RLNotes

David Peña Peralta

2024-07-11

Table of contents

Preface

Reinforced Learning is learning what to do -how to map situations to actions- so as maximize a numerical reard signal.

We formalize the problem of reinforcement learning using ideas from dynamical systems theory, specially, as the optimal control of incompletely-known Markov decission processes.

One of the challenges that arise in reinforcement learning, and not in other kinds of learning, is the trade-off betweeen exploration and exploitation. The agent has.

1 Elements of Reinforcement Learning.

Beyond the agent and the enviroment, one can identify four main subelements od a reinforcement learning system:

- A Policy
- A Reward Signal
- A Value Function

and optionally, a model of the enviroment.

A *policy* defines the learning agent's way of behaving at a given time. In general, policies may be stochastic, specifying probabilities for each action.

A *reward signal* defines the goal of a reinforcement learning problem. The agent's role objective is to maximize the total reward it receives over the long run.

Whereas the reward signal indicates what is good in immediate sense, a *value function* specifies what is good in the long run.

The fourth and final element of some reinforcement learning systems is a model od th eenviroemnt. This is something that mimics the behavior of the envirometn, or more generally, that allows inferences to be made about how the enviroment will behave.

2 An Extended Example: Tic-Tac-Toe

Consider the familiar child's game of tic-tac-toe. Two players take turns playing on a three-by-three board. One player plays Xs and the other Os until one player wins by placing three marks in a row, horizontally, vertically, or diagonally, as the X player has in the game shown to the right. If the board fills up with neither player getting three in a row, then the game is a draw.

Because a skilled player can play so as never to lose, let us assume that we are playing against an imperfect player, one whose play is sometimes incorrect and allows us to win. For the moment, in fact, let us consider draws and losses to be equally bad for us.

How might we construct a player that will find the imperfections in its opponent's play and learn to maximize its chances of winning?

2.1 Construct Decision Making Model

Here is how the tic-tac-toe problem would be approached with a method making use of a value function. First we would set up a table of numbers, one for each possible state of the game. Each number will be the latest estimate of the probability of our winning from that state. We treat this estimate as the state's value, and the whole table is the learned value function. State A has higher value than state B, or is considered "better" than state B, if the current estimate of the probability of our winning from A is higher than it is from B. Assuming we always play Xs, then for all states with three Xs in a row the probability of winning is 1, because we have already won. Similarly, for all states with three Os in a row, or that are filled up, the correct probability is 0, as we cannot win from them. We set the initial values of all the other states to 0.5, representing a guess that we have a 50% chance of winning.

2.2 Exploratory vs Greedy

We then play many games against the opponent. To select our moves we examine the states that would result from each of our possible moves (one for each blank space on the board) and look up their current values in the table. Most of the time we move greedily, selecting the move that leads to the state with greatest value, that is, with the highest estimated probability of winning. Occasionally, however, we select randomly from among the other moves instead.

These are called exploratory moves because they cause us to experience states that we might otherwise never see.

3 Excersice Code: Tic Tac Toe.

3.1 Objective

In the first excersice code going to make a Tic Tac Toe and check the probabilites of win.

3.2 Construct Tic Tac Toe Model.

The Tic Tac Toe Board is a game over a grid of 3×3 squares

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

In each square (a_{ij}) can write a $\mathbf{X}$ or $\mathbf{O}$. A player use the $\mathbf{X}$ and the other player use the $\mathbf{O}$. In the game by turns both players place us simbols $\{\mathbf{X}, \mathbf{O}\}$ in the board. The winner is the first player in obtain a three row, column or principal diagonal.

