

COMP3211 Tutorial 6: MDP/RL

Fengming ZHU

Apr. 15&18, 2024

Department of CSE
HKUST

© 2024 Fengming Zhu. All rights reserved.

MDP V.S. Search

Policy Iteration/Value Iteration

Coding Examples

Grid Worlds - via Policy Iteration

Gambler's Problem - via Value Iteration

Maximization Bias Problem - via Q Learning

MDP V.S. Search

Search:

- A set of states \mathcal{S} , initial state I , goal state G
- A set of actions \mathcal{A}
- Deterministic transitions $T : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$
- cost function $c : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- Objective: a path p from I to G that minimizes $c(p)$

MDP V.S. Search

Search:

- A set of states \mathcal{S} , initial state I , goal state G
- A set of actions \mathcal{A}
- Deterministic transitions $T : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$
- cost function $c : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- Objective: a path p from I to G that minimizes $c(p)$

MDP:

- A set of states \mathcal{S} , a terminating condition $End(s)$
- A set of actions \mathcal{A}
- Stochastic transitions $T : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$
- Reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$, with a discount factor γ
- Objective: maximize $\sum_t \gamma^t r_t$

Solution Concept: Policy

Question:

Given a sequential and stochastic decision making problem, in order to come up with an optimal solution, you'd like to know:

- (A) Only your current state
- (B) All the history from the beginning up until now
- (C) Need to know more

Solution Concept: Policy

Question:

Given a sequential and stochastic decision making problem, in order to come up with an optimal solution, you'd like to know:

- (A) Only your current state
- (B) All the history from the beginning up until now
- (C) Need to know more

Markov property:

A state S_t is Markovian iff $P[S_{t+1}|S_t, \dots, S_0] = P[S_{t+1}|S_t]$. That is, your current state is already a “sufficient statistic” that well summarizes the history, also known as the **information state**.

Solution Concept: Policy

Question:

Given a sequential and stochastic decision making problem, in order to come up with an optimal solution, you'd like to know:

- (A) Only your current state
- (B) All the history from the beginning up until now
- (C) Need to know more

Markov property:

A state S_t is Markovian iff $P[S_{t+1}|S_t, \dots, S_0] = P[S_{t+1}|S_t]$. That is, your current state is already a “sufficient statistic” that well summarizes the history, also known as the **information state**.

Policy in MDP:

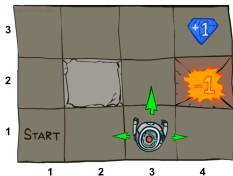
A solution is a policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$

Follow-up Question: Maze

Question:

Given a large maze, you (with deterministic actions U/D/L/R) are supposed to find a nice way from the entrance to the exit, which agent you'd like to choose

- (A) State machines with infinite memory
- (B) Agents that can do A^* search
- (C) Agents that can compute policies
- (D) None of them

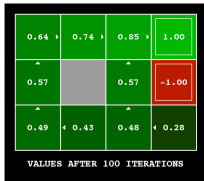
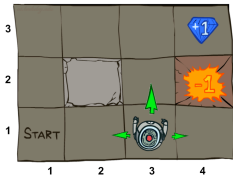


Follow-up Question: Maze

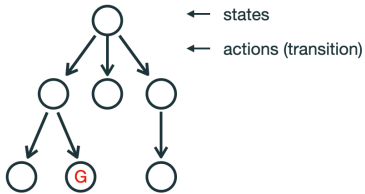
Question:

Given a large maze, you (with deterministic actions U/D/L/R) are supposed to find a nice way from the entrance to the exit, which agent you'd like to choose

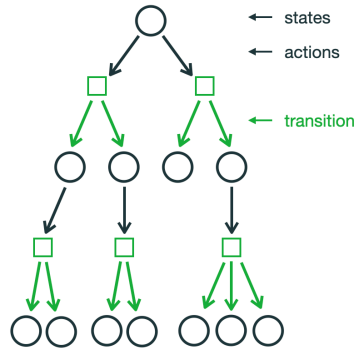
- (A) State machines with infinite memory
- (B) Agents that can do A^* search
- (C) Agents that can compute policies
- (D) None of them



