbb{E}(G_t|s_t)$
- *Return*: $G_t \triangleq \sum_{k=0}^{\infty} \lambda^k \cdot r_{t+k+1}$, λ is a discount factor.

M. Littman (2015) Reinforcement learning improves behaviour from evaluative feedback. Nature, 521: 445-451.

Reinforcement Learning

Bellman Equation

$\because$ *Value function:* $v(s) = \mathbb{E}(G_t|s_t) = \mathbb{E}(\sum_{k=0}^{\infty} \lambda^k \cdot r_{t+k+1}|s_t)$
$\therefore v(s) = \mathbb{E}(r_{t+1} + \lambda \cdot G_{t+1}|s_t)$; $v(s) = \mathbb{E}(r_{t+1} + \lambda \cdot v(s_{t+1})|s_t)$
$\therefore$ *Action-value function:*

$$Q^{\pi}(s,a) \triangleq \mathbb{E}_{s'}(r + \lambda \cdot Q^{\pi}(s',a')|s,a)$$

Optimal action-value function:

$$Q^{*}(s,a) = \mathbb{E}_{s'}(r + \lambda \cdot \max_{a'} Q^{*}(s',a')|s,a)$$

Iteratively,

$$Q_{i+1}(s,a) = \mathbb{E}_{s'}(r + \lambda \cdot \max_{a'} Q_i(s',a')|s,a) \rightarrow Q^{*}, \text{ if } i \rightarrow \infty$$

MDP: Markov Decision Process

- A state s_t is Markov if $P(s_{t+1}|s_t) = P(s_{t+1}|s_1, \cdots, s_t)$.
- Value function: $v(s) \triangleq \mathbb{E}(G_t|s_t)$
- Return: $G_t \triangleq \sum_{k=0}^{\infty} \lambda^k \cdot r_{t+k+1}$, λ is discount factor.

M. Littman (2015) Reinforcement learning improves behaviour from evaluative feedback. Nature, 521: 445-451.

Reinforcement Learning

Bellman Equation

$\because$ Value function: $v(s) = \mathbb{E}(G_t|s_t) = \mathbb{E}(\sum_{k=0}^{\infty} \lambda^k \cdot r_{t+k+1}|s_t)$
$\therefore$ $v(s) = \mathbb{E}(r_{t+1} + \lambda \cdot G_{t+1}|s_t)$; $v(s) = \mathbb{E}(r_{t+1} + \lambda \cdot v(s_{t+1})|s_t)$
$\therefore$ Action-value function:

$$Q^\pi(s,a) \triangleq \mathbb{E}_{s'}(r + \lambda \cdot Q^\pi(s',a')|s,a)$$

The optimal action-value function:

$$Q^*(s,a) = \mathbb{E}_{s'}(r + \lambda \cdot \max_{a'} Q^*(s',a')|s,a)$$

Iteratively,

$$Q_{i+1}(s,a) = \mathbb{E}_{s'}(r + \lambda \cdot \max_{a'} Q_i(s',a')|s,a) \rightarrow Q^*, \text{ if } i \rightarrow \infty$$

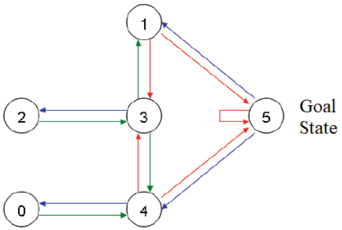
Q-Learning

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_{t+1} + \lambda \cdot \max_a Q(s_{t+1},a) - Q(s_t,a_t))$$

M. Littman (2015) Reinforcement learning improves behaviour from evaluative feedback. Nature, 521: 445-451.

Reinforcement Learning

Deep Q-Learning



Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \lambda \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

