Dynamic Programming in Tabular Case

CSE599G: Deep Reinforcement Learning

Aravind Rajeswaran and Kendall Lowrey

University of Washington Seattle

April 2, 2018

Overview

Last week:

- Introduction to the course
- Basics of MDPs

Some general comments:

- Probability and general mathematical maturity assumed as *prerequisite*
- MDP notations are indeed a bit hard to understand from just one pass

This week:

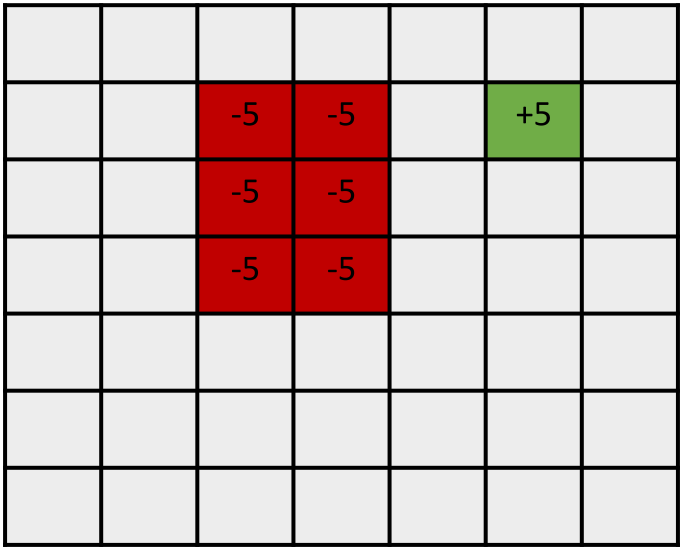
- Review MDPs
- Algorithm to solve some very simple MDPs with major assumptions
- Start moving towards learning with function approximation (deepRL)

Parts of MDP: Review

- Formally, MDP is a tuple: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho_0, \gamma, T \rangle$
  - $\mathcal{S}$ = states (joint positions in robot, concentrations in chemical reaction)
  - $\mathcal{A}$ = actions (motor torques, how much chemical to add)
  - $\mathcal{R}(s, a) \to \mathbb{R}$ is the "reward" function
  - $\mathcal{P} \equiv \mathbb{P}(s' | s, a)$ is the transition dynamics
  - ρ_0 = initial state distribution (i.e. state at time = 0)
  - T = horizon (how long does the MDP last)
  - γ = discount factor *(forget this for now, we'll come to this later)*

Grid world example

Task = find the optimal policy to go from any location to the goal location




= goal (+5 reward)

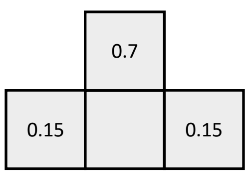

= locations to avoid (obstacles, pits etc. -5 reward)

