Agent-Based Modeling and Simulation

Finite Markov Decision Processes

Dr. Alejandro Guerra-Hernández

Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097

mailto:aguerra@uv.mx
https://www.uv.mx/personal/aguerra/abms

Doctorado en Inteligencia Artificial 2024

Universidad Veracruzana

Credits

- These slides are based on the book of Sutton and Barto [1], chapter 3. Any difference with this source is my responsibility.
-
-

Universidad Veracruzana

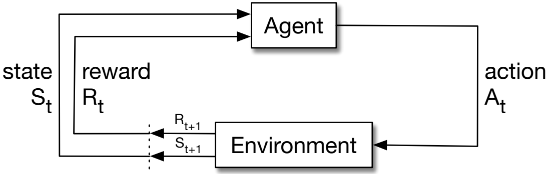
MDPs as a problem

- MDPs characterize the problem we try to solve in the following session.
- They involved evaluative feedback, but also an associative aspect –choosing different actions in different situations.
- They are a classical formalization of sequential decision making, where actions influenced not just immediate rewards, but also subsequent situations, or states, and through those future rewards.
- Thus MDPs involve delayed reward and the need to trade off immediate and delayed reward.

Universidad Veracruzana

A frame for interaction based learning

- For a set of states ($\mathcal{S}$), actions ($\mathcal{A}$), and rewards ($\mathcal{R}$).
- There is a sequence of discrete time steps, $t = 0, 1, 2, 3, \ldots,$ where:
  - At each step t, the agent receives some representation of the environment's state $S_t \in \mathcal{S}$, and on that basis selects an action, $A_t \in \mathcal{A}(s)$.
  - At the next step, in part as a consequence of its action, the agent receives a numerical reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and finds itself at a new state $S_{t+1} \in \mathcal{S}$.



Universidad Veracruzana

Trajectories

- The MDP and agent together thereby give rise to a sequence or trajectory that begins like this:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \quad (1)$$

- Observe the similarities with the run of the basic agent architecture reviewed in the MAS course.

Universidad Veracruzana

Finite MDPs

- The sets of states ($\mathcal{S}$), actions ($\mathcal{A}$), and rewards ($\mathcal{R}$) are finite.
- The random variables R_t and S_t have well defined discrete probability distributions dependent only on the preceding state and action.
- For $s' \in \mathcal{S}$ and $r \in \mathcal{R}$ there is a probability of occurrence of those values at time t, given particular values of the preceding state and action:

$$p(s', r \mid s, a) \doteq Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\} \quad (2)$$

  for all $s', s \in \mathcal{S}$, $r \in \mathcal{R}$, and $a \in \mathcal{A}(s)$.
- The function p defines the dynamics of the MDP.

Universidad Veracruzana

Constrain on p

- So, the dynamics function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is an ordinary deterministic function of four arguments.
- Since p specifies a probability distribution for each choice of s and a, then:

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1 \quad (3)$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$.

Universidad Veracruzana

The Markov Property

- In a MDP, the probabilities given by p completely characterize the environment's dynamics.
- This is best viewed as a restriction not on the decision process, but on the state.
- The state must include information about all aspects of the past agent-environment interaction that make a difference in the future.
- If it does, it is said to have the Markov property.
- The property is assumed in what follows.

Universidad Veracruzana

State-transition Probabilities

$$p(s' \mid s, a) \doteq Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\}$$
$$= \sum_{r \in \mathcal{R}} p(s', r \mid s, a). \tag{4}$$

Universidad Veracruzana

Expected Rewards

$$
\begin{aligned}
r(s, a) &\doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a\right] \\
&= \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a).
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
r(s, a, s') &\doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'\right] \\
&= \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}.
\end{aligned} \tag{6}
$$

Universidad Veracruzana

An Abstract and Flexible Framework

- Time steps need not refer to fixed intervals, but arbitrary successive stages of decision making and acting.
- Actions can be low-level controls, *e.g.*, the voltages applied to motors in a robot arm; or high-level decisions, *e.g.*, whether or not to have a lunch.
- States can be completely determined by low-level sensations, *e.g.*, sensor readings; or be more high-level, e.g., symbolic descriptions *à la* BDI.
- Actions can be mental, *i.e.*, internal; or external in the sense they affect the environment.

