CS 540 Introduction to Artificial Intelligence
Reinforcement Learning I
University of Wisconsin-Madison

Fall 2022
Outline

• Introduction to reinforcement learning
  – Basic concepts, mathematical formulation, MDPs, policies

• Valuing policies
  – Value functions, Bellman equation, value iteration
We have an **agent interacting** with the **world**

- **Agent** receives a reward based on state of the world
  - **Goal**: maximize reward / utility ($$$)
  - **Note**: data consists of actions & observations
    - Compare to unsupervised learning and supervised learning
Examples: Gameplay Agents

AlphaZero:

https://deepmind.com/research/alphago/
Examples: Video Game Agents

Pong, Atari

Mnih et al, “Human-level control through deep reinforcement learning”
Examples: Video Game Agents

Minecraft, Quake, StarCraft, and more!

Shao et al, "A Survey of Deep Reinforcement Learning in Video Games"
Examples: Robotics

Training robots to perform tasks (e.g., grasp!)

Ibarz et al, "How to Train Your Robot with Deep Reinforcement Learning – Lessons We’ve Learned"
Building The Theoretical Model

Basic setup:
• Set of states, $S$
• Set of actions $A$
• Information: at time $t$, observe state $s_t \in S$. Get reward $r_t$
• Agent makes choice $a_t \in A$. State changes to $s_{t+1}$, continue

Goal: find a map from states to actions maximize rewards.

A “policy”
Markov Decision Process (MDP)

The formal mathematical model:

- **State set** $S$. Initial state $s_0$. **Action set** $A$
- **Reward function**: $r(s_t)$
- **State transition model**: $P(s_{t+1} | s_t, a_t)$
  - Markov assumption: transition probability only depends on $s_t$ and $a_t$, and not earlier history (previous actions or states)
- More generally: $r(s_t, a_t), P(r_t, s_{t+1} | s_t, a_t)$
- **Policy**: $\pi(s) : S \rightarrow A$ action to take at a particular state

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$
Example of MDP: Grid World

Robot on a grid; goal: find the best policy

Source: P. Abbeel and D. Klein
Example of MDP: Grid World

Note: (i) Robot is unreliable    (ii) Reach target fast

\[
\begin{align*}
    r(s) &= -0.04 \text{ for every non-terminal state}
\end{align*}
\]
Grid World Abstraction

Note: (i) Robot is unreliable  (ii) Reach target fast

\[ r(s) = -0.04 \text{ for every non-terminal state} \]
Grid World Optimal Policy

Note: (i) Robot is unreliable    (ii) Reach target fast

\[
\begin{array}{cccc}
3 & \rightarrow & \rightarrow & +1 \\
\hline
2 & \uparrow & \uparrow & -1 \\
\hline
1 & \uparrow & \rightarrow & \rightarrow \\
\end{array}
\]

\[r(s) = -0.04\] for every non-terminal state
The formal mathematical model:

- **State set** $S$. Initial state $s_0$. **Action set** $A$

- **State transition model**: $P(s_{t+1} | s_t, a_t)$
  - Markov assumption: transition probability only depends on $s_t$ and $a_t$, and not previous actions or states.

- **Reward function**: $r(s_t)$

- **Policy**: $\pi(s) : S \rightarrow A$ action to take at a particular state.

How do we find the best policy?
Q 1.1 Which of the following statement about MDP is **not** true?

- A. The reward function must output a scalar value
- B. The policy maps states to actions
- C. The probability of next state can depend on current and previous states
- D. The solution of MDP is to find a policy that maximizes the cumulative rewards
Q 1.1 Which of the following statement about MDP is not true?

- A. The reward function must output a scalar value
- B. The policy maps states to actions
- C. The probability of next state can depend on current and previous states
- D. The solution of MDP is to find a policy that maximizes the cumulative rewards
Q 1.1 Which of the following statement about MDP is not true?

