Artificial Intelligence

CS 165A
Nov 10, 2020

Instructor: Prof. Yu-Xiang Wang

→ Intro to RL
→ Markov Decision Processes
Announcement

• The TAs are still grading the midterm.

• We are hoping to release your midterm grades on Thursday.

• No discussion class this week.
Announcement

• HW3 released last Thursday.

• Topics covered includes
  – Game playing
  – Markov Decision processes

• Programming question:
  – Solve PACMAN with ghosts moving around.
Recap: Expectimax

- Your opponent behave randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5,0.5]
- If you move right, your opponent will select actions with [0.6,0.4]

From MAX point of view, she is playing against a stochastic environment.
Games: Modelling, Inference, Learning

• Modelling:
  – Formulating games as a search problem
  – Modeling your opponent

• Inference:
  – How to search for a strategy
  – Minimax, Expectimax (and Expectiminimax)
  – Pruning
  – Heuristic function and cut-off search

• Learning:
  – Learning heuristic functions
  – Modeling your opponent from data

(Where are the data coming from?)
Reinforcement Learning Lecture Series

• Overview (Today)

• Markov Decision Processes (Today)

• Bandits problems and exploration

• Reinforcement Learning Algorithms
Reinforcement learning in the animal world

- Learn from rewards
- Reinforce on the states that yield positive rewards

Ivan Pavlov
(1849 - 1936)
Nobel Laureate
Reinforcement learning: Applications

Recommendations
buy or not buy

DEEPMIND AI LEARNED HOW TO WALK
Reinforcement learning problem setup

- State, Action, Reward
- Unknown reward function, unknown state-transitions.
- Agents might not even observe the state
Reinforcement learning problem setup

- **State, Action, Reward and Observation**
  \[ S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O} \]

- **Policy:**
  - When the state is observable: \( \pi : \mathcal{S} \rightarrow \mathcal{A} \)
  - Or when the state is not observable
    \[ \pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A} \]

- **Learn the best policy that maximizes the expected reward**
  - Finite horizon (episodic) RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{T} R_t \right] \)
  - Infinite horizon RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \right] \)

\( \gamma \): discount factor
RL for robot control

- **States**: The physical world, e.g., location/speed/acceleration and so on.
- **Observations**: camera images, joint angles
- **Actions**: joint torques
- **Rewards**: stay balanced, navigate to target locations, serve and protect humans, etc.
RL for Inventory Management

- **State**: Inventory level, customer demand, competitor’s inventory
- **Observations**: current inventory levels and sales history
- **Actions**: amount of each item to purchase
- **Rewards**: profit
Demonstrating the learning process

- Mountain car:
  https://www.youtube.com/watch?v=U5w9PoKCOeM
Reading materials for RL

• Introduction:
  – Sutton and Barto: Chapter 1

• Markov Decision Processes
  – AIMA Section 17.1, Sutton and Barto: Ch 3

• Policy iterations / value iterations
  – AIMA Chapter 17.2-17.3, Sutton and Barto Ch 4.

• Bandits
  – Sutton and Barto Ch 2, AIMA Ch. 21.4 (Ch. 22.4 in 4th Edition)

• RL Algorithms:  Sutton and Barto Ch 4, Ch 5, Ch 6, Ch 13
Reinforcement learning is, arguably, the most general AI framework.

• RL: State, Action, Reward, Nothing is known.

• Simplified RL models:
  – iid state $\rightarrow$ Contextual bandits
  – No state, tabular action $\rightarrow$ Multi-arm bandits
  – iid state, no reward $\rightarrow$ Supervised Learning
  – Known dynamics / reward $\rightarrow$ Markov Decision Processes (Control/Cybernetics)
  – No reward / Unknown dynamics $\rightarrow$ System Identification
Reinforcement learning is very challenging

• The agent needs to:
  – Learn the state-transitions  ------ How the world works
  – Learning the costs / rewards  ------ Cost of actions
  – Learning how to search  ------ Come up with a good strategy

• All at the same time
Let us tackle different aspects of the RL problem one at a time

• **Markov Decision Processes:**
  – Dynamics are given no need to learn

• **Bandits:** Explore-Exploit in simple settings
  – RL without dynamics

• **Full Reinforcement Learning**
  – Learning MDPs
Robot in a room. (3 min discussion)

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- what’s the strategy to achieve max reward?
- what if the transitions were deterministic?

actions: UP, DOWN, LEFT, RIGHT

e.g.,

State-transitions with action UP:

80% move up
10% move left
10% move right

*If you bump into a wall, you stay where you are.
Is this a solution?