Universidad Veracruzana

The Boundary Agent/Environment

- The boundary between agent and environment is typically not the same as the physical boundary of a robot's or animal's body.
- Example. The motors and mechanical linkages of a robot and its sensing hardware should be considered part of the environment rather than part of the agent.
- Rewards too are considered as external to the agent.
- The boundary represents the limit of the agent's absolute control, not of its knowledge.

Universidad Veracruzana

Efficiency

- The MDP framework is a considerable abstraction of the problem of goal-directed learning from interaction.
- Any problem is reduced to three signals:
  - The choices made by te agent (actions).
  - The basis on which choices are made (states).
  - The agent's goal (rewards).
- Particular states and actions vary greatly from task to task, and how they are represented can strongly affect performance.
- Representational choices are at present more art than science.

Universidad Veracruzana

Bioreactor

- The actions might target temperatures and stirring rates passed to lower-level control systems, linked to heating elements and motors to attain the targets.
- The states are likely to be thermocouple and other sensory readings, perhaps filtered and delayed, plus symbolic inputs representing the ingredients in the vat and the target chemical.
- The rewards might be moment-to-moment measures of the rate at which the useful chemical is produced by the bioreactor.
- Observe: States and actions are lists or vectors, while rewards are single numbers. This is typical of RL.

Pick-and-Place Robot

- To learn movements that are fast and smooth, the learning agent will have to control the motors directly and have low-latency information about the current positions and velocities of the mechanical linkages.
- Actions might be voltages applied to each motor at each joint.
- States might be the latest readings of joint angles and velocities.
- The reward might be +1 for each object successfully picked and placed.
- To encourage smooth movements, a small negative reward can be given as a function of the moment-to-moment "jerkiness" of the motion.

Universidad Veracruzana

Recycling Robot I

- A mobile robot has the job of collecting empty soda cans in the office environment.
- It has sensors for detecting cans, and an arm and a gripper that can pick them up and place them in an onboard bin.
- It runs on a rechargeable battery.
- The robot's control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper.
- High-level decisions about how to search for cans are made by a RL agent based on the current charge level of the battery.
- Assume that only two charge levels can be distinguished, comprising a small state set $\mathcal{S} = \{high, low\}$.

Universidad Veracruzana

Recycling Robot II

- In each state, the agent can decide whether to:
  1. Actively search for a can for a certain period of time;
  2. Remain stationary and wait for someone to bring it a can; or
  3. Head back to its home base to recharge its battery.
- When the energy level is *high*, recharging will always be foolish, so it is not included in the action set for such state. The action sets are:
  - $\mathcal{A}(high) = \{search, wait\}$;
  - $\mathcal{A}(low) = \{search, wait, recharge\}$.
- The rewards are zero most of the time, but become positive when the robot secures an empty can, or large and negative if the battery runs all the way down.

Universidad Veracruzana

Recycling Robot III

- The best way to find cans is to actively search for them, but this runs down the robot's battery, whereas waiting does not.
- Whenever the robot is searching, the possibility exists that its battery will become depleted. In this case the robot must shut down and wait to be rescued (producing a low reward).
- If the energy level is high, then a period of active search can always be completed without risk of depleting the battery.
- A period of searching that begins with a high energy level leaves the energy level high with probability α and reduces it to low with probability $1 - \alpha$.
- On the other hand, starting when the energy level is low leaves it low with probability β and depletes the battery with probability $1 - \beta$.

Universidad Veracruzana

Recycling Robot IV

- In the latter case, the robot must be rescued, and the battery is then recharged back to *high*.
- Each can collected by the robot counts as a unit reward, whereas a reward of -3 results whenever the robot has to be rescued.
- Let r_{search} and r_{wait} with $r_{search} > r_{wait}$, respectively denote the expected number of cans the robot will collect (expected reward).
- Finally, suppose that no cans can be collected during a run home for charging, neither on a step in which the battery is depleted.