Tree Search: A Conceptual Bridge



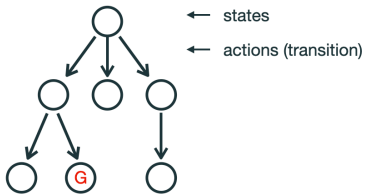
Lookahead - Backtrack



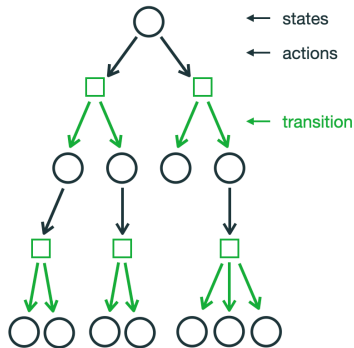
Lookahead - "Backward Induction"

A policy is nothing but a conditional plan!

Tree Search: A Conceptual Bridge



Lookahead - Backtrack



Lookahead - "Backward Induction"

A policy is nothing but a conditional plan!

What if (1) repeat states; (2) infinite horizon?

Bellman (Optimality) Equation

- For state-value function,

$$v_*(s) = \max_a \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma v_*(s')]$$

- For action-value function,

$$q_*(s, a) = \max_a \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} q_*(s', a')]$$

- However, non-linear thus no closed form solution in general!

Policy Iteration/Value Iteration

Policy iteration:

- Initialize $\pi \leftarrow \pi_0, v(\cdot) \leftarrow \vec{0}$
- While π still changing:
 - Policy evaluation: iterate **until convergence**
$$\forall s, v_{\pi}^t(s) \leftarrow \sum_a \pi(a|s) \cdot \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma v_{\pi}^{t-1}(s')]$$
 - Policy improvement: greedy update
$$\forall s, \pi_+(s) \leftarrow \arg \max_{a \in A(s)} q_{\pi}(s, a)$$
 - $\pi \leftarrow \pi_+$

Value Iteration

Value iteration:

- Initialize $\pi \leftarrow \pi_0, v(\cdot) \leftarrow \vec{0}$
- While V still changing:

$$\forall s, v_*^{t+1}(s) \leftarrow \max_a \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma v_*^t(s')]$$

- Extract π_*

$$\forall s, \pi_*(s) \leftarrow \arg \max_a \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma v_*(s')]$$

But note that, intermediate value functions might not correspond to any underlying policy π .

Value iteration (another perspective):

- Initialize $\pi \leftarrow \pi_0, v(\cdot) \leftarrow \vec{0}$
- While V still changing:
 - Policy evaluation (one sweep): iterate **only once**
 $\forall s, v(s) \leftarrow \sum_a \pi(a|s) \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma v(s')]$
 - Policy improvement: greedy update
 $\forall s, \pi_+(s) \leftarrow \arg \max_{a \in A(s)} q(s, a)$
 - $\pi \leftarrow \pi_+$ (every π will be deterministic except for π_0)

Still, intermediate value functions might not correspond to any underlying policy π .

REINFORCEjs: Dynamic Programming Demo

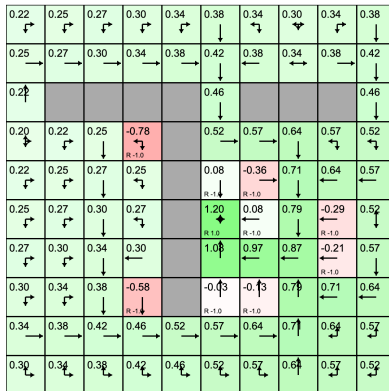
GridWorld: Dynamic Programming Demo

Policy Evaluation (one sweep)

Policy Update

Toggle Value Iteration

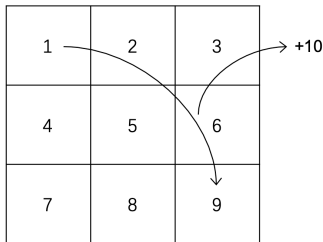
Reset



https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Coding Examples

3×3 Grid World



The agent can be in one of the nine cells at any starting time. It can then move in one of four directions: $\{E, S, W, N\}$. If the agent hits a wall, it remains in its current cell and gets a reward 1. When the agent moves to cell 1, it then immediately moves to cell 9 and gets a reward of 10. The discount factor $\gamma = 0.9$.