- only if transitions are deterministic
  - not in this case (transitions are stochastic)

- solution/policy
  - mapping from each state to an action
**Optimal policy**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>♦</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reward for each step: -2
Markov Decision Process (MDP)

- set of states $S$, set of actions $A$, initial state $S_0$
- transition model $P(s' | s,a)$
  - $P([1,2] | [1,1], \text{up}) = 0.8$
- reward function $r(s')$
  - $r([4,3]) = +1$ (Sometimes also depend on $s, a$)
- goal: maximize cumulative reward in the long run

- policy: mapping from $S$ to $A$
  - Overloading notation: $\pi(s)$ outputs an actions (for deterministic policy), or a probability distribution of actions (for stochastic policy).
  - We also use $\pi(a|s)$ as a short hand for $P_{\pi}(a|s)$ --- the conditional probability table under policy $\pi$
Tabular MDP

- **Discrete State, Discrete Action, Reward and Observation**
  \[ S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O} \]

- **Policy:**
  - When the state is observable: \( \pi : \mathcal{S} \rightarrow \mathcal{A} \)
  - Or when the state is not observable: \( \pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A} \)

- **Learn the best policy that maximizes the expected reward**
  - Finite horizon (episodic) RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{T} R_t] \)
    \( T: \) horizon
  - Infinite horizon RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} R_t] \)
    \( \gamma: \) discount factor
What is Markovian about MDPs?

• “Markov” generally means that given the present state, the future and the past are independent

• For Markov decision processes, “Markov” means action outcomes depend only on the current state

\[ P(S_{t+1} = s'|S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0) = P(S_{t+1} = s'|S_t = s_t, A_t = a_t) \]

• This is just like search, where the future (available actions, states to transition to) could only depend on the current state (not the history)

Andrey Markov (1856-1922)

(slide credit: Virtue and Rosenthal)
This is a conditional independence assumption!

- Example of a finite horizon MDP with $H = 3$, as a BayesNet
This is a **conditional independence assumption**!

- Example of an infinite horizon MDP (as a BayesNet)
State-space diagram representation of an MDP: An example with 3 states and 2 actions.

\[ r(s_2, a_1, s_1) = -2 \]

\[ r(s_2, a_1, s_2) = 50 \]

\[ r(s_2, a_2, s_3) = -1 \]

* The reward can be associated with only the state s’ you transition into.
* Or the state that you transition from s and the action a you take.
* Or all three at the same time.
Reward function and Value functions

- Immediate reward function $r(s, a, s')$
  - expected **immediate** reward

- State value function: $V^\pi(s)$
  - expected **long-term** return when starting in $s$ and following $\pi$

- State-action value function: $Q^\pi(s, a)$
  - expected **long-term** return when starting in $s$, performing $a$, and following $\pi$

- Useful for finding the optimal policy
  - can estimate from experience
  - pick the best action using $Q^\pi(s, a)$
Reward function and Value functions

- Immediate reward function $r(s,a,s')$
  - expected immediate reward
    \[ r(s, a, s') = \mathbb{E}[R_1 | S_1 = s, A_1 = a, S_2 = s'] \]
    \[ r^\pi(s) = \mathbb{E}_{a \sim \pi(a|s)}[R_1 | S_1 = s] \]

- state value function: $V^\pi(s)$
  - expected long-term return when starting in $s$ and following $\pi$
    \[ V^\pi(s) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + \ldots + \gamma^{t-1} R_t + \ldots | S_1 = s] \]

- state-action value function: $Q^\pi(s,a)$
  - expected long-term return when starting in $s$, performing $a$, and following $\pi$
    \[ Q^\pi(s, a) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + \ldots + \gamma^{t-1} R_t + \ldots | S_1 = s, A_1 = a] \]
Bellman equations – the fundamental equations of MDP and RL

• An alternative, recursive and more useful way of defining the V-function and Q function

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

• Quiz:
  – Prove Bellman equation from the definition in the previous slide.
  