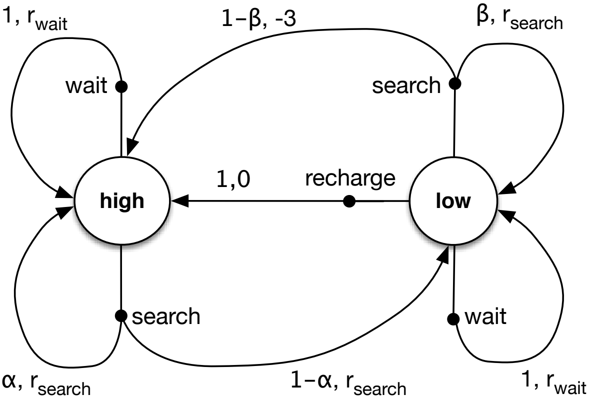
Universidad Veracruzana

The finite MDP as a table

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1 - \alpha$	r_{search}
low	search	high	$1 - \beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	-

Universidad Veracruzana

The finite MDP as a Transition Graph



Reward

- The purpose or goal of the agent is formalized in terms of a special signal, called reward, passing from the environment to the agent.
- At each time step t, the reward is a simple number, $R_t \in \mathbb{R}$.
- Informally, the agent's goal is to maximize the total amount of received reward (cumulative reward not the immediate one).
- The reward hypothesis: All of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal, called reward.
- The use of a reward signal to formalize the idea of a goal is one of the most distinctive features of RL.

Universidad Veracruzana

Examples

- Although formulating goals in terms of reward signals might appear limiting, it has proved to be flexible and widely applicable:
  - To make a robot learn to walk, researchers have provided reward on each time step proportional to the robot's forward motion.
  - In making a robot learn to escape from a maze, the reward is often -1 for every time step that passes prior to escape; encouraging the fastest possible escape.
  - To make a robot learn to find and collect empty soda cans for recycling, one might give a reward of zero most of the time, and the +1 for each can collected. Bumping into things might get negative reward.
  - An agent learning chess or checkers receive +1 as reward when winning, -1 for losing, and 0 otherwise.

Universidad Veracruzana

Considerations

- The agent always learn to maximize its reward. If we want it to do something for us, we must provide rewards to it in such a way that maximizing them achieve our goals.
- The reward signal is not the place to impart the agent prior knowledge about how to achieve what we want it to do.
- Example: A chess playing agent must be rewarded only for actually winning, not for achieving subgoals as taking its opponent's pieces or gaining control of the center of the board.
- The reward signal is your way of communicating to the robot what you want it to achieve, not how you want it achieved.

Universidad Veracruzana

Expected Return

- If the sequence of rewards received after time step t is denoted $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$, then what precise aspect of this sequence do we wish to maximize?
- In general we seek at the expected return, where the return, denoted G_t, is defined as some specific function of the reward sequence, *e.g.*, the sum of rewards:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \tag{7}$$

where T is a final time step.

Universidad Veracruzana

Episodic Tasks

- This makes sense when the agent-environment interaction breaks naturally into episodes, where a terminal state is reached at the end.
- After that, the system is reset to standard starting state or to a sample from a standard distribution of starting states.
- Tasks with episodes of this kind are called episodic tasks.
- In episodic tasks we need to distinguish the set of all non terminal states, denoted by $\mathcal{S}$, from the set of all states plus the terminal state $\mathcal{S}^+$
- The time of termination T is a random variable that normally varies from episode to episode.

Universidad Veracruzana

Continuing Tasks

- Where tasks go continually without limit.
- Examples: An on-going process control task; or an application to a robot with a long life span.
- The return formulated in Eq. 7 is problematic since $T = \infty$, and the return itself easily become infinite too.
- We need a definition of return that is slightly more complex conceptually, but much simpler mathematically.

Universidad Veracruzana

Discounted return

- The additional concept that we need is that of discounting.
- The agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized.
- It chooses A_t to maximize the expected discounted return:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}, + \dots \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \end{aligned} \tag{8}$$

where $0 \leq \gamma \leq 1$ is the discount rate.

Observations

- The discount rate determines the present value of future rewards: a reward received k steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately.
- if $\gamma < 1$ in Eq. 8, the infinite sum has a finite value as long as the reward sequence R_k is bounded.
- If $\gamma = 0$, then the agent is myopic in being concerned only with maximizing immediate rewards: Choosing A_t in order to maximize R_{t+1}.
- As γ approaches 1, the return objective takes future rewards into account more strongly, the agent becomes more farsighted.