M. Littman (2015) Reinforcement learning improves behavior from evaluative feedback. Nature, 521: 445-451.

Deep Q-Learning

- For a deep network *w.r.t.* Q and weight w,

$$Q(s, a, w) = Q^{\pi}(s, a).$$

- The loss/objective function is,

$$L(w) \triangleq \mathbb{E}([r + \gamma \cdot \max_{a'} Q(s', a', w) - Q(s, a, w)]^2)$$

- The gradient is,

$$\frac{\partial L(w)}{\partial w} = \mathbb{E}([r + \gamma \cdot \max_{a'} Q(s', a', w) - Q(s, a, w)]) \cdot \frac{\partial Q(s, a, w)}{\partial w}$$

V.Mnih, et al. (2015) Human-level control through deep reinforcement learning. Nature, 518: 529-533.

Questions?

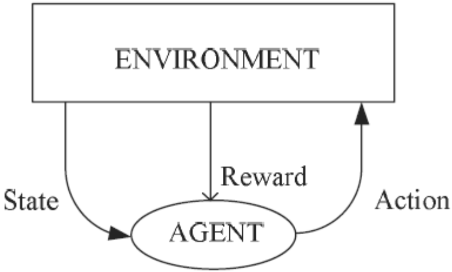
Policy and Value Iterations

- Reinforcement learning is learning what to do - how to map situations to actions — so as to *maximize a numerical reward.*
- Reinforcement learning receives *reward/penalty* or *trial/error* for its actions solve a problem.
- Reinforcement learning could *learn the best policy* and *maximize the total reward.*
- The *sequence of actions* has the maximum cumulative reward.
- For each policy π, there is a reward $v^{\pi}(s_t)$, we hope to find the *optimal policy*,

$$v^{*}(s_t) = \max_{\pi}(v^{\pi}(s_t)), \forall s_t$$

E. Alpaydn (2010) Introduction to Machine Learning (2nd Eition). The MIT Press.

Policy and Value Iterations



E. Alpaydn (2010) Introduction to Machine Learning (2nd Eition). The MIT Press.

Policy and Value Iterations

In a simple case, action $a(t) \triangleq \pi(s_t)$, $Q(a_t) \triangleq r(a_t) > 0$. If $r(a)$ is the reward,

$$Q(a_{t+1}) \leftarrow Q_t(a) + \eta \cdot [r(a_{t+1}) - Q(a_t)]$$

In a *full* reinforcement learning, the value of state s_t satisfies,

$$v(s_t) = \max_{a_t} Q(s_t, a_t);$$

where

$$a_t^* = \arg\max_{a_t} Q(s_t, a_t); \pi^*(s_t^*) = a_t^*;$$

E. Alpaydn (2010) Introduction to Machine Learning (2nd Eition). The MIT Press.

Policy and Value Iterations

Value iteration:

$$v(s_t) \leftarrow v(s_t) + \eta \cdot [r_{t+1} + \gamma \cdot v(s_{t+1}) - v(s_t)]$$

Termination condition:

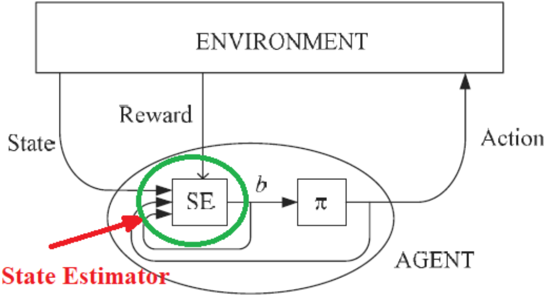
$$|v^{(l+1)}(s) - v^{(l)}(s)| < \delta; \delta > 0; l = 1, 2, 3, \cdots$$

Policy iteration:

$$\pi \leftarrow \pi' = \arg\max_{\pi}(v^{\pi}(s')); v^{\pi}(s) \leftarrow v^{\pi}(s');$$

V. Mnih, et al. (2016) Asynchronous Methods for Deep Reinforcement Learning. International Conference on Machine Learning, pp.1928-1937.

