Q-Learning

Christopher Watkins and Peter Dayan

Noga Zaslavsky

The Hebrew University of Jerusalem
Advanced Seminar in Deep Learning (67679)

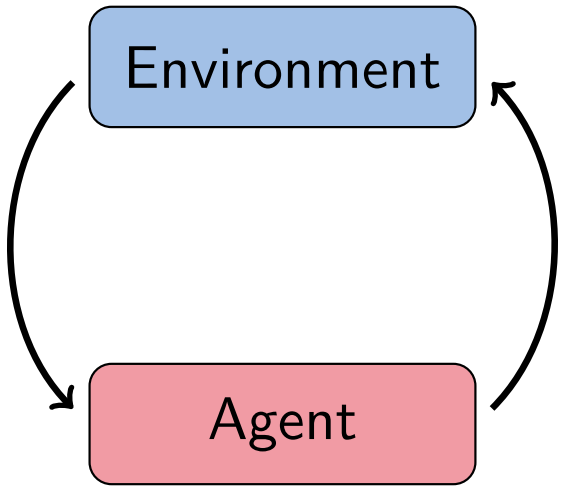
November 1, 2015

Outline

Outline

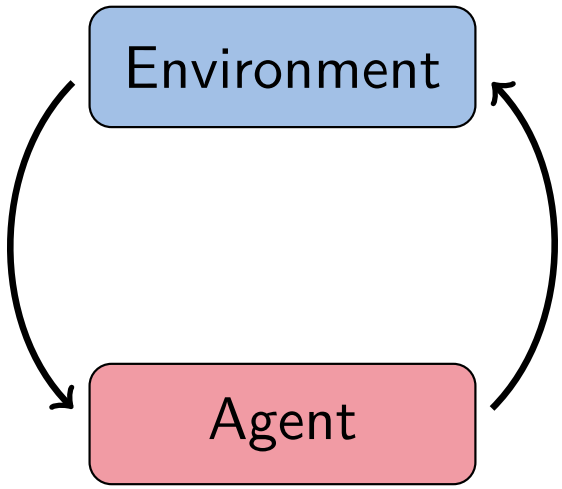
Reinforcement Learning: Agent-Environment Interaction

- Reinforcement learning is learning how to behave in order to maximize value or achieve some goal



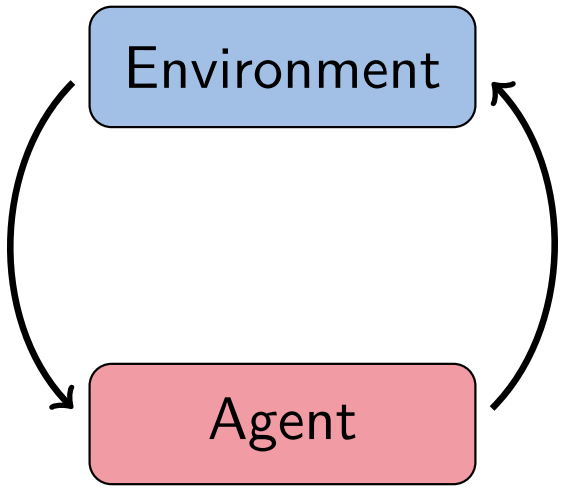
Reinforcement Learning: Agent-Environment Interaction

- Reinforcement learning is learning how to behave in order to maximize value or achieve some goal
- It is not supervised learning, and it is not unsupervised learning
  - There is no expert to tell the learning agent right from wrong - it is forced to learn from its own experience
  - The learning agent receives feedback from its environment - it can learn by trying different actions at different situations