Universidad Veracruzana

Relation over Returns

- Returns at succesive time steps are related to each other in a way that is important for the theory, and algorithms of RL:

$$
\begin{aligned}
G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}, + \gamma^3 R_{t+4} + \dots \\
&= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned} \tag{9}
$$

- Note that this works for all time steps $t < T$, even if termination occurs at $t + 1$, if we define $G_T = 0$.

Universidad Veracruzana

Finite Reward

- Observe that although the return in Eq. 8 is a sum of an infinte number of terms, it is still finite if the reward is nonzero and constant –if $\gamma < 1$.
- Example. If the reward is a constant $+1$, then the return is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma} \tag{10}$$

Universidad Veracruzana

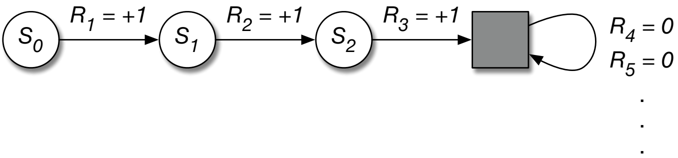
Episodic Notation

- Episodic tasks require some additional notation. Rather than one long sequence of time steps, we need to consider series of episodes, each which consists of a finite sequence of time steps.
- We number the time steps of each episode starting anew from zero.
- We need to represent the time step t at episode i, $S_{t,i}$ for the state, and similarly for $A_{t,i}, R_{t,i}, \pi_{t,i}$, T_i, etc.
- However, it turns out that when we discuss episodic tasks, we almost never have to distinguish between different episodes. Abuse of notation S_t refers to $S_{t,i}$.

Universidad Veracruzana

Absorbing States

- We have defined the return as a sum over a finite number of terms (Eq. 7) and as a sum over infinite number of terms (Eq. 8).
- Both can be unified by considering episode termination to be the entering of a special absorbing state that transitions only to itself and generates always rewards of zero:



Alternative Notation

- Then, we can write:

$$G_T \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{11}$$

including the possibility that $T = \infty$ or $\gamma = 1$, but no both.

Universidad Veracruzana

Ideas

- Almost all RL algorithms involve estimating value functions –functions of states (or of state-action pairs) that estimate how good is it for the agent to be there.
- The notion of how good is defined in terms of future rewards that can be expected, or, to be precise, in terms of expected return.
- The rewards the agent can expect to receive in the future depend on what actions it will take.
- Accordingly, value functions are defined with respect to particular ways of acting, called policies.

Universidad Veracruzana

Policy

- A policy is a mapping from states to probabilities of selecting each possible action.
- If the agent is following the policy π at time t, then $\pi(a \mid s)$ is the probability that $A_t = a$ if $S_t = s$.
- Like p, π is an ordinary function; the bar in the middle merely reminds that it defines a probability distribution over $a \in \mathcal{A}(s)$ for each $s \in \mathcal{S}$.
- Reinforcement learning methods specify how the agent's policy is changed as a result of its experience.

Universidad Veracruzana

Value function

- The value function of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting at s and following π thereafter:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \end{aligned} \tag{12}$$

  for all $s \in \mathcal{S}$.
- $\mathbb{E}_\pi[\cdot]$ denotes the expected value of a random variable, given that the agent follows π; and t is any time step.
- The value of a terminal state is zero.
- We call v_π the state-value function for policy π.

Universidad Veracruzana

Value of Taking an Action

- Similarly, the value of taking an action a in a state s under a policy π is denoted as:

$$
\begin{aligned}
q_\pi(s, a) &\doteq \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]
\end{aligned} \tag{13}
$$

- q_π is called the action-value function for policy π.

Universidad Veracruzana

Estimation of These Functions

- The value functions v_π and q_π can be estimated from experience.
- Example. If an agent follows policy π and maintains an average, for each state encountered, of the actual returns that have followed that state, then the average will converge to the state's value $v_\pi(s)$, as the number of times that state is encountered approaches infinity.
- If separate averages are kept for each action taken in each state, then these averages will converge to action values $q_\pi(s, a)$.
- Estimation methods of this kind are called Monte Carlo methods because they involve averaging over many random samples of actual returns.