Policy and Value Iterations



E. Alpaydn (2010) Introduction to Machine Learning (2nd Eition). The MIT Press.

Monte-Carlo Method

- **Episode**: $\exists T$, $(s_1, a_1, r_2, \cdots, s_T) \to \pi$
- **Monte-Carlo Method**: Using empirical mean to replace Bellman equation instead of expected return, i.e.,

$$v_\pi(s) = \frac{1}{T}\sum_{t=1}^{T}(G_t | s_t = s)$$

where $G_t = \sum_{k=1}^{T-t} \lambda^{k-1} r_{t+k}$. Hence,

$$v(s_t) \leftarrow v(s_t) - \alpha \cdot (G_t - v(s_t))$$

M. Littman (2015) Reinforcement learning improves behavior from evaluative feedback. Nature, 521: 445-451.

TD: Temporal Difference Method

- **Episode**: $\exists T$, $(s_1, a_1, r_2, \cdots, s_T) \to \pi$
- **Monte-Carlo Method**:

$$v(s_t) \leftarrow v(s_t) - \alpha \cdot (G_t - v(s_t))$$

- **Temporal Difference (TD)**:

$$v(s_t) \leftarrow v(s_t) - \alpha \cdot (r_{t+1} + \gamma \cdot v(s_{t+1}) - v(s_t))$$

  where $\delta = r_{t+1} + \gamma \cdot v(s_{t+1}) - v(s_t)$ is named as the *TD error*, $r_{t+1} + \gamma \cdot v(s_{t+1})$ is called the *TD target*.
- **Dynamic Programming Method**:

$$v(s_t) = \mathbb{E}_\pi(r_{t+1} + \gamma \cdot v(s_{t+1}))$$

M. Littman (2015) Reinforcement learning improves behavior from evaluative feedback. Nature, 521: 445-451.

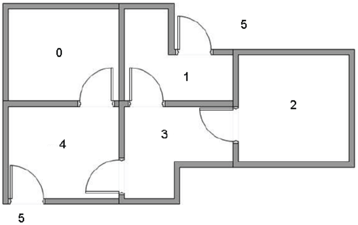
Double Q-Learning

- **Episode**: $\exists T,\ (s_1, a_1, r_2, \cdots, s_T) \to \pi$
- **SARSA (State-Action-Reward-State-Action)** :
  $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot Q(s', a') - Q(s,a)]$
  $s \leftarrow s'$, $a \leftarrow a'$.
- **Q-Learning** (an *off-policy* TD control algorithm):
  $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_a Q(s', a) - Q(s,a)]$,
  $s \leftarrow s'$, $a \leftarrow a'$.
- **Double Q-Learning:**
  $Q_1(s,a) \leftarrow$
  $Q_1(s,a) + \alpha \cdot [r + \gamma \cdot Q_2(s', \arg\max_a Q_1(s', a)) - Q_1(s,a)]$,
  $Q_2(s,a) \leftarrow$
  $Q_2(s,a) + \alpha \cdot [r + \gamma \cdot Q_1(s', \arg\max_a Q_2(s', a)) - Q_2(s,a)]$

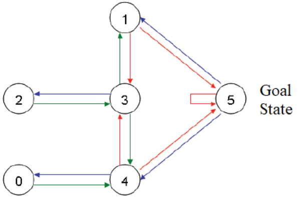
M. Littman (2015) Reinforcement learning improves behavior from evaluative feedback. Nature, 521: 445-451.

Questions?