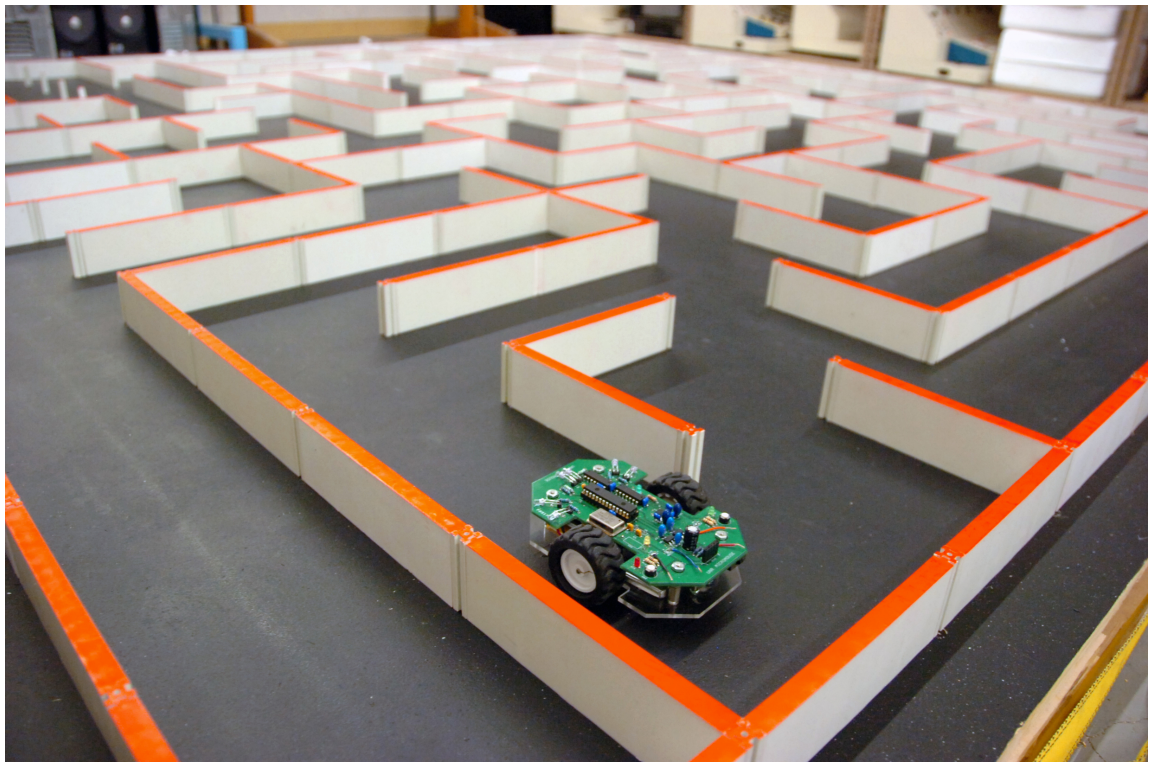


Reinforcement Learning: Agent-Environment Interaction

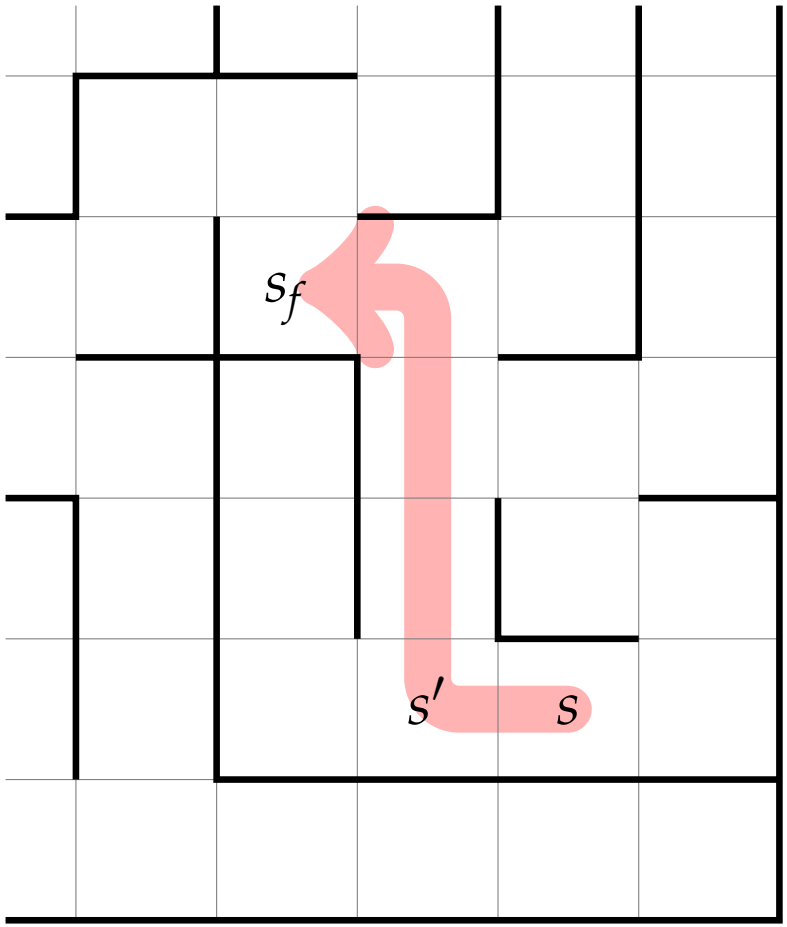
- Reinforcement learning is learning how to behave in order to maximize value or achieve some goal
- It is not supervised learning, and it is not unsupervised learning
  - There is no expert to tell the learning agent right from wrong - it is forced to learn from its own experience
  - The learning agent receives feedback from its environment - it can learn by trying different actions at different situations
- There is a tradeoff between exploration and exploitation



Reinforcement Learning: Agent-Environment Interaction



Toy Problem



Markov Decision Process (MDP)

- $S \in \mathcal{S}$ - the state of the system

Markov Decision Process (MDP)

- $S \in \mathcal{S}$ - the state of the system
- $A \in \mathcal{A}$ - the agent's action

Markov Decision Process (MDP)

- $S \in \mathcal{S}$ - the state of the system
