Markov Decision Processes (& Reinforcement Learning)

Saeed Saremi

November 14, 2024

OUTLINE

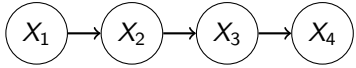
- Markov Decision Process (MDP)
  - policy and value functions
- Bellman equations[1]
  - policy and value iterations
- Deep RL

[1] Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.*

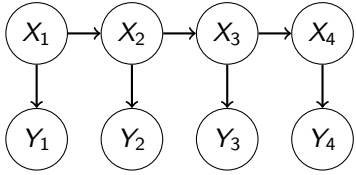
MARKOV AGAIN!

So far in the course we have seen:

- Markov chains



- Hidden Markov models



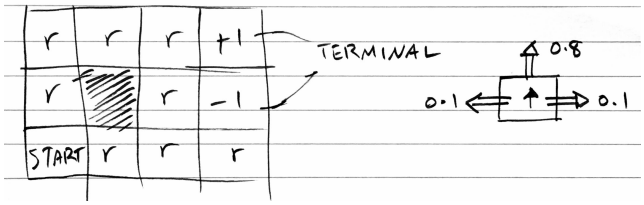
- Langevin MCMC

$$x_{t+1}|x_t \sim \mathcal{N}(x_t + h\nabla \log p(x_t), 2h\, I)$$

Today, we will explore a new concept:

Markov Decision Process

A MOTIVATING EXAMPLE[2]

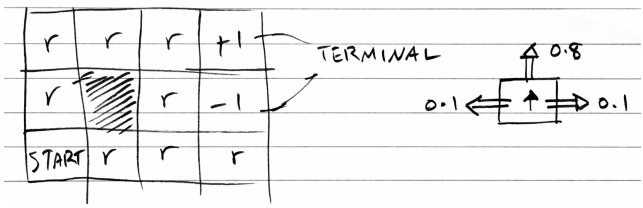


- An agent with a sequential decision problem on a simple 4×3 environment.
- $P(S_{t+1} = s' | S_t = s, A_t = a)$: the transition model of the environment. The intended outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement.
- The two TERMINAL states have a reward of $+1$ and 1, respectively, and all other states have a reward of r. Informally speaking, the agent's goal is to accumulate rewards.
- $\mathcal{A}(s)$ denotes the set of actions one can take in state s. In this case $\mathcal{A} = \{\uparrow, \downarrow, \rightarrow, \leftarrow\}$.
- Our agent knows the map of the world and starts with a *policy* (denoted by π) to act:

$$\pi : s \mapsto a(s) \in \mathcal{A}(s)$$

- What do you think a natural definition for *optimal policy* π^* is?

[2] Russell, S. J., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*.



- We pick a policy π and will run "many" simulations and collect the accumulate reward G_0:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T$$

We take the Monte Carlo average of of G_0 over many (when do you think we should stop running our sampling of trajectories?) such trajectories. This Monte Carlo average converges to what we call the *value*[3] of our policy which we write compactly as

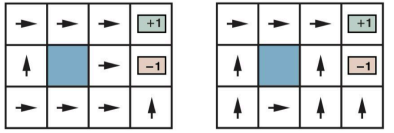
$$v_\pi = \mathbb{E}[G_0].$$

- What do you think the optimal policy $\pi^*(s)$ is for $s = (3, 1)$? Does it depend on r?