For example, if the first player is $\mathbf{X}$ and

$$s : a_{11} = \mathbf{X}, a_{12} = \mathbf{O}, ... \begin{bmatrix} \mathbf{X} & \mathbf{O} & \mathbf{O} \\ \mathbf{X} & \mathbf{X} & \\ \mathbf{X} & & \mathbf{O} \end{bmatrix}.$$

In this case, the $\mathbf{X}'s$ player win. To construct the Decision Model be consider the States Set $(\mathcal{S})$ as the Tic Tac Toe diagrams

$$\mathcal{S} = \{(k, (i, j), a)\}_k, 1 \leq k \leq 9, 1 \leq i, j \leq 3, a \in \{\mathbf{X}, \mathbf{O}\}\}.$$

Then $s \in \mathcal{S}$

$$
\begin{aligned}
s =& \Bigg\{ \Big(1, \big((1,1), \mathbf{X}\big)\Big), \Big(2, \big((1,2), \mathbf{O}\big)\Big), \Big(3, \big((2,2), \mathbf{X}\big)\Big), \dots \\
& \Big(4, \big((3,3), \mathbf{O}\big)\Big), \Big(5, \big((3,1), \mathbf{X}\big)\Big), \Big(6, \big((1,3), \mathbf{O}\big)\Big), \Big(7, \big((2,1), \mathbf{X}\big)\Big) \Bigg\} \\
s \equiv& \begin{bmatrix} \mathbf{X} & \mathbf{O} & \mathbf{O} \\ \mathbf{X} & \mathbf{X} & \\ \mathbf{X} & & \mathbf{O} \end{bmatrix}.
\end{aligned}
$$

Note that exists states $u_1, u_2, \dots, u_6$ such that

$$u_1 \to u_2 \to u_3 \to \dots \to u_6 \to s.$$

In other words, exists a function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ such that

$$S_{t+1} = f(S_t, a_t),$$

where f represent the model dynamics. In this case f was the response of the rival. Consider the initial condition S_0

$$S_0 = \{\} \equiv \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

when you are the first player, if you are the second (with $\mathbf{X}$ or $\mathbf{O}$)

$$S_0((i,j), p) = \left(1, ((i,j), p)\right), p \in \{\mathbf{X}, \mathbf{O}\}.$$

In both cases, with S_0, the agent continue choose a action a_0 and obtain the following state

$$S_1 = f(S_0, a_0).$$

Later defines the history H_t as the sucession of the following shape

$$H_t = \{S_0, a_0, S_1, a_1, \dots, a_{t-1}, S_t\}$$

4 Dynamical Programming

4.1 Introduction

The temporal structure of a typical dynamic program is

1. An initial state X_0 is given
2. $t \leftarrow 0$
3. while $t < T$ do
4. () The controller of the system observes the current state X_t
5. The controller chooses an action A_t
6. () The controller receives a reward R_t that depends on the current state and action.
7. () the state update to X_{t+1}
8. () $t \leftarrow t + 1$
9. end

The state X_t is a vector listing current values of variables deemed relevant to choosing the current action. The action A_t is a vector describing choices of a set of decision variables. If $T < \infty$ then the problem has a finite horizon. Otherwise it is an infinite horizon problem.

Dynamic programming provides a way to maximize expected lifetime reward of a decision maker who receives a prospective reward sequence $(R_t)_{t\geq 0}$ and who confronts a system that maps today's state and control into next period's state. A lifetime reward is an aggregation of the individual period rewards $(R_t)_{t\geq 0}$ into a single value. An example of lifetime reward is an expected discounted sum

$$E\left[\sum_{t\geq 0} \beta^t R_t\right]$$

for some $\beta \in (0, 1)$.

Example A manager wants to set prices and inventories to maximize a firm's expected present value (EPV), which, given interest rate r, is defined as

$$E\left[\sum_{k\geq 0} \pi_k \left(\frac{1}{1+r}\right)^k\right]. \tag{4.1}$$

Here X_t, will be a vector that quantifies the size of the inventories, prices set by competitors and other factors factors relevant to profit maximization. The action A_t sets current prices and orders of new stock. The current reward R_t is current profit π_t, and the profit stream $\{\pi_t\}_{t \geq 0}$ is aggregagted into a lifetime reward via Equation 4.1.

The core theory of dynamic programming is relative simple and concise. But implementation can be computationally demanding. That situation provides one of the major challenges facing the field of dynamic programming.

4.2 Bellman Equations

In this section we introduce the recursive structure of dynamic programming in a simple setting. After solving a finite-horizon model, we consider an infinite-horizon version and explain how it produces a system of nonlinear equations. Then we turn to methods for solving such systems.