Universidad Veracruzana

The Bellman Equation

- For any policy π and any state s, the following consistency condition holds between the value of s and the value of its possible successor states (Similar to Eq. 9):

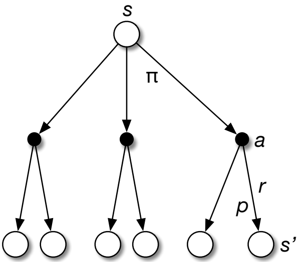
$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_\pi \left[G_{t+1} \mid S_{t+1} = s' \right] \right] \\ &= \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s') \right] \end{aligned} \tag{14}$$

for all $s \in \mathcal{S}$.

- Observe the merged sum over all values of s' and r.

Backup Diagram for v_π

- Open circles represent states.
- Black circles represent a state-action pairs.
- Staring at s the agent could take any of some set of actions, based on policy π.
- For each, the environment could respond with one of several next states s', along with a reward r, depending on its dynamics given by the MDP dynamics function p.



Universidad Veracruzana

Observations

- The Bellman equation averages over all the possibilities, weighting each by its probability of occurring.
- It states that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way.
- The value function v_π is the unique solution to its Bellman equation.
- Backup diagrams represent backup operations that transfer value information back to a state (or a pair state-action) from its successor states.

Universidad Veracruzana

Optimal Policies

- Solving an RL task means, roughly, finding a policy that achieves a lot of reward over the long run.
- For finite MDPs we can precisely define it:
  - Value functions define a partial ordering over policies.
  - $\pi \geq \pi'$ iff $v_{\pi}(s) \geq v_{\pi'}(s)$, for all $s \in \mathcal{S}$.
- There is always at least one policy that is better than or equal to all other policies –an optimal policy.
- Although there may be more than one, we denote all the optimal policies by π_*.

Universidad Veracruzana

Optimal State-Value Functions

- Optimal policies share the same state-value function, called the optimal state-value function, defined as: v_*:

$$v_* \doteq \max_\pi v_\pi(s) \tag{15}$$

for all $s \in \mathcal{S}$.

Universidad Veracruzana

Optimal Action-Value Function

- Optimal policies also share the same optimal action-value function, defined as:

$$q_*(s, a) \doteq \max_{\pi} q_\pi(s, a) \tag{16}$$

  for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

- For the state-action pair (s, a), this function gives the expected return for taking action a in state s and thereafter following an optimal policy. Thus we can write q_* in terms of v_*:

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \tag{17}$$

Universidad Veracruzana

Belleman Optimality Equation I

- Because v_* is the value function for a policy, it must satisfy the self-consistency condition given by the Bellman equation for state values (Eq. 14).
- Because it is the optimal value function, however, v_*'s consistency condition can be written without reference to any specific policy.

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r \mid s, a)[r + \gamma v_*(s')] \end{aligned} \tag{18}$$

Universidad Veracruzana

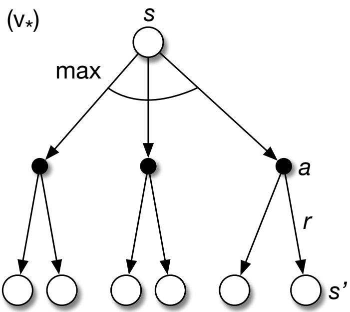
Belleman Optimality Equation II

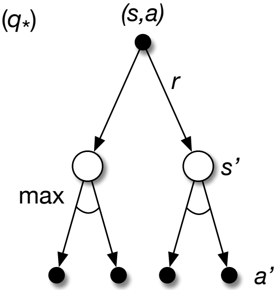
- The Bellman optimality equation for q_* is:

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'}(S_{t+1}, a') \mid S_t = s, A_t = a\Big] \\
&= \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big]
\end{aligned} \tag{19}
$$

Universidad Veracruzana

Backup Diagrams





Solution

- For finite MDPs, the Bellman optimality equation for v_π has a unique solution independent of the policy.
- This equation is actually a system of equations, one for each state –If there are n states, then there are n equations in n unknowns.
- If the dynamics p of the system are known, then in principle the system can be solve for v_*.
- The same for q_*.

Universidad Veracruzana

Getting the Optimal Policy

- Once we have v_* is relatively easy to determina an optimal policy.
- For each state s, there will be one or more actions at which the maximum is obtained in the Bellman optimality equation.
- Any policy that assigns nonzero probability only to those actions is an optimal policy.
- You can think of this as a one-step search.
- Any policy that is greedy with respect to the optimal evaluation function v_* is an optimal policy.
- Observe that v_* already takes into account the reward consequences of all possible future behavior!
- The one-step search yields the long-term optimal actions.