Actions: up, down, left, right

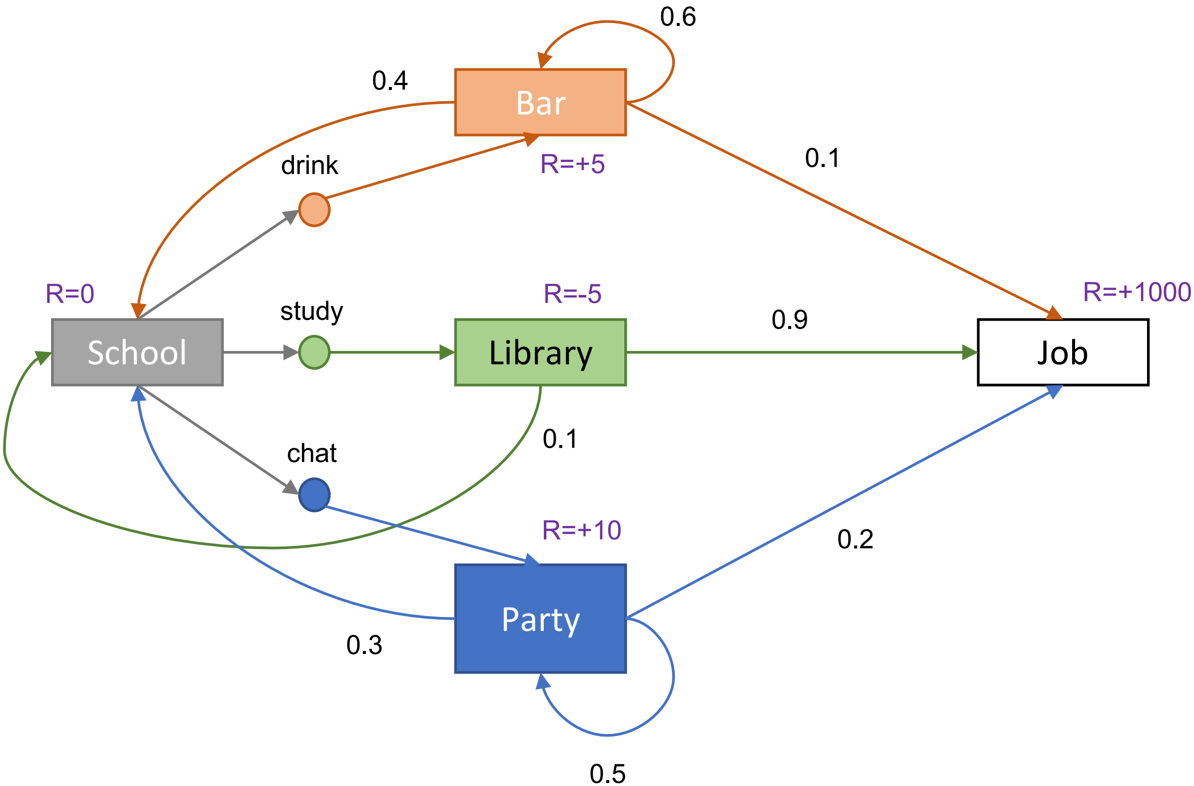
Dynamics:

- Move to desired ”adjacent” grid with $\mathbb{P} = 0.7$ and the two orthogonal directions with $\mathbb{P} = 0.15$ each.
- If at edges of grid, all probability of moving outside goes towards being in the same location.

action=“up”



Student MDP example



Value Functions: Review

- Let's focus on what it means for now; leave how to compute it for later.
- Let's forget discounting for now (we'll get to it later)
- $V^{\pi}(\boldsymbol{s}, \boldsymbol{t})$: How much cumulative reward do I **expect** to accumulate till the end of the horizon if I start from **state (s)** at **time (t)** and follow **policy** $(\boldsymbol{\pi})$

$$V^{\pi}(s,t) = \mathbb{E}\left[\sum_{t'=t}^{t'=T} R(s_{t'}, a_{t'}) \mid s_t = s\right]$$

- $s_{t'}$ and $a_{t'}$ are random variables: how are they generated?
- Define trajectory as $\tau_{t:T} = (s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots s_T, a_T, r_T)$, so that the expectation is now over trajectories. Quantities generated as $s_{t+1} \sim \mathbb{P}(. | s_t, a_t)$, $\; a_t \sim P(. | s_t)$ and $\; r_t \sim R(s_t, a_t)$

Value Functions: Review

- $V^{\pi}(s, t)$: How much cumulative reward do I **expect** to accumulate till the end of the horizon if I start from **state (s)** at **time (t)** and follow **policy** (π)

$$V^{\pi}(s,t) = \mathbb{E}\left[\sum_{t'=t}^{t'=T} R(s_{t'}, a_{t'}) \mid s_t = s\right]$$

- Forgetting efficiency, one way to compute the above quantity, that is also conceptually the easiest to understand is as follows:
    - Start from s and t (so T-t steps left), and simulate trajectories
    - For each trajectory, add up the rewards
    - Take average over trajectories

Horizon and discounting

- Summing up rewards in the infinite horizon case is problematic:

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

- The time is not required as an argument to V (time runs till infinity)
- The RHS in general need not be finite (e.g. all rewards are $\geq r_{min}$)
- One way to make the math well defined is average case:

$$V^{\pi}(s) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=0}^{T} R(s_t, a_t) \ | \ s_0 = s\right]$$