4.2.1 Finite-Horizon Job Search

We begin with a celebrated model of job search created by McCall (1970). McCall analyzed the decision problem of an unemployed worker in terms of current and prospective wage offers, impatience, and the availability of unemployment compensation.

4.2.1.1 A Two Period Problem

Imagine someone who begins her working life at time $t = 1$ without employment. While unemployed, she receives a new job offer paying wage W_t at each date t. She can accept the offer and work permanently at that wage level or reject the offer, receive unemployment compensation c, and draw a new offer next period. We assume that the wage offer sequence is i.i.d and nonegative, with distribution φ. In particular,

- $W \subset \mathbb{R}^+$ is a finite set of possible wage outcomes and
- $\varphi : W \to [0, 1]$ is a probability distribution on W, assigning a probability $\varphi(w)$ to each possible wage outcome w.

The worker is impatient. Impatiente is parametrized by a time discount factor $\beta \in (0, 1)$, so that the present value of a next-period payoff of y dollars is β_y. Since $\beta < 1$, the worker will be tempted to accept reasonable offeres, rather than to wait for better ones. A key question is how long to wait.

Suppose as a first step that working life is just two periods. To solve our problem we work backwards, starting at the final date $t = 2$ after W_2 has been observed.

If she is already employed, the worker has no decision to make: she continues working at her current wage. If she is unemployed, then she should take the largest of c and W_2.

Now we step back to $t = 1$. At this time, having received offer W_1, the unemployed worker's options are (a) accept W_1 and receive it in both periods or (b) reject it, receive unemployment compensation c, and then, in the second period, choose the maximum of W_2 and c.

Let's assume that the worker seeks to maximize expected present value. The EPV of option (a) is $W_1 + \beta W_1$, which is also called the stopping value. The EPV of option (b), also called the continuation value, is $h_1 := c + \beta E\left[\max\{c, W_2\}\right]$. More explicitly,

$$h_1 = c + \beta \sum_{w' \in W} v_2(w')\varphi(w'), \text{where } v_2(w) := \max\{c, w\}. \tag{4.2}$$

The optimal choice as $t = 1$ is now clear: accept the offer if $W_1 + \beta W_1 \geq h$ and reject otherwise.

4.2.1.2 Value Functions

A key idea in dynamic programming is to use *value functions* to track maximal lifetime rewards from a given state at a given time. The time 2 value function v_2 in Equation 4.2 returns the maximum value obtained in the final stage for each possible realization of the time 2 wage offer. The time 1 value function v_1 evaluated at $w \in W$ is

$$v_1 := \max\{w + \beta w, c + \beta \sum_{w' \in W} v_2(w)\varphi(w')\} \tag{4.3}$$

It represents the present value of expected lifetime income after receiving the first offer w, conditional conditional on choosing optimally in both periods.

Now calculate the w such that solves the indifference condition

$$w + \beta w = c + \beta \sum_{w' \in W} v_2(w')\varphi(w')$$

This is,

$$\begin{aligned} w =& \frac{c + \beta \sum_{w' \in W} v_2(w')\varphi(w')}{1 + \beta} \\ =& \frac{h_1}{1 + \beta} \end{aligned}$$

4.2.1.3 Three Periods

Now let's suppose that the works in period $t = 0$ as well as $t = 1, 2$. At $t = 0$, the value of accepting the current offer W_0 is $W_0 + \beta W_0 + \beta^2 W_0$, while maximal value of rejecting and waiting, is c plus, after discounting by β, the maximum value that can be obtained by behaving optimally from $t = 1$. We have already calculated this value: it is just $v_1(W_1)$, as given is Equation 4.3!

Maximal time zero value $v_0(w)$ is the maximum of the value of these two options, given $W_0 = w$, so we can write

$$v_0(w) = \max\{w + \beta w + \beta^2 w, c + \beta \sum_{w' \in W} v_1(w')\varphi(w')\} \tag{4.4}$$

By plugging v_1 from Equation 4.3 into this expression, we can determine v_0, as well as the optimal action, that one that achieves the largest value in the max term in Equation 4.4.

5 The multi armed bandits

References