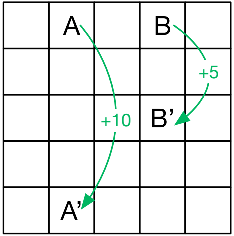
Universidad Veracruzana

Based on q_*

- Having q_* makes choosing optimal actions even easier. With q_* the agent does not even have to do a one-step-ahead search!
- For any state s, it can simply find any action that maximizes $q_*(s, a)$.
- At the cost of representing a function of state-action pairs, instead of just states, we allow optimal actions to be selected without having to know anything about the environment's dynamic!

Universidad Veracruzana

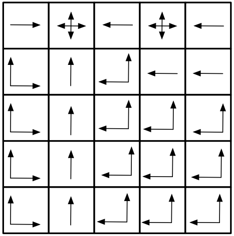
Graphically: Gridworld



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Universidad Veracruzana

Mathematically: The Recycling Robot

$$
\begin{aligned}
v_*(\mathtt{h}) &= \max \left\{ \begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l}) + \gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l}) + \gamma v_*(\mathtt{l})] \end{array} \right\} \\
&= \max \left\{ \begin{array}{l} \alpha[r_{\mathtt{s}} + \gamma v_*(\mathtt{h})] + (1-\alpha)[r_{\mathtt{s}} + \gamma v_*(\mathtt{l})], \\ 1[r_{\mathtt{w}} + \gamma v_*(\mathtt{h})] + 0[r_{\mathtt{w}} + \gamma v_*(\mathtt{l})] \end{array} \right\} \\
&= \max \left\{ \begin{array}{l} r_{\mathtt{s}} + \gamma[\alpha v_*(\mathtt{h}) + (1-\alpha) v_*(\mathtt{l})], \\ r_{\mathtt{w}} + \gamma v_*(\mathtt{h}) \end{array} \right\}.
\end{aligned}
$$

$$
v_*(\mathtt{l}) = \max \left\{ \begin{array}{l} \beta r_{\mathtt{s}} - 3(1-\beta) + \gamma[(1-\beta) v_*(\mathtt{h}) + \beta v_*(\mathtt{l})], \\ r_{\mathtt{w}} + \gamma v_*(\mathtt{l}), \\ \gamma v_*(\mathtt{h}) \end{array} \right\}.
$$

Universidad Veracruzana

Problems

- Explicitly solving the Bellman optimality equation solves the RL, but it is rarely directly useful.
- It is akin to an exhaustive search, looking ahead at all possibilities, computing their desirabilities in terms of expected rewards.
- This solution relies in three assumptions:

    1. We accurately know the dynamics of the environment;
    2. We have enough computational resources to complete the computational solution; and
    3. The Markov property.

Universidad Veracruzana

Decision Making Methods

- Many different decision-making methods can be viewed as ways of approximately solving the Bellman optimality equation.
- Example. Heuristic search methods can be viewed as expanding the right-hand side of (19) several times, up to some depth, forming a "tree" of possibilities, and then using a heuristic evaluation function to approximate v_* at the leaf nodes.
- The methods of dynamic programming can be related even more closely to the Bellman optimality equation.
- Many RL methods approximately solve the Bellman optimality equation, using actual experienced transitions in place of knowledge of the expected transitions.

Universidad Veracruzana

Cost

- We have defined optimal value functions and optimal policies.
- An agent that learns an optimal policy has done it very well, but in practice this rarely happens.
- Optimal policies can be generated only with extreme computational cost.
- Optimality is an ideal that agents can only approximate to varying degrees.
- Memory is also an issue, for the tabular case or more compact parameterized function representations.

Universidad Veracruzana

Approximation

- In approximating optimal behavior, there may be many states that the agent faces with such low probability that selecting suboptimal actions for them has little impact in the received reward.
- The online nature of RL makes it possible to approximate optimal policies in ways to put more effort into learning to make good decisions for frequently encountered states, at the expense of the rest of them.

Universidad Veracruzana

Referencias I

[1] R Sutton and AG Barto. *Reinforcement Learning: An Introduction*. 2nd. Cambridge, MA, USA: The MIT Press, 2018.

Universidad Veracruzana