Reinforcement Learning and Control

Felipe José Bravo Márquez

December 3, 2024

Introduction

- In supervised learning, algorithms try to make their outputs mimic the labels y given in the training set [1].
- The labels give an unambiguous "right answer" for each of the inputs x.
- In contrast, for many sequential decision making and control problems, it is very difficult to provide this type of explicit supervision to a learning algorithm.
- For example, if we have just built a four-legged robot and are trying to program it to walk.
- Initially we have no idea what the "correct" actions to take are to make it walk
- Hence, we don't know how to provide explicit supervision for a learning algorithm to try to mimic.

[1] These slides are based on [Ng, 2000].

Introduction

- In the reinforcement learning framework, we povide our algorithms only a reward function.
- This function indicates to the learning agent when it is doing well, and when it is doing poorly.
- In the four-legged walking example, the reward function might give the robot positive rewards for moving forwards, and negative rewards for either moving backwards or falling over.
- It will then be the learning algorithm's job to figure out how to choose actions over time so as to obtain large rewards.

Introduction

- Reinforcement learning has been successful in applications as diverse as:
  1. autonomous helicopter flight
  2. robot legged locomotion
  3. cell-phone network routing
  4. marketing strategy selection
  5. factory control
  6. efficient web-page indexing
- Our study of reinforcement learning will begin with a definition of the Markov decision processes (MDPs).
- MDPs provide the formalism in which RL problems are usually posed.

Markov Decision Process (MDP)

- A Markov Decision Process is a tuple:

$$(S, A, \{P_{SA}\}, \gamma, R)$$

  where
- S is a set os states. (For example, in autonomous helicopter flight, S might be the set of all possible positions and orientations of the helicopter.)
- A is a set of actions. (For example, the set of all possible directions in which you can push the helicopter's control sticks.)

Markov Decision Process (MDP)

- P_{sa} are the state transition probabilites. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space, i.e., it gives the distribution over what states we will transition to if we take action a in state s.

$$\sum_{s'} P_{sa}(s') = 1, \quad P_{sa}(s') \geq 0$$

- $\gamma \in [0, 1)$ is a discount factor.
- $R : S \rightarrow \mathcal{R}$ is a reward function.

Markov Decision Process (MDP)

The dynamics of an MDP proceeds as follows:

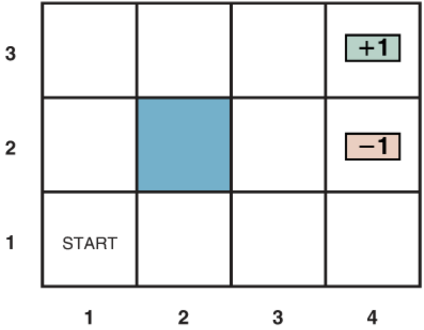
- We start in some state $s0$, and get to choose some action $a0 \in A$ to take in the MDP.
- As a result of our choice, the state of the MDP randomly transitions to some successor state $s1$, drawn according to $s1 \sim P_{s0a0}$.
- Then, we get to pick another action $a1$.
- As a result of this action, the state transitions again, now to some $s2 \sim P_{s1a1}$.
- We then pick $a2$, and so on. . . .

Pictorially, we can represent this process as follows:

$$so \xrightarrow{a0} s1 \xrightarrow{a1} s2 \xrightarrow{a2} s3 \xrightarrow{a3} ..$$

Example

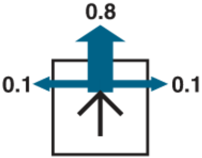
- Suppose that an agent is situated in the 4×3 environment as shown in the Figure below:



- Beginning in the start state, it must choose an action at each time step.
- There 11 states (S) and four actions $A = \{N, S, E, W\}$.
- The interaction with the environment terminates when the agent reaches one of the goal states, marked +1 or –1.

Example

- The “intended” outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction:



- For example:
  $P_{(3,1)N}((3,2)) = 0.8$
  $P_{(3,1)N}((4,1)) = 0.1$
  $P_{(3,1)N}((2,1)) = 0.1$
  $P_{(3,1)N}((3,3)) = 0$
  ... etc

Total Payoff

- Upon visiting the sequence of states $s0, s1, \ldots$ with actions $a0, a1, \ldots$, our **total payoff** is given by:

$$R(so) + \gamma R(s1) + \gamma^2 R(s2) + \ldots$$

  $0 \leq \gamma < 1$

- Our goal in reinforcement learning is to choose actions over time so as to maximize the expected value of the total payoff:

$$E[R(so) + \gamma R(s1) + \gamma^2 R(s2) + \ldots]$$

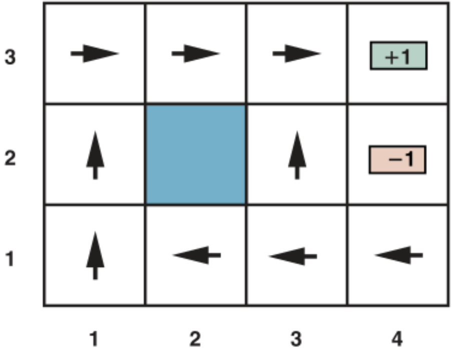
- Note that the reward at timestep t is discounted by a factor of γ^t.
- Thus, to make this expectation large, we would like to accrue positive rewards as soon as possible (and postpone negative rewards as long as possible).
- In economic applications where $R(\cdot)$ is the amount of money made, γ also has a natural interpretation in terms of the interest rate (where a dollar today is worth more than a dollar tomorrow).

