Reinforcement Learning Tutorial

CAMP 2016, Bangalore

Martin Angelhuber

Reinforcement Learning

Unsupervised Learning

Supervised Learning

Reinforcement Learning

Reinforcement Learning

Unsupervised Learning	→ No feedback at all
Supervised Learning	→ Teaching signal: right solution is provided
Reinforcement Learning	→ Reward signal: scalar value

Reinforcement Learning

Unsupervised Learning

→ No feedback at all

Supervised Learning

→ Teaching signal:
right solution is provided

Reinforcement Learning

→ Reward signal:
scalar value

→ Reward is typically delayed!

Reinforcement Learning

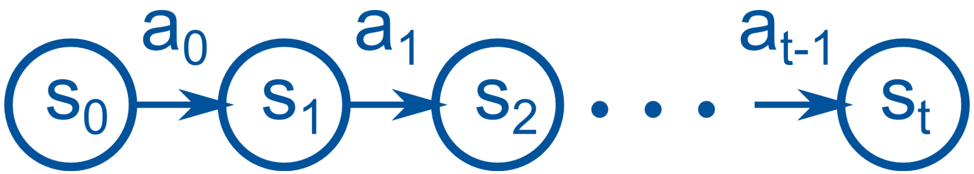
Unsupervised Learning	→ No feedback at all
Supervised Learning	→ Teaching signal: right solution is provided
Reinforcement Learning	→ Reward signal: scalar value

→ Reward is typically delayed!

→ Agent has to perform a sequence of actions before reward arrives

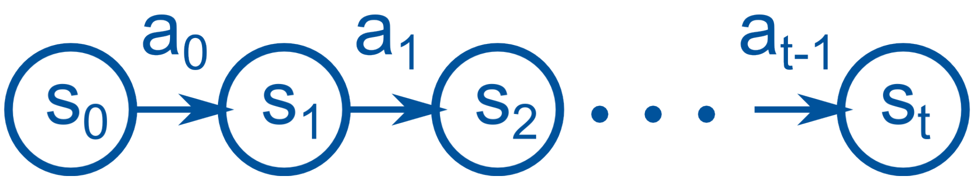
Markov Decision Process

- Sequential decision task:



Markov Decision Process

- Sequential decision task:



Discrete set of states s and actions a

Transition model: $P(s'|s,a)$

Reward function $R(s)$: $s \rightarrow \text{reward}$

Markov Decision Process

- Sequential decision task:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \cdots \xrightarrow{a_{t-1}} s_t$$

Discrete set of states s and actions a

Transition model: $P(s'|s,a)$

Reward function $R(s)$: $s \longrightarrow \text{reward}$

Policy: $\pi(s)$: $s \longrightarrow a$

→ Maximize the expected future reward

$$E[R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\ldots+\gamma^t R(s_t)] \qquad 0 \leqslant \gamma \leqslant 1$$

Markov Decision Process

- Sequential decision task:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \cdots \xrightarrow{a_{t-1}} s_t$$

Discrete set of states s and actions a

Transition model: $P(s'|s,a)$

Reward function $R(s)$: $s \longrightarrow \text{reward}$

} Environment

Policy: $\pi(s)$: $s \longrightarrow a$

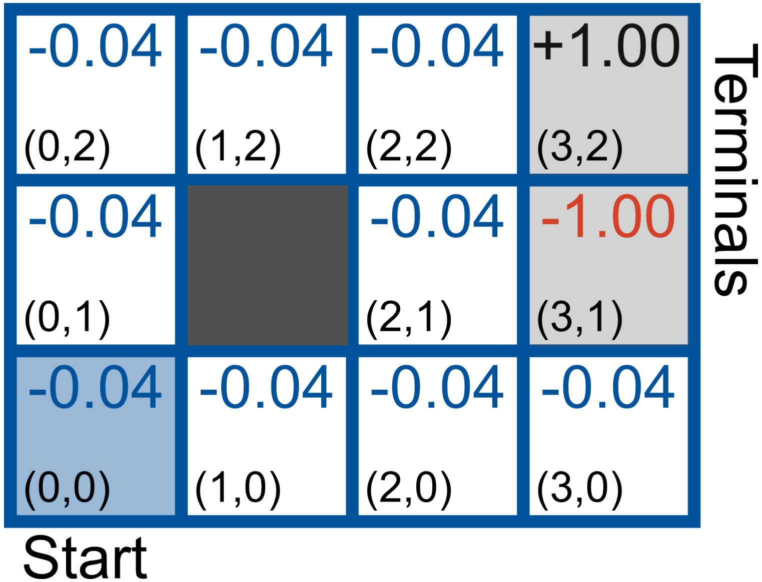
→ Maximize the expected future reward

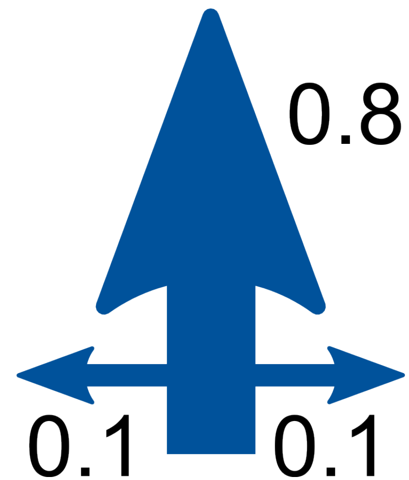
$$E\left[R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\ldots+\gamma^t R(s_t)\right]$$

} Agent

Example: Grid World

- Navigating a simple grid:





Actions (up, down, left, right) are random!