- $A \in \mathcal{A}$ - the agent's action
- $p\left(s'|s,a\right)$- the dynamics of the system

Markov Decision Process (MDP)

- $S \in \mathcal{S}$ - the state of the system
- $A \in \mathcal{A}$ - the agent's action
- $p\left(s'|s,a\right)$- the dynamics of the system
- $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ - the reward function (possibly stochastic)

Definition

MDP is the tuple $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$

Markov Decision Process (MDP)

- $S \in \mathcal{S}$ - the state of the system
- $A \in \mathcal{A}$ - the agent's action
- $p\left(s'|s,a\right)$- the dynamics of the system
- $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ - the reward function (possibly stochastic)

Definition

MDP is the tuple $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$

- $\pi\left(a|s\right)$ - the agent's policy

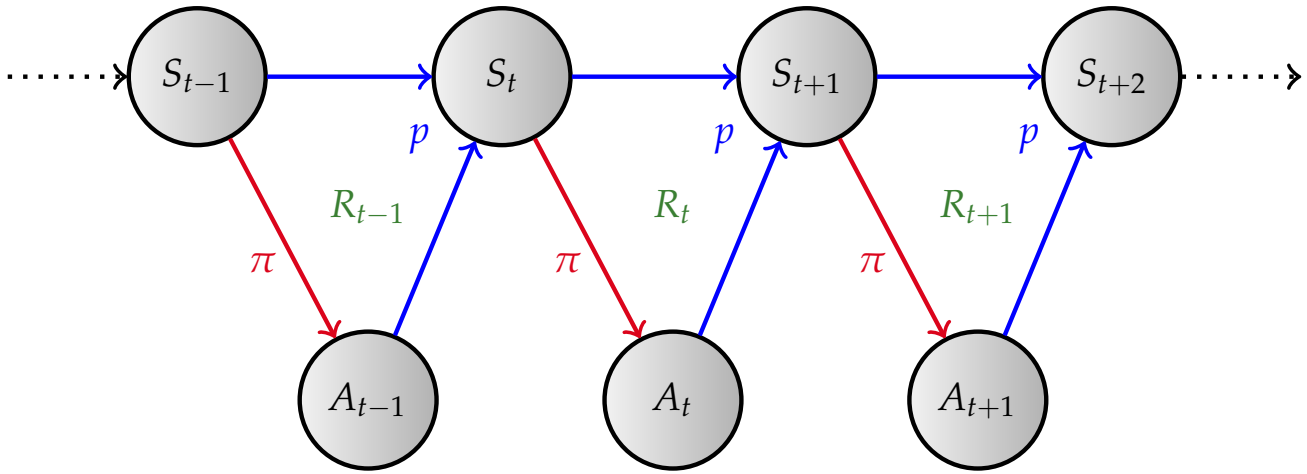
Markov Decision Process (MDP)