- Analyzing the average case is harder, but there is some theory

Horizon and discounting

- Discounting is another way to make the quantities well defined:

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

- If $R(s,a) \leq r_{max}\ \forall(s,a)$ (i.e. there is an upper bound on the reward), we have:

  $\sum_{t=0}^{t=\infty} \boldsymbol{\gamma}^{\boldsymbol{t}} R(s_t, a_t) \leq \frac{1}{1-\boldsymbol{\gamma}}\ r_{max}$ so that V^π is always well defined for $\gamma \in [0,1)$

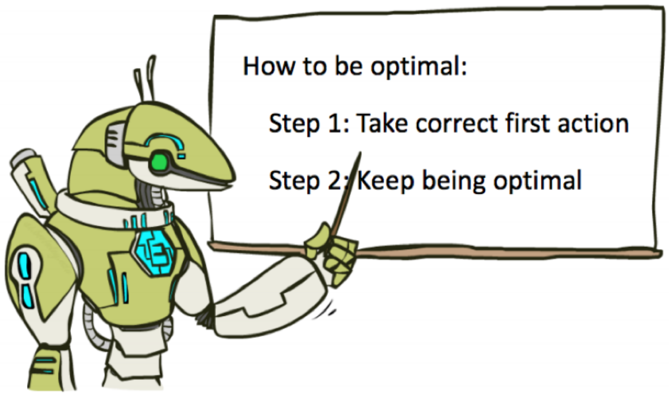
- **Intuition:** Drop an explicit or external clock. Start a new clock from each state and pretend that the horizon is $\frac{1}{1-\gamma}$. *(usually called the "effective horizon")*

- Has some other motivations like we discussed earlier: time value of money in economics, risk-sensitivity and uncertainty in neuroscience etc.

Why we need the value function

- Summarize long term quantities and abstract away temporal nature
- If we can get the optimal value function, we have solved the MDP
- Provides a recursive recipe for attacking the MDP problem

$$\pi^*(s) = argmax_a \, \mathbb{E}[\, R(s,a) + \gamma V^*(s')]$$
$$\pi^*(s) = argmax_a \, Q^*(s,a)$$

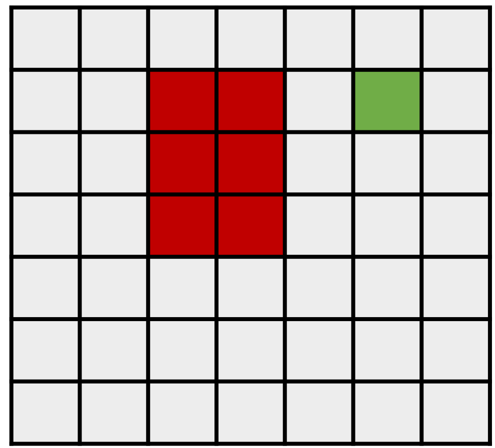


Plan for today

- Given a policy π, how to **efficiently** compute V^{π} ?
  - Called policy evaluation
  - Today, we will do this assuming we know the transition model
  - Much harder in the unknown model case, topic of research
- Given π and V^{π} , how to improve policy?
  - Called policy improvement
  - Today, we will do this assuming a tabular representation
  - Much harder with function approximation, a bit part of DeepRL
- Iteratively performing policy evaluation and policy improvement would lead us to the optimal policy. We will show that in the tabular case, this scheme would converge to the *globally optimal solution!*

Finite MDPs and tabular representation

- Finite MDPs implies a finite number of states and actions
- We can represent such problems through a table
- Grid world example: $(|.| =$ number of entries$)$
    - Each grid square is a state
    - For each state, we have 4 actions
    - So, policy is a table with $|S| \times |A|$ entries.
    - Value function is a table with $|S|$ entries
    - Q function is a table with $|S| \times |A|$ entries.
    - Transition dynamics is a table with $|S| \times |A| \times |S|$ entries.
    - Overall, space complexity is $O(|S|^2 |A|)$