If agent moves against wall, he stays put

Value functions

- Introduce value function:

$$V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^t R(s_t) | s_0 = s, \pi]$$

expected future reward, when in state s and following policy π

Value functions

- Introduce value function:

$$V^{\pi}(s)=E[R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\ldots+\gamma^t R(s_t)|s_0=s,\pi]$$

expected future reward, when in state s and following policy π

For now, assume the transition model $P(s'|s,a)$ and reward function $R(s)$ are known to the agent!

Value functions

- Introduce value function:

$$V^{\pi}(s)=E[R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\ldots+\gamma^t R(s_t)|s_0=s,\pi]$$

expected future reward, when in state s and following policy π

- Recursive form (for each state s)

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

Value functions

- Introduce value function:

$$V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^t R(s_t) | s_0 = s, \pi]$$

expected future reward, when in state s and following policy π

- Recursive form (for each state s)

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

→ can be solved iteratively:

Repeat for all s: $V_{k+1}^{\pi}(s) := R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) V_k^{\pi}(s')$

Value functions

- Introduce value function:

$$V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^t R(s_t) | s_0 = s, \pi]$$

expected future reward, when in state s and following policy π

- Recursive form (for each state s)

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

→ can be solved iteratively:

Repeat for all s: $V^{\pi}_{k+1}(s) := R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) V^{\pi}_{k}(s')$

- Optimal value function: $V^*(s) = \max_{\pi} V^{\pi}(s)$

$$V^*(s) = R(s) + \max_a \gamma \sum_{s'} P(s' | s, a) V^*(s')$$

Bellman equation

Value and policy iteration

- Two ways of solving the Bellman equation iteratively:

Value iteration:

1) Initialize: $V(s) := 0, \forall s$

2) Repeat until convergence:

$$V(s) := R(s) + \max_a \gamma \sum_{s'} P(s' | s, a) V(s'), \forall s$$

Value and policy iteration

- Two ways of solving the Bellman equation iteratively:

Value iteration:

1) Initialize: $V(s) := 0, \forall s$

2) Repeat until convergence:

$$V(s) := R(s) + \max_a \gamma \sum_{s'} P(s' | s, a) V(s'), \forall s$$

Policy iteration:

1) Start with random policy

2) Repeat until convergence:

a) Compute value function: $V(s) := V(s)^{\pi}, \forall s$

b) Update policy:

$$\pi(s) := \arg\max_a \sum_{s'} P(s' | s, a) V(s'), \forall s$$

Assignment 1:

1) Familiarize yourself with the code for the grid world.

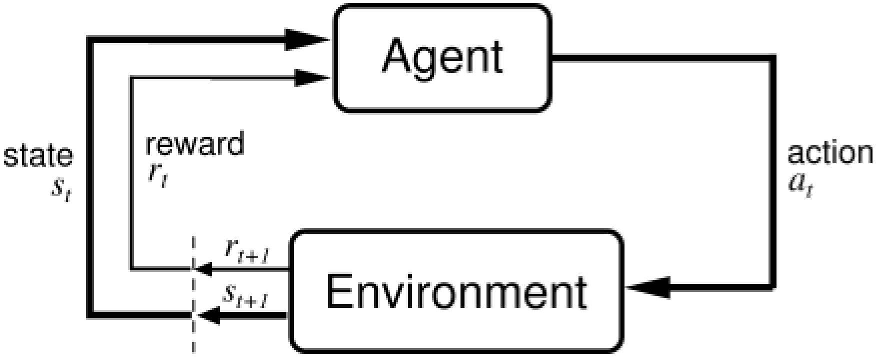
2) Complete the function policy_evaluation to compute the value function for a given policy!

3) Complete the code to implement value iteration.

4) Complete the code to implement policy iteration.