- $S \in \mathcal{S}$ - the state of the system
- $A \in \mathcal{A}$ - the agent's action
- $p\left(s'|s,a\right)$- the dynamics of the system
- $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ - the reward function (possibly stochastic)

Definition

MDP is the tuple $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$

- $\pi\left(a|s\right)$ - the agent's policy



The Value Function

- The ***un-discounted value function*** for any given policy π

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t) \middle| s_0 = s\right]$$

T is the ***horizon*** - can be finite, episodic, or infinite

The Value Function

- The ***un-discounted value function*** for any given policy π

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t) \middle| s_0 = s\right]$$

  T is the ***horizon*** - can be finite, episodic, or infinite

- The ***discounted value function*** for any given policy π

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right]$$

  $0 < \gamma < 1$ is the ***discounting factor***

The Value Function

- The ***un-discounted value function*** for any given policy π

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t) \middle| s_0 = s\right]$$

T is the ***horizon*** - can be finite, episodic, or infinite

- The ***discounted value function*** for any given policy π

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right]$$

$0 < \gamma < 1$ is the ***discounting factor***

The goal: find a policy π^* that maximizes the value $\forall s \in \mathcal{S}$

$$V^*(s) = V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s)$$

The Bellman Equation

- The value function can be written recursively

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right] = \mathop{\mathbb{E}}_{\substack{a \sim \pi(\cdot|s) \\ s' \sim p(\cdot|s,a)}} \left[r(s,a) + \gamma V^{\pi}(s')\right]$$

The Bellman Equation

- The value function can be written recursively

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right] = \mathop{\mathbb{E}}_{\substack{a \sim \pi(\cdot|s) \\ s' \sim p(\cdot|s,a)}} \left[r(s,a) + \gamma V^{\pi}(s')\right]$$

- The optimal value satisfies the Bellman equation

$$V^{*}(s) = \max_{\pi} \mathop{\mathbb{E}}_{\substack{a \sim \pi(\cdot|s) \\ s' \sim p(\cdot|s,a)}} \left[r(s,a) + \gamma V^{*}(s')\right]$$

The Bellman Equation

- The value function can be written recursively

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right] = \mathop{\mathbb{E}}_{\substack{a \sim \pi(\cdot|s) \\ s' \sim p(\cdot|s,a)}} \left[r(s,a) + \gamma V^{\pi}(s')\right]$$

- The optimal value satisfies the Bellman equation

$$V^{*}(s) = \max_{\pi} \mathop{\mathbb{E}}_{\substack{a \sim \pi(\cdot|s) \\ s' \sim p(\cdot|s,a)}} \left[r(s,a) + \gamma V^{*}(s')\right]$$

- π^* is not necessarily unique, but V^* is unique

The Q-Function

- The value of taking action a at state s:

$$Q(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)} \left[V(s') \right]$$

The Q-Function

- The value of taking action a at state s:

$$Q(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)} \left[V(s') \right]$$

- If we know V^* then an optimal policy is to decide deterministically

$$a^*(s) = \arg\max_a Q^*(s,a)$$

The Q-Function

- The value of taking action a at state s:

$$Q(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)} \left[V(s') \right]$$