Evaluate the uniform policy - Iterative procedures

Initialize $V_{\pi}^0(s) = 0$ for all s .

For each $t=1,\dots,\text{Max}$:

For each state s :

$$V_{\pi}^t(s) = \sum_{s'} T(s, \pi(s), s') [\text{Reward}(s, \pi(s), s') + \gamma V_{\pi}^{t-1}(s')].$$

V:

```
[ [ 8.69914207  2.42431251 -0.11319971]
  [ 2.42431251  0.87545285 -0.47892324]
  [-0.11319971 -0.47892324 -1.30088793] ]
```

Evaluate the uniform policy - Matrix form

$$v_{\pi} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v_{\pi}$$

P:

```
[ [0.    0.    0.    0.    0.    0.    0.    0.    1.  ]  
  [0.225 0.225 0.225 0.    0.225 0.    0.    0.    0.  ]  
  [0.    0.225 0.45  0.    0.    0.225 0.    0.    0.  ]  
  [0.225 0.    0.    0.225 0.225 0.    0.225 0.    0.  ]  
  [0.    0.225 0.    0.225 0.    0.225 0.    0.225 0.  ]  
  [0.    0.    0.225 0.    0.225 0.225 0.    0.    0.225]  
  [0.    0.    0.    0.225 0.    0.    0.45  0.225 0.  ]  
  [0.    0.    0.    0.    0.225 0.    0.225 0.225 0.225]  
  [0.    0.    0.    0.    0.    0.225 0.    0.225 0.45 ]]
```

R:

```
[10.   -0.25 -0.5  -0.25  0.   -0.25 -0.5  -0.25 -0.5 ]
```

V:

```
[ [ 8.69883122  2.42401952 -0.1134855 ]  
  [ 2.42401952  0.87516596 -0.47920629]  
  [-0.1134855  -0.47920629 -1.30116878]]
```

Find the optimal policy - Analytically

Analytically, the optimal policy for every state is to get state 1 as soon as possible. For state 9, the minimal number of steps to reach state 1 is 4 and then agent jumps back to state 9. Thus,

$$V_1 = V_9 + 10, V_9 = 0.9^4 V_1 \Rightarrow V_1 = 29.08, V_9 = 19.08$$

The value of rest states are

$$V_2 = 0.9V_1 = 26.17, V_3 = 0.9V_2 = 23.55, V_4 = 0.9V_1 = 26.17,$$

$$V_5 = 0.9V_2 = 23.55, V_6 = 0.9V_3 = 21.20, V_7 = 0.9V_4 = 23.55,$$

$$V_8 = 0.9V_7 = 21.20$$

The optimal policy will be

NULL	←	←
↑	←↑	←↑
↑	←↑	←↑

Find the optimal policy - Policy iteration

See `mdp_example.ipynb`

Optimal state-values:

```
[[29.078 26.17 23.553]
 [26.17 23.553 21.198]
 [23.553 21.198 19.078]]
```

Extracted policy (random tie-breaking):

NULL	←	←
↑	←	←
↑	←	←

Gambler's Problem

(Sutton's book, example 4.3)

A gambler has the opportunity to make bets on the outcomes of a sequence of coin flips. If the coin comes up heads, he wins as many dollars as he has staked on that flip; if it lands with tails, he loses his stake. The game ends when the gambler wins by reaching his goal of \$100, or loses by running out of money. On each flip, the gambler must decide what portion of his capital to stake, in integer numbers of dollars.

Gambler's Problem- Formulation

This problem can be formulated as an undiscounted, episodic, finite MDP. The state is the gambler's capital, $s \in \{1, 2, \dots, 99\}$ and the actions are stakes, $a \in \{0, 1, \dots, \min(s, 100 - s)\}$. The reward is zero on all transitions except those on which the gambler reaches his goal, when it is $+1$. The state-value function then gives the probability of winning from each state. A policy is a mapping from levels of capital to stakes. The optimal policy maximizes the probability of reaching the goal. Let p_h denote the probability of the coin coming up heads. If p_h is known, then the entire problem is known and it can be solved, for instance, by value iteration.