Example

- A collision with a wall results in no movement.
- Transitions into the two terminal states have reward +1 and –1, respectively.
  $R((4,3)) = +1$
  $R((4,2)) = -1$
  $R(S) = -0.02$ for all other states.
- All other transitions have a reward of –0.02
- This is common in navigation tasks to charge the robot for fuel comsumption and to (to avoid the robot wasting time).
- We will also assume that the robot stops when it reaches states $(4,3)$ or $(4,2)$.
- These terminal states can be modelled by adding an extra state in which transitions from these terminal states are made with probability 1 and the agent loops forever in this new state with no more rewards.

Policy

- A policy is any function $\pi : S \to A$, mapping states to actions.
- We say that we are executing some policy π if, whenever we are in state s, we take action $a = \pi(s)$.
- The "optimal policy" for the previous example would be:



Policy

- So when you execute this policy this will maximize your expected value of the total payoffs.
- It is worth noting that the action of going west (left) from state $(3, 1)$ is not trivial, since going north would bring the agent closer to the goal.
- However, entering state $(3, 2)$ adds the danger of ending up in state $(4, 2)$.
- Therefore, the optimal solution favored the longer and less risky path.

Value and Value-Action Functions

- For any policy π, we define the value function $V^\pi : S \to \mathbb{R}$ as:

$$V^\pi(s) = \mathbb{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s, \pi\right].$$

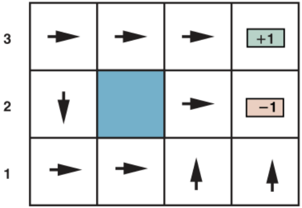
- $V^\pi(s)$ is the expected sum of discounted rewards (or total payoff) starting in state s and executing policy π.
- Similarly, we can define the action value function $q^\pi : S \times A \to \mathbb{R}$ as:

$$Q^\pi(s,a) = \mathbb{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s, a_0 = a, \pi\right].$$

- In the action-value function, for each state and action pair, the action-value function outputs the expected return if the agent starts in that state, takes that action, and then follows the policy forever after.

Value Function Example

- The following examples shows a pretty bad policy π than seems to be heading to -1 rather than +1 in most cases:



- The value function $V^{\pi}(s)$ for this policy is as follows:

3	.52	.73	.77	+1
2	-.90		-.82	-1
1	-.88	-.87	-.85	-1.00

Bellman Equation

- Given a fixed policy π, its value function V^π satisfies the Bellman equation:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s').$$

- This states that the expected sum of discounted rewards $V^\pi(s)$ for starting in state s consists of two terms:
    1. The immediate reward $R(s)$ obtained immediately upon being in state s, and
    2. The expected sum of future discounted rewards.
- Examining the second term more closely, the summation can be rewritten as $\mathbb{E}_{s' \sim P_{s\pi(s)}}[V^\pi(s')]$.
- This represents the expected value of the discounted rewards starting in state s', where s' is distributed according to $P_{s\pi(s)}$—the probability distribution over the states we transition to after taking the action $\pi(s)$ in the MDP from state s.
- Thus, the second term gives the expected sum of discounted rewards obtained after the first step in the MDP.

Bellman Equation Explained

- Start with the Definition of $V^{\pi}(s)$:

$$V^{\pi}(s) = \mathbb{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \;\middle|\; s_0 = s, \pi\right].$$

- The first term in the expectation corresponds to the immediate reward $R(s)$. The subsequent terms capture the future rewards starting from the next state:

$$V^{\pi}(s) = R(s) + \gamma \mathbb{E}\left[V^{\pi}(s_1) \mid s_0 = s, \pi\right].$$

- The expectation over the next state s_1 is computed using the state transition probabilities $P_{s\pi(s)}(s')$. Hence:

$$\mathbb{E}[V^{\pi}(s_1)|s_0 = s, \pi] = \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s').$$

- This satisfies the definition of expected value because we are calculating the weighted average of the value function $V^{\pi}(s_1)$ over all possible next states s' where the weights are given by the probabilities of transitioning to those states under the given policy π.
- Substituting this back into the equation gives:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s').$$

Example: Computing the Value Function

- Suppose the policy at state (3, 1) prescribes the action north (N), i.e., $\pi((3,1)) = N$.
- The value function for this state is:

$$V^\pi((3,1)) = R((3,1)) + \gamma\Big(0.8 \times V^\pi((3,2)) + 0.1 \times V^\pi((4,1)) + 0.1 \times V^\pi((2,1))\Big).$$

- For every state in the Markov Decision Process (MDP), a similar equation can be written.
- Each value $V^\pi(s)$ represents an unknown variable.
- In this example, there are 11 states, leading to 11 linear equations.
- Solving these equations yields the value function $V^\pi(s)$ for all states.

References I

Ng, A. (2000).
Cs229 lecture notes.
CS229 Lecture notes, 1(1):1–3.