[3] reward is "local"; value is "global"

OPTIMAL POLICY

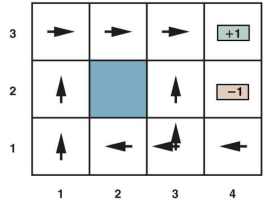
- The optimal policy for different values of r:



$r < -1.6497$ $\quad -0.7311 < r < -0.4526$

$-0.0274 < r < 0$ $\quad r > 0$

- It turns out there are two (degenerate) optimal policies for $r = -0.04$:



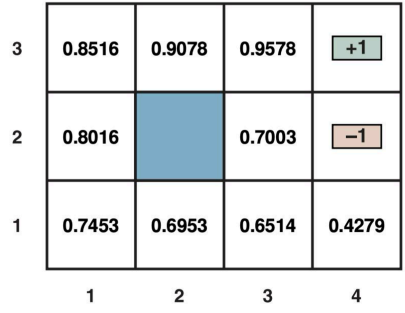


Figure: The value function corresponding to the optimal policy ($r = -0.04$).

MARKOV DECISION PROCESSES[4]

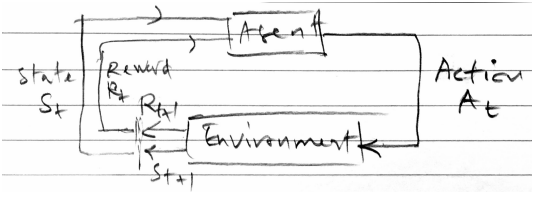
- Consider how a general environment might respond at time $t+1$ to the action taken at time t. In the most general, causal case this response may depend on everything that has happened earlier. In this case the dynamics can be dened only by specifying the complete probability distribution:

$$P(S_{t+1} = s', R_{t+1} = r \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t)$$

  for all r, s', and all possible values of the past events: $S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t$.

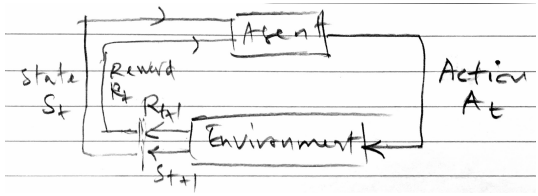
- The Markov property: the environment's response at $t+1$ depends only on the state and action representations at t, i.e., environment's dynamics can be defined by specifying only

$$P(s', r \mid s, a) = P(R_{t+1} = r, S_{t+1} = s' \mid S_t = s, A_t = a)$$



[4] Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*.

NON-DETERMINISTIC POLICY & DISCOUNT RATE



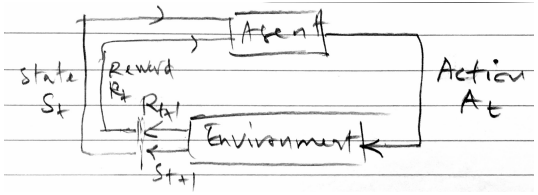
- At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's policy and is denoted π_t, where $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$.[5]
- RL methods specify how the agent changes its policy as a result of its experience. The agent's goal is to maximize the total amount of reward it receives over the long run.
- We revise our definition of the "total reward", a.k.a. *return*, for mathematical convenience:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

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

[5] A good example is the game "rock paper scissors". What's the optimal policy there?

VALUE FUNCTION

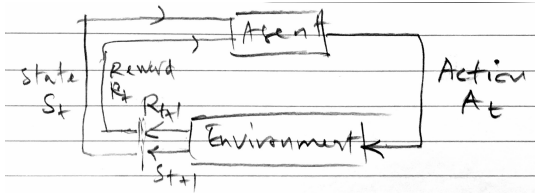


- Recall that a policy, π, is a mapping from each state, $s \in S$, and action, $a \in \mathcal{A}(s)$, to the probability $\pi(a|s)$ of taking action a when in state s. Informally, the value of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, we can define $v_\pi(s)$ formally as

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s],$$

where $\mathbb{E}_\pi$ denotes the expected value of a random variable given that the agent follows policy π, and t is any time step. We call the function v_π the state-value function for policy π.

Q FUNCTION



- Similarly, we define the value of taking action a in state s under a policy π, denoted $q_\pi(s, a)$, as the expected return starting from s, taking the action a, and thereafter following policy π:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a].$$

  We call q_π the action-value function for policy π.