Naïve Policy Evaluation

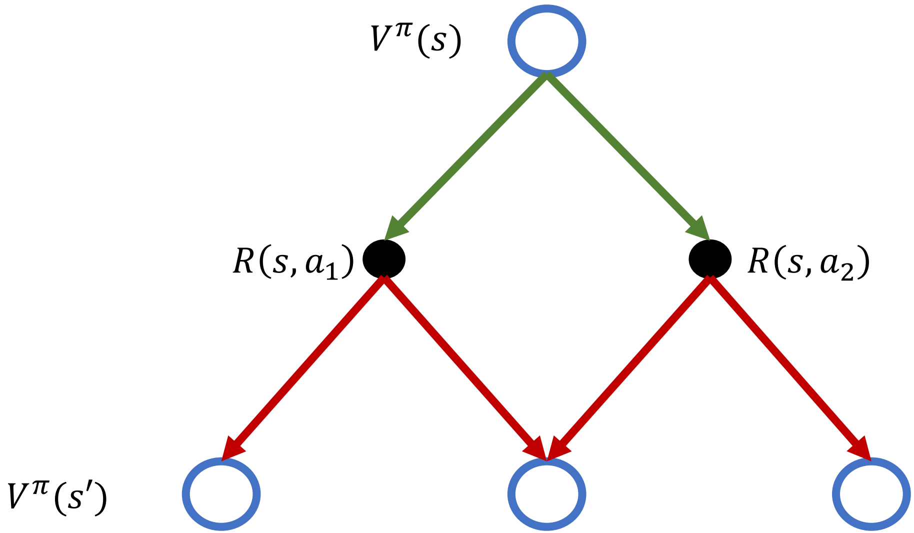
- Let us consider the naïve simulation based approach we outlined earlier
- We will simply use the definition of value function, and approximate the expectation using sample based average. We will simulate for an effective horizon of $T = \frac{1}{1-\gamma}$

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

- Complexity of this procedure is: $O(|S|\times|A|\times|S|\times T\times K)$
- Need to do this for every state, need to query policy for action, need to query MDP for the next state – we repeat this for T steps and K times (sample avg)
- For reasonable level of variance in the estimate, we need $K = O(|S|)$. Overall complexity is: $O(|S|^3|A|T)$. Can we do better? **YES!!**

Bellman Recursion

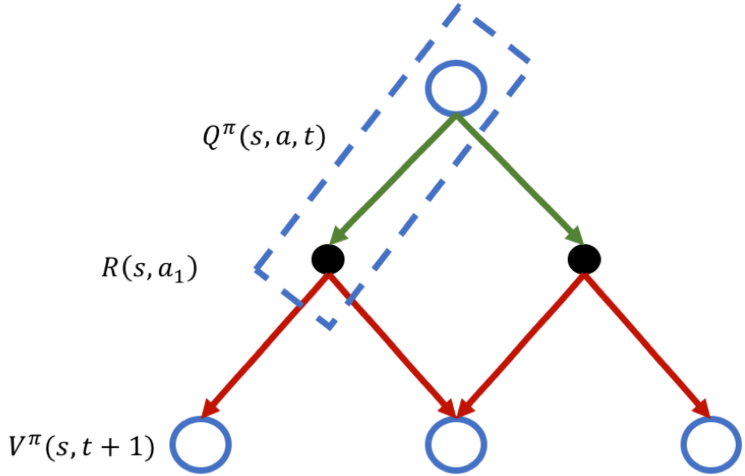
$$V^{\pi}(s) = \sum_{a} \pi(a|s) \left(R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^{\pi}(s') \right)$$



Bellman Recursion

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

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



Bellman Recursion

- Structure due to the sequential nature of the problem and Markov property
- If we know values at some state, we can ”backup” this information to other states, since values need to obey the recursion.
- This bootstrapping is the key to the efficiency of many RL methods.
- Using the recursive relationship, policy evaluation has complexity: $O(|S|^3)$
- Define following for notational convenience:

$$R^\pi(s) = \sum_a \pi(a|s) R(s,a)$$

$$\mathbb{P}^\pi(s'|s) = \sum_a \pi(a|s)\, \mathbb{P}(s'|s,a)$$

Policy evaluation

- Rewriting the recursive relationship:

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

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

- In matrix notation, this is: $V^{\pi} = R^{\pi} + \gamma P^{\pi} V^{\pi}$
- Solve this as a system of linear equations:

$$V^{\pi} = (I - \gamma P^{\pi})^{-1} R^{\pi}$$

Policy evaluation

When do we have a solution?