Example: Intelligent Navigation



Building Map



Graph

Reinforcement Learning

Example: Intelligent Navigation

$$
R = \begin{array}{c c}
 & \text{Action} \\
\text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
\end{array}
\qquad
Q = \begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Step 0

Example: Intelligent Navigation

$$Q(s,a) = r(s,a) + \lambda \cdot \max_{a'} Q(s',a')$$

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c|cccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Step 0

Reinforcement Learning

Example: Intelligent Navigation

$$Q(1,5) = R(1,5) + \lambda \cdot \max(Q(5,5), Q(5,4), Q(5,1)) = 100.$$

$$Q(3,1) = R(3,1) + \lambda \cdot \max(Q(1,3), Q(1,5)) = 80, \lambda = 0.8.$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Step 1

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Step 2

Example: Intelligent Navigation

$$Q = \begin{array}{c|c} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

Step 6x6

$$Q = \begin{array}{c|c} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

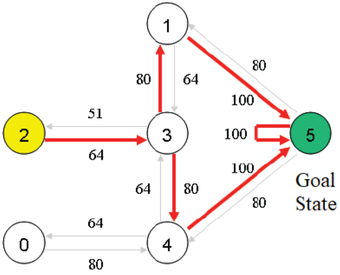
Step 6x6+1

Reinforcement Learning

Example: Intelligent Navigation

$$
Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 80 & 0 \\
0 & 0 & 0 & 64 & 0 & 100 \\
0 & 0 & 0 & 64 & 0 & 0 \\
0 & 80 & 51 & 0 & 80 & 0 \\
64 & 0 & 0 & 64 & 0 & 100 \\
0 & 80 & 0 & 0 & 80 & 100
\end{bmatrix}
\end{array}
$$

Step 36+1

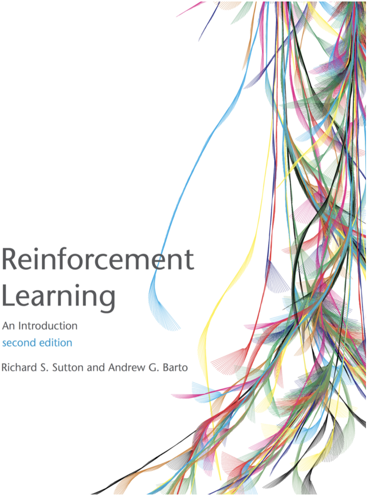


Deep Q-Learning: Implementation

```python
        start_room = index

        print("##################################",

        current_state = start_room

        step = 0

        target_state = 5

        while current_state != target_state:
            out_result = self.session.run(self.q_eval, fee
                self.q_eval_input: self.state_list[current_s

            next_state = np.argmax(out_result[0])

            print("Agent currernt state:", current_state, "

            current_state = next_state

            step += 1

        print("Agent started from room:", start_room, "

        print("##################################",


if __name__ == "__main__":
    q_network = DeepQNetwork()
    q_network.pay()
```

```
 [-1.  0. -1. -1.  0. 100.]]
################################## Agent in room:  0 starts moving ########
Agent currernt state: 0  to next state: 4
Agent currernt state: 4  to next state: 5
Agent started from room: 0  after:  2 steps arrive room 5
################################## Agent at room: 5 Agent is at Room 5 ###
####
################################## Agent in room:  1 starts moving ########
Agent currernt state: 1  to next state: 5
Agent started from room: 1  after:  1 steps arrive room 5
################################## Agent at room: 5 Agent is at Room 5 ###
####
################################## Agent in room:  2 starts moving ########
Agent currernt state: 2  to next state: 3
Agent currernt state: 3  to next state: 4
Agent currernt state: 4  to next state: 5
Agent started from room: 2  after:  3 steps arrive room 5
################################## Agent at room: 5 Agent is at Room 5 ###
####
################################## Agent in room:  3 starts moving ########
Agent currernt state: 3  to next state: 4
Agent currernt state: 4  to next state: 5
Agent started from room: 3  after:  2 steps arrive room 5
################################## Agent at room: 5 Agent is at Room 5 ###
####
################################## Agent in room:  4 starts moving ########
Agent currernt state: 4  to next state: 5
Agent started from room: 4  after:  1 steps arrive room 5
################################## Agent at room: 5 Agent is at Room 5 ###
####
>>>
```

Questions?


Reinforcement
Learning
An Introduction
second edition
Richard S. Sutton and Andrew G. Barto