- The value functions v_π and q_π can be estimated from experience by running simulations. We call estimation methods of this kind Monte Carlo methods because they involve averaging over many random samples of actual returns.

? Which value function is more "valuable": v_π or q_π?

OPTIMAL POLICY

- A policy π is dened to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. In other words, $\pi \geq \pi'$ if and only if

$$v_\pi(s) \geq v_{\pi'}(s), \ \forall s \in \mathcal{S}.$$

- There is always at least one policy that is better than or equal to all other policies. This is an optimal policy. Although there may be more than one (our example earlier), we denote all the optimal policies by π^*. They share the same state-value function, called the optimal state-value function, denoted v_*, and dened as

$$v_*(s) = \max_\pi v_\pi(s) = v_{\pi^*}(s)$$

- Relating optimal state-value function to action-value function:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi^*}(s, a)$$

THE DARK KNIGHT DYNAMIC PROGRAMMING RETURNS

Richard C. Bellman

BELLMAN OPTIMALITY EQUATION

$$
\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_{\pi^*}(s,a) \\
&= \max_a \mathbb{E}_{\pi^*}\left[G_t \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} R_{t+k+2} \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a \right] \\
&= \max_{a \in A(s)} \sum_{s',r} p(s', r \mid s, a)\left(r + \gamma v_*(s') \right)
\end{aligned}
$$

- Similarly, the Bellman optimality equation for q_* is

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

We turn the Bellman equation for v_π,

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a) \Big(r + \gamma v_\pi(s') \Big),$$

into an iterative algorithm to find v_π which we can iteratively improve via policy improvement:

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s)) \big[r + \gamma V(s') \big]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      $a \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a) \big[r + \gamma V(s') \big]$
      If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

VALUE ITERATION

Using Bellman optimality equation

$$v_*(s) = \max_a \sum_{s',r} P(s', r|s, a) \left(r + \gamma v_*(s')\right)$$

into an iterative algorithm:

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

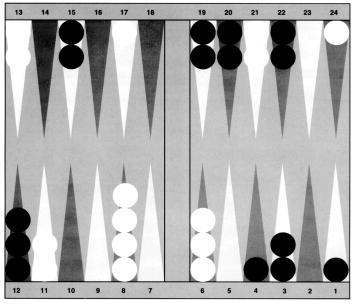
Output a deterministic policy, π, such that
  $\pi(s) = \text{argmax}_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

There are, however, some details to consider. In the first place, the effective analytic solution of a large number of even simple equations as, for example, linear equations, is a difficult affair. Lowering our sights, even a computational solution usually has a number of difficulties of both gross and subtle nature. Consequently, the determination of this maximum is quite definitely not routine when the number of variables is large.

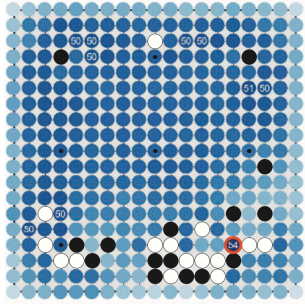
All this may be subsumed under the heading "the curse of dimensionality." Since this is a curse which has hung over the head of the physicist and astronomer for many a year, there is no need to feel discouraged about the possibility of obtaining significant results despite it.

Dynamic Programming, Richard Bellman, 1957.

🏆 CURSE OF DIMENSIONALITY?



(a) Tesauro's TD-GAMMON, 1995



(b) DeepMind's ALPHAGO, 2016

- Finite MDP $\approx$ 90% of modern reinforcement learning.[6]
- The other 10% goes around parametrizing π, v, and q with neural networks.

Deep RL Bootcamp: `https://www.youtube.com/watch?v=qaMdN6LS9rA`

"reward is enough"?[7] or just a "cherry on the cake"?[8] ...

[6] Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.*
[7] Silver, D., Singh, S., Precup, D., & Sutton, R. S. (2021). *Reward is enough.*
[8] Yann LeCun is to blame.