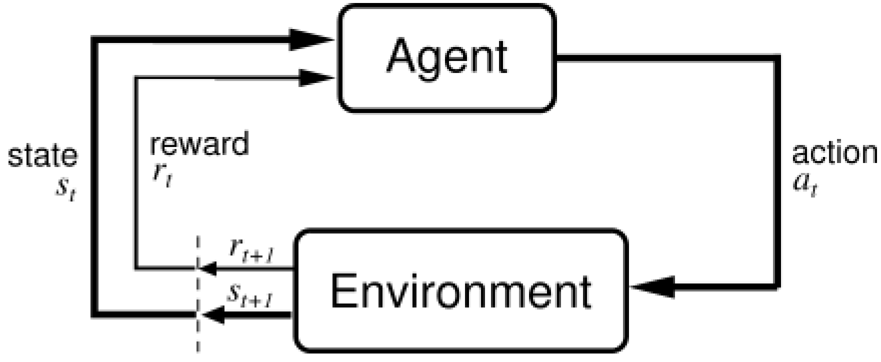
Reinforcement Learning

- Transition model and reward function unknown



Reinforcement Learning

- Transition model and reward function unknown



Agent learns purely from observing outcomes

→ Online trial-and-error learning

Reinforcement Learning

- Passive Learning: Policy is fixed

  → problem of learning value function:
  similar to policy evaluation

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

Reinforcement Learning

- Passive Learning: Policy is fixed

  → problem of learning value function:
  similar to policy evaluation

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

→ instead of knowledge of $P(s'|s,\pi)$, we use the *actual outcomes* to compute a running average:

$$\begin{aligned}V^{\pi}(s)&=(1-\alpha)V^{\pi}(s)+\alpha[R(s)+\gamma V^{\pi}(s')]\\&=V^{\pi}(s)+\alpha[R(s)+\gamma V^{\pi}(s')-V^{\pi}(s)]\end{aligned}$$

Reinforcement Learning

- Passive Learning: Policy is fixed

  → problem of learning value function:
  similar to policy evaluation

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

→ instead of knowledge of $P(s' \mid s, \pi)$, we use the *actual outcomes* to compute a running average:

$$\begin{aligned} V^{\pi}(s) &= (1-\alpha) V^{\pi}(s) + \alpha [R(s) + \gamma V^{\pi}(s')] \\ &= V^{\pi}(s) + \alpha [R(s) + \gamma V^{\pi}(s') - V^{\pi}(s)] \end{aligned}$$

Temporal-Difference-error

Reinforcement Learning

- Passive Learning: Policy is fixed

  → problem of learning value function:
  similar to policy evaluation

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

→ instead of knowledge of $P(s' | s, \pi)$, we use the *actual outcomes* to compute a running average:

$$\begin{aligned} V^{\pi}(s) &= (1-\alpha) V^{\pi}(s) + \alpha [R(s) + \gamma V^{\pi}(s')] \\ &= V^{\pi}(s) + \alpha [R(s) + \gamma V^{\pi}(s') - V^{\pi}(s)] \end{aligned}$$

Temporal-Difference-error

→ since the state transitions occur at a relative frequency dictated by $P(s' | s, \pi)$, this converges to the true $V^{\pi}(s)$

Q-Learning

- How to choose actions?

  → define value function $Q(s,a)$ on state-action pairs!

Q-Learning

- How to choose actions?

  → define value function $Q(s,a)$ on state-action pairs!

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q(s',a')$$

Q-Learning

- How to choose actions?

→ define value function $Q(s,a)$ on state-action pairs!

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

→ iterative TD update:

$$Q(s,a) := Q(s,a) + \alpha [R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Q-Learning

- How to choose actions?

→ define value function $Q(s,a)$ on state-action pairs!

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

→ iterative TD update:

$$Q(s,a) := Q(s,a) + \alpha [R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

→ Best action in state s is simply: $\arg\max_a Q(s,a)$

Exploration - Exploitation

→ Problem: if the agent strictly follows one policy, he does not learn the Q-function for the entire state space!

Exploration - Exploitation

→ Problem: if the agent strictly follows one policy, he does not learn the Q-function for the entire state space!

→ Trade-off between exploiting current knowledge and gathering additional knowledge

Exploration - Exploitation

→ Problem: if the agent strictly follows one policy, he does not learn the Q-function for the entire state space!

→ Trade-off between exploiting current knowledge and gathering additional knowledge

→ Solution: Randomly make suboptimal choices!

Exploration - Exploitation

→ Problem: if the agent strictly follows one policy, he does not learn the Q-function for the entire state space!

→ Trade-off between exploiting current knowledge and gathering additional knowledge

→ Solution: Randomly make suboptimal choices!

<u>ε-greedy agent:</u>

Make random choice with probability ε, otherwise follow optimal policy.

Assignment 2:

1) Implement passive Temporal-Difference learning by completing the function passive_td.

2) Complete the code for q_learning and the function best_policy which computes the optimal policy given a Q-function. Compare the results with the real Q-function as computed by value_iteration_q.

3*) Implement an epsilon-greedy agent.

References and further reading

- Russell, Norvig: Artificial Intelligence – A modern approach.

  (Concise introduction to Markov decision processes and reinforcement learning, source of gridworld example)

- Sutton, Barto: Reinforcement Learning – An introduction.

  (Standard textbook on reinforcement learning)