• A. The reward function must output a scalar value (True: need to be able to compare)
• B. The policy maps states to actions (True: a policy tells you what action to take for each state).
• C. The probability of next state can depend on current and previous states (False: Markov assumption).
• D. The solution of MDP is to find a policy that maximizes the cumulative rewards (True: want to maximize rewards overall).
Defining the Optimal Policy

For policy $\pi$, expected utility over all possible state sequences from $s_0$ produced by following that policy:

$$V^\pi(s_0) = \sum_{\text{sequences starting from } s_0} P(\text{sequence})U(\text{sequence})$$

Called the value function (for $\pi$, $s_0$)
Discounting Rewards

One issue: these are infinite series. **Convergence?**

• Solution

\[ U(s_0, s_1 \ldots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \ldots = \sum_{t \geq 0} \gamma^t r(s_t) \]

• Discount factor \( \gamma \) between 0 and 1
  – Set according to how important **present** is VS **future**
  – Note: has to be less than 1 for convergence
Now that $V^\pi(s_0)$ is defined what $a$ should we take?

• First, let $\pi^*$ be the optimal policy for $V^\pi(s_0)$, and $V^*$ its expected utility.

• What’s the expected utility of an action?
  – Specifically, action $a$ in state $s$?
    $$\sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to  Transition probability  Expected rewards
Obtaining the Optimal Policy

We know the expected utility of an action

- So, to get the optimal policy, compute

\[ \pi^*(s) = \arg\max_a \sum_{s'} P(s'|s, a)V^*(s') \]

All the states we could go to
Transition probability
Expected rewards

Credit L. Lazbenik
Slight Problem...

Now we can get the optimal policy by doing

\[ \pi^*(s) = \arg\max_a \sum_{s'} P(s'|s, a)V^*(s') \]

• So we need to know \( V^*(s) \).
  – But it was defined in terms of the optimal policy!
  – So we need some other approach to get \( V^*(s) \).
  – Need some other **property** of the value function!
Bellman Equation

Let’s walk over one step for the value function:

\[ V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s, a)V^*(s') \]

- Current state reward
- Discounted expected future rewards

• Bellman: inventor of dynamic programming
Bellman Equation

Let’s walk over one step for the value function:

\[ V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^*(s') \]

Current state reward

Discounted expected future rewards
Value Iteration

Q: how do we find $V^*(s)$?

• Why do we want it? Can use it to get the best policy

• Know: reward $r(s)$, transition probability $P(s' \mid s,a)$
  – Knowing $r$ and $P$ is the “planning” problem. In reality $r$ and $P$ must be estimated from interactions: “reinforcement learning”

• Also know $V^*(s)$ satisfies Bellman equation (recursion above)

A: Use the property. Start with $V_0(s)=0$. Then, update

$$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid s,a) V_i(s')$$
Value Iteration: Demo

Source: POMDPBGallery Julia Package
Q 2.1 Consider an MDP with 2 states \{A, B\} and 2 actions: “stay” at current state and “move” to other state. Let \( r \) be the reward function such that \( r(A) = 1, \ r(B) = 0 \). Let \( \gamma \) be the discounting factor. Let \( \pi: \pi(A) = \pi(B) = \text{move} \) (i.e., an “always move” policy). What is the value function \( V^\pi(A) \)?

- A. 0
- B. \( 1 / (1 - \gamma) \)
- C. \( 1 / (1 - \gamma^2) \)
- D. 1
Q 2.1 Consider an MDP with 2 states \{A, B\} and 2 actions: “stay” at current state and “move” to other state. Let \(r\) be the reward function such that \(r(A) = 1, r(B) = 0\). Let \(\gamma\) be the discounting factor. Let \(\pi: \pi(A) = \pi(B) = \text{move}\) (i.e., an “always move” policy). What is the value function \(V^\pi(A)\)?

- A. 0
- B. \(1/(1-\gamma)\)
- C. \(1/(1-\gamma^2)\)
- D. 1
Q 2.1 Consider an MDP with 2 states \( \{A, B\} \) and 2 actions: “stay” at current state and “move” to other state. Let \( r \) be the reward function such that \( r(A) = 1, r(B) = 0 \). Let \( \gamma \) be the discounting factor. Let \( \pi: \pi(A) = \pi(B) = \text{move} \) (i.e., an “always move” policy). What is the value function \( V^\pi(A) \)?

- A. 0
- B. \( 1/(1-\gamma) \)
- C. \( 1/(1-\gamma^2) \) (States: A,B,A,B,... rewards 1,0, \( \gamma^2 \),0, \( \gamma^4 \),0, ...)
- D. 1
Summary

• Reinforcement learning setup
• Mathematica formulation: MDP
• Value functions & the Bellman equation
• Value iteration