Gambler's Problem - Value iteration

See `mdp_example.ipynb`

Gambler's Problem - Results

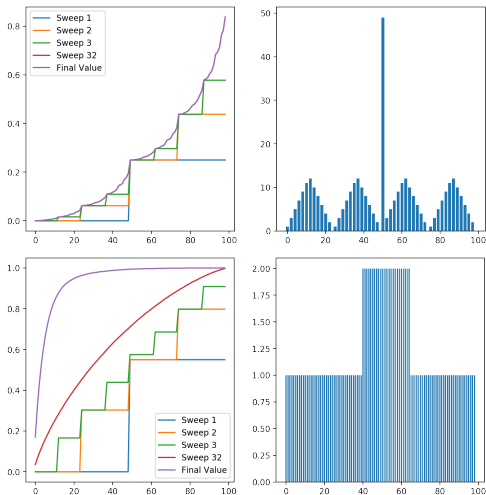


Figure 1: $p_h = 0.25$ (top), $p_h = 0.55$ (bottom)

Maximization Bias Problem

(Sutton's book, example 6.7)

Consider a single state s where there are many actions a whose true values, $q(s,a)$, are all zero but whose estimated values, $Q(s,a)$, are uncertain and thus distributed some above and some below zero.

The maximum of the true values is zero, but the maximum of the estimates is positive, a positive bias. We call this maximization bias.

Maximization Bias Problem

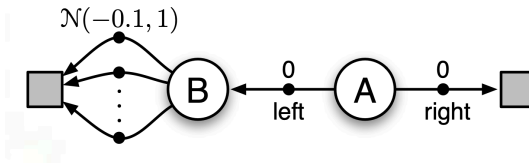


Figure 2: Example for maximization bias.

The MDP has two non-terminal states A and B. Episodes always start in A with a choice between two actions, left and right. The right action transitions immediately to the terminal state with a reward and return of zero. The left action transitions to B, also with a reward of zero, from which there are many possible actions all of which cause immediate termination with a reward drawn from a normal distribution with mean 0.1 and variance 1.0.

Maximization Bias Problem - Q Learning

While observing (S, A, R, S') ...

Q Learning

ϵ -greedy according to Q and update it

$$Q(S, A) \leftarrow (1 - \alpha)Q(S, A) + \alpha(R + \gamma \max_{a \in A} Q(S', a))$$

Double Q Learning

ϵ -greedy according to $Q_1 + Q_2$ and update them

$$Q_1(S, A) \leftarrow (1 - \alpha)Q_1(S, A) + \alpha(R + \gamma Q_2(S', \arg \max_{a \in A} Q_1(S', a)))$$

$$Q_2(S, A) \leftarrow (1 - \alpha)Q_2(S, A) + \alpha(R + \gamma Q_1(S', \arg \max_{a \in A} Q_2(S', a)))$$

See `mdp_example.ipynb`

Maximization Bias Problem - Results

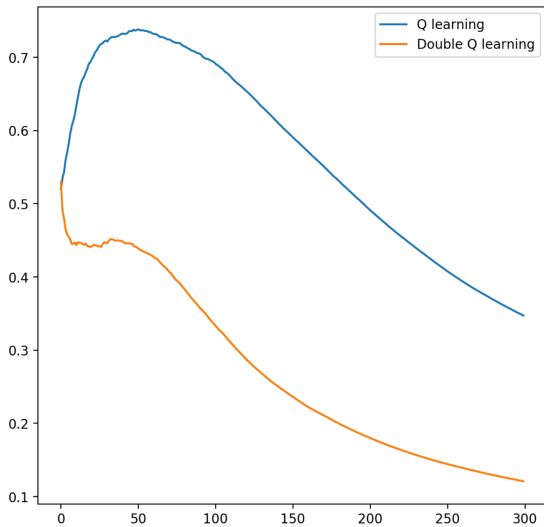


Figure 3: Ratio of selecting the wrong actions.

Thanks!