  – Write down the Bellman equation using Q function alone.
    \[ Q^\pi(s, a) = ? \]
Bellman equations – the fundamental equations of MDP and RL

• An alternative, recursive and more useful way of defining the V-function and Q function

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

• More quiz:
  – On AIMA textbook, reward is only a function of the state your transition into (Think about we collect a reward when we transition into s’). What is the Bellman equation in this special case?
  – Sometimes, the reward is conditionally independent to s’ given s, a. What is the Bellman equation in this special case?
Let’s work out the Value function for a specific policy

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>+1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>-1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>UP</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LEFT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RIGHT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**actions:** UP, DOWN, LEFT, RIGHT

**e.g., UP**

**state-transitions with action UP:**
- 80% move UP
- 10% move LEFT
- 10% move RIGHT

*If you bump into a wall, you stay where you are.*

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

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

\[
1.0 + 0.8 \times (+1-0.04 + 0) + 0.1 \times (-0.04 + V^\pi([3,2])) + 0.1 \times (-0.04 + V^\pi([3,3]))
\]
Optimal value functions

- There’s a set of *optimal* policies
  - $V^{\pi}$ defines partial ordering on policies
  - They share the same optimal value function
    \[ V^*(s) = \max_{\pi} V^{\pi}(s) \]

- Bellman optimality equation
  \[ V^*(s) = \max_a \sum_{s'} P(s'|s,a) \left[ r(s,a,s') + \gamma V^*(s') \right] \]
  - System of n non-linear equations
  - Solve for $V^*(s)$
  - Easy to extract the optimal policy

- Having $Q^*(s,a)$ makes it even simpler
  \[ \pi^*(s) = \arg \max_a Q^*(s,a) \]
Inference problem: given an MDP, how to compute its optimal policy?

• It suffices to compute its $Q^*$ function, because:

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

• It suffices to compute its $V^*$ function, because:

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$
Algorithms for calculating the V* function

- Policy evaluation, policy-improvement
- Policy iterations
- Value iterations
Dynamic programming

• main idea
  – use value functions to structure the search for good policies
  – need a known model of the environment

• two main components
  – policy evaluation: compute $V^\pi$ from $\pi$
  – policy improvement: improve $\pi$ based on $V^\pi$

  – start with an arbitrary policy
  – repeat evaluation/improvement until convergence
Policy evaluation/improvement

- policy evaluation: \( \pi \rightarrow V^\pi \)
  - Bellman eqn’s define a system of n eqn’s
  - could solve, but will use iterative version
    \[
    V_{k+1}^\pi(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^\pi(s')]
    \]
  - start with an arbitrary value function \( V_0 \), iterate until \( V_k \) converges

- policy improvement: \( V^\pi \rightarrow \pi' \)
  \[
  \pi'(s) = \arg \max_a Q^\pi(s,a) \\
  = \arg \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^\pi(s')]
  \]
  - \( \pi' \) either strictly better than \( \pi \), or \( \pi' \) is optimal (if \( \pi = \pi' \))
Policy/Value iteration

- Policy iteration
  \[ \pi_0 \rightarrow E \ V^{\pi_0} \rightarrow I \ \pi_1 \rightarrow E \ V^{\pi_1} \rightarrow I \ldots \rightarrow I \ \pi^* \rightarrow E \ V^* \]
  - two nested iterations; too slow
  - don’t need to converge to \( V^{\pi_k} \)
    - just move towards it

- Value iteration
  \[ V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a)[r(s, a, s') + \gamma V_k(s')] \]
  - use Bellman optimality equation as an update
  - converges to \( V^* \)
So far no learning at all. On Thursday:

- More on MDPs
- MDP inferences
- Start bandits and exploration