- If we know V^* then an optimal policy is to decide deterministically

$$a^*(s) = \arg\max_a Q^*(s,a)$$

- This means that $V^*(s) = \max_a Q^*(s,a)$

The Q-Function

- The value of taking action a at state s:

$$Q(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)} \left[V(s') \right]$$

- If we know V^* then an optimal policy is to decide deterministically

$$a^*(s) = \arg\max_a Q^*(s,a)$$

- This means that $V^*(s) = \max_a Q^*(s,a)$

Conclusion

Learning Q^* makes it easy to obtain an optimal policy

Outline

Learning Q^*

- If the agent knows the dynamics p and the reward function r, it can find Q^* by dynamic programing (e.g. value iteration)

Learning Q^*

- If the agent knows the dynamics p and the reward function r, it can find Q^* by dynamic programing (e.g. value iteration)

- Otherwise, it needs to estimate Q^* from its experience
  - The ***experience*** of an agent is a sequence $s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$
  - The n-th ***episode*** is (s_n, a_n, r_n, s_{n+1})

Learning Q^*

Q-Learning

Initialize $Q_0(s,a)$ for all $a \in \mathcal{A}$ and $s \in \mathcal{S}$

For each episode n

observe the current state s_n

select and execute an action a_n

observe the next state s_{n+1} and reward r_n

$$Q_n(s_n,a_n) \leftarrow (1-\alpha_n) Q_{n-1}(s_n,a_n) + \alpha_n \left(r_n + \gamma \max_a Q_{n-1}(s_{n+1},a)\right)$$

$\alpha_n \in (0,1)$ - the learning rate

Exploration vs. Exploitation

- How should the agent gain experience in order to learn?

Exploration vs. Exploitation

- How should the agent gain experience in order to learn?
  - If it explores too much it might suffer from a low value

Exploration vs. Exploitation

- How should the agent gain experience in order to learn?
  - If it explores too much it might suffer from a low value
  - If it exploits the learned Q function too early, it might get stuck at bad states without even knowing about better possibilities (overfitting)

Exploration vs. Exploitation

- How should the agent gain experience in order to learn?
  - If it explores too much it might suffer from a low value
  - If it exploits the learned Q function too early, it might get stuck at bad states without even knowing about better possibilities (overfitting)

- One common approach is to use an ϵ-greedy policy

Convergence of the Q function

Theorem

Given bounded rewards $|r_n| \leq \mathcal{R}$, *and learning rates* $0 \leq \alpha_n \leq 1$ *s.t.*

$$\sum_{n=1}^{\infty} \alpha_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

then with probability 1

$$Q_n(s,a) \underset{n \to \infty}{\longrightarrow} Q^*(s,a)$$

The exploration policy should be such that each state-action pair will be encountered infinitely many times

Convergence of the Q function

Poof - main idea

- Define the operator

$$\mathbf{B}\left[Q\right]_{s,a} = r\left(s,a\right) + \gamma \sum_{s'} p\left(s'|s,a\right) \max_{a'} Q\left(s',a'\right)$$

- $\mathbf{B}$ is a contraction under the $\|\cdot\|_\infty$ norm

$$\|\mathbf{B}\left[Q_1\right] - \mathbf{B}\left[Q_2\right]\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty$$

- It is easy to see that $\mathbf{B}\left[Q^*\right] = Q^*$
- We're not done because the updates come from a stochastic process

Outline

What assumptions did we make?

- Q-learning is model free
- The state and reward are assumed to be fully observable
  - POMDP is a much harder problem
- The Q function can be represented by a look-up table
  - otherwise we need to do something else such as function approximation, and this is where deep learning comes in!

Outline

Summary

- The basic reinforcement learning problem can be modeled as an MDP
- If the model is known we can solve precisely by dynamic programing
- Q-learning allows learning an optimal policy when the model is not known, and without trying to learn the model
- It converges asymptotically to the optimal Q if the learning rate is not too fast and not too slow, and if the exploration policy is satisfactory

Further Reading

Richard S. Sutton and Andrew G. Barto
Reinforcement Learning: An Introduction.
MIT Press, 1998