- Note that each row of the P matrix sums to 1 and each entry is >=0 and <1
- Thus, the maximum eigen value of P is 1
- $(I - \gamma P^{\pi})$ is thus invertible when $\gamma \in [0,1)$

Incremental solution method instead of matrix inversion (k is iteration counter):

$$V_{k+1} = R^{\pi} + \gamma P^{\pi} V_k$$

i.e. for each state $s \in S$ do:

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

Stop when $|V_{k+1}^{\pi} - V_k^{\pi}|_{\infty} \leq \epsilon$ where $|x|_{\infty} = \max_i |x_i|$

This incremental approach is related to ***Jacobi method*** for solving linear equations.

Policy evaluation

What is the complexity?

- For the matrix inversion procedure, $(I - \gamma P^{\pi})$ is a matrix of size $|S| \times |S|$, so this has complexity (naïve inversion methods): $O(|S|^3)$
- For the incremental approach, per iteration requires $O(|S|^2)$ given P^{π} matrix
- Incremental approach converges in $\mathrm{O}(|S|)$ iterations
- Computing P^{π} has a cost of $O(|S|^2|A|)$ (one time)
- So overall, the incremental method has complexity of $O(|S|^3 + |S|^2|A|)$ in the worst case, but in practice much faster.
- Compare to the naïve case of $O(|S|^3|A|T)$

Policy Improvement

- Given π and V^{π}, get a new policy π_{new} that is better.

- Notice that given V^{π} and the MDP (reward, transitions), we can write the Q function easily as: $Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^{\pi}(s')$

- Also, notice that $\max_{a} Q^{\pi}(s,a) \geq V^{\pi}(s)$

- Taking a cue from this, define the new policy as:

$$\pi_{new}(s) = \arg\max_{a} Q^{\pi}(s,a) \ \ \forall s$$

Policy Iteration

- Policy iteration is an iterative improvement algorithm to fine the **optimal policy**
- We will work with deterministic policies now (dynamics can still be stochastic)
- For infinite horizon finite MDPs, there will be at least one globally optimal deterministic policy (why?)

Initialize π_0 for all states (arbitrarily)

For i = 1,2,3, … (till convergence)

- Policy evaluation: compute the value of π_i (i.e. V^{π_i})

- Generate corresponding Q function

- Policy improvement: $\pi_{i+1} = \arg\max_a Q^{\pi_i}(s, a)$

Stop when policy does not change for any state

Policy Iteration

Each policy improvement step leads to monotonic improvement in the value

$$V^{\pi_i}(s) \leq \max_a Q^{\pi_i}(s,a)$$

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

$$= R\big(s, \pi_{i+1}(s)\big) + \gamma \sum_{s'} P(s'|s, \pi_{i+1}(s)) V^{\pi_i}(s')$$

$$\leq R\big(s, \pi_{i+1}(s)\big) + \gamma \sum_{s'} P(s'|s, \pi_{i+1}(s)) \left(\max_{a'} Q^{\pi_i}(s', a')\right)$$

$$\leq R\big(s, \pi_{i+1}(s)\big) + \gamma \sum_{s'} P(s'|s, \pi_{i+1}(s)) \left(R(s', \pi_{i+1}(s') + \gamma \sum_{s''} P(s''|s', \pi_{i+1}(s')) V^{\pi_i}(s'')\right)$$

$$= V^{\pi_{i+1}}(s)$$

Next Class

- Convergence of policy iteration to globally optimal solution
- Value iteration (related method) and proof of convergence
- Recitation for setting up MuJoCo (for Homework 1)
- Start DeepRL with simplest method: evolutionary strategies